SYBASE®
An SAP® Company

パフォーマンス＆チューニング・シリーズ

Sybase IQ 15.4

ドキュメント ID： DC00283-01-1540-01
改訂： 2011 年 11 月

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

目次

対象読者

このマニュアルは、Sybase® IQ の設定によってパフォーマンスを向上したいと考えているデータベース管理者、データベース設計者、開発者を対象としています。

パフォーマンスに関する考慮事項

パフォーマンスは、通常、応答時間とスループットで測定されます。適正な設計を行い、適切なインデックス付け方式を選択することによって、パフォーマンスを最大に向上させることができます。

応答時間
応答時間とは、1 つのタスクが完了するまでにかかる時間のことです。応答時間は、次の項目の影響により変化します。

- 競合の軽減と待機時間 (特にディスク I/O 待機時間) の短縮
- より高速なコンポーネントの使用
- リソースに必要な時間の短縮 (同時実行性の向上)

スループット
スループットは、一定の時間にどれだけの作業量が完了したかを表します。スループットは、通常、1 秒あたりのトランザクション数で表されますが、1 分、1 時間、1 日などの単位で測定する場合もあります。

設計上の考慮事項
正しく設定されたシステムで Sybase IQ を実行し、適切な設計を行い、適切なインデックス付け方式を選択することによって、最高のパフォーマンスを実現できます

その他、ハードウェアやネットワークを分析することによって、インストール環境のボトルネックを特定できます。

参照：
- データ・モデルの推奨事項 (43 ページ)
- システム・リソースの管理 (5 ページ)
- パフォーマンスのモニタリングとチューニング (37 ページ)
- クエリと削除の最適化 (97 ページ)

システム・リソースの管理

ハードウェアとソフトウェアの設定をチューニングすることにより、パフォーマンスとクエリの処理速度が向上します。

メモリ使用の最適化

Sybase IQ がメモリを割り付ける方法を理解することは、システムで最高のパフォーマンスを実現するのに役立ちます。

ページングによる使用可能メモリの増加

ページングにより使用可能なメモリの量は増えますが、最適なメモリ管理のために、ページ・スワッピングを回避したり、最小限に抑えたりする必要があります。

システムのメモリが不足している場合、パフォーマンスが大幅に低下することがあります。 このような場合、使用可能なメモリを増やす必要があります。 Sybase IQ に割り付け可能なメモリが多ければ多いほど、パフォーマンスも向上します。

ただし、システム内のメモリ量には常に一定の制限があるため、データの一部のみがメモリに格納され、残りのデータはディスク上に格納されるという状況が発生します。オペレーティング・システムが、ディスク上のデータを検索して取り出し、メモリ要求に対応する必要がある場合、これを「ページング」または「スワッピング」と呼びます。 メモリを適切に管理することの主な目的は、ページングやスワッピングを回避したり、最小限に抑えたりすることです。

最も頻繁に使用されるオペレーティング・システム・ファイルは、「スワップ・ファイル」です。メモリが消耗している場合、オペレーティング・システムがメモリのページをディスクにスワップして、新しいデータの領域を確保します。スワップされたページを再び呼び出すと、他のページがスワップされて、要求されたメモリ・ページが元に戻ります。ユーザのディスク使用率が高い場合、スワッピングには時間がかかります。通常は、スワッピングが起こらないようなメモリ編成にして、オペレーティング・システム・ファイルの使用を最小限に抑えてください。

Sybase IQ では、物理メモリを最大限利用するために、データベースに対するすべての読み込みと書き込みにバッファ・キャッシュを使用します。

注意： ディスク上のスワップ領域には、少なくとも物理メモリ全体を収容できるだけのサイズを確保します。 スワップ／ページ領域を複数の高速なディスクにストライピングすることが重要です。

参照：
- スワッピングをモニタするためのユーティリティ (6 ページ)
- サーバ・メモリ (6 ページ)
- バッファ・キャッシュの管理 (7 ページ)
- バッファ・キャッシュ・サイズの決定 (8 ページ)
- バッファ・キャッシュ・サイズの設定 (11 ページ)
- ページ・サイズの指定 (11 ページ)
- ユーザが多数存在する場合の最適化 (12 ページ)
- プラットフォーム固有のメモリ・オプション (14 ページ)

スワッピングをモニタするためのユーティリティ

オペレーティング・システムのユーティリティを使用して、システムでページングが過度に発生していないかどうかを調査します。

UNIX の **vmstat** コマンド、UNIX の **sar** コマンド、または Windows タスク マネージャを使用すると、実行中のプロセス数、ページアウト回数、スワップ回数についての統計を表示できます。この統計によって得た情報を使用して、システムでページングが過度に発生していないかどうかを調査し、必要に応じて調整を行ってください。たとえば、特殊な高速ディスクにスワップ・ファイルを配置します。

参照：
- ページングによる使用可能メモリの増加 (5 ページ)
- サーバ・メモリ (6 ページ)
- バッファ・キャッシュの管理 (7 ページ)
- バッファ・キャッシュ・サイズの決定 (8 ページ)
- バッファ・キャッシュ・サイズの設定 (11 ページ)
- ページ・サイズの指定 (11 ページ)
- ユーザが多数存在する場合の最適化 (12 ページ)
- プラットフォーム固有のメモリ・オプション (14 ページ)

サーバ・メモリ

Sybase IQ によって、バッファ、トランザクション、データベース、サーバのヒープ・メモリが割り付けられます。 共有メモリも使用できますが、非常に少量です。

オペレーティング・システム・レベルでは、Sybase IQ サーバ・メモリはヒープ・メモリで構成されます。 ほとんどの場合、Sybase IQ で使用されるメモリがヒープ・メモリか共有メモリかを気にする必要はありません。メモリ割り付けは、すべて自動的に処理されます。 ただし、Sybase IQ を実行する前に、オペレーティン

グ・システム・カーネルが共有メモリを使用するように正しく構成されていることを確認してください。

マルチプレックス・メモリの管理
マルチプレックスの各サーバは、独自のホスト上にある場合と、ホストを他のサーバと共有している場合があります。複数のサーバが同じシステム上にある場合、作業負荷の処理にかかる CPU 時間は、単一の組み合わされたサーバの場合とほとんど変わりません。しかし、独立した複数のサーバでは、単一の組み合わされたサーバより多くの物理メモリが必要になります。これは、各サーバが使用するメモリを他のサーバが共有できないからです。

参照：
- ページングによる使用可能メモリの増加 (5 ページ)
- スワッピングをモニタするためのユーティリティ (6 ページ)
- バッファ・キャッシュの管理 (7 ページ)
- バッファ・キャッシュ・サイズの決定 (8 ページ)
- バッファ・キャッシュ・サイズの設定 (11 ページ)
- ページ・サイズの指定 (11 ページ)
- ユーザが多数存在する場合の最適化 (12 ページ)
- プラットフォーム固有のメモリ・オプション (14 ページ)

バッファ・キャッシュの管理

デフォルトのキャッシュ・サイズ (メイン・キャッシュに 16MB、テンポラリ・キャッシュに 12MB) では、ほとんどのデータベースでサイズが不足します。IQ メイン・バッファとテンポラリ・バッファのキャッシュにできるだけ多くのメモリを割り付けます。

Sybase IQ では、バッファ・キャッシュに最も多くのメモリが必要です。Sybase IQ には、IQ ストア用とテンポラリ・ストア用の 2 つのバッファ・キャッシュがあります。 ページング、データベースへの挿入、バックアップやリストアなどのすべてのデータベース I/O 操作にこの 2 つのバッファ・キャッシュが使用されます。データがメモリ内に送られると、この 2 つのキャッシュのいずれかに格納されます。 すべてのユーザ接続は、これらのバッファ・キャッシュを共有します。Sybase IQ は、各接続にどのデータが関連付けられているかを追跡します。

参照：
- ページングによる使用可能メモリの増加 (5 ページ)
- スワッピングをモニタするためのユーティリティ (6 ページ)
- サーバ・メモリ (6 ページ)
- バッファ・キャッシュ・サイズの決定 (8 ページ)

バッファ・キャッシュ・サイズの決定

適切なバッファ・キャッシャ・サイズは、多くの要因により異なります。

- システムに搭載されている物理メモリの合計量
- Sybase IQ、オペレーティング・システム、その他のアプリケーションがそれぞれのタスクを実行するのに必要なメモリの量
- ロード、クエリ、またはその両方を実行するかどうか
- スキーマ設定とクエリ負荷

オペレーティング・システムとその他のアプリケーション

ほとんどのオペレーティング・システムは、ファイル・システム・バッファリングに使用できるメモリの多くを使用します。使用するオペレーティング・システムのバッファリング・ポリシーを理解して、メモリの過度の割り付けを回避してください。

Sybase IQ とともに動作するアプリケーションのメモリ要件については、オペレーティング・システムとアプリケーションのマニュアルを参照してください。

参照：

- メモリ・オーバヘッド (8 ページ)
- メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ (10 ページ)

メモリ・オーバヘッド

オペレーティング・システムや他のアプリケーションで使用する物理メモリ量が判明したら、Sybase IQ が必要とする残りのメモリ量を計算します。

ロー・パーティションとファイル・システム

UNIX システムの場合、ロー・パーティションではなくファイル・システムを使用するデータベースには、オペレーティング・システムによるファイル・バッファリング処理のために残りのメモリの 30% がさらに必要になります。 Windows では、`OS_FILE_CACHE_BUFFERING = 'OFF'` に設定し (新しいデータベースのデフォルト)、ファイル・システム・キャッシュを無効にしてください。

マルチユーザのデータベース・アクセス

マルチユーザ環境でデータベースのクエリを行う場合、Sybase IQ には「アクティブ」ユーザ 1 人あたり約 10MB のメモリが必要です。 アクティブ・ユーザとは、

同時にデータベースにアクセスしたり、データベースに対してクエリを実行したりするユーザのことです。 たとえば、Sybase IQ に接続しているユーザが 30 人でも、アクティブにデータベースを同時に使用しているユーザは 10 人ほどしかいないことがあります。

スレッド・スタックのメモリ
スレッドの処理には、少量のメモリが必要です。使用する Sybase IQ 処理スレッドが多くなるにつれ、必要なメモリも多くなります。 **-iqmt** サーバ・スイッチは、Sybase IQ のスレッド数を制御します。 **-iqtss** サーバ・スイッチと **-gss** サーバ・スイッチは、各スレッドに割り付けられたスタック・メモリの容量を制御します。IQ スタックに割り付けられたメモリの総量は、**(-gn * (-gss + -iqtss)) + (-iqmt * -iqtss)** の式で計算された値とほぼ同じになります。

ユーザの数に比例して、スレッドの処理に必要なメモリは増加します。**-gn** スイッチは、データベース・サーバが同時に実行できるタスクの数 (ユーザ要求とシステム要求の両方) を制御します。**-gss** スイッチは、これらのタスクを実行するサーバ実行スレッドのスタック・サイズをある程度制御します。IQ はこれらのワーカ・スレッドのスタック・サイズを、**(-gss + -iqtss)** の式を使用して計算します。

スレッドの合計数 **(-iqmt** と **-gn** の合計) が、現在のプラットフォームで使用できるスレッド数を超えないようにします。

その他のメモリ使用
すべてのコマンドとトランザクションが、ある程度のメモリを使用します。これまで説明してきた要因の他に、メモリを大量に使用する操作には次のものがあります。

- バックアップ。バックアップに使用される仮想メモリの量は、データベース作成時に指定された **IQ PAGE SIZE** によって決まります。この値はおよそ 2 * CPU の数 * 20 * (IQ PAGE SIZE/16) です。プラットフォームによっては、BACKUP コマンドの BLOCK FACTOR を調整するとバックアップのパフォーマンスが向上する場合がありますが、BLOCK FACTOR を増やすとメモリの使用量も増加します。
- データベースの検証と修復 データベース全体を検証すると、**sp_iqcheckdb** プロシージャは処理を開始する前に、すべての Sybase IQ テーブル、テーブルのそれぞれのフィールドとインデックスを開きます。Sybase IQ テーブルの数、テーブル内のカラムとインデックスの累積数によって、**sp_iqcheckdb** に必要な仮想メモリの量は大幅に異なります。必要なメモリ量を制限するには、**sp_iqcheckdb** オプションを使用して 1 つのインデックスまたはテーブルを検証または修復します。
- リーク・ブロックの削除 リーク削除操作でも、すべての Sybase IQ テーブル、ファイル、インデックスを開く必要があるため、データベース全体を検証するときに **sp_iqcheckdb** が使用するのと同じ容量の仮想メモリを使用します。

Sybase IQ テンポラリ・バッファ・キャッシュを使用して、使用ブロックを追跡します。

参照：

- オペレーティング・システムとその他のアプリケーション (8 ページ)
- メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ (10 ページ)

メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュ

キャッシュ・サイズの一般的なガイドラインでは、メイン・バッファ・キャッシュに 40%、テンポラリ・バッファ・キャッシュに 60% のメモリを割り付けます。このガイドラインに従って起動し、サーバのパフォーマンスをモニタして、必要に応じてキャッシュ・サイズを調整します。

バッファ・キャッシュと物理メモリ

Sybase IQ のメイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュに使用するメモリと、Sybase IQ メモリ・オーバヘッド、オペレーティング・システムとその他のアプリケーションに使用するメモリの合計が、システムの物理メモリを超えないようにしてください。

最良のパフォーマンスを得るためには、IQ メイン・バッファおよびテンポラリ・バッファのキャッシュにできるだけ多くのメモリを割り付けます。 たとえば、Sybase IQ を使用するマシンに 4GB の物理メモリがある場合、共有バッファ・キャッシュをメインとテンポラリに分けることができます。

その他の考慮事項

バッファ・キャッシュ・サイズの要件は、使用状況によって異なります。パフォーマンスを最大にするには、データベースへの挿入、問い合わせ、その両方を使用するそれぞれの場合に応じて設定を変更します。ただし、データベースへの挿入と問い合わせを両方使用する環境では、すべてのユーザによるデータベースの使用を中止し、バッファ・キャッシュ・オプションをリセットすることは容易でありません。このような場合は、ロードまたはクエリのどちらかのパフォーマンスを優先させてください。

注意：

- 上記のガイドラインでは、システムで同時にアクティブなデータベースは 1 つのみであると想定しています。複数のアクティブなデータベースがある場合は、使用するデータベース間で残りのメモリをさらに分ける必要があります。
- 一部の UNIX プラットフォームでは、他のサーバ・スイッチを設定してバッファ・キャッシュに使用可能なメモリを増やす必要があります。

参照：
- オペレーティング・システムとその他のアプリケーション (8 ページ)
- メモリ・オーバヘッド (8 ページ)

バッファ・キャッシュ・サイズの設定

Sybase IQ では、最初、メイン・バッファ・キャッシュのサイズは 16MB に、テンポラリ・バッファ・キャッシュのサイズは 12MB に設定されています。 アプリケーションの要件に応じて、メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュのデフォルト・サイズを変更します。

表 1 : バッファ・キャッシュ・サイズの設定

方法	使用時期	設定の有効期間
-iqmc と **-iqtc** のサーバ・スイッチ	次の方法が推奨されます。キャッシュ・サイズは起動時に設定してください。	サーバが起動してから停止するまで。 **-iqmc** と **-iqtc** のサーバ起動オプションは、サーバが実行されている間だけ有効なため、サーバを再起動するたびに指定する必要があります。

ページ・サイズの指定

ページ・サイズとバッファ・キャッシュ・サイズは、データベースのメモリ使用とディスク I/O スループットに影響します。

注意： ページ・サイズは変更できません。ページ・サイズによって、一部のデータベース・オブジェクトのサイズの上限と LOB 機能を使用できるかどうかが決まります。

ページ・サイズ
Sybase IQ は、ページ・サイズを単位として I/O を実行します。データベースを作成する場合は、カタログ・ストアと IQ ストアに別々のページ・サイズを指定します。 テンポラリ・ストアのページ・サイズは、IQ ストアと同じです。

カタログ・ストアのページ・サイズは、パフォーマンスに実質的な影響を与えません。デフォルト値 4096 バイトで十分です。IQ ページ・サイズによって、データベースのデフォルト I/O 転送ブロック・サイズと最大データ圧縮の 2 つのパフォーマンスの要因が決まります。

データ圧縮
Sybase IQ は、すべてのデータを圧縮します。圧縮量は、IQ ページ・サイズに基づいて決定されます。

メモリの節約
マシンに十分なメモリが搭載されていない場合は、メモリを増やしてバッファ・キャッシュ・サイズを小さくします。 ただし、バッファ・キャッシュを小さくしすぎると、バッファの不足によって、データのロードまたはデータの問い合わせが非効率的になったり、完了できなくなったりすることがあります。

参照：
- ページングによる使用可能メモリの増加 (5 ページ)
- スワッピングをモニタするためのユーティリティ (6 ページ)
- サーバ・メモリ (6 ページ)
- バッファ・キャッシュの管理 (7 ページ)
- バッファ・キャッシュ・サイズの決定 (8 ページ)
- バッファ・キャッシュ・サイズの設定 (11 ページ)
- ユーザが多数存在する場合の最適化 (12 ページ)
- プラットフォーム固有のメモリ・オプション (14 ページ)

ユーザが多数存在する場合の最適化

最大数のユーザをサポートするには、テンポラリ DB 領域を増やし、オペレーティング・システムのパラメータを調整し、起動パラメータを変更する必要があります。

起動オプション

多数のユーザが存在する場合には、次の起動オプションを使用します。

-gm
接続のデフォルト数を設定します。
```
-gm#_connections_to_support
```

この値は、サーバがサポートする接続の合計数を表しますが、すべての接続が同時にアクティブなわけではありません。

-iqgovern
指定すると、同時に実行されるクエリの最大数が制限されます。 **-iqgovern** の制限を超えるユーザがクエリを発行した場合は、アクティブなクエリのいずれかが完了するまで、新しいクエリはキューに入れられます。
```
-iqgovern#_ ACTIVE_ queries_to_support
```

-iqgovern の最適な値は、クエリの性質、CPU の数、Sybase IQ バッファ・キャッシュのサイズによって異なります。デフォルト値は 2**numCPU* + 10 です。接続ユーザ数が多い場合は、このオプションを 2**numCPU* + 4 に設定するとスループットが向上する場合があります。

-gn
複数のユーザが実行する場合に、カタログ・ストアと接続の要求を処理するために使用される実行スレッドの数を設定します。

```
-gn number of tasks (both user
and system requests) that the database server can execute
concurrently
```

-gn の適正値は、**-gm** の値によって決まります。**start_iq** ユーティリティが **-gn** を計算し、値を適切に設定します。**-gn** の設定値が小さすぎると、サーバが正しく機能しなくなることがあります。**-gn** は、480 以下に設定することをおすすめします。

-c
カタログ・ストア・キャッシュ・サイズを設定します。

```
-ccatalog_store_cache_size
```

カタログ・キャッシュ・サイズは、スキーマ・サイズとオブジェクト数に大きく依存します。カタログ・ストア・バッファ・キャッシュは、カタログ・ストアの汎用メモリ・プールでもあります。 MB 単位で指定するには、**-c nM** の形式を使用します。たとえば、**-c 64M** と指定します。Sybase の推奨値は次のとおりです。

表 2 : カタログ・バッファ・キャッシュの設定

ユーザ数	-c で設定する最小値
1000 まで	64MB
200 まで	48MB (64 ビットの場合の **start_iq** のデフォルト値)。ユーザ数がこれより多い場合は 64MB に設定すると有効

-cl と -ch
カタログ・ストア・キャッシュ・サイズの上限 (**-ch**) と下限 (**-cl**) を設定します。

-cl *minimum cache size* **-ch** *maximum cache size*

標準のカタログ・キャッシュ・サイズが小さすぎる場合は、**-cl** パラメータと **-ch** パラメータを設定します。32 ビット・プラットフォームでは次のように設定してみます。

```
-cl 128M -ch 256M
```

-c を、**-ch** または **-cl** と同じ設定ファイルまたはコマンド・ラインで使用しないでください。関連情報については、「**-ch cache-size** オプション」を参照してください。

警告！ カタログ・ストア・キャッシュ・サイズを明示的に制御するには、サーバ起動用の設定ファイル (`.cfg`) または UNIX コマンド・ラインで、次のいずれか一方を実行します。両方を実行しないでください。

- **-c** パラメータを設定する。
- **-ch** パラメータと **-cl** パラメータを使用して、カタログ・ストア・キャッシュ・サイズに特定の上限と下限を設定する。

上記のパラメータをこれ以外の組み合わせで指定すると、予期しない結果が生じることがあります。

-iqmt
処理スレッドの数を設定します。

-gm 設定に対して、小さすぎる値を **-iqmt** に設定すると、スレッドが不足することがあります。

プラットフォーム固有のメモリ・オプション

使用可能な合計メモリ量を制限するのは、システムの仮想メモリだけです。

連結メモリ・プール
HP と Solaris のプラットフォームでは、指定した量のメモリを連結メモリとして指定できます。 連結メモリは、物理メモリにロックされた共有メモリです。 カーネルはこのメモリを物理メモリからページ・アウトできません。

他のアプリケーションが同じマシン上で同時に実行されている場合は、連結メモリによって Sybase IQ のパフォーマンスが向上することがあります。 ただし、連結メモリを Sybase IQ 専用に割り付けると、そのメモリはマシン上の他のアプリケーションが利用できなくなります。

これらの UNIX プラットフォームにのみ連結メモリのプールを作成するには、**-iqwmem** コマンド・ライン・スイッチを指定して、連結メモリの MB 数を指定します (Solaris 以外のプラットフォームで **-iqwmem** を設定するには、**root** ユーザの権限が必要です)。 64 ビット・プラットフォームでは、**-iqwmem** の上限はマシンの物理メモリのみです。

たとえば、14GB のメモリを搭載するマシンで、10GB の連結メモリを確保するとします。そのためには、次のように指定します。

```
-iqwmem 10000
```

注意： **-iqwmem** は、専用の連結メモリを割り付けるために十分なメモリがある場合にのみ使用します。メモリが十分にないときにこのスイッチを使用すると、パフォーマンスが著しく低下することがあります。

- Solaris では、**-iqwmem** を指定すると、常に連結メモリが有効になります。
- HP では、サーバを root ユーザで起動した場合に、**-iqwmem** を指定すると連結メモリが有効になります。root ユーザ以外のユーザでサーバを起動した場合

は、非連結メモリが有効になります。この動作は、将来のバージョンで変更される可能性があります。

他のアプリケーションとデータベースの影響
サーバに使用されるメモリは、すべてのアプリケーションとデータベースに使用されるメモリ・プール内のメモリです。複数のサーバまたは複数のデータベースを同時に同じマシン上で実行したり、他のアプリケーションを実行したりしている場合は、サーバが要求するメモリ量を減らす必要があります。

また、UNIX コマンド `ipcs -mb` を発行して、実際のセグメント数を表示することもできます。

HP のメモリ問題のトラブルシューティング
HP-UX では、maxdsiz_64bit カーネル・パラメータの値を調べます。 このパラメータは、64 ビット HP プロセッサ上で Sybase IQ が使用できる仮想メモリの量を制限します。 推奨値については、『インストールおよび設定ガイド』を参照してください。

ファイル・システム・バッファリングの制御

一部のファイル・システムでは、ファイル・システム・バッファリングのオンとオフを切り替えることができます。ファイル・システム・バッファリングをオフにすると、通常、ページングが減り、パフォーマンスが向上します。

既存のデータベースの IQ メイン DB 領域のファイル・システム・バッファリングを無効にするには、次の文を発行します。

```
SET OPTION "PUBLIC".OS_FILE_CACHE_BUFFERING = OFF
```

既存のデータベースの IQ テンポラリ DB 領域のファイル・システム・バッファリングを無効にするには、次の文を発行します。

```
SET OPTION "PUBLIC".OS_FILE_CACHE_BUFFERING_TEMPDB = OFF
```

このオプションは、PUBLIC グループにのみ設定できます。変更を有効にするには、データベースを停止し、再起動します。

このダイレクト I/O パフォーマンス・オプションは、Solaris UFS、Linux、Linux IBM、AIX、Windows ファイル・システムでのみ有効です。 このオプションは HP-UX と HP-UXi には影響しません。また、ロー・ディスク上に構築されたデータベースにも影響はありません。 Linux では、ダイレクト I/O はカーネル・バージョン 2.6.x でサポートされます。

Linux カーネル・バージョン 2.6 および AIX でダイレクト I/O を有効にするには、環境変数 IQ_USE_DIRECTIO を 1 に設定します。Linux カーネル・バージョン 2.6 および AIX では、ダイレクト I/O はデフォルトで無効になっています。IQ_USE_DIRECTIO は、Solaris と Windows には影響しません。

注意：

- Sybase IQ は、Linux カーネル・バージョン 2.4 でダイレクト I/O をサポートしていません。 Linux カーネル・バージョンで IQ_USE_DIRECTIO 環境変数を設定すると、Sybase IQ サーバは起動しません。 エラー "`Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS`" が報告されます。
- Solaris には、ファイル・システム・バッファ・キャッシュのサイズを制限するカーネル・パラメータはありません。 時間の経過とともにファイル・システム・バッファ・キャッシュが大きくなり、IQ バッファ・キャッシュ・ページに取って代わるため、オペレーティング・システムに過度のページング・アクティビティが発生し、Sybase IQ のパフォーマンスが低下します。 そのため Sybase では、Solaris のデータベース用にはロー・デバイスを強くおすすめします。
- Windows では、ファイル・システムよりもアプリケーションを優先するページング・アルゴリズムを使用できます。 これは、Sybase IQ パフォーマンスの向上に役立ちます。

参照：

- Java 実行可能のデータベースのオプション (16 ページ)

Java 実行可能のデータベースのオプション

`JAVA_HEAP_SIZE` オプションを設定することにより、制御できなくなった Java アプリケーションがメモリを使いすぎないようにできます。

`SET OPTION` コマンドの `JAVA_HEAP_SIZE` オプションは、Java アプリケーションに対して接続ごとに割り付けるメモリの最大サイズ (バイト単位) を設定します。 通常、接続ごとに割り付けられるメモリは、ユーザが作業環境で使用する Java 変数と Java アプリケーション・スタック領域の組み合わせです。 Java アプリケーションの実行中、接続ごとの割り付けはデータベース・サーバの固定キャッシュを使用するため、制御できなくなった Java アプリケーションがメモリを使いすぎないようにすることが重要です。

参照：

- ファイル・システム・バッファリングの制御 (15 ページ)

プロセス・スレッド・モデル

Sybase IQ では、最大限のパフォーマンスを得るために、オペレーティング・システムのカーネル・スレッドを使用します。 デフォルトでは、Sybase IQ はシステムの CPU 数を基にしてスレッド数を割り付けます。

ライトウェイト・プロセスは、カーネルでサポートされるコントロールの基本となるスレッドです。オペレーティング・システムによって、どのライトウェイト・プロセス (LWP) をどのプロセッサでいつ実行するかが決定されます。オペレーティング・システムはユーザ・スレッドのことは関知しませんが、ユーザ・スレッドが待機中か実行可能かは認識しています。

オペレーティング・システムのカーネルによって、LWP が CPU リソース上にスケジューリングされます。この場合、LWP のスケジューリング・クラスと優先度を使用します。各 LWP は、カーネルによって個別にディスパッチされ、個別のシステム呼び出しを実行し、個別のページ・フォルトを発生させ、マルチプロセッサ・システム上では並列に実行します。

高度にスレッド化された単一のプロセスが、すべての Sybase IQ ユーザの処理を実行します。Sybase IQ は、接続によって実行される処理の種類、使用可能な合計スレッド数、さまざまなオプションの設定に基づいて、各ユーザ接続にさまざまな数のカーネル・スレッドを割り当てます。

スレッド不足エラー
クエリ処理に必要なサーバ・スレッドが不足している場合、Sybase IQ は次のエラーを生成します。

```
Not enough server threads available for this query
```

この状況は、すぐに解消される場合もあります。 他のクエリが完了してからクエリを発行すると、使用可能なスレッドが増えるため、クエリが成功する場合があります。 状況が持続する場合は、サーバを再起動し、より多くの Sybase IQ スレッドを指定する必要があります。 接続数に対して **-iqmt** に設定されている値が小さすぎる可能性もあります。

スレッド使用を管理するための Sybase IQ オプション

- 最大スレッド数を設定するには、サーバ起動オプション **-iqmt** を設定します。デフォルト値は接続数と CPU 数によって計算され、通常、デフォルト値をそのまま使用できます。
- 内部実行スレッドのスタック・サイズを設定するには、サーバ起動オプション **-iqtss** を設定します。通常はデフォルト値で十分ですが、複雑なクエリを実行

したときに、スタックの深さがこの制限を超えていることを示すエラーが返された場合は、値を増やします。
- ユーザ 1 人あたりに使用するスレッド数の最大値を設定するには、SET OPTION MAX_IQ_THREADS_PER_CONNECTION コマンドを使用します。 SET OPTION MAX_IQ_THREADS_PER_TEAM は、スレッドのチームで使用可能なスレッド数を設定します。
  特定の操作に使用するリソースの量を制御する場合にも、これらのオプションを使用します。 たとえば、DBA は INSERT、LOAD、BACKUP、または RESTORE のコマンドを発行する前にこのオプションを設定できます。

参照：
- パフォーマンスに関する考慮事項 (3 ページ)
- メモリ使用の最適化 (5 ページ)
- I/O の分散 (18 ページ)
- リソース使用を調整するオプション (24 ページ)
- リソースを効率的に利用するための他の方法 (31 ページ)
- インデックスのヒント (43 ページ)
- データベース・サイズと構造の管理 (32 ページ)
- ロードを高速化するための UNION ALL ビュー (59 ページ)
- ネットワーク・パフォーマンス (33 ページ)

I/O の分散

ディスク・ストライピング、ファイル・ディスクへのランダム／順次アクセス、メッセージ・ログ・ファイルのサイズ制御方法について説明します。

ロー I/O (UNIX オペレーティング・システム)

データベースまたは DB 領域をロー・デバイスまたはファイル・システム・ファイルに作成できます。

ディスク・パーティションへのアクセスは、通常、 ファイル・システム・モード (UFS ファイル・システム経由など) またはロー・モードの 2 種類のモードで行われます。 ロー・モードではバッファを使用しない I/O を行い、通常、読み取りまたは書き込みのシステム呼び出しごとにデバイスに対するデータ転送を行います。 UNIX のデフォルト・ファイル・システムである UFS は、バッファを使用する I/O システムであり、バッファにデータを蓄積してからバッファ全体を一度に転送します。

データベースまたは DB 領域をロー・デバイスまたはファイル・システム・ファイルに作成します。Sybase IQ は、指定されたパス名から、ロー・パーティション

かファイル・システム・ファイルのどちらかを自動的に判断します。ロー・パーティションは任意のサイズに設定できます。

詳細については、『システム管理ガイド：第 1 巻』の「データベース・オブジェクトの管理」>「データベース・オブジェクトの編集ツール」を参照してください。

参照：

- ディスク・ストライピングの使用ディスク・ストライピングは、複数のディスク・ドライブに 1 つのファイルのデータを分散する場合に使用する一般的な方法です。ストライプ・ディスクは、1 台のディスクを使用するよりも、パフォーマンスが大幅に向上します。
- 内部ストライピング (20 ページ)
- 複数のファイルの使用 DB 領域内の複数のファイルを使用することで、スループットが向上し、DB 領域の平均遅延時間が短縮します。
- 戦略的なファイルの格納 (21 ページ)
- 挿入、削除、同期のための作業領域 Sybase IQ でデータの INSERT、DELETE、同期を行うためには、IQ ストアに作業領域が必要です。これらのトランザクションがコミットされると、この作業領域は他の目的に再利用されます。
- 予約領域のオプションの設定 MAIN_RESERVED_DBSPACE_MB と TEMP_RESERVED_DBSPACE_MB のオプションは、Sybase IQ が特定の操作のために予約する領域の量を制御します。

Sybase IQ とディスク・ストライピング

複数のディスクにわたるデータのストライピングは、優れたパフォーマンスを得るために重要な手法です。

ディスク・ストライピングはシステムのさまざまな場所で実行できますが、RAID ハードウェアや RAID ソフトウェアの一部として実行することが一般的です。次のような使用例があります。

- ディスク・アレイやコントローラなどのデバイス・レイヤで実行する。
- オペレーティング・システムや、Veritas などの専用デバイス管理ソフトウェアで実行する。
- アプリケーションで実行する。

IQ のデフォルトでは、DB 領域内のすべてのファイルにわたるページのストライピングが内部的に行われています。そのため、パフォーマンス向上の目的で、ソフトウェアまたはハードウェア・レベルでさらにストライピングを行う必要はありません。もちろん、RAID-5 を使用する場合など、データベースの記憶域に冗長構成を実装する作業の一部として、ストライピングの追加が必要となる場合もあります。

記憶域を冗長構成した Sybase IQ では、「RAID-1」による単純なミラーリングによって最高のパフォーマンスが得られます。前述のように、Sybase IQ は DB 領域内の 2 台セットのミラー・ディスク全体にわたってデータを分散します。

コストの関係で、ほとんどの Sybase IQ データベースではミラーリングが行われず、RAID-5 または同レベルの RAID の実装によって冗長構成を実現しています。RAID-5 では、チャンク・サイズ (次のディスクに移る前にディスク 1 台に書き込まれるデータ量) の適切な選択が、システムのパフォーマンスに大きく影響します。これは、RAID-5 には著しい書き込みオーバヘッドがあるからです。アプリケーションで行われる読み込み、更新、削除の処理頻度や緊急度が高い場合や、クエリによって DB 領域に対する一時的な I/O が行われることが多い場合は、Sybase IQ データベース・ページの 25 ～ 50% に相当する小さなチャンク・サイズを選択することによって、最適なパフォーマンスを得られる可能性が高くなります。アプリケーションの処理の大部分が読み込みで、書き込みがほとんど行われない場合は、IQ ページ・サイズの 75 ～ 100% に相当する大きなチャンク・サイズを選択することで最適なパフォーマンスを実現できます。

Sybase IQ では通常、アクティブなクエリが 1 つだけの場合でも、複数の読み込みのプリフェッチや複数の書き込みのフラッシュが並列処理として試行されます。そのため、非常に小さなサイズのチャンクを使用して、各ページの読み込みや書き込みを多くのディスクに分散してもメリットはほとんどなく、たいていはパフォーマンスに悪影響を及ぼします。

RAID を使用する場合に最適なパフォーマンスを実現するには、通常はコントローラやアレイなどのハードウェア・ベースの RAID を使用します。ソフトウェア・ベースの RAID ツールも十分に機能しますが、サーバの CPU のパフォーマンスに適度な負荷が加わる場合があります。

内部ストライピング

ディスク・ストライピングでは、複数のディスク・スピンドルを使用して高速な並列ディスク書き込みを行います。

Sybase IQ では、サードパーティ製のソフトウェアを使用せずにディスク・ストライピングを可能にするオプションが用意されています。 サードパーティ製のソフトウェアとハードウェアによるディスク・ストライピングを使用している場合は、次の説明に従う必要はありません。 CREATE DBSPACE コマンドに STRIPING ON オプションを指定することにより、ディスク・ストライピングを有効にできます。

ディスク・ストライピングの ON/OFF

DB 領域の作成時にデフォルトのストライピングを変更するために使用する構文は、次のとおりです。

```
SET OPTION "PUBLIC".DEFAULT_DISK_STRIPING = { ON | OFF }
```

すべてのプラットフォームで DEFAULT_DISK_STRIPING オプションのデフォルト値は **ON** です。ディスク・ストライピングが **ON** の場合、入力データは、使用可能な領域があるすべての DB 領域に分散されます。ディスク・ストライピングが **OFF** の場合は、論理ファイルの先頭から DB 領域 (ディスク・セグメント) に格納され、一度に 1 つのディスク・セグメントが格納されます。

DEFAULT_DISK_STRIPING の値を変更する場合、ストライピングの優先を指定しないすべての後続の CREATE DBSPACE 操作に影響を与えます。

ディスク・ストライピングが ON の場合、ALTER DBSPACE DROP コマンドを使用して DB 領域からファイルを削除できます。ただし、DB 領域を削除する前に sp_iqemptyfile ストアド・プロシージャを使用して、DB 領域内のすべてのデータを再配置します。ディスク・ストライピングではデータが複数のファイルに分散されるため、sp_iqemptyfile プロセスには多数のテーブルとインデックスの再配置が必要になることがあります。 sp_iqdbspaceinfo ストアド・プロシージャと sp_iqdbspace ストアド・プロシージャを使用して、DB 領域に存在するテーブルとインデックスを確認します。

参照：

- ロー I/O (UNIX オペレーティング・システム) (18 ページ)
- ディスク・ストライピングの使用ディスク・ストライピングは、複数のディスク・ドライブに 1 つのファイルのデータを分散する場合に使用する一般的な方法です。ストライプ・ディスクは、1 台のディスクを使用するよりも、パフォーマンスが大幅に向上します。
- 複数のファイルの使用 DB 領域内の複数のファイルを使用することで、スループットが向上し、DB 領域の平均遅延時間が短縮します。
- 戦略的なファイルの格納 (21 ページ)
- 挿入、削除、同期のための作業領域 Sybase IQ でデータの INSERT、DELETE、同期を行うためには、IQ ストアに作業領域が必要です。これらのトランザクションがコミットされると、この作業領域は他の目的に再利用されます。
- 予約領域のオプションの設定 MAIN_RESERVED_DBSPACE_MB と TEMP_RESERVED_DBSPACE_MB のオプションは、Sybase IQ が特定の操作のために予約する領域の量を制御します。

戦略的なファイルの格納

記憶領域リソースを追加して、ファイル・ディスク I/O を改善します。

ランダム・アクセス・ファイル専用のディスク・ドライブ数と、これらのファイルに対して実行される 1 秒あたりの処理数を増やすことによって、ランダム・アクセス・ファイルに関連するパフォーマンスを改善することができます。 ランダム・ファイルには、IQ ストア、テンポラリ・ストア、カタログ・ストア、プログ

ラム (Sybase IQ 実行ファイル、ユーザ・プロシージャとストアド・プロシージャ、アプリケーション)、オペレーティング・システム・ファイルのランダム・ファイルがあります。

一方、順次アクセス・ファイルに関連するパフォーマンスは、専用ディスク・ドライブに格納し、他のプロセスとの競合をなくすことによって向上させることができます。順次ファイルには、トランザクション・ログやメッセージ・ログ・ファイルがあります。

ディスク・ボトルネックを防止するために、次の注意に従ってください。

- ランダム・ディスク I/O を順次ディスク I/O から分離する。また、パフォーマンスを最大にするために、DB 領域ごとに 1 つの物理デバイス (ディスクまたは HW RAID セット) から 1 つのパーティションのみを使用する。
- Adaptive Server® Enterprise や I/O を多用する他のアプリケーションのデータベース I/O から Sybase IQ データベースの I/O を分離する。
- データベース・ファイル、テンポラリ DB 領域、トランザクション・ログ・ファイルをデータベース・サーバと同じ物理マシン上に配置する。

トランザクション・ログ

トランザクション・ログ・ファイルにはリカバリと監査に関する情報が含まれています。

トランザクション・ログ・ファイルを移動したり、ファイル名を変更したりするには、トランザクション・ログ・ユーティリティ (**dblog**) を使用します。『ユーティリティ・ガイド』の「dblog データベース管理ユーティリティ」を参照してください。

警告！ Sybase IQ のトランザクション・ログ・ファイルは、多くのリレーショナル・データベースのトランザクション・ログ・ファイルとは異なります。 なんらかの理由で (ログ・ファイルではなく) データベース・ファイルが失われた場合は、データベースが失われます。ただし、バックアップを正しく実行している場合は、データベースを再ロードできます。

トランザクション・ログのトランケーション

トランザクション・ログは、時間経過とともに大量のディスク領域を使用する場合があります。 ディスク領域を節約するには、定期的にログをトランケートします。

ログをトランケートする頻度は、ログ・ファイルの増大の度合いとサイトの運用手順に基づいて、Sybase IQ システムのサポートを担当している DBA が決定します。

参照：

- メッセージ・ログ (23 ページ)

- 停止したデータベースのトランザクション・ログをトランケートする (23 ページ)

停止したデータベースのトランザクション・ログをトランケートする
–m 起動スイッチを使用して、トランザクション・ログをトランケートします。

–m サーバ起動スイッチを使用して Sybase IQ を起動すると、トランザクション・ログのトランケートのみが実行されます。 **–m** を永続的に設定したままにしないでください。これをどのように行うかは DBA 次第ですが、次に示す手順を参考にしてください。

1. サーバ・スイッチ `.cfg` ファイルのコピーを作成し、ログのトランケーション設定用のファイルであることを示す名前を付けます。このファイルを編集し、**–m** スイッチを追加します。
2. **–m** オプションが含まれる設定ファイルを使って Sybase IQ を起動します。 この時点では、ユーザ・アクセスやトランザクションを許可しないでください。
3. Sybase IQ を停止し、**–m** オプションが設定されていない設定ファイルを使用して再起動します。

参照：
- トランザクション・ログのトランケーション (22 ページ)
- メッセージ・ログ (23 ページ)

メッセージ・ログ
ディスク領域を節約するには、メッセージ・ログのサイズを制限します。

Sybase IQ は、エラー、状態、挿入通知の各メッセージを含むすべてのメッセージをメッセージ・ログ・ファイルに記録します。`LOAD` 文と `INSERT` 文のパラメータを使用して、通知メッセージを OFF に設定できます。

サイトによっては、挿入の数、**`LOAD`** オプションと `NOTIFY_MODULUS` データベース・オプションの低い設定、その他の条件が原因で、メッセージ・ログ・ファイルが急速に増大することがあります。 Sybase IQ では、メッセージ・ログをラッピングするか、またはファイルの最大サイズを設定してアクティブな Sybase IQ メッセージ・ログが満杯になったときにログ・ファイルをアーカイブすることで、ファイルのサイズを制限できます。

ログ・ファイルの最大サイズの設定、メッセージ・ログ・ファイルのアーカイブ、メッセージ・ログ・ラッピングの有効化の詳細については、『システム管理ガイド： 第 1 巻』の「Sybase IQ システム管理の概要」>「メッセージ・ロギング」を参照してください。

参照：

- トランザクション・ログのトランケーション (22 ページ)
- 停止したデータベースのトランザクション・ログをトランケートする (23 ページ)

リソース使用を調整するオプション

リソースを調整して、クエリをより高速に実行します。

同時クエリの制限

-iqgovern スイッチを設定して、特定のサーバでの同時クエリ数を指定します。これは、ライセンスによって規制される接続数とは異なります。

-iqgovern には、適切な同時クエリ・アクセス数を設定して、スループットを最適化するために最適な値があります。**-iqgovern** をこのしきい値より大きく設定すると、競合またはリソース不足が発生して、すべての要求の処理速度が低下します。

-iqgovern スイッチを指定することによって、Sybase IQ はディスクへのバッファ・データのページングを最適化し、メモリの過剰使用を防止できます。 **-iqgovern** のデフォルト値は (2 x CPU 数) + 10 です。場合によっては、いろいろな値を試して最適な値を見つける必要があります。 アクティブな接続が多数あるサイトの場合は、**-iqgovern** を多少低めに設定してみてください。

参照：

- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

使用可能な CPU 数の設定

-iqnumbercpus 起動スイッチを設定して、使用できる CPU 数を設定します。このパラメータは、リソース計画を目的として CPU の物理的な数を上書きします。

-iqnumbercpus スイッチは、次のマシンでのみ使用することをおすすめします。

- Intel® CPU を搭載し、ハイパースレッディングが有効になっていて、**-iqnumbercpus** が実際のコア数に設定されているマシン
- オペレーティング・システムのユーティリティを使って、Sybase IQ で使用可能な CPU が、マシンにある CPU の一部に制限されているマシン

『システム管理ガイド：第 1 巻』の「Sybase IQ の起動」>「CPU 数のスイッチ」を参照してください。

参照：

- 同時クエリの制限 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

クエリによるテンポラリ DB 領域の使用の制限

`QUERY_TEMP_SPACE_LIMIT` を設定して、クエリが受け付けられる最大のテンポラリ領域の予測容量を指定します。予測容量がこの値を超えると、クエリは拒否されます。

`QUERY_TEMP_SPACE_LIMIT` オプションを設定すると、予測されるテンポラリ領域の使用率が指定されたサイズを超える場合に、クエリが拒否されます。 デフォルトでは、クエリによるテンポラリ・ストアの使用率に制限はありません。

Sybase IQ は、クエリの解析に必要なテンポラリ領域のサイズを推定します。 推定が現在の `QUERY_TEMP_SPACE_LIMIT` 設定を超えると、Sybase IQ は次のエラーを返します。

```
Query rejected because it exceeds total space resource limit
```

このオプションを 0 (デフォルト) に設定すると、制限がないため、テンポラリ領域の条件によってクエリが拒否されることはありません。

接続ごとのテンポラリ・ストアの実際の使用率を制限するには、クエリを含むすべての DML 文に `MAX_TEMP_SPACE_PER_CONNECTION` オプションを設定します。 `MAX_TEMP_SPACE_PER_CONNECTION` は、文によるテンポラリ・ストアの実際の実行時の使用率をモニタして制限します。接続が `MAX_TEMP_SPACE_PER_CONNECTION` オプションで設定された割り当てを超えた場合は、エラーが返され、現在の文はロールバックされます。

参照：
- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

返されるローによるクエリの制限

QUERY_ROWS_RETURNED_LIMIT オプションを設定して、オプティマイザが、大量の結果セットを返す可能性のあるクエリを拒否しないようにします。

QUERY_ROWS_RETURNED_LIMIT オプションを設定すると、クエリ・オプティマイザは、大量のリソースを消費する可能性のあるクエリを拒否します。 クエリからの結果セットがこのオプションの値を超えると推定される場合、クエリ・オプティマイザはクエリを拒否し、次のメッセージが表示されます。

```
Query rejected because it exceed resource: Query_Rows_Returned_Limit
```

このオプションを使用する場合は、大量のリソースを消費するクエリのみを拒否するように設定します。

参照：
- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

カーソルのスクロールの禁止

大量の結果セットを返すクエリのテンポラリ・ストア・ノードを除去して、パフォーマンスを向上させます。

ホスト変数を宣言せずにカーソルのスクロールを使用すると、Sybase IQ は、クエリ結果をバッファするテンポラリ・ストア・ノードを作成します。 これは、テンポラリ・ストア・バッファ・キャッシュとは異なります。 テンポラリ・ストア・ノードを使用すると、アプリケーションで結果セットを検索するときに前方および後方に効率的にスクロールできます。

ただし、クエリが出力に大量 (数百万) のローを返す場合、およびアプリケーションによって行われるスクロールのほとんどが前方スクロールの場合は、テンポラリ・ストア・ノードのメモリ要件によってクエリのパフォーマンスが低下する可能性があります。パフォーマンスを向上させるには、次のコマンドを発行してテンポラリ・ストア・ノードを除去します。

```
SET TEMPORARY OPTION FORCE_NO_SCROLL_CURSORS = 'ON'
```

注意： アプリケーションが後方スクロールを行うことが多い場合、FORCE_NO_SCROLL_CURSORS オプションを ON に設定すると、クエリのパフォーマンスが実際に低下することがあります。これは、テンポラリ・キャッシュが存在しないため、後方スクロールのたびに Sybase IQ によるクエリの再実行が強制されるからです。

フロントエンド・アプリケーションで後方スクロールがほとんど行われない場合は、FORCE_NO_SCROLL_CURSORS = 'ON' を永続的な PUBLIC オプションに設定します。 メモリの節約になるため、クエリのパフォーマンスが向上します。

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

カーソル数の制限

MAX_CURSOR_COUNT オプションを設定して、単一の接続が、使用できるメモリまたは CPU のリソースを大量に使用しないようにします。

MAX_CURSOR_COUNT オプションは、1 つの接続が同時に使用できるカーソルの最大数を制限します。 デフォルトの値は 50 です。このオプションを 0 に設定すると、カーソル数は無制限になります。

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

文の数の制限

MAX_STATEMENT_COUNT オプションを設定して、1 つの接続が作成できる準備文の数を制限します。

MAX_STATEMENT_COUNT オプションは、1 つの接続が同時に使用できる準備文の最大数を制限します。 デフォルト数 (50) を超える準備文を任意の接続に対して同時にサポートする必要がある場合は、MAX_STATEMENT_COUNT オプションをより大きな値に設定できます。

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)

- プリフェッチされるローの数の制御 (30 ページ)

キャッシュ・ページのプリフェッチ

BT_PREFETCH_MAX_MISS オプションを設定して、プリフェッチ・メモリの動作を制御します。

BT_PREFETCH_MAX_MISS オプションは、特定のクエリでページのプリフェッチを継続するかどうかを決定します。 HG インデックスを使用するクエリの実行速度が予想より遅い場合は、このオプションの値を徐々に増やしてみます。

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- 一般的な使用のための最適化 (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

一般的な使用のための最適化

USER_RESOURCE_RESERVATION オプションを設定して、現在のユーザ数を考慮してメモリ使用を調整します。

Sybase IQ は、開いたカーソルの数を追跡して、メモリを割り付けます。 特定の状況においては、USER_RESOURCE_RESERVATION オプションの設定によって、製品を使用していると Sybase IQ が判断する現在のカーソル数の最小値を調整し、テンポラリ・キャッシュから割り付けるメモリをさらに節約できます。

このオプションは、慎重な分析の結果、実際に必要であると判断された場合にのみ設定する必要があります。このオプションを設定する場合は、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。

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)

- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- プリフェッチされるローの数の制御 (30 ページ)

プリフェッチされるローの数の制御

PrefetchRows パラメータと PrefetchBuffer パラメータを設定して、特定条件下でのカーソルのパフォーマンスを改善します。 これは、ODBC 接続ダイアログまたは `.odbc.ini` ファイルで設定できるクライアント・オプションです。

プリフェッチは、相対位置 1 または相対位置 0 のみをフェッチするカーソルのパフォーマンスを向上させます。2 つの接続パラメータを使用して、カーソル・プリフェッチのデフォルトを変更できます。PrefetchRows (PROWS) は、プリフェッチされるローの数を設定します。PrefetchBuffer (PBUF) は、プリフェッチされたローを格納するために、この接続に使用できるメモリを設定します。プリフェッチするローの数を増やすと、次の特定の条件ではパフォーマンスが向上する可能性があります。

- アプリケーションが数回の絶対フェッチで数多くのロー (数百ロー以上) をフェッチする場合
- アプリケーションがローを大量にフェッチし、かつ、クライアントとサーバが同じマシン上にあるか高速ネットワークで接続されている場合
- クライアント／サーバ通信がダイヤルアップ・リンクやワイド・エリア・ネットワークなどの低速ネットワークで行われている場合

参照：

- 同時クエリの制限 (24 ページ)
- 使用可能な CPU 数の設定 (24 ページ)
- クエリによるテンポラリ DB 領域の使用の制限 (25 ページ)
- 返されるローによるクエリの制限 (26 ページ)
- カーソルのスクロールの禁止 (27 ページ)
- カーソル数の制限 (28 ページ)
- 文の数の制限 (28 ページ)
- キャッシュ・ページのプリフェッチ (29 ページ)
- 一般的な使用のための最適化 (29 ページ)

リソースを効率的に利用するための他の方法

パフォーマンスを向上させ、ディスク領域をさらに有効に活用するためのシステムの調整方法がいくつかあります。

マルチプレックス・データベースのディスク領域の管理

ユーザに現在のトランザクションを定期的にコミットさせることにより、書き込みサーバで古いバージョンのテーブルが削除され、ディスク・ブロックが解放されます。 `auto_commit` オプションを指定すると、バージョン数の増大が最小限に抑えられ、領域の最小化に役立ちます。

ユーザがいずれかのサーバで、古いバージョンのテーブルを必要とするトランザクションを実行している間は、Sybase IQ はそのテーブルを削除できません。このため、マルチプレックス・データベースでテーブルの更新とクエリが同時に発生すると、Sybase IQ が大量のディスク領域を使用することがあります。 使用される領域の量は、データとインデックスの性質および更新の頻度によって決まります。

クエリする必要がなくなった古いバージョンを書き込みサーバが削除できるようにすれば、ディスク・ブロックを解放できます。古いテーブル・バージョンをリカバリできるように、すべてのサーバのユーザ全員が現在のトランザクションを定期的にコミットする必要があります。これで、サーバは稼働し続けることができ、すべての機能を利用できます。`sp_iqversionuse` ストアド・プロシージャを使用して、リモート・サーバで使用されているバージョンを表示できます。

参照：

- クエリ・サーバ間のロード・バランス (32 ページ)
- データベース・アクセスの制限使用頻度の少ない時間に更新をスケジュールして、クエリのパフォーマンスを向上させます。
- ディスクのキャッシュ大量のメモリを有効に使用して、実メモリに対する需要とディスクからのデータ読み取りのニーズのバランスを保ちます。

論理サーバを使用したマルチプレックス・リソースの管理

論理サーバを使用することにより、マルチプレックス・リソースの使用を最も効果的に管理できます。論理サーバを使用してアプリケーションごとに異なるマルチプレックス・サーバのセットを割り当て、アプリケーションの個々のパフォーマンス要件を達成します。

マルチプレックスでは、各接続は単一の論理サーバ・コンテキストの下で動作します。クエリをマルチプレックス・サーバに発行すると、接続の論理サーバの構成に応じて、その実行は 1 つまたは複数のマルチプレックス・サーバに分散されます。論理サーバに割り当てられているリソースを動的に調整し、処理を行うア

プリケーションの変化するニーズを満たすために、論理サーバに対してマルチプレックス・サーバの追加または削除を行います。

クエリ・サーバ間のロード・バランス

IQ ネットワーク・クライアントを使用して、マルチプレックス・クエリ・サーバ間でクエリ負荷を分散するには、プール内のマシンにクライアント接続をディスパッチする中間システムが必要となります。

この方法を使用するには、クライアント・システムで、中間ロード・バランス・システムの IP アドレスとポート番号および汎用サーバ名を指定し、**VerifyServerName** 接続パラメータを **NO** に設定した特別な ODBC DSN を作成します。 クライアントがこの DSN を使って接続すると、ロード・バランサは負荷が最も少ないと判断したマシンに対して接続を確立します。

クエリ・サーバのロード・バランスで使用する ODBC DSN を定義する方法の詳細については、『システム管理ガイド：第 1 巻』の「接続パラメータと通信パラメータ」>「VerifyServerName 通信パラメータ (Verify)」を参照してください。

注意： サードパーティ製ソフトウェアが必要です。**VerifyServerName** にはこの動作を許可する機能しかありません。

参照：

- マルチプレックス・データベースのディスク領域の管理 (31 ページ)
- データベース・アクセスの制限使用頻度の少ない時間に更新をスケジュールして、クエリのパフォーマンスを向上させます。
- ディスクのキャッシュ大量のメモリを有効に使用して、実メモリに対する需要とディスクからのデータ読み取りのニーズのバランスを保ちます。

データベース・サイズと構造の管理

データベースのサイズは、作成するインデックスと格納するデータ量に大きく依存します。インデックスを作成することによって、クエリ処理を高速化できます。不要なオブジェクトを削除することによって、ディスク領域を解放したり、ロード時間を短縮したりできます。

インデックスの断片化

- 内部インデックスの断片化は、インデックス・ページが最大ボリュームまで使用されていないときに発生します。
- ローの断片化は、ローが削除されると発生します。ページのロー全体を削除した場合、そのページは解放されますが、ページの一部のローが未使用の場合は、未使用領域がディスクに残ります。

- テーブルに対する DML 操作 (INSERT、UPDATE、DELETE) により、インデックスの断片化が発生することがあります。

断片化の問題についての情報を取得するには、次のストアド・プロシージャを実行します。

- **sp_iqrowdensity** は FP インデックス・レベルでのローの断片化を報告します。詳細については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「sp_iqrowdensity プロシージャ」を参照してください。
- **sp_iqindexfragmentation** は、補助インデックス内の内部断片化を報告します。『リファレンス： ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「sp_iqindexfragmentation プロシージャ」を参照してください。

出力を調べて、インデックスの再作成、再編成、再構築などの対応策をとるかどうかを判断する必要があります。 FP インデックスを補助するために別のインデックスを作成できます。

カタログ・ファイル増大の最小化
カタログ・ファイルのサイズが増加するのは正常なことで、その割合はアプリケーションとカタログの内容によって異なります。 .db ファイルのサイズがパフォーマンスに影響を与えることはなく、.db ファイル内の空きページは必要に応じて再利用されます。

カタログ・ファイルの増大を最小限に抑えるには、次の方法を使用します。

- CREATE TABLE 文で IN SYSTEM を使用しない。
- システム・ストアド・プロシージャを実行した後で COMMIT 文を発行する。
- 長時間実行されるトランザクションの場合は、途中で COMMIT 文を発行する。

ネットワーク・パフォーマンス

環境を少し変更するだけで、ネットワーク・パフォーマンスの問題を解決できることがあります。

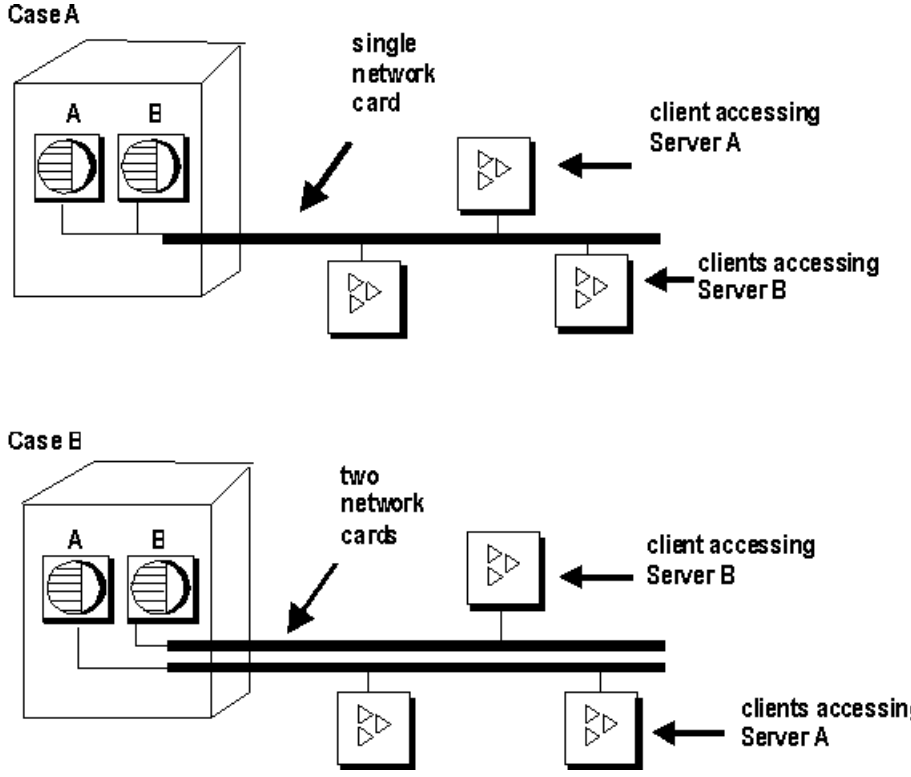
ネットワーク・スループットを改善するには、複数のネットワーク・アダプタを用意します。サービス・レベル・アグリーメントに基づき、ユーザをクラス別で異なるネットワークに割り当てることができます。

図 12-4 のケース A では、2 つの異なるデータベース・サーバにアクセスするクライアントが 1 枚のネットワーク・カードを使用しています。このため、サーバ A とサーバ B にアクセスするクライアントは、ネットワーク上とネットワーク・カードで競合します。ケース B では、サーバ A にアクセスするクライアントと

サーバ B にアクセスするクライアントが別々のネットワーク・カードを使用しています。

異なるマシンをデータベース・サーバにすると、さらにパフォーマンスが向上します。異なるデータベースのヘビー・ユーザを異なるマシンに分けることもできます。

図 1 : ヘビー・ネットワーク・ユーザの分離



少量のデータを小さなパケットに入れる
ネットワーク上で少量のデータを送信する場合は、デフォルトのネットワーク・パケット・サイズを小さいまま使用します (デフォルトは 512 バイトです)。**-p** サーバ起動オプションは、最大パケット・サイズを指定するために使用します。クライアント・アプリケーションを使用してパケット・サイズを設定できます。

大量のデータを大きなパケットに入れる。
大量のデータを送受信するアプリケーションが多い場合は、デフォルトのネットワーク・パケット・サイズを大きくします。転送の数は少なくなりますが、データ転送量は多くなります。

サーバ・レベルのプロセス
サーバ・レベルで、できるかぎり多くのデータをフィルタします。

参照：
- パフォーマンスに関する考慮事項 (3 ページ)
- メモリ使用の最適化 (5 ページ)
- プロセス・スレッド・モデル (17 ページ)
- I/O の分散 (18 ページ)
- リソース使用を調整するオプション (24 ページ)
- リソースを効率的に利用するための他の方法 (31 ページ)
- インデックスのヒント (43 ページ)
- データベース・サイズと構造の管理 (32 ページ)
- ロードを高速化するための UNION ALL ビュー (59 ページ)

パフォーマンスのモニタリングとチューニング

システムが使用可能なリソースを最大限に利用しているかどうかを確認するために使用するツールについて説明します。

ストアド・プロシージャでの情報の取得

データベース情報を表示するストアド・プロシージャがいくつか用意されています。

表 3 : 統計情報を示すストアド・プロシージャ

名前	説明
sp_iqconnection	接続およびバージョンについての情報を表示します。この情報には、テンポラリ DB 領域を使用しているユーザ、バージョンを有効にしているユーザ、各接続が Sybase IQ 内で行っている作業、接続ステータス、データベース・バージョン・ステータスなどが含まれます。 『リファレンス： ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqconnection プロシージャ」を参照してください。
sp_iqcontext	接続ごとに、現在実行されている文に関する情報を追跡して表示します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcontext プロシージャ」を参照してください。
sp_iqcheckdb	現在のデータベースの妥当性を確認します。オプションで、DB 領域またはデータベースの割り付けの問題を解決します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqcheckdb プロシージャ」を参照してください。
sp_iqdbstatistics	最後に実行された **sp_iqcheckdb** の結果をレポートします。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqdbstatistics プロシージャ」を参照してください。

名前	説明
sp_iqdbsize	現在のデータベースのサイズを表示します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqdbsize プロシージャ」を参照してください。
sp_iqspaceinfo	データベース内の各オブジェクトによる領域の使用状況を表示します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqspaceinfo プロシージャ」を参照してください。
sp_iqstatus	データベースのその他のステータス情報を表示します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqstatus プロシージャ」を参照してください。
sp_iqtablesize	現在のデータベース内の各オブジェクトが使用しているブロック数と、オブジェクトが置かれている DB 領域の名前を表示します。 『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「システム・ストアド・プロシージャ」>「sp_iqtablesize プロシージャ」を参照してください。

Sybase IQ の全ストアド・プロシージャの構文の詳細と例については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』を参照してください。

データベース・プロシージャのプロファイリング

プロシージャ・プロファイリングは、プロシージャ、他のシステム・イベントの実行時間を追跡します。Sybase Central のプロファイリングを使用して、データベースまたはデータベース・オブジェクトのパフォーマンスの問題を特定します。

プロシージャ・プロファイリング統計の表示

Sybase Central でデータベースのプロファイリング・オプションを設定し、ストアド・プロシージャ、関数、イベント、トリガの実行時間をモニタします。

データベースのプロファイル・プロパティ

表 4 : データベースのプロファイル・プロパティ

プロパティ名	説明
Name	オブジェクトの名前をリストします。
Owner	オブジェクトの所有者をリストします。
Table	トリガが属するテーブルをリストします (このカラムはデータベースの [プロファイル] タブにのみ表示されます)。
Event	システム・トリガのトリガのタイプを表示します。Update、Delete のいずれかです。
Type	オブジェクトのタイプ (たとえばプロシージャ) をリストします。
# Exes.	各オブジェクトが呼び出された回数をリストします。
#msecs.	各オブジェクトの合計実行時間をリストします。

データベース・オブジェクトのプロファイル

データベース・オブジェクトには、ストアド・プロシージャ、関数、イベント、トリガが含まれます。データベース・オブジェクトのプロファイル・プロパティは、1 行ごとに、実行時間が要約されて表示されます。

表 5 : オブジェクトのプロファイル・プロパティ

プロパティ名	説明
Calls	オブジェクトが呼び出された回数をリストします。
Milliseconds	各オブジェクトの合計実行時間をリストします。
Line	プロシージャの各行に行番号を付加します。
Source	SQL プロシージャを 1 行ずつ表示します。

Sybase Central でデータベースのプロファイリング・プロパティを設定する

Sybase Central でデータベースのプロファイリング・プロパティを設定するには、DBA 権限のあるユーザとしてデータベースに接続する必要があります。また、サーバが実行している必要があります。

1. Sybase Central で、データベースを右クリックし、[プロパティ] を選択します。
2. [プロファイリング設定] タブをクリックします。
3. 他のプロファイリング・オプションについては、オンライン・ヘルプを参照してください。

データベース・オブジェクトのクラスに関するプロファイリング情報の表示

Sybase Central でデータベース・オブジェクトのクラスに関するプロファイリング情報を表示するには、親フォルダをクリックし、オブジェクトのプロファイルを確認します。

1. オブジェクト・フォルダを開きます。
   - プロシージャと関数
   - イベント
   - トリガ
   - システム・トリガ
2. 右ウィンドウ枠で [プロファイル] タブをクリックします。

   オブジェクトのプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

特定のデータベース・オブジェクトについてのプロファイリング情報の表示

Sybase Central で特定のデータベース・オブジェクトについてのプロファイリング情報を表示するには、オブジェクトを選択し、オブジェクトのプロファイルを確認します。

1. オブジェクト・フォルダを開きます。
   - プロシージャと関数
   - イベント
   - トリガ
   - システム・トリガ
2. 親フォルダでオブジェクトをクリックします。
3. 右ウインドウ枠で [プロファイル] タブをクリックします。

   オブジェクトのプロファイリング情報が右ウィンドウ枠の [プロファイル] タブに表示されます。

プロシージャ・プロファイリング統計

データベースのプロファイリング・オプションを設定し、プロファイリング・オプションを使用して、ストアド・プロシージャ、関数、イベント、トリガのパフォーマンス統計を返します。

sa_procedure_profile_summary

sa_procedure_profile_summary はシステム・プロシージャで、データベース内で実行したすべてのプロシージャ、関数、イベント、またはトリガの実行時間についての要約情報を報告します。このプロシージャは、これらのオブジェクトに関して、Sybase Central の [プロファイル] タブと同じ情報を提供します。

表 6 : sa_procedure_profile_summary 統計

カラム名	説明
object_type	オブジェクト・タイプを識別します。 • P (ストアド・プロシージャ) • F (関数) • T (トリガ) • E (イベント) • S (システム・トリガ)
object_name	オブジェクトの名前をリストします。
executions	各オブジェクトが呼び出された回数をリストします。
owner_name	オブジェクトの所有者をリストします。
table_name	トリガのプロファイル情報を取得するテーブルを指定します。
executions	オブジェクトが呼び出された回数をリストします。
Milliseconds	行の実行時間をミリ秒単位で示します。
foreign_owner	システム・トリガのための外部テーブルを所有するデータベース・ユーザを示します。
foreign_table	システム・トリガのための外部テーブルの名前を示します。

プロシージャ・プロファイル

sa_procedure_profile は、データベース内で実行されたプロシージャ、関数、イベント、またはトリガに含まれる各行の実行時間についての情報を報告します。

表 7 : sa_procedure_profile 統計

カラム名	説明
object_type	オブジェクト・タイプを識別します。 • P (ストアド・プロシージャ) • F (関数) • T (トリガ) • E (イベント) • S (システム・トリガ)
object_name	オブジェクトの名前をリストします。
owner_name	オブジェクトの所有者をリストします。
table_name	トリガに関連付けられているテーブルを識別します (他のオブジェクト・タイプの場合は NULL)。
Line_number	プロシージャ内のライン番号を示します。
executions	オブジェクトが呼び出された回数をリストします。
Milliseconds	オブジェクトの実行時間をリストします。
percentage	特定の行で必要な実行時間が全実行時間に対してる占めるパーセンテージを示します。
foreign_owner	システム・トリガのための外部テーブルを所有するデータベース・ユーザを示します。
foreign_table	システム・トリガのための外部テーブルの名前を示します。

Interactive SQL でのデータベースのプロファイリング・オプションの設定

sa_server_option を使用して、Interactive SQL でデータベースのプロファイリング・オプションを設定します。設定するには、DBA 権限を持つユーザとしてデータベースに接続する必要があります。また、サーバが実行している必要があります。

Interactive SQL で、**sa_server_option** を実行し、procedure_profiling オプションを設定します。

次に例を示します。

```
CALL sa_server_option ( 'procedure_profiling', 'ON')
```

他のオプションについては、『SQL Anywhere サーバ ― SQL リファレンス』の「システム・プロシージャ」>「システム・プロシージャのアルファベット順リスト」の「sa_server_option システム・プロシージャ」を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

Interactive SQL でのプロファイリング情報の生成

sa_procedure_profile と **sa_procedure_profile_summary** は、プロシージャ、関数、イベント、トリガの実行統計を生成します。
Interactive SQL で、**sa_procedure_profile** または **sa_procedure_profile summary** を実行します。次に例を示します。

```
CALL sa_procedure_profile ( 'procedure_profiling', 'ON')
```

他のオプションについては、『SQL Anywhere サーバ ― SQL リファレンス』を参照してください。

注意： このリファレンスは SQL Anywhere マニュアルにリンクされています。

データ・モデルの推奨事項

優れたパフォーマンスのデータベースは、優れたデータベース設計から生まれます。時間をかけて Sybase IQ 独自の設計機能を開発のスキーマに組み入れて、応答時間の短縮とクエリ結果の迅速な取得に役立ててください。

インデックスのヒント

正しいカラム・インデックス・タイプを選択して、クエリをより高速に実行します。

Sybase IQ では、いくつかのインデックスが自動的に設定されます。射影を最適化する 1 つのインデックスがすべてのカラムに対して設定され、`UNIQUE`、`PRIMARY KEYS`、`FOREIGN KEYS` に対して `HG` インデックスが設定されます。 これらのインデックスはいくつかの目的には役立ちますが、特定のクエリをできるだけ迅速に処理するには別のインデックスが必要となります。

インデックス・アドバイザ
インデックス・アドバイザは、1 つまたは複数のカラム上にインデックスを追加で設定することによりクエリが高速に処理される可能性がある場合に、メッセージを生成します。

インデックス・アドバイザをアクティブにするには、`INDEX_ADVISOR` オプションを `ON` に設定します。メッセージはクエリ・プランの一部として出力されます。クエリ・プランが有効になっていない場合は、メッセージ・ログ (`.iqmsg`) に単

独のメッセージとして出力されます。出力は OWNER.TABLE.COLUMN 形式です。詳細については、『リファレンス：文とオプション』 の「データベース・オプション」を参照してください。

LF インデックスまたは HG インデックス
カラムが列挙型 FP 記憶領域を使用していない場合、ジョイン・クエリの WHERE 句で参照されるグループ化カラムの LF インデックスまたは HG インデックスのいずれかの作成を検討する必要があります。Sybase IQ オプティマイザは、最適なクエリ・プランを作成するために、列挙型 FP または HG/LF インデックスからのメタデータを必要とする場合があります **HAVING** 句で非集合カラムが参照される場合、クエリを最適化するには、LF インデックスまたは HG インデックスの使用が有効です。 次に例を示します。

```
SELECT c.name, SUM(l.price * (1 - l.discount))
FROM customer c, orders o, lineitem l
WHERE c.custkey = o.custkey
    AND o.orderkey = l.orderkey
    AND o.orderdate >= "1994-01-01"
    AND o.orderdate < "1995-01-01"
GROUP by c.name
HAVING c.name NOT LIKE "I%"
    AND SUM(l.price * (1 - l.discount)) > 0.50
ORDER BY 2 desc
```

インデックスの追加は、記憶領域要件とロード時間の増大につながるため、 クエリ・パフォーマンスが向上する場合にのみ実行してください。

参照：
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- HG インデックスのロード (47 ページ)
- マルチカラム・インデックス (49 ページ)

インデックスを使用する状況と場所

インデックスは Sybase IQ をチューニングするための基本的な内部メカニズムです。インデックスを使用する状況と場所を把握することにより、クエリの実行速度を向上できます。

次の状況では常にインデックスを使用します。

- ジョイン・カラム (カーディナリティに関係なく HG インデックスを使用)。
- サーチャブル・カラム (カーディナリティに基づき HG または LF インデックスを使用)。
- DATE、TIME、DATETIME/TIMESTAMP カラム (DATE、TIME、DTTM)。

DATE、TIME、または DATETIME/TIMESTAMP カラムには、データ・カーディナリティに応じて LF インデックスまたは HG インデックスも設定してください。
- カラムの使用頻度が高いかどうかわからない場合は、カラムに LF インデックスまたは HG インデックスを配置する。 その後、負荷管理を有効にしてインデックスの使用を監視できます。
- 適切な場合は、PRIMARY KEY、UNIQUE CONSTRAINT、または UNIQUE HG インデックスを使用する。これらのインデックスにより、IQ に対してインデックス・カラム内で一意のデータに関する追加情報が提供されます。
- HNG または CMP インデックスが設定されたカラムには、対応する LF または HG インデックスが必要。
- クライアントに対してのみデータが返される (射影される) カラムにはインデックスは不要。

参照：
- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

簡単なインデックス選択基準

いくつかの簡単な質問に回答することで、カラムに対して適切なインデックスを選択するために役立ちます。

データモデルに対してクエリに関係なく最適なインデックスを決めるには、各カラムについて以下の簡単な質問に回答します。

- カーディナリティは 1500 ～ 2000 を超えているか。
  回答が「はい」の場合は、このカラムに HG インデックスを配置します。「いいえ」の場合は、カラムに LF インデックスを配置します。

- カラムに DATE、TIME、DATETIME、または TIMESTAMP データが含まれているか。
  回答が「はい」の場合は、このカラムに DATE インデックス、TIME インデックス、または DTTM インデックスを配置します。 また、カラムに LF または HG を配置してください。
- カラムは範囲検索または集合に使用されるか。
  回答が「はい」の場合は、カラムに HNG インデックスを配置します。また、カラムに LF または HG を配置してください。集合にカラム以外が含まれる場合、HNG は適切でない可能性があります。 ほとんどの場合、集合処理では LF インデックスまたは HG インデックスが十分以上の機能を果たすため、HNG インデックスは不要です。 これは、DATE、TIME,、または DATETIME の各データ型には該当しません。
- このカラムは単語検索に使用されるか。
  回答が「はい」の場合は、カラムに WD インデックスを配置します。LF インデックスまたは HG インデックスは不要で、使用すると大量の記憶域を使用します。
- このカラムは全文検索に使用されるか。
  回答が「はい」の場合は、カラムに TEXT インデックスを配置します。 LF または HG は不要で、使用すると大量の記憶域を使用します。
- 同じテーブル内の 2 つのカラムが相互に比較されるか (A = B, A < B, A > B, A <= B, A > + B)。
  回答が「はい」の場合は、2 つのカラムに CMP インデックスを配置します。
- このカラム (またはカラムの組) は GROUP BY 文または ORDER BY 文で使用されるか。
  回答が「はい」の場合は、GROUP BY 文または ORDER BY 文で使用されるカラム (またはカラムの組) に HG インデックスを配置します。 また、各カラムに対応する HG インデックスまたは LF インデックスも配置してください。
- このカラムは、マルチカラム・プライマリ・キー、制約、またはインデックスの一部か。
  回答が「はい」の場合は、マルチカラム・インデックス内の各カラムに HG インデックスまたは LF インデックスを配置します。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)

- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

HG インデックスのロード

他の IQ インデックスと比較すると、HG インデックスは、データのロード中および削除中の維持にコストがかかります。 HG インデックスのパフォーマンスを最も左右する要因は、HG インデックス構造内のデータの位置、 つまり操作が分散して行われるか、または集約して行われるか (操作密度の大小) です。

HG 密度大の (集約して行われる) 操作とは、影響を受けるローが特定キーの周囲に緊密に集まっているものです。 密度小の (分散して行われる) 操作とは、影響を受けるローがキーごとに少ししかないものです。 たとえば、一般にデータの日付は、操作がログに記録された日時や、データが変更された日時などの周囲に集まります。つまり、新しいデータは HG インデックス構造の末尾に配置されます。 日付 HG インデックス内のデータを削除する場合、一般的にそのデータは、日付、週、月などのチャンクから派生するもので、HG btree の先頭から削除されるか、または少数のキーの周囲に緊密に集められて削除されます。 IQ が少数のページ上で操作を行い、非常に多数のローに対して影響を及ぼす場合と比較すると、これらの操作は非常に高速です。

Prices、Customer ID、City、Country など、比較的分散しているデータについては、状況が大きく異なります。 たとえば、「価格」データは価格ごとにロードされ、インデックス内にすでに存在するデータ全体にわたって大幅に変動します。 株価を追跡しているカラムがある場合、変更されるデータは、すでにロードされている値の範囲のほぼ全体にわたるため、データを格納する数値フィールドは密に更新されます。 影響を受ける各ローに対して維持する必要のあるインデックス・ページ数が多いため、これらの操作は速度が遅くなります。 最悪の場合、IQ はロードまたは削除対象の EACH ROW に対して、1 ページの読み取り／書き込みを強いられます。 これは最善の方法ではありませんが、Sybase IQ は、HG インデックス・ロードのフェーズ 2 と削除を並列処理して、影響を大幅に軽減できるように設計されています。

以上の点は問題ないのですが、データ・モデルの設計とインデックス作成にはどのような影響があるでしょうか。 一般に、Sybase IQ のチューニングと最適化では、結局インデックスの有無が問題になります。 インデックスがデータとロード

によってどのような影響を受けるかを把握することは、どのインデックスを導入し、どれを導入しないかを決定するときに重要な要素となります。HG インデックスは、他のインデックスより比較的ロードに時間がかかるため、使用および設計の対象となる場合は、注目を集めることがよくあります。HG インデックスがクエリのパフォーマンス向上に役立つことは確実です。しかし、インデックスを追加してもクエリに対する好影響はわずかしかなく、データ損失に対する影響の方が多いこともあります。このような状況では、ロードと削除に時間がかかる理由を把握し、どのような対処が可能であるかを理解することが重要です。

この点においては、現在ロードされているデータに関する新規データの分散または密集の度合いが重要な役割を演じます。Customer ID の比較的ランダムなカラムにインデックスを付けてクエリのパフォーマンスを改善するには、そのカラムにインデックスを付ける必要があります。ただし、テーブルにプライマリ キーが存在し、それが Customer ID の場合、トランザクションの日時は Date フィールドに格納されます。順序を (`customer_id`、`transaction_date`) のままにすると、ほとんどの場合、データは分散してロードされるか、またはテーブルから削除されます。ロードされるデータは、トランザクションの日付ごとにロードされます。しかし、Customer ID カラムはマルチカラム・インデックスの最初のカラムであるため、IQ は HG インデックス構造全体を通じたデータのタッチを強いられます。

順序を (`transaction_date`、`customer_id`) に変更するだけで、この動作は変わります。インデックスは、プライマリ・キーの参照整合性を制御するために引き続き使用されます。カラムの順序は、プライマリ・キーの強制適用には無関係です。そのため、ダウンストリームに悪影響を及ぼすことなく、カラムの順序を変更できます。順序をこのように変更すると、トランザクションの日付ごとにロードされるすべての新規データが、非常に密度の高い状態で HG インデックス構造の末尾に強制的に挿入されます。やがてロードが継続して行われるにつれ、データは常に HG 構造の末尾に追加されます。

マルチカラム・インデックスのカラムの順序を変更するだけで、パフォーマンスに大きな影響を与えることができます。データの幅は順序に関係なく同じであるため、HG インデックスのサイズはそれほど大きく変わりません。データがロードされる速度と、テーブルから削除される速度が変わります。

参照：

- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)

- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

マルチカラム・インデックス

現在、インデックスの作成で複数のカラムをサポートしているのは、HG、UNIQUE HG、UNIQUE CONSTRAINT、PRIMARY KEY の各インデックスのみです。ただし、マルチカラム・インデックスは **GROUP BY** 文と **ORDER BY** 文でも役立ちます。

統計についての観点では、マルチカラム・インデックスは、オプティマイザにジョインに関する正確な統計情報を知らせ、多対多のジョインか 1 対多のジョインかをオプティマイザが判断するための十分な情報を、マルチカラム・テーブル・ジョインとして提供します。 オプティマイザは、最適化のためにも統計情報を利用しますが、実際の作業では個別の HG/LF インデックスを使用します。 オプティマイザは、ジョインと並べ替えのすべてのシナリオについてコストを見積もり、その操作にどのインデックスが最適かを決定します。 統計情報はその決定を行うために役立ちます。

HG インデックスについては、次の項目に留意してください。

- HG の挿入は Sybase IQ で最もコストがかかる
- 挿入は確実にインデックスの末尾で行われるようにする
  トランザクションの日付やバッチ番号 (連続データ) など、一般的なインクリメント・データはインデックス・リストの先頭に配置する。 連続キーを確保する

前項の「HG インデックスのロード」を参照してください。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)

- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

ジョイン・カラム

ジョインでは、データ型を可能なかぎり狭い範囲に保つことで、ディスク I/O と必要なメモリ量を少なくできます。

整数値の方が文字列より高速に比較できるため、ジョインでは整数データ型 (可能であれば符号なし) を使用します。 データ型を可能なかぎり狭い範囲に保つことで、ディスク I/O と必要なメモリが少なくなり、ジョインのパフォーマンスが向上します。ジョインの観点では、HG インデックスの機能の方がやや優れているため、ジョイン・カラムに対してはカーディナリティが適切なインデックス (LF または HG) ではなく、HG インデックスを使用します。 この判断では、HG インデックスのロードにかかる時間は LF インデックスに比べて長くなる可能性があることを慎重に考慮する必要があります。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

プライマリ・キー

マルチカラム・プライマリ・キーには、プライマリ・キーで指定された各カラムに対して、追加の LF または HG インデックスを配置する必要があります。 IQ は複合カラムに 1 つの HG インデックスを作成するだけであるため、この操作は手動で行う必要があります。

UNIQUE constraint、UNIQUE HG、プライマリ・キーは同じ構造を共有しています。 その構造では、G-Array のない HG インデックスを使用してロー ID を格納します。 可能な場合は、テーブルでプライマリ・キーを使用します。 これにより、インデックスが使用されない場合でも、オプティマイザがクエリ・パスを決定する上で有益な情報が提供されます。 このインデックス構造は、オプティマイザがより良い選択を行うために有益な詳しい統計情報を提供するとともに、データを詳細に検討するための構造を提供します。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

外部キー

プライマリ・キーと同様に、外部キーを使用してクエリのジョイン・パフォーマンスを向上できます。 この使用により、テーブルがジョインされる方法についての情報と、そのジョインの背後にある統計情報が IQ に提供されます。 IQ は外部キー・カラムに自動で HG インデックスを作成するため、HG インデックスまたは LF インデックスを追加する必要はありません。外部キーを使用するには、参照先のテーブルにプライマリ・キーが存在する必要があります。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

データ型の適切なサイズ設定

すべてのデータ型、特に文字ベースのデータ型に対して、できるだけ正確なサイズを設定します。

カラムでどのデータ型を使用するかを決定するには、次の要素について考慮します。

- Sybase IQ には数多くのデータ型が含まれており、アプリケーションに対して適切なデータ型を使用することにより、最適なパフォーマンスを実現できる。
- HOUR、MINUTE、SECOND に関する情報が不要な場合は、DATETIME ではなく DATE を使用する。
- データが TINYINT または SMALLINT データ型に収まる場合は、INTEGER または BIGINT ではなくこれらのデータ型を使用する。
- NUMERIC( ) または DECIMAL( ) を定義する場合、その程度の精度では不要なデータにコストがかかる場合があるため、記憶域を過剰に割り付けない。
- CHAR( ) と VARCHAR() のデータ型は、デフォルトのフラット FP インデックス内の幅が固定されている。 ただし、VARCHAR() の各ローには使用されるバイト数を表す 1 バイトが追加される点が唯一異なる。

Sybase IQ には、BINARY( )、CHAR( )、VARCHAR( )、VARBINARY( ) の各データ型でよく見られる大量の繰り返しパターンを圧縮するための新しい圧縮アルゴリズムが含まれています。

参照：

- HG インデックスのロード (47 ページ)

- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

IQ UNIQUE と MINIMIZE_STORAGE

`IQ UNIQUE` と `MINIMIZE_STORAGE` を使用すると、ディスク領域を節約してパフォーマンスを向上できます。

可能なかぎり `IQ UNIQUE` と `MINIMIZE_STORAGE` を使用することで、デフォルト `FP` インデックス・タイプで使用する記憶領域を最小限に抑えることができます。デフォルトでは、これらのオプションを使用するときに最適化されたデータ圧縮が有効でないため、領域の消費量が増加する原因となる場合があります。 これらのオプションのいずれかをデータ・モデルに採用することにより、データをできるだけ圧縮して、記憶域とパフォーマンスの最大化を達成するために役立ちます。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)

- テンポラリ・テーブル (57 ページ)

NULL 値

カラムを NULL または NOT NULL として定義することにより、オプティマイザの機能をより効率化するために役立ちます。

NULL または NOT NULL を指定することで、データの特性に関する情報が追加され、オプティマイザがジョインと条件検索を行うときに、より根拠のある推測を実行できます。NULL データに指定しても、他のデータベースのようにはデータベース・ページの領域を節約できませんが、NULL データは、IQ の圧縮アルゴリズムと最適化されたインデックスによって、ディスクへの格納時に圧縮されます。

- NULL または NOT NULL を常に指定する
- テーブルが作成されるときの Open Client 接続と ODBC 接続のデフォルト動作は異なる
- オプティマイザに対して、ジョインと引数の検索に使用するデータの特性に関する追加情報を提供する

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

符号なしのデータ型

場合によっては、符号なしのデータ型を使用することで符号を比較する必要がなくなり、より高速なクエリを作成できます。

すべてのデータが常に 0 以上であり、データの符号が問題にならない場合に、符号なしのデータ型を使用します。符号が格納されないため、カラムの比較で符号

を比較する必要がなくなります。これにより、特にキー・カラムについてパフォーマンスが向上し、データのジョインと検索の操作で手順が 1 つ少なくなります。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

LONG VARCHAR と LONG VARBINARY

VARCHAR( ) と VARBINARY( ) を使用して、ラージ・オブジェクトの格納メカニズムを使用せずにカラムの記憶領域のサイズを増大します。

一般に、開発者と DBA は VARBINARY( ) データと VARCHAR( ) データが 255 バイトに制限されていると考えています。 IQ でサポートされる VARCHAR( ) データと VARBINARY( ) データのサイズは最大 32K です (それぞれ LONG VARCHAR と LONG VARBINARY とも呼ばれます)。 そのため、BLOB/CLOB データ型や IMAGE/TEXT データ型の専門性の高いラージ・オブジェクト格納メカニズムを使用しなくても、さらに大きなテキスト・データまたはバイナリ・データを格納できます。

- 適量のテキスト・データまたはバイナリ・データを格納するために使用できる
- 最大サイズは 32K (VARBINARY( ) では 16 進 ASCII データ 64K)
- WORD インデックスと TEXT インデックスのみが、255 バイトより大きいサイズの VARCHAR( ) データのインデックスとして使用可能
- 記憶域は 256 バイトのチャンクとして割り付けられる
- 257 バイトの文字列には 512 バイトの記憶域が必要
- 511 バイトの文字列にも 512 バイトの記憶域が必要

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

ラージ・オブジェクトの格納

32K を超える記憶域が必要なデータには、ラージ・オブジェクト・データ型を使用します。

- ラージ・オブジェクト・データ型には ASCII (TEXT/CLOB) データとバイナリ (IMAGE/BLOB) データが格納される。 データの各 BLOB/CLOB セルは、1 つ以上のページに格納される
  - 前提のページ・サイズは 128K
  - データが 129K の場合、情報の格納に 2 ページ必要
  - データが 1K の場合、データの格納に 1 ページ必要
  - どちらの場合も、ページはブロック・サイズの倍数に圧縮される
- バイナリまたはテキスト・ベースのオブジェクトを格納するために使用できる
- LONG BINARY データ型の最大サイズは 6K から無制限に拡大される
- 設定可能な唯一のインデックスは TEXT インデックスである
- TEXT インデックスとその検索機能を使用して完全に検索できる
- オブジェクトのサイズを返す特別な関数がある (byte_length64)
- オブジェクトの全体ではなく一部を返す特別な関数がある (byte_substr64)
- **BFILE()** 関数を使用して、バイナリ・オブジェクト・セルの内容を個別ファイルに抽出できる

参照：

- HG インデックスのロード (47 ページ)

- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)
- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)
- テンポラリ・テーブル (57 ページ)

テンポラリ・テーブル

トランザクションがコミットされてもデータを維持するには、グローバル・テンポラリ・テーブルを作成するときに `ON COMMIT PRESERVE ROWS` オプションを使用するか、ローカル・テンポラリ・テーブルを宣言します。

テンポラリ・テーブルには次の 3 種類があります。

- # (ハッシュ) テーブル
  ```
  CREATE TABLE temp table( col1 int )
  ```
- ローカル・テンポラリ・テーブル
  ```
  DECLARE LOCAL TEMPORARY TABLE temp table ( col1 int )
  ```
  ローカル・テンポラリ・テーブルの動作は # テーブルと同様です。
- グローバル・テンポラリ・テーブル
  ```
  CREATE GLOBAL TEMPORARY TABLE table temp table ( col1 int )
  ```
  グローバル・テンポラリ・テーブルの構造は、接続の変更や再起動を行っても変わりません。

標準ハッシュ (#) テーブルには `ON COMMIT PRESERVE ROWS` オプションは不要です。これは、ハッシュ・テーブル内のデータはトランザクションがコミットされても常に維持されるからです。

参照：

- HG インデックスのロード (47 ページ)
- データ型の適切なサイズ設定 (52 ページ)
- IQ UNIQUE と MINIMIZE_STORAGE (53 ページ)
- NULL 値 (54 ページ)

- 符号なしのデータ型 (54 ページ)
- LONG VARCHAR と LONG VARBINARY (55 ページ)
- ラージ・オブジェクトの格納 (56 ページ)
- インデックスを使用する状況と場所 (44 ページ)
- 簡単なインデックス選択基準 (45 ページ)
- マルチカラム・インデックス (49 ページ)
- ジョイン・カラム (50 ページ)
- プライマリ・キー (51 ページ)
- 外部キー (51 ページ)

パフォーマンス向上のための非正規化

データベースを非正規化することによりパフォーマンスが向上することがありますが、非正規化には、リスクと短所もあります。

リスク
非正規化を正しく行うには、アプリケーションに関する十分な知識が必要となるため、パフォーマンスに問題がある場合にのみ非正規化を実行してください。データを最新の状態に保つためにどれだけの作業が必要かを考慮する必要があります。

これは、大量のデータの要約が頻繁に必要とされる意志決定支援アプリケーションと個別にデータ変更を行うトランザクション処理要求との違いを示す良い例です。非正規化を行う場合、特定の処理の効率を向上させるために、他の処理の効率が低下することがあります。

非正規化を行うと、データの整合性に問題が発生する可能性があるため、アプリケーションの設計時に慎重に文書化し、注意する必要があります。

非正規化の決定
環境内のアプリケーションのデータ・アクセス要件とその実際のパフォーマンス特性を分析します。次の項目について検討します。

- 重要なクエリと予想される応答時間
- 使用するテーブルまたはカラム。1 アクセスあたりのロー数
- 通常のソート順
- 同時予測
- アクセス頻度が最も高いテーブルのサイズ
- 要約を計算するプロセスの有無

ロードを高速化するための UNION ALL ビュー

テーブル内のすべてのローに二次的なインデックスを維持するにはコストがかかりすぎる場合、UNION ALL ビューを使用するとロード・パフォーマンスが向上することがあります。

Sybase IQ では、日付などでデータを複数のベース・テーブルに分けることができます。 データは、これらの小さいテーブルにロードします。 そして、UNION ALL ビューを使ってテーブルを 1 つの論理的な統一体に結合し、この統一体に対してクエリを実行します。

これによりロード・パフォーマンスを改善することができますが、一部のクエリのパフォーマンスに悪影響を与える可能性があります。 単一のベース・テーブルに対するクエリと、小さく分割された複数のベース・テーブルにわたる UNION ALL ビューに対するクエリのパフォーマンスは、ビューの定義がすべての制約条件を満たしていれば、ほとんどのタイプのクエリでほぼ同じになります。 ただし、一部のクエリ・タイプ、特に DISTINCT を伴う、または複数のジョイン・カラムに関連するジョインを伴うクエリを UNION ALL ビューに対して実行した場合、その実行速度は単一の大きなベース・テーブルに対して実行した場合に比べると非常に遅くなる可能性があります。 この方法を使用する場合は、アプリケーションのクエリ・パフォーマンスを低下させても、ロード・パフォーマンスを改善する必要があるかどうかを検討するようにしてください。

UNION ALL ビューを使用すると効率よく管理できます。たとえば、データを月ごとに分割している場合は、テーブルを削除し、 UNION ALL ビューの定義を適切に更新することで、月全体のデータを削除できます。日付の範囲述部を追加することなく、年、四半期などに対応する多くのビュー定義を作成できます。

UNION ALL ビューを作成するには、ベース・テーブルを別々の物理テーブルに分割する論理的手段を選択します。 最も一般的な方法は、月ごとに分割することです。 たとえば、第 1 四半期のすべての月を含むビューを作成するには、次のコマンドを入力します。

```
CREATE VIEW
SELECT * JANUARY
UNION ALL
SELECT * FEBRUARY
UNION ALL
SELECT * MARCH
UNION ALL
```

月ごとに、1 つのベース・テーブル (この例では JANUARY、FEBRUARY、または MARCH) にデータをロードできます。次の月は、同じカラムと同じインデックス・タイプで構成された新しいテーブルにデータをロードします。

構文の詳細については、『リファレンス：文とオプション』の「UNION 演算」を参照してください。

注意： UNION ALL ビューに対して INSERT...SELECT を実行することはできません。UNION ALL 演算子は、このリリースでは完全に並列であるわけではありません。 使用すると、クエリの並列処理が制限される場合があります。

UNION ALL ビューを参照するクエリの最適化

UNION ALL ビューを参照するクエリのパフォーマンスを調整するには、JOIN_PREFERENCE オプションを設定します。このオプションは、**UNION ALL** ビュー間のジョインに影響を与えます。

最適化が効果を発揮するには、UNION ALL ビューが分割されたすべての部分にすべてのインデックスが定義されている必要があります。 DISTINCT を指定するクエリでは、UNION ALL ビューを使用すると、ベース・テーブルを使用するよりも実行速度が遅くなる傾向があります。

Sybase IQ には、UNION ALL ビューに関する次のような最適化が用意されています。

- UNION ALL ビューでの分割 GROUP BY
- UNION ALL ビューへのプッシュダウン・ジョイン

UNION を分割されたテーブルとして扱えるのは、以下の制約条件がすべて満たされている場合に限られます。

- 1 つまたは複数の UNION ALL のみが含まれる。
- UNION の各分岐の FROM 句にテーブルが 1 つだけ含まれており、そのテーブルは物理ベース・テーブルである。
- UNION のどの分岐にも、DISTINCT、RANK、集合関数、または GROUP BY 句はない。
- UNION の各分岐に含まれる SELECT 句の中の各項目は、カラムである。
- UNION の最初の分岐における SELECT リスト内のカラムのデータ型シーケンスが、UNION の後続の各分岐におけるシーケンスと同じである。

参照：

- UNION ALL ビューのパフォーマンスの管理 (60 ページ)

UNION ALL ビューのパフォーマンスの管理

構造クエリでは、ソート順が **ASC** の場合、**ORDER BY** の前に **DISTINCT** 演算子を評価します。

UNION より下の DISTINCT を評価する最適化は、DESC 順序に適用されません。そのため、ORDER BY が DESC の場合、UNION ALL ビュー内への DISTINCT 演

算子のプッシュをはじめとした一部の最適化は適用されません。 たとえば、次のクエリはパフォーマンスに影響を与えます。

```
SELECT DISTINCT state FROM testVU ORDER BY state DESC;
```

このパフォーマンス上の問題を回避するには、クエリで ORDER BY の前に DISTINCT 演算子を評価する必要があります。こうすることにより、ソート順が ASC になり、最適化を適用できるようになります。

```
SELECT c.state FROM (SELECT DISTINCT state
  FROM testVUA) c
ORDER BY c.state DESC;
```

参照：

- UNION ALL ビューを参照するクエリの最適化 (60 ページ)

パフォーマンス統計のモニタリング

Sybase Central のパフォーマンス・モニタを使用して、シンプレックス・サーバとマルチプレックス・サーバの統計を表示します。統計は、リアル・タイムな動的チャートで表示されます。

注意： この項では、シンプレックス・サーバのみについて説明します。 マルチプレックス・サーバについては、『Sybase IQ Multiplex の使用』を参照してください。

サーバ・レベルでのパフォーマンスのモニタリング

Sybase Central のパフォーマンス・モニタを使用して、シンプレックス・サーバとマルチプレックス・サーバの統計をモニタします。

1. パフォーマンス・モニタを起動するには、Sybase Central のフォルダ・ビューでサーバ名をクリックします。
2. 右側のウィンドウ枠で、[パフォーマンス・モニタ] タブをクリックします。

詳細情報とオプションについては、Sybase IQ ヘルプの「サーバ」>「パフォーマンスのモニタリング」を参照してください。

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)

- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)

メモリ使用状況統計

メモリ使用状況統計は、サーバのメモリ統計を示します。

表 8 : メモリ使用状況

名前	説明	デフォルトでのモニタリング
割り付けられたメモリ	IQ サーバによって割り付けられているメモリ量 (MB 単位)	可
割り付けられる最大メモリ	IQ サーバによって割り付けられる最大のメモリ容量 (MB 単位)	不可

参照：

- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

キャッシュ統計

キャッシュ統計はキャッシュの使用状況を示します。

表 9 : キャッシュ統計

名前	説明	デフォルトでのモニタリング
カタログ・キャッシュ・ヒット数	1 秒あたりのカタログ・キャッシュ・ヒット数。	不可
テンポラリ・キャッシュ・ヒット数	1 秒あたりのテンポラリ・キャッシュ・ヒット数。	不可
メイン・キャッシュ・ヒット数	1 秒あたりのメイン・キャッシュ・ヒット数。	不可
カタログ・キャッシュ・リード数	1 秒あたりのカタログ・キャッシュ・リード数。	可
テンポラリ・キャッシュ・リード数	1 秒あたりのテンポラリ・キャッシュ・リード数。	不可
メイン・キャッシュ・リード数	1 秒あたりのメイン・キャッシュ・リード数。	不可
現在のカタログ・キャッシュ・サイズ	現在のカタログ・キャッシュ・サイズ (MB 単位)。	不可
現在のテンポラリ・キャッシュ・サイズ	現在のテンポラリ・キャッシュ・サイズ (MB 単位)。	不可
現在のメイン・キャッシュ・サイズ	現在のメイン・キャッシュ・サイズ (MB 単位)。	不可
カタログ・キャッシュ使用率 (パーセンテージ)	パーセントで表現されたカタログ・キャッシュの使用率。	不可
テンポラリ・キャッシュ使用率 (パーセンテージ)	パーセントで表現されたテンポラリ・キャッシュの使用率。	不可
メイン・キャッシュ使用率 (パーセンテージ)	パーセントで表現されたメイン・キャッシュの使用率。	不可
固定されたカタログ・キャッシュ	固定されたカタログ・キャッシュ・ページ数。	不可

名前	説明	デフォルトでのモニタリング
固定されたテンポラリ・キャッシュ	固定されたテンポラリ・キャッシュ・ページ数。	不可
固定されたメイン・キャッシュ	固定されたメイン・キャッシュ・ページ数。	不可
カタログ・キャッシュの固定率 (パーセンテージ)	固定されたカタログ・キャッシュのパーセントで表現された比率。	不可
テンポラリ・キャッシュの固定率 (パーセンテージ)	固定されたテンポラリ・キャッシュのパーセントで表現された比率。	不可
メイン・キャッシュの固定率 (パーセンテージ)	固定されたメイン・キャッシュのパーセントで表現された比率。	不可
ダーティなカタログ・キャッシュ・ページの割合 (パーセンテージ)	パーセントで表現されたダーティなカタログ・キャッシュ・ページの比率。	不可
ダーティなテンポラリ・キャッシュ・ページの割合 (パーセンテージ)	パーセントで表現されたダーティなテンポラリ・キャッシュ・ページの比率。	不可
ダーティなメイン・キャッシュ・ページの割合 (パーセンテージ)	パーセントで表現されたダーティなメイン・キャッシュ・ページの比率。	不可

参照：

- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

CPU 使用率統計

CPU 使用率統計は、使用されている CPU リソースのパーセンテージを示します。

表 10 : CPU 使用率

名前	説明	デフォルトでのモニタリング
CPU 使用率	IQ プロセスの CPU 使用率をパーセンテージで示します。この値には、システムによる使用とユーザによる使用の両方が含まれます。	可
CPU システム使用率	IQ プロセスの CPU システム使用率をパーセンテージで示します。	不可
CPU ユーザ使用率	IQ プロセスの CPU ユーザ使用率をパーセンテージで示します。	不可

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

スレッド統計

スレッド統計は、スレッドの使用状況を示します。

表 11 : スレッド統計

名前	説明	デフォルトでのモニタリング
使用中の IQ スレッド数	IQ サーバによって使用されているスレッド数。	不可

名前	説明	デフォルトでのモニタリング
使用可能な IQ スレッド数	IQ サーバが使用できるスレッドの数。	不可
使用中の SA スレッド数	SQL Anywhere エンジンによって使用されているスレッド数。	不可

参照：
- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

接続統計

接続統計は、接続のアクティビティを示します。

表 12：接続統計

名前	説明	デフォルトでのモニタリング
接続数合計	ユーザ接続および INC 接続を含めた接続総数。	可
ユーザ接続数	ユーザ接続の数。	不可
INC 受信接続数	INC 受信接続の数。	不可
INC 送信接続数	INC 送信接続の数。	不可
1 分あたりのユーザ接続数	1 分あたりのユーザ接続の数。	不可
1 分あたりのユーザ切断数	1 分あたりのユーザ切断の数。	不可

参照：
- メモリ使用状況統計 (62 ページ)

- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

要求統計

要求統計は、クライアント・アプリケーションからの要求に応答するアクティビティの状況を示します。

表 13 : 要求統計

名前	説明	デフォルトでのモニタリング
要求	新しい要求処理または既存の要求の処理の継続のためにサーバに入った 1 秒あたりの回数。	不可
スケジュールされていない要求	現在キュー内で使用可能なサーバ・スレッドを待機している要求の数。	不可
IQ 待機オペレーション	リソース・ガバナを待機している IQ オペレーションの数。	不可
IQ アクティブ・オペレーション	アクティブな IQ オペレーションの数。	不可

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)

- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

トランザクション統計

トランザクション統計は、トランザクションのアクティビティを示します。

表 14 : トランザクション統計

名前	説明	デフォルトでのモニタリング
トランザクション数合計	ユーザ・トランザクションおよび INC トランザクションを含めたアクティブなトランザクションの総数。	不可
ユーザ・トランザクション数	アクティブなユーザ・トランザクションの数。	不可
INC トランザクション数	アクティブな INC トランザクションの数。	不可
アクティブな LOAD TABLE 文	アクティブな LOAD TABLE 文の数。	不可

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

ストア I/O 統計

ストア I/O 統計は、ディスクの読み取りと書き込みのアクティビティを示します。

表 15 : ストア I/O 統計

名前	説明	デフォルトでのモニタリング
カタログ・ストア・ディスク・リード数	カタログ・ストアから読み取られた 1 秒あたりのキロバイト数。	不可
テンポラリ・ストア・ディスク・リード数	テンポラリ・ストアから読み取られた 1 秒あたりのキロバイト数。	不可
メイン・ストア・ディスク・リード数	メイン・ストアから読み取られた 1 秒あたりのキロバイト数。	不可
カタログ・ストア・ディスク・ライト数	カタログ・ストアに書き込まれた 1 秒あたりのキロバイト数。	不可
テンポラリ・ストア・ディスク・ライト数	テンポラリ・ストアに書き込まれた 1 秒あたりのキロバイト数。	不可
メイン・ストア・ディスク・ライト数	メイン・ストアに書き込まれた 1 秒あたりのキロバイト数。	不可

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- DB 領域使用状況統計 (70 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

DB 領域使用状況統計

DB 領域使用状況統計は、DB 領域の可用性を示します。

表 16 : DB 領域使用状況

名前	説明	デフォルトでのモニタリング
使用中の DB 領域ファイル・サイズ	使用中の DB 領域のサイズ。DB 領域ごとにこの統計があります。	不可
使用可能な DB 領域サイズ	各 DB 領域ファイルで使用可能な空き領域をパーセンテージで示します。DB 領域のファイルごとにこの統計があります。	不可

参照：

- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- ネットワーク統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

ネットワーク統計

ネットワーク統計は、ネットワークのアクティビティを示します。

表 17 : ネットワーク統計

名前	説明	デフォルトでのモニタリング
受信バイト数	クライアント／サーバ間の通信で受信された 1 秒あたりのバイト数。	可

名前	説明	デフォルトでのモニタリング
展開した状態で受信されたバイト数	圧縮が無効であると仮定した場合にクライアント／サーバ間の通信で受信されたと想定される 1 秒あたりのバイト数。	不可
送信されたバイト数	クライアント／サーバ間の通信で送信された 1 秒あたりのバイト数。	可
展開した状態で送信されたバイト数	圧縮が無効であると仮定した場合にクライアント／サーバ間の通信で送信されたと想定される 1 秒あたりのバイト数。	不可
未使用の通信バッファ数	使用可能なネットワーク通信バッファ数。	不可
通信バッファ数合計	ネットワーク通信バッファ数合計。	不可

参照：
- メモリ使用状況統計 (62 ページ)
- キャッシュ統計 (63 ページ)
- CPU 使用率統計 (65 ページ)
- スレッド統計 (65 ページ)
- 接続統計 (66 ページ)
- 要求統計 (67 ページ)
- トランザクション統計 (68 ページ)
- ストア I/O 統計 (69 ページ)
- DB 領域使用状況統計 (70 ページ)
- サーバ・レベルでのパフォーマンスのモニタリング (61 ページ)

バッファ・キャッシュのモニタリング

バッファ・キャッシュのパフォーマンスは、全体的なパフォーマンスにとって重要な要因です。バッファ・キャッシュ・モニタは、バッファ・キャッシュ、メモリ、I/O の統計を記録します。

バッファ・キャッシュ・モニタを使用して、メイン・バッファ・キャッシュとテンポラリ・バッファ・キャッシュのメモリ割り付けを細かく調整します。あるキャッシュが他のキャッシュよりもかなり多くの I/O を実行している場合、キャッシュ割り付けの 10% ほどの少量のメモリをそのキャッシュに再割り当てし

ます (増やします)。再割り付けが終了したら、負荷を再実行し、パフォーマンス上の変化をモニタリングします。問題が継続する場合は、メモリの再割り付けを繰り返します。

参照：

- Sybase IQ 環境の表示 Sybase IQ のパフォーマンスをチューニングする最初の手順は、環境を調べることです。
- パフォーマンス統計のモニタリング (61 ページ)
- バッファ・キャッシュの構造 (86 ページ)
- バッファ・マネージャのスラッシングの回避 (87 ページ)
- CPU 使用率をモニタリングするシステム・ユーティリティ (95 ページ)
- バッファ・キャッシュ・モニタリング・チェックリスト (91 ページ)

バッファ・キャッシュ・モニタの起動

バッファ・キャッシュ・モニタを Interactive SQL から起動します。モニタを起動するたびに、各モニタは Sybase IQ 内で別々のカーネル・スレッドとして実行されます。

次の構文を使って、モニタを起動します。

```
          IQ UTILITIES { MAIN | PRIVATE }
INTO
          dummy_table_name
           START MONITOR 'monitor_options [ … ]'
```

MAIN を指定すると、接続先データベースの IQ ストア内のすべてのテーブルについて、メイン・バッファ・キャッシュのモニタリングが開始されます。

PRIVATE を指定すると、接続先データベースのテンポラリ・ストア内のすべてのテーブルについて、テンポラリ・バッファ・キャッシュのモニタリングが開始されます。

バッファ・キャッシュごとにコマンドを別々に発行する必要があります。モニタが結果を収集している間は、これらの各セッションを開いておく必要があります。接続を閉じると、モニタは実行を停止します。接続は最大で 2 つのモニタの実行まで開くことができます。1 つはメイン・バッファ・キャッシュ用で、もう 1 つはテンポラリ・バッファ・キャッシュ用です。

dummy_table_name には、任意の Sybase IQ ベース・テーブルまたはテンポラリ・テーブルを指定します。 他の **IQ UTILITIES** コマンドと構文上の互換性を持たせるために、テーブル名を指定する必要があります。最も望ましいのは、モニタリング専用のテーブルを作成することです。

モニタリング出力ファイルのディレクトリ位置を制御するには、MONITOR_OUTPUT_DIRECTORY オプションを設定します。このオプションを設定しない場合は、データベースと同じディレクトリに結果が出力されます。モニタを実行している間、すべてのモニタリング出力ファイルが使用されます。モニタの実行が停止した後も、ファイルはそのまま残ります。

マルチプレックス・クエリ・サーバを作成する前に、モニタリングで使用するテンポラリ・テーブルを宣言するか、新しいデータベースの作成時に永続的なダミー・テーブルを作成してください。これによって DDL の変更を回避し、実際の運用稼働時にデータがクエリ・サーバにとどまるようにします。

注意： モニタを簡単に使用するには、ストアド・プロシージャを作成してダミー・テーブルを宣言し、出力ロケーションを指定して、モニタを起動します。

注意： 表示する間隔は、ページ単位ではなく、出力行単位です。ただし、次の 2 つの場合は例外です。**-cache_by_type** と **-debug** では、表示ごとに新しいページが開始されます。

参照：

- 出力オプション (73 ページ)
- モニタ実行中の結果の確認 (84 ページ)
- バッファ・キャッシュ・モニタの停止 (85 ページ)
- モニタリング結果の検査と保存 (85 ページ)

出力オプション

バッファ・キャッシュ・モニタの出力は、*monitor_options* 引数で指定するスイッチにより異なります。

-summary

メインとテンポラリの両方のバッファ・キャッシュの要約情報を表示します。モニタ・オプションを何も指定しないと、サマリ・レポートが表示されます。

使用法

```
monitor_options -summary
```

出力

表 18 : -summary の出力フィールド

出力フィールド	説明
Users	バッファ・キャッシュに接続しているユーザ数。

出力フィールド	説明
IO	バッファ・キャッシュによる物理読み込みと物理書き込みの合計。

参照：

- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-cache

メイン・バッファ・キャッシュまたはテンポラリ・バッファ・キャッシュのアクティビティの詳細を表示します。重要なフィールドは、*Finds*, *HR%*、*BWaits* です。

使用法

```
monitor_options -cache
```

出力

表 19 : -cache の出力フィールド

出力フィールド	説明
Finds	バッファ・キャッシュへの検索要求。Finds の値がゼロに急降下してゼロのままなら、サーバにデッドロックが発生しています。サーバでアクティビティがあれば、Finds はゼロ以外の値を示すはずです。
Creats	データベース内の 1 ページの作成要求。
Dests	データベース内の 1 ページの破棄要求。
Dirty	バッファがダーティ (変更) された回数。

出力フィールド	説明
HR%	ヒット率。I/O 要求なしで、バッファ・キャッシュによって応じることのできたパーセンテージ。ヒット率が高いほど効率が良くなります。キャッシュ・サイズを十分な大きさに設定した場合、通常は 90% ～ 100% になります。大きいクエリの場合、最初はヒット率が下がることがありますが、プリフェッチが機能し始めると上昇します。
BWaits	ビジー・ページ (ページ・フレーム競合) のために待機させられた検索要求。通常は小さい数値ですが、特別な場合には大きくなることがあります。たとえば、まったく同じクエリが同時に開始された場合は、両方のクエリが同じページを要求するため、最初の要求がディスクからページを取得するまで 2 番目の要求は待機します。
ReReads	同一トランザクション内でストアの同じ部分がキャッシュ内に再読み込みされた概算の回数。 この数値は常に小さくあるべきですが、Sybase IQ 12.4.2 以降では数値が大きくても重要ではありません。
FMiss	不正な失敗。バッファ・キャッシュがメモリ内のページの検索に複数回のルックアップを必要とした回数。この数値は 0 か、ごく小さな値になるようにしてください。この値が大きい場合は、ロールバックが発生し、特定の操作が繰り返し要求されていると考えられます。
Cloned	Sybase IQ が同時読み込み用に既存のバッファを保持しながら、書き込み用に新しいバッファを作成する必要があった回数。 ページのクローンが作成されるのは、他のユーザがそのページを参照している場合だけです。
Reads/ Writes	バッファ・キャッシュによって実行された物理読み込みと物理書き込み。
PF/ PFRead	プリフェッチ要求、およびプリフェッチ用に実行した読み込み。
GDirty	LRU バッファがダーティな状態で取り込まれたため、Sybase IQ がそのバッファを使用する前に書き出した回数。 この数値が 0 より大きい状態が長時間続かないようにしてください。 0 より大きい状態が続く場合は、スイーパ・スレッドの数を増やすか、ウォッシュ・マーカを移動する必要があります。
Pin%	バッファ・キャッシュ内で使用中でありロックされているページ数のパーセンテージ。
Dirty%	変更されたバッファ・ブロックのパーセンテージ。この数値が 85 ～ 90% を超えないようにします。それ以上になると、[*GDirty*] が 0 より大きくなります。

参照：

- -summary (73 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-cache_by_type

-cache の結果を IQ ページ・タイプごとに集計します([Bwaits] カラムは例外で、合計だけを表示します)。この形式は、Sybase 製品の保守契約を結んでいるサポート・センタに情報を送る場合にたいへん有効です。

使用法

```
monitor_options -cache_by_type
```

参照：

- -summary (73 ページ)
- -cache (74 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-file_suffix
`<dbname>.<connid>-<main_or_temp>-<suffix>` という名前のモニタリング出力ファイルを作成します。オプションのファイル拡張子を指定しないと、デフォルトのファイル拡張子 `iqmon` が使用されます。

使用法
```
monitor_options -file_suffix {extension}
```

参照：
- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-io
指定した期間のメインまたはテンポラリ (プライベート) のバッファ・キャッシュの I/O 率と圧縮比を表示します。これらのカウンタは、サーバのすべてのアクティビティを表します。情報はデバイス別に集計されません。

使用法
```
monitor_options -io
```

出力

表 20 : -io の出力フィールド

出力フィールド	説明
Reads	バッファ・キャッシュによって実行された物理読み込み。
Lrd(KB)	読み込まれた論理キロバイト数 (ページ・サイズに要求数を乗算した数値)。

出力フィールド	説明
Prd(KB)	読み込まれた物理キロバイト数。
Rratio	読み込まれた物理データに対する論理データの圧縮比。読み込み時のディスクに対する圧縮効率の度合い。
Writes	バッファ・キャッシュによって実行された物理書き込み。
Lwrt(KB)	書き込まれた論理キロバイト数。
Pwrt(KB)	書き込まれた物理キロバイト数。
Wratio	書き込まれた物理データに対する論理データの圧縮比。

参照：

- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-bufalloc

ソート、ハッシュ、ビットマップなどのオブジェクト用にバッファ・キャッシュ内の領域を予約する、メイン・バッファ・アロケータまたはテンポラリ・バッファ・アロケータの情報を表示します。

使用法

```
monitor_options -bufalloc
```

出力

表 21 : -bufalloc の出力フィールド

出力フィールド	説明
OU	User_Resource_Reservation オプション設定 (以前は Optimize_For_This_Many_Users)。
AU	現在アクティブなユーザ数。
MaxBuf	バッファ・アロケータで制御されているバッファ数。
Avail	現在ピン・クォータ割り付けに使用可能なバッファ数。
AvPF	現在プリフェッチ・クォータ割り付けに使用可能なバッファ数。
Slots	バッファ・キャッシュ・クォータを使用中の、現在登録されているオブジェクト数。
PinUser	ピン・クォータを使用中のオブジェクト数 (ハッシュ、ソート、B ツリー・オブジェクトなど)。
PFUsr	プリフェッチ・クォータを使用中のオブジェクト数。
Posted	あらかじめプランされたクォータ・ユーザであるオブジェクト数。
UnPost	特定のクォータ・ユーザであるオブジェクト数。
Locks	バッファ・アロケータで処理されたミューテックス・ロック数。
Waits	ロックのためにスレッドが待機する必要があった回数。

参照：

- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-contention
多くの重要なバッファ・キャッシュとメモリ・マネージャ・ロックを表示します。これらのロック・カウンタとミューテックス・カウンタは、バッファ・キャッシュおよびヒープ・メモリ内のアクティビティと、これらのロックがどれだけ迅速に解消されたかを示します。タイムアウト数が 20% を超えている場合は、問題の発生を示しています。

使用法

```
monitor_options -contention
```

出力

表 22 : -contention の出力フィールド

出力フィールド	説明
AU	現在アクティブなユーザ数。
LRULks	LRU がロックされた (テンポラリ・キャッシュ用に繰り返された) 回数。
woTO	タイムアウトなしにロックが付与された (テンポラリ・キャッシュ用に繰り返された) 回数。
Loops	ロックが付与される前に Sybase IQ がリトライした (テンポラリ・キャッシュ用に繰り返された) 回数。
TOs	Sybase IQ がタイムアウトして、ロックのために待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
BWaits	キャッシュ内のバッファに対する Busy Waits の (テンポラリ・キャッシュ用に繰り返された) 回数。
IOLock	Sybase IQ が圧縮化 I/O プールをロックした (テンポラリ・キャッシュ用に繰り返された) 回数。無視してかまわない。
IOWait	圧縮化 I/O プール上のロックのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。無視してかまわない。
HTLock	Sybase IQ がブロック・マップ・ハッシュ・テーブルをロックした (テンポラリ・キャッシュ用に繰り返された) 回数。

出力フィールド	説明
HTWait	ブロック・マップ・ハッシュ・テーブルのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。HTLock と HTWait は、使用中のブロック・マップ数を示す。
FLLock	Sybase IQ がフリー・リストをロックする必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
FLWait	フリー・リスト上のロックのために Sybase IQ が待機する必要があった (テンポラリ・キャッシュ用に繰り返された) 回数。
MemLks	Sybase IQ がメモリ・マネージャ (ヒープ) をロックした回数。
MemWts	メモリ・マネージャ・ロックのために Sybase IQ が待機する必要があった回数。

注意： オペレーティング・システムが進歩したため、Sybase IQ ではスピン・ロックを使用しなくなりました。 このため、[woTO]、[Loops]、[TOs] の統計はめったに使用されません。

参照：
- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-threads
処理スレッド・マネージャのカウントを表示します。値はサーバワイドです (つまり、メインとプライベートのどちらでこのオプションを選択するかは無関係です)。レポートの最後のページ以降の新しいイベントを表します。

使用法
```
monitor_options -threads
```

出力

表 23 : -thread の出力フィールド

出力フィールド	説明
cpus	Sybase IQ が使用している CPU の数。システムに搭載されている数より少ない場合がある。
Limit	Sybase IQ が使用できるスレッドの最大数。
NTeams	現在使用中のスレッド・チームの数。
MaxTms	今まで使用されたチームの最大数。
NThrds	既存スレッドの現在の数。
Resrvd	システム (接続) での使用のために予約されているスレッドの数。
Free	割り当てに使用可能なスレッドの数。この数値が非常に小さい場合は、スレッドの不足を示しているので、モニタリングが必要。
Locks	スレッド・マネージャで処理されたロックの数。
Waits	スレッド・マネージャ上のロックのために Sybase IQ が待機する必要があった回数。

参照：

- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-interval

レポート間隔を秒単位で指定します。デフォルトは 60 秒ごとです。最小値は 2 秒ごとです。通常、クエリの実行中やパフォーマンスに問題があるときに、モニタをデフォルトの間隔で実行すると、有益な結果を取得できます。間隔が短すぎる

と、意味のある結果を取得できないことがあります。ジョブ時間に見合った間隔を指定してください。通常は 1 分で十分です。

使用法
```
monitor_options -interval
```

出力
最初の表示では、サーバの起動からのカウンタが示されます。それ以降の記録では、前の表示との差が示されます。

参照：
- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -append | - truncate (83 ページ)
- -debug (84 ページ)

-append | - truncate
前者は既存の出力ファイルに追加し、後者は既存の出力ファイルをトランケートします。デフォルトでは、トランケートされます。

使用法
```
monitor_options -append | -truncate
```

参照：
- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)

- -debug (84 ページ)

-debug
同じ情報を扱う標準表示モードがあるかどうかにかかわらず、パフォーマンス・モニタで使用可能な情報がすべて表示されます。**-debug** は、主に、Sybase 製品の保守契約を結んでいるサポート・センタに情報を提供するために使用されます。

使用法
```
monitor_options -debug
```

出力
ページの上部には、ディスク・ブロック・タイプごとの統計の配列が表示されます。次に、他のバッファ・キャッシュの統計、メモリ・マネージャの統計、スレッド・マネージャの統計、フリー・リストの統計、CPU 使用率、そして最後にバッファ・アロケータの統計が表示されます。

バッファ・アロケータの統計はさらにクライアント・タイプ (ハッシュ、ソートなど) ごとに集計され、最後に行われたバッファ割り付けのヒストグラムが表示されます。メモリ割り付けは、レポートの最後のページ以降に割り付けられた量を示します。

参照：
- -summary (73 ページ)
- -cache (74 ページ)
- -cache_by_type (76 ページ)
- -file_suffix (77 ページ)
- -io (77 ページ)
- -bufalloc (78 ページ)
- -contention (80 ページ)
- -threads (81 ページ)
- -interval (82 ページ)
- -append | - truncate (83 ページ)

モニタ実行中の結果の確認

UNIX システムでは、クエリの実行中にモニタリング出力を確認できます。

たとえば、次のコマンドを使用してモニタを起動するとします。
```
iq utilities main into monitor_tab  start monitor “-cache -interval 2
-file_suffix iqmon”
```

このコマンドを実行すると、結果が dbname.conn#-[main|temp]-iqmon という名前の ASCII ファイルに出力されます。したがって、データベース iqdemo では、結果が iqdemo.2-main-iqmon に出力されます。

結果を確認するには、システム・プロンプトで次のコマンドを入力します。

```
$ tail -f iqdemo.2-main-iqmon
```

参照：

- バッファ・キャッシュ・モニタの起動 (72 ページ)
- 出力オプション (73 ページ)
- バッファ・キャッシュ・モニタの停止 (85 ページ)
- モニタリング結果の検査と保存 (85 ページ)

バッファ・キャッシュ・モニタの停止

モニタの停止コマンドは起動コマンドとほぼ同じですが、オプションを指定する必要はありません。

次の構文を使って Sybase IQ バッファ・キャッシュ・モニタを停止します。

```
        IQ UTILITIES { MAIN | PRIVATE }
INTO
        dummy_table_name
        STOP MONITOR
```

注意： 特定のオプション設定を有効にするには、データベースを再起動します。モニタを実行している場合は、データベースを再起動できるようにモニタをシャットダウンする必要があります。

参照：

- バッファ・キャッシュ・モニタの起動 (72 ページ)
- 出力オプション (73 ページ)
- モニタ実行中の結果の確認 (84 ページ)
- モニタリング結果の検査と保存 (85 ページ)

モニタリング結果の検査と保存

バッファ・キャッシュ・モニタは、実行ごとに結果を記録します。

ログのデフォルトの名前は次のとおりです。

- dbname.connection#-main-iqmon (メイン・バッファ・キャッシュの結果)

- `dbname.connection#-temp-iqmon` (テンポラリ・バッファ・キャッシュの結果)

プレフィクス *dbname.connection#* は、データベース名と接続番号を示します。接続番号が複数あって、どれが自分のものかわからない場合は、カタログ・ストアド・プロシージャ **sa_conn_info** を実行してください。このプロシージャを実行すると、アクティブなデータベース接続のそれぞれについて、接続番号、ユーザ ID などの情報が表示されます。

IQ UTILITIES コマンドで **-file_suffix** パラメータを使用すると、サフィックス `iqmon` を任意のサフィックスに変更できます。

モニタの実行結果を表示するには、テキスト・エディタを使用するか、ファイルの表示や印刷に通常使用している方法が他にあれば、それを使用してください。

同じデータベースから同じ接続番号を使ってモニタを再度起動する場合、デフォルトでは前回の結果が上書きされます。モニタの実行結果を保存する場合は、ファイルを別の場所にコピーした後で同じデータベースからモニタを再度起動するか、**-append** オプションを使用してください。

参照：

- バッファ・キャッシュ・モニタの起動 (72 ページ)
- 出力オプション (73 ページ)
- モニタ実行中の結果の確認 (84 ページ)
- バッファ・キャッシュ・モニタの停止 (85 ページ)

バッファ・キャッシュの構造

CACHE_PARTITIONS 値を変更すると、マルチ CPU 構成でロードまたはクエリのパフォーマンスが向上することがあります。

Sybase IQ では、システム上の CPU の数に応じて、バッファ・キャッシュのキャッシュ・パーティションの数が自動的に計算されます。マルチ CPU 構成でロードまたはクエリのパフォーマンスが予想より悪い場合は、CACHE_PARTITIONS データベース・オプションの値を変更するとパフォーマンスが向上することがあります。 詳細については、『リファレンス：文とオプション』の「CACHE_PARTITIONS オプション」を参照してください。

バッファは、キャッシュの LRU (Least Recently Used) 側の終端に近づくと、ウォッシュ・マーカを越えます。Sybase IQ は最も古いページ (ウォッシュ・マーカを越えたページ) をディスクに書き出して、そのページが占有していたキャッシュ領域を再利用できるようにします。 スイーパ・スレッドと呼ばれる Sybase IQ 処理スレッドのチームが、最も古いバッファを一掃し (書き出し) ます。

データのページをキャッシュに読み込む必要がある場合、Sybase IQ は LRU バッファを取り込みます。バッファがまだ「ダーティな」(変更された) 状態の場合は、先にバッファをディスクに書き込む必要があります。モニタ **-cache** レポートの [Gdirty] カラムは、LRU バッファをダーティな状態で取り込んだために、Sybase IQ がそのバッファを使用する前に書き出す必要があった回数を示します。

通常、Sybase IQ では [Gdirty] の値が 0 に維持されます。この値が 0 より大きい状態が長時間続く場合は、スイーパ・スレッドの数とウォッシュ・マーカを制御するデータベース・オプションを調整する必要があります。『リファレンス：文とオプション』の「SWEEPER_THREADS_PERCENT オプション」または「WASH_AREA_BUFFERS_PERCENT オプション」を参照してください。

参照：

- Sybase IQ 環境の表示 Sybase IQ のパフォーマンスをチューニングする最初の手順は、環境を調べることです。
- パフォーマンス統計のモニタリング (61 ページ)
- バッファ・キャッシュのモニタリング (71 ページ)
- バッファ・マネージャのスラッシングの回避 (87 ページ)
- CPU 使用率をモニタリングするシステム・ユーティリティ (95 ページ)
- バッファ・キャッシュ・モニタリング・チェックリスト (91 ページ)

バッファ・マネージャのスラッシングの回避

システムが要求されたページを読み込む前にダーティ・ページを書き込む必要がある場合、スラッシングが発生し、システムの速度が大幅に低下します。最適なパフォーマンスを得るためには、ページ・ライタが空き領域の需要に対応できるように常に十分な空きキャッシュを割り付けます。

バッファ・キャッシュ・スラッシング
バッファ・キャッシュ・スラッシングはシステム・スラッシングに似ていて、読み込みに使用できるクリーン・バッファが十分にない場合に発生します。このスラッシングは、キャッシュで「書き込んでから読み込む」場合と同様の遅延を生じ、バッファ・キャッシュが不十分でクエリで参照されたすべてのオブジェクトに対応できない場合に発生します。

バッファ・キャッシュ・スラッシングを排除するには、バッファ・キャッシュに十分なメモリを割り付ける必要があります。バッファ・キャッシュへの過度の割り付けには注意してください。データベース・バッファ・キャッシュにメモリを過度に割り付けると、システム・スラッシングを誘発します。極端な場合、メモリの過度の割り付けはバッファ・キャッシュ・スラッシング問題を解決しないまま、複数のレベルのスラッシングを引き起こす可能性があります。

マルチユーザ・コンテキストにおいて、または、クエリに使用可能なキャッシュに適した数より非常に多くの値で HASH オブジェクトを構築する必要のある環境で複雑なクエリに起因するスキューや不確実性が原因でオプティマイザが HASH アルゴリズムを選択する場合に、より不明確なバッファ・キャッシュ・スラッシングが発生する可能性もあります。

バッファ・サイズの設定
バッファ・サイズを設定するときは、次のトレードオフに注意してください。

- Sybase IQ バッファ・キャッシュが大きすぎると、Sybase IQ が全メモリを使用しようとするため、オペレーティング・システムでページングが強制的に行われる。
- Sybase IQ バッファ・キャッシュが小さすぎると、Sybase IQ はクエリ・データをキャッシュに収めきれないため、スラッシングしてしまう。

深刻なパフォーマンスの問題が発生した場合は、ページングをモニタリングして、スラッシングが問題かどうかを確認してください。スラッシングが問題だった場合は、バッファ・サイズをリセットしてください。

クエリとハッシュ・アルゴリズム
ページングをモニタし、スラッシングが問題と判断した場合は、ハッシュ・アルゴリズムを伴うクエリが含まれる文の実行時のスラッシングの量を制限することもできます。HASH_THRASHING_PERCENT データベース・オプションを調整し、許容するハード・ディスク I/O の割合を制御します。この割合を超えると、文がロールバックされてエラーが返されます。

HASH_THRASHING_PERCENT のデフォルト値は 10% です。
HASH_THRASHING_PERCENT 値を大きくするとロールバックするまでのディスクへのページング量が増え、HASH_THRASHING_PERCENT 値を小さくするとこれが減ります。

以前のバージョンの Sybase IQ では実行されていた、ハッシュ・アルゴリズムを伴うクエリが、デフォルトの HASH_THRASHING_PERCENT の制限に達するとロールバックされるようになります。Sybase IQ はエラー `Hash insert thrashing detected` またはエラー `Hash find thrashing detected` を報告します。実行に必要なリソースをクエリに割り当てるには、次の 1 つ以上の対応策を講じてください。

- HASH_THRASHING_PERCENT の値を増やし、ページングの制限を緩和する。
- テンポラリ・キャッシュのサイズを増やす (DBA のみ)。システム・スラッシングが生じる可能性を回避するため、テンポラリ・キャッシュのサイズを増やすと、メイン・キャッシュ割り付けから同じサイズが減ることに注意してください。
- Sybase IQ がこの文の 1 つ以上のハッシュ・サイズの見積もりを誤っている原因を突き止めて改善する。 たとえば、LF または HG インデックスを必要とする

すべてのカラムにそれがあるかどうかを確認します。 また、複数カラムのインデックスが適切かどうかも検討します。
- データベース・オプション HASH_PINNABLE_CACHE_PERCENT の値を減らす。

これらのデータベース・オプションの詳細については、『リファレンス：文とオプション』の「HASH_THRASHING_PERCENT オプション」と「HASH_PINNABLE_CACHE_PERCENT オプション」を参照してください。

クエリで起きている可能性のある問題を特定するには、テンポラリ・データベース・オプションの QUERY_PLAN = 'ON' と QUERY_DETAIL = 'ON', を指定してクエリを実行し、クエリ・プランを生成します。次に、クエリ・プランの見積もりを調査します。 生成されたクエリ・プランはメッセージ・ログ・ファイルにあります。

参照：
- バッファ・キャッシュの管理 (7 ページ)

Windows システムでのページングのモニタリング

Windows パフォーマンス ツールを使用して、ページングとオブジェクトのメモリをモニタします。

システム モニターにアクセスして、[LogicalDisk] オブジェクト下の PAGEFILE.SYS, ファイルが格納されているディスクのインスタンスと [Disk Transfers/Sec] カウンタを選択します。Windows ページ・ファイルはデータベース DB 領域デバイスとは別のディスクに格納してください。オブジェクト [Memory] と [Pages/Sec] カウンタもモニタリングできます。ただし、この値はソフト・フォールトとハード・フォールトの両方を含む全メモリ・フォールトの合計となります。

参照：
- UNIX 系オペレーティング・システムでのページングのモニタリング (90 ページ)

UNIX 系オペレーティング・システムでのページングのモニタリング

vmstat、**top**、または **topas** を使用して、ページングなどのシステム・アクティビティをモニタします。

表 24 : UNIX 系オペレーティング・システムのモニタリング・ユーティリティ

コマンド	プラットフォーム	説明
vmstat	Solaris、Linux、HP-UX	**vmstat** は仮想メモリに関する統計を表示
top	Solaris、Linux、HP-UX	**top** は CPU プロセッサ・アクティビティの上位を表示
topas	AIX	**topas** はローカル・システムに関する統計を生成

注意： 構文とオプションについては、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。

参照：

- Windows システムでのページングのモニタリング (89 ページ)

バッファ・キャッシュ・モニタリング・チェックリスト

このチェックリストを確認して、通常の範囲を超えているキャッシュ動作を調整します。

表 25 : バッファ・キャッシュ・モニタリング・チェックリスト

統計	正常な動作	調整が必要な動作	推奨される対応策
HR% (Cache hit rate)	90% 以上。 GARRAY、BARRAY、Bitmap (bm)、hash object、sort object、variable-length btree (btreev)、fixed-length btree (btreef)、bit vector (bv)、dbext、dbid、vdo、store、checkpoint block (ckpt) などの個別の内部データ構造体の場合、クエリの実行中はヒット率が 90% を上回る。最初は 90% を下回る場合がある。プリフェッチが機能し始めると (PF または PrefetchReqs > 0)、ヒット率が徐々に上昇して 90% を超える。	プリフェッチが機能した後もヒット率が 90% を下回る。 **注意 :** オブジェクトによってはプリフェッチを行わないものがあるため、これらのヒット率は一般に低くなる。	**-iqmc** と **-iqtc** を調整し、メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。 PREFETCH_THREADS_PERCENT オプションを調整し、プリフェッチ・スレッド数を増やしてみる。

統計	正常な動作	調整が必要な動作	推奨される対応策
Gdirty (Grabbed Dirty)	適度なキャッシュ・サイズ (10GB 未満) が設定されたシステムでは 0。	GDirty > 0 **注意：** スイーパ・スレッドがアクティブになるのは、ダーティ・ページの数がウォッシュ・エリアの一定の割合に達した場合だけである。GDirty/GrabbedDirty が 0 より大きく、I/O 率 (Writes) が低い場合は、単にシステムに軽い負荷がかかっていると考えられるため対応策は不要。	SWEEPER_THREADS_PERCENT オプション (デフォルトは 10%) または WASH_AREA_ BUFFERS_PERCENT オプション (デフォルトは 20%) を調整し、ウォッシュ・エリアのサイズを増やす。
BWaits (Buffer Busy Waits)	0	> 0 の状態が持続し、複数のジョブが同じバッファで衝突していることを示している。	I/O 率 (Writes) が高い場合は、キャッシュのスラッシングが原因で Busy Waits が起きていると考えられる。キャッシュ・レポートでヒット率を調べて、メインとテンポラリのキャッシュのバランスをとり直す必要があるかどうかを確認する。 ほぼ同一の多数のクエリを同時にバッチ・ジョブで開始している場合は、開始時刻をずらしてみる。
LRU Waits (デバッグ・レポートでは、LRUNum TimeOuts percentage)	20% 以下	> 20%。これは重大な競合問題が起きていることを示す。	オペレーティング・システムのパッチ・レベルやその他の環境設定を確認する。これはオペレーティング・システムの問題の可能性が高い。
IOWait (IONum-Waits)	10% 以下	> 10%	ディスク・エラーや I/O リトライを調べる。

統計	正常な動作	調整が必要な動作	推奨される対応策
FLWait (FLMutexWaits)	20% 以下	> 20%	DB 領域の設定を確認する。 データベース領域が不足しかかっていないか？ DISK_STRIPING が ON になっているか？ **sp_iqcheckdb** が 15% を超える断片化を報告していないか？
HTWait (BmapHTNumWaits) MemWts (MemNtimesWaited) (PFMgrCondVarWaits)	10% 以下	> 10%	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る。
CPU time (デバッグ・レポートでは CPU Sys Seconds、CPU Total Seconds)	CPU Sys Seconds < 20%	CPU Sys Seconds > 20% CPU Total Seconds も低い使用率を示しており、システムがビジー状態になるだけのジョブがある場合は、キャッシュがスラッシングしているか、並列度が失われていると考えられる。	**-iqgovern** を調整し、実行できる同時クエリの合計数を減らす。 キャッシュ・レポートでヒット率と I/O 率を調べて、キャッシュがスラッシングしていないかどうかを確認する。また、cache_by_type (またはデバッグ) レポートでハッシュ・オブジェクトのヒット率を調べて、ハッシュ・オブジェクトがスラッシングしていないかどうかを確認する： ヒット率が 90% 未満で I/O 率 (Writes) が高くないか？ 試行された並列処理をクエリ・プランで確認する。十分なスレッドが使用可能だったか？ システムに大量の CPU が搭載されているか？マルチプレックス構成などの対策が必要な場合もある。

統計	正常な動作	調整が必要な動作	推奨される対応策
InUse% (Buffers in use)	起動時以外は 100% かそれに近い値	100% 未満	バッファ・キャッシュが大きすぎる可能性がある。 **-iqmc** と **-iqtc** を調整し、メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。
Pin% (Pinned buffers)	< 90%	> 90 〜 95%。システムがバッファ不足状態に危険なほど近づいていることを示す。この状態になるとトランザクションがロールバックされる。	メインとテンポラリのキャッシュ・サイズのバランスをとり直してみる。 バッファ・キャッシュ・サイズのバランスをとり直すことができない場合は、**-iqgovern** を減らして、同時に実行されるジョブの数を制限してみる。
Free threads (ThrNumFree)	Free > Resrvd	空きスレッドの数が予約済みの数まで減少している場合は、システムのスレッドが不足していると考えられる。	次のいずれかを試してみる。 **-iqmt** を設定してスレッドの数を増やす。 スレッド関連の次のオプションの値を減らす： `MAX_IQ_THREADS_ PER_CONNECTION`, `MAX_IQ_THREADS_ PER_TEAM`. `USER_RESOURCE_ RESERVATION` を設定し、クエリ・エンジンのリソース割り付けを制限する。 **-iqgovern** を設定し、ジョブの数を制限する。
FlOutOfSpace (デバッグのみ)	0。このストアのフリー・リストが満杯でないことを示す。割り付けられていないページが使用可能	1。このストア (メインまたはテンポラリ) が全部割り付けられていることを示す。	ストアに DB 領域を追加する。

参照：

- Sybase IQ 環境の表示 Sybase IQ のパフォーマンスをチューニングする最初の手順は、環境を調べることです。
- パフォーマンス統計のモニタリング (61 ページ)
- バッファ・キャッシュのモニタリング (71 ページ)
- バッファ・キャッシュの構造 (86 ページ)
- バッファ・マネージャのスラッシングの回避 (87 ページ)
- CPU 使用率をモニタリングするシステム・ユーティリティ (95 ページ)

CPU 使用率をモニタリングするシステム・ユーティリティ

OS 固有のユーティリティを使用して、CPU 使用率をモニタリングできます。

表 26 : OS 固有のモニタリング・ユーティリティ

OS	ユーティリティ	説明
UNIX	**top** (Solaris、Linux、HP-UX)、**topas** (IBM-AIX)	プロセッサ・アクティビティを継続してリアル・タイムで参照します。
	ps	プロセス・ステータスをレポートします。
	vmstat	システム・プロセス、メモリ、ページング、ブロック IQ、トラップ、CPU アクティビティに関する情報を表示します。
	iostat -x	ディスクのサブシステム情報を表示します。
Windows	システム モニターのタスク マネージャ	コンピュータのパフォーマンス、実行しているアプリケーション、プロセス、CPU 使用率、および他のシステム・サービスに関する詳細情報を提供します。

参照：

- Sybase IQ 環境の表示 Sybase IQ のパフォーマンスをチューニングする最初の手順は、環境を調べることです。
- パフォーマンス統計のモニタリング (61 ページ)
- バッファ・キャッシュのモニタリング (71 ページ)
- バッファ・キャッシュの構造 (86 ページ)
- バッファ・マネージャのスラッシングの回避 (87 ページ)
- バッファ・キャッシュ・モニタリング・チェックリスト (91 ページ)

クエリと削除の最適化

クエリの計画、構築、制御を行うための推奨事項

クエリ構築のヒント

クエリの構造を改善することにより、クエリの実行速度が向上することがあります。

- サブクエリを含むコマンド文をジョインとして構成することによって、実行速度を高めることができる場合があります。
- GROUP BY 句で複数のカラムをグループ化する場合、可能であれば、カラムに対応するユニークな値を基にして降順にカラムをリストします。 これによって最適なクエリのパフォーマンスが実現されます。
- 追加のカラムを使用して、頻繁に行う計算の結果を格納すると、パフォーマンスを向上させることができます。

ORDER BY クエリ・パフォーマンスの強化

マルチカラム HG インデックスを使用することにより、ORDER BY クエリのパフォーマンスを強化できます。

マルチカラム HG インデックスを使用して、単一テーブル・クエリ内の複数カラムへの参照がある ORDER BY クエリのパフォーマンスを強化できます。 この変更により、ユーザが意識することなく、クエリ・パフォーマンスが向上します。

ORDER BY 句に複数のカラムが含まれるクエリは、マルチカラム HG インデックスを使用した方が処理速度が向上する可能性があります。たとえばテーブル T にマルチカラム・インデックス HG(x,y,z) がある場合、このインデックスは射影の順序付けに使用されます。

```
SELECT abs (x) FROM T
ORDER BY x, y
```

上記の例では、HG インデックスはソート順に x と y を縦方向に射影します。

ROWID() 関数が SELECT リスト式内にある場合、マルチカラム HG インデックスも使用されます。 例を示します。

```
SELECT rowid()+x, z FROM T
ORDER BY x,y,z
```

ROWID() が ORDER BY リストの末尾にあり、ROWID() を除くそのリストのカラムがインデックス内に存在し、順序付けキーが先行する HG カラムの順序に一致す

る場合、マルチカラム・インデックスがクエリに使用されます。 次に例を示します。

```
SELECT z,y FROM T
ORDER BY x,y,z,ROWID()
```

参照：

- サブクエリのパフォーマンスの改善 (98 ページ)
- キャッシュ方法の使用 (98 ページ)
- UNION ALL での GROUP BY がクエリ・パフォーマンスに与える影響特定の状況では、分割 GROUP BY 方法を使用することにより、クエリの処理時間を短縮できます。

サブクエリのパフォーマンスの改善

SUBQUERY_FLATTENING_PREFERENCE と SUBQUERY_FLATTENING_PERCENT を使用して、サブクエリのフラット化を制御します。

サブクエリのフラット化は、オプティマイザがサブクエリの入ったクエリを、ジョインを使用するクエリに書き換える最適化方法です。Sybase IQ によって多くのサブクエリがフラット化されますが、すべてではありません。SUBQUERY_FLATTENING_PREFERENCE と SUBQUERY_FLATTENING_PERCENT を使用して、オプティマイザがこの最適化を使用する状況を制御します。

FLATTEN_SUBQUERIES オプションは、Sybase IQ 15.0 では廃止されています。

参照：

- UNION ALL での GROUP BY がクエリ・パフォーマンスに与える影響特定の状況では、分割 GROUP BY 方法を使用することにより、クエリの処理時間を短縮できます。
- キャッシュ方法の使用 (98 ページ)
- ORDER BY クエリ・パフォーマンスの強化 (97 ページ)

キャッシュ方法の使用

SUBQUERY_CACHING_PREFERENCE オプションを設定して、相関サブクエリのキャッシュ方法を選択します。

相関サブクエリにはサブクエリ外の 1 つまたは複数のテーブルへの参照が含まれており、参照されるカラムの値が変更されるたびに再実行されます。SUBQUERY_CACHING_PREFERENCE オプションを使用して、相関サブクエリを実行するためのキャッシュ方法を選択します。

参照：

- サブクエリのパフォーマンスの改善 (98 ページ)
- ORDER BY クエリ・パフォーマンスの強化 (97 ページ)
- UNION ALL での GROUP BY がクエリ・パフォーマンスに与える影響特定の状況では、分割 GROUP BY 方法を使用することにより、クエリの処理時間を短縮できます。

クエリ・プラン

クエリ・プランを生成することにより、オプティマイザが作成した実行プランを理解するのに役立ちます。

最も効果的な構文を使用していなくても、正しいインデックスを作成していれば、通常は Sybase IQ クエリ・オプティマイザによって、最も効率的な方法でクエリを実行できます。

クエリを実行する前に、Sybase IQ クエリ・オプティマイザはクエリ・プランを作成します。Sybase IQ では、これ以降の項で説明するオプションを使用して、クエリ・プランを調査および変更し、クエリを評価できます。 このオプションを指定する方法の詳細については、『リファレンス：文とオプション』を参照してください。

クエリ評価オプション

適切なオプション設定は、クエリ・プランを評価するのに役立ちます。

- `INDEX_ADVISOR – ON` に設定されていると、インデックス・アドバイザは、Sybase IQ クエリ・プランの一部として、または、クエリ・プランが無効の場合に Sybase IQ メッセージ・ログ・ファイル内の独立したメッセージとして、推奨されるインデックスを表示します。 これらのメッセージは、"Index Advisor:" という文字列で始まります。 この文字列を検索することで、Sybase IQ メッセージ・ファイルからこれらのメッセージをフィルタできます。このオプションはメッセージを `OWNER.TABLE.COLUMN` 形式で出力します。このオプションのデフォルト設定は `OFF` です。

  詳細については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「システム・プロシージャ」>「sp_iqindexadvice プロシージャ」を参照してください。
- `INDEX_ADVISOR_MAX_ROWS` ― このオプションはインデックス・アドバイザによって格納されるメッセージの数を制限します。 指定した制限値に到達すると、`INDEX_ADVISOR` は新しいアドバイスの保存を停止します。 ただし、既存のアドバイスのカウントとタイムスタンプの更新は続行します。

- NOEXEC — このオプションを ON に設定すると、Sybase IQ によってクエリ・プランが生成されますが、クエリ全体は実行されません。
  EARLY_PREDICATE_EXECUTION オプションを ON に設定すると、クエリの一部は実行されます。
  EARLY_PREDICATE_EXECUTION を OFF に設定すると、クエリを通常どおり実行する場合に比べてクエリ・プランが大きく異なる可能性があるため、OFF に設定しないでください。
- QUERY_DETAIL — このオプションと、QUERY_PLAN または QUERY_PLAN_AS_HTML のいずれかを ON に設定すると、クエリ・プランの作成時にクエリに関する追加情報が表示されます。 QUERY_PLAN と QUERY_PLAN_AS_HTML が OFF の場合は、このオプションは無視されます。
- QUERY_PLAN – このオプションが ON に設定されている場合 (デフォルト)、Sybase IQ はクエリについてのメッセージを生成します。ジョイン・インデックスの使用方法、ジョイン順序、クエリのジョイン・アルゴリズムについてのメッセージなどが生成されます。
- QUERY_PLAN_TEXT_ACCESS — このオプションが ON の場合、Interactive SQL クライアントから IQ クエリ・プランを表示、保存、出力できます。
  QUERY_PLAN_ACCESS_FROM_CLIENT オプションが OFF の場合、クエリ・プランはキャッシュされないので、クエリ・プランが関係する他のデータベース・オプションが Interactive SQL クライアントのクエリ・プランの表示に影響を与えることはありません。 このオプションはデフォルトで OFF になっています。
  『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「GRAPHICAL_PLAN 関数 [文字列]」と「HTML_PLAN 関数 [文字列]」を参照してください。
- QUERY_PLAN_AFTER_RUN – ON に設定されている場合、クエリの実行が完了するとクエリ・プランが出力されます。これにより、クエリの各ノードから渡された実際のロー数などの情報をクエリ・プランに追加できます。このオプションを使用するには、QUERY_PLAN を ON にします。 このオプションはデフォルトで OFF になっています。
- QUERY_PLAN_AS_HTML – Web ブラウザで表示できるように、HTML 形式のグラフィカルなクエリ・プランを生成します。HTML 形式では、ノード間にハイパーリンクが設定されるため、テキスト形式の .iqmsg ファイルよりはるかに使いやすくなります。 クエリ・プランのファイル名にクエリ名を含めるには、QUERY_NAME オプションを使用します。 このオプションはデフォルトで OFF になっています。
- QUERY_PLAN_AS_HTML_DIRECTORY — QUERY_PLAN_AS_HTML が ON で、ディレクトリが QUERY_PLAN_AS_HTML_DIRECTORY で指定されている場合、

Sybase IQ は指定されたディレクトリに HTML のクエリ・プランを書き込みます。

- QUERY_PLAN_TEXT_CACHING ― プランをキャッシュできるようにリソースを制御できます。 このオプションが OFF の場合 (デフォルト)、クエリ・プランは該当するユーザ接続でキャッシュされません。
  QUERY_PLAN_TEXT_ACCESS オプションを OFF にすると、QUERY_PLAN_TEXT_CACHING の設定にかかわらず、そのユーザからの接続ではクエリ・プランはキャッシュされなくなります。
  『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「GRAPHICAL_PLAN 関数 [文字列]」と「HTML_PLAN 関数 [文字列]」を参照してください。
- QUERY_TIMING ― サブクエリのタイミング統計の収集などクエリ・エンジンの反復的な機能を制御します。 非常に短い相関サブクエリの場合、各サブクエリを実行するタイミングを合わせる処理のために、全体のパフォーマンスが大幅に低下するため、このオプションは、通常、OFF (デフォルト) にします。

注意： クエリ・プランを生成すると、.iqmsg ファイルに大量のテキストが追加される場合があります。QUERY_PLAN が ON の場合で、QUERY_DETAIL も ON である場合は特に、メッセージ・ログのラッピングまたはメッセージ・ログのアーカイブを有効にして、メッセージ・ログ・ファイルが満杯にならないようにしてください。 詳細については、『システム管理ガイド：第 1 巻』の「Sybase IQ システム管理の概要」>「メッセージ・ログ・ラッピング」を参照してください。

参照：
- クエリ・ツリー (101 ページ)
- クエリ・プランの使用 (102 ページ)

クエリ・ツリー

クエリ・ツリーは、クエリのデータ・フローを表します。

クエリ・ツリーはノードで構成されます。それぞれのノードは処理の段階を表します。 ツリーの一番下のノードはリーフ・ノードです。各リーフ・ノードはクエリ内のテーブルを表します。

プランの最上部にあるのは、演算子ツリーのルートです。情報はテーブルから上方向に、ジョイン、ソート、フィルタ、格納、集合、サブクエリを表す演算子を通じて流れます。

参照：
- クエリ評価オプション (99 ページ)
- クエリ・プランの使用 (102 ページ)

クエリ・プランの使用

QUERY_PLAN_AS_HTML オプションを設定してクエリ・プランの HTML 版を生成することにより、クエリ・プランを Web ブラウザで表示できます。

HTML クエリ・プランでは、ツリーの各ノードが詳細へのハイパーリンクになっています。 各ボックスがツリーへハイパーリンクされています。任意のノードをクリックし、プラン内をすばやく移動できます。

また、サーバ上の .iqmsg ファイルまたはクエリ・プラン・ファイルにアクセスしなくても、Interactive SQL プラン・ウィンドウでクエリ・プランの表示、出力、保存ができます。

SQL 関数の GRAPHICAL_PLAN と HTML_PLAN は、IQ クエリ・プランを文字列結果セットとしてそれぞれ XML 形式と HTML 形式で返します。データベース・オプションの QUERY_PLAN_TEXT_ACCESS と QUERY_PLAN_TEXT_CACHING は、新しい関数の動作を制御します。

次の方法でクエリ・プランを Interactive SQL プラン・ウィンドウから確認します。

- クエリを実行し、プラン・ウィンドウを開く。[プラン] オプション ([ツール] - [オプション] - [プラン]) から選択したプランの種類に応じて、該当するプランがプラン・ウィンドウに表示される。
  IQ クエリ・プランが表示されるのは、[GRAPHICAL_PLAN] オプションを選択した場合のみ。他のプランではエラー・メッセージ「プラン・タイプはサポートされていません。」が表示される。
- クエリを SQL 文ウィンドウに入力し、[SQL] - [プランの取得] から選択する。[プラン] オプション ([ツール] - [オプション] - [プラン]) から選択したプランの種類に応じて、該当するプランがプラン・ウィンドウに表示される。
  IQ クエリ・プランが表示されるのは、[GRAPHICAL_PLAN] オプションを選択した場合のみ。他のプランではエラー・メッセージ「プラン・タイプはサポートされていません。」が表示される。
- SQL 関数の GRAPHICAL_PLAN と HTML_PLAN を使用して、クエリ・プランを文字列結果として返させる。

クエリ・プランにアクセスするには、SQL 関数の GRAPHICAL_PLAN と HTML_PLAN を、 クエリ SELECT、UPDATE、DELETE、INSERT SELECT、SELECT INTO に対して使用します。

Interactive SQL のクエリ・プランを保存するには、GRAPHICAL_PLAN または HTML_PLAN を使用してクエリ・プランを取得し、OUTPUT 文を使用して出力をファイルに保存します。

保存したプランを表示するには、Interactive SQL クライアント・メニューから [ファイル] - [オープン] を選択して、プランを保存したディレクトリに移動しま

す。[ファイル] - [印刷] を選択して、プラン・ウィンドウに表示されているプランを印刷することもできます。

詳細については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「GRAPHICAL_PLAN 関数 [文字列]」と「HTML_PLAN 関数 [文字列]」を参照してください。これらのクエリ・プラン関数をサポートするオプションについては、『リファレンス：文とオプション』の「QUERY_PLAN_TEXT_ACCESS オプション」と「QUERY_PLAN_TEXT_CACHING オプション」を参照してください。

参照：
- クエリ評価オプション (99 ページ)
- クエリ・ツリー (101 ページ)

クエリ処理の制御

すべてのユーザが、特定のクエリの処理にかかる時間に制限を設定できます。DBA 権限を持つユーザは、特定のユーザのクエリに他のクエリより高い優先度を与えることや、処理のアルゴリズムを変更し、クエリ処理の速度を操作できます。

クエリの時間制限の設定

MAX_QUERY_TIME オプションを設定することにより、クエリの実行時間を制限できます。 クエリの実行時間が MAX_QUERY_TIME より長くなると、Sybase IQ は適切なエラーを表示してクエリを停止します。

注意： Sybase IQ では、小数の *option-value* の設定がすべて整数値にトランケートされます。 たとえば、3.8 という値は 3 にトランケートされます。

参照：
- クエリの優先度の設定 (103 ページ)
- クエリ最適化オプションの設定 (104 ページ)
- ユーザ指定の条件ヒントの設定 (105 ページ)
- 負荷のモニタリング (106 ページ)

クエリの優先度の設定

クエリの優先度オプションを設定することにより、クエリ処理にユーザ別の優先度を割り当てることができます。

処理をキューで待機しているクエリは、そのクエリを送信したユーザの優先度、そしてクエリが送信された順序の順に実行されます。優先度の高いクエリがすべて実行されるまで、優先度の低いキューのクエリは実行されません。

次のオプションは、クエリにユーザ別の処理の優先度を割り当てます。

- IQGOVERN_PRIORITY – 処理キューで待機しているクエリに数字の優先度 (1、2、または 3 で、1 が最も高い) を割り当てます。
- IQGOVERN_MAX_PRIORITY – DBA はユーザまたはグループの IQGOVERN_PRIORITY に上限値を設定できます。
- IQ_GOVERN_PRIORITY_TIME — 優先度の高い (優先度 1 の) クエリが、指定した時間より長く -iqgovern キューで待機している場合に、優先度の高いユーザを開始できます。

クエリの優先度を調べるには、sp_iqcontext ストアド・プロシージャによって返される IQGovernPriority 属性を確認します。

参照：

- クエリの時間制限の設定 (103 ページ)
- クエリ最適化オプションの設定 (104 ページ)
- ユーザ指定の条件ヒントの設定 (105 ページ)
- 負荷のモニタリング (106 ページ)

クエリ最適化オプションの設定

最適化オプションは、クエリ処理の速度に影響します。

- AGGREGATION_PREFERENCE – 集合関数 (GROUP BY、DISTINCT、SET) を処理するときに使用するアルゴリズムを選択します。このオプションは、主に内部用として設計されているため、経験のあるデータベース管理者のみが使用してください。
- DEFAULT_HAVING_SELECTIVITY_PPM – クエリ内のすべての HAVING 述部の選択性を設定します。この設定が、HAVING 句によってフィルタされるロー数をオプティマイザが見積もった設定より優先して使用されます。
- DEFAULT_LIKE_MATCH_SELECTIVITY_PPM – たとえば、LIKE '*string* *%string*' (% はワイルドカード文字) のような一般的な LIKE 述部のデフォルトの選択性を設定します。 他に選択性に関する情報が提供されておらず、一致文字列が一連の文字定数と後続の単一文字のワイルドカードで始まっていない場合、オプティマイザはこのオプションを参照します。
- DEFAULT_LIKE_RANGE_SELECTIVITY_PPM – 先頭の文字定数 LIKE 述部のデフォルトの選択性を LIKE '*string%*' という形式で設定します。ここで、一致文字列は一連の文字定数と後続の単一文字のワイルドカード (%) で構成されます。 選択性に関する情報が提供されていない場合、オプティマイザはこのオプションを参照します。
- MAX_HASH_ROWS — クエリ・オプティマイザがハッシュ・アルゴリズムを使用するときに考慮する最大ロー数の推測値を設定します。 デフォルトは、

2,500,000 ローです。たとえば、2 つのテーブル間にジョインがあり、両方のテーブルからジョインに入力されるロー数がこのオプションで設定された値を超えると、オプティマイザはハッシュ・ジョインを選択肢から外します。TEMP_CACHE_MEMORY_MB がユーザあたり 50MB を超えるシステムの場合は、このオプションにさらに大きな値を設定します。

- MAX_JOIN_ENUMERATION ― オプティマイザによる単純化が適用された後で、ジョイン順序を最適化するためのテーブルの最大数を設定します。 通常は、このオプションを設定する必要はありません。

参照：

- クエリの時間制限の設定 (103 ページ)
- クエリの優先度の設定 (103 ページ)
- ユーザ指定の条件ヒントの設定 (105 ページ)
- 負荷のモニタリング (106 ページ)

ユーザ指定の条件ヒントの設定

選択性ヒントは、オプティマイザが適切なクエリ方式を選択するのに役立ちます。

Sybase IQ クエリ・オプティマイザは、使用可能なインデックスの情報を使用して、クエリを実行するための適切な方式を選択します。クエリ内の各条件について、オプティマイザはインデックスを使用して条件を実行できるかどうかを決定します。条件を実行できる場合、オプティマイザはインデックスを選択し、そのテーブル上の他の条件に対する順序を決定します。 これらの決定で最も重要な要因になるのは、条件の選択性、つまり条件を満たすテーブル・ローの割合です。

オプティマイザは通常、ユーザの介入なしに、一般的に最適な決定を行います。ただし、状況によっては、オプティマイザが条件の実行前にその選択性を正確に決定できない場合があります。これらの状況は通常、条件が適切なインデックスを使用できないカラムを対象としている場合、または算術演算または関数式が含まれるために条件が複雑すぎてオプティマイザが正確に予測できない場合に発生します。

構文、パラメータ、例については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「SQL 言語の要素」>「ユーザ指定の条件ヒント」を参照してください。

参照：

- クエリの時間制限の設定 (103 ページ)
- クエリの優先度の設定 (103 ページ)
- クエリ最適化オプションの設定 (104 ページ)
- 負荷のモニタリング (106 ページ)

負荷のモニタリング

テーブル、カラム、インデックスの各使用状況をモニタするストアド・プロシージャを使用して、クエリのパフォーマンスを向上させます。

最適化のメタデータを提供するため、またユニーク性とプライマリ／外部キーの関係を確保するために、インデックスが作成されることがよくあります。ただし、いったんインデックスが作成されると、インデックスがもたらす利点を数量化するという難題が DBA に発生します。

複数の接続によりアクセスされるか、または長期間アクセスされる必要があるデータを一時的に記憶するために、IQ メイン・ストアにテーブルが作成されることがよくあります。このテーブルは、貴重なディスク領域を継続的に使用しているうちに忘れられてしまう可能性があります。さらに、データ・ウェアハウス内のテーブルの数が多くなりすぎたり、負荷が複雑すぎて手作業で使用状況を分析できなくなったりします。

そのため、使用されていないインデックスとテーブルは、ディスク領域の浪費、バックアップ・タイムの延長、DML パフォーマンスの低下の原因となります。

Sybase IQ には、指定された負荷の統計情報の収集と分析を行うための各種ツールが用意されています。 DBA は、クエリで参照されているために維持する必要があるデータベース・オブジェクトをすぐに判断できます。 使用されていないテーブル、カラム、インデックスを削除して、浪費される領域の低減、DML パフォーマンスの向上、バックアップ・タイムの短縮を達成できます。

負荷のモニタリングはストアド・プロシージャを使用して実現されます。ストアド・プロシージャは、収集処理を制御し、テーブルとカラムの使用状況とインデックス情報を詳細に報告します。 これらのプロシージャは `INDEX_ADVISOR` 機能を補完します。この機能はクエリのパフォーマンスを改善する可能性があるカラム・インデックスの追加を推奨するメッセージを生成します。 推奨されたインデックスを追加したら、その使用状況を追跡することにより、保持するだけの価値があるかどうかを判断できます。

負荷のモニタリング・プロシージャの詳細については、『リファレンス：ビルディング・ブロック、テーブル、およびプロシージャ』の「sp_iqcolumnuse プロシージャ」、「sp_iqindexadvice プロシージャ」、「sp_iqindexuse プロシージャ」、「sp_iqtableuse プロシージャ」、「sp_iqunusedcolumn プロシージャ」、「sp_iqunusedindex プロシージャ」、「sp_iqunusedtable プロシージャ」、「sp_iqworkmon プロシージャ」を参照してください。

『リファレンス：文とオプション』の「INDEX_ADVISOR オプション」も参照してください。

参照：

- クエリの時間制限の設定 (103 ページ)

- クエリの優先度の設定 (103 ページ)
- クエリ最適化オプションの設定 (104 ページ)
- ユーザ指定の条件ヒントの設定 (105 ページ)

削除オペレーションの最適化

Sybase IQ は、HG インデックスと WD インデックスが設定されたカラムで削除オペレーションを処理するために最適なアルゴリズムを選択します。

HG 削除オペレーション

Sybase IQ は、HG (High_Group) インデックスが設定されたカラムで削除オペレーションを処理するために、次の 3 つのアルゴリズムから 1 つを選択します。

- スモール・デリートは、非常に少数のグループからローを削除するときに最適なパフォーマンスが得られます。 通常は、削除するローが 1 つだけか、HG インデックスが設定されたカラムに等号述部がある場合に選択されます。 スモール・デリート・アルゴリズムは、HG にランダムにアクセスできます。 最悪の場合、I/O はアクセスされるグループの数に比例します。
- ミッド・デリートは、いくつかのグループからローを削除するときに最適なパフォーマンスが得られます。ただし、それらのグループが十分に分散されているか、十分に少なくて、あまり多くの HG ページがアクセスされないことが条件です。ミッド・デリート・アルゴリズムは、HG へのアクセスを順番に提供します。最悪の場合、I/O はインデックス・ページ数によって制限されます。ミッド・デリートでは、削除するレコードをソートするというコストが加わります。
- ラージ・デリートは、多数のグループからローを削除するときに最適なパフォーマンスが得られます。 ラージ・デリートでは、すべてのローが削除されるまで HG が順番にスキャンされます。 最悪の場合、I/O はインデックス・ページ数によって制限されます。 ラージ・デリートは並列処理ですが、並列度はインデックスの内部構造および削除対象のグループの分散度によって制限されます。 HG カラムの範囲述部を使用して、ラージ・デリートのスキャン範囲を狭めることができます。

HG 削除コスト

削除コスト・モデルでは、I/O コスト、CPU コスト、使用可能なリソース、インデックス・メタデータ、並列度、クエリから使用できる述部など、多数の要素が考慮されます。

HG、LF、または enumerated FP インデックスを持つカラムの述部を指定すると、コストが大幅に改善されます。 HG コスト計算でラージ・デリート以外のアルゴリズムを選択するには、削除によって影響を受ける重複しない個別の値 (グルー

プ) の数を判定できる必要があります。個別カウント数は、初めはインデックス・グループの数および削除されるローの数より少ないものと見なされます。 述部を指定することにより、個別カウント数の見積もりを改善でき、さらには正確な見積もりを提供することも可能です。

現在のコスト計算では、ラージ・デリートにおける範囲述部の効果を考慮していません。そのため、ラージ・デリートの方が速いケースでミッド・デリートが選択されることもあります。そうしたケースでは、必要に応じて強制的にラージ・デリート・アルゴリズムを適用できます。これについては、次の項で説明します。

HG 削除パフォーマンス・オプションの使用

HG_DELETE_METHOD オプションを使用すると、HG 削除パフォーマンスを制御できます。

HG_DELETE_METHOD オプションで指定したパラメータの値に対応する削除アルゴリズムが強制的に使用されます。

- 1 = スモール・デリート
- 2 = ラージ・デリート
- 3 = ミッド・デリート
- DML_OPTIONS5 = 4 (デリート述部のプッシュを無効にします) デフォルトは 0 — HG ラージ・デリートへの範囲述部のプッシュを無効にします。

HG_DELETE_METHOD データベース・オプションの詳細については、『リファレンス：文とオプション』の「データベース・オプション」>「HG_DELETE_METHOD オプション」を参照してください。

参照：

- WD 削除オペレーション (108 ページ)
- TEXT 削除オペレーション (109 ページ)

WD 削除オペレーション

Sybase IQ は、**WD** (Word) インデックスが付いたカラムで削除オペレーションを処理するために、次の 3 つのアルゴリズムから 1 つを選択します。

- スモール・デリートは、削除されるローに個別の単語が少数しか含まれておらず、多くの WD ページにアクセスする必要がない場合に、最適なパフォーマンスが得られます。 WD スモール・デリート・アルゴリズムは、WD へのアクセスを順番に実行します。 最悪の場合、I/O はインデックス・ページ数によって制限されます。 スモール・デリートには、削除するレコード内の単語とレコード ID をソートするというコストが伴います。
- WD のミッド・デリートは、WD スモール・デリートの変形で、スモール・デリートと同じ条件下では便利です。つまり、削除されるローに個別の単語が少

数しかない場合です。 WD のミッド・デリートでは、削除するレコード内の単語のみをソートします。 このソートは並列処理ですが、並列度は単語数と使用可能な CPU スレッド数によって制限されます。 Word インデックスの場合、通常は、ミッド・デリートを使用した方がスモール・デリートより高速です。
- ラージ・デリートは、削除されるローに個別の単語が多数含まれているために、インデックス内の多数の「グループ」にアクセスする必要がある場合に、最適なパフォーマンスが得られます。 ラージ・デリートでは、すべてのローが削除されるまで WD が順番にスキャンされます。 最悪の場合、I/O はインデックス・ページ数によって制限されます。 ラージ・デリートは並列処理ですが、並列度はインデックスの内部構造および削除対象のグループの分散度によって制限されます。

WD 削除コスト
WD 削除コスト・モデルでは、I/O コスト、CPU コスト、使用可能なリソース、インデックス・メタデータ、並列度など、多数の要素が考慮されます。

WD_DELETE_METHOD データベース・オプションを使用すると、WD 削除パフォーマンスを制御できます。

WD 削除パフォーマンス・オプションの使用
WD_DELETE_METHOD オプションで指定したパラメータの値に対応する削除アルゴリズムが強制的に使用されます。

- 0 = コスト・モデルで選択されたミッド・デリートまたはラージ・デリート
- 1 = スモール・デリート
- 2 = ラージ・デリート
- 3 = ミッド・デリート

WD_DELETE_METHOD データベース・オプションの詳細については、『リファレンス：文とオプション』の「データベース・オプション」>「WD_DELETE_METHOD オプション」を参照してください。

参照：
- HG 削除オペレーション (107 ページ)
- TEXT 削除オペレーション (109 ページ)

TEXT 削除オペレーション

Sybase IQ は、TEXT インデックスが設定されたカラムで削除オペレーションを処理するために、次の 2 つのアルゴリズムから 1 つを選択します。

- スモール・デリートは、削除されるローに個別の単語が少数しか含まれておらず、多くの TEXT ページにアクセスする必要がない場合に、最適なパフォーマ

ンスが得られます。 TEXT スモール・デリート・アルゴリズムは、TEXT へのアクセスを順番に実行します。 最悪の場合、I/O はインデックス・ページ数によって制限されます。 スモール・デリートには、削除するレコード内の単語とレコード ID をソートするというコストが伴います。

- ラージ・デリートは、削除されるローに個別の単語が多数含まれているために、インデックス内の多数の「グループ」にアクセスする必要がある場合に、最適なパフォーマンスが得られます。 ラージ・デリートでは、すべてのローが削除されるまで TEXT が順番にスキャンされます。 最悪の場合、I/O はインデックス・ページ数によって制限されます。 ラージ・デリートは並列処理ですが、並列度はインデックスの内部構造および削除対象のグループの分散度によって制限されます。

TEXT 削除コスト

TEXT 削除コスト・モデルでは、I/O コスト、CPU コスト、使用可能なリソース、インデックス・メタデータ、並列度など、多数の要素が考慮されます。

TEXT_DELETE_METHOD データベース・オプションを使用すると、TEXT 削除パフォーマンスを制御できます。

TEXT 削除パフォーマンス・オプションの使用

TEXT_DELETE_METHOD オプションで指定したパラメータの値に対応する削除アルゴリズムが強制的に使用されます。

- 0 = コスト・モデルで選択されたミッド・デリートまたはラージ・デリート
- 1 = スモール・デリート
- 2 = ラージ・デリート

TEXT_DELETE_METHOD データベース・オプションの詳細については、『Sybase IQ の非構造化データ分析の概要』の「TEXT インデックスとテキスト設定オブジェクト」>「TEXT_DELETE_METHOD オプション」を参照してください。

参照：

- HG 削除オペレーション (107 ページ)
- WD 削除オペレーション (108 ページ)

索引

記号

A

B

C

D

E

F

H

I

J

L

M

N

O

P

R

S

T

U

V

W

い

お

か

き

く

さ

し

す

た

ち

て

と

ね

は

ひ

ふ

へ

ま

め

も

ゆ

ら

り

ろ