SYBASE®
An SAP Company

システム管理ガイド:
第 2 巻

# Adaptive Server® Enterprise

15.7

ドキュメント ID：DC33117-01-1570-01
改訂：2011 年9 月



Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

# 目次

第 1 章

# サーバ・リソースへのアクセスの制限

この章では、個々のログインまたはアプリケーションがクリティカルな時間帯に使用できる I/O コスト、ロー・カウント、処理時間、または tempdb 領域を制限するためのリソース制限の使用方法について説明します。また、連続する時間のブロックを指定してリソース制限を適用するために、名前付き時間範囲を作成する方法についても説明します。



## リソースの制限

「リソース制限」とは、システム管理者が指定する一連のパラメータであり、個々のログインやアプリケーションからのクエリやトランザクションによるサーバ・リソースの独占を防ぎます。

リソース制限は時間範囲にバインドされるので、システム管理者はリソース制限が適用される時期を正確に定義できます。システム管理者がリソース制限を変更すると、システム管理者も含め、ログインしているすべてのユーザに変更が反映されます。

リソースの制限を指定する一連のパラメータの中には、制限を適用する時間帯や、実行するアクションの種類を指定するものもあります。たとえば、クリティカルな時間帯に膨大なレポートが出力されることを防いだり、不要な「直積」を生成するクエリを実行するセッションを強制終了したりすることができます。

# リソース制限の計画

リソース制限の計画では、次のことを考慮します。

- 制限を施行する時間帯と曜日
- モニタ対象となるユーザとアプリケーション
- 設定する制限のタイプ
  - 大量の論理読み込みと物理読み込みが必要になる可能性のあるクエリの I/O コスト (見積もりまたは実績値)
  - 大量の結果セットを返す可能性のあるクエリのロー・カウント
  - クエリが複雑であるため、またはサーバの負荷などの外部要因のために、完了までに時間がかかる可能性のあるクエリの経過時間
- 制限を個々のクエリに適用するか、さらに広いスコープ (クエリ・バッチまたはトランザクション) を指定するか
- 接続を開始したが長時間アイドル状態にして、ロックなどのシステム・リソースを使用している可能性があるユーザに対する最大アイドル時間
- I/O コスト制限を実行前と実行中のどちらに施行するか
- 制限超過時に実行するアクション (警告の発行、クエリ・バッチまたはトランザクションのアボート、セッションの強制終了など)

# リソース制限の有効化

リソース制限を有効にするには、次の設定パラメータを使用します。

```
sp_configure "allow resource limits", 1
```

値を 1 に設定するとリソース制限が有効になり、0 に設定すると無効になります。allow resource limits は静的なパラメータであるため、設定を変更するにはサーバを再起動する必要があります。

allow resource limits が設定されているとき、サーバは、時間範囲、リソース制限、内部サーバ・アラームのための内部メモリを割り付けます。また、サーバは、ログイン・セッションに対して適用可能な範囲と制限を内部的に割り当てます。

allow resource limits を 1 に設定すると、showplan と statistics i/o の出力も次のように変わります。

- showplan では、オプティマイザによるクエリ全体の見積もりコストが単位なしの数値として表示されます。このコスト見積もりは、テーブル統計情報 ( 値の数と分散 ) と該当するバッファ・プールのサイズによって変わります。バッファ・プールの状態やアクティブ・ユーザ数などの要因の影響は受けません。『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「第 2 章 showplan の使用」を参照してください。
- statistics i/o では、オプティマイザのコスト計算式に従って計算された、文の実際の合計 I/O コストが表示されます。この値は、論理 I/O 数に論理 I/O 当たりのコストを掛けた値と、物理 I/O 数に物理 I/O 当たりのコストを掛けた値の合計を表します。

# 時間範囲の定義

時間範囲とは、週の中の 1 日または連続した複数の日における、1 日の中の連続した時間をいいます。

Adaptive Server® には事前に定義された時間範囲 “at all times” があり、月曜日から日曜日の午前 0 時から午前 0 時まですべての時間を表します。リソースを制限する上で必要であれば、他にも時間範囲を作成、修正、削除できます。

名前付き時間範囲どうしが重複してもかまいません。ただし、特定のユーザとアプリケーションの組み合わせに対する制限が複数ある場合に、それぞれに関連付けられた名前付き制限範囲が重複していてはなりません。

たとえば、“joe_user” に対して、営業時間帯に給与アプリケーションを実行したときに返されるローの数を 100 に制限したとします。次に、このユーザがピーク時間帯に取り出すローの数を制限しようとしますが、ピーク時間帯は営業時間帯と重複しています。この新しい制限は、既存の制限と重複するため失敗します。

同じ時間範囲を共有する複数の制限を作成することはできます。たとえば、ローの取り出し制限と同じ時間範囲で、“joe_user” に対する第 2 の制限を設定できます。たとえば、このユーザのロー取り出し制限を設定した時間範囲と同じ時間範囲について、クエリの実行時間の制限を設定できます。

名前付き時間範囲を作成すると、systimeranges システム・テーブルに格納されます。時間範囲にはそれぞれ範囲 ID 番号があります。範囲 “at all times” の範囲 ID は 1 です。Adaptive Server のメッセージは、特定の時間範囲を参照します。

## 必要な時間範囲の決定

次のような表を使用して、サーバごとに作成する時間範囲を決定します。1 週間のサーバ使用状況をモニタし、サーバが特にビジーな時間帯や、割り込みを避けたい重要タスクの実行時間帯を調べます。

| 日 / 時刻 | 00:00 | 01:00 | 02:00 | 03:00 | 04:00 | 05:00 | 06:00 | 07:00 | 08:00 | 09:00 | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 | 15:00 | 16:00 | 17:00 | 18:00 | 19:00 | 20:00 | 21:00 | 22:00 | 23:00 | 00:00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 月曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 火曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 水曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 木曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 金曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 土曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 日曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |

## 名前付き時間範囲の作成

sp_add_time_range を使用して、次の項目を指定します。

- 時間範囲の名前
- 時間範囲の開始曜日と終了曜日
- 時間範囲の開始時刻と終了時刻

『リファレンス・マニュアル：プロシージャ』の「sp_add_time_range」を参照してください。

### 時間範囲の例

毎週、次の時間帯に重要ジョブを 2 つ実行するとします。

- ジョブ 1 を火曜日と水曜日にそれぞれ 07:00 から 10:00 まで実行
- ジョブ 2 を土曜日の 08:00 から日曜日の 13:00 まで実行

次の表では、“1” はジョブ 1 を実行する時間帯を表し、“2” はジョブ 2 を実行する時間帯を表しています。

| 日 \ 時刻 | 00:00 | 01:00 | 02:00 | 03:00 | 04:00 | 05:00 | 06:00 | 07:00 | 08:00 | 09:00 | 10:00 | 11:00 | 12:00 | 13:00 | 14:00 | 15:00 | 16:00 | 17:00 | 18:00 | 19:00 | 20:00 | 21:00 | 22:00 | 23:00 | 00:00 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 月曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 火曜日 | | | | | | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| 水曜日 | | | | | | | | 1 | 1 | 1 | 1 | | | | | | | | | | | | | | |
| 木曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 金曜日 | | | | | | | | | | | | | | | | | | | | | | | | | |
| 土曜日 | | | | | | | | | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 日曜日 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | | | | | | | | | | | |

ジョブ 1 は単一の時間範囲 tu_wed_7_10 で表せます。

```
sp_add_time_range tu_wed_7_10, tuesday, wednesday, "7:00", "10:00"
```

ジョブ 2 には、土曜日の時間範囲と日曜日の時間範囲の 2 つが必要です。

```
sp_add_time_range saturday_night, saturday, saturday, "08:00", "23:59"
sp_add_time_range sunday_morning, sunday, sunday, "00:00", "13:00"
```

## 名前付き時間範囲の変更

sp_modify_time_range を使用して、次の項目を指定します。

- 修正する時間範囲
- 変更する曜日
- 変更する時間帯

『リファレンス・マニュアル：プロシージャ』の「sp_modify_time_range」を参照してください。

たとえば、*business_hours* 時間範囲の既存の開始曜日、開始時刻、終了時刻はそのままで、終了曜日を土曜日に変更するには、次のコマンドを発行します。

```
sp_modify_time_range business_hours, NULL, Saturday, NULL, NULL
```

*before_hours* 時間範囲の終了曜日と終了時刻を新しく指定するには、次のコマンドを発行します。

```
sp_modify_time_range before_hours, NULL, Saturday, NULL, "08:00"
```

**注意**　時間範囲 “at all times” の変更はできません。

## 名前付き時間範囲の削除

ユーザ定義の時間範囲を削除するには、sp_drop_time_range を使用します。

『リファレンス・マニュアル：プロシージャ』の「sp_drop_time_range」を参照してください。

たとえば、時間範囲 *evenings* を master データベースの systimeranges システム・テーブルから削除するには、次のコマンドを発行します。

```
sp_drop_time_range evenings
```

**注意** 時間範囲 “at all times” や、リソース制限が定義されている時間範囲は削除できません。

### 時間範囲の変更の反映時期

アクティブな時間範囲がログイン・セッションと関連付けられるのは、各クエリ・バッチの開始時です。実時間の変更が原因でサーバのアクティブな時間範囲の値が変更されても、クエリ・バッチの処理中には影響はありません。つまり、あるリソース制限によって特定の時間範囲内のクエリ・バッチの実行が制限されていても、その時間範囲がアクティブになる前にクエリ・バッチが開始していれば、実行中のクエリ・バッチは、リソース制限の影響を受けません。しかし、同じログイン・セッション中に別のクエリ・バッチを実行する場合は、そのクエリ・バッチは変更の影響を受けます。

時間範囲を追加、修正、削除しても、現在進行中のログイン・セッションのアクティブな時間範囲は影響を受けません。

リソース制限でトランザクションがそのスコープとなっている場合、トランザクション実行中にサーバのアクティブな時間範囲の変更が発生しても、現在進行中のトランザクションは新しいアクティブ時間範囲の影響を受けません。

## ユーザと制限の識別

それぞれのリソース制限に対して、制限を適用するオブジェクトを指定します。

次の項目にリソース制限を適用できます。

- 特定のログインで使用される全アプリケーション
- 特定のアプリケーションを使用する全ログイン
- 特定のログインで使用される特定のアプリケーション

ここでのアプリケーションとは、Adaptive Server 上で実行されるクライアント・プログラムで、特定のログインによりアクセスされるものです。Adaptive Server 上でアプリケーションを実行するには、cs_config (Open Client™ Client-Library™ アプリケーションの 1 つ) を使用して CS_APPNAME 接続プロパティで名前を指定するか、Open Client DB-Library™ の DBSETLAPP 関数を使用して名前を指定します。サーバで実行中の名前付きアプリケーションをすべて表示するには、master..sysprocesses テーブルから program_name カラムを選択します。

CS_APPNAME 接続プロパティの詳細については、『Open Client Client-Library/C リファレンス・マニュアル』を参照してください。DBSETLAPP 関数の詳細については、『Open Client DB-Library/C リファレンス・マニュアル』を参照してください。

## リソース使用量が多いユーザの識別

リソース制限を実施する前に、sp_reportstats を実行します。このプロシージャからの出力を参考にして、システム・リソースの使用量が多いユーザを調べることができます。次に例を示します。

```
    sp_reportstats
Name    Since         CPU     Percent CPU  I/O     Percent I/O
------  -----------   -----   ------------ -----   -------------
probe   jun 19 2007   0       0%           0       0%
julie   jun 19 2007   10000   24.9962%     5000    24.325%
jason   jun 19 2007   10002   25.0013%     5321    25.8866%
ken     jun 19 2007   10001   24.9987%     5123    24.9234%
kathy   jun 19 2007   10003   25.0038%     5111    24.865%

    Total CPU   Total I/O
    ---------   ---------
    40006       20555
```

I/O カラムと Percent I/O カラムでは、各ユーザの使用量はほぼ均等です。チャージバック・アカウンティングの詳細については、『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## リソース使用量が多いアプリケーションの識別

システム上で実行中のアプリケーションと、そのアプリケーションを実行しているユーザを調べるには、master データベースの sysprocesses システム・テーブルに対してクエリを実行します。

次のクエリにより、クライアント・プログラムのうち、名前が Adaptive Server に渡されているのは isql、payroll、perl、acctng だけであることがわかります。

```
    select spid, cpu, physical_io,
      substring(user_name(uid),1,10) user_name,
```

```
                          hostname, program_name, cmd
                        from sysprocesses
spid  cpu    physical_io  user_name hostname program_name cmd
----  ---    -----------  --------- -------- ------------ ------
  17    4          12748  dbo       sabrina  isql         SELECT
 424    5              0  dbo       HOWELL   isql         UPDATE
 526    0            365  joe       scotty   payroll      UPDATE
 568    1           8160  dbo       smokey   perl         SELECT
 595   10              1  dbo       froth    isql         DELETE
 646    1              0  guest     walker   isql         SELECT
 775    4          48723  joe_user  mohindra acctng       SELECT

(7 rows affected)
```

sysprocesses は現在のプロセスをレポートするために動的に作成されるので、クエリを繰り返し実行すると結果はそのたびに異なります。このクエリを終日定期的に実行して、使用しているシステムで実行されているアプリケーションを識別します。

CPU と物理 I/O の値は定期的に syslogins システム・テーブルへフラッシュされますが、これによって、sp_reportstats で表示される値が増加します。

使用しているシステムで実行されているアプリケーションを識別したら、showplan と statistics io を使用して、アプリケーションのクエリのリソース使用量を評価します。

## 制限タイプの選択

制限するユーザとアプリケーションを決定したら、リソース制限のタイプを選択します。

表 1-1 に、制限タイプの機能とスコープ、特定のクエリにその制限タイプの効果があるかどうかを判断するための支援ツールを示します。特定のユーザとアプリケーションに対して、複数の異なるタイプの制限を適用することもできます。「制限タイプの理解」(11 ページ) を参照してください。

**表 1-1: リソース制限のタイプ**

| 制限タイプ | 適するクエリ | リソース使用量の測定方法 | スコープ | 適用される時期 |
|---|---|---|---|---|
| io_cost | 大量の論理読み込みや物理読み込みを必要とするクエリ | クエリを実行する前に set showplan on を使用して見積もり I/O コストを表示する。set statistics io on を使用して実際の I/O コストを取得する。 | クエリ | 実行前または実行時 |
| row_count | 大量の結果セットを返すクエリ | ロー・カウントの適切な制限を決定するには、*@@rowcount* グローバル変数を利用する。 | クエリ | 実行時 |

| 制限タイプ | 適するクエリ | リソース使用量の測定方法 | スコープ | 適用される時期 |
|---|---|---|---|---|
| elapsed_time | クエリが複雑であるため、またはサーバの負荷やロック待ちなどの外部要因のために、完了までに時間がかかるクエリ | クエリを実行する前に set statistics time on を使用して経過時間をミリ秒単位で表示する。 | クエリ・バッチまたはトランザクション | 実行時 |
| tempdb_space | ワーク・テーブルやテンポラリ・テーブルを作成するときに、tempdb の全領域を使用するクエリ | セッション当たりの tempdb の使用ページ数 | クエリ・バッチまたはトランザクション | 実行時 |
| idle_time | アクティブでないクエリ | 接続がアクティブでない時間 (秒数) | 個々のプロセス | 実行前 |

spt_limit_types システム・テーブルには、各制限タイプの情報が格納されています。

## 施行時間の決定

**施行時間**は、クエリ処理の中で Adaptive Server によって特定のリソース制限が適用されるフェーズです。リソース制限が適用されるのは次のときです。

- 実行前 – オプティマイザの I/O コスト見積もりに基づいて、実行前にリソース制限が適用されます。この制限のタイプを使用すると、コストが高くなる可能性のあるクエリの実行を防ぎます。実行前に制限できるリソース・タイプは I/O コストだけです。

  条件文の句内にあるデータ操作言語文の I/O コストを評価するとき、各データ操作言語文は個々に考慮されます。実際に実行される句が 1 つだけであっても、すべての文が評価されます。

  実行前のリソース制限の制限スコープとして可能なものはクエリだけです。つまり、コンパイル時に制限されるリソースの値の計算とモニタは、クエリ単位でのみ行われます。

  トリガ内の文に対しては、実行前のリソース制限は施行されません。

- 実行時 – 実行時にリソース制限が適用されます。通常は、クエリがサーバやオペレーティング・システムのリソースを独占しないようにするために使用します。実行時制限は、実行前制限に比べて使用するリソースの量が増える (CPU 時間と I/O が増加する) ことがあります。

## リソース制限のスコープの決定

*scope* パラメータ (たとえば、sp_add_resource_limit *scope*、sp_help_resource_limit *scope* など) は、Transact-SQL 文に制限が適用される持続時間を指定します。指定できる制限スコープは、クエリ、クエリ・バッチ、トランザクションの 3 つです。

- クエリ － select、insert、update など、サーバにアクセスする単一の Transaction-SQL 文にリソース制限が適用されます。これらの文がクエリ・バッチ内で発行されたときは、各文が個別に評価されます。

  Adaptive Server は、ストアド・プロシージャを一連のデータ操作言語文と見なします。ストアド・プロシージャ内の文ごとに、リソース制限を評価します。あるストアド・プロシージャが別のストアド・プロシージャを実行する場合、ネストされたストアド・プロシージャ内の各データ操作言語文の評価は、内側のネスト・レベルで行われます。

  実行前リソース制限のスコープがクエリのとき、Adaptive Server は、ネスト・レベルごとに制限をチェックします。次のネスト・レベルに進むたびに、そのネスト・レベルの文を実行する前に、各データ操作言語文のリソース使用量の見積もりとアクティブなリソース制限とを比較します。そのネスト・レベルのデータ操作言語クエリのうち、リソース使用量見積もりがアクティブなリソース制限の制限値を超えるものが 1 つ以上あるときは、リソース制限違反となります。この場合、違反したリソース制限に対応したアクションが実行されます。

  実行時のリソース制限のスコープがクエリの場合は、各データ操作言語クエリの累積リソース使用量とリソース制限が比較されます。クエリのリソース使用量がアクティブな実行時リソース制限の制限値を超えると、制限違反になります。この場合も、違反したリソース制限に対応したアクションが実行されます。

- クエリ・バッチ － 1 つ以上の Transact-SQL 文で構成されます。たとえば、isql では、単一の go コマンド・ターミネータによって実行されるクエリのグループが 1 つのクエリ・バッチになります。

  クエリ・バッチはネスト・レベル 0 で始まります。ストアド・プロシージャを呼び出すたびにネスト・レベルが最大ネスト・レベルまで 1 ずつ増えます。ストアド・プロシージャから戻るたびに、ネスト・レベルが 1 ずつ減ります。

  スコープとしてクエリ・バッチを指定できるのは実行時のリソース制限だけです。

  スコープがクエリ・バッチである実行時リソース制限は、クエリ・バッチの文の累積リソース使用量と比較されます。クエリ・バッチのリソース使用量が、アクティブな実行時リソース制限の制限値を超えているときは、制限違反となります。この場合、違反したリソース制限に対応したアクションが実行されます。

- トランザクション － スコープがトランザクションである制限は、そのトランザクションのすべてのネスト・レベルに適用され、トランザクションの累積リソース使用量と比較されます。

  トランザクションのリソース使用量が、アクティブな実行時のリソース制限の制限値を超えているときは、制限違反となります。この場合、違反したリソース制限に対応したアクションが実行されます。

スコープとしてトランザクションを指定できるのは、実行時のリソース制限だけです。

Adaptive Server がリソース制限を適用するとき、ネストされたトランザクションは認識されません。トランザクションに対するリソース制限は、*@@trancount* が 1 に設定されると開始し、*@@trancount* が 0 に設定されると終了します。

- セッション ― アイドル時間の制限は、制限がアクティブであるセッションに適用されます。

# 制限タイプの理解

リソース制限は、さまざまな方法でリソース使用量を制限できます。

- I/O コスト
- 経過時間
- ロー・カウント
- tempdb 領域の使用状況
- アイドル時間

## I/O コストの制限

I/O コストは、クエリ処理中に使用される論理／物理アクセス (読み込み) の回数に基づいて計算されます。最も効率的な処理プランを実行前に決定するために、Adaptive Server オプティマイザは論理リソースと物理リソースを使用して、見積もり I/O コストを計算します。

Adaptive Server は、オプティマイザのコスト計算式の結果を「単位なし」の数値として使用します。つまり、秒やミリ秒などの単一の測定単位に基づいているとはかぎらない値です。

リソース制限を設定するには、これらの制限がどのように実行時のシステム・オーバヘッドに変換されるかを理解する必要があります。たとえば、*x* 回の論理 I/O と *y* 回の物理 I/O のコストを持つクエリが運用サーバに与える影響を知っておく必要があります。

io_cost を制限すると、大量の結果セットを返すクエリなど、I/O 集約クエリを制御できます。ただし、大規模テーブルの全ローを返す単純なクエリを実行するときに、そのテーブルのサイズに関する最新の統計情報がない場合は、そのクエリが io_cost リソース制限を超過することをオプティマイザは予測できません。大量の結果セットを返すクエリの実行を防ぐには、row_count に関するリソース制限を作成します。

Adaptive Server が並列クエリ処理を行うように設定されているとき、分割テーブルの I/O コスト制限トラッキング精度は、非分割テーブルの場合よりも低下することがあります。『パフォーマンス&チューニング・シリーズ：クエリ処理と抽象プラン』の「第 5 章 並列クエリ処理」を参照してください。

## I/O コストの識別

I/O コストの適切な制限を設定するには、代表的なクエリを選び、set コマンドを使用して必要な論理読み込み回数と物理読み込み回数を調べます。

- set showplan on は、オプティマイザのコスト見積もりを表示します。この情報を使用して、実行前のリソース制限を設定します。オプティマイザが見積もったクエリの I/O コストが制限値を超えると、実行前リソース制限の違反となります。このような制限により、高コストのクエリの実行を防ぎます。
- set statistics io on は、実際の論理読み込みと物理読み込みの回数を表示します。この情報を使用して、実行時のリソース制限を設定します。クエリの実際の I/O コストが制限値を超えると、実行時リソース制限違反となります。

実際の I/O コストの統計には、クエリに使用されるユーザ・テーブルとワーク・テーブルのアクセス・コストだけが含まれます。Adaptive Server は、内部処理で他のテーブルを使用することがあります。たとえば、統計を出力するために sysmessages にアクセスします。したがって、統計では示されていなくても、クエリの実際の I/O コストが制限を超える場合があります。

オプティマイザがクエリのコストを計算するとき、必要となったページに初めてアクセスするときに 1 回の物理 I/O が行われ、それ以降のアクセスではキャッシュが使用されることを前提としています。実際の I/O コストは、いくつかの理由により、オプティマイザの見積もりコストと異なる場合があります。

一部のページが既にキャッシュにある場合や、統計が正しくない場合は、見積もりコストが実際のコストよりも高くなります。オプティマイザが 16K の I/O を選択し、一部のページが 2K キャッシュ・プールにある場合は、大量の 2K の I/O が必要になるため、見積もりコストは実際のコストよりも低くなる可能性があります。また、大規模なジョインによってキャッシュ内のページが強制的にディスクにフラッシュされた場合は、2 回目のアクセスであってももう一度物理 I/O が必要となることがあります。

分散統計や密度統計が最新のものでない場合や使用できない場合は、オプティマイザの見積もりは正確ではありません。

## カーソルの I/O コストの計算

カーソル処理のコスト見積もりは、declare cursor の実行時に計算されます。ただし、実行カーソルの場合は、カーソルがオープンしたときに計算されます。

I/O コストに対する実行前のリソース制限は、どのタイプのカーソルの場合も open *cursorname* の実行時に施行されます。ユーザがカーソルをオープンしようとするたびに、オプティマイザによって制限値が再計算されます。

実行時のリソース制限は、カーソルがオープンしてからクローズするまでの間のカーソルの累積 I/O コストに適用されます。カーソルがオープンするたびに、オプティマイザは I/O 制限を再計算します。

『Transact-SQL ユーザーズ・ガイド』の「第 18 章 カーソル：データのアクセス」の章を参照してください。

### *io_cost* 制限タイプのスコープ

I/O コストを制限するリソース制限は、単一クエリにのみ適用されます。クエリ・バッチで複数の文を発行するときは、クエリごとに I/O 使用量が評価されます。「リソース制限のスコープの決定」(9 ページ) を参照してください。

## 経過時間の制限

経過時間とは、クエリ・バッチまたはトランザクションの実行に要した時間を秒数で表したものです。経過時間は、クエリの複雑さ、サーバの負荷、ロックの待ち時間などの要因によって決まります。

経過時間に対する適切な制限を決定するには、set statistics time を使用して集めた情報を利用してください。経過時間リソースを制限できるのは、実行時だけです。

set statistics time を on にした状態で、代表的なクエリを実行して、表示されるミリ秒単位の処理時間を確認します。リソース制限を作成するときは、ミリ秒単位の数値を秒単位に変換してください。

経過時間のリソース制限は、データ操作言語文だけではなく、制限のスコープ (クエリ・バッチまたはトランザクション) 内のすべての SQL 文に適用されます。対応するスコープの経過時間が制限値を超えると、リソース制限違反になります。

ストアド・プロシージャやトランザクションがネストしている場合も、それぞれに別の経過時間制限が適用されるのではありません。つまり、あるトランザクションが別のトランザクション内にネストしている場合、経過時間制限は外側のトランザクションに適用されます。外側のトランザクションの経過時間には、内側のトランザクションの経過時間全体が含まれます。したがって、トランザクションの実行時間を測定すると、その実行時間には、ネストされているすべてのトランザクションの実行時間が含まれます。

### *elapsed_time* 制限タイプのスコープ

経過時間を制限するリソース制限のスコープは、クエリ・バッチまたはトランザクションです。「リソース制限のスコープの決定」(9 ページ ) を参照してください。

## 結果セットのサイズの制限

row_count 制限タイプは、ユーザに返されるローの数を制限します。select 文で返されるローの数が制限値を超えると、制限違反となります。

リソース制限のアクションが警告の発行である場合に、クエリがローの制限を超えたときは、選択したローがすべて返され、その後で次の例のように制限値を示す警告が出力されます。

```
Row count exceeded limit of 50.
```

リソース制限のアクションがクエリ・バッチまたはトランザクションのアボートまたはセッションの強制終了である場合に、クエリがローの制限を超えたときは、選択したローのうち一部だけが返され、クエリ・バッチ、トランザクション、またはセッションがアボートされます。この場合、Adaptive Server に次のようなメッセージが表示されます。

```
Row count exceeded limit of 50.
Transaction has been aborted.
```

row_count 制限タイプは、実行時にすべての select 文に適用されます。実行前に、返されるロー数の見積もりを制限することはできません。

ロー・カウントの適切な制限を決定するには、*@@rowcount* グローバル変数を利用する。代表的なクエリを実行した後でこの変数を選択すると、クエリによって返されたローの数がわかります。

ロー・カウント制限は、カーソルがオープンしてからクローズするまでの間にそのカーソルによって返されたローの累積数に適用されます。カーソルがオープンするたびに、オプティマイザは row_count 制限を再計算します。

### *row_count* 制限タイプのスコープ

ロー・カウントを制限するリソース制限は単一のクエリだけに適用され、クエリ・バッチやトランザクションで返される累積ロー数には適用されません。「リソース制限のスコープの決定」(9 ページ) を参照してください。

## tempdb 領域の使用に対する制限を設定する

tempdb_space リソース制限は、1 つのセッションで使用できる tempdb データベースのページ数を制限します。指定された制限を超過した場合に、ユーザのセッションを終了させることも、バッチまたはトランザクションをアボートすることもできます。

並列で実行されるクエリの場合、tempdb_space リソース制限は並列スレッドに均等に配分されます。たとえば、tempdb_space のリソース制限が 1500 ページに設定されていて、ユーザが 3 分割並列処理で以下を実行した場合、各並列スレッドは tempdb に 500 ページまで作成できます。

```
select into #temptable from partitioned_table
```

システム管理者やデータベース管理者は、sp_add_resource_limit を使用して tempdb_space の制限を設定し、sp_drop_resource_limit を使用して tempdb_space の制限を削除します。

### アイドル時間の制限

アイドル時間は、ユーザ入力を待っている、接続がアクティブでない状態の時間を秒数で表したものです。接続がアクティブでなくても、サーバ・リソースは使用されているため、同じサーバで実行されている他のアクティブなプロセスをブロックする可能性のあるリソース ( ロックなど ) を保持している場合もあります。

idle_time を使用すると、アイドル接続の時間制限を設定できます。接続のアイドル状態が設定された制限を超えると、接続を実行しているプロセスが停止されるか、警告が発行されます。

構文は次のとおりです。

sp_add_resource_limit user, *application*, *time_range*, idle_time , *kill_time*, *enforcement_time*, *action*, *scope*

たとえば、次のコマンドを使用すると、ユーザ “sa” に対して isql 接続の新しい制限が作成されます。

```
sp_add_resource_limit sa, isql, 'at all times', idle_time, 10,
2, 4, 8
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

## リソース制限の作成

リソース制限を新しく作成するには、sp_add_resource_limit を使用します。

sp_add_resource_limit *name*, *appname*, *rangename*, *limittype*, *limit_value*, *enforced*, *action*, *scope*

『リファレンス・マニュアル：プロシージャ』の「sp_add_resource_limit」を参照してください。

### リソース制限の例

ここでは、リソース制限の設定例を紹介します。

#### 例

**例 1**　この例では name パラメータに NULL を指定しているため、作成されるリソース制限は payroll アプリケーションのすべてのユーザに適用されます。

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,
elapsed_time, 120, 2, 1, 2
```

この制限が有効となるのは、時間範囲 tu_wed_7_10 の間です。制限タイプは elapsed_time (経過時間) で、120 秒に設定しています。elapsed_time が施行されるのは実行時だけであるので、*enforced* パラメータは 2 に設定します。警告が発行されるように、*action* パラメータは 1 に設定します。最後のパラメータ *scope* で制限のスコープを 2、つまりクエリ・バッチに指定します。この例では、実行したクエリ・バッチの経過時間が 120 秒を超えると、警告が発行されます。

**例 2** この例では、時間範囲 saturday_night の間に “joe_user” が実行するすべてのアドホック・クエリとアプリケーションに適用されるリソース制限を作成します。

```
sp_add_resource_limit joe_user, NULL, saturday_night,
    row_count, 5000, 2, 3, 1
```

クエリ (*scope* = 1) が返すローの数が 5000 を超えると、トランザクションがアボートされます (*action* = 3)。このリソース制限は実行時に施行されます (*enforced* = 2)。

**例 3** この例でも、“joe_user” が実行するすべてのアドホック・クエリとアプリケーションに適用されるリソース制限を作成します。

```
sp_add_resource_limit joe_user, NULL, "at all times",
    io_cost, 650, 1, 3, 1
```

ただし、このリソース制限では、デフォルトの時間範囲 “at all times” を指定します。オプティマイザによって見積もられたクエリ (*scope* = 1) の io_cost が指定値の 650 を超える場合は、トランザクションはアボートされます (*action* = 3)。このリソース制限は実行前に施行されます (*enforced* = 1)。

---

**注意** 時間制限の制限値に達すると、Adaptive Server は現在のトランザクションを終了しますが、別の SQL コマンドまたはバッチを実行するまで 1105 エラー・メッセージは表示されません。このメッセージは接続を再び試みたときに初めて表示されます。

---

# 既存の制限の情報取得

既存のリソース制限についての情報を取得するには、sp_help_resource_limit を使用します。

『リファレンス・マニュアル：プロシージャ』の「sp_help_resource_limit」を参照してください。

## 既存の全リソース制限のリスト

パラメータを指定せずに sp_help_resource_limit を実行すると、サーバ内のすべてのリソース制限が表示されます。次に例を示します。

```
                        sp_help_resource_limit
name      appname rangename rangeid limitid limitvalue enforced  action scope
----      ------- --------- ------- ------- ---------- --------  ------ -----
NULL      acctng  evenings        4       2        120         2       1     2
stein     NULL    weekends        1       3       5000         2       1     1
joe_user  acctng  bus_hours       5       3       2500         2       2     1
joe_user  finance bus_hours       5       2        160         2       1     6
wong      NULL    mornings        2       3       2000         2       1     1
wong      acctng  bus_hours       5       1         75         1       3     1
```

rangeid カラムには、rangename カラムの名前に対応する systimeranges.id の値が出力されます。limitvalue カラムには、sp_add_resource_limit または sp_modify_resource_limit を使用して設定した値が出力されます。表 1-2 は、limitid、enforced、action、scope の各カラムの説明を示しています。

**表 1-2: sp_help_resource_limit 出力の値**

| カラム | 意味 | 値 |
|---|---|---|
| limitid | 制限タイプ | 1 － I/O コスト<br>2 － 経過時間<br>3 － ロー・カウント |
| enforced | 制限の施行時間 | 1 － 実行前<br>2 － 実行時<br>3 － 両方 |
| action | 制限に違反したときに実行されるアクション | 1 － 警告の発行<br>2 － クエリ・バッチのアボート<br>3 － トランザクションのアボート<br>4 － セッションの強制終了 |
| scope | 制限のスコープ | 1 － クエリ<br>2 － クエリ・バッチ<br>4 － トランザクション<br>6 － クエリ・バッチとトランザクション |

システム管理者が sp_help_resource_limit を実行するときにログイン名を指定すると、そのログインに対するすべてのリソース制限が表示されます。指定したユーザに固有のリソース制限だけではなく、指定したアプリケーションの全ユーザに関係するリソース制限もすべて出力されます。これは、指定したユーザも全ユーザに含まれるためです。

たとえば、次の例では、“joe_user” に適用されるすべてのリソース制限が出力されます。acctng アプリケーションの全ユーザを対象として定義されているリソース制限もあり、その制限も出力に含まれています。

```
                    sp_help_resource_limit joe_user
name     appname rangename rangeid limitid limitvalue enforced  action scope
----     ------- --------- ------- ------- ---------- --------  ------ -----
NULL     acctng  evenings        4       2        120         2       1     2
joe_user acctng  bus_hours       5       3       2500         2       2     1
joe_user finance bus_hours       5       2        160         2       1     6
```

# リソース制限の修正

sp_modify_resource_limit を使用して、制限値を変更することや、制限違反が発生したときに実行されるアクションを変更することができます。制限が適用されるログインやアプリケーションを変更することはできません。また、新しい時間範囲、制限タイプ、適用時間、またはスコープを指定することはできません。

sp_modify_resource_limit の構文は、次のとおりです。

```
sp_modify_resource_limit name, appname, rangename, limittype,
    limitvalue, enforced, action, scope
```

『リファレンス・マニュアル：プロシージャ』の「sp_modify_resource_limit」を参照してください。

# リソース制限の削除

リソース制限を Adaptive Server から削除するには、sp_drop_resource_limit を使用します。

構文は次のとおりです。

```
sp_drop_resource_limit {name , appname } [, rangename, limittype,
    enforced, action, scope]
```

制限をユニークに識別するために必要な情報をすべて指定してください。*name* と *appname* のどちらかには null 以外の値を指定してください。

『リファレンス・マニュアル：プロシージャ』の「sp_drop_resource_limit」を参照してください。

# リソース制限の優先度

時間範囲とリソース制限については、次のような優先度の規則があります。

## 時間範囲

現在アクティブな時間範囲の間の個々のログイン・セッションに対して、制限タイプ、施行時間、スコープの組み合わせごとにアクティブにできる制限は 1 つだけです。アクティブな制限を決定する優先度の規則は、次のとおりです。

- ログイン ID に対する制限が、時間範囲 “at all times” と現在アクティブな時間範囲のどちらについても定義されていない場合は、アクティブな制限はありません。
- ログインに対する制限が、時間範囲 “at all times” と特定の時間範囲の両方について定義されている場合は、特定時間範囲についての制限が優先されます。

## リソースの制限

リソース制限を特定するにはユーザのログイン名とアプリケーション名の一方または両方が使用されるので、ログイン・セッションに適用される制限を探すために Adaptive Server が sysresourcelimits テーブルをスキャンするとき、事前に定義された優先度に従って検索を行います。ログイン名とアプリケーション名のペアの優先度を一致する順序ごとに次の表に示します。

| レベル | ログイン名 | アプリケーション名 |
|---|---|---|
| 1 | joe_user | payroll |
| 2 | NULL | payroll |
| 3 | joe_user | NULL |

ある優先度レベルで一致するものが 1 つ以上ある場合は、それ以降のレベルは検索されません。これによって、異なるログイン／アプリケーションの組み合わせに対し、同様の制限値が適用されないようにします。

どのレベルでも一致が見つからない場合は、そのセッションでは制限は施行されません。

第 2 章

# データベース・デバイスのミラーリング



## ディスク・ミラーリング

「ディスク・ミラーリング」によって、メディア障害が起きたときのノンストップ・リカバリが実現します。disk mirror コマンドを実行すると、Adaptive Server のデータベース・デバイスが複製されます。つまり、そのデバイスに書き込まれるデータは、すべて別の物理デバイスにコピーされます。一方のデバイスに障害が起きても、もう一方のデバイスにすべてのトランザクションの最新コピーが保管されています。

ミラーリングされたデバイスの読み込みまたは書き込みに失敗した場合は、Adaptive Server はその不良デバイスの「ミラーリングを解除」し、エラー・メッセージを表示します。Adaptive Server は、ミラーリングが解除された状態で処理を続けます。

## ミラーリングの対象の決定

デバイスのミラーリングを行うかどうかを決定するときは、システムのダウン時間のコスト、パフォーマンスの低下、記憶メディアのコストといった要因を考慮する必要があります。これらの点を考慮して、トランザクション・ログだけをミラーリングするか、あるサーバ上のすべてのデバイスをミラーリングするか、または選択したデバイスだけをミラーリングするかを決定します。

**注意** ダンプ・デバイスのミラーリングはできません。

デフォルト・リスト内のデータベース・デバイスが create database コマンドや alter database コマンドの影響を受けたときも、データベースが保護されるように、すべてのデフォルトのデータベース・デバイスをミラーリングしてください。

ユーザ・データベース・デバイスをミラーリングするだけでなく、トランザクション・ログを別のデータベース・デバイスに保管するようにしてください。保護をより強化するには、トランザクション・ログに使用されるデータベース・デバイスのミラーを作成します。

データベースのトランザクション・ログ (syslogs システム・テーブル) を、残りのデータベースの部分が格納されているデバイス以外のデバイスに配置するには、データベースの作成時にデータベース・デバイスとログ・デバイスを指定します。また、alter database を使用して別のデバイスを追加してから、sp_logdevice を実行する方法もあります。

コストとパフォーマンスの間のトレードオフを考えるときは、次の事項について検討してください。

- リカバリの速度 − master データベースとユーザ・データベース (ログを含む) をミラーリングすると、ノンストップ・リカバリが可能になります。また、トランザクション・ログを再ロードしなくてもリカバリできます。
- 記憶領域 − 迅速なリカバリを行うには、完全冗長構成が必要となる (すべてのデータベースとログをミラーリングする) ため、必要なディスク領域が増えます。
- パフォーマンスへの影響 − ユーザ・データベースをミラーリングすると (図 2-2 (23 ページ) と 図 2-3 (24 ページ) を参照)、両方のディスクにトランザクションを書き込むための時間が長くかかります。

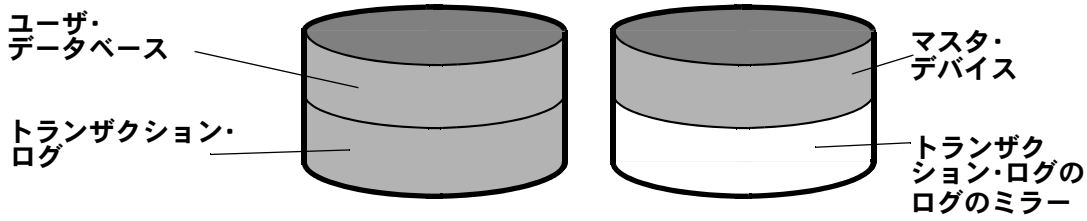
## 最小物理ディスク領域によるミラーリング

図 2-1 は、ハードウェア障害時のデータベース・リカバリを可能にする「最小保証構成」を示します。マスタ・デバイスと、ユーザ・データベースのトランザクション・ログのミラーは、1 つの物理ディスクの別々のパーティション内に保管されます。他方のディスクには、ユーザ・データベースとそのトランザクション・ログが、別々のディスク・パーティションに保管されています。

ユーザ・データベースが使用しているディスクに障害が起きた場合は、バックアップおよびミラーリングしたトランザクション・ログから、新しいディスク上にユーザ・データベースをリストアできます。

マスタ・デバイスが存在するディスクに障害が起きた場合は、master データベースのデータベース・ダンプからマスタ・デバイスをリストアし、ユーザ・データベースのトランザクション・ログを再ミラーリングします。

**図 2-1: 最小物理ディスク領域によるディスクのミラーリング**



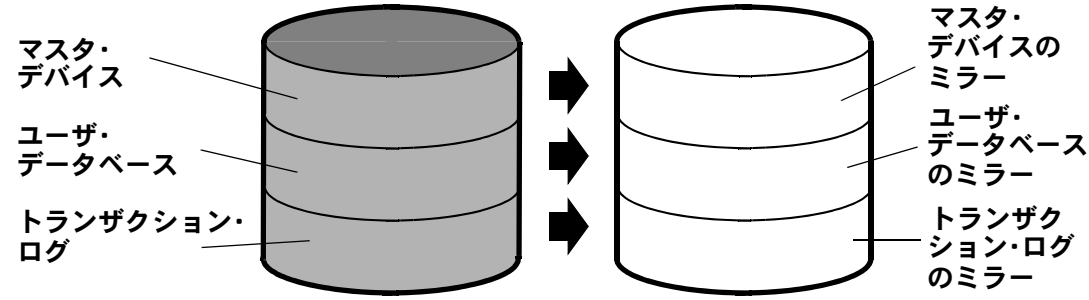

この構成では、必要なディスク領域の量が最小になりますが、トランザクション・ログのミラーリングによって完全なリカバリが保証されます。しかし、この構成では、master データベースとユーザ・データベースがミラーリングされておらず、バックアップを使用してリカバリを行わなければならないため、ノンストップ・リカバリを行うことはできません。

## ノンストップ・リカバリを可能にするためのミラーリング

図 2-2 は、別のミラーリング構成を示しています。この構成では、マスタ・デバイス、ユーザ・データベース、トランザクション・ログが同一の物理デバイスの別々のパーティションに保管され、2 番目の物理デバイスにそれぞれのミラーリングが作成されます。

図 2-2 の構成では、ハードウェア障害が起きた場合でもノンストップ・リカバリが可能です。プライマリ・ディスク上の master データベース、ユーザ・データベース、ログの作業コピーがすべてミラーリングされているため、どちらかのディスクで障害が起きても Adaptive Server のユーザの操作は中断されません。

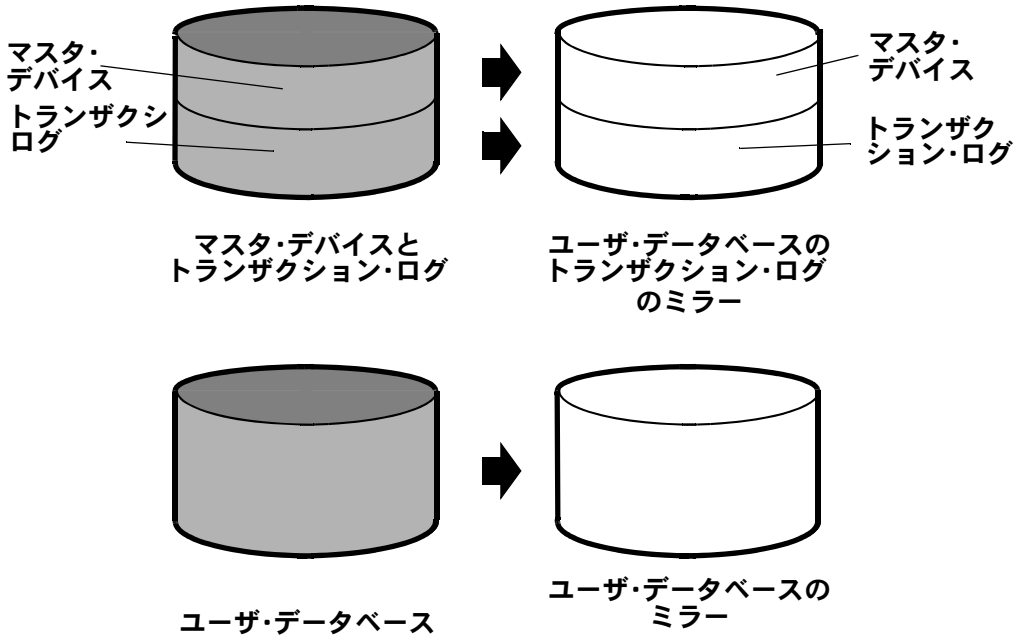
**図 2-2: 迅速なリカバリを行うためのディスク・ミラーリング**



この構成では、すべてのデータは 2 回書き込まれます。つまり、プライマリ・ディスクに 1 回、ミラーに 1 回書き込まれます。多数の書き込みを必要とするアプリケーションでは、ディスク・ミラーリングが原因で、ミラーリングを行わない場合よりも処理に時間がかかることがあります。

図 2-3 は、高レベルの冗長構成の例を示しています。この構成では、3 つのデータベース・デバイスをすべてミラーリングしますが、2 つではなく 4 つのディスクを使用します。この構成では、データベースのトランザクション・ログがユーザ・データベースとは別のデバイスに保管され、システムは少ないディスク・ヘッドの移動で両方のデバイスにアクセスできるため、書き込みトランザクションのパフォーマンスが向上します。

**図 2-3: ディスク・ミラーリング：別のディスクへのトランザクション・ログの保存**



# ミラーリングが無効にならない条件

ミラーが無効になるのは、ミラーリングされたデバイスで I/O エラーが発生した場合のみです。たとえば、ディスク上の不良ブロックに書き込もうとすると、これによって発生するエラーによって、そのデバイスのミラーリングは無効になります。ただし、影響を受けないミラーでは、処理は中断することなく続行します。

次の条件では、ミラーは無効になりません。

- デバイス上の未使用ブロックが不良である場合。その不良ブロックに Adaptive Server がアクセスするまでは I/O エラーは検出されず、ミラーリングが無効になることはありません。
- デバイス上のデータが上書きされる場合。これは、ミラーリングされたデバイスが UNIX ファイル・システムとしてマウントされている場合に、UNIX によって Adaptive Server のデータが上書きされるときに発生する可能性があります。この場合、データベースは破壊されますが、Adaptive Server が I/O エラーを検出しないため、ミラーリングは無効になりません。
- プライマリ・デバイスとセカンダリ・デバイスの両方に、不正なデータが書き込まれた場合。
- アクティブなデバイスに対するファイル・パーミッションが変更された場合。一方のデバイスに対するパーミッションを変更し、故意に I/O 障害を発生させて他方のデバイスのミラーリングを解除するという方法で、ディスクのミラーリングをテストすることをシステム管理者が考えるかもしれません。しかし、UNIX オペレーティング・システムは、デバイスをオープンした後はパーミッションを確認しません。したがって、そのデバイスが次回起動されるまで I/O 障害は発生しません。

ディスク・ミラーリングは、データベースの破壊を検出あるいは防止するようには設計されていません。前述のような状況では、データベースが破壊される可能性があります。したがって、すべてのデータベースに対して dbcc checkalloc や dbcc checkdb などの整合性チェックを定期的に実行してください。「第 11 章 データベースの一貫性の検査」を参照してください。

# ディスク・ミラーリングのコマンド

ディスク・ミラーリングを制御するコマンドには、disk mirror、disk unmirror、disk remirror があります。これらのコマンドはすべてデバイスの使用中に発行できるので、データベースが使用中であってもデータベース・デバイスのミラーリングの開始や停止が可能です。

---

**注意** disk mirror、disk unmirror、disk remirror の各コマンドは、master データベース内の sysdevices テーブルを変更します。これらのコマンドを発行した後は、master データベースが損傷しても確実にリカバリできるようにするために、必ず master データベースのダンプを実行してください。

---

## ミラーリングの初期化

disk mirror コマンドを実行すると、ディスク・ミラーリングが開始します。disk init コマンドを使用してミラーリング・デバイスを初期化しないでください。データベース・デバイスとそのミラーによって、1 つの論理デバイスが構成されます。disk mirror コマンドは、ミラー名を sysdevices テーブルの mirrorname カラムに追加します。

disk mirror の構文は次のとおりです。

```
disk mirror
    name = "device_name" ,
    mirror = "physicalname"
    [ , writes = { serial | noserial }]
```

## デバイスのミラーリング解除

2 つの物理デバイスのうちの一方に障害が発生すると、ディスク・ミラーリングは自動的に停止します。ミラーリングされたデバイスの読み込みまたは書き込みを正しく実行できなかった場合は、Adaptive Server はエラー・メッセージを出力します。Adaptive Server は、ミラーリングが解除された状態で処理を継続します。ミラーリングを再開するには、ディスクを再度ミラーリングしてください。

ハードウェアのメンテナンス中にミラーリング処理を停止するには、次の disk unmirror コマンドを使用します。

```
disk unmirror
    name = "device_name"
    [, side = { "primary" | secondary }]
    [, mode = { retain | remove }]
```

## システム・テーブルへの影響

mode オプションを指定すると、ミラーリングが無効化されたことを示すように sysdevices の status カラムが変更されます (『システム管理ガイド第 1 巻』の「第 7 章 データベース・デバイスの初期化」を参照)。sysdevices の phyname カラムと mirrorname カラムへのこのオプションの影響は、side 引数の値によっても異なります。表 2-1 を参照してください。

**表 2-1: ディスク・ミラーリングのコマンドの mode オプションと side オプションの影響**

| | | *side* | |
|---|---|---|---|
| | | primary | secondary |
| *mode* | remove | mirrorname 内の名前が phyname に移動され、mirrorname が null に設定される。status が変更される。 | mirrorname 内の名前が削除される。status が変更される。 |
| | retain | 名前は変更されない。status は、停止されたデバイスを示すように変更される。 | |

次の例では、プライマリ・デバイスのミラーリングが中断します。

```
disk unmirror
  name = "tranlog",
  side = "primary"
```

## ミラーリングの再開

デバイスの障害が原因で、または disk unmirror によって中断していたミラーリング処理を再開するには、disk remirror を使用します。構文は次のとおりです。

```
disk remirror
   name = "device_name"
```

このコマンドによって、データベース・デバイスはそのミラーにコピーされます。

## *waitfor mirrorexit*

ディスク障害が発生するとシステムのセキュリティが損なわれる可能性があるため、ディスクのミラーリング解除時に特定のタスクを実行するように、次のように waitfor mirrorexit コマンドをアプリケーションに組み込むことができます。

```
begin
    waitfor mirrorexit
  commands to be executed
end
```

指定するコマンドは、アプリケーションによって異なります。更新を実行するアプリケーションに警告を追加することや、sp_dboption を使用してディスクのミラーリング解除時は特定のデータベースを読み込み専用にすることなどが考えられます。

---

**注意** ディスクのミラーリングが解除されても、そのミラーリング・デバイスに対する I/O が試行されるまでは、Adaptive Server はミラーリング解除を認識しません。ミラーリングされたデータベース上で I/O が発生するのは、チェックポイントのとき、または Adaptive Server のバッファをディスクに書き込まなければならないときです。ミラーリングされたログに対する I/O は、データ修正を実行するトランザクションのコミットなどのログへの書き込み、チェックポイント、またはデータベース・ダンプの処理時に発生します。

---

waitfor mirrorexit、コンソールに出力されるエラー・メッセージ、およびミラー障害に関するエラー・ログは、上述のイベントによってのみアクティブになります。

## マスタ・デバイスのミラーリング

UNIX 環境で、master データベースを格納しているデバイスをミラーリングする場合は、サーバの起動時にミラーリング・デバイスも起動されるように、Adaptive Server の runserver ファイルを編集してください。

UNIX の場合は、次のように、-r フラグとミラーリング・デバイス名を追加します。

```
dataserver -d /dev/rsd1f -r /dev/rs0e -e/sybase/install/errorlog
```

Windows 環境でマスタ・デバイスをミラーリングする方法の詳細については、『ASE ユーティリティ・ガイド』を参照してください。

## デバイスとミラーリングの情報の取得

システム上のすべての Adaptive Server デバイス (ユーザ・データベース・デバイス、そのミラー、およびダンプ・デバイス) のレポートを取得するには、sp_helpdevice を実行します。

# ディスク・ミラーリングのチュートリアル

この項では、ディスク・ミラーリング・コマンドの使用方法と master..sysdevices のカラムへの影響について順を追って説明します。以下の手順では、sysdevices 内の各エントリの status の番号と、その番号を 16 進数で表したものをカッコで囲んで表記しています。

1　次のコマンドを使用して、新しいテスト・デバイスを初期化します。

```
disk init name = "test",
physname = "/usr/sybase/test.dat",
size=5120
```

master..sysdevices のカラムに次の値が挿入されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   NULL                  16386
```

ステータス 16386 は、デバイスが物理デバイス (2、0x00000002) であり、書き込みは UNIX ファイルに対して行われる (16384、0x00004000) ことを示します。mirrorname カラムは null であるため、このデバイスのミラーリングは行われません。

2　次のコマンドを使用して、テスト・デバイスをミラーリングします。

```
disk mirror name = "test",
mirror = "/usr/sybase/test.mir"
```

master..sysdevices のカラムは、次のように変更されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

ステータス 17122 は、このデバイスでのミラーリングが現在有効である (512、0x00000200) ことを示します。また、読み込みがミラーリングされ (128、0x00000080)、書き込みが UNIX ファイル・デバイスにミラーリングされ (16384、0x00004000)、デバイスがミラーリングされ (64、0x00000040)、逐次モードである (32、0x00000020) ことを示します。このデバイスは物理ディスク (2、0x00000002) です。

3　ミラーリング・デバイス ( セカンダリ側 ) を無効にします。ただしそのミラーは保持します。

```
disk unmirror name = "test",
side = secondary, mode = retain
```

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  18658
```

ステータス 18658 は、デバイスがミラーリングされ (64、0x00000040)、ミラーリング・デバイスが保持されていますが (2048、0x00000800)、ミラーリングは無効になっていて (512 ビットがオフ)、プライマリ・デバイスだけが使用されている (256 ビットがオフ) ことを示します。また、読み込みがミラーリングされ (128、0x00000080)、書き込みが UNIX ファイル・デバイスにミラーリングされ (16384、0x00004000)、逐次モードである (32、0x00000020) ことを示します。このデバイスは物理ディスク (2、0x00000002) です。

4 テスト・デバイスを再ミラーリングします。

```
disk remirror name = "test"
```

master..sysdevices のカラムは、再び次のように設定されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

ステータス 17122 は、このデバイスでのミラーリングが現在有効である (512、0x00000200) ことを示します。また、読み込みがミラーリングされ (128、0x00000080)、書き込みが UNIX ファイル・デバイスにミラーリングされ (16384、0x00004000)、デバイスがミラーリングされ (64、0x00000040)、逐次モードである (32、0x00000020) ことを示します。このデバイスは物理ディスク (2、0x00000002) です。

5 テスト・デバイス (プライマリ側) を無効にします。ただしそのミラーは保持します。

```
disk unmirror name = "test",
side = "primary", mode = retain
```

master..sysdevices のカラムは、次のように変更されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  16866
```

ステータス 16866 は、デバイスがミラーリングされているが (64、0x00000040)、ミラーリングは無効化されており (512 ビットがオフ)、セカンダリ・デバイスだけが使用されている (256, 0x00000100) ことを示します。また、読み込みがミラーリングされ (128、0x00000080)、書き込みが UNIX ファイル・デバイスにミラーリングされ (16384、0x00004000)、逐次モードである (32、0x00000020) ことを示します。このデバイスは物理ディスク (2、0x00000002) です。

6 テスト・デバイスを再ミラーリングします。

```
disk remirror name = "test"
```

master..sysdevices のカラムは、再び次のように設定されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   /usr/sybase/test.mir  17122
```

ステータス 17122 は、このデバイスでのミラーリングが現在有効である (512、0x00000200) ことを示します。また、読み込みがミラーリングされ (128、0x00000080)、書き込みが UNIX ファイル・デバイスにミラーリングされ (16384、0x00004000)、デバイスがミラーリングされ (64、0x00000040)、逐次モードである (32、0x00000020) ことを示します。このデバイスは物理ディスク (2、0x00000002) です。

7　テスト・デバイス ( プライマリ側 ) を無効にして、ミラーリングを削除します。

```
disk unmirror name = "test", side = "primary",
mode = remove
```

master..sysdevices のカラムは、次のように変更されます。

```
name  phyname                mirrorname          status
test  /usr/sybase/test.dat   NULL                    16386
```

ステータス 16386 は、デバイスが物理デバイス (2、0x00000002) であり、書き込みは UNIX ファイルに対して行われる (16384、0x00004000) ことを示します。mirrorname カラムは null であるため、このデバイスのミラーリングは行われません。

8　テスト・デバイスを削除して、チュートリアルを完了します。

```
sp_dropdevice test
```

master..sysdevices から、テスト・デバイス用のエントリがすべて削除されます。

# ディスクのサイズ変更とミラーリング

disk resize は、ミラーリングが完全に無効になっている場合にのみ使用します。ミラーリングされているデバイスで disk resize コマンドを実行しようとすると、次のようなメッセージが表示されます。

```
disk resize can proceed only when mirroring is permanently
disabled. Unmirror secondary with mode = 'remove' and re-
execute disk resize command.
```

ミラーリングが一時的に停止しているときは、次の 2 つの状況が考えられます。

- プライマリ・デバイスがアクティブで、セカンダリ・デバイスが一時的に停止した状態。前述のエラーと同じエラーが発生します。
- プライマリ・デバイスが一時的に停止しており、セカンダリ・デバイスがアクティブな状態。次のようなエラー・メッセージが表示されます。

```
disk resize can proceed only when mirroring is
permanently disabled. Unmirror primary with mode =
```

```
'remove' and re-execute the command.
```

ミラーリング・デバイスのサイズを拡大するには、次の手順に従います。

1 デバイスのミラーリングを永久的に停止します。

2 プライマリ・デバイスのサイズを拡大します。

3 (ファイルの場合) ミラーリング・デバイスを物理的に削除します。

4 ミラーリングを再開します。

第 3 章

# メモリの設定



## Adaptive Server の使用可能メモリの決定

使用できるメモリが多ければ多いほど、Adaptive Server の内部バッファとキャッシュ用のリソースも多くなります。キャッシュに使用できるメモリが十分に大きければ、Adaptive Server がデータやプロシージャ・プランをディスクから読み込む回数は少なくなります。

コンピュータで使用可能なメモリの最大量を使用するように Adaptive Server を設定しても、パフォーマンスが低下することはありません。ただし、Adaptive Server 以外がそのシステム上で必要とするメモリ量を調べて、それ以外の使用可能なメモリを Adaptive Server が使用するように設定してください。設定された量のメモリを確保できない場合は、Adaptive Server は起動できません。

システム上で Adaptive Server が使用できるメモリの最大量を調べるには、次の手順に従います。

1 使用しているコンピュータ・システムの物理メモリの合計量を算定します。

2 合計物理メモリからオペレーティング・システム用に必要なメモリを差し引きます。

3 同じマシン上で稼働する他の Adaptive Server 関連のソフトウェア (Backup Server など) に必要なメモリ量を差し引きます。

4 マシンが Adaptive Server 専用でない場合は、他のシステムで必要なメモリも差し引きます。

たとえば、Adaptive Server のマシン上で稼働するクライアント・アプリケーションが使用するメモリ量を差し引きます。X Window のようなウィンドウ・システムは多くのメモリを必要とするので、Adaptive Server と同じマシン上で使用すると、Adaptive Server のパフォーマンスに影響を与える可能性があります。

オペレーティング・システムと他のアプリケーションに必要なメモリ量を差し引いて残ったメモリが、Adaptive Server で使用できる総メモリ量です。

max memory 設定パラメータを使用して、Adaptive Server に設定できる最大メモリ量を指定します。「メモリの割り付けに影響する設定パラメータ」(45 ページ) を参照してください。

# Adaptive Server がメモリを割り付ける方法

すべてのデータベース・オブジェクトのページのサイズは、マスタ・デバイスの構築時に指定される「論理ページ・サイズ」によって決定します。すべてのデータベース、および各データベース内のすべてのオブジェクトは、同じ論理ページ・サイズを使用します。Adaptive Server の論理ページのサイズ (2K、4K、8K、16K) によって、サーバの領域の割り付けが決まります。各アロケーション・ページ、オブジェクト・アロケーション・マップ (OAM) ページ、データ・ページ、インデックス・ページ、テキスト・ページなどが論理ページ上で構築されます。たとえば、Adaptive Server の論理ページ・サイズが 8K の場合は、これらのページのサイズは 8K となります。論理ページ・サイズで指定されたサイズ全体が、これらのすべてのページによって使用されます。論理ページ・サイズを大きくすると、より大きなローを作成でき、1 ページ分の読み込みによってアクセスできるデータの量が増えるので、パフォーマンスが向上します。たとえば、論理ページ・サイズが 16K ならば 2K ページの 8 倍のデータを保持でき、8K ページの場合は 2K ページの 4 倍のデータを保持できます。

論理ページ・サイズはサーバ全体に適用される設定です。そのため、同じサーバ内で論理ページ・サイズの異なるデータベースを設定することはできません。すべてのテーブルのサイズは、ロー・サイズがサーバの現在のページ・サイズより大きくならない範囲で設定できます。つまり、ローは複数のページにまたがることはできません。

設定されている論理ページ・サイズに関係なく、オブジェクト (テーブル、インデックス、テキスト・ページ・チェーン) の領域はエクステント単位で割り付けられます。1 エクステントは 8 論理ページです。つまり、サーバが 2K の論理ページを使用するように設定されている場合は、これらのオブジェクトに 1 エクステントとして 16K が割り付けられます。16K の論理ページを使用するように設定されている場合は、各オブジェクトに 1 エクステントとして 128K が割り付けられます。

システム・テーブルに対しても、同様の割り付けが行われます。小さいテーブルが多数あるサーバの場合は、使用する論理ページ・サイズが大きいと、領域の消費量が膨大になることがあります。たとえば、2K 論理ページを使用するように設定されているサーバの場合、systypes データベース ( 約 31 の短いローと、クラスタード・インデックスおよびノンクラスタード・インデックスから成る) には 3 エクステント、つまり 48K のメモリが予約されます。サーバをマイグレートして 8K ページを使用する場合、systypes 用に予約される領域は 3 エクステントのままですが、メモリは 192K になります。16K に設定されているサーバの場合、systypes は 384K のディスク領域を必要とします。サイズが小さいテーブルの最後のエクステントで未使用となっている領域が、大きい論理ページ・サイズを使用するサーバでは重要になることがあります。

ページ・サイズを大きくすると、データベースもその影響を受けます。各データベースには、システム・カタログとそのインデックスがあります。小さい論理ページ・サイズから大きい論理ページ・サイズにマイグレートするときは、各データベースに必要なディスク領域の量を計算する必要があります。表 3-1 に、論理ページ・サイズごとのデータベース・サイズの最小値を示します。

**表 3-1: 最小データベース・サイズ**

| 論理ページのサイズ | 最小データベース・サイズ |
|---|---|
| 2K | 2MB |
| 4K | 4MB |
| 8K | 8MB |
| 16K | 16MB |

## ディスク領域の割り付け

論理ページ・サイズは、メモリのアロケーション・ページ・サイズとは異なるものです。論理ページ・サイズが 2、4、8、16K のどのサイズに設定されても、メモリのアロケーション・ページは常に 2K のままです。メモリ関連の設定パラメータのほとんどは、2K を単位としてメモリのページ・サイズを指定します。これに該当する設定パラメータには、次のものがあります。

- max memory
- total logical memory
- total physical memory
- procedure cache size
- size of process object heap
- size of shared class heap
- size of global fixed heap

## 大容量の論理ページ・サイズとバッファ

Adaptive Server は、論理ページ単位でバッファ・プールを割り付けます。たとえば、2K の論理ページを使用するサーバで、デフォルト・データ・キャッシュに 8MB が割り付けられているとします。このとき、バッファの数は約 2048 となります。論理ページ・サイズが 16K のサーバで、デフォルト・データ・キャッシュに同じ 8MB を割り付けると、デフォルト・データ・キャッシュのバッファの数は約 256 になります。処理量が多いシステムでは、バッファの数がこのように少ないことが原因で、バッファが常にウォッシュ・エリアに存在することがあり、クリーンなバッファを要求するタスクのパフォーマンスが低下します。一般に、ページ・サイズを大きくした場合に 2K 論理ページ・サイズのときと同様のバッファ管理特性を実現するには、大きなページ・サイズに合わせてキャッシュのサイズを調整します。したがって、論理ページ・サイズを 4 倍に拡大するときは、キャッシュ・サイズとプール・サイズも約 4 倍に拡大しなければなりません。

通常、Adaptive Server はメモリを動的に割り付け、必要であればロー処理用のメモリを割り付けます。また、サイズの大きなバッファが必要でない場合も、これらのバッファ用に最大サイズを割り付けます。このようなメモリ管理要求によって、Adaptive Server がワイド文字データを処理するときのパフォーマンスの低下を最小限に抑えることができます。

## ヒープ・メモリ

ヒープ・メモリ・プールは起動時に作成される内部的なメモリ・プールで、このプールからタスクが必要に応じて動的にメモリを割り付けます。このメモリ・プールは、スタックからの大量のメモリを必要とするタスク ( ワイド・カラムを使用するタスクなど ) によって使用されます。たとえば、ワイド・カラムやローを変更する場合に、タスクが使用するテンポラリ・バッファのサイズを 16K に拡大することができますが、このサイズは大きすぎるのでスタックからの割り付けはできません。したがって、Adaptive Server は、タスクの実行時にメモリの動的な割り付けと解放を行います。ヒープ・メモリ・プールを利用することにより、各タスク用にあらかじめ定義されるスタック・サイズが大幅に小さくなり、さらにサーバ内のメモリ使用効率が高まるという利点があります。タスクが使用するヒープ・メモリは、タスクが完了するとヒープ・メモリ・プールに返されます。

ヒープ・メモリを設定するには、heap memory per user 設定パラメータを使用します。

ヒープ・メモリの値はユーザあたりのバイト数です。デフォルトでは、このメモリ量は 4096 バイトに設定されます。次の例では、ヒープ・メモリの値をユーザあたり 100 バイトに設定します。

```
sp_configure 'heap memory per user', 100
```

メモリの容量をユーザあたりのバイト数として指定することもできます。たとえば、次の例では各ユーザ接続に 4K バイトのヒープ・メモリを割り付けるよう指定しています (0 は、単位値を指定する場合に sp_configure によって要求されるプレースホルダです)。

```
sp_configure 'heap memory per user', 0, "4K"
```

Adaptive Server の初期設定では、ヒープ・メモリ用に 1MB が確保されます。サーバに設定されているユーザ接続数とワーカ・プロセス数に対応できるように、追加のヒープ・メモリが割り付けられます。したがって、サーバの起動時に使用可能となるヒープ・メモリの量には、次の設定パラメータが影響します。

- number of user connections
- number of worker processes

グローバル変数 *@@heapmemsize* は、ヒープ・メモリ・プールのサイズをバイト単位で表します。

## ヒープ・メモリの計算

Adaptive Server によって確保されるヒープ・メモリの量の計算式は次のとおりです。Adaptive Server は内部構造用に少量のメモリを予約するため、数値はサイトによって異なります。

((1024 X 1024) + (ヒープ・メモリ (バイト))* (number of user connections + number of worker processes))

最初の値 (1024 X 1024) はヒープ・メモリ・プールの初期サイズの 1MB を表します。Adaptive Server は、内部構造用に少量のメモリを予約します。

たとえば、サーバが次のように設定されているとします。

- heap memory per user − 4K
- number of user connections − 25 (デフォルト)
- number of worker processes − 25 (デフォルト)

*@@heapmemsize* の値は 1378304 バイトとなります。

上記の式を使用すると、見積もりは次のようになります。

((1024 X 1024) + (4 X 1024 X 50)) = 1253376

次のように number of user connections を増やすと、それに従ってヒープ・メモリ・プールのサイズも増えます。

```
sp_configure 'user connections', 100
```

*@@heapmemsize* の値は 1716224 バイトとなります。

この場合の見積もりは次のようになります。

((1024 X 1024) + (4 * 1024 * (100 + 25) ) = 1560576

次のエラー・メッセージが表示されて、アプリケーションが異常終了したとします。

```
There is insufficient heap memory to allocate %ld bytes.
Please increase configuration parameter 'heap memory per
user' or try again when there is less activity on the
system.
```

サーバで使用可能なヒープ・メモリを増やすには、次のいずれかを大きくします。

- heap memory per user
- number of user connections
- number of worker processes

メモリ・プールのサイズは、ユーザ接続数によって異なります。Sybase® では、ユーザ 1 人あたりのヒープ・メモリは、論理ページ・サイズの 3 倍以上に設定することをおすすめします。

number of user connections または number of worker processes を増やす前に、heap memory per user の値を増やすことをおすすめします。number of user connections と number of worker processes を増やすと、他のリソース用のシステム・メモリが消費されるので、サーバの最大メモリを大きくしなければならないことがあります。

『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## Adaptive Server がメモリを使用する方法

Adaptive Server でのメモリは、次のように合計論理メモリまたは合計物理メモリとして存在します。

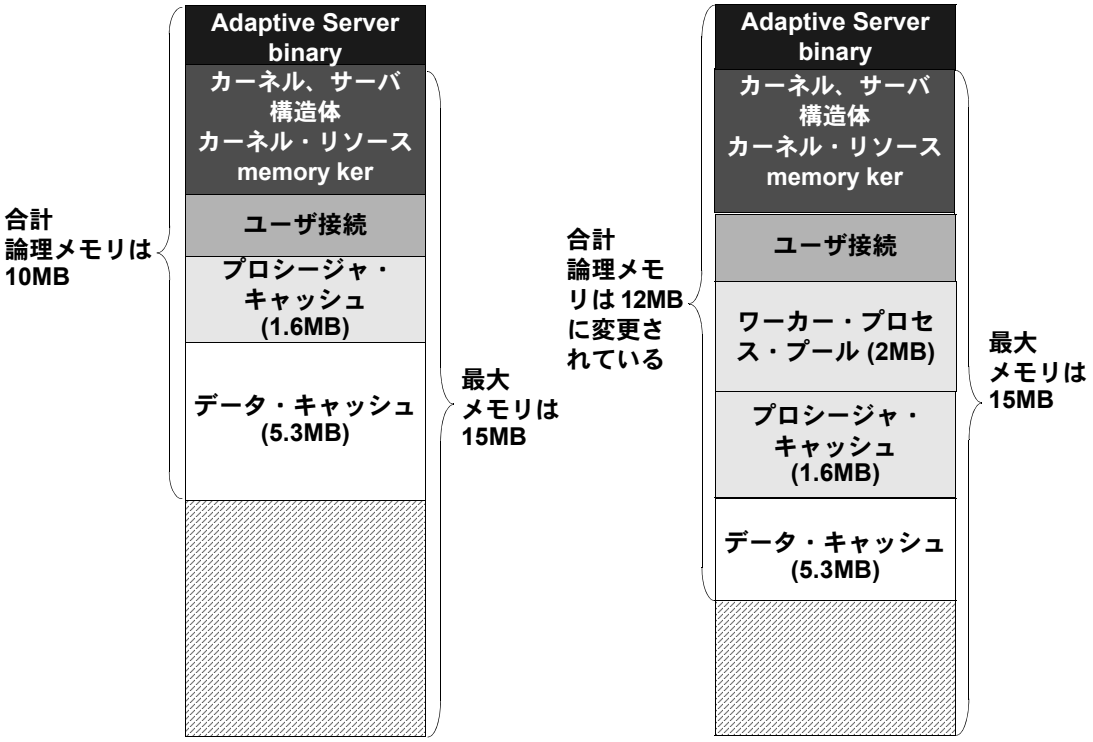
- 合計論理メモリ － すべての sp_configure パラメータに必要なメモリの合計量です。合計論理メモリは使用可能な状態でなければなりませんが、特定の時点においてその量が使用されていることもそうでないこともあります。合計論理メモリの値は、設定パラメータ値の変更に応じて変化することがあります。
- 合計物理メモリ － Adaptive Server 内の、すべての共有メモリ・セグメントの合計です。つまり、合計物理メモリとは、Adaptive Server が特定の時点で使用しているメモリの合計量です。この値を確認するには、読み込み専用の設定パラメータである total physical memory を使用します。total physical memory の値が減少することはありません。これは、いったん割り付けられたメモリ・プールのサイズが縮小することはないからです。合計物理メモリ量を少なくするには、設定パラメータを変更して Adaptive Server を再起動してください。

Adaptive Server は、起動時に次のメモリを割り付けます。

- Adaptive Server が使用するデータ構造体用のメモリ (ユーザによる設定は不可)
- ユーザ設定可能なすべてのパラメータ用のメモリ (データ・キャッシュ、プロシージャ・キャッシュ、カーネル・リソース・メモリ、デフォルト・データ・キャッシュを含む)

図 3-1 は、メモリ設定パラメータが変更されたときに Adaptive Server がどのようにメモリを割り付けるかを示します。

**図 3-1: メモリ構成の変更内容**



2MB のワーカー・プロセス・プールが Adaptive Server のメモリ構成に追加されても、プロシージャ・キャッシュとデータ・キャッシュのサイズは最初に設定されたサイズ (1.6MB と 5.3MB) のままです。max memory は total logical memory よりも 5MB 大きいため、追加されたメモリ・プールを容易に吸収します。新しいワーカー・プロセス・プールの追加によって、サーバのサイズが max memory の制限値を超える場合は、ワーカー・プロセス・プールを大きくするために発行されたコマンドは異常終了します。このような場合は、新しい構成に必要な合計論理メモリ量が、sp_configure のエラー・メッセージ内に表示されます。新しい構成に必要な合計論理メモリ (total logical memory) より大きくなるように max memory の値を設定します。次に、sp_configure をもう一度実行します。

**注意**　max memory と total logical memory の値には、Adaptive Server バイナリは含まれていません。

デフォルト・データ・キャッシュとプロシージャ・キャッシュのサイズは、全体的なパフォーマンスに大きく影響します。プロシージャ・キャッシュ・サイズを最適化する方法については、『パフォーマンス＆チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」を参照してください。

## Adaptive Server が必要とするメモリ量の決定

Adaptive Server の起動に必要な合計メモリ量は、すべてのメモリ設定パラメータの合計 + プロシージャ・キャッシュのサイズ + バッファ・キャッシュのサイズです。このプロシージャ・キャッシュのサイズとバッファ・キャッシュのサイズは、割合ではなく概数です。プロシージャ・キャッシュ・サイズとバッファ・キャッシュ・サイズは、設定されている合計メモリ量には依存しません。プロシージャ・キャッシュ・サイズとバッファ・キャッシュ・サイズは、他のサイズとは無関係に設定できます。各キャッシュの合計サイズ、キャッシュごとのプール数、各プールのサイズなどに関する情報を取得するには、sp_cacheconfig を使用します。

特定の時点で Adaptive Server が使用している合計メモリ量を調べるには、sp_configure を使用します。

```
1> sp_configure "total logical memory"

Parameter Name        Default   Memory Used   Config Value   Run Value
   Unit               Type
--------------        --------------- --- ----------- ------------ ---------
-------------------- ----------
total logical memory  33792    127550         63775         63775
   memory pages(2k)   read-only
```

“Memory Used” カラムの値は、キロバイト単位です。一方、“Config Value” カラムの値は、2K ページの数です。

Config Value カラムの値は、Adaptive Server の稼働中に使用される合計論理メモリ量を示しています。Run Value カラムの値は、現在の Adaptive Server 構成が消費している合計論理メモリ量を示しています。2 つの Adaptive Server がまったく同じように設定されていることはないため、このコマンドを実際に実行した場合、出力内容はこれとは異なります。

『リファレンス・マニュアル：プロシージャ』を参照してください。

## Adaptive Server メモリ設定の決定

システム起動時に割り付けられるメモリの合計は、Adaptive Server のすべての設定に必要なメモリの合計です。この値は、読み込み専用の設定パラメータ total logical memory から取得され、Adaptive Server によって計算されます。設定パラメータ max memory には、total logical memory 以上の値を指定する必要があります。max memory は、Adaptive Server の使用に対応できるメモリの量を示しています。

デフォルトでは、total logical memory の値に基づいてサーバの起動時にメモリが割り付けられます。ただし、設定パラメータ allocate max shared memory が設定されている場合は、max memory の値に基づいてメモリが割り付けられます。設定パラメータ allocate max shared memory を使用すると、システム管理者は Adaptive Server で使用できる最大メモリをサーバの起動時に割り付けることができます。

メモリ設定の重要な点は次のとおりです。

- システム管理者は、Adaptive Server に使用できる共有メモリのサイズを決定し、max memory をこの値に設定する必要があります。
- 設定パラメータ allocate max shared memory を起動時と実行時にオンにすると、最小限の共有メモリ・セグメントを使用してすべての共有メモリを max memory まで割り付けることができます。多数の共有メモリ・セグメントを使用すると、特定のプラットフォームでパフォーマンスが低下するという欠点があります。共有メモリ・セグメントの最適な数については、オペレーティング・システムのマニュアルを参照してください。割り付けられた共有メモリ・セグメントは、サーバが再起動されるまで解放できません。
- max memory と total logical memory の差分は、プロシージャ・キャッシュ、ステートメント・キャッシュ、データ・キャッシュ、または他の設定パラメータに使用できるメモリ量を決定します。

  起動時に Adaptive Server によって割り付けられるメモリの量は、total logical memory または max memory によって決まります。alloc max shared memory を 1 に設定する場合、Adaptive Server は max memory の値を使用します。

total logical memory または max memory のいずれかが大きすぎる場合

- マシンの物理リソースが不十分な場合は、Adaptive Server が起動しないことがあります。
- Adaptive Server が起動しても、オペレーティング・システム・ページのフォールト・レートが著しく上昇し、オペレーティング・システムを再設定する必要が生じることがあります。

## アップグレードを実行する場合

12.5 より前のバージョンの Adaptive Server では、total logical memory、procedure cache percent、min online engines の各設定値から、procedure cache size と number of engines at startup の新しい値が自動的に計算されます。デフォルト・データ・キャッシュのサイズは、アップグレード時に計算されて設定ファイルに書き込まれます。計算されたデータ・キャッシュ・サイズとプロシージャ・キャッシュ・サイズがデフォルト・サイズよりも小さい場合は、デフォルト・サイズにリセットされます。

アップグレード中に、Adaptive Server は次を設定します。

- max memory は設定ファイルで指定された total logical memory の値に設定されます。max memory の値は、リソース要件に応じて再設定してください。
- 以前の設定でのエンジン数から syb_default_pool のスレッド数

sp_configure の verify オプションを使用すると、Adaptive Server を再起動しなくても、設定ファイルに加えた変更内容を確認できます。

```
sp_configure “configuration file”, 0, “verify”, “full_path_to_file”
```

# Adaptive Server で使用可能なメモリ量の決定

表 3-2 に、Adaptive Server バージョン 12.0 以降でアドレス指定可能な共有メモリの上限を示します。

**表 3-2: アドレス指定可能なプラットフォーム別の最大メモリ量**

| プラットフォーム | 32 ビットの Adaptive Server | 64 ビットの Adaptive Server |
|---|---|---|
| HP-UX 11.x (PA-RISC プロセッサ) | 2.75 ギガバイト | 16 EB[1] |
| IBM AIX 5.x | 2.75 ギガバイト | 16 EB |
| Sun Solaris 8 (SPARC プロセッサ) | 3.78 ギガバイト | 16 EB |
| Sun Solaris 8 (Intel x86 プロセッサ) | 3.75 ギガバイト | N/A |
| Red Hat Enterprise Linux (Intel x86 プロセッサ) | 2.7 ギガバイト | N/A |

[1]1 エクサバイト (EB) は $2^{60}$ 乗、または 1024 ペタバイトと同等です。理論上の上限は 16 エクサバイトです。実際の上限はシステム上の合計メモリ量になります。Adaptive Server は最大 256GB の共有メモリで動作確認されています。
[2]Windows を /3G オプションを指定して起動すると、Adaptive Server は最大 3GB の共有メモリを使用できます。詳細については、Windows のマニュアルを参照してください。

**注意** Adaptive Server 12.5 以降では、以前のバージョンとは異なる方法でメモリが割り付けられています。既存のメモリ関連の設定パラメータが変更され、新しいメモリ・パラメータが導入されています。

オペレーティング・システムごとに、共有メモリのデフォルトの最大セグメントが決まっています。オペレーティング・システムの共有メモリ・セグメントの割り付けは、max memory を上回るように設定してください。詳細については、使用しているプラットフォームの『インストール・ガイド』を参照してください。

# メモリの割り付けに影響する設定パラメータ

Adaptive Server のメモリ構成を設定するときは、sp_configure を使用して、各メモリ要件を絶対値によって指定します。また、プロシージャ・キャッシュとデフォルト・データ・キャッシュのサイズも、絶対値で指定します。

メモリの割り付けに影響するパラメータには、max memory、allocate shared memory、および dynamic allocation on demand の 3 つがあります。

*max memory*

max memory を使用すると、Adaptive Server に割り付けることができるメモリ量の最大値を設定できます。max memory によって最大メモリを設定するときに、必要な値よりも少し大きな値に設定すると、Adaptive Server に必要なメモリ量が増えたときに利用可能な予備のメモリを確保できます。

max memory の値を増やすには、sp_configure で max memory の値を指定します。ただし、メモリの割り付けはすぐに行われない場合があります。max memory によって指定されたメモリを Adaptive Server が割り付ける方法は、allocate max shared memory と dynamic allocation on demand の設定によって異なります。

アップグレード時に max memory の値が不足している場合、max memory の値は Adaptive Server によって自動的に引き上げられます。新しいバージョンの Adaptive Server では、内部データ構造のサイズが増加しているため、より多くのメモリが必要になります。

*allocate max shared memory*

allocate max shared memory を使用すると、起動時に、max memory によって指定されたメモリをすべて割り付けるか、合計論理メモリの指定によって要求されるメモリだけを割り付けるかを実行できます。

プラットフォームによっては、アプリケーションに割り付けられた共有メモリ・セグメント数がプラットフォーム固有の最適数を超えると、パフォーマンスが低下することがあります。このような場合は、max memory を、Adaptive Server が使用できる最大量に設定してください。次に、allocate max shared memory を 1 に設定して、サーバを再起動します。これによって、max memory で指定されたすべてのメモリが起動時に割り付けられ、セグメント数は最小になります。

たとえば、allocate max shared memory が 0 (デフォルト値) に設定され、max memory が 500MB に設定されているけれども、起動時にこのサーバ構成が必要とするメモリが 100MB だけである場合は、残りの 400MB は追加のメモリが必要になるまでは割り付けられません。ただし、allocate max shared memory が 1 に設定されている場合は、Adaptive Server は起動時に 500MB 全体を割り付けます。

allocate max shared memory を 0 に設定し、max memory を増やした場合は、実際のメモリ割り付けは必要に応じて行われます。allocate max shared memory を 1 に設定し、max memory を増やした場合は、Adaptive Server はメモリをすぐに割り付けようとします。メモリの割り付けに失敗した場合、メッセージがエラー・ログに書き込まれます。

起動時にすべてのメモリを割り付ける利点は、追加メモリに応じたサーバの再調整が行われる間もパフォーマンスの低下が発生しないということです。ただし、メモリ増加の予測が適切でなく、max memory が大きな値に設定されているときは、物理メモリが無駄に消費されてしまうことになります。メモリ設定パラメータの値を動的に小さくすることはできないため、他のメモリ要件も考慮しておくことが重要です。

*dynamic allocation on demand*

dynamic allocation on demand を使用すると、メモリ・リソースの割り付けを、メモリ・リソースが要求されたときにすぐに実行するか、または必要になった場合にだけ実行するかを指定できます。dynamic allocation on demand を 1 に設定すると、メモリの変更値は必要となったときに割り付けられます。0 に設定すると、メモリ設定量は、メモリ構成を変更したときに割り付けられます。

たとえば、dynamic allocation on demand を 1 に設定して、number of user connections を 1024 に変更すると、total logical memory は、ユーザ 1 人あたりのメモリ量に 1024 を掛けた値になります。つまり、ユーザ 1 人あたりのメモリ量が 112K の場合、ユーザ接続に使用されるメモリは 112MB (1024 × 112) になります。

このメモリ量は、number of user connections 設定パラメータが使用できる最大メモリ量です。ただし、サーバに接続しているユーザ数が 500 だけの場合、number of user connections パラメータが使用する合計物理メモリ量は 56MB (500 × 112) です。

dynamic allocation on demand の値が 0 の場合は、number of user connections を 1024 に変更すると、すべてのユーザ接続リソースが直ちに設定されます。

最高のパフォーマンスを得るには、total physical memory が合計論理メモリ量を超えないように Adaptive Server のメモリを構成してください。合計論理メモリ量は、max memory 未満です。dynamic allocation on demand を 1 に設定し、allocate max shared memory を 0 に設定すると、このことを部分的に実現できます。

# メモリの動的な割り付け

Adaptive Server は物理メモリの割り付けを動的に実行します。したがって、Adaptive Server のメモリ構成の設定を変更しても、サーバを再起動する必要はありません。

---

**注意**　Adaptive Server はメモリを動的に小さくすることは行いません。メモリ設定パラメータの値を小さくして、それまで使用していた物理メモリを解放するには、サーバを再起動しなければなりません。したがって、システムに必要なメモリ量を正確に査定することが重要です。「メモリ設定パラメータの動的な低減」(47 ページ) を参照してください。

---

次のような場合は、max_memory 設定パラメータの値の変更を検討してください。

- 使用しているマシン上の RAM の量を変更する。
- 使用しているマシンの使用のパターンを変更する。
- max_memory (最大メモリ) が不足しているため、設定に失敗する。

## Adaptive Server が起動しない場合

Adaptive Server は、起動時に、total logical memory によって設定されている全メモリ量をオペレーティング・システムから取得します。十分なメモリを取得できないために Adaptive Server が起動しないときは、メモリを消費する設定パラメータの値を小さくして、必要なメモリ量を小さくします。また、場合によっては、多くのメモリ量を必要とする他の設定パラメータの値も小さくする必要があります。変更後の新しい値を使用するために、Adaptive Server を再起動します。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## メモリ設定パラメータの動的な低減

メモリ設定パラメータの値を再設定して小さくしても、使用中のメモリは動的に解放されません。メモリ構成の変更値をどのように減少させるかについて、図 3-2 と図 3-3 で説明します。

**図 3-2: dynamic allocation on demand を 1 に設定した場合 (新しいユーザ接続なし)**

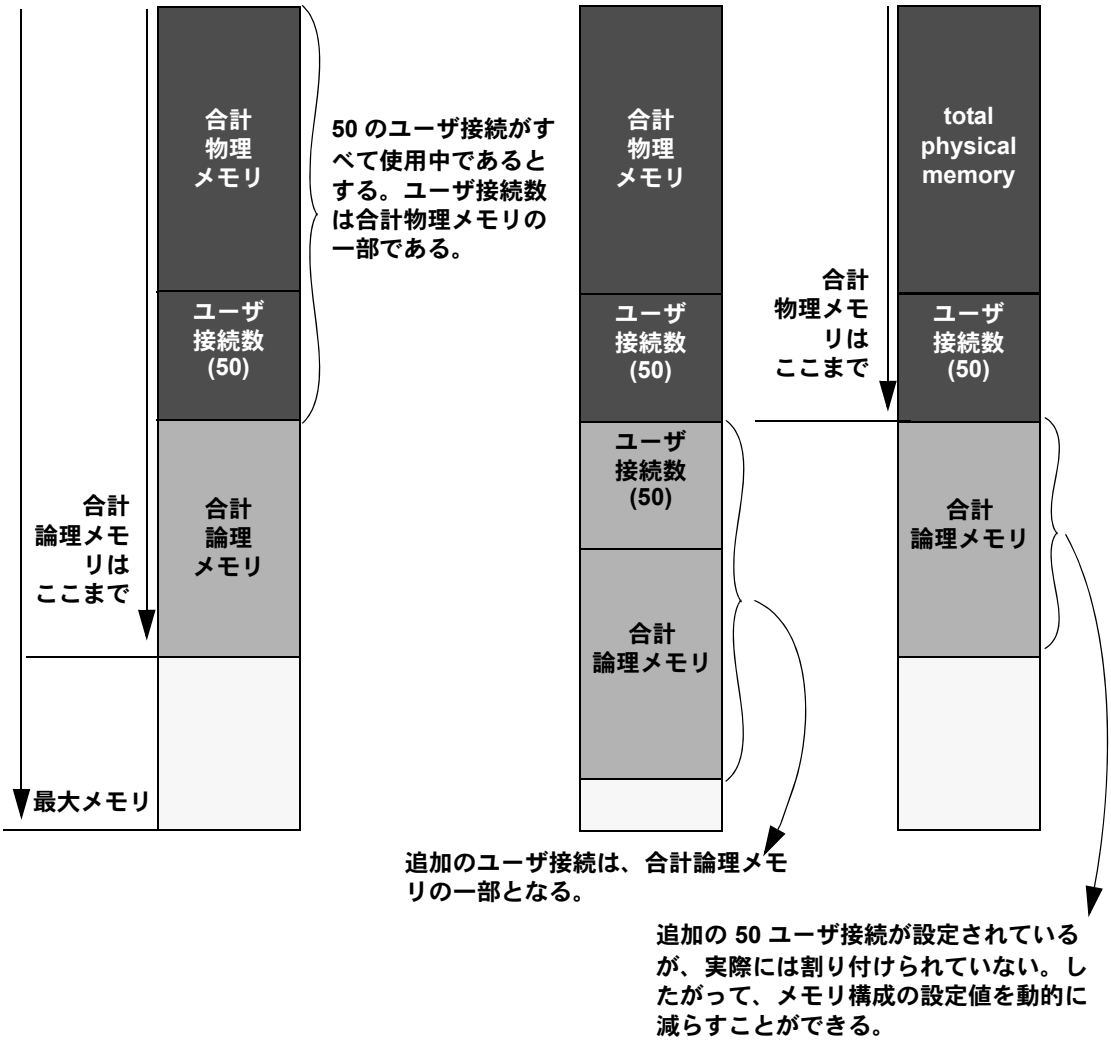



図 3-2 では、dynamic allocation on demand が 1 に設定されているため、メモリが使用されるのは、追加メモリを使用しなければならないイベントが発生したときだけとなっています。この例では、そのイベントとはユーザ接続の追加要求で、これは、クライアントが Adaptive Server にログインしようとしたときに発生します。

number of user connections の値は、実際に割り付けられているユーザ接続数まで小さくすることができます。これは、dynamic allocation on demand が 1 に設定されており、要求されるユーザ接続数が実際に増加しなければサーバから追加メモリが要求されることはないからです。

**図 3-3: dynamic allocation on demand を 1 に設定した場合 (新しいユーザ接続がログオン済み)**

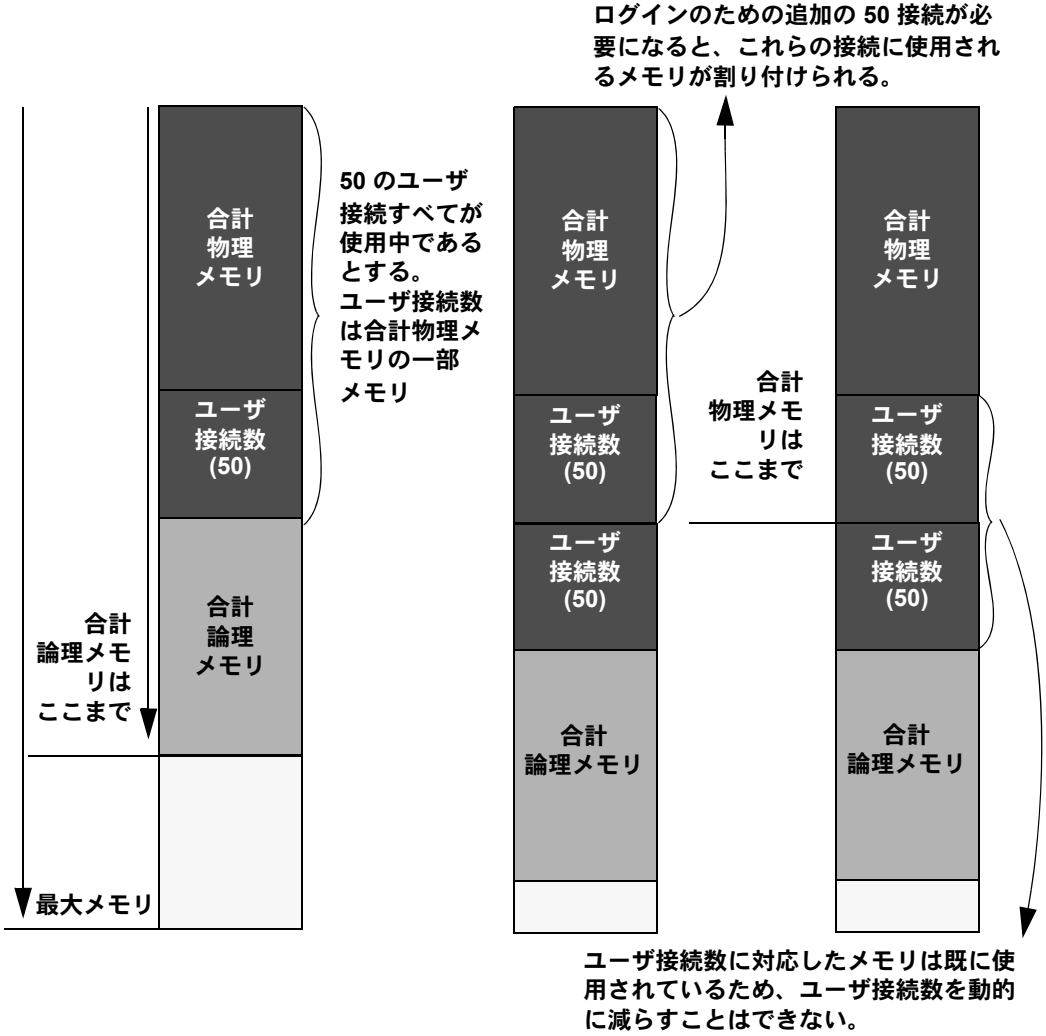
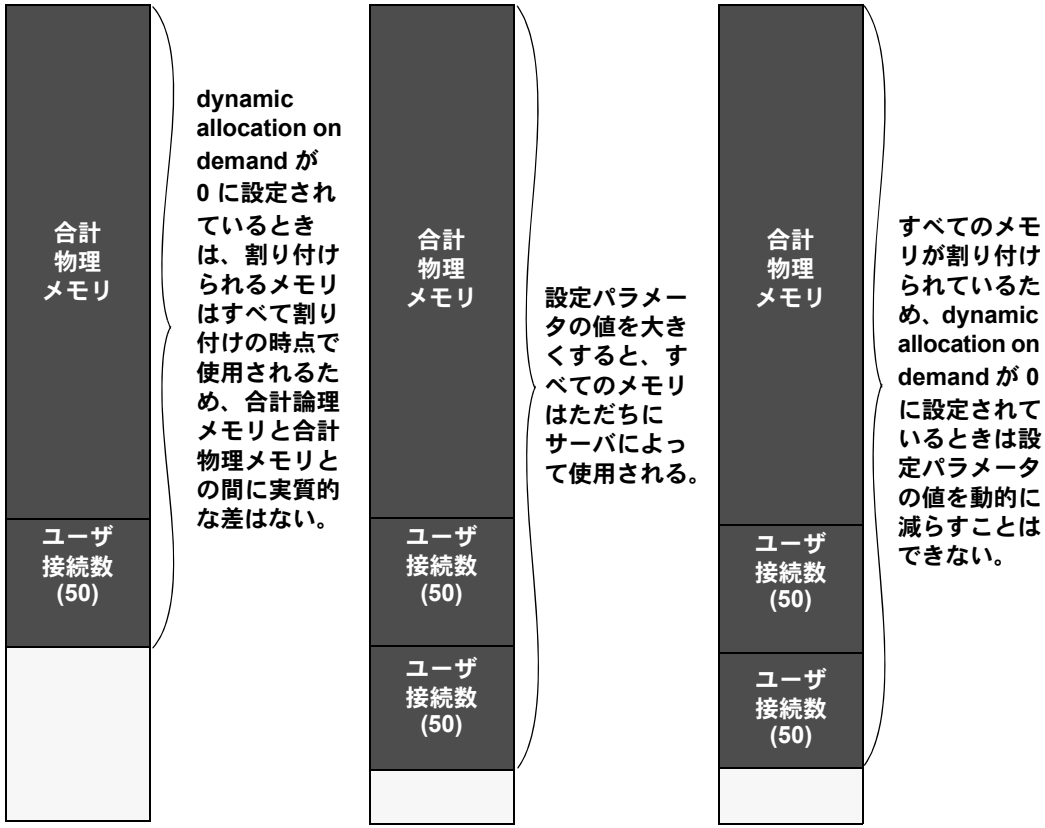


図 3-3 では、追加の 50 ユーザ接続がすべて実際に使用されていると想定しています。メモリが使用中であるので、number of user connections を減らすことはできません。sp_configure を使用してメモリ設定パラメータの変更を指定することはできますが、その変更内容はサーバを再起動するまでは有効になりません。

**図 3-4: dynamic allocation on demand が 0 に設定されている場合**




**注意** 理論的には、dynamic allocation on demand が 0 に設定されている場合は、合計論理メモリと合計物理メモリとの間に差はありません。ただし、Adaptive Server が必要なメモリ量を見積もる方法と、実際にメモリが要求される方法が多少異なることがあります。したがって、実行時にこの 2 つのメモリ量が異なる可能性もあります。

dynamic allocation on demand を 0 に設定すると、設定されているすべてのメモリ要件がただちに割り付けられます。メモリ構成を動的に低減させることはできません。

図 3-3 と図 3-4 では、メモリ設定パラメータの値をさらに小さくすることもできます。この変更は動的に反映されるものではありませんが、このように変更すると、新しいメモリの使用ができなくなります。たとえば、図 3-3 や図 3-4 に示す状況例で説明します。ここで、100 のユーザ接続に対応できるように number of user connections を設定した後に、ユーザ接続数を 50 に変更したとします。この場合、number of user connections の値を 50 に減らすことができます。Adaptive Server のメモリ使用量がこの変更の影響を受けるのはサーバの起動後ですが、このように変更した後は、新しいユーザがサーバにログインできなくなります。

# スレッド・プールの設定

Adaptive Server は、個別のスレッド・プール情報を `Thead Pool` 見出しの下にある設定ファイルに記録します。

デフォルトでは、Adaptive Server には、機能するために必要なシステムのスレッド・プールのセットがあります。これには、syb_default_pool エンジン・プール、エンジンが含まれていない複数の run to completion (RTC) スレッド・プールが含まれます。ユーザは、システムのプールに加えて、独自のスレッド・プールを作成できます。ユーザ作成のスレッド・プールは常にエンジン・プールです。

スレッド・プール型の詳細については、「スレッド・プール」(124 ページ)を参照してください。

起動時に、Adaptive Server は、デフォルト値に設定されたパラメータで syb_default_pool および syb_blocking_pool の情報をリストします。syb_system_pool の情報は含まれません。Adpative Server は、他の設定要求に基づいて実行時にスレッド数を計算するためです。

設定ファイルは、すべてのスレッド・プールのパラメータをリストします。

- number of threads − プール内の設定スレッド数。
- description − プールの簡単な説明。

Adaptive Server が起動されると、これがデフォルトのスレッド・プール設定になります。

```
[Thread Pool:syb_blocking_pool]
        number of threads = 4

[Thread Pool:syb_default_pool]
        number of threads = 1
```

スレッド・プールを追加または削除すると、Adaptive Server は設定ファイルを更新しますが、変更の有効化するために再起動する必要はありません。ユーザ作成のスレッド・プール (create thread pool で作成) は、エンジン・プールである必要があります。

この例には、Sybase が提供する 2 つのスレッド・プールおよびユーザ作成のスレッド・プール、sales_pool が含まれます。

```
[Thread Pool:sales_pool]
        description = pool for the sales force
        number of threads = 14
        idle timeout = 75

[Thread Pool:syb_blocking_pool]
         number of threads = 20

[Thread Pool:syb_default_pool]
        number of threads = 1
```

ファイル・エディタを使用して、設定ファイルでスレッド・プール情報を編集、または create thread pool コマンド、alter thread pool コマンド、および drop thread pool コマンドを使用して、スレッド・プールを管理します。『リファレンス・マニュアル：コマンド』を参照してください。

設定ファイルを編集する場合、Adaptive Server は新しいスレッド・プール設定を使用し、ログ・ファイルに変更情報を出力します (create thread pool を使用してスレッド・プールを追加する場合、Adaptive Server を再起動する必要はありません )。次は、smaller_pool スレッド・プールを Adaptive Server に追加した後、ログ・ファイルから出力されます。

```
00:0000:00000:00000:2010/06/03 16:09:56.22 kernel  Create Thread Pool 4, "smaller_pool",
type="Engine (Multiplexed)", with 10 threads
```

isql では、create thread pool を使用して、スレッド・プールを追加します。この例では、5 つのスレッドで sales_pool スレッド・プールを追加します。

```
create thread pool sales_pool with thread count = 1
```

sp_helpthread を使用して、syb_system_pool を含むスレッド・プールの実行時の値を決定します。これは、上記のスレッド・プールの sp_helpthread 出力です。

```
sp_helpthread


Name                Type                    Size   IdleTimeout
          Description
-----------------   --------------------  ----   -----------
          -------------------------------------------
sales_pool          Engine (Multiplexed)     1           100
          NULL
syb_blocking_pool   Run To Completion        4             0
          A pool dedicated to executing blocking calls
syb_default_pool    Engine (Multiplexed)     1           100
          The default pool to run query sessions
syb_system_pool     Run To Completion        4             0
          The I/O and system task pool
```

sales_pool スレッド・プールを削除するには、次を使用します。

```
drop thread pool sales_pool
```

『リファレンス・マニュアル：コマンド』を参照してください。

スレッドの作成には不十分なメモリ量 (kernel resource memory で決定 ) でスレッド・プールを作成し、使用可能なエンジン数が不十分な場合、このようなメッセージが表示される場合があります。

```
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Setting console to nonblocking mode.
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Create thread pool pubs_pool
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Create Thread Pool 4, "pubs_pool",
type="THREADPOOL_MULTIPLEXED", with 2 threads
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  could not allocate memory for dynamic
engine
00:0001:00000:00011:2010/06/11 14:46:38.32 kernel  Thread creation failed to allocate an
engine.
00:0001:00000:00011:2010/06/11 14:46:38.32 server  Configuration file
'/sybase/siena.cfg' has been written and the previous version has been renamed to
'/sybase/siena.009'.
1> 00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Network and device connection limit
is 1009.
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  ASE - Dynamic Pluggable Component
Interface is disabled
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Encryption provider initialization
succeeded on engine 1.
00:0025:00000:00000:2010/06/11 14:46:38.33 kernel  Thread 25 (LWP 15726) of Threadpool
pubs_pool online as engine 1
```

スレッド・プールを作成するには、max online engines の値を増やして、Adaptive Server を再起動します。

## スレッドの総数の決定

monThread モニタリング・テーブルには、Adaptive Server のすべてのスレッドに関する情報が含まれます。`select count(*) from monThread` を発行して、Adaptive Server のスレッドの総数を決定します。ただし、このクエリからレポートされたスレッドの多くは syb_blocking_pool からで、CPU 容量をほとんど必要とせず、リソースの使用計画の考慮も必要ありません。

リソースの使用計画中に、ユーザ・クエリを実行するクエリ処理、エンジン、またはスレッドの数に特に注意してください。`select count(*) from monEngine`、`select count(*) from sysengines` を発行、または syb_default_pool のスレッド数を任意のユーザ作成のスレッド・プールのスレッド数に追加することにより、クエリ処理スレッド数を決定します。

sp_sysmon は、各スレッドが消費する CPU リソースの量を表示します。『パフォーマンス&チューニング・シリーズ：sp_sysmon による Adaptive Server の監視』を参照してください。

### syb_blocking_pool のチューニング

monWorkQueue モニタリング・テーブルを使用して、syb_blocking_pool の適切なサイズを決定します。キューイングされた要求が過分に含まれている場合や、待機時間がかなり長い場合は、syb_blocking_pool のサイズを増やしてください。WaitCount および WaitTime がゼロの場合は、プールのサイズを減らしてください。

sp_sysmon は、出力の「カーネル」セクションにブロック・プールの統計を表示します。「呼び出しアクティビティのブロック」セクションの「キューイングされた要求」が「サービスされた要求」と比較して高率であるか、「合計待ち時間」が長い場合は、プール・サイズを増やす必要があります。

## メモリの設定に使用するシステム・プロシージャ

Adaptive Server のメモリを設定するには、次の 3 つのプロシージャを使用します。

- sp_configure
- sp_helpconfig
- sp_monitorconfig

sp_configure の完全な構文と使用方法、および各パラメータの詳細については、『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

### *sp_configure* による設定パラメータの設定

Adaptive Server のメモリ使用に関するパラメータを確認するには、次のように入力します。

```
sp_configure "Memory Use"
```

Memory Used カラムに “#” が表示されている場合は、このパラメータが他のパラメータのコンポーネントであり、そのメモリ使用量が他のコンポーネントのメモリ使用量に含まれていることを示します。たとえば、stack size と stack guard size に使用されているメモリは、ユーザ接続やワーカー・プロセスのメモリ要件の一部です。したがって、この値が 200 より大きな値に設定されている場合は、number of user connections や number of worker processes に必要なメモリに含まれます。

Memory Use の出力の値のいくつかは、計算された値です。計算された値は、sp_configure を使用して設定できるものではありませんが、どこにメモリが割り付けられているかを示すためにレポートされます。計算される値の代表的なものに、total data cache size (合計データ・キャッシュ・サイズ) があります。

## 動的増加可能なメモリ量

sp_configure memory を発行すると、すべてのメモリ・パラメータが表示され、max memory と total logical memory の差がわかります。この差が、動的増加可能なメモリ量となります。次に例を示します。

```
sp_configure memory
Msg 17411, Level 16, State 1:
Procedure 'sp_configure', Line 187:
Configuration option is not unique.
Parameter Name                 Default  Memory Used  Config Value  Run Value
Unit                      Type
------------------------------ ----------- ----------- ------------ ----------
--------------------     ----------
additional network memory                0           0            0          0
                bytes        dynamic
allocate max shared memory               0           0            0          0
               switch        dynamic
compression memory size                  0         152            0          0
     memory pages(2k)        dynamic
engine memory log size                   0           2            0          0
     memory pages(2k)        dynamic
heap memory per user                  4096           0         4096       4096
                bytes        dynamic
kernel resource memory                4096        8344         4096       4096
     memory pages(2k)        dynamic
lock shared memory                       0           0            0          0
               switch         static
max memory                           33792      300000       150000     150000
     memory pages(2k)        dynamic
memory alignment boundary            16384           0        16384      16384
                bytes         static
memory per worker process             1024           4         1024       1024
                bytes        dynamic
messaging memory                       400           0          400        400
     memory pages(2k)        dynamic
pci memory size                      32768           0        32768      32768
     memory pages(2k)        dynamic
shared memory starting address           0           0            0          0
       not applicable         static
total logical memory                 33792      110994        55497      55497
     memory pages(2k)      read-only
total physical memory                    0       97656            0      48828
     memory pages(2k)      read-only
transfer utility memory size          4096        8194         4096       4096
     memory pages(2k)        dynamic

An additional 30786 K bytes of memory is available for reconfiguration. This is
the difference between 'max memory' and 'total logical memory'.
```

## *sp_helpconfig* の使用

sp_helpconfig は、与えられた設定パラメータとその値に必要なメモリ量を見積もります。また、パラメータの簡単な説明、パラメータの最小値、最大値、デフォルト値に関する情報や、実行値、現在の実行値での使用メモリ量を表示します。sp_helpconfig が特に役立つのは、サーバに対して重要な変更を加えようとしており (たとえば、サイズの大きい既存のデータベースを他のサーバからロードする)、必要なメモリを見積もる場合です。

パラメータの設定に必要なメモリ量を調べるには、名前を特定するのに十分な長さのパラメータ名と、設定値を入力します。

```
1> sp_helpconfig "worker processes", "50"

number of worker processes is the maximum number of worker processes that can be
in use Server-wide at any one time.

Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
0              2147483647     0              0              0
number   dynamic

Configuration parameter, 'number of worker processes', will consume 7091K of memory if
configured at 50.
Changing the value of 'number of worker processes' to '50' increases the amount of memory
ASE uses by 7178 K.
```

また、特定のリソースに割り付けるメモリ量がわかっているときは、sp_helpconfig を実行することによって、sp_configure の値を求めることもできます。

```
1> sp_helpconfig "user connections", "5M"

number of user connections sets the maximum number of user connections that can
be connected to SQL Server at one time.
Minimum Value  Maximum Value  Default Value  Current Value  Memory Used
Unit     Type
-------------  -------------  -------------  -------------  ----------
-------  ------------
5              2147483647     25             25             3773
number  dynamic
Configuration parameter, 'number of user connections', can be configured to 33
to fit in 5M of memory.
```

上記の 2 つの文の構文の大きな違いは、単位の使い方です。2 番目の例では (メガバイトの “M”)、この単位によって、値が設定値ではなくサイズであることを sp_helpconfig に示しています。以下に、有効な単位を示します。

- P − ページ (Adaptive Server 2K ページ)
- K − キロバイト

- M − メガバイト
- G − ギガバイト

この構文がパラメータの型に対して意味をなさない場合や、メモリ使用量を計算できない場合がありますが、このようなときは、sp_helpconfig を実行するとエラー・メッセージが出力されます。たとえば、allow resource limits のようなオン／オフ形式のパラメータのサイズを指定しようとすると、sp_helpconfig のメッセージには、メモリを使用しない設定パラメータの機能が表示されます。

## *sp_monitorconfig* の使用

sp_monitorconfig は、特定の共有サーバ・リソース上でのメタデータ・キャッシュの使用状況情報を表示します。この情報には、次のような統計値が含まれます。

- 同時にオープンできるデータベース、オブジェクト、インデックスの数
- 参照整合性クエリが使用する補助スキャン記述子の数
- 未使用の記述子とアクティブな記述子の数
- 空きメモリ・ページ数
- アクティブな記述子の割合
- サーバが起動されてから使用された記述子の最大値
- プロシージャ・キャッシュの現在のサイズと、実際に使用された量
- クラスタ環境で実行している場合は sp_monitorconfig を実行するインスタンスの名前、クラスタ環境で実行していない場合は NULL です。

たとえば、number of open indexes 設定パラメータが 500 に設定されている場合、ピーク時に sp_monitorconfig を実行すると、インデックス記述子に対する実際のメタデータ・キャッシュ使用状況を正確に測定したデータが表示されます。

```
sp_monitorconfig "number of open indexes"
Usage information at date and time: May 28 2010  1:12PM.
Name                        Num_free    Num_active  Pct_act  Max_Used
Reuse_cnt    Instance_Name
--------------------------  ----------  ----------  -------  ----------
-----------  ------------------------------
number of open indexes           217         283    56.60         300
           0                               NULL
```

Adaptive Server のオープン・インデックス数が 500 に設定されているにもかかわらず、サーバが前回起動されてから使用されたオープン・インデックスの最大値が 300 と報告されています。したがって、number of open indexes 設定パラメータを 330 に再設定できます。これは、最大使用インデックス記述子数 300 にさらに 10% を加えた数です。

また、sp_monitorconfig "procedure cache size" を使用して、プロシージャ・キャッシュの現在のサイズを調べることもできます。このパラメータは、現在設定されているプロシージャ・キャッシュの領域の大きさと、これまでにプロシージャ・キャッシュが実際に使用した最大量を表します。この例では、プロシージャ・キャッシュが 20,000 ページに設定されています。

```
sp_configure "procedure cache size"
Parameter Name                      DefaultMemory         Used
         Config Value          Run Value            Unit
         Type
------------------------------      -------------------   ----------
         -------------------   --------------------- ---------------------
          --------------------
 procedure cache size                                7000        43914
                       20000                 20000      memory pages(2k)
                       dynamic
```

しかし、sp_montorconfig procedure cache size を実行すると、次のように、プロシージャ・キャッシュがこれまで使用した最大ページ数は 14241 ページであることが判明しました。したがって、プロシージャ・キャッシュの実行値を現在より低く設定して、メモリを節約できます。

```
sp_monitorconfig "procedure cache size"
Usage information at date and time: May 28 2010  1:35PM.
Name                      Num_free    Num_active  Pct_act  Max_Used
Reuse_cnt    Instance_Name
------------------------  ----------  ----------  -------  ----------
----------  ------------------------------
procedure cache size           5878        14122    70.61       14241
         384                              NULL
```

空きメモリ・ページ数を表示して (Num_free)、カーネル・リソース・メモリ設定が十分であるかを決定します。

```
sp_monitorconfig "kernel resource memory"
Usage information at date and time: Oct 12 2010  1:35PM.
Name                      Num_free    Num_active  Pct_act  Max_Used
Reuse_cnt    Instance_Name
------------------------  ----------  ----------  -------  ----------
----------  ------------------------------
kernel resource memory          9512         728     7.11       728
         0                              NULL
```

空きページ数が低くて、アクティブな割合 (Pct_act) が高い場合、kernel resource memory の値を増やす必要があります。

# Adaptive Server メモリを制御する設定パラメータ

大量の Adaptive Server メモリを使用する設定パラメータ、および多くの Adaptive Server 環境で一般に変更される設定パラメータは、システム管理者が Adaptive Server を初めて設定するときに確認する必要があります。システム構成を変更するときや、Adaptive Server を新しいバージョンにアップグレードした後、またはメモリを使用する他の設定変数を変更するときには、これらのパラメータを見直してください。

設定パラメータのうち、メモリ使用量が少ないものについては、「メモリを使用するその他のパラメータ」(64 ページ) で説明しています。

## Adaptive Server 実行プログラム・コードのサイズ

実行プログラム・コードのサイズは、total logical memory や max memory の計算には含まれていません。total logical memory は、実行プログラムを除く Adaptive Server 設定に必要な実際のメモリ量を表します。

実行プログラム・コードに必要なメモリはオペレーティング・システムによって管理され、Adaptive Server で制御することはできません。使用するオペレーティング・システム用のマニュアルを参照してください。

## データ・キャッシュとプロシージャ・キャッシュ

「Adaptive Server がメモリを使用する方法」(39 ページ ) の説明に従って、データ・キャッシュとプロシージャ・キャッシュのサイズを指定します。この項では、この 2 つのキャッシュの詳細と、キャッシュ・サイズをモニタリングする方法について説明します。

### プロシージャ・キャッシュ・サイズの決定

procedure cache size は、プロシージャ・キャッシュのサイズを、サーバの論理ページ・サイズに関係なく、2K のページ単位で指定します。次に例を示します。

```
sp_configure "procedure cache size"
Parameter Name        Default  Memory Used  Config Value  Run Value
Unit              Type
------------------  -------  -----------  ------------  ---------
---------------  ---------
procedure cache size     7000        15254          7000       7000
memory pages(2k)  dynamic
```

プロシージャ・キャッシュに使用されるメモリ量は 30.508KB (15254 2K ページ) です。プロシージャ・キャッシュを別のサイズに設定するには、次のコマンドを発行します。

```
sp_configure "procedure cache size", new_size
```

次のコマンド例は、プロシージャ・キャッシュのサイズを 10000 × 2K ページ (20MB) に再設定します。

```
sp_configure "procedure cache size", 10000
```

## デフォルト・データ・キャッシュ・サイズの決定

sp_cacheconfig と sp_helpcache はどちらも、現在のデフォルト・データ・キャッシュのサイズをメガバイト単位で表示します。たとえば、次の例では、Adaptive Server のデフォルト・データ・キャッシュが 19.86MB に設定されています。

```
sp_cacheconfig

Cache Name               Status     Type       Config Value    Run Value
-----------------        --------   --------   ------------    ----------
default data cache       Active     Default    0.00 Mb         19.86Mb
                                                ------------    --------
                      Total                     0.00Mb          19.86 Mb
============================================================================
Cache: default data cache, Status: Active, Type: Default
       Config Size: 0.00 Mb, Run Size: 19.86 Mb
       Config Replacement: strict LRU, Run Replacement: strict LRU
       Config Partition:    1,   Run Partition:    1
IO Size       Wash Size     Config Size     Run Size      APF Percent
--------      ---------     ------------    ----------    -----------
2 Kb          4066 Kb        0.00 Mb         19.86 Mb               10
```

デフォルト・データ・キャッシュを変更するには、sp_cacheconfig の実行時に “default data cache” を指定します。たとえば、デフォルト・データ・キャッシュを 25MB に変更するには、次のように入力します。

```
sp_cacheconfig "default data cache", "25M"
```

この変更は動的です。新しい値を有効にするために Adaptive Server を再起動する必要はありません。

デフォルト・データ・キャッシュのサイズは絶対値です。また、キャッシュ・サイズの最小値は論理ページ・サイズの 256 倍です。つまり、論理ページ・サイズが 2KB の場合は最小サイズは 512KB で、16KB の場合は最小サイズは 4MB になります。デフォルト値は 8MB です。アップグレード中に、Adaptive Server は、デフォルト・データ・キャッシュのサイズを設定ファイル内のデフォルト・データ・キャッシュの値に設定します。

## キャッシュ領域のモニタリング

データ・キャッシュの領域とプロシージャ・キャッシュの領域を確認するには、次のように入力します。

```
sp_configure "total data cache size"
```

Adaptive Server のメモリ使用状況を確認する別の方法は、Adaptive Server の起動時にエラー・ログに書き込まれたメモリ関連のメッセージを調べることです。これらのメッセージには、データ・キャッシュとプロシージャ・キャッシュの割り付け量、同時にキャッシュ内に存在できる「コンパイル済みオブジェクト」の数、およびバッファ・プールのサイズの正確な値が含まれています。

また、これらのメッセージから、Adaptive Server のキャッシュの割り付けに関する最も正確な情報が得られます。プロシージャ・キャッシュに割り付けられるメモリ量は、procedure cache size 設定パラメータの実行値によって異なります。

次に、これらのエラー・ログ・メッセージについて説明します。

### プロシージャ・キャッシュのメッセージ

プロシージャ・キャッシュの情報を示すエラー・ログ・メッセージは、次の 2 つです。

```
server: Number of proc buffers allocated: 556
server: Number of blocks left for proc headers: 629
```

#### proc buffer

proc buffer (プロシージャ・バッファ ) は、プロシージャ・キャッシュ内のコンパイル済みオブジェクトを管理するデータ構造体です。プロシージャ・キャッシュ内に格納されているコンパイル済みオブジェクトのコピーごとに、1 つのプロシージャ・バッファが使用されます。Adaptive Server は、起動時に、必要なプロシージャ・バッファの数を判定し、その数値に単一のプロシージャ・バッファのサイズ (76 バイト ) を乗算して、必要な合計メモリ量を計算します。

#### proc header

コンパイル済みオブジェクトは、プロシージャ・キャッシュ内にある間は、proc header (プロシージャ・ヘッダ) に格納されます。格納されるオブジェクトのサイズによって、1 つまたは複数のプロシージャ・ヘッダが必要になる場合があります。プロシージャ・キャッシュ内に格納できるコンパイル済みオブジェクトの総数の上限は、使用可能なプロシージャ・ヘッダ数とプロシージャ・バッファ数のいずれか少ない方です。

プロシージャ・キャッシュの合計サイズは、プロシージャ・バッファに割り付けられたメモリの総量 (最も近いページ境界に切り上げられる) に、プロシージャ・ヘッダに割り付けられたメモリを加えた値です。

### データ・キャッシュのメッセージ

Adaptive Server は、起動時に、各キャッシュの合計サイズとキャッシュ内の各プールのサイズをエラー・ログに記録します。次は、2 つのプールがあるデフォルト・データ・キャッシュと、2 つのプールがあるユーザ定義キャッシュの例です。

```
Memory allocated for the default data cache cache: 8030 Kb
Size of the 2K memory pool: 7006 Kb
Size of the 16K memory pool: 1024 Kb
Memory allocated for the tuncache cache: 1024 Kb
Size of the 2K memory pool: 512 Kb
Size of the 16K memory pool: 512 Kb
```

## カーネル・リソース・メモリ

kernel resource memory パラメータは、スレッド・プール、タスク、およびモニタ・カウンタのような内部カーネルのみで使用するためのメモリ量を 2K ページで決定します。kernel resource memory のメモリは max memory から導出され、kernel resource memory の大きな値を受け入れるための十分な大きさが必要です。

## ユーザ接続

ユーザ接続ごとに必要なメモリの量はプラットフォームによって異なります。また、次のような設定変数の変更の影響も受けます。

- default network packet size
- stack size と stack guard size
- user log cache size

これらのパラメータを変更すると、ユーザ接続ごとに使用される領域の量も変わります。サイズの差にユーザ接続数を掛けます。たとえば、ユーザ接続数が 300 のときに stack size を 34K から 40K に増やす場合、必要なメモリ量は 1,800K 増加します。

## オープンできるデータベース、インデックス、オブジェクト

一度にオープンできるデータベース、インデックス、パーティション、オブジェクトの総数を制御する設定パラメータは、「メタデータ・キャッシュ」によって管理されています。メタデータ・キャッシュは、Adaptive Server のメモリの中のサーバ構造体の部分にあります。これらのパラメータを使用して、これらのキャッシュの領域を設定します。

- number of open databases
- number of open indexes

- number of open objects
- number of open partitions

『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

Adaptive Server は、データベースをオープンするときや、インデックス、パーティション、またはオブジェクトにアクセスするときに、そのデータベース、インデックス、オブジェクトに関する情報を、対応するシステム・テーブルから読み込みます。データベースの情報は sysdatabases、インデックスの情報は sysindexes、パーティションの情報は sysobjects にあります。

メタデータ・キャッシュとは、sysdatabases、sysindexes、syspartitions、または sysobjects に保管されている、データベース、インデックス、パーティション、またはオブジェクトに関する情報にアクセスするときに、その情報を Adaptive Server のメモリ内構造体から直接取得できるようにするものです。これによって、ディスクへのアクセスが必要な高コストの呼び出しが回避されるので、パフォーマンスが向上します。また、Adaptive Server が実行時にデータベース、インデックス、パーティション、またはオブジェクトの情報を取得するときの同期やスピンロックの競合も少なくなります。

大量のインデックス、パーティション、およびオブジェクトを格納しているデータベースでは、メタデータ・キャッシュを個別に管理すると便利です。また、ユーザ間での同時実行の可能性が高い場合も、このようにメタデータ・キャッシュを個別に管理すると効果的です。

## ロックの数

Adaptive Server 内のすべてのプロセスは、ロック構造体のプールを共有しています。ロック数の設定値を初めて見積もる場合は、予想される同時ユーザ接続数に number of worker processes の設定値を加え、その値に 20 を掛けます。

クエリが必要とするロック数にはかなりのばらつきがあります。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。ワーカー・プロセスがメモリを使用する方法については、「ワーカー・プロセス」(65 ページ) を参照してください。

Adaptive Server は、ロックがなくなるとエラー・メッセージ 1204 を発行します。“sa” または sa_role を持つユーザのみがサーバにログインして、追加ロックを設定できます。この時点で、Adaptive Server は他のユーザからのログイン試行を拒否し、エラー・ログに次のメッセージを出力します。

```
login: Process with spid <process id> could not connect to
the ASE which has temporarily run out of locks
```

この状態にある場合、Adaptive Server は、ログイン中に sa_role が自動的にアクティブにならないユーザからのログイン試行を拒否します。システム・セキュリティ・オフィスが次の手順のいずれかを実行した場合、役割が自動的に有効になります。

- sa_role を直接ユーザに付与した。
- ユーザのデフォルト役割として指定されているユーザ定義役割を使用して、sa_role をユーザに間接的に付与した。
- sa_role をユーザのログイン・プロファイルに付与し、sa_role が自動的にアクティブになるようにプロファイルを変更した。

いったんログインしたら、“sa” ユーザまたは sa_role を持つユーザは直ちに sp_configure を実行して、ロックを追加する必要があります。この場合、他の文を実行すると、Adaptive Server がエラー・メッセージ 1024 を再度発行する原因になります。

Adaptive Server がロック不足のときに、多数の “sa” ユーザまたは sa_role を持つユーザが同時にログインを試行すると、Adaptive Server は回復不可能な状態になることがあります。

### データベース・デバイスとディスク I/O 構造体

number of devices 設定パラメータは、Adaptive Server がデータの格納に使用できるデータベース・デバイスの数を制御します。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

ユーザ・プロセスが物理 I/O を実行する必要があるとき、物理 I/O はディスク I/O 構造体にキューイングされます。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## メモリを使用するその他のパラメータ

この項では、メモリ使用量の少ない設定パラメータについて説明します。

### 並列処理

並列処理は、逐次処理よりも多くのメモリを必要とします。並列処理に影響を与える設定パラメータは次のとおりです。

- number of worker processes
- memory per worker processes
- number of mailboxes と number of messages

### ワーカー・プロセス

number of worker processes は、Adaptive Server 内で同時に実行できるワーカー・プロセスの総数を設定します。各ワーカー・プロセスには、1 ユーザ接続とほぼ同じ量のメモリが必要です。

次のパラメータのいずれかを変更すると、各ワーカー・プロセスに必要なメモリ量が変化します。

- default network packet size
- stack size と stack guard size
- user log cache size
- memory per worker process

memory per worker process は、全ワーカー・プロセス用のプール内に配置される追加メモリを制御します。この追加メモリには、さまざまなデータ構造体のオーバヘッドとワーカー・プロセス間通信バッファが格納されます。

#### 並列クエリとプロシージャ・キャッシュ

各ワーカー・プロセスは、プロシージャ・キャッシュから借りた領域にクエリ・プランのコピーを作成します。コーディネーティング・プロセスは、メモリ内にクエリ・プランのコピーを 2 つ保管します。

## リモート・サーバ

Adaptive Server が他の Sybase サーバ (Backup Server、コンポーネント統合サービス、XP Server など) と通信できるようにするための設定パラメータの中にも、メモリを使用するものがあります。

リモート・サーバに影響する設定パラメータのうち、メモリを使用するものは次のとおりです。

- number of remote sites
- number of remote sites
- number of remote logins
- remote server pre-read packets

### Number of remote sites

number of remote sites では、サーバ上で同時に通信する必要があるサイト数を設定します。Backup Server だけを使用して、他のリモート・サーバを使用しない場合は、このパラメータを 1 に設定すると、データ・キャッシュとプロシージャ・キャッシュのサイズを増加することができます。

Adaptive Server から XP Server への接続では、リモート・サイトを 1 つ使用します。

### RPC に使用するその他の設定パラメータ

次に示すリモート通信の設定パラメータによって接続ごとに使用されるメモリ量はごくわずかです。

- number of remote connections
- number of remote logins

ESP を実行するための Adaptive Server から XP Server への同時接続のそれぞれについて、リモート接続 1 つとリモート・ログイン 1 つが使用されます。

remote server pre-read packets パラメータは、number of remote connections パラメータで設定された各接続に必要な領域を増加させるものであるため、先読みされるパケット数を増やすと、メモリの使用量に大きな影響を与える可能性があります。

## 参照整合性

データベース内のテーブルで多くの参照整合性制約を使用している場合に、ユーザ接続が使用する補助スキャン記述子の数がデフォルトの設定値を超えるようであれば、number of aux scan descriptors パラメータを調整しなければなりません。ほとんどの場合は、デフォルト設定で十分です。ユーザ接続での使用数が現在の設定を超えると、number of aux scan descriptors パラメータの設定値の増加を提案するエラー・メッセージが返されます。

## メモリに影響するその他のパラメータ

メモリに影響するその他のパラメータは、次のとおりです。これらの設定パラメータを再設定するときは、各パラメータが使用するメモリ量を確認し、その変更がプロシージャ・キャッシュやデータ・キャッシュに与える影響を確認してください。

| | |
|---|---|
| • additional network memory | • max online engines |
| • allow resource limits | • max SQL text monitored |
| • audit queue size | • number of alarms |
| • event buffers per engine | • number of large i/o buffers |
| • max number network listeners | • permission cache entries |

# ステートメント・キャッシュ

ステートメント・キャッシュは、キャッシュしたステートメントの SQL を保存するために使用されます。Adaptive Server は、受信した SQL 文とキャッシュしている SQL 文を比較し、一致する場合は既に保存している SQL 文のプランを実行します。これにより、アプリケーションは、同じ文を繰り返し実行するたびにクエリをコンパイルする必要がなくなります。

## ステートメント・キャッシュの設定

ステートメント・キャッシュを使用すると、Adaptive Server は、新たに受信したアドホック SQL 文とキャッシュしている SQL 文を比較し、一致する場合は、最初の実行時にキャッシュしたプランを使用します。この方法により、既にプランがある SQL 文を Adaptive Server が再コンパイルする必要がなくなります。

ステートメント・キャッシュはサーバワイドなリソースであり、プロシージャ・キャッシュ・メモリ・プールのメモリを割り付け、消費します。ステートメント・キャッシュのサイズは、statement cache size 設定パラメータを使用して動的に設定します。

---

**注意** ステートメント・キャッシュのメモリ量を割り付け解除するか減らす場合、割り付けられている元のメモリは、Adaptive Server を再起動するまで解放されません。

---

構文は次のとおりです。*size_of_cache* は、2K ページ単位のサイズです。

```
sp_configure "statement cache size", size_of_cache
```

たとえば、ステートメント・キャッシュを 5000 × 2K ページに設定するには、次のように入力します。

```
sp_configure "statement cache size", 5000
```

『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

ステートメント・キャッシュのメモリを設定するときは次の点に注意してください。

- プロシージャ・キャッシュ・メモリ・プールに割り付けられるメモリ量は、statement cache size 設定パラメータと procedure cache size 設定パラメータの合計です。ステートメント・キャッシュ・メモリはプロシージャ・キャッシュ・メモリ・プールから使用されます。

- statement cache size を使用すると、キャッシュされる SQL テキストとプランで使用できるプロシージャ・キャッシュ・メモリの量が制限されます。Adaptive Server は、statement cache size 設定パラメータで設定した量を超えるメモリをステートメント・キャッシュに使用することはできません。
- *@@nestlevel* には、ユーザ・セッションでの現在の実行のネスト・レベルが含まれます。初期値は 0 です。ストアド・プロシージャまたはトリガが、別のストアド・プロシージャやトリガを呼び出すたびに、ネスト・レベルが 1 つずつ増加します。ネスト・レベルは、キャッシュされる文が作成されたときも 1 つ増えます。最大値の 16 を超えると、トランザクションはアボートします。
- statement cache size 設定パラメータで割り付けられたメモリも含め、プロシージャ・キャッシュ・メモリはすべてストアド・プロシージャに使用でき、キャッシュされた文は LRU (least recently used、最も長い間使用されていない) ベースで置換されます。
- max memory 設定パラメータは、ステートメント・キャッシュの設定と同じ量だけ増やしてください。たとえば、ステートメント・キャッシュのサイズを 100 × 2K ページに当初設定した場合、これと同じ量だけ max memory を増やします。
- statement cache size 設定パラメータを使用している場合、set statement cache を使用してセッション・レベルでステートメント・キャッシュの無効と有効を切り替えることができます。ステートメント・キャッシュは、サーバ・レベルで設定されている場合、セッション・レベルではデフォルトで有効になります。
- キャッシュされた文はそれぞれ 1 つのオブジェクト記述子を使用するため、number of open objects 設定パラメータを使用してオブジェクト記述子の数を増やす必要があります。キャッシュできる SQL 文の数を推定するには、「ステートメント・キャッシュのサイズ設定」(71 ページ) を参照してください。

## アドホック・クエリの処理

Adaptive Server が文をキャッシュすると、文はアドホック・クエリからライトウェイト・ストアド・プロシージャに変更されます。たとえば、文をキャッシュせずにデフォルトまたはルールを呼び出す文として、同じバッチで sp_bindefault または sp_bindrule を発行すると、Adaptive Server はエラー・メッセージ 540 を発行します。ただし、文をキャッシュすると、Adaptive Server はデフォルトまたはルールをカラムにバインドします。

文がキャッシュされ、ストアド・プロシージャとして実行されると、Adaptive Server は、正規化エラーの代わりに実行時エラーを発行します。たとえば、文がキャッシュされない場合、このクエリはエラー番号 241 を生成しますが、文がキャッシュされると、`Truncation error` を生成してコマンドを中止します。

```
create table t1(c1 numeric(5,2)
go
insert t1 values(3.123)
go
```

ステートメント・キャッシュを使用してアドホック SQL 文を処理するには、次の手順を実行します。

1　Adaptive Server は文を解析します。

キャッシュすべき文である場合 (「キャッシュする条件」(70 ページ) を参照 )、Adaptive Server は文のハッシュ値を計算します。このハッシュ値を使用して、ステートメント・キャッシュ内で一致する文を検索します (「文の一致基準」(70 ページ) を参照)。

- ステートメント・キャッシュ内で一致する文が見つかった場合は、手順 4 に進みます。
- 一致する文が見つからない場合は、手順 2 に進みます。

2　Adaptive Server は、この SQL 文のテキストをキャッシュします。

3　Adaptive Server は、SQL 文をライトウェイト・ストアド・プロシージャでラップし、ローカル変数が含まれる場合にはそれをプロシージャ・パラメータに変更します。この時点では、ライトウェイト・プロシージャの内部表現はプランにコンパイルされません。

4　Adaptive Server は、SQL 文を、対応するライトウェイト・プロシージャの execute 文に変換します。

- キャッシュにプランがない場合は、プロシージャをコンパイルしてプランをキャッシュします。ローカル変数に割り当てられたランタイム値を使用してプランをコンパイルします。
- プランが存在していても無効である場合は、手順 3 に戻り、キャッシュされている SQL 文のテキストを使用します。

5　Adaptive Server はプロシージャを実行します。代わりにライトウェイト・プロシージャを使用すると、*@@nestlevel* グローバル変数が増加します。

### 文の一致基準

Adaptive Server は、アドホック SQL 文とキャッシュした文の照合に、SQL テキストや、ログイン (特に両方のユーザが sa_role を持っている場合)、ユーザ ID、データベース ID、セッション状態の設定を使用します。関連するセッション状態は、次に示す set コマンド・パラメータの設定で構成されます。

- forceplan
- jtc
- parallel_degree
- prefetch
- quoted_identifier
- table count
- transaction isolation level
- chained (トランザクション・モード)

これらのパラメータの設定内容により、キャッシュした文に対して Adaptive Server が生成するプランの動作が決まります。『リファレンス・マニュアル：コマンド』を参照してください。

---

**注意** ステートメント・キャッシュを有効にする場合は、set chained on/off をバッチ内に設定してください。

---

### キャッシュする条件

Adaptive Server は、次の条件に従って文をキャッシュします。

- 現在、Adaptive Server では、少なくとも 1 つのテーブル参照を含む select 文、update 文、delete 文、insert select 文がキャッシュされます。
- abstract plan dump パラメータまたは abstract plan load パラメータが有効になっている場合、文はキャッシュされません。つまり、ステートメント・キャッシュを有効にすることと、abstract plan load 設定パラメータと abstract plan dump at 設定パラメータを有効にすることを同時に行うことはできません。
- select into 文、カーソル文、動的文、単純な insert 文 (insert select を除く) や、ストアド・プロシージャ、ビュー、トリガ内の文はキャッシュされません。また、テンポラリ・テーブルを参照する文や、バイナリ・ラージ・オブジェクト (BLOB) データ型として送信される言語パラメータを含む文もキャッシュされません。許容サイズを超える文もキャッシュされません。さらに、if exists または if not exists という条件句に含まれる select 文もキャッシュされません。

### ステートメント・キャッシュのサイズ設定

キャッシュされる文は、SQL テキストの長さによって異なりますが、それぞれ約 1K のステートメント・キャッシュ用メモリが必要です。キャッシュされるプランは、それぞれ少なくとも 2K のプロシージャ・キャッシュ・メモリが必要です。必要なステートメント・キャッシュ・メモリを推定するには、キャッシュする各文ごとの次の値を合計します。

- SQL 文の長さ (バイト単位で 256 の倍数に切り上げる)。
- 約 100 バイトのオーバヘッド。
- プロシージャ・キャッシュ内のプランのサイズ。このサイズは、キャッシュされる文のみを含むストアド・プロシージャ・プランのサイズと同じです。キャッシュされた 1 つの文を複数のユーザが同時に使用する場合、プランが複製されることがあります。

## ステートメント・キャッシュのモニタリング

sp_sysmon は、文のキャッシュとストアド・プロシージャの実行についてレポートします。ステートメント・キャッシュは次のカウンタを使用してモニタされます。

- Statements Cached － キャッシュに追加された SQL 文の数。通常、これは Statements Not Found と同数です。statements cached の値の方が小さい場合は、ステートメント・キャッシュがアクティブな文で一杯になっています。
- Statements Found in Cache － クエリ・プランが再使用された回数。キャッシュのヒット回数が少ない場合、ステートメント・キャッシュが小さすぎることを示している場合があります。
- Statements Not Found － 繰り返された SQL 文が存在しないことを示します。statements found in cache と statements not found の合計が、処理対象になる発行された SQL 文の総数です。
- Statements Dropped － キャッシュから削除された文の数。この値が大きい場合、プロシージャ・キャッシュ・メモリ量が不十分であるか、ステートメント・キャッシュ・サイズが小さすぎることがあります。
- Statements Restored － SQL テキストから再生成されたクエリ・プランの数。値が大きい場合は、プロシージャ・キャッシュ・サイズが不十分です。
- Statements Not Cached － ステートメント・キャッシュが有効になっていれば Adaptive Server がキャッシュするはずであった文の数。ただし、Statements Not Cached では、何種類のユニークな SQL 文がキャッシュされるかはわかりません。

次に、sp_sysmon のサンプル出力を示します。

```
  SQLStatement Cache:
   Statements Cached                0.0           0.0           0        n/a
   Statements Found in Cache        0.7           0.0           2        n/a
   Statements Not Found             0.0           0.0           0        n/a
   Statements Dropped               0.0           0.0           0        n/a
   Statements Recompiled            0.3           0.0           1        n/a
   Statements Not Cached            1.3           0.0           4       n/a
```

## ステートメント・キャッシュの消去

dbcc purgesqlcache を実行すると、ステートメント・キャッシュからすべての SQL 文が削除されます。現在実行している文は削除されません。

dbcc purgesqlcache を実行するには sa_role が必要です。

## 文の要約の出力

dbcc prsqlcache を実行すると、ステートメント・キャッシュ内の文の要約が出力されます。*oid* オプションを指定すると、出力する文のオブジェクト ID を指定できます。また、*printopt* オプションでは、トレースの説明を出力するか (0 を指定) または showplan オプションを出力するか (1 を指定 ) を指定できます。*oid* または *printopt* に値を指定しないで dbcc prsqlcache を実行すると、ステートメント・キャッシュの内容全体が表示されます。

dbcc prsqlcache を実行するには sa_role が必要です。

次の例は、キャッシュ内のすべての文の情報を出力します。

```
dbcc prsqlcache
Start of SSQL Hash Table at 0xfc67d830
Memory configured: 1000 2k pages             Memory used: 18 2k pages
Bucket# 625 address 0xfc67ebb8

SSQL_DESC 0xfc67f9c0
ssql_name *ss1248998166_0290284638ss*
ssql_hashkey 0x114d645e ssql_id 1248998166
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 8
SQL TEXT: select * from sysobjects where name like "sp%"

Bucket# 852 address 0xfc67f2d0
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
```

```
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

End of SSQL Hash Table

DBCC execution completed. If DBCC printed error messages, contact a user with
```

また、次の例のように、特定のオブジェクト ID の情報を得ることができます。

```
dbcc prsqlcache (1232998109, 0)
SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1      ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

次の例は、showplan の出力のために printopt パラメータに 1 を指定しています。

```
dbcc prsqlcache (1232998109, 1)

SSQL_DESC 0xfc67f840
ssql_name *ss1232998109_1393445479ss*
ssql_hashkey 0x530e4a67 ssql_id 1232998109
ssql_suid 1             ssql_uid 1       ssql_dbid 1
ssql_status 0x28        ssql_parallel_deg 1
ssql_tab_count 0        ssql_isolate 1  ssql_tranmode 0
ssql_keep 0             ssql_usecnt 1   ssql_pgcount 3
SQL TEXT: select name from systypes where allownulls = 0

QUERY PLAN FOR STATEMENT 1 (at line 1).

      STEP 1
         The type of query is SELECT.
         FROM TABLE
            systypes
         Nested iteration.
         Table Scan.
         Forward scan.

         Positioning at start of table.
         Using I/O Size 2 Kbytes for data pages.
         With LRU Buffer Replacement Strategy for data pages.
         DBCC execution completed. If DBCC printed error messages,
         contact a user with
          System Administrator (SA) role.
```

### キャッシュされた文の SQL プランの表示

次の関数を使用して、キャッシュされた文のプランを表示します。

```
show_plan(spid, batch_id, context_id, statement_number)
```

上記のパラメータの意味は、次のとおりです。

- *spid* − 任意のユーザ接続のプロセス ID です。
- *batch_id* − バッチのユニークな番号です。
- *context_id* − すべてのプロシージャ (またはトリガ) のユニークな番号です。
- *statement_number* − バッチ内の現在の文の番号です。

効率がよくない文がある場合は、オプティマイザ設定の変更または抽象プランの指定によってプランを変更できます。

既存の show_plan 引数で最初の int 変数を “-1” として指定すると、2 番目のパラメータは、show_plan によって SSQLID として処理されます。

---

**注意** ステートメント・キャッシュの単一のエントリは、複数の異なる SQL プランと関連付けられている可能性があります。show_plan では、そのうちの 1 つのプランしか表示されません。

---

第 4 章

# データ・キャッシュの設定

データ・キャッシュを管理する主な目的は、データ・キャッシュを再設定してパフォーマンスを向上させることです。この章では、主にデータ・キャッシュ上の処理メカニズムについて説明します。データ・キャッシュに関係するパフォーマンスの概念については、『パフォーマンス&チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」で説明しています。

# Adaptive Server のデータ・キャッシュ

データ・キャッシュには、現在使用されているデータ、インデックス、ログ・ページの他に、Adaptive Server が最近使用したページも保持されます。Adaptive Server をインストールした直後は、1 つのデフォルト・データ・キャッシュが割り当てられており、すべてのデータ、インデックス、ログのアクティビティに使用されます。このキャッシュのデフォルト・サイズは 8M です。他のキャッシュを作成しても、デフォルト・データ・キャッシュのサイズが縮小することはありません。また、名前付きキャッシュ内やデフォルト・キャッシュ内にプールを作成すると、大容量 I/O を実行することができます。データベース、テーブル (syslogs テーブルを含む)、インデックス、テキスト・ページ・チェーン、イメージ・ページ・チェーンを、この名前付きデータ・キャッシュにバインドできます。

大きな I/O サイズを使用すれば、データのプリフェッチによってパフォーマンスが向上するとクエリ・オプティマイザが判断した場合にプリフェッチを実行できるようになります。たとえば、16K の論理ページを使用するように設定されているサーバの I/O サイズが 128K の場合は、1 つのエクステント (8 ページ) 全体を一度に読み込むことができ、I/O を 8 回実行する必要はありません。

また、大きな I/O サイズに合わせてバッファ・プールを設定すると、ソートのパフォーマンスも向上します。

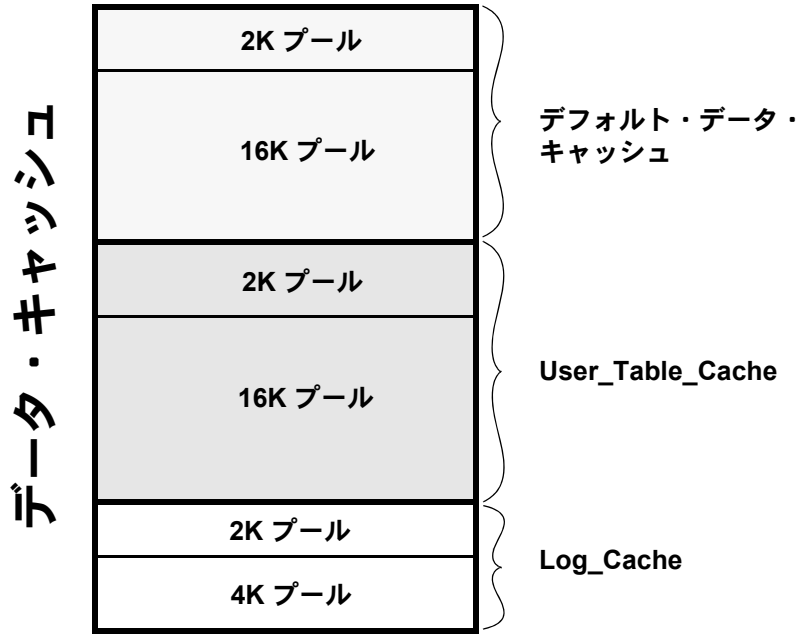
名前付きデータ・キャッシュを設定しても、デフォルト・キャッシュが別のキャッシュ構造体に分割されることはありません。作成した名前付きデータ・キャッシュを使用できるのは、そのキャッシュに明示的にバインドされたデータベースまたはデータベース・オブジェクトだけです。名前付きデータ・キャッシュに明示的にバインドされていないオブジェクトはすべて、デフォルト・データ・キャッシュを使用します。

Adaptive Server のユーザ設定可能なデータ・キャッシュは、特にマルチプロセッサ・サーバ上でのパフォーマンス改善に役立ちます。

図 4-1 は、デフォルト・データ・キャッシュと 2 つの名前付きデータ・キャッシュを持つキャッシュを示します。このサーバは 2K の論理ページを使用します。

デフォルト・キャッシュには、2K のプールと 16K のプールがあります。User_Table_Cache キャッシュにも、2K のプールと 16K のプールがあります。Log_Cache には、2K のプールと 4K のプールがあります。

**図 4-1: デフォルト・キャッシュと 2 つの名前付きデータ・キャッシュを持つデータ・キャッシュ**




# キャッシュ設定コマンドとシステム・プロシージャ

表 4-2 は、名前付きデータ・キャッシュの設定、キャッシュに対するオブジェクトのバインドとバインド解除、キャッシュのバインド情報を表示するためのコマンドとシステム・プロシージャのリストです。この他に、データベース・オブジェクトのサイズを確認するために使用するプロシージャ、およびオブジェクト、コマンド、セッションの各レベルでキャッシュの使用を制御するコマンドも示します。

**表 4-1: 名前付きデータ・キャッシュを設定するコマンドとプロシージャ**

| コマンドまたはプロシージャ | 機能 |
|---|---|
| sp_cacheconfig | 名前付きキャッシュを作成または削除する。サイズ、キャッシュ・タイプ、キャッシュ方式、キャッシュ・パーティション数を変更する。 |
| sp_poolconfig | I/O プールを作成または削除する。サイズ、ウォッシュ・サイズ、非同期プリフェッチ率の制限値を変更する。 |
| sp_bindcache | データベースまたはデータベース・オブジェクトをキャッシュにバインドする。 |
| sp_unbindcache | 特定のオブジェクトまたはデータベースとキャッシュとのバインドを解除する。 |
| sp_unbindcache_all | 指定のキャッシュにバインドされているすべてのオブジェクトのバインドを解除する。 |

| コマンドまたはプロシージャ | 機能 |
|---|---|
| sp_helpcache | データ・キャッシュについての概要をレポートし、キャッシュにバインドされているデータベースとデータベース・オブジェクトをリストする。 |
| sp_cachestrategy | テーブルまたはインデックスに設定されているキャッシュ方式についてレポートする。また、プリフェッチや MRU 方式を無効または有効にする。 |
| sp_logiosize | ログのデフォルト I/O サイズを変更する。 |
| sp_spaceused | テーブルとインデックスのサイズ、またはデータベースで使用する領域のサイズに関する情報を表示する。 |
| sp_estspace | テーブルに保存されるローの数から、テーブルとインデックスのサイズを見積もる。 |
| sp_help | テーブルがバインドされているキャッシュをレポートする。 |
| sp_helpindex | インデックスがバインドされているキャッシュをレポートする。 |
| sp_helpdb | データベースがバインドされているキャッシュをレポートする。 |
| set showplan on | クエリの I/O サイズとキャッシュ利用方式をレポートする。 |
| set statistics io on | クエリ用に実行される読み込み回数をレポートする。 |
| set prefetch [on \|off] | 個々のセッションのプリフェッチを有効または無効にする。 |
| select... (prefetch...lru \| mru) | 指定の I/O サイズまたは MRU 置換方式をサーバに使用させる。 |

sp_cacheconfig でデータ・キャッシュを設定するためのパラメータのほとんどは動的であり、サーバを再起動しなくても設定を反映できます。静的アクションと動的アクションの詳細については、表 4-2 (81 ページ) を参照してください。

コマンドを使用して名前付きデータ・キャッシュを対話形式で設定する方法の他に、*$SYBASE* ディレクトリ内の設定ファイルを編集する方法もあります。ただし、設定ファイルを編集する場合はサーバを再起動する必要があります。「設定ファイルによるデータ・キャッシュの設定」(108 ページ) を参照してください。

# データ・キャッシュに関する情報

sp_cacheconfig は、名前付きデータ・キャッシュの作成と設定を行います。Adaptive Server をインストールした直後は、default data cache という名前のキャッシュ 1 つだけが存在します。キャッシュについての情報を表示するには、次のように入力します。

```
sp_cacheconfig
```

sp_cacheconfig の実行結果は次の例のように表示されます。

```
Cache Name                 Status    Type     Config Value  Run Value
-------------------------- --------- -------- ------------ ------------
default data cache         Active    Default         0.00 Mb    59.44 Mb
                                               ------------ ------------
                                     Total           0.00 Mb    59.44 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 0.00 Mb,   Run Size: 59.44 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:             1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   12174 Kb     0.00 Mb     59.44 Mb     10
```

レポートの先頭のブロックに各キャッシュについての概要情報が表示され、最後に、設定されているすべてのキャッシュのサイズの合計が表示されます。また、キャッシュごとのブロックに、そのキャッシュ内のメモリ・プールの設定情報が表示されます。

各キャッシュ情報を表示するカラムは、次のとおりです。

- Cache Name (キャッシュ名) − キャッシュの名前が表示されます。
- Status (ステータス) − キャッシュがアクティブかどうかを示します。値は次のとおりです。
  - “Pend/Act” (保留／アクティブ) − キャッシュは作成された直後で、まだアクティブではない。
  - “Active” (アクティブ) − キャッシュは現在アクティブ状態。
  - “Pend/Del” (保留／削除) − キャッシュは削除されている。キャッシュ・サイズは、sp_cacheconfig と sp_poolconfig を使用して 0 にリセットされた。詳細については、「データ・キャッシュの設定」(81 ページ )を参照してください。

- Type (タイプ) − キャッシュにデータ・ページとログ・ページが混在可能 (Mixed) か、ログ・ページだけを保管できる (Log Only) かを示します。タイプが “Default” となるのはデフォルト・キャッシュだけです。デフォルト・データ・キャッシュのタイプを変更したり、他のキャッシュのタイプを “Default” に変更したりすることはできません。
- Config Value ( 設定値 ) − 現在の設定値が表示されます。この例では、デフォルト・データ・キャッシュは明示的に設定されていないため、サイズは 0 になっています。
- Run Value (実行値) − Adaptive Server が現在使用しているサイズが表示されます。

2 番目のブロックでは、初めにキャッシュの情報が 3 行にわたって表示されます。最初の 2 行には、先頭の概要ブロックと同じ情報が表示されます。3 行目の “Config Replacement” と “Run Replacement” には、“strict LRU” と “relaxed LRU” のいずれかのキャッシュ方式が表示されます。Run Replacement には現在有効な設定 (“strict LRU” または “relaxed LRU”) が表示されます。サーバの再起動後に方式を変更した場合は、Config Replacement と Run Replacement の設定は異なります。

次に、キャッシュ内のプールの情報が、プールごとに 1 行表示されます。

- IO Size (I/O サイズ) − プール内のバッファのサイズを示します。プールのデフォルト・サイズは、サーバの論理ページ・サイズです。新しいキャッシュの設定直後は、すべての領域がそのプールに割り付けられます。有効なサイズは、2K、4K、8K、16K です。
- Wash Size (ウォッシュ・サイズ) − プールのウォッシュ・サイズを示します。「メモリ・プールのウォッシュ・エリアの変更」(98 ページ) を参照してください。
- Config Size と Run Size − 設定されているサイズと現在使用されているサイズを示します。他にも、プール間で領域を移動しようとして解放できなかった領域があると、値が一致しないことがあります。
- Config Partition と Run Partition − 設定されているキャッシュ・パーティションの数と現在使用されているパーティション数を示します。再起動後にパーティションの数を変更した場合は、これらの数が異なることがあります。
- APF Percent ( 非同期プリフェッチ率 ) − そのプールの中で、非同期プリフェッチで取り込まれたけれども未使用であるバッファを保持できる割合が表示されます。

合計行には、表示されているキャッシュのサイズの合計が表示されます。

# データ・キャッシュの設定

Adaptive Server が使用するデフォルト・データ・キャッシュとプロシージャ・キャッシュを指定するには、絶対値を使用します。キャッシュの構成を計画して実装するには、初めに max memory 設定パラメータを設定します。max memory を設定した後で、サーバ上のデータ・キャッシュに割り付ける領域の大きさを決めます。データ・キャッシュのサイズを制限するものは、システム上で使用可能なメモリのみです。ただし、max memory は total logical memory よりも大きい値でなければなりません。デフォルト・データ・キャッシュと他のすべてのユーザ定義キャッシュのサイズには、絶対値を指定してください。Adaptive Server のメモリの使用については、「第 3 章 メモリの設定」を参照してください。

データ・キャッシュは次の 2 つの方法で設定できます。

- 対話形式 (sp_cacheconfig と sp_poolconfig を使用)。この方法は動的なので、Adaptive Server を再起動する必要はありません。
- 設定ファイルを編集します。この方法は静的なので、Adaptive Server を再起動する必要があります。

設定ファイルを使用する方法については、「設定ファイルによるデータ・キャッシュの設定」(108 ページ) を参照してください。

データ・キャッシュを変更するか、sp_cacheconfig または sp_poolconfig を実行すると、新しいキャッシュまたはプールの情報が設定ファイルに書き込まれ、変更前のファイルがバックアップ・ファイルにコピーされます。バックアップ・ファイル名を示すメッセージがエラー・ログに記録されます。

sp_cacheconfig で実行するアクションには、動的なものと静的なものがあります。

1 つのコマンドで静的パラメータと動的パラメータの両方を指定できます。動的な変更は直ちに実行され、すべての変更内容 (静的と動的の両方) が設定ファイルに書き込まれます。静的な変更内容は、次回のサーバ起動時に反映されます。

**表 4-2: sp_cacheconfig の動的アクションと静的アクション**

| sp_cacheconfig の動的アクション | sp_cacheconfig の静的アクション |
|---|---|
| 新しいキャッシュの追加 | キャッシュ・パーティション数の変更 |
| 既存のキャッシュへのメモリの追加 | キャッシュ・サイズの削減 |
| キャッシュの削除 | 置換方式の変更 |
| キャッシュ・タイプの変更 | |

静的パラメータを再設定するには、次の手順でキャッシュを削除してから再作成します。

1　キャッシュのバインドを解除します。

2　キャッシュを削除します。

3　新しい設定を使用して、キャッシュを再作成します。

4　キャッシュにオブジェクトをバインドします。

## 新しいキャッシュの作成

新しいキャッシュを作成するには sp_cacheconfig を使用します。『リファレンス・マニュアル：プロシージャ』の「sp_cacheconfig」を参照してください。

データ・キャッシュ・サイズの最大値を制限するものは、システム上で使用可能なメモリ容量のみです。Adaptive Server のグローバル・メモリの容量によって、新しいキャッシュの作成に必要なメモリが決まります。キャッシュは、次のように作成されます。

- デフォルトのウォッシュ・サイズが定義されます。
- 非同期プリフェッチ・サイズは、global async prefetch limit の値に設定されます。
- デフォルトのバッファ・プールのみが存在します。

これらの値を再設定するには、sp_poolconfig を使用します。

pubs_cache という名前の 10MB のキャッシュを作成するには、次のコマンドを実行します。

```
sp_cacheconfig pubs_cache, "10M"
```

このコマンドを実行すると、システム・テーブルが変更され、設定ファイルに新しい値が書き込まれます。キャッシュはただちにアクティブになります。sp_cacheconfig を実行すると、次のように結果が表示されます。

```
sp_cacheconfig

Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
default data cache             Active    Default       0.00 Mb      8.00 Mb
pubs_cache                     Active    Mixed        10.00 Mb     10.00 Mb
------------ ------------
Total      10.00 Mb     18.00 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
Config Size: 0.00 Mb,   Run Size: 8.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   1636 Kb      0.00 Mb      8.00 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
Config Size: 10.00 Mb,   Run Size: 10.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
```

pubs_cache はアクティブになっており、すべての領域が最小のプールに割り付けられています。

---

**注意** 新しいキャッシュを作成するとき、指定した追加メモリ量が max memory に対して検証されます。total logical memory の値と、要求した追加メモリ量の合計が max memory を超える場合はエラーとなり、変更は実行されません。

---

Adaptive Server を再起動することなく、必要な数のキャッシュを作成できます。

## 新しいキャッシュ用の領域が不足している場合

要求した量のメモリをすべて割り付けることができない場合は、使用可能なメモリだけが割り付けられ、次のメッセージが発行されます。

```
ASE is unable to get all the memory requested (%d). (%d) kilobytes have been allocated
dynamically.
```

ただし、この追加メモリが実際に割り付けられるのは、Adaptive Server の次回の再起動時です。

Adaptive Server は、リソース制約によりメモリの一部が利用できないため領域が不足しているということを通知する場合があります。システム管理者は、このリソース制約が一時的なものであることを確認する必要があります。この状態が解消されない場合は、それ以降の再起動に失敗することがあります。

たとえば、max memory が 700MB、pub_cache が 100MB、サーバの total logical memory が 600MB であるとします。pub_cache に 100MB を追加する場合の追加メモリ量は、max memory の範囲内です。ただし、90MB しか割り付けることができない場合は、その量が動的に割り付けられますが、設定ファイル内のキャッシュの `size field` は、100MB に更新されます。次の再起動時に、Adaptive Server はすべてのデータ・キャッシュ用のメモリを一度に取得するので、pub_cache のサイズは 100MB になります。

## 既存の名前付きキャッシュへのメモリの追加

既存のキャッシュにメモリを追加するには、sp_cacheconfig を使用します。

割り付けた追加メモリは、Adaptive Server のページ・サイズのプールに追加されます。たとえば、サーバの論理ページ・サイズが 4K ならば、プールの最小サイズは 4K バッファ・プールです。キャッシュにパーティションがある場合は、追加メモリは各パーティションに均等に配分されます。

使用可能なメモリが不足している場合は、使用できる量のメモリが割り付けられ、サーバを再起動したときに全量が割り付けられます。「新しいキャッシュ用の領域が不足している場合」(83 ページ) を参照してください。

たとえば、pub_cache という名前のキャッシュ (現在のサイズは 10MB) に 2MB を追加するには、次のように入力します。

```
sp_cacheconfig pub_cache, "12M"
```

メモリの追加後に sp_cacheconfig を実行すると、次のような出力が生成されます。

```
sp_cacheconfig pub_cache
 Cache Name                     Status    Type     Config Value Run Value
------------------------------ --------- -------- ------------ ------------
pub_cache                      Active    Mixed      12.00 Mb       12.00
------------ ------------
Total      12.00 Mb     12.00 Mb
================================================================================
Cache: pub_cache,   Status: Active,   Type: Mixed
Config Size: 12.00 Mb,   Run Size: 12.00 Mb
Config Replacement: strict LRU,   Run Replacement: strict LRU
Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
4 Kb   2456 Kb      0.00 Mb     12.00 Mb     10
```

この変更によって、メモリがデータベースのページ・サイズのバッファ・プールに追加され、必要に応じてウォッシュ・サイズが再計算されます。ウォッシュ・サイズに絶対値が設定されている場合は、再計算は行われません。

## キャッシュ・サイズの削減

キャッシュ・サイズを縮小する場合は、変更を反映させるために Adaptive Server を再起動してください。

次の例は、pubs_log というキャッシュについてのレポートです。

```
sp_cacheconfig pubs_log

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      7.00 Mb      7.00 Mb
                                                ------------ ------------
                                      Total       7.00 Mb      7.00 Mb
================================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    920 Kb      0.00 Mb      4.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

次のコマンドを実行して、pubs_log キャッシュのサイズを現在の 7MB から 6MB に減らします。

```
sp_cacheconfig pubs_log, "6M"
```

Adaptive Server の再起動後に sp_cacheconfig を実行すると、次のようなレポートが出力されます。

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                              ------------ ------------
                                    Total         6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

データ・キャッシュのサイズを小さくするときは、除去する領域全体がデフォルト・プール内で除去可能な大きさでなければなりません (使用可能な最小のサイズ )。データ・キャッシュのサイズを縮小できるようにするために、他のプールからデフォルトのプールに領域を移さなければならない場合があります。最後の例でキャッシュのサイズを 3MB に減らすには、sp_poolconfig を使用して、4K のプールから 2K のデフォルト・プールにメモリを移します。メモリは、「名前付きキャッシュ用に使用できるメモリ」に移動します。「メモリ・プールのサイズの変更」(102 ページ) を参照してください。

## キャッシュの削除

データ・キャッシュを完全に消去するには、次のようにサイズを 0 にリセットします。

```
sp_cacheconfig pubs_log, "0"
```

キャッシュはただちに削除されます。

**注意** デフォルト・データ・キャッシュは削除できません。

オブジェクトがバインドされているデータ・キャッシュを削除した場合、そのキャッシュはメモリ内に残り、次のメッセージが発行されます。

```
Cache cache_name not deleted dynamically. Objects are bound to the cache. Use
sp_unbindcache_all to unbind all objects bound to the cache.
```

設定ファイル内および sysconfigures 内のこのキャッシュに対応するエントリが削除されます。キャッシュ自体は Adaptive Server の次回の再起動時に削除されます。

キャッシュを作成し直して Adaptive Server を再起動すると、バインドは再び有効になります。

キャッシュにバインドされているオブジェクトをすべて表示するには、sp_helpcache を使用します。オブジェクトのバインドを解除するには、sp_unbindcache_all を使用します。『リファレンス・マニュアル：プロシージャ』を参照してください。

## デフォルト・キャッシュの明示的な設定

デフォルト・データ・キャッシュのサイズは、明示的に設定してください。キャッシュに使用できるメモリの空き容量を調べるには、sp_helpcache を使用します。次に例を示します。

```
sp_helpcache
Cache Name                Config Size    Run Size        Overhead
-----------------------   ------------  ----------      ----------
default data cache        50.00 Mb       50.00 Mb        3.65 Mb
pubs_cache                10.00 Mb       10.00 Mb        0.77 Mb

Memory Available For      Memory Configured
Named Caches              To Named Caches
--------------------      ----------------
91.26 Mb                  91.25 Mb

------------------ Cache Binding Information: ------------------

Cache Name       Entity Name    Type         Index Name     Status
----------         -----------      -----          ---------        ------
```

デフォルト・データ・キャッシュの絶対サイズを指定するには、default data cache とサイズ値を指定して sp_cacheconfig を実行します。たとえば、デフォルト・データ・キャッシュ・サイズを 25MB に設定するには、次のように入力します。

```
sp_cacheconfig "default data cache", "25M"
```

sp_helpconfig をもう一度実行すると、“Config Value” にこの値が表示されます。

```
sp_cacheconfig

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     50.00 Mb
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                                ------------ ------------
                                      Total      10.00 Mb    60.00 Mb
==========================================================================
```

```
Cache: default data cache,   Status: Active,   Type: Default
    Config Size: 25.00 Mb,   Run Size: 50.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb 10110 Kb     00.00 Mb    50.00 Mb      10
============================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
    Config Size: 10.00 Mb,   Run Size: 10.00 Mb
    Config Replacement: strict LRU,   Run Replacement: strict LRU
    Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb   0.00 Mb    10.00 Mb    10
```

名前付きキャッシュのサイズを変更するには、sp_cacheconfig を使用します。データ・キャッシュのサイズを変更しても、他のキャッシュのサイズへの影響はありません。同様に、デフォルト・データ・キャッシュのサイズを指定した後で、他のユーザ定義キャッシュを設定しても、デフォルト・データ・キャッシュのサイズが変更されることはありません。

---

**注意** デフォルト・データ・キャッシュを設定した後で、max memory の値を小さくした結果、total logical memory の値が max memory の値を上回る場合には、Adaptive Server は起動しません。設定ファイルを編集して、他のキャッシュのサイズを増やし、メモリを必要とするパラメータの値を増やして、total logical memory の値が max memory の値を超えるようにします。「第 3 章 メモリの設定」を参照してください。

---

デフォルト・データ・キャッシュとすべてのユーザ定義キャッシュは、絶対値によって明示的に設定します。さらに、多くの設定パラメータがメモリを使用します。パフォーマンスを最大限に引き出してエラー発生を防ぐには、すべてのキャッシュと、メモリを使用するすべての設定パラメータに対応できるレベルまで、max memory の設定値を大きくします。

max memory の設定値が total logical memory の設定値よりも小さい場合は、警告メッセージが発行されます。

### キャッシュ・タイプの変更

キャッシュをトランザクション・ログ専用に予約するには、キャッシュのタイプを “logonly” に変更します。この変更は動的です。

次のコマンドは、“logonly” タイプのキャッシュ pubs_log を作成します。

```
sp_cacheconfig pubs_log, "7M", "logonly"
```

キャッシュの初期状態は次のように表示されます。

```
Cache Name           Status    Type     Config Value Run Value
-------------------- --------- -------- ------------ ------------
pubs_log             Pend/Act  Log Only      7.00 Mb      0.00 Mb
                                        ------------ ------------
                       Total                7.00 Mb      0.00 Mb
```

ログ以外のオブジェクトがキャッシュにバインドされている場合を除いて、次のように既存の “mixed (混合)” キャッシュのタイプを変更できます。

```
sp_cacheconfig pubtune_cache, logonly
```

トランザクション量が多い環境でも、デフォルト値がサーバの論理ページ・サイズの 2 倍であるときに Adaptive Server のパフォーマンスが最も良好になります (サーバの論理ページ・サイズが 2K の場合は 4K、論理ページ・サイズが 4K の場合は 8K)。ページ・サイズが大きい場合に (4、8、16K)、システムの最適な設定を求めるには、sp_sysmon を実行します。「ログ・キャッシュとログ I/O サイズの一致」(92 ページ) を参照してください。

## キャッシュ置換方式の設定

キャッシュがテーブルまたはインデックス専用で、システムが安定状態のときにそのキャッシュでバッファの置換がほとんど発生しない、またはまったく発生しない場合は、リラックス LRU (Least Recently Used) 置換方式を設定できます。この方式を使用すると、バッファの置換がほとんど発生しない、またはまったく発生しない場合にキャッシュのパフォーマンスが向上し、ほとんどのログ・キャッシュのパフォーマンスも向上することがあります。『パフォーマンス＆チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」を参照してください。

リラックス置換方式を設定するには、次のコマンドを使用します。

```
sp_cacheconfig pubs_log, relaxed
```

デフォルト値は “strict” です。キャッシュ置換方式と非同期プリフェッチ率は省略可能です。指定する場合は、正しいパラメータまたは “DEFAULT” を指定してください。

**注意** キャッシュ置換方式の設定は静的であるため、変更を反映するには Adaptive Server を再起動する必要があります。

キャッシュの作成、およびキャッシュ・タイプと置換方式の指定を 1 つのコマンドで実行できます。次の例では、pubs_log と pubs_cache の 2 つのキャッシュが作成されます。

```
sp_cacheconfig pubs_log, "3M", logonly, relaxed
sp_cacheconfig pubs_cache, "10M", mixed, strict
```

結果は次のようになります。

```
sp_cacheconfig
Cache Name                     Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     42.29 Mb
pubs_cache               Active    Mixed        10.00 Mb     10.00 Mb
pubs_log                 Active    Log Only      7.00 Mb      7.00 Mb
                                             ------------ ------------
                                   Total      42.00 Mb     59.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb     10.00 Mb     10
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 7.00 Mb,   Run Size: 7.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   1432 Kb      0.00 Mb      7.00 Mb          10
```

# メモリ・プールへのデータ・キャッシュ分割

データ・キャッシュを作成した後で、それぞれ異なる I/O サイズの複数のメモリ・プールにキャッシュを分割できます。1 つのキャッシュの中で、同じ I/O サイズのプールを 2 つ以上持つことはできません。キャッシュ内のプールの合計サイズは、そのキャッシュのサイズよりも大きくすることはできません。メモリ・プールの最小サイズは、サーバの論理ページ・サイズです。メモリ・プールのサイズをこれより大きくする場合は、サイズは 2 の累乗でなければなりません。最大サイズは 1 エクステントです。

Adaptive Server が大量の I/O を実行するときは、複数のページが同時にキャッシュに読み込まれます。これらのページは、常に 1 つの単位として扱われます。つまり、その単位でキャッシュ内に保持され、ディスクに書き込まれます。

デフォルトでは、名前付きのデータ・キャッシュを作成すると、その領域すべてがデフォルトのメモリ・プールに割り当てられます。プールを追加作成すると、その領域の一部が他のプールに再度割り当てられ、デフォルトのメモリ・プールのサイズが縮小されます。たとえば、50MB の領域を持つデータ・キャッシュを作成すると、この領域はすべて 2K プールに割り付けられます。このキャッシュ内に 30MB の領域を持つ 4K プールを設定すると、2K プールは 20MB に縮小します。

複数のプールを作成した後で、プール間で領域を移すことができます。たとえば、20MB の 2K プールと 30MB の 4K プールがあるキャッシュでは、4K プールから 10MB の領域を引き出して 16K プールを作成できます。

キャッシュ内のプール間で領域を移動するコマンドを実行しても、Adaptive Server を再起動する必要はありません。したがって、サーバのアクティビティに影響を与えることなく、アプリケーションの負荷の変化に適応するようにプールの設定を変更できます。

自分で設定したキャッシュ内にプールを作成するだけでなく、デフォルト・データ・キャッシュ内に最大 16K の I/O 用メモリ・プールを追加することもできます。

メモリ・プールを設定するための構文は次のとおりです。

```
sp_poolconfig cache_name, "memsize[P|K|M|G]", "config_poolK"
[, "affected_poolK"]
```

config_pool は、コマンドで指定したサイズに設定されます。領域は、もう 1 つのプール affected_pool との間で移動します。affected_pool を指定しない場合、領域は 2K プールから取得されるか、2K プールに割り付けられます (使用可能な最小のサイズ)。プールの最小サイズは 512K です。

次の例では、データ・キャッシュ pubs_cache 内に 7MB の 16K ページ・プールを作成します。

```
sp_poolconfig pubs_cache, "7M", "16K"
```

現在の設定を確認するには、次のようにキャッシュ名だけを指定して sp_cacheconfig を実行します。

```
sp_cacheconfig pubs_cache

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_cache                Active    Mixed        10.00 Mb     10.00 Mb
                                                ------------ ------------
                                    Total        10.00 Mb     10.00 Mb
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 10.00 Mb,   Run Size: 10.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2048 Kb      0.00 Mb      3.00 Mb     10

   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
```

デフォルト・データ・キャッシュ内にメモリ・プールを作成することもできます。

たとえば、以下のように設定されているキャッシュを使用します。

```
Cache Name                 Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     42.29 Mb
                                                ------------ ------------
                                      Total      25.00 Mb     42.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:             1,   Run Partition:             1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     42.29 Mb     10
```

次に、8MB のデフォルト・データ・キャッシュ内に 16K のプールを作成するには、次のコマンドを使用します。

```
sp_poolconfig "default data cache", "8M", "16K"
```

このコマンドを実行すると、2K プールの “Run Size” は次のように設定されます。

```
Cache Name                 Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     42.29 Mb
                                                ------------ ------------
                                     Total      25.00 Mb     42.29 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 42.29 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   8662 Kb      0.00 Mb     34.29 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

作成するキャッシュ内で、2K メモリ・プールのサイズを設定する必要はありません。2K プールの Run Size カラムは、キャッシュ内の他のプールに対して明示的に設定されていないすべてのメモリを表します。

## ログ・キャッシュとログ I/O サイズの一致

データベースのトランザクション・ログ用のキャッシュを作成する場合は、そのキャッシュ内のほとんどの領域を、ログ I/O サイズと一致するように設定します。デフォルト値は、サーバの論理ページ・サイズの 2 倍です (サーバの論理ページ・サイズが 2K の場合は 4K、論理ページ・サイズが 4K の場合は 8K)。4K プールを使用できない場合は、ログには 2K の I/O が使用されます。ログ I/O サイズを変更するには、sp_logiosize を使用します。各データベースのログ I/O サイズは、Adaptive Server の起動時にエラー・ログに出力されます。sp_logiosize をパラメータなしで発行して、データベースのサイズを調べることもできます。

次の例は、pubs_log キャッシュ内に 4K プールを作成します。

```
sp_poolconfig pubs_log, "3M", "4K"
```

他のキャッシュにバインドされていないデータベースのトランザクション・ログに使用するために、デフォルト・データ・キャッシュに 4K メモリ・プールを作成することもできます。

```
sp_poolconfig "default data cache", "2.5M", "4K"
```

『パフォーマンス＆チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」の「トランザクション・ログの I/O サイズの選択」を参照してください。

# オブジェクトのキャッシュへのバインド

sp_bindcache は、データベース、テーブル、インデックス、テキスト・オブジェクト、イメージ・オブジェクトをキャッシュに割り当てます。エンティティをキャッシュにバインドするには、次の条件を満たす必要があります。

- 名前付きキャッシュが存在していて、そのステータスが “Active” であること。
- データベースまたはデータベース・オブジェクトが存在すること。
- テーブル、インデックス、オブジェクトをバインドする場合は、実行するユーザが使用しているデータベースにそのオブジェクトが保管されていること。
- トランザクション・ログ・テーブル syslogs などのシステム・テーブルをバインドする場合は、データベースがシングルユーザ・モードであること。
- データベースをバインドする場合は、実行するユーザが使用しているデータベースが master であること。
- データベース、ユーザ・テーブル、インデックス、テキスト・オブジェクト、イメージ・オブジェクトをキャッシュにバインドする場合は、キャッシュのタイプが Mixedであること。Log Onlyタイプのキャッシュにバインドできるのは、syslogs テーブルのみです。
- 実行するユーザがオブジェクトの所有者、データベース所有者、またはシステム管理者であること。

キャッシュへのオブジェクトのバインドは動的であり、サーバを再起動する必要はありません。

オブジェクトをキャッシュにバインドするための構文は次のとおりです。

```
sp_bindcache cache_name, dbname [,[owner.]tablename
[, indexname | "text only"] ]
```

次の例では、titles テーブルを pubs_cache にバインドします。

```
sp_bindcache pubs_cache, pubs2, titles
```

titles のインデックスをバインドするには、次のように 3 番目のパラメータとしてインデックス名を追加します。

```
sp_bindcache pubs_cache, pubs2, titles, titleind
```

上記の例では、pubs2 データベース内のオブジェクトの所有者が “dbo” であるため、所有者名を指定する必要はありません。他のユーザが所有しているテーブルを指定するには、その所有者名を追加します。ピリオドは特殊文字なので、次のようにパラメータ全体を引用符で囲んでください。

```
sp_bindcache pubs_cache, pubs2, "fred.sales_east"
```

次の例では、トランザクション・ログである syslogs を pubs_log キャッシュにバインドします。

```
sp_bindcache pubs_log, pubs2, syslogs
```

トランザクション・ログ syslogs などのシステム・テーブルをキャッシュにバインドするには、データベースはシングルユーザ・モードでなければなりません。次のように、master データベースから sp_dboption と use database コマンドを実行し、次に checkpoint を実行します。

```
sp_dboption pubs2, single, true
```

テーブルの text カラムと image カラムは、データベース内では別のデータ構造体に保管されます。このオブジェクトをキャッシュにバインドするには、次のように “text only” パラメータを追加します。

```
sp_bindcache pubs_cache, pubs2, au_pix, "text only"
```

次の例を master から実行すると、tempdb データベースがキャッシュにバインドされます。

```
sp_bindcache tempdb_cache, tempdb
```

既存のバインドを解除しなくても、オブジェクトを再バインドすることができます。

## キャッシュのバインドの制限

次の場合は、データベース・オブジェクトのバインドやバインド解除はできません。

- そのオブジェクトに対するダーティ・リードがアクティブである。
- そのオブジェクトに対するカーソルがオープンされている。

さらに、バインドまたはバインド解除の実行中、Adaptive Server はオブジェクトをロックする必要があり、プロシージャはロックが解放されるのを待つため、プロシージャの応答時間が遅くなることがあります。「バインドを実行するためのロック」(108 ページ) を参照してください。

# キャッシュのバインド情報の取得

sp_helpcache は、次のようにキャッシュ名を指定して実行すると、そのキャッシュと、キャッシュにバインドされているエンティティに関する情報を表示します。

```
sp_helpcache pubs_cache

Cache Name            Config Size     Run Size       Overhead
--------------------- -------------   ----------     ----------
pubs_cache             10.00 Mb       10.00 Mb        0.77 Mb

------------------ Cache Binding Information: ------------------

Cache Name      Entity Name            Type   Index Name  Status
----------      -----------            ----   ----------  ------
pubs_cache      pubs2.dbo.titles       index  titleind    V
pubs_cache      pubs2.dbo.au_pix       index  tau_pix     V
pubs_cache      pubs2.dbo.titles       table              V
pubs_cache      pubs2.fred.sales_east  table              V
```

キャッシュ名を指定しないで sp_helpcache を実行した場合は、Adaptive Server に設定されているすべてのキャッシュと、キャッシュにバインドされているすべてのオブジェクトについての情報が表示されます。

sp_helpcache は、キャッシュ名に関して文字列の一致を実行するときは、*%cachename%* を使用します。たとえば、“pubs” は “pubs_cache” と “pubs_log” の両方に一致します。

"Status” カラムには、キャッシュ・バインドが有効 (V) か無効 (I) かが表示されます。データベースまたはオブジェクトをキャッシュにバインドした後で、そのキャッシュを削除した場合は、バインド情報はシステム・テーブル内に残りますが、キャッシュ・バインドは無効 (I) と表示されます。オブジェクトのバインドが無効な場合は、そのオブジェクトにはデフォルト・データ・キャッシュが使用されます。その後で、同じ名前を持つ別のキャッシュを作成した場合は、そのキャッシュがアクティブになったときにバインドが有効になります。すべてのユーザ定義キャッシュは、ステータスが “mixed cache” または “log only” でなければなりません。

## キャッシュのオーバヘッドの検査

sp_helpcache を使用すると、指定したサイズの名前付きデータ・キャッシュを管理するために必要なオーバヘッドの量を表示できます。名前付きデータ・キャッシュを作成すると、sp_cacheconfig で要求した領域全体がキャッシュ領域として使用可能になります。キャッシュの管理に必要なメモリは、グローバル・メモリ・ブロック・プールから取得されます。

---

**注意** Adaptive Server バージョン 12.5.1 以降では、以前のバージョンと比べてキャッシュ・オーバヘッドが明確にわかるようになりました。オーバヘッドの量は同じですが、サーバのオーバヘッドの一部ではなく、キャッシュ・オーバヘッドの一部であると見なされ、レポートされるようになっています。

---

キャッシュに必要なオーバヘッドを調べるには、計画しているサイズを指定します。単位として、P (ページ)、K (キロバイト)、M (メガバイト)、G (ギガバイト) を使用します。次の例は 20,000 ページに対するオーバヘッドを検査します。

```
sp_helpcache "20000P"

2.96Mb of overhead memory will be needed to manage a cache of
size 20000P
```

ユーザ・キャッシュを設定しても、キャッシュ領域は浪費されません。1 つの大きなデータ・キャッシュを使用するか、多数の小さなキャッシュを使用するかに関係なく、メモリ内にページを保管して管理するための構造体としてメモリの約 5 パーセントが必要です。

## 合計キャッシュ領域へのオーバヘッドの影響

「データ・キャッシュに関する情報」(79 ページ) の例では、ユーザ定義のキャッシュを作成する前の状態で使用可能なキャッシュ領域として、59.44MB のデフォルト・データ・キャッシュが表示されています。サーバは 2K の論理ページを使用しています。10MB の pubs_cache を作成した後で sp_cacheconfig を実行すると、合計キャッシュ・サイズが 59.44MB であると表示されます。

データ・キャッシュを設定すると、使用可能なキャッシュの総量が増えたり減ったりしているように見えることがあります。これは、特定のサイズのキャッシュを管理するために必要なオーバヘッド量があるにもかかわらず、sp_cacheconfig によって表示される値にはそのオーバヘッドが含まれていないからです。

sp_helpcache を使用して、最初の 59.44MB のデフォルト・キャッシュと新しい 10MB のキャッシュのオーバヘッドを検査すると、領域の変化はオーバヘッドのサイズが変化したためであることがわかります。次の例は、変更前のデフォルト・データ・キャッシュに対するオーバヘッドを表示します。

```
sp_helpcache "59.44M"

4.10Mb of overhead memory will be needed to manage a cache of
size 59.44M
```

次の例は、pubs_cache に対するオーバヘッドを表示します。

```
sp_helpcache "10M"

0.73Mb of overhead memory will be needed to manage a cache of
size 10M
```

次の例は、49.44MB のキャッシュ・サイズに対するオーバヘッドを表示します。

```
sp_helpcache "49.44M"
3.46Mb of overhead memory will be needed to manage a cache of
size 49.44M
```

サイズが 10MB と 49.44MB の 2 つのキャッシュを管理するために必要なオーバーヘッド・サイズは 4.19MB (0.73MB + 3.46MB に等しい) で、これは 59.44MB の 1 つのキャッシュを管理するために必要なオーバーヘッド 4.10MB よりわずかに大きい値です。

キャッシュ・サイズは sp_cacheconfig システム・プロシージャで表示されるときに小数点以下第 2 位で丸められ、オーバヘッドは sp_helpcache システム・プロシージャで表示されるときに小数点以下第 2 位で丸められるので、出力には丸めによる多少の差異があります。

# キャッシュ・バインドの解除

キャッシュのバインドを解除するには、次のコマンドを使用します。

- キャッシュから 1 つのエンティティのバインドを解除する場合は sp_unbindcache。
- キャッシュにバインドされているすべてのオブジェクトのバインドを解除する場合は sp_unbindcache_all。

sp_unbindcache の構文は次のとおりです。

```
sp_unbindcache dbname [,[owner.]tablename
[, indexname | "text only"] ]
```

次の例は、pubs2 データベースにある titles テーブルの titleidind インデックスのバインドを解除します。

```
sp_unbindcache pubs2, titles, titleidind
```

キャッシュにバインドされているすべてのオブジェクトのバインドを解除するには、キャッシュ名を指定して sp_unbindcache_all を実行します。

```
sp_unbindcache_all pubs_cache
```

**注意** 8 つのデータベースの 9 つ以上のデータベース・オブジェクトがキャッシュにバインドされている場合は、sp_unbindcache_all を使用することはできません。個々のデータベースまたはオブジェクトに対して sp_unbindcache を実行して、関係するデータベースの数を 8 つ以下に減らしてください。

オブジェクトに対するキャッシュ・バインドを解除すると、その時点でメモリ内にあるすべてのページがそのキャッシュからクリアされます。

## メモリ・プールのウォッシュ・エリアの変更

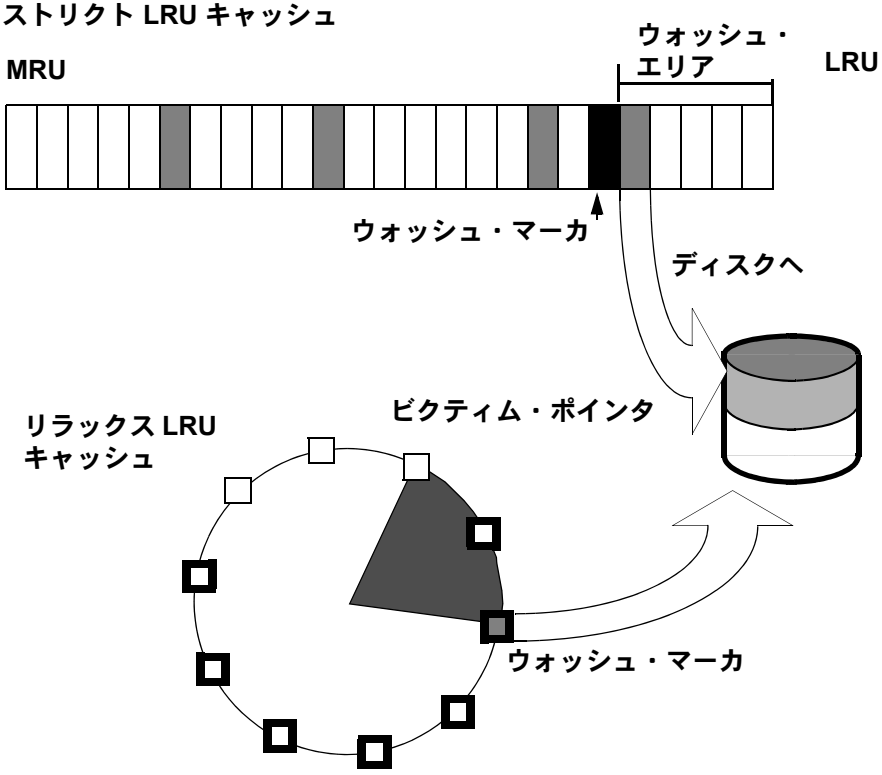
Adaptive Server は、バッファをキャッシュに読み込む場合、次のいずれかを行います。

- ストリクト LRU (least recently used、最も長い間使用されていない) 方式のキャッシュでは、各メモリ・プールの LRU 側の終端にバッファを配置します。
- リラックス LRU 方式のキャッシュでは、ビクティム・ポインタの位置にバッファを配置します。ビクティム・マーカの位置のバッファに「最近使用されたことを示す」ビットが設定されている場合は、ビクティム・ポインタはプール内の次のバッファに移動します。

各プールの一部分が、**ウォッシュ・エリア**として設定されます。ダーティ・ページ ( キャッシュ内の変更されたページ ) がウォッシュ・マーカを通過してウォッシュ・エリアに入ると、Adaptive Server はそのページについて非同期 I/O を開始します。書き込みが完了すると、そのページの状態はクリーンとなり、引き続きキャッシュ内で使用可能になります。

ウォッシュ・エリアの領域は、ページの置き換えが必要となる前にバッファの I/O を完了できるように十分な大きさが必要です。図 4-2 は、バッファ・プールのウォッシュ・エリアがストリクト LRU キャッシュとリラックス LRU キャッシュでどのように機能するかを示します。

図 4-2: バッファ・プールのウォッシュ・エリア




デフォルトでは、メモリ・プールのウォッシュ・エリアのサイズは次のように設定されます。

- プール・サイズが 300MB より小さい場合、デフォルトのウォッシュ・サイズはプール内のバッファ数の 20 パーセント。
- プール・サイズが 300MB より大きい場合、デフォルトのウォッシュ・サイズは 300MB でのバッファ数の 20 パーセント。

ウォッシュ・エリアのサイズの最小値は 10 バッファ分です。ウォッシュ・エリアの最大サイズは、プール・サイズの 80 パーセントです。2KB よりも大きなすべてのプールに対しては、プール・サイズとウォッシュ・サイズを指定してください。

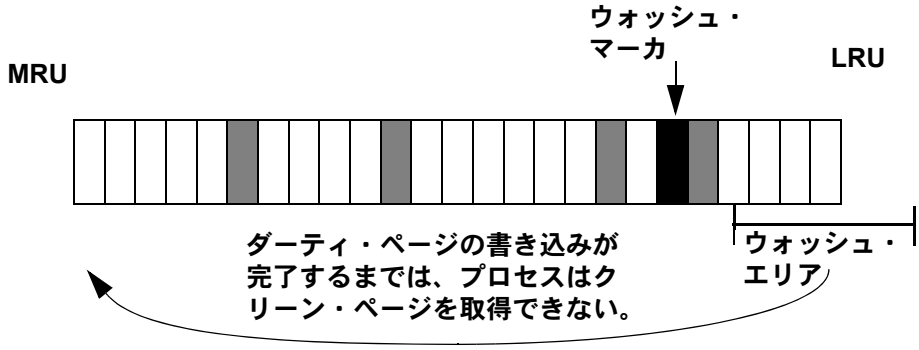
バッファとは複数のページから成るブロックで、そのプールの I/O サイズに一致します。各バッファは、常に 1 つの単位として扱われます。バッファ内のすべてのページは、その単位でキャッシュ内に読み込まれ、ディスクに書き込まれ、保持されます。ブロックのサイズを求めるには、バッファ数にプールのサイズを掛けます (2KB プール × 256 バッファ = 512KB、16KB プール × 256 バッファ = 4096KB)。

たとえば、1MB の領域を持つ 16K プールを設定した場合は、このプールのバッファ数は 64 で、その 20 パーセントは 12.8 となります。この値は 12 バッファに切り捨てられ、つまり 192K がウォッシュ・エリアに割り付けられたブロックのサイズになります。

## ウォッシュ・エリアが小さすぎる場合

バッファ・プールのウォッシュ・エリアが小さすぎると、クリーン・バッファを必要とするオペレーションは、プールの LRU 側終端またはビクティム・マーカの位置にあるダーティ・バッファの I/O が完了するまで待機しなければならないことがあります。これは「ダーティ・バッファ・グラブ」と呼ばれ、パフォーマンスの大幅な低下の原因となることがあります。図 4-3 は、ストリクト置換方式キャッシュのダーティ・バッファ・グラブを示します。

**図 4-3: ウォッシュ・エリアが小さすぎる結果発生するダーティ・バッファ・グラブ**




メモリ・プール内でダーティ・バッファ・グラブが発生しているかどうかを判断するには、sp_sysmon を使用します。通常、大量のダーティ・ページがあり、キャッシュ置換率が高いときにダーティ・バッファ・グラブが発生するので、sp_sysmon は、キャッシュの I/O と更新アクティビティが大量に発生しているときに実行します。

キャッシュ・サマリ・セクションの “Buffers Grabbed Dirty” 出力の “Count” カラムの値が 0 以外の場合は、各プールの “Grabbed Dirty” ローを確認して、どこに問題があるかを判断します。該当するプールのウォッシュ・エリアのサイズを増やします。次の例は、8K メモリ・プールのウォッシュ・エリアを 720K に設定します。

```
sp_poolconfig pubs_cache, "8K", "wash=720K"
```

プール・サイズが小さすぎる場合、特に sp_sysmon の出力からプールのターンオーバ率が高いことが判明した場合は、プール・サイズも拡大します。

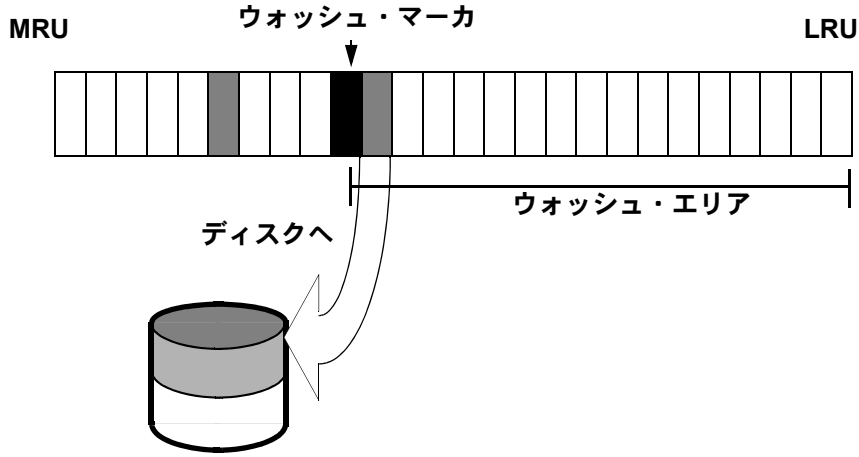
『パフォーマンス&チューニング・シリーズ：sp_sysmon による Adaptive Server の監視』を参照してください。

## ウォッシュ・エリアが大きすぎる場合

プールのウォッシュ・エリアが大きすぎると、図 4-4 に示すように、バッファがキャッシュ内のウォッシュ・マーカを通過するのが速すぎて、ダーティ・バッファの非同期書き込みが開始します。そのバッファの状態はクリーンとなり、LRU に達するまで MRU/LRU チェーンのウォッシュ・エリアにとどまります。別のクエリによってそのバッファ内のページが変更される場合は、そのバッファを再度ディスクに書き込むための追加の I/O が必要になります。

sp_sysmon の出力から、ストリクト置換方式キャッシュのウォッシュ・エリア内でバッファが見つかる確率 (Found in Wash) が高く、ダーティ・バッファ・グラブの問題がないことが判明している場合は、ウォッシュ・エリアのサイズを小さくしてみます。『パフォーマンス&チューニング・シリーズ：sp_sysmon による Adaptive Server の監視』を参照してください。

**図 *4-4:* ウォッシュ・サイズが大きすぎる場合の影響**



### キャッシュに対するウォッシュを防止するためのハウスキーピングの設定

cache status パラメータで HK ignore cache オプションを使用すると、ハウスキーピングが特定のキャッシュに対してウォッシュを実行しないように指定できます。ウォッシュに含めないキャッシュを指定することで、ハウスキーピングとキャッシュ・マネージャのスピンロックの間の競合を防止できます。HK ignore cache は、設定ファイルのキャッシュの各見出しの下に、手動で設定します。sp_cacheconfig を使用して HK ignore cache を設定することはできません。

次の設定ファイルの例は、名前付きキャッシュ newcache に対する設定です。

```
Named Cache:newcache
     cache size = 5M
     cache status = mixed cache
     cache status = HK ignore cache
     cache replacement policy = DEFAULT
local cache partition number = DEFAULT
```

HK ignore cache を設定する場合、default data cache、mixed cache、または log parameters のいずれかも設定してください。cache status パラメータに HK ignore cache だけを設定することはできません。

## プールの非同期プリフェッチ制限値の変更

非同期プリフェッチ制限値は、プールの領域のうち、非同期プリフェッチによってキャッシュに読み込まれたけれども一度もクエリで使用されていないページを保持できる割合を指定します。サーバのデフォルト値を設定するには、global async prefetch limit パラメータを使用します。特定のプールについて sp_poolconfig で設定された制限値は、デフォルトの制限値よりも優先されます。

次のコマンドは、pubs_cache 内の 2K メモリ・プールの割合を 20 に設定します。

```
sp_poolconfig pubs_cache, "2K", "local async prefetch limit=20"
```

プールのプリフェッチ制限値の変更は即座に反映されるので、Adaptive Server を再起動する必要はありません。『パフォーマンス＆チューニング・シリーズ：基本』の「第 6 章 非同期プリフェッチのチューニング」を参照してください。

## メモリ・プールのサイズの変更

sp_poolconfig を使用してメモリ・プールのサイズを変更し、キャッシュ、プールの新しいサイズ、変更するプールの I/O サイズ、バッファを取得するプールの I/O サイズを指定します。最後のパラメータを指定しなければ、すべての領域が論理ページ・サイズのプールから引き出されるか、またはそのプールに割り当てられます。

## メモリ・プールからの領域移動

pubs_log キャッシュの現在の設定を確認するには、sp_cacheconfig を使用します (次の出力は、前の項の例に基づいたものです)。

```
sp_cacheconfig pubs_log

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                                ------------ ------------
                                      Total       6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      3.50 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
```

4K プールのサイズを 5MB, に増やし、必要な領域を 2K プールから移動するには、次のように入力します。

```
sp_poolconfig pubs_log, "5M", "4K"

sp_cacheconfig pubs_log

Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
pubs_log                  Active    Log Only      6.00 Mb      6.00 Mb
                                                ------------ ------------
                                      Total       6.00 Mb      6.00 Mb
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
```

## 他のメモリ・プールからの領域移動

他のプールから領域を移動するには、キャッシュ名および「移動先」の I/O サイズと「移動元」の I/O サイズを指定します。デフォルト・データ・キャッシュの現在の設定が、次のように表示されているとします。

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                               ------------ ------------
                                    Total        25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      2.50 Mb      2.50 Mb     10
   16 Kb   1632 Kb      8.00 Mb      8.00 Mb     10
```

4K プールのサイズを 2.5MB から 4MB に増やして、16K プールから領域を引き出すには、次のコマンドを使用します。

```
sp_poolconfig "default data cache","4M", "4K","16K"
```

結果は次のような設定になります。

```
Cache Name                Status    Type     Config Value Run Value
------------------------- --------- -------- ------------ ------------
default data cache        Active    Default      25.00 Mb     29.28 Mb
                                               ------------ ------------
                                    Total        25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      0.00 Mb     18.78 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
   16 Kb   1632 Kb      6.50 Mb      6.50 Mb     10
```

キャッシュ内のプール間でバッファを移動するときに、移動できるのは空きバッファだけです。使用中のバッファや、ディスクに書き込まれていない変更が含まれているバッファは移動できません。

要求された量のバッファを移動できない場合は、要求されたサイズと変更後のメモリ・プールのサイズを示す情報メッセージが表示されます。

# キャッシュ・パーティションの追加

マルチエンジン・サーバでは、同時に複数のタスクが同じキャッシュにアクセスを試みることがあります。デフォルトでは、それぞれのキャッシュにシングル・スピンロックがあるため、そのキャッシュに対して変更またはアクセスできるのは一度に 1 つのタスクだけです。キャッシュ・スピンロックの競合が 10 パーセントを上回る場合は、キャッシュ・パーティションの数を増やすことによってスピンロックの競合が減り、パフォーマンスが向上する場合があります。

キャッシュのパーティション数は次のように設定します。

- すべてのデータ・キャッシュのパーティション数を、global cache partition number 設定パラメータを使用して設定します。
- 個々のキャッシュのパーティション数を、sp_cacheconfig を使用して設定します。

キャッシュ内のパーティション数は、必ず 2 の累乗で、範囲は 1 ～ 64 です。キャッシュ・パーティション内のプールのサイズを 512K より小さくすることはできません。個々のオブジェクトを格納するための要件を満たすようにキャッシュのサイズを調整できるので、ほとんどの場合は、スピンロックの競合が発生する特定のキャッシュにローカル設定を使用するようにしてください。

『パフォーマンス&チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」の「キャッシュ・パーティションによるスピンロック競合の低減」を参照してください。

## キャッシュ・パーティション数の設定

サーバ上のすべてのキャッシュに対するキャッシュ・パーティション数を設定するには、sp_configure を使用します。たとえば、キャッシュ・パーティションの数を 2 に設定するには、次のように入力します。

```
sp_configure "global cache partition number",2
```

変更内容を有効にするには、サーバを再起動してください。

## ローカル・キャッシュ・パーティション数の設定

ローカルのキャッシュ・パーティション数を設定するには、sp_cacheconfig または設定ファイルを使用します。次のコマンドは、デフォルト・データ・キャッシュのキャッシュ・パーティション数を 4 に設定します。

```
sp_cacheconfig "default data cache", "cache_partition=4"
```

変更内容を有効にするには、サーバを再起動してください。

### 優先度

ローカル・キャッシュ・パーティションの設定は、常に、グローバル・キャッシュ・パーティションの値よりも優先されます。

次の例は、サーバワイドなパーティション数を 4 に設定し、pubs_cache のパーティション数を 2 に設定します。

```
sp_configure "global cache partition number", 4
sp_cacheconfig "pubs_cache", "cache_partition=2"
```

ローカル・キャッシュ・パーティション数はグローバル・キャッシュ・パーティション数よりも優先されるため、pubs_cache が使用するパーティションは 2 つとなります。これ以外のすべての設定済みキャッシュは 4 つのパーティションを使用します。

pubs_cache のローカル設定を削除して、代わりにグローバル値を使用するには、次のコマンドを使用します。

```
sp_cacheconfig "pubs_cache", "cache_partition=default"
```

グローバル・キャッシュ・パーティション数をデフォルト値にリセットするには、次のコマンドを使用します。

```
sp_configure "global cache partition number", 0, "default"
```

## メモリ・プールの削除

プールを完全に削除するには、プール・サイズを 0 に設定します。次の例は、16K プールを削除して、すべての領域をデフォルトのプール内に配置します。

```
sp_poolconfig "default data cache", "0", "16K"
Cache Name               Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      25.00 Mb     29.28 Mb
                                              ------------ ------------
                                     Total      25.00 Mb     29.28 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 25.00 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
```

対象となるプール・サイズ ( 上記の例の 16K) を指定しない場合は、すべての領域がデフォルトのプール内に配置されます。キャッシュ内のデフォルトのプールは削除できません。

### ページが使用中であるためにプールが削除できない場合

削除しようとしているプールに、使用中のページが含まれている場合、またはダーティ・リードがありながらディスクに書き込まれていないページが含まれている場合は、可能なかぎりのページが指定のプールに移動され、残ったプールのサイズが情報メッセージとして表示されます。プール・サイズが最小許容プール・サイズよりも小さい場合は、プールが使用不可であることを示す警告メッセージが表示されます。これらの警告のいずれかが表示された後で sp_cacheconfig を実行すると、これらのプールの詳細情報セクションに Status カラムが追加され、そのプールの Status カラムに Unavailable/too small (使用不可／小さすぎる) または Unavailable/deleted (使用不可／削除済み) のいずれかが表示されます。

後で同じシステム・プロシージャを再発行すると、プールの削除を完了できます。Adaptive Server を再起動すると、“Unavailable/too small” または “Unavailable/deleted” 状態のプールも削除されます。

## メモリとクエリ・プランへのキャッシュ・バインドの影響

オブジェクトのバインドやバインド解除は、パフォーマンスに影響を及ぼすことがあります。テーブルまたはインデックスをバインドしたりバインドを解除したりすると、次のようになります。

- そのオブジェクトのページがキャッシュからフラッシュされます。
- バインドを実行するにはそのオブジェクトをロックする必要があります。
- プロシージャとトリガのクエリ・プランをすべて再コンパイルする必要があります。

### キャッシュからのページのフラッシュ

オブジェクトまたはデータベースをキャッシュにバインドすると、既にメモリ内にあるそのオブジェクトのページは、すべてソース・キャッシュから削除されます。そのページが次回クエリによって要求されたときは、ページは新しいキャッシュに読み込まれます。同様に、オブジェクトのバインドを解除すると、キャッシュ内のページはユーザ設定キャッシュから削除され、次回そのページがクエリによって要求されたときはデフォルト・キャッシュに読み込まれます。

### バインドを実行するためのロック

ユーザ・テーブル、インデックス、テキスト・オブジェクト、イメージ・オブジェクトをバインドまたはバインド解除するには、キャッシュ・バインド処理コマンドがそのオブジェクトでの排他テーブル・ロックを取得する必要があります。別のユーザがテーブルに対するロックを保持している場合に、そのオブジェクトに関して sp_bindcache、sp_unbindcache、sp_unbindcache_all を発行すると、システム・プロシージャは必要とするロックを取得できるまでスリープします。

データベース、システム・テーブル、システム・テーブルのインデックスの場合、そのデータベースはシングルユーザ・モードでなければなりません。したがって、他のユーザがそのオブジェクトに対するロックを保持していることはありません。

### ストアド・プロシージャとトリガへのキャッシュ・バインドの影響

キャッシュのバインドと I/O サイズは、ストアド・プロシージャとトリガのクエリ・プランの一部です。オブジェクトのキャッシュ・バインドを変更すると、そのオブジェクトを参照するすべてのストアド・プロシージャは、次回実行されるときに再コンパイルされます。データベースのキャッシュ・バインドを変更すると、そのデータベース内のオブジェクトのうち、キャッシュに明示的にバインドされていないものを参照するすべてのストアド・プロシージャは、次回実行されるときに再コンパイルされます。

## 設定ファイルによるデータ・キャッシュの設定

Adaptive Server の起動時に使用される設定ファイルを編集することによって、名前付きデータ・キャッシュを追加または削除したり、既存のキャッシュとそのメモリ・プールを再設定したりできます。

---

**注意** 稼働中のサーバのキャッシュとプールの設定は変更できません。既にサーバ上で設定されているものとは異なるキャッシュ設定とプール設定が格納されている設定ファイルを読み込もうとすると、その読み込みは失敗します。

---

## 設定ファイル内のキャッシュ・エントリとプール・エントリ

サーバ上の設定済みデータ・キャッシュごとに、次の情報ブロックが設定ファイル内にあります。

```
[Named Cache:cache_name]
     cache size = {size | DEFAULT}
     cache status = {mixed cache | log only | default data cache}
     cache replacement policy = {DEFAULT |
          relaxed LRU replacement| strict LRU replacement }
```

サイズは、次の単位によって指定できます。

- P － ページ (Adaptive Server ページ)
- K － キロバイト (デフォルト)
- M － メガバイト
- G － ギガバイト

次の例は、設定ファイル内のデフォルト・データ・キャッシュのエントリを示します。

```
[Named Cache:default data cache]
      cache size = DEFAULT
      cache status = default data cache
      cache replacement policy = strict LRU replacement
```

デフォルト・データ・キャッシュは、Adaptive Server の起動に必要な唯一のキャッシュ・エントリです。このエントリには、キャッシュ・サイズとキャッシュ・ステータスを割り当ててください。ステータスは “default data cache” でなければなりません。

キャッシュに設定済みプールがある場合は、上記の例のブロックの後に次のようなプールごとの情報ブロックが続きます。

```
    [16K I/O Buffer Pool]
            pool size = size
            wash size = size
            local async prefetch limit = DEFAULT
```

**注意** 場合によっては、キャッシュ内のプールのエントリが設定ファイルにないことがあります。sp_poolconfig によって非同期プリフェッチ率を変更すると、その変更内容はシステム・テーブルだけに書き込まれ、設定ファイルには書き込まれません。

次の例は、sp_cacheconfig からの出力です。その後に、このキャッシュとプールの設定に一致する設定ファイルのエントリを示します。

```
Cache Name                Status    Type     Config Value Run Value
------------------------ --------- -------- ------------ ------------
default data cache       Active    Default      29.28 Mb     25.00 Mb
pubs_cache               Active    Mixed        20.00 Mb     20.00 Mb
pubs_log                 Active    Log Only      6.00 Mb     6.00 Mb
tempdb_cache             Active    Mixed         4.00 Mb     4.00 Mb
                                                ------------ ------------
                                     Total      59.28 Mb     55.00 Mb
===========================================================================
Cache: default data cache,   Status: Active,   Type: Default
     Config Size: 29.28 Mb,   Run Size: 29.28 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   3844 Kb      6.50 Mb     25.28 Mb     10
    4 Kb    512 Kb      4.00 Mb      4.00 Mb     10
===========================================================================
Cache: pubs_cache,   Status: Active,   Type: Mixed
     Config Size: 20.00 Mb,   Run Size: 20.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb   2662 Kb      0.00 Mb     13.00 Mb     10
   16 Kb   1424 Kb      7.00 Mb      7.00 Mb     10
===========================================================================
Cache: pubs_log,   Status: Active,   Type: Log Only
     Config Size: 6.00 Mb,   Run Size: 6.00 Mb
     Config Replacement: relaxed LRU,   Run Replacement: relaxed LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    716 Kb      0.00 Mb      1.00 Mb     10
    4 Kb   1024 Kb      5.00 Mb      5.00 Mb     10
===========================================================================
Cache: tempdb_cache,   Status: Active,   Type: Mixed
     Config Size: 4.00 Mb,   Run Size: 4.00 Mb
     Config Replacement: strict LRU,   Run Replacement: strict LRU
     Config Partition:            1,   Run Partition:            1

IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
    2 Kb    818 Kb      0.00 Mb      4.00 Mb     10
```

このキャッシュとプールの設定に一致する設定ファイルの情報は次のとおりです。

```
[Named Cache:default data cache]
         cache size = 29.28M
         cache status = default data cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT

[2K I/O Buffer Pool]
        pool size = 6656.0000k
        wash size = 3844 K
        local async prefetch limit = DEFAULT

[4K I/O Buffer Pool]
        pool size = 4M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_cache]
        cache size = 20M
        cache status = mixed cache
        cache replacement policy = strict LRU replacement
        local cache partition number = DEFAULT

[16K I/O Buffer Pool]
        pool size = 7M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT

[Named Cache:pubs_log]
        cache size = 6M
        cache status = log only
        cache replacement policy = relaxed LRU replacement
        local cache partition number = DEFAULT

[4K I/O Buffer Pool]
        pool size = 5.0000M
        wash size = DEFAULT
        local async prefetch limit = DEFAULT


[Named Cache:tempdb_cache]
        cache size = 4M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = DEFAULT
```

『システム管理ガイド 第1巻』の「第5章 設定パラメータ」を参照してください。

**警告！** max memory 設定パラメータを確認し、Adaptive Server の他の部分の要件を満たす十分なメモリを割り付けてください。設定ファイル内でデータ・キャッシュに割り付けたメモリ量が大きすぎると、Adaptive Server は起動しません。この場合は、設定ファイルを編集してデータ・キャッシュ領域の量を減らすか、Adaptive Server に割り付けられる max memory の設定値を大きくしてください。『システム管理ガイド 第1巻』の「第5章 設定パラメータ」を参照してください。

## キャッシュ設定のガイドライン

ユーザが定義できるキャッシュを設定する場合は、次の一般的なガイドラインに従います。

- デフォルト・データ・キャッシュのサイズには、バインドされていないテーブルとインデックスに対するすべてのキャッシュ動作に対応できる十分な大きさを設定してください。キャッシュに明示的にバインドされていないオブジェクトはすべてデフォルト・キャッシュを使用します。これには、ユーザ・データベース内のバインドされていないシステム・テーブル、master データベース内のシステム・テーブル、キャッシュに明示的にバインドされていないその他のオブジェクトが該当します。
- リカバリ中にアクティブになるのは、デフォルト・キャッシュだけです。ロールバックまたはロールフォワードが必要なすべてのトランザクションは、デフォルト・データ・キャッシュ内にデータ・ページを読み込む必要があります。デフォルト・データ・キャッシュが小さすぎると、リカバリの時間が長くなります。
- キャッシュ内の 2K プールの領域不足が発生することがないようにしてください。多くのタイプのデータ・アクセスでは、大容量の I/O は必要としません。たとえば、インデックスを使用して 1 つのローをユーザに返すだけの簡単なクエリは、4 つか 5 つの 2K の I/O を使用し、16K の I/O からは何も取得しません。
- 特定の dbcc コマンドおよび drop table では、2K の I/O しか実行できません。dbcc checktable では大容量 I/O を実行でき、dbcc checkdb は、テーブルに対しては大容量 I/O を実行できますが、インデックスに対しては 2K の I/O を実行します。
- トランザクション・ログによって使用されるキャッシュの場合は、デフォルト・ログ I/O サイズに適応する I/O プールを設定します。このサイズは、sp_logiosize を使用してデータベースごとに設定します。デフォルト値は 4K です。

- すべてのインデックスとオブジェクト、およびそれらのキャッシュ処理を管理しようとすると、キャッシュ領域を浪費することになります。作成したキャッシュやプールが、バインドされているテーブルまたはインデックスによって適切に使用されていなければ、キャッシュやプールの領域が無駄になり、他のキャッシュでの余分な I/O が増えることになります。
- アプリケーションで tempdb を使用する頻度が高い場合は、tempdb を専用のキャッシュにバインドしてください。tempdb データベースの全体をバインドすることだけはできますが、tempdb の個々のオブジェクトをバインドすることはできません。
- 更新が多く、置き換え率も高いキャッシュの場合は、必ずウォッシュ・サイズを十分な大きさにしてください。
- マルチ CPU システムでは、スピンロックの競合を避けるために、最も頻繁に使用されるテーブルとそのインデックスを複数のキャッシュに分配します。
- 変化する作業負荷に適応するように、キャッシュまたはキャッシュ内のメモリ・プールの再設定を検討してください。キャッシュの設定を変更すると、サーバの再起動が必要になります。ただし、メモリ・プールの設定を変更した場合、再起動は必要ありません。

  たとえば、1 か月の大半で OLTP (オンライン・トランザクション処理) を実行し、負荷の大きな DSS (意思決定支援システム) アクティビティを数日間だけ実行するシステムの場合は、多量の DSS アクティビティ用に 2K プールから 16K プールへの領域の移動を行い、DSS の作業負荷がなくなったときにプールのサイズを OLTP 用に再設定します。

## 設定ファイルのエラー

設定ファイルを手動で編集する場合は、キャッシュ・サイズ、プール・サイズ、ウォッシュ・サイズを十分に確認してください。全キャッシュ・サイズの合計が、max memory の値から Adaptive Server の他の部分に必要なメモリ量を差し引いた値を超えることはできません。

ほとんどの場合、エントリがないという問題があるときは、サイズ、ステータス、またはその他の情報が省略されたエントリの直後の行に「フォーマットが正しくありません」というエラーがレポートされます。他のエラーでは、エラーが発生したキャッシュの名前と、エラーのタイプが示されます。たとえば、プールのウォッシュ・サイズの指定が正しくない場合は、次のようなエラーが表示されます。

```
The wash size for the 4k buffer pool in cache pubs_cache has
been incorrectly configured. It must be a minimum of 10 buffers
and a maximum of 80 percent of the number of buffers in the
pool.
```

第 5 章

# マルチプロセッサ・サーバの管理

Adaptive Server は、Sybase Virtual Server Architecture™ を実装しており、これによって SMP (対称型マルチプロセッシング) システムの並列処理機能を利用できるようになります。使用可能な CPU の数とサーバ・マシンに要求される処理量に応じて、Adaptive Server を 1 つのプロセス、1 つのマルチスレッド・プロセス、または複数の協調プロセスとして実行することもできます。この章では、次の内容について説明します。

- SMP Adaptive Server 用のターゲット・マシン・アーキテクチャ
- SMP 環境向けの Adaptive Server アーキテクチャ
- SMP 環境での Adaptive Server のタスク管理
- 複数のエンジンの管理

SMP システム向けのアプリケーション設計の詳細については、『パフォーマンス&チューニング・シリーズ：基本』の「第 3 章 エンジンと CPU の使用方法」を参照してください。



## Adaptive Server カーネル

Adaptive Server のバージョン 15.7 以降には、スレッド・カーネルおよびプロセス・カーネルの 2 つのカーネルが含まれます。Adaptive Server を設定したカーネルによって、Adaptive Server が実行するモードが決まります。

- スレッド・モード － Adaptive Server は 1 つのマルチスレッド・オペレーティング・システム・プロセスとして実行され、スレッド・プールのスレッドで実行されるエンジンにより SQL クエリが処理されます。スレッド・モードでは、エンジンを使用しないスレッドを活用して I/O を管理します。管理者は追加スレッド・プールを設定して負荷を管理できます。

- プロセス・モード − Adaptive Server が 1 つのサーバとして機能する複数のオペレーティング・システム・プロセスとして実行されます。プロセス・モードでは、エンジンを使用して I/O を管理し、管理者はエンジン・グループを設定して負荷を管理します。

**注意** プロセス・モードは Windows では使用できません。

スレッド・モードは、多くの負荷に対してプロセス・モードよりも格段に少ない CPU を使用し、同等以上のパフォーマンスを提供します。スレッド・モードには、タスクとエンジンとの結び付きがあまり必要ないため、I/O 集中利用負荷と CPU 集中利用負荷が混在する場合でも、より一定したパフォーマンスを発揮します。

スレッド・カーネルを使用すると、以前のバージョンのカーネルよりも多くのプロセッサ、プロセッサ・コア、ハードウェア・スレッドを持つ並列ハードウェアとサポート・システムを Adaptive Server で利用することができます。

バージョン 15.7 ではカーネルが変更されていますが、クエリ・プロセッサに変更はありません。スレッド・カーネル・モードを実行するには、以前のバージョンの Adaptive Server 用に作成したほとんどのスクリプトは変更の必要がありませんが、一部のコマンドとストアド・プロシージャが変更されています。アプリケーションはスレッド・モードと完全互換です。

# ターゲットのアーキテクチャ

SMP 製品は、次の機能を持つマシンを対象としています。

- 対称型マルチプロセッシング・オペレーティング・システム
- 共通バス上の共有メモリ
- 1 ～ 1024 プロセッサ、コア、またはハードウェア・スレッド
- マスタ・プロセッサなし
- 超高速スループット

Adaptive Server は、オペレーティング・システムによって物理 CPU 上にスケジュール調整されている 1 つまたは複数の協調プロセスで構成されています。

Adaptive Server は、プロセス・モードで実行中の場合、複数の協調プロセスを使用して並行ハードウェアを活用し、スレッド・モードで実行中の場合、同じプロセスから複数スレッドを使用します。プロセス・モード・カーネルの各プロセスは、Adaptive Server のエンジンです。スレッド・モード・カーネルは、いくつかのスレッドをエンジンとして使用し、追加の非エンジン・スレッドを持ちます。

プロセス・モードはマルチスレッド・プロセスを使用します。ただし、Adaptive Server では、各プロセスのメイン・スレッドで作業のほとんどを実行するため、CPU リソースの調査および調整の際、これらのプロセスはシングル・スレッドと見なされます。

Adaptive Server はエンジンをプロセッサとして使用し、SQL クエリを実行します。プロセス・モードでは、エンジンは各プロセスのメイン・スレッドです。スレッド・モードでは、エンジンは 1 つ以上のエンジン・スレッド・プールからのスレッドです。複数エンジンは共有メモリを使用して通信します。プロセスおよびスレッド・モードは、単一エンジンまたは単一プロセッサ環境を含む共有メモリを使用します。

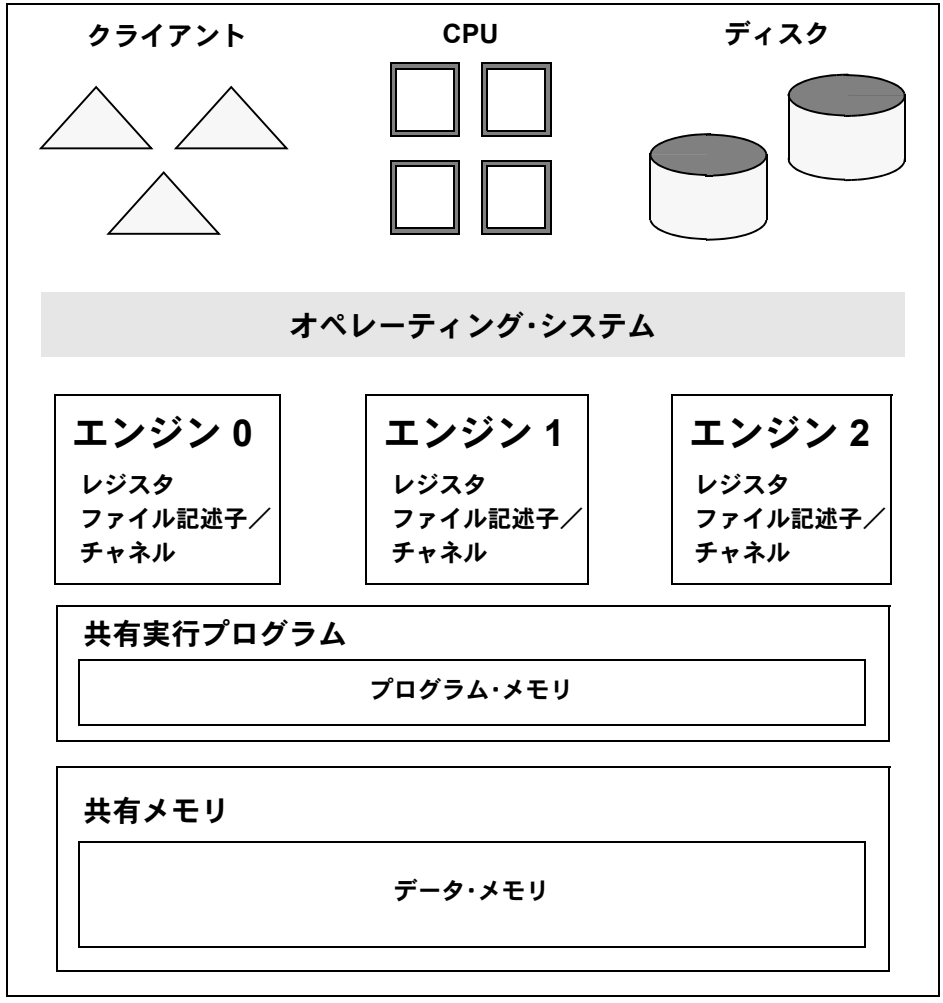
オペレーティング・システムは、CPU リソースに対して Adaptive Server スレッドをスケジュールします。Adaptive Server は物理プロセッサ、コア、またはサブコア・スレッドを区別しません。

カーネル・モード用に設定すると、Adaptive Server は単一オペレーティング・システム・プロセスのスレッド内のエンジンを実行します。Adaptive Server はエンジンをサポートするためにエンジン・スレッド・プールからスレッドを入手します。

Adaptive Server には、特定のタスクに使用される非エンジン・スレッドが含まれますが、エンジンとは見なされません。

図 5-1: スレッド・モード・アーキテクチャ

**図 5-2: プロセス・モード・アーキテクチャ**




オペレーティング・システムは、物理 CPU リソースに対して Adaptive Server スレッド (エンジンおよび非エンジン) をスケジュールします。リソースはプロセッサ、コア、サブコア・スレッドのいずれかです。Adaptive Server ではなく、オペレーティング・システムが、スレッドを CPU リソースに割り当てます。Adaptive Server パフォーマンスは、オペレーティング・システムからの CPU 時間によって異なります。

Adaptive Server エンジンは、更新およびロギングを含むすべてのデータベース機能を実行します。オペレーティング・システムではなく、Adaptive Server が、使用可能なエンジンに対してクライアント・タスクを動的にスケジュールします。タスクは、Adaptive Server 内の実行環境です。

「結び付き」は、特定の Adaptive Server タスクが特定のエンジン上だけで実行されるプロセス (タスクとの結び付き)、特定のエンジンが特定のタスク用のネットワーク I/O を処理するプロセス (ネットワーク I/O との結び付き)、または特定のエンジンが特定の CPU 上だけで実行されるプロセス (エンジンとの結び付き) です。

プロセス・モードでは、Adaptive Server への接続には接続を受け入れるエンジンとのネットワーク I/O の結び付きがあります。このエンジンは、その接続のすべてのネットワーク I/O を処理する必要があります。エンジンはどの接続に対してでも I/O を実行できるため、スレッド・モードにはネットワーク I/O との結び付きは存在しません。このため、コンテキストの切り替えが減少し、パフォーマンスが向上します。

マネージャがタスクを特定のエンジンのみで実行できるように、論理プロセス・マネージャを使用してタスクとの結び付きを確立できます。スレッド・モードでは、スレッド・プールを使用して、タスクとの結びつきを確立します。プロセス・モードでは、エンジン・グループを使用します。

スレッド・プールおよびエンジン・グループの動作は異なります。スレッド・プールのエンジンは、そのスレッド・プールのみに割り当てられたタスクを実行します。スケジューラの検索領域 (スケジューラが実行可能なタスクを検索する領域) はそのスレッド・プールのエンジンに制限されます。エンジン・グループのエンジンは、異なるグループのエンジンに制約されていない限り、どのタスクでも実行できます。つまり、タスクの実行場所が制限されているにもかかわらず、そのタスクのグループのエンジンを予約しないエンジン・グループのタスクも含みます。

Adaptive Server 論理プロセス・マネージャを使用して、タスクとの結び付きを設定します (『パフォーマンス＆チューニング・シリーズ：基本』の「第 4 章 エンジン・リソースの分配」を参照)。dbcc tune または同等のオペレーティング・システムのコマンドを使用して、エンジンまたは CPU との結び付きを設定します (『リファレンス・マニュアル：コマンド』およびオペレーティング・システムのマニュアルを参照)。

# カーネル・モード

デフォルトでは、Adaptive Server はスレッド・モードで起動され、オペレーティング・システムで単一プロセスとして表示され、並列処理のためにネイティブのオペレーティング・システム・スレッドを使用します。デフォルトでは、Adaptive Server はネイティブ・スレッドとの I/O を処理します。プロセス・モードでは、Adaptive Server はオペレーティング・システムに複数プロセスとして表示されます。

プロセス・モードでは、Adaptive Server の新しいバージョンで使用可能な機能でサポートされていないものもあります。Sybase ではスレッド・モードでの実行に問題がある場合のフォールバックとして、プロセス・モードが使用されます。

**注意** アーキテクチャ上の理由から、プロセス・モードは Windows プラットフォームでは使用できません。Adaptive Server の以前のバージョンも、Windows 上でマルチスレッドの単一プロセス・カーネルを使用しています。

Adaptive Server の現在のモードを確認するには、次を使用します。

```
select @@kernelmode
-------------------------------------
threaded
```

## カーネル・モードを切り替える

sp_configure “kernel mode” を使用して、カーネル・モードを切り替えます。変更を有効にするには、Adaptive Server を再起動します。モードを切り替えると、Adaptive Server は次のようなエラー・メッセージを返します。

```
sp_configure "kernel mode", 0, process
Parameter Name                 Default              Memory Used
        Config Value         Run Value            Unit
        Type
------------------------------ -------------------- -----------
        -------------------  -------------------  --------------------
        --------------------
kernel mode                      threaded             0
             process        threaded             not applicable
        static

(1 row affected)
Configuration option changed. Since the option is static, Adaptive Server must be rebooted
in order for the change to take effect. Changing the value of 'kernel mode' does not
increase the amount of memory Adaptive Server uses
```

次のことを考慮してください。

- スレッド・モードからプロセス・モードに切り替えると、Adaptive Server の起動時に設定ファイルのスレッド・プール情報が無視されます。
- プロセス・カーネル・モードからスレッド・モードに切り換えると、すべての実行クラスにはエンジン・グループがなくなり、syb_default_pool に関連付けられます。管理者は、実行クラスをエンジン・スレッド・プールに関連付けることができます。スレッド・モードからプロセス・モードに切り替えると、実行クラスにはスレッド・プールの関連付けがなくなり、既存の ANYENGINE エンジン・グループに関連付けられます。管理者は、実行クラスをエンジン・グループに関連付けることができます。スレッド・プール (またはエンジン・グループ) への実行クラスの再割り当ては、新しいログインおよびその実行クラスに関連付けられている既存の接続に適用されます。
- スレッド・モードの実行中は、エンジン・グループは廃止されます。このモードでは、sp_addexeclass を使用して実行クラスをユーザ・スレッド・プールにバインドします。スレッド・プールは、Adaptive Server 内のアプリケーション間のプロセッサ・リソースを分離します。

# タスク

タスクは Adaptive Server 内部の実行プログラムです (たとえば、ユーザ接続、ハウスキーピング・タスクなどのデーモン、I/O 処理などのカーネル・タスク)。Adaptive Server はタスクをスレッドに対してスケジュールします。これにより、CPU 時間が各タスクに割り当てられます。

Adaptive Server でのタスクには、次が含まれます。

- ユーザ・タスク ― ユーザ接続を表し、ユーザ・クエリを実行できます。isql 接続はユーザ・タスクの例です。
- サービス・タスク ― 特定のユーザに関連付けられていません。ユーザ・クエリを発行しません。サービス・タスクには、ハウスキーピング・タスク、チェックポイント、複写エージェント・スレッドなどが含まれます。
- システム・タスク ― サービス・タスクのカーネルレベル・バージョンです。システム・タスクには、I/O ハンドラ、クロック・ハンドラ、クラスタ・メンバシップ・サービス、IP リンク・モニタなどが含まれます。

Adaptive Server はスレッド全体でタスクを多重化します。実行完了 (RTC) スレッド・プールは、タスクをスレッド・プールに配置し、次に使用可能なスレッドがタスクを選択します。タスクは完了するまでスレッドに保持され、実行の終了時に削除されます。

個別のタスクにタスクの優先度を設定する方法については、『パフォーマンス＆チューニング・シリーズ：基本』の「第 4 章 エンジン・リソースの分配」の「基本優先度」を参照してください。

monTask モニタリング・テーブルには、Adaptive Server で実行中の各タスクのローが含まれます。『リファレンス・マニュアル：テーブル』を参照してください。

### スレッドを使用したタスクの実行

Adaptive Server はタスクをスレッド・プールに割り当てます。すべてのスレッド・プールにはスレッドがあります。各スレッド・プールにはタスクをスレッドに割り当てるスケジューラが含まれます。スケジューラは、多重化タスクを作業中のスレッドに割り当てたり、割り当てを解除します。スケジューラがいったん RTC タスクをスレッドに割り当てると、タスクは作業が完了するまでそのスレッドに留まります。

Adaptive Server には、システム作成のスレッド・プールおよびユーザ作成のスレッド・プールが含まれています。すべてのシステム定義スレッド・プールは、syb_ プレフィクスで始まります (たとえば、syb_default_pool)。ユーザ定義スレッド・プールの名前を syb_ プレフィクスで始めることはできません。オブジェクトの命名規則に従う必要があります (『リファレンス・マニュアル：ブロック』の「第 1 章 SQL ビルディング・ブロック」を参照)。デフォルトでは、Adaptive Server はユーザ・タスクを syb_default_pool スレッド・プールに関連付けます。sp_bindexeclass を使用して、ユーザ作成スレッド・プールに 1 つ以上のタスクを関連付けます。

## SMP 環境の設定

SMP 環境の設定方法はユニプロセッサの場合と似ていますが、通常は SMP マシンの方が強力で、はるかに多くのユーザを処理できます。SMP 環境では、エンジンの数を制御するための機能が追加されています。

## スレッド・プール

スレッド・プールは CPU リソースをグループ分けします。スレッド・プールには、関連付けられている Adaptive Server タスクの実行に使用されるスレッドが含まれています。スレッドは、ユーザ・タスクを実行するエンジンをホストし、特定のジョブ ( シグナル処理など ) を実行して、ワーク・キューからの要求を処理します。Adaptive Server には、システム定義スレッド・プール、およびユーザ作成スレッド・プール ( 存在する場合 ) が含まれています。

**注意** スレッド・プールは、Adaptive Server がスレッド・モードに設定されている場合にのみ使用できます。

エンジンが含まれているスレッド・プールはエンジン・プールと呼ばれ、カーネル・プロセス ID (KPID) を持つ Adaptive Server タスクを実行します。Adaptive Server は、エンジン・プールの各スレッドにエンジンを割り当てます。

Adaptive Server は、次の 2 種類のスレッドをサポートします。

- エンジン (または多重化) スレッド − データベース・クエリ・プロセスを実行し、複数のデータベース・プロセスの間で共有されます。多重化スレッドは、他のタスクと 1 つ以上のスレッドを共有するタスクを実行します。デフォルトでは、ユーザ・タスクおよびサービス・タスクは多重化です。定義されたタイム・スライスのコンシューム後、またはブロックされた場合に、多重化タスクを停止する必要があります。すべての多重化スレッドにはエンジンが割り当てられているため、プロセス・モードのエンジンと同等です。
- 実行完了 (RTC) スレッド − システム・タスクによって使用され、複数タスク間では共有されません。RTC スレッドは、タスクが完了するまで単一タスクを実行します。これは、Adapative Server によるスケジューリングの対象外です。

  RTC スレッドは、スレッドが完了するまでスケジュールから削除できないタスクを実行し、タイムスライスでは停止せずにブロックされている間スレッドに接続されたままです。すべての RTC スレッドが現在タスクを実行中の場合、RTC タスクは、スレッドが実行されるまで待機します。

スレッド・プールを作成する際、スレッドの種類を指定することはできません。ユーザ作成スレッド・プールは、常に多重化です。

Adaptive Server には、これらのシステム定義スレッド・プールが含まれます。

- syb_default_pool − デフォルトのエンジン・スレッド・プールです。syb_default_pool の各スレッドはエンジンです。すべてのユーザ・タスクおよびすべての多重化システム・タスク (ハウスキーピングなど) は syb_default_pool で実行されます。ただし、追加のスレッド・プールを作成することで、一部のタスクを syb_default_pool から移動することができます。
- syb_system_pool − システム・スレッドに使用される RTC スレッド・プールです。syb_system_pool の各スレッドは、特定タスクの実行に指定されています。syb_system_pool には、システム・クロック用のスレッドが最低 1 つ、および他の非同期シグナルが含まれています。すべての I/O 処理スレッドは、syb_system_pool で実行されます。
- syb_blocking_pool − Adaptive Server が多重化タスクからの呼び出し要求のブロックを処理するために使用する RTC プールで、通常は、多重化スレッドまたはエンジン・スレッドが許容不可の時間量をブロックする原因となるオペレーティング・システムの呼び出しです。syb_blocking_pool のスレッドは、通常、CPU リソースのほんのわずかしか消費しません。

スレッド・プールは属性によって定義されます。そのうちの一部は Adaptive Server によって自動的に割り当てられ、その他はスレッド・プールの作成時に決定されます。スレッド・プールの属性は次のとおりです。

- ID − スレッド・プールのシステム割り当て ID。Adaptive Server は、起動時に新しい ID をスレッド・プールに割り当てる場合があるため、Sybase では、スレッド・プール ID の静的参照をおすすめしません。
- Name − スレッド・プール名。システム・スレッド・プールのみが syb_ プレフィックスで始められます。
- Description − (オプション) 最大 255 文字のスレッド・プールの説明。
- Type − `Engine` または `RTC` のいずれかです。
- Current number of threads − プールに現在含まれているスレッド数 (プール・サイズの変更中は、一時的にプールの設定スレッド数と異なる場合があります)。
- Configured number of threads − スレッド・プールが設定されるスレッド数。
- idle timeout − スレッドがアイドル状態からスリープ状態になる間の時間 (マイクロ秒)。

Adaptive Server は、スレッド・プール設定を設定ファイルに記録します。この例では、3 つのデフォルト・スレッド・プール (syb_blocking_pool、syb_system_pool、および syb_default_pool) と、big_pool という名前のユーザ作成スレッド・プールを示しています。

```
[Thread Pool:big_pool]
     description = Big thread pool
```

```
        number of threads = 15

    [Thread Pool:syb_blocking_pool]
        number of threads = 20

    [Thread Pool:syb_default_pool]
         number of threads = 1
```

sp_helpthread または monThreadPool モニタリング・テーブルを使用して、現在のスレッド・プール設定を表示します。Adaptive Server の起動前に設定ファイルのスレッド・プール情報を編集するか、alter thread pool を使用してスレッド・プール設定を変更します。

『システム管理ガイド 第 2 巻』の「第 3 章 メモリの設定」を参照してください。

## エンジンの管理

SMP システムのパフォーマンスを最高にするには、適切なエンジン数を維持する必要があります。

エンジンは、一定量の CPU パワーに相当します。エンジンは、メモリと同様の設定可能リソースです。

---

**注意** Adaptive Server は、ロード・バランシング・アルゴリズムを使用して、エンジン間の負荷を均等に分散します。プロセス・モードでは、サーバへの接続にコンポーネント統合サービス (CIS) を使用する場合、サーバ接続はすべて単一のエンジンに結び付けられ、あるエンジンから別のエンジンにマイグレートすることはできません。この制限は、スレッド・モードには適用されません。

---

### プロセス・モードでのエンジンの設定

Adaptive Server をインストールした直後は、システムは単一エンジンで実行するように設定されています。複数のエンジンを使用するには、初めてサーバを再起動するときにエンジン数を再設定してください。また、他のときにエンジン数を再設定してもかまいません。たとえば、次のような場合です。

- 現在のパフォーマンスがアプリケーションの要求を満たすには十分でないときに、マシン上に十分な数の CPU があれば、エンジン数を増やすことができます。
- Adaptive Server が既存のエンジンを十分に活用していない場合は、エンジンの数を減らします。エンジンの実行にはオーバヘッドが伴うため、不要なエンジンがあると、Adaptive Server によってリソースが無駄に消費されます。

## スレッド・モードでのエンジンの設定

設定ファイルのスレッド・プール情報を編集するか、create thread pool、alter thread pool、および drop thread pool を使用して、スレッド・プールを管理します。「スレッド・プールの設定」(51 ページ) を参照してください。

max online engines は、Adaptive Server で使用できるエンジン総数を制御します。

sp_configure を使用して、max online engines を再設定します。たとえば、エンジン数を 3 に設定するには、次のコマンドを発行します。

```
sp_configure "max online engines", 3
```

エンジン数を再設定するには、サーバを再起動します。

## 適正なエンジン数の選択

Adaptive Server のエンジン数を正しく選択することが重要です。

- CPU の数を超えるエンジンは設定できません。CPU がオフラインになる場合、sp_engine (プロセス・モード)、または alter thread pool (スレッド・モード) を使用してエンジン数を減らす必要があります。
- Adaptive Server は追加のオペレーティング・システム・スレッドを使用して、スレッド・モードで I/O 操作を処理します。大量 I/O の場合、Adaptive Server にこれらのスレッドを実行するために十分な CPU 容量があるかどうかを確認する必要があります。たとえば、8 CPU システムで大量 I/O を実行している Adaptive Server は、プロセス・モードで 8 つのエンジンを使用して最大のパフォーマンスを得ますが、スレッド・モードでは 7 つのエンジンだけが必要です。I/O スレッドが CPU 時間でエンジン・スレッドと競合する場合、パフォーマンスが深刻に低下することがあります。
- 割り当てられている使用可能な CPU と同数のエンジン数だけを設定します。クライアント・プロセスやその他の Adaptive Server 以外のプロセスによる大量の処理がある場合、CPU 1 つにつき 1 つのエンジンでは多すぎることがあります。オペレーティング・システムでさえ、1 つの CPU の一部を占有する場合があります。
- 十分な数のエンジンを設定します。初めはエンジン数を少なくして、既存の CPU の使用率が 100% に近づいたときにエンジンを追加します。エンジン数が少なすぎる場合は、処理量が既存のエンジンの能力を超え、ボトルネックが生じることがあります。

Sybase では、サイトに必要な数のエンジンのみを持つことをおすすめします。エンジン数の増加がよりよいパフォーマンスにつながるとは限りません。一般的には、40% ビジー状態の 8 つのエンジンよりも、80 % ビジー状態の 4 つのエンジンの方が、Adaptive Server のパフォーマンスが良くなります。

## エンジンの起動と停止

sp_engine を使用して、Adaptive Server エンジンを起動または停止します。

---

**注意** Adaptive Server で sp_engine を使用するためにプロセス・モードに設定する必要があります。Adaptive Server がカーネル・モードに設定されている場合、create、alter、および drop thread を使用して、メモリ・プールを設定します。「スレッド・プールの設定」(51 ページ) を参照してください。

---

### エンジンのステータスのモニタ

エンジンをオンラインまたはオフラインにする前に、現在稼動しているエンジンのステータスを確認する必要があります。sysengines の status カラムには、次のいずれかが表示されます。

- online – エンジンがオンラインであることを示します。
- in offline – sp_engine offline が実行されたことを示します。エンジンはサーバに割り付けられた状態のままですが、他のエンジンにタスクをマイグレートする処理を実行しています。
- in destroy – このエンジンからすべてのタスクが正常にマイグレートされ、サーバは OS レベルのタスクによってエンジンの割り付けが解除されるのを待機していることを示します。
- in create – エンジンをオンラインにする処理が進行中であることを示します。
- dormant – sp_engine offline が、エンジンからすべてのタスクをマイグレートできなかったことを示します。タスクが ( タイムアウトによって) 終了する場合、エンジンは永続的にオフラインに切り替わります。休止状態のエンジンは、休止状態を発生させるタスクのみを処理し、他のタスクを処理することはできません。

以下のコマンドを実行すると、エンジン番号、ステータス、結び付けられたタスクの数、エンジンがオンラインになった時刻が表示されます。

```
select engine, status, affinitied, starttime
from sysengines
engine status       affinitied  starttime
------ ------------ ----------- --------------------------
     0 online                12        Mar  5 2007 9:40PM
     1 online                 9        Mar  5 2007 9:41PM
     2 online                12        Mar  5 2007 9:41PM
     3 online                14        Mar  5 2007 9:51PM
     4 online                 8        Mar  5 2007 9:51PM
     5 in offline            10        Mar  5 2007 9:51PM
```

### *sp_engine* によるエンジンの起動と停止

sp_engine を使用してエンジンを動的に起動または停止できます。これを利用すると、時間に伴う処理要件の変動に応じて CPU リソースの設定を変更できます。

sp_engine の構文は次のとおりです。

```
sp_engine {"online" | [offline | can_offline] [, engine_id] |
["shutdown", engine_id]
```

たとえば、次のコマンドを実行するとエンジン 1 がオンラインになります。メッセージはプラットフォーム固有です (この例では Sun Solaris が使用されています)。

```
sp_engine "online", 1
02:00000:00000:2001/10/26 08:53:40.61 kernel  Network and device connection limit is
3042.
02:00000:00000:2001/10/26 08:53:40.61 kernel  SSL Plus security modules loaded
successfully.
02:00000:00000:2001/10/26 08:53:40.67 kernel  engine 1, os pid 8624  online
02:00000:00000:2001/10/26 08:53:40.67 kernel  Enabling Sun Kernel asynchronous disk I/O
strategy
00:00000:00000:2001/10/26 08:53:40.70 kernel  ncheck: Network fc0330c8 online
```

特定のエンジンをオフラインにできるかどうかを調べるには、can_offline を使用します。たとえば、エンジン 1 をオフラインにできるかどうかを調べるには、次のコマンドを使用します。

```
sp_engine can_offline, 1
```

sp_engine のリターン・コードが 0 の場合は、指定したエンジンをオフラインにできます。*engine_id* を指定しないで sp_engine を実行すると、sysengines のエンジンのうち *engine_id* が最大のエンジンのステータスが表示されます。

max online engines の値が、その時点の online ステータスのエンジン数より大きく、CPU が十分に追加エンジンをサポートできる場合にだけ、エンジンをオンラインにできます。

エンジンをオフラインにするには、エンジン ID を入力します。たとえば、エンジン 1 をオフラインにするには、次のコマンドを使用します。

```
sp_engine offline, 1
```

Adaptive Server は、このエンジンに関連付けられたタスクがすべて終了するのを待ってエンジンをオフラインにし、次のようなメッセージを返します。

```
01:00000:00000:2001/11/09 16:11:11.85 kernel  Engine 1 waiting for affinitied process(es)
before going offline
00:00000:00000:2001/11/09 16:16:01.90 kernel  engine 1, os pid 21127  offline
```

エンジン 0 をオフラインにすることはできません。

sp_engine “shutdown” は、指定のエンジンに関連付けられたすべてのタスクを 5 秒以内に強制終了してから、エンジンを停止します。エンジンが休止状態またはオフラインの場合、sp_engine shutdown を使用できます。sp_engine を使用して、エンジンを通常のオフラインにすることを妨げている残りのプロセスをすべて停止します。次のコマンドでは、エンジン 1 が停止します。

```
sp_engine "shutdown", 1
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

### ネットワーク接続とエンジンの関係

(プロセス・モードのみ) UNIX オペレーティング・システムの制約によってプロセスあたりのファイル記述子数が決められているので、エンジン数を減らすと、サーバで使用できるネットワーク接続数が減ります。Windows では、ネットワーク接続の数はエンジン数とは関係ありません。

サーバ間のリモート・プロシージャ・コールを実行するために作成されるネットワーク接続 (Replication Server や XP Server への接続など) はマイグレートできません。したがって、このような接続を管理しているエンジンをオフラインにすることはできません。

### 論理プロセス管理と *dbcc engine(offline)*

論理プロセス管理を使用して、特定のログインやアプリケーションをエンジン・グループにバインドする場合、dbcc engine(offline) の使用には注意が必要です。エンジン・グループのすべてのエンジンをオフラインにすると、次のようになります。

- そのログインまたはアプリケーションは、どのエンジンでも実行できる。
- サーバにログインしている接続に、通知メッセージが送信される。

エンジンとの結び付きはクライアントのログイン時に割り当てられるため、エンジン・グループのエンジンを dbcc engine(“online”) コマンドで再びオンラインにしても、すでにログインしているユーザがマイグレートされることはありません。

## ユーザ接続の管理 (プロセス・モードのみ)

プロセス・モード中、SMP システムがネットワークとの結び付きのマイグレーション (あるエンジンから別のエンジンにネットワーク I/O を移動するプロセス。このマイグレーションをサポートする SMP システムでは、Adaptive Server のすべてのエンジン間にネットワーク I/O の負荷を分散させることができます) をサポートしている場合、各エンジンはその接続のネットワーク I/O を処理します。ログイン時に、クライアント接続タスクは、エンジン 0 から、その時点でサービスしている接続数が最も少ないエンジンにマイグレートされます。クライアントのタスクのネットワーク I/O は、接続が終了するまでそのエンジン (ネットワークとの結び付き) 上で実行されます。SMP システムがこのマイグレーションをサポートしているかどうかを調べるには、プラットフォームの『Adaptive Server Enterprise 設定ガイド』を参照してください。

ネットワーク I/O をエンジン間に分散させることにより、Adaptive Server はより多くのユーザ接続を処理できます。プロセスごとにオープン可能なファイル記述子の最大数に関する制限によって接続数が制限されることはなくなりました。エンジンを追加すると、グローバル変数 *@@max_connections* に保管されている最大ファイル記述子数がそれに比例して増えます。

エンジン数を増やすと、増加した *@@max_connections* の値が標準出力に出力され、サーバを再起動したときにこの値がエラー・ログ・ファイルに出力されます。次のコマンドを使用すると、その値を問い合わせることができます。

```
select @@max_connections
```

この値は、プロセスのファイル記述子の最大数としてオペレーティング・システムで定められている数から、Adaptive Server が使用する次のようなファイル記述子の数を差し引いたものです。

- エンジン 0 上のマスタ・ネットワーク・リスナごとに 1 つ (*interfaces* ファイル内の該当する Adaptive Server のエントリの “master” 行ごとに 1 つ)
- エンジンの標準出力ごとに 1 つ
- エンジンのエラー・ログ・ファイルごとに 1 つ
- エンジンのネットワークとの結び付きのマイグレーション・チャネルごとに 2 つ
- エンジンごとに 1 つ (設定用)
- エンジンごとに 1 つ (*interfaces* ファイル用)

たとえば Adaptive Server が 1 つのエンジンを使用するように設定されていて、*@@max_connections* の値が 1,019 である場合に、2 つ目のエンジンを追加すると、マスタ・ネットワーク・リスナが 1 つだけであれば、*@@max_connections* の値は 2,039 に増えます。

増加した *@@max_connections* の制限値が使用されるように、number of user connections パラメータを設定できます。ただし、max online engines を使用してエンジン数を減らすときは、それに従って number of user connections の値も調整する必要があります。max online engines と number of user connections の再設定は動的ではありません。したがって、これらの設定値を変更するにはサーバを再起動する必要があります。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

**注意** スレッド・モードは単一プロセスを実行するため、プロセスごとの記述子の制限によって、Adaptive Server がサポートできるユーザ接続の数が制限されます。Adaptive Server を起動するシェルで使用できるファイル記述子の数を調整する必要があります。ハード制限値を増やすためには、オペレーティング・システム管理者に連絡してください。

## SMP システムに作用する設定パラメータ

一部の設定パラメータ (スピンロック率など) は、SMP システムのみにあてはまります。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

### スピンロック率パラメータの設定

スピンロック率のパラメータは、1 つの **spinlock** によって保護される内部システム・リソース ( 内部テーブル内のロー、キャッシュなど ) の数を指定します。スピンロックは、あるプロセスが使用しているシステム・リソースに別のプロセスがアクセスできないようにする、簡単なロック・メカニズムです。そのリソースにアクセスしようとするすべてのプロセスは、ロックが解放されるまで待つ (つまり「スピンする」) 必要があります。

スピンロック率設定パラメータは、マルチプロセッシング・システムでのみ有効です。Adaptive Server に設定されているエンジンが 1 つだけの場合は、スピンロック率設定パラメータの設定値に関係なく、スピンロックは 1 つだけになります。

表 5-1 は、スピンロックによって保護されるシステム・リソースと、デフォルトのスピンロック率を変更するために使用する設定パラメータのリストです。

**表 5-1: スピンロック率設定パラメータ**

| 設定パラメータ | 保護されるシステム・リソース |
|---|---|
| lock spinlock ratio | ロック・ハッシュ・バケットの数 |
| open index hash spinlock ratio | インデックス・メタデータ記述子のハッシュ・テーブル |
| open index spinlock ratio | インデックス・メタデータ記述子 |
| open object spinlock ratio | オブジェクト・メタデータ記述子 |
| partition spinlock ratio | 内部パーティション・キャッシュ内のロー |
| user log cache spinlock ratio | ユーザ・ログ・キャッシュ |

スピンロック率パラメータに指定する値は、スピンロックの数ではなく、リソースとスピンロックの比率を定義するものです。たとえば、スピンロック率に 100 を指定すると、100 リソースごとに 1 つのスピンロックが割り付けられます。Adaptive Server が割り付けるスピンロックの数は、リソースの総数と、指定された率によって決まります。スピンロック率の値を小さくすればするほど、スピンロック数は増えていきます。

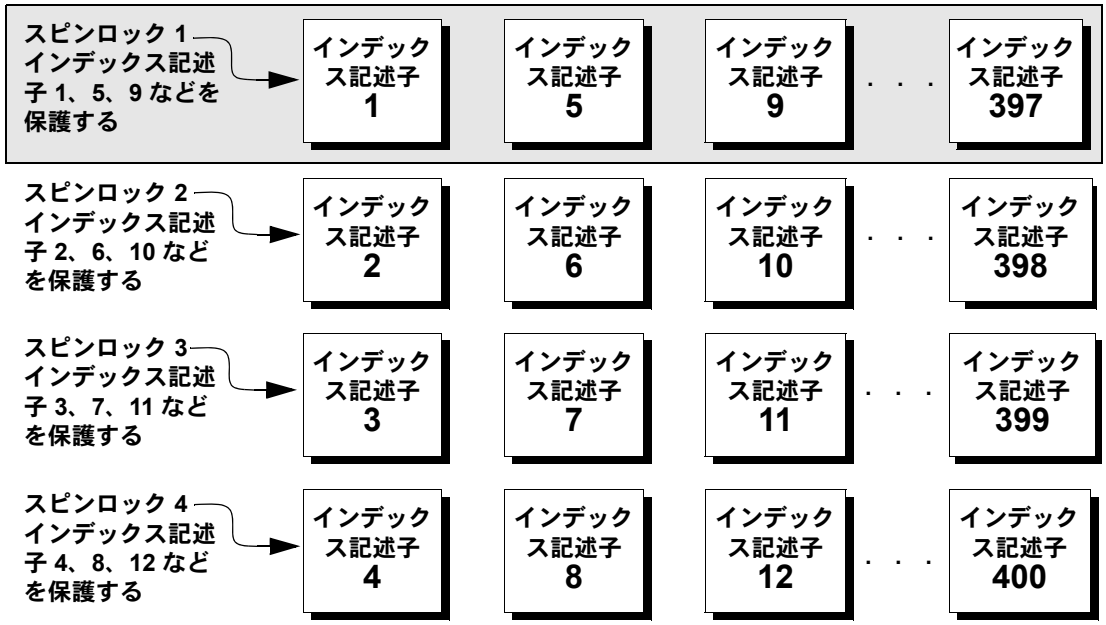
スピンロックは、ラウンドロビン方式またはシーケンシャル方式でシステム・リソースに割り当てられます。

### ラウンドロビン割り当て

open index hash spinlock ratio、open index spinlock ratio、open object spinlock ratio の各パラメータによって設定されるメタデータ・キャッシュ・スピンロックは、ラウンドロビン割り当てメソッドを使用します。

図 5-2 は、ラウンドロビン割り当てメソッドの 1 つの例を示し、スピンロックとインデックス・メタデータ記述子の関係を説明しています。

図 5-3: スピンロックとインデックス記述子の関係




インデックス・メタデータ記述子の数が 400、つまりインデックス記述子の内部テーブルに 400 ローがあり、スピンロック率が 100 に設定されているとします。このとき、スピンロックは 4 つになります。つまり、スピンロック 1 はロー 1、スピンロック 2 はロー 2、スピンロック 3 はロー 3、スピンロック 4 はロー 4 をそれぞれ保護します。その後、スピンロック 1 は次に利用可能なインデックス記述子であるインデックス記述子 5 を保護します。そして、すべてのインデックス記述子がスピンロックによって保護されるまで、これが続けられます。このラウンドロビン・メソッドによる記述子の割り当てによって、スピンロックが競合する可能性が低くなります。

### シーケンシャル割り当て

lock table spinlock ratio パラメータによって設定されるテーブル・ロックのスピンロックは、シーケンシャル割り当てメソッドを使用します。lock table spinlock ratio のデフォルト設定は 20 です。この設定では、内部ハッシュ・テーブル内の 20 ローが各スピンロックに割り当てられます。ローはシーケンシャルに分割されます。最初のスピンロックは最初の 20 ローを保護し、2 番目のスピンロックは次の 20 ローを保護し、3 番目以降も同様の保護を行います。

理論上は、1 つのリソースを 1 つのスピンロックで保護することによってスピンロックの競合が最小になり、その結果、同時実行性が最大になります。ほとんどの場合、これらのスピンロック率のデフォルト値は、使用しているシステムに最適な値となります。スピンロック率は、スピンロックの競合が発生した場合のみ変更します。

スピンロック競合のレポートを取得するには、sp_sysmon を使用します。『パフォーマンス＆チューニング・シリーズ：sp_sysmon による Adaptive Server の監視』を参照してください。

第 6 章

# ユーザ・データベースの作成と管理



## ユーザ・データベースの作成と管理のコマンド

表 6-1 に、ユーザ・データベースとそのトランザクション・ログを作成、変更、削除するために使用するコマンドの要約を示します。

**表 6-1: ユーザ・データベースの管理用コマンド**

| コマンド | タスク |
|---|---|
| create database...on *dev_name*<br>または<br>alter database...on *dev_name* | 特定の Adaptive Server データベースで使用できるようにデータベース・デバイスを設定する。<br>on *dev_name* 句を指定しない場合は、データベース・デバイスのデフォルト・プールから領域が割り付けられる。 |
| dbcc checktable(syslogs) | ログのサイズをレポートする。 |
| sp_logdevice | 現在のログ・デバイスに空きがなくなった場合にログを保管するデバイスを指定する。 |
| sp_helpdb | データベースのサイズとデバイスについての情報を表示する。 |
| sp_spaceused | データベースが使用する記憶領域量の要約情報を表示する。 |

# ユーザ・データベースを管理するためのパーミッション

デフォルトでは、create database のパーミッションを持つのはシステム管理者だけですが、システム管理者は create database コマンドを使用するためのパーミッションを付与できます。ただし、通常のインストール環境では、データベースの配置とデータベース・デバイスの割り付けを集中管理するために、システム管理者だけが create database のパーミッションを持ちます。このような状況では、システム管理者が他のユーザに代わって新しいデータベースを作成してから、該当するユーザに所有権を譲渡します。

システム管理者は、データベースを作成してから別のユーザに所有権を譲渡するには、次の手順に従います。

1 create database コマンドを発行します。

2 use *database* コマンドを使用して新しいデータベースに切り替えます。

3 sp_changedbowner を実行します。詳細については、「データベース所有権の変更」(158 ページ) を参照してください。

また、システム管理者がデータベース作成のパーミッションをユーザに付与する場合、そのパーミッションを受け取るユーザも、master データベースの正当なユーザでなければなりません。データベースの作成は、必ず master を使用している状態で行うからです。

システム管理者が行う操作 (たとえば、バックアップやダンプ) が保護システムの外部で行われるように見えるということには、安全対策としての役割があります。たとえば、あるデータベース所有者が自分のパスワードを忘れてしまったときや、sysusers 内のすべてのエントリを誤って削除したときも、システム管理者は、定期的に取得しているバックアップまたはダンプを使用することによって損傷を修復できます。

alter database コマンドまたは drop database コマンドを使用するパーミッションは、デフォルトではデータベース所有者に付与されています。このパーミッションは、データベースの所有権とともに自動的に譲渡されます。grant コマンドまたは revoke コマンドを使用して、alter database および drop database のパーミッションを変更することはできません。

# *create database* コマンドの使用方法

create database を使用するには、create database のパーミッションが必要です。また、master データベースの正当なユーザでなければなりません。必ず、use master を実行してから新しいデータベースを作成します。『リファレンス・マニュアル：コマンド』を参照してください。

---

**注意**　create database コマンドを実行するたびに、master データベースをダンプしてください。これによって、master に損傷が生じた場合のリカバリを、容易に、また安全に行うことができます。「第 14 章 システム・データベースのリストア」参照。

---

一度に作成できるデータベースの数は 1 つだけです。

次に示す最も単純な形式の create database では、master..sysdevices に登録されているデフォルトのデータベース・デバイス上に 1 つのデータベースが作成されます。

```
create database newpubs
```

必要なパーミッションを持っているユーザが create database を発行すると、Adaptive Server は次の処理を実行します。

- 指定されたデータベース名がユニークで、識別子の規則に従っているかどうかを確認します。
- 指定されたデータベース・デバイス名を使用できるかどうかを確認します。
- 新しいデータベース用に未使用の ID 番号を検索します。
- 指定されたデータベース・デバイス上でデータベースに領域を割り当て、master..sysusages を更新してその割り当てを反映させます。
- sysdatabases にローを挿入します。
- 新しいデータベース領域に model データベースのコピーを作成することによって、新しいデータベースのシステム・テーブルを作成します。
- データベース・デバイス内の残りのページをすべてクリアします。データベース・ダンプをロードするためのデータベースを作成する場合は、for load を指定すればページのクリア処理は省略されます (ページのクリアはロード完了後に実行されます)。

新しいデータベースの初期状態では、システム・テーブルが格納されており、システム・テーブルにはシステム・テーブル自身のエントリが保管されています。新しいデータベースは、model データベースに加えられた変更をすべて継承します。継承される変更内容には、次のものがあります。

- ユーザ名の追加。
- オブジェクトの追加。
- データベース・オプションの設定。model データベースのオプションは、初期設定ではオフになっています。特定のオプションをすべてのユーザ・データベースに継承させるには、sp_dboption を使用して model データベースのオプションを変更します。『システム管理ガイド 第 1 巻』の「第 2 章 システム・データベースとオプションのデータベース」と「第 8 章 データベース・オプションの設定」を参照してください。

データベースを新規作成した後、システム管理者またはデータベース所有者は、sp_adduser を使用して、手動でデータベースにユーザを追加できます。新しい Adaptive Server ログインを追加する場合は、システム・セキュリティ担当者が必要な場合もあります。『システム管理ガイド 第 1 巻』の「第 13 章 Adaptive Server のセキュリティ管理について」を参照してください。

# データベースへの領域とデバイスの割り当て

create database コマンドまたは alter database コマンドを実行すると、データベースに記憶領域が割り付けられます。create database では、1 つまたは複数のデータベース・デバイスを指定でき、このときに個々のデータベース・デバイスから新しいデータベースに割り付ける領域の大きさを指定できます。

---

**警告！** 小さいデータベースや重要でないデータベースを作成する場合を除いて、ログは常に別のデータベース・デバイスに配置してください。運用データベースを作成するには、「別のデバイスへのトランザクション・ログの保管」(142 ページ) の手順に従ってください。

---

default キーワードを使用するか、on 句を省略すると、master..sysdevices で指定されている 1 つまたは複数のデフォルト・データベース・デバイスにデータベースが配置されます。『システム管理ガイド 第 1 巻』の「第 7 章 データベース・デバイスの初期化」を参照してください。

デフォルト・ロケーションに保管するデータベースのサイズ (次の例では 4MB) を指定するには、以下を使用します。

```
create database newpubs
```

```
on default = "4M"
```

データベースを特定のデータベース・デバイスに配置する場合は、保管するデータベース・デバイスの名前をコマンドに含めます。1 つのデータベースを、複数のデータベース・デバイスに保管することを要求し、それぞれのデバイスに異なる領域の大きさを指定することができます。create database で指定するすべてのデータベース・デバイスは、sysdevices 内に登録されている必要があります (つまり、データベース・デバイスは disk init を使用して初期化されている必要があります )。『システム管理ガイド 第 1 巻』の「第 7 章 データベース・デバイスの初期化」を参照してください。

次の文では、newdb データベースを作成し、mydata に 3MB を割り付け、newdata に 2MB を割り付けます。データベースとトランザクション・ログは別々のデバイスには配置しません。

```
create database newdb
on mydata = "3M", newdata = "2M"
```

要求した大きさの領域が指定のデータベース・デバイス上で確保できない場合は、各デバイス上で可能なかぎりの大きさの領域を割り付けてデータベースが作成され、各データベース・デバイスに割り付けられた領域のサイズを通知するメッセージが表示されます。指定したデータベース・デバイス上にデータベースの作成に必要な最低限の空き領域がない場合、create database コマンドは失敗します。

dsync 設定値を使用していない UNIX デバイス・ファイル上でデータベースを作成または変更すると、エラー・ログ・ファイルに次のようなエラー・メッセージが出力されます。

```
Warning: The database 'newdb' is using an unsafe virtual device 'mydata'. The recovery
of this database can not be guaranteed.
```

## デフォルトのデータベース・サイズとデバイス

on 句の size パラメータを省略すると、作成されるデータベースにはデフォルトの大きさの領域が割り当てられます。この領域の大きさは、default database size 設定パラメータで指定されるサイズと、model データベースのサイズのいずれか大きい方です。

作成できる最小のデータベースのサイズは model データベースのサイズです。このサイズはインストール環境の論理ページ・サイズによって決まります。データベースの最小サイズを増やすには、alter database を使用して model データベースを大きくします。default database size 設定パラメータを使用して、データベースのデフォルトのサイズを決めることもできます。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

on 句を省略すると、上述のように、データベースのサイズはデフォルト・サイズになります。領域は、データベース・デバイス名のアルファベット順に、master..sysdevices で指定されているデフォルト・データベース・デバイスから割り付けられます。

デフォルトのデータベース・デバイスの論理名を調べるには、次のように入力します。

```
select name
    from sysdevices
    where status & 1 = 1
    order by name
```

また、sp_helpdevice を実行すると、データベース・デバイスの説明の部分に「デフォルト・ディスク」が表示されます。

### 必要な領域の見積もり

割り当てが完了した記憶領域を元に戻すことは難しいので、サイズの割り付けの決定は重要な作業です。領域の追加はいつでもできますが、データベースに割り付けた領域の解除は、データベースを削除してからでないとできません。

データベースのテーブルとインデックスのサイズを見積もるには、sp_estspace を使用するか、自分で値を計算します。『パフォーマンス&チューニング・シリーズ：物理データベースのチューニング』の「第 4 章 テーブルとインデックスのサイズの決定」を参照してください。

## 別のデバイスへのトランザクション・ログの保管

create database コマンドの log on 句によって、トランザクション・ログ (syslogs テーブル) が別のデータベース・デバイスに配置されます。小さいデータベースや重要でないデータベースを作成する場合を除いて、ログは常に別のデータベース・デバイスに配置してください。ログを別のデータベース・デバイスに配置する理由は次のとおりです。

- dump database コマンドではなく dump transaction コマンドを実行できるため、時間とテープを節約できる。
- ログの固定のサイズを設定できるため、他のデータベースのアクティビティとの領域の競合がない。
- ログ・セグメントに対するデフォルトの空き領域スレッショルドのモニタリングが作成され、さらに、データベースのログとデータ部分に対する空き領域のモニタリングをユーザが作成できる。「第 17 章 スレッショルドによる空き領域の管理」を参照してください。

- パフォーマンスが向上する。
- ハード・ディスクがクラッシュした場合に、安全かつ完全にリカバリができる。また、データ・デバイスが損傷を受けたディスク上にあっても、dump transaction の特別な引数を使用することによってトランザクション・ログをダンプできる。

トランザクション・ログのサイズとデバイスを指定するには、log on *device* = *size* 句を使用して create database を実行します。サイズは、単位指定子 “k” または “K” (キロバイト)、“m” または “M” (メガバイト)、“g” または “G” (ギガバイト)、“t” または “T” (テラバイト) で表されます。次の文は、newdb データベースを作成して、mydata に 8MB、newdata に 4MB を割り付ける例です。また、3 番目のデータベース・デバイス tranlog に 3MB のトランザクション・ログを配置します。

```
create database newdb
on mydata = "8M", newdata = "4M"
log on tranlog = "3M"
```

## トランザクション・ログ・サイズの見積もり

トランザクション・ログのサイズは、次の 2 つの要因を考慮して決定します。

- 対応するデータベースの更新アクティビティの量
- トランザクション・ログ・ダンプの頻度

これはトランザクション・ログ・ダンプを手動で行う場合も、スレッショルド・プロシージャを使用して自動で行う場合も同じです。データベースに割り付ける領域の 10 ～ 25 パーセント程度をログに割り付けてください。

挿入、削除、更新を行うと、ログのサイズは増加します。dump transaction を実行すると、コミットされたトランザクションがディスクに書き込まれ、ログから削除されるので、ログのサイズが減少します。update 文ではローの更新前イメージと更新後イメージをログに記録する必要があるので、一度に大量のローを更新するアプリケーションでは、同時に更新するローの少なくとも 2 倍、または最大のテーブルの少なくとも 2 倍のトランザクション・ログ領域を用意してください。または、更新を小さなグループのバッチに分け、バッチとバッチの間にトランザクション・ダンプを実行します。

大量の挿入や更新処理が行われるデータベースでは、ログは急速に増大します。必要なログ・サイズを決定するには、定期的にログ・サイズを確認してください。これは、ログのスレッショルドを選択したり、トランザクション・ログ・ダンプの実行のタイミングを計画したりするときにも役立ちます。データベースのトランザクション・ログによって使用されている領域を確認するには、use コマンドで対象のデータベースへ移動してから、次のように入力します。

```
dbcc checktable(syslogs)
```

dbcc は、ログによって使用されているデータ・ページ数をレポートします。ログが別のデバイスにある場合は、dbcc checktable を実行して、使用されている領域と空き領域の大きさを調べることもできます。次に示すのは、2MB のログの出力例です。

```
Checking syslogs
The total number of data pages in this table is 199.
*** NOTICE: Space used on the log segment is 0.39 Mbytes, 19.43%.
*** NOTICE: Space free on the log segment is 1.61 Mbytes, 80.57%.
Table has 1661 data rows.
```

ログの増加をチェックするには、次のように入力します。

```
select count(*) from syslogs
```

どちらかのコマンドを定期的に繰り返し実行して、ログの増加速度を調べてください。

## デフォルトのログ・サイズとデバイス

log on 句から *size* パラメータを省略すると、許される最小限の大きさの記憶領域が配置されます。log on 句全体を省略すると、トランザクション・ログ領域はデータ・テーブルと同じデータベース・デバイス上に配置されます。

## 別のデバイスへのトランザクション・ログの移動

create database に log on 句を使用しなかった場合は、次の手順でトランザクション・ログを別のデータベース・デバイスに移動できます。

sp_logdevice は、指定されたデバイスに存在する既存のデータベースの部分に、トランザクション・ログ用に予約されているものとしてマークを付けます。このとき既存のデータは移動されません。このデバイスに既にデータがある場合、このデータは、適切なセグメント上に存在していないと解釈されません。ただし、dbcc はこれをエラーとして通知するので、既存のログの部分が指定したデバイスに移動することはありません。ログが新しいデバイスまで拡張され、dump transaction を使用してログのその部分がクリアされるまで、現在のログ・データはその場所に残ります。また、sp_logdevice は、データベースに新しい領域を割り付けたり、デバイスを初期化したりすることはありません。代わりに、指定されたデバイスのその部分を、指定されたデータベースに既に属するログ用に予約します。

sp_logdevice の構文は次のとおりです。

```
sp_logdevice database_name, devname
```

指定するデータベース・デバイスは、disk init で初期化され、create database または alter database でそのデータベースに割り付けられている必要があります。

トランザクション・ログ全体を別のデバイスに移動させるには、次の手順に従います。

1　sp_logdevice を実行して、新しいデータベース・デバイスの名前を指定します。

2　現在使用中のページがいっぱいになるまでトランザクションを実行します。更新に必要な領域量は、使用している論理ページのサイズによって異なります。dbcc checktable(syslogs) を更新の前後で実行すると、新しいページが使用されているかどうかを確認できます。

3　現在アクティブなトランザクションがすべて終了するまで待ちます。sp_dboption を使用して、データベースをシングルユーザ・モードにすることもできます。

4　dump transaction を実行します。ディスクに書き込まれたログ・ページがすべて削除されます。元のデバイスのログにアクティブなトランザクションのログ・ページがなければ、すべてのページが削除されます。「第 12 章 バックアップおよびリカバリ・プランの作成」を参照してください。

5　sp_helplog を実行して、完全なログが新しいログ・デバイス上にあることを確認します。

**注意**　トランザクション・ログを移動すると、トランザクション・ログに使用されていた領域をデータ用に使用できるようになります。ただし、トランザクション・ログの移動によって、デバイスに割り付けた領域を減らすことはできません。

トランザクション・ログの詳細については、「第 12 章 バックアップおよびリカバリ・プランの作成」を参照してください。

## ログ領域の縮小

alter database コマンドには、データベース・ログの不要な部分を削除する log off パラメータが含まれ、データベースを再作成しなくてもログ領域を縮小して記憶領域を解放することができます。

構文は次のとおりです。

```
alter database database_name [log off database_device
    [= size | [from logical_page_number] [to logical_page_number]]
    [, database_device
    [= size | [from logical_page_number] [to logical_page_number]]
```

パラメータは、select into、alter table、reorg rebuild などのデータベース操作のフル・ログを取るオプションを実行した後、データベースに割り当てられた追加スペースが不要になった場合に特に有益になります。『リファレンス・マニュアル：コマンド』の dump transaction を参照してください。

## ログ領域縮小時の *dump* および *load database* の使用

dump および load database を実行すると、次のようになります。

- ロードしているデータベースには、少なくともダンプされた時の領域と同じ物理領域が必要です。
- ロードしているデータベースの物理領域は、ダンプされたデータベースの物理的なフラグメントに応じて割り当てられます。これは「穴」となり、前の alter database log off コマンドの結果として生成された、対応する物理記憶領域のないアロケーション・ユニットのことを意味します。ロードするデータベースでは、ロードを実行した後に穴が必ずしも保持されるとは限りません。
- ローディングしているデータベースの残りの領域は、穴のない dump および load database の場合と同じ方法で割り当てられます。
- load database with headeronly コマンドを実行することで、ダンプされたデータベースの物理領域の量、穴があるかどうか、およびこれらの穴のサイズと位置に関する情報を調べることができます。

### ❖ dump および load database を実行する前にログを縮小する

このシナリオでは、ダンプされるデータベースのログを縮小します。load database with headeronly コマンドおよび sp_helpdb システム・プロシージャを使用して、ダンプとともにロードされる前に、ターゲット・データベースの大きさを指定します。これらのコマンドを表示する完全なシーケンスについては、「dump と load database を使用したシーケンスの例」(147 ページ)を参照してください。

1. 必要なだけログ・デバイスを使用してデータベースを作成します。この例では、2 つのログ・デバイスが作成されます。
2. (オプション) select * を実行し、データベースの作成を確認して、データベースを構成するデバイス・フラグメントを表示します。
3. alter database log off コマンドを使用し、データベース・ダンプ・シーケンスをブレークせずに、データベースからログの不要な部分を削除します。データベースのダンプ・シーケンスがすでにブレークされている場合は、alter database log off によって自動的にデータベースの最後から縮小された領域が削除されます。データベースの最後ではない削除された領域は、常に穴になります。

この例では、データベースの中間にある縮小された領域が穴になっています。

4　sysusages 出力は、穴の位置とサイズを示します (例では segmap = 0、location は 4)。sp_helpdb 出力は、穴を除くデータベースのサイズの概要 (例では 9MB)、およびデータベースの穴の合計サイズ (例では唯一使用できない領域としてログの 3072KB (3MB)) を示します。

5　ロードするデータベースは合計で 12MB ですが、穴が 3MB あるため、このうち 9MB がデータベースの実際の物理領域になります。dump database with headeronly コマンドによって、データベースには論理ページの 12MB と物理ページの 9MB が含まれていることが確認されます。このデータベースをロードするには、少なくとも 9MB のサイズの新しいデータベースを作成してください。

6　データベースをロードし、オンラインにします。

新しくロードされたデータベースの sysusages ローでは、ダンプされたデータベースの領域と一致するように 9MB の物理領域が再配列され、データベースは 9MB の物理ページと 3MB の穴で構成された 12MB サイズになります。

## dump と load database を使用したシーケンスの例

この例では、dump と load database を使用するときに実行する完全なシーケンスを示します。詳細については、「dump および load database を実行する前にログを縮小する」(146 ページ) を参照してください。

```
1> create database sales_db on sales_db_dev=3 log on sales_db_log1=3,
    sales_db_log2=3, sales_db_log1=3, sales_db_log2=3
2> go
00:00:00000:00015:2011/01/21 09:38:28.29 server  Timestamp for database
'sales_db' is (0x0000,
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log2'
(1536 logical pages requested).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log2'
(1536 logical pages requested).
Warning: The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'.
The recovery of this database can not be guaranteed.
Database 'sales_db' is now online.

1> select * from sysusages where dbid=4
2> go

dbid segmap lstart size vstart location unreservedpgs  crdate            vdevno
```

```
---- ------ ------ ---- ------ -------- -------------  ------              ------
    4      3      0 1536      0        0           670  Jan 21 2011 9:38AM     1
    4      4   1536 1536      0        0          1530  Jan 21 2011 9:38AM     2
    4      4   3072 1536      0        0          1530  Jan 21 2011 9:38AM     3
    4      4   4608 1536   1536        0          1530  Jan 21 2011 9:38AM     2
    4      4   6144 1536   1536        0          1530  Jan 21 2011 9:38AM     3

(5 rows affected)

1> alter database sales_db log off sales_db_log2
2> select * from sysusages where dbid=4
3> go
dbid segmap lstart size vstart location unreservedpgs  crdate              vdevno
---- ------ ------ ---- ------ -------- -------------  ------              ------
    4      3      0 1536      0        0           670  Jan 21 2011 9:38AM     1
    4      4   1536 1536      0        0          1530  Jan 21 2011 9:38AM     2
    4      0   3072 1536   3072        4          1530  Jan 21 2011 9:38AM    -4
    4      4   4608 1536   1536        0          1530  Jan 21 2011 9:38AM     2

(4rowsaffected)

1> sp_helpdb sales_db
2> go
 name     db_size       owner dbid  created      durability status
 -------- ------------- ----- ----  ------------ ---------- --------------
sales_db  9.0 MB        sa       4  Jan 21, 2011 full       no options set

(1 row affected)
device_fragments   size    usage      created               free kbytes
------------------ ------  ---------- ------------------    --------------
sales_db_dev       3.0 MB  data only  Jan 21 2011 9:38AM             1340
sales_db_log1      3.0 MB  log only   Jan 21 2011 9:38AM    not applicable
sales_db_log1      3.0 MB  log only   Jan 21 2011 9:38AM    not applicable

------------------------------------------------------------
log only free kbytes = 6082, log only unavailable kbytes = 3072
(return status = 0)

1> dump database sales_db to "c:/temp/sales_db.dmp"
2> go
Backup Server session id is: 45. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from
the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file c:/temp/sales_db.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db11021087C7  ' section number 1
mounted on disk file 'c:/temp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db: 848 kilobytes (67%) DUMPED.
Backup Server: 4.188.1.1: Database sales_db: 862 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database sales_db: 870 kilobytes (100%) DUMPED.
```

```
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> load database sales_db from "c:/temp/sales_db.dmp" with headeronly
2> go
Backup Server session id is: 48. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db11021087C7  ' section number 1 mounted
on disk file 'c:/temp/sales_db.dmp'
This is a database dump of database ID 4, name 'sales_db', from Jan 21 2011 9:39AM. ASE
version: lite_642236-1/Adaptive Server Enterprise/15.7/EBF 18567 SMP
Drop#2/B/X64/Windows Server/aseasap/ENG/. Backup Server version: Backup
Server/15.7/B/X64/Windows Server/aseasap/ENG/64-bit/DEBUG/Thu Jan 20 11:12:51 2011.
Database page size is 2048.
Database contains 6144 pages; checkpoint RID=(Rid pageid = 0x604; row num = 0x12); next
object ID=560001995; sort order ID=50, status=0; charset ID=2.
Database log version=7; database upgrade version=35; database durability=UNDEFINED.
segmap: 0x00000003 lstart=0 vstart=[vpgdevno=1 vpvpn=0] lsize=1536 unrsvd=670
segmap: 0x00000004 lstart=1536 vstart=[vpgdevno=2 vpvpn=0] lsize=1536 unrsvd=1530
Unavailable disk fragment: lstart=3072 lsize=1536
segmap: 0x00000004 lstart=4608 vstart=[vpgdevno=2 vpvpn=1536] lsize=1536 unrsvd=1530
The database contains 6144 logical pages (12 MB) and 4608 physical pages (9 MB).

1> create database sales_db2 on sales_db_dev=3 log on sales_db_log1=6
2> go
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE: allocating 3072 logical pages (6.0 megabytes) on disk 'sales_db_log1'
(3072 logical pages requested).
Warning: The database 'sales_db2' is using an unsafe virtual device 'sales_db_dev'.
The recovery of this database can not be guaranteed.
Database 'sales_db2' is now online.

1> select * from sysusages where dbid=db_id("sales_db2")
2> go
dbid segmap lstart size vstart location unreservedpgs crdate             vdevno
---- ------ ------ ---- ------ -------- ------------- ------             ------
   5      3      0 1536   1536        0           670 Jan 26 2011 1:22AM      1
   5      4   1536 3072   3072        0          3060 Jan 26 2011 1:22AM      2


1> load database sales_db2 from "/tmp/sales_db.dmp"
2> go
Backup Server session id is: 10. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db1102602564  ' section number 1 mounted
on disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db2: 6148 kilobytes (33%) LOADED.
Backup Server: 4.188.1.1: Database sales_db2: 9222 kilobytes (50%) LOADED.
Backup Server: 4.188.1.1: Database sales_db2: 9230 kilobytes (100%) LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db2).
Started estimating recovery log boundaries for database 'sales_db2'.
Database 'sales_db2', checkpoint=(1544, 22), first=(1544, 22), last=(1544, 22).
Completed estimating recovery log boundaries for database 'sales_db2'.
```

```
Started ANALYSIS pass for database 'sales_db2'.
Completed ANALYSIS pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:15.86 server  Log contains all committed transactions
until 2011/01/26 01:55:15.71 for database sales_db2.
Started REDO pass for database 'sales_db2'. The total number of log records to process
is 1.
Completed REDO pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:15.88 server  Timestamp for database 'sales_db2' is
(0x0000, 0x00001612).
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> online database sales_db2
2> go
Started estimating recovery log boundaries for database 'sales_db2'.
Database 'sales_db2', checkpoint=(1544, 22), first=(1544, 22), last=(1544, 22).
Completed estimating recovery log boundaries for database 'sales_db2'.
Started ANALYSIS pass for database 'sales_db2'.
Completed ANALYSIS pass for database 'sales_db2'.
00:00:00000:00011:2011/01/26 05:12:22.49 server  Log contains all committed transactions
until 2011/01/26 01:55:15.71 for database sales_db2.
Recovery of database 'sales_db2' will undo incomplete nested top actions.
Database 'sales_db2' is now online.

1> select * from sysusages where dbid=db_id("sales_db2")
2> go
dbid segmap lstart size vstart location unreservedpgs crdate              vdevno
---- ------ ------ ---- ------ -------- ------------ ------              ------
   5      3      0 1536   1536        0          670 Jan 26 2011 5:12AM      1
   5      4   1536 1536   3072        0         1530 Jan 26 2011 5:12AM      2
   5      0   3072 1536   3072        4         1530 Jan 26 2011 5:12AM     -5
   5      4   4608 1536   4608        0         1530 Jan 26 2011 5:12AM      2
```

## ログ領域の縮小時の *dump* および *load transaction* の使用

ログのサイズは、データベースが最初にダンプされるダンプ・シーケンス中に変更される場合があり、トランザクション・ログのダンプは定期的に実行されます。完全にログされた select into によって行われるログの増加ボリュームに対応するために、ログ・セグメントが増加し、そのログがコマンドの完了後に元のサイズに戻るように縮小された場合は、特にその可能性が高くなります。このようなダンプ・シーケンスをロードするには、次のガイドラインを使用します。

- ダンプ・シーケンス中に最大サイズのダンプからロードされるデータベースを作成します。これを決定するには、ロードされる最後のトランザクション・ダンプに dump tran with headeronly を実行します。
- ログは、必ずトランザクション・ログのシーケンスが完了し、データベースをオンラインにした後で必要なサイズに縮小します。

### ❖ ログが縮小されているデータベースのダンプ・シーケンスのロード

番号付きの手順の後に、これらのコマンドと出力を表示する例が、「dump と load transaction を使用したシーケンスの例」(152 ページ) に表示されています。

1. データベースを作成します。この例では sales_db を作成します。
2. sp_dboption システム・プロシージャの 'full logging for all' データベース・オプションを使用して、データベースの完全なロギングをオンにします。
3. データベースをダンプします。
4. 完全にログされた select into コマンドの実行できる形式で、alter database log on を使用してログ・セグメントのサイズを大きくします。
5. 増加したログ・セグメントを使用する、完全にログされた select into コマンドを実行します。
6. トランザクション・ログをダンプし、ログをトランケートして、ログ・セグメントを縮小するために準備します。
7. 前の手順で追加されたログ領域を削除するには、alter database log off を使用してデータベースのログを縮小します。
8. 縮小したログ・セグメントを持つデータベースのトランザクション・ログをダンプします。
9. ダンプのシーケンスをロードする前に、ロード・シーケンスの最後のファイルからデータベースの論理サイズを取得します。この例では、サイズは 16MB です。

   **注意** ロード・シーケンスの最後のダンプから取得したデータベースの論理サイズは、少なくともダンプ・シーケンス全体のデータベースの最大物理サイズと同じ大きさであることが保証されています。これにより、シーケンス内のすべてのダンプをロードするために必要なターゲット・データベースのサイズを決める便利なメソッドが生成されます。

   シーケンス内のすべてのダンプに対応するために、load transaction with headeronly コマンドを使用して、必要なターゲット・データベースのサイズを決めます。
10. 必要なだけログ・デバイスを使用して新しいデータベースを作成します。この例では、2 つのログ・デバイスを持つ 16MB のデータベースとして sales_db1 データベースを作成します。
11. このデータベースをロードします。
12. 1 番目と 2 番目のトランザクション・ログ・ダンプからデータベースにトランザクション・ログをロードします。

13 データベースをオンラインにします。

14 ログ・セグメントから領域を削除して、データベースのサイズを小さくします。この例では、ログ・セグメントは 10MB のサイズに縮小されます。

15 select * from sysusages を実行して、データベースの最後から領域が削除されたことを確認します。削除された領域はデータベースの穴になっています。

16 dump database の with shrink_log オプションを使用して、データベースの最後にある穴を削除します。

17 select * from sysusages を再度実行し、Adaptive Server によってデータベースの最後から穴が正常に削除されたことを確認します。

## dump と load transaction を使用したシーケンスの例

この例では、dump と load transaction を使用するときに実行する完全なシーケンスを示します。詳細については、「ログが縮小されているデータベースのダンプ・シーケンスのロード」(151 ページ) を参照してください。

```
1> create database sales_db on sales_db_dev=3 log on sales_db_log1=3
2> go
00:00:00000:00018:2011/05/05 12:45:06.36 server  Timestamp for database 'sales_db' is
(0x0000, 0x00002aa9).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested). CREATE DATABASE: allocating 1536 logical pages (3.0
megabytes) on disk 'sales_db_log1' (1536 logical pages requested).
Warning: The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'. The
recovery of this database can not be guaranteed.
Database 'sales_db' is now online.

1> sp_dboption sales_db,'full logging for all',true
2> go
Database option 'full logging for all' turned ON for database 'sales_db'.
Running CHECKPOINT on database 'sales_db' for option 'full logging for all' to take
effect.
(return status = 0)

1> use master
2> go
1> dump database sales_db to "/tmp/sales_db.dmp"
2> go
Backup Server session id is: 120. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /tmp/sales_db.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db11137014BC' section number 1 mounted on
disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db: 852 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
```

```
Backup Server: 4.188.1.1: Database sales_db: 856 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database sales_db: 860 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> alter database sales_db log on sales_db_log2=10
2> go
Extending database by 5120 pages (10.0 megabytes) on disk sales_db_log2
Warning: The database 'sales_db' is using an unsafe virtual device 'sales_db_dev'. The
recovery of this database can not be guaranteed.
Warning: Using ALTER DATABASE to extend the log segment will cause user thresholds on the
log segment within 128 pages of the last chance threshold to be disabled.

use sales_db
go
select * into bigtab2 from bigtab
go
(20000 rows affected)

1> dump tran sales_db to "/tmp/sales_db.trn1"
2> go
Backup Server session id is: 9. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /tmp/sales_db.trn1.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D37' section number 1 mounted on
disk file '/tmp/sales_db.trn1'
Backup Server: 4.58.1.1: Database sales_db: 250 kilobytes DUMPED.
Backup Server: 4.58.1.1: Database sales_db: 254 kilobytes DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database sales_db: 258 kilobytes DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> use master
2> go
1> alter database sales_db log off sales_db_log2
2> go
Removing 5120 pages (10.0 MB) from disk 'sales_db_log2' in database 'sales_db'.

1> dump tran sales_db to "/tmp/sales_db.trn2"
2> go
Backup Server session id is: 11. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /tmp/sales_db.trn2.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
Backup Server: 4.58.1.1: Database sales_db: 6 kilobytes DUMPED.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.58.1.1: Database sales_db: 10 kilobytes DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db).

1> load tran sales_db from "/tmp/sales_db.trn2" with headeronly
2> go
Backup Server session id is: 13. Use this value when executing the 'sp_volchanged' system
```

```
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
This is a log dump of database ID 5, name 'sales_db', from May 19 2011  4:22AM.
ASE version: lite_670673-1/Adaptive Server Enterprise/15.7.0/EBF 19186 SMP GA
FS3b/B/x86_64/Enterprise Linux/asea. Backup Server version: Backup
Server/15.7/EBF 19186 Drop#3B Prelim/B/Linux AMD Opteron/Enterprise
Linux/aseasap/3556/64-bi. Database page size is 2048.
Log begins on page 1986; checkpoint RID=Rid pageid = 0x7c2; row num = 0x14;
previous BEGIN XACT RID=(Rid pageid = 0x7c2; row num = 0x4); sequence dates:
(old=May 19 2011  4:21:11:356AM, new=May 19 2011  4:22:31:043AM); truncation
page=1986; 123 pages deallocated; requires database with 8192 pages.
Database log version=7; database upgrade version=35; database
durability=UNDEFINED.
segmap: 0x00000003 lstart=0 vstart=[vpgdevno=1 vpvpn=0] lsize=1536 unrsvd=192
segmap: 0x00000004 lstart=1536 vstart=[vpgdevno=2 vpvpn=0] lsize=1536
unrsvd=1530
Unavailable disk fragment: lstart=3072 lsize=5120
The database contains 8192 logical pages (16 MB) and 3072 physical pages (6MB).

1> create database sales_db1 on sales_db_dev=3 log on sales_db_log1=3,
    sales_db_log2=10
2> go
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_dev'
(1536 logical pages requested).
CREATE DATABASE: allocating 1536 logical pages (3.0 megabytes) on disk 'sales_db_log1'
(1536 logical pages requested).
CREATE DATABASE: allocating 5120 logical pages (10.0 megabytes) on disk 'sales_db_log2'
(5120 logical pages requested).
Warning: The database 'sales_db1' is using an unsafe virtual device 'sales_db_dev'. The
recovery of this database can not be guaranteed.
Database 'sales_db1' is now online.

1> load database sales_db1 from "/tmp/sales_db.dmp"
2> go
Backup Backup Server session id is: 15. Use this value when executing the 'sp_volchanged'
system stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db111390340B' section number 1 mounted on
disk file '/tmp/sales_db.dmp'
Backup Server: 4.188.1.1: Database sales_db1: 6148 kilobytes (37%) LOADED.
Backup Server: 4.188.1.1: Database sales_db1: 6160 kilobytes (100%) LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
All dumped pages have been loaded. ASE is now clearing pages above page 3072, which were
not present in the database just loaded.
ASE has finished clearing database pages.
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1863, 13), first=(1863, 13), last=(1865, 7).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'. The total number of log records to process
is 22.
```

```
Redo pass of recovery has processed 2 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not
bring it online automatically.

1> load tran sales_db1 from "/tmp/sales_db.trn1"
2> go
Backup Server session id is: 17. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D37' section number 1 mounted on
disk file '/tmp/sales_db.trn1'
Backup Server: 4.58.1.1: Database sales_db1: 250 kilobytes LOADED.
Backup Server: 4.58.1.1: Database sales_db1: 258 kilobytes LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1863, 13), first=(1863, 13), last=(1986, 3).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'. The total number of log records to process
is 365.
Redo pass of recovery has processed 8 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> load tran sales_db1 from "/tmp/sales_db.trn2"
2> go
Backup Server session id is: 19. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'ales_db1113903D87' section number 1 mounted on
disk file '/tmp/sales_db.trn2'
Backup Server: 4.58.1.1: Database sales_db1: 10 kilobytes LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database sales_db1).
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1986, 3), first=(1986, 3), last=(1986, 20).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
Completed ANALYSIS pass for database 'sales_db1'.
Started REDO pass for database 'sales_db1'. The total number of log records to process
is 16.
Redo pass of recovery has processed 2 committed and 0 aborted transactions.
Completed REDO pass for database 'sales_db1'.
Use the ONLINE DATABASE command to bring this database online; ASE will not bring it
online automatically.

1> online database sales_db1
2> go
Started estimating recovery log boundaries for database 'sales_db1'.
Database 'sales_db1', checkpoint=(1986, 20), first=(1986, 19), last=(1986, 20).
Completed estimating recovery log boundaries for database 'sales_db1'.
Started ANALYSIS pass for database 'sales_db1'.
```

```
Completed ANALYSIS pass for database 'sales_db1'.
Recovery of database 'sales_db1' will undo incomplete nested top actions.
Started UNDO pass for database 'sales_db1'. The total number of log records to process
is 2.
Undo pass of recovery has processed 1 incomplete transactions.
Completed UNDO pass for database 'sales_db1'.
Database 'sales_db1' is now online.

1> alter database sales_db1 log off sales_db_log2
2> go
Removing 5120 pages (10.0 MB) from disk 'sales_db_log2' in database 'sales_db1'.

1> select * from sysusages where dbid=db_id("sales_db1")
2> go
 dbid  segmap  lstart   size  vstart location unreservedpgs  crdate              vdevno
 ----  ------  -------  ----  ------ -------- -------------- ------------------ ------
    6       3       0   1536    1536        0            192  May 19 2011 4:25AM      1
    6       4    1536   1536    1536        0           1536  May 19 2011 4:25AM      2
    6       0    3072   5120    3072        4           5100  May 19 2011 4:25AM     -6

(3 rows affected)

1> dump database sales_db1 to "/tmp/sales_db1.dmp" with shrink_log
2> go
Backup Server session id is: 22. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 4.41.1.1: Creating new disk file /tmp/sales_db1.dmp.
Backup Server: 6.28.1.1: Dumpfile name 'sales_db11113903EC3' section number 1 mounted
on disk file '/tmp/sales_db1.dmp'
Backup Server: 4.188.1.1: Database sales_db1: 3100 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database sales_db1: 3108 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database sales_db1).

1> select * from sysusages where dbid=db_id("sales_db1")
2> go
 dbid  segmap  lstart   size  vstart location unreservedpgs  crdate              vdevno
 ----  ------  -------  ----  ------ -------- -------------- ------------------ ------
    6       3       0   1536    1536        0            192  May 19 2011  4:25AM      3
    6       4    1536   1536    1536        0           1530  May 19 2011  4:25AM      4

(2 rows affected)
```

# データベース・リカバリのための *for load* オプションの使用方法

データベースを新規作成するとき、通常は、データベース・デバイス内の未使用ページがすべてクリアされます。データベースのサイズと使用しているシステムのスピードに応じて、ページのクリアには数秒から数分かかります。

メディア障害からのリカバリ、またはあるマシンから別のマシンへデータベースを移動する目的で、データベース・ダンプからロードを行うために使用するデータベースには、for load オプションを使用します。for load を使用して create database を実行するとページのクリア処理は省略され、ダンプのロードだけに使用可能なターゲット・データベースが作成されます。

for load を使用してデータベースを作成する場合、データベース・ダンプをロードするまでは、新しいデータベース内で実行できるのは次のコマンドだけです。

- alter database...for load
- drop database
- load database

データベース・ダンプをロードするとき、そのデータベース用の新しいデータベース・デバイスの割り付けは、ダンプされたデータベースでの割り付けと一致する必要があります。領域割り付けの複写については、「第 13 章 ユーザ・データベースのバックアップとリストア」を参照してください。

データベース・ダンプを新しいデータベースにロードした後は、使用できるコマンドについての制約はありません。

# *create database* の *with override* オプションの使用方法

with overide オプションを使用すると、領域が限定されているマシンで、データとは別のデバイス・フラグメント上にログを保持できます。これは推奨される処理方法ではありませんが、特にハード・ディスクの障害が発生した後で、データベースをオンライン状態に戻さなければならないとき、記憶領域が限定されているマシンで使用できるのはこの方法だけです。

この場合もトランザクション・ログのダンプは実行できますが、ログがデータと同じデバイス上にあるので、メディア障害が発生した場合に現在のログにアクセスできなくなります。そのため、リカバリできるのは最後のトランザクション・ログ・ダンプまでとなり、その時点から障害が起きた時点までのトランザクションはすべて失われます。

次は、ログとデータが同じ論理デバイスの別々のフラグメントにある場合の例です。

```
create database littledb
    on diskdev1 = "4M"
    log on diskdev1 = "1M"
    with override
```

作成できる最小データベース・サイズは、model のサイズです。

# データベース所有権の変更

システム管理者は、ユーザ・データベースを作成する場合に、一部の初期作業を行ってからそのデータベースの所有権を別のユーザに渡すことができます。sp_changedbowner を使用すると、データベースの所有権を変更できますが、所有権が変更されるデータベース内でシステム管理者だけが実行できます。構文は次のとおりです。

sp_changedbowner *loginame* [, true ]

次の例では、“albert” というユーザを現在のデータベースの所有者に変更します。

```
sp_changedbowner albert
```

この新しい所有者のログイン名が Adaptive Server に登録されている必要がありますが、そのデータベースのユーザを所有者とすることはできません。また、そのデータベース内でエイリアスを持っていてはなりません。そのような場合は、sp_dropuser または sp_dropalias を実行してからでなければ、データベースの所有権は変更できません (『リファレンス・マニュアル：プロシージャ』を参照)。所有権の変更の詳細については、『システム管理ガイド 第 1 巻』の「第 13 章 Adaptive Server のセキュリティ管理について」を参照してください。

**注意** master データベースの所有権を変更することはできません。このデータベースは、常に “sa” ログインが所有しています。

# データベースの変更

データベースまたはトランザクション・ログのデータが増えて、create database によって割り付けられた領域がすべていっぱいになった場合は、alter database を使用して記憶領域を追加することができます。データベース・オブジェクトまたはトランザクション・ログ、あるいはその両方に領域を追加できます。また、データベースをバックアップからロードするための準備を行うときも、alter database を使用できます。

alter database を使用するパーミッションは、デフォルトではデータベース所有者に付与されています。このパーミッションは、データベースの所有権とともに自動的に譲渡されます。alter database パーミッションは、grant コマンドや revoke コマンドによって変更することはできません。「データベース所有権の変更」(158 ページ) を参照してください。

**注意**　alter database for proxy update を指定すると、プロキシ・データベースのすべてのプロキシ・テーブルが削除されます。

## *alter database* の構文

データベースを拡張し、このときに記憶領域を追加するデバイスを指定するには、alter database を使用します。

最も単純な形式の alter database を実行すると、デフォルトのデータベース・デバイスから、設定されているデフォルトの大きさの領域が追加されます。ログとデータが分かれているデータベースの場合は、追加する領域はデータにだけ使用されます。sp_helpdevice を使用すると、デフォルト・リストのデータベース・デバイス名を調べることができます。

『リファレンス・マニュアル：コマンド』を参照してください。

デフォルトのデータベース・デバイスから領域を newpubs データベースに追加するには、次のように入力します。

```
alter database newpubs
```

on 句と log on 句の働きは、create database でのこれらの句と同様です。デフォルトのデータベース・デバイスまたは他のデータベース・デバイス上の領域の大きさを指定でき、複数のデータベース・デバイスを指定できます。alter database を使用して master データベースを拡張する場合に、拡張できるのはマスタ・デバイス上だけです。指定できる最小増加量は、1MB または 1 アロケーション・ユニットのどちらか大きい方です。

要求したサイズの領域を割り付けることができない場合は、各データベース・デバイスで可能な限りの領域が割り付けられます。最小割り付け量は、1 デバイスあたり 256 論理ページです。alter database が完了すると、割り付けた領域を知らせるメッセージが次のように表示されます。

```
Extending database by 1536 pages on disk pubsdata1
```

このメッセージで、要求した領域が追加されたかどうかを確認してください。

次のコマンドを実行すると、newpubs 用に pubsdata1 上で割り付けられる領域に 2MB が追加され、新しいデバイス pubsdata2 上の 3MB が追加され、ログ用に tranlog 上の 1MB が追加されます。

```
alter database newpubs
on pubsdata1 = "2M", pubsdata2 =" 3M"
log on tranlog
```

---

**注意** alter database コマンドを発行するたびに、master データベースをダンプしてください。

---

with override 句を使用すると、既にデータが存在するデバイスにログ領域を格納するためのデバイス・フラグメントを作成することや、既にログ用に使用されているデバイスにデータ・フラグメントを作成することができます。他の記憶領域オプションがなく、最新のデータを完全にリカバリする必要がない場合にだけ、このオプションを使用してください。

for load を使用できるのは、create database for load を実行した後だけです。ダンプからロードされるデータベースの領域割り付けを再作成するときに使用します。ダンプを新しいデータベースにロードする場合の記憶領域の割り付けの複写方法については、「第 13 章 ユーザ・データベースのバックアップとリストア」を参照してください。

## *drop database* コマンドの使用方法

drop database を使用すると Adaptive Server からデータベースが削除され、その結果、データベースとデータベース内のすべてのオブジェクトが消去されます。また、次の操作も実行されます。

- データベースに割り付けられている記憶領域を解放する
- master データベース内のシステム・テーブルから、そのデータベースへの参照を消去する

データベースを削除できるのは、データベースの所有者だけです。データベースの削除は、master データベースを使用している状態で行う必要があります。ユーザが読み込みまたは書き込みを実行するためにオープンしているデータベースを削除することはできません。

構文は次のとおりです。

drop database *database_name* [, *database_name*]...

1 つの文で複数のデータベースを削除できます。次に例を示します。

```
drop database newpubs, newdb
```

データベース・デバイスそのものを削除するには、その前にそのデータベース・デバイスのデータベースをすべて削除する必要があります。デバイスを削除するコマンドは、sp_dropdevice です。

master データベースが損傷を受けても確実にリカバリできるように、データベースを削除した後は master データベースのダンプを実行してください。

# 領域の割り付けを管理するシステム・テーブル

データベース・デバイス上にデータベースを作成して領域を割り付けるとき、Adaptive Server は、まず sysdatabases 内に新しいデータベースのエントリを作成します。次に、master..sysdevices を調べて、create database で指定されたデバイス名が実際に存在することと、そのデバイス名がデータベース・デバイスであることを確認します。データベース・デバイスが指定されていない場合、または default オプションが使用された場合は、master..sysdevices と master..sysusages を調べて、デフォルトの記憶領域として使用できるすべてのデバイスの空き領域を確認します。この確認作業は、デバイスの名前のアルファベット順に実行されます。

指定の量の記憶領域を確保するときに、領域が連続している必要はありません。データベースの記憶領域が、複数のデータベース・デバイスにまたがることもあります。1 つのデータベースは、複数のデータベース・デバイス上に格納されても、1 つの論理単位として扱われます。

データベースの個々の記憶領域の大きさは、1 アロケーション・ユニット以上でなければなりません。各アロケーション・ユニットの最初のページが、アロケーション・ページです。このページは、他のページとは異なり、データベースのローは格納されませんが、残りのページがどのように使用されているかを示す配列が格納されています。

## *sysusages テーブル*

データベース記憶領域の情報は、master..sysusages テーブルに登録されています。master..sysusages 内の各ローは、特定のデータベースに対応する領域割り付けを示します。したがって、create database または alter database によってディスク領域のフラグメントがデータベースに割り当てられるたびに、sysusages 内にはそのデータベースのローが 1 つ作成されることになります。

Adaptive Server をインストールすると、sysusages には、以下のデータベースが作成されます。

- master (dbid は 1)
- テンポラリ・データベース tempdb (dbid は 2)

- model (dbid は 3)
- sybsystemdb (dbid は 31513)
- sybsystemprocs (dbid は 31514)

Adaptive Server を以前のバージョンからアップグレードした場合は、データベース sybsystemdb と sybsystemprocs に別のデータベース ID が設定されていることがあります。

監査機能をインストールすると、sybsecurity データベースの dbid は 5 になります。

新しいデータベースを作成するときや、既存のデータベースを拡張するときは、新しいデータベース割り付けを表す新しいローが sysusages に追加されます。

次に、5 つのシステム・データベースと 1 つのユーザ・データベースを含む Adaptive Server の sysusages の内容の例を示します。ユーザ・データベースは、`log on` オプションを指定して作成され、alter database を使用して 1 回拡張されています。データベース ID (dbid) は 4 です。

```
select dbid, segmap, lstart, size, vdevno, vstart from
sysusages
order by 1
```

```
dbid     segmap      lstart      size        vdevno      vstart
----     ---------   ----------  --------    --------    -----------
1        7           0           6656        0           4
2        7           0           2048        0           8196
3        7           0           1536        0           6660
4        3           0           5120        2           0
4        4           5120        2560        3           0
4        3           7680        5120        2           5120
31513    7           0           1536        0           10244
31514    7           0            63488       1            0
```

この例では、lstart カラムと size カラムには論理ページが示されています。ページのサイズは 2KB ～ 16KB バイトまで変動する可能性があります。vstart カラムには、仮想ページが示されています (サイズは常に 2KB)。次のグローバル変数は、ページ・サイズ情報を示します。

- *@@maxpagesize* － 論理ページ・サイズ
- *@@pagesize* － 仮想ページ・サイズ

次の表には、データベース ID と名前の対応、size カラムで表されたメガバイト数、リスト中の各 vdevno に対する論理デバイス名、各データベースに割り付けられている総メガバイト数の計算値が示されています。出力例では、dbid 4 に対する結果だけが示されており、結果は、読みやすくするため、フォーマットされています。

```
select  dbid, db_name(dbid) as 'database name', lstart,
     size / (power(2,20)/@@maxpagesize) as 'MB',
     d.name
from sysusages u, sysdevices d
where u.vdevno = d.vdevno
and d.status & 2 = 2
order by 1
compute sum(size / (power(2,20)/@@maxpagesize)) by dbid
dbid     database name     lstart      MB          device name
------    --------------    --------    --------    --------------------
4         test              0           10          datadev
4         test              5120        5           logdev
4         test              7680        10          datadev

Compute Result:
-----------
25
```

次に示すのは、セグメントを追加したときの sysusages テーブルの segmap 値に対する変更です。sysusages テーブルからの次の出力で示されているように、この例のサーバには、最初、デフォルトのデータベース、testdb という名前のユーザ・データベース (データのみのデータベース)、および testlog デバイス上のログがあります。

```
select dbid, segmap from master..sysusages where dbid = 6
 dbid    segmap
------  -----------
  6          3
  6           4
```

ユーザ・セグメント newseg をテスト・データベースに追加し、newseg 上にテーブル abcd を作成して、再び sysusages からセグメント情報を選択すると、次のようになります。

```
sp_addsegment newseg, testdb, datadev

create table abcd ( int c1 ) on newseg

select dbid, segmap from sysusages
where dbid=6
dbid     segmap
------   -----------
6           11
6            4
```

ユーザ・データベースに対するセグメント・マッピングの値が 3 から 11 に変化しています。これは、ユーザ・データベースに対するセグメント・マッピングがデータベースを再設定すると変化することを示しています。

セグメントのステータスを確認するには、次のコマンドを実行します。

```
sp_helpsegment
 segment     name            status
-------    ---------------- ----------
0           system              0
1           default             1
2           logsegment          0
3           newseg              0
```

セグメント newseg は、デフォルト・プールの一部ではありません。

別のセグメント newseg1 を testdb データベースに追加し、もう一度 sysusages からセグメント情報を選択すると、newseg に対するセグメント・マッピングが 11 から 27 に変化しています。

```
sp_addsegment newseg1, testdb, datadev

select dbid, segmap from sysusages
 dbid    segmap
------  -----------
6            27
6              4
```

### *segmap* カラム

segmap は、表されているデータベース・フラグメントに対して許可されている記憶領域を示します。このマスクのビット値を制御するには、セグメント管理用のストアド・プロシージャを使用します。マスク内の有効なビットの番号は、ローカル・データベースの syssegments から生成されます (ローカル・データベースは、現在使用しているデータベースであり、ログインの際のデフォルト・データベースまたは use database で最後に使用したデータベースのいずれかです)。

Adaptive Server では、3 つの名前付きセグメントが提供されます。

- system (セグメント 0)
- default (セグメント 1)
- logsegment (セグメント 2)

追加セグメントを作成するには、sp_addsegment を使用します。model データベースにセグメントを作成すると、それ以降に作成するすべてのデータベースにそのセグメントが存在することになります。他のデータベースに作成したセグメントは、そのデータベースのみに存在します。異なるデータベース内の異なるセグメント名に対しては、同じセグメント番号を付けることができます。たとえば、データベース testdb の newseg1 と、データベース mydb の mysegment の両方に、同じセグメント番号 4 を設定できます。

segmap カラムは、ユーザ・データベースの syssegments テーブルにある segment カラムにリンクされているビットマスクです。各ユーザ・データベース内の logsegment はセグメント 2 で、これらのユーザ・データベースはログを別のデバイスに持つので、segmap の値は、log on 文で指定されたデバイスの場合は 4 ($2^2$)、システム・セグメント ($2^0 = 1$) とデフォルト・セグメント ($2^1 = 2$) を持つデータ・セグメントの場合は 3 となります。

データまたはログが格納されるセグメントの値の例を次に示します。

| 値 | セグメント |
|---|---|
| 3 | データのみ (システムおよびデフォルトのセグメント) |
| 4 | ログのみ |
| 7 | データおよびログ |

8 以上の値は、ユーザ定義のセグメントを示します。segmap カラムの詳細については、「第 9 章 セグメントの作成と使用」の「セグメント・チュートリアル」を参照してください。

次のクエリは、syssegments のセグメントと master..sysusages の segmap の間の接続を示しています。このクエリでは、現在のデータベースに対するセグメント・マッピングがリストされ、syssegments の各セグメント番号が master..sysusages のビット番号になることが示されています。

```
select dbid, lstart, segmap, name as 'segment name'
from syssegments s, master..sysusages u
where u.segmap & power(2,s.segment) != 0
and dbid = db_id()
order by 1,2
dbid   lstart    segmap       segment name
-----   ------    --------     --------------
4       0         3           system
4       0         3           default
4       5120      4           logsegment
4       7680      3           system
4       7680      3           default
```

この例は、lstart 値 0 のディスク・フラグメントと lstart 値 7680 のフラグメントが system (セグメント番号 0) と default (セグメント番号 1) を使用するのに対し、lstart 値 5120 のフラグメントが logsegment (セグメント番号 2) を使用することを示しています。このデータベースは、create database の on 句と log on 句の両方を使用して作成された後、alter database の on 句を使用して 1 回だけ拡張されました。

sysusages segmap で使用されているデータ型は int であり、このデータ型は 32 ビットしかないので、データベースが保持できるのは最大 32 セグメント (番号 0 ～ 31) です。segmap は符号付きの値 (正と負の値を表現可能) であり、セグメント 31 は非常に大きい負の値と見なされるので、セグメント 31 を使用するデータベースでこのクエリを使用すると、算術オーバーフローが発生します。

### *lstart* カラム、*size* カラム、および *vstart* カラム

- lstart カラム － データベースのこのアロケーション・ユニットの開始ページ番号を表します。各データベースの先頭は論理アドレス 0 です。データベースに割り付けが追加されると、dbid 7 と同様に、lstart カラムにも割り付けの追加が反映されます。
- size カラム － 同じデータベースに割り当てられた連続するページの数です。データベースのこの部分の終了論理アドレスを調べるには、lstart の値に size の値を加算します。
- vstart カラム － このデータベースに割り当てられた領域の先頭アドレスです。
- vdevno － そのデータベース・フラグメントが存在するデバイスです。

## データベース記憶領域に関する情報の取得

この項では、現在どのデータベース・デバイスがデータベースに割り付けられているかと、それぞれのデータベースがどれくらいの大きさの領域を使用しているかを調べる方法について説明します。

## データベース・デバイス名とオプション

特定のデータベースが存在するデータベース・デバイスの名前を調べるには、次のように、データベース名を引数として sp_helpdb を実行します。

```
sp_helpdb pubs2

name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa            4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data

device_fragments    size          usage         created      free kbytes
------------------- ------------- ------------- -----------  ------------
master               10.0MB        data and log  Apr 13 2005          1792
pubs_2_dev          10.0MB        data and log  Apr 13 2005         9888

device                 segment
---------------------- ----------------------
master                 default
master                 logsegment
master                 system
pubs_2_dev             default
pubs_2_dev             logsegment
pubs_2_dev             system
pubs_2_dev             seg1
pubs_2_dev             seg2
```

sp_helpdb は、指定のデータベースで使用されているデバイスのサイズと使用状況を表示します。status カラムには、データベース・オプションが表示されます。これらのオプションについては、『システム管理ガイド 第 1 巻』の「第 8 章 データベース・オプションの設定」を参照してください。

現在自分が使用しているデータベースを指定して、sp_helpdb を実行すると、データベース内のセグメントと、そのセグメントによって指定されているデバイスも表示されます。「第 9 章 セグメントの作成と使用」を参照してください。

次のように引数を付けずに sp_helpdb を実行すると、Adaptive Server 内のすべてのデータベースの情報が表示されます。

```
sp_helpdb
name            db_size  owner dbid   created            status
--------------  -------- ----- ----   -------------      --------------------
master          48.0 MB   sa      1   Apr 12, 2005       mixed log and data
model            8.0 MB   sa      3   Apr 12, 2005       mixed log and data
pubs2           20.0 MB   sa      6   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
sybsystemdb      8.0 MB   sa      5   Apr 12, 2005       mixed log and data
sybsystemprocs 112.0 MB   sa      4   Apr 12, 2005       trunc log on chkpt,
     mixed log and data
tempdb           8.0 MB   sa      2   Apr 12, 2005       select into/
     bulkcopy/pllsort, trunc log on chkpt, mixed log and data
```

## 領域使用量の確認

sp_spaceused は、以下の使用領域に関する要約情報をレポートします。

- データベース。
- テーブルとそのインデックスおよび text/image 記憶領域。
- テーブル。インデックスと text/image 記憶領域の情報は別に出力されます。

## データベース内の使用領域の確認

データベースが使用している記憶領域の大きさに関する要約情報を表示するには、そのデータベース内で次のように sp_spaceused を実行します。

```
sp_spaceused
database_name                  database_size
------------------------------ -------------
pubs2                          2.0 MB

reserved      data          index_size      unused
------------- ------------- --------------- --------
1720 KB       536 KB        344 KB          840 KB
```

表 6-2 は、レポートされるカラムの説明です。

**表 6-2: sp_spaceused で表示されるカラム**

| カラム | 説明 |
|---|---|
| database_name | 調査するデータベースの名前 |
| database_size | create database または alter database によってデータベースに割り付けられた領域の合計 |
| reserved | データベース内に作成されたすべてのテーブルとインデックスに割り付けられている領域の合計(データベース内では、1 エクステントつまり 8 ページ単位でデータベース・オブジェクトに領域が割り付けられる) |
| data、index_size | データとインデックスに使用されている合計領域 |
| unused | 既存のテーブルとインデックスによって予約されているが、まだ使用されていない領域の合計 |

unused カラム、index_size カラム、data カラムの値の合計は、reserved カラムの値に一致します。database_size から reserved を差し引くと、予約されていない領域の大きさが得られます。この領域は、新しいオブジェクトを作成するときや、既存のオブジェクトをその予約済みの領域を超えて拡張するときに使用できます。

sp_spaceused を定期的に実行することにより、使用できるデータベース領域の大きさをモニタできます。たとえば、reserved の値が database_size の値に近い場合は、新しいオブジェクトに使用できる領域がほとんどないことを示します。unused の値が小さい場合は、データを追加するための領域もほとんどないことを示します。

## テーブルの概要情報の確認

sp_spaceused には、次のようにテーブル名をパラメータとして指定することもでき、テーブルで使用されている領域の情報を表示することができます。

```
sp_spaceused titles
name   rowtotal reserved  data    index_size unused
------ -------- --------- ------- ---------- -----
titles 18       48 KB     6 KB    4 KB       38 KB
```

rowtotal カラムの値は、そのテーブルに対して select count(*) を実行した結果と異なる場合があります。この相違は、sp_spaceused が組み込み関数 rowcnt を使用して値を計算していることによるものです。この関数は、アロケーション・ページに保管されている値を使用します。ただし、この値は定期的に更新されるものではないので、アクティビティの量が多いテーブルの場合はこの差が大きくなることがあります。update statistics、dbcc checktable、dbcc checkdb を実行するとページあたりロー数の見積もりが更新されます。したがって、rowtotal が最も正確な値を示すのは、これらのコマンドが実行された直後となります。

データベースの変更が頻繁に行われると、トランザクション・ログが急速に大きくなるため、syslogs に対する sp_spaceused を定期的に実行してください。特に、トランザクション・ログが別のデバイス上に配置されていない場合は、データベースの他の部分との間で領域の競合が発生するので、このことが問題になります。

## テーブルとそのインデックスの情報の確認

個々のインデックスが使用する領域の情報を確認するには、次のように入力します。

```
                    sp_spaceused titles, 1
index_name          size       reserved   unused
------------------- ---------- ---------- ----------
titleidind          2 KB       32 KB      24 KB
titleind            2 KB       16 KB      14 KB

name       rowtotal   reserved   data   index_size unused
---------- --------  ---------  ------- ---------- ----------
titles          18      46 KB     6 KB      4 KB      36 KB
```

text/image ページが占有する記憶領域は、テーブルが使用する領域とは別にレポートされます。text および image 記憶領域のオブジェクト名は常に、テーブル名に “t” を加えたものです。

```
                                sp_spaceused blurbs,1
index_name            size        reserved   unused
--------------------- ----------- ---------- ----------
blurbs                0 KB        14 KB      12 KB
tblurbs               14 KB       16 KB      2 KB

name        rowtotal reserved    data    index_size unused
----------- -------- ----------- ------- ---------- ----------
blurbs             6       30 KB    2 KB       14 KB      14 KB
```

## 領域の使用情報を示すシステム・テーブルの問い合わせ

独自のクエリを作成することによって、物理記憶領域についての補足情報を得ることもできます。たとえば、Adaptive Server 上に存在する 2K ブロックの記憶領域の総数を調べるには、次のように sysdevices に対するクエリを実行します。

```
select sum(convert(numeric(20,0), high - low + 1))
from sysdevices
where status & 2 = 2

-----------------
              230224
```

この例では、status カラム (3 行目) の 2 は物理デバイスを示します。high はデバイス上で最も大きい有効な 2KB ブロックであるため、合計に整数値を加算してオーバーフローが発生するのを防ぐため、減算 (1 行目の `high - low`) した値に 1 を加えて真のカウントを取得し、カウントを numeric(20,0) に変換する必要があります。

第 7 章

# データベースのマウントとマウント解除



## 概要

mount コマンドと unmount コマンドを使用して、次の処理を行います。

- 自分が所有するデータベースを簡単に、たとえば SQL スクリプトではなくデータ・ファイルとしてパッケージできます。このような SQL スクリプトの実行に必要なアクション (デバイスやデータベースのセットアップなど) を省略できます。
- データベースの移動 － 移送元 Adaptive Server から移送先 Adaptive Server に一連のデータベースを移動するには、データベースが存在するデバイスを物理的に移動します。
- Adaptive Server を停止せずに、データベースをコピーします。コマンド・ラインからデータベースをコピーするときは、Adaptive Server の外部で処理する必要があり、UNIX の dd や ftp などのコマンドを使用して、1 つまたは複数のデータベース内のすべてのページのバイト単位コピーを作成します。

  isql プロンプトから mount および unmount を実行します。プライマリ Adaptive Server が移送元で、セカンダリ Adaptive Server が移送先です。複数の移送元から 1 つの移送先にデータベースをコピーできるので、quiesce database を利用して、複数のプライマリのデータベース用のスタンバイとして 1 つのセカンダリ Adaptive Server を設定することもできます。

データベースのマウントを解除した後、再度マウントする場合

1 unmount コマンドを使用して、データベースとそのデバイスをサーバから削除します。このコマンドの句で指定したロケーションに、データベースに対するマニフェスト・ファイルが作成されます。マニフェスト・ファイルには、移送元 Adaptive Server でのそのデータベースに関する情報 (データベース・デバイス、サーバ情報、データベース情報など) が書き込まれます。「マニフェスト・ファイル」(173 ページ) を参照してください。

2 移送先 Adaptive Server にデータベースをコピーまたは移動します。

3 mount コマンドを使用して、そのデータベースのデバイスや属性などを追加します。

4 database online を使用して、移送先 Adaptive Server 上でそのデータベースをオンラインにします。サーバを再起動する必要はありません。

mount database および unmount database の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

---

**注意** mount database と unmount database は Cluster Edition でサポートされます。これらのコマンドの使用中にインスタンスのフェールオーバ・リカバリが行われると、コマンドがアボートされる場合があります。この場合、インスタンスのフェールオーバ・リカバリが完了したらコマンドを再発行する必要があります。

---

**警告！** 移送元の Adaptive Server 上でデータベースへのアクセスを許可されているログインのそれぞれについて、同じ suid を持つ対応するログインが移送先の Adaptive Server に存在している必要があります。

パーミッションが変わらないようにするには、移送先 Adaptive Server 上のログイン・マップを移送元 Adaptive Server 上のログイン・マップと同一にする必要があります。

---

## マニフェスト・ファイル

マニフェスト・ファイルは、データベースに関する情報 (データベース・デバイス、サーバ情報、データベース情報など) が格納されているバイナリ・ファイルです。該当のデバイスを占有する一連のデータベースが切り離されていて独立している場合にのみ、マニフェスト・ファイルを作成できます。マニフェスト・ファイルには、次の情報が含まれています。

- サーバ固有またはすべてのデータベースに共通する移送元サーバ情報 (移送元 Adaptive Server のバージョン、アップグレード・バージョン、ページ・サイズ、文字セット、ソート順、マニフェスト・ファイルの作成日など)。
- sysdevices カタログから抽出されたデバイス情報。マニフェスト・ファイルには、最初と最後の仮想ページに関する情報、ステータス、関係するデバイスの論理名と物理名が格納されます。
- sysdatabases カタログから抽出されたデータベース情報。マニフェスト・ファイルには、データベース名、dbid、データベース管理者 (dba) のログイン名、所有者の suid、トランザクション・ログへのポインタ、作成日、sysdatabases のステータス・フィールドからのステータス・ビット、最後の dump tran 実行日、関係するデータベースの diskmap エントリに関する情報が格納されます。

**警告！** マニフェスト・ファイルはバイナリ・ファイルであるため、ファイルの内容の文字変換操作 (ftp など) は、バイナリ・モードで実行しないと、ファイルが破損します。

## データベースのコピーと移動

移動やコピーの操作は、データベース・レベルで行い、Adaptive Server の外部でのアクティビティが必要です。デバイスとデータベースの移動またはコピーを実行するには、移送元 Adaptive Server 上で、そのデバイスとデータベースが物理的移送可能な単位に設定されていることを確認してください。

たとえば、複数のデータベースによって使用されているデバイスがある場合は、そのデータベースをすべて 1 回の操作で移送してください。

データベースをコピーするときは、データベースが存在するデバイスを物理的にコピーすることによって移送元から移送先に一連のデータベースを複製します。一連のデータベースを移送元 Adaptive Server から移送先 Adaptive Server にコピーします。

quiesce database コマンドを使用すると、外部ダンプを行うためのマニフェスト・ファイルを作成するパラメータを指定できます。Adaptive Server 外部のユーティリティまたはコマンド (tar、zip、または UNIX の dd コマンド) を使用して、移送先 Adaptive Server にデータベースを移動またはコピーします。移送先では、同じ外部のコマンドまたはユーティリティを使用してデータを抽出し、Adaptive Server のデバイス上に配置します。

1 つのデバイスが複数のデータベースに使用されている場合は、そのデバイス上のすべてのデータベースを 1 回の操作で移動する必要があります。

移送元 Adaptive Server を初期設定するときは注意が必要です。Adaptive Server は、デバイスが 1 つの単位として移送可能であるかどうかは認識しません。データベースが存在するディスクを切り離すことによって、移動対象外のデータベースのデータが失われないように注意してください。Adaptive Server は、ドライブが単一物理ディスク上のパーティションであるかどうかは識別できません。すべてのデータベースを 1 つにまとめて移動してください。

**警告！** mount コマンドと unmount コマンドを使用するときは、複数のデータベースを指定して一度に移動できます。ただし、1 つのデバイスが複数のデータベースに使用されている場合は、そのデータベースをすべて 1 回の操作で移動する必要があります。移送する一連のデータベースのセットを指定します。これらのデータベースに使用されているデバイスは、コマンドで指定されたデータベース以外の外部のデータベースによって共有されていてはなりません。

## パフォーマンスの考慮事項

データベースを一時的に使用するためにマウントする場合を除き、移送するデータベースのデータベース ID は、移送先 Adaptive Server 上でも同一でなければなりません。その場合、checkalloc を実行してデータベース ID を修正する必要があります。

dbid が変更されると、そのデータベースのすべてのストアド・プロシージャが再コンパイル対象としてマークされます。これにより、送信先でデータベースのリカバリに要する時間が長くなり、プロシージャの最初の実行が遅れます。

# デバイスの検証

移送先 Adaptive Server は、各データベースのデバイス割り付けの境界をスキャンすることによって、マニフェスト・ファイルで指定されているデバイスのセットを検証します。このスキャンによって、マウントされるデバイスが、マニフェスト・ファイルに記述されている割り付けに対応していることを確認します。また、sysusages の各エントリの最初と最後のアロケーション・ページについて、アロケーション・ページ内の dbid をマニフェスト・ファイル内の dbid と比較して検証します。

より厳密なデバイス検査が必要な場合は、mount コマンドで with verify を使用します。これによって、データベース内のすべてのアロケーション・ページの dbid が検証されます。

デバイスのコピーを混同しないよう、十分に注意してください。

たとえば、ディスク dsk1、dsk2、dsk3 のコピーで構成されるデータベース・コピーを作成したとします。8 月に、3 月に作成したデータベースのコピーから dsk1 と dsk2 のコピーをマウントし、6 月に作成したコピーから dsk3 をマウントしようとしたとします。このとき、mount コマンドで with verify を使用した場合でも、アロケーション・ページの検査は合格します。使用しているバージョンに関するデータベース情報が正しくないので、リカバリは失敗します。

ただし、dsk3 にアクセスしなければ、リカバリは失敗しないこともあります。これは、データベースはオンラインになるけれども、データが破損している可能性があることを意味します。

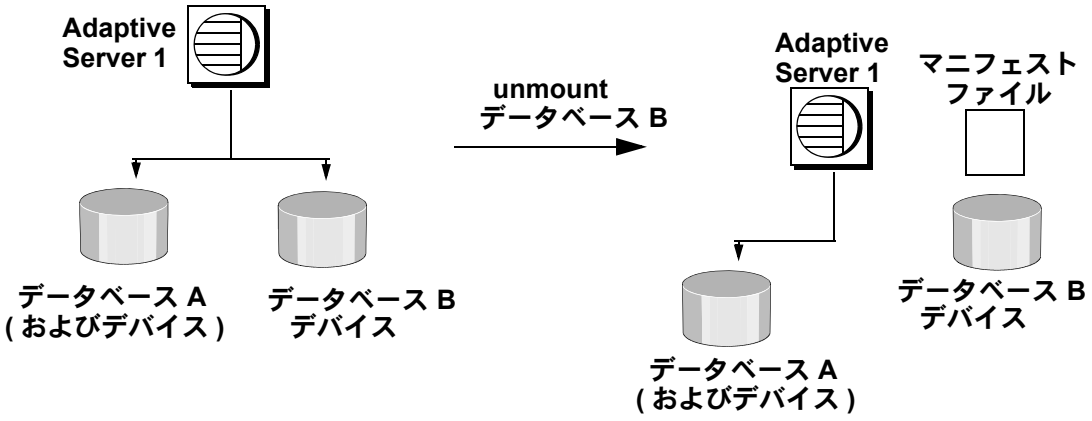
# データベースのマウントとマウント解除

この項では、mount コマンドと unmount コマンドの使用方法について説明します。quiesce database コマンドには、mount コマンドと unmount コマンドを利用しやすくするための句があります。

## データベースのマウントの解除

データベースのマウントを解除すると、データベースとそのデバイスが Adaptive Server から削除されます。unmount コマンドを実行すると、データベースは停止します。そのデータベースを使用するタスクはすべて終了します。データベースとそのページは変更されず、そのオペレーティング・システムのデバイス上に残ります。表 7-1 は、システムからデータベースのマウントを解除した場合の例を示します。

**図 7-1: unmount コマンド**




**注意** unmount コマンドでは、1 回の移動操作に対して複数のデータベースを指定できます。ただし、1 つのデバイスが複数のデータベースに使用されている場合は、そのデータベースをすべて 1 回の操作で移動する必要があります。移送する一連のデータベースのセットを指定します。これらのデータベースに使用されているデバイスは、コマンドで指定されたデータベース以外の外部のデータベースによって共有されていてはなりません。

1 回の unmount コマンドで移動できるデータベースの数は 8 個までです。

unmount コマンドは、次の処理を行います。

- データベースを停止します。
- そのデータベースを Adaptive Server から削除します。
- デバイスを非アクティブ化し、削除します。
- *manifest_file* 句を使用してマニフェスト・ファイルを作成します。

unmount コマンドが完了したら、必要に応じて、移送元 Adaptive Server でデバイスの接続を解除し、移動することができます。

```
unmount database <dbname list> to <manifest_file> [with
{override, [waitfor=<delay time]} ]
```

たとえば、次のように結果が表示されます。

```
unmount database pubs2 to "/work2/Devices/Mpubs2_file"
```

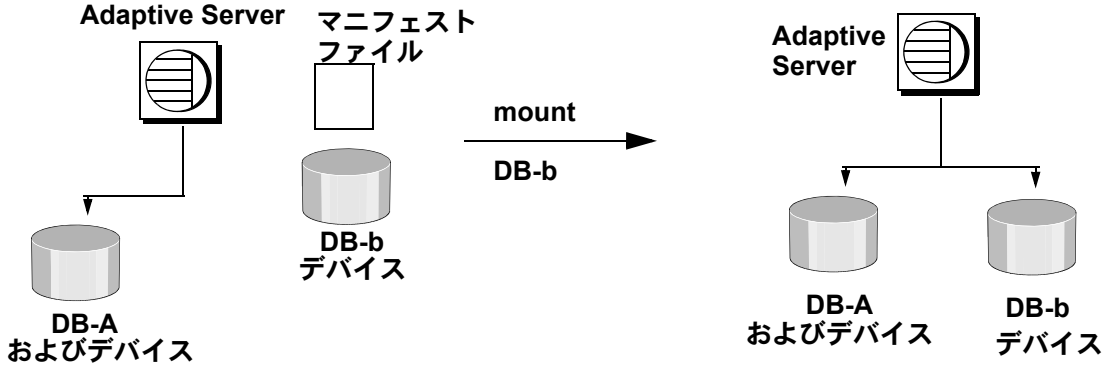
ここで、pubs2 データベースを使用しようとすると、次のエラーが表示されます。

```
Attempt to locate entry in sysdatabases for database 'pubs2'
by name failed - no entry found under that name. Make sure that
name is entered properly.
```

**注意** unmount コマンドで with override を指定することによって参照元データベースが削除されると、参照制約 ( 依存性 ) やテーブルを削除することができなくなります。

## データベースのマウント

**図 7-2: mount コマンド**




mount コマンドを使用して、データベースを移送先またはセカンダリの Adaptive Server に接続します。mount コマンドによって、マニフェスト・ファイル内の情報が復号化され、そのセットのデータベースがオンラインになります。データベース・デバイスの追加 ( 必要な場合 ) とそのアクティブ化、新しいデータベースのカタログ・エントリの作成、データベースのリカバリおよびオンライン化など、必要なサポート・アクティビティがすべて実行されます。

1 つの mount コマンドで指定できるデータベースの数は 8 個に制限されています。

詳細については、『リファレンス・マニュアル：コマンド』の「mount」を参照してください。

---

**注意** mount コマンドでは、1 回の移動操作に対して複数のデータベースを指定できます。ただし、1 つのデバイスが複数のデータベースに使用されている場合は、そのデータベースをすべて 1 回の操作で移動する必要があります。移送する一連のデータベースのセットを指定します。これらのデータベースに使用されているデバイスは、コマンドで指定されたデータベース以外の外部のデータベースによって共有されていてはなりません。

---

mount コマンドは、以下の方法で使用できます。

- 移送先 Adaptive Server で mount コマンドを実行します。次に例を示します。

  ```
  mount database all from "/data/sybase2/mfile1"
  using "/data/sybase1/d0.dbs" = "ldev1"
  ```

  データベースとそのデバイスが移送先 Adaptive Server に登録され、その状態はマウント中となります。システム・カタログが更新され、データベースに関する情報がシステムに登録されますが、データベースそのものはリカバリされていない状態です。ただし、システム障害が発生してもそのデータベースは失われません。

  次に、移送先 Adaptive Server は、データベースを 1 つずつリカバリします。リカバリ後のデータベースはオフラインのままです。

  あるデータベースのリカバリが失敗しても、影響を受けるのはそのデータベースのみです。他のデータベースのリカバリが続行されます。

  database online コマンドを使用してデータベースをオンラインにしてください。

  移送先サーバを再起動する必要はありません。

- mount コマンドの実行時に listonly を指定すると、データベースはマウントされませんが、移送元 Adaptive Server からのマニフェスト・ファイル内のパス名が表示されます。

  移送先 Adaptive Server で、データベースをマウントする前に、listonly パラメータを使用してデバイス・パス名の一覧を表示します。次に例を示します。

  ```
  mount database all from "/data/sybase2/mfile1" with
  listonly

  /data/sybase1/d0.dbs = ldev1
  ```

  次に mount を使用してデータベースを実際にマウントします。パス名が表示されたら、それらを検証または変更して、送信先 Adaptive Server での基準に合わせることができます。

Adaptive Server へのデータベースの mount を実行するときは、以下の点に留意してください。

- マニフェスト・ファイルに記述されている一連のデータベースのサブセットをマウントすることはできません。マニフェスト・ファイルに記述されているデータベースとデバイスは、すべて同時にマウントする必要があります。
- マウントするデータベースのページ・サイズは、前の Adaptive Server のページ・サイズと同じでなければなりません。
- マウントするデータベースに属するすべてのデバイスを正常に追加できるだけの十分なデバイスが、セカンダリ Adaptive Server において設定されている必要があります。
- 設定パラメータ number of devices を正しく設定する必要があります。
- 移送先には、マウントするデータベースと同じ名前のデータベースやデバイスが存在してはなりません。
- Adaptive Server のバージョンは、マウントするデータベースと同じでなければなりません。
- マウントするデータベースは、同じプラットフォームの Adaptive Server で作成されたものでなければなりません。

## データベースのマウント可能コピーの作成

1. manifest 句を指定して quiesce database コマンドを実行し、データベースをクワイス状態にします。このコマンドは、そのデータベースを記述するマニフェスト・ファイルを作成します。
2. listonly を指定して mount コマンドを実行し、コピー対象デバイスのリストを表示します。
3. cp、dd、split mirror などの外部コピー・ユーティリティを使用して、データベース・デバイスを別の Adaptive Server にコピーします。

   デバイスのコピーとマニフェスト・ファイルを合わせて、データベースのマウント可能コピーと呼びます。

## Adaptive Server 間のデータベースの移動

1 unmount コマンドを実行して、元の Adaptive Server からデータベースのマウントを解除します。そのデータベースを記述するマニフェスト・ファイルが作成されます。

2 データベース・デバイスが、まだセカンダリ Adaptive Server で利用可能な状態でない場合は、利用できるようにします。セカンダリ Adaptive Server が別のマシン上にある場合は、オペレーティング・システム管理者の支援が必要になることがあります。

3 手順 1 で作成したマニフェスト・ファイルを使用して、セカンダリ Adaptive Server 上で mount コマンドを実行します。

## システムの制約事項

- システム・データベースのマウントは解除できません。ただし、sybsystemprocs のマウント解除は可能です。
- プロキシ・データベースのマウントは解除できません。
- mount database コマンドと unmount database コマンドは、トランザクション内では使用できません。
- mount database は HA 設定サーバでは実行できません。

## *quiesce database* の拡張機能

データベースを複写またはコピーするには、quiesce database の拡張機能を使用してマニフェスト・ファイルを作成します。quiesce database によって、データベースへの書き込みがブロックされて quiesce hold の状態になり、マニフェスト・ファイルが作成されます。その後、データベースの制御はユーザに返されます。

クワイス ( 静止 ) 状態の一連のデータベースのセットに、そのセットの外にあるデータベースへの参照が含まれている場合は、マニフェスト・ファイルを作成することはできません。この制約を回避するには、with override オプションを使用します。

次に、ユーティリティを使用して、データベースを別の Adaptive Server にコピーします。コピー操作を行うときは、quiesce database hold の次のルールに従ってください。

- コピー操作を始めることができるのは、quiesce database hold プロセスの完了後です。
- quiesce database コマンドで指定したすべてのデータベースに関してすべてのデバイスをコピーします。
- コピー処理が終了してから quiesce database release コマンドを呼び出します。

『リファレンス・マニュアル：コマンド』の「quiesce database」を参照してください。

第 8 章

# 分散トランザクション管理



Adaptive Server には、次に示す分散トランザクション管理機能が導入されました。

- 改良されたトランザクションおよびスレッド管理。Adaptive Server はすべてのトランザクションをサーバ・リソースとして管理し、トランザクションにスレッドを付加および分離する機能を提供します。これらの新しい機能は、ローカル・サーバ・トランザクションのクライアントおよび X/Open XA と MSDTC 機能のクライアントをサポートする共通のインタフェースを提供します。「トランザクション・リソースの設定」(188 ページ) を参照してください。
- 新しい分散トランザクション・コーディネーション・サービス。Adaptive Server は、RPC と CIS を通してリモート Adaptive Server のデータを修正するトランザクションに、一貫したロールバックとコミット機能を提供します。新しいトランザクション・コーディネーション・サービスは、外部トランザクション・マネージャが存在しなくても、そのような分散トランザクションの整合性を保証します。「Adaptive Server コーディネーション・サービスの使用」(191 ページ) を参照してください。
- 準備済みトランザクション用に改良されたリカバリ。リカバリ中、Adaptive Server は、X/Open XA プロトコルと Adaptive Server のネイティブ・トランザクション・コーディネーション・サービスによってコーディネートされた準備済みトランザクションを識別します。Adaptive Server はリカバリ前の状態にこれらのトランザクションをリストアし、以前のバージョンのサーバよりも短時間で、関連するデータベースをオンラインにします。「分散トランザクションにおけるクラッシュのリカバリ手順」(206 ページ) を参照してください。
- 分散トランザクションを自発的に完了する新しい dbcc コマンド。詳細については、「トランザクションの自発的完了」(207 ページ ) を参照してください。

# 影響を受けるトランザクションのタイプ

新しい Adaptive Server の DTM 機能は、次に示すトランザクションに影響を与えます。

- 外部トランザクション・マネージャによってコーディネートされる分散トランザクション
- RPC および CIS を使用してデータを更新するトランザクション

## 外部トランザクション・マネージャによってコーディネートされる分散トランザクション

分散トランザクションは、外部トランザクション・マネージャが X/Open XA などの特定のプロトコルを使用してトランザクションの実行をコーディネートする環境で発生します。Adaptive Server は、Adaptive Server への DTM XA インタフェースを介した CICS、Encina、TUXEDO、MSDTC トランザクション・マネージャを使用して、トランザクションをサポートします。

---

**注意** DTM XA インタフェースを持つ Adaptive Server は、以前 XA-Server 製品の一部であった機能を提供します。現在は、XA-Server 製品ではなく、Adaptive Server のオプションとして提供されています。DTM XA インタフェースの詳細については、『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』を参照してください。

---

### トランザクション・マネージャがコーディネートするトランザクションの動作

Adaptive Server は、XA-Library と XA-Server 製品の一部であったいくつかの機能をネイティブに実装し、X/Open XA プロトコルを通してコーディネートされる準備済みトランザクションに新しいリカバリ手順を提供します。詳細については、「トランザクション・リソースの設定」(188 ページ ) と「分散トランザクションにおけるクラッシュのリカバリ手順」(206 ページ) を参照してください。

Adaptive Server への XA インタフェースは、サーバの新しい分散トランザクション管理機能を実行できるように修正されました。XA インタフェースへの変更は、X/Open XA クライアント・アプリケーションに対して透過的です。しかし、リソース・マネージャとして Adaptive Server を使用するには、X/Open XA トランザクション・マネージャに Adaptive Server の DTM XA インタフェースをリンクする必要があります。すべての XA インタフェースの変更の詳細については、『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』を参照してください。

また、Adaptive Server には、MSDTC によってコーディネートされる分散トランザクションのサポートが導入されています。MSDTC クライアントは、ネイティブ・インタフェースを使用して Adaptive Server に直接通信できます。また、クライアントは DTM XA インタフェースを使用することによって、UNIX で実行している 1 つ以上の Adaptive Server と通信できます。

**注意** DTM XA インタフェースを使用する MSDTC クライアントは、アクセスする Adaptive Server で dtm_tm_role を処理する必要があります。dtm_tm_role の詳細については、『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』を参照してください。

## Adaptive Server バージョン 15.0.3 以降の拡張されたトランザクション・マネージャ

バージョン 15.0.3 よりも前の Adaptive Server では、Adaptive Server が外部トランザクションを暗黙にアボートしたことをアプリケーションが認識しない場合、通常はこのトランザクション内で実行されるはずの DML コマンドが、明示的なトランザクションの外側で実行される可能性があります。これらの DML コマンドは Adaptive Server が起動した暗黙のトランザクション内で実行されるため、ビジネス・データに矛盾が生じることになります。この問題に対応するには、外部トランザクションがアクティブになっているかどうかをユーザ・アプリケーション側で常にチェックし、状況に応じてコマンドを発行する必要があります。

バージョン 15.0.3 以降では、外部トランザクションの暗黙なロールバックが発生した場合、トランザクション・マネージャからの分離要求がないかぎり、Adaptive Server はこの外部トランザクションに関連付けられている接続での DML コマンドの実行を許可しません。分離要求は、コマンドのバッチの終わりで外部トランザクションを実行することを意味します。

15.0.3 以降では、Adaptive Server は、分散トランザクション内で実行されるべき SQL コマンドが、分散トランザクションの外側で実行されないように自動的に制御します。つまり、すべてのコマンドを発行する前に、ユーザ・アプリケーションでグローバル変数 *@@trancount* を確認し、トランザクションが暗黙的にアボートされると、「外部トランザクションがロールバックされたため、コマンドを実行できません」というエラー・メッセージ (3953) が表示されるかを確認する必要がなくなりました。detach transaction コマンドが発行されると、このメッセージの表示が消えます。

3953 エラー・メッセージの表示を消し、Adaptive Server で以前の動作をリストアする (つまり、DTM トランザクションがアクティブでない場合であっても SQL コマンドを実行する) には、トレース・フラグ -T3995 を使用して Adaptive Server を起動します。

## RPC および CIS トランザクション

ローカル Adaptive Server トランザクションは、Transact-SQL リモート・プロシージャ・コール (RPC) およびコンポーネント統合サービス (CIS) を使用することによって、リモート・サーバにあるデータを更新できます。RPC 更新は、ローカルに作成されたトランザクション内から RPC を実行することによって行います。次に例を示します。

```
sp_addserver westcoastsrv, ASEnterprise, hqsales
begin transaction rpc_tran1
update sales set commission=300 where salesid="12OZ"
exec westcoastsrv.salesdb..recordsalesproc
commit rpc_tran1
```

上記のトランザクションは、ローカル Adaptive Server にある sales テーブルを更新しますが、RPC の recordsalesproc を使用してリモート・サーバにあるデータも更新します。

CIS は、これらのテーブルがローカルな場合に、リモート・テーブルにあるデータを更新する方法を提供します。sp_addobjectdef を使用することによって、リモート・データを参照する Adaptive Server にローカル・オブジェクトを作成することができます。ローカル・オブジェクトの更新は、リモート Adaptive Server にあるデータを修正します。次に例を示します。

```
sp_addobjectdef salesrec,
"westcoastsrv.salesdb..sales", "table"
begin transaction cis_tran1
update sales set commission=300 where salesid="12OZ"
update salesrec set commission=300 where salesid="12OZ"
commit cis_tran1
```

## RPC および CIS トランザクションの新しい動作

バージョン 12.0 より前の Adaptive Server では、RPC と CIS を通してデータを更新するトランザクションはリモート・サーバの作業をロールバックできず、またこれらのトランザクションでリモート作業が実際にコミットされることを保証することもできませんでした。Adaptive Server は、新しいトランザクション・コーディネーション・サービスを提供することによって、RPC や CIS 更新が初期トランザクションでの作業のコミットまたはロールバックを保証します。詳細については、「Adaptive Server コーディネーション・サービスの使用」(191 ページ) を参照してください。

RPC と CIS 更新の以前の動作を利用するアプリケーションに対しては、トランザクション・コーディネーション・サービスを無効にすることができます。詳細については、「enable xact coordination パラメータ」(188 ページ) を参照してください。

### SYB2PC トランザクション

SYB2PC トランザクションは、Sybase 2 フェーズ・コミット・プロトコルを使用して、論理単位としてコミットまたはロールバックされる分散トランザクションの作業を保証します。

Adaptive Server は、SYB2PC トランザクションの動作を変更していません。しかし、SYB2PC トランザクションを実装するアプリケーション開発者によっては、代わりに Adaptive Server トランザクション・コーディネーション・サービスの使用を検討したい場合もあります。SYB2PC トランザクションと比較すると、Adaptive Server によって直接コーディネートされるトランザクションは、分散トランザクションの整合性を保証しながら、より少ないネットワーク接続を使用して、より短時間で実行されます。また、アプリケーションではなく Adaptive Server がリモート・トランザクションをコーディネートする場合は、アプリケーション・コードをよりシンプルにすることができます。詳細については、「Adaptive Server コーディネーション・サービスの使用」(191 ページ) を参照してください。

## DTM 機能の有効化

### DTM 機能の有効化

Adaptive Server のライセンスを購入してインストールした後、sp_configure に設定パラメータ enable dtm と enable xact coordination を使用して DTM 機能を有効にできます。

#### *enable dtm パラメータ*

enable dtm パラメータにより、基本的な DTM 機能を有効にしたり無効にしたりできます。enable dtm を 1 (オン) に設定すると、Adaptive Server は MSDTC および DTM XA インタフェースを介した X/Open XA トランザクション・マネージャからの外部トランザクションをサポートします。詳細については、『XA インタフェース統合ガイド for CICS、Encina、TUXEDO』を参照してください。

基本的な DTM 機能を有効にするには、次のコマンドを使用します。

```
sp_configure 'enable dtm', 1
```

この変更を有効にするには、Adaptive Server を再起動します。

### *enable xact coordination* パラメータ

enable xact coordination は、Adaptive Server トランザクション・コーディネーション・サービスを有効にしたり無効にしたりします。このパラメータを有効にすると、Adaptive Server では、リモート Adaptive Server データへの更新は、オリジナルのトランザクションでコミットまたはロールバックするようになります。詳細については、「Adaptive Server コーディネーション・サービスの使用」(191 ページ) を参照してください。

トランザクション・コーディネーションを有効にするには、次のコマンドを使用します。

```
sp_configure 'enable xact coordination', 1
```

この変更を有効にするには、Adaptive Server を再起動します。

## トランザクション・リソースの設定

Adaptive Server は、ローカル・サーバ・トランザクションと分散トランザクション・プロトコルによってコーディネートされる外部トランザクションの両方をサポートする、共通のインタフェースを提供します。分散トランザクション・プロトコルは、X/Open XA、MSDTC、ネイティブ Adaptive Server トランザクション・コーディネーション・サービスによってサポートされます。

Adaptive Server はすべてのトランザクションを設定可能なサーバ・リソースとして管理し、システム管理者は任意のサーバの使用可能リソースの合計数を設定できます。X/Open XA 環境で Adaptive Server にアクセスするクライアント・タスクは、必要に応じてトランザクション・リソースのスレッドを中断したりジョインしたりできます。

この項では、Adaptive Server が利用可能なトランザクション・リソースの合計数を決定したり設定したりする方法について説明します。

### 必要なトランザクション記述子の計算

Adaptive Server は**トランザクション記述子**リソースを使用して、サーバ内でのトランザクションを管理します。トランザクション記述子は内部メモリ構造で、Adaptive Server はこれを使用してトランザクションを表現します。

起動時に、Adaptive Server は設定パラメータ txn to pss ratio の値に基づいて固定数のトランザクション記述子を割り当て、プールに配置します。Adaptive Server は、新しいトランザクションに対して必要になったときに、プールからトランザクション記述子を取得します。トランザクションが完了すると、記述子はプールに返却されます。使用できるトランザクション記述子がない場合は、Adaptive Server が記述子の解放を待つ間、トランザクションは遅延されます。

トランザクション記述子の数を適切に設定するには、グローバル・プールから Adaptive Server が新しい記述子をいつ取得しようとするのか、正確に理解することが重要です。新しいトランザクション記述子は、次のような場合に必要となります。

- クライアント接続が、新しい外部レベルのトランザクションを開始する場合。これは、クライアントが外部レベルの begin transaction コマンドを実行するときに、明示的に発生します。クライアントが begin transaction コマンドを入力せずにデータを修正した場合、暗黙的に発生することもあります。

  外部レベルのトランザクションが開始されると、その後のネストした begin transaction コマンドにはそれ以上のトランザクション記述子は必要ありません。トランザクション記述子の割り付けと割り付け解除は、トランザクションの最も外側のブロックによって指示されます。

- 既存のトランザクションが、2 番目のデータベースを修正した場合 ( マルチデータベース・トランザクション )。マルチデータベース・トランザクションには、アクセスする各データベースに対して、専用トランザクション記述子が必要です。

図 8-1 は、Adaptive Server が異なるタイプのトランザクションに対して、トランザクション記述子を取得し、解放する方法について示しています。

**図 8-1: トランザクション記述子の割り付けおよび割り付け解除**

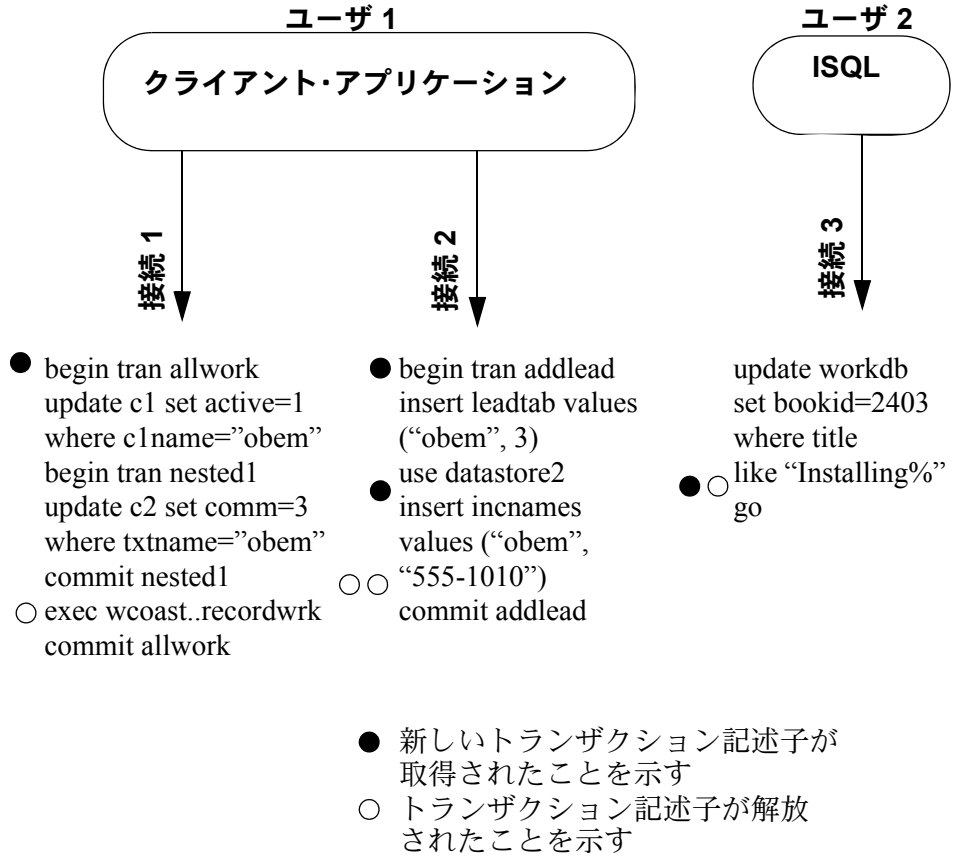



図 8-1 では、Adaptive Server は 1 組のクライアント接続を介してサーバにアクセスするユーザ 1 に対して、合計 3 つのトランザクション記述子を使用しています。サーバは、トランザクション allwork に対して 1 つの記述子を割り付けます。記述子は、トランザクションがコミットされたときに解放されます。ネストされたトランザクション nested1 は、専用トランザクション記述子を必要としません。

トランザクション addlead はマルチデータベース・トランザクションで、2 つのトランザクション記述子を必要とします。1 つは、外部トランザクション・ブロック用で、もう 1 つは 2 番目のデータベース datastore2 の修正用です。このトランザクション記述子は両方とも、外部トランザクション・ブロックがコミットしたときに解放されます。

isql から Adaptive Server にアクセスしているユーザ 2 も、専用トランザクション記述子が必要です。ユーザ 2 は begin transaction を使用して外部トランザクション・ブロックを明示的に作成しませんが、Adaptive Server はトランザクション・ブロックを暗黙的に作成して、update コマンドを実行します。このブロックに関連するトランザクション記述子は、go コマンドの後に取得され、挿入が完了した後に解放されます。

トランザクション記述子は、ほかの Adaptive Server サービスが使用する分のメモリを消費することがあるので、常に、必要となるトランザクションの最大数に足りるだけの記述子を使用することが重要です。

### トランザクション記述子の数の設定

システムで使用するトランザクション記述子の数を決定したら、sp_configure を使用して txn to pss ratio の値を設定します。txn to pss ratio は、サーバで使用できるトランザクション記述子の合計数を決定します。起動時に、この比率に number of user connections パラメータを掛け合わせ、トランザクション記述子プールを作成します。

```
# of transaction descriptors = number of user connections * txn
to pss ratio
```

txn to pss ratio のデフォルト値 16 は、Adaptive Server の以前のバージョンとの互換性を保ちます。バージョン 12.0 より前の Adaptive Server では、各ユーザ接続に対して 16 個のトランザクション記述子を割り付けていました。バージョン 12.0 以降では、同時トランザクションの数は、サーバで利用できるトランザクション記述子の数のみによって制限されます。

たとえば、各ユーザ接続に対して 25 個のトランザクション記述子を割り付けるには、次のコマンドを使用します。

```
sp_configure 'txn to pss ratio', 25
```

この変更を有効にするには、Adaptive Server を再起動します。

## Adaptive Server コーディネーション・サービスの使用

ローカル Adaptive Server トランザクションの作業は、リモート・データを修正するリモート・サーバに分散されることがあります。これは、ローカル・トランザクションが別の Adaptive Server テーブルにあるデータを更新するためにリモート・プロシージャ・コール (RPC) を実行したり、ローカル・トランザクションがコンポーネント統合サービス (CIS) を使用してリモート・テーブルにあるデータを修正する場合に発生します。

## トランザクション・コーディネーション・サービスの概要

バージョン 12.0 より前の Adaptive Server では、RPC を実行したローカル・トランザクションまたは CIS でデータを更新したローカル・トランザクションは、リモート Adaptive Server で行われた作業をロールバックできませんでした。さらに、ローカル・トランザクションを実行するクライアントは、たとえばリモート・サーバがシステム障害を検出したときなどに、リモート作業が実際にコミットされることを保証できませんでした。

Adaptive Server は、リモート・サーバにトランザクションを送信し、すべてのサーバの作業をコーディネートするサービスを提供し、すべての作業が論理単位としてコミットまたはロールバックされることを保証します。トランザクション・コーディネーション・サービスを使用すると、Adaptive Server 自身が複数の Adaptive Server にあるデータを更新するトランザクションの分散トランザクション・マネージャとして動作できます。

### 階層的なトランザクション・コーディネーション

分散トランザクションに関連する別の Adaptive Server はリモートのパティシパントもコーディネートするため、トランザクションは階層的な方法でさらに追加のサーバに送信できます。たとえば図 8-2 では、ASE1 に接続するクライアントは、ASE2 で RPC を実行し、ASE3 で RPC を実行するトランザクションを開始します。ASE1 のコーディネーション・サービスは、ASE2 と ASE3 にトランザクションを送信します。

ASE2 はトランザクション・コーディネーション・サービスも有効なため、追加のリモート・パティシパントにトランザクションを送信できます。ここで、ASE2 は CIS を使用してデータが更新される ASE4 にトランザクションを送信します。

**図 8-2: 階層的なトランザクション・コーディネーション**

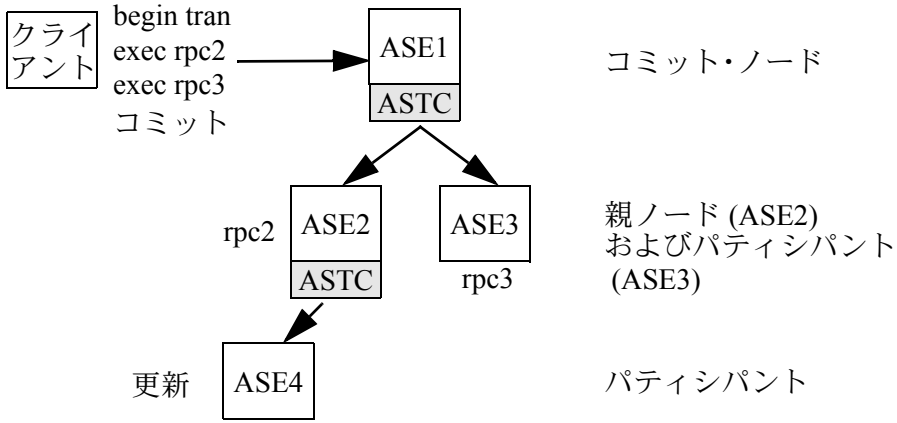


図 8-2 では、ASE1 は分散トランザクションのコミット・ノードとしています。ASE1 のトランザクションがコミットしたとき、ASE1 のコーディネーション・サービスは ASE2 と ASE3 にトランザクションを送信する準備をするよう指示します。ASE3 は、ローカル作業のコミットが準備されたときに、そのトランザクションが準備されることを示します。ASE2 はそのローカル作業を完了する必要があり、ASE3 にそのトランザクションを準備するよう指示します。トランザクションが ASE2 と ASE3 で準備されると、ASE1 のコーディネーション・サービスはオリジナルのトランザクションをコミットします。次に、従属するトランザクションをコミットする指示が ASE2、ASE3、および最終的に ASE3 に転送され、準備のための指示と同じ方法で転送されます。

### DTP 環境における X/Open XA 準拠の動作

X/Open XA プロトコルは、リモート・リソース・マネージャに送信されるトランザクションのコーディネーション・サービスを提供するリソース・マネージャが必要です。この要件は、外部トランザクション・マネージャ (場合によっては、トランザクションが発生するクライアント ) がトランザクションのリモート・サーバへの送信を認識せず、リモート・トランザクションが必要に応じて完了またはアボートすることを保証できないために必要です。

新しいトランザクション・コーディネーション・サービスを使用すると、Adaptive Server が X/Open XA プロトコルに完全に準拠したリソース・マネージャとしての役割を果たすようにすることができます。分散トランザクションは RPC や CIS を使用してリモート・サーバに暗黙的に送信でき、Adaptive Server はコーディネートするリモート・サーバにグローバル・トランザクションのコミットまたはロールバック・ステータスを保存することを保証します。

## 要件と動作

Adaptive Server トランザクション・コーディネーション・サービスは、各リモート Adaptive Server がバージョン 12.0 以降の場合、リモート・サーバの作業が論理的にコミットまたはロールバックされることを保証できます。

トランザクション・コーディネーション・サービスは、分散トランザクションを実行するクライアントに対して透過的です。ローカル・クライアント・トランザクションが RPC を実行したり、CIS を使用してデータを更新したりする場合、コーディネーション・サービスはリモート作業に新しいトランザクション名を作成し、従属するリモート・サーバにそのトランザクションを送信します。ローカル・クライアントがローカル・トランザクションをコミットまたはロールバックすると、Adaptive Server は従属サーバのそれぞれにその要求をコーディネートし、リモート・トランザクションが同じようにコミットまたはロールバックされることを保証します。

Adaptive Server トランザクション・コーディネーション・サービスは、“ASTC HANDLER” という 1 つ以上のバックグラウンド・タスクとして実行され、sp_who を使用して参照できます。複数の Adaptive Server エンジンを使用するシステムでは、“ASTC HANDLER” プロセスの数 (最も近い整数に丸められます) は、次のようになります。

```
number of engines * 2/3
```

Adaptive Server で実行される “ASTC HANDLER” プロセスの最大数は 4 です。

次に示す sp_who からの出力は、1 つの “ASTC HANDLER” を表示します。

```
sp_who
fid spid  status   loginame   origname   hostname           blk_spid   dbname
   tempdbname   cmd              block_xloid    threadpool
--- ----  --------- --------- ---------  -----------------   --------   -----------
   -----------   ----------------- -----------    ------------------
0     1   running       sa        sa         dtmsol1         0       master
        tempdb             SELECT           0    syb_default_pool
0     2  sleeping     NULL     NULL          master         0       master
        tempdb   NETWORK HANDLER           0    syb_default_pool
0     3  sleeping     NULL     NULL                         0       master
        tempdb     DEADLOCK TUNE           0    syb_default_pool
0     4  sleeping     NULL     NULL                         0       master
        tempdb    MIRROR HANDLER           0    syb_default_pool
0     5  sleeping     NULL     NULL                         0       master
        tempdb       HOUSEKEEPER            0    syb_default_pool
0     6  sleeping     NULL     NULL                         0       master
        tempdb  CHECKPOINT SLEEP           0    syb_default_pool
0     7  sleeping     NULL     NULL      metin1_dtm         0  sybsystemdb
        tempdb      ASTC HANDLER           0    syb_default_pool
```

## パティシパント・サーバ・リソースの設定

デフォルトでは、トランザクション・コーディネーション・サービスは常に有効です。システム管理者は、enable xact coordination 設定パラメータを使用してこれらのサービスを有効または無効にできます。このパラメータの完全な説明については、『システム管理ガイド』を参照してください。

また、システム管理者は、Adaptive Server にすべての RPC およびトランザクションによって要求される CIS 更新のコーディネートに必要なリソースがあることを保証する必要があります。トランザクションが RPC または CIS 更新を発行するたびに、トランザクション・コーディネータはフリーの **DTX** パティシパントを取得する必要があります。DTX パティシパントまたは「分散トランザクション・パティシパント」は、Adaptive Server が従属 Adaptive Server に送信されたトランザクションをコーディネートするのに使用する内部メモリ構造体です。図 8-2 では、ASE1 は 3 つのフリー DTX パティシパントが必要で、ASE2 は 2 つのフリー DTX パティシパントが必要です(この場合、1 つの DTX パティシパントは送信されるトランザクションのローカル作業をコーディネートするのに使用されます)。

DTX パティシパント・リソースは、関連するリモート・トランザクションがコミットされるまで、コーディネーティング Adaptive Server によって使用されたままになります。初期トランザクションはすべての従属トランザクションが正常に作業の準備を整えるとすぐにコミットするため、これは一般に、初期トランザクションがコミットされたあとのある時期に発生します。

使用できる DTX パティシパントがない場合、RPC 要求と CIS 更新要求は処理されず、トランザクションがアボートされます。

### *number of dtx participants* パラメータ

システム管理者は、number of dtx participants 設定パラメータを使用して、Adaptive Server で使用できる DTX パティシパントの合計数を設定できます。number of dtx participants は、Adaptive Server トランザクション・コーディネーション・サービスが一度に送信およびコーディネートできるリモート・トランザクションの合計数を設定します。

デフォルトでは、Adaptive Server は 500 のリモート・トランザクションをコーディネートできます。number of dtx participants の設定値を小さくすると、サーバが管理できるリモート・トランザクションの数が減少します。使用できる DTX パティシパントがない場合、新しい分散トランザクションは開始できません。新しいリモート・トランザクションを送信するために使用できる DTX パティシパントがない場合は、進行中の分散トランザクションがアボートすることがあります。

number of dtx participants の設定値を大きくすると、Adaptive Server が処理できるリモート・トランザクション分岐の数が増加しますが、メモリの消費量も増加します。

### 使用しているシステムの *number of dtx participants* の最適化

ピークの時間帯に、sp_monitorconfig を使用して DTX パティシパントの使用状況を調査します。

```
sp_monitorconfig "number of dtx participants"
Usage information at date and time: Jun 18 1999 9:00AM.
Name                          Num_Free   Num_Active  Pct_act   Max_Used
```

```
      Reuse_cnt    Instance_Name
--------------------------   --------   ----------  --------  -----------
      --------  ------------------------
number of dtx participant           480           20      4.00           37
         210                        NULL
participants
```

#Free の値がゼロまたは非常に低い場合、新しい分散トランザクションは DTX パティシパントの不足により、開始できない可能性があります。この場合は、number of dtx participants の値を増やしてください。

#Max Ever Used の値が非常に低い場合、未使用の DTX パティシパントによってメモリが消費されているため、他のサーバ機能がメモリを使用できなくなっている可能性があります。この場合には、number of dtx participants の値を減らしてください。

## 異機種間環境でのトランザクション・コーディネーション・サービスの使用

Adaptive Server が別のバージョン 12.0 以降の Adaptive Server にトランザクションを送信する場合、全体としての分散トランザクションの整合性を保証できます。しかし、ローカル Adaptive Server トランザクションの作業は、バージョン 12.0 以降のトランザクション・コーディネーション・サービスをサポートしないリモート・サーバに分散されることがあります。これは、前の Adaptive Server バージョンでデータの更新に RPC を使用していたり、CIS サービスが Sybase 以外のデータベースでデータを更新するのに使用されている場合に発生します。このような状況では、コーディネーティング Adaptive Server は、リモート・サービスの作業がオリジナル・トランザクションでコミットまたはロールバックされることを保証できません。

### *strict dtm enforcement パラメータ*

Adaptive Server では、システム管理者は strict dtm enforcement 設定パラメータを設定することによって、分散トランザクションを論理単位としてコミットまたはロールバックする要件を強制または緩和することができます。

---

**注意** また、セッション・レベルの set コマンドと strict dtm enforcement オプションを使用して、strict_dtm_enforcement の値を上書きできます。

---

strict dtm enforcement は、Adaptive Server トランザクション・コーディネーション・サービスが分散トランザクションの ACID プロパティを厳密に適用するかどうかを決定します。

strict dtm enforcement を 1 ( オン ) に設定すると、Adaptive Server がコーディネートするトランザクションに参加できるサーバだけに、トランザクションが送信されることを保証します。トランザクション・コーディネーション・サービスをサポートしないサーバにあるデータをトランザクションが更新しようとすると、Adaptive Server はそのトランザクションをアボートします。

異機種間環境では、トランザクション・コーディネーションをサポートしないサーバを使用することがあります。これには、Adaptive Server の古いバージョンや CIS を使用して設定された Sybase 以外のデータベース・ストアも含まれます。これらの状況では、strict dtm enforcement を 0 (オフ) に設定できます。これによって、Adaptive Server は従来の Adaptive Server や他のデータ・ストアにトランザクションを送信できますが、これらのサーバのリモート作業がオリジナル・トランザクションでロールバックまたはコミットされることは保証しません。

## コーディネートされたトランザクションとパティシパントのモニタリング

Adaptive Server は新しいシステム・テーブル sybsystemdbdbo.syscoordinations にあるデータを使用して、従属サーバで行われた作業のステータス情報を追跡します。このテーブルの完全な説明については、『リファレンス・マニュアル：テーブル』を参照してください。

sp_transactions プロシージャも、syscoordinations テーブルから、処理中のリモート・トランザクションについてのいくつかのデータを表示します。詳細については、「分散トランザクション情報の取得」(199 ページ ) を参照してください。

# DTM 管理とトラブルシューティング

## トランザクションと制御スレッド

Adaptive Server バージョン 12.0 より前では、すべてのトランザクション・リソースは 1 つのサーバ・タスクによってプライベートに専有されていました。サーバは、トランザクションを開始したタスク以外のタスクを持つトランザクションを共有できませんでした。

Adaptive Server バージョン 12.5 以降では、Encina や TUXEDO などの X/Open XA 準拠のトランザクション・マネージャが使用する、「サスペンド」と「ジョイン」セマンティックのネイティブ・サポートが利用できます。トランザクションは異なる実行スレッド間で共有したり、関連するスレッドをまったく持たないようにすることもできます。

トランザクションが関連するスレッドを持たない場合、その状態を「分離されている」と表現します。分離されているトランザクションは、spid 値 0 が割り当てられます。トランザクションの spid 値は、新しい master.dbo.systransactions テーブルまたは新しい sp_transactions プロシージャの出力で参照できます。詳細については、「分散トランザクション情報の取得」(199 ページ) を参照してください。

### システム管理者の作業

分離されているトランザクションは、クライアント・アプリケーションがトランザクションにオリジナル・スレッドを再度付加したり、新しいスレッドを付加できるようにするため、Adaptive Server に保持されます。スレッドはトランザクションに付加されていないため、システム管理者は関連する spid を強制終了してトランザクションをロールバックできなくなりました。

分離状態のトランザクションは、dump transaction コマンドを使用してログがトランケートされるのを防ぐことができます。極端な状況では、自発的にトランザクションを完了する新しい dbcc complete_xact コマンドを使用して、分離されたトランザクションをロールバックできます。「トランザクションの自発的完了」(207 ページ) を参照してください。

#### *dtm detach timeout period* パラメータ

システム管理者は、トランザクションが分離されてから Adaptive Server が自動的にロールバックするまでサーバワイドな時間を指定することもできます。dtm detach timeout period は、分散トランザクション分岐を分離状態に維持する時間を、分単位で設定します。この時間が過ぎると、Adaptive Server は分離されたトランザクションをロールバックします。

たとえば、分離の 30 分後に自動的にロールバックするには、次のコマンドを使用します。

```
sp_configure 'dtm detach timeout period', 30
```

## 分離されたトランザクションをサポートするロック・マネージャへの変更点

バージョン 12.0 以前の Adaptive Server では、ロック・マネージャはトランザクション・スレッドの値 spid を使用してトランザクションのロックをユニークに識別できました。新しいトランザクション・マネージャでは、トランザクションはオリジナルのスレッドから分離でき、関連する spid を持ちません。さらに、異なる spid 値を持つ複数のスレッドが同じトランザクション・ロックを共有し、分散トランザクションの作業を実行できなければなりません。

これらの変更を反映するため、Adaptive Server バージョン 12.5 以降のロック・マネージャは、spid ではなく、ユニークなロック所有者 ID を使用してトランザクションのロックを識別します。ロック所有者 ID は、トランザクションを作成した spid とは独立しており、トランザクションがスレッドから分離されても保持されます。ロックの所有者 ID を使って、トランザクションが関連するスレッドを持たない場合、または新しいスレッドがトランザクションに付加される場合に、トランザクションのロックをサポートできます。

ロックの所有者 ID は、master.dbo.syslocks の新しい loid カラムに保管されます。トランザクションの loid 値を確認するには、sp_lock または sp_transactions 出力を調べます。

sp_transactions 出力からの spid と loid カラムを調べると、トランザクションとその制御スレッドについての情報を取得できます。spid 値がゼロの場合は、トランザクションがその制御スレッドから分離されていることを示します。spid 値がゼロ以外の場合は、現在制御スレッドがトランザクションに付加されていることを示します。

sp_transactions 出力の loid 値が偶数のときは、ローカル・トランザクションがロックを所有しています。loid 値が奇数のときは、外部トランザクションがロックを所有していることを示します。

sp_transactions 出力の詳細については、「分散トランザクション情報の取得」(199 ページ) を参照してください。

## 分散トランザクション情報の取得

Adaptive Server には、すべてのサーバ・トランザクションについての情報を保管するシステム・テーブル master.dbo.systransactions があります。systransactions は、各トランザクションを識別し、トランザクションの状態とトランザクションに関連するスレッドについての情報を管理します。

新しいシステム・プロシージャ sp_transactions は、systransactions と syscoordinations テーブルからの情報を変換して、アクティブなトランザクションのステータス状況を表示します。

### *systransactions* のトランザクション識別

Adaptive Server は varchar(255) のカラム (前バージョンでは varchar(64)) にトランザクション名を保管し、異なる分散トランザクション・プロトコルから提供されるトランザクション名の長さとフォーマットを収めます。たとえば、X/Open XA プロトコルでは、分散トランザクションにはグローバル・トランザクション ID (gtrid) と分岐修飾子の両方から構成されるトランザクション名が割り当てられます。Adaptive Server 内では、この情報は systransactions テーブルの xactname カラムに結合されます。

systransactions.xactname は、X/Open XA トランザクション・マネージャまたは MSDTC によって定義される外部作成の分散トランザクションと、ローカル・サーバ・トランザクションの両方の名前を保管します。ローカル・トランザクションを定義するクライアントは、これらのトランザクションに varchar(255) カラムの制限内で任意の名前を付けることができます。同じように、外部トランザクション・マネージャは、さまざまなフォーマットを使用して分散トランザクションの名前を付けられます。

#### トランザクション・キー

トランザクション・キーは systransactions の xactkey カラムに保管され、サーバ・トランザクションへのユニークな内部ハンドルとして動作します。ローカル・トランザクションの場合、xactkey はトランザクション名がサーバに対してユニークでないときも、トランザクションがお互いに区別できることを保証します。

Adaptive Server バージョン 12.0 から、すべてのシステム・テーブルは systransactions.xactkey を参照してユニークにトランザクションを識別します。sysprocesses と syslogshold テーブルは、この規則の唯一の例外です。これらのテーブルは、systransactions.xactname を参照し、varchar(64) (sysprocesses の場合 ) と varchar(67) (syslogshold の場合 ) の長さに値をトランケートし、Adaptive Server の以前のバージョンとの下位互換性を維持します。

### *sp_transactions* を使用したアクティブ・トランザクションの表示

sp_transactions プロシージャは、systransactions と syscoordinations からの情報を変換して、アクティブ・トランザクションについての情報を提供します。キーワードなしで sp_transactions を使用すると、すべてのアクティブ・トランザクションの情報を表示します。

```
                    sp_transactions
xactkey                      type        coordinator starttime
 state             connection dbid   spid   loid
 failover                  srvname                        namelen
xactname
 ------------------------------ ----------- -----------
 --------------------
 ----------------- ---------- ------ ------ -----------
 -------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
 Begun             Attached        1   1      2
 Resident Tx               NULL                                17
$user_transaction
 0x00000b1700040000dd6821390001 Remote      ASTC        Jun 1 1999 3:47PM
 Begun             NA              0   8      0
 Resident Tx               caserv2                          108

 00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-aa01f04ebb9a-
caserv1-caserv1-0002
```

#### ローカル・トランザクション、リモート・トランザクション、外部トランザクションの識別

“type” カラムは、トランザクションがローカル・トランザクション、リモート・トランザクション、外部トランザクションのいずれであるかを示します。ローカル・トランザクションは、ローカル・サーバ (sp_transactions を実行するサーバ ) で実行します。トランザクションは現在のサーバで実行されるため、ローカル・トランザクション用の “srvname” カラムの値は null です。

リモート・トランザクションの場合、sp_transactions は “srvname” カラムにトランザクションを実行するサーバの名前をリストします。前述の sp_transactions 出力は、caserv2 というサーバで実行されているリモート・トランザクションを示しています。

外部トランザクションとは、CICS、Encina、別の Adaptive Server の “ASTC HANDLER” プロセスなど、外部トランザクション・コーディネータによってコーディネートされるトランザクションのことを示します。外部トランザクションも、“srvname” カラムの値は null です。

### トランザクション・コーディネータの識別

“coordinator” カラムは、トランザクションを管理するのに使用されるメソッドまたはプロトコルを示します。前述の出力では、ローカル・トランザクション $user_transaction には外部コーディネータはありません。caserv2 でのリモート・トランザクションのコーディネータの値は “ASTC” です。これは、トランザクションはネイティブ Adaptive Server コーディネーション・サービスを使用してコーディネートされていることを示しています。このサービスについては、「Adaptive Server コーディネーション・サービスの使用」(191 ページ) で説明します。

使用可能なコーディネータ値の全リストと説明については、『リファレンス・マニュアル』の「sp_transactions」を参照してください。

### トランザクションの制御スレッドの表示

spid カラムは、トランザクションに付加されたプロセスのプロセス ID (トランザクションが制御スレッドから分離されている場合は 0) を示します。ローカル・トランザクションの場合、spid 値はローカル・サーバで実行されているプロセス ID を示します。リモート・トランザクションの場合、spid 値は指定したリモート・サーバで実行されているタスクのプロセス ID を示します。前述の出力では、spid 値として、リモート・サーバ caserv2 で実行されているプロセス ID 8 を表示しています。

### トランザクション・ステータス情報の理解

“state” カラムは、各トランザクションの現在のステータス情報を表示します。ローカル・トランザクションまたは外部トランザクションでは、状況によってコマンドの実行、アボート、コミットなどが行われている可能性があります。さらに、分散トランザクションは準備ステータスである場合や、または自発的完了や自発的ロールバックを行っていることもあります。

“connection” カラムは、トランザクションの接続のステータス情報を表示します。この情報を使用して、トランザクションが現在プロセスに付加されているか分離されているかを判断します。X/Open XA 環境のトランザクションは、トランザクション・マネージャからの要求に対応して、それらの開始プロセスから分離されることがあります。

使用可能なコーディネータ値の全リストと説明については、『リファレンス・マニュアル：プロシージャ』の「sp_transactions」を参照してください。

### sp_transactions 出力の特定ステータスへの限定

sp_transactions と state キーワードを使用して、指定したトランザクションのステータスだけに出力を制限することができます。次に例を示します。

```
sp_transactions "state", "Prepared"
```

準備済み分散トランザクションの情報だけを表示します。

### トランザクション・フェールオーバ情報

“failover” カラムは、可用性の高い環境で動作しているサーバの特別な情報を表示します。可用性の高い環境では、オリジナルのサーバで重大な障害が発生した場合、準備済みトランザクションがセカンダリ・コンパニオン・サーバに転換されることがあります。“failover” カラムに表示される、トランザクションを実行している方法や場所を示すフェールオーバ・ステータスには、次の 3 種類があります。

- “Resident Tx” は、通常の動作状態で、Adaptive Server 高可用性機能を利用しないシステムで表示されます。“Resident Tx” は、トランザクションがプライマリ Adaptive Server で起動されて実行中であることを示します。
- Failed-over Tx” は、セカンダリ・コンパニオン・サーバのフェールオーバがあった後で表示されます。“Failed-over Tx” は、トランザクションが最初はプライマリ・サーバで開始され、準備ステータスに達したが、プライマリ・サーバでのシステム障害などのために、自動的にセカンダリ・コンパニオン・サーバにマイグレートされたことを意味します。準備済みトランザクションのマイグレーションは、外部コーディネーティング・サービスに対して透過的に発生します。
- “Tx by Failover-Conn” は、セカンダリ・コンパニオン・サーバにフェールオーバされた後に表示されます。“Tx by Failover-Conn” は、アプリケーションまたはクライアントがプライマリ・サーバでトランザクションを開始したが、プライマリ・サーバが接続フェールオーバによって使用できないことを示します。この状況が発生すると、トランザクションは自動的にセカンダリ・コンパニオン・サーバで開始され、トランザクションは “Tx by Failover-Conn” としてマーク付けされます。

Adaptive Server のフェールオーバ機能の詳細については、「トランザクション・フェールオーバ情報」(203 ページ) を参照してください。

## *sp_transactions* によるコミット・ノードと *gtrid* の指定

sp_transactions に xid キーワードを使用すると、「sp_transactions を使用したアクティブ・トランザクションの表示」(201 ページ) に説明されている出力に加えて、コミット・ノード、親ノード、特定のトランザクションの gtrid が表示されます。この sp_transactions のフォームには、特定のトランザクション名を指定する必要があります。次に例を示します。

```
sp_transactions "xid", "00000b1700040000dd6821390001-aa01f04ebb9a-
00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-caserv1-0002"
xactkey                          type        coordinator starttime
 state             connection dbid   spid   loid
 failover                   srvname                          namelen
xactname
 commit_node parent_node
gtrid
 ------------------------------ ----------- -----------
 --------------------
```

```
 ----------------- ---------- ------ ------ -----------
 ------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 -------------
 0x00000b2500080000dd6821960001 External    ASTC        Jun 1 1999 3:47PM
 Begun             Attached        1      8         139
 Resident Tx               NULL                                108

 00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-aa01f04ebb9a-
caserv1-caserv1-0002

 caserv1 caserv1
 00000b1700040000dd6821390001-aa01f04ebb9a
```

### コミット・ノードと親ノード

Adaptive Server がコーディネートする分散トランザクションの場合、“commit node” カラムは分散トランザクションの最上部の分岐を実行するサーバの名前をリストします。このトランザクションは、トランザクションのすべての分岐のコミットまたはロールバック・ステータスを決定します。詳細については、「階層的なトランザクション・コーディネーション」(192 ページ ) を参照してください。

“parent node” カラムは、トランザクションを開始したサーバの名前をリストします。上記の sp_transactions 出力では、“commit node” カラムと “parent node” カラムには同じサーバ caserv1 がリストされています。これは、分散トランザクションが caserv1 で発生し、caserv1 が現在のサーバにトランザクションの分岐を送信したことを示します。

### グローバル・トランザクション ID

“gtrid” カラムは、Adaptive Server によってコーディネートされた分散トランザクションのグローバル・トランザクション ID を表示します。同じ分散トランザクションの一部であるトランザクション分岐は、同じ gtrid を共有します。特定の gtrid に sp_transactions gtrid キーワードを使用して、現在のサーバで実行されている他のトランザクション分岐のステータスを調べられます。これは、システム管理者が、分散トランザクションの特定の分岐に自発的コミットまたはロールバックを行うべきかどうかを判断するのに役立ちます。sp_transactions と gtrid キーワードの使用例については、「Adaptive Server トランザクションのコミット・ステータスの確認」(210 ページ) を参照してください。

---

**注意** X/Open XA に準拠するトランザクション・マネージャ、MSDTC、または SYB2PC によってコーディネートされるトランザクションの場合、gtrid カラムには、外部コーディネータが指定する完全なトランザクション名が示されます。

---

## 外部トランザクションを実行する手順

すべてのバージョンにおいて、Adaptive Server は外部トランザクションを実行するために次の手順を行います。

1　TM が begin transaction を開始します。

2　TM が attach transaction を開始します。

**注意**　TM は手順 1 と 2 を同時に実行する場合があります。

3　アプリケーションが DML コマンドを実行します。

4　TM が detach transaction を開始します。

5　必要に応じて、手順 2 ～ 4 を繰り返します。

6　トランザクションがロールバックされない場合、TM は prepare transaction を開始します。

7　TM が commit transaction または rollback transaction を開始します。

手順 3 を実行すると、分散トランザクションがロールバックされます。

すべてのコマンドを発行する前にグローバル変数を確認することは面倒であるため、多くのユーザ・アプリケーションはこの確認をまったく行いません。バージョン 15.0.3 以前では、分散トランザクションがロールバックされた場合に、Adaptive Server はユーザ・アプリケーションが引き続き SQL コマンドを発行することを許可していました。これらのコマンドは、独立したトランザクションとして分散トランザクションの外側で実行されました。ロールバック・トランザクションに含める必要のある SQL コマンドが独立してコミットされることで、トランザクションにデータの矛盾が発生しました。

15.0.3 以降では、Adaptive Server は、分散トランザクション内で実行されるべき SQL コマンドが、分散トランザクションの外側で実行されないように自動的に制御します。つまり、すべてのコマンドを発行する前に、ユーザ・アプリケーションでグローバル変数を確認する必要がなくなりました。トランザクションが暗黙的にアボートされると、「外部トランザクションがロールバックされたため、コマンドを実行できません」というエラー・メッセージ (3953) が表示されます。detach transaction コマンドが発行されると、このメッセージの表示が消えます。

3953 エラー・メッセージの表示を消し、Adaptive Server で以前の動作をリストアする (つまり DTM トランザクションがアクティブでない場合であっても SQL コマンドを実行する) には、トレース・フラグ -T3995 を使用して Adaptive Server を起動します。

## 分散トランザクションにおけるクラッシュのリカバリ手順

クラッシュのリカバリ中、Adaptive Server は準備済みステータスで発見された分散トランザクションを解決する必要があります。準備済みトランザクションの解決に使用する方法は、分散トランザクションの管理に使用されたコーディネーション方法およびコーディネーション・プロトコルによって異なります。

---

**注意** 次に示すクラッシュのリカバリ手順は、load database または load transaction コマンドを使用した通常のデータベース・リカバリ中には実行されません。load database または load transaction が準備済みまたは疑わしいトランザクションを適用した場合、Adaptive Server は関連するデータベースをオンラインにする前に、それらのトランザクションをアボートします。

---

### MSDTC によってコーディネートされたトランザクション

MSDTC を使用してコーディネートされている準備済みトランザクションは、マスタ・トランザクションのコミット・ステータスに応じてロールフォワードまたはロールバックされます。リカバリ中、Adaptive Server は MSDTC とのコンタクトを開始し、マスタ・トランザクションのコミット・ステータスを判断して、それに従って準備済みトランザクションをコミットまたはロールバックします。MSDTC とコンタクトできない場合、リカバリ処理はコンタクトができるまで待機します。それ以上のリカバリは、Adaptive Server が MSDTC とのコンタクトが取れるまで実行されません。

### Adaptive Server または X/Open XA によってコーディネートされたトランザクション

クラッシュのリカバリ中、Adaptive Server は、Adaptive Server トランザクション・コーディネーション・サービスまたは X/Open XA プロトコルを使用してコーディネートされた準備済みトランザクションを検出します。これらのトランザクションを検出すると、コーディネートしている Adaptive Server または外部トランザクション・コーディネータがコンタクトを開始し、準備済みトランザクションがコミットまたはロールバックを実行すべきか判定するまで、ローカル・サーバは待機しなければなりません。

リカバリ処理の速度を上げるため、Adaptive Server はこれらの各トランザクションを障害の前の状態に戻します。トランザクション・マネージャはオリジナルのトランザクション ID を使って新しいトランザクションを作成し、ロック・マネージャはロックを適用してオリジナル・トランザクションが修正したデータを保護します。リストアされたトランザクションは準備済みステータスのままですが、関連するスレッドを持たず、分離されます。

トランザクション・コーディネータが Adaptive Server にコンタクトすると、トランザクション・マネージャはトランザクションをコミットまたはロールバックできます。

このリカバリ・メカニズムを使用すると、サーバはコーディネートする Adaptive Server または外部トランザクション・マネージャがまだ準備済みトランザクションを解決していなくても、データベースをオンラインにすることができます。準備済みトランザクションはリカバリの前に実行したロックを保持するため、他のクライアントやトランザクションはローカル・データでの作業を再開できます。準備済みトランザクション自体は、そのコーディネータによってコンタクトが行われると、コミットまたはロールバックを準備します。

制御している Adaptive Server または外部トランザクション・マネージャがトランザクションを完了できない場合、システム管理者はそのロックとトランザクション・リソースを解放するために、トランザクションの自発的完了を実行できます。詳細については、「トランザクションの自発的完了」(207 ページ ) を参照してください。

### SYB2PC によってコーディネートされたトランザクション

SYB2PC プロトコルを使用してコーディネートされた準備済みトランザクションは、マスタ・トランザクションのコミット・ステータスに応じてロールフォワードまたはロールバックされます。リカバリ中、Adaptive Server はコミット・サービスとのコンタクトを開始し、マスタ・トランザクションのコミット・ステータスを判断して、それに従って準備済みトランザクションをコミットまたはロールバックします。コミット・サービスにコンタクトできない場合、Adaptive Server はデータベースをオンラインにしません。しかし、Adaptive Server はシステム内の他のデータベースのリカバリ処理を進めます。

このリカバリ方法は、Adaptive Server の以前のバージョンでの SYB2PC トランザクションで利用されており、Adaptive Server バージョン 12.5 以降でも変更はありません。

## トランザクションの自発的完了

Adaptive Server の dbcc complete_xact コマンドは、トランザクションの自発的完了を行います。dbcc complete_xact はその作業をコミットまたはロールバックすることによってトランザクションを解決し、トランザクションが使用していたリソースを解放します。

dbcc complete_xact は、システム管理者だけが準備済みトランザクションを正しく解決できる場合、またはシステム管理者がトランザクション・コーディネータを待たずにトランザクションを解決しなければならない場合に使用するように提供されています。

たとえば、図 8-2 (192 ページ) に示すように、すべてのリモート Adaptive Server がトランザクションの準備を完了していても、ASE1 へのネットワーク接続が永久に失われた場合に、自発的完了を検討できます。リモート Adaptive Server は、ASE1 からのコーディネーション・サービスによってコンタクトされるまで、それらのトランザクションを準備ステータスのままにします。この場合、ASE2、ASE3、ASE4 のシステム管理者は、準備済みトランザクションを適切に解決できます。ASE3 における準備済みトランザクションの自発的完了は、トランザクションとロック・リソースを解放し、後にトランザクション・コーディネータが使用するために systransactions にコミット・ステータスを記録します。ASE2 におけるトランザクションの自発的完了も、ASE4 に送信されたトランザクションを完了します。

## 準備済みトランザクションの完了

**警告！** 準備済みトランザクションの自発的完了は、分散トランザクション全体において一貫性のない結果となる場合があります。システム管理者がトランザクションを自発的にコミットまたはロールバックすることに決定すると、Adaptive Server またはトランザクション・プロトコルを調整した場合の決定内容と矛盾してしまう可能性があります。

トランザクションの自発的完了を実行する前に、システム管理者はコーディネーティング Adaptive Server またはトランザクション・プロトコルが分散トランザクションのコミットまたはロールバックを決断するかどうかを調べる必要があります ( 詳細については、「Adaptive Server トランザクションのコミット・ステータスの確認」(210 ページ) 参照)。

dbcc complete_xact を使用して、システム管理者は Adaptive Server に分散トランザクションの分岐をコミットまたはロールバックするよう強制します。準備済みトランザクションの自発的完了を実行した後、Adaptive Server は master.dbo.systransactions にトランザクションのコミット・ステータスを記録するので、トランザクションのコーディネータ (Adaptive Server、MSDTC、または X/Open XA トランザクション・マネージャ ) はトランザクションがコミットまたはロールバックされたかどうかを確認できます。

Adaptive Server はコマンドを送信して、トランザクション分岐をコーディネートした対象のサーバへのトランザクションを自発的にコミットするか、またはアボートします。たとえば、図 8-2 (192 ページ ) に示すように、ASE2 でトランザクションに自発的コミットを実行すると、ASE2 は ASE4 にコマンドを送信し、ASE4 でのトランザクションもコミットします。

dbcc complete_xact には、アクティブ・トランザクション名と希望のトランザクション結果を指定する必要があります。たとえば、次のコマンドはトランザクションの自発的コミットを実行します。

```
dbcc complete_xact "00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-aa01f04ebb9a-
```

```
caserv1-caserv1-0002", "commit"
```

### トランザクションの自発的完了の削除

システム管理者が準備済みトランザクションに自発的完了を実行すると、Adaptive Server は master.dbo.systransactions にトランザクションのコミット・ステータスについての情報を保持します。この情報を管理することによって、外部トランザクション・コーディネータは自発的に完了されたトランザクションの存在を検出できます。

外部コーディネータが別の Adaptive Server の場合、サーバはコミット・ステータスを調べ、自発的完了が分散トランザクションのコミット・ステータスと矛盾しているときは警告メッセージをログします。コミット・ステータスを確認したら、コーディネーティング Adaptive Server は systransactions からコミット・ステータス情報をクリアします。

外部コーディネータが X/Open XA 準拠のトランザクション・マネージャの場合、トランザクション・マネージャは自発的完了が分散トランザクションと矛盾していても警告メッセージをログしません。しかし、X/Open XA 準拠のトランザクション・マネージャは、systransactions からコミット・ステータス情報をクリアします。

### コミット・ステータスの手動クリア

dbcc forget_xact は、systransactions から自発的に完了されたトランザクションのコミット・ステータスをパージします。これは、システム管理者がコーディネーティング・サービスにトランザクションが自発的に完了されることを認識させたくない場合、または外部コーディネータが systransactions からの情報をクリアできない場合に使用できます。

dbcc forget_xact の使用方法の詳細については、『リファレンス・マニュアル：コマンド』の「dbcc」を参照してください。

## 準備済みでないトランザクションの完了

dbcc complete_xact は、Adaptive Server によってコーディネートされた準備ステータスになっていないトランザクションをロールバックするのに使用することもできます。コーディネーティング・サーバはトランザクションが作業の準備に失敗したことを認識できるため、準備されていないトランザクションの自発的ロールバックは分散トランザクションに対してリスクにはなりません。このような状況では、コーディネーティング Adaptive Server は一貫性を保つために分散トランザクション全体をロールバックできます。

準備されていない Adaptive Server トランザクションに自発的ロールバックを行う場合、Adaptive Server は systransactions に自発的ロールバックを記録しません。代わりに、情報メッセージが画面に出力され、サーバのエラー・ログに記録されます。

## Adaptive Server トランザクションのコミット・ステータスの確認

コミットまたはロールバックしたい分散トランザクション分岐が Adaptive Server によってコーディネートされているときは、sp_transactions を使用して分散トランザクションのコミット・ステータスを確認できます。これを行うには、次の手順に従います。

**注意** これらの手順は、X/Open XA プロトコル、MSDTC、または SYB2PC によってコーディネートされている分散トランザクションには使用できません。

1 完了するトランザクション分岐を実行しているサーバでは、sp_transactions と xid キーワードを使用して、トランザクションの情報を表示します。コミット・ノードとトランザクションの gtrid を記録します。次に例を示します。

```
sp_transactions "xid", "00000b1700040000dd6821390001-aa01f04ebb9a-
00000b1700040000dd6821390001-aa01f04ebb9a-caserv1-caserv1-0002"
xactkey                         type        coordinator starttime
 state             connection dbid   spid   loid
 failover                   srvname                         namelen
xactname
 commit_node parent_node
gtrid
 ------------------------------ ----------- -----------
 --------------------
 ----------------- ---------- ------ ------ -----------
 ---------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 -------------
 0x00000b2500080000dd6821960001 External    ASTC        Jun 1 1999 3:47PM
 Begun             Attached        1      8         139
 Resident Tx                NULL                                  108

 00000b1700040000dd6821390001-aa01f04ebb9a-00000b1700040000dd6821390001-
aa01f04ebb9a-caserv1-caserv1-0002

 caserv1 sfserv
 00000b1700040000dd6821390001-aa01f04ebb9a
```

この例では、分散トランザクションのコミット・ノードは “caserv1” で、gtrid は “00000b1700040000dd6821390001-aa01f04ebb9a” です。

2 コミット・ノードによって示されたサーバにログオンします。次に例を示します。

```
isql -Usa -Psa_password -Scaserv1
```

3　sp_transactions と gtrid キーワードを使用して、手順 1 で取得した gtrid を持つ分散トランザクションのコミット・ステータスを確認します。

```
sp_transactions "gtrid", "00000b1700040000dd6821390001-aa01f04ebb9a"
xactkey type coordinator starttime
state connection dbid   spid   loid
failover srvname namelen
xactname
commit_node
parent_node
------------------------------ ----------- -----------
--------------------
----------------- ---------- ------ ------ -----------
------------------------- ------------------------------ -------
 ------------------------------------------------------------------
 -------------
 -------------
 0x00000b1700040000dd6821390001 Local       None        Jun 1 1999 3:47PM
 Committed Attached        1      1           2
 Resident Tx                NULL                                17
 $user_transaction

 caserv1
 caserv1
```

この例では、“state” カラムによって示されているように、指定した gtrid を持つローカル・トランザクションはコミットされています。システム管理者は、手順 1 で確認した準備済みトランザクションに自発的コミットを実行する必要があります。

4　システム管理者権限を持つアカウントを使用して、完了させるトランザクション分岐を実行するサーバにログオンします。

```
isql -Usa -Psa_password -Ssfserv
```

5　dbcc complete_xact を使用して、トランザクションをコミットします。この例では、システム管理者は commit キーワードを使用して、分散トランザクションとの一貫性を保つ必要があります。

```
dbcc complete_xact "00000b1700040000dd6821390001-
aa01f04ebb9a-00000b1700040000dd6821390001-aa01f04ebb9a-
caserv1-caserv1-0002", "commit"
```

## プログラミングと設定の考慮事項

この項では、トラブルシューティングを行うときに考慮が必要な設定オプションについて説明します。

## 分散トランザクションでの DDL の動作

X/Open XA プロトコルを使用する外部トランザクション・マネージャ、または別の Adaptive Server の Adaptive Server トランザクション・コーディネーション・サービスによってトランザクションがコーディネートされる場合、DDL コマンドはトランザクション内で使用できません。この動作は、データベース・オプション ddl in tran が有効になっている場合でも適用されます。

## 外部トランザクションでの Adaptive Server の暗黙のロールバック

外部トランザクションでエラー (デッドロック、更新トリガのアボートなど) が発生した場合、Adaptive Server によって外部トランザクションがアボートされることがあります。

Adaptive Server は障害に関するエラー・メッセージを送信しますが、アプリケーションは常にメッセージを確認するわけではありません。特に、DBA によって追加されたトリガのような簡単な挿入は、アプリケーションで認識されない場合があります。XA トランザクションが終了したことが、エラー・メッセージで確認できない場合もあります。

Adaptive Server が外部トランザクションをアボートして SQLException をスローした場合は、`select @@trancount` を発行することができます。*@@trancount* の値がゼロの場合は、DTM トランザクションがアボートされたことを意味します。

アプリケーションはトランザクション・マネージャ (通常はアプリケーション・サーバ) を呼び出して、トランザクションがアボートされたことを知らせる必要があります。エラー・メッセージを無視すると、後続の更新が DTM トランザクションのコンテキスト外 ( たとえばローカル・トランザクション ) で実行される可能性があります。エラー・メッセージのログを取り *@@transtate* または *@@trancount* を調べて、更新が行われたことを確認することができます。

Adaptive Server に外部トランザクションをロールバックさせるトリガを次に示します。失敗する可能性のあるトリガは insert 文に含まれています。Adaptive Server が insert を発行できない場合は、更新で ut.commit 関数が実行されます。

次の例 (疑似コード) は、JTA/XA UserTransaction を実行していることを前提としています。

```
try {

insert into table values (xx.... )
update table
ut.commit();

} catch (SQLException sqe) {

if this is a known error then process
else
select @@trancount into count
if count == 0
then ut.rollback() }
```

rollback 関数を含めない場合、ほかの更新は JTA/XA トランザクション外で行われます。

第 9 章

# セグメントの作成と使用



セグメントによってシステムのパフォーマンスを改善する方法については、『パフォーマンス＆チューニング・シリーズ：物理データベースのチューニング』の「第 1 章 データの物理的配置の制御」を参照してください。

## Adaptive Server セグメント

セグメントとは、1 つまたは複数のデータベース・デバイスを指すラベルです。セグメント名を create table コマンドや create index コマンドで使用することによって、特定のデータベース・デバイス上にテーブルまたはインデックスを配置します。セグメントを使用すると、Adaptive Server のパフォーマンスが向上します。また、システム管理者やデータベース所有者が、データベース・オブジェクトの配置、サイズ、領域量を制御できるようになります。

セグメントはデータベース内に作成し、これによってそのデータベースに割り付けられているデータベース・デバイスの集合を表します。Adaptive Server のデータベースあたりのセグメント数は、システム定義セグメントを含めて最大で 32 個です (システム定義セグメントについては、「システム定義セグメント」(216 ページ) を参照)。disk init でデータベース・デバイスを初期化し、create database または alter database コマンドでそのデータベース・デバイスをデータベースで使用可能な状態にしてから、セグメント名を割り当ててください。

## システム定義セグメント

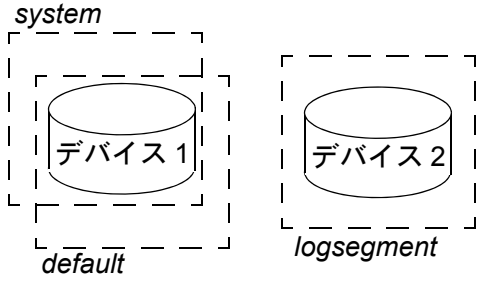
データベースを作成すると、表 9-1 に示す 3 つのセグメントがデータベース内に作成されます。

**表 9-1: システム定義セグメント**

| セグメント | 機能 |
|---|---|
| system | データベースのシステム・テーブルを保管する。 |
| logsegment | データベースのトランザクション・ログを保管する。 |
| default | その他のすべてのデータベース・オブジェクトを保管する (ただし、ユーザが追加のセグメントを作成し、create table...on *segment_name* や create index...on *segment_name* を使ってその新しいセグメントにテーブルやインデックスを保管する場合を除く)。 |

データベースを単一のデータベース・デバイス上に作成する場合、system、default、logsegment の各セグメントは同じデバイス上に作成されます。log on 句を使用してトランザクション・ログを別のデバイス上に置くと、セグメント構造は 図 9-1 のようになります。

**図 9-1: システム定義セグメント**




ユーザ定義セグメントは追加したり削除したりできますが、default、system、logsegment の 3 つのセグメントはデータベースから削除することはできません。データベースには、system、logsegment、および default という各タイプのシステム定義セグメントが少なくとも 1 つずつ必要になります。

以降に、セグメントを管理するためのコマンドとシステム・プロシージャを示します。

- sp_addsegment − データベース内にセグメントを定義します。
- create table と create index − セグメント上にデータベース・オブジェクトを作成します。
- sp_dropsegment − データベースからセグメントを削除します。またはセグメントのスコープから 1 つのデバイスを削除します。
- sp_extendsegment − 既存のセグメントにデバイスを追加します。

- sp_placeobject － 特定のセグメントに、テーブルまたはインデックス・パーティションが使用する将来の領域の割り付けを指定します。
- sp_helpsegment － 特定のセグメント上のデータベースまたはデータ用のセグメント割り付けを表示します。
- sp_helpdb － 各データベース・デバイス上のセグメントを表示します。例については、「第 6 章 ユーザ・データベースの作成と管理」を参照してください。
- sp_help － テーブルについての情報 (そのテーブルが存在するセグメントなど) を表示します。
- sp_helpindex － テーブルのインデックスについての情報 (そのインデックスが存在するセグメントなど) を表示します。

# Adaptive Server がセグメントを使用する方法

新しいデバイスをデータベースに追加すると、そのデバイスは、デフォルトの領域プール (データベースの default セグメントと system セグメント) 内に配置されます。これによって、データベースが使用できる領域の合計は増加しますが、どのオブジェクトをその新しい領域に保管するかを指定することはできません。テーブルやインデックスは、領域のプール全体をいっぱいにするまで拡大する可能性があります。その結果、重要なテーブルを拡張するための領域が残らなくなります。また、使用頻度の高いテーブルとインデックスが、デフォルト領域プール内の単一の物理デバイスに配置されて、その結果 I/O のパフォーマンスが低下することもあります。

オブジェクトをセグメント上に作成すると、そのオブジェクトはセグメント内で利用できるすべてのデータベース・デバイスを使用できますが、それ以外のデバイスは使用できません。セグメントを使用すると、個々のオブジェクトが使用できる領域を制御できます。

以降の各項では、セグメントを使用してディスク領域の使用率を制御し、パフォーマンスを向上させる方法について説明します。

## 領域使用量の制御

重要でないオブジェクトをセグメントに割り当てれば、セグメントのデバイス内で使用できる領域を超えてオブジェクトが拡大することはありません。逆に、重要なテーブルをセグメントに割り当てて、そのセグメントのデバイスが他のセグメントに使用できないように設定されていれば、そのテーブルと他のオブジェクトとの間で領域の競合が発生することはありません。

セグメント内のデバイスに空き領域がなくなったときは、必要に応じてセグメントを拡張し、別のデバイスまたはデバイス・フラグメントを組み込むことができます。また、特定のデータベース・セグメントの領域が残り少なくなったときに、スレッショルドを使って警告を出すこともできます。

データ用に追加セグメントを作成する場合は、セグメントごとに新しいスレッショルド・プロシージャを作成できます。「第 17 章 スレッショルドによる空き領域の管理」を参照してください。

## パフォーマンスの改善

複数のデータベースや複数のドライブで構成される大規模な Adaptive Server 環境では、データベースに対する領域の割り付けと、物理デバイス上のデータベース・オブジェクトの配置を適正に行うことで、システムのパフォーマンスが向上します。個々のデータベースがデータベース・デバイスを独占的に使用する、つまり他のデータベースと物理ディスクを共用しないようにするのが理想的です。通常は、使用頻度の高いデータベース・オブジェクトを専用の物理ディスク上に置くか、容量の大きいテーブルを「分割」して複数の物理ディスクに置くことで、パフォーマンスを改善できます。

### テーブル、インデックス、ログの分離

一般に、テーブルを 1 つの物理デバイス上に置き、そのノンクラスタード・インデックスを 2 番目の物理デバイス上に、トランザクション・ログを 3 番目の物理デバイス上に置くと、パフォーマンスが向上します。別々の物理デバイス (ディスク・コントローラ) を使用することで、ディスクの読み書きに必要な時間を短縮できます。この方法でデバイス全体を専用にできない場合は、少なくともすべてのノンクラスタード・インデックスを専用の物理デバイスに置くようにしてください。

create database の log on 句 (または sp_logdevice) を使用して、別の物理ディスクにトランザクション・ログを配置します。テーブルとインデックスを特定の物理デバイスに置くには、セグメントを使用します。「セグメントへのデータベース・オブジェクトの割り当て」(222 ページ) を参照してください。

### テーブルの分割

テーブルの全般的な読み取りパフォーマンスを向上させるには、使用頻度の高い大きなテーブルを、それぞれ別のディスク・コントローラ上にある複数のデバイスに分割します。大きなテーブルを複数のデバイスに配置すると、複数のディスク上で小さな読み取りが同時に行われる可能性が高くなります。

テーブルを複数のデバイスに分割するには、次の 3 つの方法がありますが、いずれもセグメントを使用します。

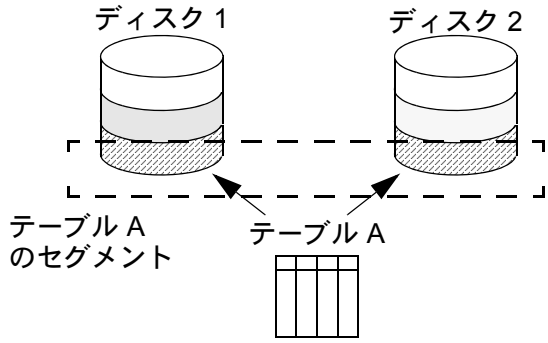
- テーブルを分割します。
- テーブルにクラスタード・インデックスがある場合は、部分的なロードを使用します。
- テーブルに text データ型または image データ型がある場合は、テキスト・チェーンを他のデータから分離します。

## 分割されたテーブル

テーブルを分割すると、そのテーブルに対する複数のページ・チェーンが作成され、これらのページ・チェーンがそのテーブルのセグメント内の全デバイスに分散します。また、挿入時に複数のページ・チェーンが使用できるので、読み取りパフォーマンスと挿入パフォーマンスの両方が向上します。

図 9-2 は、セグメント内の 2 つのデバイスに分割されたテーブルを示しています。

**図 9-2: 複数の物理デバイスへのテーブルの分割**




テーブルを分割するには、指定した数のデバイスを持つセグメント上にテーブルを作成しておく必要があります。alter table を使用してテーブルを分割する方法については、『パフォーマンス＆チューニング・シリーズ：物理データベースのチューニング』の「第 1 章 データの物理的配置の制御」を参照してください。

## 部分的なロード

クラスタード・インデックスを持つテーブルを分割するには、sp_placeobject を実行し、さらに load コマンドを複数回実行することによって、テーブルの各部分をそれぞれ異なるセグメントにロードします。この方法は実行や管理が多少面倒ですが、テーブルとそのクラスタード・インデックスを複数の物理デバイスに分割できます。「セグメントへの既存オブジェクトの配置」(224 ページ) を参照してください。

### text カラムと image カラムの分離

Adaptive Server は、text カラムと image カラムのデータを別のデータ・ページ・チェーンに保管します。このテキスト・チェーンは、デフォルトではテーブルの他のデータと同じセグメントに配置されます。テキスト・カラムの読み込みには、ベース・テーブルでのテキスト・ポインタの読み取りオペレーションに加えて、別のテキスト・チェーンにあるテキスト・ページの読み取りオペレーションも必要です。したがって、テキスト・チェーンとベース・テーブル・データを別の物理デバイスに置くと、パフォーマンスが向上します。「別のデバイスへのテキスト・ページの配置」(227 ページ) を参照してください。

## 別のデバイスへのテーブルの移動

セグメントを使用すると、create clustered index コマンドでテーブルをあるデバイスから別のデバイスに移動することもできます。クラスタード・インデックスは、インデックスの最下位レベル、つまりリーフ・レベルに実際のデータがあるので、テーブルと同じセグメント上にあります。したがって、クラスタード・インデックスがある場合は削除して、目的のセグメント上にクラスタード・インデックスを作成するか作り直すことで、テーブル全体を移動できます。「セグメント上のクラスタード・インデックスの作成」(227 ページ) を参照してください。

# セグメントの作成

データベースにセグメントを作成するには、次の手順に従います。

- disk init を使用して物理デバイスを初期化します。
- create database または alter database に on 句を使用して、そのデータベース・デバイスをデータベースで使用できるようにします。これによって、新しいデバイスはデータベースの default セグメントと system セグメントに自動的に追加されます。

データベース・デバイスが存在し、データベースで使用できる状態になったら、sp_addsegment を使用して、そのデータベースのセグメントを定義してください。

『リファレンス・マニュアル：プロシージャ』を参照してください。

次に、mydisk1 データベース・デバイスに seg_mydisk1 セグメントを作成する文を示します。

```
sp_addsegment seg_mydisk1, mydata, mydisk1
```

# セグメント・スコープの変更

セグメントを使用するには、セグメント・スコープ (各セグメントが指すデータベース・デバイスの数) も管理する必要があります。次の処理を行います。

- セグメントのスコープを拡張するには、そのセグメントが指すデバイスの数を増やします。
- セグメントのスコープを縮小するには、そのセグメントが指すデバイスの数を減らします。

## セグメント・スコープの拡張

セグメントに割り当てられたデータベース・オブジェクトが領域を使いきった場合は、セグメントを拡張する必要があります。sp_extendsegment を使用して、既存のセグメントにデータベース・デバイスを追加します。

セグメントを拡張するには、以下の条件を満たす必要があります。

- 追加するデータベース・デバイスは、sysdevices に登録されている必要があります。
- データベース・デバイスは、拡張するデータベースで使用できる状態でなければなりません。
- 指定するセグメント名が、現在のデータベース内に存在している必要があります。

次の例では、pubs_dev2 というデータベース・デバイスを、bigseg という名前の既存セグメントに追加します。

```
sp_extendsegment bigseg, pubs2, pubs_dev2
```

データベース内の default セグメントを拡張するには、次のように default を引用符で囲んで指定します。

```
sp_extendsegment "default", mydata, newdevice
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

## セグメント・スコープの自動的な拡張

alter database コマンドでデータベース・デバイス上の領域を追加するとき、そのデバイスをそのデータベースで使用するのが初めてである場合は、新しい領域を組み込むように system セグメントと default セグメントが拡張されます。つまり、system セグメントと default セグメントのスコープは、データベースに新しいデバイスが追加されるたびに拡張されます。

alter database コマンドで既存のデータベース・デバイス上の追加領域を割り当てると、既存のデバイスにマップされているすべてのセグメントは、新しいデバイス・フラグメントを組み込むように拡張されます。たとえば、newdev という 4MB のデバイスを初期化して、そのうちの 2MB を mydata データベースに割り付け、その 2MB を testseg セグメントに割り当てるには、次のように入力します。

```
alter database mydata on newdev = "2M"
sp_addsegment testseg, mydata, newdev
```

その後で、newdev 上の残りの領域を使用するために次のように mydata を変更すると、残りの領域フラグメントは自動的に testseg セグメントにマップされます。

```
alter database mydata on newdev = "2M"
```

「セグメント・チュートリアル」(233 ページ) を参照してください。

### セグメント・スコープの縮小

セグメントに含まれているデータベース・デバイスを他のセグメントで排他的に使用できるように予約したい場合は、そのセグメントのスコープを縮小します。たとえば、あるテーブル専用の新しいデータベース・デバイスを追加する場合に、default セグメントと system セグメントのスコープを縮小すれば、このデバイスをこれらのセグメントで使用しないようにすることができます。

sp_dropsegment を使用してセグメントから 1 つのデータベース・デバイスを削除することによって、セグメント・スコープを縮小できます。

sp_dropsegment を実行すると、そのセグメントの配下にあるデバイスのスコープから、指定した device だけが削除されます。また、「セグメントの削除」(228 ページ) で説明するように、sp_dropsegment を使ってセグメント全体をデータベースから削除することもできます。

次の例では、bigseg というスコープから pubs_dev2 というデータベース・デバイスを削除します。

```
sp_dropsegment bigseg, pubs2, pubs_dev2
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

## セグメントへのデータベース・オブジェクトの割り当て

ユーザ定義セグメントに新しいデータベース・オブジェクトまたは既存のデータベース・オブジェクトを割り当てます。この手順は、次の場合に必要になります。

- 新しいオブジェクトの保管を 1 つまたは複数のデータベース・デバイスに限定する場合
- パフォーマンスを向上させるために、テーブルとそのインデックスを別々のデバイスに配置する場合
- 既存のオブジェクトを複数のデータベース・デバイスに分割する場合

## セグメント上の新しいオブジェクトの作成

新しいオブジェクトをセグメント上に置くには、まず新しいセグメントを作成します。目的のデータベース・デバイスだけを指すように、このセグメント (または他のセグメント) のスコープを変更することもできます。新しいデータベース・デバイスをデータベースに追加すると、そのデータベース・デバイスは自動的に default セグメントと system セグメントのスコープに追加されます。

「セグメント上のクラスタード・インデックスの作成」(227 ページ ) を参照してください。

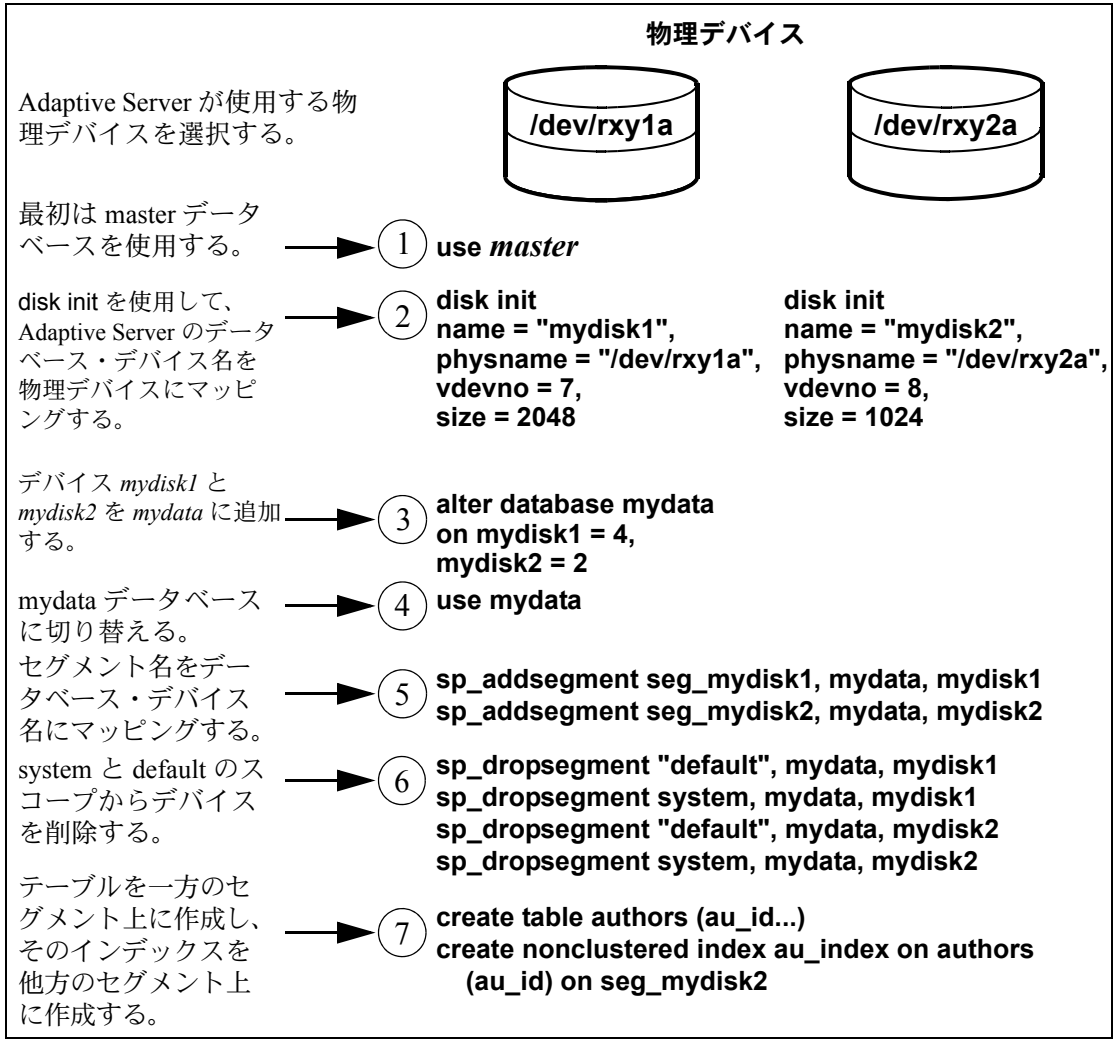
現在のデータベース内にセグメントを定義した後で、create table または create index にオプションの on *segment_name* 句を指定して、そのセグメント上にオブジェクトを作成します。

『リファレンス・マニュアル：プロシージャ』を参照してください。

### 例：別のセグメント上へのテーブルとインデックスの作成

図 9-3 は、2K の論理ページを使用するサーバ上の特定の物理ディスクにテーブルとインデックスを作成するために使用する、Transact-SQL コマンドの手順を要約したものです。

図 9-3: セグメントを使用した特定のデバイスへのオブジェクトの作成




## セグメントへの既存オブジェクトの配置

sp_placeobject では、セグメントへのオブジェクトの割り付けを解除することはできません。ただし、このプロシージャを実行すると、それ以降のそのオブジェクトに対するディスクの割り付けは、指定した新しいセグメント上で行われるようになります。

たとえば、mytab テーブルへのディスクの割り付けをすべて bigseg 上で行うには、次のコマンドを使用します。

```
sp_placeobject bigseg, mytab
```

sp_placeobject では、データベース・デバイス間でのオブジェクトの移動は実行されません。最初のデバイスに割り付けられたページは割り付けられたままです。つまり、最初のデバイスに書き込まれたデータは、そのデバイス上に残ります。sp_placeobject は、それ以降に行われる領域の割り付けだけに作用します。

sp_placeobject の実行後に dbcc checkalloc を実行すると、複数セグメントに分割された各オブジェクトに対して次のメッセージが表示されます。

```
Extent not within segment: Object object_name, indid
index_id includes extents on allocation page page_number
which is not in segment segment_name.
```

このメッセージは無視してもかまいません。

『リファレンス・マニュアル：プロシージャ』を参照してください。

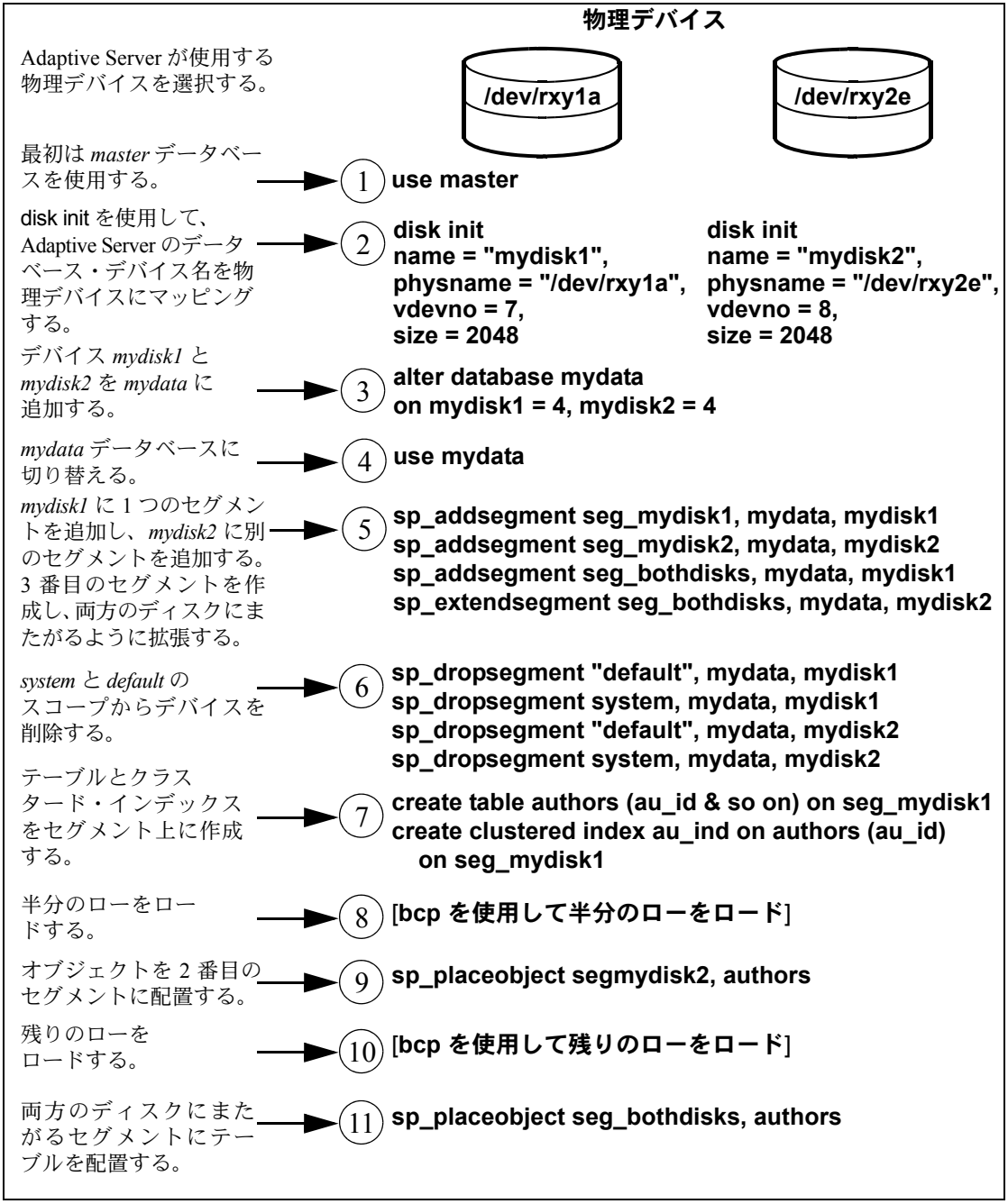
### 例：複数物理デバイスへのテーブルとクラスタード・インデックスの分割

大量のデータを扱うマルチユーザ・アプリケーションでは、別々のディスク・コントローラにある複数のセグメントに大型のテーブルを分割すると、パフォーマンスが向上します。

手順の実行順序が重要になります。特に、クラスタード・インデックスを作成してから、別のセグメントにテーブルを配置してください。

図 9-4 は、2K の論理ページ・サイズを使用しているサーバで、1 つのテーブルを 2 つのセグメントに分割する方法の概要を示します。

**図 9-4: 2 つのセグメントへの大型テーブルの分割**



テーブルが頻繁に更新される場合は、時間が経つにつれてディスク割り付けのバランスが変化する可能性があります。そのため、高速性という利点を保つためには、テーブルを削除して再作成する必要があります。

## 別のデバイスへのテキスト・ページの配置

text カラムまたは image カラムを持つテーブルを作成すると、そのデータは別のテキスト・ページ・チェーンに保管されます。テーブルに text カラムまたは image カラムがある場合は、sysindexes 内にテキスト・チェーンを表すエントリが追加されます。このエントリの名前カラムの値はテーブル名の前に tを付けたもので、indid の値は 255 です。sp_placeobject コマンドを使用してテキスト・チェーンを別のデバイスに配置するとき、次のようにテーブル名と sysindexes に登録されているテキスト・チェーン名の両方を指定します。

```
sp_placeobject textseg, "mytab.tmytab"
```

**注意** デフォルトでは、テキスト・ページのチェーンはそのテーブルと同一のセグメント上に置かれます。sp_placeobject を実行した後は、元のデバイスに以前に書き込んだページは割り付けられたままになりますが、新しい割り付けはすべて新しいセグメント上で行われます。

テキスト・ページを特定のセグメント上に配置するには、テーブルをそのセグメント上に作成してから (最初のエクステントがそのセグメントに割り付けられる )、そのテーブルのクラスタード・インデックスを作成することによって、残りのデータをそのセグメントに移動します。

## セグメント上のクラスタード・インデックスの作成

クラスタード・インデックスの最下位レベル、つまりリーフ・レベルにはデータが入っています。したがって、テーブルとそのクラスタード・インデックスは同一のセグメント上に存在します。1 つのセグメントにテーブルを作成して別のセグメントにクラスタード・インデックスを作成すると、テーブルはクラスタード・インデックスを作成したセグメントに移行されます。これを利用すると、テーブルをデータベース内の別のデバイスに移動する処理を、高速かつ簡単に実行できます。

『リファレンス・マニュアル：コマンド』の「create index」を参照してください。

次の例では、new_space セグメントのテーブルを使用して、セグメント名を指定せずにクラスタード・インデックスを作成します ( このテーブルを作成する手順については、「セグメント・チュートリアル」(233 ページ) を参照してください)。

```
create clustered index mytabl_cix
```

```
    on mytabl(c1)
sp_helpsegment new_space


segment     name           status
-------     ------------- ------
      3     new_space           0

device            size       free_pages
----------------- ------     ----------
newdevice         3.0MB            1523

total_size     total_pages     free_pages  used_pages    reserved_pages
-------------- -----------     ----------- -----------    --------------
3.0MB                  1536            1530           6                  0
```

テーブルをセグメント上に配置した後でクラスタード・インデックスを作成するときは、on *segment_name* 句を使用してください。使用しない場合は、テーブルが default セグメントに移行されます。

# セグメントの削除

セグメント名とデータベース名だけを指定して sp_dropsegment を実行すると、指定したセグメントがデータベースから削除されます。ただし、そのセグメントにデータベース・オブジェクトが割り当てられている場合は、削除できません。最初にそのオブジェクトを別のセグメントに割り当てるか、オブジェクトを削除してから、セグメントを削除してください。

default、system、logsegment をデータベースから完全に削除することはできません。データベースには、default、system、logsegment の 3 つが少なくとも必要です。ただし、これらのセグメントのスコープを縮小することはできます。詳細については、「セグメント・スコープの縮小」(222 ページ) を参照してください。

**注意** セグメントを削除すると、そのセグメント名がデータベースのセグメントのリストから削除されますが、データベースへのデバイスの割り付けが解除されることはなく、デバイスからオブジェクトが削除されることもありません。
データベース・デバイスからすべてのセグメントを削除した場合は、その領域はデータベースに割り付けられたままですが、データベース・オブジェクトの保管に使用することはできなくなります。dbcc checkcatalog を実行すると「sysusages segmap 内にセグメントがありません」と表示されます。デバイスをデータベースで使用できるようにするには、次のように sp_extendsegment を実行して、そのデバイスをデータベースのデフォルト・セグメントにマッピングしてください。

```
sp_extendsegment "default", dbname, devname
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

# セグメント情報の取得

セグメントについての情報を得るには、次のシステム・プロシージャを使います。

- sp_helpsegment － データベースのセグメントのリスト、またはデータベース内の特定セグメントの情報を表示する。
- sp_helpdb － データベース内のデバイスとセグメントの関係についての情報を表示する。
- sp_help と sp_helpindex － テーブルとインデックスについての情報を表示する。これには、オブジェクトが割り当てられているセグメントの情報が含まれる。

## *sp_helpsegment*

sp_helpsegment システム・プロシージャを引数なしで実行すると、このプロシージャを実行したデータベース内の全セグメントの情報が、次のように表示されます。

```
sp_helpsegment
segment name                          status
------- ----------------------------- ------
      0 system                             0
      1 default                            1
      2 logsegment                         0
      3 seg1                               0
      4 seg2                               0
```

特定のセグメントについての情報を得るには、引数としてセグメント名を指定します。default セグメントの情報を表示する場合は、次のように引用符で囲みます。

```
sp_helpsegment "default"
```

次の例では、seg1 についての情報が表示されます。

```
sp_helpsegment seg1

segment name                           status
------- ------------------------------ ------
      4 seg1                                0
```

```
device                 size             free_pages
---------------------- ---------------- -----------
user_data10            15.0MB                  6440
user_data11            15.0MB                  6440
user_data12            15.0MB                  6440

table_name             index_name            indid
---------------------- ---------------------- ------
customer               customer                   0

total_size      total_pages free_pages  used_pages
--------------- ----------- ----------- -----------
45.0MB                23040       19320        3720
```

## *sp_helpdb*

現在使用しているデータベースの名前を指定して sp_helpdb システム・プロシージャを実行すると、データベース内のセグメントについての情報が表示されます。

次に例を示します。

```
sp_helpdb pubs2

name      db_size    owner     dbid created        status
--------- ---------- --------- ---- -------------- --------------
pubs2     20.0 MB     sa           4 Apr 25, 2005   select
     into/bulkcopy/pllsort, trunc log on chkpt, mixed log and data

device_fragments    size          usage         created      free kbytes
------------------- ------------- ------------- ----------   ------------
master              10.0MB        data and log  Apr 13 2005          1792
pubs_2_dev         10.0MB        data and log  Apr 13 2005         9888

device                 segment
---------------------- ----------------------
master                 default
master                 logsegment
master                 system
pubs_2_dev             default
pubs_2_dev             logsegment
pubs_2_dev             system
pubs_2_dev             seg1
pubs_2_dev             seg2
```

## *sp_help と sp_helpindex*

特定のデータベースを使用しているときに、テーブル名を指定して sp_help または sp_helpindex を実行すると、そのテーブルまたはそのインデックスが保管されているセグメントについての情報が表示されます。

次に例を示します。

```
sp_helpindex authors

index_name  index_keys  index_description  index_max_rows_per_page
     index_fillfactor  index_reservepagegap  index_created
     index_local
----------  ----------  -----------------  ----------------------
     ----------------  --------------------  -------------
     -----------
auidind     au_id       clustered,unique                          0
                    0                    0   Apr26 2005  4:04PM
     Global Index
aunwind    au_lname,au_fname  nonclustered,unique               0
                    0                    0   Apr26 2005  4:04PM
     Global Index
(2 rows affected)
index_ptn_name        index_ptn_seg
--------------------  --------------
auidind_400001425     default
aunmind_400001425     default
```

# セグメントとシステム・テーブル

セグメントに関する情報の保管先は、master..sysusages システム・テーブルと、ユーザ・データベース内の 2 つのシステム・テーブル (sysindexes と syssegments) の 3 つです。sp_helpsegment はこれらのテーブルを使用し、sysdevices 内のデータベース・デバイス名を検索します。

create database または alter database を使用してデバイスをデータベースに追加すると、master..sysusages にローが 1 つ追加されます。sysusages の segmap カラムの内容は、各デバイスに対応するデータベース内のセグメントのビットマップです。

create database を実行すると、ユーザ・データベース内に syssegments テーブルも作成され、次のようなデフォルトのエントリが挿入されます。

```
segment name            status
------- --------------- ------
      0 system                0
      1 default               1
      2 logsegment            0
```

sp_addsegment を実行すると、次の処理が実行されます。

- ユーザ・データベースの syssegments テーブルに新しいローを追加する。
- master..sysusages 内の segmap を更新する。

テーブルまたはインデックス・パーティションを作成すると、Adaptive Server によって sysindexes に新しいローが追加されます。そのテーブルの segment カラムには、サーバがオブジェクトの新しい領域を割り付ける場所を示すセグメント番号が保管されます。オブジェクトの作成時にセグメント名を指定しない場合は、オブジェクトは default セグメントに配置されますが、指定した場合はそのセグメントに配置されます。

text カラムまたは image カラムを持つテーブルを作成すると、リンクされたテキスト・ページ・リストを表すローが sysindexes に追加されます。デフォルトでは、テキスト・ページのチェーンはテーブルと同じセグメント上に保管されます。sp_placeobject を使用して、テキスト・チェーンを専用のセグメントに配置する例については、「セグメント・チュートリアル」(233 ページ) で説明します。

syssegments の name **カラムにある名前**は、create table 文と create index 文内で使用されます。status カラムは、どのセグメントがデフォルト・セグメントであるかを示します。

---

**注意** segmap カラムと、記憶領域を管理するシステム・テーブルの詳細については、「領域の割り付けを管理するシステム・テーブル」(161 ページ) を参照してください。

---

# セグメント・チュートリアル

このチュートリアルでは、ユーザ・セグメントを作成する方法と、デバイスから他のセグメント・マッピングをすべて削除する方法を説明します。このチュートリアルでは、2K の論理ページを使用しているサーバを想定して、さまざまな例を示します。

セグメントとデバイスを操作する場合は、以下のことを考慮してください。

- 複数のフラグメントから成る領域を割り当てる場合、フラグメントごとに 1 つのエントリが sysusages に存在することになります。
- あるデバイスの追加フラグメントをデータベースに割り当てた場合、既存のフラグメントにマップされたすべてのセグメントが、新しいフラグメントにマップされます。
- alter database コマンドを使用してデバイス上の領域を追加するとき、そのデバイスをそのデータベースで使用するのが初めての場合は、system セグメントと default セグメントが、新しい領域に自動的にマップされます。

このチュートリアルでは、初めに、新しいデータベースを作成します。この例のデータベースでは、データベース・オブジェクト用にデバイスを 1 つ使用し、トランザクション・ログ用に別のデバイスを使用します。

```
create database mydata on bigdevice = "5M"
     log on logdev = "4M"
```

次に、use mydata を実行し、さらに sp_helpdb を実行すると、次のような出力が表示されます。

```
sp_helpdb mydata

name        db_size  owner     dbid   created      status
----------- -------- --------- ------ ------------ ---------------
mydata       9.0 MB sa              5 May 27, 2005  no options set

device_fragments  size    usage      created              free kbytes
----------------  ------  ---------- -----------          -----------
bigdevice         5.0 MB  data only  May 25 2005 3:42PM   3650
logdev            4.0 MB  log only   May 25 2005 3:42PM   not applicable
------------------------------------------------------------------------
log only free kbytes = 4078
device                  segment
----------------------- ----------------------
bigdevice               default
bigdevice               system
logdev                  logsegment
(return status = 0)
```

新しく作成されるすべてのデータベースと同様に、mydata データベースには default、system、logsegment という名前のセグメントがあります。create database 文で log on を使用したので、logsegment は専用のデバイスである logdev にマップされ、default と system の両セグメントは bigdevice にマップされます。

次のように、同じデータベース・デバイス上の領域を mydata に追加して、sp_helpdb をもう一度実行すると、追加したフラグメントのエントリが表示されます。

```
use master
alter database mydata on bigdevice = "2M"
    log on logdev = "1M"
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created       status
---------- -------- --------- ------ ------------  ---------------
mydata     12.0 MB  sa             4  May  25,  2005  no  options  set

device_fragments  size     usage      created                free kbytes
----------------- -------- ---------- -----------------      -----------
bigdevice         5.0 MB   data only  May 25 2005 3:42PM            2048
logdev            4.0 MB   data only  May 25 2005 3:42PM     not applicable
data only         2.0 MB   log only   May 25 2005 3:55PM            2040
log only          1.0 MB   log only   May 25 2005 3:55PM     not applicable
--------------------------------------------------------------------------------
log only free kybytes = 5098
device                  segment
----------------------- ----------------------
bigdevice               default
bigdevice               system
logdev                  logsegment
```

常に、ログ領域はログ領域に追加し、データ領域はデータ領域に追加してください。既にデータ用に使用されているセグメントをログに割り付けようとしたり、逆にログ用のセグメントをデータに割り付けようとしたりすると、with override を使用するように要求するメッセージが表示されます。セグメントは、領域のフラグメントだけではなく、デバイス全体にマップされます。デバイスのセグメントの割り当てを一部でも変更すると、すべてのフラグメントが変更されます。

次に、まだ mydata によって使用されていない新しいデータベース・デバイスを割り付ける例を示します。

```
use master
alter database mydata on newdevice = 3
use mydata
sp_helpdb mydata
```

```
name       db_size  owner     dbid   created      status
```

```
---------- -------- --------- ------ ------------ ---------------
mydata      15.0 MB sa             5 May 25, 2005  no options set

device_fragments   size      usage       created             free kbytes
------------------ --------- ----------- ------------------  --------------
bigdevice          5.0 MB    data only   May 25 2005 3:42PM            3650
logdev             4.0 MB    log only    May 25 2005 3:42PM  not applicable
bigdevice          2.0 MB    data only   May 25 2005 3:55PM            2040
logdev             1.0 MB    log only    May 25 2005 3:55PM  not applicable
newdevice          3.0 MB    data only   May 26 2005 11:59AM           3060
------------------------------------------------------------------------------
log only free kbytes = 5098
device           segment
---------------- ----------
bigdevice        default
bigdevice        system
logdev           logsegment
newdevice        default
newdevice        system
```

次に、newdevice 上に new_space という名前のセグメントを作成する例を示します。

```
sp_addsegment new_space, mydata, newdevice
```

次は、sp_helpdb のレポートのうち、セグメント・マッピングのリストの部分です。

```
device                         segment
------------------------------ ------------------
bigdevice                      default
bigdevice                      system
logdev                         logsegment
newdevice                      default
newdevice                      new_space
newdevice                      system
```

default セグメントと system セグメントは、まだ newdevice にマッピングされています。パフォーマンスを向上させるために、特定のテーブルまたはインデックスを new_space に保管する計画があり、他のユーザ・オブジェクトがデフォルトでそのデバイス上に保管されないようにするには、次のように、sp_dropsegment を使用して default と system のスコープを縮小します。

```
sp_dropsegment system, mydata, newdevice
sp_dropsegment "default", mydata, newdevice
```

defaultは Transact-SQL の予約語であるため、引用符で囲む必要があります。

次に示すのは、sp_helpdb のレポートのセグメント・マッピングを示す部分です。

```
device                          segment
------------------------------ --------------------
bigdevice                       default
bigdevice                       system
logdev                          logsegment
newdevice                       new_space
```

この時点では、new_space だけが newdevice にマップされています。オブジェクトを作成するユーザは、on new_space を使用すると、そのセグメントに対応するデバイスにテーブルまたはインデックスを配置することができます。default セグメントはそのデータベース・デバイスを指してはいないので、テーブルやインデックスを作成するときに on 句を使用しなければ、自分で特別に用意したデバイスにそのオブジェクトを置くことはできません。

alter database on newdevice をもう一度実行すると、新しい領域フラグメントのセグメント・マッピングは、そのデバイスの既存のフラグメントと同じ (new_space セグメントだけ) となります。

ここで、セグメントとして new_space を指定して create table を実行し、その後で sp_helpsegment を実行すると、結果は次のようになります。

```
create table mytabl (c1 int, c2 datetime)
    on new_space

sp_helpsegment new_space

segment    name                           status
------- ------------------------------    ------
      3    new_space                           0

device           size        free_pages
---------------- ----------- -----------
newdevice        3.0MB              1523

Objects on segment 'new_space':

table_name        index_name         indid        partition_name
----------------- ------------------ --------     ----------------
mytabl            mytabl                    0     mytabl_400001425

Objects currently bound to segment 'new_space':

table_name      index_name     indid

total_size    total_pages    free_pages  used_pages   reserved_pages
------------ -----------    ----------- -----------   --------------
3.0MB                1536           1523          13                0
```

第 10 章

# *reorg* コマンドの使用方法

テーブルに対する更新アクティビティは、いずれは領域の使用効率低下の原因となり、パフォーマンスの低下をもたらす可能性があります。テーブル領域の使用を再編成し、パフォーマンスを向上させるには、reorg コマンドを使用します。



## *reorg* コマンドとそのパラメータ

reorg は、次のような場合に使用すると便利です。

- 大量の転送されたローが存在するため、読み込みオペレーション中に余分な I/O が発生する。
- 挿入や直列化可能な読み込みでアクセスするページに、再利用化が必要な不連続の空き領域があるため、これらのオペレーションが遅くなる。
- データ・ページとインデックス・ページのクラスタ率が低いため、大容量 I/O オペレーションが遅くなる。
- sp_chgattribute を使用して領域管理の設定 (reservepagegap、fillfactor、または exp_row_size) を変更したが、その変更を将来の更新だけでなく、テーブル内の既存のすべてのローとページに適用したい。

reorg コマンドには、さまざまなタイプとレベルの再編成を実行するための次の 4 つのパラメータが用意されています。

- reorg forwarded_rows － ローの転送を取り消します。
- reorg reclaim_space － ローの削除や短縮などの更新の結果としてページ上に残された未使用の領域を再利用できるようにします。
- reorg compact － 領域の再利用化とローの転送の取り消しの両方の操作を行います。

- reorg rebuild － 領域の再利用化とローの転送の取り消しの操作を行います。それ以外にも次の処理を行います。
  - テーブルにクラスタード・インデックスがある場合は、それに従うようにすべてのローを書き込み直します。
  - データ・パーティションとインデックス・パーティションの領域を書き込み直します。
  - 個々のパーティションに対して使用できます。
  - sp_chgattribute によって変更された領域管理設定に従うように、ローをデータ・ページに書き込みます。
  - テーブルに属するすべてのインデックスの削除と再作成を行う。

reorg rebuild を実行する前に、以下の点について考慮してください。

- reorg rebuild は、その実行が終了するまで排他ロックを保持します。大きなテーブルを処理するときは、これはかなりの時間となります。ただし、reorg rebuild はクラスタード・インデックスの削除と再作成に必要なすべての処理を行うので、結果的には効率的です。さらに、reorg rebuild は、テーブルの現在の領域管理設定値をすべて使用してテーブルを再構築します。インデックスの削除と再作成を行うときは、reservepagegap の領域管理設定は使用されません。
- 多くの場合、reorg rebuild を実行するには、再構築するテーブルとそのインデックスが使用する領域と同じサイズの領域が別に必要となります。

さらに、次の制約があります。

- コマンドでテーブルを指定する場合は、そのテーブルに使用されるロック・スキームは datarows または datapages でなければなりません。
- システム管理者またはオブジェクト所有者だけが reorg を実行できます。
- トランザクション内で reorg を実行することはできません。

## *reorg* の必要性を確認するための *optdiag* ユーティリティの使用方法

reorg の実行が必要かどうかを判定するには、systabstats テーブルの統計と optdiag ユーティリティを使用します。systabstats にはテーブル領域の使用状況に関する統計が保存されていますが、optdiag は systabstats テーブルと sysstatistics テーブルの両方の統計に基づいてレポートを作成します。

systabstats テーブルの詳細については、『パフォーマンス＆チューニング・シリーズ：物理データベースのチューニング』の「第 2 章 統計テーブルおよび optdiag を使った統計の表示」を参照してください。optdiag については、『ユーティリティ・ガイド』を参照してください。

# 転送されたローのホーム・ページへの移動

更新の結果、ローが長くなり、現在のページに収まらない場合は、そのローは別のページに転送されます。そのローへの参照は、元のページであるホーム・ページに保持され、転送されたローへのアクセスは、すべてこの参照を通じて行われます。したがって、転送されたローにアクセスするためには、2 ページへのアクセスが必要となります。スキャンのときに大量の転送されたページを読み込む必要がある場合は、その I/O によってページへのアクセスが増えるので、パフォーマンスが低下します。

reorg forwarded_rows を使用すると、ローの転送を取り消すことができます。このとき、転送されたローは、そのホーム・ページに戻される (十分な領域がある場合) か、または削除されて新しいホーム・ページに再挿入されます。テーブルが複数のパーティションにまたがっている場合、*partition_name* パラメータを使用してパーティションを指定できます。

テーブル内の転送されたローの数に関する統計を表示するには、systabstats に対するクエリを実行するか、optdiag を使用します。

reorg forwarded_rows の構文は、次のとおりです。

```
reorg forwarded_rows table_name partition partition_name
    [with {resume, time = no_of_minutes}]
```

resume オプションと time オプションの詳細については、「resume オプションと time オプションによる大きなテーブルの再編成」(245 ページ) を参照してください。

インデックスには転送されたローはないので、reorg forwarded_rows はインデックスには使用できません。

## *reorg compact* によるロー転送の取り消し

reorg forwarded_rows は、アロケーション・ページ・ヒントを使って、転送されたローを探します。テーブル全体を検索する必要はないので、このコマンドの実行には時間はかかりませんが、転送されたローを見落とすこともあります。reorg forwarded_rows の実行後に optdiag を使用して「転送されたローの数」をチェックすることで、このコマンドの効果を評価できます。「転送されたローの数」が多い場合は、reorg compact を実行します。このコマンドは、テーブルのページを 1 つずつ調べ、すべてのロー転送を取り消します。

# 削除や更新の結果生じた未使用領域の再利用

タスクによって**削除**が行われたとき、またはローが短くなるような更新が行われたとき、空いた領域は、トランザクションがロールバックされる場合に備えてそのまま残されます。このような削除や行が短くなるような更新が頻繁に実行されるテーブルでは、再利用化できない領域が増えて、ついにはパフォーマンスに影響するようになることがあります。

reorg reclaim_space は、削除や更新操作の結果として未使用状態となった領域を、再利用可能にします。コミットされた削除やローを短くする更新の結果として未使用の領域ができたページのそれぞれについて、残ったローを連続するように書き込み直し、未使用の領域をページの終わりに集めます。残っているローがない場合は、reorg reclaim_space はそのページの割り付けを解除します。

テーブルが 1 つ以上のパーティションに拡張されている場合、*partition_name* を使用することで、パーティション上の利用可能領域を再利用できます。

テーブル内の、削除後に再利用化処理が行われていないロー数に関する統計は、systabstats テーブルから、または optdiag ユーティリティを使って表示できます。ローが短くなる更新の結果の未使用領域がどれくらいあるかを直接調べる方法はありません。

テーブル名のみを指定すると、そのテーブルのデータ・ページのみが再編成されて未使用の領域が利用可能となります。つまり、インデックスは処理されません。インデックス名を指定すると、そのインデックスのページのみが再編成されます。パーティションを使用する場合、そのパーティション上のテーブル部分のみが影響を受けます。

「resume オプションと time オプションによる大きなテーブルの再編成」(245 ページ) を参照してください。

## *reorg* コマンドを使わずに領域の再利用を行う方法

テーブル内の領域をページ単位で再利用可能にする、あるいは再編成する処理は、次のようなアクティビティによって行われます。

- 挿入において、ページ内の未使用領域の再利用化処理を実行すればそのページへの挿入が可能と判断される場合。
- update statistics コマンドの実行 (インデックス・ページに対してのみ有効)。
- クラスタード・インデックスの再作成。
- ハウスキーピング・ガーベジ・コレクション・タスクの実行 (enable housekeeper GC の値が 1 以上に設定されている場合)。

上記の方法は、それぞれ制約があり、ページ数が多い場合には効率が悪いこともあります。たとえば、挿入操作で領域の再利用化処理が必要になると挿入の動作が遅くなることがあります。また、再編成可能なページが多数存在していても処理できません。ハウスキーピング・ガーベジ・コレクション・タスクによる領域の再利用化では未使用領域が圧縮されますが、ユーザ優先度で実行されるハウスキーピング・ガーベジ・コレクション・タスクの 1 回の実行で、再利用化が必要なページをすべて処理できないこともあります。

## 未使用領域の再利用とローの転送の取り消し

reorg compact の機能は、reorg reclaim_space と reorg forwarded_rows の機能を合わせたものです。reorg compact は、次のような場合に使用します。

- 1 つのテーブル全体を再構築する必要がない場合 (必要があれば reorg rebuild を使用する)。ただし、ローの転送と、削除や更新操作の結果できた未使用の領域のどちらもパフォーマンスを低下させる恐れがあります。
- 転送されたローが大量にある場合。詳細については、「reorg compact によるロー転送の取り消し」(239 ページ) を参照してください。

パーティションを使用する場合、そのパーティション上のテーブル部分のみが影響を受けます。

「resume オプションと time オプションによる大きなテーブルの再編成」(245 ページ) を参照してください。

## テーブルの再構築

次のような場合に reorg rebuild を使います。

- 通常であれば大容量 I/O を使用するクエリに大容量 I/O が選択されず、かつ optdiag の結果からデータ・ページ、データ・ロー、またはインデックス・ページのクラスタ率が低いことがわかっている場合。
- sp_chgattribute を使用して領域管理設定 exp_row_size、reservepagegap、fillfactor の値を変更したが、その変更内容を将来のデータだけではなく、既存のローとページにも適用したい場合。sp_chgattribute の詳細については、『リファレンス・マニュアル：プロシージャ』を参照してください。

クラスタ率が低いのでテーブルを再構築する必要がある場合は、そのテーブルの領域管理設定値も変更する必要がある可能性があります (「reorg rebuild の実行前に領域管理設定値を変更する」(243 ページ) を参照)。

reorg rebuild は、現在のテーブルが他のセッションによって使用されていることを検出した場合、トランザクション全体をアボートします。

reorg rebuild は、テーブルの現在の領域管理設定値を使い、そのテーブルにクラスタード・インデックスがある場合はそのインデックスに従ってテーブルのローを書き込み直します。そのテーブルのすべてのインデックスは、削除され、現在の reservepagegap と fillfactor の領域管理設定値を使って再作成されます。rebuild の実行後、テーブルには転送されたローはなくなり、削除や更新の結果生じた未使用の領域もなくなります。

テーブルとパーティションに対して reorg rebuild を実行すると、次の処理が行われます。

- 排他テーブル・ロックを取得します
- 古いページから新しいページにデータをコピーします
- 古いデータ・ページの割り付けを解除します
- システム・テーブルを更新するためにロックします (sysindexes、sysobjects、syspartitions、systabstats など)
- 新しいデータ・ページに対してクラスタード・インデックスとノンクラスタード・インデックスを再構築します
- すべてのオープン・トランザクションをコミットします
- システム・テーブルのロックを解放します

テーブルが大きく、多数のインデックスがある場合は、更新のためにシステム・テーブルが長時間ロックされることがあります。このロックによって、reorg を実行しているユーザ・テーブルのシステム・テーブルにある情報へのアクセスがプロセスからブロックされます。ただし、systabstats は既にデータロー・ロックされているので、ブロックに影響しません。

reorg rebuild でのクラスタード・インデックスの構築には with sorted data オプションが使用されるので、このインデックスの構築中のデータの再ソートは必要ありません。

## *reorg rebuild* 実行のための前提条件

テーブルに対して reorg rebuild を実行する前に、次の準備が必要です。

- select into/bulkcopy/pllsort データベース・オプションを true に設定します。
- テーブルでデータページ・ロック・スキームまたはデータロー・ロック・スキームのどちらを使用するかを決定します。
- テーブルとそのインデックスのサイズに等しいディスク空き領域があることを確認します。

select into/bulkcopy/pllsort を true に設定するには、次のように入力します。

```
1> use master
2> go
1> sp_dboption pubs2,
    "select into/bulkcopy/pllsort", true
2> go
```

テーブルに対して rebuild を実行した後は、次のようになります。

- トランザクション・ログをダンプする前に、そのテーブルが含まれているデータベースをダンプする必要があります。
- テーブルの分散統計が更新されます。
- そのテーブルを参照するすべてのストアド・プロシージャが、次に実行されるときに再コンパイルされます。

### *reorg rebuild* の実行前に領域管理設定値を変更する

reorg rebuild によってテーブルが再構築されるとき、テーブルとインデックスのすべてのローが、そのテーブルの reservepagegap、fillfactor、exp_row_size の現在の設定に従って書き換えられます。これらのプロパティは、挿入によりテーブルがフラグメント化される (クラスタ率が低下する) 速度に影響します。

テーブルのフラグメント化が速く進み、非常に頻繁に再構築する必要がある場合は、reorg rebuild を実行する前にそのテーブルの領域管理設定値を変えなければならない場合があります。

領域管理設定値を変更するには、sp_chgattribute を使用します (『リファレンス・マニュアル：プロシージャ』を参照してください)。sp_chgattribute の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。また、記憶領域管理の詳細については『パフォーマンス&チューニング・シリーズ：物理データベースのチューニング』の「第 3 章 記憶領域管理プロパティの設定」を参照してください。

# インデックスに対する *reorg rebuild* コマンドの使用方法

reorg rebuild コマンドを使用すると、テーブルそのものは読み込みと更新が可能な状態で、インデックスを再構築できます。

## *reorg rebuild index_name partition_name* を使用したインデックスの再構築

1 つのテーブルまたはパーティション・インデックスを再構築すると、すべてのインデックス・ローが新しいページに再度書き込まれます。これにより、次の方法でパフォーマンスを向上できます。

- インデックスのリーフ・レベルのクラスタ化を改善する。
- 保存されているフィルファクタの値をインデックスに適用してページ分割を減らす。
- 保存されている reservepagegap の値を適用する。これにより、ページを将来の分割のために予約できます。

reorg rebuild は、クエリでそのインデックスを使用するユーザとの競合を減らすために、一度に少数のページだけをロックします。インデックスの再構築は、一連の独立したトランザクションを使って行われますが、ネストされたトランザクションも使われます。ネストされた各トランザクションで約 32 ページが再構築され、外側の各トランザクションで約 256 ページが再構築されます。修正されるページに対してはアドレス・ロックが取得され、トップ・アクションの終わりで解放されます。トランザクション内で割り付けを解除されたページが再利用可能になるのは、次のトランザクションが開始したときです。

reorg rebuild コマンドの実行が停止したとき、既にコミットされたトランザクションはロールバックされません。したがって、再編成された部分はクラスタ化され、意図したとおりに領域が使用されていますが、再編成されなかった部分はコマンドを実行する前の状態のままです。インデックスの論理的な一貫性は維持されます。

---

**注意** クラスタード・インデックスを再構築しても、テーブルのデータ・ページには影響しません。影響を受けるのは、リーフ・ページと上位のインデックス・レベルだけです。レベル 1 より上の非リーフ・ページは再構築されません。

---

### インデックスの再構築に必要な領域

fill_factor または reservepagegap が指定されていない場合、インデックスを再構築するには最大約 256 ページの追加の領域がデータ・セグメント内に必要となります。インデックスを削除し、create index を使用して再作成する場合よりも多くのログ領域が必要ですが、実際のインデックスのサイズと比べると非常に小さな領域です。インデックスのクラスタ化は、利用可能な空き領域が大きいほど効率的に行われます。

**注意**　reorg rebuild を実行しても、インデックスのうち、クラスタ率が既に高く、許容できる領域が使用されている部分に対しては再構築が行われないことがあります。

### ステータス・メッセージ

大きなテーブルに対して reorg rebuild *indexname* を実行すると、時間がかかることがあります。定期的に出力されるステータス・メッセージが表示されます。メッセージの開始と終了は、エラー・ログと、reorg を実行するクライアント・プロセスに書き込まれます。進行状況を示すメッセージは、クライアントにのみ表示されます。

ステータスの報告を行う間隔は、処理対象ページ数と 10,000 ページのいずれか大きい方の 10% として計算されます。計算されたページ数の処理が終わると、ステータス・メッセージが出力されます。したがって、インデックスのサイズにかかわらず、出力されるメッセージ数は 10 以下となります。既存の reorg コマンドのステータス・メッセージは、より頻繁に出力されます。

## *resume* オプションと *time* オプションによる大きなテーブルの再編成

テーブル全体を再編成すると非常に時間がかかり、他のデータベース・アクティビティに影響することが予測される場合は、reorg コマンドの resume オプションと time オプションを使用します。time では、reorg を実行する時間の長さを指定できます。resume を使用すると、テーブル内の前回 reorg を終了した位置から reorg の実行を開始できます。この 2 つのオプションを組み合わせて使用すると、たとえば、営業時間外に部分的な再編成を続けて実行することによって大きなテーブルを再編成できます。

resume オプションと time オプションは、reorg rebuild では使用できません。

## *time* オプションで *no_of_minutes* を指定する

time オプションの *no_of_minutes* 引数は、CPU 時間ではなく経過時間を表します。たとえば、前回 reorg compact が停止した位置から reorg compact を 30 分間実行するには、次のように入力します。

```
reorg compact tablename with resume, time=30
```

この 30 分の間に reorg 処理がスリープした場合も、その時間は経過時間にカウントされます。reorg の実行時間が延長されるわけではありません。

指定の時間が経過すると、reorg は、処理されたテーブルまたはインデックスの部分の統計を systabstats テーブルに保存します。この情報は、reorg の resume オプションが指定されたときに再開位置を決めるために使用されます。resume オプションと time オプションを取る 3 つのパラメータの再開位置は、それぞれ別に保存されます。したがって、たとえば、初めに reorg reclaim_space を使用して、次に reorg compact を使用するプロセスを再開することはできません。

*no_of_minutes* で指定した時間が経過する前に reorg の処理がテーブルまたはインデックスの終わりまで行われると、オブジェクトの先頭に戻り、指定時間が経過するまで処理が実行されます。

---

**注意** resume と time を使うと、複数回の実行によりテーブル全体またはインデックス全体を再編成できます。ただし、reorg の実行と実行の間に更新が行われると、同じページが 2 回処理されたり、あるページがまったく処理されなかったりすることがあります。

---

第 11 章

# データベースの一貫性の検査



## データベース一貫性チェッカの定義

データベース一貫性チェッカ (dbcc) は、データベースの論理的および物理的一貫性を検査するためのコマンドです。dbcc の主な検査対象は次のとおりです。

- checkstorage、または checktable と checkdb を使用した、ページ・レベルとロー・レベルの両方におけるページ・リンクとデータ・ポインタ
- checkstorage、checkalloc、checkverify、tablealloc、textalloc、indexalloc を使用した、ページ割り付け
- checkcatalog コマンドを使用した、特定のデータベースにおけるシステム・テーブル内およびシステム・テーブル間の一貫性

dbcc checkstorage は、dbccdb データベースに検査結果を格納します。dbcc ストアド・プロシージャを使用して、dbccdb のレポートを出力できます。

dbcc コマンドは次のように使用します。

- 通常のデータベース管理として使用します。データベースの内部構造の整合性が保たれるかどうかは、システム管理者やデータベース所有者が定期的にデータベースの一貫性チェックを行うかどうかに左右されます。
- システム・エラーが発生した後に、損傷の可能性がある範囲を判断します。
- データベースのバックアップ前に使用することによって、バックアップの整合性に関する信頼度を高めます。
- データベースに損傷の可能性がある場合に使用します。たとえば、あるテーブルの使用中に「テーブルが破損しました」というメッセージが表示された場合、dbcc を使用して、データベース内の他のテーブルにも損傷があるかどうかを確認できます。

コンポーネント統合サービス (CIS) を使用している場合は、この他にもリモート・データベースに対して使用可能な dbcc コマンドがあります。『コンポーネント統合サービス・ユーザーズ・ガイド』を参照してください。

# ページとオブジェクト割り付け

データベース・デバイスを初期化するときに、disk init コマンドによって新しい記憶領域が「アロケーション・ユニット」に分割されます。アロケーション・ユニットのサイズは、サーバが使用する論理ページ・サイズ (2、4、8、16K) によって決まります。アロケーション・ユニットの最初のページは**アロケーション・ページ**で、このページには、アロケーション・ユニットのすべてのページの使用状況が記録されます。アロケーション・ページには、オブジェクト ID 99 が付与されます。このオブジェクトは実際のデータベース・オブジェクトではないため、システム・テーブル内にはありませんが、dbcc でアロケーション・ページのエラーが見つかると、この ID 番号が表示されます。

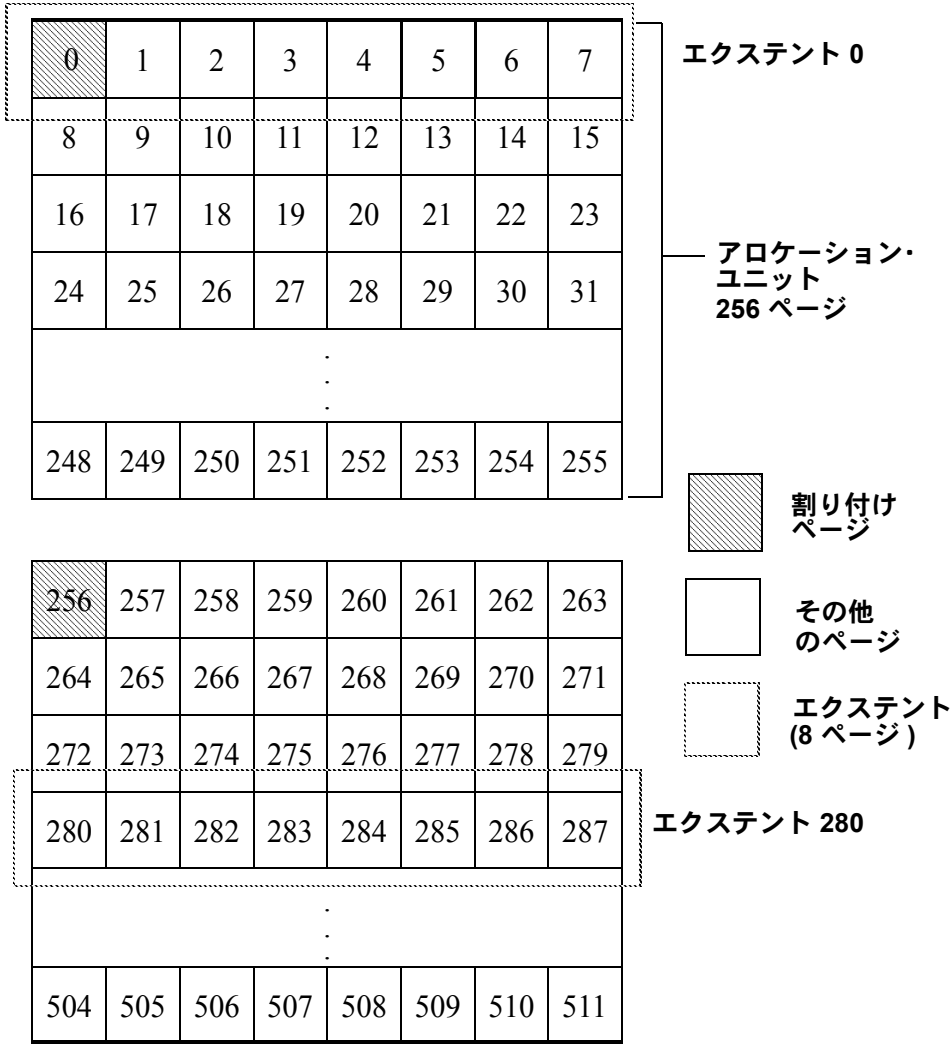
インデックス・パーティションのテーブルに領域が必要な場合、Adaptive Server は 8 ページから成るブロックをそのオブジェクトに割り付けます。この 8 ページ・ブロックを、エクステントと呼びます。各アロケーション・ユニットには、32 エクステントが収納されます。エクステントのサイズも、サーバの論理ページのサイズによって決まります。Adaptive Server は、エクステントを記憶領域管理の単位として使用し、次のように記憶領域の割り付けと解除を行います。

- インデックス・パーティションのテーブルを作成するとき、そのオブジェクトにエクステントを 1 つ割り付けます。
- 既存のテーブルにローを追加するときに、既存のページに空きがない場合は、新しいページを割り付けます。エクステントのページがすべて満杯の場合は、追加のエクステントを割り付けます。
- インデックス・パーティションのテーブルを削除するときは、使用されていたエクステントの割り付けを解除します。
- テーブルからローを削除した結果、テーブルのサイズが 1 ページ分小さくなる場合は、そのページの割り付けを解除します。テーブルが縮小した結果、エクステント全体が不要となった場合は、そのエクステントの割り付けを解除します。

エクステントに領域を割り付けたり、割り付けを解除したりするたびに、Adaptive Server は、そのオブジェクトのエクステントを追跡するアロケーション・ページ上にそのイベントを記録します。これにより、オブジェクトが縮小したり拡張したりしても余分なオーバヘッドは発生しないので、データベースの領域の割り付けをすばやく追跡できます。

図 11-1 に、Adaptive Server データベースのエクステントとアロケーション・ユニット内でのデータ・ページの設定方法を示します。

**図 11-1: エクステントによるページ管理**




dbcc checkalloc は、データベース内のすべてのアロケーション・ページ (ページ 0 と、256 で割り切れるすべてのページ) を検査し、割り付け情報をレポートします。dbcc indexalloc と dbcc tablealloc は、特定のデータベース・オブジェクトの割り付けを検査します。

## OAM (オブジェクト・アロケーション・マップ) について

テーブルとテーブルのインデックスのそれぞれに、オブジェクト・アロケーション・マップ (**OAM: object allocation map**) があります。OAM は、テーブルやインデックスに割り付けられたページに保管され、そのインデックスやテーブルに新しいページが必要となったときに検査されます。1 つの OAM ページは、2,000 ～ 63,750 のデータまたはインデックス・ページ用のアロケーション・マップを保持できます。OAM ページのサイズは、1 論理ページのサイズです。たとえば、4K の論理ページ・サイズを使用するサーバでは、各 OAM ページは 4K です。

OAM ページあたりのエントリ数も、サーバが使用している論理ページ・サイズによって決まります。次の表は、論理ページ・サイズ別の OAM エントリ数です。

| 2K 論理ページ・サイズ | 4K 論理ページ・サイズ | 8K 論理ページ・サイズ | 16K 論理ページ・サイズ |
|---|---|---|---|
| 250 | 506 | 1018 | 2042 |

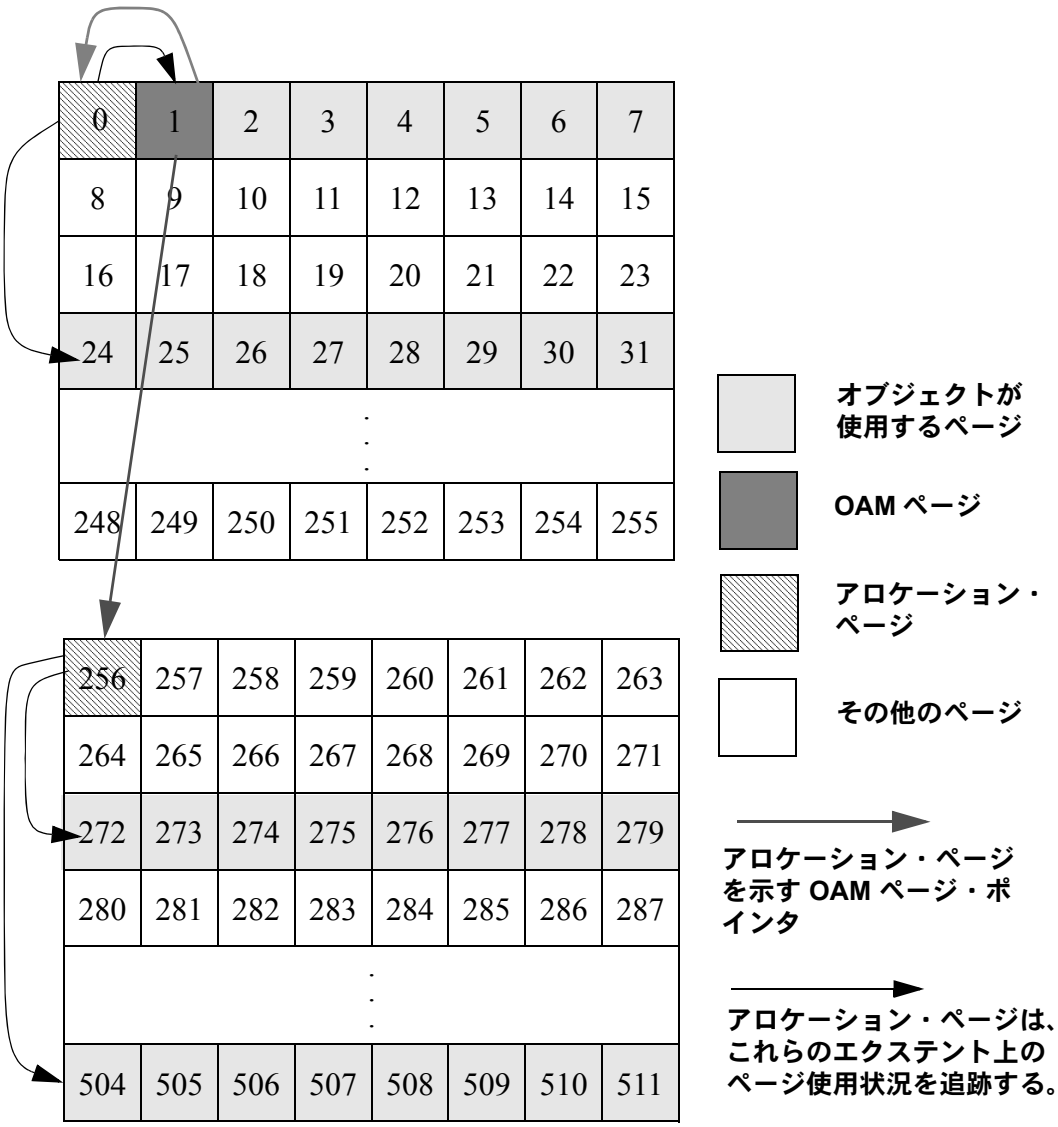
OAM ページは、オブジェクトが領域を使用するそれぞれのアロケーション・ユニットのアロケーション・ページを指します。アロケーション・ページには、アロケーション・ユニット内のエクステントとページの使用状況についての情報が記録されています。つまり、titles テーブルがエクステント 24 と 272 に保管されているとすれば、titles テーブルの OAM ページは、ページ 0 と 256 を指します。

図 11-2 は、2K の論理ページを使用しているサーバで、1 つのオブジェクトが 4 つのエクステント (0、24、272、504) に保管されていることを示します。OAM は、最初のセグメントの最初のページに保管されます。この場合、アロケーション・ページがページ 0 を占有するので、OAM はページ 1 に配置されます。

この OAM は、ページ 0 とページ 256 の 2 つのアロケーション・ページを指します。

これらのアロケーション・ページは、そのアロケーション・ユニット内に記憶領域を持つオブジェクトが使用する各エクステント内で使用されているページを記録します。この例のオブジェクトでは、エクステント (0、24、272、504) のページの割り付けの設定と解除が記録されます。

**図 11-2: OAM ページとアロケーション・ページのポインタ**

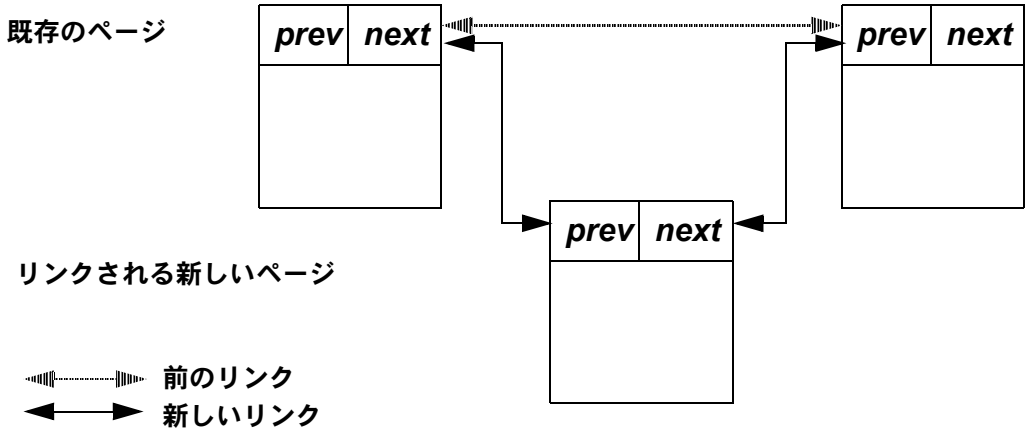



dbcc checkalloc コマンドと dbcc tablealloc コマンドは、ページ・リンクの確認に加えて、この OAM ページ情報も調べます。詳細については、「ページ・リンクについて」(253 ページ) を参照してください。

## ページ・リンクについて

インデックス・パーティションのテーブルにページが割り付けられると、そのページは、同じオブジェクトに対して使用される他のページとリンクされます。図 11-3 にこのようなリンクの例を示します。各ページにはヘッダがあり、直前のページの番号 (prev) と直後のページの番号 (next) が格納されています。新しいページが割り付けられると、前後のページのヘッダ情報が、そのページを指すように変更されます。dbcc checktable と dbcc checkdb はページ・リンクを確認します。dbcc checkalloc、tablealloc、indexalloc は、ページ・リンクをアロケーション・ページ上の情報と比較します。

**図 11-3: 新しく割り付けられたページと他のページのリンクのメカニズム**



# *dbcc* によって実行できる検査

表 11-1 は、dbcc コマンドを使用して行う一貫性検査の要約を示します。表 11-2 (269 ページ ) は、さまざまな dbcc コマンドを使用した場合のパフォーマンスと影響の比較です。

**表 11-1: dbcc コマンドによって実行される検査の比較**

| 実行される検査 | checkstorage | checktable | checkdb | checkalloc | indexalloc | tablealloc | checkcatalog | textalloc |
|---|---|---|---|---|---|---|---|---|
| テキスト値カラムの割り付け | X | | | | | | | X[1] |
| インデックスの一貫性 | | X | X | | | | | |
| インデックスのソート順 | | X | X | | | | | |
| OAM ページ・エントリ | X | X | X | X | X | X | | X |
| ページの割り付け | X | | | X | X | X | | X |
| ページの一貫性 | X | X | X | | | | | |
| ポインタの一貫性 | X | X | X | | | | | |
| システム・テーブル | | | | | | | X | X[2] |
| テキスト・カラム・チェーン | X | X | X | | | | | X |
| テキスト値カラム | X | X | X | | | | | X[3] |

[1]textalloc は、テキスト・ページの割り当てステータスを検査します (カラムがあるデータ・ページの割り当てステータスは検査しません)。

[2]textalloc は、text または image カラムに対応する syspartition エントリを検査します。

[3]textalloc の検査対象は次のとおりです。

- テキスト値カラムがあるデータ・ページ
- テキスト値が保存されているテキスト・ページ

---

**注意** dbrepair と checkdb 以外のすべての dbcc コマンドは、データベースがアクティブなときに、fix オプションを付けて実行できます。

---

checktable、fix_text または reindex キーワードを使用して dbcc を実行できるのは、テーブル所有者だけです。checkstorage、checkdb、checkcatalog、checkalloc、indexalloc、textalloc、tablealloc の各キーワードを使用できるのは、データベース所有者だけです。dbrepair キーワードを使用できるのは、システム管理者だけです。

## *dbcc* コマンドの出力について

dbcc checkstorage は、dbccdb データベースに検査の結果を保管します。このデータベースから、さまざまなレポートを出力できます。「dbcc checkstorage」(256 ページ) を参照してください。

他の多くの dbcc コマンドの出力には、検査中のオブジェクトを識別する情報や、コマンドによってオブジェクト内に発見された問題の内容を示すエラー・メッセージが含まれます。fix オプションを付けて dbcc tablealloc と dbcc indexalloc を実行した場合は、コマンドによって行われた修復も出力に含まれます。

次の例は、割り付けエラーのあるテーブルに対して dbcc tablealloc を実行したときの出力を示します。

```
dbcc tablealloc(rrtab)
```

```
The default report option of OPTIMIZED is used for this run.
The default fix option of FIX   is used for this run.
************************************************************
TABLE: rrtab            OBJID = 416001482
PARTITION ID=432001539 FIRST=2032 ROOT=2040 SORT=1
Data level: indid 1, partition 432001539. 2 Data pages allocated and 2 Extents
allocated.
Indid : 1, partition : 432001539. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=448001596 FIRST=2064 ROOT=2072 SORT=1
Data level: indid 1, partition 448001596. 2 Data pages allocated and 2 Extents
allocated.

Indid : 1, partition : 448001596. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=480001710 FIRST=2080 ROOT=2080 SORT=0
Indid : 2, partition : 480001710. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=496001767 FIRST=2096 ROOT=2096 SORT=0
Indid : 2, partition : 496001767. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=512001824 FIRST=2112 ROOT=2112 SORT=0
Indid : 3, partition : 512001824. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=528001881 FIRST=2128 ROOT=2128 SORT=0
Indid : 3, partition : 528001881. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=680 ROOT=680 SORT=0
Indid : 4, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 18
Alloc page 1792 (# of extent=2 used pages=2 ref pages=2)
Alloc page 1792 (# of extent=2 used pages=3 ref pages=3)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
Alloc page 2048 (# of extent=1 used pages=1 ref pages=1)
Alloc page 1792 (# of extent=1 used pages=1 ref pages=1)
```

```
Alloc page 2048 (# of extent=1 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=3 ref pages=3)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 2048 (# of extent=2 used pages=2 ref pages=2)
Alloc page 256 (# of extent=1 used pages=1 ref pages=1)
Alloc page 512 (# of extent=1 used pages=1 ref pages=1)
Total (# of extent=18 used pages=21 ref pages=21) in this database
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role..
```

# データベースとテーブルの一貫性の検査

データベースとテーブルの一貫性を検査するには、次の dbcc コマンドを使用します。

- dbcc checkstorage
- dbcc checktable
- dbcc checkdb

## *dbcc checkstorage*

dbcc checkstorage を使用して、次の検査を行います。

- テキスト値カラムの割り付け
- ページの割り付けと一貫性
- OAM ページ・エントリ
- パーティションごとの OAM ページの有無
- ポインタの一貫性
- テキスト値カラムとテキスト・カラム・チェーン

dbcc checkstorage は、ディスク上のデータベースに対して検査を実行するコマンドです。破損がメモリ内だけであれば、dbcc checkstorage を実行してもその破損を検出できない場合があります。dbcc checkstorage を 2 回実行した場合に、これらの実行結果が矛盾しないようにするには、dbcc checkstorage の前に checkpoint を実行します。ただし、このようにすると、メモリ内の一時的な破損がディスクに反映されてしまう場合があります。

### *dbcc checkstorage* を使用する利点

dbcc checkstorage を使用する利点は、次のとおりです。

- 他のさまざまな dbcc コマンドによる検査がまとめて実行されます。
- テーブルやページを長時間ロックしません。これにより、同時更新が行われている間も dbcc によって正確にエラーを見つけることができます。
- 集約的な I/O スループットによる直線的な拡張性 (リニア・スケーラビリティ ) を実現します。
- 検査とレポートの機能が分かれているので、独自の評価とレポートの生成が可能になります。
- ターゲット・データベースの領域の使用状況について、詳細な説明が得られます。
- dbcc checkstorage のアクティビティと結果は dbccdb データベース内に記録されます。これによって傾向の分析が可能になり、このデータを基に正確な診断を行うことができます。

### *dbcc checkstorage* と他の *dbcc* コマンドの比較

dbcc checkstorage は、他の dbcc コマンドと異なり、次のものを必要とします。

- dbccdb データベース － 設定情報およびターゲット・データベース上で行われた検査の結果を格納します。ただし、どのデータベースからでも dbcc checkstorage を実行できます。
- 2 つ以上の作業領域 － 検査オペレーション中に使用される領域。『リファレンス・マニュアル：テーブル』の「dbccdb のテーブル」を参照してください。
- システム・プロシージャとストアド・プロシージャ － 使用しているシステムが dbcc checkstorage を実行できるようにします。また、dbccdb データベースに格納されているデータに関するレポートを生成できるようにします。

dbcc checkstorage はどのようなフォールトも修復しません。dbcc checkstorage を実行し、レポートを生成してフォールトを確認した後で、該当する dbcc コマンドを実行してフォールトを修復します。

### *dbcc checkstorage* オペレーションについて

dbcc checkstorage のオペレーションは、次の手順で実行されます。

1 調査 − 検査対象のデータベースのデバイス割り付けとセグメント定義、max worker processing 設定パラメータ、および named cache 設定パラメータを使用して、可能な並列処理のレベルを決定します。また、設定パラメータ max worker processes と dbcc named cache によって、使用できる並列処理のレベルを制限します。

2 プラン作成 − 手順 1 で決定した並列処理を利用するようにオペレーションの実行プランを作成します。

3 実行と最適化 − 複数の Adaptive Server ワーカー・プロセスを使用して、ターゲット・データベースの検査と記憶領域の分析を並列で実行します。このとき、各ワーカー・プロセスで実行される作業を平均化し、処理量が少ないワーカー・プロセスの作業を統合するようにします。検査オペレーションの進行中に、検査オペレーション中に集められた追加情報を利用できるように、dbcc checkstorage は、手順 2 で作成されたプランの拡張と調整を行います。

4 レポート出力と制御 − dbcc checkstorage の検査オペレーションの実行中にターゲット・データベース内で見つかったフォールトはすべて dbccdb に記録されます。この情報は、後でレポート出力や評価に使用できます。また、記憶領域の分析結果も dbccdb に記録されます。dbcc checkstorage は、フォールトが見つかると、リカバリしてオペレーションを続行しようとしますが、そのフォールトの後では正しく実行できないオペレーションは終了します。たとえば、ディスクに欠陥があっても dbcc checkstorage が異常終了することはありませんが、欠陥のあるディスクに対して検査オペレーションを実行しても目的を達することができないため、そのオペレーションは実行されません。

   別のセッションが drop table コマンドを同時に実行すると、dbcc checkstorage は初期化フェーズで異常終了することがあります。その場合は、drop table の処理が終了してから dbcc checkstorage を再実行してください。

**注意** dbcc checkstorage のエラー・メッセージの詳細については、『トラブルシューティング＆エラー・メッセージ・ガイド』を参照してください。

## パフォーマンスとスケーラビリティ

dbcc checkstorage は、集約型の I/O スループットによってリニア・スケーラビリティ (処理能力や容量の増加、システムの拡張に即座に対応できる拡張性) を実現しており、パフォーマンスは dbcc checkalloc よりも大幅に優れています。dbcc checkstorage の拡張性 ( スケーラビリティ・プロパティ ) とは、データベースのサイズが 2 倍に増えて、ハードウェアの処理能力 (実感できる I/O スループット) が 2 倍に増えた場合でも、dbcc の検査に必要な時間は変わらないということです。通常、処理能力や容量が 2 倍になると、ディスクのスピンドル数も 2 倍になり、また I/O チャネル、システム・バス、および CPU の処理能力も十分なレベルに増加します。これによって、集約的なディスク・スループットがさらに向上します。

テキスト・カラム・チェーンの検証などの、dbcc checkalloc と dbcc checkdb を使用して行われる検査のほとんどは、dbcc checkstorage を使用すれば一度に検査できるので、無駄な検査オペレーションを排除できます。

dbcc checkstorage は、未使用のページも含めたデータベース全体を検査するため、実行時間とデータベースのサイズの間には相関関係があります。そのため、dbcc checkstorage を使用する場合は、他の dbcc コマンドを使用する場合と異なり、ほとんど空のデータベースでも、満杯に近いデータベースでも、検査に必要な時間に大きな違いはありません。

他の dbcc コマンドとは異なり、dbcc checkstorage のパフォーマンスは、データの配置にはほとんど左右されません。そのため、セッションとセッションの間にデータの配置が変更されても、各セッションのパフォーマンスは一定です。

dbcc checkstorage は並列オペレーションの設定と dbccdb への大量データの記録という作業も行うため、ターゲット・データベースが小さいときは、他の dbcc コマンドの方が処理時間は短くなります。

dbcc checkstorage によって使用される作業領域のロケーションと割り付けは、パフォーマンスとスケーラビリティに影響します。『リファレンス・マニュアル：テーブル』の「dbccdb のテーブル」を参照してください。

dbcc checkstorage とレポート生成用のシステム・プロシージャの 1 つを単一コマンドで実行するには、sp_dbcc_runcheck を使用してください。

## *dbcc checktable*

dbcc checktable を使用して、次のような検査を行います。

- インデックスとデータ・ページが正確にリンクされていること
- インデックスが適切にソートされていること
- ポインタに矛盾がないこと
- すべてのインデックスとデータ・パーティションが正しくリンクされていること

- 各ページのデータ・ローが、ロー・オフセット・テーブルにエントリを持つこと。これらのエントリが、そのページのデータ・ローのロケーションと一致していること
- 分割されたテーブルの分割情報が正しいこと

skip_ncindex オプションを使用すると、ノンクラスタード・インデックスのページ・リンク、ポインタ、ソート順の検査を省略できます。クラスタード・インデックスとデータ・ページとのリンクとポインタは、テーブルの整合性を保つには欠かせません。ノンクラスタード・インデックスのページ・リンクまたはポインタについての問題がレポートされたときは、そのインデックスを削除して再生成します。

*partition_name* は、チェックするパーティションの名前です (テーブルは複数のパーティションにまたがることができるため、パーティションがテーブル全体を含む場合と含まない場合があります )。*partition_id* は、チェックするパーティションの ID です。

*partition_name* または *partition_id* を使用する場合、このパーティションにあるテーブルまたはテーブルの一部のみが dbcc checktable によってチェックされます。他のパーティションはチェックされません。また、次のような制限があります。

- テーブルが複数のパーティションから構成される場合、インデックスの処理はローカル・インデックスに限定されます。
- *partition_name* パラメータまたは *partition_id* パラメータを指定する場合は、2 番目のパラメータも指定する必要があります。2 番目のパラメータは skip_ncindex、fix_spacebits または NULL です。次の例は NULL を指定しています。

```
dbcc checkalloc(titles, null, 560001995)
```

- char または varchar データ型で定義されたカラムを持つテーブルのソート順または文字セットが正しくない場合、dbcc checktable はこれらの値を修正しません。これらのエラーを修正するには、テーブル全体に対して dbcc checktable を実行してください。
- ソート順が変更されたためにインデックスに「読み込み専用」を示すマークが付いている場合、テーブルの sysobjects エントリの sysstat フィールドに設定された O_READONLY ビットは、dbcc checktable によってクリアされません。このステータス・ビットをクリアするには、テーブル全体に対して dbcc checktable を実行してください。
- dbcc checktable を syslogs に対して実行する場合、dbcc checktable は領域の使用量 ( 空き領域と使用済み領域の比較 ) をレポートしません。ただし、*partition_name* または *partition_id* パラメータを指定しない場合、dbcc checktable は領域の使用量をレポートします。

checkstorage からフォールト・コード 100035 が返され、このスペースビット・フォールトがハード・フォールトであることを checkverify によって確認したときは、dbcc checktable を使用して、レポートされたフォールトを修正することができます。

次のコマンドでは、smallsales パーティション (本の総売上額 5000 未満) にある titles テーブルの一部をチェックします。

```
dbcc checktable(titles, NULL, "smallsales")
Checking partition 'smallsales' (partition ID 1120003990) of table 'titles'.
The logical page size of this table is 8192 bytes. The total number of data
pages in partition 'smallsales' (partition ID 1120003990) is 1.
Partition 'smallsales' (partition ID 1120003990) has 14 data rows.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

dbcc checktable にはテーブル名とオブジェクト ID のどちらも指定できます。sysobjects テーブルの name カラムと id カラムにこれらの情報が格納されています。

次の例は、損傷を受けていないテーブルについてのレポートを示します。

```
dbcc checktable(titles)
```

```
Checking table 'titles' (object ID 576002052):Logical page size is 8192 bytes.
The total number of data pages in partition 'titleidind_576002052' (partition ID
576002052) is 1.
The total number of data pages in this table is 1.
Table has 18 data rows.
DBCC execution completed. If DBCC printed error messages, contact a user with
System Administrator (SA) role.
```

現在のデータベースにないテーブルを検査するには、データベース名を入力してください。別のオブジェクトが所有するテーブルを検査する場合は、その所有者の名前を入力してください。修飾されたテーブル名は、次に示すように引用符で囲んでください。次に例を示します。

```
dbcc checktable("pubs2.newuser.testtable")
```

dbcc checktable は次のような問題を調べます。

- ページ・リンクが不正な場合、dbcc checktable はエラー・メッセージを表示します。
- char データ型または varchar データ型のカラムがあるテーブルのソート順 (sysindexes.soid) や文字セット (sysindexes.csid) が誤っているが、テーブルのソート順が Adaptive Server のデフォルトのソート順と互換性があるものである場合は、dbcc checktable は、そのテーブルの値を訂正します。すべての文字セットと互換性があるのは、バイナリのソート順だけです。

  **注意** ソート順を変更すると、文字ベースのユーザ・インデックスの状態は「読み込み専用」となるため、確認のうえ、必要であれば再構築してください。

- データ・ローがオブジェクトの最初の OAM ページの計算に含まれていない場合は、dbcc checktable は、そのページのロー数を更新します。これは重大な問題ではありません。sp_spaceused などのプロシージャで使用される組み込み関数 row_count は、この数値を使用してローの見積りを迅速に行います。

強化されたページ・フェッチを使用することにより、dbcc checktable のパフォーマンスを向上させることができます。

dbcc checkindex

dbcc checkindex コマンドは、dbcc checktable と同様の検査を行いますが、対象はテーブル全体ではなく指定されたインデックスのみです。

*partition_name* は、チェックするパーティションの名前です。*partition_id* は、チェックするパーティション ID です。bottom_up は、checkindex が下位のパーティションからチェックするかどうかを指定します。下位のパーティションからチェックする場合は、インデックスのリーフからチェックが開始されます。bottom_up はデータオンリー・ロック・テーブルに対してのみ適用されます。このオプションを checkindex または checktable とともに指定すると、インデックスはボトムアップ方式でチェックされます。

## *dbcc checkdb*

dbcc checkdb コマンドは、指定されたデータベースの各テーブルについて、dbcc checktable と同様の検査を実行します。データベース名が省略された場合は、dbcc checkdb は現在のデータベースを検査します。dbcc checkdb は、dbcc checktable と同様のメッセージを表示し、同じ修正を実行します。

skip_ncindex は省略可能です。指定した場合は、データベース内のユーザ・テーブルのノンクラスタード・インデックスは一切検査されません。

データベースが一連のパーティションに拡張されている場合、dbcc checkdb は各パーティションをチェックします。

# ページ割り付けの検査

ページ割り付けを検査するには、次の dbcc コマンドを使用します。

- dbcc checkalloc
- dbcc indexalloc
- dbcc tablealloc
- dbcc textalloc

## *dbcc checkalloc*

dbcc checkalloc は、次のことを確認します。

- すべてのページが正しく割り付けられていること
- アロケーション・ページにある分割情報が正しいこと
- 割り付けられているページがすべて使用されていること
- すべてのページが個々のパーティションに正しく割り付けられていること
- 割り付けられているページのみが使用されていること

データベース名が省略された場合は、dbcc checkalloc は現在のデータベースを検査します。

fix オプションを選択して dbcc checkalloc を使用する場合は、dbcc tablealloc によっても修復可能なすべての割り付けエラーが修復され、また、データベースから削除されたオブジェクトへ割り付けられたままのページも修復されます。fix オプションを指定して dbcc checkalloc を実行する前に、データベースをシングルユーザ・モードにしてください。fix オプションと no fix オプションの使用方法の詳細については、「fix | nofix オプションによる割り付けエラーの修正」(266 ページ) を参照してください。

dbcc checkalloc の出力では、各テーブルのデータが 1 つのブロックとして表示され、これにはシステム・テーブルと各テーブルのインデックスも含まれます。テーブルまたはインデックスごとに、使用されているページ数とエクステント数がレポートされます。INDID 値は次のとおりです。

- クラスタード・インデックスのないテーブルでは INDID = 0 となります。

- クラスタード・インデックスのある全ページ・ロック・テーブルでは、テーブル・データ・パーティションとクラスタード・インデックス・パーティションは統合され、データ・パーティション (またはクラスタード・インデックス・パーティション) は INDID = 1 となります。
- クラスタード・インデックスのあるデータオンリー・ロック・テーブルでは、テーブル・データ・パーティションは INDID=0 となります。クラスタード・インデックスとノンクラスタード・インデックスには、INDID = 2 から始まる続き番号が付与されます。

パーティションとページ情報は、PARTITION ID = *partition_number* の後にリストされます。

次の pubs2 のレポートには、titleauthor、titles、stores の各テーブルの情報が出力されています。

```
***************************************************************
TABLE: titleauthor OBJID = 544001938
PARTITION ID=544001938 FIRST=904 ROOT=920 SORT=1
Data level: indid 1, partition 544001938. 1 Data pages allocated and 2 Extents
allocated.
Indid : 1, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=928 ROOT=928 SORT=0
Indid : 2, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
PARTITION ID=544001938 FIRST=944 ROOT=944 SORT=0
Indid : 3, partition : 544001938. 1 Index pages allocated and 2 Extents
allocated.
TOTAL # of extents = 8
***************************************************************
TABLE: titles OBJID = 576002052
PARTITION ID=1120003990 FIRST=1282 ROOT=1282 SORT=1
Data level: indid 0, partition 1120003990. 1 Data pages allocated and 1 Extents
allocated.
PARTITION ID=1136004047 FIRST=1289 ROOT=1289 SORT=1
Data level: indid 0, partition 1136004047. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 2
***************************************************************
TABLE: stores OBJID = 608002166
PARTITION ID=608002166 FIRST=745 ROOT=745 SORT=0
Data level: indid 0, partition 608002166. 1 Data pages allocated and 1 Extents
allocated.
TOTAL # of extents = 1
```

## *dbcc indexalloc*

dbcc indexalloc コマンドは、指定されたインデックスに対して次の検査を行います。

- すべてのページが正しく割り付けられていること
- 割り付けられているページがすべて使用されていること
- 割り付けられているページのみが使用されていること

dbcc indexalloc は、dbcc checkalloc のインデックス・レベルのバージョンであり、個々のインデックスに対して同様の整合性の検査を行います。これにはインデックス・パーティションの検査も含まれます。オブジェクト ID、オブジェクト名、またはパーティション ID のいずれかを指定する必要があります。さらに、インデックス ID を指定する必要があります。dbcc checkalloc と dbcc indexalloc の出力にはインデックス ID が含まれます。

dbcc indexalloc の fix オプションまたは nofix オプションを使用するには、レポート・オプション (full、optimized、fast、または null) のいずれかも指定してください。パーティション ID を指定すると、そのパーティションのみが検査されます。詳細については、「fix | nofix オプションによる割り付けエラーの修正」(266 ページ) と 「dbcc tablealloc と dbcc indexalloc によるレポートの生成」(267 ページ) を参照してください。

dbcc indexalloc によって、非分割インデックスは単一パーティションのある 1 つのインデックスとして扱われます。

sp_indsuspect を実行して、インデックスのソート順の一貫性を検査でき、dbcc reindex を実行して、不一致を修復できます。

## *dbcc tablealloc*

dbcc tablealloc は、指定されたユーザ・テーブルに対して次の検査を行います。

- すべてのページが正しく割り付けられていること
- アロケーション・ページにある分割情報が正しいこと
- 割り付けられているページがすべて使用されていること
- 割り付けられているページのみが使用されていること
- すべてのページが指定したテーブルのパーティションに正しく割り付けられていること

sysobjects の ID カラムにあるテーブル名、データ・パーティション ID、またはテーブルのオブジェクト ID を指定します。データ・パーティション ID を指定すると、dbcc tablealloc によってこのパーティションとすべてのローカル・インデックス・パーティションが検査されます。オブジェクト名またはオブジェクト ID を指定すると、dbcc tablealloc によってテーブル全体が検査されます。インデックス・パーティション ID を指定すると、dbcc tablealloc はエラー 15046 を返します。

dbcc tablealloc の fix オプションまたは nofix オプションを使用する場合は、レポート・オプション (full、optimized、fast、または null) のいずれかも指定してください。詳細については、「fix | nofix オプションによる割り付けエラーの修正」(266 ページ) と 「dbcc tablealloc と dbcc indexalloc によるレポートの生成」(267 ページ) を参照してください。

### dbcc textalloc

dbcc textalloc は、オブジェクトの text および image データの割り当て整合性をチェックし、問題が見つかった場合は、これをレポートし、修正します。dbcc textalloc は、指定したオブジェクトまたは現在のデータベースの text または image データを持つすべてのオブジェクトをチェックして、次の内容を確認します。

- システム・テーブルの text および image エントリが正しいこと
- text および image データの OAM ページ・チェーンおよび OAM ページの割り当てが正しいこと
- すべての text および image ページが正しく割り付けられていること
- 使用されていない text ページが割り付けられていないこと
- 割り付けられていない text ページが使用されていないこと

dbcc textalloc は、パラメータなしで発行できます。デフォルトでは、dbcc textalloc はデフォルト・モードで実行されます。パラメータを指定しない場合、dbcc textalloc は text または image カラムがある現在のデータベースの各テーブルをチェックします。

dbcc textalloc は、異なるオブジェクトで相互に干渉することなく同時に実行できます。

## *fix | nofix* オプションによる割り付けエラーの修正

fix | nofix オプションは、dbcc checkalloc、dbcc tablealloc、textalloc、および dbcc indexalloc で使用でき、これらのコマンドを使用してテーブル内の割り付けエラーを修正するかどうかを指定します。ユーザ・テーブルのデフォルトは fix です。システム・テーブルのデフォルトは nofix です。

システム・テーブルに対して fix オプションを使用するときは、その前に次のコマンドを実行してデータベースをシングルユーザ・モードにしてください。

```
sp_dboption dbname, "single user", true
```

このコマンドはだれもデータベースを使用していないときに発行できます。

fix を指定して dbcc tablealloc コマンドを実行すると、出力には、割り付けエラーと修正されたエラーが表示されます。次に、fix オプションを使用しているかどうかにかかわらず表示されるエラー・メッセージの例を示します。

```
Msg 7939, Level 22, State 1:
Line 2:
Table Corrupt: The entry is missing from the OAM for object
id 144003544 indid 0 for allocation page 2560.
```

fix を使用すると、失われていたエントリがリストアされたことを示す、次のようなメッセージが表示されます。

```
The missing OAM entry has been inserted.
```

fix|nofix オプションは、dbcc indexalloc と同様に、dbcc indexalloc でも機能します。

## *dbcc tablealloc と dbcc indexalloc によるレポートの生成*

dbcc tablealloc または dbcc indexalloc を使用して、次の 3 種類のレポートを生成できます。

- full − すべてのタイプの割り付けエラーのレポートを作成します。full オプションを dbcc tablealloc で使用した場合の結果は、テーブル・レベルで dbcc checkalloc を実行した場合と同じになります。
- optimized − テーブルの OAM ページに登録されているアロケーション・ページを基にレポートを作成します。optimized オプションを使用して dbcc tablealloc を実行したとき、OAM ページに登録されていないアロケーション・ページ上に参照されていないエクステントが存在していても、レポートは行われず、修復はできません。レポート・タイプを指定しない場合や、null を指定した場合は、optimized がデフォルトとなります。
- fast − 参照されているが、エクステントに割り付けられていないページ (2521 レベルのエラー ) の例外レポートを作成します。割り付けのレポートは作成しません。

## システム・テーブルの一貫性の検査

dbcc checkcatalog コマンドは、特定のデータベースにおけるシステム・テーブル内およびシステム・テーブル間の一貫性を検査します。たとえば、次の検査を行います。

- syscolumns の個々のデータ型に対応するエントリが systypes 内に存在すること
- sysobjects のすべてのテーブルとビューが、syscolumns に少なくとも 1 つのカラムを持っていること
- sysindexes が一貫しており、エラーがすべて修正されていること
- syspartitions の各ローに対して、一致するローが sysegments にあること
- syslogs の最後のチェックポイントが有効であること

データベースで使用するように定義されたセグメントがリストされます。また検出されたエラーはすべて修正されます。

データベース名が省略された場合は、dbcc checkcatalog は現在のデータベースを検査します。

```
dbcc checkcatalog (testdb)
Checking testdb
The following segments have been defined for database 5 (database name testdb).
virtual start addr      size      segments
--------------------    ------    --------------------------
33554432                4096       0
                                   1
16777216                102        2
DBCC execution completed. If DBCC printed error messages, see your System
Administrator.
```

# 一貫性検査のコマンドの使用方式

以下の各項では、dbcc コマンドのパフォーマンスを比較し、パフォーマンスへの重大な影響を避けるための計画と方式について提案し、さらに dbcc の出力についての情報を示します。

表 11-2 は、dbcc コマンドを比較したものです。dbcc checkdb、dbcc checktable、dbcc checkcatalog で実行される整合性検査は、dbcc checkalloc、dbcc tablealloc、dbcc indexalloc で実行されるものとは異なることに注意してください。dbcc checkstorage は、他のコマンドで実行される検査のいくつかを組み合わせた検査を行います。各コマンドによって実行される検査の内容は、表 11-1 (254 ページ) を参照してください。

**表 11-2: dbcc コマンドのパフォーマンスの比較**

| コマンド／オプション | レベル | ロックとパフォーマンス | 処理速度 | 完全性 |
|---|---|---|---|---|
| checkstorage | アロケーション・ページのページ・チェーンとデータ・ロー、インデックスのページ間リンク、OAM ページ、デバイスと分割の情報 | ロックしない。大量の I/O を行い、システム I/O を飽和状態にする。専用キャッシュを使用することにより、他のキャッシュへの影響を最小にできる。 | 高速 | 高 |
| checktable checkdb | すべてのインデックスのページ・チェーン、ソート順、データ・ロー、分割の情報 | 共有テーブル・ロック。dbcc checkdb は、一度に 1 つのテーブルをロックし、そのテーブルの検査が完了するとロックを解放する。 | 低速 | 高 |
| skip_ncindex オプション付き checktable checkdb | テーブルとクラスタード・インデックスのページ・チェーン、ソート順、データ・ロー | 共有テーブル・ロック。dbcc checkdb は、一度に 1 つのテーブルをロックし、そのテーブルの検査が完了するとロックを解放する。 | skip_ncindex オプションを指定しない場合より最大 40% 速くなる。 | 中 |
| checkalloc | ページ・チェーンと分割の情報 | ロックしない。大量の I/O を行い、I/O 呼び出しを飽和状態にする。アロケーション・ページのみがキャッシュされる。 | 低速 | 高 |
| full オプション付き tablealloc、indexalloc、および textalloc | ページ・チェーン | 共有テーブル・ロック。大量の I/O を行う。アロケーション・ページのみがキャッシュされる。 | 低速 | 高 |
| optimized オプション付き tablealloc、indexalloc、および textalloc | アロケーション・ページ | 共有テーブル・ロック。大量の I/O を行う。アロケーション・ページのみがキャッシュされる。 | 普通 | 中 |
| fast オプション付き tablealloc、indexalloc、および textalloc | OAM ページ | 共有テーブル・ロック。 | 高速 | 低 |
| checkcatalog | システム・テーブルのロー | システム・カタログに対する共有ページ・ロック。各ページの検査後にロックを解放する。ページ・キャッシュはほとんど行われない。 | 普通 | 中 |

## 大容量 I/O と非同期プリフェッチの使用方法

いくつかの dbcc コマンドでは、大容量 I/O と非同期プリフェッチを使用できます。ただし、検査対象のデータベースまたはオブジェクトが使用するキャッシュにこれらのオプションが設定されている場合に限ります。

大容量 I/O が設定されたキャッシュを使用するテーブルについては、dbcc checkdb と dbcc checktable でそのテーブルのページ・チェーンを検査するときに、大容量 I/O プールが使用されます。この場合、使用可能な I/O サイズのうち最大のものが使用されます。dbcc でインデックスを検査するときは、2K バッファだけが使用されます。

dbcc checkdb と dbcc checktable、および dbcc の割り付け検査コマンドである checkalloc、tablealloc、indexalloc では、使用しているプールで非同期プリフェッチが使用可能ならば、非同期プリフェッチが使用されます。『パフォーマンス＆チューニング・シリーズ：クエリ処理と抽象プラン』の「第 6 章 非同期プリフェッチのチューニング」の「dbcc 用の非同期プリフェッチ制限値の設定」を参照してください。

キャッシュをバインドするコマンドと、プールのサイズおよび非同期プリフェッチ率を変更するコマンドは動的コマンドです。ユーザ・アプリケーションへの影響が小さいオフピーク時にこれらの dbcc コマンドを使用する場合は、これらの設定を変更すると dbcc のパフォーマンスを高速化できます。dbcc の検査が終了したら、通常の設定に戻します。「第 4 章 データ・キャッシュの設定」を参照してください。

## 使用しているシステムにおけるデータベース管理のスケジュール

次の項では、dbcc コマンドの実行頻度や実行するコマンドの種類を決定するいくつかの要因について説明します。

### データベースの使用状況

Adaptive Server が主に月曜日から金曜日までの午前 8 時から午後 5 時の時間帯に使用される場合は、夜間または週末に dbcc による検査を行うようにすれば、一般ユーザへの影響が少なくなります。テーブルが極端に大きくなければ、すべての dbcc コマンドは頻繁に実行できます。

dbcc checkstorage と dbcc checkcatalog は、最小のコストで最大範囲の検査を行い、バックアップからのリカバリを保証するのに十分な範囲の検査を行います。dbcc checkdb または dbcc checktable によるインデックスのソート順と一貫性の検査は、それほど頻繁に行う必要はありません。この検査は、他のデータベース管理作業に合わせて行う必要はありません。オブジェクト・レベルの dbcc 検査と fix オプション付きの検査は、より精密な診断や、dbcc checkstorage で発見されたフォールトの修正が必要になったときに行うようにしてください。

Adaptive Server が、週 7 日間、終日使用される場合は、ワーカー・プロセスの数を制限するか、アプリケーション・キューを使用することにより、dbcc checkstorage のリソース使用を制限することを検討します。dbcc checkstorage を使用しない場合は、dbcc checktable、dbcc tablealloc、dbcc indexalloc を使用して個々のテーブルとインデックスを順に検査するスケジュールを作成します。検査が一巡して、すべてのテーブルが検査済みの状態になったら、dbcc checkcatalog を実行し、データベースをバックアップしてください。『パフォーマンス&チューニング・シリーズ：基本』の「第 5 章 タスク間でのエンジン・リソースの分配」を参照してください。

高いパフォーマンスを要求される 24 時間運用のシステムでは、次の方法で dbcc 検査を実行します。

- データベースをテープにダンプします。
- このデータベース・ダンプを別の Adaptive Server にロードして複製データベースを作成します。
- 複製データベース上で dbcc コマンドを実行します。
- fix オプションによって修復できるエラーが見つかった場合は、元のデータベースの該当するオブジェクトに対して fix オプション付きの dbcc コマンドを実行します。

ダンプはデータベース・ページの論理コピーであるため、元のデータベースに存在する問題は複製データベースにも存在することになります。この方式は、ダンプを使用して、レポートの作成などの他の目的のために複製データベースを作成する場合に役立ちます。

オブジェクトをロックする dbcc コマンドは計画を立てて使用し、通常業務を中断させることのないようにしてください。たとえば、dbcc checkdb は、データベースの検査が実行されているテーブルごとにロックを取得し、検査が終了して次のテーブルに検査が移行すると、ロックを解放します。これらのテーブルは、dbcc checkdb がロックを保持している間はアクセスできません。テーブルのロックが必要な他の業務が存在する場合は、dbcc checkdb (または、同様の影響を持つ他の dbcc コマンド) の実行をスケジュールしないでください。

## バックアップのスケジュール

データベースのバックアップとトランザクション・ログのダンプを頻繁に行うと、障害時により多くのデータを復旧できます。障害が発生した場合に失っても問題のないデータ量を決定しておき、その決定をサポートするダンプ計画を作成する必要があります。

ダンプのスケジュールを作成した後、そのスケジュールに dbcc コマンドを組み込む方法を決定してください。各ダンプを実行する前に dbcc 検査を行うことは必須ではありません。ただし、ダンプの実行中に破壊が発生した場合は、追加データが失われることがあります。

データベースをダンプするのに理想的な時期は、dbcc checkstorage と dbcc checkcatalog を使用して、そのデータベースを完全に検査した後です。これらのコマンドでデータベース内にエラーが見つからなかった場合は、バックアップしたデータベースにはまったくエラーがありません。ダンプのロード後に発生した問題は、インデックスの再作成によって修正できます。個々のテーブルやインデックス上で dbcc tablealloc や indexalloc を使用して、dbcc checkalloc によってレポートされた割り付けエラーを修正してください。

## テーブルのサイズについて、およびデータの重要性について

使用しているデータについて、次のことを確認してください。

- きわめて重要なデータが格納されているテーブルの数
- そのデータの変更頻度
- それらのテーブルのサイズ

dbcc checkstorage は、データベースレベルのオペレーションです。重要なデータや頻繁に変更されるデータが格納されているテーブル数が少ない場合は、データベース全体に対して dbcc checkstorage を実行するよりも頻繁に、テーブル・レベルやインデックス・レベルの dbcc コマンドを実行します。

## データベースの一貫性の問題によって発生するエラー

dbcc checkstorage を使用して見つかったデータベースの一貫性の問題によって発生するエラーは、dbcc_types テーブルに記録されます。ほとんどのエラーは、5010 ～ 5024 と 100,000 ～ 100,038 の範囲です。特定のエラーの説明については、『リファレンス・マニュアル：テーブル』の「第 2 章 dbccdb のテーブル」を参照してください。

dbcc checkstorage 以外の dbcc コマンドによって発見されるデータベースの一貫性の問題によって発生するエラーは、通常は番号が 2500 ～ 2599 または 7900 ～ 7999 のエラーです。このようなメッセージと、データベースの一貫性の問題により出力される他のメッセージ ( メッセージ 605 など ) には、「テーブルが破損しました」や「エクステントがセグメント内にありません」という語句が含まれていることがあります。

メッセージの中には、データベースの一貫性に深刻な問題があることを示すものもありますが、それほど急を要さないものもあります。Sybase 製品の保守契約を結んでいるサポート・センタのサポートを必要とする問題は少なく、多くは次の操作で解決できます。

- fix オプションを付けて dbcc コマンドを実行します。
- 『ASE トラブルシューティング＆エラー・メッセージ・ガイド』の指示に従います。このガイドには、多数の dbcc データベース・エラーを解決するための手順が記載されています。

問題の解決に要する手法がどのようなものであっても、破壊や矛盾が発生した直後に問題を早期発見すれば、それだけ解決が容易になります。一貫性の問題は、使用頻度が低いデータ・ページ、たとえば、1 か月に一度しか更新しないテーブルなどに存在することもあります。dbcc を使用することでこのような問題を発見でき、多くの場合は解決できます。

## アボートされた checkstorage と checkverify オペレーションに関するレポート

checkstorage または checkverify オペレーションがアボートされるとき、オペレーション ID (opid) と、オペレーションのアボート時に調査されたデータベース名を含むメッセージが出力されます。checkverify オペレーションがアボートされた場合は、メッセージにシーケンス番号も出力され、指定された *dbname* と opid、さらに (checkverify オペレーションの場合は) シーケンス番号 *seq* を使用して sp_dbcc_patch_finishtime を実行するようにユーザに指示します。sp_dbcc_patch_finishtime を実行した後、アボートされたオペレーションに関するフォールト・レポートを作成できます。

### エラー 100032 が原因のアボート

checkstorage は、ページ・リンク・エラー (100032) が発生すると、オブジェクトの検査をアボートする場合があります。

checkstorage は、最新バージョンのページを指定することでこのページ・リンク・エラーを解決できるか、またはページ・リンク・エラー数が設定した最大リンク・エラー値より少ないかどうかについてオブジェクトを検証します。

最大リンク・エラー値を設定するには、sp_dbcc_updateconfig を使用します。この例では、great_big_db の値を 8 に設定します。

```
sp_dbcc_updateconfig great_big_db, "linkage error abort", "8"
```

『リファレンス・マニュアル：ビルディング・ブロック』の「第 4 章 dbcc ストアド・プロシージャ」を参照してください。

checkstorage は、以下の場合に、ページ・リンク・エラーが発生する前でも検査をアボートする場合があります。

- checkstorage が残りのページ・チェーンにアクセスできないため、APL インデックスを使用するオブジェクトの同時更新によってページ・リンクが妨害された場合
- ページ・リンクの検査中にインデックスが削除された場合

閑散時に checkstorage を実行すると、インデックスの検査のアボートの原因となる一時的なエラーが減少 (解消) します。一時的なエラーを解消するには、dbcc checkstorage を実行してすぐに checkverify を実行します。

## ソフト・フォールトとハード・フォールトの比較

dbcc checkstorage は、ターゲット・データベース内でフォールトを発見すると、そのフォールトを「ソフト・フォールト」または「ハード・フォールト」として dbcc_faults テーブルに記録します。

### ソフト・フォールト

ソフト・フォールトは、Adaptive Server における矛盾であり、通常は永続性のないものを指します。ソフト・フォールトの多くは、ターゲット・データベース内の一時的な矛盾によるもので、この矛盾は、dbcc checkstorage のオペレーション中にユーザがターゲット・データベースを更新したときや、dbcc checkstorage の実行中にデータ定義言語 (DDL: data definition language) が実行されたときに発生します。これらのフォールトは、そのコマンドをもう一度実行したときには発生しません。ソフト・フォールトを再分類するには、2 回の dbcc checkstorage の実行結果を比較します。または、dbcc checkstorage によってソフト・フォールトが検出された後で、dbcc tablealloc や dbcc checktable を実行します。

dbcc checkstorage を再び実行したときも同じソフト・フォールトが起きる場合は、「永続的な」ソフト・フォールトと見なされます。このフォールトは、破壊の発生を示すことがあります。dbcc checkstorage がシングルユーザ・モードで実行された場合は、レポートされたソフト・フォールトは「永続的な」ソフト・フォールトです。これらのフォールトは、sp_dbcc_differentialreport を使用するか、dbcc tablealloc と dbcc checktable を実行することにより解決できます。dbcc tablealloc コマンドと dbcc checktable コマンドを使用する場合は、ソフト・フォールトが発生したテーブルやインデックスだけを検査してください。

### ハード・フォールト

ハード・フォールトは、Adaptive Server の永続的な破壊を指します。これは、Adaptive Server を再起動しても修正できません。すべてのハード・フォールトが同じように深刻なわけではありません。たとえば、以下の状況はいずれもハード・フォールトの原因となりますが、結果は異なります。

- 存在しないテーブルに割り付けられたページにより、使用可能なディスク記憶領域がわずかに減少します。
- テーブルのローの一部がスキャンできない場合に、誤った結果が返されることがあります。
- 別のテーブルへリンクされたテーブルにより、クエリが停止されます。

ハード・フォールトの中には、影響を受けたテーブルのトランケートなどの簡単な処理で修正できるものもあります。また、バックアップからデータベースをリストアするしか修正方法がないものもあります。

# *dbcc checkverify* によるフォールトの検証

dbcc checkverify は、最新の checkstorage オペレーションの結果を検査して、個々のソフト・フォールトをハード・フォールトと無視できるフォールトのいずれかに再分類します。checkverify は、checkstorage の結果から実際にはフォールトではないものを除去する第 2 のフィルタとして機能します。

## *dbcc checkverify* の動作

checkverify は、記録されたフォールトを dbcc_faults から読み出し、checkstorage オペレーションで使用されるのと同様のプロシージャを使用して個々のソフト・フォールトを解決します。

---

**注意** checkverify は、同時更新ができないようにテーブルをロックするので、ソフト・フォールトが正しく再分類されます。checkverify は、checkstorage の最後の実行以降に発生したエラーは検出しません。

---

checkverify は、checkstorage と同様に dbcc_operation_log テーブルと dbcc_operation_results テーブルに情報を記録します。記録される opid の値は、最後に実行された checkstorage オペレーションの opid の値と同じです。checkverify は、dbcc_faults テーブル内の status カラムを更新し、処理するフォールトに対応するローを dbcc_fault_params テーブルに挿入します。

checkverify は、scan 作業領域や text 作業領域は使用しません。

checkstorage で発見されたフォールトを checkverify を使用して検証すると、その結果は次のいずれかとなります。

- checkstorage で分類されたものと同様のハード・フォールト。
- 同時処理がフォールトの原因であるとして禁止されたために、checkverify によってハード・フォールトとして再分類されたソフト・フォールト。
- checkverify によってソフト・フォールトであると確認されたソフト・フォールト。データベース内で同時処理が行われていないときに発生するソフト・フォールトには、明らかな危険を示さないものがあり、これらはハード・フォールトとしては再分類されません。情報を示すだけで破壊ではないソフト・フォールトは再分類されません。
- 同時処理が原因であるか、以降の処理によって当初の矛盾が検出できなくなったために、無視できるフォールトとして再分類されたソフト・フォールト。

checkstorage によってコード 100011 (テキスト・ポインタ・フォールト) を割り当てられたフォールトは、テキスト・カラムにハード・フォールトがある場合にはハード・フォールトに分類されます。テキスト・カラムにハード・フォールトがなければ、ソフト・フォールトに再分類されます。

checkstorage によってコード 100016 ( ページは割り付けられているがリンクされていない ) を割り当てられたフォールトは、連続した 2 つの checkstorage オペレーションで同じフォールトが発生した場合にはハード・フォールトに分類されます。そうでない場合は、ソフト・フォールトに再分類されます。

checkstorage によってコード 100035 (スペースビットの不一致) を割り当てられたフォールトがハード・フォールトに分類された場合は、dbcc checktable を使用して修復できます。

checkverify によって、データベース内にハード・フォールトがあることが確認された場合は、適切な手順に従ってフォールトを修正します。

checkverify でソフト・フォールトに分類されるフォールトは次のとおりです。

- 100020 − 検査がアボートされた。
- 100025 − ロー・カウントにフォールトがある。
- 100028 − 現在のセグメントでページ割り付けがオフになっている。

## *dbcc checkverify* を使用する時期

checkstorage を実行した後、checkverify を実行して永続的なフォールトを検証します。これは数時間あるいは数日経った後でも可能です。ただし、計画を決定する際には、時間が経つとデータベースのステータスが変わり、その変更によってソフト・フォールトとハード・フォールトの両方がマスクされるということを考慮に入れておいてください。

たとえば、テーブルにリンクされているが割り付けられていないページは、ハード・フォールトです。テーブルが削除されると、フォールトは解決されて検出できなくなります。そのページが別のテーブルに割り付けられると、フォールトは残りますが、シグニチャが変更されます。このとき、そのページが 2 つの異なるテーブルにリンクされているように見えます。このページが同じテーブルに再び割り付けられると、ページ・チェーンが破壊されているというフォールトが発生します。

永続的なフォールトが、以降のデータベースの変更によって修正された場合は、通常は運用上の問題は発生しません。しかし、これらのフォールトを検出してすぐに検証を行うようにすれば、より深刻な問題が発生したり、元のフォールトのシグニチャが変更されたりする前に、破壊の原因を特定できることがあります。したがって、Sybase では、dbcc checkstorage の実行後、できるだけ早く checkverify を実行することをおすすめします。

**注意**　シングルユーザ・モードのターゲット・データベースで checkstorage を実行すると、一時的なソフト・フォールトは発生しないため checkverify を実行する必要はありません。

checkverify は、checkstorage を 1 回実行するたびに 1 回だけ実行します。ただし、checkverify が中断されて完了しなかった場合は、再度実行できます。オペレーションは中断したところから再開します。

checkverify で非常に大きなデータベースを処理する場合は、完了までに時間がかかります。その間、checkverify がいつ完了するかを通知するメッセージなどは表示されません。checkverify の実行中に進行状況レポートを確認するには、command_status_reporting コマンドを使用します。

```
set command_status_reporting on
```

checkverify を実行すると、結果は次のようになります。

```
dbcc checkverify (pubs2)
Verifying faults for ’pubs2’.
Verifying faults for table 't1'. The total number of tables to verify is 5. This
is table number 1.
Verifying faults for table 't2'. The total number of tables to verify is 5. This
is table number 2.
Verifying faults for table 't3'. The total number of tables to verify is 5. This
is table number 3.
Verifying faults for table 't4'. The total number of tables to verify is 5. This
is table number 4.
Verifying faults for table 't5'. The total number of tables to verify is 5. This
is table number 5.
DBCC CHECKVERIFY for database ’pubs2’ sequence 4 completed at Apr  9 2003
2:40PM. 72 suspect conditions were resolved as faults, and 11 suspect conditions
were resolved as harmless. 0 objects could not be checked.
```

### *dbcc checkverify* の使用方法

checkverify は、指定のデータベースに対する、最後に完了した checkstorage オペレーションの結果のみを処理します。

checkverify オペレーションが完了すると、Adaptive Server から次のメッセージが返されます。

```
DBCC checkverify for database name, sequence
n completed at date time. n suspect conditions resolved as
faults and n resolved as innocuous.
n checks were aborted.
```

sp_dbcc_runcheck を使用すると、checkstorage の実行後に自動的に checkverify を実行できます。

checkverify からテーブル、フォールト、フォールトとテーブルの組み合わせを除外できます。checkverify から除外する項目は、sp_dbcc_exclusions を使用して指定します。sp_dbcc_exclusions は動的です。つまり、checkverify によって、sp_dbcc_exclusions で指定した項目はすぐに除外されます。checkverify からこれらのオブジェクトをいったん除外すると、2 回目のコマンド実行時には除外したオブジェクトに関してレポートされません。

## *dbcc checkstorage* を使用するための準備

dbcc checkstorage を使用するには、Adaptive Server を設定し、dbccdb データベースをセットアップする必要があります。表 11-3 は、手順とコマンドを使用する順にまとめたものです。

それぞれのアクションについての詳細は、以降の項を参照してください。この項の例では、2K の論理ページを使用しているサーバを想定します。

**表 11-3: dbcc checkstorage を使用するための準備**

| アクションの内容 | 参照箇所 | 使用するコマンド |
|---|---|---|
| 1. ターゲット・データベースに応じたデータベース・サイズ、デバイス (dbccdb データベースが存在しない場合)、作業領域サイズ、キャッシュ・サイズ、ワーカー・プロセス数の推奨値を取得する。 | 「リソースの計画」(279 ページ)<br>「作業領域サイズの計画」(281 ページ) | use master<br>sp_plan_dbccdb |
| 2. 必要であれば、Adaptive Server が使用するワーカー・プロセス数を調整する。 | 「ワーカー・プロセスの設定」(283 ページ) | sp_configure<br>number of worker processes<br>memory per worker processes |
| 3. dbcc 用の名前付きキャッシュを作成する (省略可能)。 | 「dbcc が使用する名前付きキャッシュの設定」(284 ページ) | sp_cacheconfig |

| アクションの内容 | 参照箇所 | 使用するコマンド |
|---|---|---|
| 4. 使用するバッファ・プールを設定する。 | 「8 ページ I/O バッファ・プールの設定」(285 ページ) | sp_poolconfig |
| 5. dbccdb が既に存在する場合は、新しい dbccdb データベースを作成する前に、dbccdb および関連するすべてのデバイスを削除する。 | | drop database |
| 6. dbccdb のデータとログ用のディスク・デバイスを初期化する。 | 「dbccdb 用のディスク領域の割り付け」(286 ページ) | disk init |
| 7. データ・ディスク・デバイス上に dbccdb を作成する (省略可能)。 | | create database |
| 8. ディスク・セグメントを追加する (省略可能)。 | 「作業領域用のセグメント」(286 ページ) | use dbccdb |
| 9. dbccdb データベースを移植して、dbcc のストアド・プロシージャをインストールする。 | | `isql -Usa -P -i $SYBASE/$SYBASE_ASE/scripts/installdbccdb` |

**注意**　dbcc checkstorage は、ディスク上のデータベースに対して検査を実行するコマンドです。破損がメモリ内だけであれば、dbcc を実行してもその破損を検出できない場合があります。dbcc checkstorage コマンドを 2 回連続して実行した場合に、これらの結果が矛盾しないようにするには、最初に checkpoint を実行します。ただし、checkpoint を実行すると、メモリ内の一時的な破損がディスクに反映されてしまう場合があります。

## リソースの計画

dbccdb のデバイスとサイズを正しく選択することは、dbcc checkstorage オペレーションのパフォーマンスの観点からは重要なことです。システム・プロシージャ sp_plan_dbccdb を実行すると、dbccdb が存在するかどうかに応じて、指定したターゲット・データベースの推奨設定値が表示されます。この情報を使用して、Adaptive Server を設定し、dbccdb データベースをセットアップしてください。

### *sp_plan_dbccdb の出力例*

dbccdb が存在しない場合、sp_plan_dbccdb は次の情報を返します。

- dbccdb の最小サイズ
- dbccdb に適したデバイス
- scan 作業領域と text 作業領域の最小サイズ
- 最小キャッシュ・サイズ
- ワーカー・プロセス数

ターゲット・データベース内のページ割り付けパターンによって dbccdb に適したキャッシュ・サイズが異なるため、キャッシュ・サイズの推奨値はおおよその値です。次の例は、dbccdb が存在しない場合の、pubs2 データベースに対する sp_plan_dbccdb の出力を示しています。

```
sp_plan_dbccdb pubs2

Recommended size for dbccdb is 4MB.

Recommended devices for dbccdb are:

Logical Device Name   Device Size   Physical Device Name

sprocdev              28672         /remote/SERV/sprocs_dat
tun_dat               8192          /remote/SERV/tun_dat
tun_log               4096          /remote/SERV/tun_log

Recommended values for workspace size, cache size and process count are:

dbname        scan ws    text ws    cache    process count

pubs2        64K        64K        640K    1

(return status = 0)
```

dbccdb が既に存在している場合、sp_plan_dbccdb は次の情報を返します。

- dbccdb の最小サイズ
- 既存の dbccdb データベースのサイズ
- scan 作業領域と text 作業領域の最小サイズ
- 最小キャッシュ・サイズ
- ワーカー・プロセス数

次の例は、dbccdb が既に存在する場合の、pubs2 データベースに対する sp_plan_dbccdb の出力を示します。

```
sp_plan_dbccdb pubs2

Recommended size for dbccdb database is 23MB (data = 21MB, log = 2MB).
dbccdb database already exists with size 8MB.
Recommended values for workspace size, cache size and process count are:
dbname                          scan ws    text ws    cache    process count
pubs2                          64K        48K        640K    1
(return status = 0)
```

複数のデータベースを検査する場合は、ターゲット・データベースには最も大きいデータベースの名前を使用します。ターゲット・データベース名を指定しない場合は、sp_plan_dbccdb は master..sysdatabases に登録されているすべてのデータベースの設定値を返します。

```
sp_plan_dbccdb

Recommended size for dbccdb is 4MB.

dbccdb database already exists with size 8MB.

Recommended values for workspace size, cache size and process count are:

dbname            scan ws    text ws    cache    process count

master             64K        64K        640K     1
tempdb             64K        64K        640K     1
model              64K        64K        640K     1
sybsystemprocs    384K       112K       1280K     2
pubs2              64K        64K        640K     1
pubs3              64K        64K        640K     1
pubtune           160K        96K       1280K     2
sybsecurity        96K        96K       1280K     2
dbccdb            112K        96K       1280K     2
```

『リファレンス・マニュアル：プロシージャ』の「sp_plan_dbccdb」を参照してください。

## 作業領域サイズの計画

dbccdb を実行するには、scan と text の 2 つの作業領域が必要です。これらの作業領域に必要な領域サイズは、検査する最も大きいデータベースのサイズによって決まります。また、複数の dbcc checkstorage オペレーションを同時に実行する場合は、追加の作業領域を設定します。

### 検査される最も大きいデータベースのサイズの決定

さまざまなデータベースが同じ作業領域を使用できます。このため、作業領域の大きさは、その作業領域を使用するデータベースのうち、最大のものを収容できる大きさにしてください。

---

**注意**　sp_plan_dbccdb を実行すると作業領域サイズの推奨値が表示されます。つまり、以下で説明している作業領域サイズの決定方法は、単なる参考情報です。

---

ターゲット・データベース内の各ページは、scan 作業領域内では長さ 18 バイトの 1 つのローによって表されます。この作業領域は、ターゲット・データベース・サイズの約 1.1% にしてください。

ターゲット・データベース内の非 null の text カラムごとに、22 バイトのローが text 作業領域内に挿入されます。ターゲット・データベース内に非 null の text 型カラムが *n* 個ある場合は、text 作業領域のサイズは (22 * *n*) バイト以上になるようにしてください。非 null の text カラムの数は変動するので、将来的な必要性に適応できるように、text 作業領域には十分な領域を割り付けてください。text 作業領域に必要な最小領域サイズは 24 ページです。

### 同時に使用できる作業領域数

複数のデータベース上で dbcc checkstorage を同時に実行できるように dbccdb データベースを設定できます。これは、2 番目以降の dbcc checkstorage オペレーションが専用のリソースを持っている場合にだけ可能です。複数の dbcc checkstorage オペレーションを同時に実行するには、各オペレーション専用の scan 作業領域と text 作業領域、ワーカー・プロセス、予約済みキャッシュが必要です。

作業領域に必要な合計領域サイズは、ユーザ・データベースを逐次検査するか、並行して検査するかによって異なります。複数の dbcc checkstorage オペレーションを逐次に実行する場合は、最大の scan 作業領域と text 作業領域をすべてのユーザ・データベースに使用できます。複数の dbcc checkstorage オペレーションを同時に実行する場合は、同時に使用される作業領域の中で最も大きい作業領域を収容できるように dbccdb データベースの大きさを設定してください。同時に検査する最大のデータベースのサイズを加算して、作業領域のサイズを求めてください。

『リファレンス・マニュアル：テーブル』の「dbccdb のテーブル」を参照してください。

### 作業領域の自動拡張

sp_dbcc_updateconfig. . automatic workspace expansion オプションを使用することで、checkstorage で作業領域のサイズを必要に応じて自動的に変更するかどうかを指定します。

checkstorage を実行すると、最初に作業領域のサイズが検証されます。より多くの領域が必要で、automatic workspace expansion が有効な場合は、各セグメントに適切な領域があれば、作業領域は自動的に checkstorage によって拡張されます。より多くの領域が必要で、automatic workspace expansion が無効な場合は、checkstorage は終了し、情報を知らせるメッセージが出力されます。

1 (デフォルト値) の値を指定すると、automatic workspace expansion が有効になります。0 の値を指定すると、automatic workspace expansion が無効になります。『リファレンス・マニュアル：プロシージャ』を参照してください。

### デフォルトのスキャン作業領域とテキスト作業領域

デフォルトのスキャンおよびテキスト作業領域は、installdbccdb スクリプトの実行時に作成されます。デフォルトのスキャン作業領域の名前は def$scan$ws で、サイズは 256K です。デフォルトのテキスト作業領域の名前は def$text$ws で、サイズは 128K です。デフォルトの作業領域が設定されていなく、ターゲット・データベースに作業領域の値が設定されていない場合、これらのデフォルトの作業領域が使用されます。

## ワーカー・プロセスの設定

次の各パラメータは、dbcc checkstorage に影響します。

- max worker processes － このパラメータは、sp_dbcc_updateconfig を使用して設定します。これは、各ターゲット・データベースに対応する dbcc_config テーブル内の max worker processes の値を更新します。
- number of worker processes － この設定パラメータは、sp_configure を使用して設定します。これは、*server_name.cfg* ファイルを更新します。
- memory per worker process － この設定パラメータは、sp_configure を使用して設定します。これは、*server_name.cfg* ファイルを更新します。

max worker processes パラメータは、dbcc checkstorage によって使用される、ターゲット・データベースごとのワーカー・プロセスの最大数を指定します。number of worker processes パラメータは、Adaptive Server がサポートするワーカー・プロセスの総数を指定します。ワーカー・プロセスは、dbcc checkstorage オペレーション専用ではありません。

number of worker processes の値には、max worker processes によって指定されているプロセス数の合計を見込んだ大きい値を設定してください。ワーカー・プロセス数が少ないと、dbcc checkstorage のパフォーマンスとリソース消費量が低下します。dbcc checkstorage が使用するプロセス数は、そのデータベースによって使用されるデータベース・デバイス数を超えることできません。キャッシュ・サイズ、CPU パフォーマンス、デバイス・サイズによっては、ワーカー・プロセス数がそれより少なくても済むこともあります。Adaptive Server に設定されているワーカー・プロセス数が十分でないときは、dbcc checkstorage は実行されません。

設定パラメータ maximum parallel degree と maximum scan parallel degree は、dbcc checkstorage の並列機能には影響を与えません。maximum parallel degree が 1 に設定されていても、dbcc checkstorage の並列処理ができなくなることはありません。

dbcc checkstorage は複数のプロセスを必要とするので、number of worker processes は、親プロセスとワーカー・プロセスを 1 つ実行できるように、最低でも 1 に設定する必要があります。

sp_plan_dbccdb は、データベース・サイズ、デバイス数などの要素に従って、ワーカー・プロセス数の推奨値を示します。システムに負荷がかかりすぎないようにするために、ワーカー・プロセス数を少なくすることもできます。dbcc checkstorage が使用するワーカー・プロセス数は、sp_plan_dbccdb の推奨値や設定値よりも少ないことがあります。

使用するワーカー・プロセス数を増やせばパフォーマンスが向上するとは限りません。次の例では、2 つの異なる設定の効果を説明します。

8 GB のデータベースのデータのうち、ディスク A に 4 GB があり、ディスク B、C、D、E、F、G、H、I にそれぞれ 0.5 GB があるとします。

9 つのワーカー・プロセスがアクティブな場合に、dbcc checkstorage の実行に要する時間が 2 時間であるとします。これは、ディスク A の検査に要する時間です。他の 8 つのワーカー・プロセスは 15 分で完了し、ディスク A のワーカー・プロセスが完了するのを待ちます。

2 つのワーカー・プロセスがアクティブな場合も、dbcc checkstorage の実行に要する時間は 2 時間です。最初のワーカー・プロセスはディスク A を処理し、他のワーカー・プロセスはディスク B、C、D、E、F、G、H、I を処理します。この場合、待機は発生せず、リソースがより効率的に使用されます。

memory per worker process は、Adaptive Server でサポートされるワーカー・プロセスの合計メモリ割り付けを指定します。デフォルト値は、dbcc checkstorage に適切な値です。

## *dbcc* が使用する名前付きキャッシュの設定

dbcc checkstorage に名前付きキャッシュを使用する場合は、Adaptive Server の設定パラメータを調整しなければならないことがあります。

dbcc checkstorage のオペレーション中は、作業領域はターゲット・データベースを読み込むキャッシュに一時的にバインドされます。dbcc 専用の名前付きキャッシュを使用すると、データベースの検査が他のユーザに与える影響が最小限になり、パフォーマンスが向上します。同時に実行される dbcc checkstorage オペレーションごとに個別のキャッシュを作成するか、同時処理の合計必要量を満たす大きさのキャッシュを 1 つ作成します。パフォーマンスを最適化するために必要なサイズは、ターゲット・データベースのサイズとそのデータベース内でのデータの分布によって異なります。dbcc checkstorage は、名前付きキャッシュ内で、ワーカー・プロセス 1 つあたり最低 640K のバッファ (各バッファは 1 エクステント) を必要とします。

最大限のパフォーマンスを引き出すには、専用キャッシュのほとんどをバッファ・プールに割り当てます。キャッシュは分割しません。キャッシュ・サイズの推奨値は、バッファ・プールの最小サイズです。1 ページ・プールのサイズをこの値に加算します。

dbcc checkstorage に専用のキャッシュを割り付けた場合は、このコマンドの実行時に最小の 1 ページ・バッファ・プールより大きなサイズが要求されることはありません。キャッシュを共有する場合は、オペレーションを実行する前にバッファ・プールのサイズを増やし、処理が完了したら値を減らすことで、dbcc checkstorage のパフォーマンスを向上できます。バッファ・プールの要件は、共有キャッシュと同じです。ただし、共有キャッシュのサイズが要件を満たしていても、他の処理でそのキャッシュが要求されるときは、dbcc checkstorage が使用できるバッファが制限され、checkstorage と Adaptive Server 全体の両方のパフォーマンスに大きな影響を与えることがあります。

---

**注意**　dbcc checkstorage に専用のキャッシュ・パーティションを割り付けると、エラー・メッセージ 9946 または 9947 を発行する場合があるため、Sybase では dbcc checkstorage に専用の名前付きキャッシュを割り付けることをおすすめします。

---

dbcc checkstorage のオペレーションに名前付きキャッシュを使用するように Adaptive Server を設定するには、sp_cacheconfig と sp_poolconfig を使用します。「第 4 章 データ・キャッシュの設定」を参照してください。

## 8 ページ I/O バッファ・プールの設定

dbcc checkstorage を実行するには、1 エクステントの I/O バッファ・プールを設定する必要があります。プール・サイズの設定と、プールが正常に設定されているかどうかの確認には、sp_poolconfig を使用します。プール・サイズの情報は、dbcc_config テーブルに格納されます。

dbcc checkstorage バッファ・プールのサイズは、ページ・サイズの 16 倍です。異なるページ・サイズに対する dbcc checkstorage のバッファ・プールの要件は、次のとおりです。

- (2KB ページ・サーバ) * (8 エクステント) = 16k バッファ・プール
- (4KB ページ・サーバ) * (8 エクステント) = 32k バッファ・プール
- (8KB ページ・サーバ) * (8 エクステント) = 64k バッファ・プール
- (16KB ページ・サーバ) * (8 エクステント) = 128k バッファ・プール

次の例は、2K の論理ページを使用するように設定されているサーバ上で master_cache 用に 16K バッファ・プールを設定するための sp_poolconfig の使用方法を示します。この名前付きキャッシュは、master データベース用に作成されています。

```
1> sp_poolconfig "master_cache", "1024K", "16K"
2> go

(return status = 0)
```

次の例は、プライベート・キャッシュ “master_cache” のバッファ・プールが設定されていることを示します。

```
1> sp_poolconfig "master_cache"
2> go
Cache Name     Status    Type     Config Value Run Value
-------------  --------- -------- ------------ ------------
master_cache   Active    Mixed         2.00 Mb      2.00 Mb
                                    ------------ ------------
                          Total        2.00 Mb      2.00 Mb
===========================================================================
Cache: master_cache,   Status: Active,   Type: Mixed
   Config Size: 2.00 Mb,   Run Size: 2.00 Mb
   Config Replacement: strict LRU,   Run Replacement: strict LRU
IO Size  Wash Size Config Size  Run Size     APF Percent
-------- --------- ------------ ------------ -----------
2 Kb     512 Kb    0.00 Mb      1.00 Mb      10
16 Kb    192 Kb    1.00 Mb      1.00 Mb      10
(return status = 0)
```

『リファレンス・マニュアル：プロシージャ』を参照してください。

## *dbccdb* 用のディスク領域の割り付け

dbccdb データベースを格納するための、追加のディスク記憶領域が必要です。dbcc checkstorage は dbccdb を頻繁に使用するので、dbccdb は他のデータベース・デバイスとは別のデバイスに配置してください。

**注意** マスタ・デバイスに dbccdb を作成しないでください。dbccdb 用のログ・デバイスとデータ・デバイスは、必ず別にしてください。

## 作業領域用のセグメント

作業領域に専用のセグメントを割り付けることで、作業領域の配置を制御して、dbcc checkstorage のパフォーマンスを向上できます。作業領域を排他的に使用するために専用のセグメントを新しく割り付ける場合は、sp_dropsegment を使用して、そのセグメントに付加されているデバイスとデフォルト・セグメントとのマッピングを解除してください。

## *dbccdb* データベースの作成

### ❖ *dbccdb* データベースの作成

1. master データベース内で sp_plan_dbccdb を実行して、ターゲット・データベースに応じたデータベース・サイズ、デバイス、作業領域のサイズ、キャッシュ・サイズ、ワーカー・プロセス数の推奨値を取得します。

   sp_plan_dbccdb によって表示される情報の詳細については、「リソースの計画」(279 ページ) を参照してください。

2. dbccdb が既に存在する場合は、新しい dbccdb データベースを作成する前に、dbccdb および関連するすべてのデバイスを削除します。

   ```
   use master
   go
   if exists (select * from master.dbo.sysdatabases
        where name = "dbccdb")
   begin
        print "+++ Dropping the dbccdb database"
        drop database dbccdb
   end
   go
   ```

3. dbccdb のデータとログ用のディスク・デバイスを初期化します。

   ```
   use master
   go
   disk init
      name = "dbccdb_dat",
      physname = "/remote/disks/masters/",
      size = "4096"
   go
   disk init
      name = "dbccdb_log",
      physname = "/remote/disks/masters/",
   size = "1024"
   go
   ```

4. 手順 3 で初期化したデータ・ディスク・デバイス上に dbccdb データベースを作成します。

   ```
   use master
   go

   create database dbccdb
      on dbccdb_dat = 6
      log on dbccdb_log = 2
   go
   ```

5 (この手順は省略可能) scan 作業領域と text 作業領域用のセグメントを dbccdb のデータ・デバイスに追加します。

```
use dbccdb
go
sp_addsegment scanseg, dbccdb, dbccdb_dat
go
sp_addsegment textseg, dbccdb, dbccdb_dat
go
```

6 dbccdb のテーブルを作成し、dbcc_types テーブルを初期化します。

```
isql -Ujms -P***** -iinstalldbccdb
```

installdbccdb は、テーブルの作成を開始する前に、データベースが存在しているかどうかを検査します。このスクリプトは、dbccdb 内に存在していないテーブルだけを作成します。dbccdb のテーブルの中に破損しているものがある場合は、drop table を使用してそのテーブルを削除してから、installdbccdb を使用してテーブルを再作成してください。

7 scan 作業領域と text 作業領域を作成して初期化します。

```
use dbccdb|
go|
sp_dbcc_createws dbccdb, scanseg, scan_pubs2, scan, "64K"|
sp_dbccvreatews dbccdb, textseg, text_pubs2, text, "64K"
```

dbccdb のインストールが終了した後で、dbcc_config テーブルを更新する必要があります。

# *dbcc_config* テーブルの更新

dbcc_config テーブルは、現在実行中の、または最後に完了した dbcc checkstorage オペレーションを記述するもので、次の内容を含みます。

- dbcc checkstorage オペレーション専用リソースのロケーション
- dbcc checkstorage オペレーションでのリソース使用率の制限値

この項では、このテーブルの値を更新する方法について説明します。

## *sp_dbcc_updateconfig* によるデフォルト設定値の追加

sp_dbcc_updateconfig では、dbcc の設定パラメータにデフォルト設定値を指定できます。checkstorage によって分析された個々のデータベースにこれらの設定値を指定するか、対応する設定値がないすべてのデータベースに適用できるデフォルト値を設定できます (つまり、設定値は競合しません)。

### *sp_dbcc_updateconfig* による設定値の削除

sp_dbcc_updateconfig の *str1* パラメータの値に delete を指定できます。これにより、データベースに設定した設定値を削除できます。

たとえば、max worker processes 設定パラメータのデフォルト値を削除するには、次のように入力します。

```
sp_dbcc_updateconfig null, 'max worker processes', 'delete'
```

データベース my_db の max worker processes 設定パラメータの値を削除するには、次のように入力します。

```
sp_dbcc_updateconfig my_db, 'max worker processes', 'delete'
```

### 現在の設定値の表示

sp_dbcc_configreport の *defaults* パラメータによって、デフォルトの設定値を表示できます。*defaults* パラメータは省略可能で、*dbname* が NULL でない場合は無視されます。*defaults* パラメータの有効な値は、true または false (デフォルト) です。true の値は、デフォルトの設定値のみを表示することを指定します。false の値は、すべての設定値を表示することを指定します。

たとえば、デフォルトの設定値のみを表示するには、*defaults* パラメータに true を指定します。

```
sp_dbcc_configreport null, 'true'
```

すべての設定値を表示するには、defaults パラメータの値を指定しないか、“false” を指定します。

```
sp_dbcc_configreport
```

または、

```
sp_dbcc_configreport null, 'false'
```

## *dbccdb* データベースの管理

dbccdb 上で、次のような管理タスクを定期的に実行する必要がある場合があります。

- 次のシステム・プロシージャを使用して、設定を再評価し、更新します。
  - sp_dbcc_evaluatedb － 前回の dbcc checkstorage オペレーションの結果を使用して、設定パラメータの推奨値を表示します。
  - sp_dbcc_updateconfig － 指定のデータベースの設定パラメータを更新します。

- 次のシステム・プロシージャを使用して、古いデータをクリーンアップします。
  - sp_dbcc_deletedb – 指定のデータベースのすべての情報を dbccdb から削除します。
  - sp_dbcc_deletehistory – 指定のデータベースに対する dbcc checkstorage オペレーションの結果を dbccdb から削除します。
- 不要な作業領域を削除します。
- dbccdb データベースそのものの一貫性の検査を実行します。

## *dbccdb* 設定の再評価と更新

ユーザ・データベースの特性が変わった場合は、ストアド・プロシージャ sp_dbcc_evaluatedb を使用して、現在の dbccdb 設定を再評価し、より適切な値を指定します。

ユーザ・データベースの変更は、次のように dbccdb の設定に影響を与えることがあります。

- ユーザ・データベースを作成、削除、変更した場合は、dbcc_config テーブルに格納されている作業領域や名前付きキャッシュのサイズ、またはワーカー・スレッド数が影響を受けることがあります。
- dbcc_checkstorage 用の名前付きキャッシュのサイズ、またはワーカー・プロセス数を変更した場合は、バッファ・キャッシュとワーカー・プロセスを再設定しなければならないことがあります。

ターゲット・データベースの dbcc checkstorage オペレーションの結果を使用できる場合は、sp_dbcc_evaluatedb を使用して新しい設定値を決定します。sp_dbcc_configreport も、指定のデータベースの設定パラメータをレポートします。

sp_dbcc_updateconfig を使用して、新しいデータベースを dbcc_config テーブルに追加します。また、sp_dbcc_evaluatedb による推奨値を反映するように dbcc_config 内の設定値を変更します。

## *dbccdb* のクリーンアップ

Adaptive Server は、dbcc checkstorage で生成されたデータを dbccdb に格納します。sp_dbcc_deletehistory を使用してデータベースからデータを削除することにより、dbccdb を定期的にクリーンアップします。データベースは、指定した日付より前に作成されたものである必要があります。

データベースを削除したときは、そのデータベースに関係するすべての設定情報と dbcc checkstorage の結果を dbccdb から削除してください。dbccdb からデータベース情報をすべて削除するには、sp_dbcc_deletedb を使用します。

## 作業領域の削除

不要な作業領域を削除するには、dbccdb で次の文を発行します。

```
drop table workspace_name
```

## *dbccdb* での一貫性チェックの実行

dbccdb テーブルにおける更新アクティビティを制限することで、破壊の可能性が低くなります。dbccdb における破壊の兆候を次に示します。

- dbcc checkstorage が、初期化フェーズにおいて、実行が必要な処理を評価するときに異常終了する。または、完了フェーズにおいて結果を書き込むときに異常終了する。
- フォールト記録の破壊によるフォールト情報の損失。dbcc checkstorage によって検出される。

dbccdb の破壊の程度が大きい場合は、dbcc checkstorage が異常終了することがあります。dbcc checkstorage を実行して dbccdb 内のひどく破壊された場所を特定するには、dbccdb の検査のみに使用する代替データベース dbccalt を作成します。dbccalt は、dbccdb と同じ方法で作成します。手順については、「dbcc checkstorage を使用するための準備」(278 ページ) を参照してください。

dbccalt 用に使用できる空きデバイスがない場合は、master データベースや dbccdb が使用しているデバイス以外のデバイスであれば使用できます。

dbcc checkstorage と dbcc システム・プロシージャは、dbccalt に対しても dbccdb と同様に機能します。ターゲット・データベースが dbccdb のとき、dbccalt が存在する場合は、dbcc checkstorage はdbccalt を使用します。dbccalt が存在しない場合は、dbccdb 自体を管理データベースとして使用して dbccdb を検査できます。ターゲット・データベースが dbccdb で、dbccalt が存在する場合は、dbccdb に対する dbcc checkstorage オペレーションの結果は dbccalt に格納されます。dbccalt が存在しない場合、結果は dbccdb 自体に格納されます。

また、dbcc checkalloc と dbcc checktable を使用して dbccdb を検査することもできます。

dbccdb が破壊された場合は、削除して、再作成するかバックアップから以前のバージョンをロードします。削除すると、診断履歴の一部が失われます。

# *dbccdb* からのレポートの生成

dbccdb には、dbccdb 内のデータからレポートを生成するための多数の dbcc ストアド・プロシージャがあります。

## *dbcc checkstorage* オペレーションの要約レポート

sp_dbcc_summaryreport は、指定のデータベース上で、指定の日付またはそれ以前に完了したすべての dbcc checkstorage オペレーションをレポートします。次に、このコマンドの出力例を示します。

```
    sp_dbcc_summaryreport

DBCC Operation : checkstorage
Database Name      Start time           End Time   Operation ID
         Hard Faults Soft Faults Text Columns Abort Count
         User Name
------------------ -------------------- ---------- ------------
         ---------- ---------- ------------ -----------
         ------------------------------
sybsystemprocs     05/12/1997 10:54:45  10:54:53              1
         0           0            0           0
     sa
sybsystemprocs     05/12/1997 11:14:10  11:14:19              2
         0           0            0           0
     sa
```

『リファレンス・マニュアル：プロシージャ』の「第 4 章 dbcc ストアド・プロシージャ」を参照してください。

## 設定、統計、フォールト情報のレポート

次の dbcc システム・プロシージャは、設定情報および統計情報についてレポートします。

- sp_dbcc_summaryreport
- sp_dbcc_configreport
- sp_dbcc_statisticsreport
- sp_dbcc_faultreport short

sp_dbcc_fullreport は、これらの dbcc システム・プロシージャのすべてについて、完全なレポートを実行します。

『リファレンス・マニュアル：プロシージャ』を参照してください。

# *dbcc upgrade_object* を使用したコンパイル済みオブジェクトのアップグレード

Adaptive Server は、コンパイル済みオブジェクトをそのソース・テキストに基づいてアップグレードします。コンパイル済みオブジェクトは、次に示すものです。

- 検査制約
- デフォルト
- ルール
- ストアド・プロシージャ (拡張ストアド・プロシージャを含む)
- トリガ
- ビュー

各コンパイル済みオブジェクトのソース・テキストは、手動で削除されていない限り syscomments テーブルに格納されます。サーバをアップグレードする場合は、ソース・テキストが syscomments に存在するかどうか、そのプロセスで検証されます。ただし、コンパイル済みオブジェクトは、それらが呼び出されるまで実際にはアップグレードされません。

たとえば、list_proc というユーザ定義のストアド・プロシージャがあるとします。最新バージョンの Adaptive Server へアップグレードするときに、list_proc のソース・テキストが存在するかどうか検証されます。アップグレード後、最初に list_proc が呼び出されると、Adaptive Server はコンパイル済みオブジェクトである list_proc がアップグレードされていないことを検出します。Adaptive Server は、syscomments 内のソース・テキストに基づいて list_proc を再コンパイルします。次いで、その新しいコンパイル済みオブジェクトが実行されます。

アップグレードされたオブジェクトは、アップグレードの前に使用していたのと同じオブジェクト ID とパーミッションを持ちます。

sp_hidetext を使用してソース・テキストが隠されているコンパイル済みオブジェクトも、ソース・テキストが隠されていないオブジェクトと同様にアップグレードされます。sp_hidetext については、『リファレンス・マニュアル：プロシージャ』を参照してください。

---

**注意** 32 ビットのインストール環境をアップグレードして 64 ビットの Adaptive Server を使用するようにすると、各データベースの sysprocedures テーブルに含まれている 64 ビットのコンパイル済みオブジェクトのサイズは、アップグレード後に約 55% 大きくなります。正確なサイズは、アップグレード前のプロセスで計算されます。この値に従って、アップグレードされるデータベースのサイズを大きくしてください。

---

コンパイル済みオブジェクトが、アップグレードの成功後に呼び出されることを確実にするために、dbcc upgrade_object コマンドを使用して手動でアップグレードできます。「運用前にコンパイル済みオブジェクトのエラーを探す」(294 ページ) を参照してください。

## 運用前にコンパイル済みオブジェクトのエラーを探す

12.5.x より前のバージョンの Adaptive Server で行った変更が原因で、新しいバージョンにアップグレードすると、コンパイル済みオブジェクトが異なる動作をする場合があります。dbcc upgrade_object を使用することにより、次のようなエラーと発生する可能性のある問題点を見つけることができます。正しく動作させるには、これらに手動で変更を加える必要があります。

- 予約語エラー
- 削除、トランケート、または破損したソース・テキスト
- 引用符で囲まれた識別子のエラー
- テンポラリ・テーブルの参照
- select * で発生する可能性がある問題

エラーと発生する可能性のある問題点を調べて、変更の必要がある部分を修正した後に、dbcc upgrade_object を使用することにより、サーバがオブジェクトを自動的にアップグレードするまで待たないでコンパイル済みオブジェクトを手動でアップグレードすることができます。『リファレンス・マニュアル：コマンド』を参照してください。

### 予約語エラー

dbcc upgrade_object がコンパイル済みオブジェクト内でオブジェクト名として使用される予約語を検出すると、エラーが表示されそのオブジェクトはアップグレードされません。エラーを修正するには、手動でオブジェクト名を変更するか、オブジェクト名を引用符で囲んで set quoted identifiers on コマンドを実行します。その後、コンパイル済みオブジェクトを削除して再作成します。

たとえば、データベース・ダンプを古いバージョンの Adaptive Server から現在のバージョンの Adaptive Server にロードするときに、現在のバージョンの予約語である “XYZ” という単語を使用しているストアド・プロシージャがダンプに含まれているとします。そのストアド・プロシージャに対して dbcc upgrade_object を実行すると、古いバージョンでは予約語ではなかった XYZが現在のバージョンでは予約語になっているため、エラーになります。このエラー表示によって、ストアド・プロシージャおよび関連するテーブルを事前に変更してから運用環境で使用することができます。

## 削除、トランケート、または破損したソース・テキスト

syscomments 内のソース・テキストが、削除、トランケート、または損傷している場合、dbcc upgrade_object が構文エラーを表示することがあります。ソース・テキストが隠されていない場合は、sp_helptext を使用してソース・テキストが完全なものかどうか調べることができます。トランケートまたはその他の破損が発生している場合は、コンパイル済みオブジェクトを削除して再作成します。

## 引用符で囲まれた識別子のエラー

dbcc upgrade_object は、次の場合に引用符で囲まれた識別子のエラーを返します。

- 11.9.2 より前のリリースで引用符で囲まれた識別子をアクティブにしてコンパイル済みオブジェクトが作成された (set quoted identifiers on)。
- 引用符で囲まれた識別子が、現在のセッションでアクティブではない (set quoted identifiers off)。

このエラーを避けるには、引用符で囲まれた識別子をアクティブにしてから dbcc upgrade_object を実行してください。引用符で囲まれた識別子がアクティブな場合は、二重引用符ではなく一重引用符で dbcc upgrade_object キーワードを囲む必要があります。

引用符で囲まれた識別子のエラーが発生する場合は、set コマンドを使用して quoted identifiers をアクティブにしてから、dbcc upgrade_object を実行してオブジェクトをアップグレードします。

---

**注意** 引用符で囲まれた識別子は、二重引用符で囲まれたリテラルと同じではありません。リテラルの場合は、アップグレードの前に特別なアクションを行う必要はありません。

---

## テンポラリ・テーブルの参照

ストアド・プロシージャやトリガなどのコンパイル済みオブジェクトがテンポラリ・テーブル (#temp *table_name*) を参照する場合、それがオブジェクト本体の外に作成されるとアップグレードは失敗して、dbcc upgrade_object はエラーを返します。このエラーを修正するには、コンパイル済みオブジェクトが必要とするのと同じテンポラリ・テーブルを作成してから、dbcc upgrade_object を再実行してください。コンパイル済みオブジェクトを自動的にアップグレードする場合は、この作業は必要はありません。

### *select* * で発生する可能性がある問題

dbcc upgrade_object は、ストアド・プロシージャの最も外側のクエリ・ブロックに select * 句を検出するとエラーを返し、オブジェクトをアップグレードしません。

たとえば、次のようなストアド・プロシージャがあります。

```
create procedure myproc as
  select * from employees
go
create procedure yourproc as
  if exists (select * from employees)
    print "Found one!"
go
```

myproc が最も外側のクエリ・ブロックに select * 句を持った文を含んでいるので、dbcc upgrade_object は myproc に関してエラーを返します。このプロシージャはアップグレードされません。

dbcc upgrade_object は、yourproc に関してはエラーを返しません。これは、select * 句がサブクエリ内にあるからです。このプロシージャはアップグレードされます。

**select * をビュー内で変更する必要があるかどうか調べる方法**

dbcc upgrade_object がビュー内に select * が存在するとレポートする場合は、元のビューの syscolumns の出力をテーブルの出力と比較して、ビューが作成された以降にテーブルに対してカラムが追加または削除されていないかどうか調べてください。

たとえば、次の文があるとします。

```
create view all_emps as select * from employees
```

all_emps ビューをアップグレードする前に、次のクエリを使用して、元のビューのカラム数と更新後のテーブルのカラム数を調べます。

```
select name from syscolumns
  where id = object_id("all_emps")
select name from syscolumns
  where id = object_id("employees")
```

2 つのクエリの出力を比較します。テーブルがビューより多数のカラムを含んでいて、select * 文のアップグレード前の結果を保持することが重要な場合は、select * 文を、特定のカラム名を指定する select 文に変更してください。ビューが複数のテーブルから作成された場合は、ビューを構成するすべてのテーブルのカラムを調べて、必要に応じて select 文を書き換えてください。

---

**警告！** select * 文をビューから実行しないでください。実行すると、ビューがアップグレードされて、syscolumns 内の元のカラム情報に関する情報が上書きされます。

---

ビュー内のカラムと新しいテーブル内のカラムとの相違を調べる別の方法は、sp_help をビューとビューを構成するテーブルの両方について実行することです。

この比較は、ビューに対してだけ実行でき、他のコンパイル済みオブジェクトに対しては実行できません。他のコンパイル済みオブジェクト内の select * 文の変更が必要かどうかを調べるには、各コンパイル済みオブジェクトのソース・テキストを調べてください。

## アップグレードにデータベース・ダンプを使用する

### ダンプおよびロードを使用してアップグレードする

12.5 より前のデータベース・ダンプとトランザクション・ログをロードして、データベースをアップグレードできます。

以下の点に注意してください。

- アップグレードには、データをコピーするディスク領域と、アップグレード・プロセス中にシステム・テーブルへの変更のログを取るディスク領域が必要です。ダンプ内のソース・データベースが満杯に近くなると、領域が不足してアップグレード・プロセスが失敗する可能性があります。この問題の発生頻度は低いはずですが、alter database を使用して領域不足エラーが発生しないように空き領域を増やすことができます。
- 古いダンプを再ロードしたら、新しいインストール環境からロードしたデータベース上で sp_checkreswords を実行し、予約語をチェックしてください。

### データベース・ダンプ内のコンパイル済みオブジェクトのアップグレード

現在の Adaptive Server より前のバージョンで作成されたデータベース・ダンプをロードする場合は、ダンプをロードする前にアップグレード前の作業を実行する必要はありません。したがって、データベース・ダンプ内のコンパイル済みオブジェクトのソース・テキストが削除されていても何も通知されません。データベース・ダンプのロードが終了したら、sp_checksource を実行してデータベース内のすべてのコンパイル済みオブジェクトについてソース・テキストが存在するか確認してください。確認ができれば、コンパイル済みオブジェクトが実行されるときにアップグレードすることができます。または、発生する可能性のある問題を見つけるために dbcc upgrade_object を実行して、オブジェクトを手動でアップグレードすることができます。

『リファレンス・マニュアル：プロシージャ』を参照してください。

## コンパイル済みオブジェクトがアップグレードされているか調べる方法

コンパイル済みオブジェクトがアップグレードされているか調べるには、次のいずれかを実行します。

- sysprocedures.version カラムを確認します。オブジェクトがアップグレードされた場合は、このカラムに数値 12500 が含まれます。
- ポインタのサイズを同じバージョンの 64 ビット・ポインタにアップグレードする場合は、sysprocedures.status カラムを調べます。オブジェクトがアップグレード済みで、64 ビット・ポインタを使用する場合、このカラムは、オブジェクトが 64 ビット・ポインタを使用することを示す 0x2 という 16 進数ビット設定を含んでいます。このビットが設定されていない場合、オブジェクトはまだ 32 ビット・オブジェクトであり、アップグレードされていないことを意味します。

第 12 章

# バックアップおよびリカバリ・プランの作成

停電やコンピュータの不具合に対処できるように、Adaptive Server には自動リカバリプロシージャが用意されています。メディア障害の発生に備えて、定期的なデータベースのバックアップを頻繁に実行してください。

この章では、バックアップとリカバリ・プランの作成に役立つ情報を説明します。この章の最初の部分は、Adaptive Server のバックアップとリカバリのプロセスの概要です。次に、システムの運用を実際に始める前に解決しなければならないバックアップとリカバリの問題について説明します。

サイトで IBM Tivoli Storage Manager が提供する記憶領域管理サービスを使用する場合は、「IBM Tivoli Storage Manager と Backup Server の使用」も参照してください。

# データベースの変更の追跡

Adaptive Server は、トランザクションを使用して、データベースのすべての変更を記録します。トランザクションとは、Adaptive Server の作業の単位です。トランザクションは、1 つまたは複数の Transact-SQL 文から構成されており、全体が 1 つの単位として正常終了するか、異常終了します。

データを変更するための SQL 文は、それぞれが 1 つの**トランザクション**と考えられます。ユーザは、begin transaction...end transaction ブロックの中に一連の文を組み込むことで、トランザクションを定義することができます。『Transact-SQL ユーザーズ・ガイド』の「第 18 章 トランザクション：データの一貫性およびリカバリ」を参照してください。

個々のデータベースには、そのデータベースの**トランザクション・ログ**であるシステム・テーブル syslogs があります。トランザクション・ログには、データベースの各ユーザが発行したすべてのトランザクションが自動的に記録されます。トランザクション・ログへの記録をオフにすることはできません。

トランザクション・ログは、**先書きログ**です。ユーザが発行した文がデータベースを変更するものである場合は、Adaptive Server はその変更をログに書き込みます。文による変更がすべてログに記録された後で、その変更がキャッシュ内のデータ・ページのコピーに書き込まれます。キャッシュ内のデータ・ページは、別のデータベース・ページによってそのメモリが要求されるまではキャッシュ内にとどまります。別のデータベース・ページによってメモリが要求されると、データ・ページはディスクに書き込まれます。

トランザクション内の文のいずれかが異常終了した場合は、Adaptive Server は、そのトランザクションによって行われたすべての変更を取り消します。Adaptive Server は、個々のトランザクションの終了時に「トランザクション終了」レコードをログに書き込み、トランザクションのステータス (成功または失敗) を記録します。

## トランザクション・ログに関する情報の取得

トランザクション・ログには、それぞれのトランザクションを確実にリカバリできる情報が含まれています。トランザクション・ログに含まれている情報をテープまたはディスクにコピーするには、dump transaction コマンドを使用します。ログのサイズをチェックするには sp_spaceused syslogs を使用し、ログの拡張が可能な空き領域をチェックするには sp_helpsegment logsegment を使用します。

---

**警告！** insert コマンド、update コマンド、delete コマンドを使用して syslogs を変更しないでください。

---

## ログ・レコードをコミットするときを決定するための *delayed_commit* の使用

リレーショナル・データベースは、原子性、一貫性、整合性、持続性 (ACID プロパティとも呼ばれます) など、さまざまなトランザクション・プロパティを保証する必要があります。そのため、Adaptive Server は以下のルールに従います。

- すべての操作をトランザクション・ログに書き込みます。
- ログ・レコードの書き込みは、データまたはインデックス・ページを変更する前に行います。
- トランザクションの commit が発行されたら、ログ・ページをディスクに書き込みます。
- commit が成功したことをクライアント・アプリケーションに通知するのは、Adaptive Server が、基本となるオペレーティング・システムおよび I/O サブシステムから、ディスクへの書き込みが成功したことの通知を受け取った後でのみ行います。

set delayed_commit は、特定のアプリケーションに対してのみ適したパフォーマンス・オプションです。Adaptive Server の データ操作言語 (DML) の操作 (insert、update、delete など) に対するパフォーマンスが向上しますが、システム障害の間にデータが失われるリスクも増加します。パフォーマンスの向上は、使用するアプリケーションによって異なります。

set delayed_commit によるメリットがあるのは、一般に、短時間で逐次的に Adaptive Server に送信される短いトランザクションを実行するアプリケーションです。たとえば、多数の insert 文を発行し、各 insert が別のトランザクションであるようなバッチ・アプリケーションなどです。

delayed_commit を有効にするには、セッションに対しては set コマンドを使用し、データベースに対しては sp_dboption を使用します。

set delayed_commit を有効にすると、それ以降、対応するログ・レコードがディスクに書き込まれる前に、commit の成功がアプリケーションに通知されます。これにより、最後を除くすべてのログ・ページがディスクに書き込まれて、最後のアクティブなログ・ページでの競合が緩和されるので、パフォーマンスが向上します。

set delayed_commit を有効にする前に、次の点を考慮してください。

- サーバをシャット・ダウンする前に checkpoint を発行してそれが完了していない場合、shudown with nowait を発行するとデータの持続性に関する問題が発生する可能性があります。
- セッションに対して set delayed_commit を有効にすると、そのセッションだけが影響を受けます。他のすべてのセッションのトランザクションでは、ACID プロパティを含めて、そのトランザクションのすべてのプロパティが適用されます。これは、他のセッションの物理ログ書き込みでは、set delayed_commit が有効になっているセッションに対応する最後のログ・ページとそのログ・レコードが書き込まれることも意味します。

- テンポラリ・データベースの場合、set delayed_commit の使用は冗長であり、パフォーマンスは向上しません。
- set delayed_commit は、アプリケーションと運用の両面における要件および環境を慎重に検討した後でのみ、使用してください。データの持続性に対するリスクが非常に低くても、データベースが大きく、データの欠落に対する許容度が低い場合には、リカバリのオプションを実行すると非常に時間がかかる可能性があります。

**ロギング動作の変化**

以下は、delayed_commit を有効にしたときのロギング動作の変化です。

セッションが暗黙的または明示的にトランザクションをコミットする場合は、次のように変化します。

- ユーザ・ログ・キャッシュ (ULC) は、メモリ上のトランザクション・ログにフラッシュされます。
- タスクは、commit を含む最後のログ・ページを除く、書き込まれていないすべてのログ・ページに対して書き込みを実行します。
- タスクは、IO の完了を待たずに、commit の成功をクライアント・アプリケーションに通知します。

---

**注意** トランザクションの「最後のログ・ページ」は、以下によって書き込まれます。

- 「最後のログ・ページ」ではなくなった時点で、別のトランザクションによって書き込まれます。
- トランザクションの完了後、別の非遅延トランザクションによって書き込まれます。
- checkpoint またはハウスキーピングのバッファ・ウォッシュ・メカニズムによって書き込まれます。
- 暗黙的なチェックポイントによって書き込まれます (shutdown、dump database、dump tran、sp_dboption truncate log on checkpoint など)。

---

- タスクは、次のトランザクションを続行できる状態になります。

**delayed_commit を使用したときのリスク**

set delayed_commit を有効にすると、Adaptive Server は、実際の物理ディスク書き込みが完了する前に、クライアント・アプリケーションに通知します。このため、アプリケーションは、物理ディスク書き込みが成功したかどうかに関係なく、トランザクションが完了したものと見なします。システム障害 ( ディスク・エラー、システム・クラッシュなど ) が発生した場合、ディスクに書き込まれていなかったトランザクション (commit レコードが最後のログ・ページにあったレコード) は、アプリケーションがコミット済みの通知を受け取っていたとしても、リカバリ後には存在しません。

Real Time Data Services (RTDS) を使用するメッセージング・システムなどによる緊密なシステム依存関係を必要とするシステムの場合、set delayed_commit を使用したときの結果はさらに複雑になります。

アプリケーションは、次の 2 つの状況でリスクを管理できます。

- アプリケーションが独自のトレースまたはログを保持しており、システム障害の後で、データベースの状態が独自のトレースまたはログに対応している場合。
- データベースを、アプリケーションを実行する前の状態にリストアできる場合。ただし、バッチ・ジョブ・タイプのアプリケーションを実行する前に、データベースの完全なバックアップを行っておく必要があります。障害が発生した場合は、データベースのバックアップをロードして、バッチ・ジョブを再度実行します。

*set delayed_commit* の有効化

データベースまたはセッションに対して、set delayed_commit を有効にできます。セッションに対する設定は、データベースに対する設定より優先されます。つまり、セッションでこのオプションを有効にすると、データベースの設定に関係なく、delayed_commit が有効になります。

## バックアップに対する責任の割り当て

システム部門には、通常、すべてのバックアップとリカバリの操作を担当するオペレータがいます。システム管理者、データベース所有者、またはオペレータだけが dump コマンドと load コマンドを実行できます。ただし、データベース所有者は自身のデータベースしかダンプできません。オペレータとシステム管理者は任意のデータベースをダンプまたはロードできます。

「バックアップとリカバリのための Backup Server の使用方法」(348 ページ) を参照してください。

# データベースとそのログの同期化：チェックポイント

チェックポイントとは、ダーティ・ページ、つまり、最後のチェックポイント以降にメモリ内で変更されたページ (ディスク上では変更されていない) をすべてデータベース・デバイスに書き込むことです。Adaptive Server の自動チェックポイント・メカニズムによって、完了したトランザクションで変更されたデータ・ページをメモリ内のキャッシュからデータベース・デバイスに書き込む処理が定期的に行われます。データベースとそのトランザクション・ログを同期化すると、システム障害後のデータベースのリカバリに要する時間が短縮されます。

## リカバリ間隔の設定

通常、自動リカバリ・プロセスの所要時間は、1 つのデータベースにつき数秒から数分です。所要時間は、データベースのサイズ、トランザクション・ログのサイズ、コミットまたはロールバックが必要なトランザクションの数とサイズによって異なります。

許容される最大リカバリ時間を指定するには、recovery interval in minutes パラメータを指定して sp_configure を実行します。Adaptive Server は、指定された時間内でデータベースをリカバリできるように、自動チェックポイントを実行する間隔を調整します。

デフォルト値 5 を使用すると、データベースあたりの許容されるリカバリ時間は最大 5 分となります。リカバリ間隔を 3 分に変更するには、次のコマンドを使用します。

```
sp_configure "recovery interval in minutes", 3
```

---

**注意** Adaptive Server の障害発生時にアクティブになっていたトランザクションのうち、実行時間が長く、ログの量が非常に少ないトランザクション (たとえば、create index) には、リカバリ間隔の効果はありません。このようなトランザクションの処理を元に戻すには、トランザクションを実行するのと同じぐらいの時間がかかります。あまり時間がかからないようにするには、データベースのテーブルにインデックスを作成したらすぐにそのデータベースをダンプしてください。

---

## 自動チェックポイント・プロシージャ

チェックポイント・タスクは、約 1 分に 1 回サーバ上の個々のデータベースをチェックし、最後のチェックポイント以降にトランザクション・ログに追加されたレコードの数を調べます。これらのトランザクションをリカバリするために必要な時間がデータベースのリカバリ間隔よりも長いと判断されると、Adaptive Server はチェックポイントを発行します。

変更されたページはキャッシュからデータベース・デバイス上に書き込まれ、チェックポイント・イベントがトランザクション・ログに記録されます。その後、チェックポイント・タスクは 1 分間「スリープ」します。

チェックポイント・タスクを表示するには、sp_who を実行します。通常、チェックポイント・タスクの “cmd” カラムには “CHECKPOINT SLEEP” と表示されます。

```
fid  spid  status      loginame  origname  hostname         blk_spid  dbname
        tempdbname    cmd               block_xloid   threadpool
---  ----  ----------  --------  --------  ----------------  --------  ------
        ----------    ----------------  -----------   -------------------

. . .
0       2    sleeping      NULL      NULL                NULL       0  master
```

```
            tempdb     DEADLOCK TUNE              0    syb_default_pool
0       3    sleeping      NULL       NULL              NULL       0  master
            tempdb     ASTC HANDLER               0    syb_default_pool
0       4    sleeping      NULL       NULL              NULL       0  master
            tempdb     CHECKPOINT SLEEP           0    syb_default_pool
. . .
```

## ユーザ・データベースのアップグレード後のチェックポイント

Adaptive Server は、ユーザ・データベースをアップグレードした直後にチェックポイント・レコードを挿入します。このレコードは、アップグレードされたデータベースに対して dump transaction を実行する前に、必ず dump database が実行されるようにするためのものです。

## 自動チェックポイント後のログのトランケート

システム管理者は、Adaptive Server によって自動チェックポイントが実行されるときにトランザクション・ログをトランケートするように設定できます。

trunc log on chkpt データベース・オプションを設定すると、自動チェックポイントが発生したときにトランザクション・ログがトランケートされます。このオプションを設定するには、master データベースから次のコマンドを実行します。

```
sp_dboption database_name, "trunc log on chkpt", true
```

このオプションは、トランケートする前にトランザクション・ログのコピーを作成しないため、運用環境には適しません。trunc log on chkpt は、次のような場合に使用します。

- トランザクション・ログを保管するセグメントが分かれていないためにトランザクション・ログをバックアップできないデータベース
- 最新のバックアップが重要ではないテスト・データベース

**注意**　trunc log on chkpt オプションを off のままにしておく (デフォルトの状態) と、トランザクション・ログは、dump transaction コマンドを使用してトランケートするまで増え続けます。

ログが領域を使い切るのを防ぐには、トランザクション・ログをダンプするようにラストチャンス・スレッショルド・プロシージャを設計してください。「第 17 章 スレッショルドによる空き領域の管理」を参照してください。

### フリー・チェックポイント

Adaptive Server が処理すべきユーザ・タスクがなくなると、ハウスキーピング・ウォッシュ・タスクは、変更があったバッファをディスクに書き込む処理を自動的に開始します。ハウスキーピング・タスクが、すべての設定済みキャッシュ内のすべてのアクティブ・バッファ・プールをフラッシュできる場合は、チェックポイント・タスクをウェイクアップさせます。チェックポイント・タスクは、データベースでチェックポイントを実行する必要があるかどうか調べます。

ハウスキーピング・ウォッシュ・タスクの結果として発生するチェックポイントを、フリー・チェックポイントと呼びます。変更されたページをデータベース・デバイスに書き込む作業はハウスキーピング・ウォッシュ・タスクによって既に行われているため、フリー・チェックポイントではこの作業は行われません。そのため、データベースのリカバリ時間を短縮できる可能性があります。

『パフォーマンス&チューニング・シリーズ：基本』の「第 3 章 エンジンと CPU の使用方法」を参照してください。

### 手動によるチェックポイントの要求

データベース所有者は、checkpoint コマンドを発行して、メモリ内で変更されたすべてのページを強制的にディスクに書き込むことができます。sp_dboption の trunc log on chkpt オプションが “on” に設定されていても、手動チェックポイントのときはログはトランケートされません。

checkpoint コマンドは次のような場合に使用します。

- 特殊な状況 (計画的な shutdown with nowait の直前など) における予防策として、Adaptive Server のリカバリ・メカニズムがリカバリ間隔の時間内に実行されるようにします (通常の shutdown は、チェックポイントを実行します)。
- sp_dboption の実行後にデータベース・オプションの変更を有効にするために使用します(sp_dboption を実行すると、checkpoint の実行を指示する情報メッセージが表示されます)。

checkpoint を使用して、1 つまたは複数のデータベースを識別したり、all 句を使用して、すべてのデータベースのチェックポイントを実行したりできます。『リファレンス・マニュアル：コマンド』を参照してください。

## システム障害または停止後の自動リカバリ

停電、オペレーティング・システム障害、shutdown コマンドの実行などの後に Adaptive Server を再起動するたびに、各データベースに対して一連のリカバリ手順が自動的に実行されます。

リカバリ・メカニズムは、個々のデータベースをそのトランザクション・ログと比較します。特定の変更のログ・レコードがデータ・ページより新しい場合は、その変更をトランザクション・ログから再適用します。障害の発生時にトランザクションが進行中であった場合は、リカバリ・メカニズムは、そのトランザクションによるすべての変更を取り消します。

Adaptive Server を起動すると、次の順序でデータベースのリカバリが実行されます。

1　master がリカバリされます。

2　sybsystemprocs がリカバリされます。

3　model がリカバリされます。

4　(model をコピーして) tempdb が作成されます。

5　sybsystemdb がリカバリされます。

6　sybsecurity がリカバリされます。

7　sysdatabases.dbid による順序での、または sp_dbrecovery_order により指定された順序での、ユーザ・データベースのリカバリ。「リカバリ順序」を参照してください。

システム・データベースがリカバリされると、ユーザはすぐに Adaptive Server にログインできますが、他のデータベースにはそのデータベースのリカバリが終了するまでアクセスできません。

print recovery information 設定パラメータは、リカバリ中に個々のトランザクションに関する詳しいメッセージをコンソール画面に表示するかどうかを決定します。デフォルトでは、このメッセージは表示されません。メッセージを表示するには、次のコマンドを使用します。

```
sp_configure "print recovery information", 1
```

## 高速リカバリ

予定された停止または予定外の停止後にサーバが再起動するとき、または高可用性フェールオーバのときにかかる時間のほとんどは、データベース・リカバリを行うための時間です。リカバリが高速であれば、それだけデータベースのダウン時間を短くすることができます。高速リカバリの目標は次のとおりです。

- データベース・リカバリのパフォーマンスを向上する。
- 使用できるサーバ・リソースを活用し、自動的に調整することにより、複数のデータベースを並行してリカバリする。
- 複数のチェックポイント・タスクを同時に実行することにより、リカバリ時の作業を最小限に抑える。

## Adaptive Server の起動シーケンス

Adaptive Server 起動時のイベント順序は、次のとおりです。

1 システム・データベースのリカバリがエンジン 0 上で実行されます。

2 Adaptive Server はユーザ接続を受け付けます。

3 起動時にオンラインになるように設定されたエンジンがすべてオンラインになります。

4 リカバリ・タスクによって、複数のユーザ・データベースのリカバリが並行して行われます。リカバリ・タスクの数は、リカバリのパフォーマンスが最高になるようにチューニングされたデフォルトのデータ・キャッシュを使用するように「セルフチューニング」されます。

「データベース・リカバリ」(309 ページ ) を参照してください。リカバリ・タスクの数の詳細については、「並列リカバリ」(308 ページ) を参照してください。

HA フェールオーバ時は、フェールオーバされたユーザ・データベースのリカバリおよびオンライン化は並行して行われます。

## エンジンを先にオンラインにする

エンジンをオンラインにする処理は、システム・データベースのリカバリ後、ユーザ・データベースをリカバリする前に行われます。これによって、複数のユーザ・データベースを並行してリカバリすることが可能になり、また、エンジンをオンライン・アクティビティに使用できるようになります。

エンジンがこのタイミングでオンラインになるのは起動時だけです。それ以外のあらゆる状況、たとえばフェールオーバのときは、エンジンはセカンダリ・サーバ上で既にオンラインになっています。

## 並列リカバリ

Adaptive Server 12.5.1 以降では、起動時とフェールオーバ時に、複数のリカバリ・タスクによって、複数のデータベースのリカバリが並行して行われます。データベースのリカバリは、I/O 集約型のプロセスです。並列リカバリによる Adaptive Server のリカバリ時間は、I/O サブシステムの帯域幅に左右されます。この I/O サブシステムには、Adaptive Server の同時実行 I/O 要求を処理できるだけの能力が必要です。

並列リカバリでは、複数のタスクが並行して複数のユーザ・データベースを同時にリカバリします。リカバリ・タスクの数は、設定パラメータ max concurrently recovered db によって決まります。デフォルト値 0 は、データ保存のアーキテクチャに関する一切の仮定を行わないセルフチューニング・アプローチをサーバが使用することを示します。統計的 I/O サンプリング法によって、I/O サブシステムの性能に応じた最適なリカバリ・タスク数が計算されます。最適なリカバリ・タスク数の推奨値が提示されます。設定値が 0 以外の場合は、この設定パラメータか number of open databases パラメータで指定された数のタスクが生成されます。

並列リカバリの実行時に、システム管理者は、max concurrently recovered db パラメータの設定値を 1 に変更することによって強制的に逐次リカバリを実行できます。アクティブなリカバリ・タスクは、処理中のデータベースのリカバリが完了すると消滅します。残りのデータベースのリカバリは逐次方式で行われます。

『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## データベース・リカバリ

Adaptive Server のデータベース・リカバリには、次のものが含まれます。

- ログ I/O サイズ － Adaptive Server は、デフォルト・データ・キャッシュ内で使用可能な最大のバッファ・プールをログの I/O に使用します。最大バッファ・サイズのプールを使用できない場合は、このプールを動的に作成してログの I/O に使用します。このプールのバッファは、デフォルトのプールから取得されます。リカバリのパフォーマンスが最大になるように、大容量バッファ・プールのサイズがチューニングされます。大容量バッファ・プールを使用できても、サイズが最適ではない場合は、リカバリのパフォーマンスが最大になるように大容量プールとデフォルト・プールのサイズを動的に変更します。バッファ・プールの設定は、リカバリ処理の終了時に復元されます。

  『パフォーマンス&チューニング・シリーズ：基本』の「第 5 章 メモリの使い方とパフォーマンス」を参照してください。

- async prefetch limit － リカバリの実行中は、デフォルト・データ・キャッシュ内のプールに対するローカルの**非同期プリフェッチ**率をサーバが自動的に最適な値に設定します。リカバリ実行中は、ユーザが指定した値よりもこの設定が優先されます。

  リカバリが完了すると、元の設定値が復元されます。

## リカバリ順序

ユーザ・データベースのすべてまたは一部に対して、ユーザがデータベースのリカバリ順序を指定できます。重要度の高いデータベースを先にリカバリするように設定するには、sp_dbrecovery_order を使用します。

sp_dbrecovery_order を使用してユーザ定義のリカバリ順序を入力または修正するには、システム管理者の権限が必要です。また、master データベース内から実行する必要があります。sp_dbrecovery_order を使用してユーザ定義のリカバリ順序を表示することは、どのユーザでも、どのデータベースからでも実行できます。『リファレンス・マニュアル：プロシージャ』を参照してください。

sp_dbrecovery_order には、オンラインにする順序を示す追加のパラメータがあります。

```
sp_dbrecovery_order [database_name [, rec_order [, force [
relax | strict ]]]]
```

- relax − データベースは、リカバリが完了するとオンラインになる (デフォルト)。
- strict − データベースは指定されたリカバリ順序でオンラインになる。

デフォルトは relax で、データベースはリカバリ完了直後にオンラインになります。

リカバリ順序には、1 から始まる連続した整数を指定してください。先に 1、2、4 をデータベースに割り当て、後から別のデータベースに 3 を指定することはできません。

既に定義されている順序の途中にデータベースを挿入するには、*rec_order* を入力して force を指定します。たとえば、データベース A、B、C にリカバリ順序 1、2、3 がそれぞれ割り当てられている場合に、pubs2 というデータベースをリカバリ順序が 2 番目になるように追加するには、次のように入力します。

```
sp_dbrecovery_order pubs2, 2, force
```

このコマンドを実行すると、データベース B のリカバリ順序は 3、データベース C のリカバリ順序は 4 となります。

Adaptive Server 12.5.1 以降は並列リカバリ・タスクを使用し、ユーザが指定した順序に従って、次にどのデータベースをリカバリするかを決定します。残りのデータベースのリカバリは、データベース ID の順に行われます。1 つのデータベースのリカバリに必要な時間は、リカバリ可能ログのサイズなどのさまざまな要因によって左右されます。そのため、sp_dbrecovery_order でリカバリ順序を決めておいても、データベース・リカバリが開始された順序とは異なる順序でリカバリが完了する場合があります。リカバリ順序と同じ順序でデータベースをオンラインにする必要のあるアプリケーションに対応できるように、sp_dbrecovery_order には strict オプションが用意されています。

## データベースに割り当てられたリカバリ順序の変更および削除

データベースのリカバリ順序を変更するには、対象のデータベースをユーザ定義のリカバリ・シーケンスからいったん削除して、新しい位置に挿入します。リカバリ順序を最後にする場合以外は、force オプションを使用します。

データベースをリカバリ・シーケンスから削除するには、順序に -1 を指定します。

たとえば、データベース pubs2 のリカバリ順序を 2 から 1 に変更するには、次に示すように、まずこのデータベースをリカバリ・シーケンスから削除し、次に新しい順序を割り当てます。

```
sp_dbrecovery_order pubs2, -1
sp_dbrecovery_order pubs2, 1, "force"
```

## ユーザ定義のデータベース・リカバリ順序のリスト

リカバリ順序が割り当てられているすべてのデータベースについて、その順序をリストするには、次のコマンドを使用します。

```
sp_dbrecovery_order
```

次のように出力されます。

```
The following databases have user specified recovery order:
Recovery Order Database Name        Database Id
-------------- -------------------- -----------
    1          dbccdb                   8
    2          pubs2                    5
    3          pubs3                    6
    4          pubtune                  7
The rest of the databases will be recovered in default database id order.
```

特定のデータベースのリカバリ順序を表示するには、そのデータベース名を入力します。

```
1> sp_dbrecovery_order pubs2
2> go

Database Name Database id Recovery Order
 ------------- ----------- --------------
pubs2              5           2
```

## 並列チェックポイント

複数のチェックポイント・タスクから成るプールが、アクティブなデータベースのリストを並列処理します。このプールは、設定パラメータ number of checkpoint tasks によって制御されます。チェックポイントのボトルネックが発生している場合に、チェックポイント・タスクを増やすことは、リカバリ可能ログが短くなるということを意味するので、障害が発生してもリカバリの処理量が少なくて済み、可用性が向上します。

number of checkpoint tasks のデフォルト値は 1 で、逐次チェックポイントを表します。このパラメータの値は、**エンジン数と、オープンしているデータベース数**によって制限されます。並列リカバリをスムーズに行うには、できるだけ多くのエンジンを起動時にオンラインにします。このパラメータの値を小さくするとチェックポイント・タスク数が減少し、値を大きくすると追加のタスクが生成されます。

チェックポイントは I/O 集約型のプロセスです。そのため、並列チェックポイントの効果は、データベースのレイアウトと、I/O サブシステムのパフォーマンスに依存します。アクティブなデータベースの数と、I/O サブシステムの書き込み処理能力に基づいて、number of checkpoint tasks をチューニングします。

『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## リカバリ状態

グローバル変数 *@@ recovery_state* を利用すると、Adaptive Server がリカバリ中かどうかを判別できます。*@@ recovery_state* の有効な値は次のとおりです。

- NOT_IN_RECOVERY － Adaptive Server は起動時のリカバリ中でもフェールオーバ時のリカバリ中でもありません。リカバリは完了しており、オンラインにできるすべてのデータベースがオンラインになっています。
- RECOVERY_TUNING － Adaptive Server はリカバリ中 (起動時またはフェールオーバ時) で、最適なリカバリ・タスク数をチューニングしています。

- BOOTIME_RECOVERY − Adaptive Server は起動時のリカバリを行っており、最適なタスク数のチューニングを完了しますた。すべてのデータベースがリカバリされたわけではありません。
- FAILOVER_RECOVERY − Adaptive Server は HA フェールオーバ時のリカバリを行っており、最適なリカバリ・タスク数のチューニングを完了しました。すべてのデータベースがオンラインになっているわけではありません。

アプリケーションで *@@recovery_state* を使用すると、すべてのデータベースがリカバリされてオンラインになったかどうかを判断できます。

## 高速リカバリを実現するためのチューニング

この項では、Adaptive Server をチューニングしてリカバリ時間を短縮するためのガイドラインについて説明します。

### データベースのレイアウト

- データベースのログとデータを、それぞれ専用の物理デバイスに保存してください。ログとデータへのアクセス・パターンは異なるので、分けて保存します。
- Adaptive Server 内の複数のデータベースから同時に発生する I/O 要求を処理できるように、基本となる I/O サブシステムを設定します。

### 実行時設定についてのアドバイス

- 空きサイクルの間にダーティ・ページが書き出されるように、housekeeper free write percent を使用して最適なハウスキーピング・ウォッシュ・タスクのパーセンテージを設定します。通常はデフォルト値が最適値です。
- 長時間実行されるトランザクションは、最低限に抑えるようにします。長時間実行されるトランザクションはリソースを占有し、リカバリの時間を長引かせる原因にもなります。
- サーバの停止には polite shutdown を使用して、リカバリの時間が長くならないようにします。

### 領域計算の設定

データ領域の計算が重要でないデータベースの場合は、sp_dboption を使用して、空き領域計算をオフにするようにデータベース・オプションを設定します。このようにすると、データ・セグメントに対してスレッショルド・アクションが実行されないことになります。

# リカバリ中のフォールト・アイソレーション

単に「リカバリ」と呼ばれているリカバリ・プロシージャは、トランザクション・ログからサーバのデータベースを再構築する処理です。次のような場合に、リカバリが実行されます。

- Adaptive Server の起動
- load database コマンドの使用
- load transaction コマンドの使用

リカバリ・アイソレーション・モード設定は、データベース内でトランザクションの処理を元に戻したり再適用したりするときに破壊されたデータが検出された場合のリカバリの動作を制御します。

インデックスが suspect (疑わしい) とマーク付けされている場合は、インデックスを削除してから再作成することによって、このインデックスを修復できます。

リカバリ・フォールト・アイソレーションにより次のことが可能になります。

- リカバリによって破壊が検出されたときに、データベース全体をアクセスできないようにするか、疑わしいページだけをアクセスできないようにするかを設定します。
- 疑わしいページのあるデータベース全体を read_only モードでオンラインにするか、オンラインのページを更新可能にするかを設定します。
- 疑わしいページが含まれているデータベースをリストします。
- 指定したデータベース内の疑わしいページを、ページ ID、インデックス ID、オブジェクト名ごとに表示します。
- 疑わしいページを修復する間、システム管理者が使用できるようにそのページをオンラインにします。
- 疑わしいページを修復後、そのページをすべてのデータベース・ユーザに対してオンラインにします。

疑わしいページだけを分離 ( アイソレート ) し、残りのデータベースをオンラインにできるため、データ破壊時の処理の柔軟度が高まります。システム管理者が問題を診断し、問題を修正している間も、データベースのほとんどの部分はユーザからのアクセスが可能な状態が保たれます。損傷の範囲を調べて、応急修復を行うか、妥当な時期に再ロードを行います。

リカバリ・フォールト・アイソレーションが適用できるのは、ユーザ・データベースだけです。リカバリ時に、システム・データベース内で破壊されたページが見つかった場合は、必ずシステム・データベース全体がオフラインになります。破壊されたページをすべて修復するか削除してからでないと、システム・データベースをリカバリすることはできません。

## オフライン・ページの持続性

疑わしいページをオフラインにした場合は、サーバを再起動してもそのページはオフラインのままです。オフライン・ページについての情報は、master.dbo.sysattributes に格納されます。

疑わしいページのエントリを master.dbo.sysattributes からクリアするには、drop database コマンドと load database コマンドを使用してください。

## リカバリ・フォールト・アイソレーションの設定

Adaptive Server をインストールするときに設定されるリカバリ・アイソレーション・モードのデフォルトは databases です。このモードでは、破壊されたページが検出されると、データベースは「疑わしい」とマーク付けされ、そのデータベース全体がオフラインになります。

### 疑わしいページのアイソレート

疑わしいページだけをオフラインにするために分離し、データベースの残りの部分にはユーザがアクセスできるようにするには、sp_setsuspect_granularity を使用してリカバリ・アイソレーション・モードを page に設定します。この設定は、そのデータベースに対して次回リカバリが実行されるときに有効になります。『リファレンス・マニュアル：プロシージャ』を参照してください。

database と page のどちらの引数も指定しないで sp_setsuspect_granularity を実行すると、指定したデータベースに現在設定されているリカバリ・アイソレーション・モードの値が表示されます。引数をすべて省略すると、現在のデータベースの設定値が表示されます。

破壊が発生したページを特定して分離できない場合は、リカバリ・アイソレーション・モードが page に設定されていても、データベース全体が「疑わしい」とマーク付けされます。たとえば、トランザクション・ログが破壊されている場合や、使用できないグローバル・リソースがある場合に、この状態になります。

リカバリによって特定のページが疑わしいとマーク付けされたとき、デフォルトでは、データベースに対しては読み込みと書き込みが可能になりますが、疑わしいページはオフラインになり、したがってアクセス不可能になります。ただし、sp_setsuspect_granularity に read_only オプションを指定すると、リカバリによりいずれかのページに疑わしいとマーク付けされた場合、データベース全体が read_only モードでオンラインになり、変更はできなくなります。通常は read_only オプションを選択した状態にするけれども、場合によっては疑わしいページ以外のページをユーザが変更できるようにするには、sp_dboption を使用してデータベースのオンライン部分を書き込み可能にします。

```
sp_dboption pubs2, "read only", false
```

この場合、疑わしいページは、修復するか強制的にオンラインにするまではオフラインのままになります。詳細については、「オフライン・ページのオンライン化」(317 ページ) を参照してください。

### 疑わしいページ数の許容値の増加

サスペクト・エスカレーション・スレッショルドは、リカバリ・アイソレーション・モードが page に設定されている場合でもデータベース全体を「疑わしい」とマーク付けするかどうかを決定する基準となる、疑わしいページの数です。デフォルトでは、データベースあたり 20 ページに設定されています。sp_setsuspect_threshold を使用して、サスペクト・エスカレーション・スレッショルドの値を変更します。『リファレンス・マニュアル：プロシージャ』を参照してください。

リカバリ・フォールト・アイソレーションとサスペクト・エスカレーション・スレッショルドは、データベース・レベルで設定します。

この例は、pubs2 データベースのリカバリ・アイソレーション・モードが page であり、このデータベースで最後にリカバリが実行されたときのエスカレーション・スレッショルド (現在のサスペクト・スレッショルド値) が 20 であったことを示しています。このデータベースで次回リカバリが実行されるときのリカバリ・アイソレーション・モードは page であり、エスカレーション・スレッショルドは 30 (設定値) になります。

```
sp_setsuspect_granularity pubs2

DB Name  Cur. Suspect Gran.  Cfg. Suspect Gran.  Online mode
-------  ------------------  ------------------  -----------
pubs2    page                page                read/write


sp_setsuspect_threshold pubs2

DB Name       Cur. Suspect threshold   Cfg. Suspect threshold
------------- ------------------------ ----------------------
pubs2         20                       30
```

引数を指定せずに sp_setsuspect_granularity と sp_setsuspect_threshold を実行すると、現在のデータベースがユーザ・データベースならば、現在のデータベースの現在の設定値が表示されます。

## オフライン・データベースとオフライン・ページ情報の取得

どのデータベースにオフライン・ページがあるかを表示するには、sp_listsuspect_db を使用します。

次の例は、疑わしいページについての一般情報を表示します。

```
sp_listsuspect_db
The database 'dbt1' has 3 suspect pages belonging to 2 objects.
```

個々のオフライン・ページの詳細情報を表示するには、sp_listsuspect_page を使用します。

dbname を省略した場合のデフォルトは、現在のデータベースです。次の例は、sp_listsuspect_page を使用して出力した、dbt1 データベースについてのページレベルの詳細情報を示します。

```
sp_listsuspect_page dbt1

DBName   Pageid       Object    Index   Access
-------- ------------ --------- ------- --------------------
dbt1     384          tab1      0       BLOCK_ALL
dbt1     390          tab1      0       BLOCK_ALL
dbt1     416          tab1      1       SA_ONLY

(3 rows affected, return status = 0)
```

Access カラムの値が SA_ONLY で、疑わしいページが 1 の場合、sa_role を付与されているユーザのみがその疑わしいページにアクセスできます。値が BLOCK_ALL の場合は、そのページにはどのユーザもアクセスできません。

sp_listsuspect_db と sp_listsuspect_page は、すべてのユーザが、どのデータベースからでも実行できます。

## オフライン・ページのオンライン化

データベース内のすべてのオフライン・ページをアクセス可能にするには、sp_forceonline_db を使用します。個々のオフライン・ページをアクセス可能にするには、sp_forceonline_page を使用します。『リファレンス・マニュアル：プロシージャ』を参照してください。

どちらのプロシージャを使用する場合も、アクセスのタイプを指定します。

- sa_on は、疑わしいページやデータベースにアクセスできるユーザを、sa_role を付与されたユーザだけに限定します。これは、データベースが起動されて稼働しているときに疑わしいページの修復と修復結果のテストを行い、その間は一般のユーザが疑わしいページにアクセスできないようにする場合に便利です。また、ページがオフラインの場合には禁止されている、疑わしいページのあるデータベース上での dump database や dump transaction with no_log の実行にも使用できます。
- “sa_off” は、システム管理者を含むすべてのユーザのアクセスをブロックします。これは、前回 “sa_on” を指定して実行された sp_forceonline_db または sp_forceonline_page の処理を元に戻します。
- “all_users” は、ページが修復された後に、すべてのユーザに対してオフライン・ページをオンラインにします。

"sa_on" を指定して疑わしいページをオンラインにしてから "sa_off" を指定して再度オフラインにする場合とは異なり、sp_forceonline_page または sp_forceonline_db で "all users" を指定してページをオンラインにした場合は、この処理を元に戻すことはできません。つまり、オンライン・ページを再度オフラインにすることはできません。

**警告！** オンラインにされるページに対して Adaptive Server が何らかの検査を行うことはありません。オンラインにするページが修復済みかどうかを確認してください。

sp_forceonline_db と sp_forceonline_page は、トランザクション内では実行できません。

sp_forceonline_db や sp_forceonline_page を実行するには、sa_role を付与されている必要があります。また、これらの操作は、master データベース内から行う必要があります。

## データオンリーロック・テーブルに対するインデックス・レベルのフォールト・アイソレーション

リカバリ中に、データオンリーロック・テーブルのインデックスのページが疑わしいとマーク付けされると、インデックス全体がオフラインになります。オフライン・インデックスを管理するために、次の 2 つのシステム・プロシージャがあります。

- sp_listsuspect_object
- sp_forceonline_object

多くの場合、システム管理者は、sa_role を持つユーザだけが疑わしいインデックスにアクセスできるように sp_forceonline_object を使用して設定します。そのインデックスがユーザ・テーブルのインデックスの場合は、いったん削除して再作成することによって疑わしいインデックスを修復できます。

『リファレンス・マニュアル：プロシージャ』を参照してください。

## オフライン・ページの二次的な影響

オフライン・ページがあるデータベースに対して、次の制限が適用されます。

- オフラインのデータを直接的または間接的に必要とする (たとえば参照整合性制約のため) トランザクションは異常終了し、メッセージを生成します。

- データベースの中に、オフラインになっている部分がある場合は、dump database は実行できません。

  システム管理者は、sa_on を指定して sp_forceonline_db を実行することによりオフライン・ページを強制的にオンラインにしてから、データベースをダンプします。ダンプが完了したら、“sa_off” を指定して sp_forceonline_db を実行します。

- データベースの中にオフラインになっている部分がある場合は、dump transaction with no_log や dump transaction with truncate_only を実行することはできません。

  システム管理者は、“sa_on” を指定して sp_forceonline_db を実行することによりオフライン・ページを強制的にオンラインにしてから、with no_log オプションを使用してトランザクション・ログをダンプします。ダンプが完了したら、“sa_off” を指定して sp_forceonline_db を実行します。

- オフライン・ページが含まれているテーブルやインデックスを削除するには、master データベース内でトランザクションを使用する必要があります。このようにしないと、疑わしいページのエントリを master.dbo.sysattributes から削除する必要があるため、削除は失敗します。次の例では、オブジェクトを削除し、そのオフライン・ページについての情報を master.dbo.sysattributes から削除しています。

  pubs2 データベースから、疑わしいページが含まれている authors_au_id_ind という名前のインデックスを削除するには、次のように master データベースのトランザクション内でこのインデックスを削除します。

```
use master
go
sp_dboption pubs2, "ddl in tran", true
go
checkpoint pubs2
go
begin transaction
drop index authors.au_id_ind
commit
go
use master
go
sp_dboption pubs2, "ddl in tran", false
go
checkpoint pubs2
go
```

## リカバリ・フォールト・アイソレーションを使用したリカバリ方式

疑わしいページのあるデータベースを、ユーザがアクセスしているときに一貫性のある状態に戻すには、再ロードと修復の 2 つの主な方式があります。

どちらの方式も、次のものが必要です。

- クリーン・データベース・ダンプ
- 疑わしいページがある状態でデータベースがリカバリされた時点までの、信頼性のある一連のトランザクション・ログ・ダンプ
- オフライン・ページに対する変更を取得するために、データベースのリカバリ直後に作成されたトランザクション・ログ・ダンプ
- 部分的にオフラインになっているデータベースをユーザが使用している間に作成された、連続的なトランザクション・ログ・ダンプ

### 再ロード方式

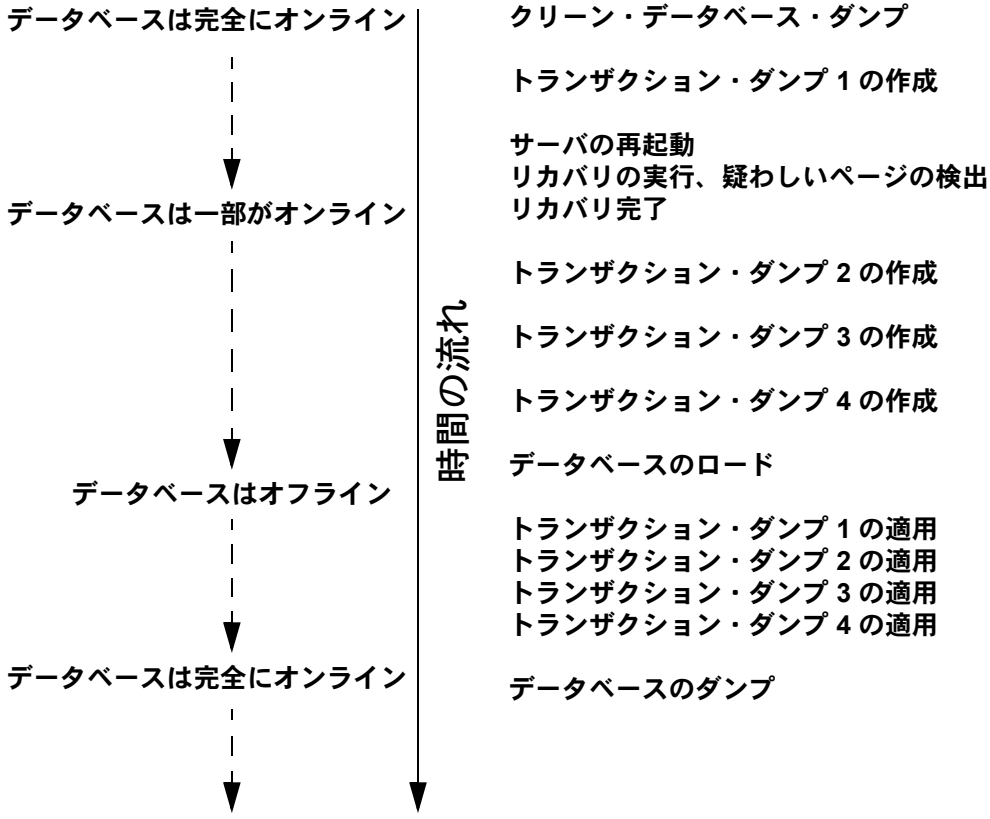
再ロード方式では、クリーン・データベースをバックアップからリストアします。適切なときに、最新のクリーン・データベース・ダンプをロードし、トランザクション・ログを適用してデータベースをリストアします。

load database を実行すると、master.dbo.sysdatabases システム・テーブルと master.dbo.sysattributes システム・テーブルから疑わしいページの情報がクリアされます。

リストアしたデータベースがオンラインになった直後に、そのデータベースをダンプします。

図 12-1 は、データベースを再ロードする方式を示します。

図 12-1: 再ロード方式



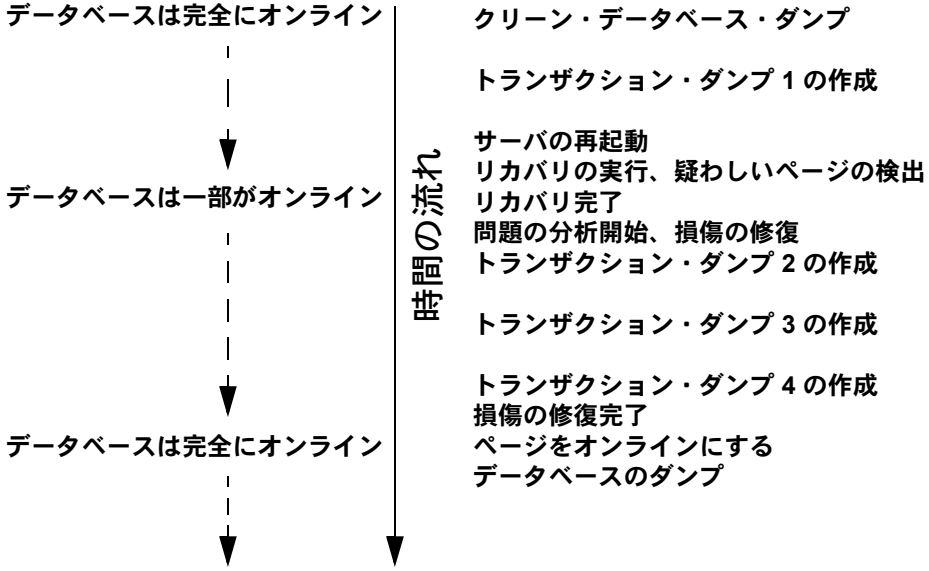

## 修復方式

修復方式では、データベースが部分的にオフラインになっているときに、破壊されたページを修復します。問題を診断して修復するには、dbcc コマンドなどの既知の方法を使用して、システム管理者が専用に使用するために強制的にオンラインにされた疑わしいページに対して、結果がわかっているクエリを実行します。必要であれば、Sybase 製品の保守契約を結んでいるサポート・センタに問い合わせてください。損傷を修復するために、疑わしいページが含まれるオブジェクトを削除して再作成することもあります。

sp_forceonline_page を使用して、修復されたオフライン・ページを個別にオンラインにすることも、すべてのオフライン・ページが修復されるまで待ち、sp_forceonline_db を使用して一度にオンラインにすることもできます。

修復方式では、データベース全体をオフラインにする必要はありません。図 12-2 は、破壊されたページを修復する方式を示します。

**図 12-2: 修復方式**




## 破壊の範囲の調査

場合によっては、破壊された範囲を調べるためにリカバリ・フォールト・アイソレーションを使用します。このとき、強制的にリカバリを実行して、疑わしいとマーク付けされたページと、そのページが属するオブジェクトを検査します。

たとえば、特定のデータベースを使用しているユーザから問題が報告された場合は、リカバリ・アイソレーション・モードを “page” に設定し、Adaptive Server を再起動して強制的にリカバリを行います。リカバリが完了したら、sp_listsuspect_db または sp_listsuspect_page を実行して、疑わしいページのページ数と、影響のあるデータベース・オブジェクトを判断します。

データベース全体が疑わしいとマーク付けされていて、次のようなメッセージが表示された場合は、

```
Reached suspect threshold '%d' for database '%.*s'.
Increase suspect threshold using sp_setsuspect_threshold.
```

sp_setsuspect_threshold を使用して、サスペクト・エスカレーション・スレッショルドの値を増やし、もう一度リカバリを強制的に実行してください。このメッセージを受け取るたびに、スレッショルドの値を増やしてリカバリを実行し、データベースがオンラインになるまで続けます。このメッセージを受け取らなかった場合は、破壊が発生しているページを分離できず、この方式によって疑わしいページ数を調べることはできません。

## *dump* コマンドおよび *load* コマンドの使用方法

ディスク・クラッシュなどのメディア障害が発生した場合も、データベースとそのトランザクション・ログを定期的にバックアップしていれば、データベースをリストアできます。完全なリカバリができるかどうかは、定期的に dump database コマンドと dump transaction コマンドを使用してデータベースをバックアップし、load database コマンドと load transaction コマンドを使用してデータベースをリストアしているかどうかによって決まります。これらのコマンドについて、以降で簡単に説明します。詳細については、「第 13 章 ユーザ・データベースのバックアップとリストア」と「第 14 章 システム・データベースのリストア」を参照してください。

---

**警告！** オペレーティング・システムのコピー・コマンドを使用して、動作しているデータベース・デバイスをコピーしないでください。コピーされたデバイスに対して Adaptive Server を実行すると、データベースの破壊の原因となります。Adaptive Server をシャットダウンするか、quiesce database コマンドを使用してコピー操作を保護します。

---

ダンプ・コマンドは、データベースが破壊されていても正常に終了します。データベースをバックアップする前に、dbcc コマンドを使用して一貫性を検査してください。「第 11 章 データベースの一貫性の検査」を参照してください。

---

**警告！** テープに直接ダンプするときは、そのテープに他のタイプのファイル (UNIX バックアップ、tar ファイルなど) を保存しないでください。そのようなファイルを保存すると、Sybase のダンプ・ファイルが無効になる可能性があります。ただし、UNIX のファイル・システムにダンプすると、ダンプを含むファイルをテープに保管することができます。

---

## 日常のデータベース・ダンプの作成：*dump database*

dump database コマンドは、データとトランザクション・ログの両方を含むデータベース全体のコピーを作成します。dump database は、ログをトランケートしません。

dump database では **dynamic dumps** が可能です。つまり、ダンプの実行中も、ユーザは引き続きデータベースを変更することができます。この機能は、定期的にデータベースをバックアップする場合に便利です。

## 日常のトランザクション・ログ・ダンプの作成：*dump transaction*

dump transaction コマンドを使用して、トランザクション・ログの日常のバックアップを実行します。dump transaction は、多くのオペレーティング・システムで可能なインクリメンタル・バックアップに似ています。dump transaction は、前回のトランザクション・ログのダンプ以降に行われたデータベースの変更の記録であるトランザクション・ログをコピーします。dump transaction は、ログをコピーした後、アクティブでない部分をトランケートします。

dump transaction は、必要な時間と記憶領域はデータベース全体のバックアップよりも少なく、通常はデータベースのバックアップよりも頻繁に実行されます。ダンプをとっている間、ユーザは引き続きデータベースを変更できます。dump transaction を実行できるのは、データベースのログが別のセグメントに保管されている場合だけです。

メディア障害の発生後は、dump transaction の with no_truncate オプションを使用して、トランザクション・ログをバックアップしてください。こうすることで、障害が発生した時点までのトランザクション・ログを残すことができます。

## デバイス障害後のログのコピー：*dump tran with no_truncate*

データ・デバイスに障害が発生し、データベースにアクセスできない場合は、dump transaction の with no_truncate オプションを使用して、ログの最新のコピーを取得します。このオプションはログをトランケートしません。トランザクション・ログが別のセグメントにあり、master データベースにアクセスできる場合にのみ、このオプションを使用できます。

## データベース全体のリストア：*load database*

dump database によって作成されたバックアップをロードするには、load database コマンドを使用します。既存のデータベースにダンプをロードすることも、for load オプションを使用して新しいデータベースを作成することもできます。新しいデータベースを作成する場合は、少なくとも、元のデータベースに割り付けた領域と同じ量の領域を割り付けてください。

load database コマンドは、データベースのステータスを「オフライン」に設定します。つまり、データベースをロードする前に sp_dboption の no chkpt on recovery、dbo use only、read only オプションを使用する必要はありません。ただし、データベースのロード中とそれに続くトランザクション・ログのロード中は、いかなるユーザもデータベースを使用できません。そのデータベースをユーザが使用できるようにするには、online database コマンドを発行します。

データベースをロードした後、Adaptive Server は次の作業を行う必要があります。

- ロード先のデータベースがダンプされたデータベースよりも大きい場合は、未使用のページをすべて「空白」にします。
- トランザクション・ログの変更をデータに反映しながら、リカバリを行います。

割り付けられていないページの数や長いトランザクションの数によっては、この作業が数秒で終了することもありますが、非常に大きいデータベースでは長時間かかる場合もあります。Adaptive Server は、未使用のページを初期化していること、またはリカバリを開始したことを通知します。このメッセージは通常、バッファリングされます。メッセージを見るには、次のコマンドを発行します。

```
set flushmessage on
```

## データベースへの変更の適用：*load transaction*

データベースをロードした後、load transaction コマンドを使用して、個々のトランザクション・ログ・ダンプを作成された順序でロードしてください。このプロセスによって、トランザクション・ログに記録されている変更が再実行され、データベースが再構成されます。必要であれば、load transaction の until_time オプションを使用して、データベースをトランザクション・ログ内の特定の時点までロール・フォワードするという方法でデータベースをリカバリできます。

load database コマンドと load transaction コマンドの間は、load database によってステータスが「オフライン」に設定されているため、ユーザがデータベースを変更することはできません。

Adaptive Server の同じリリース・レベルに関連するデータベースのトランザクション・ログ・ダンプだけをロードできます。

一連のトランザクション・ログ・ダンプがすべてロードされると、データベースは、最後のトランザクション・ログ・ダンプの時点でコミットされていたすべてのトランザクションが反映された状態になります。

## データベースをユーザが使用できるようにする：*online database*

ロード・シーケンスが完了したら、ユーザがそのデータベースを使用できるように、ステータスを「オンライン」に変更します。load database によってロードされたデータベースには、online database コマンドを発行するまでアクセスできません。

必要なトランザクション・ログをすべてロードしたことを確認してから、online database コマンドを発行してください。

## プラットフォーム間でのデータベースのダンプとロード

Adaptive Server では、エンディアン・アーキテクチャの異なるプラットフォーム間でもデータベースをダンプしてロードできます。これは dump database と load database をビッグ・エンディアン・プラットフォームからリトル・エンディアン・プラットフォームに対して ( またはリトル・エンディアン・プラットフォームからビッグ・エンディアン・プラットフォームに対して) 実行できることを意味します。

ビッグ・エンディアン・システムでは、記憶域 (integer や long など) の最上位のバイトから順にアドレスが付けられます。リトル・エンディアン・システムはその逆です。

---

**注意** 同じエンディアン・アーキテクチャのプラットフォーム間で dump database と load database を実行するときは、ユーザやシステムのデータを変換する必要はありません。データベースのダンプとロードに関して操作上の制限は特にありません。

---

Adaptive Server は、load database の実行中に自動的にデータベース・ダンプ・ファイルのコピー元システムのアーキテクチャのタイプを検出し、必要な変換を行います。11.9、12.0、12.5 などの旧バージョンからのロードもサポートしています。32 ビットから 64 ビット・プラットフォームへのダンプとロード (またはその逆) も可能です。

サポート対象プラットフォーム：

| | | | |
|---|---|---|---|
| ビッグ・エンディアン | Sun Solaris | IBM AIX | HPPA、HPIA 上の HP-UX |
| リトル・エンディアン | Linux IA | Windows | Sun Solaris x86 |

特定のプラットフォームの組み合わせについて load database が実行されると、ストアド・プロシージャやその他のコンパイル済みオブジェクトは、その後最初に実行されるとき *syscomments* 内の SQL テキストから再コンパイルされます。

## データベースのダンプ

プラットフォーム間のダンプとロードでは、dump database を実行する前に、次の手順でデータベースをトランザクションが実行されていない状態に移行する必要があります。

1　dbcc checkdb と dbcc checkalloc を実行して、データベースの整合性に問題がないことを確認します。

2　sp_dboption を使用して、データベースをシングルユーザ・モードにします。

3　sp_flushstats で統計値を systabstats へフラッシュします。

4　10 ～ 30 秒待ちます。これはデータベースの規模と活動状況によって異なります。

5　データベースに対して checkpoint を実行して、更新済みのページをフラッシュします。

6　dump database を実行します。

## データベースのロード

データベースをロードすると、Adaptive Server はダンプ・ファイルのエンディアン・タイプを自動的に調べ、load database コマンドと online database コマンドの実行中に必要なすべての変換を実行します。

Adaptive Server がインデックス・ローを変換すると、インデックス・ローの順序が正しくなくなることがあります。Adaptive Server は、online database の実行中にユーザ・テーブルの次のインデックスをサスペクト (疑わしい) インデックスとマークします。

- APL テーブルのノンクラスタード・インデックス
- DOL テーブルのクラスタード・インデックス
- DOL テーブルのノンクラスタード・インデックス

プラットフォーム間のダンプ操作とロード操作時、疑わしいパーティションは次のように処理されます。

- 初めての online database コマンドの実行中にエンディアン・タイプの異なる 2 つのプラットフォームで load database を実行すると、ハッシュ分割に suspect のマークが付けられます。
- unichar または varchar 分割キーで内部生成されたパーティション条件を持つラウンドロビン分割のグローバル・クラスタード・インデックスは、“suspect (疑わしい)” のマークが付けられます。
- データベースがオンラインになったら、sp_post_xpload を使用して疑わしい分割およびインデックスを修正します。

## データベースとトランザクションのダンプおよびロードについての制限事項

- dump transaction と load transaction をプラットフォーム間で実行することはできません。
- リモート Backup Server に対する dump database と load database をプラットフォーム間で実行することはできません。
- パスワードで保護されたダンプ・ファイルをプラットフォーム間でロードすることはできません。
- 解析済み XML オブジェクトに対して dump database と load database を実行する場合は、load database の実行後にテキストを再度解析する必要があります。
- ダンプのロードは、ダンプ元のサーバと同じソート順のサーバに対してのみ実行できます。たとえば、辞書順で大文字と小文字の区別がありアクセントを区別するソート順を使用するサーバから、辞書順で大文字と小文字の区別がなくアクセントを区別しないソート順を使用するサーバへはダンプをロードできません。
- dump database と load database を Adaptive Servers バージョン 11.9 より前のプラットフォーム間で実行することはできません。
- Adaptive Server は、binary、varbinary、または image カラムとして格納されている埋め込みデータ構造を変換することはできません。
- master データベースに対する load database をプラットフォーム間で実行することはできません。
- ストアド・プロシージャやその他のコンパイル済みオブジェクトは、load database の実行後最初に実行されるとき syscomments 内の SQL テキストから再コンパイルされます。

  テキストから再コンパイルするパーミッションがない場合は、パーミッションを持つ人が dbcc upgrade_object でテキストから再コンパイルしてオブジェクトをアップグレードする必要があります。

---

**注意** master データベースの syslogins システム・テーブル内のログイン・レコードを Solaris から Linux へ移行する場合は、bcp を文字フォーマットで使用できます。今回のリリースから、Solaris プラットフォームのログイン・パスワードは、トレース・フラグのない Linux にも適合します。これ以外のプラットフォームの組み合わせについては、パスワードが適合しないのでログイン・レコードを作り直す必要があります。

---

## パフォーマンスに関する注意

インデックス・ローは、テーブルのデータ・ローに高速にアクセスできるように並べ替えられます。ユーザ・テーブルへの高速なアクセスを実現するために、ロー識別子を持つインデックス・ローはバイナリとして扱われます。

同じアーキテクチャのプラットフォーム内であればインデックス・ローの順序は相変わらず有効で、与えられた選択基準に対する検索順序はいつものパスをとります。しかし、インデックス・ローが異なるアーキテクチャ間で変換されると最適化が行われた順序が無効になり、プラットフォーム間のダンプとロードでユーザ・テーブルに無効なインデックスが作成されます。

ビッグ・エンディアンからリトル・エンディアンというように、異なるアーキテクチャのデータベース・ダンプをロードすると、一部のインデックスがサスペクトとマークされます。

- APL テーブルのノンクラスタード・インデックス
- DOL テーブルのクラスタード・インデックス
- DOL テーブルのノンクラスタード・インデックス

ターゲット・システムのインデックスを修復するには、アーキテクチャの異なるダンプのロード後、次の 2 つの方法のどちらかを使うことができます。

- すべてのインデックスを削除して作り直す。
- sp_post_xpload を使用する。

一般的に、大きなテーブルでインデックスを作り直すと相当時間がかかるので、計画的に行う必要があります。

sp_post_xpload は、インデックスを検証し、無効なインデックスを削除し、削除したインデックスを作り直すまでの一連の操作をデータベースに対して単一のコマンドで実行します。sp_post_xpload は多くの操作を実行するので、インデックスを削除して作り直す方法よりも時間がかかることがあります。10G 以下のデータベースでは、sp_post_xpload を使用してください。10GB を超えるデータベースに対しては、インデックスを削除して作り直す方法を使用することをおすすめします。

## 別の Adaptive Server へのデータベースの移動

dump database と load database を使用して、データベースを特定の Adaptive Server から別の Adaptive Server に移動させることができます。ただし、必ず、移動先の Adaptive Server と移動元の Adaptive Server のデバイス割り付けが一致するようにしてください。一致していない場合は、新しいデータベース内のシステム定義セグメントとユーザ定義セグメントが、移動元のデータベース内のものと一致しないことになります。

新しい Adaptive Server のデータベース・ダンプをロードするときにデバイス割り付けを維持するには、障害が発生したデバイスからユーザ・データベースをリカバリする場合と同じ手順に従ってください。「領域使用量の調査」(402 ページ) を参照してください。

また、システム・データベースを異なるデバイスに移動する場合は、次の一般ガイドラインに従ってください。

- master データベースを移動する前には必ず、マスタ・デバイスのミラーリングを解除してください。このようにしないと、新しいデバイスを使用する Adaptive Server を起動するときに、Adaptive Server は古いミラー・デバイス・ファイルを使用しようとします。
- master データベースを移動するときは、sysdevices での割り付けエラーを防ぐために、新しいデバイスのサイズが元のデバイスと同じになるようにしてください。
- sybsecurity データベースを移動するには、新しいデータベースをシングルユーザ・モードに設定してから、そのデータベースに古いデータをロードしてください。

## ユーザ・データベースのアップグレード

バージョン 11.9 以降の Adaptive Server のダンプは、最新バージョンの Adaptive Server にロードすることができます。ロードされたデータベースは、online database を発行するまではアップグレードされません。

ユーザ・データベースのアップグレード手順は、次のように、システム・データベースのアップグレード手順と同じです。

1 load database を使用して、Adaptive Server バージョン 11.9 以降のデータベース・ダンプをロードします。load database を実行すると、データベースのステータスは「オフライン」になります。

2 load transaction を使用して、前回のデータベース・ダンプ以後に作成されたすべてのトランザクション・ログを順番にロードします。すべてのトランザクション・ログをロードしてから、手順 3 に進みます。

3 online database を使用してアップグレードを実行します。online database コマンドによってデータベースがアップグレードされるのは、そのデータベースの現在の状態が Adaptive Server の最新バージョンと互換性がないためです。アップグレードが完了すると、データベースのステータスは「オンライン」になり、一般のユーザが使用できるようになります。

4 アップグレードしたデータベースのダンプを作成します。dump database を実行しなければ、dump transaction コマンドは実行できません。

『リファレンス・マニュアル：コマンド』を参照してください。

## *dump transaction* の特別なオプションの使用方法

状況によっては、前述の単純なモデルがあてはまらないことがあります。表 12-1 は、標準の dump transaction コマンドの代わりに、特別な with no_log オプションと with truncate_only オプションを使用する場合の説明です。

**警告！** 特殊な dump transaction コマンドは、表 12-1 に示す場合のみに使用してください。特に dump transaction with no_log は最後の手段として使用し、dump transaction with no_truncate が失敗した後に一度だけ使用するようにしてください。dump transaction with no_log コマンドは、トランザクション・ログ内のほんのわずかな領域しか解放しません。dump transaction with no_log を入力した後もデータのロードを継続すると、ログが完全に満杯になって、以降の dump transaction コマンドが失敗する可能性があります。データベースに追加領域を割り付けるには、alter database コマンドを使用してください。

**表 12-1: dump transaction with truncate_only または dump transaction with no_log を使用する条件**

| 条件 | 使用 |
|---|---|
| ログがデータと同じセグメントにある場合 | ログをトランケートするための dump transaction with truncate_only<br>ログを含むデータベース全体をコピーするための dump database |
| 最近のトランザクションのリカバリを必要としない場合 (初期段階の開発環境など) | ログをトランケートするための dump transaction with truncate_only<br>データベース全体をコピーするための dump database |
| ログ領域の空きがなくなり、通常のトランザクション・ログのダンプ (標準の dump transaction コマンドまたは dump transaction with truncate_only) が失敗した場合 | イベントを記録しないでログをトランケートするための dump transaction with no_log<br>ログを含むデータベース全体をコピーするために dump database を直後に使用 |

## ダンプ・ファイルを確認するための特殊なロード・オプションの使用方法

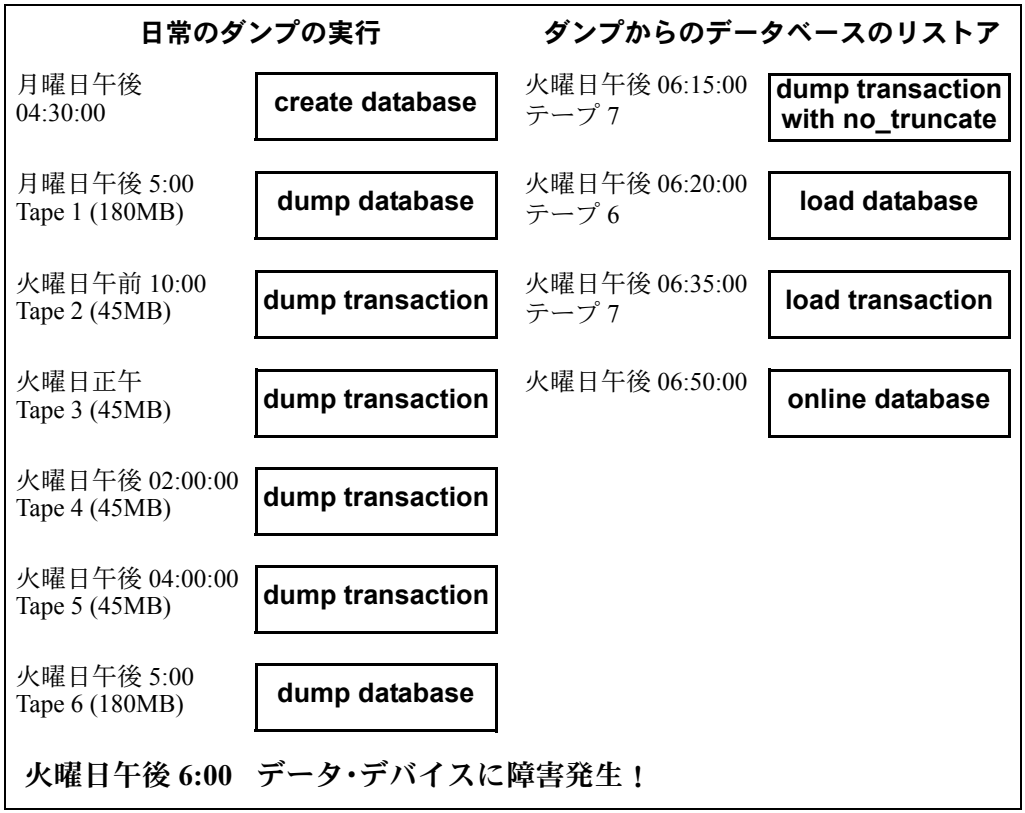
指定したファイルまたはテープ上の最初のファイルのヘッダ情報を表示するには、with headeronly オプションを使用します。テープ上のすべてのファイルに関する情報を返すには、with listonly オプションを使用してください。これらのオプションは、実際にはテープ上のデータベースまたはトランザクション・ログをロードしません。

**注意** これらのオプションはどちらか 1 つだけしか指定できません。両方を指定すると、with listonly が優先されます。

## バックアップからのデータベースのリストア

図 12-3 は、月曜日の午後 4 時 30 分に作成され、その直後にダンプされたデータベースをリストアするプロセスを示します。データベース全体のダンプは毎日午後 5 時に行われます。トランザクション・ログ・ダンプは、毎日午前 10 時、正午、午後 2 時、午後 4 時に行われます。

**図 12-3: データベースのリストア (ケース 1)**




データを保管するディスクの障害が火曜日の午後 6 時に発生した場合は、次の手順でデータベースをリストアします。

1 dump transaction with no_truncate を使用して、最新のトランザクション・ログ・ダンプを取得します。

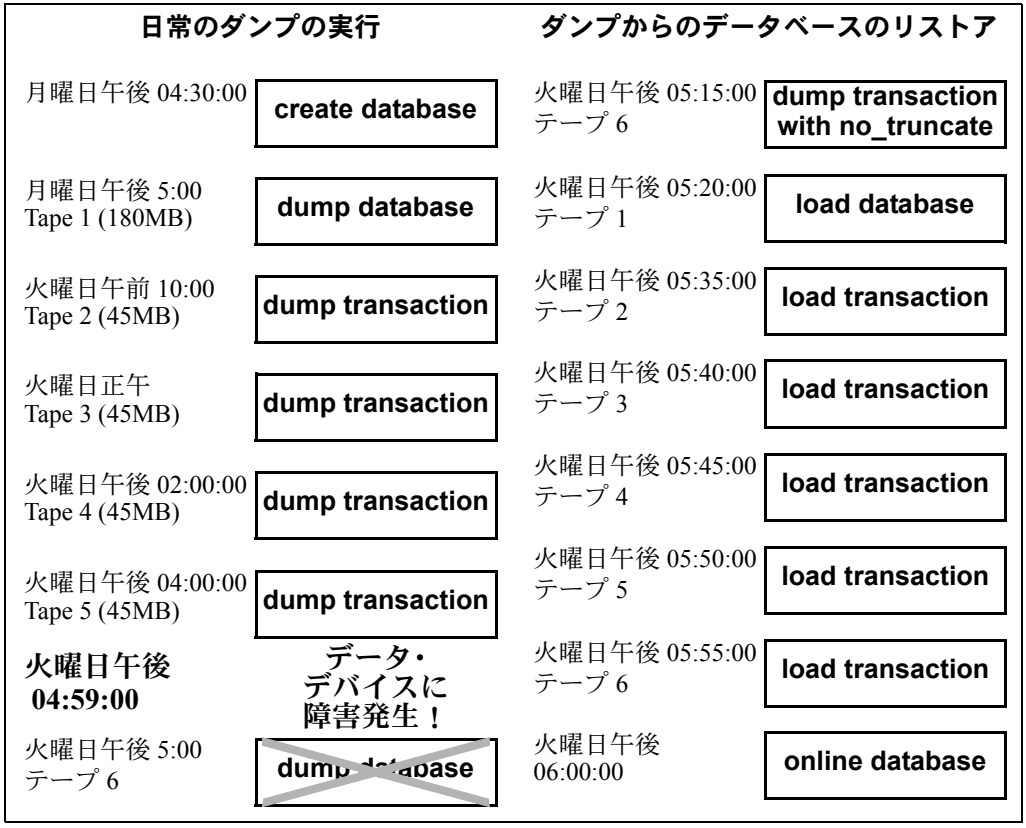
2 load database を使用して、最新のデータベース・ダンプであるテープ 6 をロードします。load database を実行すると、データベースのステータスは「オフライン」になります。

3 load transaction を使用して、最新のトランザクション・ログ・ダンプであるテープ 7 を適用します。

4　online database を使用して、データベースのステータスを「オンライン」にします。

図 12-4 は、データ・デバイスの障害が火曜日の午後 4 時 59 分、つまりオペレータが毎日のデータベース・ダンプを行う予定の時刻の直前に発生した場合に、データベースをリストアする方法を示します。

**図 12-4: データベースのリストア (ケース 2)**




次の手順でデータベースをリストアします。

1　dump transaction with no_truncate を使用して、テープ 6 (本来であれば日常のデータベース・ダンプに使用するテープ) に最新のトランザクション・ログをダンプします。

2　load database を使用して、最新のデータベース・ダンプであるテープ 1 をロードします。load database を実行すると、データベースのステータスは「オフライン」になります。

3　load transaction を使用して、テープ 2、3、4、5、および最新のトランザクション・ログ・ダンプであるテープ 6 をロードします。

4 online database を使用して、データベースのステータスを「オンライン」にします。

# データベースへの更新の中断と再開

quiesce database hold を使用すると、ディスクのミラーリング解除や個々のデータベース・デバイスの外部コピーを実行する間、そのデータベースに対する更新をブロックすることができます。この間は書き込みが実行されないため、データベースの外部コピー (セカンダリ・イメージ) はプライマリ・イメージと全く同じです。データベースがクワイス (静止) 状態のときは、そのデータベースに対して実行できるのは読み込みクエリだけです。データベースへの更新を再開するには、外部コピーの操作が完了したときに、quiesce database release を発行します。データベースの外部コピーをセカンダリ・サーバにロードするときも、プライマリ・イメージのコピーのトランザクションの一貫性が保証されます。isql 接続から quiesce database hold を発行し、次に別の isql 接続からログインして quiesce database release を発行することができます。『リファレンス・マニュアル：コマンド』を参照してください。

---

**注意** *tag_name* は、識別子の規則に従っていなければなりません。quiesce database...hold と quiesce database...release の両方に、同じ *tag_name* を使用してください。

---

たとえば、pubs2 データベースへの更新を中断するには、次のコマンドを発行します。

```
quiesce database pubs_tag hold pubs2
```

Adaptive Server は、次のようなメッセージをエラー・ログに書き込みます。

```
QUIESCE DATABASE command with tag pubs_tag is being executed by process 9.
Process 9 successfully executed QUIESCE DATABASE with HOLD option for tag pubs_tag.
Processes trying to issue IO operation on the quiesced database(s) will be suspended until
user executes Quiesce Database command with RELEASE option.
```

pubs2 データベースへの更新は、このデータベースが解放されるまで遅延し、解放されると更新が完了します。pubs2 データベースを解放するには、次のコマンドを発行します。

```
quiesce database pubs_tag release
```

for external dump 句を使用していた場合は、データベースを解放した後で、-q パラメータを使用してセカンダリ・サーバを起動することができます。リカバリの実行によって、データベースのトランザクションの一貫性が保たれた状態になります。または、データベースがオンラインになるまで待ってからトランザクション・ログを適用することができます。

## quiesce database の使用法のガイドライン

quiesce database を使用する一番簡単な方法は、インストール済みシステム全体を完全にコピーします。これによって、システムのマッピングは一貫性を保ちます。システムのマッピングが格納されているシステム・データベースを、quiesce database hold のデータベース・セットの一部として物理的にコピーすると、そのマッピング内容がセカンダリのインストール済みシステムに移植されます。このマッピングが実行されるのは、元のインストール済みシステム内のすべてのユーザ・データベースが、同じセットの一部としてコピーされたときです。quiesce database では、1 回の操作で 8 つまでデータベース名を指定できます。元のインストール済みシステムに 9 つ以上のデータベースがある場合は、quiesce database hold コマンドを 2 回以上発行して、複数のデータベース・セットに対応する複数の同時クワイス (静止) 状態を作り出すことができます。

元のインストール済みシステムを新たに作成するときは、プライマリ・システムを作成するスクリプトとセカンダリ・システムを作成するスクリプトはほとんど同じですが、セカンダリ・システムのスクリプトでは、disk init コマンドに渡す物理デバイス名が異なる場合があります。この方法では、プライマリ・サーバでのシステム・デバイスに対する更新を、同一の変更によってセカンダリ・サーバにも反映する必要があります。たとえば、alter database コマンドをプライマリ・サーバ上で実行するときは、同じコマンドを、同じパラメータを使ってセカンダリ・サーバ上でも実行しなければなりません。また、この方法では、データベース・デバイスがボリューム・マネージャによってサポートされている必要があります。ボリューム・マネージャとは、物理的に明確に区別されているデバイスに対して、プライマリ・サーバとセカンダリ・サーバの両方で同じ物理名を使用できるようにするためのものです。

使用しているシステムで、データベース・デバイスの外部コピーを作成する独自のプロシージャを開発することができますが、次のような作業を実行することをおすすめします。

- master データベースを quiesce database のデータベース・リストに組み込みます。
- クワイスしたデータベースでディスクへの書き込みを防ぐプロセスは、別のプロセスの実行を防ぐリソースを保持している可能性があります。たとえば、特定のプロセスがトランザクションでデータベース・ページを変更しても、**コミット**時のログ・ページのフラッシュを防いでいる場合、このプロセスは排他ページ・ロックを保持していることが考えられ、quiesce database 操作中にリーダが同じページの共有ページ・ロックを取得しようとしてもブロックする場合があります。

この問題はシステム・データベース (監査が有効の場合、sybsystemprocs、sybsystemdb、または sybsecurity) をクワイス状態にすると発生しますが、master データベースに頻繁に使用されるシステム・テーブルが多数存在すると、この master データベースを**クワイス状態にした**ときに最も深刻になります。たとえば、特定のプロセスが create login で syslogins を変更しても、master データベースのクワイス時にトランザクションのコミットを防止されていると、syslogins を変更するために取得した排他ロックによってログインがブロックされます。これは、ログインするには syslogins に対する共有ページのロックを入手する必要があるためです。

**注意** master データベースやその他のシステム・データベースをクワイス状態にすると、クワイス状態になっているデータベースの更新を試みるプロセスをブロックすることになるため、サーバ・パフォーマンスに重大な影響を及ぼす可能性があります。

- プライマリ・サーバとセカンダリ・サーバの両方で、同じ文字列を使ってデバイス名を指定します。
- プライマリおよびセカンダリのインストール済みシステム内にある master、model、sybsystemprocs の各システム・データベース用の環境を同じにします。特に、コピーされたデータベースの sysusages マッピングとデータベース ID が、プライマリ・サーバとセカンダリ・サーバで同一になるようにします。また、両方のサーバで、sysdatabases 内に反映されるデータベース ID がまったく同一になるようにします。
- syslogins.suid と sysusers.suid との間のマッピングの一貫性が、セカンダリ・サーバでも保たれるようにします。
- プライマリ・サーバとセカンダリ・サーバが master のコピーを共有し、コピーされた各デバイスの sysdevices エントリで同じ文字列を使用する場合は、両方のサーバ内の *physname* 値が物理的に明確に区別されているようにします。
- 次の制限事項に従って、データベースの外部コピーを作成します。
  - コピー処理を始めることができるのは、quiesce database hold の完了後です。
  - quiesce database のデータベース・リスト内にあるすべてのデータベースに対応する全デバイスをコピーします。
  - 外部コピーが終了してから quiesce database release コマンドを呼び出します。

- quiesce database で指定された、外部コピー操作を行うための期間内は、quiesce database のデータベース・リストで指定されたデータベースに属するディスク領域に対する更新を行うことはできません。この領域は、sysusages 内で定義されています。ただし、quiesce database のデータベース・リストにあるデータベースとリストにないデータベースの間でデバイス上の領域が共有されている場合は、外部コピーの作成中にこの共有デバイスへの更新が行われる可能性もあります。外部コピーを作成するシステム内のどの場所にデータベースを配置するかを決定するには、次のいずれかを実行します。
  - quiesce database を使用する環境内でデータベースがデバイスを共有しないように、データベースを隔離します。
  - デバイス上のデータベースすべてをコピーします (インストール済みシステム全体のコピーをすすめている上記の説明のとおりです)。
- データベース上での更新アクティビティがほとんどない場合のみ、quiesce database を使用します (読み込みのみのアクティビティが実行されている間が望ましい )。アクティビティが少ない時間帯にデータベースをクワイス状態にすると、ユーザの作業への影響を少なくできるだけでなく、コピー操作に関与するデバイスを同期化するための時間も短縮できる可能性があります。ただし、これは、外部コピーを実行するサードパーティの I/O サブシステムに依存します。
- mount コマンドと unmount コマンドを使用すると、データベースの移動やコピーを簡単に実行できます。データベースを特定の Adaptive Server から別の Adaptive Server に移動したりコピーしたりする際に、サーバを再起動する必要はありません。また、複数のデータベースを同時に移動またはコピーすることができます。

  これらのコマンドを使用すると、デバイスを物理的に移動してデータベースを再度アクティブにすることもできます。

  データベースに unmount を実行すると、データベースとそのデバイスが Adaptive Server から削除されます。unmount はデータベースを停止して Adaptive Server から削除するコマンドです。デバイスも非アクティブ化されて削除されます。マウントが解除されているときは、データベースとそのページに対する変更は行われません。

## プライマリ・サーバとセカンダリ・サーバ間の役割関係の管理

システムが 2 つの Adaptive Server で構成されており、1 つがプライマリ・サーバとして機能し、もう 1 つがプライマリ・サーバのデータベースの外部コピーを受け取るセカンダリ・サーバとして動作する場合は、絶対に両者の役割を混合させないでください。つまり、各サーバが果たす役割を入れ替える (プライマリ・サーバをセカンダリ・サーバに変更、またはその逆に変更する) ことはできますが、この 2 つの役割を同時に 1 つのサーバに与えることはできません。

## -*q* オプションによるセカンダリ・サーバの起動

dataserver -q オプションを使用して、セカンダリ・サーバを指定します。-q オプションは、プライマリ・サーバを起動するときは使用しないでください。-q オプションを使用すると、quiesce database for external dump の期間にコピーされたユーザ・データベースは、次のような操作が行われるまでオフライン状態のままになります。

- プライマリ・サーバ上にあるデータベースのトランザクション・ログを、**スタンバイ・アクセスによって** (つまり dump tran with standby_access を使用して) ダンプしてから、load tran を使用してセカンダリ・サーバ上にあるこのデータベースのコピーにダンプし、次にこのデータベースに対して online database for standby access を実行します。
- 読み込みと書き込みアクセスを実行できるように online database を発行してデータベースを強制的にオンラインにします。ただし、この操作を行うと、データベースのリカバリ時に補正ログ・レコードが書き込まれます。また、データベースをロードするか、quiesce database を使用してプライマリ・デバイスの新しいコピーを作成しなければ、トランザクション・ログはロードできません。

システムのデータベースは、-q オプションに関係なくオンラインになり、また、トランザクションがロールバックされたときは必ず補正ログ・レコードが書き込まれます。

## 更新された「クワイス状態」データベースのログ・レコード値

dataserver の -q オプションを使用してセカンダリ・サーバを起動すると、in quiesce (クワイス状態) であるとして内部的にマーク付けされているユーザ・データベースごとに、そのデータベースが "`in quiesce`" であることを通知するメッセージが出力されます。

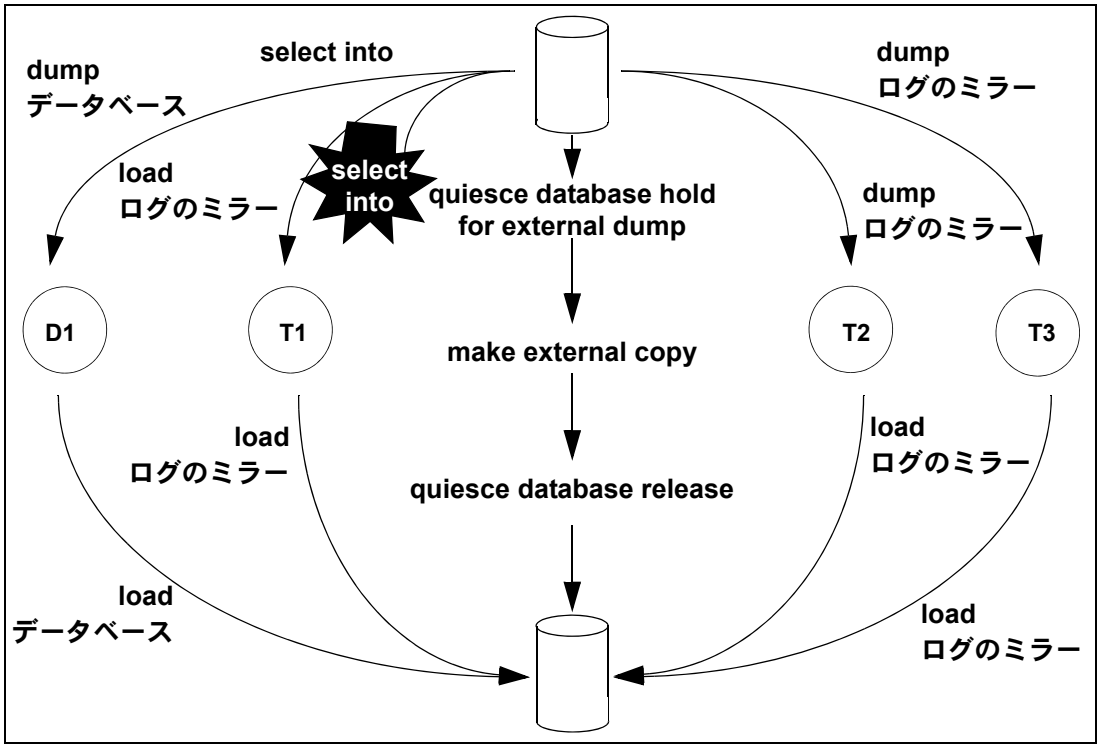
quiesce database for external dump を使用してコピーされたデータベースに対する -q によるリカバリの動作は、load database の実行時のリカバリとほとんど同じです。load database に対応するリカバリと同様に、dataserver -q はカレント最終ログ・レコードのアドレスを内部的に記録します。これによって、後続の load transaction が、このアドレスを前回のカレント最終ログ・レコードと比較することができます。この 2 つのアドレス値が一致しない場合は、セカンダリ・データベースにおいてアクティビティが発生したことを表し、エラー番号 4306 が出力されます。

## ダンプ・シーケンス番号の更新

dump database と同様に、quiesce database は、ログを取らない書き込みがある場合に、ダンプ・シーケンス番号を更新します。これによって、ダンプ・シーケンス番号の起点として古いデータベース・ダンプや外部コピーを誤って使用することを防ぎます。

たとえば、図 12-5 で説明しているウォーム・スタンバイ方法では、dump database (D1)、dump transaction (T1)、quiesce database、dump transaction (T2)、および dump transaction (T3) によってアーカイブが作成されます。

図 12-5: ウォーム・スタンバイのダンプ・シーケンス




一般に、ログを取らない更新が行われ、dump tran with truncate_only を実行しない環境では、quiesce database hold を省略して D1、T1、T2、T3 を順番にロードします。この方法はウォーム・スタンバイ・モードのときに使用されます。このモードでは、プライマリ・サーバ上でその後にデータベース・ダンプを実行することにより、メディア障害が発生しても単純な処理でリカバリできるようになります。意思決定支援システムに使用されるセカンダリ・サーバ (すなわちスタンバイ・サーバ) では、外部コピー操作によって中断する代わりに、load transaction を連続して実行することによって差分を適用してもかまいません。

ただし、dump transaction によって T1 が作成された後で、ログを取らない操作 ( たとえば 図 12-5 の select into) が発生すると、その後で dump transaction to archive を実行することはできません。この場合、データベースのダンプを作成し直すか、quiesce database for external copy を発行してからデータベースの新しい外部コピーを作成する必要があります。これらのコマンドのいずれかを発行すると、ダンプ・シーケンス番号が更新され、dump transaction to archive をブロックしているマークがクリアされます。

for external dump 句を使用するかどうかは、クワイス状態 (`in quiesce`) としてマーク付けされるデータベースをリカバリ時にどのように処理するかによって決まります。

quiesce database hold

for external dump 句を使用せずに quiesce database を発行した場合、外部コピー操作によってデータベースのセカンダリ・セットを作成する間はセカンダリ・サーバは稼働しません。-q が指定されたときのリカバリでは、コピーされたデータベースの中に in quiesce (クワイス状態) のものは存在しません。つまり、起動時のリカバリでは、各サーバは通常の方法でリカバリされ、既に説明したような for load database に対応するリカバリは行われません。その後で、これらのデータベースのいずれかに対して load tran を実行しようとしても、ロードは許可されず、エラー 4306「最後のロード以来、データベースに変更があったため ...」またはエラー 4305「指定されたファイル '%.*s' は、シーケンス外です ...」が発生します。

ログを取らないアクティビティがプライマリ・データベース内にあるかどうかに関係なく、quiesce database hold を実行してもダンプ・シーケンス番号が増えることはありません。また、quiesce database release を実行しても、ログを取らない書き込みのビットがクリアされることはありません。

クワイスされたデータベースに対してクエリを実行しようとすると、Adaptive Server は次のようなエラー・メッセージ 880 を返します。

```
Your query is blocked because it tried to write and
database '%.*s' is in quiesce state. Your query will
proceed after the DBA performs QUIESCE DATABASE RELEASE
```

データベースがクワイス状態でなくなると、クエリが実行されます。

quiesce database hold for external dump

外部ダンプを行うために quiesce database for external dump を発行すると、そのデータベースの外部ダンプは、クワイス状態の期間に作成されたことを「記憶」します。その結果、-q が指定された場合のリカバリ時に、その外部ダンプは load database の実行時と同じようにリカバリされます。quiesce database release を実行すると、この情報はプライマリ・データベースからクリアされます。ログを取らない書き込みが存在するためにプライマリ・サーバ上で dump tran *to archive* を実行できなかった場合も、この時点では、dump tran *to archive* を実行できるようになっています。

quiesce database のリストで指定されたデータベースに対して、前回の dump database または quiesce database hold for external dump の実行以降にログを取らない書き込みが発生している場合は、quiesce database hold for external dump によってダンプ・シーケンス番号が更新されます。また、ログを取らない書き込みの情報は、quiesce database release によってクリアされます。更新されたシーケンス番号のダンプを load tran を使用して適用するとき、適用先が、そのシーケンス番号を更新した quiesce database によって作成された外部コピー以外のものである場合は、load tran は異常終了します。これは、ログを取らない書き込みが存在するデータベースに対して dump database を実行したときの動作に似ています。

---

**警告！** quiesce database for external dump を実行すると、dump transaction to *archive_device* の実行を禁止する内部フラグがクリアされます。これは、実際に外部コピーを作成するか、データベース・ダンプを実行するかには関係ありません。quiesce database コマンドは、外部コピーが作成されたかどうかを認識することはできません。外部コピーの作成は、システム管理者の責任において行ってください。quiesce database hold for external dump を使用する目的が、新しいダンプ・シーケンスの起点の役割となるコピーを実際に実行することではなく、一時的な書き込み保護であり、アプリケーション内でログを取らない書き込みが行われる場合も、トランザクション・ログ・ダンプの作成は可能ですが、このログ・ダンプは使用できません。この場合、dump transaction to *archive_device* initially は、最初は正しく実行されますが、その後で実行するトランザクションのロードのコマンドでは、シーケンス番号が正しくないため、このアーカイブは拒否されます。

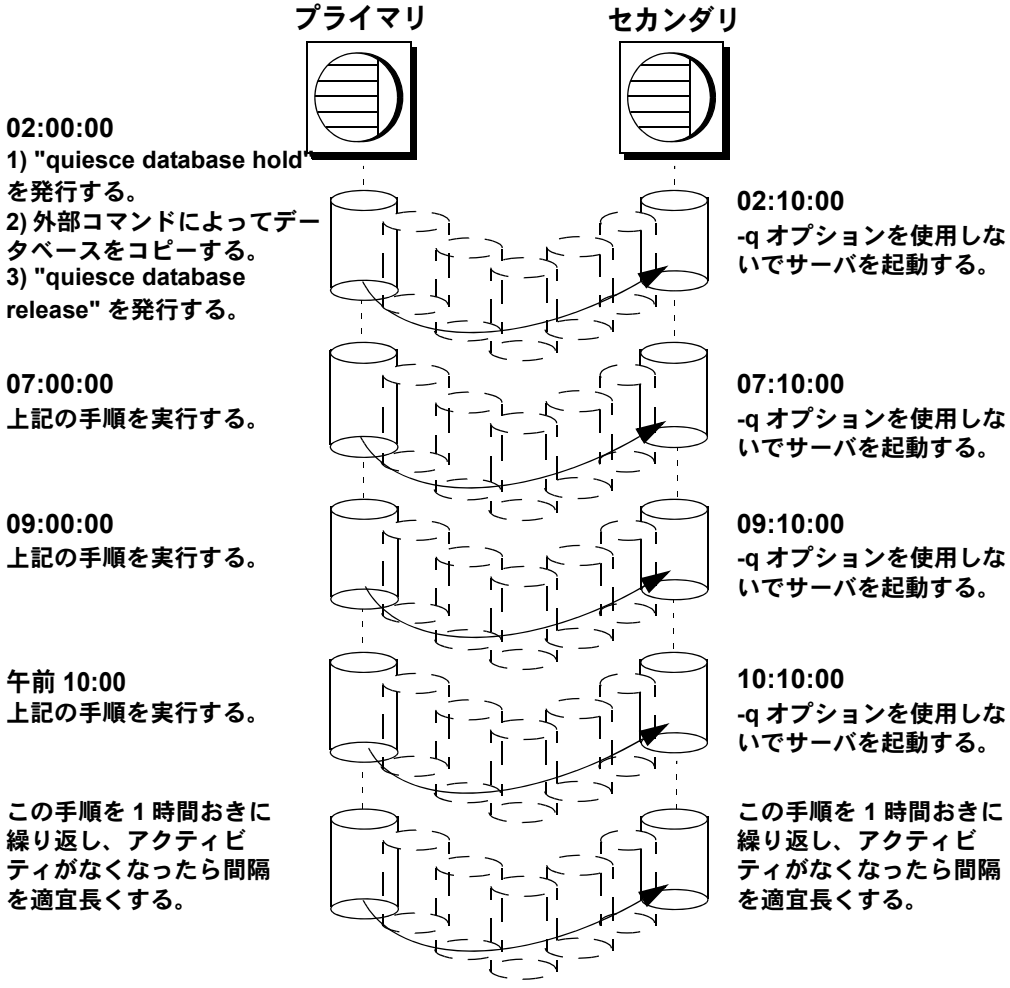
---

## *quiesce database* によるプライマリ・デバイスのバックアップ

一般的に、quiesce database を使用してユーザ・データベースをバックアップする場合は、以下で説明する 2 つのいずれかの方式でバックアップを実行します。どちらの方式でも、通常の操作中に、意思決定支援アプリケーションの負荷を OLTP (オンライン・トランザクション処理) サーバから別のサーバに移すことができます。

- プライマリ・デバイスの反復リフレッシュ方式 – リフレッシュ間隔が経過するたびに、プライマリ・デバイスをセカンダリ・デバイスにコピーします。各リフレッシュの前にデータベースをクワイス状態にします。図 12-6 に、この方式を使用した週 1 回のバックアップの例を示します。

**図 12-6: 反復リフレッシュ方式のバックアップ・スケジュール**

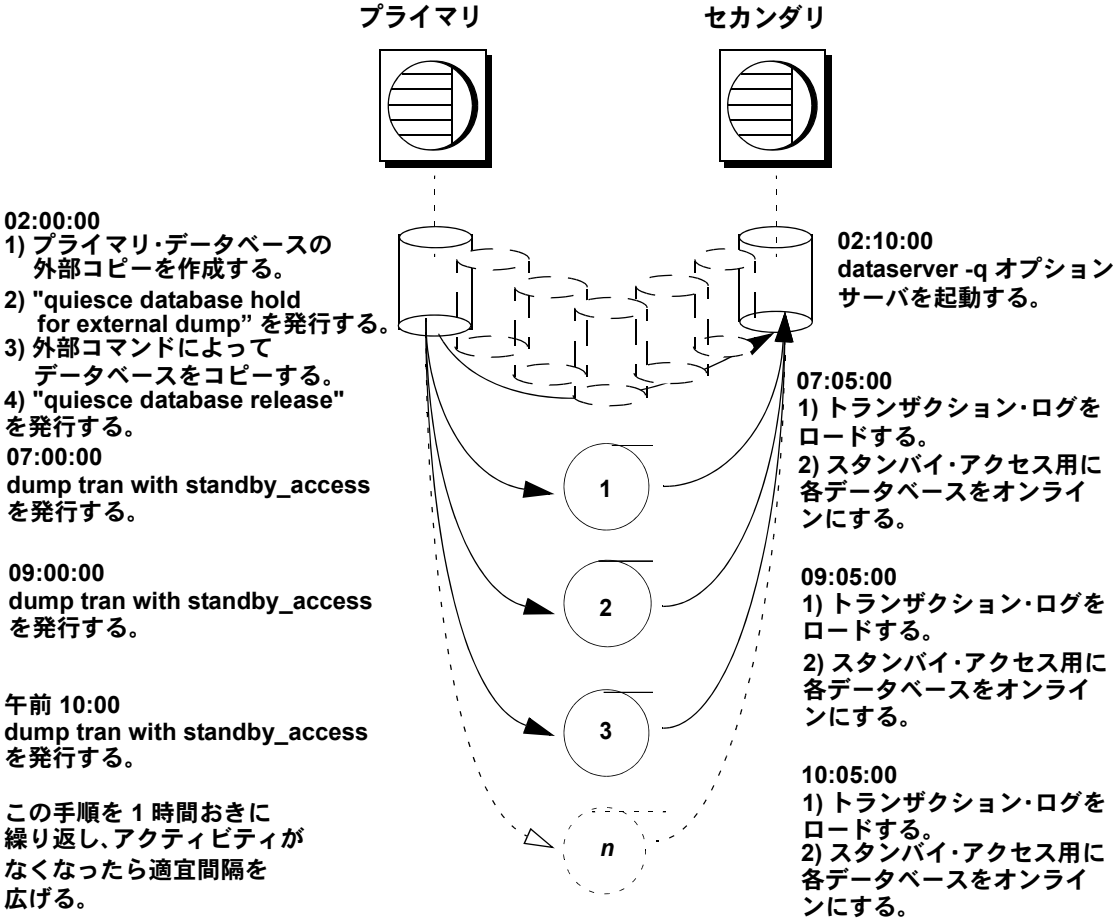



反復リフレッシュ方式を使用しているときは、クラッシュ発生後またはシステムのメンテナンス後にセカンダリ・サーバを起動するときに -q オプションを使用する必要はありません。トランザクションが正しく完了されなかった場合は、補正ログ・レコードが生成され、影響を受けたデータベースが通常の方法でオンラインになります。

- ウォーム・スタンバイ方式 − 書き込みをブロックしないため、OLTP サーバの完全な同時実行が可能になります。

for external dump 句を使用してプライマリ・デバイスの外部 (セカンダリ) コピーを作成した後、プライマリ・サーバからダンプしたトランザクション・ログを定期的に適用することによって、セカンダリ・データベースをリフレッシュします。この方式では、データベースを一度クワイス状態にしてデータベース・セットの外部コピーを作成し、次に、dump tran with standby_access を使用して各データベースを定期的にリフレッシュします。図 12-7 に、プライマリ・デバイスの更新を毎日実行し、1 時間ごとにトランザクション・ログのバックアップを作成する方式を示します。

**図 12-7: ウォーム・スタンバイ方式のバックアップ・スケジュール**



## ウォーム・スタンバイ方式のデータベース・リカバリ

ウォーム・スタンバイ方式を使用する場合は、プライマリ・サーバとセカンダリ・サーバのどちらを起動しようとしているのかを Adaptive Server に認識させる必要があります。セカンダリ・サーバを起動するように指定する場合は、dataserver コマンドの -q オプションを使用します。サーバの起動時に -q オプションを使用しない場合は、次のような状態になります。

- 各データベースは正常にリカバリされるが、load database に対応するリカバリは行われない。
- quiesce database の発行時にコミットされていなかったトランザクションはロールバックされる。

「-q オプションによるセカンダリ・サーバの起動」(338 ページ ) を参照してください。

リカバリ・シーケンスは、データベースが in quiesce (静止) としてマーク付けされているかどうかによって異なります。

### "in quiesce" のマークが付いていないデータベースのリカバリ

-q オプションを指定したとき、in quiesce のマークが付いていないデータベースは、プライマリ・サーバ内にある場合と同じようにリカバリされます。つまり、前回のオペレーションからのロード・シーケンス内に現在そのデータベースがなくても、そのデータベースは完全にリカバリされ、オンライン化されます。完了していないトランザクションはリカバリ時にロールバックされ、補正ログ・レコードが書き込まれます。

### "in quiesce" のマークが付いているデータベースのリカバリ

- ユーザ・データベース ― in quiesce のマークが付けられたユーザ・データベースは、load database の実行時に行われるデータベース・リカバリと同じ方法でリカバリされます。これによって load tran は、前回サーバが停止されてからプライマリ・データベース内で行われたアクティビティを検出することができます。-q オプションを指定してセカンダリ・サーバが起動された後、リカバリ・プロセスは in quiesce マークを検出します。Adaptive Server は、データベースがロード・シーケンス内にあることと、オフライン状態のままであることを通知するメッセージを出力します。ウォーム・スタンバイ方式を使用するときは、意思決定支援システムの役割に対応するデータベースをオンラインにする前に、必ず、dump tran with standby_access によって作成された最初のトランザクション・ダンプをロードしてください。その後で、online database for standby_access を使用してください。
- システム・データベース ― システム・データベースはすぐに完全にオンラインになります。in quiesce マークは消去されて無視されます。

## クワイス・ステータス中のアーカイブ・コピーの作成

quiesce database hold for external dump は、クワイス・ステータス中にデータベースの外部コピーを作成する意思を示します。外部コピーの作成は quiesce database hold の発行後に行われるため、データベースのトランザクションの一貫性が保たれます。これは、quiesce database hold が発行された後 quiesce database release が発行されるまでは書き込みが発生しないことが保証され、起動時のリカバリと同じ方法でリカバリが実行できるからです。このプロセスについては、図 12-5 (340 ページ) で説明しています。

ログを取らない更新が発生せず、dump tran with truncate_only も行われない環境では、quiesce database...hold コマンドを省略して D1、T1、T2、T3 を順番にロードすることになります。ただし、トランザクションのダンプによって T1 が作成された後に、ログを取らないオペレーション ( たとえば 図 12-5 の select into) が発生した場合は、トランザクションのダンプによってアーカイブを作成することはできなくなります。

quiesce database hold for external dump 句を使用すると、この問題が解決されます。この句は、次回実行される dump transaction to archive の使用を禁止するステータス・ビットをクリアします。また、外部コピーのシーケンス番号を変更して、ロード・シーケンスの起点を作成します。ただし、ログを取らない書き込みがまったく発生しなければ、シーケンス番号は増えません。

for external ダンプ句を使用してもしなくても、データベースの外部コピーは作成できます。ただし、後続のトランザクション・ダンプをプライマリ・サーバからセカンダリ・サーバに適用するには、次のように、for external dump 句をコマンド内に指定する必要があります。

```
quiesce database tag_name hold db_name [, db_name] ... [for
external dump]
```

次に例を示します。

```
quiesce database pubs_tag hold pubs2 for external dump
```

プライマリ・データベースが前回起動されてからデータベースのマッピングが変更されていない場合、1 つのデータベースの外部ダンプを作成する手順は次のようになります。

1 次のコマンドを発行します。

```
quiesce database pubs_tag hold pubs2 for external dump
```

2 システムに適した方法を使用して、データベースの外部コピーを作成します。

3 次のコマンドを発行します。

```
quiesce database pubs_tag release
```

**警告！** ステータス・ビットがクリアされ、シーケンス番号が更新されるので、quiesce database の発行後に実際に外部コピーを作成するかどうかにかかわらず、トランザクションのダンプの実行が可能になります。Adaptive Server は、quiesce database... hold for external dump が実行されてから quiesce database... release が実行されるまでの間に外部コピーが作成されたかどうかは認識できません。quiesce database hold for external dump コマンドを使用する目的が、新しいダンプ・シーケンスの起点となるコピーを実際に実行することではなく、一時的な書き込み保護であり、アプリケーションによってログを取らない書き込みが行われる場合も、トランザクション・ログ・ダンプの作成は可能ですが、このダンプは使用できません。dump transaction to *archive_device* は正しく実行されますが、load transaction は、シーケンスが正しくないので、そのアーカイブを拒否します。

# *mount* コマンドと *unmount* コマンドの使用

mount コマンドと unmount コマンドを使用すると、データベースの移動やコピーを簡単に実行できます。サーバを再起動しないで、Adaptive Server から別の Adaptive Server にデータベースを移動またはコピーできます (データベースをテープまたはディスクにコピーする dump および load database とは対照的です)。mount コマンドと unmount コマンドでは、一度に複数のデータベースの移動またはコピーが可能です。

これらのコマンドを使用すると、デバイスを物理的に移動してデータベースを再度アクティブにすることもできます。

「第 7 章 データベースのマウントとマウント解除」を参照してください。

**警告！** ログイン名への直接マッピングは、Adaptive Server のデータベース内に保持されているのではありません。つまり、移送元の Adaptive Server でデータベースへのアクセスを許可されているログインのそれぞれについて、同じ suid を持つ対応するログインが移送先の Adaptive Server に存在している必要があります。

パーミッションと保護が変わらないようにするには、セカンダリ Adaptive Server 上のログイン・マップを元の Adaptive Server 上のログイン・マップと同一にする必要があります。

# バックアップとリカバリのための Backup Server の使用方法

ダンプとロードは、Adaptive Server と同じマシン上で動作する Open Server プログラムである Backup Server によって実行されます。リモート・コンピュータ上にある Backup Server を使用し、ローカル・コンピュータ上で別の Backup Server を使用することによって、ネットワーク上でバックアップを実行できます。

**注意** Backup Server は、マルチディスク・ボリュームへのダンプは実行できません。

Backup Server は次の作業を行います。

- 「ストライプ・ダンプ」の作成とロード。ダンプ・ストライピングを利用すると、最大 32 個のバックアップ・デバイスを同時に使用できます。これによって、データベースはほぼ均等に分割されて、それぞれの部分が別々のデバイスにバックアップされます。
- 複数テープにまたがっているシングル・ダンプの作成とロード。
- ネットワーク上の、別のマシン上で稼働している Backup Server との間のダンプとロード。
- 複数のデータベースまたはトランザクション・ログを 1 本のテープにダンプします。
- 多数のデータベース・ダンプまたはログ・ダンプが保存されているテープから、1 つのファイルをロードします。
- プラットフォーム固有のテープ処理オプションをサポートします。
- ボリューム処理要求を、**ダンプ**・コマンドまたは**ロード**・コマンドが発行されたセッション、またはそのオペレータ・コンソールに送信します。
- ダンプ・デバイスの物理的な特性を検出して、プロトコル、ブロック・サイズとその他の特性を判別します。

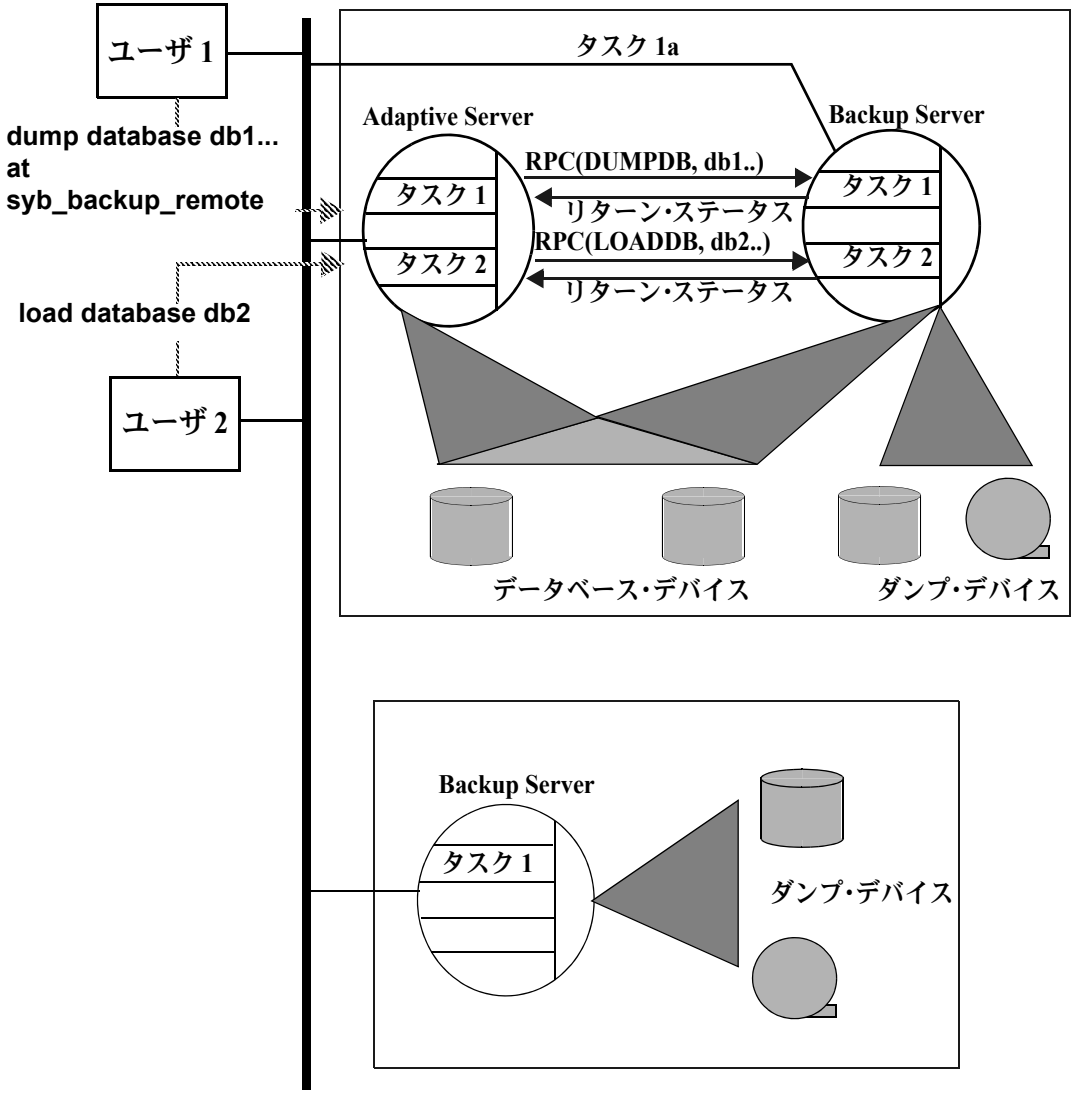
図 12-8 は、2 人のユーザが 2 つのデータベースに対して同時にバックアップを実行する様子を示します。

- ユーザ 1 は、データベース db1 をリモート Backup Server にダンプしています。
- ユーザ 2 は、データベース db2 をローカル Backup Server にロードしています。

ユーザはそれぞれ、Adaptive Server セッションからダンプ・コマンドまたはロード・コマンドを発行します。Adaptive Server はそのコマンドを解釈し、RPC (リモート・プロシージャ・コール) を Backup Server に送信します。この RPC では、どのデータベース・ページをダンプまたはロードするか、どのダンプ・デバイスを使用するか、およびその他のオプションを指定します。

ダンプやロードを実行するとき、Adaptive Server と Backup Server は RPC を使用して命令とステータス・メッセージを交換します。**ダンプ**・コマンドと**ロード**・コマンドに必要なデータ転送は、Adaptive Server ではなく、Backup Server がすべて実行します。

**図 12-8: リモート Backup Server のある Backup Server と Adaptive Server**




ローカル Backup Server は、ユーザ 1 のダンプ命令を受け取ると、指定されたページをデータベース・デバイスから読み取り、リモート Backup Server に送信します。リモート Backup Server は、データをオフライン・メディアに保管します。

同時に、ローカル Backup Server はユーザ 2 のロード・コマンドを実行します。データをローカル・ダンプ・デバイスから読み取り、データベース・デバイスに書き込みます。

## Backup Server との通信

dump コマンドと load コマンドを使用するには、Adaptive Server が Backup Server と通信できることが必要です。そのための要件を以下に示します。

- Backup Server が、Adaptive Server と同じマシン (OpenVMS の場合は同じクラスタ) 上で稼働している必要があります。
- また、Backup Server は master..sysservers テーブルにリストされている必要があります。Backup Server のエントリ SYB_BACKUP は、Adaptive Server のインストール時に sysservers 内に作成されます。この情報を表示するには、sp_helpserver を使用します。
- Backup Server が interfaces ファイルに登録されている。ローカル Backup Server のエントリは、Adaptive Server のインストール時に作成されます。interfaces ファイルに登録されている Backup Server の名前は、master..sysservers 内の SYB_BACKUP エントリのカラム srvnetname の名前と一致している必要があります。リモート Backup Server を別のマシンにインストールした場合は、両方のマシンで共有されるファイル・システム上に interfaces ファイルを作成するか、またはエントリをローカルの interfaces ファイルにコピーします。リモート Backup Server 名は、両方の interfaces ファイルで同じでなければなりません。
- Backup Server プロセスを起動するユーザが、ダンプ・デバイスに対する書き込みパーミッションを持っている必要があります。通常、Adaptive Server と Backup Server を起動するのは "sybase" ユーザですが、このユーザはデータベース・デバイスに対する読み込みと書き込みが可能です。
- リモート・アクセスを行うように Adaptive Server が設定されている。デフォルトでは、Adaptive Server のインストール時に、リモート・アクセスが使用可能になるように設定されます。詳細については、「リモート・アクセスを行うためのサーバの設定」(353 ページ) を参照してください。

## 新しいボリュームのマウント

バックアップとリストアのプロセスの途中で、テープ・ボリュームの交換が必要になる場合があります。Backup Server は、現在マウントされているボリュームで問題を検出すると、クライアントまたはそのオペレータ・コンソールにメッセージを送信して、ボリューム交換を要求します。別のボリュームをマウントしたら、オペレータは Adaptive Server 上で sp_volchanged を実行して Backup Server に通知します。

UNIX システムでは、テープの空きがなくなると、Backup Server がボリュームの交換を要求します。オペレータは、別のテープをマウントして、sp_volchanged を実行します (表 12-2 を参照)。

**表 12-2: UNIX システムでのテープ・ボリュームの変更**

| 手順 | オペレータ、isql を使用 | Adaptive Server | Backup Server |
|---|---|---|---|
| 1 | dump database コマンドを発行する。 | | |
| 2 | | Backup Server にダンプ要求を送信する。 | |
| 3 | | | Adaptive Server からダンプ要求メッセージを受信する。<br>テープ・マウントのメッセージをオペレータに送信する。<br>オペレータからの応答を待つ。 |
| 4 | Backup Server からボリューム変更要求を受信する。<br>テープをマウントする。<br>sp_volchanged を実行する。 | | |
| 5 | | | テープを調べる。<br>テープが使用可能であれば、ダンプを開始する。<br>テープに空き領域がない場合は、ボリューム交換要求をオペレータに送信する。 |
| 6 | Backup Server からボリューム変更要求を受信する。<br>テープをマウントする。<br>sp_volchanged を実行する。 | | |
| 7 | | | ダンプを続行する。<br>ダンプが完了したら、オペレータと Adaptive Server にメッセージを送信する。 |
| 8 | ダンプ完了のメッセージを受信する。<br>テープを取り出してラベルを付ける。 | ダンプ完了のメッセージを受信する。<br>ロックを解除する。<br>dump database コマンドを完了する。 | |

## Backup Server の起動と停止

ほとんどの UNIX システムでは、startserver ユーティリティを使用して、Adaptive Server と同じマシン上で Backup Server を起動します。Windows では、Sybase Central から Backup Server を起動します。Backup Server を起動する方法については、プラットフォームの『Adaptive Server Enterprise 設定ガイド』を参照してください。

Backup Server を停止するには、shutdown を使用します。『システム管理ガイド 第 1 巻』の「第 11 章 システムの問題の診断」、「第 2 章 システム・データベースとオプションのデータベース」と『リファレンス・マニュアル：コマンド』を参照してください。

## リモート・アクセスを行うためのサーバの設定

デフォルトでは、remote access 設定パラメータは、Adaptive Server のインストール時に 1 に設定されます。これによって、Adaptive Server は Backup Server へのリモート・プロシージャ・コールを実行できます。

セキュリティの観点から、ダンプとロードを行っている間以外はリモート・アクセスを禁止する場合もあります。リモート・アクセスを禁止するには、次のコマンドを使用します。

```
sp_configure "allow remote access", 0
```

ダンプまたはロードを実行するには、リモート・アクセスを再度有効にします。

```
sp_configure "allow remote access", 1
```

allow remote access は動的であり、有効にするために Adaptive Server を再起動する必要はありません。allow remote access パラメータを設定できるのは、システム・セキュリティ担当者だけです。

## バックアップ・メディアの選択

ダンプ・デバイスのメディアとして適しているのはテープです。テープを使用すれば、データベースとトランザクション・ログ・ダンプのライブラリをオフラインで保管することができます。大きなデータベースを複数のテープに分けて保管することができます。UNIX システムで Backup Server を使用してダンプとロードを実行するには、ノンリワインディング・テープ・デバイスが必要です。

サポートされているダンプ・デバイスのリストについては、プラットフォームの『Adaptive Server Enterprise 設定ガイド』を参照してください。

### バックアップ・テープの上書きの防止

tape retention in days 設定パラメータは、バックアップ・テープの上書きを禁止する日数を決定します。tape retention in days のデフォルト値は 0 です。これは、すぐにバックアップ・テープに上書きできることを意味します。

tape retention in days の値を変更するには、sp_configure を使用します。新しい値は、次回 Adaptive Server を再起動したときに有効になります。

```
sp_configure "tape retention in days", 14
```

dump database と dump transaction コマンドの両方に retaindays オプションがあり、そのダンプについてはこの値が tape retention in days の値よりも優先されます。

### ファイルまたはディスクへのダンプ

通常は、ファイルまたはディスクをダンプしないようにすることをおすすめします。ファイルが格納されているディスクまたはコンピュータで障害が発生すると、データをまったくリカバリできなくなることもあります。UNIX システムと PC システムでは、master データベース全体を 1 つのボリュームに収める必要があります。これらのシステムで、master データベースのサイズが大きすぎて 1 本のテープ・ボリュームに収まらないときは、sp_volchanged 要求を発行できる別の Adaptive Server がないかぎり、ファイルまたはディスクにダンプするより他に方法はありません。

ファイルまたはディスクに出力したダンプをテープにコピーしてオフラインで保管することができますが、そのテープを Adaptive Server で読み込むには、オンライン・ファイルに再びコピーする必要があります。ディスク・ファイルに出力されてテープにコピーされたダンプを、Backup Server が直接読み込むことはできません。

## ローカル・ダンプ・デバイスの論理デバイス名の作成

ローカル・デバイスにダンプする、またはローカル・デバイスからロードする場合 ( つまり、ネットワークを使用したリモート Backup Server へのバックアップを実行するのではない場合) は、ダンプ・デバイスを指定するときに物理ロケーションと論理デバイス名のどちらも使用できます。論理デバイス名を指定する場合は、master データベース内のシステム・テーブル sysdevices に論理ダンプ・デバイス名を作成できます。

---

**注意** リモート Backup Server にダンプする場合、またはリモート Backup Server からロードする場合は、ダンプ・デバイスの絶対パス名を指定してください。論理デバイス名を使用することはできません。

---

sysdevices テーブルには、各データベース・デバイスとバックアップ・デバイスの情報が保管されています。この中に、*physical_name* (実際のオペレーティング・システムのデバイス名またはファイル名) とその *device_name* (Adaptive Server 内でのみ認識される論理名 ) があります。ほとんどのプラットフォームでは、sysdevices に、テープ・デバイスのエイリアスが 1 つまたは 2 つインストールされています。これらのデバイスの物理名は、プラットフォームの通常のディスク・ドライブ名です。論理名は tapedump1 と tapedump2 です。

バックアップ・スクリプトとスレッショルド・プロシージャを作成するときは、物理デバイス名ではなく論理名を使用してください。また、可能なかぎり、バックアップ・デバイスを交換するたびに、実際のデバイス名を参照しているスクリプトとプロシージャを修正してください。論理デバイス名を使用していれば、デバイスに障害が発生したときも、そのデバイスのエントリを sysdevices から削除して新しいエントリを作成し、その論理名を別の物理デバイスに対応付けるだけで済みます。

**注意** ダンプ・コマンドにはデバイス・ドライバ・オプションを正確に指定してください。Backup Server は、ダンプ・コマンドに指定されたデバイス・ドライバ・オプションを検証しないためです。たとえば、テープを使用する前に強制的に巻き戻すオプションを指定すると、Backup Server は、ダンプの箇所からテープを読み込むのではなく、必ずテープを先頭まで巻き戻します。

## 現在のデバイス名のリスト

システムのバックアップ・デバイスのリストを表示するには、次のコマンドを発行します。

```
select * from master..sysdevices
  where status = 16 or status = 24
```

データベース・デバイスまたはバックアップ・デバイスの物理名と論理名の両方を表示するには、sp_helpdevice システム・プロシージャを使用します。

```
sp_helpdevice tapedump1
device_name physical_name
 description
 status cntrltype vdevno vpn_low      vpn_high
 ------ --------- ------------- -------- -------
tapedump1 /dev/nrmt4
tape, 625 MB, dump device
     16         3             0        0   20000
```

### バックアップ・デバイスの追加

バックアップデバイスを追加するには、sp_addumpdevice システム・プロシージャを使用します。

既存の論理デバイス名を別の物理デバイスに使用するには、sp_dropdevice を使用してデバイスを削除してから、sp_addumpdevice を使用して追加します。次に例を示します。

```
sp_dropdevice tapedump2
sp_addumpdevice "tape", tapedump2, "/dev/nrmt8", 625
```

## ユーザ・データベースのバックアップのスケジューリング

バックアップ・プランを作成するときの主な作業に、データベースをバックアップする頻度の決定があります。バックアップの頻度によって、メディア障害が発生した場合に失われる作業の量が決まります。この項では、ユーザ・データベースとトランザクション・ログをダンプする時期についてのガイドラインを示します。

### 日常のバックアップの計画

ユーザ・データベースの作成直後にダンプを行います。これがベース・ポイントとなります。その後は、一定のスケジュールでダンプします。トランザクション・ログを毎日 1 回バックアップし、データベースを毎週 1 回バックアップするといったペースは極力避けてください。多くの場合は、サイズが大きくアクティブなデータベースのデータベース・ダンプは毎日作成し、トランザクション・ログのダンプは 30 分または 1 時間ごとに作成します。

相互依存のデータベース、つまりデータベース間のトランザクション、トリガ、参照整合性があるデータベースについては、データベース間でのデータ変更が行われない期間に同時にバックアップを作成する必要があります。このデータベースのいずれかに障害が発生して再ロードが必要な場合は、この同時実行のダンプからすべてのデータベースを再ロードする必要があります。

---

**警告！** データベース間の制約を追加、変更、削除した場合や、データベース間の制約が含まれるテーブルを削除した場合は、その直後に必ず両方のデータベースをダンプしてください。

---

## データベースのバックアップを取るその他のタイミング

日常のダンプの他に、ユーザ・データベースのアップグレード、新しいインデックスの作成、ログを取らないオペレーションの実行、dump transaction with no_log および dump transaction with truncate_only コマンドの実行を行うたびにデータベースをダンプしてください。

### アップグレード後のユーザ・データベースのダンプ

ユーザ・データベースを最新バージョンの Adaptive Server にアップグレードした後は、新たにアップグレードしたデータベースをダンプして、最新リリースと互換性のあるダンプを作成してください。アップグレードしたユーザ・データベースに対して dump database を実行しなければ、dump transaction を使用することはできません。

### インデックス作成後のデータベースのダンプ

インデックスをテーブルに追加すると、create index がトランザクション・ログに記録されます。ただし、インデックス・ページに情報が格納されるときは、その変更はログには記録されません。

インデックスの作成後にデータベース・デバイスで障害が発生すると、load transaction でインデックスを再構築するために要する時間が、create index でインデックスを作成するのと同じぐらい長くなることがあります。あまり時間がかからないようにするには、データベースのテーブルのインデックスを作成したらすぐに、そのデータベースをダンプしてください。

### ログを取らないオペレーション後のデータベースのダンプ

次のコマンドのデータはディスクに直接書き込まれ、トランザクション・ログへのエントリの追加は行われません (bcp の場合は、最小限のエントリが追加されます)。

- ログに記録されない writetext
- 永久テーブルでの select into
- トリガやインデックスを持たないテーブルへの高速バルク・コピー (bcp)

これらのコマンドを発行した後にデータベースに加えられる変更は、リカバリできません。これらのコマンドの結果をリカバリできるようにするには、コマンドの実行直前に dump database コマンドを発行します。

### ログがトランケートされた場合のデータベースのダンプ

dump transaction with truncate_only と dump transaction with no_log は、バックアップ・コピーを作成しないでログからトランザクションを削除します。確実にリカバリできるようにするには、ディスクの空き領域不足が発生したためにこれらのコマンドを実行するときに、必ずデータベースをダンプしてください。データベースをダンプしてからでないと、トランザクション・ログはコピーできません。「dump transaction の特別なオプションの使用方法」(331 ページ) を参照してください。

trunc log on chkpt データベース・オプションが true に設定されている場合に、トランザクション・ログのロー数が 50 以上になると、自動チェックポイントの実行時にログが自動的にトランケートされます。この場合、リカバリ性を保証するには、トランザクション・ログではなくデータベース全体をダンプしてください。

## *master* のバックアップのスケジューリング

master データベースのバックアップは、master データベースに影響を与える障害が発生した場合に、リカバリ・プロシージャの一部として使用されます。master の最新のバックアップが存在しない場合は、データベースを再び利用可能な状態にするために、システム・テーブルを作り直さなければならないことがあります。

### 各変更後の *master* のダンプ

master 内でのデータベース・オブジェクトの作成をシステム管理者が制限していても、create login、drop login、alter login、などのシステム・プロシージャを使用すれば、ユーザがデータベース内のシステム・テーブルを変更することができます。このような変更を記録するには、master データベースを頻繁にバックアップしてください。

ディスク、記憶領域、データベース、またはセグメントに影響を与えるコマンドを実行した後は、master データベースをバックアップします。次のコマンドまたはシステム・プロシージャを発行したときは、必ず master をバックアップしてください。

- disk init、sp_addumpdevice、または sp_dropdevice
- ディスク・ミラーリングのコマンド
- セグメントのシステム・プロシージャ sp_addsegment、sp_dropsegment、または sp_extendsegment
- create procedure、drop procedure
- sp_logdevice

- sp_configure
- create database、alter database

## スクリプトとシステム・テーブルの保存

さらに保護を強化するために、disk init、create database、alter database の各コマンドを実行するスクリプトをすべて保存しておき、これらのコマンドを発行するたびに sysdatabases テーブル、sysusages テーブル、sysdevices テーブルのハードコピーを作成してください。

これらのコマンドによって生じる変更内容は、dataserver コマンドの実行時に自動的にリカバリすることはできません。スクリプト (Transact-SQL 文を格納したファイル) を保管していれば、スクリプトを実行することによって変更を再現できます。スクリプトがない場合は、再構築された master データベースに対して個々のコマンドを再発行してください。

syslogins のハードコピーを保管してください。ダンプから master をリカバリするときは、ハードコピーとテーブルの最新バージョンを比較して、ユーザのユーザ ID が変わっていないことを確認してください。

システム・テーブルに対して実行するクエリの詳細については、『システム管理ガイド 第 1 巻』の「第 2 章 システム・データベースとオプションのデータベース」を参照してください。

## *master* データベースのトランザクション・ログのトランケート

master データベースのトランザクション・ログはデータと同じデータベース・デバイス上にあるので、そのトランザクション・ログだけをバックアップすることはできません。また、master データベースのログを移動することはできません。master データベースをバックアップするときは、必ず dump database を使用する必要があります。truncate_only オプションを指定して dump transaction を定期的 (個々のデータベースのダンプ後など) に実行し、master データベースのトランザクション・ログを削除してください。

## ボリュームの変更とリカバリの回避

master データベースをダンプするとき、ダンプ全体が 1 つのボリュームに収まるようにしてください。ただし、Backup Server と通信できる Adaptive Server が 2 つ以上ある場合はこのかぎりではありません。master データベースをロードする前に、Adaptive Server をシングルユーザ・モードで起動する必要があります。この状態では、ロード中に Backup Server から送信されるボリューム交換メッセージに、別のユーザ接続を使用して応答することはできません。通常、master のサイズは小さいため、バックアップは 1 本のテープ・ボリュームに収まります。

# *model* データベースのバックアップのスケジューリング

model データベースの最新のデータベース・ダンプを保持してください。model を変更するたびに、新しいバックアップを作成します。バックアップが保管されていない場合は、model が損傷したときに、model をリストアするために実行済みの変更をすべて実行し直す必要があります。

## *model* データベースのトランザクション・ログのトランケート

model のトランザクション・ログは、master と同様に、データと同じデータベース・デバイスに保管されます。model データベースをバックアップするには、必ず dump database を使用してください。また、データベースのダンプ後に truncate_only を指定して dump transaction を実行し、トランザクション・ログを削除してください。

# *sybsystemprocs* データベースのバックアップのスケジューリング

sybsystemprocs データベースは、システム・プロシージャだけを保管します。このデータベースへの変更を何も加えていない場合は、installmaster スクリプトを実行すればデータベースをリストアできます。

システム・プロシージャに関するパーミッションを変更する場合、または、sybsystemprocs 内に独自のシステム・プロシージャを作成する場合は、リカバリ方法として次の 2 つがあります。

- installmaster を実行した後で、プロシージャを再作成するか、grant コマンドと revoke コマンドを再実行して、すべての変更内容を再入力します。
- sybsystemprocs を変更するたびにバックアップします。

これらのリカバリ・オプションの説明は、「第 14 章 システム・データベースのリストア」を参照してください。

他のシステム・データベースと同様に、sybsystemprocs のトランザクション・ログはデータと同じデバイス上に保管されます。sybsystemprocs をバックアップするには、必ず dump database を使用してください。デフォルトでは、sybsystemprocs の trunc log on chkpt オプションは true (on) に設定されています。したがって、ユーザがトランザクション・ログをトランケートする必要はありません。このデータベース・オプションを変更する場合は、データベースのダンプ時に必ずログをトランケートしてください。

UNIX システムまたは PC を使用中で、Backup Server と通信できる Adaptive Server が 1 つしかない場合は、sybsystemprocs のダンプ全体が必ず 1 つのダンプ・デバイスに収まるようにしてください。ボリューム交換を知らせるには、sp_volchanged が必要ですが、sybsystemprocs のリカバリ中に、サーバ上でこのプロシージャを実行することはできません。

## 同時ロードを行うための Adaptive Server の設定

Adaptive Server は、複数の load コマンドや dump コマンドを同時に実行できます。データベースをロードするには、アクティブなデータベースのロードごとに 1 つの 16K バッファが必要です。デフォルトでは、6 つのロードを同時に実行できるように Adaptive Server が設定されます。同時に実行するロードの数を増やすには、大容量 I/O バッファ数の値を増やします。

```
sp_configure "number of large i/o buffers", 12
```

このパラメータを有効にするには、Adaptive Server を再起動する必要があります。これらのバッファは、dump コマンドや load transaction では使用されません。『システム管理ガイド 第 1 巻』の「第 5 章 設定パラメータ」を参照してください。

## バックアップ統計値の収集

dump database を使用して実際のユーザ・データベースの練習用のバックアップをいくつか行い、dump transaction を使用してトランザクション・ログをバックアップしてください。load database を使用してデータベースをリカバリし、以降のトランザクション・ログ・ダンプを、load transaction を使用してロードしてください。

個々のダンプとロードに要する時間と必要な領域の量に関する統計を記録してください。実際のバックアップ条件に近いほど、練習時の予想が有益なものになります。

バックアップ・プロシージャを作成してテストした後は、その手順を文書化します。無理のないバックアップ・スケジュールを決定し、厳守してください。事前にバックアップ・プロシージャを作成して文書化し、テストを行うことで、障害発生時にデータベースをより効率的にオンライン化することができます。

第 13 章

# ユーザ・データベースのバックアップとリストア

データベース・デバイスに障害が発生した場合にデータベースを損傷から守る唯一の方法は、定期的なバックアップを頻繁に行うことです。

この章では、Backup Server を使用してデータベースをバックアップおよびリストアする方法について説明します。

Tivoli Storage Manager (TSM) がサイトでサポートされている場合、「IBM Tivoli Storage Manager と Backup Server の使用」も参照してください。ただし、この章の構文および使用法のほとんどは、サイトでサポートされている TSM に関連するものです。



dump database、dump transaction、load database、load transaction の各コマンドの構文は、並列構文です。日常のダンプおよびロードを実行するには、データベースの名前と少なくとも 1 つのダンプ・デバイスが必要です。各コマンドにはこれらのオプションを指定できます。

- compression= − ダンプ・ファイルを 1 つのローカル・ファイルに圧縮します。
- at *server_name* − リモート Backup Server を指定します。
- density、blocksize、capacity − テープ記憶装置の特性を指定します。
- dumpvolume − ANSI テープ・ラベルのボリューム識別子を指定します。
- file = *file_name* − ダンプ先またはロード元のファイルの名前を指定します。
- stripe on *stripe_device* − ダンプ・デバイスを追加指定します。
- dismount、unload、init、retaindays − テープ処理オプションを指定します。
- notify − Backup Server のメッセージを、ダンプまたはロードを開始した client に送信するか、operator_console に送信するかを指定します。

**注意** ユーザ・データベースをダンプしても、データベース・オプションはダンプされません。データベース・オプションは master データベースの sysdatabases テーブルに格納されているからです。これは、ダンプ済みのデータベースをそのデータベース自身にロードする場合は問題にはなりません。このデータベースを記述する sysdatabases のローが master に残っているためです。ただし、そのデータベースを削除してから load database を実行する場合や、別のサーバ上でそのデータベースのダンプをロードする場合は、そのデータベースのオプションはリストアされません。ユーザ・データベースのイメージをリストアするには、データベース・オプションも再作成してください。

dump transaction または dump transaction with truncate_only コマンドを実行するのに空き領域が不足している場合は、dump transaction with no_log を使用します。

『リファレンス・マニュアル：コマンド』を参照してください。

**ダンプとロードの完了割合**

dump と load database は、実行中に完了した割合をパーセントで表示します。dump database は、ダンプしたデータベースの完了割合を表示し、load database は、ターゲット・データベースのロード割合をパーセントで表示します。

**注意** dump と load transaction コマンドは、完了した割合を表示しません。

たとえば、sybsystemprocs データベースを *pubs2.dump* という名前のファイルにダンプすると、Adaptive Server は次を表示します。

```
dump database sybsystemprocs to "pubs2.dump"
Backup Server session id is:  13.  Use this value when executing the
```

```
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.

Backup Server: 4.41.1.1: Creating new disk file
/linuxkernel_eng3/Pubs/REL1502/ASE-15_0/bin/pubs2.dump.
Backup Server: 6.28.1.1: Dumpfile name 'pubs20805209785  ' section number 1
mounted on disk file '/linuxkernel_eng3/Pubs/REL1502/ASE-15_0/bin/pubs2.dump'
Backup Server: 4.188.1.1: Database pubs2: 876 kilobytes (46%) DUMPED.
Backup Server: 4.188.1.1: Database pubs2: 1122 kilobytes (100%) DUMPED.
Backup Server: 3.43.1.1: Dump phase number 1 completed.
Backup Server: 3.43.1.1: Dump phase number 2 completed.
Backup Server: 3.43.1.1: Dump phase number 3 completed.
Backup Server: 4.188.1.1: Database pubs2: 1130 kilobytes (100%) DUMPED.
Backup Server: 3.42.1.1: DUMP is complete (database pubs2).
```

*pubs2.dump* をデータベースにロードすると、Adaptive Server は次をレポートします。

```
load database pubs2 from pubs2.dump
Backup Server session id is:  17.  Use this value when executing the
'sp_volchanged' system stored procedure after fulfilling any volume change
request from the Backup Server.
Backup Server: 6.28.1.1: Dumpfile name 'pubs20805209785  ' section number 1
mounted on disk file '/linuxkernel_eng3/Pubs/REL1502/ASE-15_0/bin/pubs2.dump'
Backup Server: 4.188.1.1: Database pubs2: 1880 kilobytes (45%) LOADED.
Backup Server: 4.188.1.1: Database pubs2: 4102 kilobytes (100%) LOADED.
Backup Server: 4.188.1.1: Database pubs2: 4110 kilobytes (100%) LOADED.
Backup Server: 3.42.1.1: LOAD is complete (database pubs2).
Started estimating recovery log boundaries for database 'pubs2'.
Database 'pubs2', checkpoint=(1503, 22), first=(1503, 22), last=(1503, 22).
Completed estimating recovery log boundaries for database 'pubs2'.
Started ANALYSIS pass for database 'pubs2'.
Completed ANALYSIS pass for database 'pubs2'.
Started REDO pass for database 'pubs2'. The total number of log records to
process is 1.
Completed REDO pass for database 'pubs2'.
Use the ONLINE DATABASE command to bring this database online; ASE will not
bring it online automatically.
```

dump database では、表示されるパーセントは、ダンプされたデータベースの総サイズによって予測されます。ただし、load database では、表示されるパーセントは受信するデータベースの総サイズに従って予測されます。たとえば、500MB のデータベース・ダンプを 100 メガバイトのデータベースにロードする場合、完了割合値は 50MB のダンプではなく、100MB のデータベースに従って予測されます。

以下の各項では、ダンプ・コマンドとロード・コマンドで指定する情報とボリューム交換メッセージについて詳しく説明します。

ダンプ・コマンドとロード・コマンドを実行するのに必要なパーミッションについては、「バックアップに対する責任の割り当て」(303 ページ ) を参照してください。

# データベースおよびダンプ・デバイスの指定

すべての dump コマンドおよび load コマンドには、少なくとも、ダンプまたはロードしようとするデータベースの名前を指定してください。トランザクション・ログをトランケートするだけでなく、データのダンプまたはロードを行うコマンドには、ダンプ・デバイスも指定する必要があります。

『リファレンス・マニュアル：コマンド』を参照してください。

## データベース名の指定に関する規則

データベース名は、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。

データベースをダンプからロードする場合は、次のことに注意してください。

- そのデータベースが存在している必要があります。create database の for load オプションを指定してデータベースを作成することも、既存のデータベースにロードすることもできます。データベースをロードする場合は、常に、既存のデータベースの情報がすべて上書きされます。
- ダンプしたデータベース名と同じデータベース名を使用しなくてもかまいません。たとえば、データベース pubs2 をダンプしてから、pubs2_archive という名前の別のデータベースを作成し、その新しいデータベースにダンプをロードできます。

**警告！** 他のデータベースから参照する場合に必要な、プライマリ・キーを含むデータベースの名前は変更しないでください。そのようなデータベースからのダンプをロードして別の名前を付ける必要がある場合は、初めに、他のデータベースからの参照を削除してください。

## ダンプ・デバイスの指定に関する規則

ダンプ・デバイスを指定するには、次の規則に従ってください。

- ダンプ・デバイスは、リテラル、ローカル変数、またはストアド・プロシージャへのパラメータとして指定できます。
- 「null デバイス」(UNIX では */dev/null*、PC プラットフォームには該当しない) に対するダンプやロードは実行できません。
- ローカル・デバイスへダンプする場合やローカル・デバイスからロードする場合は、これら 3 つの方法でダンプ・デバイスを指定できます。
  - 絶対パス名
  - 相対パス名

    - sysdevices システム・テーブルの論理デバイス名

    Backup Server は、Adaptive Server の現在の作業ディレクトリを使用して相対パス名を解決します。

- ネットワーク上でダンプまたはロードを行う場合には、次のことに注意してください。
    - ダンプ・デバイスの絶対パス名を指定してください。相対パス名や、sysdevices システム・テーブルからの論理デバイス名は使用できません。
    - パス名は、Backup Server が稼働しているマシン上で有効なものでなければなりません。
    - 名前に文字、数字、またはアンダースコア (_) 以外の文字が含まれる場合は、引用符で囲んでください。
- with standby_access を使用してトランザクション・ログをダンプする場合は、そのダンプをロードするときに with standby_access を使用する必要があります。

**例**

次の例では、単一のテープ・デバイスを使用してダンプとロードを行います。ダンプとロードに同じデバイスを使う必要はありません。

- UNIX の場合：

```
dump database pubs2 to "/dev/nrmt4"
load database pubs2 from "/dev/nrmt4"
```

- Windows の場合：

```
dump database pubs2 to "\\.\tape0"
load database pubs2 from "\\.\tape0"
```

オペレーティング・システム・ファイルにダンプを出力することもできます。次に Windows での例を示します。

```
dump database pubs2 to "d:\backups\backup1.dat"
load database pubs2 from "d:\backupbackup1.dat"
```

## Backup Server によるテープ・デバイスの決定

dump database コマンドまたは dump transaction コマンドが発行されると、Backup Server は、指定されたダンプ・デバイスのデバイス・タイプが、Adaptive Server で認識されている (内部的に指定される) ものであるかどうかを確認します。認識されているタイプではないデバイスの場合は、テープ設定ファイル (デフォルトのロケーションは *$SYBASE/backup_tape.cfg*) にそのデバイスの設定があるかどうかを調べます。

設定が見つかった場合は、dump コマンドの処理を続行します。

テープ・デバイス設定ファイルに設定が見つからない場合は、このエラー・メッセージが表示されて、dump コマンドは異常終了します。

```
Device not found in configuration file. INIT needs to be
specified to configure the device.
```

そのデバイスを設定するには、init パラメータを指定して dump database コマンドまたは dump transaction コマンドを実行してください。Backup Server は、オペレーティング・システム・コールを行ってデバイスの特性を決定しようとし、特性の決定が正しく行われると、テープ設定ファイルにそのデバイス特性を保存します。

Backup Server がダンプ・デバイスの特性を判断できない場合のデフォルトは、テープあたり 1 つのダンプとなります。その設定でダンプ・ファイルを 1 つ以上書き込むことができない場合は、そのデバイスは使用できません。

Backup Server によるテープ設定は、UNIX プラットフォームでのみ有効です。

## テープ・デバイス設定ファイル

テープ・デバイス設定ファイルには、dump コマンドだけに使用されるテープ・デバイス情報が含まれています。このファイルは、1 行に 1 つのテープ・デバイス・エントリが保存されるようにフォーマットされています。フィールドは、ブランクまたはタブで区切られています。

テープ・デバイス設定ファイルは、Backup Server による書き込みの準備ができたときに作成されます。Backup Server が初めてこのファイルに書き込みを行おうとすると、以下が表示されます。

```
Warning, unable to open device configuration file for
reading. Operating system error. No such file or directory.
```

上記の警告メッセージは無視してください。Backup Server はこの警告を表示してからファイルを作成し、そのファイルに設定情報を書き込みます。

ユーザが設定ファイルを編集する必要があるのは、このエラー・メッセージが表示された場合だけです。

```
Device does not match the current configuration. Please
reconfigure this tape device by removing the configuration
file entry and issuing a dump with the INIT qualifier.
```

上記のエラー・メッセージは、テープ・ハードウェア設定においてデバイス名についての変更があったことを意味します。デバイス名の行エントリを削除し、指示どおりに init 修飾子を指定して、dump コマンドを発行してください。

設定ファイルのデフォルトのパス名は *$SYBASE/backup_tape.cfg* です。これは Sybase のインストール・ユーティリティを使用して、デフォルト・ロケーションを変更できます。

# ダンプの圧縮

dump コマンドには、Backup Server を使用してデータベースとトランザクション・ログを圧縮できる 2 つのオプションがあります。これにより、アーカイブされたデータベースに必要な領域を減らすことができます。パラメータは次のとおりです。

- compression = *compression_level* − リモート・サーバに圧縮します。Backup Server は、独自のネイティブな圧縮方式を使用します。この圧縮オプションをおすすめします。
- compress::[*compression_level*::] − ローカル・ファイルに圧縮します。実行すると、Backup Server は外部フィルタを呼び出します。下位互換性のためにサポートされています。

*compression_level* を 0 ～ 9、100、または 101 の数字で指定します。1 桁の圧縮レベルでは、0 は圧縮なし、9 は圧縮の最高レベルです。圧縮レベル 100 および 101 は、より高速で効率的な圧縮です。100 はより高速の圧縮、101 はより効率的な圧縮を行います。

---

**注意**　compress:: パラメータは、圧縮レベル 100 および 101 をサポートしません。

---

『リファレンス・マニュアル：コマンド』を参照してください。

dump コマンドの compression= パラメータを使用すると、アーカイブされたデータベースに必要な領域を減らすことができます。Adaptive Server 12.5.2 以降で compression= パラメータを使用すると、リモート・マシンにダンプを圧縮できます。

古い形式の compress:: オプションを使用する場合は、データベース・ダンプをロードするときに圧縮レベルを指定する必要はありません。ただし、load with listonly=full を発行すると、ダンプが作成されたときの圧縮レベルを判別できます。

ネイティブ形式の compression= オプションを使用する場合、データベース・ダンプをロードするときに compression= オプションを指定する必要はありません。

たとえば、pubs2 データベースを *compress_file* ファイルにダンプするには、次のように入力します。

```
dump database pubs2 to compress_file...compression=100
```

表 13-1 は、pubs2 データベースに対する圧縮レベルの例を示します。表の中の数値は参考値です。実際に使用するシステムでは、これらの数値が OS のレベルや構成によって異なる可能性があります。

**表 13-1: pub2 の圧縮レベルと圧縮されたファイルのサイズ**

| 圧縮レベル | 圧縮されたファイルのサイズ |
|---|---|
| レベル 1 | 254K |
| レベル 9 | 222K |
| レベル 100 | 324K |
| レベル 101 | 314K |

圧縮レベル 100 および 101 は、レベル 0 ～ 9 よりも CPU 集中が少なく、より高速な圧縮です。ただし、100 および 101 レベルを使用すると、ダンプ・ファイルが大きくなる可能性があります。レベル 100 はより高速な圧縮、レベル 101 はより完全な圧縮を提供します。実際の圧縮結果は、ファイルの内容によって異なります。

パフォーマンスの要件に応じて圧縮レベルのセットを選択することをおすすめします。CPU 集中の少ない圧縮では、圧縮レベル 100 を使用し、アーカイブに必要な領域に基づいてレベル 101 に切り替えます。通常の圧縮では、圧縮レベル 6 を使用し、パフォーマンスの要件に応じて圧縮レベルを調整します。

**例**

**例 1** pubs2 データベースを圧縮レベル 4 で “remotemachine” という名前のリモート・マシンにダンプします。

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
with compression ="4"
```

**例 2** pubs2 データベースを圧縮レベル 100 で remotemachine という名前のリモート・マシンにダンプします。

```
dump database pubs2 to "/Syb_backup/mydb.db" at remotemachine
with compression ="100"
```

dump database と dump transaction の構文の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

## Backup Server のダンプ・ファイルと圧縮ダンプ

この項では、compress:: オプションに関する問題について説明します。これらの問題は、優先される compression= オプションには適用されません。

既存のアーカイブ・ファイルを使用し、テープ・デバイスに対して dump database または dump transaction を実行すると、Backup Server は、自動的に既存のダンプ・アーカイブのヘッダを確認します。このヘッダが判読できない場合は、Backup Server はこのファイルを有効な非アーカイブ・ファイルであると見なし、ダンプ・アーカイブの交換を要求します。

```
Backup Server: 6.52.1.1: OPERATOR: Volume to be overwritten on
'/opt/SYBASE/DUMPS/model.dmp' has unrecognized label data.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
    @session_id = 5,
    @devname = '/opt/SYBASE/DUMPS/model.dmp',
    @action = { 'PROCEED' | 'RETRY' |
'ABORT' },
    @vname = <new_volume_name>
```

したがって、dump database または dump transaction を実行して既存のファイルに出力するときに、compress:: オプションを使用しないで既存の圧縮済みダンプ・アーカイブを指定した場合は、そのアーカイブは圧縮されているため、Backup Server はそのヘッダ情報を認識しません。

**例**

2 番目の dump database はエラーをレポートし、sp_volchanged を実行するためのプロンプトを表示します。

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
go
```

このエラーを防止するには、後続の dump database コマンドや dump transaction コマンドで、with init オプションを指定します。

```
dump database model to 'compress::model.cmp'
go
dump database model to 'model.cmp'
   with init
go
```

## 圧縮ダンプのロード

この項では、compress:: オプションに関する問題について説明します。ネイティブの compression= オプションを指定してダンプを行う場合は、ロードのために固有の構文は必要ありません。

dump ... compress:: を使用してデータベースまたはトランザクション・ログをダンプする場合は、このダンプをロードするときに load ... compress:: オプションを使用してください。

次に、load database ... compress:: と load transaction ... compress:: の構文の一部を示します。

```
load database database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…
load transaction database_name
 from compress::stripe_device
…[stripe on compress::stripe_device]…
```

この構文の *database_name* は、アーカイブ処理されたデータベースです。compress:: を指定すると、アーカイブされたデータベースまたはトランザクション・ログの圧縮解除が行われます。*archive_name* は、ロードするアーカイブ済みデータベースまたはトランザクション・ログへの完全パスです。ダンプ・ファイルの作成時に完全パスを指定していなかった場合は、Adaptive Server が起動されたディレクトリ内にダンプ・ファイルが作成されています。

compress:: オプションを使用する場合は、各ダンプ・デバイスに対する stripe on 句の一部でなければなりません。compression= オプションを使用する場合は、デバイス・リストの後で 1 回指定します。「追加のダンプ・デバイスの指定：stripe on 句」(383 ページ) を参照してください。

**注意** load コマンドでは *compression_level* 変数は使用しないでください。

load database と load transaction の構文の詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

# リモート Backup Server の指定

at *backup_server_name* 句を使用して、ネットワークを介したダンプ要求およびロード要求を、別のマシン上で実行中の Backup Server に送信します。『リファレンス・マニュアル：コマンド』を参照してください。

**注意** compress:: オプションが正しく動作するのは、ローカルのアーカイブの場合のみです。*backup_server_name* オプションは使用できません。

ダンプとロードの要求をネットワークを介して送信するという方法は、複数のテープ・デバイスを持つ 1 台のマシンですべてのバックアップとロードを行う場合に理想的です。すべてのテープ交換要求に対応できるように、そのマシンが設置されている場所にオペレータを配置します。

次の例では、リモート Backup Server である REMOTE_BKP_SERVER との間でダンプおよびロードを行います。

```
dump database pubs2 to "/dev/nrmt0" at REMOTE_BKP_SERVER
load database pubs2 from "/dev/nrmt0" at REMOTE_BKP_SERVER
```

*backup_server_name* は、Adaptive Server が稼働しているマシンの interfaces ファイル内に登録されていなければなりませんが、sysservers テーブル内には登録されていなくてもかまいません。ローカルとリモートの両方の interfaces ファイルで、*backup_server_name* が同じになるようにしてください。

# テープ密度、ブロック・サイズ、容量の指定

通常、Backup Server が使用するデフォルトのテープ密度とブロック・サイズは、そのオペレーティング・システムに最適なものです。これらのデフォルト値を使用することをおすすめします。

テープの密度、ブロック・サイズ、容量は、ダンプ・デバイスごとに指定できます。さらに、すべてのダンプ・デバイスに対する density オプション、blocksize オプション、capacity オプションを with 句内で指定できます。個々のテープ・デバイスに対して指定されている特性は、with 句で指定される特性に優先します。

『リファレンス・マニュアル：コマンド』を参照してください。

## デフォルト密度の上書き

dump コマンドと load コマンドでは、オペレーティング・システムのデフォルトのテープ密度が使用されます。通常、この値がテープのダンプに最も適しています。

density パラメータを指定しても、UNIX と PC プラットフォームでのダンプとロードには効果はありません。

---

**注意**　テープ密度を指定するのは、init テープ処理オプションを使用する場合だけです。「ダンプ前のボリュームの再初期化」(386 ページ ) を参照してください。

---

## デフォルトのブロック・サイズの上書き

blocksize パラメータは、ダンプ・デバイスの I/O 操作 1 回あたりのバイト数を指定します。
デフォルトでは、ダンプ・コマンドおよびロード・コマンドはオペレーティング・システムに最適な値を選択します。可能な限り、このデフォルト値を使用してください。

特定のダンプ・デバイスについて、デフォルト値に優先する値を指定するには、blocksize = *number_bytes* オプションを使用します。ブロック・サイズの大きさは、データベース・ページ 1 ページ分 (2048 バイト ) 以上で、データベース・ページ・サイズの整数倍でなければなりません。

UNIX システムでは、ロード・コマンドで指定したブロック・サイズは無視されます。Backup Server は、ダンプの作成時に使用されたブロック・サイズを使用します。

### 大きなブロック・サイズ値の指定

dump database コマンドまたは dump transaction コマンドを使用してテープにダンプするときに、Backup Server が決定するデバイスのブロック・サイズ最大値よりも大きいブロック・サイズ値を指定した場合は、テープ・ドライブによってはダンプまたはロードが異常終了することがあります。たとえば、HP で、8 mm テープ・ドライブにダンプする場合、オペレーティング・システムのエラー・メッセージは次のようになります。

```
Backup Server: 4.141.2.22: [2] The 'write' call failed for
device 'xxx' with error number 22 (Invalid argument). Refer
to your operating system documentation for further details.
```

*$SYBASE/backup_tape.cfg* 内のテープ・デバイス設定ファイルで指定されているブロック・サイズよりも大きなブロック・サイズは指定しないでください。デバイスのブロック・サイズは、テープ・デバイス設定ファイル内の行の 5 番目のフィールドにあります。

このエラーは、テープの自動設定が実行されるテープ・ドライブ上でのみ発生します。つまり、デバイス・モデルは、Backup Server のコード内にハードコードされているのではありません。

## ダンプ・コマンドでのテープ容量の指定

テープ終了マーカを検出できない UNIX プラットフォームでは、テープにダンプできる容量 (キロバイト数) を指定する必要があります。

ダンプ・デバイスの物理パス名を指定する場合は、capacity = *number_kilobytes* パラメータをダンプ・コマンドに追加してください。論理ダンプ・デバイス名を指定すると、capacity = *number_kilobytes* パラメータを使用して上書きしないかぎり、sysdevices テーブルに保管されている *size* パラメータが使用されます。

指定する容量は、データベース・ページ 5 ページ分 (1 ページにつき 2,048 バイト必要) 以上でなければなりません。使用するデバイスの定格容量を少し下回る容量を指定することをおすすめします。

容量を計算するには、一般に、デバイスの製造者仕様に定められている最大容量の 70 % を使用し、オーバヘッド ( レコード間ギャップ、テープ・マークなど ) の分として 30 % を残しておきます。この規則はほとんどの場合にあてはまりますが、ベンダ間およびデバイス間でオーバヘッドが異なるため、すべての場合にあてはまるわけではありません。

## Backup Server に対するノンリワインディング・テープの機能

ノンリワインディング・テープには、有効なダンプ・データの終端に自動的に位置付ける機能があるので、複数のダンプ操作を行うときに時間を短縮できます。

Backup Server は、各ダンプ操作の終了時にファイル終了ラベル EOF3 を書き込みます。

### テープ操作

新しいダンプが実行されると、Backup Server はスキャンを実行して最後の EOF3 ラベルを探します。関係する情報が保存され、テープは次のファイルの先頭に位置付けられます。この位置より後に、新しいデータが追加されます。

EOF3 ラベルが見つからないか、その他の問題がある場合は、テープを巻き戻して前方をスキャンします。これらの手順の実行中に発生したエラーが原因でダンプ操作がアボートすることはありませんが、エラーが発生すると、Backup Server のデフォルトの動作は「巻き戻し＆スキャン」となります。巻き戻し＆スキャン動作の間もエラーが解消されない場合は、dump コマンドはアボートします。

### ダンプのバージョンの互換性

Backup Server は、テープのラベル・バージョンが 5 以上の場合にのみ、ノンリワインディング・ロジックをアクティブにします。したがって、このロジックをアクティブにするには、dump コマンドの実行時に with init 句を指定する必要があります。ラベル・バージョンが 5 未満のボリュームに対して dump without init を実行すると、ボリュームを交換するよう指示が表示され、次のボリュームでダンプが開始されます。マルチボリューム・ダンプのラベル・バージョンは、ボリューム・セットの途中では変化しません。

表 13-2 は、この動作が可能なラベル・バージョンの定義です。

**表 13-2: ラベル・バージョンの互換性**

| ラベル・バージョン | 使用可 |
|---|---|
| '3' | 不可 |
| '4' | 不可 |
| '5' | 可 |
| '6' | 可 |

# ボリューム名の指定

ボリューム名を指定するには、with dumpvolume = *volume_name* オプションを使用します。dump database コマンドと dump transaction コマンドは、SQL テープ・ラベルにボリューム名を書き込みます。load database コマンドと load transaction コマンドは、このラベルを確認します。ロードされているボリュームが正しくない場合は、Backup Server はエラーメッセージを生成します。

ボリューム名はダンプ・デバイスごとに指定できます。with 句を使用して、すべてのデバイスのボリューム名を指定することもできます。デバイスごとに指定されたボリューム名は、with 句で指定されたボリューム名に優先します。

『リファレンス・マニュアル：コマンド』を参照してください。

## マルチファイル・ボリュームからのロード

複数のダンプ・ファイルが保存されているボリュームからデータベース・ダンプをロードするときは、ダンプ・ファイル名を指定してください。データベース名だけを指定すると、指定したデータベースに最初のダンプ・ファイルがロードされます。たとえば、このコマンドを入力すると、pubs2 からのデータが含まれているかどうかにかかわらず、テープの最初のダンプ・ファイルが pubs2 にロードされます。

```
load database pubs2 from "/dev/rdsk/clt3d0s6"
```

この問題を避けるには、データをダンプまたはロードするときは必ずユニークなダンプ・ファイル名を指定してください。特定のテープのダンプ・ファイルについての情報を得るには、load database コマンドの listonly = full オプションを使用してください。

# ダンプの識別

データベースまたはトランザクション・ログをダンプするときのデフォルトのダンプ・ファイル名は、次の要素を連結して作成されます。

- データベース名の最後の 7 文字
- 西暦年の下 2 桁
- 3 桁の年間通し日付 (1 ～ 366)
- 0 時からの通し秒数 (16 進数)

このデフォルトに優先する値を指定するには、file = *file_name* オプションを使用します。ファイル名は 17 文字以下でなければなりません。また、オペレーティング・システムのファイル命名規則に従ってください。

ファイル名は、ダンプ・デバイスごとに指定できます。with 句を使用してすべてのデバイス用のファイル名を指定することもできます。デバイスごとに指定されたファイル名は、with 句で指定されたファイル名に優先します。

データベースまたはトランザクション・ログをロードするときは、file = *file_name* 句を使用して、複数のダンプが保存されているボリュームからどのダンプをロードするかを指定します。

マルチファイル・ボリュームからダンプをロードするときは、正しいファイル名を指定してください。

```
dump tran publications
    to "/dev/nrmt3"
load tran publications
    from "/dev/nrmt4"
    with file = "cations930590E100"
```

次の例では、ユーザ定義のファイル命名規則を使用します。15 文字のファイル名 *mydb97jul141800* から、ダンプを実行したデータベース (mydb)、日付 (1997 年 7 月 14 日 )、時間 (18:00、つまり午後 6:00) を識別できます。load コマンドを実行すると、ロードの前にテープが *mydb97jul141800* まで先送りされます。

```
dump database mydb
    to "/dev/nrmt3"
    with file = "mydb97jul141800"
load database mydb
    from "/dev/nrmt4"
    with file = "mydb97jul141800"
```

# ダンプまたはロードのパフォーマンスの向上

Backup Server を起動するときに -m パラメータを使用し、Backup Server が使用する共有メモリを増やすことによって、dump コマンドと load コマンドのパフォーマンスを向上させます。-m パラメータには、Backup Server が使用する共有メモリの最大値を指定します。指定した共有メモリ量を Backup Server が確実に使用できるようにするには、オペレーティング・システムでの設定も必要です。ダンプまたはロード操作が完了すると、その共有メモリ・セグメントは解放されます。

**注意** 追加の共有メモリ量を設定することによりダンプやロードのパフォーマンスが向上するのは、ハードウェア設定のパフォーマンス限界を超えない場合に限られます。-m パラメータの値が増加しても、QIC のような低速のテープ・デバイスに対して行うダンプのパフォーマンスはさほど向上しない場合がありますが、DLT のような高速のデバイスに対するダンプのパフォーマンスは大幅に向上します。

## 前バージョンとの互換性

ダンプ・ファイルと Backup Server との互換性に関しては、いくつか注意すべき点があります。表 13-3 は、現在実行しているバージョンと以前のバージョンのローカル Backup Server でロードできるダンプ・ファイルの形式を示します。

**表 13-3: ローカル・オペレーションのサーバ**

| | 新しいダンプ・ファイルの形式 | 古いダンプ・ファイルの形式 |
|---|---|---|
| 現在のバージョンのサーバ | 可 | 可 |
| 以前のバージョンのサーバ | 不可 | 可 |

表 13-4 と 表 13-5 は、現在のバージョンと以前のバージョンのリモート Backup Server でロードできるダンプ・ファイルの形式を示します。リモート Backup Server のシナリオでは、マスタ・サーバとは、データベースおよび Adaptive Server と同じマシン上にある Backup Server を指し、スレーブ・サーバは、アーカイブ・デバイスと同じリモート・マシン上にある Backup Server を指します。

表 13-4 に、マスタ・サーバが現在のバージョンの Backup Server である場合に load オペレーションが可能かどうかを示します。

**表 13-4: 新しいバージョンのマスタ・サーバ**

| | 新しいダンプ・ファイルの形式 | 古いダンプ・ファイルの形式 |
|---|---|---|
| 新しいバージョンのスレーブ・サーバ | 可 | 可 |
| 古いバージョンのスレーブ・サーバ | 不可 | 可 |

表 13-5 に、マスタ・サーバが以前のバージョンの場合に load オペレーションが可能かどうかを示します。

**表 13-5: 古いバージョンのマスタ・サーバ**

| | 新しいダンプ・ファイルの形式 | 古いダンプ・ファイルの形式 |
|---|---|---|
| 新しいバージョンのスレーブ・サーバ | 不可 | 可 |
| 古いバージョンのスレーブ・サーバ | 不可 | 可 |

## 整数フォーマットで保管されたラベル

バージョン 12.0 以降の Backup Server は、整数形式でストライプ番号を保管します。それより前のバージョンの Backup Server は、ASCII 形式の HDR1 ラベルに 4 バイトのストライプ番号を保管していました。このような旧バージョンの Backup Server は、新しいダンプ・ファイルの形式を使用するダンプ・ファイルはロードできません。ただし、バージョン 12.0 以降の Backup Server は、旧バージョンの形式のダンプを読み込んだり書き込んだりすることができます。

リモート・サーバを使用してダンプまたはロードを実行しているときに次のことが発生すると、エラー・メッセージが表示されてオペレーションがアボートされます。

- リモートの Backup Server の中にバージョンが 12.0 よりも前のものがあり、データベースをダンプまたはロードするストライプの数が 32 を超える場合。
- Backup Server がロード時に読み込むダンプ・ファイルの中に旧バージョンの形式のものがあり、データベースのロード元であるストライプの数が 32 を超える場合。

## システム・リソースの設定

ダンプやロードを実行する前に、ローカル Backup Server とリモートの Backup Server の起動時にコマンド・ラインのオプションでシステム・リソースの値を正しく指定することによって、ローカル Backup Server とリモートの Backup Server を設定する必要があります。コマンド・ライン・オプションの完全なリストについては、『ASE ユーティリティ・ガイド』を参照してください。

システム・リソースが適切に設定されていないと、ダンプまたはロードが異常終了することがあります。たとえば、デフォルト設定で起動されたローカル Backup Server とリモート Backup Server を使用して 26 以上のストライプへのリモート・ダンプを実行すると、このダンプは失敗します。これは、Backup Server が開始できる最大ネットワーク接続数 (-N オプションによって指定される) が 25 であるからです。ただし、デフォルトでは、リモート Backup Server への最大サーバ接続数 (-C オプションによって指定される) は 30 です。

ストライプ数の上限を上げるようにシステムを設定するには、次のオペレーティング・システム・パラメータを設定します。

- 1 つのプロセスに付加できる共有メモリ・セグメント数
- 共有メモリ識別子数
- スワップ領域

上記のパラメータが正しく設定されていない場合は、多数のストライプへのダンプ ( また多数のストライプからのロード ) が開始されたときに、システム・リソース不足のためにオペレーションがアボートすることがあります。Backup Server が共有メモリ・セグメントを作成できないか付加できないため SYBMULTBUF プロセスが終了したことを通知するエラー・メッセージが表示されます。

### 共有メモリ使用量の設定

-m パラメータを使用して Backup Server を起動するための構文は、次のとおりです。

```
backupserver [-m nnn]
```

ここで *nnn* は、Backup Server がすべての dump セッションや load セッションで使用できる共有メモリの最大メガバイト値です。

-m パラメータは、共有メモリ使用量の上限値を設定します。ただし、メモリを追加してもパフォーマンスが向上しないと Backup Server が判断した場合は、指定した値よりも実際に使用される量が少なくなることがあります。

Backup Server は、-m の値をサービス・スレッド数の設定値 (-P パラメータ) で除算することで、各ストライプで使用できる共有メモリのサイズを決定します。

-m のデフォルト値は、サービス・スレッド数に 1MB を乗算した数値です。-P のデフォルト値は 48 なので、共有メモリ使用量のデフォルトでの最大値は 48MB となります。ただし、Backup Server のメモリ使用量がこの値に達するのは、48 のすべてのサービス・スレッドが同時にアクティブになっているときだけです。-P の最大値は、サービス・スレッドの最大数である 12,288 です。『ASE ユーティリティ・ガイド』を参照してください。

Backup Server が使用できるストライプ 1 つあたりの共有メモリ量は、割り付けられているサービス・スレッド数に比例します。サービス・スレッド数の最大値を増やす場合は、ストライプあたりの共有メモリ量が変わらないように、-m の値も増やす必要があります。-m の値を増やさずに -P の値を増やした場合は、ストライプあたりの割り付けられる共有メモリが少なすぎて、ダンプまたはロードを処理できなくなることがあります。

-m の値をどのくらいまで増やすかを判断するには、この式を使用してください。

(-m の MB 値) × 1024/(-P の値)

この式で得られた値が 128 KB 未満の場合は、Backup Server は起動できません。

-m の最小値は 6MB です。-m の最大値は、オペレーティング・システムの共有メモリの制限値によって異なります。

共有メモリの値が大きい Backup Server を使ってダンプを作成し、共有メモリの値がそれより小さい Backup Server を使ってロードを実行しようとすると、Backup Server は使用可能なメモリしか使用せず、ロード・パフォーマンスの低下の原因となります。

ロード時に使用可能なストライプあたりの共有メモリのサイズが、ダンプ時に使用されたブロック・サイズの 2 倍より小さい場合は、Backup Server はエラーメッセージを生成してアボートします。

## 最大ストライプ数の設定

Backup Server が使用できる最大ストライプ数は、Open Server が作成できる Open Server スレッドの最大数によって制限されます。Open Server では、1 つのアプリケーションが使用できるスレッド数は最大 12K という制限があります。

Backup Server は、ストライプごとに 1 つのサービス・スレッドを作成します。したがって、Backup Server がダンプやロードに使用できるローカル・ストライプ数は、最大 12,286 です。

その他の制限事項として、Backup Server はストライプごとに 2 つのファイル記述子を使用します。これは、エラー・ログ・ファイル、interfaces ファイル、その他のシステム・ファイルに対応するファイル記述子とは別に使用されるものです。ただし、スレッドごとのファイル記述子の数は、オペレーティング・システムによって制限されています。Open Server では、アプリケーションが追跡できるファイル記述子の数は 1280 までに制限されています。

Backup Server がダンプを出力できるローカル・ストライプ数のおおよその最大値を求めるには、次の式を使用します。

$$\frac{(\text{OS の制限値と Open Server の制限値のいずれか小さい方}) - 2}{2}$$

Backup Server がダンプを出力できるリモート・ストライプ数のおおよその最大値を求めるには、次の式を使用します。

$$\frac{(\text{OS の制限値と Open Server の制限値のいずれか小さい方}) - 2}{3}$$

ファイル記述子のデフォルト数と最大数の詳細については、使用しているオペレーティング・システムのマニュアルを参照してください。

## 最大ネットワーク接続数の設定

ローカルの Backup Server が起点となるネットワーク接続数は Open Server によって制限され、最大数は 9118 です。したがって、Backup Server が 1 回のダンプまたはロードのオペレーションに使用できる最大リモート・ストライプ数は 9118 となります。

1 つのリモート Backup Server が同時に受け入れ可能なサーバ接続数は最大 4096 です。したがって、リモート Backup Server 1 つに対する最大リモート・ストライプ数も 4096 です。

## 最大サービス・スレッド数の設定

Backup Server の -P パラメータは、Open Server が作成するサービス・スレッド数を設定します。最大サービス・スレッド数は 12,228 です。最小値は 6 です。最大スレッド数は、使用できる最大ストライプ数に等しくなります。Backup Server の起動時に -P で設定された値が小さすぎて、データベースをダンプまたはロードするストライプの数がスレッド数を超える場合は、そのダンプまたはロードは失敗します。

# 追加のダンプ・デバイスの指定：*stripe on* 句

**Striping** を利用すると、単一のダンプ・コマンドまたはロード・コマンドで複数のダンプ・デバイスを使用できます。デバイスごとに別の stripe on 句を使用して、名前 (および必要であれば特性) を指定する必要があります。

1 つのダンプ・コマンドまたはロード・コマンドに、複数の stripe on 句を指定できます。

『リファレンス・マニュアル：コマンド』を参照してください。

## 複数デバイスへのダンプ

Backup Server はデータベースをほぼ等分し、各部分を異なるデバイスに送信します。ダンプはすべてのデバイスで同時に実行されるため、1 つのデータベースまたはトランザクション・ログのダンプに必要な時間を短縮できます。それぞれのテープはデータベースの一部分しか格納しないため、特定のデバイスにテープを新しくマウントする必要はありません。

---

**警告！** master データベースを複数のテープ・デバイスにダンプしないでください。master データベースをテープなどのリムーバブル・メディアからロードする場合は、ボリューム交換メッセージに応答できる別の Adaptive Server がなければ、ボリュームを交換できません。

---

## 複数のデバイスからのロード

データベースまたはトランザクション・ログのロードに、複数のデバイスを使用できます。複数デバイスを使用すると、ロードに必要な時間が短くなり、特定のデバイスに複数のテープをマウントする必要性も少なくなります。

## ロードに使用するデバイス数がダンプ時よりも少ない場合

ダンプ・デバイスのいずれかがダンプとロードの間に使用できなくなった場合でも、データベースまたはログのロードは実行できます。ダンプ・コマンドで指定した数より少ない stripe 句をロード・コマンドに指定してください。

---

**注意**　ネットワーク上でダンプおよびロードを実行する場合には、両方のオペレーションで同じドライブ数を使用してください。

---

データベースのダンプに 3 つのデバイスを使用し、ロードには 2 つだけ使用する例を示します。

```
dump database pubs2 to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    stripe on "/dev/nrmt2"
load database pubs2 from "/dev/nrmt0"
    stripe on "/dev/nrmt1"
```

最初の 2 本のテープがロードされると、3 番目のテープのロードを要求するメッセージがオペレータに通知されます。

複数のオペレーティング・システム・ファイルに、データベースをダンプすることもできます。Windows での例を示します。

```
dump database pubs2 to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
load database pubs2 from "/dev/nrmt0"
stripe on "d:\backups\backup2.dat"
    stripe on "d:\backups\backup3.dat"
```

## 各デバイスの特性の指定

リモート Backup Server に接続したストライプ・デバイスごとに、別の at *server_name* 句を使用してください。リモート Backup Server 名を指定しない場合、ローカル Backup Server はローカル・マシン上でダンプ・デバイスを探します。必要であれば、ストライプ・デバイスごとに別のテープ・デバイス特性 (density、blocksize、capacity、dumpvolume、file) を指定することもできます。

UNIX での例では、3 つのダンプ・デバイスを使用します。これらのデバイスは、リモート Backup Server である REMOTE_BKP_SERVER に接続されています。

```
dump database pubs2
    to "/dev/nrmt0" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt1" at REMOTE_BKP_SERVER
    stripe on "/dev/nrmt2" at REMOTE_BKP_SERVER
```

# テープ処理オプション

with 句で指定するテープ処理オプションは、そのダンプまたはロードに使用するすべてのデバイスに適用され、次を含みます。

- nodismount：そのテープを次のダンプやロードに使用できるようにします。
- unload：ダンプまたはロード後にテープを巻き戻し、アンロードします。
- retaindays：ファイルが上書きされないように保護します。
- init：最後のテープ終了マークの後にダンプファイルを追加するのではなく、テープを再初期化します。

『リファレンス・マニュアル：コマンド』を参照してください。

## テープのマウント解除の指定

論理マウント解除をサポートするプラットフォームでは、ダンプまたはロードが完了するとテープのマウントが解除されます。テープをマウントしたままにして次のダンプまたはロードに使用できるようにするには、nodismount オプションを指定してください。このコマンドは、UNIX システムや PC システムでは効果はありません。

## テープの巻き戻し

デフォルトでは、ダンプ・コマンドとロードコマンドはどちらも nounload テープ処理オプションを使用します。

UNIX システムでは、このオプションが指定されている場合はダンプやロードが完了してもテープは巻き戻されません。このため、同じボリュームへのデータベースやトランザクション・ログのダンプ、または同じボリュームからのデータベースやトランザクション・ログのロードを続けて実行できます。コマンドが完了した時点でテープを巻き戻してアンロードするには、そのテープへの最後のダンプを行うときに unload オプションを使用してください。

## ダンプ・ファイルの上書き防止

tape retention in days 設定パラメータは、テープ・ファイルの作成後、もう一度ダンプを実行してそのテープ・ファイルを上書きできるようになるまでの日数を指定します。sp_configure を使用して設定できるこのサーバワイドなオプションは、単一の Adaptive Server からのすべてのダンプ要求に適用されます。

単一のデータベースまたはトランザクション・ログのダンプで、この tape retention in days パラメータに優先する値を指定するには、retaindays = *number_days* オプションを使用します。日数は正の整数でなければなりません。テープをただちに上書きする場合は、ゼロを指定してください。

---

**注意** tape retention in days および retaindays は、ディスク、1/4 インチ・カートリッジ、および単一ファイル・メディアにだけ有効です。マルチファイル・メディアでは、Backup Server は最初のファイルの有効期限だけを確認します。

---

## ダンプ前のボリュームの再初期化

デフォルトでは、各ダンプはテープの最後のテープ終了マークの後に追加されていきます。テープ・ボリュームは再初期化されません。これによって、単一のボリュームに複数のデータベースをダンプできます新しいダンプは、マルチボリューム・ダンプの最後のボリュームの末尾にしか追加できません。

既存のテープの内容を上書きするには、init オプションを使用してください。init を指定すると、Backup Server は以下のものがあるかどうかを確認しないで、テープを再初期化します。

- ANSI アクセス制限
- 有効期限の切れていないファイル
- Sybase 以外のデータ

デフォルトの noinit を指定した場合は、この 3 つの条件がすべて確認され、いずれかに該当する場合はボリューム交換プロンプトが送信されます。

次の例では、2 つのデバイスを初期化し、既存の内容を新しいトランザクション・ログのダンプで上書きします。

```
dump transaction pubs2
    to "/dev/nrmt0"
    stripe on "/dev/nrmt1"
    with init
```

データベースをオペレーティング・システム・ファイルにダンプするときは、init オプションを使用して既存のファイルを上書きすることもできます。Windows での例を示します。

```
dump transaction pubs2
    to "d:\backups\backup1.dat"
    stripe on "d:\backups\backup2.dat"
    with init
```

単一ボリュームへの複数のデータベースのダンプの詳細については、『リファレンス・マニュアル：コマンド』を参照してください。

# パスワード保護を使用したデータベースのダンプとロード

dump database コマンドの password パラメータを使用して、権限を持たないユーザがデータベース・ダンプをロードできないように保護します。データベースをロードするときにこのパスワードも指定する必要があります。

『リファレンス・マニュアル：コマンド』を参照してください。

パスワードの長さは、6 ～ 30 文字にする必要があります。

この例では、パスワード “bluesky” を使用して pubs2 データベースのデータベース・ダンプを保護します。

```
dump database pubs2 to "/Syb_backup/mydb.db" with passwd = "bluesky"
```

この例では、同じパスワードを使用してデータベース・ダンプをロードします。

```
load database pubs2 from "/Syb_backup/mydb.db" with passwd = "bluesky"
```

パスワード保護に対応する dump コマンドと load コマンドを使用できるのは、Adaptive Server バージョン 12.5.2 以降のみです。Adaptive Server バージョン 12.5.2 のダンプに password パラメータを使用した場合、そのダンプを以前のバージョンの Adaptive Server にロードしようとすると失敗します。

ダンプをロードできるサーバは同じ文字セットを使用しているサーバのみです。たとえば、ASCII 文字セットを使用するサーバから ASCII 以外の文字セットを使用するサーバにダンプをロードしようとすると、ASCII のパスワードの値は ASCII ではないパスワードと異なるためロードが失敗します。

ユーザが入力したパスワードは、Adaptive Server のローカル文字セットに変換されます。ASCII 文字は通常は文字セット間で値の表現が同じであるため、ユーザのパスワードが ASCII 文字セットであれば、dump と load のパスワードはすべての文字セットで認識されます。

# デフォルトのメッセージ送信先の変更

BackupServer のメッセージは、オペレータに、テープ・ボリュームを交換するタイミングおよびダンプまたはロードの進行状況を通知します。これらのメッセージのデフォルト送信先は、オペレーティング・システムにオペレータ端末機能があるかどうかによって決まります。

with 句内で notify オプションを指定すると、ダンプまたはロードのデフォルトのメッセージ送信先を変更できます。このオプションを有効にするには、Backup Server が起動された端末セッションまたはログイン・セッションが、Backup Server が稼働している間はアクティブになっている必要があります。アクティブでないと、sp_volchanged メッセージは失われます。

オペレータ端末機能を持つオペレーティング・システムでは、ボリューム交換メッセージは Backup Server が実行されているマシン上のオペレータの端末に送信されます。他の Backup Server のメッセージを、dump または load 要求が開始されたターミナル・セッションに送信するには、notify = client を使用してください。

オペレータ端末機能が備わっていない UNIX などのシステムでは、ダンプ要求またはロード要求を開始したクライアントにメッセージが送信されます。リモート Backup Server が起動された端末へメッセージを送信するには、notify = operator_console を使用してください。

## *with standby_access* を使用してデータベースをオンラインにする

with standby_access を指定して dump transaction を実行すると、完了したトランザクションだけがダンプされます。このとき、トランザクション・ログは、アクティブなトランザクションがなくなる箇所までダンプされます。with standby_access を使用しない場合は、すべてのオープン・トランザクションのレコードを含むトランザクション・ログ全体がダンプされます。

**図 13-1: standby_access を使用したトランザクション・ダンプ時のダンプのカットオフ・ポイント**

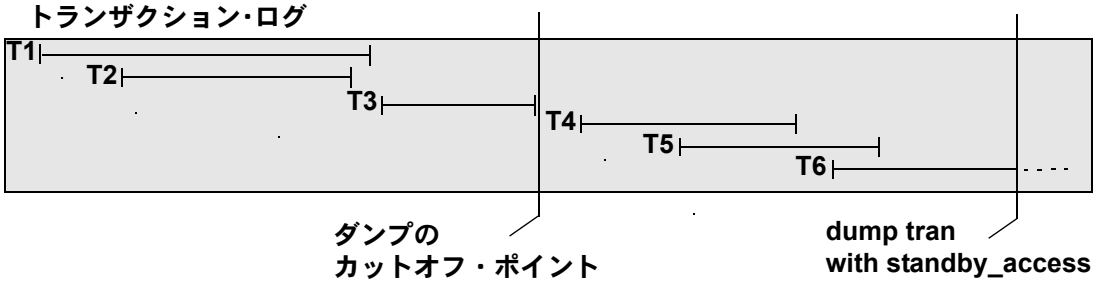



図 13-1 で、dump transaction...with standby_access コマンドが発行される時点では、トランザクション T1 から T5 までが完了し、トランザクション T6 はまだオープンな状態です。T6 がまだオープンであるので、T5 はこのダンプに含まれません。また、T5 がまだオープンであるので T4 もダンプに含まれません。ダンプは T3 (ダンプのカットオフ・ポイントの直前) の最後で停止し、このダンプには完了したトランザクション T1 から T3 までが含まれます。

## *with standby_access* の使用時期の決定

複数のトランザクション・ログを順番にロードするとき、ロードとロードの間にデータベースをオンラインの状態にする場合は、dump tran[saction]...with standby_access when を使用します。たとえば、プライマリ・データベースからのトランザクション・ダンプをロードしてデータを取得する読み込み専用データベースの場合です。プライマリ・データベース内のトランザクションに基づく日次レポートを生成するのに読み込み専用のデータベースが使用され、プライマリ・データベースのトランザクション・ログが一日の終わりにダンプされるとすると、毎日の操作サイクルは次のようになります。

1　プライマリ・データベースで実行：dump tran[saction]...with standby_access

2　読み込み専用データベースで実行：load tran[saction]...

3　読み込み専用データベースで実行：online database for standby_access

**警告！** トランザクション・ログにオープン・トランザクションが含まれているときに、with standby_access を使用しないでそのトランザクション・ログのダンプを行うと、そのログをロードしてデータベースをオンライン化し、その後で以降のトランザクション・ダンプをロードすることはできなくなります。一連のトランザクション・ダンプをロードする場合は、with standby_access が指定されたダンプをロードした後か、一連のダンプすべてをロードした後にのみデータベースをオンラインにできます。

## *with standby_access* を使用してデータベースをオンラインにする

online database コマンドには、with standby_access オプションもあります。with standby_access オプションを使用して作成されたダンプをロードした後にデータベースをオンライン状態にするには、for standby_access を使用してください。

**警告！** with standby_access オプションを指定せずにダンプしたトランザクション・ログを使用して online database for standby_access を実行しようとすると、このコマンドは失敗します。

『リファレンス・マニュアル：コマンド』を参照してください。

# ダンプ・ファイルに関する情報の取得

テープの内容がわからない場合は、ロード・コマンドの with headeronly オプションまたは with listonly オプションを使用して情報を表示します。

**注意** with headeronly と with listonly のどちらの場合も、レポートの表示後にダンプ・ファイルがロードされることはありません。

## ダンプ・ヘッダ情報の要求

with headeronly は、1 つのファイルのヘッダ情報を返します。ファイル名を指定しなければ、with headeronly はテープ上の最初のファイルについての情報を返します。

ヘッダは、データベース・ダンプとトランザクション・ログ・ダンプの区別、データベース ID、ファイル名、ダンプの作成日を示します。データベース・ダンプの場合は、ヘッダは、文字セット、ソート順、ページ数、次のオブジェクト ID も示します。トランザクション・ログのダンプの場合は、ログの中のチェックポイントの位置、最も古い begin transaction レコードの位置、および新旧のシーケンス日付を示します。

この例では、テープ上の最初のファイルのヘッダ情報が返され、次に *mydb9229510945* ファイルのヘッダ情報が返されます。

```
load database mydb
    from "/dev/nrmt4"
    with headeronly
load database mydb
    from "/dev/nrmt4"
    with headeronly, file = "mydb9229510945"
```

次は、headeronly の出力の例です。

```
Backup Server session id is: 44. Use this value when executing the 'sp_volchanged' system
stored procedure after fulfilling any volume change request from the Backup Server.
Backup Server: 4.28.1.1: Dumpfile name 'mydb9232610BC8 ' section number 0001 mounted on
device 'backup/SQL_SERVER/mydb.db.dump'
This is a database dump of database ID 5 from Nov 21 1992 7:02PM.
Database contains 1536 pages; checkpoint RID=(Rid pageid = 0x404; row num = 0xa); next
object ID=3031; sort order ID=50, status=0; charset ID=1.
```

## データベース、デバイス、ファイル名、日付の確認

with listonly は、ボリューム上の各ダンプ・ファイルについての簡単な説明を返します。with listonly = full を指定すると、詳細が表示されます。どちらのレポートも、SQL テープ・ラベルでソートされています。

load database コマンドの with listonly の出力例

```
Backup Server: 4.36.1.1: Device '/dev/nrst0':
File name: 'model9320715138  '
Create date & time: Monday, Jul 26, 2007, 23:58:48
Expiration date & time:  Monday, Jul 26, 2007, 00:00:00
Database name: 'model                          '
```

次に with listonly = full コマンドからの出力例を示します。

```
Backup Server: 4.37.1.1: Device '/dev/nrst0':
Label id: 'HDR1'
File name:'model9320715138  '
Stripe count:0001
Device typecount:01
Archive volume number:0001
Stripe position:0000
Generation number:0001
Generation version:00
Create date & time:Monday, Jul 26, 2007, 23:58:48
Expiration date & time:Monday, Jul 26, 2007, 00:00:00
Access code:' '
File block count:000000
Sybase id string:
'Sybase  'Reserved:'       '
Backup Server: 4.38.1.1: Device '/dev/nrst0':
Label id:'HDR2'
Record format:'F'
Max. bytes/block:55296
Record length:02048
Backup format version:01
Reserved:'    '
Database name:'model                          '
Buffer offset length:00
Reserved:'                              '
```

ボリューム上のすべてのファイルのリストが表示された後に、Backup Server からのボリューム交換要求が送信されます。

```
Backup Server: 6.30.1.2: Device /dev/nrst0: Volume cataloguing
complete.
Backup Server: 6.51.1.1: OPERATOR: Mount the next volume to search.
Backup Server: 6.78.1.1: EXECUTE sp_volchanged
@session_id = 5,
          @devname = '/dev/nrst0',
          @action = { 'PROCEED' | 'RETRY' | 'ABORT' },
          @fname = '
```

オペレータは、sp_volchanged を使用して、別のボリュームをマウントしてボリューム交換を通知することも、すべてのストライプ・デバイスの検索操作を終了することもできます。

# デバイス障害が発生した後のログのコピー

通常、dump transaction は、ログをコピーした後、そのログのアクティブでない部分をトランケートします。トランケートせずにログをコピーするには、with no_truncate オプションを使用してください。

no_truncate オプションを指定すると、データを保持するデバイスに障害が発生した後でもトランザクション・ログをコピーできます。このオプションでは、システム・テーブル sysdatabases および sysindexes 内のポインタを使用して、トランザクション・ログの物理的なロケーションを判断します。no_truncate が使用できるのは、別のセグメントにトランザクション・ログがあり、master データベースにアクセス可能な場合のみです。

---

**警告！** no_truncate は、メディア障害が発生してデータ・セグメントがアクセスできなくなった場合のみに使用してください。使用中のデータベースには no_truncate を使用しないでください。

---

with no_truncate を使用したログのコピーは、「データベースのリカバリ：実行手順」(401 ページ) の項で説明する最初の手順です。

no_truncate は、ストライプ・ダンプ、テープの初期化、リモート Backup Server とともに使用できます。

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

# ログのトランケート

この項では、ログのトランケートに関する問題について説明します。

## 別のセグメント上にないログのトランケート

データベースのログ・セグメントがデータ・セグメントとは別のデバイス上にない場合、dump transaction を使用したログのコピーや、トランケートはできません。その代わり、以下を実行します。

1. 領域不足が発生しないようにするために、dump transaction の with truncate_only オプションを使用してログをトランケートします。truncate_only オプションはデータをコピーしないので、データベース名だけを指定します。
2. dump database を使用して、そのログを含むデータベース全体をコピーします。

この例では、データ・セグメントとログ・セグメントのデバイスが分かれていないデータベース mydb のダンプを実行し、その後でログをトランケートします。

```
dump database mydb to mydevice
dump transaction mydb with truncate_only
```

## 初期開発環境でのログのトランケート

初期開発環境では、ストアド・プロシージャとトリガの作成、削除、再作成、および整合性制約の検査が実行されるので、トランザクション・ログがすぐにいっぱいになる可能性があります。データのリカバリよりも、データベース・デバイス上に十分な領域を確保する方が重要なこともあります。

with truncate_only を使用すると、バックアップ・コピーを作成しないでトランザクション・ログをトランケートできます。

```
dump transaction database_name with truncate_only
```

dump transaction with truncate_only を実行したときは、日常のログ・ダンプを実行する前に、データベースをダンプしてください。

## 空き領域のないログのトランケート

トランザクション・ログが満杯になると、通常の方法ではダンプできないことがあります。dump transaction または dump transaction with truncate_only を実行したときに、ログの領域不足が原因でコマンドが失敗した場合は、dump transaction の with no_log オプションを使用してください。

```
dump transaction database_name with no_log
```

このオプションは、ダンプ・トランザクション・イベントをログに記録せずに、ログをトランケートします。このオプションはデータをコピーしないので、データベース名だけを指定します。

---

**警告！** dump transaction with no_log は最後の手段として使用し、dump transaction with truncate_only が失敗した後に一度だけ使用するようにしてください。dump transaction with no_log を入力した後もデータのロードを継続すると、ログが完全に満杯になって、以降の dump transaction コマンドが失敗する可能性があります。データベースに追加領域を割り付けるには、alter database コマンドを使用してください。

---

dump tran with no_log の実行はすべて Adaptive Server のエラー・ログに記録されます。成功か失敗かを示すメッセージもエラー・ログに出力されます。no_log は、ダンプ時にエラー・ログ・メッセージを生成する唯一のオプションです。

### *with truncate_only* および *with no_log* を使用する場合のリスク

with truncate_only および with no_log を使用すると、ログの空き領域が極端に少なくなった場合にログをトランケートできます。ただし、これらのオプションを使用した場合は、前回の日常ダンプ以降にコミットされたトランザクションはリカバリできません。

---

**警告！** データをリカバリできるようにするには、できるだけ早い段階で dump database を実行してください。

---

この例では、mydb のトランザクション・ログをトランケートしてから、データベースをダンプします。

```
dump transaction mydb
    with no_log
dump database mydb to ...
```

### 十分なログ領域の用意

dump transaction...with no_log を使用することは、エラーの発生と見なされ、サーバのエラー・ログに記録されます。ログ・セグメントをデータ・セグメントとは別のデバイスに配置するようにデータベースが作成されており、頻繁にトランザクション・ログをダンプするラストチャンス・スレッショルド・プロシージャが作成済みで、ログとデータベースに十分な領域が割り付けられている状態であれば、このオプションを使う必要はありません。

ただし、場合によっては、頻繁にログ・ダンプを行っても依然としてトランザクション・ログが満杯になることがあります。dump transaction コマンドは、ログの先頭から、コミットされていないトランザクション・レコード (最も古いアクティブ・トランザクション) が含まれているページの直前までのすべてのページを削除することによって、ログをトランケートします。このアクティブ・トランザクションがコミットされない状態が長く続くほど、トランザクション・ログ内の空き領域が少なくなります。これは、dump transaction コマンドでは追加ページをトランケートできないためです。

このような状況は、データベース内のテーブルを変更するアプリケーションのトランザクションが非常に長く、そのデータベースのトランザクション・ログの容量が小さい場合に起こります。また、暗黙の begin transaction が連鎖トランザクション・モードを使用する場合や、ユーザがトランザクションを終了させるのを忘れた場合のように、トランザクションが長時間コミットされないままでいる場合にも発生します。syslogshold システム・テーブルを問い合わせることによって、データベース内の最も古いアクティブ・トランザクションを調べることができます。

### *syslogshold* テーブル

syslogshold テーブルは、master データベース内にあります。このテーブル内の各ローは次のいずれかを表します。

- データベース内の最も古いアクティブ・トランザクション
- データベース・ログに対する Replication Server のトランケーション・ポイント

データベースに対応するローが syslogshold 内に存在しないこともあります。あるいは、上記のいずれかを表す 1 つのローが存在するか、上記の両方を表す 2 つのローが存在します。Replication Server のトランケーション・ポイントがデータベースのトランザクション・ログに与える影響については、Replication Server のマニュアルを参照してください。

syslogshold を問い合わせることによって、それぞれのデータベース内の現在の状況のスナップショットを得ることができます。ほとんどのトランザクションは存続時間が非常に短いので、問い合わせを実行するたびに結果が異なる場合もあります。たとえば、syslogshold の最初のローに記述されている最も古いアクティブ・トランザクションは、syslogshold への問い合わせが完了する前に終了してしまうことがあります。ただし、syslogshold へのクエリを、時間をかけて何度も実行したときに、特定のデータベースに対して同じローが返される場合は、そのトランザクションが存在するために、dump transaction によるログ領域のトランケートができない状態にある可能性があります。

トランザクション・ログがラストチャンス・スレッショルドに達したときに、dump transaction によってログ内の領域を解放できない場合は、syslogshold と sysindexes を問い合わせることによって、トランケートを妨げているトランザクションを調べることができます。次に例を示します。

```
select H.spid, H.name
from master..syslogshold H, threshdb..sysindexes I
where H.dbid = db_id("threshdb")
and I.id = 8
and H.page = I.first

spid    name
------  -------------------------------------
     8  $user_transaction

(1 row affected)
```

このクエリは、threshdb データベース内の syslogs に対応するオブジェクト ID (8) を使用して、そのトランザクション・ログの最初のページに一致する syslogshold 内の最も古いアクティブ・トランザクションの最初のページを検索します。

master データベース内の syslogshold と sysprocesses を問い合わせて、最も古いアクティブ・トランザクションを所有しているホストとアプリケーションを特定することもできます。次に例を示します。

```
select P.hostname, P.hostprocess, P.program_name,
   H.name, H.starttime
from sysprocesses P, syslogshold H
where P.spid = H.spid
and H.spid != 0
```

```
hostname hostprocess program_name name                starttime
-------- ----------- ------------ ------------------ ------------------
eagle          15826 isql         $user_transaction  Sep  6 1997 4:29PM
hawk           15859 isql         $user_transaction  Sep  6 1997 5:00PM
condor         15866 isql         $user_transaction  Sep  6 1997 5:08PM

(3 rows affected)
```

この情報を使用して、最も古いアクティブ・トランザクションを所有しているユーザ・プロセスを通知するか強制終了して、dump transaction 処理を続行します。データベースのスレッショルド・プロシージャに、上記のタイプのクエリを自動警報メカニズムとして組み込むこともできます。たとえば、トランザクション・ログが絶対にラストチャンス・スレッショルドに達しないようにするという目標を立てたとします。ラストチャンス・スレッショルドに達してしまった場合は、ラストチャンス・スレッショルド・プロシージャ (sp_thresholdaction) からの警告に、トランザクション・ダンプを妨げている最も古いアクティブ・トランザクションについての情報が表示されます。

---

**注意** トランザクションの最初のログ・レコードは、ユーザのログ・キャッシュ内に存在する場合がありますが、そのレコードがログにフラッシュされる (たとえばチェックポイントの後) までは、syslogshold には反映されません。

---

詳細については、『リファレンス・マニュアル：テーブル』および「第 17 章 スレッショルドによる空き領域の管理」を参照してください。

# ボリューム交換要求への応答

UNIX システムおよび PC システムでは、sp_volchanged システム・プロシージャを使用して、正しいボリュームをマウントしたことを Backup Server に通知します。

sp_volchanged を使用するには、ボリューム交換要求を発行した Backup Server と、ダンプまたはロードを開始した Adaptive Server の両方と通信できる任意の Adaptive Server にログインしてください。

既存のダンプへの上書き、新しいテープへのダンプ、または内容が認識できないテープへのダンプを行う場合、Backup Server は ANSI テープ・ラベルに *vname* を書き込みます。ロードを実行するとき、Backup Server は *vname* を使用して、正しいテープがマウントされていることを確認します。*vname* が指定されていない場合は、ダンプ・コマンドまたはロード・コマンドで指定されたボリューム名を使用します。sp_volchanged とコマンドのどちらにもボリューム名が指定されていない場合は、ANSI テープ・ラベルのこのフィールドは確認されません。

## ダンプのボリューム交換プロンプト

データベースまたはトランザクション・ログのダンプ中に、ボリューム交換プロンプトが表示されます。プロンプトには、オペレータがとることができるアクション、および sp_volchanged による応答方法が表示されます。

- `Mount the next volume to search.`

  既存のボリュームにダンプを追加するときに、ファイルの終了マークが見つからない場合は、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged` *`session_id`*`,` *`devname`*`,`<br>`abort` |
| 新しいボリュームのマウントおよびダンプの続行 | `sp_volchanged` *`session_id`*`,` *`devname`*`,`<br>`proceed`<br>`[,` *`fname`* `[,` *`vname`*`]]` |

- `Mount the next volume to write.`

  テープの終わりに到達すると、このメッセージが発行されます。これは、Backup Server がテープの終了マークを検出した場合や、ダンプの量が dump コマンドの capacity パラメータによって指定されたキロバイト数に達した場合、または sysdevices システム・テーブルで指定されたそのデバイスの *high* の値に達した場合に発生します。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged session_id, devname, abort` |
| 新しいボリュームのマウントおよびダンプの続行 | `sp_volchanged session_id, devname, proceed[, fname[, vname]]` |

- `Volume on device devname has restricted access (code access_code).`

  init オプションを指定してダンプを実行すると、既存の内容は上書きされます。ANSI アクセス制限を持つテープに init オプションを指定しないでダンプしようとすると、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged session_id, devname, abort` |
| 別のボリュームのマウントおよびダンプの再試行 | `sp_volchanged session_id, devname, retry [, fname[, vname]]` |
| ダンプの続行および既存の内容の上書き | `sp_volchanged session_id, devname, proceed[, fname[, vname]]` |

- `Volume on device devname is expired and will be overwritten.`

  init オプションを指定してダンプを実行すると、既存の内容は上書きされます。init オプションを指定しないで単一ファイル・メディアにダンプを行うとき、テープに有効期限切れのダンプが存在する場合は、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged session_id, devname, abort` |
| 別のボリュームのマウントおよびダンプの再試行 | `sp_volchanged session_id, session_id, retry[, session_id[, session_id]]` |
| ダンプの続行および既存の内容の上書き | `sp_volchanged session_id, session_id, proceed[, session_id[, session_id]]` |

- `Volume to be overwritten on 'devname' has not expired: creation date on this volume is creation_date, expiration date is expiration_date.`

  単一ファイル・メディアでは、init オプションを指定した場合を除いて、既存のダンプの有効期限が検査されます。ダンプが有効期限内の場合に、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged session_id, session_id, abort` |
| 別のボリュームのマウントおよびダンプの再試行 | `sp_volchanged session_id, session_id, retry[, session_id[, session_id]]` |
| ダンプの続行および既存の内容の上書き | `sp_volchanged session_id, session_id, proceed[, session_id[, session_id]]` |

- `Volume to be overwritten on 'devname' has unrecognized label data.`

  init オプションを指定してダンプを実行すると、既存の内容は上書きされます。init オプションを指定せずに、新しいテープまたは Sybase 以外のデータがあるテープにダンプしようとすると、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ダンプのアボート | `sp_volchanged session_id, session_id, abort` |
| 別のボリュームのマウントおよびダンプの再試行 | `sp_volchanged session_id, session_id, retry[, session_id[, session_id]]` |
| ダンプの続行および既存の内容の上書き | `sp_volchanged session_id, session_id, proceed[, session_id[, session_id]]` |

## ロード中のボリューム交換プロンプト

次に示すのは、ロード中のボリューム交換プロンプトおよびオペレータによって実行可能な処理です。

- `Dumpfile 'fname' section vname found instead of 'fname' section vname.`

  単一ファイル・メディアで指定のファイルが見つからない場合に、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ロードのアボート | `sp_volchanged` *session_id*, *session_id*, `abort` |
| 別のボリュームのマウントおよびロードの再試行 | `sp_volchanged` *session_id*, *session_id*, `retry[,` *session_id*`[,` *session_id*`]]` |
| 現在マウントされているボリュームのファイルのロード (指定されたファイルではなくてもロードするが、望ましい処理ではない) | `sp_volchanged` *session_id*, *session_id*, `proceed[,` *session_id*`[,` *session_id*`]]` |

- `Mount the next volume to read.`

  マルチボリューム・ダンプからダンプ・ファイルの次のセクションを読み込む準備が整うと、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ロードのアボート | `sp_volchanged` *session_id*, *session_id*, `abort` |
| 次のボリュームのマウントおよびダンプの再試行 | `sp_volchanged` *session_id*, *session_id*, `proceed[,` *session_id*`[,` *session_id*`]]` |

- `Mount the next volume to search.`

  マルチファイル・メディアで指定のファイルが見つからない場合に、このメッセージが発行されます。

| オペレータにできる処理 | 応答 |
|---|---|
| ロードのアボート | `sp_volchanged` *session_id*, *session_id*, `abort` |
| 別のボリュームのマウントおよびロードの続行 | `sp_volchanged` *session_id*, *session_id*, `proceed[,` *session_id*`[,` *session_id*`]]` |

# データベースのリカバリ：実行手順

メディア障害の症状はその原因と同じくらいさまざまです。ディスク上の不良ブロックが 1 つだけならば、dbcc コマンドを頻繁に実行しない限り、データベースは不良ブロックが発生した後でもしばらくは正常に機能しているように見えます。ディスク全体またはディスク・コントローラが不良の場合、Adaptive Server はそのデータベースを疑わしい (suspect) としてマーク付けし、警告メッセージを表示します。master データベースを保管しているディスクに障害が発生すると、ユーザはサーバにログインできなくなり、既にログインしているユーザは、master システム・テーブルにアクセスするアクションを実行できなくなります。

データベース・デバイスに障害が発生した場合は、次の手順に従ってください。

1　そのデバイス上の全データベースの最新のログ・ダンプを取得します。

2　そのデバイス上の全データベースの領域の使用状況を調べます。

3　そのデバイスの全データベースについてこの情報を取得したら、各データベースを削除します。

4　sp_dropdevice を使用して、障害が発生したデバイスを削除します。『リファレンス・マニュアル：プロシージャ』を参照してください。

5　disk init を使用して、新しいデータベース・デバイスを初期化します。『システム管理ガイド 第 1 巻』の「第 7 章 データベース・デバイスの初期化」を参照してください。

6　データベースを 1 つずつ再作成します。

7　最新のデータベース・ダンプを各データベースにロードします。

8　各トランザクション・ログ・ダンプを作成順に適用します。

次のセクションでは、ここに記載されたものより詳細な情報が必要な手順について説明します。

## トランザクション・ログの最新のダンプの取得

障害が発生したデバイスにある各データベースの最新のトランザクション・ログ・ダンプを取得するには、dump transaction with no_truncate を使用します。たとえば、mydb の最新のトランザクション・ログのダンプを取得するには、次のように入力します。

```
dump transaction mydb
to "/dev/nrmt0" at REMOTE_BKP_SERVER
with init, no_truncate,
notify = "operator_console"
```

## 領域使用量の調査

このセクションで説明された手順を使用して、データベースに使用されているデバイスと各デバイスに割り付けられた領域の量、およびその領域がデータまたはログ、あるいはその両方に使用されているかを判断します。データベースを再作成する場合にこの情報を使用して、ログ、データ、インデックスをそれぞれ別のデバイスに配置し、作成したユーザ・セグメントのスコープを維持します。

---

**注意** ハードウェアとソフトウェアのプラットフォームが同じであるサーバ間でデータベース・ダンプを移動する場合にも、次に示す手順に従うことにより、セグメントのマッピングを維持できます。

---

この情報を使用しないで、損傷したデータベースのデバイス割り付けを再作成した場合は、load database コマンドの実行後に、矛盾を解決するために sysusages テーブルの再マッピングが行われます。これは、データベースのシステム定義セグメントとユーザ定義セグメントが、適切なデバイス割り付けと一致しなくなることを意味します。sysusages テーブル内の情報が正しくないときは、リカバリ前にはデータとログが別のデバイスに配置されていた場合でも、ログとデータが同じデバイス上に保管される可能性があります。このとき、ユーザ定義セグメントがどう変更されるかが予測できない場合もあり、その結果、標準の create database コマンドを使用してもデータベースを作成できなくなることもあります。

すべての損傷したデータベースのデバイス割り付けを調べて記録するには、次の手順に従ってください。

1 master 内で、次のクエリを使用して、損傷したデータベースのデバイス割り付けと使用状況を調べます。

```
select segmap, size from sysusages
    where dbid = db_id("database_name")
```

2 クエリの出力を調べます。segmap が 3 のローは、データの割り付けを表します。segmap が 4 のローは、ログの割り付けを表します。これよりも大きい値は、ユーザ定義のセグメントを示します。これらは、セグメントのスコープを保持するために、データ割り付けとして扱います。size カラムは、データのブロック数を示します。各ディスク部分の順序、使用量、およびサイズに注意してください。

たとえば、2K 論理ページを使用しているサーバからの出力が次のとおりであるとすると、変換後のサイズと使用量は表 13-6 のようになります。

```
segmap      size
-------     --------
     3        10240
     3         5120
     4         5120
```

```
8           1024
4           2048
```

**表 13-6: デバイスの割り付けの例**

| デバイスの割り付け | メガバイト |
|---|---|
| データ | 20 |
| データ | 10 |
| ログ | 10 |
| データ (ユーザ定義セグメント) | 2 |
| ログ | 4 |

**注意** segmap カラムの値が 7 の場合は、データとログが同じデバイスにあるため、最新のデータベース・ダンプの時点までしかリカバリできないことを示します。create database には log on オプションを使用しないでください。sysusages からレポートされた合計と同じか、それ以上の領域を割り付けるだけにしてください。

3　データベースに対応する sp_helpdb *database_name* を実行します。このクエリは、データとログが配置されているデバイスのリストを表示します。

```
name        db_size owner   dbid    created      status
--------    ------- ------  ----    -----------  -------------
mydb        46.0 MB sa        15    May 26 2005  no_options set

device_fragments  size    usage      created              free kbytes
----------------  -----   --------   ------------         -----------
datadev1          20 MB   data only  June 7 2005 2:05PM         13850
datadev2          10 MB   data only  June 7 2005 2:05PM          8160
datadev3           2 MB   data only  June 7 2005 2:05PM          2040
logdev1           10 MB   log only   June 7 2005 2:05PM  not applicable
logdev2            4 MB   log only   June 7 2005 2:05PM  not applicable
```

## データベースの削除

障害が発生したデバイスの全データベースに対して前述の手順を実行した後に、drop database コマンドを使用して各データベースを削除します。

**注意** 他のデータベースのテーブルに、削除しようとするデータベース内のテーブルへの参照が含まれている場合は、alter table を使用して参照整合性制約を解除しなければデータベースは削除できません。

drop database コマンドを発行したときに、データベースが損傷していることが原因でシステムのエラーがレポートされた場合は、以下を使用します。

```
dbcc dbrepair (mydb, dropdb)
```

複写データベースを使用している場合には、dbcc dbrepair を使用して、以前のバージョンの Adaptive Server からさらに新しいバージョンに、ダンプをロードしてください。次に例を示します。

- 旧バージョンの Adaptive Server の運用システムから最新バージョンの Adaptive Server のテスト・システムに、ダンプをロードします。
- ウォーム・スタンバイ・アプリケーションで、最新バージョンの Adaptive Server のスタンバイ・データベースを、旧バージョンの Adaptive Server のアクティブ・データベースからのデータベース・ダンプで初期化します。

『ASE トラブルシューティング＆エラー・メッセージ・ガイド』および『リファレンス・マニュアル：コマンド』を参照してください。

## データベースの再作成

既に収集したセグメント情報を利用して各データベースを再作成します。

---

**注意** セグメントの使用状況の情報を収集しなかった場合は、create database...for load を使用して、元のデータベース以上のサイズのデータベースを新しく作成してください。

---

1 for load オプションを指定して create database コマンドを実行します。sysusages テーブル内のそのデータベースに対応する各ローのとおりに、最初のログ・デバイスまでの (最初のログ・デバイスも含む) デバイス・フラグメントのマッピングとサイズを指定します。必ず、sysusages でのローの順序のとおりに指定してください (sp_helpdb の結果は、割り付けられた順序ではなくデバイス名のアルファベット順で表示されます)。たとえば、表 13-6 (403 ページ) のとおりに mydb データベースの割り付けを再作成するには、次のコマンドを入力します。

```
create database mydb
    on datadev1 = 20,
        datadev2 = 10
log on logdev1 = 10
for load
```

**注意** create database...for load を実行すると、新しく作成されたデータベースは一時的にユーザが使用できない状態になり、load database を実行すると、データベースは通常使用ができるようにオフラインとしてマーク付けされます。このようなしくみにより、リカバリ中は、ログに記録されるトランザクションをユーザが実行できないようにします。

2　for load オプションを指定して alter database コマンドを実行し、残りのエントリを順番に再作成します。ユーザ・セグメントに対するデバイス割り付けは、データ割り付けと同様に行ってください。

この例で、datadev3 にデータ領域を、logdev1 にログ領域をさらに割り付けるには、次のコマンドを実行します。

```
alter database mydb
    on datadev3 = "2M"
log on logdev1= "4M"
for load
```

## データベースのロード

load database を使用してデータベースを再ロードします。元のデータベースでオブジェクトがユーザ定義のセグメント (sysusages の segmap が 7 より大きい) に格納されている場合に、新しいデバイス割り付けがダンプしたデータベースのものと一致していれば、ユーザ・セグメントのマッピングは維持されます。

作成した新しいデバイス割り付けが、ダンプされたデータベースのデバイス割り付けと一致しない場合は、セグメントは使用可能なデバイス割り付けに再マッピングされます。このとき、この再マッピングによって、同じ物理デバイス上にログとデータが混在してしまうこともあります。

**注意** データベースをロードしているときに新しい障害が発生すると、Adaptive Server は部分的にロードしたデータベースをリカバリせず、ユーザに通知します。load コマンドを繰り返してデータベースのロードを再開してください。

## トランザクション・ログのロード

load transaction を使用して、作成時と同じシーケンスでトランザクション・ログのバックアップを適用します。最新のダンプを最後にロードします。

ダンプしたデータベースおよびトランザクション・ログごとにタイムスタンプの確認が行われます。ダンプをロードする順序が正しくない場合や、ロードとロードの間にユーザ・トランザクションによってトランザクション・ログが変更された場合は、ロードは失敗します。

with standby_access を使用してトランザクション・ログをダンプした場合は、ロードするときも必ず standby_access を使用してください。

データベースが最新の状態になったら、dbcc コマンドを使用して一貫性を検査してください。

### 特定の時点までのトランザクション・ログのロード

トランザクション・ログ内の特定の時点までデータベースをリカバリすることができます。このようにするには、load transaction コマンドの until_time オプションを使用します。この機能は、たとえば、不注意により重要なテーブルを削除してしまった場合などに役立ちます。until_time オプションを使用すれば、テーブルが削除される直前までのデータベースに加えられた変更を、削除されたテーブルを含めてリカバリできます。

データが破壊された後で、この until_time オプションを効果的に使用するには、エラーが発生した正確な時刻を知る必要があります。時刻を調べるには、エラー発生時に select getdate を発行します。たとえば、ユーザが誤って重要なテーブルを削除してしまったとします。管理者は、現在の時刻を次のようにミリ秒単位で表示します。

```
select convert(char(26), getdate(), 109)

--------------------------
Mar 26 2007 12:45:59:650PM
```

エラーが含まれているトランザクション・ログをダンプして、最新のデータベース・ダンプをロードします。次に、最後のデータベース・ダンプ後に作成されたトランザクション・ログをロードします。そして、エラーが含まれているトランザクション・ログを、until_time を使用してロードします。この例では、10 分ほど早く実行されます。

```
load transaction employees_db
from "/dev/nrmt5"
with until_time = "Mar 26 2007 12:35:59: 650PM"
```

until_time を使用してトランザクション・ログをロードすると、その時点からデータベースのログ・シーケンスが再び始まります。したがって、until_time を使用して load transaction を実行した後は、データベースを再度ダンプするまでは後続のトランザクション・ログをロードすることはできません。別のトランザクション・ログをダンプするには、その前にデータベースをダンプする必要があります。

### データベースをオンラインにする方法

すべてのトランザクション・ログ・ダンプをデータベースに適用した後で、online database コマンドを実行して、ユーザがデータベースを使用できるようにします。たとえば、mydb データベースをオンラインに設定するには、次のように入力します。

```
online database mydb
```

### 複写データベース

複写データベースを現在のバージョンの Adaptive Server にアップグレードする前に、データベースをオンライン状態にしてください。ただし、ログが排出されるまでは、複写データベースをオンラインにすることはできません。ログが排出される前に複写データベースをオンライン状態にしようとすると、Adaptive Server は次を表示します。

```
Database is replicated, but the log is not yet drained.
This database will come online automatically after the log
is drained.
```

Replication Server が LTM (Log Transfer Manager) を使用してログを排出すると、online database コマンドが自動的に実行されます。

Adaptive Server ユーザが複写データベースを使用している場合のアップグレード手順については、プラットフォームの『ASE インストール・ガイド』を参照してください。

ロード・シーケンス

複写データベースをロードするためのロード・シーケンスは、load database、replicate、load transaction、replicate などです。ロード・シーケンスの最後に、必ず online database コマンドを実行して、データベースをオンライン状態にしてください。ロード・シーケンス中であるためにオフライン状態になっているデータベースを、Replication Server が自動的にオンライン状態にすることはありません。

---

**警告！** すべてのトランザクション・ログがロードされるまでは、online database コマンドを発行しないでください。

---

## 旧バージョンからのデータベース・ダンプのロード

ほとんどのアップグレード中、サーバに関連するすべてのデータベースは自動的にアップグレードされます (たとえば、バージョン 12.5.x からバージョン 15.5 へのアップグレードなど主要アップグレードでは、preupgrade および upgrade ユーティリティを使用してアップグレードを完了する必要があります)。

したがって、以前のバージョンの Adaptive Server で作成されたデータベース・ダンプとトランザクション・ログ・ダンプを現在のバージョンの Adaptive Server で使用するには、これらのダンプもアップグレードする必要があります。

Adaptive Server には、データベース単位で自動アップグレードを行うメカニズムがあります。このメカニズムは、データベースまたはトランザクション・ログ・ダンプを現在のバージョンの Adaptive Server にアップグレードすることによって、ダンプを、互換性のある使用可能な状態にするものです。このメカニズムは完全に Adaptive Server の内部的な機能であり、外部プログラムを必要としません。これにより、必要に応じて個々のダンプを柔軟にアップグレードできます。

ただし、これらのタスクは、この自動アップグレード機能によってサポートされていません。

- 以前のバージョンの master データベースのロード。つまり、Adaptive Server を最新バージョンにアップグレードすると、アップグレード前のバージョンの Adaptive Server で作成した master データベースのダンプはロードできなくなります。
- 新しいストアド・プロシージャまたは変更したストアド・プロシージャのインストール。引き続き installmaster を使用できます。

## 最新バージョンの Adaptive Server へのダンプのアップグレード

ユーザ・データベースまたはトランザクション・ログのダンプを最新版の Adaptive Server にアップグレードするには、次の手順に従います。

1 load database と load transaction を使用して、アップグレード対象のダンプをロードします。

Adaptive Server は、ダンプ・ヘッダを調べて、ロードするバージョンを判断します。ダンプ・ヘッダが読み込まれた後、Backup Server がロードを始める前に、データベースは load database コマンドまたは load transaction コマンドによってオフラインとしてマーク付けされます。これによって、データベースの通常使用はできなくなり (クエリと use *database* は許可されません)、ロード・シーケンスを自由に制御できるようになります。また、他のユーザが誤ってロード・シーケンスを中断してしまうおそれはなくなります。

2　ダンプのロードが成功したら、online database コマンドを使用してアップグレード処理をアクティブにします。

**注意** すべてのトランザクション・ログがロードされるまでは、online database コマンドを発行しないでください。

バージョン 12.0 以降からのダンプをロードする場合は、online database コマンドを実行するとアップグレード処理がアクティブになり、ロード済みのダンプがアップグレードされます。アップグレードが正常に終了すると、データベースはオンライン状態になり、使用可能になります。

現在実行中の Adaptive Server からのダンプをロードする場合は、アップグレード処理はアクティブになりません。必ず online database コマンドを実行してデータベースをオンライン状態にしてください (load database コマンドを実行すると、データベースはオフラインになります)。

アップグレード処理中は、個々のアップグレード手順が実行される前に、その手順での処理内容を示すメッセージが表示されます。

アップグレードが失敗すると、データベースはオフラインのままになり、アップグレードが失敗したことを示すメッセージが表示されます。この場合は、ユーザが修正する必要があります。

『リファレンス・マニュアル：コマンド』を参照してください。

3　online database コマンドの実行が正常に終了したら、dump database コマンドを実行します。データベースをダンプしなければ、dump transaction は実行できません。新たに作成したデータベースまたはアップグレードしたデータベースに対して dump transaction を実行するには、その前に dump database が正常に終了していなければなりません。

## データベース・オフラインのステータス・ビット

「データベース・オフライン」ステータス・ビットは、データベースが通常使用できる状態ではないことを示します。データベースがオフラインかどうかを調べるには、sp_helpdb を使用します。このビットが設定されていると、データベースがオフラインであることを示しています。

load database によってデータベースがオフラインになると、sysdatabases テーブル内のステータス・ビットが設定され、online database が正常に終了するまで設定されたままになります。

「データベース・オフライン」ステータス・ビットは、他の既存のステータス・ビットと組み合わせることによって機能します。次のステータス・ビットを強化して、より詳細な制御が可能になります。

- リカバリ中

「データベース・オフライン」ステータス・ビットは、これらのステータス・ビットよりも優先されます。

- DBO 使用のみ
- 読み込み専用

これらのステータス・ビットは「データベース・オフライン」ステータス・ビットよりも優先されます。

- アップグレードを開始
- リカバリを無視
- ロード中
- リカバリされていない
- 疑わしい
- 使用がリカバリされていない

データベースがオフラインの場合、データベースの通常の使用はできませんが、次のコマンドを使用することはできます。

- dump database と dump transaction
- load database と load transaction
- alter database on device
- drop database
- online database
- dbcc 診断 (dbcc の制限事項に従う)

### バージョン識別子

自動アップグレード機能によって、次のように Adaptive Server、データベース、ログ・レコード・フォーマットのバージョン識別子が設定されます。

- 設定アップグレード・バージョン識別子 － Adaptive Server の現在のバージョンを示すもので、sysconfigures テーブルに格納されます。sp_configure の出力では、Adaptive Server の現在のバージョンは「アップグレード・バージョン」と表示されます。
- アップグレード・バージョン識別子 － データベースの現在のバージョンを示すもので、データベースとダンプのヘッダに格納されます。Adaptive Server のリカバリ機能は、データベースを通常使用可能な状態にする前に、アップグレードが必要かどうかをこの値を使用して調べます。
- ログ互換性バージョン識別子 － データベース、データベース・ダンプ、またはトランザクション・ログ・ダンプ内のログ・レコードのフォーマットを表すもので、これによって Adaptive Server の以前のバージョンと最新バージョンのログを区別します。この定数は、データベースとダンプのヘッダに格納され、Adaptive Server がリカバリ時にログ・レコードのフォーマットを調べるために使用します。

## キャッシュのバインドとデータベースのロード

データベースをダンプして、別のキャッシュ・バインドを使用するサーバにロードする場合は、データベースとその中のオブジェクトに対するキャッシュ・バインドについて注意が必要です。たとえば、チューニングまたは開発作業のために別のサーバ上にデータベースをロードする場合です。あるいは、サーバから削除したデータベースをロードするときに、そのキャッシュ・バインドがダンプの作成後に変更されている場合です。

リカバリ後にデータベースをオンライン状態にするとき、または、ロード後に online database コマンドを使用してデータベースをオンライン状態にするときは、そのデータベースとデータベース・オブジェクトに対するキャッシュ・バインドの検証が実行されます。キャッシュが存在しない場合は、エラー・ログに警告が書き込まれ、sysattributes ではそのバインドに無効のマークが付けられます。次のメッセージは、エラー・ログに書きこまれるメッセージの例です。

```
Cache binding for database '5', object '208003772', index
'3' is being marked invalid in Sysattributes.
```

無効なキャッシュ・バインドは削除されません。同じ名前のキャッシュを作成して Adaptive Server を再起動すれば、そのバインドには有効のマークが付けられ、作成したキャッシュが使用されます。同じ名前を持つキャッシュを作成しない場合は、そのオブジェクトを別のキャッシュにバインドするか、デフォルト・キャッシュを使用するようにします。

キャッシュ・バインドについて説明する以下の各項では、ロード先サーバはデータベースがロードされるサーバのことを指し、オリジナル・サーバはダンプが行われるサーバのことを指します。

可能な限り、オリジナル・サーバでのバインドと同じ名前を持つキャッシュをロード先サーバ上で再作成してください。ロード先データベースの使用目的がオリジナル・サーバと同様である場合や、パフォーマンス・テストと開発を行ってその結果をオリジナル・サーバに移植する場合には、プールの設定をオリジナル・サーバとロード先サーバとで同一にすることも検討してください。意思決定支援を目的として、または dbcc コマンドを実行するためにロード先データベースを使用する場合は、16K メモリ・プールでさらに多くの領域を使用できるようにプールを設定してください。

## データベースとキャッシュ・バインド

データベースのバインド情報は、master..sysattributes に格納されます。データベースのバインド情報は、データベース自体には保管されません。キャッシュにバインドされている既存のデータベースに load database コマンドを使用してダンプをロードするとき、ロード・コマンドを実行する前にそのデータベースを削除するのでなければ、バインドへの影響はありません。

ロードするデータベースが、オリジナル・サーバではキャッシュにバインドされていたものである場合は、次のいずれかを行ってください。

- ロード先サーバ上のデータベースを、そのサーバでの必要量に応じて設定されたキャッシュにバインドする。
- ロード先サーバ上のデフォルト・データ・キャッシュ内のプールを、そのサーバ上のアプリケーションの必要量に応じて設定し、そのデータベースを名前付きデータ・キャッシュにはバインドしない。

## データベース・オブジェクトとキャッシュ・バインド

オブジェクトのバインド情報は、データベース自体の sysattributes テーブルに格納されます。ロード先サーバ上にデータベースを頻繁にロードする場合は、ロード先サーバ上に同じ名前のキャッシュを設定するのが最も簡単な解決法です。

ロード先サーバ上に、オリジナル・サーバと同じ名前のキャッシュが設定されていない場合は、ロード先サーバで、データベースをオンライン状態にしてから適切なキャッシュにオブジェクトをバインドするか、ロード先サーバ上の必要量に応じてデフォルト・キャッシュが設定されていることを確認します。

### キャッシュ・バインドの確認

sp_helpcache を使用すると、データベース・オブジェクトのキャッシュ・バインドが表示されます。これには、無効なキャッシュ・バインドも含まれます。

次の SQL 文は、ユーザ・データベースの sysattributes テーブルにある情報を使用して、キャッシュ・バインド・コマンドを再生成します。

```
/* create a bindcache statement for tables */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", " + object_name(object)
from sysattributes
where class = 3
    and object_type = "T"

/* create a bindcache statement for indexes */

select "sp_bindcache "+ char_value + ", "
    + db_name() + ", "   + i.name
from sysattributes, sysindexes i
where class = 3
    and object_type = "I"
    and i.indid = convert(tinyint, object_info1)
    and i.id = object
```

# データベース間の制約とデータベースのロード

create table または alter table の references (参照) 制約を使用して、複数データベース間でテーブルを参照する場合、そのデータベースのいずれかのダンプをロードしようとしたときに問題が発生することがあります。

- ダンプされたデータベースが、他のデータベース内のテーブルから参照されている場合は、ダンプされたデータベースを別の名前でロードしたり、ダンプされたときとは異なるサーバ上でロードしたりすると、参照整合性エラーが発生します。再ロードするときにデータベースの名前またはロケーションを変更する場合は、そのデータベースをダンプする前に、参照元データベース内で alter database を使用して外部参照整合性制約をすべて削除してください。
- ロードする参照先データベースのダンプが、参照元データベースよりも前のバージョンのものである場合は、一貫性の問題またはデータ破壊が起こる可能性があります。このことを防ぐには、データベース間制約を追加したり削除したりするたびに、またはデータベース間制約を含むテーブルを削除するたびに、影響を受ける両方のデータベースをダンプしてください。
- 相互参照するすべてのデータベースを同時にダンプしてください。同期に関する問題が起きないようにするには、両方のデータベースをダンプするときにシングルユーザ・モードにしてください。これらのデータベースをロードするときは、両方を同時にオンラインにします。

次のような場合に、データベース間の制約の一貫性が失われることがあります。

- データベース・ダンプのロードが日付順に行われない (たとえば、1997 年 8 月 12 日に作成されたダンプを、8 月 13 日に作成されたダンプの後にロードする) 場合
- 新しい名前のデータベースにダンプをロードする場合

データベース・ダンプのロードを行わないと、データベース間の制約は一貫性を失うことがあります。

この問題を解決するには、次の手順に従います。

1 両方のデータベースをシングルユーザ・モードにします。

2 一貫性が失われた参照制約を削除します。

3 次のようなクエリを使用して、データの一貫性を検査します。

```
select foreign_key_col from table1
where foreign_key not in
(select primary_key_col from otherdb..othertable)
```

4 データの一貫性に問題がある場合は、解決します。

5 制約を再作成します。

第 14 章

# システム・データベースのリストア



## システム・データベースのリカバリ

システム・データベースに対するリカバリの手順は、どのデータベースが関係するかと、システム上の問題点によって異なります。一般に、リカバリでは次の手順を実行します。

- load database を使用して、データベースのバックアップをロードします。
- dataserver、installmaster、installmodel を使用して、データベースの初期状態にリストアします。
- これらのタスクを組み合わせます。

システム・データベースのリカバリをできる限り効率的に行うには、システム管理において次の点に注意してください。

- ユーザ・データベースなどの、master、tempdb、model、sybsystemdb 以外のデータベースを、マスタ・デバイス上に保存しないでください。
- 重要なシステム・テーブルについては、必ず最新の状態のプリントアウトを保存してください。
- データベース・デバイスの初期化、データベースの作成や変更、新しいサーバ・ログインの追加などを行った後は、必ず master データベースをバックアップしてください。

# *master* データベースのリカバリ

master データベースの損傷の原因には、master が保管されている領域のメディア障害や、データベースの内部的な破壊があります。次の場合は、master データベースが損傷しています。

- Adaptive Server が起動できない
- 頻繁な、あるいは悪影響を与えるようなセグメンテーション・フォールトまたは I/O エラーの発生
- dbcc (データベースの一貫性チェッカ) によるデータベースの定期検査時に損傷がレポートされる

この項では、master データベースのリカバリとマスタ・デバイスの再構築の手順を説明するために、次のような状況を想定します。

- master データベースが破壊された、またはマスタ・デバイスが損傷している。
- システム・テーブルの最新のプリントアウトが保管されている。詳細については『システム管理ガイド 第 1 巻』の「第 2 章 システム・データベースとオプションのデータベース」を参照してください。
- マスタ・デバイスには、master データベース、tempdb、model、sybsystemdb だけが格納されている。
- master データベースの最新のバックアップが保管されており、最後に master をダンプした後にデバイスの初期化や、データベースの作成や変更を行っていない。
- サーバはデフォルトのソート順序を使用している。

リカバリ方法の詳細については、『ASE トラブルシューティング＆エラー・メッセージ・ガイド』を参照してください。

## リカバリ処理について

master データベースとマスタ・デバイスは、集中管理を行うという特性があるため、特殊な手順が必要になります。master データベース内のテーブルにより、Adaptive Server のすべての機能、データベース、およびデータ・デバイスが設定され、制御されます。リカバリ・プロセスでの処理内容は、次のとおりです。

- マスタ・デバイスを、サーバを新しくインストールしたときのデフォルト状態に再構築する。
- master データベースをデフォルト状態にリストアする。
- master データベースを、最後のバックアップ時の状態にリストアする。

master データベースのリカバリの初期段階では、システム・ストアド・プロシージャは使用できません。

## リカバリ手順の概要

破壊されたマスタ・デバイスをリストアするには、次の手順に従ってください。各手順の詳細については、表に示している参照先のページで説明しています。

1　ディスク、データベース、ログインのリストアに必要なシステム・テーブルのハードコピーを確認する。

2　Adaptive Server を停止し、dataserver を使用して新しい master データベースとマスタ・デバイスを構築する。

3　Adaptive Server をマスタ・リカバリ・モードで再起動する。

4　master データベースの割り付けを、sysusages のとおりに正確に再作成する。

5　sysservers テーブル内の Backup Server のネットワーク名を更新する。

6　Backup Server が実行中であることを確認する。

7　load database を使用して、master の最新のデータベース・ダンプをロードする。master が正常にロードされると、Adaptive Server は自動的に停止する。

8　設定ファイル内の number of devices 設定パラメータを更新する。

9　Adaptive Server をシングルユーザ・モードで再起動する。

10　master のバックアップに最新のシステム・テーブル情報が格納されていることを確認する。

11　Adaptive Server を再起動する。

12　syslogins を調べて、master の最後のバックアップ以降に新しいログインが追加されているかどうかを確認する。

13　model データベースをリストアする。

14　sysusages と sysdatabases のハードコピーを新しいオンライン・バージョンと比較する。それぞれのデータベースに対して dbcc checkalloc を実行し、データベース内の重要なテーブルを検査する。

15　master データベースをダンプする。

## システム・テーブルのコピーの確認

ファイルに保存したシステム・テーブル、sysdatabases、sysdevices、sysusages、sysloginroles、および syslogins のコピーを確認します。これらのコピーは、これらのプロシージャの完了時にシステムが完全にリストアされていることを確認するために使用します。

『システム管理ガイド 第 1 巻』の「第 2 章 システム・データベースとオプションのデータベース」を参照してください。

## 新しいマスタ・デバイスの構築

新しいマスタ・デバイスを構築する必要があるのは、古いマスタ・デバイスが修復不能なほど損傷した場合だけです。それ以外の場合は、既存のマスタ・デバイス上に master データベースと model データベースを作成し直すことができます。

master データベースを再作成するプロシージャには、マスタ・デバイスを置き換えるものと、Adaptive Server に強制的に設定エリアを再作成させるものの 2 種類があります。マスタ・デバイスが損傷しているときに、マスタ・デバイスを置き換えます。マスタ・デバイスの設定エリアも壊れている場合は、Adaptive Server に強制的に設定エリアを再作成させます。多くの場合、サーバが動作しなくなり、設定エリアが破損しているというエラー・メッセージが表示されることで、設定エリアが破損したことがわかります。

次の例では、UNIX の dataserver コマンドを使用します。Windows の場合は、sqlsrvr コマンドを使用します。

### ❖ マスタ・デバイスの置換

1 dataserver -w オプションを使用してマスタ・デバイスを再構築します。

```
dataserver -w master
```

**注意** dataserver コマンドには、デバイス・パス名、サーバ名、interfaces ファイル名などを指定するコマンド・ライン・フラグといった他のオプションを追加することもできます。これらのオプションは、そのサーバを通常起動する *RUN_servername* ファイルにリストされています。

-w master オプションを指定すると、Adaptive Server は、デバイスで master データベースに属するデータベース・フラグメントが発生しているかどうかを検索します。フラグメントを検出すると、デフォルトの master データベースをその領域に書き込んだ後、停止します。

2 *RUN_servername* ファイルを使用して Adaptive Server を再起動します。

### ❖ 設定エリアの再構築

設定エリアが壊れている場合は、-f オプションを使用して、このエリアを再構築する必要があります。

**注意** dataserver コマンドには、デバイス・パス名、サーバ名、interfaces ファイル名などを指定するコマンド・ライン・フラグといった他のオプションを追加することもできます。これらのオプションは、そのサーバを通常起動する *RUN_servername* ファイルにリストされています。

次の制限があります。

- ページ・サイズが正しくない場合にはページ・サイズを指定できます (たとえば -z8k)。
- デバイス・サイズが正しくない場合にはデバイス・サイズを指定できます (たとえば -b125M)。
- 壊れていると考えられるディスク上のすべてのアロケーション・ユニット、または現在割り付けられていないすべてのアロケーション・ユニットが master データベースに割り付けられます。

---

**警告！** この方法でサーバの論理ページ・サイズを変更しないでください。この操作により、リカバリ不能になるまでデバイスが壊れます。すべてのデバイス・サイズを正確に指定してください。dataserver -w を実行してもデバイスのサイズは変更されませんが、指定するサイズが小さすぎると一部のデータベースの部分を発見できない可能性があり、指定するサイズが大きすぎると全体が失敗する可能性があります。

---

-w master オプションは、-f の指定の有無にかかわらず、master データベースだけを再作成します。ディスク上の他のすべてのアロケーション・ユニットは変更されません。このため、disk refit を使用してデータをリカバリできます。

マスタ・デバイス全体が壊れている場合 ( ディスク障害の場合など ) は、dataserver -z*page_size*. . .-b*device_size* を使用して Adaptive Server を開始することで、デバイス全体を置き換えます。

1　既存のマスタ・デバイスがロー・パーティション上になく、マスタ・デバイスを再使用する場合は、古いマスタ・デバイス・ファイルを削除してください。

2　dataserver -z*page_size*.. .-b*device_size* によりサーバを起動します。

---

**警告！** この時点では、このコマンドを使用してサーバのページ・サイズを変更することはできません。このサーバの他のデバイスはすべて設定されているページ・サイズを使用するので、異なるページ・サイズを使用すると、正しくリカバリされません。ただし、新しいファイルを作成しているので、デバイス・サイズを変更することはできます。

---

作成するマスタ・デバイスの大きさを決定するときは、このデバイスでは設定エリア用に 8KB が予約されていることに注意してください。たとえば、デバイス・サイズを 96MB と指定すると、若干の領域は、アロケーション・ユニットに十分な大きさがないので、無駄になります。このオーバーヘッドを考慮する場合、デバイスに必要な領域に 0.01MB を追加します。たとえば、デバイスの 96MB を完全に使用するには、-b96.01M と指定します。

この方法を使用してマスタ・デバイスを再構築すると、Adaptive Server により、master、model、tempdb、sybsystemdb のデフォルトのデータベースが新しく作成されます。これらのデータベースはすべて、インストールの論理ページ・サイズに対して可能な限り小さくなるように作成されます。これらのデータベースにバックアップをロードする場合、この時点ではデータベースが小さすぎる可能性があります。これらのバックアップをロードする前に、alter database を使用してデータベースのサイズを増やしてください。

マスタ・デバイスをリストアすると、リストア方法にかかわらず master データベース中のすべてのユーザとパスワードは失われます。残っている唯一の権限付きログインは “sa” で、パスワードは設定されていません。

ユーザおよびパスワードの再作成については、『ユーティリティ・ガイド』を参照してください。

## マスタ・リカバリ・モードでの Adaptive Server の起動

-m (UNIX および Windows) オプションを指定して、マスタ・リカバリ・モードで Adaptive Server を起動します。

- UNIX の場合は、runserver ファイルをコピーして *m_RUN_server_name* という名前を付けます。この新しいファイルを編集して、dataserver コマンド・ラインに -m パラメータを追加します。次に、サーバをマスタ・リカバリ・モードで起動します。

  ```
  startserver -f m_RUN_server_name
  ```

- Windows の場合は、コマンド・ラインから sqlsrver コマンドを使用して Adaptive Server を起動します。-m パラメータと他の必要なパラメータを指定します。次に例を示します。

```
sqlsrver.exe -dD:\Sybase\DATA\MASTER.dat -sPIANO -eD:\Sybase\install\errorlog
-iD:\Sybase\ini -MD:\Sybase -m
```

これらのコマンドの構文の詳細については、『ユーティリティ・ガイド』を参照してください。

Adaptive Server をマスタ・リカバリ・モードで起動する場合は、1 人のユーザ、つまりシステム管理者による 1 つのログインだけが許可されます。master データベースで dataserver コマンドを実行した直後は、“sa” アカウントだけが存在します。このアカウントのパスワードは NULL です。

---

**警告！** サイトによっては、起動時に “sa” ログインによってサーバにログインする自動ジョブがあります。このような自動ジョブが実行されないようにしてください。

---

マスタ・リカバリ・モードが必要となるのは、dataserver によって作成された汎用 master データベースが Adaptive Server の実際の状況と一致しないためです。たとえば、汎用 master データベースには、実際のデータベース・デバイスについての情報は含まれていません。master データベースに対して何らかの操作を行うと、リカバリが不可能になるか、複雑で時間がかかるようになる可能性があります。

マスタ・リカバリ・モードで Adaptive Server を起動すると、システム・テーブルを直接更新できる状態に自動的に設定されます。他の特定のオペレーションは使用できません。

---

**警告！** システム・テーブルにアドホックな (定められた以外の) 変更を加えないでください。変更内容によっては、Adaptive Server が起動できなくなることがあります。この章で説明されている変更だけを行い、変更は必ずユーザ定義トランザクション内で行ってください。

---

## *master* のデバイス割り付けの再作成

手順 2 で説明されているプロセスに従ってマスタ・デバイスを再作成した場合は、マスタ・デバイスが小さすぎる可能性があります。master データベースに追加の領域を割り付けるには、次の手順に従います。

1　sysusages のハードコピー・バージョンで、dbid 1 (master データベースの dbid) に対して示されている size の値を合計します。その値を、現在の master データベースのサイズと比較します。この値は、次のコマンドを発行することで確認できます。

```
select sum(size)
from sysusages
where dbid = 1
```

2　現在の master データベースが小さすぎる場合は、alter database を使用して必要なサイズに拡大します。論理ページをメガバイトの値に変換するには、次を使用します。

```
select N / (power(2,20) / @@maxpagesize)
```

*N* は論理ページの数です。

-m master オプションを使用して master データベースを書き直した場合は、master データベースのサイズを変更する必要はありません。Adaptive Server は、そのデバイス上の全データベースで使用されていたアロケーション・ユニットを記録しているため、master のダンプをロードするのに十分な大きさの領域が既に存在します。

---

**注意** sysusages のハードコピーがない場合は、データベースをロードしてみることで、データベースに必要な大きさがわかります。小さすぎる場合は、増やす必要のあるサイズがエラー・メッセージに表示されます。

15.0 よりも前のバージョンの Adaptive Server では、古いバージョンに新しいマスタ・デバイスの割り付けを再作成するには、複雑な手順を実行する必要がありました。この手順は不要になりました。Adaptive Server バージョン 15.0 以降では、ほとんどの作業が自動的に行われます。

---

## Backup Server の *sysservers* 情報のチェック

“sa” としてサーバにログインします。パスワードは null です。

Backup Server のネットワーク名が SYB_BACKUP でない場合は、Adaptive Server が Backup Server と通信できるように、sysservers を更新する必要があります。interfaces ファイルを調べて Backup Server 名を確認し、次のコマンドを発行します。

```
select *
from sysservers
where srvname = "SYB_BACKUP"
```

このコマンドからの出力の srvnetname を確認してください。interfaces ファイル内の Backup Server のエントリと一致している場合は、「Backup Server が実行中であることの確認」(423 ページ) に進んでください。

出力された srvnetname が interfaces ファイル内の Backup Server の名前と同じでない場合は、sysservers を更新してください。次の例では、Backup Server のネットワーク名を PRODUCTION_BSRV に変更します。

```
begin transaction
update sysservers
set srvnetname = "PRODUCTION_BSRV"
where srvname = "SYB_BACKUP"
```

このコマンドを実行し、1 つのローだけが変更されたことを確認してください。再度 select コマンドを実行し、正しいローが変更されたことと、その値が正しいことを確認してください。update コマンドで複数のローを変更した場合、または誤ったローを変更した場合は、rollback transaction コマンドを実行してから、再度更新を行ってください。

コマンドによって Backup Server のローが正確に変更された場合は、commit transaction コマンドを発行します。

## Backup Server が実行中であることの確認

UNIX の場合は、showserver コマンドを使用して Backup Server が稼働しているかどうかを確認し、必要であれば Backup Server を再起動します。『ユーティリティ・ガイド』の「showserver」と「startserver」の項を参照してください。

Windows の場合は、ローカルにインストールされた Sybase Central と Services Manager ユーティリティを使用して、Backup Server が稼働しているかどうかを確認します。

Backup Server を起動するコマンドについては、『ユーティリティ・ガイド』を参照してください。

## *master* のバックアップのロード

master データベースの最新のバックアップをロードします。

- たとえば、UNIX プラットフォームでは、次のコマンドを使用します。

```
load database master from "/dev/nrmt4"
```

- Windows では、次のコマンドを使用します。

```
load database master from "\\.\TAPE0"
```

コマンド構文の詳細については、「第 13 章 ユーザ・データベースのバックアップとリストア」を参照してください。

load database コマンドが正常終了すると、Adaptive Server が停止します。ロードおよび停止の処理中は、エラー・メッセージが表示されたかどうかに注意してください。

## *number of devices* 設定パラメータの更新

この手順は、デフォルトの数よりも多くのデータベース・デバイスを使用する場合にだけ実行します。それ以外の場合は、「マスタ・リカバリ・モードでの Adaptive Server の再起動」(424 ページ) に進んでください。

設定値が Adaptive Server で使用されるようになるのは、master データベースのリカバリ終了後です。したがって、起動時に設定ファイルから number of devices パラメータの値を読み込むように Adaptive Server に指示する必要があります。

最新の設定ファイルを使用できない場合は、number of devices パラメータの正しい値を反映するように新しい設定ファイルを編集してください。

runserver ファイルを編集します。dataserver または sqlsrver コマンドの最後に -c パラメータを追加して、設定ファイルの名前とロケーションを指定します。Adaptive Server の起動時に、指定した設定ファイルからパラメータが読み込まれます。

## マスタ・リカバリ・モードでの Adaptive Server の再起動

startserver を使用して、マスタ・リカバリ・モードで Adaptive Server を再起動します。詳細については、「マスタ・リカバリ・モードでの Adaptive Server の起動」(420 ページ) を参照してください。リカバリ中は、エラー・メッセージが表示されるかどうかを監視してください。

master のバックアップをロードすると、“sa” アカウントは以前の状態にリストアされます。これにより、“sa” アカウントのパスワードが設定されている場合は、そのパスワードがリストアされます。バックアップを行う前に sp_locklogin を使用して “sa” アカウントをロックした場合は、リストア後もこのアカウントはロックされています。以降のリカバリ手順は、sa_role のアカウントを使用して実行してください。

## *master* の最新のバックアップを確認するためのシステム・テーブルの検査

disk init コマンド、create database コマンド、または alter database コマンドの最後の実行より後に master データベースをバックアップしていれば、sysusages、sysdatabases、sysdevices の内容はハードコピーと一致します。

リカバリ対象のサーバ内の sysusages テーブル、sysdatabases テーブル、sysdevices テーブルを、ハードコピーと照合します。特に、以下のような問題に注意してください。

- ハードコピーにあるデバイスが、リストアされた sysdevices テーブル内に存在しない場合は、最後のバックアップ以降にデバイスが追加されています。disk reinit と disk refit を実行してください。「disk reinit と disk refit によるシステム・テーブルのリストア」(433 ページ) を参照してください。
- ハードコピーにあるデータベースが、リストアされた sysdatabases テーブル内に存在しない場合は、master の最後のバックアップ以降にデータベースが追加されていることを意味します。disk refit を実行してください。

---

**注意** disk refit を実行する前に、トレース・フラグ 3608 を有効にして Adaptive Server を起動する必要があります。ただし、トレース・フラグを使用して Adaptive Server を開始する前に、必ず『トラブルシューティング&エラー・メッセージ・ガイド』の情報に目を通してください。

---

## Adaptive Server の再起動

Adaptive Server を通常モード (マルチユーザ・モード) で再起動します。

## サーバ・ユーザ ID のリストア

syslogins のハード・コピーと、リストアした syslogins テーブルを確認します。

- master の最後のバックアップ以降にサーバ・ログインを追加した場合は、create login コマンドを再発行してください。
- サーバ・ログインを削除した場合は、drop login コマンドを再発行してください。
- サーバ・アカウントをロックした場合は、sp_locklogin コマンドを再発行してください。
- ユーザまたはシステム管理者が alter login プロシージャを実行したことが原因で発生した不一致が他にあるかどうかを確認してください。

ユーザに割り当てられている suid が正しいことを確認してください。データベース内の suid 値が一致していなければ、パーミッション上の問題が発生する可能性があり、ユーザがテーブルにアクセスできなくなることや、コマンドを実行できなくなることがあります。

既存の suid の検査を効率的に行うには、ユーザ・データベースの各 sysusers テーブルに対して union を実行します。master を使用するパーミッションをユーザに与えている場合は、このプロシージャに master を含めます。

次に例を示します。

```
select suid, name from master..sysusers
union
select suid, name from sales..sysusers
union
select suid, name from parts..sysusers
union
select suid, name from accounting..sysusers
```

この結果のリストで、ログインをリストアする範囲内の suid 値の中に連続していない部分がある場合は、欠番となっている値に対応するプレースホルダを追加し、次に drop login を使用してその値を削除するか、sp_locklogin を使用してロックしてください。

## *model* データベースのリストア

model データベースのリストア

- model のバックアップがある場合は、そのバックアップをロードします。
- バックアップがなければ、ほとんどのプラットフォームの場合、次の installmodel スクリプトを実行します。

```
cd $SYBASE/$SYBASE_ASE/scripts
isql -Usa -Ppassword -Sserver_name < installmodel
```

Windows の場合：

```
cd $SYBASE/$SYBASE_ASE/scripts
isql -Usa -Ppassword -Sserver_name < instmodl
```

- model に対して行った変更をすべて再実行します。

## Adaptive Server の確認

次の手順に従い、Adaptive Server の確認を十分に行います。

1. sysusages のハードコピーを新しいオンライン・バージョンと比較します。
2. sysdatabases のハードコピーを新しいオンライン・バージョンと比較します。
3. 各データベースに対して dbcc checkalloc を実行します。
4. 各データベースの重要なテーブルを調べます。

**警告！** sysusages 内に不一致がある場合は、Sybase 製品の保守契約を結んでいるサポート・センタに連絡してください。

## *master* のバックアップ

master データベースが完全にリストアされ、dbcc による完全な整合性検査が完了したら、通常のダンプ・コマンドを使用してデータベースのバックアップを作成してください。

## *model* データベースのリカバリ

この項では、リストアが必要なデータベースが model データベースだけである場合の、そのリカバリ方法について説明します。

dataserver を使用して、master に影響を与えることなく model データベースをリストアします。

---

**警告！** dataserver コマンドを実行する前に、Adaptive Server を停止してください。

---

- UNIX の場合：

```
dataserver -d /devname  -w model
```

- Windows の場合：

```
sqlsrvr -d physicalname -w model
```

use model コマンドを正常に発行できる場合は、load database コマンドを使用して model データベースをバックアップからリストアできます。

データベースが使用できない場合は、次の手順に従います。

1　前述の dataserver コマンドを発行します。

2　model のサイズを変更した場合は、alter database コマンドを再発行します。

3　load database を使用してバックアップをロードします。

model データベースを変更し、バックアップがない場合は、次の操作を行います。

1　前述の dataserver コマンドを発行します。

2　model の変更時に発行したコマンドをすべて再発行します。

## *sybsystemprocs* データベースのリカバリ

sybsystemprocs データベースには、システム・テーブルの変更およびレポートに使用されるシステム・プロシージャが保管されています。日常の dbcc 検査によってこのデータベースの損傷が報告されたが、このデータベースのバックアップを作成していない場合は、*installmaster* を使用してリストアします。sybsystemprocs のバックアップが保管されている場合は、load database を使用して sybsystemprocs をリストアします。

## *installmaster* による *sybsystemprocs* のリストア

この項では、sybsystemprocs データベースは存在するが壊れていることを前提としています。sybsystemprocs が存在しない場合は、create database を使用して作成する必要があります。

*installmaster* を使用して sybsystemprocs をリストアするには、次の手順に従います。

1 sybsystemprocs が現在どのデバイスに保管されているかを調べます。

```
select  lstart,
size / (power(2,20)/@@maxpagesize) as 'MB',
d.name as 'device name',
case when segmap = 4 then 'log'
when segmap & 4 = 0 then 'data'
else 'log and data'
end as 'usage'
from sysusages u, sysdevices d
where d.vdevno = u.vdevno
and d.status & 2 = 2
and dbid = db_id('sybsystemprocs')
order by 1
```

通常この結果には、1 つのディスク・フラグメント上にあり、用途が `log and data` の sybsystemprocs がすべて表示されます。ただし、さらに複雑なレイアウトが表示されることもあります。後で使用するため、このクエリの結果を保存しておきます。

2 データベースを削除します。

```
drop database sybsystemprocs
```

削除が成功し、デバイスが損傷を受けていない場合は、手順 3 に進みます。

- sybsystemprocs がひどく破壊されている場合、drop database が失敗する可能性があります。データベースを識別する情報を削除することで、データベースを手動で削除できます。

```
sp_configure ‘allow updates’, 1
go
delete from sysusages
where dbid = db_id('sybsystemprocs')
delete from sysdatabases
where name = 'sybsystemprocs'
go
sp_configure 'allow updates', 0
go
```

- 物理ディスクが破壊されている場合は、デバイスを削除します。

```
sp_dropdevice name_of_sybsystemprocs_device
```

手動で sybsystemprocs を削除した場合、sybsystemprocs デバイスを再作成し、shutdown with nowait を使用して Adaptive Server をシャットダウンします。sybsystemprocs デバイスを削除し、それがロー・パーティションではなかった場合は、物理ファイルを削除します。Adaptive Server を再起動します。

3　sybsystemprocs デバイスを再作成します。sybsystemprocs デバイスを削除した場合は、disk init を使用して新しいデバイスを作成します。その後、手順 1 で保存しておいた結果を使用し、次のいずれかの方法を用いて sybsystemprocs を再作成します。

---

**注意**　sybsystemprocs のバックアップ・コピーをロードする予定の場合は、create database コマンドまたは alter database コマンドに for load オプションを含めることができます。ただし、他の目的でバックアップ・コピーを使用する前に、load database でロードする必要があります。

---

- 表示された usage がすべて log and data の場合は、次のコマンドを使用して単純なデータベースを作成します。

  ```
  create database sybsystemprocs on device_name = N
  ```

  *N* は、デバイスの合計サイズです。必要なサイズを確保するためには、複数のデバイスに新しいデータベースを作成することが必要になる場合があります。

- 表示された用途に log または data のエントリが含まれている場合は、create database と alter database を使用して同じレイアウトを再作成します連続するデータ・セクションまたはログ・セクションを 1 つのデバイスにグループ化することはできますが、ログとデータを混在させることはできません。データ・セクションとログ・セクションの最初のグループを再作成します。

  ```
  create database sybsystemprocs
  on device_1 = M
  log on device_2 = N
  ```

  *M* は最初のデータ・グループのサイズの合計で、*N* は最初のログ・グループのサイズの合計です。create database の代わりに alter database を使用して、後に続く各グループに対してこの処理を繰り返し、データベースを拡大します。

4　*installmaster* スクリプトを実行して、Sybase 提供のシステム・プロシージャを作成します。

UNIX の場合：

```
isql -Usa -Ppassword -Sserver_name -i
$SYBASE_/$SYBASE_ASE/scripts/installmaster
```

Windows の場合 (*%SYBASE%\%SYBASE_ASE%\scripts* ディレクトリから)：

```
isql -Usa -Ppassword -S<server_name> -i instmstr
```

5 サイトで sybsystemprocs にプロシージャを追加した場合、または他の変更を行った場合は、新しい sybsystemprocs データベースにも同じ変更を加える必要があります。

### *load database* による *sybsystemprocs* のリストア

作成したシステム・プロシージャを sybsystemprocs に保管する場合は、このデータベースが損傷したときのリカバリ方法には次の 2 つがあります。

- 「installmaster による sybsystemprocs のリストア」(428 ページ) の手順 4 の説明に従い、installmaster からデータベースをリストアします。次に、create procedure コマンドを再発行してプロシージャを再作成します。
- データベースのバックアップを保管しておき、load database を使用してバックアップをロードします。

データベースのバックアップを保管するという方法を選択した場合は、必ずバックアップ全体が 1 本のテープ・ボリュームに収まるようにするか、複数の Adaptive Server が Backup Server と通信できるようにしてください。ダンプが複数のテープ・ボリュームにわたる場合は、sybsystemprocs に保管されている sp_volchanged システム・プロシージャを使用して、ボリューム交換コマンドを実行します。このコマンドは、データベースのリカバリ中に実行することはできません。

- たとえば、UNIX では、次のコマンドを使用します。

```
load database sybsystemprocs from "/dev/nrmt4"
```

- Windows では、次のコマンドを使用します。

```
load database sybsystemprocs from "\\.\TAPE0"
```

## *tempdb* のサイズ縮小方法

tempdb (テンポラリ) データベースは、テンポラリ・テーブルやその他の一時的な作業領域に必要な記憶領域を提供します。tempdb の一部を含んでいる破損ディスクがある場合、最初に tempdb をデフォルト・サイズに縮小し、次に新しいデバイスまで拡張する必要があります。

この項では、tempdb をデフォルト・サイズ (データの 2MB およびマスタ・デバイスへのログオン) に縮小する方法について説明しています。

## *tempdb* をデフォルト・サイズにリセットする

sysusages を手動で更新する間に他のユーザがデータベースを変更できないように、処理前に Adaptive Server をシングル・ユーザ・モードで起動します。

1　tempdb をマスタ上で 4mb まで増やします。

2　システム管理者として Adaptive Server にログインします。

3　障害が発生した場合、master データベースをダンプし、バックアップからリストアします。

```
dump database master to "dump_device"
```

*dump_device* は、ターゲット・ダンプ・デバイスの名前です。

4　必要であれば、master データベース・リカバリを支援するため、bcp..out コマンドを使用して次のキー・システム・テーブルをデータ・ファイルに保存します。

- master..sysusages
- master..sysdevices
- master..sysdatabases
- master..syslogins
- master..sysconfigures
- master..syscharsets
- master..sysloginroles
- master..sysservers
- master..sysremotelogins
- master..sysresourcelimits
- master..systimeranges

**警告！** このプロシージャは、tempdb にのみ使用してください。システムがシャット・ダウンされて再起動されるたびに tempdb が再構築されるため、このプロシージャが機能します。他のデータベースでこのプロシージャを使用すると、データベース破損の原因になります。

5　master データベースから Adaptive Server を再設定して、システム・カタログの変更を可能にします。

```
sp_configure "allow updates", 1
```

6 tempdb に属する現在のローを sysusages から表示し、影響されるローの数を記録します。

```
begin transaction
select * from sysusages
where dbid = db_id('tempdb')
```

db_id 関数は、データベースの ID 番号を返します。この場合、tempdb のデータベース ID が返されます。

7 データとログが区切られている場合、tempdb の最初の 2MB をデータとログに設定しなおします。

```
update sysusages
set segmap = 7 where dbid = db_id('tempdb')
and lstart = 0
```

8 tempdb に属する他のすべてのローを sysusages から削除します。影響されるローの数は、以前の select コマンドに影響されるローの数より 1 つ少ない必要があります。

```
delete sysusages where dbid = db_id('tempdb')
and lstart != 0
```

**警告！** Adaptive Server を停止して再起動するたびに、model データベースは tempdb にコピーされます。したがって、model データベースがデフォルト・サイズを超えている場合、tempdb のサイズが model よりも小さくなるように縮小しないでください。

9 tempdb に次のようなエントリがあることを確認します。

```
select * from sysusages
where dbid = db_id('tempdb')
```

10 情報が正しい場合は、手順 10 に進んでトランザクションをコミットします。

問題がある場合は、次のコマンドを入力して変更を取り消します。

```
rollback transaction
```

プロシージャを継続しないでください。実行した手順を確認して、問題の原因を判別します。

11 トランザクションを完了します。

```
commit transaction
```

12 Adaptive Server を再設定して、システム・カタログへの変更を禁止します (Adaptive Server の通常状態)。

```
sp_configure "allow updates", 0
```

13　checkpoint を即座に発行して、Adaptive Server を停止します。

**警告！** tempdb を再度変更する前に、Adaptive Server を停止する必要があります。停止および再起動せずに実行を継続する場合、tempdb に深刻なエラーが発生します。

```
checkpoint
go
shutdown
go
```

14　Adaptive Server を再起動する。

# *disk reinit* と *disk refit* によるシステム・テーブルのリストア

master データベースをダンプからリストアするときに、最近実行した disk init コマンド、または create database コマンドや alter database コマンドがそのダンプに反映されていない場合は、この項で説明する手順に従って、sysusages テーブル、sysdatabases テーブル、sysdevices テーブル内の正しい情報をリストアしてください。

## *disk reinit* による *sysdevices* のリストア

最後のダンプ以降にデータベース・デバイスを追加している場合 (つまり disk init コマンドを発行した場合) は、disk reinit コマンドを実行して新しいデバイスをそれぞれ sysdevices に追加する必要があります。最初に disk init コマンドを実行したときのスクリプトが保存されている場合は、そのスクリプトを基に disk reinit コマンドのパラメータを決定してください (vstart の元の値を含む)。指定したサイズが小さすぎるか、vstart に指定した値が異なっている場合は、データベースが破壊されることがあります。

disk init のスクリプトが保存されていない場合は、sysdevices の最新のハードコピーを調べて disk reinit の正しいパラメータを決定してください。元の disk init コマンドで vstart に独自の値を指定した場合でも、vstart の元の値を確認しておく必要があります。

表 14-1 は、disk reinit のパラメータおよび対応する sysdevices のデータを示します。

**表 14-1: sysdevices の使用による disk reinit パラメータの決定**

| *disk reinit* パラメータ | *sysdevices* データ | 注意 |
|---|---|---|
| name | name | 特に、スクリプトを使用してデータベースを作成または変更、あるいはセグメントを追加する場合は、同じ名前を使用する。 |
| physname | *phyname* | デバイスへのフル・パス名。すべての相対パスは、サーバの現在の作業ディレクトリが基準になる。 |
| vdevno | vdevno | 既に使用されていない値を選択する。 |
| size | (*high* -*low*) +1 | 正確なサイズ情報を指定する必要がある。 |

また、エラー・ログから *name*、*physname*、*vdevno* の値を読み取ってデバイスの情報を取得し、オペレーティング・システムのコマンドを使用してデバイスのサイズを判断するという方法もあります。

sybsystemprocs データベースを別の物理デバイス上に保管する場合に、この時点でそのデバイスが sysdevices に登録されていないときは、必ず sybsystemprocs に対応する disk reinit コマンドも実行してください。

disk reinit を実行したら、sysdevices テーブルを、dataserver の実行前に作成したコピーと比較してください。

disk reinit コマンドは、master データベースからのみ実行できます。このコマンドを実行できるのはシステム管理者だけです。パーミッションを他のユーザに譲渡することはできません。

『システム管理ガイド 第 1 巻』または『リファレンス・マニュアル：コマンド』の「第 7 章 データベース・デバイスの初期化」を参照してください。

## *disk refit* による *sysusages* と *sysdatabase* のリストア

最後のデータベース・ダンプ以降に、データベース・デバイスを追加したり、データベースを作成または変更したりした場合は、disk refit を使用して sysusages テーブルと sysdatabases テーブルを再構築してください。

disk refit は、master データベースからのみ実行できます。このコマンドを実行できるのはシステム管理者だけです。パーミッションを他のユーザに譲渡することはできません。構文は次のとおりです。

```
disk refit
```

disk refit によってシステム・テーブルが再構築されると、Adaptive Server は自動的に停止します。disk refit の実行中と停止処理中の出力を参照して、エラーが発生したかどうかを調べてください。

**警告！** disk reinit コマンドに指定した情報が正確でない場合は、データの更新時に永久的な破壊が発生することがあります。disk refit の実行後は、dbcc を使用して Adaptive Server の検査を行ってください。

第 15 章

# アーカイブ・データベースへのアクセス



アーカイブ・データベースへのアクセス機能により、データベース管理者は、従来の読み取り専用データベース (このタイプのデータベースは「アーカイブ・データベース」と呼ばれます) と同様の方法でダンプにアクセスし、データベース・ダンプ (アーカイブ) のデータを検証したり、選択的にリカバリしたりできます。

従来のデータベースとは異なり、アーカイブ・データベースは実際のデータベース・ダンプを主要なディスク記憶領域デバイスとして使用し、従来の記憶領域を最小限にすることによって、データベース・ダンプのリカバリの結果生成される新しいページまたは変更済みのページを表します。データベース・ダンプには、既に多くの ( またはほとんどの ) データベース・ページのイメージが含まれているため、Backup Server を使用してページをアーカイブから従来のデータベース記憶領域に転送しなくてもアーカイブ・データベースをロードできます。その結果、従来のデータベースよりもロードが大幅に速くなります。

# 概要

アーカイブ・データベースのアクセスでは、さまざまな操作をデータベース・ダンプに対して直接実行できます。

従来のデータベースをロードするには、ソース・データベースのサイズと同じか、それ以上の記憶領域が必要になります。Backup Server を使用してデータベース・ダンプをロードするには、データベース・ダンプから、従来のデータベースとは別に設定されたストレージにページをコピーすることが必要です。

対照的に、アーカイブ・データベースは最小限の従来のディスク記憶領域を使用して作成できます。アーカイブ・データベースをロードするとき、データベース・ダンプ内に存在するページは Backup Server によりコピーされません。代わりに、アーカイブ内のページの「論理ページから仮想ページへの」マッピングを表すマップが Adaptive Server により作成されます。これにより、データベース・ダンプのデータの表示に必要な時間が大幅に短縮され、ダンプのロードに必要な記憶領域のサイズも減少します。

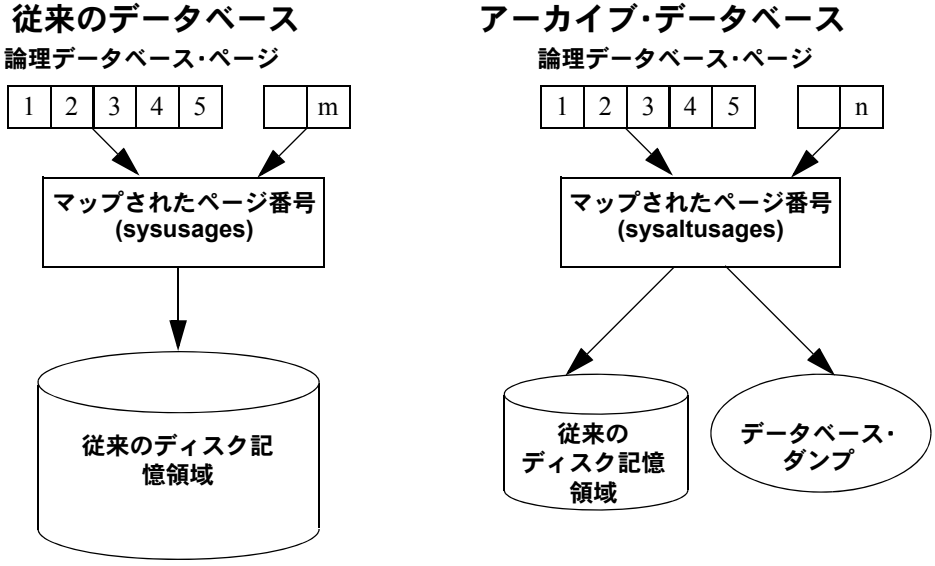
アーカイブ・データベースでは、元のデータベース全体をコピーする必要はありません。sp_dumpoptimize を使用してデータベースをダンプするときに行われる最適化に応じて、データがアーカイブ・データベースに完全に格納される (データベースのすべてのページがデータベース・ダンプに格納される) か、部分的に格納されます (割り付けられたページだけがデータベース・ダンプに格納される)。

データベース・ダンプは、読み込み専用のデータベースとして存在するため、データベース管理者は次のような使いやすいツールと手法を使用してクエリを実行できます。

- 運用データベースからダンプされた最新のコピーに対する、データベース一貫性検査。これらの検査を別のサーバで実行することにより、運用環境におけるリソースの競合を避けることができます。リソースを考慮する必要のない場合は、アーカイブが作成されたのと同じサーバでアーカイブを検査できます。アーカイブでの検証により、リストア操作の実行前に必要な保証を得られます。
- データベース・ダンプの整合性が問題となる場合、アーカイブ・データベースにデータベース・ダンプをロードすると問題ないかどうかを簡単にテストできるため、従来のデータベースのリストアに使用される適切なデータベース・ダンプを特定するのに良いツールとなります。
- データベース・ダンプからのオブジェクト・レベルのリストア。失われたデータは、select into を使用してアーカイブ・データベース内のテーブルからリストア対象のローをコピーすることにより、リカバリされます。select into 操作は、アーカイブ・データベースを持つサーバで直接実行するか、オブジェクトのリストアが必要なサーバとは別のサーバでアーカイブ・データベースが使用可能な場合はコンポーネント統合サービス・プロキシ・テーブルを使用することにより実行できます。

また、トランザクション・ログをアーカイブ・データベースにロードでき、それによってリストア・オペレーションを実行する際に同じロード・シーケンスを適用できます。図 15-1 は、アーカイブ・データベースと従来のデータベース構造の差異を示します。

**図 15-1: アーカイブ・データベースのコンポーネント**




## アーカイブ・データベースのコンポーネント

アーカイブ・データベースは、連携して機能するこれらのコンポーネントで構成されており、データベース・ダンプが従来のデータベースとして機能しているかのように見せかけることができます。

- データベース・ダンプ (アーカイブ)
- 変更済みページ・セクションの格納に使用される従来のオプションのディスク記憶領域
- sysaltusages テーブルを持つスクラッチ・データベース

### データベース・ダンプ

読み込み専用のデータベース・ダンプは、ほとんどの変更されていないページのレポジトリとして使用されます。

データベース・ダンプに変更を加えることはできません。ダンプ内のデータに加えた変更はすべて、変更済みページ・セクションに格納されます。

Adaptive Server では、このデータベース・ダンプとストライプは、アーカイブ・データベースによってのみ使用可能なデータベース・デバイスと見なされます。

## 変更済みページ・セクション

データベース・ダンプには、特定の時間のデータベースのスナップショットが反映されます。データベース・ダンプを表すアーカイブ・データベースは読み込み専用です。ユーザ・トランザクションは実行できませんが、ある種の変更については実行可能です。次に例を示します。

- アーカイブ・データベースがソース・データベースと整合性を保つようにリカバリを実行できます。
- テーブルの修正バージョンをリストアするための修正を実行する dbcc コマンドを使用できます。

これらの変更済みデータベース・ページ、および新しく割り付けられたデータベース・ページはデータベース・ダンプとそのストライプ内に格納できないため、アーカイブ・データベースには従来のデータベース記憶領域が若干必要です。このディスク領域は変更済みページ・セクションと呼ばれ、create archive database コマンドおよび alter database コマンドを使用して割り付けることができます。

変更済みページ・セクションは、次の 2 つのセグメントに分割されます。

- 処分可能な変更セグメントには、リカバリ再実行パスによって変更されているか割り付けられているページ、またはいつでも変更されているか割り付けられているログ・ページが格納されます。処分可能な変更セグメントへの入力は 1 ページに 1 つのみです。
- 永続的変更セグメントには、その他のページの変更または割り付けが格納されます。永続的変更セグメントへの入力は 1 ページに 1 つのみです。

変更済みページ・セクションのサイズを変更する場合、sysusages ローはマスター・データベースで更新されます。

## sysaltusages テーブルとスクラッチ・データベース

sysaltusages は、アーカイブ・データベースのページ番号を、データベース・ダンプとそのストライプ内、または変更済みページ・セクション内の実際のページにマップするために使用されるデータオンリーロック・テーブルです。しかし、従来のデータベースの sysusages テーブルとは異なり、sysaltusages テーブルはデータベースのすべての論理ページをマップするわけではありません。以下のページのみマップします。

- データベース・ダンプに格納されたページ
- 変更されたために、変更済みページ・セクションに移動されたページ

『リファレンス・マニュアル：テーブル』を参照してください。

---

**注意** sysaltusages はロー・ロック・カタログであるため、reorg を定期的に使用して、論理的に削除された領域を再利用する必要があります。

---

スクラッチ・データベースには、sysaltusages テーブルが格納されます。スクラッチ・データベースは、sysaltusages テーブルを柔軟に配置するために使用されます。

スクラッチ・データベースは、任意のデータベースにすることができます (master データベースや**テンポラリ**・データベースなどの例外を除く)。以下の理由から、スクラッチ・データベース専用のデータベースを使用することをおすすめします。

- sysaltusages のサイズが、サポートされるアーカイブ・データベースの数によって変化する場合があります。データベースのサイズを小さくすることはできないが、大きすぎる場合はそれを削除して、必要に応じて再作成できます。
- データベース・ログが自動的にトランケートされるように、trunc log on checkpoint を有効にできます。

sysaltusages テーブルを持つ点を除けば、このデータベースは他のデータベースと同様です。スレッショルド・プロシージャと他の領域管理メカニズムを使用して、データベース内の領域を管理できます。

スクラッチ・データベースとして使用するデータベースを指定するには、次のように入力します。

```
sp_dboption <db name>, "scratch database", "true"
```

各アーカイブ・データベースに割り当てることのできるスクラッチ・データベースは一度に 1 つのみですが、複数のデータベースが同じスクラッチ・データベースを使用することはできます。多数のアーカイブ・データベースがある場合は、複数のスクラッチ・データベースを定義できます。

## アーカイブ・データベースの操作

アーカイブ・データベースで従来のデータベース・オペレーションを数多く実行できます。ただし、データベースを変更するユーザ定義トランザクションやコマンド (insert、update、delete など) は実行できません。

データが格納されたアーカイブ・データベースは、sp_dboption を使用して readonly オプションが適用された読み込み専用データベースと同様です。

### アーカイブ・データベースのアクセスの DDLGen サポート

すべてのアーカイブ・データベースの DDL を生成するには、拡張フィルタ・オプション “OA” を使用します。

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -N% -XOA
```

1 つのアーカイブ・データベースの DDL を生成するには、標準的なデータベースの構文を使用します。次の例では、アーカイブ・データベース archivedb の DDL を作成します。

```
ddlgen -Uroy -Proy123 -SHARBAR:1955 -TDB -Narchivedb
```

# アーカイブ・データベースの設定

アーカイブ・データベースの作成には、create archive database を使用します。『リファレンス・マニュアル：コマンド』を参照してください。

## 変更済みページ・セクションのサイズ設定

変更済みページ・セクションは、変更されたデータベース・ページまたは新しく割り付けられたデータベース・ページの格納に使用されます。これらのページは永続的変更セグメント、処分可能な変更セグメント、またはその両方に格納されます。

- ページは永続的変更セクションに 1 度だけ再マップできます。
- ページは処分可能な変更セクションに 1 度だけ再マップできます。
- リカバリは、ほとんどのページ再マップにおいて重要な位置を占めます。
- dbcc checkalloc にも、有効な領域割り付けが必要です。
- 変更済みページ・セクションのサイズは、alter database コマンドを使用して増やすことができます。ただし、変更済みページ・セクションのサイズを小さくするには、アーカイブ・データベースを削除して再作成する必要があります。

永続的変更セグメントと処分可能な変更セグメントは理論上異なります。永続的変更セグメントは、永続的変更セグメントに対して segmap を含む sysusages フラグメントのセットによって定義されます。処分可能な変更セグメントは、処分可能な変更セグメントに対して segmap を含む sysusages フラグメントのセットによって定義されます。処分可能な変更セグメントは、各 load tran コマンドの最初に廃棄されます。

---

**注意** Adaptive Server は、永続的変更セグメントと処分可能な変更セグメント間の変更済みページ・セクションの領域の分割を自動的に管理します。このように自動的にサイズが変更されると、sysusages ローは master データベースで更新されます。

---

変更済みページ・セクションの最小サイズは、データベースで変更されたページの数または新しく割り付けられたページの数によって決まります。これらのページの多くは、リカバリの再実行およびリカバリの取り消しによって変更されます。

変更済みページの数を最小限にし、変更済みページ・セクションに必要な領域の量を減らすには、load database with norecovery コマンドを使用します。ただし、これを行うことには、マイナス面もあります。

**注意** dbcc checkalloc は、nofix オプションを使用する場合でも、変更済みページ・セクションの領域を大量に消費します。dbcc checkalloc を実行すると、すべてのアロケーション・ページ (256 ページごと) に情報が書き込まれます。これらのアロケーション・ページの変更結果は、変更済みページ・セクションに格納されます。つまり、dbcc checkalloc を使用する場合は、少なくとも元のデータベースの 256 分の 1 のサイズの変更済みページ・セクションが必要です。

変更済みページ・セクションに十分な領域がない場合、領域が必要なコマンドがサスペンドされ、次のようなエラーが表示されます。

```
There is no more space in the modified pages section for
the archive database <database name>. Use the ALTER
DATABASE command to increase the amount of space
available to the database.
```

変更済みページ・セクションの領域を増やすには、次のいずれかの操作を行います。

- alter database を使用して、変更済みページ・セクションのサイズを大きくします。
- 変更済みページ・セクションに追加の領域を割り付けない場合は、[Ctrl+C] を押して現在のコマンドをアボートします。

**注意** スレッショルドを使用して、変更済みページ・セクションの領域を管理することはできません。

## 変更済みページ・セクションに割り付けられた領域の拡張

alter database を使用すると、アーカイブ・データベースの変更済みページ・セクションの領域を追加できます。変更済みページ・セクションの領域を増やすと、サスペンドされたコマンドの処理を再開できます。

alter database は、領域がなくなったときに限らず、いつでも実行して変更済みページ・セクションのサイズを大きくすることができます。

## アーカイブ・データベースのマテリアライズ

アーカイブ・データベースは、データベース・ダンプとともにロードされたときにだけ有用なプレースホルダです。ロード処理では、実際にはページはコピーされませんが、ページ・マッピングを使用してデータベースがマテリアライズされます。

アーカイブ・データベースをマテリアライズするには、load database コマンドを使用します。

---

**注意** load database ... norecovery は、アーカイブ・データベースのアクセスを可能にするために、Adaptive Server バージョン 12.5.4 と 15.0.2 で導入されました。従来のデータベースでは、norecovery を使用できません。

データベース・ダンプをアーカイブ・データベースにロードするときに、Backup Server が実行されている必要はありません。

---

### *load database with norecovery* の使用

load database コマンドの with norecovery オプションを使用すると、リカバリを行わずにデータベース・ダンプをアーカイブ・データベースにロードすることで、ロードに必要な時間を減らすことができます。多くのデータベース・ページは、リカバリ中に変更や割り付けを行い、変更済みページ・セクションに格納することができます。そのため、リカバリを省略すると、消費される変更済みページ・セクションの領域が最小限に抑えられます。with norecovery オプションを使用すると、アーカイブ・データベースをすばやく表示できます。

with norecovery を使用した場合、データベースは自動的にオンラインになります。

ただし、リカバリが必要なデータベースに load database with norecovery を使用すると、トランザクションの一貫性と物理的な一貫性が保たれないままになる可能性があります。物理的な一貫性がないデータベースで dbcc 検査を実行すると、エラーが発生することがあります。

with norecovery を使用してアーカイブ・データベースをロードした場合、アーカイブ・データベースを使用するには sa_role またはデータベース所有者の権限が必要です。

## アーカイブ・データベースでの論理デバイスの使用

sp_addumpdevice を使用すると、アーカイブ・データベースをロード可能な論理デバイスを作成できます。

sp_addumpdevice コマンドを実行すると、load database コマンドの *dump_device* または *stripe_device* として、*physical_name* の代わりに *logical_name* を使用できるようになります。

---

**注意** 従来のデータベースへのロードのデバイス指定として、または従来のデータベースをダンプする際に、アーカイブ・データベース論理デバイスは使用できません。

---

## アーカイブ・データベースでの *load database* の制限

load database をアーカイブ・データベースに対して使用する場合は、次の制限があります。

- アーカイブ・データベースのデータベース・ダンプは、ローカル・マシンにマウントされたファイル・システムのディスク・ダンプでなければなりません。これは、ローカル記憶領域または NFS 記憶領域です。load database ... at *remote_server* 構文はサポートされず、テープ上のデータベース・ダンプもサポートされません。
- 異なるアーキテクチャ間のロードはサポートされません。バイト順序を合わせるために、データベース・ダンプおよび load database コマンドは同じアーキテクチャで実行する必要があります。
- ダンプしたデータベースのページ・サイズは、アーカイブ・データベースをホストするサーバが使用するページ・サイズと同じにします。
- ダンプが行われたサーバのメジャー・バージョンは、アーカイブ・データベースをホストするサーバのメジャー・バージョン以前のバージョンでなければなりません。
- データベース・ダンプが行われたサーバの文字セットおよびソート順は、アーカイブ・データベースをホストするサーバの文字セットおよびソート順と同じでなければなりません。

# アーカイブ・データベースのオンライン化

アーカイブ・データベースをオンラインにするには、online database を使用します。

online database は、変更済みページおよび割り付けられたページを変更済みページ・セクションに再マップ可能なときに、リカバリの取り消しを実行します。

データベースをロードすると、リカバリの取り消しパスを実行しなくても自動的にオンラインになるため、with norecovery を使用してデータベースをロードした場合はデータベースをオンラインにする必要はありません。

### アーカイブ・データベースへのトランザクション・ログのロード

アーカイブ・データベースにトランザクション・ログをロードするには、load tran を使用します。

アーカイブ・データベースにトランザクション・ログをロードすると、load tran はリカバリ再実行パスを実行します。変更済みのデータベース・ページと新しいデータベース・ページは、永続的変更セグメントに書き込まれます。これらの変更に対応するために、変更済みページ・セクションに十分な領域が必要です。必要に応じて、alter database を使用して、変更済みページ・セクションに領域を増やし、アーカイブ・データベースに割り付けられた通常のデータベース領域を増やします。

従来のデータベースとは異なり、アーカイブ・データベースは、ロード・シーケンスをブレークしないでロード・シーケンス中にオンラインにできます。従来のデータベースでは、ロードしてから for standby_access 句を使用せずにオンラインにすると、ロード・シーケンスで次のトランザクション・ログをロードできなくなりますが、アーカイブ・データベースでは、for standby_access 句以降を使用せずにオンラインにし、ロード・シーケンスで次のトランザクション・ログを使用してロードできます。これによって、ロード・シーケンス中にいつでも一貫性チェックの実行といった読み込み専用のオペレーションを実行できます。トランザクション・ログをアーカイブ・データベースにロードする際に、Adaptive Server は処分可能な変更セグメントを変更済みページ・セクションから自動的に削除します。こうすることで、効率的にアーカイブ・データベースを前回のロードが実行された後の状態に戻し、シーケンスの次のトランザクション・ログをロードできるようにします。

### アーカイブ・データベースの削除

アーカイブ・データベースの削除には、drop database を使用します。

アーカイブ・データベースを削除すると、そのデータベースのすべてのローが、スクラッチ・データベースの sysaltusages テーブルから削除されます。これにより、スクラッチ・データベースにログ領域が必要になります。

## アーカイブ・データベースの使用

この項では、アーカイブ・データベースで実行するコマンドについて説明します。

## アーカイブ・データベースでの SQL コマンドの使用

既に説明したコマンド (alter database、load database、online database、drop database、および load tran) に加えて、アーカイブ・データベースでは次の SQL コマンドを使用できます。

- use
- select
- select into − ターゲット・データベースがアーカイブ・データベースでない場合に使用します。
- 読み込みを実行するカーソル処理。次のコマンドが含まれます。
  - declare cursor
  - deallocate cursor
  - open
  - fetch

  更新可能カーソルは使用できません。
- checkpoint − サポートされるコマンド。ただし、アーカイブ・データベースはチェックポイント処理により自動的にチェックポイントを実行されません。
- execute − アーカイブ・データベースを参照する文がアーカイブ・データベース内で使用可能であれば、実行できます。ストアド・プロシージャ内部または外部でのトランザクションは、execute コマンドでは使用できません。
- lock table
- readtext

---

**注意** insert、update、および delete などの DML コマンドは使用できず、ユーザ・トランザクションを開始できません。

---

## アーカイブ・データベースでの dbcc コマンドの使用

アーカイブ・データベースでは、次の dbcc コマンドを使用できます。

- checkdb
- checkcatalog

**注意** checkcatalog の fix バージョンはサポートされていません。

- checktable
- checkindex
- checkalloc
- checkstorage
- indexalloc
- tablealloc
- textalloc

dbcc コマンドの実行中は、その他のユーザはアーカイブ・データベースにアクセスできません。dbcc コマンドの実行中にアーカイブ・データベースにアクセスしようとすると、データベースがシングルユーザ・モードになっていることを示すメッセージが表示されます。

上記の dbcc コマンドの変形は、アーカイブ・データベースがオンラインでもオフラインでも使用できます。ただし、fix オプションが設定された dbcc コマンドは、オンラインのアーカイブ・データベースでのみ使用できます。

## 一般的なアーカイブ・データベース・コマンド・シーケンス

次の構文は、一般的なアーカイブ・データベース・コマンド・シーケンスです。

まず、必要な場合はスクラッチ・データベースを作成します。

```
create database scratchdb
      on datadev1 = 100
      log on logdev1 = 50
```

これにより、scratchdb という名前の 150MB の従来のデータベースが作成されます。

上の手順で作成したデータベースをスクラッチ・データベースとして指定します。

```
sp_dboption "scratchdb", "scratch database", "true"
```

アーカイブ・データベースを作成します。

```
create archive database archivedb
      on datadev2 = 20
      with scratch_database = scratchdb
```

これにより、20MB の変更済みページ・セクションが含まれる、archivedb という名前のアーカイブ・データベースが作成されます。

load database を使用して、アーカイブ・データベースをマテリアライズします。

```
load database archivedb
      from "/dev/dumps/050615/proddb_01.dmp"
      stripe on "/dev/dumps/050615/proddb_02.dmp"
```

データベースをオンラインにします。

```
online database archivedb
```

アーカイブ・データベースの一貫性を検査します。次に例を示します。

```
dbcc checkdb(archivedb)
```

load tran を使用してトランザクション・ログ・ダンプをロードし、select into または bcp を使用してアーカイブ・データベースからオブジェクトをリストアします。

```
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_01.dmp"
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_02.dmp"
online database archivedb
select * into proddb.dbo.orders from
      archivedb.dbo.orders
load tran archivedb
      from "/dev/dumps/050615/proddb1_log_03.dmp"
online database archivedb
```

# アーカイブ・データベースの圧縮ダンプ

アーカイブ・データベースの圧縮ダンプを使用するには、次の操作を行う必要があります。

- dump database または dump tran コマンドの with compression = <compression level> オプションを使用して、圧縮ダンプを作成します。
- アーカイブ・データベースにアクセスするためのメモリ・プールを作成します。

---

**注意** compress:: を使用して生成されたダンプは、アーカイブ・データベースにロードできません。そのため、この章で圧縮に言及している場合は、with compression = <compression level> オプションを使用して生成されたダンプのことを指しています。

---

## 圧縮メモリ・プールの作成

Adaptive Server は、圧縮ダンプからページを読み込むとき、ダンプから圧縮されたブロックを選択してその圧縮を解除し、必要なページを抽出します。Adaptive Server での圧縮解除は、特別メモリ・プールの大容量バッファを使用して行われます。次のコマンドを使用してプールのサイズを設定します。

```
sp_configure 'compression memory size', size
```

これは、動的な設定パラメータで、サイズは 2KB のページで指定されます。サイズが 0 に設定された場合、プールは作成されず、圧縮ダンプをロードできません。

プールに最適なサイズを判断するには、次の 2 つの要素について考慮します。

- Backup Server により使用されるブロック I/O。デフォルトでは、このブロック I/O は 64KB だが、dump database コマンドで with blocksize オプションを使用して変更できます。
- すべてのアーカイブ・データベース内のブロックを圧縮解除する「同時」ユーザの数。各同時ユーザには、ブロック I/O と同じサイズごとに 2 つのバッファが必要です。

絶対最小値として、アーカイブ・データベースごとに 1 人の同時ユーザ、2 つのバッファを許可します。

# アーカイブ・データベースのアップグレードとダウングレード

この項では、アーカイブ・データベースのアップグレードおよびダウングレードの方法について説明します。

## アーカイブ・データベースが含まれる Adaptive Server のアップグレード

アーカイブ・データベースはアップグレードできません。データベース・ダンプを以前のバージョンの Adaptive Server から、新しいバージョンの Adaptive Server にホストされているアーカイブ・データベースにロードする場合、online database を実行するとデータベースは内部的にアップグレードされません。

アーカイブ・データベースが含まれる Adaptive Server をアップグレードした場合、アーカイブ・データベースを除くすべてのデータベースがアップグレードされます。アーカイブ・データベースは、古いバージョンの Adaptive Server のままになります。

すでにアップグレードされたデータベースから生成されたダンプを使用して、アーカイブ・データベースを再ロードすることをおすすめします。

Adaptive Server のアップグレード方法の詳細については、使用しているプラットフォームの『インストール・ガイド』を参照してください。

## アーカイブ・データベースが含まれる Adaptive Server のダウングレード

アーカイブ・データベースがサポートされないバージョンの Adaptive Server にダウングレードするときは、次の点に注意してください。

- アーカイブ・データベースを含む Adaptive Server を、アーカイブ・データベースがサポートされないバージョンの Adaptive Server にダウングレードする場合は、ダウングレードする前にアーカイブ・データベースを削除することが推奨されます。

  新しい sysaltusages テーブルを消去するには、ダウングレード手順を実行する前にスクラッチ・データベースを削除します。スクラッチ・データベースが削除された場合は、sysaltuages により問題が発生することはありません。

- Backup Server バージョン 15.0 ESD #2 以降は、ダンプをアーカイブ・データベースにロードできるように、圧縮を新しいフォーマットで書き込みます (with compression = *compression_level*)。そのため、アーカイブ・データベースのアクセスがサポートされないバージョンの Adaptive Server に圧縮ダンプをロードする必要がある場合は、同じバージョンの Backup Server を使用して圧縮データベース・ダンプを作成しロードします。以前のバージョンの Backup Server では、新しいフォーマットの圧縮データベース・ダンプはサポートされません。

圧縮せずにダウングレードする場合は、Backup Server に関する注意は不要です。

### 圧縮ダンプの互換性の問題

- “compress::” を使用して生成されたダンプは、アーカイブ・データベースにロードできません。従来のデータベースでこの圧縮オプションを使用する場合、ダンプに関する互換性の問題は発生しません。
- Backup Server 15.0 ESD #2 では、with compression = *compression_level* オプションを使用して生成された圧縮ダンプのフォーマットが変更されました。そのため、次の結果が生じます。
  - Backup Server バージョン 15.0 ESD #2 以降を使用して作成された圧縮済みダンプは、Backup Server バージョン 15.0 ESD #2 以降を使用した 15.0 ESD #2 以前のインストールにだけロードできます。
  - 15.0 ESD #2 以前のインストールを使用中にダンプをアーカイブ・データベースに使用する場合、Backup Server バージョン 15.0 ESD #2 以上を使用して圧縮済みデータベース・ダンプを作成します。

  **注意** Backup Server バージョン 15.0 ESD #2 以降は、15.0 ESD #2 の圧縮フォーマットとそれより前の圧縮フォーマットの両方を認識できるため、どちらのダンプとロードにも 15.0 ESD #2 Backup Server を使用できます。

## アーカイブ・データベースの制限

アーカイブ・データベースには、次の制限があります。

- アーカイブ・データベースは読み込み専用です。
- コマンドとストアド・プロシージャを実行するパーミッション、およびアーカイブ・データベースのオブジェクトへのアクセス権は、同じサーバ上の同じデータベース・ダンプとともにロードされた従来のデータベースのものと同じです。
- with norecovery を使用してアーカイブ・データベースがロードされる場合、そのデータベースへのアクセス権は sa_role を持つユーザまたはデータベース所有者に制限されます。
- インメモリ・データベースはアーカイブ・データベースとして使用できません。さらに、インメモリ・データベースはスクラッチ・データベースとしても使用しないことをおすすめします。

- サーバ全体をマイグレートする場合、sybmigrate を実行してもアーカイブ・データベースはマイグレートされません。
- sybmigrate は、アーカイブ・データベースがマイグレーション対象として明確に選択されている場合のみ、アーカイブ・データベースをマイグレートします。アーカイブ・データベースをターゲット・サーバにマイグレートすると、sybmigrate によりターゲット・サーバにはアーカイブ・データベースではなく従来のデータベースが作成されます。
- アーカイブ・データベースに変更を加えるコマンド (dbcc コマンドなど ) を実行すると、アーカイブ・データベースは自動的にシングルユーザ・モードになります。
- アーカイブ・データベースは、ディスク上のデータベース・ダンプまたはトランザクション・ログ・ダンプのみを使用します。テープ・ダンプはサポートされません。
- データベース・ダンプまたはトランザクション・ログ・ダンプは、アーカイブ・データベースをホストするサーバから確認する必要があります。リモート・ダンプはサポートされません。
- アーカイブ・データベースが圧縮ダンプにアクセスするには、ダンプが compress:: オプションではなく with compression オプションを使用して作成されている必要があります。
- チェックポイント処理では、アーカイブ・データベースに対してチェックポイントが自動的に実行されません。アーカイブ・データベースにチェックポイントを実行するには、checkpoint コマンドを使用します。
- データベース・リカバリ・シーケンスで、sp_dbrecovery_order を使用してアーカイブ・データベースを指定することはできません。アーカイブ・データベースは、*dbid* 順の最後にリカバリされます。
- ページがアーカイブ・データベースにキャッシュされると、キャッシュされたページは、サーバと同じページ・サイズでメモリ・プール内にとどまります。つまり、2K のサーバでは、ページは常に 2K のプールにキャッシュされます。16K のサーバでは、ページは常に 16K のプールにキャッシュされます。
- アーカイブ・データベースおよびデータベース内のオブジェクトは、ユーザ定義キャッシュにバインドすることはできません。アーカイブ・データベース内のオブジェクトのデフォルトは、デフォルトのデータ・キャッシュに設定されます。
- disk resize は、アーカイブ・データベースにより使用されるデバイス、またはデータベース・ダンプやトランザクション・ログにマップされるデバイスでは機能しません。

- disk refit を実行しても、master データベースの sysusages エントリは、アーカイブ・データベースによって使用されるデバイスから再構築されません。これは、ダンプ・デバイスと、変更済みページ・セクションに使用されるデバイスの両方に該当します。ただし、アーカイブ・データベースの既存の sysusages エントリは残ります。
- アーカイブ・データベースは複写できません。
- アーカイブ・データベースは、高可用性サーバではフェイルオーバされません。
- アーカイブ・データベースでは、空き領域スレッショルドを確立できません。

第 16 章

# データベースの自動拡張



データベース領域が不足したときに、自動的に拡張するようにデータベースを設定できます。

自動データベース拡張のストアド・プロシージャ sp_dbextend を使用してスレッショルドをインストールすることにより、空き領域を持つデバイスを特定し、それに応じてそのデバイス上のデータベース、およびスレッショルドに達したセグメントを変更できます。

自動拡張を行うようにデータベースをセットアップした後は、データベースのサイズが拡大してその空き領域スレッショルドに達すると、内部メカニズムが起動し、拡張ポリシーで指定された大きさの領域がデータベースに追加されます。この自動拡張プロセスは、データベースにバインドされたすべてのデバイスの空き領域の大きさを調べます。デバイスに十分な空き領域がある場合は、データベースは拡張を続けます。デフォルトでは、デバイスのサイズが 40MB を超えると、データベースのサイズが 10 パーセント増加します。40MB より小さいデータベースのサイズは 4MB ずつ増加します。サイトのニーズに応じて、データベースのサイズ変更の限界を指定できます。

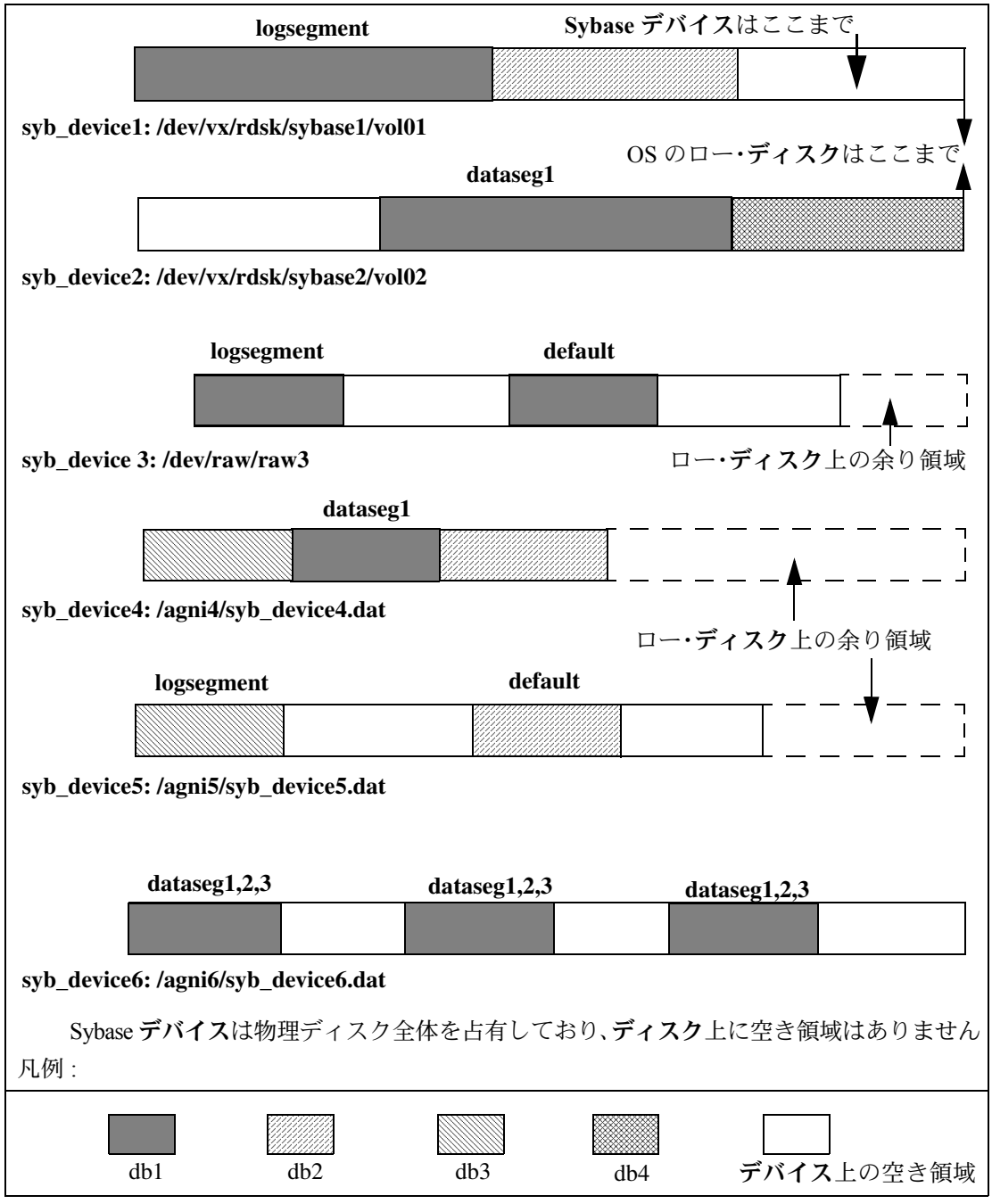
拡張用に設定されているデバイスがある場合は、次にそのデバイスが拡張に使用されます。最終的に、拡張されたデータベースはこれらのデバイス上に存在することになります。

この自動拡張プロセスは、バックグラウンド・タスクとして実行され、情報メッセージがサーバのエラー・ログに生成されます。

# ディスク、デバイス、データベース、セグメントについて

図 16-1 (455 ページ) に、一連の disk init、create database、alter database を実行した後の Adaptive Server インストール環境の物理リソースのレイアウトとして考えられるものをいくつか示します。ストアド・プロシージャのテストにおいて、物理的および論理的な領域レイアウトの計画を複数立案するときに、この情報を参考にしてください。

**図 16-1: データベースとデバイスのレイアウト**




Sybase デバイスは物理ディスク全体を占有しており、ディスク上に空き領域はありません

ロー・ディスクの一部を Sybase のデバイスが占有することがあります。syb_device2 では、ロー・ディスク全体を単一の Sybase デバイスが占有しており、このデバイス上に複数のデータベースが作成されています。このロー・ディスク (*/dev/vx/rdsk/sybase2/vol02*) 上には、デバイスの先頭部分に空き領域が残っていますが、これは、以前この領域を占有していたデータベースが削除された場合に発生します。

syb_device4 と syb_device5 は、ファイル・システム・ディスク上にある Sybase デバイス */agni4/syb_device4.dat* と */agni5/syb_device5.dat* のレイアウトを示します。これらのデバイスは、ディスクの一部を占有していますが、まだ空き領域があり、デバイス (たとえば 1 つのオペレーティング・システム・ファイル) の拡張は可能です。

syb_device6 に示す Sybase デバイスは、物理ディスク上の使用可能な領域全体を占有していますが、デバイス上には未使用の領域が残っています。この未使用領域を使用して、既存のデータベースをこのデバイス上に拡張できます。

これらのデバイスはいずれも、複数のデータベースのフラグメントを格納しています。特定のデータベースに対応するデバイスのそれぞれに、そのデバイス全体に広がるセグメントが 1 つ以上あります。

syb_device6 (*/agni6/syb_device6.dat*) では、db1 がこのデバイスの 3 つの部分を占有しています。このデバイスは、3 つの異なるセグメント、つまりデータ・セグメント 1、2、3 にも属しています。sysusages 内のこのデータベース db1 に対応する 3 つのエントリはいずれも、そのセグメント・マップ値にこの 3 つのセグメントが含まれています。

ただし、*/dev/raw/raw3* 上のデバイス syb_device3 では、データベースはこのデバイスの 2 つの部分を占有しており、1 つはログ・セグメント専用で、もう 1 つはデフォルト・セグメントとなっています。最初の create database コマンドが実行済みであるときに、次の SQL コマンドを実行するとこのような結果になります。

```
alter database db1 on syb_device3 = "30M"
alter database db1 log on syb_device3 = "10M" with
      override
```

最初の alter database コマンドはデータベースのデフォルト・セグメント部分を作成し、2 番目のコマンドはデータベースの logsegment 部分を作成します。両方の部分が同じデバイス上にありますが、強制的にこれらを混在させます。データベースの領域は、個々のセグメント内で拡張されます。

# スレッショルド・アクション・プロシージャ

データベースの拡張は、セグメントの空き領域が設定されたスレッショルドを下回ると起動される一連のスレッショルド・アクション・プロシージャによって実行されます。sp_dbextend をインタフェースとして、指定のセグメントまたはデバイスに対する拡張プロセスの管理を行います。

自動拡張機能を、サーバ全体のデフォルトの拡張ポリシーに従って実行するように設定することも、指定したデータベース内のセグメントごとにカスタマイズすることもできます。重要なデータが格納されるテーブルが存在する主要セグメントにスレッショルドをインストールすることにより、テーブルの種類ごとに異なるデータ領域に関する要件を Adaptive Server がどのようにして満たすかを細かく設定することができます。主要なテーブルに大量の挿入が行われる場合は、テーブルを特定のセグメントにバインドし、そのセグメントの拡張に関してはサイト固有のルールを適用します。このようにすれば、運用環境でそのような主要なテーブルに大量のロードが行われても、業務が中断されるような事態を回避することができます。

スレッショルドを使用して、データベースまたはそのセグメントを縮小することはできません。

『リファレンス・マニュアル：プロシージャ』を参照してください。

# 自動データベース拡張プロシージャのインストール

自動拡張機能をインストールするには、*installdbextend* スクリプトを使用します。このスクリプトは、データベースやデバイスの自動拡張機能のデフォルト設定が記述されている master.dbo.sysattributes にローをロードします。また、model データベースと tempdb データベースの中に制御テーブルを作成します。

Adaptive Server バージョン 12.5.1 以降にアップグレードする場合は、アップグレード・プロセスの一部としてこのスクリプトを別にインストールしてください。

### ❖ 自動データベース拡張機能のインストール

1. sa_role のパーミッションを持つユーザとしてログインします。UNIX では、*installdbextend* は、*$SYBASE/$SYBASE_ASE/scripts* にあります。Windows では、*%SYBASE%/%SYABASE_ASE%/scripts* にあります。
2. UNIX の場合、以下のコマンドを実行します。

```
isql -Usa -P -Sserver_name <$SYBASE/$SYBASE_ASE/scripts/installdbextend
```

Windows の場合、以下のコマンドを実行します。

```
isql -Usa -P -Sserver_name <%SYBASE%/%SYBASE_ASE%/scripts/installdbextend
```

*installdbextend* スクリプトを実行すると、一連のスレッショルド・アクション・プロシージャと sp_dbextend が sybsystemprocs データベースにインストールされます。

# *sp_dbextend* の実行

sp_dbextend を使用すると、サイト固有のルールに基づいてデータベースとデバイスの拡張プロセスをカスタマイズできます。データベース管理者は、データベースとセグメントのペアごとの拡張するサイズまたは割合、およびデバイスごとの拡張するサイズまたは割合を設定できます。

また、拡張の上限となる最大サイズを指定することで、データベース・セグメントまたはデバイスの拡張を制限できます。

sp_dbexpand をテスト・モードで使用すれば、選択した拡張ポリシーに基づく拡張プロセスのシミュレーションが可能です。

sp_dbextend には、現在の設定のリストを表示する機能や、サイト固有のポリシー・ルールを削除する機能があります。

この情報は、新しい属性定義として master.dbo.sysattributes に格納されます。

## sp_dbextend インタフェースでのコマンド・オプション

sp_dbextend は次の構文を使用します。

```
sp_dbextend [ command [, arguments...] ]
```

command は後で説明するオプションのいずれかです。arguments はデータベース名、セグメント名、空き領域量などを指定します。『リファレンス・マニュアル：プロシージャ』を参照してください。

引数をまったく指定しないで sp_dbextend を実行すると、デフォルトの help を指定したことになります。『リファレンス・マニュアル：プロシージャ』を参照してください。

---

**注意** 自動拡張プロシージャによって新しいデバイスが作成されることはありません。このプロシージャは、セグメントが現在マッピングされている既存デバイス上での、データベースとセグメントのサイズの変更だけを行います。

スレッショルド・アクション・プロシージャの使用を中止するには、sp_dropthreshold を実行するか、*clear* オプションを指定して sp_dbextend を実行することにより、スレッショルドをクリアします。『リファレンス・マニュアル：プロシージャ』を参照してください。

---

## 現在のスレッショルドの検証

check パラメータを使用すると、さまざまなスレッショルドの現在の設定を検証できます。たとえば、複数のセグメントが同じデバイス・セットを共有し、どのセグメントも自動拡張するように設定されている場合や、logsegment に対して自動拡張をトリガするように現在設定されているスレッショルドが、logsegment の現在のラストチャンス・スレッショルドに近すぎる場合は、check を実行すると警告が表示されます。この状況では、自動スレッショルドに達することはなく、check によって警告がレポートされます。

sp_dbextend には豊富な機能を持つシミュレーション・モードがあり、sa_role パーミッションを付与されたユーザであれば、トップレベルのスレッショルド・アクション・プロシージャの実行をシミュレートできます。

- pubs2 データベースの logsegment に対する拡張ポリシーを定義するには、次のコマンドを実行します。

```
sp_dbextend 'set', 'database', pubs2, logsegment,'3M'

sp_dbextend 'set', 'threshold', pubs2, logsegment, '1M'
```

- このポリシーでの拡張をシミュレートするには、次のコマンドを実行します。

```
sp_dbextend 'simulate', pubs2, logsegment
------------------------------
```

この入力の後に、サーバからのメッセージが表示されます。

次の例は、データベース pubs2 のセグメント logsegment に対するスレッショルドに 1 回達した場合に行われる、一連のデータベースとディスクの拡張を示します。

```
sp_dbextend 'simulate', pubs2, logsegment
---------------
NO REAL WORK WILL BE DONE.
Simulate database / device expansion in a dry-run mode 1 time(s).
These are the series of database/device expansions that would have
happened if the threshold on database'pubs2',
segment 'logsegment' were to fire 1 time(s).

Threshold fires: Iteration: 1.
============================

Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on segment
'logsegment'.

Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M'
-- Segment: logsegment

Database 'pubs2' was altered by total size '3M' for
segment 'logsegment'.
```

```
Summary of device/database sizes after 1 simulated extensions:
================================================
devicename initial size final size
---------- ------------ ----------
pubs2_data 20.0M        20.0M

(1 row affected)

Database 'pubs2', segment 'logsegment' would be altered from an initial
size of '4M' by '3M' for a resultant total size of '7M'.

To actually expand the database manually for this threshold, issue:
sp_dbextend 'execute', 'pubs2','logsegment', '1'

(return status = 0)
```

このスレッショルドに対してデータベースを手動で拡張するには、次のコマンドを実行します。

```
sp_dbextend 'execute', 'pubs2', 'logsegment'
--------------
```

次の例では、このレベルのスレッショルドに達した場合に、alter database コマンドによって logsegment の pubs2_data デバイスに対する変更が行われることを示します。

```
sp_dbextend 'execute', pubs2, logsegment

Threshold fires: Iteration: 1.
================================
Threshold action procedure 'sp_dbxt_extend_db' fired in db 'pubs2' on
segment 'logsegment'. Space left: 512 logical pages ('1M').

ALTER DATABASE pubs2 log on pubs2_data = '3.0M' -- Segment: logsegment
Extending database by 1536 pages (3.0 megabytes) on disk pubs2_data
Warning: The database 'pubs2' is using an unsafe virtual device
'pubs2_data'. The recovery of this database can not be guaranteed.

Warning: Using ALTER DATABASE to extend the log segment will cause user
thresholds on the log segment within 128 pages of the last chance
threshold to be disabled.

Database 'pubs2' was altered by total size '3M' for segment 'logsegment'.

(return status = 0)
```

- 特定のセグメント上のスレッショルドに <*n*> 回連続して達した場合をシミュレートするには、繰り返し回数を指定してこのコマンドを実行します。

```
sp_dbextend 'simulate', pubs2, logsegment, 5
-----------------
```

このデータベースを 5 回拡張するには、次のように入力します。

```
sp_dbextend 'execute', 'pubs2', 'logsegment', 5
------------------
```

このコマンドを実行すると、スレッショルドが 5 回連続して起動された場合、データベースに対して一連の alter database オペレーションが実行され、次に disk resize オペレーションが 1 回以上実行され、最後に、指定のデバイス上での alter database オペレーションが 1 回実行されることが表示されます。

# データベースに対する自動拡張の設定

データベースで自動拡張を行うように各セグメントを設定する手順は、次のとおりです。この項では、pubs2 データベースを使用します。

ここで示す手順がすべて必須というわけではありません。たとえば、特定のデバイスに対しては *growby* や *maxsize* を設定せず、システムのデフォルトのポリシーだけを使用することもできます。

### ❖ データベースの設定

1　データベースを作成します。

```
create database pubs2 on pubs2_data = "10m" log on pubs2_log = "5m"
```

2　pubs2_data デバイスの *growby* ポリシーと *maxsize* ポリシーを、それぞれ 10MB と 512MB に設定します。これらのポリシーの設定は、現在使用しているデータベースとは無関係に設定できます。次のように入力します。

```
exec sp_dbextend 'set', 'device', pubs2_data, '10m', '512m'
```

3　システム・デフォルトの *growby* ポリシーは、デバイスの 10% と設定されています。pubs2_log デバイスに対する新しいポリシーを設定する代わりに、このシステム・デフォルトを変更して、*growby* に別の値を選択することもできます。その場合は、pubs2_log は、この割合で拡張します。次のように入力します。

```
exec sp_dbextend 'modify', 'device', 'default', 'growby', '3m'
```

4　デフォルト・セグメントに対して *growby* 率を設定します。ただし、最大サイズは指定しません。次のように入力します。

```
exec sp_dbextend 'set', 'database', pubs2, 'default', '5m'
```

デフォルト・セグメントの *growby* 率は、セグメントが存在するデバイスでの率とは異なることがあります。*growby* は、セグメントの空き領域がなくなったときのセグメントの拡張を制御するもので、セグメントの拡張時だけに使用されます。

5 logsegment に対する *growby* 変数と *maxsize* 変数を設定します。

```
exec sp_dbextend 'set', 'database', pubs2, 'logsegment', '4m', '100m'
```

6 pubs2 データベースの各セグメントに設定されているポリシーを調べます。

```
exec sp_dbextend 'list', 'database', pubs2
```

7 pubs2 が占有する各デバイスでのポリシーを調べます。*devicename* にパターン指定子 (% を使用して、該当するすべてのデバイスを検索します。

```
exec sp_dbextend 'list', 'device', "pubs2%"
```

8 pubs2 のデフォルト・セグメントと logsegments セグメントに拡張スレッショルドをインストールします。これにより、拡張プロセスが設定されて有効になり、拡張プロセスをトリガする空き領域スレッショルドを選択できるようになります。次のように入力します。

```
use pubs2
-----------------------------------------------------------------

    exec sp_dbextend 'set', 'threshold', pubs2, 'default',   '4m'
    exec sp_dbextend   'set', 'threshold',  pubs2,  'logsegment', '3m'
```

9 上記のコマンドでインストールされたスレッショルドを調べます。

```
exec sp_dbextend list, 'threshold'

segment name free pages  free pages (KB) threshold procedure status
------------ ----------- --------------- ------------------- -----------
default          2048    4096             sp_dbxt_extend_db enabled
logsegment        160  320           sp_thresholdaction lastchance
logsegment       1536  3072          sp_dbxt_extend_db      enabled
Log segment free space currently is 2548 logical pages (5096K).
(1 row affected, return status = 0)
```

この出力では、sp_dbxt_extend_db が、実行時に拡張プロセスを起動するスレッショルド・プロシージャであることがわかります。現在、拡張スレッショルドは、デフォルトと logsegment の両方のセグメントで有効です。

10 simulate を実行して、拡張の様子を確認します。

```
exec sp_dbextend 'simulate', pubs2, logsegment
exec sp_dbextend 'simulate', pubs2, 'default', '5'
```

11 必要であれば、modify を使用してポリシーを変更します。

```
exec sp_dbextend 'modify', 'database', pubs2, logsegment, 'growby','10m'
```

12 特定のセグメントでの拡張を一時的に無効にするには、disable を使用します。

```
exec sp_dbextend 'disable', 'database', pubs2, logsegment
```

13　データベースとデバイスに対する拡張ポリシーの状態を調べます。

```
exec sp_dbextend list, 'database'
name         segment    item    value status
-----------  ---------- ------- ----- --------
server-wide (n/a)      (n/a)   (n/a) enabled
default     (all)      growby  10%   enabled
pubs2       default    growby  5m    enabled
pubs2       logsegment growby  10m   disabled
pubs2       logsegment maxsize 100m  disabled
(1 row affected, return status = 0)
```

状態が無効 (disabled) の場合は、現在、pubs2 の logsegment に対しては拡張プロセスが無効であることを示します。

```
exec sp_dbextend list, 'device'
name            segment item   value status
--------------- ------- ------ ----- -------
server-wide    (n/a)   (n/a)  (n/a) enabled
default        (n/a)   growby 3m    enabled
mypubs2_data_0 (n/a)   growby 10m   enabled
mypubs2_data_1 (n/a)   growby 100m  enabled
mypubs2_log_1  (n/a)   growby 20m   enabled
mypubs2_log_2  (n/a)   growby 30m   enabled
(1 row affected, return status = 0)
```

14　enable を使用して、拡張プロセスを再度有効にします。

```
exec sp_dbextend 'enable', 'database', pubs2, logsegment
```

# 制約と制限事項

スレッショルドの設定には、次の制約と制限事項が適用されます。

- 1 つ以上のデータベースで、複数のセグメントにスレッショルド・プロシージャがインストールされているときは、スレッショルドに達した順序で拡張が実行されます。logsegment に対して abort tran on log full がオフになっている場合は、logsegment のスレッショルド・プロシージャがそのデータベースを変更するようスケジュールされるまでタスクは待機します。
- ログが記録されないセグメントでは、空き領域スレッショルドを超えた後もタスクの処理は続行しますが、スレッショルド・プロシージャはキューに残ります。その結果、データ・セグメントの「領域不足」エラーが発生することがあります。タスクを完了できるだけの十分な空き領域をデータベース内に確保できるように、スレッショルドを設計してください。

- 多数のスレッショルド・プロシージャが同時に起動されると、プロシージャ・キャッシュに多大な負荷がかかることがあります。非常に多数のデータベース、多数のセグメント、多数のスレッショルド・アクション・プロシージャがインストールされた環境では、このことが発生する可能性が高くなります。
- tempdb の空き領域が非常に少なくなったときに、他のオペレーションによって tempdb のリソースが要求されると、スレッショルド・プロシージャは、この状況の改善を試みる間に異常終了することがあります。この問題を回避するには、十分な空き領域 (2MB 以上 ) が確保されるように、tempdb にインストールするスレッショルド・プロシージャを設定してください。

データベースとデバイスの拡張方法を決定するサイト固有のポリシーを管理するために、ダンプとロードの手順を変更しなければならない場合があります。

データベースをダンプしても master.dbo.sysattributes に格納されている情報は転送されません。したがって、サーバ間でデータベースを移行するためにダンプとロードを使用する場合は、sysattributes データベース内のデータとしてコード化されたサイト固有のポリシーを手動で移行する必要があります。この場合は、次の 2 つの対処方法が考えられます。

- master.dbo.sysattributes のクラス番号 19 のエントリにアクセスするように定義したビューから bcp out を実行することによって、データを手動で master.dbo.sysattributes から抽出します。次に、bcp in を実行して、移行先サーバにデータをロードします。この場合は、2 つのサーバにある両方のデータベースで、セグメント ID が同じでなければなりません。
- Sybase Central の ddlgen 機能を使用して、ポリシー・ルールの再作成に必要な sp_dbextend set の呼び出しを再生成することもできます。再作成するには、移行先サーバで ddlgen スクリプトを実行します。ただし、サーバ・プロシージャ間で論理デバイスの名前が変更されている場合は、ddlgen による方法では対応できません。移行先サーバのデバイス名を、手動で変更する必要があります。

次の制約は、エラーの原因にはなりません。

- データベースで sp_dboption ‘no free space acctg’ をオンにした場合、ログを取らないセグメントにはスレッショルド・アクションをインストールできます (『リファレンス・マニュアル：プロシージャ』を参照)。このオプションがオフの場合はスレッショルド・アクションは起動されないので、このオプションの意味は、データベース拡張が実行されないということだけです。このオプションをオンのままにしておくと、警告メッセージが表示されます。
- また、拡張が発生した場合は、定期的に master データベースをダンプすることをおすすめします。このようにすれば、何度も拡張が行われた後に障害が発生しても、master データベースを再作成できます。
- これらの汎用スレッショルド・プロシージャは、システム・データベース、特に master データベースにはインストールしないでください。master の領域使用方法を変更するには、特殊な処置が必要だからです (『リファレンス・マニュアル：コマンド』を参照)。
- スレッショルドを使用して、データベースまたはそのセグメントを縮小することはできません。

第 17 章

# スレッショルドによる空き領域の管理



データベースの作成時または変更時には、一定量の空き領域をデータベースのデータ・セグメントおよびログ・セグメントに割り付けます。オブジェクトを作成してデータを挿入するにつれて、データベース内の空き領域量が減少します。

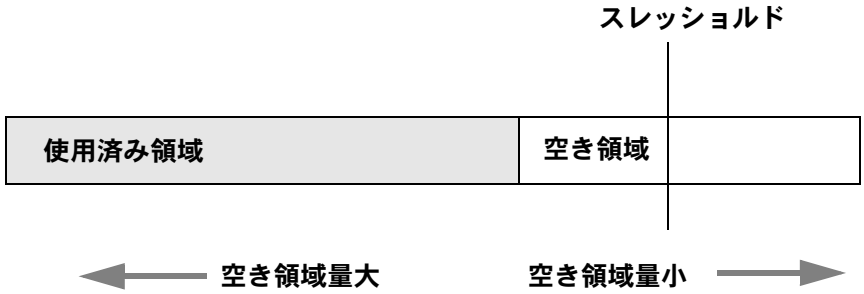
## ラストチャンス・スレッショルドによる空き領域のモニタ

master データベースを含むすべてのデータベースに、「ラストチャンス・スレッショルド」があります。スレッショルドとは、トランザクション・ログのバックアップが必要となる空きログ・ページ数の見積もり値のことです。ログ・セグメントに割り付ける領域を増やすと、それに従ってラストチャンス・スレッショルドが自動的に調整されます。

ログ・セグメント内の空き領域の量がラストチャンス・スレッショルドを下回ると、Adaptive Server は、sp_thresholdaction という特別なストアド・プロシージャを自動的に実行します (sp_modifythreshold を使用して、別のラストチャンス・スレッショルド・プロシージャを指定できます)。

図 17-1 に、ラストチャンス・スレッショルドを持つログ・セグメントを示します。網かけの部分は、使用済みのログ領域を表しています。網かけのない部分は、空きログ領域を表しています。ラストチャンス・スレッショルドはまだ超えていません。

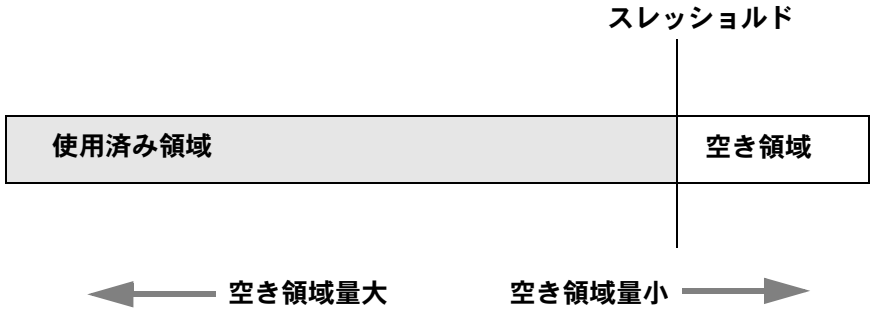
図 17-1: ラストチャンス・スレッショルドを持つログ・セグメント




## スレッショルドの超過

ユーザがトランザクションを実行するにつれて、空きログ領域の量が減少していきます。空き領域の量がラストチャンス・スレッショルドを超えると、Adaptive Server は sp_thresholdaction を自動的に実行します。

図 17-2: ラストチャンス・スレッショルドに到達した場合の sp_thresholdaction の実行



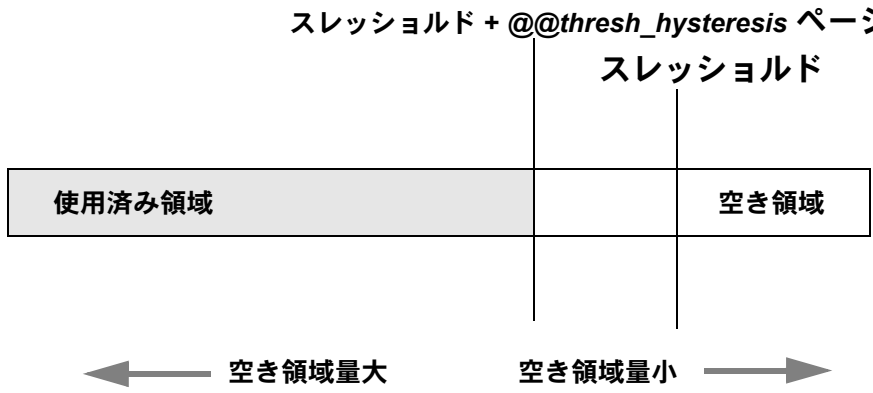

## *sp_thresholdaction* の実行頻度の制御

Adaptive Server は、ヒステリシス値 (グローバル変数 *@@thresh_hysteresis*) を使用して、空き領域量の変化に対するスレッショルドの感度を制御します。

sp_thresholdaction を実行したスレッショルドは非アクティブ状態になり、セグメント内の空き領域の量がスレッショルドより *@@thresh_hysteresis* ページ分増加するまで非アクティブ状態が続きます。これにより、空き領域のわずかな変化にスレッショルドが反応して、そのプロシージャが繰り返し実行されることを防止します。*@@thresh_hysteresis* の値は変更できません。

たとえば、図 17-2 のスレッショルドは、sp_thresholdaction を実行すると非アクティブになります。図 17-3 では、空き領域の量が *@@thresh_hysteresis* ページ分増加したときに、スレッショルドが再びアクティブになります。

**図 17-3: 空き領域が @@thresh_hysteresis 分増加したときのスレッショルドの再アクティブ化**




# ロールバック・レコードとラストチャンス・スレッショルド

Adaptive Server バージョン 11.9 以降ではトランザクション・ログに「ロールバック・レコード」が含まれます。ロールバック・レコードは、トランザクションのロールバックが行われるたびにログに記録されます。オープン・トランザクションに属するすべての更新に対応するロールバック・レコードをログに記録できるだけの十分な領域がサーバによって確保されます。トランザクションが正常に終了した場合は、ロールバック・レコードはログに記録されず、確保されていた領域は解放されます。

長時間実行されるトランザクションでは、ロールバック・レコードによって大量の領域が確保される可能性があります。

syslogs によって使用されている領域を調べるには、sp_spaceused を実行します。

```
sp_spaceused syslogs
```

出力は次のようになります。

```
name            total_pages     free_pages      used_pages      reserved_pages
--------------- --------------- --------------- --------------- --------------
syslogs                5632            1179            3783            670
```

dbcc checktable(syslogs) を実行すると、次のような出力が生成されます。

```
Checking syslogs: Logical pagesize is 2048 bytes
The total number of data pages in this table is 3761.
*** NOTICE: Space used on the log segment is 3783 pages (7.39 Mbytes), 67.17%.
*** NOTICE: Space reserved on the log segment is 670 pages (1.31 Mbytes), 11.90%.
*** NOTICE: Space free on the log segment is 1179 pages (2.30 Mbytes), 20.93%.
```

十分な空き領域があるように見えるときにトランザクション・ログのラストチャンス・スレッショルドに到達した場合は、その領域はロールバック用に確保されている領域であり、このような状態は、問題発生の原因となる可能性があります。「ロールバック・レコード用の現在の領域の算定」(471 ページ ) を参照してください。

## ロールバック・レコード用の領域の計算

ロールバック・レコードを記録するために必要な、トランザクション・ログに追加する領域の大きさを計算するには、次の数を見積もります。

- トランザクション・ログ内の更新レコードのうち、既にロールバックされたトランザクションに属するものの数
- トランザクション・ログ内の更新レコードのうち、オープン・トランザクションに属するものの最大数

更新レコードは、タイムスタンプの値を変更します。これには、データ・ページ、インデックス・ページ、アロケーション・ページなどの変更が含まれます。

各ロールバック・レコードは、約 60 バイト、つまり 1 ページの 100 分の 3 の領域を必要とします。したがって、トランザクション・ログに追加する RR (ロールバック・レコード) の量は次のようにして計算します。

ページ単位の追加領域量 = (ログに記録される RR の数 + オープンな更新の数) × 3/100

また、ラストチャンス・スレッショルドとユーザ定義のスレッショルドへのロールバック・レコードの影響に備えて、ログ領域を追加することもできます。これについては、次の項で説明します。

### lct_admin による空きログ領域の算定

専用のログ・セグメントの空き領域量を算定するには、logsegment_freepages を使用します。

pubs2 データベースのログ・セグメントの空きページの数を調べるには、次のコマンドを発行します。

```
select lct_admin("logsegment_freepages", 4)
---------------------
                   79
```

## ロールバック・レコード用の現在の領域の算定

データベースがロールバック用に現在確保しているページ数を算定するには、lct_admin に reserved_for_rollbacks パラメータを付けて実行します。

返されるページ数は、ロールバック・レコード用に確保されているけれども、まだ割り付けられていないページ数です。

たとえば、pubs2 データベース (dbid は 5) 内でロールバック用に確保されているページ数を算定するには、次のコマンドを発行します。

```
select lct_admin("reserved_for_rollbacks", 5)
```

『リファレンス・マニュアル：コマンド』を参照してください。

## ラストチャンス・スレッショルドへのロールバック・レコードの影響

ロールバック・レコードを使用する Adaptive Server では、ラストチャンス・スレッショルドの値に余裕を持たせる必要があります。また、既にログに記録されているロールバック・レコードが使用している領域や、ロールバック・レコードが出力される可能性のあるオープン・トランザクションに備えて確保されている領域が原因となって、ラストチャンス・スレッショルドへの到達が早まることがあります。

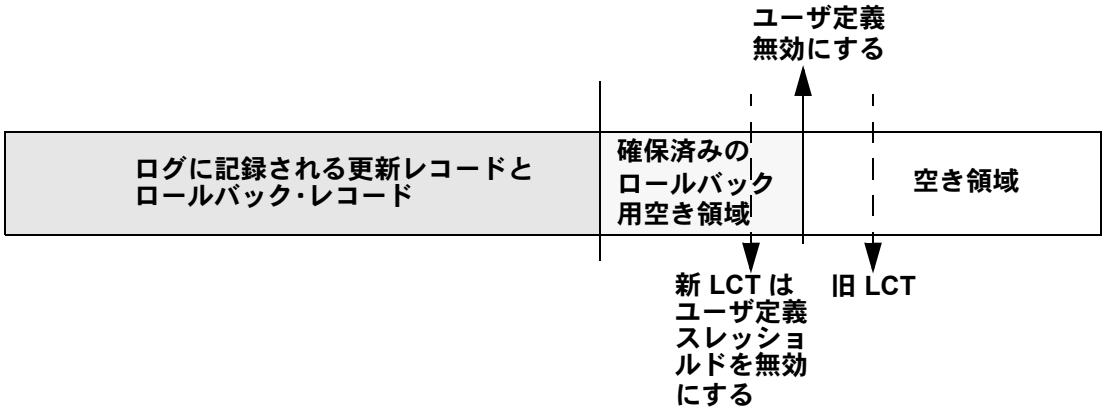
## ユーザ定義のスレッショルド

ロールバック・レコードが占有する分、トランザクション・ログの使用済み領域が増えるため、ロールバック・レコードを使用するバージョンの Adaptive Server では、使用しないバージョンよりも、ダンプ実行用のユーザ定義のスレッショルドを過ぎたときに残される空き領域の大きさが小さくなります。しかし、ラストチャンス・スレッショルドが増えたことによるダンプ用の空き領域の減少量は、オープン・トランザクションのロールバック・レコード用に確保される領域量により十分に補完されます。

ユーザ定義のスレッショルドを使用して、dump transaction を起動することができます。使用済みの領域がラストチャンス・スレッショルドに達して、ログ内のすべてのオープン・トランザクションが中断させられる前に、ダンプを完了できるように、このスレッショルドを設定して十分な領域を確保します。

ログとデータが混在するデータベースでは、ラストチャンス・スレッショルドは動的に移動します。その値がユーザ定義のスレッショルドよりも小さくなるように自動的に設定することもできます。その場合、ユーザ定義のスレッショルドは無効になり、ユーザ定義のスレッショルドよりも前にラストチャンス・スレッショルドに到達します (図 17-4 を参照)。

**図 17-4: ユーザ定義のスレッショルドよりも前に LCT に到達する場合**




ラストチャンス・スレッショルドの設定値がユーザ定義のスレッショルドよりも大きくなると (たとえば、ラストチャンス・スレッショルドが図 17-4 の「旧 LCT」の値に再設定された場合 )、ユーザ定義のスレッショルドは再び有効になります。

ログ・セグメントが分かれているデータベースでは、ログには専用の領域が確保されるため、ラストチャンス・スレッショルドは静的に設定されます。ユーザ定義のスレッショルドはラストチャンス・スレッショルドには影響されません。

# ログとデータがセグメントを共有する場合のラストチャンス・スレッショルドとユーザ・ログ・キャッシュ

Adaptive Server のすべてのデータベースにはラストチャンス・スレッショルドがあります。また、トランザクションをユーザ・ログ・キャッシュにバッファすることが可能です。ログとデータがセグメントを共有するデータベースを作成するとき、そのデータベースのラストチャンス・スレッショルドは、model データベースのサイズに基づいて設定されます。データが追加されてログへの記録が始まると、使用可能な領域と現在オープンなトランザクションに基づいて、ラストチャンス・スレッショルドが動的に再計算されます。ログとデータに専用のセグメントを確保しているデータベースの場合は、ラストチャンス・スレッショルドはログ・セグメントのサイズに基づいて設定され、動的には変化しません。

データベースの現在のラストチャンス・スレッショルドを調べるには、lct_admin を使用します。次のように、reserve パラメータを指定し、ログ・ページ数には 0 を指定します。

```
select lct_admin("reserve",0)
```

データベースのラストチャンス・スレッショルドは systhresholds テーブルに格納されています。また、sp_helpthreshold を使ってアクセスすることもできます。ただし、次の点に注意してください。

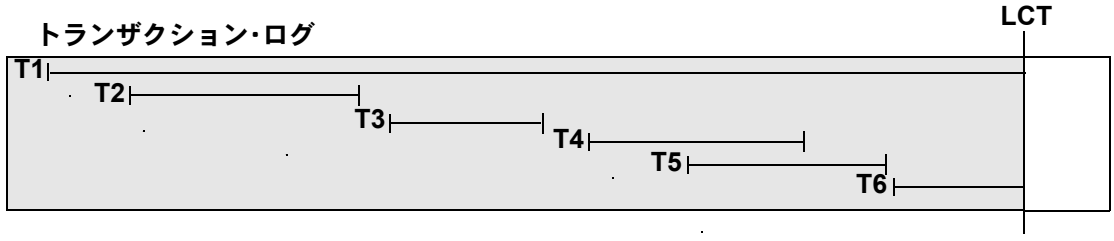
- sp_helpthreshold は、ユーザ定義のスレッショルドとその他のデータとともに、ラストチャンス・スレッショルドの現在の値を返します。ラストチャンス・スレッショルドの現在の値だけが必要な場合は、lct_admin を使う方が簡単です。どちらの場合も、ラストチャンス・スレッショルドの最新値が返されます。
- ログとデータがセグメントを共有しているデータベースの場合は、systhresholds 内のラストチャンス・スレッショルドの値は最新のものとは限りません。

## *lct_admin abort* による、中断されたトランザクションのアボート

トランザクション・ログがラストチャンス・スレッショルドに到達すると、すべてのオープン・トランザクションが中断されます。通常は、空き領域を作成するためにトランザクション・ログがダンプされます。ダンプによって、コミットされたトランザクションがログの先頭から削除されるためです。しかし、ログの先頭にあるトランザクションがまだオープンである場合は、トランザクション・ログのダンプは実行できません。

ログのダンプを妨げている、中断されたトランザクションを終了させるには、lct_admin abort を使用します。トランザクションを終了させると、そのトランザクションはクローズされるので、ダンプが実行可能になります。図 17-5 は、lct_admin abort が使用される状況を示します。トランザクション・ログが LCT に到達したため、オープン・トランザクション T1 と T6 が中断しています。T1 がログの先頭にあるため、ダンプを実行してクローズド・トランザクション T2 から T5 までを削除することができず、ログへの記録を続けるための空き領域を作成できません。ここで lct_admin abort を実行して T1 を終了させると、T1 をクローズできるため、ダンプを実行して T1 から T5 までのトランザクションをログから削除できます。

**図 17-5: lct_admin abort を使用する状況**



### ❖ 中断されたトランザクションのアボート

トランザクションをアボートするには、まずその ID を確認する必要があります。

1 ラストチャンス・スレッショルドに到達したトランザクション・ログ内で、最も古いオープン・トランザクションの spid を検索するには、次のクエリを使用します。

```
use master
go
select dbid, spid from syslogshold
where dbid = db_id("name_of_database")
```

たとえば、pubs2 データベース上で、最も古い実行中のトランザクションを検索するには、次のように入力します。

```
select dbid, spid from syslogshold
where dbid = db_id ("pubs2")
dbid   spid
------ ------
     7      1
```

2 最も古いトランザクションを終了するには、そのトランザクションを開始したプロセスの ID (spid) を入力します。そのプロセスに属する中断しているトランザクションがログ内にあるときは、そのトランザクションもこれで終了します。

たとえば、中断しているログの最も古いオープン・トランザクションがプロセス 83 に属していて、このトランザクションを終了する場合は、次のように入力します。

```
select lct_admin("abort", 83)
```

これで、同じトランザクション・ログ内の、プロセス 83 に属する他のオープン・トランザクションも終了します。

ログ内のオープンなトランザクションをすべて終了させるには、次のように入力します。

```
select lct_admin("abort", 0, 12)
```

『リファレンス・マニュアル：コマンド』を参照してください。

# master データベースのトランザクション・ログへの領域の追加

master データベースのラストチャンス・スレッショルドに達したとき、alter database を使用して master データベースのトランザクション・ログに領域を追加できます。これにより、そのログ内の中断していたトランザクションが再開されるため、サーバでより多くのアクティビティが可能になります。しかし、master データベースのトランザクション・ログがラストチャンス・スレッショルドに到達している場合に、alter database を使って他のデータベースを変更することはできません。このため、master データベースと他のデータベースの両方がそれぞれのラストチャンス・スレッショルドに到達している場合は、まず alter database を使って master データベースにログ領域を追加し、その後でもう 1 つのデータベースに同じように領域を追加します。

# プロセスの自動アボートまたは自動中断

ラストチャンス・スレッショルドを使用すると、ログの十分な空き領域に dump transaction コマンドを記録できます。ただし、そのデータベースに対する追加のユーザ・トランザクションを記録するには空き領域が不十分な場合もあります。

ラストチャンス・スレッショルドに達すると、Adaptive Server はユーザ・プロセスを中断して、次のようなメッセージを表示します。

```
Space available in the log segment has fallen critically low
in database 'mydb'. All future modifications to this database
will be suspended until the log is successfully dumped and
space becomes available.
```

この場合は、トランザクション・ログに記録されないコマンド (select および readtext) や、ログ領域を解放するのに必要なコマンド (dump transaction、dump database、alter database) だけを実行できます。

## *abort tran on log full* によるトランザクションのアボート

オープン・トランザクションを中断せずに、自動的にアボートするようにラストチャンス・スレッショルドを設定するには、次のコマンドを入力します。

```
sp_dboption database_name "abort tran on log full", true
```

Adaptive Server の旧バージョンからアップグレードする場合は、新バージョンのサーバに abort tran on log full の設定が引き継がれます。

# 中断されたプロセスの再起動

dump transaction コマンドを使用して十分なログ領域を解放すると、中断していたプロセスが自動的に起動して最後まで実行されます。最後のバックアップ以降に writetext や select into が行われたために、ログを取らないデータベースへの変更が存在する場合は、dump tran with truncate_only を実行します。このコマンドはログを取らない書き込みがある場合でも実行できます。ログ領域不足で dump tran with truncate_only が失敗した場合は、dump tran with no_log を実行できます。ただし、dump tran with no_log は、最後の手段として緊急を要する場合にのみ実行してください。

トランザクションのダンプが完了し、トランザクションがログ中断状態から正しく解放されると、システム管理者またはデータベース所有者はデータベースをダンプできます。

この方法で中断したプロセスを再起動するための十分な領域が解放されない場合は、alter database の log on オプションを使用してトランザクション・ログのサイズを大きくする必要があります。

コマンドの強制終了に代わる手段として、lct_admin(“abort”, *spid*) を使用できます。これを使用すると接続を維持できるため、接続を強制終了するよりも適している場合があります。『リファレンス・マニュアル：ビルディング・ブロック』を参照してください。

システム管理者の権限がある場合は、sp_who コマンドを使用してどのプロセスがログ中断状態にあるかを判断し、kill コマンドを使用してスリープしているプロセスを起動します。

# スレッショルドの追加、変更、削除

データベース所有者またはシステム管理者は、データベース内のセグメントの空き領域をモニタするために、追加のスレッショルドを作成できます。追加のスレッショルドを、「空き領域スレッショルド」と呼びます。データベースにはそれぞれ、ラストチャンス・スレッショルドを含めて最大 256 個のスレッショルドを作成できます。

システム・プロシージャ sp_addthreshold、sp_modifythreshold、sp_dropthreshold を使用して、スレッショルドを作成、変更、削除します。これらのプロシージャを使用するときは、現在のデータベースの名前を指定する必要があります。

## 既存のスレッショルドに関する情報の表示

データベース内のすべてのスレッショルドについての情報を表示するには、sp_helpthreshold システム・プロシージャを使用します。特定のセグメントのスレッショルドに関する情報を取得するには、sp_helpthreshold *segment_name* を使用します。

次の例では、データベースのデフォルト・セグメントのスレッショルドの情報が表示されています。“default” は予約語であるため、引用符で囲む必要があります。次の sp_helpthreshold の出力例では、このセグメントにスレッショルドが 1 つあり、200 ページに設定されていることがわかります。“last chance” カラムの値 0 は、このスレッショルドがラストチャンス・スレッショルドではなく、空き領域スレッショルドであることを示します。

```
sp_helpthreshold "default"

segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   -------------------
default        200                     0   space_dataseg

(1 row affected, return status = 0)
```

## スレッショルドとシステム・テーブル

システム・テーブル systhresholds に、スレッショルドについての情報が保管されています。sp_helpthreshold は、このテーブルを使用します。このテーブルには、セグメント名、空きページ、ラストチャンスかどうかの状態、スレッショルド・プロシージャ名の他に、スレッショルドを作成したユーザのサーバ・ユーザ ID と、その作成時点でのユーザの役割も記録されています。

Adaptive Server は、curunreservedpgs からの値を使用して、セグメントの残りの空き領域の量を調べ、スレッショルドをアクティブにするかどうかを決定します。

## 空き領域スレッショルドの追加

空き領域スレッショルドを作成するには、sp_addthreshold システム・プロシージャを使用します。『リファレンス・マニュアル：プロシージャ』を参照してください。

セグメント内の空き領域量がスレッショルドを下回ると、Adaptive Server の内部プロセスによって、関連付けられたプロシージャが実行されます。このプロセスには、ユーザが sp_addthreshold を実行してスレッショルドを作成したときのユーザのパーミッションから、それ以降に取り消されたパーミッションを差し引いたパーミッションが付与されます。

スレッショルドからは、同じデータベース、別のユーザ・データベース、sybsystemprocs、または master 内のプロシージャを実行できます。また Open Server 上でリモート・プロシージャを呼び出すこともできます。sp_addthreshold を使用してスレッショルドを作成するとき、スレッショルド・プロシージャが存在しているかどうかの確認は行われません。

## 空き領域スレッショルドの変更または新しい空き領域スレッショルドの指定

空き領域スレッショルドに新しいスレッショルド・プロシージャ、空き領域の値、またはセグメントを関連付けるか、ラストチャンス・スレッショルドに関連付けられているプロシージャの名前を変更するには、sp_modifythreshold システム・プロシージャを使用します。sp_modifythreshold によって既存のスレッショルドが削除され、代わりに新しいスレッショルドが作成されます。『リファレンス・マニュアル：プロシージャ』を参照してください。

たとえば、セグメントの空き領域が 200 ページではなく 175 ページを下回ったときにスレッショルド・プロシージャを実行するには、次のように入力します。

```
sp_modifythreshold mydb, "default", 200, NULL, 175
```

この例の NULL は、パラメータ・リストの正しい位置に *new_free_space* を置くためのプレースホルダです。スレッショルド・プロシージャ名は変更されません。

sp_modifythreshold を実行するには、ラストチャンス・スレッショルドに関連付けられている空きページ数を指定する必要があります。この値を調べるには、sp_helpthreshold を使用します。

スレッショルドを変更したユーザが、新しいスレッショルド所有者になります。セグメント上の空き領域量がスレッショルドを下回ると、Adaptive Server によってスレッショルド・プロシージャが実行されますが、このときのパーミッションは、sp_modifythreshold の実行時点で所有者に付与されていたパーミッションから、それ以後に取り消されたパーミッションを差し引いたものとなります。

次の例では、ラストチャンス・スレッショルドについての情報を表示します。次に、このスレッショルドを超えたときに実行される新しいプロシージャ sp_new_thresh_proc を指定しています。

```
sp_helpthreshold logsegment

segment name   free pages   last chance?   threshold procedure
------------   ----------   ------------   -------------------
logsegment     40                      1   sp_thresholdaction

(1 row affected, return status = 0)


sp_modifythreshold mydb, logsegment, 40, sp_new_thresh_proc
```

### スレッショルドの削除

セグメントから空き領域スレッショルドを削除するには、sp_dropthreshold システム・プロシージャを使用します。『リファレンス・マニュアル：プロシージャ』を参照してください。

*dbname* には、現在のデータベースの名前を指定します。1 つのセグメントに複数のスレッショルドが存在することもあるため、セグメント名と空きページ数の両方を指定します。次に例を示します。

```
sp_dropthreshold mydb, "default", 200
```

## ログ・セグメントの空き領域スレッショルドの作成

ラストチャンス・スレッショルドを超えると、十分なログ領域が解放されるまで、すべてのトランザクションがアボートまたは中断されます。運用環境では、このことがユーザに大きな影響を与えることがあります。適正な値の空き領域スレッショルドをログ・セグメントに追加することによって、ラストチャンス・スレッショルド超過の可能性を最小限に抑えることができます。

追加するスレッショルドは、ラストチャンス・スレッショルドの超過がほとんど発生しないような頻度でトランザクション・ログをダンプするように設定します。ただし、頻度が高すぎると、データベースのリストア時にロードするテープの数が多くなります。

この項では、第 2 のログ・スレッショルドを適切に設定するのに役立つ情報を取り上げます。まず、*free_space* 値をログ・サイズの 45% に設定してスレッショルドを追加し、次にサイトでの実際の領域の使用状況に基づいてこのスレッショルドを調整します。

#### ❖ ログ・サイズの 45% のログ・スレッショルドの追加

*free_space* 値をログ・サイズの 45% に設定してログ・スレッショルドを追加するには、次の手順に従います。

1　次のクエリを使用して、ページ単位で表したログ・サイズを調べます。

```
select sum(size)
from master..sysusages
where dbid = db_id("database_name")
and (segmap & 4) = 4
```

2　sp_addthreshold を使用して、*free_space* 値を 45% に設定した新しいスレッショルドを追加します。たとえば、ログの容量が 2048 ページならば、*free_space* の値に 922 ページを指定してスレッショルドを追加します。

```
sp_addthreshold mydb, logsegment, 922, thresh_proc
```

3　該当するデバイスにトランザクション・ログをダンプする、簡単なスレッショルド・プロシージャを作成します。「スレッショルド・プロシージャの作成」(483 ページ) を参照してください。

## 新しいスレッショルドのテストおよび調整

dump transaction コマンドを使用して、トランザクション・ログの使用済み領域が 55% 未満となるようにします。次に、以下の手順に従って、新しいスレッショルドをテストします。

1 日常行っているユーザの動作をシミュレートして、トランザクション・ログにデータを追加します。通常のトランザクションを実行する自動化スクリプトを使用します。

45% の空き領域スレッショルドを超えた時点で、スレッショルド・プロシージャはトランザクション・ログをダンプします。このスレッショルドはラストチャンス・スレッショルドではないため、トランザクションは中断もアボートもされず、ダンプ実行中もログは増え続けます。

2 ダンプの進行中に、sp_helpsegment を使用してログ・セグメントの領域の使用状況をモニタします。ダンプが完了する直前に、トランザクション・ログの最大サイズを記録しておきます。

3 図 17-6 に示すように、ダンプが完了した時点でログにかなりの空き領域が残っている場合は、トランザクション・ログのダンプの頻度を減らすことができます。

**図 17-6: 45% のスレッショルドを追加したトランザクション・ログ**

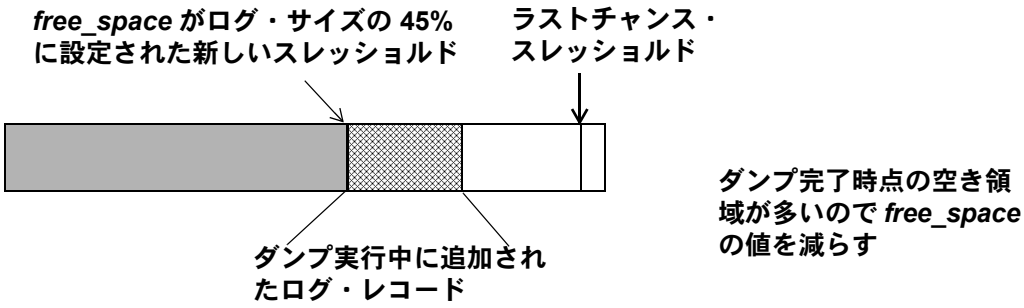
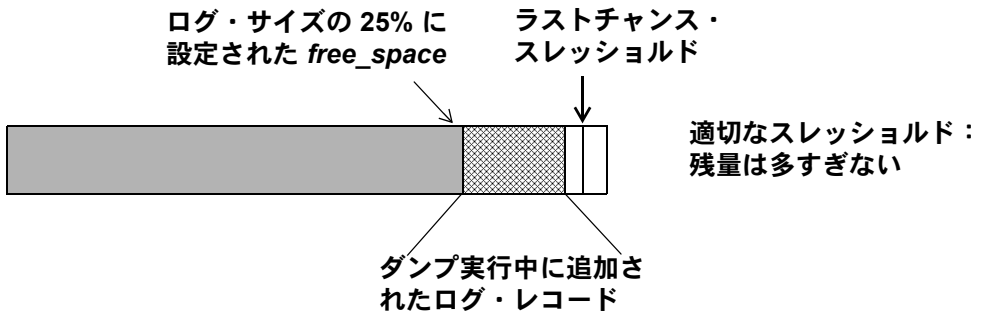



図 17-7 に示すように、ログ領域の残りが 25% になるまで待ちます。

**図 17-7: スレッショルドの変更によってダンプ完了後の空き領域を減らす場合**

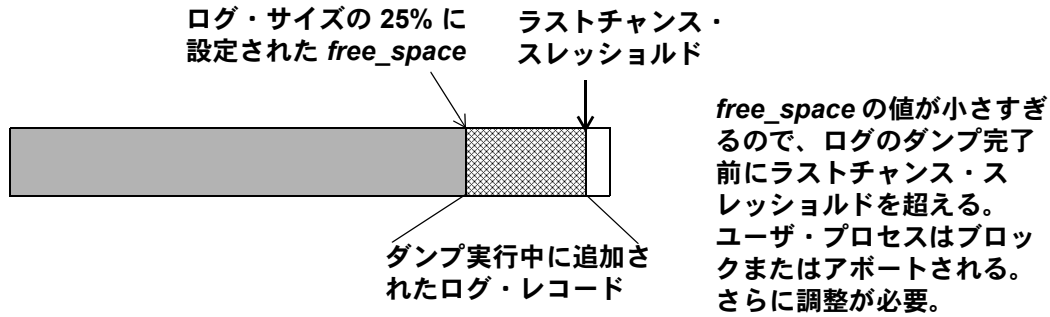



sp_modifythreshold を使用して、*free_space* 値をログ・サイズの 25% に調整します。次に例を示します。

```
sp_modifythreshold mydb, logsegment, 512,
    thresh_proc
```

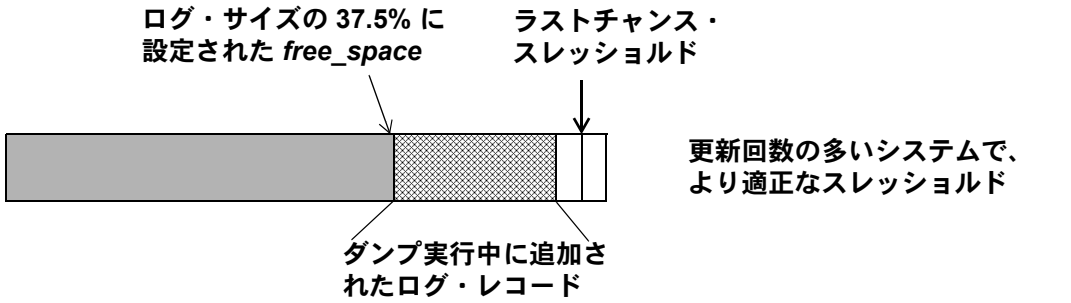
4　トランザクション・ログをダンプして、新しい *free_space* 値をテストします。図 17-8 に示すように、ダンプ完了前にログがラストチャンス・スレッショルドを超える場合は、dump transaction の開始が遅すぎます。

**図 17-8: 追加のログ・スレッショルドによるダンプの開始が遅すぎる場合**




25% の空き領域では十分ではありません。図 17-9 に示すように、ログの空き領域が 37.5% になった時点でダンプ・トランザクションが開始されるように設定します。

**図 17-9: スレッショルドの変更によって十分な空き領域を残してダンプが完了する場合**




sp_modifythreshold を使用して、*free_space* 値をログ容量の 37.5% に変更します。次に例を示します。

```
sp_modifythreshold mydb, logsegment, 768,
    thresh_proc
```

# 他のセグメントでの追加のスレッショルドの作成

ログ・セグメント上だけでなく、データ・セグメント上にも空き領域スレッショルドを作成できます。たとえば、テーブルとインデックスを保管するために使用する default セグメントに、空き領域スレッショルドを作成します。default セグメントの領域がこのスレッショルドを下回った時点でエラー・ログにメッセージを出力するように、関連付けるストアド・プロシージャを作成することもできます。エラー・ログ内のこのようなメッセージをモニタすれば、ユーザへの影響が発生する前にデータベース・デバイスに領域を追加することができます。

mydb の default セグメントに空き領域スレッショルドを作成する例を以下に示します。このセグメントの空き領域が 200 ページを下回った時点で、space_dataseg というスレッショルド・プロシージャが実行されます。

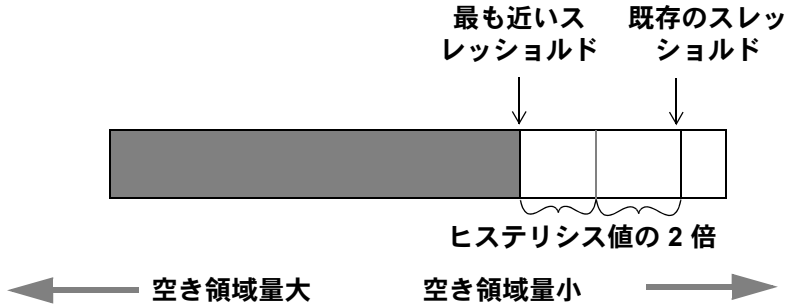
```
sp_addthreshold mydb, "default", 200, space_dataseg
```

「スレッショルド・プロシージャの作成」(483 ページ) を参照してください。

## スレッショルドの位置の決定

新しいスレッショルドを作成するとき、図 17-10 に示すように、最も近いスレッショルドとの間に *@@thresh_hysteresis* の値の 2 倍以上の間隔を空ける必要があります。

**図 17-10: スレッショルドの位置の決定**




次のコマンドを使用して、データベースのヒステリシス値を調べます。

```
select @@thresh_hysteresis
```

この例では、セグメントのスレッショルドが 100 ページに設定されています。データベースのヒステリシス値は 64 ページです。次に作成するスレッショルドは、100 + (2 * 64) ページ、つまり 228 ページ以上でなければなりません。

```
select @@thresh_hysteresis

------------
        64

sp_addthreshold mydb, user_log_dev, 228,
    sp_thresholdaction
```

# スレッショルド・プロシージャの作成

Sybase 提供のスレッショルド・プロシージャはありません。実際のサイトの要求に合わせて、ユーザ自身でスレッショルド・プロシージャを作成する必要があります。

スレッショルド・プロシージャが実行すべき処理には、サーバのエラー・ログへの書き込みや、ログ空き領域を増やすためのトランザクション・ログのダンプなどがあります。また、Open Server や XP Server へのリモート・プロシージャ・コールを実行することもできます。たとえば、次のコマンドを sp_thresholdaction に追加すると、Open Server 上で mail_me というプロシージャが実行されます。

```
exec openserv...mail_me @dbname, @segment
```

『Transact-SQL ユーザーズ・ガイド』を参照してください。

この項では、スレッショルド・プロシージャを作成するためのガイドラインを説明し、2 つのサンプル・プロシージャも示します。

## プロシージャ・パラメータの宣言

Adaptive Server は、次の 4 つのパラメータをスレッショルド・プロシージャに渡します。

- *@dbname*, varchar(30) − データベース名
- *@segmentname*, varchar(30) − セグメント名
- *@space_left*, int − スレッショルドの空き領域の残量
- *@status*、int：ラストチャンス・スレッショルドでは 1、その他のスレッショルドでは 0 です。

これらのパラメータは、名前ではなく位置で渡されます。作成するプロシージャでは、これらのパラメータに他の名前を使用することもできますが、上に示した順序とデータ型で宣言する必要があります。

## エラー・ログ・メッセージの生成

エラー・ログにデータベース名、セグメント名、スレッショルド・サイズを記録するには、プロシージャの始めの方に print 文を追加してください。プロシージャに print 文または raiserror 文がない場合は、エラー・ログにスレッショルド・イベントのレコードは出力されません。

スレッショルド・プロシージャを実行するプロセスは、Adaptive Server の内部プロセスです。このプロセスには、関連するユーザ端末やネットワーク接続はありません。スレッショルド・プロシージャをテストするために、端末セッション中にスレッショルド・プロシージャを直接実行 (execute *procedure_name* を使用 ) すると、print と raiserror のメッセージの出力が画面上に表示されます。スレッショルドに到達したことによって同じプロシージャが実行されるときは、日付と時刻を含むメッセージがエラー・ログに出力されます。

たとえば、sp_thresholdaction に次の文が含まれているとします。

```
print "LOG DUMP: log for '%1!' dumped", @dbname
```

Adaptive Server は次のメッセージをエラー・ログに書き込みます。

```
00: 92/09/04 15:44:23.04 server: background task message: LOG DUMP: log for 'pubs2' dumped
```

## トランザクション・ログのダンプ

sp_thresholdaction プロシージャに dump transaction コマンドが含まれていれば、プロシージャで指定したデバイスにログがダンプされます。dump transaction コマンドは、ログの先頭から、コミットされていないトランザクション・レコードが格納されているページの直前までのすべてのページを削除することで、トランザクション・ログをトランケートします。

ログ領域が十分に確保されると、中断されていたトランザクションが再開されます。トランザクションを中断ではなくアボートした場合は、ユーザが再度トランザクションを実行する必要があります。

ログを取らない書き込み状態のために sp_thresholdaction が失敗する場合は、代わりに dump tran with no_log を発行できます。

ディスクへのダンプ、特にデータベース・ディスクと同じマシンまたはディスク・コントローラにあるディスクへのダンプは行わないでください。ただし、スレッショルド超過によって開始されるダンプはいつ実行されるかわからないため、ディスクにダンプしてから、そのファイルをオフラインのメディアにコピーすることもできます。(この場合、そのファイルをロードし直すには、ファイルをディスクにコピーする必要があります)。

ダンプの方法は、以下の事項を考慮して決定します。

- ダンプされたデータを受け入れることができる、ロード済みの専用オンライン・ダンプ・デバイスがあるか
- データベースが使用可能な状態にある間、テープをマウントできるオペレータが常にいるか
- トランザクション・ログのサイズ
- トランザクション・レート
- 定期的にデータベースとトランザクション・ログをダンプするスケジュール
- 使用可能なディスクの領域
- サイト特有のその他のダンプ・リソースと制約

## 簡単なスレッショルド・プロシージャ

トランザクション・ログをダンプし、メッセージをエラー・ログに出力する簡単なプロシージャの例を次に示します。このプロシージャは、データベース名に変数 (*@dbname*) を使用するため、Adaptive Server のすべてのデータベースに使用できます。

```
create procedure sp_thresholdaction
    @dbname varchar(30),
    @segmentname varchar(30),
    @free_space int,
    @status int
as
dump transaction @dbname
    to tapedump1
print "LOG DUMP: '%1!' for '%2!' dumped",
        @segmentname, @dbname
```

## 複雑なプロシージャ

次に示すスレッショルド・プロシージャは、渡されるパラメータの値によって、異なるアクションを実行します。また、プロシージャに条件文が含まれているので、ログ・セグメントとデータ・セグメントの両方に使用できます。

このプロシージャは、次の処理を実行します。

- ログのラストチャンス・スレッショルドに到達した結果として呼び出された場合は、“LOG FULL” メッセージを出力します。ステータス・ビットは、ラストチャンス・スレッショルドでは 1、それ以外のスレッショルドでは 0 です。if (@status&1) = 1 は、ラストチャンス・スレッショルドの場合にのみ true の値を返します。
- 与えられたセグメント名がログ・セグメントであることを確認します。名前が変更されても、ログ・セグメントのセグメント ID は、常に 2 です。
- ダンプの実行前と実行後のトランザクション・ログのサイズ情報を出力します。ログの縮小量がそれほど大きくない場合は、長時間実行されるトランザクションによってログが満杯になっている可能性があります。
- トランザクション・ログのダンプの開始時刻と終了時刻を出力します。これは、ダンプ実行時間のデータ収集に役立ちます。
- スレッショルドがログ・セグメントのスレッショルドではない場合は、エラー・ログにメッセージを出力します。メッセージには、データベース名、セグメント名、およびスレッショルドのサイズが含まれ、データベースのデータ・セグメントが満杯に近いことを知らせます。

```
create procedure sp_thresholdaction
     @dbname           varchar(30),
     @segmentname      varchar(30),
     @space_left       int,
     @status           int
as
declare @devname varchar(100),
           @before_size int,
           @after_size int,
           @before_time datetime,
           @after_time datetime,
           @error int

/*
** if this is a last-chance threshold, print a LOG FULL msg
** @status is 1 for last-chance thresholds,0 for all others
*/
if (@status&1) = 1
begin
      print "LOG FULL: database '%1!'", @dbname
end

/*
```

```
** if the segment is the logsegment, dump the log
** log segment is always "2" in syssegments
*/
if @segmentname = (select name from syssegments
               where segment = 2)
begin
     /* get the time and log size
     ** just before the dump starts
     */
     select  @before_time = getdate(),
      @before_size = reserved_pages(db_id(), object_id("syslogs"))
      print "LOG DUMP: database '%1!', threshold '%2!'",
        @dbname, @space_left

      select @devname = "/backup/" + @dbname + "_" +
        convert(char(8), getdate(),4) + "_" +
        convert(char(8), getdate(), 8)

      dump transaction @dbname to @devname

      /* error checking */
      select @error = @@error
      if @error != 0
      begin
           print "LOG DUMP ERROR: %1!", @error
      end
/* get size of log and time after dump */
    select  @after_time = getdate(),
      @after_size = reserved_pages(db_id(), object_id("syslogs"))
      /* print messages to error log */
      print "LOG DUMPED TO: device '%1!", @devname
      print "LOG DUMP PAGES: Before: '%1!', After '%2!'",
           @before_size, @after_size
      print "LOG DUMP TIME: %1!, %2!", @before_time, @after_time end
        /* end of 'if segment = 2' section */
      else
       /* this is a data segment, print a message */
      begin
      print "THRESHOLD WARNING: database '%1!', segment '%2!' at '%3!' pages",
@dbname, @segmentname, @space_left
end
```

### スレッショルド・プロシージャの配置場所の決定

トランザクション・ログをダンプするプロシージャは、スレッショルドごとに別々に作成することもできますが、すべてのログ・セグメントのスレッショルドが実行する、単一のスレッショルド・プロシージャを作成する方が簡単です。セグメントの空き領域量がスレッショルドを下回ると、Adaptive Server は影響を受けるデータベースの systhresholds テーブルを読み込み、関連付けられているストアド・プロシージャの名前を検索します。このプロシージャは、次のいずれかです。

- Open Server に対するリモート・プロシージャ・コール
- データベース名によって修飾されたプロシージャ名 (例：sybsystemprocs.dbo.sp_thresholdaction)
- 修飾されていないプロシージャ名

プロシージャ名にデータベース修飾子が含まれない場合、Adaptive Server は空き領域不足が発生したデータベースの中でプロシージャを探します。その中にプロシージャが見つからず、プロシージャ名が文字 sp_ で始まる場合は、sybsystemprocs データベースの中で探し、次に master データベースの中で探します。

スレッショルド・プロシージャが見つからない場合、またはそのプロシージャを実行できない場合は、エラー・ログにメッセージが出力されます。

## データ・セグメントの空き領域計算の無効化

ログ・セグメント以外のセグメントの空き領域の計算を無効にするには、no free space acctg オプションを指定して sp_dboption を実行し、次に checkpoint コマンドを実行します。ログ・セグメントについては、空き領域の計算を無効にすることはできません。

空き領域の計算を無効にした場合、ログ・セグメントのスレッショルドのみが領域の使用状況をモニタします。データ・セグメントのスレッショルドを超えても、スレッショルド・プロシージャは実行されません。空き領域の計算を無効にすると、リカバリ時間が短くなります。これは、ログ・セグメント以外のセグメントのリカバリ中には、空き領域の再計算が行われないからです。

次に、production データベースの空き領域の計算を無効にする例を示します。

```
sp_dboption production,
     "no free space acctg", true
```

**警告！** 空き領域の計算が無効のときは、システム・プロシージャが領域の割り付けに関する正確な情報を返すことはできません。

データ・セグメントの空き領域の計算を無効にすると、no free space acctg を false に設定しても、カウントが不正確になる場合があります。再計算させるには、shutdown with nowait を発行して Adaptive Server を再起動します。この場合、リカバリに時間がかかることがあります。

# 索引

## 記号



## A



## B



## C

## D

## E



## F



## G



## H



## I

## L



## M



## N

## O



## P



## Q



## R



## S

## T

## U



## V



## W



## X



## あ

## い



## う



## え

## お



## か



## き



## く



## け

## こ



## さ

## し

## す



## せ

## そ



## た

## ち



## つ



## て

## と

## な



## ね



## の



## は

## ひ



## ふ

## へ



## ほ



## ま



## み



## む



## め

## れ

## ろ



## わ