新機能ガイド

Adaptive Server® Enterprise 16.0

ドキュメント ID： DC01058-01-1600-01
改訂： 2014 年 3 月

目次

バージョン 16.0 の概要 1
パーティションロックによるデータ可用性の向上 5
パーティションロック 5
パーティションロック名 6
パーティションロックの有効化 6
sp_lock によるパーティションロックの表示 7
sp_familylock によるパーティションロックの表示 8
monLocks および monDeadLock によるパーティションロックの表示 8
デッドロックとロックタイムアウト 9
パーティションロックプロモーション 9
パーティションロックプロモーションスレッショルドの設定 10
パーティションロックプロモーションスレッショルドの削除 10
ロックの共存性とロックの十分性 10
パーティションロックの共存性 11
パーティションロックの十分性 12
スキーマロックの共存性 12
スキーマロックの十分性 13
パーティションレベルオンラインオペレーションの同時実行性の向上 13
パーティションレベルオンラインオペレーションの構文 13
パーティションレベルオンラインオペレーションの同時実行性 14
グローバルインデックスを使用するパーティションレベルオンラインオペレーション 15

スキーマロック 15
HANA サーバのコンポーネント統合サービスサポート 17
HANA 対応 CIS の設定 17
Windows での ODBC データソースとしての SAP HANA の作成 17
SAP ASE interfaces ファイルへの SAP HANA の追加 18
PCI ブリッジと PCA/ODBC の設定 18
サーバクラス HANAODBC の追加 20
SAP ASE と HANA 間のデータ型のマッピング 21
制限 23
クエリ制限の緩和 27
スタージョインによるクエリプランの最適化 29
スタージョインヒント 30
use fact_table ヒントに従ったスタージョインクエリプラン 32
クエリのパフォーマンス強化 39
動的スレッド割り当て 39
SORT 演算子のパフォーマンス強化 40
ハッシュジョイン演算子のパフォーマンス強化 42
全文監査 45
ストアドプロシージャ内の権限チェックの監査 47
オブジェクト定義の置換 49
インストールスクリプトの変更 50
データセグメントおよびログセグメントの変更 51
HTML 形式のクエリプランと実行統計 53
生成されたファイルのプレフィクスのオプション 55
インデックス圧縮 57
インデックス圧縮の有効化 58
インデックス圧縮テーブルの作成 59
圧縮インデックスの作成 59
圧縮状態の変更 60

SAP JVM サポート 63
データベースの完全暗号化 65
データベースの完全暗号化とカラムの暗号化 65
データベース暗号化キーの作成 66
データベース暗号化キーの変更 67
データベース暗号化キーの削除 68
暗号化データベースの作成 69
既存データベースの暗号化 70
暗号化のステータスと進行状況 73
パフォーマンスの考慮事項 73
暗号化処理のサスペンド 77
quiesce database コマンドと完全に暗号化されたデータベース 77
暗号化処理の再開 77
暗号化されたデータベースの復号化 78
完全暗号化データベースのリカバリ 78
完全暗号化データベースのバックアップ (ダンプ) 79
データベース暗号化キーのバックアップ 79
完全に暗号化されたデータベースのバックアップのリストア (ロード) 80
暗号化されたデータベースのロード動作 80
暗号化中のデータベースの削除 82
完全暗号化データベースのマウントおよびマウント解除 83
アーカイブデータベースと完全暗号化 83
データベースの完全暗号化とシステム変更 84
スケーラビリティの拡張と機能 87
実行時ロギングの拡張 87
ロック管理の拡張 88
メタデータとラッチ管理の拡張 89
スレッショルドベースイベントの監視 93
複数のトリガ 95

複数のトリガの作成 95
トリガ起動時の順序の変更 96
Merge 文のトリガの順序 96
複数のトリガによるトランザクション動作 97
トリガの無効化と再有効化 97
@@trigger_name グローバル変数 97
残存データの削除 99
設定履歴の追跡 101
設定変更を追跡するための SAP ASE の設定 101
取得された変更 102
ch_events のクエリによる変更表示 106
dump database 用巡回冗長検査 109
トランザクションログの増大率の計算 111
システムの変更 113
設定パラメータ 113
新しい設定パラメータ 113
変更された設定パラメータ 117
組み込み関数 117
dbencryption_status 118
コマンド 119
データベースの完全暗号化に使用される alter database 119
alter index 122
インデックス圧縮用の alter table 123
複数のトリガに使用される alter table 125
残存データの削除に使用される alter table 125
データベースの完全暗号化に使用される create archive database 126
データベースの完全暗号化に使用される create database 127
create default 128
create encryption key 130
create function 133

create function (SQLJ) 136
create index 138
create procedure 139
create procedure (SQLJ) 143
create rule 145
インデックス圧縮用の create table 147
残存データの削除に使用される create table 149
複数のトリガで使用される create trigger 152
create trigger for or replace 152
create view 155
drop encryption key 159
drop trigger 160
dump database 161
kill 161
load database 162
select 162
select into 163
set 163
システムプロシージャ 166
変更されたシステムプロシージャ 166
新しいシステムプロシージャ 173
システムテーブル 187
変更されたシステムテーブル 187
新しいシステムテーブル 189
変更されたモニタリングテーブル 191
新しいモニタリングテーブル 195
ユーティリティ 197
ddlgen 198
sybmigrate 199
sybrestore 200
グローバル変数 200

バージョン 16.0 の概要

SAP® Adaptive Server®Enterprise バージョン 16.0 は、メジャーリリースです。スケーラビリティとパフォーマンス、大きなデータサイズの管理、データ可用性の向上、セキュリティと監査、容易な管理と維持に関わる重要な機能拡張が含まれます。

- スケーラビリティとパフォーマンスの強化には、競合を最小化する最適化バッファ管理が含まれます。
- メタデータとラッチ管理機能の強化には、高スループットへの対応が含まれます。
- パーティションレベルロックからのデータ可用性の向上により、1 つのテーブル上での DDL と DML の同時実行が可能になっています。
- 大きなデータ管理機能の拡張は、SAP® ASE バージョン 15.7 の拡張の上に構築されており、クエリ制限の緩和、スタージョインによるクエリプランの最適化、`SORT` 演算子のパフォーマンス強化を含む関連するクエリパフォーマンスの拡張が含まれます。
- インデックス圧縮など、新しいデータ圧縮方法が、ページおよびカラムレベルの圧縮、その他以前のリリースで追加された圧縮手法と組み合わせて提供されています。
- SCC の拡張機能には、監視と管理の拡張、スレッショルドベースイベントの拡張、HTML 形式でのクエリプランと実行統計、および圧縮データについて情報に基づいた決断を下すためのアドバイザが含まれます。

機能	説明
スケーラビリティとパフォーマンスの機能	
リニアなスケールアップ	リニアなスケールアップを 64 コアまでサポート
動的スレッドのサポート	少ないリソースで並列クエリプランをより早く実行する
インデックス圧縮	より効率的なデータの格納、メモリ消費量の削減、I/O 要求の緩和によるパフォーマンスの向上
複数トリガのサポート	複数のトリガを作成し、文の実行後ファイルをトリガする順序を指定する
パーティションロック	ロックの細分性をより細かくして、同時実行による DDL 文と DML 文用の追加パーティションへのアクセスを可能にすることで、データの可用性を向上させる

機能	説明
設定履歴の追跡	**sp_confighistory** システムプロシージャを使用して、サーバ設定への変更を追跡する
スレッショルドベースのイベント監視テーブル	スレッショルドイベントを設定、記録、および一覧表示する
実行時間監視の拡張	`monThresholdEvent` モニタリングテーブルには、SAP ASE により記録された各イベントに 1 ロー登録
monCachedStatement の更新	完了済みまたは処理中のクエリについて、約 5 秒ごとに選択されるカラムの測定基準を更新する
インストーラの拡張	SAP ASE プロセスのユーザを指定する
データベース制限の引き上げ	カラムとテーブルの制限を増加させる
create または replace コマンド	新しいオブジェクトを作成する、または同じ名前の既存オブジェクトを置換する
セキュリティ機能と監査機能	
データベース全体の暗号化	データベース全体を暗号化する
残存データの削除	データを機密とマーキングする、削除操作や更新操作後の残存データをディスクから削除するよう SAP ASE を設定する
全文監査	次のコマンドに対する全文監査情報: • **select** • **insert** • **delete** • **update** • **select into**
HANA との相互運用性	
HANA 対応の CIS サポート	SAP ASE から HANA サーバに直接接続できるように、コンポーネント統合サービス (CIS) にネイティブ ODBC インタフェースを追加する
SCC 機能	
データベースとトランザクションのバックアップのスケジュール	データベースとトランザクションのバックアップをスケジュールする
圧縮アドバイザ	圧縮のメリットがあるテーブルを特定し、圧縮についての予測と推奨事項を提供する

機能	説明
システム設定の制限アラート	特定リソースが、設定されているスレッショルドを超えるとアラートが生成されるように設定する
スケーラビリティ	最大 250 サーバを管理する
ツールサポート	
sybrestore の機能強化	マスタデータベースの破損後 SAP ASE サーバをリストアする 対話モードおよび非対話モードでの新しいパラメータ

パーティションロックによるデータ可用性の向上

パーティションレベルのロック機能を使用して、ロックの細分性をより細かくして同時 DDL 文と DML 文の他のパーティションへのアクセスを可能にすることで、データ可用性が向上します。

パーティションロック

パーティションロックは、分割されたテーブルの属性であり、すべてのタイプの分割に適用できます。

- 意図的パーティションロック - テーブルのパーティションでページレベルまたはローレベルのロックが保持されることを示します。 SAP ASE は、意図的パーティションロックを各共有パーティションロックまたは排他パーティションロックと共に適用します。そのため、意図的ロックは、排他ロックまたは共有ロックのいずれかです。 意図的ロックを設定すると、他のトランザクションがテーブル上の競合するパーティションレベルのロックをパーティションロックとして取得することができなくなります。
- 共有パーティションロック - テーブル全体に影響を及ぼす点を除き、共有ページロックまたは共有ローロックと似ています。 たとえば、SAP ASE は、**holdlock** 句を持つ **select** コマンドに対し、このコマンドがインデックスを使用しない場合に共有パーティションロックを適用します。 また、**create nonclustered index** コマンドも共有パーティションロックを取得します。
- 排他パーティションロック - テーブル全体に影響を及ぼす点を除き、排他ページロックまたは排他ローロックと似ています。 たとえば、SAP ASE は、**create clustered index** コマンドに対し排他パーティションロックを適用します。 データオンリーロックパーティションに対する **update** 文と **delete** 文は、検索引数がオブジェクトのインデックスカラムを参照しない場合に、排他パーティションロックを必要とします。
- カバーリングパーティションロック - これらのロックは、詳細なロックの取得時に DDL および DML が *partitionid* を認識しない場合に必要となります。

その他のロックタイプについては、『パフォーマンス＆チューニングシリーズ: ロックと同時実行制御』を参照してください。

このテーブル例には、基本的な SQL 文に対して SAP ASE が使用するページロックまたはローロックのそれぞれのパーティションロックが示されています。 これらの例では、`acct_number` に対するインデックスがあります。

文	全ページロックテーブル	データローロックテーブル
`select balance from account where acct_number = 25`	意図的共有パーティションロック 共有ページロック	意図的共有パーティションロック 共有ローロック
`insert account values (34, 500)`	意図的排他パーティションロック データページに対する排他ページロック リーフインデックスページに対する排他ページロック	意図的排他パーティションロック 排他ローロック
`delete account where acct_number = 25`	意図的排他パーティションロック 更新ページロックの後に、データページとリーフレベルのインデックスページに対する排他ページロック	意図的排他パーティションロック 更新ローロックの後に、データローに対する排他ローロック
`update account set balance = 0 where acct_number = 25`	意図的排他パーティションロック 更新ページロックの後に、データページとリーフレベルのインデックスページに対する排他ページロック	意図的排他パーティションロック 更新ローロックの後に、データローに対する排他ローロック

パーティションロック名

SAP ASE は、データベース ID (*dbid*)、オブジェクト ID (*objid*)、およびパーティション ID (*ptnid*) の組み合わせでパーティションをユニークに識別します。

不明なパーティションロック

不明なパーティションロックの場合、SAP ASE は、パーティション ID の値が '-1' の特別なパーティションロックを使用します。 これは、すべてのテーブルパーティション上のパーティションロックを示しています。 特定のケースでは、SAP ASE がデータローのパーティション ID を認識せず、パーティション ID '-1' のパーティションロックを取得することがあります。

パーティションロックの有効化

sp_chgattribute を使用してパーティションレベルのロックを有効または無効にします。 デフォルトでは、パーティションロックは無効です。

パーティションレベルのロックは、複数のデータパーティションを持つユーザテーブルに対してのみ有効化できます。パーティションレベルのロックはシステムテーブルおよびテンポラリテーブルに対して有効化することはできません。

構文は次のとおりです。

```
sp_chgattribute objectname, 'ptn_locking', value
```

パラメータ:

objectname - **ptn_locking** を変更するテーブルの名前です。

`value` - パーティションレベルのロックを有効にする場合は 1 に設定し、無効にする場合は 0 に設定します。

パーミッション:

sp_chgattribute を実行できるのは、オブジェクト所有者だけです。

例

この例では、*authors* テーブルに対してパーティションレベルのロックを有効にします。

```
sp_chgattribute authors, "ptn_locking", 1
```

この例では、*authors* テーブルに対してパーティションレベルのロックを無効にします。

```
sp_chgattribute authors, "ptn_locking", 0
```

sp_lock によるパーティションロックの表示

sp_lock を使用して、SAP ASE によって現在保持されているパーティションロックを表示します。

sp_lock レポートの *partitionid* カラム内のロックされたパーティションを表示します。

この例では、SAP ASE によって現在保持されているパーティションロックなどのロックを表示します。

```
sp_lock
go

fid spid loid locktype table_id partitionid page row dbname class
context
--- ---- ---- -------- -------- ----------- ---- --- ------ ------
-------
0 13 26 Ex_intent 420193516 0 0 0 master Non Cursor Lock
0 13 26 Ex_intent_partition  420193516 452193630 0 0 master Non
Cursor Lock
0 13 26 Ex_page 420193516  452193630 4993 0 master Non Cursor Lock
0 14 28 Ex_intent 420193516 0 0 0  master Non Cursor Lock
0 14 28 Ex_intent_partition 420193516 468193687 0 0  master Non
```

```
Cursor Lock
0 14 28 Ex_page 420193516 468193687 5001 0 master Non Cursor Lock
0 16 32 Sh_intent 1006623598 0 0 0 master Non Cursor Lock
```

参照：

- sp_lock (172 ページ)

sp_familylock によるパーティションロックの表示

sp_familylock を使用して、ファミリによって保持されているロックを表示します。

sp_familylock レポートの *partitionid* カラム内のロックされたパーティションを表示します。

この例では、ファミリによって現在保持されているパーティションロックを表示します。 class カラムには、現在のユーザのカーソルと他のユーザのカーソル ID に関連付けられたロックのカーソル名が表示されます。

```
sp_familylock
go

Fid spid loid locktype table_id partitionid page dbname class context
------ ------ ------ ------------------- ---------------
---------------- -------- ----------- ----------
------------------------------

25 19 50 Sh_cpartition 672002394 -1 0 userdb Non Cursor Lock LOCK
CONTEXT VALUES
25 19 50 Sh_partition 672002394 688002451 0 userdb Non Cursor Lock
LOCK CONTEXT VALUES
25 20 50 Sh_cpartition 672002394 -1 0 userdb Non Cursor Lock LOCK
CONTEXT VALUES
25 20 50 Sh_intent_partition 672002394 688002451 0 userdb Non Cursor
Lock LOCK CONTEXT VALUES
25 20 50 Sh_partition 672002394 704002508 0 userdb Non Cursor Lock
LOCK CONTEXT VALUES
25 25 50 Sh_intent 672002394 0 0 userdb Non Cursor Lock LOCK CONTEXT
VALUES

(6 rows affected)
(return status = 0)
```

参照：

- sp_familylock (170 ページ)

monLocks および monDeadLock によるパーティションロックの表示

`monLocks` モニタリングテーブルと `monDeadLock` モニタリングテーブルの *partitionid* カラムに、ロックされたパーティションを表示します。

monLocks

説明	データ型	属性	説明
partitionid	int	Null	パーティションのユニークな識別子。

monDeadLock

説明	データ型	属性	説明
partitionid	int	Null	パーティションのユニークな識別子。

『パフォーマンス&チューニングシリーズ: モニタリングテーブル』を参照してください。

デッドロックとロックタイムアウト

パーティションロックのデッドロック検出とロックタイムアウトは、テーブルロックの場合と同じ方法で実行されます。

『パフォーマンス&チューニングシリーズ: ロックと同時実行制御』を参照してください。

パーティションロックプロモーション

テーブルロックプロモーションでは、ロックが詳細なロックからテーブルロックに拡大されます。 パーティションロックプロモーションでは、ロックが詳細なロックからパーティションロックに拡大されます。

パーティションロックプロモーションを行うには、パーティションロックプロモーションスレッショルドをゼロ以外の値に設定します。

パーティションロックは、テーブルロックと同じセマンティックを使用します。ロックプロモーションスレッショルドを超える 1 つのパーティションスキャンの下のパーティションに属するページロックまたはローロックは、パーティションロックへのロックプロモーションを開始できます。

不明なパーティションでページロックまたはローロックが取得されると、パーティションロックへのロックプロモーションは完全に無効化されます。 このような場合、ロックはテーブルレベルにのみ昇格することができます。

SAP ASE は、次の 2 つのタイプのロックプロモーションをサポートしています。

- ローロックまたはページロックからテーブルロックへ - 1 回のテーブルのスキャンで、テーブルレベルのロックプロモーションスレッショルドを超える数のローまたはページでタスクがロックを取得すると、SAP ASE は対応するテーブル上で共有テーブルロックまたは排他テーブルロックを取得して、そのテー

ブル内の既存のローまたはページをすべて置き換えようとします。 また、テーブルレベルロックプロモーションも、所有するパーティションがロックプロモーションにとって未知でない場合に、ローロックまたはページロックがカバーリングパーティションで取得されたときに開始されます。

- ローロックまたはページロックからパーティションロックへ - パーティションロックプロモーション時に、該当する共有パーティションロックまたは排他パーティションロックが取得され、スキャン (または DML) の一部として取得された、そのパーティションに属する詳細なロックすべてが解除されます。
  - パーティションロックプロモーションによって、詳細な共有ロックが共有パーティションロックに拡大されます。
  - パーティションロックプロモーションによって、詳細な排他ロックが排他パーティションロックに拡大されます。

パーティションロックプロモーションスレッショルドの設定

sp_setpglockpromote_ptn システムプロシージャと **sp_setrowlockpromote_ptn** システムプロシージャは、サーバレベル、データベースレベル、およびテーブルレベルでパーティションロックプロモーションスレッショルドを設定します。

参照：

- sp_setpglockpromote_ptn (184 ページ)
- sp_setrowlockpromote_ptn (185 ページ)

パーティションロックプロモーションスレッショルドの削除

dropglockpromote_ptn システムプロシージャと **sp_droprowlockpromote_ptn** システムプロシージャは、サーバレベル、データベースレベル、およびテーブルレベルのパーティションロックプロモーションスレッショルドを削除します。

参照：

- sp_ dropglockpromote_ptn (175 ページ)
- sp_droprowlockpromote_ptn (177 ページ)

ロックの共存性とロックの十分性

ロックの共存性とロックの十分性は、ロックと同時実行性の問題をサポートする 2 つの基本概念です。

- ロックの共存性 - タスクがリソース (ロー、ページ、パーティション、テーブルなど) にロックを保持している場合、別のタスクも同じリソースにロックを保持できるかどうか。

- ロックの十分性 - 現在のタスクがリソース (ロー、ページ、パーティション、テーブルなど) に再びアクセスする必要がある場合、そのリソースに保持されている現在のロックが十分かどうか。

『パフォーマンス&チューニングシリーズ: ロックと同時実行制御』>「ロックの概要」を参照してください。

パーティションロックの共存性

次のテーブルに、ロックの共存性についてまとめ、ロックをすぐに取得できる状況を示します。

	別のプロセスが取得できるロック					
保持されているロックのタイプ	**排他パーティションロック**	**共有パーティションロック**	**意図的排他パーティションロック**	**意図的共有パーティションロック**	**排他カバーリングパーティションロック**	**共有カバーリングパーティションロック**
排他パーティションロック	No	No	No	No	No	No
共有パーティションロック	No	Yes	No	Yes	No	Yes
意図的排他パーティションロック	No	No	Yes	Yes	No	No
意図的共有パーティションロック	No	Yes	Yes	Yes	No	Yes
排他カバーリングパーティションロック	No	No	No	No	Yes	Yes
共有カバーリングパーティションロック	No	Yes	No	Yes	Yes	Yes

パーティションロックの十分性

パーティションロックの十分性マトリックステーブルを次に示します。

	タスクが次のロックを必要とする場合のロックの十分性					
保持されているロックのタイプ	排他パーティションロック	共有パーティションロック	意図的排他パーティションロック	意図的共有パーティションロック	排他カバーリングパーティションロック	共有カバーリングパーティションロック
排他パーティションロック	Yes	Yes	Yes	Yes	Yes	Yes
共有パーティションロック	No	Yes	No	Yes	No	Yes
意図的排他パーティションロック	No	No	Yes	Yes	No	No
意図的共有パーティションロック	No	No	No	Yes	No	No
排他カバーリングパーティションロック	No	No	No	No	Yes	Yes
共有カバーリングパーティションロック	No	No	No	No	No	Yes

スキーマロックの共存性

次のテーブルに、スキーマロックの共存性についてまとめ、ロックをすぐに取得できる状況を示します。

	別のプロセスが取得できるロック	
保持されているロックのタイプ	排他スキーマロック	共有スキーマロック
排他スキーマロック	No	No
共有スキーマロック	No	Yes

スキーマロックの十分性

スキーマロックの十分性マトリックステーブルを次に示します。

	タスクが次のロックを必要とする場合のロックの十分性	
保持されているロックのタイプ	**排他スキーマロック**	**共有スキーマロック**
排他スキーマロック	Yes	Yes
共有スキーマロック	No	Yes

パーティションレベルオンラインオペレーションの同時実行性の向上

特定のパーティションレベルオペレーションは、テーブルの異なるパーティションで同時に実行することができます。 パーティションレベルオンラインオペレーションの実行中に、DML もテーブル上で同時に実行することができます。

パーティションレベルオペレーションは次のとおりです。

- **alter table … split partition**
- **alter table … merge partition**
- **alter table … move partition**
- **alter table … drop partition**
- **truncate partition**
- **dbcc checkindex**
- **dbcc checktable**
- **dbcc tablealloc**
- **dbcc indexalloc**

パーティションレベルオンラインオペレーションの構文

データ可用性の向上を可能にするために、**alter table** コマンドと **truncate partition** コマンドでは partition 句 に **with online** サブ句を含めます。

最初にパーティションロックを有効にする必要があります。

alter table の構文を次に示します。

```
alter table table_name
{merge | drop | move | split} partition partition_name
existing_clause
with online
```

構文の説明は次のとおりです。

table_name - 修正するテーブルの名前です。

partition_name - マージ、削除、移動、または分割するパーティションの名前を指定します。

existing_clause - 指定したパーティションアクションに応じた、partition のサブ句です。

with online - オンラインモードで実行します。 テーブルへの同時アクセスを可能にします。

alter truncate partition の構文を次に示します。

```
truncate table table_name
   [partition partition_name]
   [with online]
```

構文の説明は次のとおりです。

table_name - トランケートするテーブルの名前です。

partition_name - トランケートするパーティションの名前を指定します。

with online - オンラインモードで実行します。 テーブルへの同時アクセスを可能にします。

パーティションレベルオンラインオペレーションの同時実行性

複数のパーティションレベルオンラインオペレーションをテーブルの異なるパーティションで同時に実行することができます。

パーティションレベルオンラインオペレーションの実行中に、すべての DML、つまり、**select** (**select into** は対象外)、**insert**、**update**、および **delete** においてテーブルの操作が可能になります。

DML は、パーティションレベルオンラインオペレーションによって操作されるパーティションを除くすべてのパーティションで実行できます。

DML で適切な述部を使用すると、SAP ASE はパーティション削除手法を利用できるようになります。 これにより、DML は必要最低限の数のパーティションに対してのみ実行されます。

パーティション削除を使用しないと、DML はテーブルのすべてのパーティションで実行されます。

DML は、パーティションレベルオペレーションによって同時に操作されているパーティションで実行すると、アボートされます。

注意： 同時 DML によるエラーの発生を回避するには、パーティション削除手法を利用する適切な述部を使用して DML を記述します。 このようにすると、DML は最低数のパーティションに対して実行されます。

パーティションレベルオンラインオペレーション (パーティションの分割やパーティションの移動など) では、テーブルにローカルユニークインデックスが設定されている必要があります。 move コマンドは、split によって操作されるパーティションも含め、テーブル内のすべてのパーティションに対する同時 DML を許されています。

グローバルインデックスを使用するパーティションレベルオンラインオペレーション

グローバルインデックスを持つテーブルに対するパーティションレベルオペレーション動作について説明します。

テーブルに対してパーティションレベルオペレーションが実行されている間は、そのテーブルへの同時 DML アクセスでグローバルインデックスは使用されません。 SAP ASE は、同時 DML スキャンに代替ローカルインデックススキャンまたはテーブルスキャンを使用します。

異なるパーティションに対する複数のパーティションレベルオペレーションは同時に実行できます。 個々のパーティションレベルオペレーションは、グローバルインデックスの再構築を行いません。 最後にコミットされたパーティションレベルオペレーションのみがグローバルインデックスの再構築を行います。

最後のパーティションレベルオペレーションがアボートされると、グローバルインデックスは suspect (疑わしい)とマーク付けされることがあります。 そのような場合、そのグローバルインデックスを明示的に再構築する必要があります。

スキーマロック

スキーマロックにより、同時オペレーションからの独立性を達成することにより、テーブルスキーマまたはメタデータを更新するパーティションレベルオペレーションを向上することができます。

新しいスキーマロックは、次のとおりです。

- 共有スキーマロック - タスクがクエリ実行でテーブルの現在のスキーマを使用することを示します。 スキャンと DML はクエリの実行を開始する前にこのロックを取得します。
- 排他スキーマロック - タスクがテーブルのスキーマを変更することを示します。 パーティションレベルオペレーションはこのロックを取得してテーブルのスキーマを更新します。

HANA サーバのコンポーネント統合サービスサポート

SAP ASE バージョン 16.0 から HANA サーバに直接接続できるようにコンポーネント統合サービス (CIS) にネイティブ ODBC インタフェースを追加します。

CIS を使用してリモートサーバに接続する方法については、『コンポーネント統合サービスユーザーズガイド』を参照してください。

HANA 対応 CIS の設定

SAP ASE を実行する同じマシンに、ODBC ドライブを含む SAP HANA クライアントパッケージをインストールする必要があります。

SAP HANA データベースマニュアルの『Client Installation and Update Guide』を参照してください。

Windows での ODBC データソースとしての SAP HANA の作成

ODBC アドミニストレータは、Windows プラットフォーム上で ODBC データソースを一元的に作成および管理できる場所を提供します。

32 ビット ODBC ドライバを使用する場合は、32 ビットの ODBC アドミニストレータを使用してデータソースを管理してください。64 ビットのクライアントに対しては 64 ビットの ODBC アドミニストレータを使用してください。

1. Windows の [スタート] メニューから [設定] > [コントロール パネル] > [管理ツール] > [データソース (ODBC)] の順に選択します。
2. [追加] をクリックします。
3. ドライバリストから [HDBODBC] を選択します。
4. [完了] をクリックします。
5. [ODBC Configuration] ウィンドウで、[Data Source Name] と [Server:Port] を入力します。 たとえば、TCP/IP プロトコルを使用してアドレス 157.133.66.75 のホスト hanaserver 上のサーバ HANA_DB に接続する場合、TCP/IP を選択して、次のように入力します。

   ```
   host=hanaserver:1870
   ```

   ホストネットワークアドレスを使用する場合は、次のように入力します。

   ```
   host=157.133.66.75:1870
   ```

6. [OK] をクリックします。

SAP ASE interfaces ファイルへの SAP HANA の追加

HANA サーバに接続するには、SAP ASE interfaces ファイルに HANA サーバエントリを追加しておく必要があります。
vi または同様のテキストエディタを使用して SAP HANA サーバエントリを interfaces ファイルまたは sql.ini ファイルに追加します。

次の例では、UNIX 上で、ポート番号 1870 と `libodbcHDB.so` SAP HANA ODBC ドライバを使用する HANA_DB エントリが追加されます。

```
HANA_DB
      query tcp ether 157.133.66.75 1870 libodbcHDB.so
```

次の例では、Windows 上で、Windows レジストリに登録されている HDBODBC (Windows 32-ビット対応は HDBODBC32) ドライバを使用する HANA_DB エントリが追加されます。

```
[HANA_DB]
Query=NLWNSCK,157.133.66.75,1870,HDBODBC
```

注意： PCI ブリッジはデフォルト値として HDBODBC を使用するため、Windows 対応の interfaces ファイル (`sql.ini`) に HDBODBC エントリを含める必要はありません。

PCI ブリッジと PCA/ODBC の設定

SAP ASE は、ターゲット関数を起動する場合、オンデマンドソフトウェアディスパッチを実装する Pluggable Component Interface (PCI) ブリッジを使用して、共有オブジェクトをロードします。

ODBC ドライバマネージャは通常、PCI ブリッジで設定された Pluggable Component Adapter for ODBC (PCA/OBDC) から起動します。 PCA/ODBC はブローカとして動作し、SAP ASE と ODBC ドライバマネージャ間のサービス要求を管理します。 PCA/ODBC は、SAP ASE と ODBC ドライバマネージャ間の双方向で要求を転送し、制御します。

ODBC ドライバマネージャコンポーネントは、SAP ASE から独立しています。 SAP ASE のアップグレードまたは再構築の必要なく、いつでもこれらのソフトウェアコンポーネントを変更またはアップグレードできます。

注意： ODBC ドライバマネージャを使用する前に PCA/ODBC 用の PCI ブリッジを有効にする必要があります。 PCI ブリッジを有効にする際、SAP ASE は `sybpcidb` データベースを必要とします。 `sybpcidb` には、PCI ブリッジと、関連付けられている PCA (PCA/ODBC など) の設定データがすべて含まれています。

1. システム管理者が、**srvbuild** や **sybconfig** のインストールユーティリティまたは設定ユーティリティを使用するか、`${SYBASE}/${SYBASE_ASE}/scripts/installpcidb` スクリプトを実行して、`sybpcidb` データベースを作成してください (『インストールガイド』を参照)。
2. PCI ブリッジを有効にします。
   ```
   sp_configure 'enable pci', 1
   ```
   『システム管理ガイド: 第 1 巻』の「設定パラメータ」 を参照してください。
3. SAP ASE を再起動します。
4. **sp_odbcconfig** を使用して、以下の PCA/ODBC パラメータを設定します。
   - **pca_odbc_load_dir** - (UNIX のみ) ODBC ドライバのロケーションの絶対パスを設定します。 **pca_odbc_load_dir** は、複数ロケーションの配列を作成します。 つまり、同じマシン上に複数の ODBC ドライバがインストールされている場合、**pca_odbc_load_dir** により配列が作成されます。

     pca_odbc_load_dir のデフォルト値は `/usr/sap/hdbclient` です。

     a. `sybpcidb` データベースに移動します。
        ```
        use sybpcidb
        ```
     b. **pca_odbc_load_dir** のパスを設定します。
        ```
        sp_odbcconfig 'add', 'pca_odbc_load_dir',
        'path_to_driver_location'
        ```
     c. **pca_odbc_load_dir** を変更した後、SAP ASE を再起動します。

     使用しなくなったドライバを削除する方法は次のとおりです。
     ```
     use sybpcidb
     go
     sp_odbcconfig 'delete', 'pca_odbc_load_dir',
     'path_to_driver_location'
     ```

     注意： Windows 上では、ドライバコンポーネントのインストール中、ODBC ドライバはシステムレジストリにより管理および保持されます。 Windows プラットフォームのマニュアル、および各ドライバベンダから提供されているマニュアルを参照してください。
   - **pca_odbc_root_dir** - UNIX システムは ODBC データソースを `odbc.ini` ファイルに格納します。 **pca_odbc_root_dir** は、`odbc.ini` が含まれるディレクトリのパスを示します。 **pca_odbc_root_dir** のデフォルト値は $SYBASE です。

     HANA_DB サーバ用の `odbc.ini` 内のエントリ形式は次のとおりです。
     ```
     [HANA_DB]
     DRIVER = libodbcHDB.so
     SERVERNODE = hanaserver:1870
     ```

注意： Windows 上では、ODBC レジストリを使用して ODBC データソースを設定します。 詳細については、Windows のマニュアルを参照してください。

UNIX システムでは、**sp_odbcconfig** を使用して、**pca_odbc_root_dir** の値を設定します。

a. `sybpcidb` データベースに移動します。

```
use sybpcidb
```

b. 新しいパスを入力します。

```
sp_odbcconfig 'update', 'pca_odbc_root_dir', 'old_path',
'new_path'
```

new_path は、$SYBASE への相対ロケーションを示します。 *new_path* の前にスラッシュを付けると、ルートディレクトリへの絶対ロケーションになります。

注意： ODBC ドライバがすでに PCA/ODBC の制御下にロードされている場合は、SAP ASE を再起動する必要があります。 SAP ASE は、**pca_odbc_dsn_enabled** が有効になっている場合のみ `odbc.ini` 内に格納されている DSN レコードを参照します。

- **pca_odbc_dsn_enabled** - PCA/ODBC サブシステムに対して DSN モードを有効または無効にします。 値は次のとおりです。
  - 0 - DSN を無効にします (デフォルト)。 SAP ASE は、interfaces ファイル内の SAP ディレクトリ制御レイヤ (DCL) から DSN 情報を自動的に収集します。
  - 1 - DSN を有効にします。 SAP ASE は、`odbc.ini` 内に格納されている DSN レコードを参照します。

5. (UNIX のみ) `$SYBASE/DataAccess64/ODBC/dm/lib64` を LD_LIBRARY_PATH に追加します。

サーバクラス HANAODBC の追加

SAP ASE バージョン 16.0 では、HANAODBC サーバクラスが追加されています。これにより、ODBC を使用して SAP HANA サーバに接続できるようになります。

SAP ASE を実行する同じマシンに、HANA ODBC ドライバをインストールする必要があります。

1. HANAODBC サーバクラスを追加するには、次のコマンドを実行します。

```
sp_addserver logical_server_name, HANAODBC, ODBC_DSN_name
```

次の例では、`big_hana_server` が `Big_company` DSN として追加されます。

```
sp_addserver big_hana_server, HANAODBC, Big_company
```

2. **sp_addexternlogin** を使用して、SAP ASE ログインアカウントとパスワードをマッピングします。

   次の例では、システム管理者が名前 'HANA_login' と "HANA_login_password" を使用して "big_hana_server" という名前のリモートサーバにログインするように設定します。

   ```
   sp_addexternlogin big_hana_server, sa, HANA_login,
   HANA_login_password
   ```

SAP ASE と HANA 間のデータ型のマッピング

プロキシテーブルを作成するときに、SAP ASE データ型は対応する HANA データ型にマッピングされます。

create table を使用してプロキシテーブルを作成すると、HANAODBC によって次の SAP ASE データ型が HANA データ型にマッピングされます。

SAP ASE データ型	HANA データ型
tinyint	tinyint
smallint	smallint
int	integer
integer	integer
bigint	bigint
unsigned smallint	integer
unsigned int	bigint
unsigned bigint	サポートされていない
numeric (precision, scale)	decimal (precision, scale)
decimal (precision, scale)	decimal (precision, scale)
float (precision)	float (precision)
double (precision)	float
real	real
smallmoney	decimal (10,4)
money	decimal (19,4)

SAP ASE データ型	HANA データ型
smalldatetime	timestamp
datetime	timestamp
date	date
time	time
char(n)	char(n)
varchar(n)	varchar(n)
unichar(n)	nchar(n)
univarchar(n)	nvarchar(n)
nchar(n)	char(n)
nvarchar(n)	varchar(n)
text	clob
unitext	nclob
binary(n)	binary(n)
varbinary(n)	varbinary(n)
image	blob
bit	tinyint
bigdatetime	timestamp
bigtime	サポートされていない

次のデータ型は、リモートの HANA のカラムをローカルのプロキシテーブルのカラムにマッピングするときに使用できます。

リモートの HANA のデータ型	SAP ASE のデータ型
tinyint	tinyint
smallint	smallint
int	integer
integer	integer
bigint	bigint
decimal(precision, scale)	decimal(precision, scale)

リモートの HANA のデータ型	SAP ASE のデータ型
real	real
double	double precision
float(precision)	double precision
date	date
time	time
seconddate	datetime
timestamp	bigdatetime
varchar(n)	varchar(n)
nvarchar(n)	univarchar(n)
alphanum(n)	univarchar(n)
char(n)	char(n)
nchar(n)	unichar(n)
varbinary(n)	varbinary(n)
clob	text
binary (n)	binary (n)
blob	image
nclob	unitext

制限

HANA 対応の CIS サポートにはいくつかの制限があります。

- HANA 対応 CIS は、スレッドカーネルモード用に設定されている SAP ASE サーバでのみサポートされています。 プロセスモード用に設定されている SAP ASE サーバでは HANA 対応 CIS を使用することはできません。
- HANA 対応 CIS は、**where current of** *cursor_name* 句を含む **update** コマンドまたは **delete** コマンドをサポートしていません。
- HANA 対応 CIS は、sensitive カーソルをサポートしていません。
- HANA 対応 CIS は、ラージオブジェクト (LOB) パラメータをサポートしていません。

- HANA LOB はテキストポインタをサポートしていないため、リモート HANA サーバへのテキストポインタを含むクエリを発行することはできません。
- **create index...desc** を発行して、プロキシテーブルにインデックスを作成した場合、**create index** 文の **desc** パラメータは生成されません。 代わりに、リモート HANA テーブル内のインデックスは **asc** パラメータを使用して作成されます。
- リモートテーブルに対する **alter table** 文には制限があります。 次の処理はできません。
  - partition 句を含める
  - 参照整合性制約を含める
  - 制約の追加
  - カラムの置換
  - ロックスキームの変更
  - **check** 条件が設定されたカラムの追加
  - ユニークキー制約またはプライマリキー制約が設定されたカラムの追加
- 1 つの **alter table** コマンドに次のパラメータのうちの 2 つ以上を含めることはできません。
  - **add** *column_name*
  - **modify** *column_name*
  - **drop** *column_name*
- ベーステーブルに位置付け **update** 文および **delete** 文用にユニークインデックスを定義する必要があります。また、このユニークインデックスには、NULL 入力不可のカラムが最低 1 つ存在する必要があります。 このようなユニークインデックスがない場合、SAP ASE はエラーメッセージを発行します。
- HANA プロキシテーブルに対して 2 フェーズコミットによる分散トランザクション管理 (DTM) を使用することはできません。
- HANA プロキシテーブルに影響を及ぼすトランザクションに **save tran** 文と **rollback to savepoint** 文を含めることはできません。
- HANA プロキシテーブルに影響を及ぼすトランザクションで発行された **rollback tran** オペレーションは、現在のトランザクションの最初の文まですべての文をロールバックします。 **rollback tran** オペレーションの後のすべての文は正常に実行されます。
- HANA プロキシテーブルが含まれたトリガで **rollback trigger** を発行することはできません。
- 検索句には、外部 HANA プロシージャにマッピングされているプロキシテーブルを使用するクエリの入力パラメータのみ (結果のカラムはなし) を含めることができます。 たとえば、次のようにこのテーブルを作成するとします。

```
create existing table rpc_table
(
   col1 int,
```

```
   col2 int
) external procedure at "odbchana..SYSTEM.MYPROC"
```

その後、次のクエリを発行します。このクエリでは、col1 が指定されているので検索結果は絞り込まれません。

```
select * from rpc_table where col1 > 100
```

- 外部 HANA プロシージャにマッピングされたプロキシテーブルの入力パラメータの値として null を使用することはできません。 たとえば、次のような文は認められません。

```
select * from rpc_table where _p1 = null
```

- HANA 対応 CIS は、バルクオペレーションをサポートしていません。
- 暗黙的な変換によって、HANA の timestamp データ型にマッピングされた SAP ASE の bigdatetime データ型と bigtime データ型のミリ秒の値の精度が損なわれることがあります。 SAP ASE では、bigdatetime 値と bigtime 値のミリ秒部分を 6 桁を使用して格納します (たとえば、*hh:mi:ss.zzzzzz*)。 しかし、HANA では、bigdatetime 値と bigtime 値のミリ秒部分を、HANA の timestamp データ型に 7 桁で格納します。 ODBC を使用して値を転送すると、ちょうど 3 桁までになります。 利用できる変換フォーマットは最大 3 桁しかサポートしていないため、ODBC からの値を暗黙的に SAP ASE の内部フォーマットに変換すると、精度がさらに低下する可能性があります。
- 検索句でこれらを指定する場合は、バイナリ値に埋め込みを行う必要があります。 次の例の最初の select 文では、c1 の値 (0x123) は完全な埋め込みを行われていないため、結果は返されません。 2 番目の select 文では、c1 の完全な埋め込みが行われた値 (0x012300) が使用されるため、有効な結果が返されます。

```
1> create table cb(c1 binary(3)) at 'odbchana..SYSTEM.cb'
2> go
1> insert into cb values(0x123)
2> go
(1 row affected)

1> select * from cb where c1 = 0x123
2> go
c 1
--------
(0 rows affected)

3> select * from cb where c1 = 0x012300
c1
--------
0x012300
(1 row affected)
```

HANA 対応 CIS 使用時の引用符付き識別子の制限事項は次のとおりです。

- DDL 文に含まれているオブジェクト定義内のオブジェクト名には、少なくとも 1 つの大文字が含まれている必要があります。
- DML 文の引用符付き識別子には、少なくとも 1 つの大文字が含まれている必要があります。
- DDL 文に含まれているオブジェクト定義に小文字のみが含まれている場合は、HANA 対応 CIS でエラーが発生します。 たとえば、次の文ではエラーが発生します。 `<hanaserver>..<schema>.table1`.
- SAP ASE は、プロキシテーブル内のオブジェクト名を引用符を使用せずに作成します。 そのため、リモートの引用符付き識別子に小文字のみが含まれている (たとえば、`column1`) と、SAP ASE は DML 文の実行時に引用符付き識別子に引用符を追加するかどうかを判断できません。 つまり、SAP ASE はデフォルトで小文字を使用するため、`column1` のオブジェクト名を `COLUMN1` の場合と同様に作成します。 対処方法として、すべて小文字の識別子を引用符で囲むトレースフラグ 11242 を使用します。
- HANA 対応 CIS は、角カッコ付き識別子 ([and] など) をサポートしていません。

クエリ制限の緩和

SAP ASE バージョン 16.0 では、一部のカラムとテーブルの制限が引き上げられています。

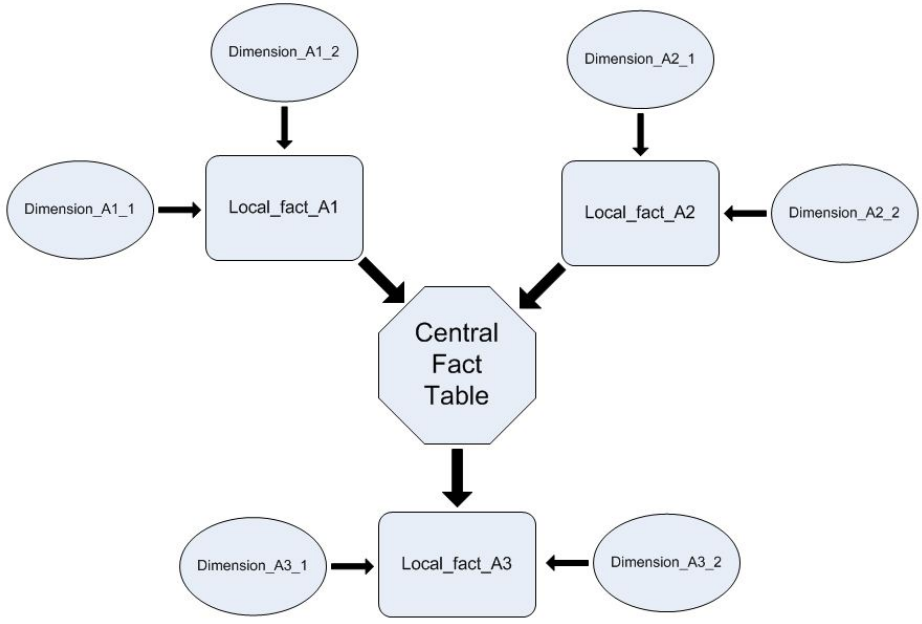
制限	16.0 よりも前のバージョンでの制限	16.0 以降のバージョンでの制限
select クエリ内のテーブルの数	50 テーブル、46 ワークテーブル	250 テーブル、46 ワークテーブル
select クエリ内のサブクエリの数	50	250
order by 句で使用できる最大カラム数	31	400

スタージョインによるクエリプランの最適化

SAP ASE バージョン 16.0 では、スタージョインのパフォーマンスが向上しています。

スタージョインは、一般的に使用されているデータウェアハウスクエリで、ディメンジョンテーブルに囲まれた大型テーブル (ファクトテーブルとも呼ばれる) で構成されるスタースキーマデータベースに対して実行されます。 通常、ファクトテーブルには 2 つのタイプのカラムが含まれています。1 つはビジネスプロセスに関する計測値、測定基準、またはファクトを含み、もう 1 つはディメンジョンテーブルへの外部キーを含みます。 ディメンジョンテーブルには通常、記述的属性が含まれています。 たとえば、ファクトテーブルには特定の作者によるハイキングブックの特定の日における販売数に関する情報が含まれているとします。 対応するディメンジョンテーブルには、作者の住所、年齢、性別が含まれています。

スタージョインでは、ファクトテーブルと 1 つ以上のディメンジョンテーブルが外部キーに従ってジョインされます。 スタージョインは、ディメンジョンテーブルに対するフィルタ述部 (たとえば、**where** gender = 'M') を含みますが、ディメンジョンテーブル間のジョインは含んでいません。



スノーフレークスキーマはスタージョインを拡張したもので、各ディメンジョンテーブルがローカルのファクトテーブルとして機能して別の一連のディメンジョンテーブルとジョインできます。

次のスタージョインでは、Orders ファクトテーブルと Time、Customer、および Location の各ディメンジョンテーブルがジョインされます。

```
select C.dattr, sum(O.measure1), count(*)
from Orders O
    join Time T on O.key_dim1 = T.key_col
    join Customer C on O.key_dim2 = C.key_col
    join Location L on O.key_dim3 = L.key_col
where T.dattr between 1 and 1000
  and C.dattr between 1001 and 2000
  and L.dattr between 901 and 1900
group by C.dattr
```

バージョン 16.0 では、スタージョインクエリ評価のパフォーマンスを向上させるために次の変更が行われています。

- 並列プランでは、ブルームフィルタプローブが Xchg 演算子より下に移動しました。
- 新しいスタージョインヒントでは抽象プランが使用されるため、クエリプロセッサのスタージョインクエリ検出性能が向上します。
- スタージョインの最終プランは、**use fact_table** ヒントに従ってブルームフィルタの使用を強制されるようになりました。
- ハッシュジョインとジョイン順は、**use fact_table** ヒントに従ってスタージョインクエリのファクトテーブルとディメンジョンテーブル間の選択性に基づいて強制されます。 クエリプロセッサは、ブルームフィルタのメリットが最大になるようにクエリプランを調整します。

クエリオプティマイザは、必要に応じてスタージョインを使用します。 スタージョインを使用するように SAP ASE を設定する必要があります。 ただし、SAP では、スタージョインを使用する際に **set join_bloom_filter** オプションと並列クエリ処理を有効にすることをおすすめします。

注意： 最適化目標を **allrows_dss** に設定すると、クエリプロセッサは **set join_bloom_filter** と並列クエリ処理を有効にします。

スタージョインヒント

SAP ASE バージョン 16.0 では、**use fact_table** 抽象プランヒントを導入しています。これは、スタージョインクエリで中央ファクトテーブルを指定し、スタージョインクエリの特別なクエリプラン最適化方式をトリガします。

構文は次のとおりです。

```
PLAN '(use fact_table fact_table_name_or_alias_name)'
```

構文の説明は次のとおりです。

- **fact_table** - クエリにスタージョインのファクトテーブルが含まれていることを示します。
- *fact_table_name_or_alias_name* - 中央ファクトテーブルに使用する名前またはエイリアス名を指定します。 次の例では、F がエイリアス名です。

```
use fact_table 'F'
```

たとえば、次のスタージョインでは、Orders ファクトテーブルを Time、Customer、および Location の各ディメンジョンテーブルにジョインします。

```
select C.dattr, sum(O.measure1), count(*)
from Orders O
join Time T on O.key_dim1 = T.key_col
join Customer C on O.key_dim2 = C.key_col
join Location L on O.key_dim3 = L.key_col
where T.dattr between 1 and 1000
and C.dattr between 1001 and 2000
and L.dattr between 901 and 1900
group by C.dattr
plan '(use fact_table O)'
```

use fact_table ヒントでは、クエリプロセッサのスタークエリ検出アルゴリズムとプラン生成に次の変更が加えられました。

- 検出アルゴリズムは、*fact_table_alias_name* から開始します。 このテーブルは常に、スノーフレークスキーマの中央にあります。
- **or** 句を使用できます。
- ファクトテーブルとディメンジョンテーブル間のジョインを複数カラム等価ジョインにすることができます。 プライマリキーと外部キー間に明示的制約定義を含める必要はありません。
- クエリプロセッサがスタージョインのクエリプランの生成時に (インデックスがない、または選択性の制限が少ないなどの理由で) ネストループジョインに適したプランを見つけられなかった場合は、強制されたジョイン順でハッシュジョインプランを選択します。 ジョイン順は、ディメンジョンテーブルとファクトテーブル間の選択性に基づきます。 選択性が高いほど (つまり ratio 値が低いほど)、ディメンジョンテーブルはそのジョイン順でファクトテーブルにより近くなります。

クエリプロセッサは、クエリが次の条件を満たしていない場合に **use fact_table** ヒントを無視します。

- クエリに少なくとも 3 つのテーブルがある。
- クエリに外部ジョインまたはネストされたサブクエリがない。
- 示唆されたファクトテーブルがクエリブロック内で最も大きなジョインテーブルである。

- ディメンジョンテーブルとファクトテーブルは等価ジョインでジョインされている。
- ディメンジョンテーブル間にジョイン述部がない。

use fact_table ヒントに従ったスタージョインクエリプラン

use fact_table 抽象プランヒントにより、クエリプロセッサはスタージョインクエリに並列ハッシュジョインプランを選択できます。 並列プランにより、クエリプロセッサはブルームフィルタプローブ (ディメンジョンテーブルとファクトテーブル間のより高速なジョインを実現する) を EXCHANGE 演算子より下に移動できるので、ファクトテーブルの該当するローがさらに減ります。

SAP ASE 16.0 より前のバージョンでは、ブルームフィルタは **sp_showplan** 出力内の EXCHANGE 演算子より上に含まれていました。

次の例には、示唆する中央ファクトテーブルとしてテーブル F を含んだヒントに従った並列プランが含まれています。

```
QUERY PLAN FOR STATEMENT 1 (at line 1).
Optimized using Parallel ModeExecuted in parallel by coordinating
process and 66 worker processes.
STEP 1
The type of query is SELECT.

29 operator(s) under root

|ROOT:EMIT Operator (VA = 29)
|
| |HASH VECTOR AGGREGATE Operator (VA = 28)
| | GROUP BY
| | Evaluate Grouped COUNT AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable10 for internal storage.
| | Key Count: 3
| |
```

```
| | |HASH JOIN PROBE Operator (VA = 27) (Join Type: Inner Join)
| | | Using Worktable1 for internal storage.
| | | Key Count: 1
| | |
| | | |HASH JOIN BUILD Operator (VA = 1) (Join Type: Inner Join)
| | | | Using Worktable1 for internal storage.
| | | | Key Count: 1
| | | |
| | | | |SCAN Operator (VA = 0)
| | | | | FROM TABLE
| | | | | /B49/DBENCH01P
| | | | | DP
| | | | | Index : /B49/DBENCH01P~010
| | | | | Forward Scan.
| | | | | Positioning by key.
| | | | | Keys are:
| | | | | SID_0CHNGID ASC
| | | | | SID_0RECORDTP ASC
| | | | | SID_0REQUID ASC
| | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index leaf pages.
| | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | With LRU Buffer Replacement Strategy for data pages.
| | | |
| | | |HASH JOIN PROBE Operator (VA = 26) (Join Type: Inner Join)
| | | | Using Worktable2 for internal storage.
| | | | Key Count: 1
| | | | |
| | | | |HASH JOIN BUILD Operator (VA = 3) (Join Type: Inner Join)
| | | | | Using Worktable2 for internal storage.
| | | | | Key Count: 1
| | | | | |
| | | | | |SCAN Operator (VA = 2)
| | | | | | FROM TABLE
| | | | | | /B49/DBENCH01U
| | | | | | DU
| | | | | | Index : "/B49/DBENCH01U~020"
| | | | | | Forward Scan.
| | | | | | Positioning at index start.
| | | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | With LRU Buffer Replacement Strategy for index leaf pages.
| | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | |
| | | | |HASH JOIN PROBE Operator (VA = 25) (Join Type: Inner Join)
| | | | | Using Worktable3 for internal storage.
| | | | | Key Count: 2
| | | | | |
| | | | | |HASH JOIN BUILD Operator (VA = 5) (Join Type: Inner Join)
| | | | | | Using Worktable3 for internal storage.
| | | | | | Key Count: 2
| | | | | | |
| | | | | | | |SCAN Operator (VA = 4)
| | | | | | | | FROM TABLE
| | | | | | | | /B49/DBENCH01T
| | | | | | | | DT
```

```
| | | | | | | Index : /B49/DBENCH01T~20
| | | | | | | Forward Scan.
| | | | | | | Positioning by key.
| | | | | | | Keys are:
| | | | | | | SID_0CALMONTH ASC
| | | | | | | Using I/O Size 16 Kbytes for index leaf pages.
| | | | | | | With LRU Buffer Replacement Strategy for index leaf
pages.
| | | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | |
| | | | |HASH JOIN PROBE Operator (VA = 24) (Join Type: Inner Join)
| | | | | Using Worktable4 for internal storage.
| | | | | Key Count: 1
| | | | | |
| | | | | |HASH JOIN BUILD Operator (VA = 7) (Join Type: Inner Join)
| | | | | | Using Worktable4 for internal storage.
| | | | | | Building pushdown bloom filter (bv7)
| | | | | | Key Count: 1
| | | | | | |
| | | | | | |SCAN Operator (VA = 6)
| | | | | | | FROM TABLE
| | | | | | | /B49/XCUSTOMER
| | | | | | | X1
| | | | | | | Table Scan.
| | | | | | | Forward Scan.
| | | | | | | Positioning at start of table.
| | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | |
| | | | | |HASH JOIN PROBE Operator (VA = 23) (Join Type: Inner
Join)
| | | | | | Using Worktable5 for internal storage.
| | | | | | Key Count: 1
| | | | | | |
| | | | | | |HASH JOIN BUILD Operator (VA = 9) (Join Type: Inner
Join)
| | | | | | | Using Worktable5 for internal storage.
| | | | | | | Building pushdown bloom filter (bv9)
| | | | | | | Key Count: 1
| | | | | | | |
| | | | | | | |SCAN Operator (VA = 8)
| | | | | | | | FROM TABLE
| | | | | | | | /B49/DBENCH011
| | | | | | | | D1
| | | | | | | | Table Scan.
| | | | | | | | Forward Scan.
| | | | | | | | Positioning at start of table.
| | | | | | | | Using pushdown bloom filter (bv7).
| | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | | |
| | | | | | |EXCHANGE Operator (VA = 22) (Merged)
| | | | | | |Executed in parallel by 64 Producer and 1 Consumer
processes.
```

```
| | | | | | | |
| | | | | | | | |EXCHANGE:EMIT Operator (VA = 21)
| | | | | | | | |
| | | | | | | | | |HASH JOIN PROBE Operator (VA = 20) (Join Type:
Inner Join)
| | | | | | | | | | Using Worktable9 for internal storage.
| | | | | | | | | | Key Count: 1
| | | | | | | | | |
| | | | | | | | | | |EXCHANGE Operator (VA = 13) (Replicated)
| | | | | | | | | | |Executed in parallel by 1 Producer and 64
Consumer processes.

| | | | | | | | | | |
| | | | | | | | | | | |EXCHANGE:EMIT Operator (VA = 12)
| | | | | | | | | | | |
| | | | | | | | | | | |HASH JOIN BUILD Operator (VA = 11) (Join Type:
Inner Join)
| | | | | | | | | | | | Using Worktable6 for internal storage.
| | | | | | | | | | | | Building pushdown bloom filter (bv11)
| | | | | | | | | | | | Key Count: 1
| | | | | | | | | | | |
| | | | | | | | | | | | |SCAN Operator (VA = 10)
| | | | | | | | | | | | | FROM TABLE
| | | | | | | | | | | | | /B49/SSALESORG
| | | | | | | | | | | | | S1
| | | | | | | | | | | | | Index : "/B49/SSALESORG~0"
| | | | | | | | | | | | | Forward Scan.
| | | | | | | | | | | | | Positioning by key.
| | | | | | | | | | | | | Keys are:
| | | | | | | | | | | | | /B49/S_SALESORG ASC
| | | | | | | | | | | | | Using I/O Size 16 Kbytes for index leaf
pages.
| | | | | | | | | | | | | With LRU Buffer Replacement Strategy for
index leaf pages.
| | | | | | | | | | | | | Using I/O Size 16 Kbytes for data pages.
| | | | | | | | | | | | | With LRU Buffer Replacement Strategy for
data pages.
| | | | | | | | | |
| | | | | | | | | | |HASH JOIN PROBE Operator (VA = 19) (Join Type:
Inner Join)
| | | | | | | | | | | Using Worktable8 for internal storage.
| | | | | | | | | | | Key Count: 1
| | | | | | | | | | |
| | | | | | | | | | | |EXCHANGE Operator (VA = 17) (Replicated)
| | | | | | | | | | | |Executed in parallel by 1 Producer and 64
Consumer processes.

| | | | | | | | | | | |
| | | | | | | | | | | | |EXCHANGE:EMIT Operator (VA = 16)
| | | | | | | | | | | | |
| | | | | | | | | | | | | |HASH JOIN BUILD Operator (VA = 15) (Join
Type: Inner Join)
| | | | | | | | | | | | | Using Worktable7 for internal storage.
| | | | | | | | | | | | | Building pushdown bloom filter (bv15)
| | | | | | | | | | | | | Key Count: 1
| | | | | | | | | | | | | |
```

```
| | | | | | | | | | | | | | |SCAN Operator (VA = 14)
| | | | | | | | | | | | | | | FROM TABLE
| | | | | | | | | | | | | | | /B49/DBENCH013
| | | | | | | | | | | | | | | D3
| | | | | | | | | | | | | | | Table Scan.
| | | | | | | | | | | | | | | Forward Scan.
| | | | | | | | | | | | | | | Positioning at start of table.
| | | | | | | | | | | | | | | Using pushdown bloom filter (bv11).
| | | | | | | | | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | | | | | | | | With LRU Buffer Replacement Strategy for
data pages.
| | | | | | | | | | |
| | | | | | | | | | | |SCAN Operator (VA = 18)
| | | | | | | | | | | | FROM TABLE
| | | | | | | | | | | | /B49/FBENCH01
| | | | | | | | | | | | F
| | | | | | | | | | | | Index : /B49/FBENCH01~060
| | | | | | | | | | | | Forward Scan.
| | | | | | | | | | | | Positioning at index start.
| | | | | | | | | | | | Using pushdown bloom filter (bv15, bv9).
| | | | | | | | | | | | Executed in parallel with a 64-way partition
scan.
| | | | | | | | | | | | Using I/O Size 128 Kbytes for index leaf pages.
| | | | | | | | | | | | With LRU Buffer Replacement Strategy for index
leaf pages.
| | | | | | | | | | | | Using I/O Size 128 Kbytes for data pages.
| | | | | | | | | | | | With MRU Buffer Replacement Strategy for data
pages.
```

次の例では、ヒントに従った逐次プランが含まれています。

```
QUERY PLAN FOR STATEMENT 1 (at line 1).
Optimized using Serial Mode


STEP 1
The type of query is SELECT.

8 operator(s) under root

|ROOT:EMIT Operator (VA = 8)
|
| |HASH VECTOR AGGREGATE Operator (VA = 7)
| | GROUP BY
| | Evaluate Grouped COUNT AGGREGATE.
| | Evaluate Grouped SUM OR AVERAGE AGGREGATE.
| | Using Worktable4 for internal storage.
| | Key Count: 1
| |
| | |HASH JOIN Operator (VA = 6) (Join Type: Inner Join)
| | | Using Worktable3 for internal storage.
| | | Building pushdown bloom filter (bv6)
| | | Key Count: 1
| | |
| | | |SCAN Operator (VA = 0)
| | | | FROM TABLE
```

```
| | | | Location
| | | | L
| | | | Table Scan.
| | | | Forward Scan.
| | | | Positioning at start of table.
| | | | Using I/O Size 2 Kbytes for data pages.
| | | | With LRU Buffer Replacement Strategy for data pages.
| | |
| | | |HASH JOIN Operator (VA = 5) (Join Type: Inner Join)
| | | | Using Worktable2 for internal storage.
| | | | Building pushdown bloom filter (bv5)
| | | | Key Count: 1
| | | |
| | | | |SCAN Operator (VA = 1)
| | | | | FROM TABLE
| | | | | Customer
| | | | | C
| | | | | Table Scan.
| | | | | Forward Scan.
| | | | | Positioning at start of table.
| | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | With LRU Buffer Replacement Strategy for data pages.
| | | |
| | | | |HASH JOIN Operator (VA = 4) (Join Type: Inner Join)
| | | | | Using Worktable1 for internal storage.
| | | | | Building pushdown bloom filter (bv4)
| | | | | Key Count: 1
| | | | |
| | | | | |SCAN Operator (VA = 2)
| | | | | | FROM TABLE
| | | | | | Time
| | | | | | T
| | | | | | Table Scan.
| | | | | | Forward Scan.
| | | | | | Positioning at start of table.
| | | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | | With LRU Buffer Replacement Strategy for data pages.
| | | | | |
| | | | | | |SCAN Operator (VA = 3)
| | | | | | | FROM TABLE
| | | | | | | Orders
| | | | | | | O
| | | | | | | Table Scan.
| | | | | | | Forward Scan.
| | | | | | | Positioning at start of table.
| | | | | | | Using pushdown bloom filter (bv4, bv5, bv6).
| | | | | | | Using I/O Size 2 Kbytes for data pages.
| | | | | | | With LRU Buffer Replacement Strategy for data pages.
```

クエリのパフォーマンス強化

SAP ASE バージョン 16.0 では、クエリ機能が強化されています。

動的スレッド割り当て

SAP ASE バージョン 16.0 以降では、静的スレッド割り当てではなく動的スレッド割り当てを使用します。

動的スレッド割り当てにより、SAP ASE は並列クエリプランをより高速に、かつより少ないリソースで実行できます。 SAP ASE は、動的スレッド割り当てを、**select** クエリに生成される並列 Lava クエリプランに適用します。 ただし、次のコマンドは引き続き静的スレッド割り当てを使用します。

- **select into**
- データコピーを含む **reorg** コマンド
- データコピーを含む **alter table** コマンド
- **create index**

動的スレッド割り当てを使用して、次を行うことでパフォーマンスが向上します。

- より少ないスレッドによるクエリプランの並列実行 (静的スレッド割り当てでは逐次クエリ実行が必要)。
- ワーカスレッド間での動的負荷分散の実行。 作業単位よりも少ない数のスレッドがクエリプランを実行している場合、最も小さい作業単位を実行するスレッドはより迅速にタスクを完了し、より大きな作業単位を実行しているスレッドがタスクを実施している間に残りの作業単位を実行しようとします。
- ジョイン内の既存のセマンティック分割をより有効に使用。 動的スレッド割り当てでは、ジョインカラムですでに分割された 2 つのテーブルをジョインするときに、パーティション単位でのジョインが可能です。これに対して、静的スレッド割り当てでは、1 つのオペレーションで外部テーブルのすべてのパーティションを内部テーブルのすべてのパーティションにジョインします。 動的スレッド割り当てでは、1 つのワーカスレッドが外部テーブルの最初のパーティションを内部テーブルの最初のパーティションにジョインし、次にクエリプランフラグメントを再実行して 2 番目のパーティションをジョインします (以下同様)。

動的スレッド割り当ては、**number of worker processes** パラメータと **max parallel degree** パラメータを指定して並列処理を行うように SAP ASE を設定すると有効に

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

SORT 演算子のパフォーマンス強化

SAP ASE バージョン 16.0 以降では、SORT 演算子を含む特定の並列クエリでのパフォーマンスが向上しています。

このメリットを享受するには、並列クエリに以下が含まれている必要があります。

- Exchange 演算子の上に SORT 演算子がある
- Exchange 演算子が単一のプロデューサと複数のコンシューマを持つ Replicated Exchange である

クエリがこの条件を満たしていない場合、クエリプロセッサは、16.0 より前のバージョンの SAP ASE 付属の標準 SORT 演算子 (DEFAULT ソートとも呼ばれます) を使用します。

この例は、16.0 より前のバージョンの SAP ASE からのもので、SORT 演算子の変更が含まれていません。

```
|   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join) (VA = 5)
|   |   |   |    | Using Worktable2 for internal storage.
|   |   |   |    |  Key Count: 1
|   |   |   |    |  Key Ordering: ASC
|   |   |   |    |
|   |   |   |    |    |SORT  Operator (VA = 3)
|   |   |   |    |    | Using Worktable1 for internal storage.
|   |   |   |    |    |
|   |   |   |    |    |    |EXCHANGE Operator (VA = 2) (Replicated)
|   |   |   |    |    |    |Executed in parallel by 1 Producer and 8
Consumer processes.
|   |   |   |    |    |    |    |
|   |   |   |    |    |    |    |    |EXCHANGE:EMIT Operator (VA = 1)
|   |   |   |    |    |    |    |    |
|   |   |   |    |    |    |    |    |    |SCAN Operator (VA = 0)
|   |   |   |    |    |    |    |    |    |  FROM TABLE
|   |   |   |    |    |    |    |    |    |  t2
|   |   |   |    |    |    |    |    |    |  Table Scan.
|   |   |   |    |    |    |    |    |    |  Forward Scan.
|   |   |   |    |    |    |    |    |    |  Positioning at start of table.
|   |   |   |   |   |   |   |   |   |  Using I/O Size 16 Kbytes for data
pages.
|   |   |   |   |   |   |   |   |   |  With LRU Buffer Replacement Strategy
for data pages.
```

ただし、SAP ASE バージョン 16.0 以降では、SORT 演算子のビルド部分が Exchange Operator の下に移動し、ソートテーブル読み込み部分が Exchange の上にあり、同じクエリに対してこのクエリ実行プランが提供されます。

```
|   |   |   |   |MERGE JOIN Operator (Join Type: Inner Join) (VA = 6)
|   |   |   |   | Using Worktable3 for internal storage.
|   |   |   |   |  Key Count: 1
|   |   |   |   |  Key Ordering: ASC
|   |   |   |   |
|   |   |   |   |   |SORT (GETSORTED) Operator (VA = 4)
|   |   |   |   |   | Using Worktable2 for internal storage.
|   |   |   |   |   |
|   |   |   |   |   |   |EXCHANGE Operator (VA = 3) (Replicated)
|   |   |   |   |   |   |Executed in parallel by 1 Producer and 8
Consumer processes.
|   |   |   |   |   |   |
|   |   |   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 2)
|   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |SORT (SORTBUILD) Operator (VA = 1)
|   |   |   |   |   |   |   |   | Using Worktable1 for internal storage.
|   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |SCAN Operator (VA = 0)
|   |   |   |   |   |   |   |   |   |   |  FROM TABLE
|   |   |   |   |   |   |   |   |   |   |  t2
|   |   |   |   |   |   |   |   |   |   |  Table Scan.
|   |   |   |   |   |   |   |   |   |   |  Forward Scan.
|   |   |   |   |   |   |   |   |   |   |  Positioning at start of table.
|   |   |   |   |   |   |   |   |   |   |  Using I/O Size 16 Kbytes for data
pages.
|   |   |   |   |   |   |   |   |   |   |  With LRU Buffer Replacement
Strategy for data pages.
```

新しい並列 SORT 演算子のメリットは、以下のとおりです。

- SORT 演算子が 2 つの演算子として SORTBUILD と GETSORTED に分割されました。 SORTBUILD によりビルドされた単一ソートテーブルは、ソート済みローを並列で読み込む GETSORTED の複数のインスタンスによって使用され、I/O を減らします。 DEFAULT SORT 演算子プランは、SortTables (最初の例では 8 つの Consumer processes) の複数のコピーを作成します。
- SORTBUILD 演算子から GETSORTED 演算子の複数インスタンスにメタデータを含む単一ローが送信されます。これにより、ソートテーブルに直接アクセスするための参照が提供され、Exchange 経由で多くのローをコピーして伝搬することを回避できます。

並列 SORT 演算子には、以下の制限があります。

- Repartitioned Exchange をサポートしません。 ただし、Repartitioned Exchange の上で発生する SORT 演算子はクエリ実行プラン内の DEFAULT ソートを使用します。
- 抽象プランは GETSORTED および SORTBUILD の強制実行を明示的にはサポートしませんが、SORT 演算子を Replicated Exchange の上に配置することにより、これらのプランを強制的に実行できます。

- GETSORTED 演算子および SORTBUILD 演算子を含むクエリ実行プランに対しては、動的スレッド割り当て (DTA) が暗黙で無効になります。

ハッシュジョイン演算子のパフォーマンス強化

SAP ASE バージョン 16.0 以降では、HASH JOIN 演算子を含む特定の並列クエリプランのパフォーマンスの向上とリソース使用量の削減を実現しています。

HASH JOIN 操作には以下の 2 つのステップが含まれます。

1. ジョインするローの外部ストリームのジョインカラムを含むハッシュテーブルを構築する
2. ジョインする内部ストリーム内の各ローのジョインカラム値でハッシュテーブルをプローブする

SAP ASE 16.0 では HASH JOIN 演算子は HASH PROBE 演算子と HASH BUILD 演算子に置き換えられ、これらの演算子間に複写 EXCHANGE 演算子が挿入されました。 HASH BUILD 演算子は、ハッシュテーブルを構築し、HASH PROBE 演算子は、内部ストリームを読み込んで、ハッシュテーブルをプローブして一致するローを検索します (以前のリリースの SAP ASE では HASH JOIN 演算子がこの両方のステップを実行していました)。

単一のワーカスレッドが HASH BUILD 演算子を実行し、1 つのハッシュテーブルを構築します。 クエリエンジンがこのハッシュテーブルをメモリパイプ経由で HASH PROBE 演算子を実行しているすべてのプロデューサに渡します。 これらのプロデューサは、ハッシュテーブルを共有し、内部ストリーム内のジョインカラムとの一致をプローブします。 以前のリリースの SAP ASE では、HASH JOIN 演算子を実行するのに複数のプロデューサが必要でした。

これは、HASH JOIN 演算子を使用したクエリプランの **showplan** 出力例です。

```
|
|   |EXCHANGE Operator (VA = 6) (Merged)
|   |Executed in parallel by 4 Producer and 1 Consumer processes.

|   |
|   |   |EXCHANGE:EMIT Operator (VA = 5)
|   |   |
|   |   |   |HASH JOIN Operator (VA = 4) (Join Type: Inner Join)
|   |   |   | Using Worktable1 for internal storage.
|   |   |   |  Key Count: 1
|   |   |   |
|   |   |   |   |EXCHANGE Operator (VA = 2) (Replicated)
|   |   |   |   |Executed in parallel by 1 Producer and 4 Consumer
processes.

|   |   |   |   |
```

```
|   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 1)
|   |   |   |   |   |
```

HASH JOIN の拡張機能を使用するには、クエリプランにクエリプロセッサの次の属性が含まれている必要があります。

- クリエプロセッサが HASH JOIN 演算子 (VA = 4) を並列で実行する (上記の例では EXCHANGE Operator (VA = 6) に 4 つの Producers が含まれています)。
- HASH JOIN Operator の残りの子 Operator が 1 つの Producer を持つ Replicated EXCHANGE Operator (VA = 2)。

クリエがこの条件を満たさない場合、クエリプロセッサは単一の HASH JOIN 演算子を使用します。

SAP ASE 16.0 クエリプランのメリットは以下のとおりです。

- HASH JOIN 演算子を実行するプロデューサにメモリパイプ経由ですべてのローをコピーすることによるパフォーマンス低下を回避できます。 代わりに、クエリエンジンは、HASH PROBE 演算子により構築されたハッシュテーブルの参照を含む 1 つのローをパイプ経由で渡します。
- 各プロデューサが独自のハッシュテーブルを構築するために余分なメモリやディスクリソースが必要ありません。 代わりに、1 つのハッシュテーブルを共有します。

これは、改善されたハッシュジョイン処理を使用したクエリプランの showplan 出力例です。

```
|
|   |EXCHANGE Operator (VA = 7) (Merged)
|   |Executed in parallel by 4 Producer and 1 Consumer processes.

|   |
|   |   |EXCHANGE:EMIT Operator (VA = 6)
|   |   |
|   |   |   |HASH JOIN PROBE Operator (VA = 5) (Join Type: Inner Join)
|   |   |   | Using Worktable2 for internal storage.
|   |   |   |  Key Count: 1
|   |   |   |
|   |   |   |   |EXCHANGE Operator (VA = 3) (Replicated)
|   |   |   |   |Executed in parallel by 1 Producer and 4 Consumer
processes.

|   |   |   |   |
|   |   |   |   |   |EXCHANGE:EMIT Operator (VA = 2)
|   |   |   |   |   |
|   |   |   |   |   |   |HASH JOIN BUILD Operator (VA = 1) (Join Type:
Inner Join)
|   |   |   |   |   |   | Using Worktable1 for internal storage.
```

```
|   |   |   |   |   |   |  Key Count: 1
|   |   |   |   |   |   |
```

全文監査

SAP ASE バージョン 16.0 以降は、全文 DML 監査をサポートしています。DML 監査オプション (*table_access* と *view_access* を含む) を有効にすると、機密パラメータはマスクされた状態でパラメータ名とその値が出力されます。

全文監査については、『セキュリティ管理ガイド』>「監査」>「監査システムの管理」>「監査システムの設定」>「グローバル監査オプションの設定」>「DML 文の監査」を参照してください。

ストアドプロシージャ内の権限チェックの監査

監査オプション sproc_auth を指定すると、システムストアドプロシージャ内で実行される権限チェックの監査が有効になります。

細密なパーミッション	イベント
有効	146
無効	80

監査イベント 80 は、監査オプション security が有効になっている場合か、監査オプション sproc_auth が有効になっている場合に監査されます。監査イベント 146 は、オプション sproc_auth が有効になっている場合に監査されます。

オブジェクト定義の置換

元の名前、オブジェクト ID、監査オプションやパーミッションなどのセキュリティ属性、複写属性を維持して、コンパイルされた既存オブジェクトの定義を新しい定義に置き換えます。

create or replace 機能は、既存のものがなければ新しいオブジェクトを作成し、既存であれば同じ名前でオブジェクトを置き換えます。 **or replace** 句は、既存オブジェクトを暗黙的に削除し、同じ名前とタイプでデータベース内に再作成します。オブジェクトの定義は変更されますが、既存のセキュリティおよび複写属性は維持されます。 コンパイルされたオブジェクトのテキストが置換前に非表示であった場合は、置換後も非表示のままになります。 この機能に新しいキーワードは導入されていません。

or replace 句が追加されたコマンドは次のとおりです。

- **create default**
- **create function**
- **create function (SQLJ)**
- **create procedure**
- **create procedure (SQLJ)**
- **create rule**
- **create trigger**
- **create view**

or replace 機能は、データを格納しないオブジェクトでのみサポートされます。

置き換えの際にオブジェクトが使用中であると、エラー 3702 が発生します。

```
"Cannot drop or replace the %S_MSG '%.*s' because it is currently in
use."
```

オブジェクトが置き換えられると、SAP ASE はシステムテーブル `sysprocedures`、`syscomments`、`sysdepends`、`syscolumns` で定義の置き換えを行います。 `sysobjects` テーブルの一部のフィールドも更新されます。 オブジェクトのクエリツリーは、`sysprocedures` で置き換えられる前に正規化されます。

置き換えられたオブジェクトは、他のオブジェクト定義で使用することができます。 SAP ASE は、置き換えられたオブジェクトを使用時に再コンパイルしますが、置き換え後のオブジェクトのインタフェースが呼び出し元オブジェクトで使用されているものと一致しない場合、状況によっては呼び出し元オブジェクトの置き換えが必要になります。 置き換えられるオブジェクトに対して **sp_depends** を

実行すると、呼び出し元オブジェクトが存在するかを確認して置き換えることができます。詳細については、「create procedure」、「create view」、および「create function」で、置き換えられたオブジェクトに依存するオブジェクトに関する説明を参照してください。

細密なパーミッションが有効であっても無効であっても、コンパイルされたオブジェクトを置き換えるにはオブジェクトの所有者であることが必要です。 エイリアスまたは **setuser** を使用してオブジェクト所有者と同一化しても、コンパイルされたオブジェクトを置き換えることはできません。 ただし、**set proxy** を使用した所有者であれば、コンパイルされたオブジェクトを置き換えることができます。

注意： **create or replace** 機能は、1 つのトランザクション内で暗黙的に **drop** を実行してから **create** を実行します。 このため、追加のトランザクションログ領域が必要です。 オブジェクトを削除してから作成する代わりに、**create or replace** を使用してからオブジェクトを作成する場合は、トランザクションログのサイズの拡大が必要な場合があります。

参照：

- create default (128 ページ)
- create function (133 ページ)
- create function (SQLJ) (136 ページ)
- create procedure (139 ページ)
- create procedure (SQLJ) (143 ページ)
- create rule (145 ページ)
- create trigger for or replace (152 ページ)
- create view (155 ページ)

インストールスクリプトの変更

インストールスクリプトで使用されるストアドプロシージャは、**create or replace** を使用するように変更されました。

SAP ASE 16.0 より前のバージョンでは、インストールスクリプトで使用されるストアドプロシージャは削除されて再作成されていました。 SAP ASE 16.0 以降、以下のインストールスクリプトで使用されるストアドプロシージャは、**or replace** 機能を使用します。

installmaster	installmodel	installmsgsvss	installpcidb
installjconnect	installcommit	installdbccalt	installdbccdb
installupgrade	installjsb	installsecurity	installoledb

installodbc	installhasvss	installpubs2	installpubs3

データセグメントおよびログセグメントの変更

各種データセグメントデータベースのデータセグメントおよびログセグメントのサイズの変更が実施されました。

create or replace の場合、デバイスには、追加領域要件を提供できるだけの容量がある必要があります。

デフォルトのデータベースサイズは次のように変更されました。

データベース	ページサイズ	16.0 より前のデフォルトサイズ	16. 0 以降のデフォルトサイズ
master	2KB	13MB	18MB
	4KB	26MB	26MB
	8KB	52MB	52MB
	16KB	104MB	104MB
sybpcidb	2KB	24MB	48MB
	4KB	48MB	96MB
	8KB	96MB	192MB
	16KB	192MB	384MB

データベース	16.0 より前のデフォルトサイズ	16. 0 以降のデフォルトサイズ
sybsystemprocs	172MB	196MB
sybmgmtdb	75MB	76MB
dbccdb	5MB	33MB
dbcalt	5MB	33MB

SAP ASE サーバを設定するためのデフォルト値と最小要件の詳細については、『インストールガイド』>「SAP ASE のインストール」>「最低限の SAP ASE サーバの設定」を参照してください。

HTML 形式のクエリプランと実行統計

新しい動的スレッド割り当て (DTA) モデルでは、ワーカスレッドの使用が拡張されています。

バージョン 15.7 SP100 で、Web ブラウザでデータを表示するために、静的スレッド割り当て (STA) ワーカスレッドモデルを使用して HTML 形式のグラフィカルなクエリプランを生成する機能が導入されました。 STA を使用する場合、プランフラグメントとワーカスレッドの間には直接的な関係があります。クエリを実行するために割り当てられたワーカスレッドと同じ数のプランフラグメントが生成されます。 ワーカスレッドへのプランフラグメントの割り当ては、静的かつユニークです。ワーカスレッドが実行できるのは、割り当てられたプランフラグメントだけです。 このモデルを使用すると、HTML 表現には各スレッドとそのスレッドの単一のプランフラグメント実行の詳細が出力されます。

バージョン 16.0 で、DTA ワーカスレッドモデルが実装されました。 DTA の導入によって、"HTML 形式のクエリプランと実行統計" 機能は、並列実行モデルを正しく反映するように更新されました。 DTA ワーカスレッドモデルの詳細については、「クエリのパフォーマンス強化」を参照してください。

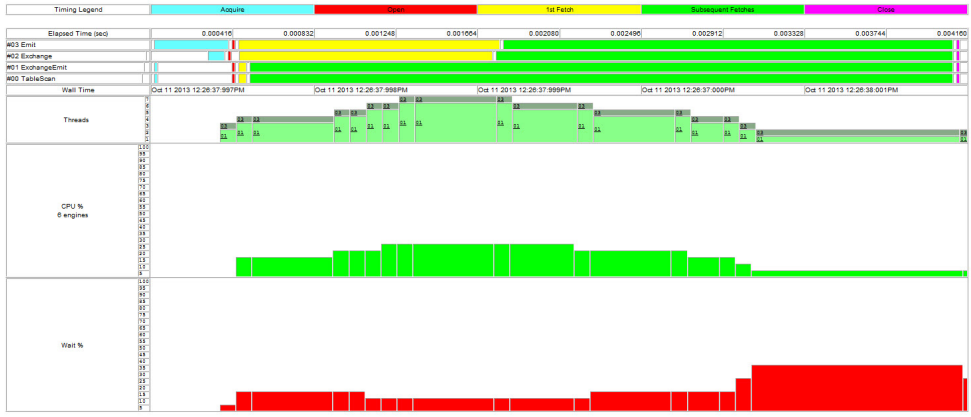
HTML 表現では、プランフラグメント実行 (または作業単位) ごとの実行統計をレポートするため、同じプランフラグメントの複数の実行を別々にレポートすることができます。 新しい DTA モデルに従って作業単位実行をより詳細に識別するために、プランフラグメント実行ごとに、出力が "Work unit execution" で示され、スレッドの SPID と、(プランフラグメントの) ルート演算子識別子やプロデューサ ID などの追加の値を提供します。

次に、複数のワーカスレッドを使用して並列で実行されたクエリの例を示します。

Adaptive Server Enterprise ® Query Plan
Query: [spid: 45]
Version: Adaptive Server Enterprise

Query Tree

Query Timings



Query Text

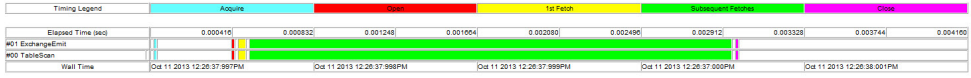
```
select * from RA2 (parallel 6)
```

Threads

SPID	Thread ID	Rows produced
34	2686997	168
35	2656924	168
36	2424851	168
37	2293778	166
38	2162705	162
39	2031632	168

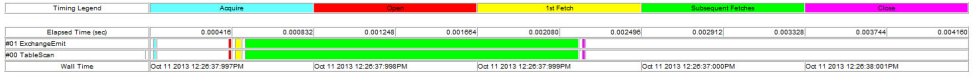
Work unit execution:
Thread SPID: 34 Root operator: 1 Producer ID: 0

Query Timings



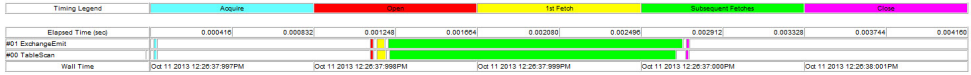
Work unit execution:
Thread SPID: 35 Root operator: 1 Producer ID: 1

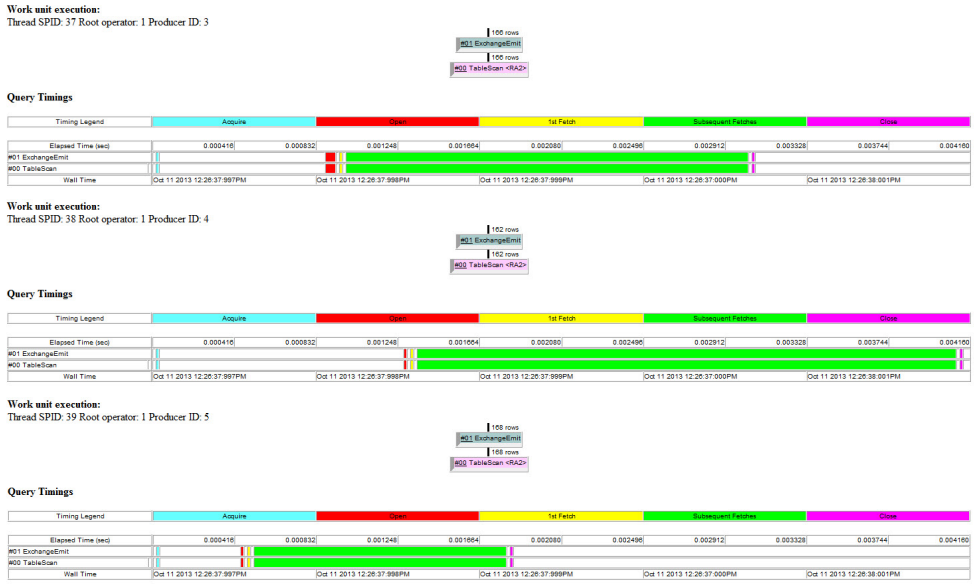
Query Timings



Work unit execution:
Thread SPID: 36 Root operator: 1 Producer ID: 2

Query Timings





参照：

- クエリのパフォーマンス強化 (39 ページ)

生成されたファイルのプレフィクスのオプション

SET STATISTICS QUERY_NAME_HTML コマンドは、同じクエリの複数の実行に関連付けられたファイルを区別または識別する際に役立ちます。

クエリ名を指定すると、HTML 出力の書き込み先となる内部的に生成されたファイルのプレフィクスとしてそのクエリ名が使用されます。

```
SET STATISTICS QUERY_NAME_HTML [queryname | ON | OFF]
```

インデックス圧縮

リレーショナルデータベースのインデックス圧縮により、データのより効率的な格納、メモリ消費量の削減、I/O 要求の減少によるパフォーマンス向上が実現します。

インデックス圧縮は以下をサポートします。

- インデックスリーフページ圧縮
- DOL および APL インデックスリーフページ形式
- テーブル、インデックス、およびローカルインデックスパーティションレベルの圧縮

インデックス圧縮は **sp_configure** および **set compression** を使用して、サーバレベル、またはセッションレベルで有効化/無効化できます。

テーブル、インデックス、またはローカルインデックスパーティションレベルでインデックス圧縮を指定するには、次のコマンドを使用します。

- **create table**
- **alter table**
- **create index**
- **alter index**
- **select into**

インデックスレベルの圧縮を指定すると、テーブルレベルで指定されたインデックス圧縮が上書きされます。 ローカルインデックスパーティションレベルで指定すると、インデックスレベルの指定が上書きされます。

サポート

- **reorg rebuild** は、インデックス圧縮をサポートするように拡張されています。
- DML は、インデックス圧縮をサポートするように拡張されています。
- インデックス圧縮はトリガによってサポートされます。
- インデックスの圧縮が定義されたテーブルにバルクコピーされたデータは、自動的に圧縮されます。
- **dbcc checktable** と **dbcc checkstorage** はいずれも、インデックスの整合性をチェックできます。 これらのコマンドは、特定のテーブル内のすべてのページをスキャンすることによって、各ページの整合性をチェックするように拡張されています。 **dbcc checktable** のほうがチェック対象の情報が多いため、チェック機能が優れています。
- 当該データベースに圧縮インデックスのあるテーブルが含まれる場合は、**dump database**、**dump transaction**、および **load transaction** を使用することがで

きます。 ただし、圧縮インデックスのあるテーブルが存在する場合、XPDL は使用できません。

テーブルの作成時にすべてのインデックスに圧縮が指定されている場合でも、複写インデックスは常に圧縮なしで作成されます。

インデックス圧縮テーブルでは APL クラスタードインデックスのサポートはありません。

1 カラムのみのユニークインデックスは圧縮されません。

インデックス圧縮の有効化

サーバレベルまたはデータベースレベルでインデックス圧縮を有効化します。

サーバレベルでのインデックス圧縮の有効化
特定のサーバ上のすべてのデータベースのすべてのインデックスの圧縮を有効にするには、サーバレベルでインデックス圧縮を設定します。

構文は次のとおりです。

```
sp_configure "enable compression", 0 | 1
```

デフォルト値は 0 です。

インデックス圧縮が有効化されていない場合にインデックス圧縮テーブルまたはインデックスを作成しようとすると、エラーが発生します。

セッションレベルでのインデックス圧縮の有効化
特定のセッションのすべてのデータベースのすべてのインデックスの圧縮を有効にするには、データベースレベルでインデックス圧縮を設定します。

構文は次のとおりです。

```
set {compression
   [= {default | ON | OFF} ]
   |index_compression
   [= {default | ON | OFF} ] }
```

デフォルト値は off です。 このコマンドは、コマンドの実行後に圧縮インデックスに構築されるリーフローのみに作用します。

- **set index_compression** を off に設定すると、圧縮インデックスに新たに挿入されるローはすべてインデックス圧縮が行われません。
- **set index_compression** を on に設定すると、圧縮インデックスに新たに挿入されるローのすべてでインデックス圧縮が行われます。

インデックス圧縮テーブルの作成

テーブルに対してインデックス圧縮を指定するには、**create table** コマンドまたは **select into** コマンドを使用します。

システムカタログとワークテーブルを除き、テンポラリテーブルを含むすべてのテーブルまたはインデックスでインデックス圧縮を指定できます。

インデックスレベルの圧縮を指定すると、テーブルレベルで指定されたインデックス圧縮が上書きされます。 ローカルインデックスパーティションレベルで指定すると、インデックスレベルの指定が上書きされます。

***create table** コマンドの使用方法*
with index_compression 句で指定される圧縮オプションは次のとおりです。

- NONE - 指定したテーブルのインデックスは圧縮されません。
  index_compression = PAGE を指定して作成されたインデックスは圧縮されます。
- PAGE - 指定したテーブルのすべてのインデックスが圧縮されます。
  index_compression = NONE を指定して作成されたインデックスは圧縮されません。

テーブル DDL で圧縮がまったく指定されていない場合、そのテーブルのインデックスは圧縮されません。

***select into** コマンドの使用方法*
既存テーブルを選択してインデックス圧縮テーブルを作成するには、**select into** を使用します。 **with index_compression** 句の構文は、**create table** コマンドの場合と同じです。

参照：
- インデックス圧縮用の create table (147 ページ)
- select into (163 ページ)

圧縮インデックスの作成

インデックスまたはローカルインデックスパーティションレベルでインデックス圧縮を指定するには、**index_compression** 句を使用します。

リーフページのみが圧縮されます。 単一のインデックスリーフページで圧縮と非圧縮のインデックスローが共存する場合があります。 テーブルまたはインデックス DDL で圧縮がまったく指定されていない場合、インデックスは圧縮されませ

ん。 インデックス圧縮では、APL クラスタードインデックスがサポートされません。 1 カラムしかないユニークインデックスは圧縮されません。

インデックスレベルの圧縮を指定すると、テーブルレベルで指定されたインデックス圧縮が上書きされます。 ローカルインデックスパーティションレベルで指定すると、インデックスレベルの指定が上書きされます。

***create index** コマンドの使用方法*

with index_compression 句で指定される圧縮オプションは次のとおりです。

- NONE - 指定したインデックスのインデックスページは圧縮されません。 **index_compression = PAGE** を指定して作成されたローカルインデックスパーティションは圧縮されます。
- PAGE - ページがいっぱいになると、ページプレフィクス圧縮を使用して既存のインデックスローが圧縮されます。 ローが追加されると、チェックが実行され、そのローが圧縮に適しているかどうかが判別されます。

参照：

- create index (138 ページ)

圧縮状態の変更

今後のインデックスの挿入または更新におけるテーブルの圧縮状態を変更するには、**alter table** または **alter index** を使用します。

圧縮されているかどうかにかかわりなく、既存のインデックスページには作用しません。 テーブルの圧縮状態を変更するには、テーブルに対する排他的アクセスが必要です。

ローカルインデックスパーティションの圧縮状態の変更は、当該のパーティションで新しく挿入または更新されたインデックスローにのみ作用します。

新たに作成したインデックスのデフォルトの動作は、テーブルの圧縮設定によって異なります。

- インデックス圧縮テーブルの場合、新たに作成されたインデックスに対してインデックス圧縮が設定されます。
- インデックスを圧縮しないテーブルの場合、新たに作成されたインデックスも圧縮されないままになります。

***alter table** コマンドの使用方法*

alter table コマンドでは、スキーマ変更とプロパティ変更をさまざまに組み合わせることができます。 カタログ更新のみが必要なコマンドもあれば、既存インデックスの再構築を伴うデータの移動が必要なものもあります。 インデックスの再構

築が必要で、インデックス圧縮がオンに設定されている場合、再構築の一環としてインデックスページが圧縮されます。 インデックスの再構築後、生成されたインデックスには、インデックス圧縮の状態に応じて圧縮または非圧縮のインデックスローが格納されます。

set index_compression 句は、テーブル、インデックス、ローカルインデックスパーティションのインデックス圧縮の有効または無効を指定します。 インデックスが圧縮されるようにテーブルを変更すると、新たに作成されるインデックスは圧縮されます。

modify partition *partition_name* 句は、後続の set compression 句の指定に従って圧縮状態が変更されるローカルインデックスパーティションを指定します。

alter index *コマンドの使用方法*

alter index のパーミッションはデフォルトでインデックス所有者にあり、**setuser** コマンドを実行してインデックス所有者と同一化できるデータベース所有者以外にこのパーミッションを譲渡することはできません。 システム管理者もユーザインデックスを変更できます。

制限事項

インデックス圧縮では、APL クラスタードインデックスがサポートされません。

1 カラムしかないユニークインデックスは圧縮されません。

インデックスからの圧縮の削除

インデックスから圧縮を削除するには、インデックスを削除してから、**set index_compression off** を指定して再作成します。

参照：

- インデックス圧縮用の alter table (123 ページ)
- alter index (122 ページ)

SAP JVM サポート

SAP ASE は SAP JRE を使用して Java アプリケーションをサポートします。

SAP JRE は、デフォルトで次の場所にインストールされます。

```
$SYBASE/shared/SAPJRE-7_*
```

インストーラによって、*SAP_JRE7*、*SAP_JRE7_32*、および *SAP_JRE7_64* の各環境変数が自動的に設定されます。

注意： (IBM AIX の場合のみ) Java アプリケーションを使用する場合は、次のようにデータサイズリソース制限を `unlimited` に設定する必要があります。

```
limit datasize unlimited
```

使用しているオペレーティングシステムのマニュアルを参照してください。

データベースの完全暗号化

SAP ASE バージョン 16.0 では、データベースを完全に暗号化する機能が導入されています。

以前のバージョンの SAP ASE では、カラムを暗号化することができました。 バージョン 16.0 では、データベース全体を完全に暗号化して、データベース全体を保護することができます。 データベースを完全に暗号化すると、データ、インデックス、トランザクションログのすべてが暗号化されます。 この暗号化は透過的であるため、ユーザはその違いに気づくことなく、テーブル、インデックスなどの操作を実行できます。

SAP ASE ではページレベルでデータが暗号化されます。 データベースの暗号化を設定すると、暗号化と復号化のプロセスが自動的に行われます。 データはページがディスクに書き込まれる直前に暗号化され、データページがメモリにロードされるとすぐに復号化されます。

データベースの完全暗号化とカラムの暗号化

ニーズに応じてデータベース全体またはカラムのみを暗号化します。

SAP ASE の認証とアクセス制御のメカニズムでは、正しく識別され、権限を持つユーザのみがデータにアクセスできることが保証されます。 データを暗号化すると、機密データの盗難やセキュリティ侵害からの保護を強化できます。

暗号化カラムと完全暗号化データベースは双方ともセキュリティおよびプライバシーの要件に対応できますが、用途によっては、どちらかの機能を展開するのが便利な場合があります。 いずれを使用するかを決定するには、以下の事項を検討してください。

- 機密データが格納されているカラムを容易に識別できる場合は暗号化カラム。
- 機密データカラムに対する範囲検索の実行が必要な場合、およびデータモデルに関する知識が十分でなく機密データカラムを識別できない場合 (何千ものテーブルを含むパッケージ化アプリケーションの場合など) は暗号化データベース。 さらに、機密データ (個人データなど) の定義がロケーション間 (州や国など) で異なる場合は、データベース全体の暗号化によって、多様なデータセキュリティ要件を満たすことが可能になります。

データベース暗号化キーの作成

データベース暗号化キーは 256 ビットの対称キーで、master データベースで作成され、データベースの暗号化に使用されます。

前提条件

データベース暗号化キー (DEK) 作成の前提として、以下を実行します。

- 有効な SAP ASE 暗号化機能ライセンス (ASE_ENCRYPTION) を保持していることを確認します。
- **enable encrypted columns** 設定パラメータを設定します。
- マスタキー、さらに必要に応じてデュアルマスタキーを `master` データベースに作成します。これらによってデータベース暗号化キーが保護されます。『暗号化カラムユーザーズガイド』の「データベースレベルマスタキーとデュアルマスタキーの使用」を参照してください。
- 適切な権限があることを確認します。
  - 細密なパーミッションが有効化されている場合、キーの作成にはシステムパーミッション `manage database encryption key` が必要です。
  - 細密なパーミッションが無効化されている場合、sso_role、keycustodian_role、または **create encryption key** パーミッションを保持している必要があります。

手順

`master` データベースで **create encryption key** コマンドを 使用して、データベース暗号化キーを作成します。 構文は次のとおりです。

```
create encryption key keyname
    [for algorithm]
    for database encryption
    [with
        {[master key]
        [key_length 256]
        [init_vector random]
        [[no] dual_control]}
```

構文の説明は次のとおりです。

- *keyname* - `master` データベース内のユーザのテーブル、ビュー、プロシージャ ネームスペースでユニークにする必要があります。
- `for` *algorithm* - アルゴリズムを指定します。 現在、サポートされている唯一のアルゴリズムは Advanced Encryption Standard (AES) です。

- `for database encryption` - カラムではなくデータベース全体を暗号化する暗号化キーの作成であることを明確に指定します。
- `master key` - データベースの完全暗号化に必須です。 マスタキーが存在していない場合は SAP ASE からエラーが返されます。
- `key_length 256` - 作成するキーのビット単位のサイズです。 データベース暗号化キーの有効長は 256 のみです。他のサイズを使用すると SAP ASE からエラーメッセージが返されます。
- `init_vector random` - データベースの完全暗号化に必須です。 カラムの暗号化キーの作成時に使用可能な `init_vector null` を指定すると、SAP ASE からエラーが返されます。
- `[no] dual control` - デュアルコントロールを使用してデータベース暗号化キーを暗号化する必要があるかどうかを指定します。 デフォルトでは、デュアルコントロールは設定されていません。

次の例は、マスタキーで保護されたデータベース暗号化キーを作成します。

```
sp_configure 'enable encrypted columns', 1
create encryption key master with passwd "testpassword"
set encryption passwd 'testpassword' for key master
create encryption key dbkey for database encryption
```

参照：

- データベース暗号化キーの変更 (67 ページ)
- データベース暗号化キーの削除 (68 ページ)
- データベース暗号化キーのバックアップ (79 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

データベース暗号化キーの変更

データベース暗号化キーの保護方法、および所有者を変更するには、**alter encryption key** コマンドを使用します。

特定のデータベースについて、データベース暗号化キーを再生成することはできません。

- データベース暗号化キーを変更する方法は次のとおりです。
  1. データベース暗号化キーで保護されているデータベースを復号化します。
  2. データベース暗号化キーを削除して再作成します。

  注意： カラム暗号化キーをデータベース暗号化キーに変換することはできません。 **for database encryption** オプションを使用して別のタイプの暗号化キーを

データベース暗号化キーに変更すると、SAP ASE ではエラーメッセージが表示されます。

- データベース暗号化キーを完全に変更するのではなく、単にデータベース暗号化キーの保護方法を変更する場合は、次の構文を使用します。

```
alter encryption key key_name
for database encryption
modify encryption with {[master key]
    [[no] dual_control}
```

- データベース暗号化キーの所有者を変更するには、次を使用します。

```
alter encryption key [[database.][owner].]dek_name
    modify owner user_name
```

このオプションを実行するパーミッションは、**alter encryption key** のパーミッションと同じです。

参照：

- データベース暗号化キーの削除 (68 ページ)
- データベース暗号化キーのバックアップ (79 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)
- データベース暗号化キーの作成 (66 ページ)

データベース暗号化キーの削除

データベース暗号化キーを削除するには、**drop encryption key** コマンドを使用します。

構文は次のとおりです。

```
drop encryption key key_name
```

このコマンドによって、`master` データベースの `sysencryptkeys` テーブルにあるデータベース暗号化キーが削除されます。

注意： 削除するデータベース暗号化キーが、データベースの暗号化に使用されている場合、このコマンドは失敗します。

参照：

- データベース暗号化キーの変更 (67 ページ)
- データベース暗号化キーのバックアップ (79 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)
- データベース暗号化キーの作成 (66 ページ)

暗号化データベースの作成

完全暗号化データベースを作成するには、**create database** コマンドを使用します。

作成時にデータベースを暗号化するか、また、このデータベースに挿入されるデータを自動的に暗号化するかを指定します。 暗号化してもデータベースのサイズは変わりません。また、記憶領域アクセス機能はデータベースが暗号化されていてもいなくても同様に機能します。 暗号化をサポートするデータベースのタイプは次のとおりです。

- 通常のユーザデータベース
- テンポラリデータベース
- アーカイブデータベース

インメモリデータベースを暗号化することはできません。

暗号化データベースを作成するには、以下を使用します。

```
create [temporary] database database_name
    encrypt with key_name
```

構文の説明は次のとおりです。

- *database_name* は、作成する暗号化データベースの名前です。
- *key_name* は、データベース暗号化キーの名前です。

暗号化アーカイブデータベースを作成するには、以下を使用します。

```
create archive database database_name
    encrypt with key_name
```

構文の説明は次のとおりです。

- *database_name* は、作成するアーカイブデータベースの名前です。
- *key_name* は、バックアップしたデータベースの暗号化に使用したキーです。 SAP ASE は、データベースのロード時に *key_name* が一致していることを検証します。 一致しない場合は、リストアが失敗します。

例

`demodb` という名前の暗号化データベースをデバイス `demodev` 上に作成し、`dbkey` という暗号化キーを使用してデバイス `demologdev` にログオンします。

```
create database demodb on demodev log on demologdev encrypt with
dbkey
```

使用法
create database コマンドの `encrypt with` オプションの使用に必要なパーミッションは特にありません。 ただし、*key_name* としての参照を可能にするには、ユーザにデータベース暗号化キーに対する **select** パーミッションが必要です。

参照：
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)

既存データベースの暗号化

SAP ASE バージョン 16.0 では、**alter database** コマンドを使用して非暗号化データベースを暗号化することができます。

データベースのサイズによっては、暗号化にしばらく時間がかかる場合があります。 このため、このコマンドでは、データベースが暗号化対象としてマークされたことがすぐに返されます。 暗号化はバックグラウンドで行われ、この処理はユーザに対して透過的です。 データベース暗号化のステータスおよび進行状況をチェックするには、**sp_helpdb** システムプロシージャ、**dbencryption_status()** 組み込み関数、または SAP Control Center ユーザインタフェースを実行します。 このとき、次の点に留意してください。

- データベースの暗号化は、データベースがオンラインの状態で行われます。 このため、暗号化の進行中も他のユーザからデータベースへのアクセスが可能で、シングルユーザモードを設定する必要はありません。
- 暗号化処理によってデータベースに対するユーザクエリ、更新、挿入の操作が中断されることはありません。
- データベースの暗号化はサスペンドと再開が可能であるため、SAP ASE の再起動後にデータベースの暗号化を再開することができます。
- 暗号化操作はページごとに実行されます。
- アーカイブデータベースの暗号化と復号化は変更できません。
- SAP ASE によってデータベース暗号化の進行状況が記録され、そのステータスをレポートするユーティリティが提供されます。

制限事項:

- `master`、`model`、`dbccdb`、および `dbccalt` のデータベースを暗号化することはできません。

- 暗号化処理が進行中のデータベースの復号化、または復号化処理が進行中のデータベースの暗号化はできません。
- 暗号化処理が進行中のデータベースをマウント解除することはできません。
- 暗号化が進行中のデータベース上に別のデータベースをロードすることはできません。
- データベースの暗号化の進行中は、データベースサイズを縮小するコマンドを実行しないでください。

構文は次のとおりです。

```
alter database database_name
{encrypt with key_name [parallel degree_of_parallelism]
| resume encryption [parallel degree_of_parallelism]
| suspend encryption
}
```

構文の説明は次のとおりです。

- *key_name* を指定して暗号化すると、SAP ASE に対して *key_name* を使用してデータベースを暗号化するように指示します。
  具体的に言うと、このコマンドは `master` データベースの `sysencryptkeys` システムテーブルから対応するキー ID を取得し、関連付けられた `sysdatabases` ローの `encrkeyid` カラムを設定します。
  すでにデータベースが以下の状態にある場合、SAP ASE は **alter database** の実行に失敗し、エラーメッセージを表示します。
  - 別のキーによって暗号化されている。
  - 暗号化中である。

  暗号化が現在進行中ではなく、一部が暗号化されているデータベースに対してこのコマンドを実行した場合、キー名が以前指定したキーと同じであれば、`resume encryption` オプションを指定した場合と同様にコマンドが処理されます。
- `parallel` *degree_of_parallelism* は、タスクを開始するワーカスレッド数を決定します。
  数が "`number of worker processes`" の設定以下である場合は、データベース記憶領域仮想デバイスのそれぞれにスレッドを作成します。 ワーカスレッド数が追加されても暗号化パフォーマンスは向上しないため、*degree_of_parallelism* の数値をデータベースデバイスの数より大きくしないでください。 *degree_of_parallelism* を指定しない場合、オンラインエンジン数、および各デバイス間におけるデータベースの分散状況に応じて、SAP ASE によってこの値が内部的に定義されます。
- `resume encryption` は、暗号化が前回サスペンドされたページから暗号化処理を再開します。
  次の場合はコマンドが失敗します。

  - SAP ASE ですでに暗号化処理が実行中である。
  - 対象データベースで暗号化が開始されたことがない。
  - 暗号化処理がすでに完了している。

  `parallel degree_of_parallelism` は、`resume encrypt` とともに使用することができます。
- `suspend encryption` は、データを暗号化中のすべての暗号化ワーカスレッドを強制終了します。 SAP ASE は暗号化の進行状況を記録して、`resume encryption` が前回の暗号化処理の停止位置から暗号化を再開できるようにします。 進行中の暗号化が存在しない場合、SAP ASE はこのコマンドを無視します。

次の例では、`dbkey` という暗号化キーを使用して、`existdb` という名前の既存データベースを暗号化するように変更します。

```
alter database existdb encrypt with dbkey
```

この例では、並列度を指定していないため、SAP ASE によって `existdb` の暗号化を並列的に開始するワーカスレッド数が決定されます。

並列度以外にデータベース暗号化のパフォーマンスに影響する主要要因としてバッファプールのサイズがあります。 SAP ASE でディスク読み込みのたびにページの大きなまとまりをロードし、暗号化を実行して、書き戻すことを可能にするには、バッファキャッシュが十分で、バッファプールのサイズが適切であることが必要です。

次の例は、暗号化される `demodb` というデータベースのバッファキャッシュとバッファプールサイズの両方を設定する際に使用できる手順を示しています。

1. `demodb` に固有のデータキャッシュを作成します。

   ```
   sp_cacheconfig demodb_cache, '10M'
   ```

   これによって、データベースページ領域 10MB で `demodb_cache` という名前のバッファキャッシュが作成されます。
2. サイズ指定のバッファプールを作成します。 バッファプールサイズは、データベースのページサイズの 8 倍にする必要があります。 たとえば、デフォルトでデータベースページサイズが 2K の場合、バッファプールサイズは 8 x 2 = 16K にする必要があります。

   ```
   sp_poolconfig demodb_cache, '10M' , '16k'
   ```

   上の例では、`demodb_cache` という指定キャッシュにサイズ 16K のバッファの 10MB のバッファプールが作成されます。
3. データベースをバッファキャッシュにバインドします。

   ```
   sp_bindcache demodb_cache, demo_db
   ```

参照：

- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- 暗号化処理のサスペンド (77 ページ)
- 暗号化処理の再開 (77 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- 暗号化されたデータベースの復号化 (78 ページ)

暗号化のステータスと進行状況

データベースの暗号化ステータスを確認するには 2 とおりの方法があります。

データベースが暗号化されているかどうか、および暗号化中のデータベースの暗号化処理がどの程度進んでいるかの情報を取得するには、以下を使用します。

- **sp_helpdb** システムプロシージャ。 構文は次のとおりです。*database_name* はデータベースの名前です。
  ```
  sp_helpdb database_name
  ```
- **dbencryption_status** 組み込み関数。 ステータスはデータベースが暗号化されているかどうか、進行状況は暗号化プロセスがどの程度進んでいるかの確認に使用します。
  - `select dbencryption_status("status", db_id("existdb"))`
  - `select dbencryption_status("progress", db_id("existdb"))`

パフォーマンスの考慮事項

既存のデータベースの暗号化時は、オンラインとして保持されます。 データベースへのユーザアクセス、および一般的な SAP ASE の応答時間に対する影響を緩和するため、パフォーマンス上の問題を考慮してください。

データベース暗号化のパフォーマンスを良好に保持するために考慮が必要な要素は次のとおりです。

- マルチプロセッサマシン上の SAP ASE エンジンの数
- データベースの格納に使用されるディスク数
- データベースに関連付けられているバッファプールサイズ

暗号化または復号化の際に **alter database** で並列度の値を指定すると、実質的には SAP ASE に対して操作の実行時に開始するワーカスレッド数を指定することになります。 ワーカスレッドは同時に実行されるため、複数の CPU 間に分散することをおすすめします。 同時に、SAP ASE からの一般的な応答時間を減速する可能性

があるため、CPU リソースの過剰占有を回避することをおすすめします。 このため、並列度の値を決定する際は、SAP ASE エンジンの数を考慮します。

データベースの暗号化では、デバイス I/O が大きなボトルネックです。 SAP ASE では次の 2 つの角度からこの問題に対処します。

- すべての別個のデバイスにワーカスレッドが割り当てられると、デバイス I/O は独立して同時に実行されるため、最良の処理能力が得られます。 このため、並列度ではデータベースの格納に使用されるディスク数を考慮する必要があります。
- ページの大きなまとまりをすべてのデバイスの読み込み/書き込みで処理できればパフォーマンス上有利です。 暗号化/復号化の進行中はデータベースがオンラインである必要があります。 このため、専用バッファを割り当てるのではなく、既存のバッファマネージャのメカニズムを利用して同期の問題を解決する必要があります。 この点については、十分なバッファキャッシュを作成し、バッファプールの I/O サイズを大きくすることで、SAP ASE の暗号化パフォーマンスを向上できます。

次の例は、demodb というデータベースを完全に暗号化するため、バッファキャッシュとプールサイズの両方を設定する方法を示しています。このデータベースのデータとログは 11 個のデバイスに分散しています。

```
> select dbid, segmap, lstart, size, vstart, vdevno from sysusages
where dbid=db_id('demodb')

dbid   segmap lstart     size       vstart      vdevno
------ ------ ---------- ---------- ----------- -----------
   4        3          0      92160           0           1
   4        4      92160      30720           0           2
   4        3     122880     184320       92160           1
   4        4     307200      61440       30720           2
   4        3     368640     419840      276480           1
   4        4     788480      61440       92160           2
   4        3     849920     122880      696320           1
   4        4     972800     153600      153600           2
   4        3    1126400     819200      819200           1
   4        3    1945600    1638400           0           3
   4        3    3584000    1638400           0           4
   4        3    5222400    1638400           0           5
   4        3    6860800    1638400           0           6
   4        3    8499200    1638400           0           7
   4        3   10137600    1638400           0           8
   4        3   11776000    1638400           0           9
   4        3   13414400    1638400           0          10
   4        3   15052800    1638400           0          11
   4        4   16691200     204800      307200           2
```

1. バッファキャッシュとバッファプールサイズを設定します。
   a. demodb 専用のデータキャッシュを作成します。

      ```
      sp_cacheconfig demodb_cache, '100M'
      ```

これによって、データベースページ領域 100MB で demodb_cache という名前のバッファキャッシュが作成されます。

b. バッファプールサイズをデータベースページサイズの 8 倍にして、指定サイズのバッファプールを作成します。

```
sp_poolconfig demodb_cache, '100M' , '16k'
```

デフォルトのデータベースページサイズは 2K であるため、バッファプールサイズは 8 X 2 = 16KB となります。
これによって、指定キャッシュ demodb_cache に 16K のバッファの 100MB のバッファプールが作成されます。

c. データベースをバッファキャッシュにバインドします。

```
sp_bindcache demodb_cache, demo_db
```

これによって、データベース demo_db が作成したバッファキャッシュ demodb_cache にバインドされます。

2. 使用する並列度を決定します。 この例では、8 個の SAP ASE エンジンが設定されています。

```
[Thread Pool:syb_default_pool]
```

スレッド数は 8 です。
ワーカスレッドの最大数を 8 より大きくすることはできません。
一方、11 個のデータベースデバイスを使用する SAP ASE では、デバイス I/O の並列実行に最大で 11 のワーカスレッドが必要です。 8 個のエンジンに対して 11 のワーカスレッドではエンジンの負担が重くなるため、並列度の設定は 8 にする必要があります。 ただし、SAP ASE で応答時間を維持し、他の操作の実行を可能にするには、並列度 6 を選択して、CPU リソースすべてを占有することを回避します。

a. 十分な数のワーカスレッドが設定されていることを確認します。

```
sp_configure 'number of worker processes', 6
```

b. 暗号化するようにデータベース demodb を変更します。

```
alter database demodb encrypt with dbkey parallel degree 6
```

sp_who では、次のように 6 個のワーカスレッドが示されます。

```
>sp_who
fid    spid    status    loginame    origname
    hostname    blk_spid    dbname
  tempdbname    cmd
        block_xloid        threadpool
------ -------- ---------------- ---------
……
  0    16    sleeping   NULL       NULL     NULL            0
master
   master DB    ENCRYPTION CONTROLLER        0
NULL
  16    1    sleeping   NULL       NULL     NULL            0
```

```
master
   master      WORKER PROCESS              0                   NULL
  16    17    sleeping   NULL       NULL     NULL              0
master
   master      WORKER PROCESS              0                   NULL
……
```

sp_helpdb では次のように暗号化の進行状況とステータスがレポートされます。

```
1> sp_helpdb demodb
2> go
name     db_size    owner  dbid  created       durability
lobcomplvl inrowlen status
-------- ---------- ------ ----- ------------- ----------
---------- -------- ----------------
demodb   33000.0 MB sa      4   Sept 27, 2013 full             0
NULL     encryption in progress: 18%
```

また、次のように **dbencryption_status** 関数を使用しても暗号化の進行状況とステータスを把握できます。

```
1> select dbencryption_status("status", db_id('demodb'))
2> go
-----------
2
1> select dbencryption_status("progress", db_id('demodb'))
2> go
-----------
21
```

次の例は、データベースページの 21 パーセントが暗号化されたことを示しています。

また、**dbencryption_status** を使用して特定のフラグメントの進行状況を確認することもできます。

```
1> select dbencryption_status("progress", db_id('demodb'),
92160)
2> go
-----------
83
```

この例は、論理ページの開始が 92160 のフラグメントのページの 83 パーセントが暗号化されていることを示しています。

暗号化されたデータベースは暗号化されていないデータベースと比較して、暗号化と復号化により多くのバッファを消費します。 暗号化と復号化のためにクリーンバッファが使用できない場合は次のようにします。

- バッファプールサイズおよびバッファプールウォッシュを拡大
- **housekeeper free write percent** をハウスキーピングタスクによるバッファのウォッシュ頻度を高めることができる値に設定

暗号化処理のサスペンド

暗号化処理中にデータベースの暗号化を停止するには、**alter database** コマンドの `suspend encrypt` オプションを使用します。

```
alter database database_name
    suspend encryption
```

参照：

- 既存データベースの暗号化 (70 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- 暗号化処理の再開 (77 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)

quiesce database コマンドと完全に暗号化されたデータベース

暗号化中のデータベースに対して **quiesce database** コマンドを実行すると、SAP ASE は暗号化処理を保留します。

quiesce database の実行後に **alter database** の `suspend encryption` オプションを実行する必要はありません。**quiesce database** によってデータベースの I/O オペレーションが自動的にサスペンドされます。

静止モードから解放された後に、暗号化 (または復号化) のタスクが自動的に再開します。**alter database** で **resume encryption** オプションを実行する必要はありません。

暗号化処理の再開

暗号化処理が中断またはサスペンドされたデータベースの暗号化を再開するには、**alter database** コマンドの `resume encryption` オプションを使用します。

```
alter database database_name
    resume encryption [parallel degree_of_parallelism]
```

参照：

- 既存データベースの暗号化 (70 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- 暗号化処理のサスペンド (77 ページ)

- データベースの完全暗号化に使用される alter database (119 ページ)

暗号化されたデータベースの復号化

完全に暗号化されたデータベースを復号化するには、**alter database** コマンドを使用します。

構文は次のとおりです。

```
alter database database_name
  {decrypt [with key_name] [parallel degree_of_parallelism]
| resume decryption [parallel degree_of_parallelism]
| suspend decryption}
```

構文の説明は次のとおりです。

- *database_name* - 復号化する完全暗号化データベースの名前です。
- *key_name* - (オプション) データベースの暗号化に使用したデータベース暗号化キーです。 別のキー名を指定するとコマンドが失敗し、SAP ASE にエラーメッセージが表示されます。
- `resume decryption` - 復号化処理が前回サスペンドされたデータベースの復号化処理を再開します。 *database_name* がすでに完全に復号化されている場合、SAP ASE はこのコマンドを無視します。
- `parallel` *degree_of_parallelism* - タスクを開始するワーカスレッド数を指定します。
- `suspend decryption` - 復号化処理を強制終了します。 SAP ASE は処理が停止した位置を記録して、`resume decrypt` がデータベースの適切な位置から復号化処理を再開できるようにします。

このコマンドを使用するには、データベースの *key_name* に対する **select** パーミッションを保持している必要があります。

完全暗号化データベースのリカバリ

マスタまたはデュアルマスタキーが使用できないために、起動時にデータベース暗号化キーを取得できない場合、SAP ASE はその暗号化データベースを無視します。

完全に暗号化されたデータベースをどのように処理するかを認識するには、SAP ASE がデータベース暗号化キーにアクセス可能である必要があります。 データベース暗号化キー自体も暗号化され、マスタキーを使用して復号化されます。

SAP ASE の再起動後にサーバに接続するには、マスタまたはデュアルマスタキーのパスワードホルダで次のように暗号化パスワードを設定できます。

```
set encryption passwd for key [dual] master
```

これでマスタキーによるデータベース暗号化キーの復号化が可能になり、その時点でデータベース暗号化キーによって完全に暗号化されたデータベースをオンラインにすることができます。

```
online database encrypted_database_name
```

サーバがオンラインに復帰すると、データベースリカバリが行われます。

自動リカバリを設定することもできます。『暗号化カラムユーザーズガイド』の「無人起動モードでの Adaptive Server の起動」を参照してください。

完全暗号化データベースのバックアップ (ダンプ)

完全暗号化データベースのバックアップは、暗号化処理が透過的に実行されるため、通常の非暗号化データベースの場合と同様です。

暗号化データベースのバックアップダンプをロードするには、ダンプの暗号化に使用したものと同じ暗号化キーを使用する必要があります。

データベースの暗号化キーは、バックアップ対象のデータベースの外部の `master` データベースに格納されます。 このため、**dump database** コマンドの実行時にバックアップ処理はデータベース暗号化キーに自動では適用されません。データベース暗号化キーとマスタキーは、データベースのバックアップとは切り離して、別途バックアップする必要があります。

キー値をバックアップするには、次のいずれかの方法を使用します。

- **ddlgen** ユーティリティを使用して DDL 文を生成します。
- 直接、バックアップします。

データベース暗号化キーのバックアップ

リカバリ性を回復するには、データベース暗号化キー、マスタまたはデュアルマスタキー、および暗号化データベースをバックアップする必要があります。

この例では、**ddlgen** ユーティリティを使用してデータベース暗号化キーに対する SQL 文を生成します。

```
ddlgen -Usa -P -Sserver -TEK -Nmaster.owner.dek_name
```

この構文は、デュアルマスタキーに対する SQL 文を生成する場合と似ています。

参照：

- データベース暗号化キーの変更 (67 ページ)
- データベース暗号化キーの削除 (68 ページ)
- create encryption key (130 ページ)

- drop encryption key (159 ページ)
- データベース暗号化キーの作成 (66 ページ)

完全に暗号化されたデータベースのバックアップのリストア (ロード)

完全に暗号化されたデータベースは、通常の暗号化されていないデータベースの場合と同じ方法でリストアします。

暗号化されたデータベースのダンプのロードを可能にするには、以下を実行します。

1. マスタキーおよびデータベース暗号化キーをリストアします。
2. ロードするデータベースに使用したデータベース暗号化キーを使用して、暗号化するターゲットデータベースを作成します。

次のコマンドを使用して、暗号化されたデータベースをリストアします。*database_name* は、リストアする暗号化データベースの名前です。

```
load database database_name
```

注意： 暗号化されたデータベースでは、検証オプション (`load database database_name with verify only = full`) を使用できません。 このオプションを指定すると、Backup Server によってすべてのローが読み込まれ、ローフォーマットが有効であることが確認されます。 Backup Server では、暗号化されたテキストが認識できないため、コマンドが失敗し、Backup Server にエラーメッセージが表示されます。

暗号化されたデータベースをリストアするために **load database** を実行すると、SAP ASE によって、ターゲットデータベースに関して以下の事項が検証されます。

- 暗号化されたデータベースであるか。 暗号化されたデータベースでない場合は、SAP ASE にエラーメッセージが表示され、**load database** コマンドが失敗します。
- 正しいデータベース暗号化キーが存在しているか。 データベース暗号化キーが一致しない場合、SAP ASE にエラーメッセージが表示されます。

暗号化されたデータベースのロード動作

ロード動作は、ターゲットデータベースと、リストアするデータベースまたはトランザクションログの両方の暗号化ステータスによって異なります。

ロード動作	暗号化されていないターゲットデータベース	暗号化されたターゲットデータベース	一部が暗号化されたターゲットデータベース	一部が複合化されたデータベース
暗号化されていないデータベースダンプ	可能。	with override 句を使用した場合のみ可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。	with override 句を使用した場合のみ可能。 ダンプステータスがターゲットデータベースに反映される。	with override 句を使用した場合のみ可能。 ダンプのセキュリティステータスが反映される。
暗号化されていないトランザクションダンプ	可能。	可能。 一部が暗号化されているとしてターゲットデータベースにマークされる。	可能。 ターゲットデータベースは、一部暗号化済みのステータスを保持。	可能。 ターゲットデータベースは、一部暗号化済みのステータスを保持。
暗号化されたデータベースダンプ	不可。	可能。	可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。	with override 句を使用した場合のみ可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。
暗号化されたトランザクションダンプ	不可。	可能。	可能。 ターゲットデータベースは、一部暗号化済みのステータスを保持。	不可。
一部が暗号化されたデータベースダンプ	不可。	可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。	可能。 ターゲットデータベースは、一部暗号化済みのステータスを保持。	with override 句を使用した場合のみ可能。 ダンプステータスがターゲットデータベースに反映される。

ロード動作	暗号化されていないターゲットデータベース	暗号化されたターゲットデータベース	一部が暗号化されたターゲットデータベース	一部が複合化されたデータベース
一部が暗号化されたトランザクションダンプ	不可。	可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。	可能。 ターゲットデータベースは、一部暗号化済みのステータスを保持。	不可。
一部が復号化されたデータベースダンプ	不可。	`with override` 句を使用した場合のみ可能。 ダンプステータスがターゲットデータベースに反映される。	`with override` 句を使用した場合のみ可能。 ダンプのセキュリティステータスがターゲットデータベースに反映される。	可能。 ターゲットデータベースは、一部復号化済みのステータスを保持。
一部が復号化されたトランザクションダンプ	不可。	不可。	不可。	可能。 ターゲットデータベースは、一部復号化済みのステータスを保持。

暗号化中のデータベースの削除

暗号化中または復号化中のデータベースを削除できます。

暗号化または復号化中のデータベースに対して **drop database** コマンドを実行すると、**drop database** によって暗号化/復号化処理が強制終了され、`sysattributes` システムテーブルの検索、すべての進行情報のクリーンアップが行われた後にデータベースが削除されます。

完全暗号化データベースのマウントおよびマウント解除

暗号化または復号化中のデータベースはマウントできません。

暗号化されたデータベースはマウントできません。

暗号化されたデータベースに対して **umount database** コマンドを使用しないでください。コマンドが失敗して SAP ASE に暗号化されたデータベースのマウント解除には事前に復号化が必要であることを意味するメッセージが表示されます。

```
Could not unmount encrypted database 'mydatabase'.
```

参照：

- 暗号化されたデータベースの復号化 (78 ページ)

アーカイブデータベースと完全暗号化

アーカイブデータベースは読み込み専用です。 暗号化構文によってアーカイブデータベースは暗号化されたデータベースダンプのロードが可能であることが示されます。

データベースのバックアップおよびロードの場合と同様に、マスタキーとデータベース暗号化キーをリストアして、DEK とアーカイブデータベースを関連付けます。

完全に暗号化されたアーカイブデータベースをダンプまたはロードするには、通常のデータベースの場合と同様の手順を実行します。

アーカイブデータベースを作成するには、以下を使用します。

```
create archive database database_name
    encrypt with key_name
```

各パラメータの意味は次のとおりです。

- *database_name* は、作成するアーカイブデータベースの名前です。
- *key_name* は、バックアップ (ダンプ) したデータベースの暗号化に使用したキーです。 SAP ASE は、データベースのダンプの際に *key_name* が一致していることを検証します。 一致しない場合は、リストアが失敗します。

通常のデータベースとは異なり、アーカイブデータベースには、再実行/取り消し、トランザクションのロード操作によるページの変更や割り付け情報を格納する変更ページセクションがあります。 アーカイブデータベースを暗号化する際は、アーカイブデータベースからのデータベース暗号化キーを使用して変更ページセクションも暗号化されます。

create archive database コマンドの `encrypt with` オプションの使用に必要なパーミッションは特にありません。 ただし、ユーザが *key_name* として参照する場合は、データベース暗号化キーに対する **select** パーミッションが必要です。

参照：
- 完全暗号化データベースのバックアップ (ダンプ) (79 ページ)
- 完全に暗号化されたデータベースのバックアップのリストア (ロード) (80 ページ)

データベースの完全暗号化とシステム変更

SAP ASE 16.0 のデータベースの完全暗号化機能に関連するシステム変更があります。

- **alter database** コマンド
- **alter encryption key** コマンド
- **create archive database** コマンド
- **create database** コマンド
- **create encryption key** コマンド
- **dbencryption_status()** 組み込み関数
- **ddlgen** ユーティリティ
- **sp_encryption** システムプロシージャ
- **sybmigrate** ユーティリティ
- `sysdatabases` システムテーブル

参照：
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)

- drop encryption key (159 ページ)

スケーラビリティの拡張と機能

SAP ASE バージョン 16.0 では、実行時のロギングとロック、メタデータ、およびラッチ管理の強化により、スケーラビリティ機能が向上しています。

実行時ロギングの拡張

SAP ASE 16.0 には、実行時ログインの拡張が含まれています。

ログレコードを `syslogs` に転送するには、SAP ASE がログロックを取得する必要があります。ログロックは、処理量の多い OLTP システムでは競合のポイントになる可能性があります。 SAP ASE は、トランザクションごとにユーザログキャッシュを使用することで、競合を減らし、実行時ロギングのパフォーマンスを向上させます。 ログレコードを `syslogs` に転送する前にユーザログキャッシュがログレコードをバッファするため、SAP ASE は個々のログレコードの代わりにログレコードバッチを syslogs に送信することができます。

ユーザログキャッシュを最大限に活用するために、SAP ASE は可能であればトランザクションの完了時に 1 回だけログレコードをログに転送します。 しかし、データローロックテーブルを使用する処理量の多い OLTP システムで同時にオープンしている複数のトランザクションが同じデータページに変更を行った場合、SAP ASE はトランザクションが完了する前にログレコードをユーザログキャッシュから `syslogs` に転送することを余儀なくされることがあります。この場合は、バッチ処理の量が減り、ログロックでの競合が増えます。 多数の同時トランザクションで同じテーブルを更新する処理量の多い OLTP システムでは、SAP ASE がトランザクション完了前に頻繁にユーザログキャッシュからログにログレコードを転送すると、ユーザログキャッシュでログレコードをキャッシュするメリットはまったくなくなる可能性があります。

SAP ASE バージョン 16.0 以降では、競合を減らすために各ユーザログキャッシュを複数のより小さなブロック (それぞれのサイズはサーバの論理ページサイズ) に分割します。 SAP ASE バージョン 16.0 は、ログレコードをユーザログキャッシュ内の現在のアクティブブロックに直接追加します (以前のリリースでログレコードをユーザログキャッシュに直接追加したのと同じ方法)。 同時オープントランザクションが同じデータページに変更を行った場合、SAP ASE はログレコードを直接ユーザログキャッシュから `syslogs` に移動する代わりに、ユーザログキャッシュ内の現在のブロックをグローバルキューの末尾に追加 (リンク) します。ログレコードは、後でグローバルキューから `syslogs` に転送できます。 これにより、

SAP ASE が `syslogs` への追加を効率的にバッチ処理できるため、データローロックテーブルに関連する問題とは関係なく実行時ロギングのパフォーマンスは向上します。

現在のブロックがグローバルキューに追加されると、ユーザログキャッシュ内の新しい空きブロックが現在のブロックになり、トランザクションからのログレコードを継続的に受け入れます。

この機能を有効または無効にするには、**user log cache queue size** 設定パラメータを使用します。 **user log cache queue size** を次のように設定することができます。

- 1 - SAP ASE はキューイング方式を使用します (デフォルト)。
- 0 - SAP ASE は、16.0 より前のバージョンの動作を使用します。

キューイング方式を使用するように SAP ASE を設定する場合は、**user log cache size** をサーバの論理ページサイズの 4 倍以上のサイズに設定する必要があります。 つまり、各ユーザログキャッシュに 4 つ以上のログキャッシュブロック (それぞれのサイズはサーバの論理ページサイズ) が存在する必要があります。 値が DEFAULT の場合、SAP ASE バージョン 16.0 は値をサーバの論理ページサイズの 4 倍の値に設定します。

user log cache size と **user log cache queue size** の値を設定する場合は、次のことに留意してください。

- **user log cache queue size** を 0 より大きい値に設定した場合は、**user log cache size** を論理ページサイズの 4 倍より小さい値に設定することはできません。
- **user log cache size** を論理ページサイズの 4 倍かそれより小さい値に設定した場合は、**user log cache queue size** の値を 0 から 1 に変更することはできません。

ロック管理の拡張

SAP ASE バージョン 16.0 以降には、ロック管理用の複数の拡張機能が含まれています。

16.0 より前のバージョンの SAP ASE では、スピンロックにより、ロック競合が起こることがありました。 SAP ASE バージョン 16.0 では、以下の領域を最適化することにより、ロック競合発生率が低下しています。

- エンジンロックの転送 - グローバルプールとエンジンローカルキャッシュ間のロックの転送機能が向上しています。
- エンジンロックのキャッシュ - エンジンがローカルにキャッシュできるロック数が以下の方法により最適化されています。
  - デフォルトエンジンのローカルキャッシュサイズの増加

    - グローバルキャッシュからローカルキャッシュへのロック転送サイズの増加
    - 個々のロックではなく、ロックのブロックを使用したローカルとグローバル間でのロック転送の強化
    - ローカルキャッシュからグローバルキャッシュへのロックの排出または再利用の減少
- `LOCK_VERIFY` オペレーション - `LOCK_VERIFY` オペレーションを最適化し、可能な限りオプティミスティックアプローチを配備し、ロック取得を回避します。
- ロックプロモーション - 繰り返し失敗するロックプロモーション試行を追跡し、一定の試行回数を超えると、失敗したロックプロモーションを発生させている DML 文に対してロックプロモーションを無効にします。
- ログセマフォのロック - `semawait` によって保持できるのは 1 つのロックのみなので、ログセマフォのロック取得を回避します。
- デッドロックのチェック - **deadlock checking period** に小さい値を指定する場合に、1 より小さい値が指定されないようにし、デッドロックチェックが専用サービススレッド上で実行されるようにします。
- ホット DOL テーブル - DOL テーブルがホットテーブルであることを許可し、テーブルロック競合率を低下させます。

これらの機能は、パフォーマンス向上を目的としており、SAP ASE がこれらの拡張機能を使用したときに、パフォーマンスの低下が発生することはありません。ロック管理拡張機能は、デフォルトで有効になっており、これを有効化するための設定は必要ありません。

メタデータとラッチ管理の拡張

SAP ASE Enterprise バージョン 16.0 以降では、トランザクション率が非常に高い環境においてラッチ競合のための CPU 使用率が低下しています。

以下の場合の競合が減少しています。

- データベースクエリおよびトランザクション間でのトランザクション率が非常に高いときの内部構造における競合。暗黙的または明示的なデータベース間参照によるロック、ラッチ、データキャッシュの競合を含みます。
- パーティションが多いテーブルでの高トランザクション率オペレーションの実行中の競合。
- テーブルの作成および削除の発生率が非常に高いときの内部構造における競合。複数のシステムテーブル上でのロックおよびラッチ競合の低減を含みます。

これらの機能は、パフォーマンス向上を目的としており、SAP ASE がこれらの拡張機能を使用したときに、パフォーマンスの低下が発生することはありません。

メタデータおよびラッチ管理拡張機能は、デフォルトで有効になっており、これを有効化するための設定は必要ありません。

SAP ASE バージョン 16.0 以降では、非常に高いトランザクション率である環境においてラッチ競合中の CPU 使用率が低減されています。

単一ページでの同時実行オペレーションは、トランザクション率が非常に高くなる期間において競合の原因になります。これは、単一キャッシュパーティションのデータキャッシュスピンロック競合として発生します。 通常、この競合は、単一データページ上での大量のラッチ要求により起きることがあります。 **sp_chgattribute exp_row_size** パラメータにより、サーバ領域を管理できます。 SAP ASE は、**exp_row_size** で設定されているサイズに従って、データページ上の領域を予約します。これにより、ページ単位のロー数が制限されます。

SAP ASE バージョン 16.0 以降では、固定長カラムのみのテーブルに対して **exp_row_size** に値を設定できます。 競合を減少させるには、**exp_row_size** を論理ページサイズ (8,192 など) に設定することにより、ページ単位のロー数を減らします。

高ワークロードシナリオでは、プロシージャキャッシュの競合が増加する可能性があります。SAP ASE は、各エンジン用にローカルメモリを確保することにより、これを低減します。 このメモリはローカルなので、このメモリにアクセスしても競合の原因になりません。

16.0 より前のバージョンの SAP ASE では、プロシージャキャッシュの 25% を、特定サイズの要求で使用できるエンジンローカルキャッシュ (ELC) として確保していました。 トレースフラグ 758 を使用することにより、このサイズがプロシージャキャッシュの 50% まで増加し、これで ELC はすべての要求サイズに対応できていました。

SAP ASE バージョン 16.0 以降は、デフォルトで ELC が有効になっています。 プロシージャキャッシュの 50% が ELC 用に使用され、すべてのサービスサイズに対応します。 **engine local cache percent** 設定パラメータが ELC のサイズを決定します。 デフォルト値 50 は、ローカルキャッシュがプロシージャキャッシュの 50% を使用することを意味し、各エンジンは以下を受信します。

```
((0.50 * procedure_cache_size) / number_of_online_engines)
```

ELC のサイズを増やすことで、プロシージャキャッシュでの競合を減らすことができます。 たとえば、ELC をデフォルトの 50% から 60% まで増やすには、次のように指定します。

```
sp_configure 'engine local cache percent', 60
```

さらに、SAP ASE バージョン 16.0 以降には、エンジンローカルキャッシュを制御する以下のような設定パラメータが含まれています。

- **enable large chunk elc** (トレースフラグ 758 の置き換え)
- **large allocation auto tune** (トレースフラグ 753 の置き換え)

これらの設定パラメータは、デフォルトで有効です。

スレッショルドベースイベントの監視

SAP ASE バージョン 16.0 以降では、スレッショルドイベントを設定、記録、リストする機能が含まれています。

クエリが CPU の使用に費やす時間を設定するには、**sp_add_resource_limit cpu_time** 制限タイプを使用します。

`monThresholdEvent` テーブルにイベントを記録するには、**sp_add_resource_limit** *action* パラメータを 5 に設定します (『リファレンスマニュアル: プロシージャ』を参照)。

構文は次のとおりです。

```
 sp_add_resource_limit name, appname, rangename, limittype,
limitvalue [, enforced [, action [, scope ]]]
```

この例では、クエリバッチの実行に費やした CPU 時間が 120 秒より長い場合に SAP ASE が警告を発行して、`monThresholdEvent` にそのイベントを記録します。

```
sp_add_resource_limit NULL, payroll, tu_wed_7_10,cpu_time, 120, 2,
5, 2
```

`monThresholdEvent` では、設定されているリソース制限に対するすべての違反を記録し、次の項目のスレッショルドを指定できます。

- トランザクションあたりの `tempdb` 使用量
- クエリまたはトランザクションの合計実行時間
- クエリによってフェッチまたは書き込まれるローの数
- クエリによって読み込まれるローの数
- クエリの推定プランコスト
- クエリの合計論理/物理 I/O 数
- クエリの合計 CPU 時間

複数のトリガ

SAP ASE バージョン 16.0 では、複数のトリガの作成機能、および文の実行後のトリガの起動順序の指定機能が導入されています。

複数のトリガの作成

1 つのテーブルでは各コマンド (**insert**、**update**、**delete**) に対して、最大で 50 のトリガを作成できます。 **create trigger** コマンドで新しい **order** パラメータを使用して、トリガの起動順序を指定します。 複数のトリガは、order 句なしでも作成できます。

構文は次のとおりです。

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    for {insert | update | delete}
    [order integer]
    as sql_statements
```

`order` *integer* は、トリガ機能の全順序または順序の一部を指定します。

- `order` 句を使用してすべてのトリガを作成すると、全順序の指定が行われます。
- 部分順序指定は、トリガの一部で `order` 句を指定しない場合に行われます。`order` 句がないトリガは、暗黙的に順序番号 0 を取得し、`order` 句を使用して作成されたトリガの後に起動されることを除き、順序は定義されません。

注意： `order` *integer* 句は、`for {insert | update | delete}` でのみ使用することができます。`instead of {insert | update | delete}` トリガで使用することはできません。

`order` で指定する数値に重複があると、SAP ASE でエラーがレポートされます。`order` の番号が連続している必要はありません。むしろ順序の途中に新しいトリガを挿入できるため、非連続の番号のほうが望ましいです。

sp_helptrigger は、次のリストに使用します。

- *tablename* によって指定されたテーブルで作成されたすべてのトリガ
- トリガアクションで指定された **insert**、**update**、**delete** コマンド
- トリガの順序番号

参照：

- 複数のトリガで使用される create trigger (152 ページ)

トリガ起動時の順序の変更

起動されるトリガの順序を変更するには、**create trigger** コマンドで **or replace** オプションを使用し、元のトリガと同じトリガ名、アクション、トリガ本文を使用し、新しい **order** 番号を指定します。

Merge 文のトリガの順序

`merge` 文は特定の順序でトリガを起動します。

merge 文は、ターゲットテーブルのトリガを次の順序で起動します。

1. **insert**
2. **update**
3. **delete**

このため、**update** 文の順序番号が **insert** の順序番号より小さい場合も、**insert** 文のトリガが先に起動されます。

ただし、同じ操作のトリガが複数存在する場合は、順序が指定されます。 つまり、まず、すべての **insert** のトリガが順序に従って起動され、次に **update** のすべてのトリガが順序に従って起動されるというように処理されます。

この例では、dbo によって `GlobalSales` テーブルに次のトリガが作成されます。

- trigger1 は順序 1 の delete
- trigger2 は順序 4 の delete
- trigger3 は順序 1 の insert
- trigger4 は順序 5 の insert

merge 文によってデータが `GlobalSales` にマージされます。 merge に `GlobalSales` に対する insert 操作と delete 操作の両方が含まれる場合、SAP ASE 実行後に次の順序でトリガを起動します。

- trigger3
- trigger4
- trigger1
- trigger2

複数のトリガによるトランザクション動作

トリガでロールバックトランザクションが実行されると、トリガを起動した **insert**、**update**、または **delete** 文が、そのトリガによって実行されたすべての作業とともにロールバックされます。

複数のトリガの場合にいずれかのトリガからロールバックトランザクションが実行されると、すでに起動されている他のトリガの作業もロールバックされ、当該の insert、update、または delete コマンドの残りのトリガの起動が保留されます。

トリガの無効化と再有効化

`alter table` コマンドを使用すると、複数のトリガの無効化と再有効化を実行できます。

alter table コマンドを使用して、複数のトリガを個別に無効化および再有効化します。

トリガを無効化または再有効化するには、次のように使用します。

```
alter table table_name {disable | enable} trigger trigger_name
```

@@trigger_name グローバル変数

@@trigger_name グローバル変数は、現在実行中のトリガの名前を返します。

トリガの本文またはトリガから (いずれかのネストレベルで) 呼び出されたストアドプロシージャの本文に以下を配置できます。

```
select @@trigger_name
```

ネストされたトリガが起動されると、`@@trigger_name` に直前に起動されたトリガの名前が保管されます。

残存データの削除

残存データを削除してデータベースデータのセキュリティを強化します。

スペースを削除するデータベース操作の中には、データが常に物理的に消去されるとは限らないものがあります。 この残存データは **dbcc** ユーティリティを使用してユーザが見ることができるため、セキュリティ脅威がもたらされる可能性があります。 SAP ASE バージョン 16.0 で導入された機能を有効化すると、ユーザがこのようなデータベース操作を行ったときに自動的に残存データをゼロ設定することができます。

バージョン 16.0 より前のバージョンの SAP ASE では、ディスクページをゼロ設定して残存データを物理的に削除しなかったため、ディスク上に機密データを残している可能性がありました。 バージョン 16.0 以降では、データを機密としてマークし、削除操作または更新操作を実行した後に残存データを消去するように SAP ASE を設定することができます。

注意： ユーザがこのようなデータベース操作を実行したときに残存データを自動的にゼロ設定できます。`master`、`sybsystemdb`、`sybsystemprocs` などのシステムデータベースや、各データベース内にあるシステムテーブルから残存データを削除することはできません。

残存データが発生する操作については、『セキュリティ管理ガイド』の「残存データの削除」を参照してください。

設定履歴の追跡

SAP ASE バージョン 16.0 では、サーバに対して行われた設定変更の履歴を追跡する機能が追加されています。

sp_confighistory システムプロシージャは設定変更の履歴を管理し、変更に関するデータを `sybsecurity` データベースに格納します。

追跡される設定プロパティは次のとおりです。
- サーバワイドな設定パラメータ
- データベースオプション
- データキャッシュとデータキャッシュプールのプロパティ
- エンジンスレッド
- サーバ設定ファイルに対する変更

これらのプロパティを追跡するには、`sybsecurity` 監査データベースをインストールする必要があります。

sp_confighistory は、変更された設定オプション、変更前の値および変更後の値、変更したユーザ、および変更した時刻など、SAP ASE 設定の変更を表示します。SAP ASE は、設定変更のレコードを `sybsecurity` データベースに格納します。`ch_events` ビューにクエリを実行し、**sp_confighistory** を実行してそれらのレコードにアクセスします。

たとえば、次の出力には監査の有効化と **max online engines** の値の 5 から 7 への変更を含む変更が含まれています。

```
area type target element oldvalue newvalue mode timestamp username
instanceid
---- ---- ------ ------- -------- -------- ---- --------- --------
----------
AUDIT global auditing  NULL NULL off on NULL Jul 15 2013 2:22PM sa
NULL
SERVER sp_configure NULL max online engines 5 7 static Jul 15 2013
2:23PM sa NULL
```

設定変更を追跡するための SAP ASE の設定

sp_confighistory をインストールするには、`installsecurity` スクリプトを実行します。

前提条件

注意： `ch_events` はすべての監査テーブルから情報を収集するため、監査テーブルが追加または削除されると同期しなくなります。 同期していない場合に

ch_events に対してクエリを実行すると、ch_events には一部の設定履歴レコードの変更が含まれない可能性があります。または、エラーメッセージ 208 (「table not found」) と 4413 (「view unusable」) が表示されます。

監査テーブルを追加または削除した場合は、**sp_confighistory create_view** を使用して ch_events を更新してください。 **sp_confighistory create_view** は、ビューが存在する場合はそのビューを削除して、現在の監査テーブルに対応する新しいビューを作成します。

監査システムをインストールして有効にします。『セキュリティ管理ガイド』を参照してください。

手順

1. 設定履歴追跡を有効化します (細密なパーミッションが有効化されている場合は、sa_role、sso_role、または manage auditing が必要)。
   ```
   sp_audit "config_history", "all", "all", "on"
   ```
   注意： sp_audit の発行は、設定履歴に記録されます。
2. 監査を有効化します。
   ```
   sp_configure 'auditing', 1
   ```
3. sybsecurity データベースに移動します。
   ```
   use sybsecurity
   ```
4. ch_events ビューを作成します。
   ```
   sp_confighistory create_view
   ```

取得された変更

設定履歴監査が有効になっている場合、SAP ASE は多数のイベントを取得します。

ch_events ビューでは、新しい値が古い値と同じである場合は変更が記録されません。

ch_events では、次の設定変更が記録されます (詳細については以下で説明)。

- 設定ファイルの変更
- 設定ファイルの読み込み
- **sp_configure** の変更
- サーバオプションの変更
- データベースオプションの変更
- キャッシュ設定の変更

- トレースフラグとスイッチの変更
- エンジン数の変更
- SAP ASE の起動イベントとシャットダウンイベント
- 監査の有効化と無効化

起動設定の変更
SAP ASE のシャットダウン時に SAP ASE 設定ファイルを変更すると、SAP ASE によって起動された ch_events テーブルに設定の変更が記録されます (これらの変更の場合は `mode` 値および `username` 値として NULL が記録されます)。

設定ファイルの読み込み
`ch_events` は、ユーザが設定ファイルの読み込み、書き込み、確認、およびリストアを行ったことは記録しますが、設定値の変更は記録しません。 たとえば、**number of user connections** の値を変更して、次のコマンドを発行した場合、

```
sp_configure "configuration file", 0, "read", "srv.config"
```

`ch_events` は、設定ファイルが読み込まれたことは記録しますが、設定値の変更は記録しません。

sp_configure の変更
SAP ASE は、**sp_configure** によって行われた、次の変更をすべて記録します。

- 設定パラメータの名前
- 古い設定値
- 新しい設定値
- パラメータが動的と静的のいずれであるか
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン

設定ファイルの読み込みによって発生した設定の変更は記録されません。 つまり、SAP ASE は、読み込み操作、書き込み操作、確認操作、リストア操作を記録しますが、読み込み操作によって発生した設定の変更は記録しません。 ユーザが別の設定ファイルまたは手動で変更した設定ファイルを読み込んで設定値を変更することもあります。 SAP ASE はファイルを読み込んだことは記録しますが、個別のパラメータの変更は記録しません。

サーバオプションの変更
`ch_events` は、**sp_serveroption** によって行われた、次の変更をすべて記録します。

- 影響を受けたサーバの名前
- 変更されたオプションの名前

- 古いオプション値
- 新しいオプション値
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン

データベースオプションの変更
`ch_events` は、**sp_dboption** によって行われた、次の変更を記録します。

- 影響を受けたデータベースの名前
- 変更されたオプションの名前
- 古いオプション値
- 新しいオプション値
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン

キャッシュ設定の変更
`ch_events` は、**sp_cacheconfig** と **sp_poolconfig** によって行われたキャッシュ設定の変更をすべて記録します。

記録される **sp_cacheconfig** による変更は次のとおりです。

- 影響を受けたキャッシュの名前
- 古いキャッシュサイズ
- 新しいキャッシュサイズ
- 属性 (cache type、cache replacement policy、partition number) (該当する場合)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン
- (Cluster Edition のみ) この変更が適用されるインスタンス

記録される **sp_poolconfig** による変更は次のとおりです。

- 影響を受けたキャッシュの名前
- 設定プール
- 古いキャッシュプールサイズ
- 新しキャッシュプールサイズ
- 変更された属性の名前 (affected pool、wash size、asynchronous prefetch (APF) percentage)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン
- (Cluster Edition のみ) この変更が適用されるインスタンス

トレースフラグとスイッチの変更
`ch_events` は、**dbcc traceon | off** と **set switch on | off** の変更を記録します。

記録される **dbcc traceon | off** の変更は次のとおりです。

- 影響を受けたトレースフラグ
- セッション ID
- 古いトレースフラグステータス (**on** または **off**)
- 新しいトレースフラグステータス (**on** または **off**)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン
- (Cluster Edition のみ) この変更が適用されるインスタンス

記録される **set switch on | off** の変更は次のとおりです。

- 影響を受けたスイッチの番号
- 古いスイッチステータス (**on** または **off**)
- 新しいスイッチステータス (**on** または **off**)
- 変更された属性の名前 (server-wide または session-specific、with override、with no_info)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン

エンジン数の変更

create thread pool、**alter thread pool**、および **drop thread pool** によって追跡される変更は次のとおりです。

- 影響を受けたプールの名前
- 古いプールサイズ
- 新しいプールサイズ
- 変更された属性の名前 (new pool name、idle timeout など)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン
- (Cluster Edition のみ) この変更が適用されるインスタンス

SAP ASE の起動イベントとシャットダウンイベント

`ch_events` は、SAP ASE と Cluster Edition の インスタンスの startup、**shutdown**、**shutdown with nowait**、および abrupt (unscheduled) shutdown の各イベントについて次の情報を記録します。

- アクションの名前 (startup、**shutdown**、**shutdown with nowait**、abrupt shutdown)。
- シャットダウンの待機に費やした時間。 **shutdown with no_wait** の場合は適用されません。
- サーバが起動したホストの名前。

- 変更日時を示すタイムスタンプ。
- 変更ユーザのログイン。
- (Cluster Edition のみ) この変更が適用されるインスタンス。

注意： ch_events は、ユーザが **shutdown** コマンドを発行したときにシャットダウンを記録するため、正常なシャットダウン時にこのコマンドを複数回発行すると、ch_events は複数のシャットダウンを記録します。

監査の有効化と無効化
ch_events は、追跡、グローバル監査、および設定履歴監査についての以下の情報を記録します。

- アクション名 (**enable** または **disable**)
- 変更日時を示すタイムスタンプ
- 変更ユーザのログイン

ch_events のクエリによる変更表示

SAP ASE には、sybsecurity データベースの一部として ch_events ビューが含まれています。

ch_events は、設定変更履歴を読みやすい形式で表示します。 ch_events のクエリを直接実行するか、**sp_confighistory** システムプロシージャを使用して、設定変更履歴のレポートを生成できます。 どちらの方法でも同じ情報が提供されます。

select コマンドを使用すると、Transact-SQL™ 言語の柔軟性により結果セットにフィルタをかけることができます (最初に sybsecurity データベースに移動してから、ch_events ビューから選択する必要があります)。 **sp_confighistory** では、さらに簡素化された結果セットが提供されます。

たとえば、SAP ASE で次の設定変更を行います。

```
sp_dboption sybsystemprocs, "delayed commit", false
sp_cacheconfig pub_cache, '10M'
sp_cacheconfig pub_log_cache, '2000K', logonly
```

次に、サーバを停止してから再起動すると、**sp_confighistory** によって次の結果セットが返されます。

```
sp_confighistory

 area     type            target         element             oldvalue
         newvalue    mode timestamp           username instanceid
 -------- --------------- -------------- -------------------
--------
          ----------- ---- ------------------- -------- ----------
```

```
 AUDIT    global auditing NULL           NULL                off
         on          NULL Aug 22 2013 11:56AM sa             NULL
 DATABASE sp_dboption     sybsystemprocs delayed commit      true
         false       NULL Aug 22 2013  3:16PM sa             NULL
 CACHE    sp_cacheconfig  pub_cache      NULL                10240
         not changed NULL Aug 22 2013  3:18PM sa             NULL
 CACHE    sp_cacheconfig  pub_log_cache  cache type: logonly 2000
         not changed NULL Aug 22 2013  3:19PM sa             NULL
 SUSD     shutdown        NULL           NULL                NULL
         NULL        NULL Aug 22 2013  3:49PM sa             NULL
 SUSD     startup         NULL           tigger              NULL
         NULL        NULL Aug 22 2013  3:50PM NULL           NULL
```

sp_confighistory に日付を含めて、特定期間における変更を選択します。 この例では、2013 年 8 月 23 日後に行われたすべての変更が表示されます。

```
sp_confighistory "Aug 23 2013"

 area type     target element oldvalue newvalue mode timestamp
         username instanceid
 ---- -------- ------ ------- -------- -------- ----
-------------------
         -------- ----------
 SUSD shutdown NULL   NULL    NULL     NULL     NULL Aug 23 2013
9:00AM
         sa             NULL
 SUSD startup  NULL   tigger  NULL     NULL     NULL Aug 23 2013
10:38AM
         NULL           NULL
```

select を発行すると、次の結果セットが提供されます。

```
use sybsecurity
go
select * from ch_events
go

area        type                target           element
oldvalue    newvalue     mode
                 timestamp                username        instanceid
----------  ------------------  ----------------
-----------------    ----------  ---------    --------------
                    ------------------------   -----------
---------------
 AUDIT      global auditing      NULL             NULL
off          on            NULL
                    Aug 22 2013 11:56AM       sa                NULL
 DATABASE   sp_dboption         sybsystemprocs    delayed commit
true         false         NULL
                    Aug 22 2013  3:16PM       sa                NULL
 CACHE      sp_cacheconfig       pub_cache         NULL
10240        not changed  NULL
                    Aug 22 2013  3:18PM       sa                NULL
 CACHE     sp_cacheconfig      pub_log_cache    cache type: logonly
2000         not changed  NULL
                    Aug 22 2013  3:19PM       sa                NULL
```

```
 SUSD       shutdown             NULL              NULL
NULL         NULL          NULL
                    Aug 22 2013  3:49PM       sa               NULL
 SUSD       startup              NULL              tiger
NULL         NULL          NULL
                    Aug 22 2013  3:50PM       NULL             NULL
```

last パラメータを含めると、一番最後に変更された項目が表示されます。

```
sp_confighistory last

 area type    target element oldvalue newvalue mode timestamp
         username instanceid
 ---- ------- ------ ------- -------- -------- ----
--------------------
         -------- ----------
 SUSD startup NULL   tigger  NULL     NULL     NULL Aug 22 2013  3:50PM
         NULL           NULL
```

dump database 用巡回冗長検査

SAP ASE では、圧縮で作成されたデータベースダンプまたはトランザクションダンプのローデータに対する偶発的な変更をチェックし、この圧縮ブロックを正しく読み込んで、圧縮解除できるかを検証するための巡回冗長検査が追加されています。

構文は次のとおりです。

```
dump database database_name to dump_device with
compression=n,verify={crc | read_after_write}
load database database_name from dump_device with verify[only]=crc
```

各パラメータの意味は、次のとおりです。

- **verify=crc** - 巡回冗長検査を実行することを示します。
- **verify=read_after_write** - 圧縮ブロックの書き込みおよび圧縮解除後 Backup Server が圧縮ブロックごとに再読み込みを行います。 Backup Server がエラーを検出すると、エラーが検出されたファイル内のオフセットが出力されます。 **verify=read_after_write** は **dump database** コマンドでのみ使用可能です。

この例では、データベース `new_dump` が `mydumpdev` デバイスにダンプされる前に検証されます。

```
dump database new_dump to mydumpdev with
compression=x,verify=read_after_write
```

この例では、`new_dump` データベースダンプがロードされるときに巡回冗長検査が実行されます。

```
load database new_dump from mydumpdev with verify[only]=crc
```

この例では、データベース `new_dump` が `mydumpdev` デバイスにダンプされる前に、巡回冗長検査が実行され、圧縮ブロックごとに再読み込みが行われます。

```
dump database new_dump to mydumpdev with
compression=x,verify=read_after_write,verify=crc
```

使用法:

- **verify=crc** パラメータなしで作成されるダンプは、Backup Server の以前のバージョンと同じ形式を使用します。
- データベースを **verify=crc** を使用してダンプしていない場合、**verify=crc** オプションは無視されます。
- 巡回冗長検査機能を含まないバージョンの Backup Server で、巡回冗長検査を含むダンプをロードすることはできません。

- **verify={crc | read_after_write}** 検査は、**with compression** パラメータを使用して作成されたファイルに対してのみ適用可能です。
- **verify=crc** は、ローデバイス、ディスクファイル、テープ、パイプ、API を含む、あらゆるファイルタイプに対応します。 ただし、**verify=read_after_write** は、ブロックを再読み込みするために "シークバック" を必要とし、ローデバイスおよびディスクファイルでのみ適用可能です。
- 適用可能でない **verify={crc | read_after_write}** パラメータが使用された場合でも、SAP ASE はこれを無視し、エラーメッセージを生成しません。

トランザクションログの増大率の計算

SAP ASE バージョン 16.0 では、指定期間におけるトランザクションログの増大率を計算する機能が追加されています。

sp_logging_rate は、システムプロシージャを実行している期間のトランザクションログ増大率の最小値、最大値、平均値をギガバイト/時で表示します。結果は、計算の平均合計または反復結果として提供されます。

この例では、24 時間、1 時間間隔で計算されたトランザクションログの増大率の概要情報が表示されます。

```
sp_logging_rate 'sum', '1,00:00:00', '01:00:00'
=========================
Total Summary Information
=========================
Transaction Log Growth Rate      Min GB/h        Max GB/h        Avg
GB/h
---------------------------   --------------  --------------
--------------
                                  0.000000        1.970084
1.566053
```

システムの変更

SAP ASE 16.0 では、設定パラメータ、コマンド、システムプロシージャ、関数、モニタリングテーブル、およびユーティリティに変更が加えられています。

設定パラメータ

SAP ASE バージョン 16.0 では、設定パラメータに変更が加えられています。

新しい設定パラメータ

SAP ASE 16.0 には、新しい設定パラメータが含まれています。

enable utility lvl 0 scan wait

表 1 : 概要情報

デフォルト値	0
値の範囲	0 (オフ)、1 (オン)
ステータス	動的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	アプリケーション機能

Adaptive Server が独立性レベル 0 のスキャンを実行している間に、**alter table ... add | drop partition** コマンドを実行することができます。

注意： **enable utility lvl 0 scan wait** のデフォルト値は、**enable functionality group** に設定されている値によって異なります。 enable functionality group の設定値によって、デフォルト値は次のようになります。

- 0 - **enable utility lvl 0 scan wait** のデフォルト値は 0 です。
- 1 - **enable utility lvl 0 scan wait** のデフォルト値は 1 です。

注意： ただし、**enable utility lvl 0 scan wait** を 1 に設定すると、**enable functionality group** に設定した値に関係なく、値 1 を使用します。 「**enable functionality group**」を参照してください。

(UNIX のみ) max network peek depth

表 2 : 概要情報

デフォルト値	0
値の範囲	0 〜 2147483647
ステータス	動的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	ネットワーク通信

SAP ASE が保留中のキャンセルを確認するために接続オペレーティングシステムの受信バッファを読み取る深さのレベル数を指定します。 たとえば、クライアントが、SAP ASE が現在のコマンドの処理を完了する前に新しいコマンドを送信し、その後にキャンセルを送信した場合、SAP ASE は **max network peek depth** で指定されている深さまでオペレーティングシステムの受信バッファを読み取ります。 指定された深さの範囲内にそのキャンセルが存在する場合、SAP ASE は現在のコマンドと、キャンセルの前に送信されたコマンドの両方を破棄して、次のコマンドを待ちます。

aggressive task stealing

表 3 : 概要情報

デフォルト値	1 (オン)
値の範囲	0 (オフ)、1 (オン)
ステータス	動的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	SQL Server 管理

aggressive task stealing を有効にすると、SAP ASE スケジューラタスクスチールポリシーが aggressive (積極的) に設定されます。

enable large chunk elc

表 4 : 概要情報

デフォルト値	1 (オン)
値の範囲	0 (オフ)、1 (オン)
ステータス	静的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	メタデータキャッシュ

エンジンローカルキャッシュの大量の割り付けを有効にします。

engine local cache percent

表 5 : 概要情報

デフォルト値	50
値の範囲	0 〜 100
ステータス	静的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	メタデータキャッシュ

プロシージャキャッシュのパーセンテージとしてエンジンローカルキャッシュを変更できます。

large allocation auto tune

表 6 : 概要情報

デフォルト値	1 (オン)
値の範囲	0 (オフ)、1 (オン)
ステータス	静的
表示レベル	包括

必要な役割	System Administrator (システム管理者)
設定グループ	メタデータキャッシュ

クエリ実行用に大量メモリを事前割り付けするよう SAP ASE を設定します。これによりプロシージャキャッシュ競合が減少します。

threshold event max messages

表 7 : 概要情報

デフォルト値	0
値の範囲	0 〜 2147483647
ステータス	動的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	メモリユーザ、モニタリング

SAP ASE が `monThresholdEvent` テーブルに格納するイベントの数を決定します。`monThresholdEvent` モニタリングテーブル内のイベントの数がこの値を超えると、SAP ASE は最も古い未読のイベントを新しいイベントで上書きします。

threshold event monitoring

表 8 : 概要情報

デフォルト値	0 (オフ)
値の範囲	0 (オフ)、1 (オン)
ステータス	動的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	モニタリング

SAP ASE によるスレッショルドイベントの記録処理を有効化または無効化します。

user log cache queue size

表 9 : 概要情報

デフォルト値	1 (オン)
値の範囲	0 (オフ)、1 (オン)
ステータス	静的
表示レベル	包括
必要な役割	System Administrator (システム管理者)
設定グループ	ユーザ環境

ロギングでキューイング方式が使用されるかどうかを決定します。 **user log cache queue size** の設定値により、次のようになります。

- 1 - ユーザログキャッシュのキューイングを有効化します。 ユーザログキャッシュは複数のキャッシュレットに分けられます。キャッシュレットの数は、**user log cache size** の値に依存します。
- 0 - ユーザログキャッシュのキューイングを無効化します。 設定した **user log cache size** の値に関係なく、ユーザログキャッシュは複数のキャッシュレットに分けられません。

変更された設定パラメータ

SAP ASE バージョン 16.0 には、設定パラメータへの変更が含まれています。

allow nested triggers
複数トリガ機能は、**sp_configure "allow nested triggers"** 設定パラメータの動作を変更しません。

stack guard size
(UNIX プラットフォームのみ) stack guard size をデフォルト値以外の値に設定して、スタック保護領域の使用可能部分を増やす場合、追加の 4096 バイト分を含めることをおすすめします。

組み込み関数

SAP ASE バージョン 16.0 では、新しい組み込み関数が導入されました。

dbencryption_status

SAP ASE 16.0 では、新しい組み込み関数 **dbencryption_status()** が導入されてます。この関数は、暗号化/復号化のステータスおよびデータベースの進行状況をレポートすることでデータベースの完全暗号化機能をサポートします。

データベースの暗号化/復号化ステータスと進行状況をレポートします。 構文は次のとおりです。

```
dbencryption_status ('status'|'progess', dbid[,
    lstart])
```

構文の説明は次のとおりです。

- `status` は *dbid* で指定したデータベースの暗号化ステータスを返します。 `status` を使用するには、*dbid* を指定する必要があります。 戻り値は次のとおりです。
  - 0 - 通常のデータベースを示します。
  - 1 - データベースが暗号化されていることを示します。
  - 2 - データベースが暗号化中であることを示します。
  - 3 - データベースの一部が暗号化されている (暗号化処理の進行中ではない) ことを示します。
  - 4 - データベースの復号化が進行中であることを示します。
  - 5 - データベースの一部が復号化されている (復号化処理の進行中ではない) ことを示します。
- `progress` は、暗号化/復号化の進行状況を割合でレポートします。 次のように指定できます。
  - *dbid* - `progress` からデータベース全体に対する処理済みの割合が返されます。
  - *dbid* と *lstart* (論理開始ページ) の両方 - `progress` から *lstart* で指定されたフラグメントの処理済みページの割合が返されます。

  暗号化または復号化操作が行われていない場合、暗号化/復号化処理が終了している場合など、"`progress`" を使用しても SAP ASE で進行状況の情報を見つけられなかった場合、SAP ASE は "`-1`" を返します。
- *dbid* - データベース ID です。

参照：

- 暗号化データベースの作成 (69 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- 既存データベースの暗号化 (70 ページ)

- sp_helpdb (171 ページ)
- 暗号化処理のサスペンド (77 ページ)
- 暗号化処理の再開 (77 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

コマンド

SAP ASE バージョン 16.0 では、コマンドに変更が加えられています。

データベースの完全暗号化に使用される alter database

SAP ASE バージョン 16.0 では、**alter database** コマンドを使用してデータベースの暗号化と復号化を実行できます。

構文

```
alter database database_name
{[encrypt with key_name | decrypt [with key_name]] [parallel
degree_of_parallelism]
| resume [encryption | decryption [parallel degree_of_parallelism]]
| suspend [encryption | decryption]
}
```

パラメータ

- ***database_name*** – 暗号化または復号化するデータベースの名前です。
- **encrypt with *key_name* –** SAP ASE に対してデータベースを完全に暗号化するように指示します。

  具体的に言うと、このコマンドは `master` データベースの `sysencryptkeys` システムテーブルから対応するキー ID を取得し、関連付けられた `sysdatabases` ローの `encrkeyid` カラムを設定します。

key_name はデータベースの暗号化に使用したデータベース暗号化キーです。別のキー名を指定すると、コマンドが失敗して SAP ASE にエラーメッセージが表示されます。

すでにデータベースが以下の状態にある場合、SAP ASE は **alter database** の実行に失敗し、エラーメッセージを表示します。

- 別のキーによって暗号化されている。
- 暗号化中である。

暗号化が現在進行中ではなく、一部が暗号化されているデータベースに対してこのコマンドを実行した場合、キー名が以前指定したキーと同じであれば、`resume encryption` オプションを指定した場合と同様にコマンドが処理されます。

- **decrypt [with *key_name*]** – SAP ASE に対してデータベースを復号化するように指示します。 SAP ASE は、`sysdatabases` システムテーブル内のキー ID を検索するため、データベースの復号化の場合、`[with key_name]` はオプションです。 ただし、データベースの暗号化に使用されたものとは異なるキー名で *key_name* を指定すると、コマンドが失敗します。
- **resume decryption** – SAP ASE に対して前回復号化処理がサスペンドされたデータベースの復号化処理を再開するように指示します。 *database_name* がすでに完全に復号化されている場合、SAP ASE はこのコマンドを無視します。
- **parallel *degree_of_parallelism*** – タスクを開始するワーカスレッド数を決定します。

  数が "`number of worker processes`" の設定以下である場合は、データベース記憶領域仮想デバイスのそれぞれにスレッドを作成します。 ワーカスレッド数が追加されても暗号化パフォーマンスは向上しないため、*degree_of_parallelism* の数値をデータベースデバイスの数より大きくしないでください。 *degree_of_parallelism* を指定しない場合、オンラインエンジン数、および各デバイス間におけるデータベースの分散状況に応じて、SAP ASE によってこの値が内部的に定義されます。
- **resume encryption** – 暗号化が前回サスペンドされたページから暗号化処理を再開します。

  次の場合はコマンドが失敗します。

  - SAP ASE ですでに暗号化処理が実行中である。
  - 対象データベースで暗号化が開始されたことがない。
  - 暗号化処理がすでに完了している。

  `parallel` *`degree_of_parallelism`* は、`resume encrypt` とともに使用することができます。 `parallel` *`degree_of_parallelism`* を指定しない

場合、SAP ASE は各エンジン間でのデータベースの分散方法に従って値を決定します。

- **suspend encryption** – データを暗号化中のすべての暗号化ワーカスレッドを強制終了します。 SAP ASE は暗号化の進行状況を記録して、`resume encryption` が前回の暗号化処理の停止位置から暗号化を再開できるようにします。 進行中の暗号化が存在しない場合、SAP ASE はこのコマンドを無視します。

例

- **例 1 –** 次の例では、`dbkey` という暗号化キーを使用して、`existdb` という名前の既存データベースを暗号化するように変更します。
  ```
  alter database existdb encrypt with dbkey
  ```
  この例では、並列度を指定していないため、SAP ASE によって `existdb` の暗号化を並列的に開始するワーカスレッド数が決定されます。
- **例 2 –** 次の例は `existdb` というデータベースに対する暗号化処理をサスペンドします。
  ```
  alter database existdb suspend encryption
  ```
- **例 3 –** 次の例は、サスペンドされたデータベース `existdb` に対する暗号化処理を再開します。
  ```
  alter database existdb resume encryption
  ```

使用法

- データベースの暗号化は、データベースの実行中に行われます。 このため、暗号化の進行中も他のユーザからデータベースへのアクセスが可能です。シングルユーザモードにする必要はありません。
- 暗号化処理によってデータベースに対するユーザクエリ、更新、挿入の操作が中断されることはありません。
- データベースの暗号化はサスペンドと再開が可能であるため、SAP ASE の再起動後にデータベースの暗号化を再開することができます。
- 暗号化処理はトランザクション指向ではありません。
- アーカイブデータベースとテンポラリデータベースはいずれも暗号化と復号化を変更できます。
- SAP ASE によってデータベース暗号化の進行状況が記録され、そのステータスをレポートするユーティリティが提供されます。

次の場合はコマンドが失敗します。

- すでに暗号化されているデータベースに対する使用。
- 暗号化が進行中のデータベースに対する使用。 このコマンドを一部が暗号化されたデータベースに対して使用し、SAP ASE で暗号化処理が進行中ではない

場合、データベースの以前の暗号化のコマンドで使用されたデータベース暗号化キー名を使用すれば、このコマンドによって最後にサスペンドされた位置から暗号化が再開されます。

制限事項:

- `master`、および `model` のデータベースを暗号化することはできません。
- 暗号化が進行中のデータベースの復号化、または復号化が進行中のデータベースの暗号化はできません。
- 暗号化中にデータベースを削除またはマウント解除することはできません。
- 暗号化が進行中のデータベース上に別のデータベースをロードすることはできません。
- データベースの暗号化中にデータベースのバックアップ (ダンプ) を実行することはできません。

参照：

- 既存データベースの暗号化 (70 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- 暗号化処理のサスペンド (77 ページ)
- 暗号化処理の再開 (77 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)

alter index

index_compression 句を使用して、今後のインデックスの挿入または更新の圧縮状態を変更します。

構文

```
alter index [[database.][owner].table_name.index_name
set index_compression [= {none | page} ]
```

パラメータ変更
index_compression - ローカルインデックスパーティションの圧縮状態の変更は、このパーティションで新しく挿入または更新されたインデックスローにのみ影響します。

- NONE - 指定したインデックスのインデックスページは圧縮されません。 **index_compression = page** を指定して作成されたインデックスは圧縮されます。
- page - ページがいっぱいになると、ページプレフィクス圧縮を使用して既存のインデックスローが圧縮されます。 ローが追加されると、チェックが実行されて、そのローが圧縮に適しているかどうかが判別されます。

例
インデックス idx_char の圧縮状態を on に設定します。

```
alter index order_line. idx_char
    set index_compression = page
```

参照：
- 圧縮状態の変更 (60 ページ)
- インデックス圧縮用の alter table (123 ページ)

インデックス圧縮用の alter table

index_compression 句を使用して、今後のインデックスの挿入または更新の圧縮状態を変更します。

構文

```
alter table [[database.][owner].table_name
{add column_name datatype}
   [default {constant_expression | user | null}]
    {identity | null | not null}
    [off row | in row]
    [[constraint constraint_name]
       {{unique | primary key}
          [clustered | nonclustered] [asc | desc]
        [with {fillfactor = pct,
               max_rows_per_page = num_rows,
               reservepagegap = num_pages}]
        [on segment_name]
| references [[database.]owner.]ref_table
      [(ref_column)] [match full]
| check (search_condition)]
[encrypt [with key_name] [decrypt_default value]],
[[not] compressed]
[, next_column]...
| add {[constraint constraint_name]
        {unique | primary key}
           [clustered | nonclustered]
```

```
           (column_name [asc | desc]
        [, column_name [asc | desc]...])
           [with {fillfactor = pct,
                  max_rows_per_page = num_rows,
                  reservepagegap = num_pages}]
           [on segment_name]
| foreign key (column_name [{, column_name}...])
    references [[database.]owner.]ref_table
       [(ref_column [{, ref_column}...])]
  [match full]
| check (search_condition)}
| set {  [ dml_logging = {full | minimal | default}]
      | [,compression = {NONE | PAGE | ROW | ALL} ]
      | [,index_compression = {NONE | PAGE} ]
     }
| drop {column_name [, column_name]...
| constraint constraint_name}
| modify column_name
    [datatype [null | not null]]
    [[[encrypt [with key_name]
        [decrypt_default value]]
      | decrypt
    ]
    [[not] compressed ]
[, next_column]...
| replace column_name
     default {constant_expression | user | null}
| decrypt_default {constant_expression | null}
| drop decrypt_default
| lock {allpages | datarows | datapages}
| with exp_row_size=num_bytes
| partition number_of_partitions
| unpartition
| partition_clause
| alter_partition_clause


    alter_partition_clause ::=
{add_partition_clause
| drop_partition_clause
| modify partition partition_name
          [, partition_name ...]
  set compression [= {none | row | page | ALL} ]
  set index_compression [= {none | page} ]
```

パラメータ変更

index_compression - テーブル、インデックス、またはローカルインデックスパーティションに対してインデックス圧縮が有効または無効になるよう指定します。インデックスが圧縮されるようにテーブルを変更すると、新しく作成されるインデックスは圧縮されます。

- NONE - 指定したテーブルのインデックスは圧縮されません。

- PAGE - 指定したテーブルのすべてのインデックスが圧縮されます。

例
例 1

圧縮状態を NONE に変更して、既存のテーブル order_line を変更します。

```
alter table order_line set index_compress = NONE
```

例 2

ローカルインデックスパーティション Y2009 の圧縮の状態を PAGE に変更して、既存のテーブル sales を変更します。

```
alter table sales
modify partition Y2009 set index_compression = PAGE
```

参照：
- 圧縮状態の変更 (60 ページ)
- alter index (122 ページ)

複数のトリガに使用される alter table

1 つのテーブルに複数のトリガが存在する場合、テーブル所有者はそのテーブルで定義された複数トリガの一部またはすべてを無効にすることができます。

残存データの削除に使用される alter table

alter table コマンドは、SAP ASE の削除操作で残存データを削除する機能をサポートします。

構文
既存のテーブルでこれを指定する構文は次のとおりです。

```
alter table table_name
    set "erase residual data" {on | off}
```

使用法
テーブルに対してこのオプションを設定すると、残存データが発生するテーブル操作 (**drop table**、**delete row**、**alter table**、**drop index**) が実行されると、割り付け解除された領域が自動的にクリーンアップされます。 デフォルトは **off** に設定されています。

パーミッション
テーブル所有者、または **alter any table** パーミッションを持つユーザのみが "erase residual data" オプションを使用できます。

データベースの完全暗号化に使用される create archive database

create archive database コマンドはデータベースの完全暗号化機能をサポートします。

構文

```
create archive database database_name
    encrypt with key_name
```

パラメータ

- *database_name* は、作成するアーカイブデータベースの名前です。
- *key_name* は、バックアップ (ダンプ) したデータベースの暗号化に使用したキーです。 SAP ASE は、データベースのダンプの際に *key_name* が一致していることを検証します。 一致しない場合は、リストアが失敗します。

パーミッション

create archive database コマンドの `encrypt with` オプションの使用に必要なパーミッションは特にありません。 ただし、ユーザが *key_name* として参照できるようにするには、データベース暗号化キーに対する **select** パーミッションが必要です。

参照：

- 暗号化データベースの作成 (69 ページ)
- dbencryption_status (118 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

データベースの完全暗号化に使用される create database

create database コマンドを使用して完全に暗号化されたデータベースを作成できます。

作成時にデータベースを暗号化するか、また、このデータベースに挿入されるすべてのデータを自動的に暗号化するかを指定します。 暗号化してもデータベースのサイズは変わりません。また、記憶領域アクセス機能はデータベースが暗号化されていてもいなくても同様に機能します。 暗号化をサポートするデータベースのタイプは次のとおりです。

- 通常のユーザデータベース
- テンポラリデータベース
- アーカイブデータベース

インメモリデータベースを暗号化することはできません。

構文

```
create [temporary] database database_name
    encrypt with key_name
```

パラメータ

- ***database_name*** – 作成する暗号化データベースの名前です。
- ***key_name*** – データベース暗号化キーの名前です。

例

- **暗号化データベースのゼロからの作成** – マシン `demologdev` 上に暗号化データベース `demodb` を、ログ `demodev` とともに作成します。暗号化キーには、`dbkey` を使用します。

  ```
  create database demodb on demodev log on demologdev encrypt with
  dbkey
  ```

参照：

- 暗号化データベースの作成 (69 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)

- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)

create default

or replace 句を使用すると、**create default** を使用してデフォルトの定義を置き換えることができます。

構文
変更は太字で表示されています。

```
create [or replace] [owner.] default_name
as constant_expression
```

***create or replace default** のパラメータ変更*

- **create** - デフォルトが存在しない場合にデフォルトを作成します。
- **or replace** - デフォルトが既に存在する場合は、そのデフォルト定義を新しい定義に置き換えます。
- *default_name* - 指定したデフォルト名が既に存在する場合は、そのデフォルト名を新しいデフォルト定義に置き換えます。 オブジェクトの名前と ID は変わりません。
- *constant_expression* - デフォルトを置き換えるときにデフォルトの定義を変更できます。 新しいデフォルト値は、古いデフォルトを上書きします。

例
この例では、UNKNOWN として定義されている電話番号でデフォルトを作成します。

```
create default phonedflt as "UNKNOWN"

select object_id("phonedflt")
-----------
1001051571
```

or replace 句を使用して、前に作成したデフォルトがこのデフォルトに置き換えられます。 電話番号は変更されますが、デフォルトのオブジェクト ID は変わりません。

```
create or replace default phonedflt as "999-999-9999"

select object_id("phonedflt")
-----------
1001051571
```

置き換えられたデフォルトに依存するオブジェクト

- 置き換えられたデフォルトに多数のカラムをバインドできます。
- 置き換えられたデフォルトにユーザ定義データ型をバインドできます。

これらのカラムにアクセスするプロシージャは、デフォルトが置き換えられてそのプロシージャが実行されるときに再コンパイルされます。

***create or replace default** のパーミッション変更*
エイリアスまたは **setuser** を使用してデフォルト所有者と同一化したユーザはデフォルトを置き換えることはできません。

デフォルトを置き換えるための変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、`create default` 権限が必要。別のユーザのデフォルトを作成するには、`create any default` 権限が必要。 **デフォルトを置き換えるにはデフォルト所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、**sa_role** が付与されている、または `create default` 権限を持つユーザであることが必要。 **デフォルトを置き換えるにはデフォルト所有者であることが必要。**

***create or replace default** の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
14	create	create default	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create encryption key

create encryption key コマンドは、データベースの完全暗号化機能をサポートします。

データベース暗号化キーは 256 ビットの対称キーで、master データベースで作成され、データベースの暗号化に使用されます。

構文

```
create encryption key keyname
    [for algorithm]
    for database encryption
    [with
        {[master key]
        [key_length 256]
        [init_vector random]
        [[no] dual_control]}
```

パラメータ

- ***keyname*** – 現在のデータベース内のユーザのテーブル、ビュー、プロシージャネームスペース内でユニークである必要があります。 キーが別のデータベース内にある場合はデータベース名を指定し、データベース内にその名前のキーが複数ある場合は所有者名を指定します。 所有者のデフォルト値は現在のユーザで、デフォルト値は現在のデータベースです。 他のユーザのキーを作成できるのは、システムセキュリティ担当者だけです。
- ***algorithm*** – アルゴリズムを指定します。 サポートされる唯一のアルゴリズムは Advanced Encryption Standard (AES) です。 AES では、128、192、256 ビットのキーサイズ、および 16 バイトのブロックサイズがサポートされています。 ブロックサイズは、暗号化ユニットのバイト数です。 大きいデータは、分割して暗号化されます。
- **for database encryption** – カラムではなくデータベース全体を暗号化する暗号化キーの作成であることを指定します。
- **master key** – master データベース内にマスタキーを作成し、SAP ASE に対してそのキーを使用してデータベース暗号化キーを保護するように指示します。 デフォルトでは、SAP ASE はこのマスタキー(存在する場合)を使用してカラム暗号化キーを保護します。
- **key_length 256** – 作成するキーのビット単位のサイズです。 データベース暗号化キーの有効長は 256 のみです。これ以外のサイズを使用するとエラーメッセージが表示されます。
- **init_vector random** – 暗号化中に初期化ベクトルを使用するように指定します。 暗号化アルゴリズムで初期化ベクトルが使用される場合、2 つの同一のプレーンテキストの暗号化テキストが異なるものになり、これによって、

データパターンの検出を防止できます。 初期化ベクトルを使用すると、データのセキュリティが強化されます。 データベースの暗号化はカラムの暗号化と比較してセキュリティが強化されます。カラムの暗号化キーの作成時に使用可能な `init_vector null` を指定すると、SAP ASE によってエラーが返されます。

- **`[no] dual control`** – デュアルコントロールを使用して新しいキーを暗号化する必要があるかどうかを示します。 デフォルトでは、デュアルコントロールは設定されていません。 `dual control` を使用するには、`master` データベースにマスタキーとデュアルマスタキーの両方が存在している必要があります。

例

- **例 1 –** "testkey" を保護するマスタキーを作成します。

```
create encryption key testkey for database encryption
    with master key
```

- **例 2 –** "testkey" を保護するデュアルマスタキーを作成します。

```
create encryption key testkey for database encryption
    with dual_control
```

- **例 3 –** "testkey" を保護するマスタキーとデュアルマスタキーの両方を作成します。

```
create encryption key testkey for database encryption
    with master key dual_control
```

- **例 4 –** デュアルマスタキーを明示的に除外して、"testkey" を保護するマスタキーを作成します。

```
create encryption key testkey for database encryption
    with master key no dual_control
```

- **例 5 –** デュアルマスタキーを明示的に除外して、"testkey" を保護するマスタキーを作成します。

```
create encryption key testkey for database encryption
    with no dual control
```

- **例 6 –**

```
sp_configure 'enable encrypted columns', 1
create encryption key master with passwd "testpassword"
set encryption passwd 'testpassword' for key master
create encryption key dbkey for database encryption
```

使用法

- データベース暗号化キーは、**create encryption key** コマンドの `pad` オプションをサポートしません。

- データベース暗号化キーをカラム暗号化のデフォルトキーにすることはできません。
- 適正に作成された暗号化キーは、`master` データベースの `sysencryptkeys` テーブルに格納され、次のキータイプによって指定されます。

```
#define EK_DBENCKEY       0x1000
```

標準

ANSI SQL - 準拠レベル: Transact-SQL 拡張機能

パーミッション

create encryption key のパーミッションチェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	SAP ASE によって "`manage database encryption key`" という新しいパーミッションが作成される。 データベース暗号化キー作成のためのパーミッションが必要。
細密なパーミッションが無効	sso_role、keycustodian_role が付与されたユーザであるか、**create encryption key** 権限が必要。

参照：

- drop encryption key (159 ページ)
- データベース暗号化キーの変更 (67 ページ)
- データベース暗号化キーの削除 (68 ページ)
- データベース暗号化キーのバックアップ (79 ページ)
- データベース暗号化キーの作成 (66 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)

create function

or replace 句を使用すると、**create function** を使用してユーザ定義関数の定義を置き換えることができます。

構文
変更は太字で表示されています。

```
create [or replace] function
    [ owner_name.] function_name
    [ ( @parameter_name [as] parameter_datatype
    [ = default ][ ,...n ] ) ]
    returns  return_datatype
    [ with recompile]
    as
    [begin]
    function_body
    return scalar_expression
    [end]
```

***create or replace function** のパラメータ変更*

- **create** - ビューが存在しない場合は関数を作成します。
- **or replace** - 既存の関数を再定義します。 この句を使用して、関数に対して以前に付与したオブジェクト権限の削除、再作成、および再付与を行わずに、既存のユーザ定義関数の定義を変更します。 関数が再定義されると、その関数の使用時に再コンパイルされます。
- *function_name* - 定義は変更されますが、関数の名前は変わりません。
- **@parameter_name** - 関数の定義の置き換え時に、パラメータの名前と数を変更できます。
- *parameter_datatype* - 関数に対するパラメータのデータ型を変更できます。
- **with recompile** - 関数が置き換えられるたびに、再コンパイルするか、しないかのオプションを変更できます。
- *return_datatype* - 関数のリターンデータ型を変更できます。
- *scalar_expression* - 関数から返される値を変更できます。
- *function_body* - 関数の値を定義する SQL 文を変更できます。

例
この例は、文字列 `firstname` と `lastname` を連結する関数を定義します。

```
create function fullname(
    @firstname char(30),
    @lastname char(30))
returns char(61)
as
begin
declare @name char(61)
```

```
set @name = @firstname|| ' ' ||@lastname
return @name
end

select object_id("fullname")
-----------
473049690
```

この関数は、前に作成した `fullname` 関数を、**or replace** 句を使用して置き換えます。 関数の置き換え後にローカル変数 `@name` が削除されます。 関数のオブジェクト ID は変わりません。

```
create or replace function fullname(
    @firstname char(30),
    @lastname char(30))
returns char(61)
as
begin
return(@firstname|| ' ' ||@lastname)
end

select object_id("fullname")
-----------
473049690
```

置き換えられた関数に依存するオブジェクト
置き換えられた関数が別の関数によって呼び出される場合、呼び出し時に両方の関数が再コンパイルされます。 置き換えられた関数のインタフェースが呼び出し元関数と一致しない場合は、呼び出し元関数を置き換える必要があります。置き換えないと呼び出し元関数でエラーが発生します。 置き換えられた関数に対して **sp_depends** を実行して、呼び出し元オブジェクトが存在するかをチェックします。

たとえば、`testfun1` が置き換えられ、設定されるパラメータが 1 つではなく 2 つになっています。 2 つ目のパラメータの処理を可能にするため、呼び出し元関数 `testfun2` を置き換える必要があります。

```
create function testfun1 (@para1 int)
    returns int
    as
    begin
        declare @retval int
        set @retval = @para1
        return @retval
    end
create function testfun2 (@para int)
    returns int
    as
    begin
        declare @retval int
        select @retval= dbo.testfun1 (@para)
        return @retval
```

```
    end
create or replace function testfun1 (@para1 int,@para2 int)
    returns int
    as
    begin
        declare @retval int
        set @retval = @para1+@para2
        return @retval
    end
```

制限事項
関数が計算カラムまたは機能インデックスで参照される場合は、その関数を置き換えることはできません。

***create or replace function** のパーミッション変更*
エイリアスまたは **setuser** を使用して関数所有者と同一化したユーザは関数を置き換えることはできません。

関数の置き換えに関する変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、`create function` 権限が必要。 他のユーザに対して **create function** を実行するには、`create any function` 権限が必要。 **関数を置き換えるには、関数所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、`create function` 権限が必要。 **関数を置き換えるには、関数所有者であることが必要。**

***create or replace function** の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
97	create	create function	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create function (SQLJ)

or replace 句を使用すると、**create function** を使用してユーザ定義 SQLJ 関数の定義を置き換えることができます。

構文
変更は太字で表示されています。

```
create [or replace] function
    [owner_name.]sql_function_name
        ([ sql_parameter_name sql_datatype
            [(length)| (precision[, scale ])]
        [[,sql_parameter_name sql_datatype
            [(length)| (precision[, scale])]]
        ...]])
    returns sql_datatype
    [(length)| (precision[, scale])]
    [modifies sql data]
    [returns null on null input |    called on null input]
    [deterministic | not deterministic]
    [exportable]
    language java
    parameter style java
    external name 'java_method_name
        [([java_datatype[, java_datatype
        ...]])] '
```

create or replace function (SQLJ) のパラメータ変更

- **create** - SQLJ 関数が存在しない場合に SQLJ 関数を作成します。
- **or replace** - 既存の関数を再定義します。 この句を使用して、関数に対して以前に付与したオブジェクト権限の削除、再作成、および再付与を行わずに、既存のユーザ定義 SQLJ 関数の定義を変更します。
- *sql_function_name* - 定義は変更されますが、関数の名前は変わりません。
- **sql_parameter_name** - 関数の定義を置き換えるときにパラメータの名前と数を変更できます。
- *sql_datatype* - 関数へのパラメータの Transact-SQL データ型を変更できます。
- **returns** *sql_datatype* - 関数の結果データ型を変更できます。
- **external** - 外部ルーチンの名前を変更できます。
- *java_method_name* - Java メソッド名を変更できます。
- *java_datatype* - Java データ型を変更できます。

例
この例では、`sqlj_testfun` という名前の SQLJ 関数を作成します。

```
create function sqlj_testfun (p1 int)
                      returns int
                      language java
```

```
                        parameter style java
                        external name 'UDFSample.sample(int)'
```

次の例では、**or replace** 句を使用して、前に作成した SQLJ 関数を置き換えます。パラメータ p2 を追加して、外部 java メソッドを変更しますが SQLJ 関数のオブジェクト ID は変わりません。

```
create or replace function sqlj_testfun (p1 int,p2 int)
                        returns int
                        language java
                        parameter style java
                        external name 'UDFSample.sample2(int,int)'
```

置き換えられた関数に依存するオブジェクト
置き換えられた SQLJ 関数が別の関数によって呼び出される場合、呼び出し時に両方の関数が再コンパイルされます。

置き換えられた関数のインタフェースが呼び出し側関数のインタフェースと一致しない場合、呼び出し側関数を置き換える必要があります。そのようにしないと、呼び出し側関数でエラーが発生します。 置き換えられた関数に対して **sp_depends** を実行して、呼び出し側オブジェクトがないかチェックします。

制限事項
関数が計算カラムまたは機能インデックスで参照される場合は、その関数を置き換えることはできません。

***create or replace function** (SQLJ) のパーミッション変更*
エイリアスまたは **setuser** を使用して関数所有者と同一化したユーザは関数を置き換えることはできません。

関数を置き換えるための変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、create function 権限が必要。 他のユーザに対して **create function** を実行するには、create any function 権限が必要。 **関数を置き換えるには関数所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、create function 権限が必要。 **関数を置き換えるには関数所有者であることが必要。**

***create or replace function** (SQLJ) の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
97	create	create function	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create index

index_compression 句を使用して、インデックスまたはインデックスパーティションを圧縮できます。

構文

```
create [unique] [clustered | nonclustered] index index_name
    on [[database.]owner.]table_name
    (column_expression [asc | desc]
    [, column_expression [asc | desc]]...)
    [with {fillfactor = pct,
    index_compression = { NONE | PAGE },
    max_rows_per_page = num_rows,
    reservepagegap = num_pages,
    consumers = x, ignore_dup_key, sorted_data,
    [ignore_dup_row | allow_dup_row],
    statistics using num_steps values}]
    [on segment_name]
[index_partition_clause]
Syntax to create index partitions
index_partition_clause::=
[local index
[partition_name [on segment_name]
[with index_compression = { NONE | PAGE }]
[, partition_name [on segment_name]
[with index_compression = { NONE | PAGE }]...]]]
```

パラメータ変更

index_compression

- NONE - 指定したインデックスのインデックスページは圧縮されません。
  index_compression = PAGE を指定して作成されたインデックスは圧縮されます。

- PAGE - ページがいっぱいになると、ページプレフィクス圧縮を使用して既存のインデックスローが圧縮されます。 ローが追加されると、チェックによってそのローが圧縮に適しているかどうかが判別されます。

例
例 1

ol_delivery_d カラムと ol_dist_info カラムに idx_order_line という圧縮インデックスを作成します。

```
create index idx_order_line
    on order_line (ol_delivery_d, ol_dist_info)
with index_compression = page
```

インデックスのインデックスローの長さが短すぎて圧縮するメリットがない場合は、そのインデックスが圧縮されないことを示す警告が表示されます。

例 2

idx_Sales という圧縮インデックスを作成します。 このインデックスには圧縮できるローカルインデックスパーティションが含まれています。 インデックスプレフィクス圧縮がローカルインデックスパーティションに適用されます。 インデックスページがいっぱいになっている間は、ページプレフィクス圧縮が適用されます。

```
create index idx_sales
    on Sales(store_id, order_num)
    local index ip1 with index_compression = PAGE,
    ip2 with index_compression = PAGE,
ip3
```

参照：
- 圧縮インデックスの作成 (59 ページ)

create procedure

or replace 句を使用すると、**create procedure** を使用してプロシージャの定義を置き換えることができます。

置き換えられたプロシージャに対して以前に付与した権限は保持されます。

構文
変更は太字で表示されています。

```
create [or replace] procedure

    [owner.]procedure_name[;number]
    [[(@parameter_name datatype [(length)
    (precision[,scale])]
    [= default][output]…)]]
```

```
[with {recompile | execute as {owner | caller}} ]
as {SQL_statements | external name dll_name}
```

create or replace procedure のパラメータ変更

- **create** - **create** のみを指定した場合は新しいプロシージャが作成されます。
- **or replace** - 指定したプロシージャが存在しない場合は、新しいプロシージャが作成されます。 プロシージャが存在しない場合はプロシージャ定義が変更されます。既存のパーミッション、監査オプション、および複写属性は維持されます。
- *procedure_name* - プロシージャが置き換えられる場合、個別のすべてのカタログで、プロシージャの名前は同じままになり、プロシージャのオブジェクト識別子も同じままになります。
- *number* - **or replace** が指定されず、プロシージャが既存の場合は、SAP ASE でこのグループ番号でプロシージャがすでに作成されていることを示すエラーが発生します。この場合は別のグループ番号を指定してプロシージャを作成する必要があります。 **or replace** 句が指定され、グループ番号が指定されず、別のグループ番号でこのプロシージャが存在する場合は、プロシージャがグループの一部であり置き換えられないため、エラーが発生します。 グループ番号を指定してそのグループ番号の既存プロシージャを置き換えることはできません。プロシージャの削除についても同様で、グループ内の個別のプロシージャは削除できなません。
- *parameter_name* - プロシージャ定義を置き換えるときは、名前とパラメータの数を変更できます。
- *datatype* [(*length*) (*precision*[,*scale*])] - パラメータの型、長さ、精度、および位取りを変更できます。
- *default* - プロシージャを置き換えるときに、パラメータのデフォルトを NULL に変更するか、別の値を設定できます。
- **output** - プロシージャのリターンパラメータを変更できます。
- **with recompile** - このオプションを指定して既存のパラメータが作成されている場合に、**or replace** 句を使用して変更し、プロシージャの実行ごとに SAP ASE で新しいプランが作成されないようにすることができます。 既存のプロシージャの作成時に **with recompile** が指定されていない場合は、新しい定義に置き換えて、プロシージャの実行ごとにプランが作成されるようにすることができます。
- **with execute as** - 既存プロシージャの **with execute as** 句を所有者から呼び出し元に変更できます。また、その逆の変更についても同様です。 **or replace** 句を使用すると、**with execute as** 句なしでプロシージャを再作成することもできます。
- *SQL_statements* - プロシージャの本文を変更して、既存プロシージャと異なる文を組み込むことができます。
- **external name** - 拡張ストアドプロシージャも置き換えることができます。

- *dll_name* - 拡張ストアドプロシージャを実装する動的リンクライブラリの名前を変更できます。

例
この例は、次のように定義される製品に関する情報のテーブルに基づいています。

```
create table Products (
    ProductID int,
    ProductName varchar(30),
    Discontinued varchar(10))

create procedure ProductType
    @product_ID int,
    @type char(10) output
as
declare @prod_name char(20)
select @prod_name = ProductName, @type =
    case @prod_name
        when 'Tee Shirt' then 'Shirt'
        when 'Sweatshirt' then 'Shirt'
        when 'Baseball Cap' then 'Hat'
        when 'Visor' then 'Hat'
        when 'Shorts' then 'Shorts'
        else 'UNKNOWN'
    end
from Products
where ProductID = @product_ID

select object_id("ProductType")
-----------
425049519
```

次のコマンドは、**or replace** 句を使用して `ProductType` プロシージャを置き換えます。 `Tee Shirt` と `Sweatshirt` のパラメータは更新されますが、プロシージャのオブジェクト ID は変わらずに保持されます。

```
create or replace procedure ProductType
    @product_ID int,
    @type char(10) output
as
declare @prod_name char(20)
select @prod_name = ProductName, @type =
    case @prod_name
        when 'Tee Shirt' then ' T Shirt'
        when 'Sweatshirt' then 'Long Sleeve Shirt'
        when 'Baseball Cap' then 'Hat'
        when 'Visor' then 'Hat'
        when 'Shorts' then 'Shorts'
        else 'UNKNOWN'
    end
from Products
where ProductID = @product_ID

select object_id("ProductType")
```

```
-----------
425049519
```

置き換えられたプロシージャに依存するオブジェクト
置き換えられたプロシージャを呼び出すプロシージャは、実行時に再コンパイルされます。 プロシージャの置き換えによってパラメータの数または型が変更される場合は、呼び出し元プロシージャを置き換える必要があります。 置き換えられたプロシージャに対して **sp_depends** を実行して、変更された定義の影響を受ける呼び出し元プロシージャが存在するかどうかを確認できます。

***create or replace procedure** のパーミッション変更*
エイリアスまたは **setuser** を使用してプロシージャ所有者と同一化したユーザはプロシージャを置き換えることはできません。

プロシージャの置き換えに関する変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合は、`create procedure` 権限が必要。 他のユーザに対して **create procedure** を実行するには、`create any procedure` 権限が必要。 **プロシージャを置き換えるにはプロシージャ所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、`create procedure` 権限が必要。 **プロシージャを置き換えるにはプロシージャ所有者であることが必要。**

***create or replace procedure** の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
11	create	create procedure	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create procedure (SQLJ)

or replace 句を使用すると、**create procedure** を使用して SQLJ プロシージャの定義を置き換えることができます。

置き換えられたプロシージャに対して以前に付与した権限は保持されます。

構文
変更は太字で表示されています。

```
create [or replace] procedure

    [owner_name.]sql_procedure_name
        ([[in | out | inout]  sql_parameter_name
        sql_datatype
            [(length)| (precision[, scale ])]
            [=default]
        ...])
        [,[in | out | inout]  sql_parameter_name
        sql_datatype
            [(length)| (precision[, scale])]]
            [=default]
        ...])
    [modifies sql data]
    [dynamic result sets integer]
    [deterministic | not deterministic]
    language java
    parameter style java
    external name 'java_method_name
        [([java_datatype[, java_datatype
        ...]])] '
```

***create or replace procedure** (SQlJ) のパラメータ変更*

- **create** - **create** のみを指定した場合、新しい SQLJ プロシージャが作成されます。
- **or replace** - 指定した SQLJ プロシージャが存在しない場合、新しい SQLJ プロシージャが作成されます。 SQLJ プロシージャが存在する場合は、定義が変更されます。
- *sql_procedure_name* - パラメータの名前と数を変更できます。
- **in** | **out** | **inout** - リストされたパラメータのモードを変更できます。
- *sql_datatype* [(*length*) (*precision*[,*scale*])] - パラメータの型、長さ、精度、および位取りを変更できます。
- *default* - プロシージャを変更するときに、パラメータのデフォルトを NULL に変更するか、別の値を設定することができます。
- **deterministic** | **not deterministic** - deterministic 値を変更できます。
- **external** - 外部ルーチンの名前を変更できます。
- *java_method_name* - Java メソッド名を変更できます。

- *java_datatype* - Java データ型を変更できます。

例
この例では、proc_name という名前の SQLJ プロシージャを作成します。

```
create procedure sqlj_proc (param int)
                             language java
                             parameter style java
                      external name 'UDFSample.sample(int)'
```

このプロシージャは、**or replace** 句を使用して、前に作成した SQLJ プロシージャを置き換えます。 パラメータ p2 を追加して、外部 java メソッドを変更しますが SQLJ プロシージャのオブジェクト ID は変わりません。

```
create or replace procedure sqlj_proc (p1 int, p2 int)
                             language java
                             parameter style java
                      external name 'UDFSample.add(int,int)'
```

置き換えられたプロシージャに依存するオブジェクト
置き換えられた SQLJ プロシージャを呼び出す Transact-SQL プロシージャは、実行時に再コンパイルされます。 SQLJ プロシージャの置き換えによってパラメータの数または型が変更される場合は、呼び出し側プロシージャを置き換える必要があります。 置き換えられたプロシージャに対して **sp_depends** を実行して、変更された定義の影響を受ける呼び出し側プロシージャがあるかどうかを確認できます。

***create or replace procedure** (SQLJ) のパーミッション変更*
エイリアスまたは **setuser** を使用してプロシージャ所有者と同一化したユーザはプロシージャを置き換えることはできません。

SQLJ プロシージャを置き換えるための変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、create procedure 権限が必要。 他のユーザに対して **create procedure** を実行するには、create any procedure 権限が必要。 **プロシージャを置き換えるにはプロシージャ所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、create database procedure 権限が必要。 **プロシージャを置き換えるにはプロシージャ所有者であることが必要。**

***create or replace procedure** (SQLJ) の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
11	create	create procedure	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create rule

or replace 句を使用すると、**create rule** を使用してルールの定義を置き換えることができます。

構文
変更は太字で表示されています。

```
create [or replace] [[and | or] access] rule
    [owner.]rule_name
    as condition_expression
```

***create or replace rule** のパラメータ変更*

- **create** - ルールが存在しない場合にルールを作成します。
- **or replace** - ルールが既に存在する場合は、そのルール定義を新しい定義に置き換えます。
- **access** - アクセスルールを “and” ルールから “or” ルールに変更できます (その逆の変更も同様です)。 アクセスルールを同じ名前のドメインルールに置き換えることはできません (その逆の置き換えも同様です)。
- *rule_name* - 指定したルール名が既に存在する場合、そのルールは新しいルール定義に置き換えられますが、名前は保持されます。
- *constant_expression* - ルールを置き換えるときにルールの定義を変更できます。新しいルール値は、古いルール値を上書きします。

例
例 1

この例では、advance の値を 1000 ドルに制限する、`limit` という名前のルールを作成します。

```
 create rule limit
    as @advance < $1000

select object_id("limit")
 -----------
1017051628
```

次のコマンドで、作成したルールを置き換えます。 制限は、**or replace** 句を使用して変更されます。 ルールのオブジェクト ID は変わりません。

```
 create or replace rule limit
    as @advance < $2000

select object_id("limit")
 -----------
1017051628
```

例 2

テーブル所有者が `uname_acc_rule` という名前の **AND access rule** を作成します。

```
create access rule uname_acc_rule
as @username = suser_name()

select object_id("uname_acc_rule")

-----------
1033051685
```

`uname_acc_rule` を **OR access rule** に置き換えます。

```
create or replace or access rule uname_acc_rule
as @username = suser_name()

select object_id("uname_acc_rule")

-----------
1033051685
```

置き換えられたルールに依存するオブジェクト

- 置き換えられたルールに多数のテーブルのカラムをバインドできます。
- 置き換えられたルールにユーザ定義データ型をバインドできます。

これらのカラムにアクセスするプロシージャは、ルールが置き換えられてそのプロシージャが実行されるときに再コンパイルされます。

create or replace rule *のパーミッション変更*

エイリアスまたは **setuser** を使用してルール所有者と同一化したユーザはルールを置き換えることはできません。

ルールを置き換えるための変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、`create rule` 権限が必要。 他のユーザに対して **create rule** を使用するには、`create any rule` 権限が必要。 **ルールを置き換えるにはルール所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、`create rule` 権限を保持しているか、データベース所有者であるか、sa_role が付与されたユーザであることが必要。 他のユーザに対して **create rule** を使用するには、sa_role が付与されたユーザであることが必要。 **ルールを置き換えるにはルール所有者であることが必要。**

***create or replace rule** の監査変更*

変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
13	create	create rule	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

インデックス圧縮用の create table

index_compression 句を使用して、指定したテーブルのインデックスを圧縮できます。

構文

```
create table [database.[owner].]table_name
(column_name datatype
[default {constant_expression | user | null}]
{[{identity | null | not null}]
[off row | [in row [(size_in_bytes)]]
[[constraint constraint_name]
{{unique | primary key}
[clustered | nonclustered] [asc | desc]
[with {fillfactor = pct,
max_rows_per_page = num_rows,}
reservepagegap = num_pages}]
[on segment_name]
| references [[database.]owner.]ref_table
```

```
[(ref_column )]
[match full]
| check (search_condition)}]}
[match full]...
[encrypt [with key_name]
    [decrypt_default constant_expression | null]]
[[constraint [[database.[owner].]key_name]
    {unique | primary key}
        [clustered | nonclustered]
        (column_name [asc | desc]
        [{, column_name [asc | desc]}...])
        [with {fillfactor = pct
             max_rows_per_page = num_rows,
            reservepagegap = num_pages}]
    [on segment_name]
| foreign key (column_name [{,column_name}...])
references [[database.]owner.]ref_table
[(ref_column [{, ref_column}...])]
[match full]
| check (search_condition) ...}
 [{, {next_column | next_constraint}}...]
 [lock {datarows | datapages | allpages}]
[with {max_rows_per_page = num_rows,
    exp_row_size = num_bytes,
    reservepagegap = num_pages,
    identity_gap = value,
    transfer table [on | off],
    compression [={NONE | ROW | PAGE}],
    index_compression [={NONE | PAGE} ]
      }
]
[on segment_name]
     [[ external table ] at pathname ]
[partition_clause]
```

パラメータ変更
index_compression

- NONE - 指定したテーブルのインデックスは圧縮されません。
  index_compression = PAGE を指定して作成されたインデックスは圧縮されます。
- PAGE - 指定したテーブルのすべてのインデックスが圧縮されます。
  index_compression = NONE を指定して作成されたインデックスは圧縮されません。

例
この例では、`ol_delivery_d` カラムと `ol_dist_info` カラムを持ち、ページレベルの圧縮を使用するインデックス圧縮テーブル `order_line` を作成します。

```
create table order_line (
    ol_o_id         int,
    ol_d_id         tinyint,
    ol_w_id         smallint,
```

```
    ol_number       tinyint,
    ol_i_id         int,
    ol_supply_w_id  smallint,
    ol_delivery_d   datetime,
    ol_quantity     smallint,
    ol_amount       float,
    ol_dist_info    char(24) )
lock datapages
    with index_compression = page
```

デフォルトでは、このテーブルに作成されたインデックスは圧縮されます。 ただし、インデックスのインデックスローの長さが短すぎて圧縮するメリットがない場合は、そのインデックスが圧縮されないことを示す警告が表示されます。

参照：

- インデックス圧縮テーブルの作成 (59 ページ)
- select into (163 ページ)

残存データの削除に使用される create table

create table コマンドは、SAP ASE の削除操作で残存データを削除する機能をサポートします。

構文
新規テーブルでこれを指定する構文は次のとおりです。

```
create table table_name
    with "erase residual data" {on | off}
```

例
以下の例では次の 2 つのテーブルを使用します。

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

例 1

データベースレベルで設定されているため、テーブル `t1` では残存データを消去するオプションがオンであり、`t1` に対して実行される **drop table** コマンドと **truncate table** コマンドは両方とも、ページのすべての残存データをクリーンアップすることになります。

一方、テーブル `t2` は "**erase residual data off**" 句を指定して作成されたため、**erase residual data** オプションが明示的にオフに設定されています。 データベースレベルで "`erase residual data`" オプションが `true` に設定されていても、残存データは削除されません。 この結果、`t2` に対して **drop table** および **truncate table** を実行した後も、残存データが保持されます。

```
create database db1
go
```

```
sp_dboption db1, "erase residual data", true
go
use db1
go
create table t1 (col int)
go
insert t1 values ...
go
create table t2 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
drop table t2
go
```

例 2

この例では次のようになります。

- テーブル `t1` には "`erase residual data off`" が明示的に設定されてはいませんが、データベースレベルで設定されているため、**truncate table t1** を実行すると `t1` から残存データが削除されることになります。
- テーブル `t2` は、このオプションがデータベースレベルで設定されているため、作成時に "`erase residual data`" オプションが設定されています。 このため、**truncate table t2** を実行すると、`t2` から残存データが削除されます。
- テーブル `t3` には "`erase residual data off`" が明示的に指定されています。このため **sp_dboption** で "`erase residual data`" が true に設定され、SAP ASE が **truncate table t3** を実行しても、残存データは削除されません。

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
truncate table t2
go
```

```
truncate table t3
go
```

例 3

この例では次のようになります。

- t1 テーブルと t2 テーブルは両方ともデフォルトでは "erase residual data" オプションが設定されていませんが、**truncate table** コマンドの実行直前にセッションレベルで "erase_residual_data" がオンになったため、t1 と t2 の両方から残存データが削除されます。
- テーブル t3 は "erase residual data" オプションが明示的に off に設定されていますが、セッションレベルで "erase_residual_data" オプションが設定されているため、この場合も **truncate** コマンドが実行されると、残存データは削除されます。

```
create database db1
go
use db1
go
create table t1(col int)
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
set erase_residual_data on
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

使用法
テーブルに対してこのオプションを設定すると、残存データが発生するテーブル操作 (**drop table**、**delete row**、**alter table**、**drop index**) が実行されると、割り付け解除された領域が自動的にクリーンアップされます。

パーミッション
テーブル所有者、または **create any table** パーミッションを持つユーザのみが "**erase residual data**" オプションを使用できます。

参照：

- set (163 ページ)
- sp_dboption (167 ページ)

複数のトリガで使用される create trigger

create trigger コマンドを使用して複数のトリガを作成し、新しい `order` パラメータでトリガを起動する順序を指定します。

構文

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    {for | instead of} {insert | update | delete}
    [order integer]
    as sql_statements
```

パラメータ

`order integer` は、トリガ機能の全順序または順序の一部を指定します。

- `order` 句を使用してすべてのトリガを作成すると、全順序の指定が行われます。
- 部分順序指定は、トリガの一部で `order` 句を指定しない場合に行われます。`order` 句がないトリガは、暗黙的に順序番号 0 を取得し、`order` 句を使用して作成されたトリガの後に起動されることを除き、順序は定義されません。

使用法

注意： `order integer` 句は、`for {insert | update | delete}` でのみ使用することができます。`instead of {insert | update | delete}` トリガで使用することはできません。

`order` で指定する数値に重複があると、SAP ASE でエラーがレポートされます。`order` の番号が連続している必要はありません。むしろ順序の途中に新しいトリガを挿入できるため、非連続の番号のほうが望ましいです。

参照：

- 複数のトリガの作成 (95 ページ)

create trigger for or replace

or replace 句を使用すると、**create trigger** を使用してトリガの定義を置き換えることができます。

SAP ASE 16.0 より前のバージョンでは、**create trigger** コマンドを連続すると、古いトリガが削除され、新しいトリガ定義に置き換えられていました。 ただし、トリガの監査オプションも削除されていました。 オプションの **or replace** 句を使用すると、定義が置き換えられ、監査オプションが維持されます。

構文
変更は太字で表示されています。

```
create [or replace] trigger [owner.]trigger_name
    on [owner.]table_name
    {for {insert , update}
        | instead of {insert, update, delete}}
     [order integer]
     [as
        [if update (column_name)
        [{and | or} update (column_name)]...]
            SQL_statements] ...]
        [if update (column_name)
        [{and | or} update (column_name)]...
            SQL_statements]...]
```

***create or replace trigger** のパラメータ変更*
create - トリガが存在しない場合はトリガを作成します。

- SAP ASE バージョン 16.0 以降では、複数のトリガが存在するときに **or replace** を指定しないで **create** を指定したときに、トリガ名が同一であるとエラーが発生します。 別のトリガ名を指定した場合は、新しいトリガが作成され、古いトリガも保持されます。 次のセクションも参照してください。 複数のトリガ
- 16.0 より前のバージョンの SAP ASE では、create without or replace を指定して既存のトリガを新しいトリガに置き換える場合に、新しいトリガの名前は古いトリガの名前と同じでなくてもかまいませんでした。

- **or replace** - 既存のトリガを再作成します。 この句は、トリガの定義の変更に使用します。 **or replace** を指定すると、トリガの監査オプションは削除されません。 入力した名前で既存のトリガがない場合、新しいトリガが作成され、古いトリガも保持されます。 これは複数のトリガと関連します。
- *trigger_name* - トリガ定義が置き換えられるときにトリガの名前は変更されません。 新しいトリガ定義の名前は、置き換え対象の古い名前と一致している必要があります。 トリガ名が既存トリガ名と異なる場合は、新しいトリガが作成され、古いトリガは削除されません。
- *table_name* - トリガを置き換えるときにテーブルの名前は変更できません。 既存トリガを変更して別のテーブルに関連付けようとすると、そのテーブルにはトリガがすでに存在し、置き換えることはできないとのエラーメッセージが示されます。
- **for | instead of** - "instead of" トリガを "for" トリガに変更することはできません。 逆の変更も同様です。
- **insert,update,delete - or replace** 句を使用すると、これらのアクションを変更することができます。 たとえば、古いトリガ定義ですべての句が指定されている

場合、置き換える定義ですべての句、つまりアクションの組み合わせを指定できます。

- *SQL_statements* - トリガ定義を置き換えるときに、トリガ条件とアクションを変更できます。
- **if update** - **if update** を削除または追加し、この句が参照するカラム名を変更することができます。
- **order** *integer* - トリガ定義を置き換えるときにトリガの起動の順序を変更することもできます。

例

次の例では、`titles` テーブル内のデータが挿入または更新されたときにメッセージを出力するトリガを作成します。

```
create trigger reminder
on titles
for insert, update as
print "Don't forget to print a report for accounting."

select object_id("reminder")
-----------
1312004674
```

次のコマンドでは、**or replace** 句を使用して、titles テーブル内のデータが更新されたときに出力されるトリガのメッセージを変更します。

```
create or replace trigger reminder
on titles
for update as
print "Don't forget to give a report to accounting."

select object_id("reminder")
-----------
1312004674
```

***create or replace trigger** のパーミッション変更*

エイリアスまたは **setuser** を使用してトリガ所有者と同一化したユーザはトリガを置き換えることはできません。

トリガを置き換えるための変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、データベーステーブル所有者であることが必要。また `create trigger` 権限が無効であってはならない。 他のユーザのテーブルに対して **create trigger** を実行するには、`create any trigger` 権限が必要。 **トリガを置き換えるにはトリガ所有者であることが必要。**

細密なパーミッションが無効	細密なパーミッションが無効化されている場合は、次のようになる。 トリガ作成パーミッションの付与や取り消しを実行できるのは、システムセキュリティ担当者だけである。 データベース所有者には、ユーザテーブルにトリガを作成する暗黙のパーミッションが付与される。 ユーザは、各自が所有しているテーブルにのみトリガを作成できる。 システムセキュリティ担当者はトリガを作成するユーザパーミッションを取り消すことができる。 トリガ作成パーミッションの取り消しは、システムセキュリティ担当者が **revoke** コマンドを発行したデータベースでのみ実行される。 パーミッションを取り消されたユーザに対して、システムセキュリティ担当者が明示的に **create trigger** パーミッションを付与した場合、`create trigger` コマンドを実行するパーミッションはリストアされる。 **トリガを置き換えるにはトリガ所有者であることが必要。**

***create or replace trigger** の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
12	create	create trigger	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

create view

or replace 句を使用すると、**create view** を使用してビューの定義を置き換えることができます。

構文
変更は太字で表示されています。

```
create [or replace] view [owner.]view_name
    [(column_name[, column_name]...)]
    as
```

```
    select [distinct] select_statement
    [with check option]
```

***create or replace view** のパラメータ変更*

- **create** - ビューが存在しない場合はビューを作成します。
- **or replace** - 既存のビュー定義を置き換えます。その際にビューのセキュリティ属性は変更しません。
- *view_name* - 置き換えられるビューの名前。 既存のビューのみを置き換えることができます。 オブジェクトの名前と ID は変わりません。
- *column_name* - 見出しとして使用する名前をビューのカラムとして指定します。 **or replace** 句を指定すると、ビューのカラム名を次のように変更することができます。
  - 以前のビュー定義にカラム名の見出しが含まれていた場合は、ビューの新しい定義で見出しを省略することも、カラム名に別の見出しを設定することもできます。
  - 以前のビュー定義にカラム名の見出しが含まれていなかった場合は、新しい定義にビューのカラム名の見出しを含めることができます。
  - **select_statement** のカラム名にしたがって、カラムの見出しの数を変更することができます。
- **distinct** - 重複ローが許可されます。ビューの元の定義で `distinct` 句を指定していない場合は、このパラメータを変更して新しいビューで重複ローを許可しないように設定できます。
- *select_statement* - ビューの定義で別のテーブルとビューを指定できます。 置き換えられたビューの **select_statement** のターゲットリストで指定されたカラムを変更して、カラムを削除または追加することができます。
- **with check option** - **with check option** 句を指定して作成されたビューはこの句を指定せずに置き換えることができます。逆の場合も同様です。

例
例 1

次の例は、`Current_Product_List` に基づいています。このビューは、テーブル `Products` からアクティブな製品のすべてをリストします。 ビューの定義は次のとおりです。

```
create view Current_Product_List as
select ProductID, ProductName
from Products
where Discontinued = "No"

select object_id("Current_Product_List")
-----------
889051172
```

次のコマンドは、**or replace** 句を使用して Category カラムを Current_Product_list に追加します。 ビューのオブジェクト ID は変わりません。

```
create or replace view Current_Product_List as
select ProductID, ProductName, Category
from Products
where Discontinued = "No"

select object_id("Current_Product_List")
-----------
889051172
```

例 2

次の例では、従属オブジェクトを持つビュー V1 が置き換えられます。

```
create table T1(C1 int, C2 int)
create table T2(C1 int, C2 int)

create view V1 as select * from T1
create view V2 as select * from V1

create function foo1
returns int
as
begin
    declare @number int
    select @number = C1 from V2
end
return @number
select object_id("V1")
-----------
985051514
```

```
create or replace V1 as select * from T2

select * from V2

select dbo.foo1()

select object_id("V1")
-----------
985051514
```

置き換えバージョンの V1 は、T1 ではなく T2 を参照します。 V2 と foo1 は両方とも再コンパイルされます。 select * from v2 では、V2 が再コンパイルされますが foo1 は再コンパイルされません。これは UDF が起動されたときに再コンパイルされます。

置き換えられたビューに依存するオブジェクト
ビューには他のオブジェクト定義を組み込むことができます。

- 置き換えられるビューが別のビューに組み込まれている場合はアクセスされたときに親ビューが自動的に再コンパイルされます。
- 置き換えによってビューのカラム数が変更される場合は、そのビューを参照する他のビューとプロシージャの定義の修正が必要になる場合があります。
  次の例では、所有者がカラム数とカラム名が異なる `V1` に置き換えます。 この場合 `P` の置き換えも必要になります。

  ```
  create view V1 as select C1 from T1
  create procedure P as select * from V1
  ```

  ```
  create or replace V1 as select C1, C2 from T1
  ```

  次の例では、所有者が定義から C2 を削除して V2 を置き換えています。 P を実行すると、V2 に C2 が存在しなくなったため、エラーが発生します。 このため P の置き換えが必要になります。

  ```
  create view V2 as select C1, C2 from T2
  create procedure P as select C2 from V2
  ```

  ```
  create or replace V2 as select C1 from T2
  ```

  ビューを置き換える前に **sp_depends** を実行して、置き換え対象のビューに依存するストアドプロシージャまたは親ビューが存在するかどうかを判定します。 このようなストアドプロシージャまたは親ビューが存在する場合は、ビューの置換後に必要に応じてストアドプロシージャまたは親ビューを置き換えます。
- ビューに定義されている `instead of` トリガは、ビューが置き換えられると削除されます。
- 置き換えられたビューに依存する PRS を使用可能な状態にリストアするには、完全リフレッシュが必要になります。 再コンパイルが行われるまで、これらはリフレッシュすることも、クエリの再書き込みに使用することもできません。

制限事項
カラムレベルのパーミッションがビューに付与されている場合は、置き換えることができず、エラー 2014 が発生します。 このようなビューを置き換えるには、事前にパーミッションを取り消すか、ビューを削除してから再作成する必要があります。

***create or replace view** のパーミッション変更*
エイリアスまたは **setuser** を使用してビュー所有者と同一化したユーザはビューを置き換えることはできません。

ビューの置き換えに関する変更は太字で表示されています。

細密なパーミッションが有効	細密なパーミッションが有効の場合、`create view` 権限が必要。 他のユーザに対して **create view** を実行するには、`create any view` 権限が必要。 **ビューを置き換えるにはビュー所有者であることが必要。**
細密なパーミッションが無効	細密なパーミッションが無効の場合、データベース所有者であるか、`create view` 権限が必要。 **ビューを置き換えるにはビュー所有者であることが必要。**

***create or replace view** の監査変更*
変更は太字で表示されています。

イベント	監査オプション	監査されるコマンドまたはアクセス	extrainfo の情報
13	create	create view	• Roles - 現在のアクティブな役割 • Keywords or options - NULL • Previous value - NULL • Other information - NULL • Current value - NULL • Proxy information - set proxy が有効な場合は元のログイン名 • **or replace - create or replace の場合**

drop encryption key

`master` データベースの `sysencryptkeys` テーブルから、データベース暗号化キーを削除します。

構文

```
drop encryption key key_name
```

パラメータ

- ***key_name*** – データベース暗号化キーの名前です。

使用法

削除するデータベース暗号化キーが、データベースの暗号化に使用されている場合、このコマンドは失敗します。

標準

ANSI SQL - 準拠レベル: Transact-SQL 拡張機能

パーミッション

drop encryption key のパーミッションチェックは、細密なパーミッションの設定によって異なります。

細密なパーミッションが有効	SAP ASE によって "manage database encryption key" という新しいパーミッションが作成される。 データベース暗号化キーの作成のためのパーミッションが必要。
細密なパーミッションが無効	sso_role、keycustodian_role または、**create encryption key** 権限を持つユーザであることが必要。

参照：

- create encryption key (130 ページ)
- データベース暗号化キーの変更 (67 ページ)
- データベース暗号化キーの削除 (68 ページ)
- データベース暗号化キーのバックアップ (79 ページ)
- データベース暗号化キーの作成 (66 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)

drop trigger

drop trigger は、既存のトリガまたは複数のトリガの削除や置換に使用することができます。

SAP ASE バージョン 16.0 では、複数のトリガの作成機能、および文の実行後のトリガの起動順序の指定機能が導入されています。

drop trigger コマンドは単一のトリガを削除します。 テーブルに複数のトリガが存在する場合は、個別に削除することができます。

dump database

dump database は、圧縮で作成されたデータベースダンプまたはトランザクションダンプのローデータに対する偶発的な変更をチェックし、この圧縮ブロックを正しく読み込んで、圧縮解除できるかを検証するための巡回冗長検査を追加します。

構文は次のとおりです。

```
dump database database_name to dump_device with
compression=n,verify={crc | read_after_write}
```

各パラメータの意味は、次のとおりです。

- **verify=crc** - 巡回冗長検査を実行することを示します。
- **verify=read_after_write** - 圧縮ブロックの書き込みおよび圧縮解除後 Backup Server が圧縮ブロックごとに再読み込みを行います。 Backup Server がエラーを検出すると、エラーが検出されたファイル内のオフセットが出力されます。 **load database** コマンドでは **verify=read_after_write** を使用できません。

kill

kill コマンドには、**with force** パラメータが追加されました。

with force パラメータは、通常の **kill** *spid* パラメータでプロセスを終了できない場合に使用します。

構文
構文は次のとおりです。

```
kill spid with force
```

with force は、指定の spid を強制的に終了させることを示しています。

例
この例では、spid 16 が終了されます。

```
kill 16 with force
```

使用法

- スピンロックのある spid を終了させようとして **with force** を使用すると、SAP ASE により次のメッセージが表示されます。

  ```
  You cannot kill spid 'spid_number' with force option as it is
  holding spinlock(s).
  ```

- **with force** パラメータの発行パーミッションは、**kill** *spid_number* の発行パーミッションと同じです。
- *spid_number* は定数である必要があります。ストアドプロシージャにパラメータとして渡したり、ローカル変数として使用したりすることはできません。

load database

load database は、圧縮されたデータベースダンプまたはトランザクションダンプのローデータに対する偶発的な変更をチェックするための巡回冗長検査を追加します。

構文は次のとおりです。

```
load database database_name from dump_device with verify[only]=crc
```

各パラメータの意味は、次のとおりです。

- **verify=crc** - 巡回冗長検査を実行することを示します。

select

16.0 以降のバージョンの SAP ASE 上で @変数、@@グローバル変数、および定数のみを連鎖モードで参照する **select** 文を発行すると、新しいトランザクションは起動しません。

これにより、次のような SQL 構成体を作成できます。

```
set chained ON
<... Perform some DML or other commands ...>
select @@error, @@trancount, @@transtate
```

状況によっては、SAP ASE は、DML 文またはその他の文の実行中にエラーが発生した場合、アクティブなトランザクションをロールバックします。 **select** 文を使用して、サーバが連鎖モードであるときのトランザクションの状態をチェックし、新しいトランザクションを自動的に起動することなく、ロールバックをトリガしたエラーの発生日時を収集できます。

大部分の関数は **select** 文で使用されると新しいトランザクションを開始しますが、以下の診断関数では新しいトランザクションは開始しません。

- **getdate**
- **getutcdate**
- **current_date**
- **current_time**
- **current_bigdatetime**
- **current_bigtime**
- **abs**
- **asehostname**
- **hostname**
- **switchprop**

select into

select into 構文は、既存のテーブルからインデックス圧縮テーブルを選択してインデックス圧縮テーブルを作成するように拡張されました。

構文

```
select [ all | distinct ] select_list
    into [[database.][owner].table_name
    [with {max_rows_per_page = num_rows,
        exp_row_size = num_bytes,
        reservepagegap = num_pages,
        identity_gap = value,
        transfer table [on | off],
        compression [={NONE | PAGE | ROW | ALL} ] ,
        index_compression [={NONE | PAGE} ]
        }
    ]
    [on segment_name]
```

パラメータ変更

- NONE - 指定したテーブルのインデックスは圧縮されません。
  index_compression = PAGE を指定して作成されたインデックスは圧縮されます。
- PAGE - 指定したテーブルのすべてのインデックスが圧縮されます。
  index_compression = NONE を指定して作成されたインデックスは圧縮されません。

参照：

- インデックス圧縮テーブルの作成 (59 ページ)
- インデックス圧縮用の create table (147 ページ)

set

SAP ASE バージョン 16.0 では、**set** コマンドに変更が加えられています。

set spinlock_aggregation {on | off}
結果セットに `SpinlockName` の値が同じであるローが複数含まれる場合に、**set spinlock_aggregation** パラメータを使用して、`monSpinlockActivity` でレポートされるスピンロックメトリックを SAP ASE で集約するかどうかを設定します。

デフォルトで、SAP ASE は同じ値を持つすべてのスピンロックについて、`Grabs`、`Waits`、`Spins` の値を集約します。 `monSpinlockActivity` テーブル内のスピンロックインスタンスのそれぞれで個別のローが返されるように SAP ASE を設定する場合は、**set spinlock_aggregation** をオフに設定します。

次の例は、**set spinlock_aggregation** を無効にした場合に monSpinlockActivity から返される結果セットを示しています。

```
1> set spinlock_aggregation off
2> go
1> select * from monSpinlockActivity
2> where SpinlockName like "default data cache%"
3> order by Contention
4> go
Grabs               Spins               Waits               OwnerPID
        LastOwnerPID Contention                  InstanceID
SpinlockSlotID
         SpinlockName
 ------------------- ------------------- -------------------
-----------
         ------------ -------------------------- ----------
--------------

-------------------------------------------------------------------
---
              37697                978                   1          0
            1638413                  0.000027         0          2338
        default data cache               16396                15
               2           0
            1638413                  0.000122         0          2340
         default data cache
              17317                 24                    3          0
            1638413                  0.000173         0          2339
         default data cache
              27533                629                   5          0
            1638413                  0.000182         0          2341
         default data cache
```

この monSpinlockActivity インスタンスから複数のローを選択する場合は次のようになります。

```
select count(*) from monSpinlockActivity
-----------
        2384
```

しかし、**set spinlock_aggregation** を有効にして同じクエリを実行すると、次のようになります。

```
1> set spinlock_aggregation on
2> go
1> select * from monSpinlockActivity
2> where SpinlockName like "default data cache%"
3> order by Contention
4> go

Grabs               Spins               Waits               OwnerPID
        LastOwnerPID Contention                  InstanceID
SpinlockSlotID
         SpinlockName
 ------------------- ------------------- -------------------
-----------
```

```
         ------------ -------------------------- ----------
--------------

-------------------------------------------------------------------
              99235               1646                  11          0
            1769486                 0.000111         0          2338
         default data cache
```

これによって、monSpinlockActivityのインスタンスで表示されるローがかなり少なくなります。

```
select count(*) from monSpinlockActivity
-----------
         324
```

set erase_residual_data {on | off}
SAP ASE バージョン 16.0 では、残存データを消去する機能の有効化が導入されています。

set を使用することで、必要に応じて残存データの削除を有効化または無効化することができます。

このオプションをセッションレベルで有効化すると、セッション内で発生したすべてのページ割り付け解除から残存データが削除されます。 これには、"erase residual data" オプションを明示的に OFF にしたテーブルのページ割り付け解除も含まれます。

このオプションは、個別セッションのすべてのユーザが設定できます。必要なパーミッションは特にありません。

set statistics query_name_html [queryname | on | off]
set statistics query_name_html は、同一クエリの実行に関連するファイルの区別または特定に利用できます。

set lock {wait [numsecs] | nowait | default}
SAP ASE バージョン 16.0 では、**default** パラメータが **set** コマンドに追加されています。これは現在のセッションレベルのロック待機設定を無効にし、代わりに現在のサーバ全体を対象とする **lock wait period** 設定パラメータの設定を使用します。

参照：
- 複数のトリガ (95 ページ)
- HTML 形式のクエリプランと実行統計 (53 ページ)

システムプロシージャ

SAP ASE 16.0 には、新しいシステムプロシージャと変更されたシステムプロシージャが含まれています。

変更されたシステムプロシージャ

SAP ASE 16.0 では、既存のシステムプロシージャが変更されています。

sp_audit

SAP ASE 16.0 では、**config_history** 監査オプションが追加されています。

オプション	説明
config_history	設定履歴の監査を有効または無効にする。

sp_chgattribute

pnt_locking テーブル属性は、**sp_chgattrbute** によってパーティションレベルのロックを有効または無効にします。 デフォルトでは、パーティションロックは無効です。

構文

```
sp_chgattribute objectname, 'ptn_locking', value
```

パラメータ

- ***objectname*** – **ptn_locking** を変更するテーブルの名前です。
- **ptn_locking** – value - パーティションレベルのロックを有効にする場合は 1 に設定し、無効にする場合は 0 に設定します。

例

- **例 1** – この例では、`authors` テーブルに対してパーティションレベルのロックを有効にします。

  ```
  sp_chgattribute authors, "ptn_locking", 1
  ```

- **例 2** – この例では、`authors` テーブルに対してパーティションレベルのロックを無効にします。

  ```
  sp_chgattribute authors, "ptn_locking", 0
  ```

パーミッション

sp_chgattribute を実行できるのは、オブジェクト所有者だけです。

sp_clusterlockusage

sp_clusterlockusage の出力は、パーティションロックに固有のクラスタロックの使用状況情報を出力するように拡張されました。

```
Lock Usage                          count      % of total
-------------------------  ------------   ------------
Total Locks                      1479860        n/a
 Free Locks                      1469318       99.29 %
 Used Locks                       10542        0.71 %
 Object Locks                      7948        0.54 %
 Physical Locks                    1864        0.13 %
 Table Locks                          0        0.00 %
 Partition Locks     8 0.00 %
 Page Locks                           0        0.00 %
 Row Locks                           42        0.00 %
 Others                             680          0.05 %
```

sp_dboption

sp_dboption システムプロシージャは SAP ASE での削除から生じた残存データの削除機能をサポートします。

構文

データベースレベルで残存データの削除を有効化する構文は次のようになります。

```
sp_dboption dbname, "erase residual data", true
```

例

以下の例では次の 2 つのテーブルを使用します。

- `create table t1 (col1 int) with "erase residual data" on`
- `create table t2 (col1 int) with "erase residual data" off`

例 1

データベースレベルで設定されているため、テーブル `t1` では残存データを消去するオプションがオンであり、`t1` に対して実行される **drop table** コマンドと **truncate table** コマンドは両方とも、ページのすべての残存データをクリーンアップすることになります。

一方、テーブル `t2` は "**erase residual data off**" 句を指定して作成されたため、**erase residual data** オプションが明示的にオフに設定されています。 データベースレベルで `"erase residual data"` オプションが `true` に設定されていても、残存データは削除されません。 この結果、`t2` に対して **drop table** および **truncate table** を実行した後も、残存データが保持されます。

```
create database db1
go
sp_dboption db1, "erase residual data", true
go
use db1
```

```
go
create table t1 (col int)
go
insert t1 values ...
go
create table t2 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
drop table t1
go
truncate table t2
go
drop table t2
go
```

例 2

この例では次のようになります。

- テーブル `t1` には `"erase residual data off"` が明示的に設定されてはいませんが、データベースレベルで設定されているため、**truncate table t1** を実行すると `t1` から残存データが削除されることになります。
- テーブル `t2` は、このオプションがデータベースレベルで設定されているため、作成時に `"erase residual data"` オプションが設定されています。 このため、**truncate table t2** を実行すると、`t2` から残存データが削除されます。
- テーブル `t3` には `"erase residual data off"` が明示的に指定されています。このため **sp_dboption** で `"erase residual data"` が true に設定され、SAP ASE が **truncate table t3** を実行しても、残存データは削除されません。

```
create database db1
go
use db1
go
create table t1 (col int)
go
sp_dboption db1, "erase residual data", true
go
create table t2 (col1 int, col2 char(10))
go
create table t3 (col1 int, col2 char(10)) with "erase residual data"
off
go
truncate table t1
go
truncate table t2
go
truncate table t3
go
```

使用法
この設定をデータベースレベルで有効化すると、割り付け解除が発生する操作のすべてで実行後にページのクリーニングが行われます。 デフォルトでは、このオプションは無効です。

注意： 詳細レベルでの制御ができなくなるため、このオプションの使用はパフォーマンスに大きく影響する場合があります。

パーミッション
sp_dboption を使用してデータベースレベルのオプションを設定するには、ユーザがシステム管理者またはデータベース所有者である必要があります。

参照：
- 残存データの削除に使用される create table (149 ページ)
- set (163 ページ)

sp_depends
SAP ASE バージョン 16.0 では、**sp_depends** を使用して特定のテーブルに関連付けられた複数のトリガをリストすることができます。

sp_depends に対する構文上の変更はありません。 次の構文で、各 DML アクションに関連付けられた複数のトリガのリストを表示します。

```
sp_depends table_name
```

sp_encryption
sp_encryption システムプロシージャは、"database encryption key" という新しいキータイプをレポートして、データベースが完全に暗号化されているかどうかを示します。

次に例を示します。

```
1> create encryption key key1 as default for database encryption
2> go
1> sp_encryption helpkey, key1

Key Name    Key Owner    Key Length   Key Algorithm
    Key Type                            Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    --------------------------------   ------ ---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
    symmetric database encryption key      0                       1
  master key       0                              0
1> create encryption key key2 for database encryption with master key
2> create encryption key key3 for database encryption with
dual_control
```

```
3> go
1> sp_encryption helpkey, 'key%'
Key Name    Key Owner    Key Length   Key Algorithm
    Key Type                           Pad    Initialization Vector
  Protected By     Key Recovery
     # of Key Copies
----------- ------------ ------------ -------------
    ---------------------------------   ------ ---------------------
  -------------- ----------------    -----------------
key1        dbo          256          AES
    symmetric database encryption key      0                        1
  master key       0                              0
key2        dbo          256          AES
    symmetric database encryption key      0                        1
  master key       0                              0
key3        dbo          256          AES
    symmetric database encryption key      0                        1
  dual_control(master key + dual master key)  0                  0

1> create database encr_db1 encrypt with key1
2> create database encr_db2 encrypt with key2
3> create database encr_db3 encrypt with key3
4> go
1> sp_encryption helpkey, '%', "display_dbs"
Key Name Key Owner Encrypted Database
-------------------- --------------------------------------
key1 dbo encr_db1
key1 dbo encr_db2
key3 dbo encr_db3
```

参照：

- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

sp_familylock

sp_familylock ストアドプロシージャの出力に `partitionid` カラムが追加されました。

構文は次のとおりです。

```
sp_familylock
```

テーブルロックと詳細なロックの partitionid の値は 0 です。 partitionid は、パーティションレベルのロックの場合にのみ値が入力されます。

```
spid  locktype             table_id     partitionid     page   row…
----  -------------------  -----------  -------------   -----  ----
0     Ex_partition           576002052      576004423       0     0
0     Sh_partition_intent   1417053053     1417053053       0     0
```

参照：

- sp_familylock によるパーティションロックの表示 (8 ページ)

sp_helpdb

sp_helpdb システムプロシージャは、データベースの完全暗号化機能をサポートします。

完全暗号化データベースに対して **sp_helpdb** を実行すると、次のいずれかの暗号化ステータスがレポートされます。

- 暗号化済み
- 暗号化の進行中
- 復号化の進行中

データベースの暗号化または復号化が進行中の場合、**sp_helpdb** は処理の完了率をレポートします。

例

例 1

暗号化中のデータベースのステータスをレポートします。

```
>sp_helpdb
>go
name         db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db      6.0 MB     sa       4  Aug 07, 2013 full
              0 NULL
    encryption in progress: 35%
……
```

例 2

一部が暗号化されたデータベースのステータスをレポートします。

```
>sp_helpdb
>go
name         db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
```

```
……
test_db     6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    encrypted partly

……
```

例 3

一部が復号化されたデータベースのステータスをレポートします。

```
>sp_helpdb
>go
name        db_size    owner dbid  created      durability
    lobcomplvl inrowlen
status
……
test_db     6.0 MB     sa       4  Aug 07, 2013 full
             0 NULL
    decrypted partly

……
```

参照：

- 既存データベースの暗号化 (70 ページ)
- dbencryption_status (118 ページ)
- 暗号化処理のサスペンド (77 ページ)
- 暗号化処理の再開 (77 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

sp_lock

sp_lock ストアドプロシージャの出力に `partitionid` カラムが追加されました。

テーブルロックと詳細なロックの `partitionid` の値は 0 です。 `partitionid` は、パーティションレベルのロックの場合にのみ値が入力されます。

この例では、SAP ASE によって現在保持されているパーティションロックなどのすべてのロックを表示します。

```
sp_lock
go

fid spid loid locktype table_id partitionid page row dbname class
context
--- ---- ---- -------- -------- ----------- ---- --- ------ ------
-------
0 13 26 Ex_intent 420193516 0 0 0 master Non Cursor Lock
0 13 26 Ex_intent_partition  420193516 452193630 0 0 master Non
Cursor Lock
0 13 26 Ex_page 420193516  452193630 4993 0 master Non Cursor Lock
0 14 28 Ex_intent 420193516 0 0 0  master Non Cursor Lock
0 14 28 Ex_intent_partition 420193516 468193687 0 0  master Non
Cursor Lock
0 14 28 Ex_page 420193516 468193687 5001 0 master Non Cursor Lock
0 16 32 Sh_intent 1006623598 0 0 0 master Non Cursor Lock
```

参照：

- sp_lock によるパーティションロックの表示 (7 ページ)

新しいシステムプロシージャ

SAP ASE バージョン 16.0 では、新しいシステムプロシージャが導入されました。

sp_confighistory

`ch_events` ビューを作成し、SAP ASE 設定に行われた変更を表示します。

構文

```
sp_confighistory create_view
    begin_date[, end_date]]
    last[items_num]
    {area | type | target | element}[, item_name]
    help
```

パラメータ

- **create_view** – `ch_events` ビューを作成することを示します。
- ***begin_date*, [*end_date* –** *begin_date* 値から *end_date* 値までのすべての項目を表示します。
- **last** – 最新の設定履歴項目を表示します。
- ***items_num*** – 最新の設定履歴項目のリストから表示する項目の数です。
- **area | type | target | element** – 指定した領域の項目を表示します。
  - **area** - 監査可能イベントが発生した領域。 下記のいずれかです。

    - `server` - サーバレベルのイベント。
    - `database` - データベースレベルのイベント。
    - `cache` - キャッシュレベルのイベント。
    - `traceflag` - **dbcc traceflag** イベントと **set switch** イベント。
    - `SUSD` - 起動またはシャットダウン。
    - `audit` - 監査ステータスの変更。
  - **type** - 監査可能イベントのタイプ。 下記のいずれかです。
    - **sp_configure**
    - **sp_serveroption**
    - **sp_dboption**
    - **sp_cacheconfig**
    - **sp_poolconfig**
    - **create thread pool**
    - **alter thread pool**
    - **drop thread pool**
    - **dbcc traceflag**
    - **set switch**
    - configuration file change
    - startup
    - shutdown
    - shutdown with wait
    - shutdown with nowait
    - abrupt shutdown
    - global auditing
    - config history auditing
  - **target** - 変更が適用されるターゲットオブジェクトの名前 (server、cache、thread pool、database names、traceflag number など)。
  - **element** - 設定またはその他のオプション名 (“`enable monitoring`”、“`config pool:  4K, option: wash size`” など)。
- **help – sp_confighistory** の使用方法を表示します。

例

-

パーミッション

- このプロシージャを使用して `ch_events` ビューを作成できるのは、システム管理者 (sa_role を持つユーザ) だけです。

- このプロシージャを使用して ch_events ビューにクエリを実行できるのは、システム管理者 (sa_role を持つユーザ) と mon_role を持つユーザだけです。

パーミッションチェックは、細密なパーミッションの設定によって異なります。

設定	説明
有効	次のパーミッションを持つユーザのみ • select any audit table パーミッション。ch_events ビューに対するクエリの実行が可能。 • manage auditing パーミッション。設定履歴監査のオプションステータスの変更が可能。 • select any audit table パーミッション。ch_events ビューに対するクエリの実行が可能。 • select any audit table パーミッション。監査テーブルに対するクエリの実行が可能。
無効	次のユーザのみ • システムセキュリティ担当者 (sso_role を持つユーザ)。設定履歴監査のオプションステータスの変更が可能。 • ch_events ビューにクエリを実行できるのは、システム管理者 (sa_role を持つユーザ) と mon_role を持つユーザのみ。

sp_ dropglockpromote_ptn

パーティションロックプロモーション値を削除します。

構文

サーバワイドなパーティションロックプロモーション設定を削除するための構文は次のとおりです

```
sp_dropglockpromote_ptn "server"
```

データベースレベルまたはテーブルレベルでは、次の構文でパーティションロックプロモーションスレッショルドを削除します。

```
sp_dropglockpromote_ptn {"database" | "table"}, objname
```

パラメータ

- **server** – パーティションロックプロモーションスレッショルドに使用するサーバワイドな値を削除します。
- **"database" | "table"** – パーティションロックプロモーションスレッショルドをデータベースから削除するか、テーブルから削除するかを指定します。 これらは、Transact-SQL キーワードであるため、引用符が必要です。

- ***objname*** – パーティションロックプロモーションスレッショルドを削除するテーブルまたはデータベースの名前です。

例

- **例 1** – *titles* からパーティションロックプロモーション値を削除します。 *titles* のロックプロモーションが、データベースワイドまたはサーバワイドな値を使用するようになります。

```
sp_ dropglockpromote_ptn "table", titles
```

使用法

sp_dropglockpromote_ptn を使用する場合のその他の注意事項を次に示します。

- **sp_setpglockpromote_ptn** によって設定されたパーティションロックプロモーション値を削除するには、**sp_dropglockpromote_ptn** を使用します。
- データベースのパーティションロックプロモーションスレッショルドを削除すると、パーティションロックプロモーションスレッショルドが設定されていないテーブルはサーバワイドな値を使用します。
- テーブルの値が削除されると、SAP ASE サーバは、データベースのロックプロモーションスレッショルドが設定されていればその値を使用し、設定されていない場合にはサーバワイドな値を使用します。
- サーバワイドなパーティションロックプロモーションスレッショルドを削除すると、テーブルレベルで設定されたパーティションロックプロモーションスレッショルド値が使用されます。 それ以外の場合は、データベースレベルで設定されたパーティションロックプロモーションスレッショルド値が使用されます。 パーティションロックプロモーションスレッショルド値がデータベースレベルでもテーブルレベルでも設定されていない場合は、パーティションロックプロモーションは無効化されます。 パーティションロックプロモーションは、**sp_setrowlockpromote_ptn** を使用して有効化できます。

パーミッション

sp_dropglockpromote_ptn のパーミッションチェックは、細密なパーミッションの設定によって異なります。

設定	説明
有効	細密なパーミッションが有効の場合、`manage lock promotion thresh-old` 権限を持つユーザであることが必要。
無効	細密なパーミッションが無効の場合、**sa_role** を持つユーザであることが必要。

監査

sysaudits テーブルの event カラムと extrainfo カラムの値は次のとおりです。

情報	値
イベント	38
監査オプション	**exec_procedure**
監査されるコマンドまたはアクセス	プロシージャの実行
extrainfo の情報	• *Roles* - 現在のアクティブな役割 • *Keywords or options* - NULL • *Previous value* - NULL • *Current value* - NULL • *Other information* - すべての入力パラメータ • *Proxy information* - **set proxy** が有効な場合は元のログイン名

参照：

- sp_droprowlockpromote_ptn (177 ページ)
- パーティションロックプロモーションスレッショルドの削除 (10 ページ)

sp_droprowlockpromote_ptn

サーバレベル、データベースレベル、またはテーブルレベルのパーティションロックプロモーションスレッショルド値を削除します。

構文

サーバワイドなパーティションロックプロモーション設定を削除するための構文は次のとおりです

```
sp_droprowlockpromote_ptn "server"
```

データベースレベルまたはテーブルレベルでは、次の構文でパーティションロックプロモーションスレッショルドを削除します。

```
sp_droprowlockpromote_ptn {"database" | "table"}, objname
```

パラメータ

- **server** – パーティションロックプロモーションスレッショルドに使用するサーバワイドな値を削除します。

- **"database" | "table"** – パーティションロックプロモーションスレッショルドをデータベースから削除するか、テーブルから削除するかを指定します。 これらは、Transact-SQL キーワードであるため、引用符が必要です。
- ***objname*** – パーティションロックプロモーションスレッショルドを削除するテーブルまたはデータベースの名前です。

例

- **例 1** – `sales` テーブルからパーティションロックプロモーション値を削除します。 `sales` のパーティションロックプロモーションが、データベースワイドまたはサーバワイドな値を使用するようになります。

```
sp_droprowlockpromote_ptn "table", "sales"
```

使用法

sp_droprowlockpromote_ptn を使用する場合のその他の注意事項を次に示します。

- **sp_setrowlockpromote_ptn** によって設定されたパーティションロックプロモーション値を削除するには、**sp_droprowlockpromote_ptn** を使用します。
- データベースのパーティションロックプロモーションスレッショルドを削除すると、テーブルレベルでパーティションロックプロモーションスレッショルドが設定されていないデータローロックテーブルはサーバワイドな値を使用します。 パーティションロックプロモーション設定パラメータの値を確認するには、**sp_configure** を使用します。
- テーブルのパーティションロックプロモーション値が削除されると、SAP ASE サーバは、データベースのパーティションロックプロモーションスレッショルドが設定されている場合にはそのスレッショルドを使用し、データベース用にスレッショルドが設定されていない場合にはサーバワイドな値を使用します。
- データベースに使用されるパーティションロックプロモーションスレッショルドを変更するには、`master` データベースを使用してください。 データベース内のテーブルに使用されるパーティションロックプロモーションスレッショルドを変更するには、そのテーブルがあるデータベースを使用してください。
- サーバワイドなパーティションロックプロモーションスレッショルドを削除すると、テーブルレベルで設定されたパーティションロックプロモーションスレッショルド値が使用されます。 それ以外の場合は、データベースレベルで設定されたパーティションロックプロモーションスレッショルド値が使用されます。 パーティションロックプロモーションスレッショルド値がデータベースレベルでもテーブルレベルでも設定されていない場合は、パーティションロックプロモーションは無効化されます。 パーティションロックプロモーションは、**sp_setrowlockpromote_ptn** を使用して有効化できます。

パーミッション

sp_droprowlockpromote_ptn のパーミッションチェックは、細密なパーミッションの設定によって異なります。

設定	説明
有効	細密なパーミッションが有効の場合、`manage lock promotion threshold` 権限を持つユーザであることが必要。
無効	細密なパーミッションが無効の場合、**sa_role** を持つユーザであることが必要。

監査

`sysaudits` テーブルの `event` カラムと `extrainfo` カラムの値は次のとおりです。

情報	値
イベント	38
監査オプション	**exec_procedure**
監査されるコマンドまたはアクセス	プロシージャの実行
`extrainfo` の情報	• *Roles* - 現在のアクティブな役割 • *Keywords or options* - NULL • *Previous value* - NULL • *Current value* - NULL • *Other information* - すべての入力パラメータ • *Proxy information* - **set proxy** が有効な場合は元のログイン名

参照：

- sp_ dropglockpromote_ptn (175 ページ)
- パーティションロックプロモーションスレッショルドの削除 (10 ページ)

sp_helptrigger

SAP ASE バージョン 16.0 には、sp_helptrigger システムプロシージャが組み込まれています。

sp_helptrigger でリストされるものは次のとおりです。

- *tablename* によって指定されたテーブルで作成されたすべてのトリガ

- トリガを起動するコマンド (**insert**、**update**、または **delete**)
- トリガの順序番号

構文

```
sp_helptrigger tablename
```

パラメータ

- ***tablename*** – テーブルの名前を指定します。

パーミッション

sp_helptrigger はすべてのユーザが実行できます。

sp_jsconfigure

Job Scheduler エージェントを設定します。

構文

```
sp_jsconfigure [option [, value]]
```

パラメータ

- ***option*** – option は次のいずれかになります。
  - *interfaces path* - interfaces ファイルのパス
  - *errorlog* - errorlog のパス
  - *help* - **sp_jsconfigure** の構文を表示する
- ***value*** – *option* に設定する値を指定します。

例

- **例 1 – sp_jsconfigure** 構文を表示します。

  ```
  sp_jsconfigure "help"
  Usage: sp_jsconfigure [option [, value]]
          where option : 'interfaces path', ' errorlog ', 'help'
                value  : value to set for the 'option'
  ```

- **例 2 –** interfaces ファイルのパスを設定します。

  ```
  sp_jsconfigure 'interfaces path', "/SAP_ASE/data"
  ```

- **例 3 –** interfaces ファイルのパスを設定します。

  ```
  1> sp_jsconfigure "errorlog", "/SAP_ASE/data/js.log"
  ```

- **例 4 – sp_jsconfigure** に設定されている値を表示します。

  ```
  sp_jsconfigure
  Parameter Name                    Config
  ```

```
Value

 ------------------------------
------------------------------------------------------------------------
----------------------------------------
 interfaces path                  /SAP_ASE/
data
 errorlog                         /SAP_ASE/data/
js.log
```

使用法

- `installjsdb` スクリプトを使用して **sp_jsconfigure** (`$SYBASE/$SYBASE_ASE/scripts` にある) をインストールします。
- 設定の変更を有効にするには、JS Agent を再起動する必要があります。
- *interfaces path* または *errorlog* に値を指定していない場合、**sp_jsconfigure** はデフォルト値を使用します。
- *interfaces path* オプションまたは *errorlog* オプションに存在しないパスを指定すると、Job Scheduler は起動しません。
- *interfaces path* には、スケジュール済みジョブが実行される Job Scheduler のホストサーバとターゲットサーバ (ターゲットサーバとホストサーバは同じマシンである場合もあります) のエントリが含まれる interfaces ファイルを指定する必要があります。

パーミッション

sp_jsconfigure を実行するには `js_admin_role` を持っていることが必要です。

sp_logging_rate

指定期間におけるトランザクションログの増大率を計算します。

構文

```
sp_logging_rate {'full'|'sum', '[day,]hh:mm:ss'}[,
interval='hh:mm:ss' | clear_option='y'|'n']
```

パラメータ

- **full – sp_logging_rate** は、各コレクションの詳細なレポートを提供します。
- **sum – sp_logging_rate** は、平均値、最小値、最大値、最大率を含む概要情報を提供します。 期間を指定しない場合、**sp_logging_rate** は 10 秒ごとに情報を収集します。

- **day, *hh:mm:ss*** – *date, hour:minute:second* の形式を使用して、**sp_logging_rate** の実行期間を指定します。
- **interval = '*hh:mm:ss*'** – *hour:minute:second* の形式を使用して、実行期間を指定します。
- **clear_option = 'y' | 'n"** – データ収集中にモニタカウンタを消去するかどうかを決定します。

例

- **sum パラメータの使用例** – **sp_logging_rate** は、1 日と 8 時間分の情報を収集し、10 分ごとにサンプルを取り、期間の最後に概要情報を出力します。

```
sp_logging_rate 'sum', '1,08:00:00', '00:10:00'
=========================
Total Summary Information
=========================
Transaction Log Growth Rate     Min GB/h       Max GB/h       Avg
GB/h
---------------------------   --------------  --------------
--------------
                                  0.000000        2.870076
1.823028
```

- **full パラメータの使用例** – **sp_logging_rate** は、3 分間分の情報を収集し、10 秒ごとにサンプルを取り (デフォルト)、期間の最後に概要情報を出力します。

```
sp_logging_rate 'full', '00:03:00'
Date Time                       Transaction Log Growth Rate GB/h
-------------------------     ---------------------------------
Oct 22 2013  6:00:32:480AM                0.406779

Oct 22 2013  6:00:42:483AM                0.000000

Oct 22 2013  6:00:52:483AM                0.000000

Oct 22 2013  6:01:02:483AM                0.000000

Oct 22 2013  6:01:12:490AM                0.000000

Oct 22 2013  6:01:22:500AM                0.000000

Oct 22 2013  6:01:32:476AM                2.341870

Oct 22 2013  6:01:42:483AM                2.828132

Oct 22 2013  6:01:52:480AM                2.850305
```

```
Oct 22 2013  6:02:02:483AM                 2.782750

Oct 22 2013  6:02:12:483AM                 2.853574

Oct 22 2013  6:02:22:480AM                 2.002917

Oct 22 2013  6:02:32:483AM                 2.848995

Oct 22 2013  6:02:42:483AM                 2.754143

Oct 22 2013  6:02:52:483AM                 2.854949

Oct 22 2013  6:03:02:480AM                 2.722928

Oct 22 2013  6:03:12:476AM                 2.870076

Oct 22 2013  6:03:22:480AM                 2.697094


=========================
Total Summary Information
=========================
Transaction Log Growth Rate     Min GB/h       Max GB/h       Avg
GB/h
----------------------------   --------------  --------------
--------------
                                   0.000000        2.870076
1.823028
```

使用法

- **sp_logging_rate** の実行中に、モニタリングデータを収集するスクリプトやプロシージャ (sp_sysmon など) を実行することはできません。 **sp_logging_rate** は実行中にモニタカウンタを収集し消去するので、スクリプトやプロシージャが収集するモニタカウンタ情報は正確な情報ではなくなります。
- **sp_logging_rate** は、**'day,***hh:mm:ss*' で指定する期間より大きな値を **interval =** '*hh:mm:ss*' に指定した場合、信頼できない結果を生成します。
- **interval =** '*hh:mm:ss* ' と **'day,***hh:mm:ss* ' の値を指定する際、以下のことに注意してください。

- **'day,***hh:mm:ss'* に指定する値より大きな値を **interval =** '*hh:mm:ss*' に指定すると、エラーメッセージが生成され、**sp_logging_rate** は結果セットを生成しません。
- **sp_logging_rate** は、**'day,***hh:mm:ss'* と **interval =** '*hh:mm:ss'* の比率があまりにも小さい場合、信頼できない結果を生成する可能性があります。 たとえば、`day,00:10:00` および `interval='00:04:00'` と指定した場合、**sp_logging_rate** が、2 つの値のみを収集し、最初の値を最大値、2 つ目の値を最小値、そしてその平均値を出力することになります。 この比率が適切であれば、結果セットの信頼性も高まります。

パーミッション

sp_logging_rate の実行には、システム管理者権限が必要です。

sp_setpglockpromote_ptn

sp_setpglockpromote_ptn システムプロシージャは、サーバレベル、データベースレベル、およびテーブルレベルでパーティションロックプロモーションスレッショルドを設定します。

構文

サーバレベルでは、次の構文でパーティションロックプロモーションスレッショルドを設定します。

```
sp_setpglockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

データベースレベルまたはテーブルレベルでは、次の構文でパーティションロックプロモーションスレッショルドを設定します。

```
sp_setpglockpromote_ptn "database | table", objname, new_lwm,
new_hwm, new_pct
```

パラメータ

- **server** – ロックプロモーションスレッショルドに使用するサーバワイドな値を設定します。
- **"database" | "table"** – ロックプロモーションスレッショルドをデータベース用に設定するか、またはテーブル用に設定するかを指定します。 これらは、Transact-SQL キーワードであるため、引用符が必要です。
- ***objname*** – パーティション、テーブル、またはデータベースのロックプロモーションスレッショルドを設定する場合は、そのパーティション、テーブル、またはデータベースの名前を指定します。サーバワイドな値を設定する場合は、null を指定します。 パーティションワイドな値を設定する場合は、*objname* に *table_name.partition_name* フォーマットを使用します。

- ***new_lwm*** – SAP ASE がパーティションロックを取得する前に取得する必要のあるページロックの最低数を指定します。
- ***new_hwm*** – SAP ASE がパーティションロックに拡大する前に、オブジェクトに使用できるページロックの最大数を指定します。
- ***new_pct*** – ロックされたページのパーセンテージ (ページサイズに基づく) の上限値を指定します。この値を超えると、ロック数が *new_hwm* ロックプロモーションと *new_lwm* ロックプロモーションの間である場合に SAP ASE がパーティションロックの取得を試みます。

例

- **例 1 –** サーバワイドなパーティションロックプロモーションスレッショルド値 LWM を 200 に設定し、HWM を 300 に、PCT を 50 に設定します。
  ```
  sp_setpglockpromote_ptn "server", NULL, 200, 300, 50
  ```
- **例 2 –** master データベースに使用されるパーティションロックプロモーションスレッショルドを設定します。
  ```
  sp_setpglockpromote_ptn "database", master, 1000, 1100, 45
  ```
- **例 3 –** pubs2 データベース内の titles テーブルに使用されるパーティションロックプロモーションスレッショルドを設定します。 このコマンドは、pubs2 データベースから発行してください。
  ```
  sp_setpglockpromote_ptn "table", "pubs2..titles", 500, 700, 10
  ```

パーミッション

sp_setpglockpromote_ptn は、すべてのユーザが実行できます。

参照：

- sp_setrowlockpromote_ptn (185 ページ)
- パーティションロックプロモーションスレッショルドの設定 (10 ページ)

sp_setrowlockpromote_ptn

sp_setrowlockpromote_ptn システムプロシージャは、サーバレベル、データベースレベル、およびテーブルレベルでパーティションロックプロモーションスレッショルドを設定します。

構文

サーバレベルでは、次の構文でパーティションロックプロモーションスレッショルドを設定します。

```
sp_setrowlockpromote_ptn "server", null, new_lwm, new_hwm, new_pct
```

データベースレベルまたはテーブルレベルでは、次の構文でパーティションロックプロモーションスレッショルドを設定します。

```
sp_setrowlockpromote_ptn "database | table", objname, new_lwm,
new_hwm, new_pct
```

パラメータ

- **server** – ロックプロモーションスレッショルドに使用するサーバワイドな値を設定します。
- **"database" | "table"** – ロックプロモーションスレッショルドをデータベース用に設定するか、またはテーブル用に設定するかを指定します。 これらは、Transact-SQL キーワードであるため、引用符が必要です。
- ***objname*** – パーティション、テーブル、またはデータベースのロックプロモーションスレッショルドを設定する場合は、そのパーティション、テーブル、またはデータベースの名前を指定します。サーバワイドな値を設定する場合は、null を指定します。 パーティションワイドな値を設定する場合は、*objname* に *table_name.partition_name* フォーマットを使用します。
- ***new_lwm*** – SAP ASE がパーティションロックを取得する前に取得する必要のあるローロックの最低数を指定します。
- ***new_hwm*** – SAP ASE がパーティションロックに拡大する前に、オブジェクトに使用できるローロックの最大数を指定します。
- ***new_pct*** – ロックされたローのパーセンテージ (テーブルサイズに基づく) の上限値を指定します。この値を超えると、ロック数が *new_hwm* ロックプロモーションと *new_lwm* ロックプロモーションの間である場合に SAP ASE がパーティションロックの取得を試みます。

例

- **例 1** – サーバワイドなパーティションロックプロモーションスレッショルド値 LWM を 200 に設定し、HWM を 300 に、PCT を 50 に設定します。
  ```
  sp_setrowlockpromote_ptn "server", NULL, 200, 300, 50
  ```
- **例 2** – master データベースに使用されるパーティションロックプロモーションスレッショルドを設定します。
  ```
  sp_setrowlockpromote_ptn "database", master, 1000, 1100, 45
  ```
- **例 3** – pubs2 データベース内の titles テーブルに使用されるパーティションロックプロモーションスレッショルドを設定します。 このコマンドは、pubs2 データベースから発行してください。
  ```
  sp_setrowlockpromote_ptn "table", "pubs2..titles", 500, 700, 10
  ```

パーミッション

sp_setrowlockpromote_ptn は、すべてのユーザが実行できます。

参照：

- sp_setpglockpromote_ptn (184 ページ)
- パーティションロックプロモーションスレッショルドの設定 (10 ページ)

システムテーブル

SAP ASE 16.0 には、新しいテーブルと変更されたテーブルが含まれています。

変更されたシステムテーブル

SAP ASE バージョン 16.0 では、システムテーブルが変更されています。

sysattributes

sysattributes システムテーブルは、データベースの完全暗号化機能をサポートします。 暗号化データベース機能により、データベースの完全暗号化を示すための新しい systtributes クラス 43 が導入されました。 暗号化が行われるデータベースの記憶領域の割り付けごとに、SAP ASE によってロー sysattributes に次の値が挿入されます。

カラム名	値
クラス	43
オブジェクト	*dbid* (データベース ID)
object_info1	開始論理ページ ID
object_info2	終了論理ページ ID
int_value	1 つの記憶領域の割り付けにおける最終暗号化論理ページ ID

このローは、SAP ASE がデータベースの暗号化を終了すると、削除されます。

sysconstraints

16.0 より前のバージョンの SAP ASE では、sysconstraints にチェック制約、ユニークキー制約、プライマリキー制約、参照制約、ルールおよび計算カラム定義に関する情報が保存されていました。 SAP ASE 16.0 では、テーブルに関連付けられた複数のトリガに関する次の情報も sysconstraints に保存されます。

名前	データ型	説明
colid	smallint	トリガの順序番号。 デフォルトは 0。

名前	データ型	説明
constrid	int	トリガ ID。
tableid	int	トリガが宣言されたテーブルの ID。
error	int	トリガには使用されない。
status	int	0x0080 = **delete** トリガ 0x0100 = **insert** トリガ 0x0200 = **update** トリガ 0x0400 = トリガ無効
spare	int	未使用。

sysdatabases

sysdatabases システムテーブルは、データベースの完全暗号化機能をサポートします。 データベース完全暗号化機能によって、データベースの暗号化ステータスを示す新しいカラム status5 が導入されました。 値は次のとおりです。

16 進数	説明
0x00000001	データベースが暗号化されているかどうかを示す。
0x00000002	データベースが暗号化されている、および暗号化が進行中である。
0x00000004	データベースが復号化されている、および復号化が進行中である。
0x00000008	エラーによって、または処理がユーザによってサスペンドされたため、データベースの一部のみが暗号化されている。
0x00000010	エラーによって、または処理がユーザによってサスペンドされたため、データベースの一部のみが復号化されている。

sysobjects

16.0 より前のバージョンの SAP ASE の sysobjects で保存されていたものは次のとおりです。

- **delete** トリガ用の deltrig カラム
- **insert** トリガ用の instrig カラム
- **update** トリガ用の updtrig カラム

SAP ASE 16.0 では、特定のテーブルで **delete**、**insert**、および **update** 操作用として最初に作成されるトリガが order 句 (または order 0) なしで作成された場合、sysobjects 内の前述カラムのいずれかでテーブルと関連付けられます。

特定のアクション用として作成される 2 番目以降のトリガは、syscontraints のローによってテーブルに関連付けられます。 さらに、order 1 以上のすべてのトリガでは、テーブルとトリガの関連付けに常に sysconstraints が使用されます。

従属テーブルの sysobjects ローは、sysstat2 フィールドのビットを使用して、トリガが無効であることを示します。 16.0 より前のバージョンの SAP ASE 16.0 では、insert、delete、または upgrade トリガの 1 つのみに次の 3 つのビットが使用される場合がありました。

- disable_instrig 0x001000000
- disable_deltrig 0x002000000
- disable_updtrig 0x004000000

これらのビットは SAP ASE 16.0 にも存在し、テーブルのトリガ ID が前述のように sysobjects に保存されている場合に使用されます。

参照：
- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- sybmigrate (199 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

新しいシステムテーブル

SAP ASE バージョン 16.0 では、新しいシステムテーブルが導入されました。

ch_events

各設定変更イベントに 1 つのローを含みます。 ch_events は、sysmgmtdb データベース内にあります。

ch_events は、extrainfo カラムに基づいたビューです。 ch_events を表示するには、mon_role が必要です。

カラム

ch_events のカラムは次のとおりです。

名前	データ型	説明
area	varchar(10) not null	イベントが発生した領域。 下記のいずれか。 • server - サーバレベルのイベント。 • database - データベースレベルのイベント。 • cache - キャッシュレベルのイベント。 • traceflag - **dbcc traceflag** イベントと **set switch** イベント。 • SUSD - 起動またはシャットダウンの場合。 • audit - 監査ステータスの変更。
type	varchar(30) not null	監査可能イベントのタイプ。 下記のいずれか。 • sp_configure • sp_serveroption • sp_dboption • sp_cacheconfig • sp_poolconfig • create thread pool • alter thread pool • drop thread pool • dbcc traceflag • set switch • configuration file change • startup • shutdown • shutdown with wait • shutdown with nowait • abrupt shutdown • global auditing • config history auditing .
target	varchar(30) null	変更が適用されるオブジェクトの名前。
element	varchar(255) null	設定パラメータまたはその他のオプション名。

名前	データ型	説明
old-value	varchar(255) null	変更前のイベントの値。
new-value	varchar(255) null	変更後のイベントの値。
mode	varchar(10) null	設定パラメータのステータス (static または dynamic)。
time-stamp	datetime not null	イベントが発生した日付と時刻。 設定ファイル変更 (configuration file change) および突然のシャットダウン (abrupt shutdown) の場合、timestamp はイベントが発生した時間ではなくイベントが検出された時間を示す。
user-name	varchar(30) null	変更を行ったユーザの名前。 以下の場合は null に設定。 • startup • configuration file change • abrupt shutdown
in-stan-ceid	tinyint null	(Cluster Edition のみ) インスタンスの ID。

変更されたモニタリングテーブル

SAP ASE 16.0 では、既存のモニタリングテーブルに変更が加えられています。

monCachedStatement

SAP ASE 16.0 以降の monCachedStatement

- 完了済みまたは処理中のクエリについて、約 5 秒ごとに次のカラムの測定基準を更新します。

説明	データ型	属性	説明
TotalLIO	bigint	カウンタ	累積論理 I/O
TotalPIO	bigint	カウンタ	累積物理 I/O
TotalCPUTime	bigint	カウンタ	この文の実行に CPU が使用された累積経過時間 (秒単位)

説明	データ型	属性	説明
TotalElapsedTime	bigint	カウンタ	この文の実行に費やした累積時間数(秒単位)

SAP ASE 16.0 より前のバージョンでは、文の終了時に monCachedStatement の測定基準が更新されていました。 しかし、SAP ASE 16.0 以降がステートメントキャッシュを実行する場合は、クエリの実行中に次の値を定期的に更新します。

- TotalLIO
- MaxLIO
- TotalPIO
- MaxPIO
- TotalCPUTime
- MaxCPUTime
- TotalElapsedTime
- MaxElapsedTime

その他の測定基準 (MinLIO や AvgLIO など) は、クエリ実行の終了後に更新されます。

- 文が実行を開始したときに UseCount カラムを増分します。 UseCount の値は次のとおりです。

```
(number of completed queries) + (number of ongoing queries)
```

CurrentUsageCount カラムには、文のアクティブなクエリの数が含まれています。 文の完了済み実行の数は次のとおりです。

```
(Value of UseCount)  (value of CurrentUsageCount)
```

- カラム (この場合は CpuTime) で示されている測定基準が、中間更新中に使用された最大値を超えた場合は、現在実行中の文の最大値を示すカラム (MaxCPUTime など) の値を増分します。 最大値のカラムは、最新の測定基準 (アクティブクエリの測定基準など) を反映します。これは、現在実行中のクエリが以前の使用量または通常の使用量を超えるリソースを消費しているかどうかを判断する際に役立ちます。

monDeadLock

説明	データ型	属性	説明
partitionid	int	Null	パーティションのユニークな識別子

monLocks

説明	データ型	属性	説明
partitionid	int	Null	パーティションのユニークな識別子

monOpenObjectActivity

SAP ASE バージョン 16.0 以降では、monOpenObjectActivity モニタリングテーブルに次のカラムが追加されています。

説明	データ型	属性	説明
Scans	int	カウンタ	このオブジェクトに対して実行されたスキャンの数
LastScanDate	datetime	カウンタ	このオブジェクトに対する最後のスキャンの日付
Updates	int	カウンタ	このオブジェクトに対して実行された更新の数
LastUpdateDate	datetime	カウンタ	このオブジェクトに対する最後の更新の日付
Inserts	int	カウンタ	このオブジェクトに対して実行された挿入の数
LastInsertDate	datetime	カウンタ	このオブジェクトに対する最後の挿入の日付
Deletes	int	カウンタ	このオブジェクトに対して実行された削除の数
LastDeleteDate	datetime	カウンタ	このオブジェクトに対する最後の削除の日付

monOpenPartitionActivity

SAP ASE バージョン 16.0 以降では、monOpenPartitionActivity モニタリングテーブルに次のカラムが追加されています。

説明	データ型	属性	説明
Scans	int	カウンタ	このオブジェクトに対して実行されたスキャンの数
LastScanDate	datetime	カウンタ	このオブジェクトに対する最後のスキャンの日付

説明	データ型	属性	説明
Updates	int	カウンタ	このオブジェクトに対して実行された更新の数
LastUpdateDate	datetime	カウンタ	このオブジェクトに対する最後の更新の日付
Inserts	int	カウンタ	このオブジェクトに対して実行された挿入の数
LastInsertDate	datetime	カウンタ	このオブジェクトに対する最後の挿入の日付
Deletes	int	カウンタ	このオブジェクトに対して実行された削除の数
LastDeleteDate	datetime	カウンタ	このオブジェクトに対する最後の削除の日付

monProcess

説明	データ型	属性	説明
ClientDriverVersion	varchar16		クライアントプログラムが使用する接続ドライバのバージョン

monRepLogActivity、*monRepScannersTotalTime*、*および monRepSenders*
SAP ASE バージョン 16.0 以降では、モニタリングデータの収集を開始するために、monRepLogActivity、monRepScannersTotalTime、および monRepSenders モニタリングテーブルの **activate monitoring** 設定パラメータを有効にする必要があります。

monRepScanners
SAP ASE バージョン 16.0 では、monRepScanners モニタリングテーブルの Status カラムに次の変更が行われています。

- 値 spawned を削除
- 値 sync mode、async mode、および near sync mode を追加

monRepScannersTotalTime
SAP ASE バージョン 16.0 では、MRPBootstrapTime カラムの名前が BootstrapTime に変更されています。

monSysExecutionTime
SAP ASE バージョン 16.0 以降では、`OperationName` カラムに次の値が含まれています。

- NetworkIO - ネットワークデータの送受信に要した時間をレポートします。
- DeviceIO - ディスク IO 操作の実行に要した時間をレポートします。
- CIPCIO - クラスタ相互接続ネットワーク操作の実行に要した時間をレポートします (Cluster Edition のみ)。

monTables
SAP ASE 16.0 以降の `Description` カラムは 512 文字をサポートします。 以前のリリースは 255 文字をサポートしていました。

monTableColumns

- SAP ASE バージョン 16.0 以降の `Description` カラムは 512 文字をサポートします。 以前のバージョンは 255 文字をサポートしていました。
- SAP ASE バージョン 16.0 以降の `Label` カラムは 150 文字をサポートします。 以前のバージョンは 50 文字をサポートしていました。

新しいモニタリングテーブル

SAP ASE 16.0 では 新しいモニタリングテーブルが 2 つ追加されています。

monThresholdEvent

`monThresholdEvent` モニタリングテーブルには、SAP ASE により記録された各イベントにつき 1 ローが含まれます。

リソース制限収集を有効にするには、**allow resource limits** 設定パラメータを有効にします。 このモニタリングテーブルでデータを収集するには、**enable monitoring**、**threshold event monitoring**、および **set threshold event max messages** の各設定パラメータを有効にします。

`monThresholdEvent` は、ステートフルな履歴モニタリングテーブルです (『パフォーマンス＆チューニングガイド: モニタリングテーブル』を参照)。 **threshold event max messages** 設定パラメータで、`monThresholdEvent` に格納するイベントの数を指定します。

カラム

名前	データ型	属性	説明
SPID	int4		サーバプロセス ID
InstanceID	int1		クラスタ内でのインスタンスの ID。
KPID	int4		SAP ASE カーネルプロセス ID。
KTID	int4		カーネルタスクの ID。
ServerUserID	int4		この SQL テキストを実行したユーザのサーバユーザ識別子 (SUID)。 ServerUserID の値は、syslogins.suid の値に一致する。 対応する名前を取得するには **suser_name** 関数を使用する。
FamilyID	int4	NULL	親プロセスの spid。
Login	varchar(30)	NULL	ログインユーザ名。
Application	varchar(30)	NULL	アプリケーション名。
HostName	varchar(30)	NULL	クライアントホスト名。
ClientName	varchar(30)	NULL	**set clientname** で設定されたクライアント名。
ClientHostName	varchar(30)	NULL	アプリケーションによって設定された **clienthostname** プロパティの値。
ClientApplName	varchar(30)	NULL	アプリケーションによって設定された **clientapplname** プロパティの値。
ClientIP	varchar(64)	NULL	クライアントの IP アドレス。
Command	varchar(30)	NULL	プロセスのカテゴリ、またはプロセスが現在実行中のコマンド。
DBID	int4		プロセスが現在使用しているデータベースのユニークな識別子。
DBName	int4	NULL	プロセスを実行しているプログラムの名前。
ProcedureID	int4		プロシージャのユニークな識別子。
BatchID	int4		実行中の文を含む SQL バッチのユニークな識別子。

名前	データ型	属性	説明
LineNumber	int4		SQL バッチ内の現在実行されている文の行番号。
BlockingSPID	int4	NULL	このプロセスが要求したロックを保持しているプロセスのセッションプロセス識別子 (ロックを待機中の場合)。
TempDbObjects	int4	カウンタ	このプロセスによって作成されたテンポラリテーブルの合計数。
RangeID	int2		制限の範囲 ID。
LimitType	varchar(30)		制限タイプ。
LimitID	int2		制限の識別子。
limitvalue	int4		違反された制限の値。
Enforced	int1		クエリの実行前、または実行中に制限を実施するかどうかを指定する。
Action	int1		制限値を超えたときに実行されるアクション。
Scope	int1		制限の範囲。
ReportDatetime	datetime		制限違反があったためにレポートが発行された日付と時刻。
SQLText	varchar(255)		イベントの SQL テキスト。

ユーティリティ

SAP ASE バージョン 16.0 では、既存ユーティリティの一部が変更されています。

特定の 1 ユーザのディレクトリに SAP ASE をインストールすると、他のユーザはインストールされた `$SYBASE` ディレクトリに対する書き込みパーミッションがないために使用できない場合があります。

SAP ASE 16.0 では、インストール時およびインストール後に新しいサーバを設定するためのユーザを別途に指定できるので、複数のユーザが同一の SAP ASE インストールディレクトリを使用できます。

この新機能をサポートするため、SAP ASE 16.0 ではこれらのユーティリティに `-D` *`data_directory`* オプションを使用してデフォルト以外のディレクトリを指定します。

UNIX の場合:

- **auditinit**
- **sqlloc[res]**
- **sqlupgrade[res]**
- **srvbuild[res]**
- **updatease**

Windows の場合:

- **auditinit.exe**
- **sybatch.exe**

各ユーティリティについては、『ユーティリティガイド』を参照してください。

ddlgen

ddlgen ユーティリティは、SAP ASE 16.0 の新機能を追加します。

ddlgen によって次のサポートが追加されます。

- 完全暗号化データベースの作成
- データベース暗号化キーの作成

ddlgen は、次の **-TEK** パラメータの値によって、透過的なデータベースの暗号化に対するサポートを追加します。

- -XOCE - カラム暗号化キーからのみ DDL を生成します。
- -XOMK - マスタキーまたはデュアルマスタキーからのみ DDL を生成します。
- -XODE - データベース暗号化キーからのみ DDL を生成します。

参照：

- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- sybmigrate (199 ページ)
- 変更されたシステムテーブル (187 ページ)
- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

sybmigrate

SAP ASE バージョン 16.0 では、sybmigrate ユーティリティに新機能が追加されています。

内容:

- **sybmigrate** ユーティリティを使用すると、非暗号化データベースの場合と同様に、完全暗号化データベースを移行できます。 ただし、ソースデータベースと同じデータベース暗号化キー (DEK) を使用してターゲットデータベースを暗号化するには、ソースデータベースの移行前に以下の手順を実行する必要があります。
  1. **ddlgen** を使用して、マスタキー、デュアルマスタキー、および DEK の DDL を生成します (これらのキーはすべてマスタデータベースにあります)。

     ```
     $SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
     TEK -Nmaster -Dmaster -XOD
     $SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
     TEK -Ndualmaster -Dmaster -XOD
     $SYBASE/ASE-16_0/bin/ddlgen -Uusername -Ppwd -Shostname:port -
     TEK -Ndekname -Dmaster XOD
     ```

     注意： DEK がマスタキーのみで暗号化されている場合は、デュアルマスタキーの DDL を生成する必要はありません。
  2. 生成された DDL をターゲットサーバで実行して、同じ raw-key 値の DEK が生成されるようにします。
- **source_ase** または **target_ase** 属性の **ssl** キーワードを組み込むと、SSL 対応サーバへの接続が可能になります。 構文は次のとおりです。
  - リソースファイルモードの場合

    ```
    [server]
    source_ase=ssl : host_name : port_number
    source_ase_login=sa
    source_ase_password=
    ```
  - GUI モードの場合は、サーバテキストフィールドに "`ssl : host_name : port_number`" を含めます。

参照：

- データベースの完全暗号化とシステム変更 (84 ページ)
- データベースの完全暗号化に使用される create archive database (126 ページ)
- dbencryption_status (118 ページ)
- sp_helpdb (171 ページ)
- sp_encryption (169 ページ)
- ddlgen (198 ページ)
- 変更されたシステムテーブル (187 ページ)

- データベースの完全暗号化に使用される alter database (119 ページ)
- データベースの完全暗号化に使用される create database (127 ページ)
- create encryption key (130 ページ)
- drop encryption key (159 ページ)

sybrestore

sybrestore ユーティリティは、マスタデータベースの破損後の SAP ASE サーバのリストアをサポートします。

対話型モードおよび非対話型モードの追加パラメータが追加されました。

対話型モード

```
sybrestore
    [-J character set ]
    [-R  Restore from master database corruption ]
    [-d dump directory ]
    [-s list system databases except master database ]
    [-v version ]
    [-z language ]
    [-o Log output]
```

非対話型モード

```
sybrestore
    [-o Log output]
```

『ユーティリティガイド』の「**sybrestore**」を参照してください。

グローバル変数

SAP ASE バージョン 16.0 ではグローバル変数が追加されています。

グローバル変数	説明
@@trigger_name	実行中のトリガ名を返す。

グローバル変数	説明

@@tranroll-back	ロールバックが発生している場合、ロールバックのタイプを返す。 それぞれの戻り値の意味は以下のとおり。
	• < 0 - 複数文トランザクションの暗黙的ロールバックをサーバが引き起こした。 **@@tranrollback** は、暗黙的トランザクションロールバックをもたらしたエラー番号の符号を逆にした値を格納する。
	• 0 - 現在アクティブなトランザクションのこのセッションでは暗黙のロールバックは起こっていない。
	• > 0 < 10 - 一番最近のトランザクションロールバックは、以下の SQL コマンドの 1 つによるユーザ発行のロールバックだった。
	  • **rollback tran** (SQL バッチ、プロシージャ、またはトリガ内)
	  • **rollback trigger** (トリガ範囲外)
	  @@transtate の戻り値が、ユーザの発行した **rollback** コマンドを示す。
	  • 1 - ユーザは明示的な **rollback tran** コマンドを発行した。
	  • 2 - ユーザは **rollback tran to***savepoint* を発行した。 トランザクションはまだアクティブである。
	• > 100 - 一番最近のトランザクションロールバックは、単一文トランザクションで起動された。 **@@transtate** は、文のロールバックをもたらしたエラー番号を格納する。
	SAP ASE は、次の **rollback tran** または **commit tran** が発行されるまで、**@@tranrollback** の負の値を変更しない。これにより、セッションに暗黙のトランザクションロールバックが発生したことを示す。 次の **rollback tran** または **commit tran** の適用が成功したら、SAP ASE は **@@tranrollback** を 0 にリセットする。 この例の最後では **@@tranrollback** の値は 0 になる。
	`set chained on`
	`go`
	`<... Execute a DML statement ...>`
	`if (@@error != 0) and (@@tranrollback < 0)`
	`begin`
	`  rollback tran`
	`end`
	`go`