SYBASE®
An SAP® Company

暗号化カラム・ユーザーズ・ガイド

Adaptive Server® Enterprise

15.7

ドキュメント ID：DC00544-01-1570-01
改訂：2011 年 9 月

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

目次

第 1 章

暗号化の概要

このマニュアルでは、Adaptive Server® Enterprise の暗号化カラム機能、外部パスワードの暗号化、ストアド・プロシージャ、ビュー、およびトリガの SQL テキストについて説明します。

Adaptive Server 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、正しい権限を持つユーザのみがデータにアクセスできます。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。

Adaptive Server の暗号化カラム機能を使用すると、アプリケーションを変更しなくても、静止しているカラムレベルのデータを暗号化できます。次の機能がネイティブでサポートされています。

- カラムレベルの細分化
- NIST (National Institute of Standards and Techology) が承認した対称 AES (Advanced Encryption Standard) アルゴリズムの使用
- パフォーマンスの最適化
- 作業の分割方式の実行
- 完全統合および自動化キー管理
- アプリケーションの透過性 (アプリケーションの変更は不要)
- システム管理者の権限からのデータのプライバシー保護

データの暗号化と復号化は、自動的で透過的です。テーブルに対する insert または update パーミッションを持っていれば、挿入または修正したすべてのデータは格納される前に自動的に暗号化されます。日常の作業が中断されることはありません。

暗号化カラムから復号化されたデータを選択するには、select パーミッションに加え、decrypt パーミッションも必要です。decrypt パーミッションは、特定のデータベース・ユーザ、グループ、または役割に対して付与できます。Sybase® には、機密データに対する細分化されたアクセス制御機能があり、より詳細な制御を実行できます。また、decrypt パーミッションを持つユーザに対して選択されたデータが自動的に復号化されます。

Adaptive Server におけるカラムの暗号化は、中間層やクライアント・アプリケーションにおける暗号化よりも簡素化されます。SQL 文を使用して暗号化キーの作成と暗号化カラムの指定を行うことができ、既存のアプリケーションを変更せずに継続して実行できます。

カラム暗号化キーは、暗号化された形式でデータベースに格納されます。以下から導出されたキー暗号化キー (KEK) を使用してカラム暗号化キーを暗号化できます。

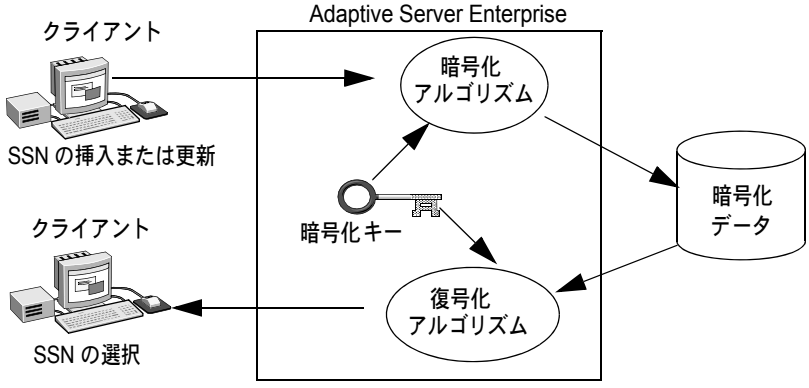
- システムレベルのユーザが指定したパスワード
- ユーザが指定したパスワードから導出された KEK (ユーザのログイン・パスワードとして使用できます)
- 個別に作成したデータベースレベルの KEK (マスタ・キーまたはデュアル・マスタ・キー)

選択したパスワードによって、システム管理者からもデータのプライバシーを保護できます。デュアルコントロール・モードを使用してカラム暗号化キーを保護し、セキュリティを強化できます。「第 2 章 カラム暗号化キーの作成と管理」を参照してください。

暗号化されたデータは、「暗号テキスト」と呼ばれるコード化された形式で格納されます。暗号テキストでは、暗号化カラムの長さに数バイトから最大 32 バイトまでの長さが追加されます。「暗号化カラムの長さ」(54 ページ) を参照してください。暗号化されていないデータは、プレーン・テキストとして格納されます。

図 1-1 は、Adaptive Server の暗号化処理と復号化処理について説明しています。この例では、クライアントが社会保障番号 (SSN) の更新と暗号化を行っています。

図 1-1: Adaptive Server での暗号化と復号化



カラムの暗号化では、対称暗号化アルゴリズムが使用されます。これは、暗号化と復号化に同じキーが使用されることを意味します。特定のカラムを暗号化するキーは、 Adaptive Server によって追跡されます。

暗号化カラムに対してデータの insert または update を行うとき、Adaptive Server はローを書き込む直前にデータを透過的に暗号化します。暗号化カラムに対して select を実行すると、ローから該当データが読み込まれ、復号化されます。整数データと浮動小数点データは、すべてのプラットフォームで次の形式で暗号化されます。

- 整数データの場合は最上位ビット形式
- 浮動小数点データの場合は、MSB 形式に対応する IEEE (Institute of Electrical and Electronics Engineers) 浮動小数点標準

あるプラットフォームでデータを暗号化し、別のプラットフォームで復号化できます。ただし、両方のプラットフォームで同じ文字セットを使用している必要があります。

一般的に、暗号化カラムを使用するには、次の手順を実行する必要があります。

1　ライセンス・オプション ASE_ENCRYPTION をインストールします。Adaptive Server Enterprise の『インストール・ガイド』を参照してください。

2　システム・セキュリティ担当者 (SSO) は、次のコマンドを使用して、Adaptive Server での暗号化を有効にすることができます。

```
sp_configure 'enable encrypted columns', 1
```

3　カラム暗号化キーの保護方法に応じて、データベースレベルのマスタ・キーを作成するか、システム暗号化パスワードを設定します。マスタ・キーの作成の詳細については、「マスタ・キーとデュアル・マスタ・キーの作成」(20 ページ) を参照してください。

「システム暗号化パスワードを使用したカラム暗号化キーの保護」(15 ページ) では、システム暗号化パスワードの設定方法について説明します。

4　1 つ以上の名前付き暗号化キーを作成します。「第 2 章 カラム暗号化キーの作成と管理」を参照してください。パスワードを使用してデータベース管理者からもデータを保護することを検討します。「第 7 章 管理者からのデータのプライバシーの保護」 を参照してください。

5　暗号化するカラムを指定します。詳細については、「新しいテーブルに対する暗号化の指定」(40 ページ) と 「既存テーブルのデータの暗号化」(42 ページ) を参照してください。

6　データを確認する必要があるユーザに decrypt パーミッションを付与します。「復号化デフォルト値」と呼ばれるデフォルトのプレーン・テキスト値を指定できます。Adaptive Server は、decrypt パーミッションを持たないユーザに対して保護されたデータではなくこのデフォルト値を返します。詳細については、「復号化のためのパーミッション」(59 ページ) を参照してください。

これらの手順を実行すると、既存のテーブルやカラムに対して既存のアプリケーションを実行できますが、データベース内のデータは盗難や不正使用に対して確実に保護されます。Sybase Central のユーティリティやその他の Sybase 製品ではデータを暗号化された形式で処理し、企業全体でデータを保護できます。たとえば次のことを実現できます。

- Sybase Control Center または Sybase Central Adaptive Server プラグインを使用して、グラフィカル・インタフェースで暗号化カラムを管理できます。オンライン・ヘルプを参照してください。
- バルク・コピー・ユーティリティ (bcp) を使用して、サーバとの間で暗号化データを安全にコピー・インおよびコピー・アウトできます。『ユーティリティ・ガイド』を参照してください。
- Adaptive Server のマイグレーション・ツール sybmigrate を使用して、サーバ間でデータを安全にマイグレートできます。詳細については、Adaptive Server Enterprise の『システム管理ガイド』を参照してください。
- Sybase Replication Server を使用して、暗号化キーおよびデータをサーバとプラットフォームに安全に配布できます。複写時の暗号化の詳細については、『Replication Server 管理ガイド』を参照してください。

外部パスワードと隠し SQL テキストの強力な暗号化の詳細については、「第 4 章 外部パスワードと隠しテキストのセキュリティ保護」を参照してください。

第 2 章

カラム暗号化キーの作成と管理

Adaptive Server には、カラム暗号化キーの作成、カラム暗号化キーのプロパティの変更、および使用されていないカラム暗号化キーの削除を実行するためのコマンドが用意されています。キー所有者は、特定のキーまたは複数のキーを使用してカラム・レベルで暗号化を設定するためのパーミッションをテーブル所有者に付与する必要があります。

カラム暗号化キーを使用したデータの保護

Adaptive Server では、キーが使用中でない場合、暗号化された状態で保持されます。ユーザは、暗号化データにアクセスする際にカラム暗号化キー (CEK) にアクセスする必要がありますが、CEK はディスクまたはメモリに格納される前に暗号化される必要があります。Adaptive Server はキー暗号化キー (KEK) を使用して CEK を暗号化し、暗号化された形式で sysencryptkeys に保存します。KEK は、CEK を復号化するためにも使用され、ユーザが暗号化データにアクセスすることができます。「キー暗号化」(13 ページ) を参照してください。

CEK 管理では、カラム暗号化キーの作成、削除、および変更、パスワードの配布、キー・コピーの作成、そしてパスワードを忘れた場合のキー・リカバリの提供を行う必要があります。

カラム暗号化キーの作成

テーブル所有者がカラムを新しいテーブルまたは既存のテーブルの暗号化用として指定するには、カラム暗号化キーが必要です。暗号化キーを初めて設定する場合は、次の点について考慮してください。

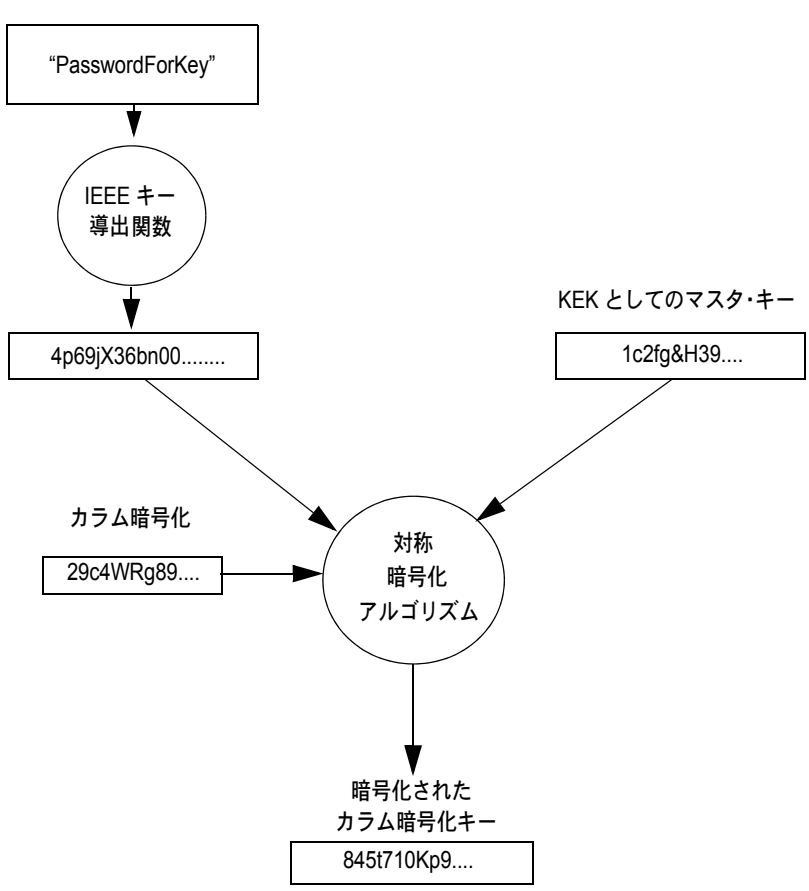
- キー所有者またはキー管理者の割り当て – システム・セキュリティ担当者 (SSO) は、キーを作成するための create encryption key パーミッションを付与する必要があります。デフォルトでは、sso_role と keycustodian_role は、create encryption key パーミッションを持っています。「キー管理者の役割」(61 ページ) を参照してください。

- キーを別のキー・データベースで作成する必要があるかどうか － 特に、システム暗号化パスワードでキーを暗号化する場合は、キー用に別のデータベースを使用することをおすすめします。
- 必要なキーの数 － 暗号化カラムごとに個別のキーを作成することも、同じキーを使って複数のテーブルのカラムを暗号化することもできます。パフォーマンスの観点では、別のテーブルの同じカラムとジョインしている暗号化カラムは同じキーを共有する必要があります。セキュリティ目的では、関連付けられていないカラムは異なるキーを使用する必要があります。

Adaptive Server のカラム暗号化では、AES (Advanced Encryption Standard) 対称キー暗号化アルゴリズムが使用されます。使用可能なキー・サイズは、128、192、256 ビットです。ランダムキーの生成と暗号化の機能は、FIPS 140-2 準拠のモジュールによって提供されます。

キー値を保護するために、Adaptive Server は 256 ビットのキー暗号化キー (KEK) を使用します。このキーは、マスタ・キーか、システム暗号化パスワードまたはユーザ指定のパスワードから取得される内部キーです。「第 3 章 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。Adaptive Server は新しいキー (カラム暗号化キー) を暗号化し、その結果を sysencryptkeys に保存します。

図 2-1: KEK を使用したカラム暗号化キーの暗号化



デフォルトでは、Adaptive Server は 256 ビットのキー暗号化キーを作成します。15.7 以前のバージョンとの互換性を保つために、KEK がシステム暗号化パスワードから抽出されている場合は 128 ビット・キーを使用します。

create column encryption key の構文

create column encryption key の構文は次のとおりです。

```
create encryption key [[database.][owner].]keyname
    [as default] [for algorithm]
    [with
        {[key_length num_bits]
        [{passwd 'passwd_phrase' | passwd system_encr_passwd |
            master key}]
        [init_vector {null | random}]
        [pad {null | random}]
        [[no] dual_control]
    }]
```

各パラメータの意味は、次のとおりです。

- *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な名前を使用してください。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。owner のデフォルト値は現在のユーザで、database の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。

  注意 "#" で始まるテンポラリ・キー名を作成することはできません。

- as default − システム・セキュリティ担当者またはキー管理者は、暗号化のためのデータベースのデフォルト・キーを作成できます。これにより、テーブルの作成者が、create table、alter table、select into で keyname を使用せずに暗号化を指定できます。Adaptive Server は同じデータベースのデフォルト・キーを使用します。デフォルト・キーは変更できます。
- *for algorithm* − AES (Advanced Encryption Standard) のみがサポートされます。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。
- keylength *num_bits* − 作成するキーのサイズです。ビット単位で指定します。AES の有効なキー長は 128、192、256 ビットです。デフォルトの keylength は 128 ビットです。
- passwd *password_phrase* − ASE に対して、ユーザ・パスワード *password_phrase* を使用して CEK を保護するよう指示します。ユーザ・パスワードには、引用符で囲まれた最長 255 バイトの英数字文字列を指定できます。
- passwd *system_encr_passwd* − ASE に対して、システム暗号化パスワードを使用して CEK を保護するよう指示します。
- master key − ASE に対して、マスタ・キーを使用して CEK を保護するよう指示します。デフォルトでは、Adaptive Server はマスタ・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。「キーの保護」(11 ページ) を参照してください。
- 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ます。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。

    初期化ベクトルを使用すると、CBC (Cipher Block Chaining) モードが暗号化に使用されます (このモードでは、データの各ブロックを前のブロックと結合してから暗号化します。先頭のブロックは初期化ベクトルと結合されます)。

ただし、初期化ベクトルを使用すると、パフォーマンスに影響が生じる場合があります。インデックス作成によるカラムのジョインと検索の最適化は、暗号化キーで初期化ベクトルが指定されていないカラムに対してのみ行えます。「第 10 章 パフォーマンスの考慮事項」を参照してください。

- null − 暗号化中に初期化ベクトルを使用しません。この指定により、カラムがインデックスに対応できるようになります。

  デフォルトは、初期化ベクトルの使用、つまり init_vector random です。

  init_vector null を設定すると、ECB (Electronic Code Book) モードが使用されます。このモードでは、データの各ブロックが個別に暗号化されます。

  あるカラムは初期化ベクトルを使用して暗号化し、別のカラムは初期化ベクトルを使用しないで暗号化するには、初期化ベクトルを指定するキーと、初期化ベクトルを指定しないキーの 2 つのキーを作成します。

- pad

  - null − デフォルト値で、データのランダム埋め込みを行いません。

    カラムでインデックスをサポートする必要がある場合は、埋め込みを使用できません。

  - random − ランダムなバイトをデータに自動的に埋め込んでから暗号化します。暗号テキストのランダム化のために初期化ベクトルではなく埋め込みを使用できます。埋め込みは、プレーン・テキストの長さがブロック長の半分より短いカラムにのみ適切です。AES アルゴリズムのブロック長は 16 バイトです。

- 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。「第 3 章 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。

create encryption key の例

次の例では、カラム暗号化キーを作成する際にさまざまな暗号化属性を使用します。多くの場合、既にマスタ・キーを作成しているか、システム暗号化パスワードを設定済みであることを前提としています (「キーの保護」(11 ページ)を参照してください)。

- 例 1 − “safe_key” という 256 ビットのキーをデータベースのデフォルト・キーとして指定します。このキーはパスワードを指定しないため、Adaptive Server では safe_key の KEK としてデータベースレベルのマスタ・キーが使用されます。マスタ・キーがない場合は、システム暗号化パスワードが使用されます。

```
create encryption key safe_key as default for AES
    with keylength 256
```

システム・セキュリティ担当者または keycustodian_role を持つユーザは、デフォルト・キーを作成できます。

- 例 2 – ランダム埋め込みを使用してカラムを暗号化する、“salary_key” という 128 ビットのキーを作成します。

```
create encryption key salary_key for AES with
     init_vector null pad random
```

- 例 3 – カラムの暗号化に初期化ベクトルを使用する、名前が “mykey” の 192 ビットのキーが作成されます。

```
create encryption key mykey for AES
     with keylength 192 init_vector random
```

- 例 4 – ユーザ指定のパスワードによって保護されるキーを作成します。

```
create encryption key key1
     with passwd 'Worlds1Biggest6Secret'
```

キーがユーザ指定のパスワードで保護されている場合、キーで暗号化されたカラムにアクセスする前に、そのパスワードを入力する必要があります。明示的なパスワードが設定されたキーを使用する方法の詳細については、「第 7 章 管理者からのデータのプライバシーの保護」を参照してください。

- 例 5 – デュアル・コントロールによって保護されるキーを作成します。

```
create encryption key dualprotectedkey
with passwd "Pass4Tomorrow"
dual_control
```

キー “dualprotectedkey” は、マスタ・キーとユーザ・パスワード (デュアル・コントロールを使用) で保護されます。キーにアクセスするには、キーのユーザ・パスワードとマスタ・キーのパスワードの両方を入力する必要があります。

create encryption key パーミッション

sso_role と keycustodian_role は暗号化キーを作成するためのパーミッションを暗黙で持っています。システム・セキュリティ担当者またはキー管理者は次の構文を使用して、create encryption key パーミッションを他のユーザに付与します。

```
grant create encryption key
     to user_name | role_name | group_name
```

次に例を示します。

```
grant create encryption key to key_admin_role
```

キーを作成するパーミッションを取り消すには、次の構文を使用します。

```
revoke create encryption key
    {to | from} user_name | role_name | group_name
```

注意 grant all コマンドでは、create encryption key パーミッションはユーザに付与されません。このパーミッションは、システム・セキュリティ担当者が明示的に付与する必要があります。

キーの保護

キー管理者は、キーを保存する場所、キーを更新する時期、および特定のキーを使用してテーブルのカラムを暗号化できるテーブル所有者を決定する必要があります。

キーへのアクセス権付与

キー所有者または sso_role を持つユーザがキーの select パーミッションを付与しないと、他のユーザは create table 文、alter table 文、select into 文でキーを指定できません。キー所有者は、システム・セキュリティ担当者、キー管理者、またはデフォルト以外のキーの場合は、create encryption key パーミッションを持つ任意のユーザになります。キー所有者は、必要に応じてキーの select パーミッションを付与してください。

次の例では、db_admin_role を持つユーザが create table、alter table、select into 文で暗号化を指定するときに、“safe_key” という名前の暗号化キーを使用できるようにします。

```
grant select on safe_key to db_admin_role
```

注意 insert、update、delete、select で暗号化カラムを処理するユーザには、暗号化キーの select パーミッションは必要ありません。

キーの変更

カラムの暗号化に使用されるキーを定期的に変更してください。create encryption key を使用して新しいキーを作成し、alter table...modify を使用して新しいキーでカラムを暗号化してください。

次の例では、“creditcard” カラムがすでに暗号化されています。alter table コマンドによって、customer テーブルで cc_key_new が使用されているすべてのローのクレジット・カード値が復号化され、再暗号化されます。

```
create encryption key cc_key_new for AES

alter table customer modify creditcard encrypt with
      cc_key_new
```

キーとデータとの分離

暗号化するカラムを指定するときに、同じデータベースまたは別のデータベースの名前付きキーを指定して使用できます。指定のキーを使用しない場合、そのカラムは自動的に同じデータベースのデフォルト・キーで暗号化されます。

別のデータベースのキーを使用して暗号化すると、セキュリティが強化されます。データベース・ダンプが盗まれた場合でも、キーと暗号化されたデータ両方へのアクセスを防止できるためです。管理者は、データベース・ダンプごとに異なるパスワードを設定して保護できます。これにより、不正アクセスがより困難になります。

別のデータベースのキーを使用して暗号化する場合、分散システムにおいてデータとキーの整合性の問題が起こらないように注意する必要があります。データベースのダンプとロードを注意して調整してください。別のデータベースの名前付きキーを使用する場合、次のようにすることをおすすめします。

- 暗号化カラムを含むデータベースをダンプするときは、対応するキーが作成されたデータベースもダンプします。最後のダンプ以降に新しいキーを追加した場合は、これを行う必要があります。
- 暗号化キーを含むデータベースをダンプするときは、そのキーで暗号化されたカラムを含むすべてのデータベースをダンプします。これによって、暗号化データと使用可能なキーの同期が保たれます。

システム・セキュリティ担当者またはキー管理者は、sp_encryption を使用して、特定のキーで暗号化されたすべてのカラムを識別できます。

暗号化キーの削除

暗号化キーを削除するには、次の文を使用します。

drop encryption key [[*database*.][*owner*.]*keyname*

次の例では、cc_key という名前の暗号化キーが削除されます。

```
drop encryption key cust.dbo.cc_key
```

キー所有者は自分が所有しているキーを削除できます。システム・セキュリティ担当者はどのキーでも削除できます。該当するキーが使用されているすべてのデータベースに暗号化カラムが存在しない場合のみ、そのキーを削除できます。

drop encryption key を実行するときに、Adptive Server では、疑わしいデータベース、アーカイブされたデータベース、リカバリされていないデータベース、または現在ロードされているデータベース内の暗号化カラムはチェックされません。これらのいずれの場合でも、このコマンドによって、使用不可のデータベースの名前を示した警告メッセージが発行されますが、コマンド自体は正常に実行されます。該当するデータベースがオンラインになると、削除されたキーによって暗号化されていたカラムを持つすべてのテーブルが使用不可になります。このキーをリストアするには、システム管理者が、削除したキーのデータベースのダンプ (キーを削除する前のもの) をロードする必要があります。

システム・セキュリティ担当者は、sp_encryption を使用して、特定のキーで暗号化されたすべてのカラムを識別できます。

キー暗号化

ユーザとデータの間には、実際には 2 つのキーが存在します。1 つはカラム暗号化キー (CEK) で、もう 1 つはキー暗号化キー (KEK) です。CEK はデータを暗号化します。ユーザは、暗号化データにアクセスする際に CEK にアクセスする必要があります。しかし、暗号化されていない形式で CEK をディスクに格納できません。そのため、ユーザが暗号化キーを作成または変更したとき、CEK は 1 個の KEK またはデュアル・コントロールであれば 2 個の KEK を使用して暗号化されます。KEK は、ユーザが暗号化データにアクセスするときに CEK の復号化も行います。CEK は、暗号化された形式で sysencryptkeys に格納されます。

KEK は、システム・セキュリティ担当者またはキー管理者によって個別に作成されたマスタ・キーで、create 文と alter encryption key 文でキーの暗号化を指定した方法に応じて、システム暗号化パスワード、ユーザ定義のパスワード、またはログイン・パスワードから内部的に導出されたキーです。システム暗号化パスワードとマスタ・キーはどちらも暗号化された形式で格納されます。

図 2-2 (14 ページ) では、create encryption key 文のカラム暗号化キーの作成および格納について説明します。KEK がパスワードから導出され、ロー CEK が暗号化関数に渡されて、暗号化 CEK が生成されます。

図 2-2: 暗号化キーの作成手順

```
create encryption key. . .
```

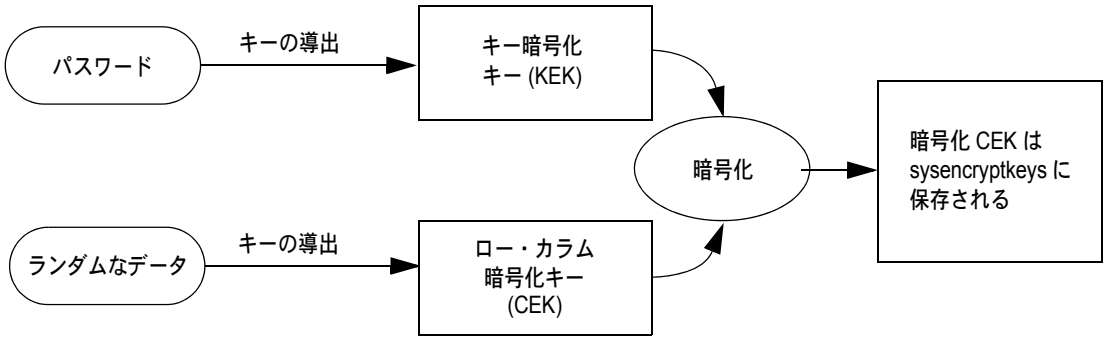
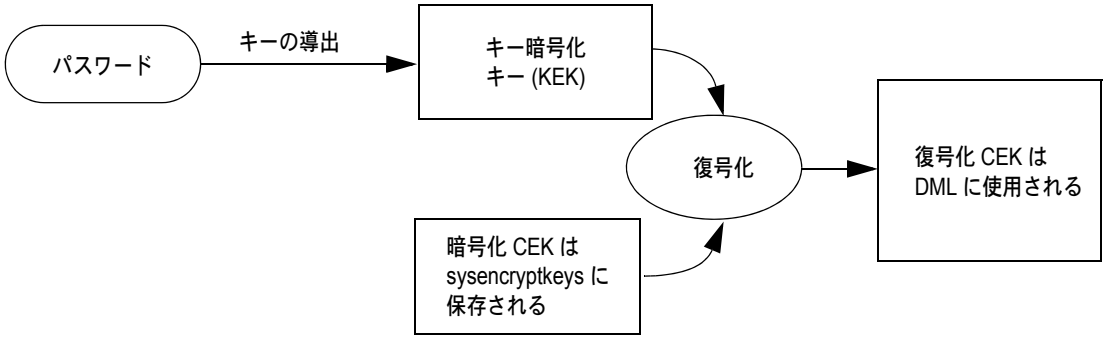


図 2-3 は、CEK を復号化する DML オペレーションでの KEK の使用方法を示します。ロー CEK は、データを暗号化または復号化するために使用されます。

図 2-3: DML 文で暗号化または復号化するための CEK へのアクセス



マスタ・キーを使用したカラム暗号化キーの保護

マスタ・キーは、sso_role または keycustodian_role のユーザにより作成されたデータベースレベルのキーで、ユーザが作成した暗号化キーに対する KEK として使用されます。作成したマスタ・キーは、ユーザが作成したキーのデフォルト KEK としてシステム暗号化パスワードに置き換わります。Adaptive Server では、15.7 以前のバージョンとの互換性を保つためにシステム暗号化パスワードをサポートしていますが、マスタ・キーを使用することをおすすめします。

デュアル・マスタ・キーとともにマスタ・キーを使用し、ユーザが作成したすべてのキーに対してデュアル・コントロールと知識分割を提供する複合キーを作成できます。また、CEK の明示的なパスワードでマスタ・キーを使用して複合キーを作成することもできます。

マスタ・キーを使用すると、暗号化データの管理が簡略化されます。その理由を以下に示します。

- キーのパスワードの管理がマスタ・キーのパスワードの設定に制限されます。
- create と alter encryption key 文でパスワードを指定する必要がありません。
- パスワード配布およびカラム暗号化キーのパスワード紛失の際のリカバリができます。
- カラムの decript パーミッションにより暗号化されたデータのアクセス・コントロールが適用されます。「decrypt パーミッションの制限」(46 ページ)を参照してください。
- アプリケーションを変更する必要はありません。

マスタ・キーを作成するための構文は、次のとおりです。

```
create encryption key master
    [for AES] with passwd char_literal
```

『リファレンス・マニュアル：コマンド』を参照してください。

システム暗号化パスワードを使用したカラム暗号化キーの保護

システム暗号化パスワードは、データベース固有のパスワードで、CEK のセカンダリのデフォルトの暗号化方法です。つまり、データベースのマスタ・キーを作成する必要はありません。Adaptive Server は、システム暗号化パスワードを使用して、指定したデータベースで作成されたキーを明示的な password 句を使用せずに暗号化します。システム・セキュリティ担当者またはキー管理者がシステム暗号化パスワードを設定した後は、暗号化カラムを処理するときにこのパスワードを指定する必要がなくなります。Adaptive Server は、カラム暗号化キーを暗号化または復号化する必要があるときに、システム暗号化パスワードに内部でアクセスします。

システム・セキュリティ担当者またはキー管理者は、sp_encryption を使用してシステム暗号化パスワードを設定します。システム・パスワードは sp_encryption を使用するデータベースに固有であり、暗号化された値はそのデータベースの sysattributes システム・テーブルに格納されます。

```
sp_encryption system_encr_passwd, password
```

password には、最大 255 バイトのパスワードを指定できます。

システム暗号化パスワードは、暗号化キーを作成したすべてのデータベースにだけ設定してください。

システム暗号化パスワードによって、暗号化キーが保護されます。長くて複雑なシステム暗号化パスワードを選択してください。パスワードが長くなれば、当て推量によって推測または解読されにくくなります。システム暗号化パスワードには、大文字と小文字、数字、および特殊文字を含めます。システム暗号化パスワードの長さは 16 バイト以上にすることをおすすめします。

Adaptive Server では、minimum password length および check password for digit 設定パラメータを使用してシステム暗号化パスワードが順守されます。

sp_encryption に古いパスワードを指定して、システム・パスワードを変更します。

```
sp_encryption system_encr_passwd, password [ , old_password]
```

特にシステム暗号化パスワードを把握している管理者が退社する場合は、システム暗号化パスワードは定期的に変更してください。システム・パスワードが変更されると、Adaptive Server は自動的にデータベース内のすべてのキーを新しいパスワードで再暗号化します。暗号化されたデータは、システム・パスワードが変更されても影響を受けません。つまり、データは復号化されたり、再暗号化されたりすることはありません。

システム暗号化パスワードの設定を解除するには、*password* の引数に “null” を指定し、*old_password* の値を指定します。システム・パスワードは、システム暗号化パスワードで暗号化されたそのデータベース内のすべての暗号化キーを削除している場合にのみ設定解除します。

ユーザ指定のパスワードを使用したカラム暗号化キーの保護

create encryption key または alter encryption key を使用してキーにパスワードを指定すると、プライベート・データにアクセスするシステム管理者またはデータベース所有者の権限を制限できます。キーに明示的なパスワードが設定されている場合、データを復号化するには以下が必要です。

- カラムの decrypt パーミッション
- 暗号化キーのパスワード

ユーザは、データを暗号化する DML コマンドを実行するためのパスワードも知っている必要があります。

「ユーザ指定のパスワードを使用したキーの保護」(64 ページ) を参照してください。

デュアル・コントロールを使用したカラム暗号化キーの保護

create encryption key コマンドを使用したデュアル・コントロールで、カラム暗号化キーのセキュリティを確保できます。

create encryption key with dual_control を指定して、ユーザ・パスワードを指定しない場合、カラム暗号化キーはマスタ・キーとデュアル・マスタ・キーで保護されます。

with dual_control を指定してユーザ・パスワードを含めると、カラム暗号化キーはマスタ・キーとユーザ・パスワードで保護されます。

- 例 1 － マスタ・キーとデュアル・マスタ・キーの両方を使用して CEK の “Reallysecret” を保護します。両方のキーがデータベースに存在しないと失敗します。

```
create encryption key Reallysecret
     with init_vector random dual_control
```

- 例 2 － マスタ・キーとユーザ・パスワード “Whybother” の両方を使用して CEK “k3” を暗号化します。

```
create encryption key k3
     with passwd 'Whybother'
     dual_control
```

既存のキーを変更してデュアル・コントロールを使用するには、「キーの保護方法の変更」(65 ページ) を参照してください。

第 3 章

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

Adaptive Server では、マスタ・キーとデュアル・マスタ・キーと呼ばれるデータベースレベルの暗号化キーを作成できます。どちらのキーもキー暗号化キーとして動作し、カラム暗号化キーやサービス・キーなどのその他のキーを保護するために使用されます。マスタ・キーを作成すると、カラム暗号化キーのデフォルトの保護方法になります。デュアル・マスタ・キーは、カラム暗号化キーのデュアル・コントロールに対してのみ必要です。

マスタ・キーとデュアル・マスタ・キーを作成できるのは、sso_role または keycustodian_role を持っているユーザだけです。マスタ・キーとデュアル・マスタ・キーに設定できるのは、1 つのデータベースで 1 つのみです。

マスタ・キーとデュアル・マスタ・キーの所有者は別々である必要があります。isql を使用するか、Adaptive Server でのみアクセスできるサーバ・プライベート・ファイルを経由して、マスタ・キーのパスワードを指定できます。これらのキーのパスワードはデータベースに格納されません。

マスタ・キーとデュアル・マスタ・キーの作成

マスタ・キーとデュアル・マスタ・キーを作成するには、次の構文を使用します。

```
create encryption key [dual] master
    [for AES] with passwd char_literal
```

各要素の意味は次のとおりです。

- master と dual master は、それらが定義されているデータベース内のその他のキーの暗号化に使用するデータベースレベルのキーを参照します。これらのキーはデータの暗号化には使用されません。マスタ・キーは、sysobjects 内で sybencrmasterkey という名前が付けられ、デュアル・マスタ・キーは sysobjects 内で sybencrdualmasterkey という名前が付けられます。
- with passwd の後ろには、sp_passwordpolicy に従う文字列パスワードを続ける必要があります。

『リファレンス・マニュアル：コマンド』を参照してください。

- 例 1 – データベース tdb1 にマスタ・キーを作成します。

```
use database tdb1
create encryption key master with passwd
'unforgetablethatswhatyouare'
```

- 例 2 – データベース tdb1 にデュアル・マスタ・キーを作成します。

```
use database tb1
create encryption key dual master with passwd
'dualunforgettable'
```

- 例 3 – マスタ・キーをカラム暗号化キーとして使用することはできないので、エラーが生成されます。

```
create table t2 (c1 int encrypt with master)
```

マスタ・キーまたはデュアル・マスタ・キーのパスワードを変更するには、次の構文を使用します。

```
alter encryption key [dual] master
    with passwd <char_literal>
    modify encryption
    with passwd <char_literal>
```

マスタ・キー・コピーの作成

sso_role または keycustodian_role を持つユーザまたはマスタ・キー所有者は、マスタ・キーのコピーを作成できます。コピーは、次の操作を行う際に必要になる可能性があります。

- マスタ・キーまたはデュアル・マスタ・キーにアクセスして Adaptive Server を無人起動できるようにする。このようなキー・コピーを automatic_startup コピーと呼びます。
- パスワードが失われた場合、マスタ・キーのリカバリをサポートする。このようなキー・コピーをリカバリ・コピーと呼びます。「第 8 章 失われたパスワードからのキーのリカバリ」を参照してください。
- ベース・キーの所有者ではないユーザがマスタ・キーまたはデュアル・マスタ・キーの暗号化パスワードを設定できるようにする。このキー・コピーを通常コピーと呼びます。

マスタ・キーのコピーをデータベースに追加するには、次の構文を使用します。

```
alter encryption key [dual] master
    with passwd char_string
    add encryption
    {with passwd char_string
    for user user_name
    [ for recovery ] | [ for automatic_startup ] }
```

それぞれの意味は、次のとおりです。

- *char_string* − (最初の参照) マスタ・キーまたはデュアル・マスタ・キーのベース・コピーを現在暗号化しているパスワードを指定します。
- *char_string* − (2 番目の参照) 通常コピーまたはリカバリ・コピーのパスワードを指定します。automatic_startup コピーには使用しないでください。
- for user − 通常コピーまたはリカバリ・コピーが割り当てられる必要のあるユーザを指定します。このパラメータを使用して、automatic_startup コピーのパスワードを入力しないでください。
- for recovery − パスワードを紛失した場合に、キー・コピーを使用してマスタ・キーをリカバリすることを示します。
- for automatic_startup − 自動マスタ・キー・アクセスを有効にしてサーバが再起動された後にキー・コピーを使用してマスタ・キーまたはデュアル・マスタ・キーにアクセスすることを示します。
- 例 1 − マスタ・キーの所有者が Mary のキー・コピーを作成します。

```
alter encryption key master
     with passwd 'unforgettablethatswhatur'
     add encryption
     with passwd 'just4now'
     for user mary
```

- 例 2 – デュアル・マスタ・キーの所有者 Smith が次の構文を使用して automatic_startup のキー・コピーを作成します。

```
alter encryption key dual master
     with passwd 'Never4Getable'
     add encryption
     for automatic_startup
```

マスタ・キーおよびデュアル・マスタ・キーのパスワードの設定

ベース・キーの所有者、または通常のキー・コピーを所有するユーザは、マスタ・キーとデュアル・マスタ・キーのパスワードを設定できます。パスワードは、マスター・キーを使用する前に設定する必要があります。マスタ・キーのパスワードを設定するには、次のいずれかを使用します。

- set encryption passwd コマンド
- 無人起動機能
- (マスタ・キーのみ) dataserver コマンド

set encryption コマンドは次のとおりです。

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

各要素の意味は次のとおりです。

- *char_literal* – ユーザがキーの所有者である場合は、マスタ・キーまたはデュアル・マスタ・キーのベース・コピーを現在暗号化しているパスワードです。ユーザがキーの所有者でない場合は、キーのユーザのコピーを現在暗号化しているパスワードです。

例 – データベース tdb1 のマスタ・キーのパスワード “MasterSecret” を設定します。

```
use tdb1
set encryption passwd 'MasterSecret' for key master
```

Adaptive Server では、マスタ・キーまたはデュアル・マスタ・キーが定義されるデータベースのサーバ・メモリにパスワードを設定し、パスワードを設定するユーザの ID を記録します。パスワードを設定すると、データベースのマスタ・キーへのすべてのアクセスが可能になります。

マスタ・キーのコピーのパスワードとキー暗号化キーの変更

マスタ・キーのコピーを所有しているユーザは、次の構文を使用してキー・コピーのパスワードを変更できます。

```
alter encryption key [dual] master
    with passwd char_string
    modify encryption
    {with passwd char_string [for recovery]
        | for automatic_startup}
```

それぞれの意味は、次のとおりです。

- *char_string* − (最初のインスタンス) ユーザがキーの所有者である場合は、マスタ・キーまたはデュアル・マスタ・キーのベース・コピーを現在暗号化しているパスワードです。ユーザがキーの所有者でない場合は、キーのユーザのコピーを現在暗号化しているパスワードです。
- *char_string* − (2 番目の参照) 通常コピーまたはリカバリ・コピーの新しいパスワードを指定します。このパラメータを使用して、automatic_startup コピーのパスワードを入力しないでください。
- for automatic_startup − 新しい KEK を生成して使用し、新しい automatic_startup キー・コピーを作成します。

for recovery も for automatic startup も指定されていない場合は、コマンドはキーの所有者によって実行され、Adaptive Server でベース・キーのコピーのパスワードが変更されます。コマンドがキーの所有者によって実行されない場合は、現在のユーザが sso_role または keycustodian_role を持っている場合にのみ、Adaptive Server でベース・キーのコピーのパスワードが変更されます。

- 例 1 − マスタ・キーの所有者 “Jones” が、次の構文を使用して “Mary” のキー・コピーを作成します。

```
alter encryption key master
     with passwd 'unforgettablethatswhatyouare'
     add encryption
     with passwd 'just4now'
     for user Mary
```

- 例 2 − “Mary” が、次の構文を使用して自分のコピーのパスワードを変更します。

```
alter encryption key master
     with passwd 'just4now'
     modify encryption
     with passwd 'maryspasswd'
```

- 例 3 − マスタ・キーの所有者 “John” が、次の構文を使用してベース・キーのパスワードを変更します。

```
alter encryption key master
     with passwd 'unforgettablethatswhatyouare'
     modify encryption
     with passwd 'notunforgettable'
```

sso_role または keycustodian_role を持つユーザは、automatic_startup キー・コピーを変更してキー暗号化キーを変更できます。たとえば、マスタ・キーのパスワードを知っているユーザは、次の構文を使用して automatic_startup キー・コピーのキー暗号化キーを変更できます。

```
alter encryption key master
     with passwd 'unforgettablethatswhatyouare'
     modify encryption for automatic_startup
```

Adaptive Server では、次の操作が行われます。

- パスワードから導出されたキー暗号化キーを使用してベース・マスタ・キーを暗号化する。
- 新しいマスタ・キーの暗号化キーを作成し、この新しいキーを使用してマスタ・キーのスタートアップ・ファイルの古いキーを置き換える。
- 新しいマスタ・キーの暗号化キーを使用してマスタ・キーを暗号化し、新しいコピーを使用して sysencryptkeys の古い automatic_startup キー・コピーを置き換えることで、新しい automatic_startup キー・コピーを作成します。

マスタ・キーの生成

マスタ・キーとデュアル・マスタ・キーは定期的に変更してください。ただし、マスタ・キーとデュアル・マスタ・キーを変更するたびに、新しいマスタ・キーとデュアル・マスタ・キーを使用してすべてのカラム暗号化キーを再暗号化する必要もあります。このプロセスを自動化するために、Adaptive Server ではマスタ・キーとデュアル・マスタ・キーの値を新しい値に置き換える regenerate key オプションを使用し、再生成されているマスタ・キーまたはデュアル・マスタ・キーによって現在暗号化されているすべてのカラム暗号化キーを再暗号化します。

```
alter encryption key [dual] master
     with passwd char_string
     regenerate key
     [with passwd char_string]
```

regenerate key コマンドを実行すると、Adaptive Server では次の操作が行われます。

- 指定したパスワードがベース・マスタ・キーまたはデュアル・マスタ・キーを復号化していることを検証する。
- 新しいマスタ・キーまたはデュアル・マスタ・キーを作成する。
- マスタ・キーまたはデュアル・マスタ・キーによって単独で、または部分的に暗号化されるすべてのカラム暗号化キーを復号化する。Adaptive Server は、新しいマスタ・キーまたはデュアル・マスタ・キーを使用してカラム暗号化キーを再暗号化します。

- ベース・マスタ・キーまたはデュアル・マスタ・キーを 2 行目のパスワードで暗号化された新しいキーに置き換える。2 行目のパスワードが指定されていない場合、Adaptive Server は現在設定されているパスワードを使用して新しいキーを暗号化します。
- 通常のキー・コピーを削除する。マスタ・キーの所有者は alter encryption key を使用して、指定されたユーザの通常のキー・コピーを再作成する必要があります。
- キー・リカバリ・コピーを削除する。マスタ・キーの所有者は、alter encryption key を使用して新しいリカバリ・キー・コピーを追加し、リカバリ・キーの所有者に新しいパスワードを知らせる必要があります。
- automatic_startup コピーをランダムに生成された新しいマスタ・キーの暗号化キーで暗号化して作成された新しいキー・コピーに置き換える。Adaptive Server は、新しいマスタ・キーの暗号化キーをマスタ・キーのスタートアップ・ファイルに書き込みます。

マスタ・キーとキー・コピーの削除

sso_role または keycustodian_role を持つユーザは、マスタ・キーまたはデュアル・マスタ・キーを使用して現在暗号化されているその他のカラム暗号化キーがない場合、マスタ・キーまたはデュアル・マスタ・キーを削除できます。次の操作を行います。

```
drop encryption key [dual] master
```

マスタ・キーまたはデュアル・マスタ・キーを削除すると、Adaptive Server は次の操作を行います。

- マスタ・キーまたはデュアル・マスタ・キーとそのキー・コピーを削除する。すべての通常のキー・コピー、automatic_startup キー・コピー、およびリカバリ・キー・コピーがデータベースから削除されます。
- automatic_startup キー・コピーが存在している場合は、マスタ・キーの暗号化キーをマスタ・キーのスタートアップ・ファイルから削除します。

通常のキー・コピーのみを削除するには、次の構文を使用します。

```
alter encryption key [dual] master
    drop encryption for user username
```

リカバリ・キー・コピーのみを削除するには、次の構文を使用します。

```
alter encryption key [dual] master
    drop encryption for recovery
```

automatic_startup キー・コピーのみを削除するには、次の構文を使用します。

```
alter encryption key [dual] master
    drop encryption for automatic_startup
```

マスタ・キーとデュアル・マスタ・キーのリカバリ

sso_role または keycustodian_role を持つユーザは、次の構文を使用してマスタ・キーとデュアル・マスタ・キーをリカバリできます。

```
alter encryption key [dual] master
    with passwd char_string
    recover encryption
    with passwd char_string
```

ここで、passwd の最初の参照はリカバリ・キー・コピーのパスワードで、passwd の 2 番目の参照はベース・キーの新しいパスワードです。

無人起動モードでの Adaptive Server の起動

パスワードの所有者がいない場合、無人起動モードを使用してマスタ・キーにアクセスできるようにします。

一般に、無人起動モードを使用するには、次の手順に従う必要があります。

1 automatic master key access 設定パラメータを有効にします。

2 (オプション) マスタ・キーのスタートアップ・ファイルのパスと名前を設定します。設定しない場合は、Adaptive Server はデフォルトのファイルのパスと名前を使用します。

3 無人起動するデータベースのマスタ・キーまたはデュアル・マスタ・キーの automatic_startup コピーを追加します。

無人起動モードの設定

sso_role を持つユーザは、次の構文を設定して Adaptive Server が無人起動モードを使用するように設定できます。

```
sp_configure 'automatic master key access', 1
```

このモードでは、パスワードがサーバのメモリに存在しない場合でも、Adaptive Server はマスタ・キーのスタートアップ・ファイルからマスタ・キーの暗号化キーを読み込んで復号化することでマスタ・キーにアクセスします。無人起動モードを使用するには、データベースに master key と dual master key の automatic_startup キー・コピーも作成する必要があります。

マスタ・キーのスタートアップ・ファイル

automatic master key access を有効にすると、Adaptive Server はマスタ・キーのスタートアップ・ファイル (存在する場合) からキー暗号化キーを読み込みます。存在しない場合、Adaptive Server はマスタ・キーのスタートアップ・ファイルを作成しますが、マスタ・キーまたはデュアル・マスタ・キーの automatic_startup キー・コピーが作成されるまで、キー暗号化キーの値をファイルに書き込みません。

automatic master key access を無効にすると、Adaptive Server はマスタ・キーとデュアル・マスタ・キーのキー暗号化キーをサーバのメモリから削除します。Adaptive Server は、キー暗号化キーの値をマスタ・キーのスタートアップ・ファイルから消去しません。

マスタ・キーのスタートアップ・ファイルの作成

sso_role を持つユーザは、次の構文を使用してマスタ・キーのスタートアップ・ファイルのパスと名前を指定できます。

```
sp_encryption mkey_startup_file
        [, {new_path | default_location | null}]
        [, {sync_with_mem | sync_with_qrm}]
```

各パラメータの意味は、次のとおりです。

- *new_path* − マスタ・キーのスタートアップ・ファイルの場所と名前を指定します。new_path は、スタンドアロンの Adaptive Server Cluster Edition インストールではサポートされていません。
- default location − マスタ・キーのスタートアップ・ファイルをデフォルトのパスと名前 *$SYBASE_ASE/security/ase_encrcols_mk_<servername>.dat* に設定します。default location は、スタンドアロンの Adaptive Server Cluster Edition インストールではサポートされていません。
- null − 現在のマスタ・キーのスタートアップ・ファイルのパスと名前を表示します。
- sync_with_mem − 設定オプションの自動マスタ・キー・アクセスが有効である場合、サーバのメモリに存在しているマスタ・キーの暗号化キーをマスタ・キーのスタートアップ・ファイルに書き込みます。sync_with_mem は、スタンドアロンの Adaptive Server Cluster Edition インストールではサポートされていません。
- sync_with_qrm − (スタンドアロンの Cluster Edition インストールでのみ使用可能) ローカル・マスタ・キーのスタートアップ・ファイルにあるキー・コピーをクォーラム・デバイスのコピーで更新します。

Adaptive Server によるマスタ・キーのスタートアップ・ファイルの使用方法

Adaptive Server は、次の場合に、マスタ・キーとデュアル・マスタ・キーの暗号化キーをマスタ・キーのスタートアップ・ファイルからサーバのメモリに読み込みます。

- サーバを automatic master key access を有効にして起動した場合、または
- サーバの稼働中に automatic master key access が有効になっている場合。

指定する値は次のとおりです。

- マスタ・キーまたはデュアル・マスタ・キーの automatic_startup キー・コピーが作成され、Adaptive Server はマスタ・キーまたはデュアル・マスタ・キーの暗号化キーをファイルに書き込みます。
- マスタ・キーまたはデュアル・マスタ・キーの automatic_startup キー・コピーのキー暗号化キーが変更され、Adaptive Server は新しいマスタ・キーまたはデュアル・マスタ・キーの暗号化キーをファイルに書き込みます。
- automatic_startup キー・コピーが削除され、Adaptive Server はファイルの対応するレコードを削除します。
- データベースが削除され、Adaptive Server は削除されたデータベースに属するすべてのレコードを削除します。
- マスタ・キーまたはデュアル・マスタ・キーが削除され、Adaptive Server は対応するレコードを削除します。
- 新しいマスタ・キーのスタートアップ・ファイルが sp_encryption mkey_startup_file を使用して指定され、Adaptive Server はサーバのメモリを新しいファイルの内容と同期します。

マスタ・キーの暗号化キーがメモリにあると、マスタ・キーのパスワードが設定されていない場合でも automatic_startup コピーを使用してマスタ・キーにアクセスすることができます。

第 4 章

外部パスワードと隠しテキストのセキュリティ保護

Adaptive Server では、AES-256 対称暗号化アルゴリズムを使用して外部ログイン・パスワードと隠しテキストの暗号化を強化しています。以下に対して、外部パスワードの強力な暗号化を選択できます。

- Replication Agent － 複写データベース。
- CIS － リモート記述子とログイン。
- Job Scheduler － Job Scheduler Agent。
- RTMS － リアルタイム・メッセージング。
- SSL (Secure Sockets Layer) と LDAP (Lightweight Directory Access Protocol) － SSL と LDAP アクセス・アカウント。ストアド・プロシージャ sp_ldapadmin および sp_ssladmin を使用して管理するパスワードは保護することができます。

ストアド・プロシージャ、ユーザ定義関数、計算カラムなどの syscomments に保存されている SQL テキストを持つオブジェクトは、オプションで sp_hidetext を使用する強力な暗号化を使用して暗号化できます。

注意 外部パスワードと隠しテキストの暗号化には ASE_ENCRYPTION ライセンスが必要です。

サービス・キー

サービス・キーは 256 ビットの永続的な暗号化キーで、外部ログイン・パスワードと隠しテキストを強力に暗号化するのに使用され、sysencryptkeys に保存されます。

次のいずれかを使用してサービス・キーを暗号化します。

- 静的なキー – サービス・キーのデフォルトのキー暗号化キーです。マスタ・キーが現在のデータベースに作成されていない場合に使用できます。この方法では、Adaptive Server は無人起動した後、サービス・キーを使用できます。
- マスタ・キー – 静的なキーよりも強力な保護を提供します。Adaptive Server は、パスワードを使用してデータベース固有のマスタ・キーを復号化する必要があります。

これらのサービス・キーを記述するデータベース・オブジェクトには、次のようなものがあります。

- syb_extpasswdkey – sysattributes の外部ログイン・パスワードの暗号化するためのサービス・キーを識別します。データベースに対して存在する syb_extpasswdkey は 1 つだけです。syb_extpasswdkey を変更すると、そのキーを使用して暗号化されたすべてのデータが新しいキーを使用して再暗号化されます。

  外部ログイン・パスワードは通常、マスタ・データベースに保存されますが、RepAgent ではこの情報は複写データベースに保存されます。

- syb_syscommkey_dddddd – syscomments の隠しテキストを暗号化するためのサービス・キーを識別します。ここで、“dddddd” は、Adaptive Server で生成されたグローバル識別子で、キーをユニークに識別します。グローバル識別子には、同じオブジェクトに関連付けられている syb_syscommkey キーが多数ある場合に名前を区別する名前が含まれています。グローバル識別子は、ローカル・データベースと複写データベースの両方でキーを区別します。

  隠しテキストの強力な暗号には、各データベースにサービス・キーが必要です。ここで、sp_hidetext を実行して SQL テキストを隠します。新しいサービス・キーを作成すると、データベースにある既存のサービス・キーは明示的に削除されるまで継続し、隠しテキストは sp_hidetext を再発行するまで再暗号化されません。

注意 システム暗号化パスワードではサービス・キーを暗号化しません。

サービス・キーの作成

sso_role または keycustodian_role を持つユーザは、次の構文を使用してサービス・キーを作成できます。

```
create encryption key [syb_extpasswdkey | syb_syscommkey]
    [ with { static key | master key }]
```

デフォルトでは、静的なキーはキーを暗号化します。マスタ・キーを使用するには、with master key パラメータを使用します。

サービス・キーを作成するユーザがそのキーの所有者になります。

syb_extpasswdkey を作成すると、sysattributes にあるすべての外部パスワードがすべて強力な暗号化を使用して新しいキーで再暗号化されます。

syb_syscommkey を作成すると、それ以降の sp_hidetext 実行では、強力な暗号化を使用した新しいキーが使用されます。sp_hidetext は新しいキーを使用して暗号化されるオブジェクトに対して既存のデータベース・オブジェクト上で実行する必要があります。隠しテキストの再暗号化では、非常に大量のデータが影響を受ける可能性があるため、データベース管理者はシステム要求が少ない時間まで sp_hidetext の実行を遅延させる必要があります。

サービス・キーを作成するには、次の条件を満たす必要があります。

- ASE_ENCRYPTION ライセンスが必要である。
- enable encrypted columns 設定パラメータを設定する必要がある。
- サービス・キーを作成するユーザには sso_role または keycustodian_role が必要である。
- マスタ・キーを使用してサービス・キーを保護している場合は、サービス・キーの前にマスタ・キーを作成する必要がある。

注意　サービス・キーでデュアル・コントロールを使用することはできません。

サービス・キーの削除

drop encryption key は、暗号化キーへの参照が残っていないことを確認してから削除します。存在していない syb_extpasswdkey または syb_syscommkey_dddddd を削除することはできません。すべての隠しテキスト・キーを確実に削除するには、sp_encryption を使用してすべての既存のキーを識別します。

注意　ASE_ENCRYPTION ライセンスの有効期限が切れると、暗号化データは使用できなくなるため、drop encryption key コマンドを実行することはできません。一時的なライセンスを取得するには、Sybase サポート・センタに問い合わせてください。

sso_role または keycustodian_role を持つユーザは、次の構文を使用して外部ログインの未使用のサービス・キーを削除できます。

```
drop encryption key syb_extpasswdkey
    with password encryption downgrade
```

with password encryption downgrade を指定すると、15.7 より前のバージョンで使用されているアルゴリズムで外部ログイン・パスワードが再設定されます。Replication Agent のパスワード、および CIS と RTMS の外部ログイン・パスワードは無効な値に再設定されます。キーが削除された後で、管理者は手動でパスワードを再入力し、対応するサービスを再開する必要があります。

sso_role または keycustodian_role を持つユーザは、次の構文を使用して隠しテキストの未使用のサービス・キーを削除できます。

- 単一のキーを削除していることを示す構文は、次のとおりです。

  ```
  drop encryption key syb_syscommkey_dddddd
  ```

  Adaptive Server は、指定されたキー _dddddd への参照があるかどうかを確認し、参照が見つからない場合はキーを削除します。

  syb_syscommkey_dddddd は単一のキーを示すため、with text encryption downgrade パラメータを使用して syb_syscommkey_dddddd を指定することはできません。

- 複数のキーを削除していることを示す構文は、次のとおりです。

  ```
  drop encryption key syb_syscommkey with text encryption downgrade
  ```

  - with text encryption downgrade を指定する場合は、syb_syscommkey_dddddd を使用して単一のサービス・キーを指定することはできません。syb_syscommkey のみを使用します。
  - syb_syscommkey に “dddddd” サフィックスが付いていない場合、Adaptive Server は 15.7 よりも以前のバージョンで使用されたアルゴリズムを使用して syscomments のすべての隠しテキストを再暗号化し、すべての syb_syscommkey_dddddd キーを削除します。

サービス・キーの変更

syb_extpasswdkey を再生成したり、その保護の暗号化をマスタ・キーから静的なキーまたはその逆に変更したりすることができます。syb_syscommkey を再生成することはできません。

syb_extpasswdkey の変更

次の構文を使用して syb_extpasswdkey を変更します。

```
alter encryption key syb_extpasswdkey
    [ with { static key | master key} ]
        { regenerate key [ with { static key | master key } ]
        | modify encryption [ with { static key | master key } ] }
```

それぞれの意味は、次のとおりです。

- with {static key | master key} の最初のインスタンスはオプションで、syb_extpasswdkey が現在暗号化されている方法を示します。
- with {static key | master key} の 2 番目のインスタンスを使用すると、管理者は再生成されたキーの暗号化を静的から動的またはその逆に変更できます。このパラメータを省略すると、再生成されたキーはこのコマンドの発行前と同様に暗号化されたままになります。
- with {static key | master key} の 3 番目のインスタンスは、指定に応じて静的なキーまたはマスタ・キーを使用するように既存のキーの保護を変更します。このパラメータを省略すると、デフォルトで静的なキーが使用されます。

syb_extpasswdkey の再生成は、次の操作を行う 1 つのトランザクションです。

1　外部ログイン・パスワードの新しいサービス・キーを作成する。

2　新しいキーを使用して sysattributes でパスワードを再暗号化する。

3　古いキーを削除する。

次に例を示します。

- 外部ログイン・パスワードのサービス・キーを作成し、静的なキーで保護されたサービス・キーを使用してすべての外部ログイン・パスワードを暗号化します。

```
create encryption key syb_extpasswdkey
```

- 外部ログイン・パスワードのサービス・キーを再生成し、静的なキーで保護された新しいサービス・キーはそのままにして、古いサービス・キーで暗号化されたすべての外部パスワードを再暗号化します。

```
alter encryption key syb_extpasswdkey
       regenerate key
```

- サービス・キーの保護がマスタ・キーで暗号化されるように変更します。サービス・キーは変更されず、外部ログイン・パスワードは再暗号化されません。

```
alter encryption key syb_extpasswdkey
      modify encryption with master key
```

注意　このコマンドを発行する前に、マスタ・キーのパスワードがマスタ・キーの所有者によって既に入力されていることを確認してください。

syb_syscommkey の変更

syb_syscommkey を変更するには、新しいキーを作成して sp_hidetext を使用し、新しいキーで再暗号化します。

次に例を示します。

- 例 1 － 新しい隠しテキストの暗号化キーを作成し、新しく作成したキーを使用して syscomments テーブルのすべての SQL テキスト・オブジェクトを暗号化します。

```
create encryption key syb_syscommkey
go
sp_hidetext
go
```

注意 新しい syb_syscommkey を作成すると、そのデータベースの sp_hidetext によって使用されるデフォルト・キーになります。

- 例 2 － 新しい隠しテキストの暗号化キーを作成し、新しく作成したキーを使用して syscomments の特定のストアド・プロシージャのテキストを暗号化し、マスタ・キーを使用してキーを保護します。

```
create encryption key syb_syscommkey
       with master_key
go
sp_hidetext sp_mysproc
go
```

この例では、syscomments のその他すべての隠しテキスト・ローは前の暗号化キーを使用して暗号化されたままになります。

外部パスワードを使用したサービス・キーの使用

サービス・キーは、SSL を使用してネットワーク・リスナのプライベート・キーのパスワードを復号化します。プライベート・キーのパスワードは SSL 証明書を初期化します。

SSL パスワード

サービス・キーがマスタ・キーで暗号化されている場合、マスタ・キーは使用できません。

- interfaces ファイルに SSL リスナしか指定されていない場合、ログインしてマスタ・キーまたはデュアル・マスタ・キーのパスワードを入力できるユーザはいません。Adaptive Server は、リスナを起動できないので停止します。

- SSL リスナと非 SSL リスナの両方を interfaces ファイルで指定すると、非 SSL リスナはログイン要求を受け入れることができます。SSL リスナは、次の構文を使用して 非 SSL リスナ・ポートの Adaptive Server に接続した後、マスタ・キーのパスワードが認証されたユーザによって手動で入力されるまでブロックされます。

  ```
  use master
  go
  set encryption passwd password for key master
  go
  ```

  マスタ・キーのパスワードを正しく入力すると、Adaptive Server は SSL リスナのプロセスをウェイクアップさせ、それらが受信ログイン要求の受け入れを開始します。

LDAP パスワード

サービス・キーは、Adaptive Server が LDAP ユーザ認証プロセス時にユーザを認証する際、LDAP 管理者アカウントのパスワードを復号化するために必要です。認証が完了するまで、ユーザは LDAP を使用してログインできません。Adaptive Server 認証を使用してローカルで認証できる権限を持つユーザは、次の構文を使用してマスタ・キーのパスワードを手動で入力できます。

```
use master
go
set encryption passwd password for key master
go
```

LDAP ユーザ認証の設定の詳細については、『セキュリティ管理ガイド』の外部認証に関する章を参照してください。

Replication Agent のパスワード

サービス・キーは、ユーザ・データベースの Replication Agent で接続を開始するパスワードを復号化します。自動的に開始するよう設定されているエージェントは、サービス・キーがマスタ・キーで暗号化されている場合、権限を持つユーザがマスタ・キーのパスワードを手動で入力するまでブロックされます。

サービス・キーが複写されたユーザ・データベースにある場合、暗号化キーを保存している sysencryptkeys テーブルも複写されているため、複写データベースでもサービス・キーを使用できます。マスタ・キーは、複写された sysencryptkeys テーブルにも保存されており、複写データベースでも使用できます。サービス・キーは暗号化されているため、複写プロセス時は保護されたままになります。

Adaptive Server が起動されると、権限を持つユーザは次の構文を使用してデータベースごとに接続し、マスタ・キーのパスワードを設定できます。

```
use mydb
go
set encryption passwd password for key master
go
```

マスタ・キーのパスワードを待機している Replication Agent は、ステータス値 “passwd sleep” で識別できます。

```
sp_who
go

fid spid status loginame orignname hostname blk_spid dbname
tempdbname cmd block_xloid
--- ---- ----------- ------ -------- -------- ------
------ ---------- --------- -----------
0   38  passwd sleep NULL   NULL      NULL     0
tdb4   tempdb     REP AGENT  0
```

アップグレード時の隠しテキストを使用したサービス・キーの使用

15.7 へのアップグレード時に、プロシージャ・オブジェクトはソースから再コンパイルされます。接続されているユーザは、隠しテキストの強力な暗号化が有効になっており、サービス・キーがマスタ・キーによって保護されているデータベースのマスタ・キーのパスワードが入力されるまで、実行できる操作が制限されています。権限を持つユーザは、次の構文を使用してそのようなデータベースにマスタ・キーのパスワードを設定する必要があります。

```
use mydb
go
set encryption passwd password for key master
go
```

マスタ・キーで暗号化されたサービス・キーの考慮事項

サービス・キーがマスタ・キーで暗号化されている場合、マスタ・キーの指定方法に応じて、マスタ・キーのパスワードを自動または手動で Adaptive Server に入力する必要があります。

自動マスタ・キー・アクセスを使用しない場合は通常、set encryption passwd を使用してマスタ・キーのパスワードを入力します。ただし、起動時にネットワーク・リスナのプライベート・キーのパスワードを復号化するためにサービス・キーが必要である場合は、コマンド・ラインで、またはコマンド・ラインのプロンプトを使用してマスタ・キーを指定することができます。

dataserver . . . -- master_key_password パラメータを使用して、Adaptive Server の起動時にマスタ・キーのパスワードの入力を要求するプロンプトを表示します。-- master_key_passwd パラメータを発行するユーザは、マスタ・データベースのマスタ・キーのパスワードを認識し、コンソールとキーボードに物理的にアクセスしてパスワードを入力する必要があります。

パスワードを指定しないと、-- master_key_password はコマンド・ラインでパスワードの入力を要求するプロンプトを表示します。次に例を示します。

```
dataserver --master_key_passwd  -dd_master  -eerrorlog

master_key_passwd:_
```

パスワード文字は表示されず、パスワードは Adaptive Server の起動順序の後半まで検証されません。

--master_key_passwd パラメータにパスワードを指定すると、次のようになります。

```
dataserver --master_key_passwd=mysecret  -dd_master  -eerrorlog
```

パスワード mysecret は、読み込まれて使用された後、メモリ内で無効になります。ただし、クリア・パスワードはメモリが無効になるまで表示されます。

間違ったパスワードを入力すると、サービス・キーの使用は失敗し、サービス・キーを必要とする Adaptive Server サービスは使用できないままになります。サービスが起動されると、権限を持つユーザは次の構文を使用してマスタ・データベースに接続し、マスタ・キーのパスワードを設定できます。

```
use master
go
set encryption passwd password for key master
go
```

SSL リスナのみを設定している場合に間違ったパスワードを入力すると、Adaptive Server はリスナを起動できないため停止します。

パスワードは、次の場所に表示されるため、コマンド・ラインでは使用しないことをおすすめします。

- UNIX ps コマンドを使用して表示できるメモリ内
- メモリ内、無人端末画面上、コマンド履歴バッファおよびファイルのディスク上
- 画面上

無人起動の使用時のこれらの脆弱性を回避するために、カスタマ・サイトはパスワードの入力を要求するプロンプトを表示することをおすすめします。

第 5 章

データの暗号化

次のデータ型を暗号化できます。

- int、smallint、tinyint
- unsigned int、unsigned smallint、unsigned tinyint
- bigint、unsigned bigint
- decimal、numeric
- float4、float8
- money、smallmoney
- date、time、smalldatetime、datetime
- char、varchar
- unichar、univarchar
- binary、varbinary
- bit

新しいテーブルに対する暗号化の指定

新しいテーブルのカラムを暗号化するには、create table 文で encrypt column 修飾子を使用します。

次に示す create table の部分構文には、暗号化に固有の句のみ含まれます。『リファレンス・マニュアル：コマンド』を参照してください。

```
create table table_name
(column_name
. . .

[constraint_specification]
[encrypt [with [database.[owner].]keyname]]
[, next_column_specification . . .]
)
```

keyname － create encryption key を使用して作成したキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しない場合、create encryption key に as default 句を指定して作成したデフォルト・キーが検索されます。

注意 計算カラムは暗号化できません。また、計算カラムを定義している式に暗号化カラムを含めることはできません。テーブルの *partition_clause* には、暗号化カラムを指定できません。

次の例では、2 つのキーが作成されます。1 つはデータベース・デフォルト・キーで、もう 1 つは、create table コマンドで名前を指定する必要のあるキー (cc_key) です。どちらのキーでも、長さおよび初期化ベクトルにデフォルト値を使用します。employee テーブルの ssn カラムはデフォルト・キーを使用して暗号化され、customer テーブルの creditcard カラムは cc_key を使用して暗号化されます。

```
create encryption key new_key as default for AES
create encryption key cc_key

create table employee_table (ssn char(15) encrypt,
   ename char(50), ...))

create table customer (creditcard char(20)
   encrypt with cc_key, cc_name char(50), ...)
```

次の例では、初期化ベクトルとランダム埋め込みにデフォルト以外の値を使用する、名前が k1 のキーが作成されます。employee esalary カラムにランダムなデータを埋め込んでから暗号化します。

```
create encryption key k1 init_vector null pad random
create table employee (eid int, esalary money encrypt with k1, ...)
```

select into での暗号化の指定

デフォルトでは、ソース・テーブルに 1 つ以上の暗号化カラムがある場合でも、暗号化されていないターゲット・テーブルが select into によって作成されます。ターゲット・テーブルで任意のカラムを暗号化するには、次のようにターゲット・テーブルを encrypt 句で修飾する必要があります。

```
select [all|distinct] column_list
      into table_name
      [(colname encrypt [with [[database.][owner].]keyname]
                  [, colname encrypt
                  [with[[database.][owner].]keyname]])]
            from table_name | view_name
```

ソース・テーブルでデータが暗号化されていなくても、ターゲット・テーブルの特定のカラムを暗号化できます。ソース・テーブルのカラムが、ターゲット・カラムに指定するものと同じキーによって暗号化されている場合、Adaptive Server によってソース・テーブルの復号化手順とターゲット・テーブルの暗号化手順が省略され、処理が最適化されます。

ターゲット・テーブルの暗号化を指定するためのルールは、次の点に関して、create table で指定された暗号化に対するルールと同じです。

- 暗号化するカラムに使用できるデータ型
- *keyname* が省略された場合のデータベース・デフォルト・キーの使用
- ターゲット・カラムの暗号化に使用されるキーの select パーミッションの必要性

次の例では、daily_xacts テーブルから暗号化カラム creditcard を選択し、#bigspenders テンポラリ・テーブル内の暗号化されたフォームにそのカラムを保存します。

```
select creditcard, custid, sum(amount) into      #bigspenders
     (creditcard encrypt with cust.dbo.new_cc_key)
     from daily_xacts group by creditcard
     having sum(amount) > $5000
```

注意　select into を実行するには、decrypt を含む、ソース・テーブルに対するカラム・レベルのパーミッションが必要です。

既存テーブルのデータの暗号化

既存のテーブルのカラムを暗号化するには、alter table 文の encrypt 句で modify column オプションを使用します。

```
alter table table_name modify column_name
    [encrypt [with [[database.][owner].]keyname]]
```

ここで、*keyname* は create encryption key を使用して作成するキーを指定します。テーブルの作成者は、*keyname* の select パーミッションが必要です。*keyname* を指定しないと、Adaptive Server は、create encryption key の as default 句を使用して作成されたデフォルト・キーを探します。

『リファレンス・マニュアル：コマンド』を参照してください。

暗号化カラムを変更する場合、次のような制限があります。

- トリガが作成されているカラムは、暗号化または復号化の対象として変更できません。次のことを行ってください。
  a トリガを削除します。
  b カラムを暗号化または復号化します。
  c トリガを再作成します。
- 既存の暗号化カラムを変更したり、テーブルの暗号化または復号化の対象としてカラムを変更したり、暗号化カラムがクラスタード・インデックスまたは配置インデックス内のキーである場合にそのカラムのデータ型を変更できません。次のことを行ってください。
  a インデックスを削除します。
  b テーブルを変更するか、カラムのデータ型を変更します。
  c インデックスを再作成します。

他の属性を変更するのと同時にカラムの暗号化プロパティを変更できます。また、alter table を使用して、暗号化カラムの追加もできます。

次に例を示します。

```
alter table customer modify custid null encrypt with cc_key
alter table customer add address varchar(50) encrypt with cc_key
```

暗号化カラムのインデックスおよび制約の作成

暗号化キーで初期化ベクトルまたはランダム埋め込みが指定されていない場合、暗号化カラムにインデックスを作成できます。初期化ベクトルまたはランダム埋め込みが使用されている暗号化カラムに対して create index を実行すると、エラーが発生します。一般的に、暗号化カラムのインデックスは等号および不等号を使用する一致には役立ちますが、大文字小文字を区別しないデータの一致やデータの範囲検索には役立ちません。

注意 関数インデックスの式には、暗号化カラムを使用できません。

次の例では、cc_key は初期化ベクトルまたは埋め込みを使用せずに暗号化を指定しています。これにより、cc_key を使用して暗号化されたすべてのカラムにインデックスを作成できます。

```
create encryption key cc_key
    with init_vector null

create table customer(custid int,
    creditcard varchar(16) encrypt with cc_key)

create index cust_idx on customer(creditcard)
```

プライマリ・キーまたはユニーク・キーとして宣言されたカラムを暗号化できます。

次の場合、暗号化カラムに参照整合性制約を定義できます。

- 参照先カラムと参照元カラムの両方が同じキーで暗号化されている。
- カラムの暗号化に使用するキーで init_vector null を指定し、pad random を指定していない。

参照整合性チェックは暗号化テキスト値に実行されるため、効率的です。

次の例では、ssn_key によって、プライマリ・テーブルと外部テーブル両方の ssn カラムが暗号化されます。

```
create encryption key ssn_key for AES
    with init_vector null
create table user_info (ssn char(9) primary key encrypt
    with ssn_key, uname char(50), uaddr char(100))
create table tax_detail (ssn char(9) references user_info encrypt
    with ssn_key, return_info text)
```

暗号化カラムのドメイン・ルールとアクセス・ルールの作成

暗号化カラムでドメイン・ルール、検査制約、またはアクセス・ルールを作成できます。ただし、暗号化カラムをターゲット・リスト、where 句などで使用する場合は、暗号化カラムに対する decrypt パーミッションが必要です。次の例では、ドメイン・ルールが定義されている creditcard カラムに rule_creditcard ルールを作成します。

```
create encryption key cc_key
    with init_vector null

create table customer(custid int,
        creditcard varchar(16) encrypt with cc_key)

create rule rule_creditcard
as @value like '%[0-9]'
sp_bindrule rule_creditcard, creditcard
```

Adaptive Server では bcp in -C で高速 bcp を使用するため、bcp in -C では暗号化カラムに対するドメイン・ルールまたは検査制約はバイパスされます。暗号化カラムにアクセス・ルールが存在する場合、bcp out -C でエラー番号 2929 が生成されます。Adaptive Server では、ドメイン・ルールまたは検索制約を含む暗号化カラムを複製する場合、insert 文と update 文のルールまたは制約はバイパスされます。また、update、delete、select 文では、アクセス・ルールを含む暗号化カラムを複製すると、エラー番号 2929 が生成されます。

decrypt パーミッション

暗号化カラムからプレーン・テキスト・データを選択したり、暗号化カラムを検索またはジョインしたりするためには、decrypt パーミッションが必要です。

テーブル所有者または sso_role を持つユーザは、grant decrypt を使用して、テーブル内の 1 つ以上のカラムを復号化する明示的なパーミッションを他のユーザ、グループ、ロールに付与します。プロシージャまたはビューの所有者が次のパーミッションを付与するときに、decrypt パーミッションが暗黙に付与される場合もあります。

- 暗号化カラムを選択するストアド・プロシージャまたはユーザ定義関数の exec パーミッション (プロシージャまたは関数の所有者が暗号化カラムを含むテーブルも所有している場合)
- 暗号化カラムを選択するビュー・カラムに対する decrypt パーミッション (ビューの所有者がテーブルも所有している場合)

どちらの場合も、ベース・テーブルの暗号化カラムに decrypt パーミッションを付与する必要はありません。

構文は次のとおりです。

```
grant decrypt on [owner.] table
    [( column[{,column}])]
    to user| group | role
    [with grant option]
```

テーブル・レベルまたはビュー・レベルで decrypt パーミッションを付与すると、テーブルのすべての暗号化カラムの decrypt パーミッションが付与されます。

customer テーブルのすべての暗号化カラムに対する decrypt パーミッションを付与するには、次のコマンドを入力します。

```
grant decrypt on customer to accounts_role
```

次の例では、ベース・テーブル “employee” の ssn カラムの user2 の暗黙的な decrypt パーミッションを表示します。user1 は employee テーブルと employee_view を次のように設定します。

```
create table employee (ssn varchar(12)encrypt,
    dept_id int, start_date date, salary money)

create view emp_salary as select
    ssn, salary from employee

grant select, decrypt on emp_salary to user2
```

user2 は、次のように emp_salary ビューを選択して、復号化された社会保障番号にアクセスできます。

```
select * from emp_salary
```

注意　テーブルまたはビューに対して grant all を実行しても、decrypt パーミッションは付与されません。decrypt パーミッションは別途付与する必要があります。

---

restricted decrypt permission で Adaptive Server を設定し、暗黙的な decrypt パーミッションがユーザに付与されないようにします。詳細については、「decrypt パーミッションの制限」(46 ページ) を参照してください。

暗号化カラムの select パーミッションだけしか持たないユーザであっても、bulk copy コマンドで暗号化カラムを暗号テキストとして処理できます。また、暗号化カラムで復号化デフォルト値を指定している場合、データを復号化するパーミッションを持っていないユーザは select ターゲット・リストや where 句内でカラムの名前を指定できます。「復号化されたデータの代わりに返されるデフォルト値」(47 ページ) を参照してください。

復号化のパーミッションの取り消し

次の文を使用してユーザの復号化のパーミッションを取り消すことができます。

```
revoke decrypt on [ owner.] table[( column[ {,column}])] from user
    | group | role
```

次に例を示します。

```
revoke decrypt on customer from public
```

decrypt パーミッションの制限

Adaptive Server では、キーを保護するためにマスタ・キーまたはシステム暗号化パスワードを使用する場合でも、管理者の権限からデータ・プライバシーを保護します。パスワード管理を行わず、マスタ・キーまたはシステム暗号化パスワードを使用して暗号化キーを保護する場合は、restricted decrypt permission 設定パラメータを設定することで、データベース所有者からのプライベート・データへのアクセスを制限できます。システム・セキュリティ担当者 (SSO) はこのパラメータを使用して、decrypt パーミッションを持つユーザを制御できます。restricted decrypt permission を有効にすると、SSO だけが、暗黙的な decrypt パーミッションを取得し、このパーミッションを他のユーザに付与する暗黙的な権限を持ちます。SSO は、with grant オプション付きの decrypt パーミッションを付与することで、decrypt パーミッションを取得するユーザを決定したり、このジョブを他のユーザに委任したりします。テーブルの所有者は、所有しているテーブルの decrypt パーミッションを自動的には取得しません。

ストアド・プロシージャまたはユーザ定義関数に対する execute パーミッションを持つユーザは、プロシージャまたは関数によって選択されたデータを復号化するための暗黙的なパーミッションを取得しません。ビュー・カラムに対する decrypt パーミッションを持つユーザは、ビューによって選択されたデータを復号化するための暗黙的なパーミッションを取得しません。

注意 エイリアスを持つユーザは、それらのユーザにエイリアスを指定したユーザのすべての decrypt パーミッションを引き続き継承します。set proxy/set user 文を使用すると、管理者またはデータベース所有者に対して、このコマンドで推測される ID を持つユーザの decrypt パーミッションが引き続き許可されます。

制限のある decrypt パーミッションの権限の割り当て

制限のある decrypt パーミッションを使用している場合、タスクのスキーマを作成し、キーを管理する権限を次のように割り当てることができます。

- システム・セキュリティ担当者 − 制限のある decrypt パーミッションを設定して暗号化キーを作成し、キーに対する select パーミッションをデータベース所有者に付与して、decrypt パーミッションをエンド・ユーザに付与する。
- データベース所有者 − スキーマを作成し、データをロードする。

復号化されたデータの代わりに返されるデフォルト値

この項では、暗号化カラムで復号化デフォルト値を使用する方法を示します。機密データの参照を許可されていないユーザが暗号化カラムに対してクエリを実行すると、復号化されたデータではなく、復号化デフォルト値が表示されます。復号化デフォルト値を使用すると、機密データの参照を許可されていないユーザであってもレガシー・アプリケーションとレポートをエラーなく実行できます。

復号化デフォルト値の定義

create table および alter table の decrypt_default パラメータを使用すると、decrypt パーミッションのないユーザが暗号化カラムの情報を選択しようとしたときに暗号化カラムでユーザ定義の値を返すことができます。この場合、エラー メッセージ 10330 は表示されません。

```
Decrypt permission denied on object <table_name>,
     database <database name>, owner <owner name>
```

暗号化カラムで復号化デフォルト値を使用すると、既存のレポートをエラーなしで実行できるので、ユーザは、暗号化されていない情報を表示できます。たとえば、customer テーブルに暗号化カラム creditcard が含まれている場合、次のような処理を行うテーブル・スキーマを設計できます。

```
select * from customer
```

decrypt パーミッションのないユーザには、クレジット・カード・データではなく、値 "**********" が返されます。

暗号化のデフォルト値の追加または削除

新しいカラムの復号化デフォルト値を create table で指定します。以下は、create table 構文の一部です。

```
create table table_name (column_name datatype
        [[encrypt [with keyname]] [decrypt_default value]], ....)
```

- decrypt_default － このカラムでは decrypt パーミッションのないユーザに select 文のデフォルト値を返すことを指定します。
- *value* － select 文を実行したときに、復号化された値の代わりに Adaptive Server から返される固定値です。定数式はデータベース・カラムを参照できませんが、それ自体がテーブルやカラムを参照するユーザ定義の関数を含めることができます。NULL が許可されているカラムについては、NULL を指定することができます。

たとえば、テーブル t2 の ssnum カラムに対して decrypt パーミッションのないユーザがクエリを実行した場合、"??????????" が返されます。

```
create table t2 (ssnum char(11)
     encrypt decrypt_default '???????????', ...)
```

以前に暗号化されていなかった既存のカラムに暗号化および復号化デフォルト値を追加するには、次の構文を使用します。

```
alter table table_name modify column_name [type]
   [[encrypt [with keyname]] [decrypt_default value]], …
```

次の例では、emp テーブルを変更して、ssn カラムを暗号化し、復号化デフォルト値を指定します。

```
alter table emp modify ssn encrypt
     with key1 decrypt_default '000-00-0000'
```

既存の暗号化カラムに復号化デフォルト値を追加する場合、またはすでに復号化デフォルト値があるカラムの復号化デフォルト値を変更する場合は、次の構文を使用します。

```
alter table table_name replace column_name decrypt_default value
```

次の例では、すでに暗号化されている salary カラムに復号化デフォルト値を追加します。

```
alter table employee replace salary
     decrypt_default $0.00
```

次の例では、前の decrypt_default 値を新しい値に置き換え、ユーザ定義関数 (UDF) を使用してデフォルト値を生成します。

```
alter table employee replace salary
     decrypt_default dbo.mask_salary()
```

暗号化プロパティを削除せずに暗号化カラムから復号化デフォルト値を削除する場合は、次の構文を使用します。

```
alter table table_name replace column_name drop decrypt_default
```

次の例では、暗号化プロパティを削除せずに salary の復号化デフォルト値を削除します。

```
alter table employee replace salary
     drop decrypt_default
```

パーミッションおよび復号化デフォルト値

暗号化カラムでユーザまたは役割が暗号化データを選択または検索できるようにするには、その暗号化カラムの decrypt パーミッションを付与する必要があります。暗号化カラムに復号化のデフォルト属性がある場合、decrypt パーミッションのないユーザは、そのカラムを選択または検索するクエリを実行できますが、プレーン・テキスト・データは表示されず、検索に使用されません。

次の例では、テーブル emp の所有者が、hr_role を持つユーザに emp.ssn の表示を許可します。ssn カラムには復号化デフォルト値があるので、emp の select パーミッションだけを持っていて hr_role を持たないユーザには実際の復号化データは表示されず、*decrypt_default* の値が表示されます。

```
create table emp (name char(50), ssn (char(11) encrypt
     decrypt_default '000-00-000', ...)
grant select permission on table emp to public
grant decrypt on emp(ssn) to hr_role
```

hr_role パーミッションがあり、このテーブルに対して select 文を実行した場合は、ssn の値が表示されます。

```
select name, ssn from emp
name                           ssn
-----------------------------  ------------
Joe Cool                       123-45-6789
Tinna Salt                     321-54-9879
```

hr_role もテーブルの select パーミッションがなく、このテーブルに対して select 文を実行した場合は、復号化デフォルト値が表示されます。

```
select name, ssn from emp
name                              ssn
-----------------------------  -----------
Joe Cool                       000-00-0000
Tinna Salt                     000-00-0000
```

このテーブルの hr_role がない場合、order by 句は結果セットに影響しません。

復号化デフォルト値を含むカラム

クエリで復号化デフォルト属性を含むカラムの使用方法に制限はなく、ターゲット・リストの式、where 句、order by、group by、またはサブクエリで使用できます。復号化デフォルト固定値の式には特定の使用方法はありませんが、カラムに復号化デフォルト値を配置しても、Transact-SQL™ 文でのカラムの使用は構文で制限されません。

次の例では、ターゲット・リスト内の復号化デフォルト値を含むカラムで select 文を使用します。

```
create table emp_benefits (col1 name char(30),
     salary float encrypt decrypt_default -99.99)

select salary/12 as monthly_salary from emp_benefits
     where name = 'Bill Smith'
```

このテーブルに対して select 文を実行すると、decrypt パーミッションがないユーザには次の結果が表示されます。

```
monthly_salary
----------------------
8.332500
```

select into コマンドでカラムの復号化デフォルト値が返されるとき、この復号化デフォルト値がターゲット・テーブルに挿入されます。しかし、ターゲット・カラムは、復号化デフォルト・プロパティを継承しません。alter table を使用して、ターゲット・テーブルの復号化デフォルトを指定する必要があります。

復号化デフォルト・カラムおよびクエリ条件

decrypt パーミッションのないユーザが復号化デフォルト・プロパティを含むカラムを where 句で使用すると、条件は false に評価されます。以下の例では、上記の emp テーブルを使用します。hr_role を持つユーザだけが ssn の decrypt パーミッションを持っています。

1 hr_role を持っているユーザが次のクエリを発行すると、1 つのローが返されます。

```
select name from emp where ssn = '123-456-7890'

name
------------------------------
Joe Cool
```

2 hr_role がない場合は、ローは返されません。

```
select name from emp where ssn = '123-456-7890'

name
------------------------------
(0 rows affected)
```

3　hr_role を持っているユーザが非暗号化カラムで or 文を含めると、適切なローが返されます。

```
select name from emp where ssn = '123-456-7890' or
name like 'Tinna%'

name
------------------------------
Joe Cool
Tinna Salt
```

4　hr_role がない場合に同じコマンドを発行すると、1 つのローだけが返されます。

```
select name from emp where ssn = '123-456-7890' or name
like 'Tinna%'

name
------------------------------
Tinna Salt
```

この場合、復号化デフォルト・プロパティを含む暗号化カラムに対する条件は false に評価されますが、非暗号化カラムに対する条件は成功します。

暗号化カラムの decrypt パーミッションがないユーザが、復号化デフォルト値を含むこのカラムに group by 文を発行すると、復号デフォルト固定値によるグループ化が行われます。

decrypt default および暗黙的な付与

テーブルの明示的または暗黙的なパーミッションがない場合、decrypt default 値が返されます。

次の例 (上記の emp テーブルを使用します) では、データベース所有者が p_emp プロシージャを作成します。このプロシージャは、データベース所有者が所有する emp テーブルから選択します。

```
create procedure p_emp as
     select name, ssn from emp
grant exec on p_emp to corp_role
```

corp_role を持つユーザには、emp に対する暗黙的な select および decrypt パーミッションがあります。

```
exec p_emp

name                           ssn
------------------------------ ------------
Tinna Salt                     123-45-6789
Joe Cool                       321-54-9879
```

emp テーブルおよび p_emp ストアド・プロシージャがそれぞれ別のユーザによって作成された場合、emp の select パーミッションがないとパーミッション・エラーが発生します。select パーミッションを持っていても decrypt パーミッションがない場合、emp.ssn の decrypt default 値が返されます。

次の例では、データベースを所有していない “joe” が v_emp ビューを作成します。このビューは、emp テーブルから選択します。ビューに付与されているパーミッションは、ベース・テーブルに暗黙的に適用されません。

```
create view v_emp as
     select name, ssn from emp
grant select on v_emp to emp_role
grant select on emp to emp_role
grant decrypt on v_emp to emp_role
```

emp_role を持つユーザが次のコマンドを発行します。

```
select * from joe.v_emp
```

emp_role には dbo.emp.ssn の decrypt パーミッションが付与されておらず、dbo.emp.ssn における emp_role への暗黙的な grant もないので、以下の結果が返されます。

```
name                        ssn
--------------------------  ---------------
Tinna Salt                  000-00-0000
Joe Cool                    000-00-0000
```

decrypt default と *insert*、*update*、および *delete* 文

decrypt default パラメータは、insert および update 文のターゲット・リストに影響しません。

復号化デフォルト値を含むカラムを update または delete 文の where 句で使用した場合、ローが更新または削除されないことがあります。たとえば、emp テーブルおよび前の例のパーミッションで、hr_role のないユーザが次のクエリを発行しても、ユーザの名前は削除されません。

```
delete emp where ssn = '123-45-6789'
(0 rows affected)
```

復号化デフォルト属性は、グラフィカル・ユーザ・インタフェース (GUI) を備えたアプリケーションで以下のような処理を行った場合にデータの挿入および更新に間接的に影響することがあります。

1 データの選択

2 ユーザによるデータの更新の許可

3 同一または別のテーブルへの変更されたローの適用

ユーザが暗号化カラムの decrypt パーミッションを持たない場合、アプリケーションが復号化デフォルト値を取得し、変更されていない復号化デフォルト値をテーブルに自動的に書き込むことがあります。有効な値が復号化デフォルト値で上書きされることを防止するには、検査制約を使用して、これらの値が自動的に適用されないようにします。次に例を示します。

```
create table customer (name char(30)),
cc_num int check (cc_num != -1)
encrypt decrypt_default -1
```

cc_num に対する decrypt パーミッションを持たないユーザが、customer テーブルからデータを選択すると、次のデータが表示されます。

```
name                               cc_num
---------------------              ------------
Paul Jones                         -1
Mick Watts                         -1
```

しかし、ユーザが名前を変更してデータベースを更新すると、アプリケーションは表示されている値のすべてのフィールドを cc_num 句のデフォルト値で更新しようとするので、エラー 548 が発行されます。

```
"Check constraint violation occurred, dbname =
<dbname>, table name = <table_name>, constraint name =
<internal_constraint _name>"
```

検査制約を設定すると、データの整合性が保証されます。アプリケーションのロジックを記述する際に、これらの更新をフィルタして整合性の再確認もできます。

復号化デフォルト値の削除

以下のコマンドを使用して、復号化デフォルト値を削除できます。

- drop table
- alter table .. modify .. drop col
- alter table .. modify .. decrypt
- alter table .. replace .. drop decrypt_default

たとえば、ssn カラムから復号化デフォルト属性を削除するには、次のコマンドを入力します。

```
alter table emp replace ssn drop decrypt_default
```

テーブルの所有者が復号化デフォルト値を削除した後に hr_role を持たないユーザが emp テーブルから選択すると、エラー・メッセージ 10330 が返されます。

暗号化カラムの長さ

create table、alter table、select into オペレーションの間に、Adaptive Server によって暗号化カラム内の最大長が計算されます。スキーマ配置やページ・サイズを決定するために、データベース所有者は暗号化カラムの最大長を把握する必要があります。

AES はブロック暗号化アルゴリズムです。ブロック暗号化アルゴリズムの暗号化データの長さは、暗号化アルゴリズムのブロック・サイズの倍数です。AES のブロック・サイズは 128 ビットすなわち 16 バイトです。このため、暗号化カラムは、少なくとも 16 バイトに次の領域を加えたサイズになります。

- 初期化ベクトル。初期化ベクトルを使用する場合、各暗号化カラムに 16 バイトが追加されます。デフォルトでは、暗号化プロセスで初期化ベクトルが使用されます。init_vector null を create encryption key に指定すると、初期化ベクトルが省略されます。
- プレーン・テキスト・データの長さ。カラムの型が char、varchar、binary、または varbinary である場合、暗号化前に、データの先頭に 2 バイトが追加されます。この 2 バイトは、プレーン・テキスト・データの長さを表します。プレフィクスの 2 バイトが追加されることで暗号テキストがさらに 1 ブロックを必要としないかぎり、暗号化カラムによってこれ以上の領域が使用されることはありません。
- 識別バイト。後続のゼロがデータベース・システムによってトリムされることを防ぐために、暗号テキストに追加されるバイトです。

表 5-1 の「暗号化データの最大長」の長さは、指定された型と長さのカラムの sycolumns.encrlen の値を表しています。

表 5-1: 暗号化カラムのデータ型長

ユーザが指定するカラムのデータ型	入力データ長	暗号化カラムのタイプ	暗号化データの最大長 (init_vector なし)	暗号化データの実際の長さ (init_vector なし)	暗号化データの最大長 (init_vector 使用)	暗号化データの実際の長さ (init_vector 使用)
bigint	8	varbinary	17	17	33	33
unsigned bigint	8	varbinary	17	17	33	33
tinyint、smallint、int (符号付きまたは符号なし)	1、2、または 4	varbinary	17	17	33	33
tinyint、smallint、int (符号付きまたは符号なし)	0 (null)	varbinary	17	0	33	0
float、float(4)、real	4	varbinary	17	17	33	33
float、float(4)、real	0 (null)	varbinary	17	0	33	0
float(8)、double	8	varbinary	17	17	33	33
float(8)、double	0 (null)	varbinary	17	0	33	0
numeric(10,2)	3	varbinary	17	17	33	33

ユーザが指定するカラムのデータ型	入力データ長	暗号化カラムのタイプ	暗号化データの最大長 (init_vector なし)	暗号化データの実際の長さ (init_vector なし)	暗号化データの最大長 (init_vector 使用)	暗号化データの実際の長さ (init_vector 使用)
numeric (38,2)	18	varbinary	33	33	49	49
numeric (38,2)	0 (null)	varbinary	33	0	49	0
char、varchar (100)	1	varbinary	113	17	129	33
char、varchar (100)	14	varbinary	113	17	129	33
char、varchar (100)	15	varbinary	113	33	129	49
char、varchar (100)	31	varbinary	113	49	129	65
char、varchar (100)	0 (null)	varbinary	113	0	129	0
binary、varbinary(100)	1	varbinary	113	17	129	33
binary、varbinary(100)	14	varbinary	113	17	129	33
binary、varbinary(100)	15	varbinary	113	33	129	49
binary、varbinary(100)	31	varbinary	113	49	129	65
binary、varbinary(100)	0 (null)	varbinary	113	0	65	0
unichar(10)	2 (1 unichar 文字)	varbinary	33	17	49	33
unichar(10)	20 (10 unichar 文字)	varbinary	33	33	49	49
univarchar(20)	20 (10 unichar 文字)	varbinary	49	33	65	49
date	4	varbinary	17	17	33	33
time	4	varbinary	17	17	33	33
time	NULL	varbinary	17	0	33	0
smalldatetime	4	varbinary	17	17	33	33
datetime	8	varbinary	17	17	33	33
smallmoney	4	varbinary	17	17	33	33
money	8	varbinary	17	17	33	33
money	NULL	varbinary	17	0	33	0
bit	1	varbinary	17	17	33	33

注意

- timestamp データ型は、Adaptive Server でサポートされません。
- char と binary は可変長データ型として扱われ、暗号化の前にブランクとゼロの埋め込みが削除されます。データが復号化されるときはブランクと 0 の埋め込みが適用されます。
- ディスク上のカラム長は、カラムの暗号化に合わせて増加しますが、この増加分はツールやコマンドでは認識されません。たとえば、sp_help にも元のサイズのみが表示されます。

第 6 章

暗号化データへのアクセス

暗号化カラムのデータを処理すると、Adaptive Server で暗号化と復号化が自動的に行われます。Adaptive Server では、暗号化カラムにデータの更新または挿入を行うとデータが暗号化され、そのデータを選択するか where 句で使用すると、データが復号化されます。

暗号化カラムの処理

暗号化カラムに対して select、insert、update、または delete コマンドを発行すると、Adaptive Server では、暗号化カラムに関連付けられている暗号化キーを使ってデータが自動的に暗号化または復号化されます。

- 暗号化カラムで insert または update を発行する場合は、次のようになります。
  - 暗号化カラムに対する insert または update パーミッションがない場合、コマンドは失敗します。
  - ユーザ指定のパスワードを使用してキーでカラムを暗号化している場合、Adaptive Server はパスワードを使用可能とみなします。ユーザ指定のパスワードが設定されていない場合、コマンドは失敗します。「ユーザ・パスワードを使用した暗号化データへのアクセス」(70 ページ) を参照してください。
  - Adaptive Server は暗号化キーを復号化します。
  - Adaptive Server はカラムの暗号化キーを使用してデータを暗号化します。
  - Adaptive Server は varbinary 暗号テキスト・データをテーブルに挿入します。
  - 挿入または更新のあと、Adaptive Server はプレーン・テキストを保持しているメモリをクリアします。文の終わりに、生の暗号化キーを保持するメモリがクリアされます。

- 暗号化カラムからのデータに対して select コマンドを発行する場合は、次のようになります。
  - 暗号化カラムに対する select パーミッションがない場合、コマンドは失敗します。
  - 暗号化キーがユーザ指定のパスワードを使って暗号化されたカラムと関連付けられている場合、Adaptive Server はパスワードを使用可能とみなします。ユーザ指定のパスワードが設定されていない場合、select 文は失敗します。「ユーザ・パスワードを使用した暗号化データへのアクセス」(70 ページ) を参照してください。それ以外の場合、Adaptive Server は暗号化キーを復号化します。
  - カラムに対する decrypt パーミッションがある場合、選択したデータの復号化は成功し、Adaptive Server はプレーン・テキスト・データをユーザに返します。
  - 暗号化カラムで復号化のデフォルトが宣言されている場合、およびカラムに対する decrypt パーミッションがない場合、Adaptive Server は復号化のデフォルト値を返します。
- 暗号化カラムを where 句に含める場合は、次のようになります。
  - カラムに対する decrypt パーミッションがなく、カラムに復号化のデフォルトが含まれている場合、where 句の述部は false に評価されます。「復号化デフォルト・カラムおよびクエリ条件」(50 ページ) を参照してください。
  - 次に該当する場合、可能であれば Adaptive Server はデータを復号化せずに比較を行います。
    - where 句で、初期化ベクトルまたはランダム埋め込みを使用せずに暗号化カラムと別の暗号化カラムが同じキーでジョインされている場合
    - カラム・データが等号または不等号条件で定数値と照合されている場合

    「パフォーマンスの考慮事項」(85 ページ) を参照してください。

復号化のためのパーミッション

暗号化データを表示または処理するために、ユーザには以下が必要です。

- ターゲット・リスト内および where、having、order by、group by、およびその他の句内で使用されているカラムに対する select パーミッションと decrypt パーミッション。
- passwd *password_phrase* 句で create または alter encryption key コマンドを使用する場合に、キーの暗号化に使用するパスワード。「第 7 章 管理者からのデータのプライバシーの保護」を参照してください。

制限されている decrypt permission に対して Adaptive Server を設定すると、暗黙的な復号化パーミッションが制限されます。テーブルの所有者に decrypt パーミッションを明示的に付与して、所有者が所有しているテーブルの暗号化カラムから選択できるようにする必要があります。ストアド・プロシージャに対する execute パーミッションまたはビューに対する select パーミッションは、基になるテーブルに対する decrypt パーミッションをユーザに明示的に付与するものではありません。ユーザには、基本テーブルに対する明示的な decrypt パーミッションも必要です。

暗号化の削除

テーブルを所有している場合は、alter table で decrypt オプションを指定して、カラムの暗号化を削除できます。

たとえば、customer テーブルの creditcard カラムの暗号化を削除するには、次のコマンドを入力します。

```
alter table customer modify creditcard decrypt
```

creditcard カラムが明示的なユーザ・パスワードを使用してキーで暗号化されている場合は、最初にそのパスワードを設定する必要があります。

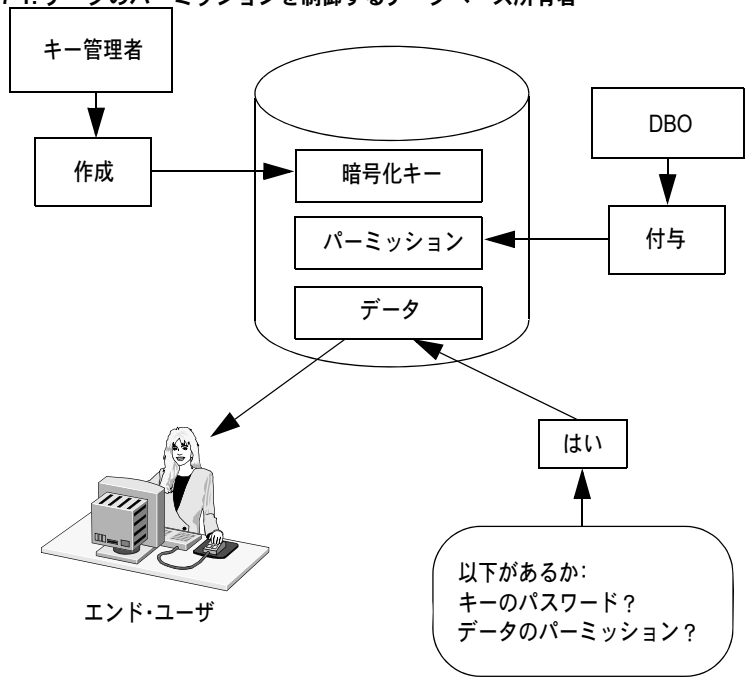
第 7 章

管理者からのデータのプライバシーの保護

キー管理者の役割

keycustodian_role が割り当てられる必要のあるキー管理者は、暗号化キーを管理します。keycustodian_role の役割を使用すると、管理者がデータに暗黙的にアクセスしないようにすることで、機密データを管理する作業を分離できます。図 7-1 は、スキーマ所有者であるデータベース所有者はデータにアクセスするためのパーミッションを制御していても、キーのパスワードがわからない場合はそのデータにアクセスできないことを示しています。一方、キー管理者は暗号化キーとパスワードを管理しますが、データに対するパーミッションは持っていません。データにアクセスできるのは、データに対するパーミッションを持ち、暗号化キーのパスワードを知っている適格なエンド・ユーザだけです。

図 7-1: データのパーミッションを制御するデータベース所有者



システム管理者およびデータベース所有者には、暗黙的なキー管理責任はありません。Adaptive Server ではシステム役割 keycustodian_role が提供されているため、SSO はすべての暗号化の責任を負う必要はありません。キー管理者は暗号化キーを所有しますが、データに対する明示的または暗黙的なパーミッションを持つべきではありません。データベース所有者は、カラム・パーミッションを介してユーザにデータへのアクセスを許可しますが、キー管理者は、ユーザにキーのパスワードへのアクセスを提供します。keycustodian_role は sso_role に自動的に付与されますが、sso_role を持つユーザは付与もできます。

キー管理者は、以下のことを行うことができます。

- 暗号化キーの作成および変更
- 所有するキーのデータベース・デフォルト・キーとしての割り当て (現在のデフォルト・キーがある場合はそれを所有している必要があります)
- 特定のユーザのキー・コピーの設定 (各ユーザは、指定したパスワードまたはログイン・パスワードを使用してキーにアクセスできます)
- エンド・ユーザとのキー暗号化パスワードの共有
- キー管理者が所有しているキーの暗号化キーに対する select アクセスのスキーマ所有者への付与
- マスタ・キーの作成またはシステム暗号化パスワードの設定

- 暗号化キーのリカバリ
- 所有する暗号化キーの削除
- 所有するキーの所有権の変更

複数のキー管理者を設定できます。それぞれのキー管理者は、独自のキー・セットを所有します。キー管理者は、スキーマ所有者に create table、alter table、および select into でキーを使用するパーミッションを付与します。また、権限を持つユーザにキー・パスワードを開示することや、ユーザによるキー・コピーへの個人パスワードまたはログイン・パスワードの割り当てを許可できます。キー管理者は、パスワードが失われたときや災害が発生したときに「キー・リカバリ・ユーザ」とともにキーをリカバリできます。キー管理者が退職した場合は、SSO が alter encryption key コマンドを使用してキーの所有権を新しいキー管理者に変更できます。

ユーザ、役割、およびデータ・アクセス

暗号化キーにユーザ指定のパスワードを設定することにより、データ・プライバシーがシステム管理者から保護されます。表 7-1 に次の方法を示します。

- キー管理者がキーを所有する方法 (ただし、データを表示することはできません)
- データベース所有者がスキーマを所有する方法 (ただし、データを表示することはできません)
- 以下を使用してユーザがデータを表示および処理する方法
  - キー管理者から付与されたキー・アクセス
  - テーブル所有者から付与されたデータ・アクセス

表 7-1: 暗号化カラムに関するユーザおよび役割のパーミッション

役割	キーの作成	スキーマ定義でのキーの使用	暗号化データの復号化
sso_role	可	不可。create table パーミッションが必要です。	不可。役割を持つユーザはパスワードを知っている可能性がありますが、テーブルの select パーミッションが必要です (SSO は暗黙的な decrypt パーミッションを持っています)。
sa_role	不可。create encryption key パーミッションが必要です。	可。しかし、キーの select パーミッションが付与されている必要があります。	不可。パスワードを知っている必要があります。
keycustodian_role	可	不可。create table パーミッションが必要です。	不可。役割を持つユーザはパスワードを知っている可能性がありますが、テーブルまたはカラムの decrypt および select パーミッションが必要です。

役割	キーの作成	スキーマ定義でのキーの使用	暗号化データの復号化
データベース所有者またはスキーマ所有者	不可。create encryption key パーミッションが必要です。	可。しかし、キーの select パーミッションが付与されている必要があります。	不可。パスワードを知っている必要があります。
ユーザ	不可。	不可。	可。しかし、decrypt または select パーミッションが付与されていて、キーのパスワードを知っている必要があります。

ユーザ指定のパスワードを使用したキーの保護

create encryption key コマンドを使用して、パスワードをキーに関連付けます。

```
create encryption key [[db.][owner].]keyname  [as default]
    [ for algorithm_name ]
    [with {[keylength num_bits]
    [passwd 'password_phrase']
    [init_vector {NULL | random}]
    [pad {NULL | random}]}]
```

password_phrase は、引用符で囲まれた最長 255 バイトの英数字文字列で、キー暗号化キー (KEK) を生成するために使用されます。

ユーザ指定のパスワードは Adaptive Server には保存されません。保存されるのは、sysencryptkeys.eksalt 内の「ソルト」として知られる検証バイトの文字列です。この文字列により、後で暗号化または復号化を行う際に使用されるパスワードがキーの正しいパスワードであるかどうかが Adaptive Server で認識されます。keyname によって暗号化されたカラムにアクセスするには、パスワードを入力する必要があります。

暗号化キーを作成すると、sysencryptkeys テーブル内のそのエントリはベース・キーとして認識されます。一部のユーザとアプリケーションについては、マスタ・キー、システム暗号化パスワード、または明示的なパスワードで暗号化された「ベース・キー」で十分です。明示的なパスワードは、キーへのアクセスを必要とするユーザ間で共有されます。また、さまざまなユーザとアプリケーションに対する「キー・コピー」を作成できます。各キー・コピーは個々のパスワードで暗号化され、sysencryptkeys 内の別のローとして格納されます。暗号化キーは常に 1 つのベース・キーおよび 0 以上のキー・コピーで表されます。

次の例では、キーに対してパスワードを使用する方法、および暗号化の設定におけるキー管理者の役割を示します。キーのパスワードは、暗号化データを処理する業務ニーズのあるすべてのユーザで共有されます。

1　キー管理者 “razi” が暗号化キーを作成します。

```
create encryption key key1
     with passwd 'Worlds1Biggest6Secret'
```

2　“razi” は、暗号化データにアクセスする必要のあるすべてのユーザにパスワードを配布します。

3　各ユーザは、暗号化カラムを含むテーブルを処理する際にパスワードを入力します。

```
set encryption passwd 'Worlds1Biggest6Secret'
     for key razi.key1
```

4　不正なユーザがパスワードを取得してキーの機密性が損なわれた場合、“razi” は、キーを変更してパスワードを変更します。

キーの保護方法の変更

次の alter encryption key コマンドを使用して、暗号化キーの保護方法を変更できます。

alter encryption key [[*database*.database][*owner*].] *keyname*
[with {passwd {'old_passwd' | system_encr_passwd
| login_passwd} | master key}]
modify encryption
[with [{passwd {'old_passwd' | system_encr_passwd | login_passwd} | master key}] [[no] dual_control]]

それぞれの意味は、次のとおりです。

- *keyname* − カラム暗号化キーを識別します。
- with passwd '*old_password*' − ベース・キーまたはキー・コピーを暗号化するために create encryption key または alter encryption key 文で以前に指定されていたユーザ定義のパスワードを指定します。パスワードの最大長は 255 バイトです。ベース・キーに with passwd を指定しない場合、デフォルト値はマスタ・キーまたはシステム暗号化パスワードになります。
- with passwd '*new_password*' − カラム暗号化キーまたはキー・コピーを暗号化するために Adaptive Server が使用する新しいパスワードを指定します。パスワードの最大長は 255 バイトです。ベース・キーに with passwd を指定せず、ベース・キーを暗号化している場合、デフォルト値は、system_encr_passwd になります。

- system_encr_passwd – デフォルトの暗号化パスワードです。1 つまたは複数のキー・コピーがすでに存在する場合は、システム暗号化パスワードで暗号化されるベース・キーは変更できません。この制限により、キー管理者が各ユーザによる制限使用のためにキーを設定した後に、キー管理者が暗号化キーを誤って管理者に公開することが防止できます。システム暗号化パスワードを使用して暗号化するキー・コピーは変更できません。
- *login_passwd* – 現在のセッションのログイン・パスワードです。暗号化に *login_password* を使用するベース・キーを変更することはできません。ユーザは、自分のログイン・パスワードで暗号化する自分のキー・コピーを変更できます。

  alter encryption key コマンドを実行するためにユーザのログイン・パスワードが割り当てられたキー・コピーを必要とせずにユーザのログイン・パスワードでキー・コピーを暗号化する別の方法については、「キー・コピーにログイン・パスワードを使用したアプリケーションの透過性」(73 ページ) を参照してください。
- master key – 最初のインスタンスでは、現在の暗号化にマスタ・キーを使用していることを示します。2 番目のインスタンスでは、CEK をマスタ・キーで再暗号化する必要があることを示します。

例 1 – 次の例では、パスワードの機密性が失われた場合、またはパスワードを知っているユーザが退職した場合にキー管理者がベース・キーを変更します。

1 キー管理者 “razi” が暗号化キーを作成します。

```
create encryption key key1
     with passwd 'MotherOfSecrets'
```

2 “razi” は、暗号化データを処理する必要がある “joe” および “bill” とベース・キーのパスワードを共有します (ここでは、キー・コピーは考慮の対象に含まれません)。

3 “joe” が退職します。

4 “razi” は暗号化キーのパスワードを変更して、“bill”、および “joe” の後任ユーザである “pete” とパスワードを共有します。変更されたのはキーを保護する方法だけで、基盤となるキーは変更されていないので、データを再暗号化する必要はありません。以下の文は、古いパスワードを使用して key1 を復号化し、新しいパスワードで再暗号化します。

```
alter encryption key key1
    with passwd 'MotherOfSecrets'
    modify encryption
    with passwd 'FatherOfSecrets'
```

例 2 – マスタ・キーを使用して既存の CEK “k2” を暗号化します。

```
alter encryption key k2
        with passwd 'goodbye'
        modify encryption
        with master key
```

例 3 − 現在マスタ・キーによって暗号化されている既存の CEK “k3” を再暗号化し、デュアル・コントロールを使用するようにします。

```
alter encryption key k3
        modify encryption
        with master key
        dual_control
```

注意　この例では with master key を省略しても同じ暗号化が得られます。

例 4 − 現在マスタ・キーおよびパスワード “k4_password” によって暗号化されている既存の CEK “k4” を再暗号化し、デュアル・コントロールを使用するようにします。CEK およびそのすべてのキー・コピーは、“k4_new_password” から導出された単一のキーによって制御されます。

```
alter encryption key k4
        with passwd 'k4_password'
        modify encryption
        with passwd 'k4_new_password'
        no dual_control
```

例 5 − マスタ・キーおよびパスワード “k5_password” によって暗号化されているデュアル・コントロールに対し、現在マスタ・キーで暗号化されている既存の CEK “k5” を暗号化します。

```
alter encryption key k5
        modify encryption
        with passwd 'k5_password'
        dual_control
```

例 6 − マスタ・キーおよびパスワード “k6_password” によるデュアル・コントロールに対して CEK を暗号化します。

```
create encryption key k6
        with passwd 'k6_password'
        dual_control
```

ユーザ “ned” の場合、マスタ・キーおよびパスワード “k6_ned_password” によるデュアル・コントロールに対して、現在マスタ・キーおよびパスワード “k6_password” によるデュアル・コントロールで暗号化されている CEK “k6” の現在のキー・コピーを暗号化します。

```
alter encryption key k6
        with passwd 'k6_password'
        add encryption
        with passwd 'k6_ned_password'
        for user ned
```

注意　ユーザ “ned” は、自分のキー・コピーの dual control プロパティを変更できません。

例 7 – 現在マスタ・キーおよびデュアル・マスタ・キーで暗号化されている CEK “k7” を暗号化し、システム暗号化パスワードを使用するようにします。

```
alter encryption key k7
        modify encryption
        with passwd system_encr_passwd
        no dual control
```

キー・コピーの作成

キー管理者は、暗号化カラムにデータをロードする必要のある管理者またはオペレータが一時的にキー・コピーを利用できるようにする必要があります。このオペレータは暗号化データにアクセスするパーミッションを持っていないため、キーに永続的にはアクセスできません。

次に示すように、キー・コピーを各ユーザが使用できるようにすることができます。

- キー管理者が create encryption key を使用して、ユーザ定義のパスワードでキーを作成します。このキーは「ベース」キーと呼ばれます。
- キー管理者が alter encryption key を使用して、ベース・キーのコピーを各ユーザのパスワードで各ユーザに割り当てます。

次の構文は、目的のユーザに明示的なパスワードを使用して暗号化されたキーを追加する方法を示します。

```
alter encryption key [database.[ owner ].]key
        with passwd 'base_key_password'
        add encryption with passwd 'key_copy_password'
        for user_name ''
```

それぞれの意味は、次のとおりです。

- *base_key_password* – ベース・キーを暗号化するために使用するパスワードです。このパスワードを知っているのはキー管理者だけです。パスワードの最大長は 255 バイトです。Adaptive Server では、最初のパスワードを使用してベース・カラム暗号化キーが復号化されます。
- *key_copy_password* – キー・コピーを暗号化するために使用するパスワードです。パスワードの最大長は 255 バイトです。Adaptive Server では、復号化ベース・キーのコピーが作成され、*key_copy_password* から導出されたキー暗号化キーで暗号化されます。次に、暗号化ベース・キーのコピーが新しいローとして sysencryptkeys に保存されます。
- *user_name* – キー・コピーを割り当てるユーザを識別します。sysencryptkeys には、キーのコピーを持つ各ユーザのローが含まれます。このローは、ユーザ ID (uid) によって識別されます。
- キー管理者は、プライベート・パスワードを使用したアクセスを必要とするユーザの数だけキー・コピーを追加します。

- ユーザは、暗号化キーのコピーを変更して、別のパスワードで暗号化できます。

次の例は、キー・コピーを作成して暗号化カラムで使用する方法を示します。

1　キー管理者 “razi” が、ユーザ指定のパスワードでベース暗号化キーを作成します。

```
create encryption key key1 with passwd 'WorldsBiggestSecret'
```

2　“razi” は、スキーマ作成用に key1 の select パーミッションをデータベース所有者に付与します。

```
grant select on key key1 to dbo
```

3　データベース所有者がスキーマを作成して、テーブルおよびカラムレベルのアクセスを “bill” に付与します。

```
create table employee (empname char(50), emp_salary money encrypt with
     razi.key1, emp_address varchar(200))
     grant select on employee to bill
     grant decrypt on employee(emp_salary) to bill
```

4　キー管理者が “bill” のキー・コピーを作成して、そのキー・コピーのパスワードを “bill” に渡します。このパスワードを知っているのは、キー管理者と “bill” だけです。

```
alter encryption key key1 with passwd 'WorldsBiggestSecret'
     add encryption with passwd 'justforBill'
     for user 'bill'
```

5　“bill” が employee.emp_salary にアクセスするとき、最初に自分のパスワードを入力します。

```
set encryption passwd 'justforBill' for key razi.key1
     select empname, emp_salary from dbo.employee
```

Adaptive Server がユーザのキーにアクセスするとき、そのユーザのキー・コピーが検索されます。ユーザのキー・コピーが存在しない場合、ユーザはベース・キーにアクセスしようとしているものと見なされます。

キー・コピーのパスワードの変更

キー・コピーがユーザに割り当てられた後、ユーザは、alter encryption key を使用してキー・コピーのパスワードを変更できます。

次の例は、キー・コピーが割り当てられたユーザが、自分の個人パスワードでデータにアクセスするためにキー・コピーを変更する方法を示します。

- キー管理者 “razi” が “bill” のために既存のキーのキー・コピーを設定して、テンポラリ・パスワードで暗号化します。

```
alter encryption key key1 with passwd 'MotherOfSecrets'
     add encryption with passwd 'just4bill' for user bill
```

- “razi” は、key1 でデータにアクセスするためのパスワードを “bill” に送信します。
- “bill” は、自分のキー・コピーにプライベート・パスワードを割り当てます。

```
alter encryption key razi.key1 with passwd 'just4bill'
     modify encryption with passwd 'billswifesname'
```

“bill” のキー・コピーのパスワードを変更できるのは、“bill” 自身だけです。“bill” が上記のコマンドを入力すると、“bill” のキー・コピーが存在することが確認されます。“bill” のキー・コピーが存在しない場合は、ユーザがベース・キーのパスワードを変更しようとしたと見なされ、エラー・メッセージが発行されます。

```
Only the owner of object '<keyname>' or a user with
     sso_role can run this command.
```

ユーザ "guest" のログイン関連付けのためのキー・コピーは作成できません。キー・コピーをログイン・パスワードで暗号化するには、2 つの手順を実行する必要があります。

ユーザ・パスワードを使用した暗号化データへのアクセス

insert、update、delete、select、alter table、または select into 文でデータを暗号化または復号化するには、暗号化キーのパスワードを入力する必要があります。システム暗号化パスワードによって暗号化キーが保護されている場合は、Adaptive Server がシステム暗号化パスワードにアクセスできるので、ユーザはシステム暗号化パスワードを入力する必要はありません。同様に、キー・コピーがユーザのログイン・パスワードで暗号化されている場合、ユーザがサーバにログインしている間、Adaptive Server は、このパスワードにアクセスできます (「キー・コピーにログイン・パスワードを使用したアプリケーションの透過性」(73 ページ) を参照してください)。キーが明示的なパスワードで暗号化されている場合は、暗号化カラムを暗号化または復号化するコマンドを実行する前に、次の構文を使用してパスワードを設定する必要があります。

```
set encryption passwd 'password_phrase'
     for {key | column} {keyname | column_name}
```

それぞれの意味は、次のとおりです。

- *password_phrase* − キーを保護するために create encryption key または alter encryption key コマンドで指定された明示的なパスワードです。
- key − 名前付きキーによって暗号化されたカラムにアクセスするときに Adaptive Server がこのパスワードを使用してキーを復号化することを示します。
- *keyname* − 完全修飾名として入力できます。次に例を示します。

  [[*database*.][*owner*.]*keyname*

- column − 名前付きカラムを暗号化または復号化するコンテキスト内でのみ Adaptive Server がこのパスワードを使用することを示します。エンド・ユーザは、カラムを暗号化するキーの名前を必ずしも知っておく必要はありません。
- *column_name* − 暗号化パスワードを設定するカラムの名前です。*column_name* は次のように入力します。

  [[*database*.][*owner*].]*table_name.column_name*

明示的なパスワードによって暗号化されたキーにアクセスする必要のあるユーザは、パスワードを入力する必要があります。パスワードは、暗号化された形式でユーザ・セッションの内部コンテキストに保存されます。セッションが終了すると、Adaptive Server のメモリが 0 で上書きされてキーがメモリから削除されます。

次の例は、データを暗号化または復号化する必要があるときに Adaptive Server でパスワードが判断される方法を示します。次のスキーマ作成文に示されるように、employee および payroll テーブルの ssn カラムが key1 で暗号化されていると仮定します。

```
create encryption key key1 with passwd "Ynot387"
create table employee (ssn char (11) encrypt with key1, ename char(50))
create table payroll (ssn char(11) encrypt with key1, base_salary float)
```

1. キー管理者は、employee.ssn へのアクセスに必要なパスワードを “susan” と共有します。その際、キー管理者はキーの名前を開示する必要はありません。
2. “susan” が employee に対する select および decrypt パーミッションを持っている場合、“susan” は、employee.ssn のパスワードを使用して従業員 (employee) データを選択できます。

```
set encryption passwd "Ynot387" for column employee.ssn
     select ename from employee where ssn = '111-22-3456'

ename
----------------------
Priscilla Kramnik
```

3. “susan” が payroll.ssn のパスワードを指定せずに payroll のデータを選択しようとすると、次の select は失敗します (“susan” が payroll の select および decrypt パーミッションを持っていたとしても同じです)。

```
select base_salary from payroll where ssn = '111-22-3456'

You cannot execute 'SELECT' command because the user encryption password
has not been set.
```

このエラーを回避するには、“susan” は最初に次のコマンドを入力する必要があります。

```
set encryption passwd "Ynot387" for column payroll.ssn
```

キー管理者は、ユーザがアプリケーション・コードにキー名をハード・コードすることを防止するために、キー名ではなくカラム名に基づいてパスワードを共有できます。キー名がアプリケーション・コードにハード・コードされている場合、データの暗号化に使用されるキーをデータベース所有者が変更することが困難になることがあります。しかし、1 つのキーで複数のカラムが暗号化されている場合は、パスワードの入力を 1 回だけにする方が便利なことがあります。次に例を示します。

```
set encryption passwd "Ynot387" for key key1
select base_salary from payroll p, employee e
     where p.ssn = e.ssn
        and e.ename = "Priscilla Kramnik"
```

1 つのキーで複数のカラムが暗号化されていて、ユーザがカラムにパスワードを設定する場合、ユーザは、処理するすべてのカラムにパスワードを設定する必要があります。次に例を示します。

```
set encryption passwd 'Ynot387' for column payroll.ssn
set encryption passwd 'Ynot387' for column employee.ssn
select base_salary from payroll  p, employee e
     where p.ssn = e.ssn
     and e.ename = 'Priscilla Kramnik'
```

パスワードが 1 つのカラムに設定されていて、カラムを暗号化するキーに対してキー・レベルでパスワードが設定されている場合、Adaptive Server では、カラムに関連付けられたパスワードが破棄され、キー・レベルのパスワードが維持されます。同じキーまたはカラムに対して 2 つの入力が連続して行われた場合、最後に入力された内容だけが保持されます。次に例を示します。

1 ユーザが、カラム employee.ssn の正しいパスワード "Ynot387" を誤って "Unot387" と入力したとします。

```
set encryption passwd "Unot387"
     for column employee.snn
```

2 その後、ユーザが正しいパスワードを再入力したすると、Adaptive Server では 2 回目の入力だけが保持されます。

```
set encryption passwd "Ynot387"
     for column employee.ssn
```

3 さらに、ユーザが同じパスワードをキー・レベルで入力したすると、Adaptive Server では、この入力だけが保持されます。

```
set encryption passwd "Ynot387" for key key1
```

4 次に、ユーザが同じパスワードをカラム・レベルで入力すると、Adaptive Server では、そのパスワードがすでにキー・レベルで保持されているので、この入力は破棄されます。

```
set encryption passwd "Ynot387"
     for column payroll.ssn
```

ストアド・プロシージャまたはトリガがユーザ指定のパスワードによって暗号化されているカラムを参照する場合、プロシージャ、またはトリガを呼び出す文を実行する前に暗号化パスワードを設定する必要があります。

注意 sp_helptext を介してパスワードが意図せずに開示される可能性があるので、トリガやプロシージャの内部に set encryption passwd 文を配置することは推奨されません。さらに、ハードコードされたパスワードの場合、パスワードが変更されたときにプロシージャまたはトリガを変更する必要があります。

キー・コピーにログイン・パスワードを使用したアプリケーションの透過性

キー管理者は、暗号化用のキー・コピーにユーザのログイン・パスワードを設定できます。キー・コピーにログイン・パスワードを設定した場合、以下の利点があります。

- 利便性 − ログイン・パスワードがキーに割り当てられているユーザは、パスワードを入力せずに暗号化データにアクセスできます。
- セキュリティの向上 − ユーザが管理する必要のあるパスワードの数が減るので、ユーザがパスワードを紙にメモする可能性が減少します。
- キー管理者の管理オーバーヘッドの軽減 − キー管理者は、プライベート・パスワードによるキー・アクセスを必要とする個々のユーザにテンポラリ・パスワードを手動で配布する必要がありません。
- アプリケーションの透過性 − 暗号化データを処理するためのパスワードがアプリケーションから要求されません。既存のアプリケーションで管理者の権限からデータのプライバシーを保護できます。

ユーザのログイン・パスワードでキー・コピーを暗号化するには、次の構文を使用します。

```
alter encryption key [[database.][owner].]keyname
    with passwd 'base_key_password'
    add encryption for user 'user_name' for login_association
```

ここで、login_association は、指定されたユーザのキー・コピーを作成するよう Adaptive Server に指示します。このユーザは、自分のキー・コピーを後で自分のログイン・パスワードで暗号化します。キー・コピーをログイン・パスワードで暗号化するには、次の手順を実行する必要があります。

1　キー管理者は、alter encryption key を使用して、ログイン・パスワードでのキー・アクセスを必要とする各ユーザのキー・コピーを作成します。キー・コピーには、キー・コピーを特定のユーザに確実に関連付けるための情報が添付されています。識別情報およびキーは、マスタ・キーまたはシステム暗号化パスワード (マスタ・キーが存在しない場合) から導出されたキーを使用して一時的に暗号化されます。キー・コピーが sysencryptkeys に保存されます。

2 キーの検索を必要とするカラムをユーザが処理すると、そのユーザに対して識別されている暗号化キーのコピーにログイン・パスワードを関連付けできることが確認されます。Adaptive Server では、キー・コピーに含まれる情報がマスタ・キーまたはシステム暗号化パスワードで復号化され、キー・コピーに関連付けられたユーザ情報がユーザのログイン・クレデンシャルと比較されます。次に、現在のログイン・セッションで入力されたユーザのログイン・パスワードから導出された KEK でキー・コピーが暗号化されます。

login_association に alter encryption key を使用してキー・コピーを追加する場合は、キー・コピーの暗号化にマスタ・キーまたはシステム暗号化パスワードを使用できるようにする必要があります。システム暗号化パスワードは、ユーザがログインするときにキー・コピーを復号化するためにも Adaptive Server で使用可能である必要があります。キー・コピーがユーザのログイン・パスワードによって再暗号化された後は、システム暗号化パスワードは必要なくなります。

次の例では、ユーザの暗号化キー・コピー key1 をユーザのログイン・パスワードで暗号化する方法を示します。

1 キー管理者 “razi” が暗号化キーを作成します。

```
create encryption key key1 for AES
     with passwd 'MotherofSecrets'
```

2 “razi” は、“bill” のために key1 のコピーを作成し、マスタ・キーまたはシステム暗号化パスワードで一時的に暗号化します (このコピーは、後で “bill” のログイン・パスワードで暗号化されます)。

```
alter encryption key key1 with
     passwd 'MotherofSecrets'
     add encryption
     for user 'bill'
     for login_association
```

3 キーと “bill” のキー・コピーを識別する情報の組み合わせがマスタ・キーまたはシステム暗号化パスワードで暗号化され、その結果が sysencryptkeys に格納されます。

4 “bill” が Adaptive Server にログインして、key1 を使用する必要があるデータ処理を行います。たとえば、emp.ssn が key1 によって暗号化されている場合、次のように入力します。

```
select * from emp
```

Adaptive Server では、“bill” の key1 のコピーを Bill のログイン・パスワードで暗号化する必要があることが認識されます。手順 4 で保存されたキー値データがマスタ・キーまたはシステム暗号化パスワードで復号化されます。この情報は、Adaptive Server によって現在のログイン・クレデンシャルと比較され、key1 のキー値が “bill” のログイン・パスワードから導出された KEK で暗号化されます。

5　次回以降のログインでは、“bill” が key1 によって暗号化されたカラムを処理するとき、Adaptive Server では、“bill” の内部セッション・コンテキストを介して Adaptive Server で使用できる “bill” のログイン・パスワードで key1 を復号化して key1 に直接アクセスします。

“bill” のエイリアスが設定されているユーザのログイン・パスワードでは key1 を復号化できないので、これらのユーザは key1 によって暗号化されているデータにはアクセスできません。

6　機密データを処理する権限を “bill” が失った場合、キー管理者は、キーに対する “bill” のアクセスを破棄します。

```
alter encryption key key1
     drop encryption
     for user 'bill'
```

ユーザは、alter encryption key で with passwd login_passwd 句を使用して、ログイン・パスワードでキー・コピーを直接暗号化できます。しかし、この方法には、ログインにおける次のような欠点があります。

- キー管理者は、ユーザに割り当てたパスワードが設定されたキー・コピーを配布する必要があります。
- ユーザは、alter encryption key を発行して、ログイン・パスワードでキー・コピーを再暗号化する必要があります。

次に例を示します。

- “razi” が、明示的なパスワードで暗号化されたキー・コピーをユーザ “bill” 用に追加します。

```
alter encryption key key1
     with passwd 'MotherofSecrets'
add encryption with passwd 'just4bill'
     for user bill
```

- “razi” は、キー・コピーのパスワードを “bill” と共有します。
- “bill” は、自分のキー・コピーを自分のログイン・パスワードで暗号化して利便性を図ります。

```
alter encryption key key1 with passwd "just4bill" modify
encryption with passwd login_passwd
```

- “bill” が暗号化カラムを処理するとき、Adaptive Server は、“bill” のログイン・パスワードを介して、そのキー・コピーにアクセスします。

ログイン・パスワードの変更およびキー・コピー

1 つまたは複数のキーにおいてログイン・パスワードで暗号化されたキー・コピーを持っているユーザがログイン・パスワードを変更した後は、キー・コピーを変更する必要はありません。ログイン・パスワードの変更の一環として、キー・コピーは sp_password によって古いログイン・パスワードで復号化された後に新しいログイン・パスワードで再暗号化されます。

SSO が sp_password を使用して、ユーザの古いパスワードを提供せずにユーザのパスワードを変更した場合、ユーザのキー・コピーは sp_password によって破棄されます。この処理により、管理者が既知のパスワードを使用してキーにアクセスすることが防止されます。この種の必須のパスワード変更の後、キー管理者は、alter encryption key を使用して、パスワードが変更されたユーザの login_association にキー・コピーを追加する必要があります。sp_password ではオフライン・データベースが無視されるので、キー管理者は、オフライン・データベースに格納されているキーに対して、データベースがオンラインに戻ったときにキー・コピーの失われたパスワードのリカバリ手順を実行する必要があります。「ログイン・パスワードが失われた場合」(78 ページ) を参照してください。

ユーザのパスワードが変更されたとき、キー管理者は、-p フラグを使用してサーバを起動した後にこれらの手順を実行する必要がある場合もあります。-p フラグを使用する SSO も自分のログイン・パスワードで暗号化されたキー・コピーを使用してキーにアクセスできる場合は、キー管理者は SSO のキー・コピーを破棄して再作成する必要があります。

キー・コピーの破棄

ユーザが異動した場合や退職した場合、キー管理者は、次のコマンドを使用して、このユーザのキー・コピーを破棄します。

```
alter encryption key keyname
    drop encryption for user user_name
```

たとえば、ユーザ “bill” が退職した場合、キー所有者は、“bill” のキー・コピーを破棄して、“bill” が key1 にアクセスすることを禁止できます。

```
alter encryption key key1
    drop encryption for user bill
```

キーの復号化は必要ないので、このコマンドではパスワードは不要です。

drop encryption key により、ベース・キーおよびそのすべてのコピーが削除されます。

第 8 章

失われたパスワードからのキーのリカバリ

キー・コピーのパスワードが失われた場合

ユーザが暗号化キーのパスワードを失った場合、キー管理者は、そのユーザの暗号化キー・コピーを削除して、新しいパスワードが設定された別の暗号化キー・コピーをそのユーザに発行します。

次の例では、キー管理者が key1 のコピーを “bill” に割り当て、“bill” が key1 のパスワードを自分だけが知っているパスワードに変更しています。そのパスワードを失った “bill” は、キー管理者に新しいキー・コピーを要求します。

1 キー管理者は、Bill のキー・コピーを削除します。

```
alter encryption key key1
     drop encryption for user bill
```

2 キー管理者は、ユーザ “bill” に key1 の新しいコピーを作成して、“bill” にパスワードを与えます。

```
alter encryption key key1
     with passwd 'MotherofSecrets'
     add encryption with passwd 'over2bill'
     for user bill
```

3 “bill” には、key1 の自分のコピーを変更するパーミッションが自動的に付与されます。

```
alter encryption key key1
     with passwd 'over2bill'
     modify encryption
     with passwd 'billsnupasswd'
```

ログイン・パスワードが失われた場合

自分のログイン・パスワードで暗号化されたキー・コピーを持つユーザ “bill” がログイン・パスワードを失った場合、次の手順を実行して、暗号化キーに対する “bill” のアクセスをリカバリできます。

1 SSO が sp_password を使用して、“bill” に新しいログイン・パスワードを発行します。Adaptive Server では、“bill” のログイン・パスワードに割り当てられたキー・コピー、または “bill” のログイン・パスワードで暗号化されたキー・コピーが削除されます。

2 キー管理者は、ログイン関連付けによるキー暗号化の設定の通常の手順を実行します。キー管理者は、マスタ・キーまたはシステム暗号化パスワードが設定されたことを確認し、“bill” のキー・コピーを次のように作成します。

```
alter encryption key k1
     with passwd 'masterofsecrets'
     add encryption for bill
     for login_association
```

ここでは、キー管理者がベース・キーのパスワードを知っていると仮定します。キーの暗号化パスワードが不明な場合、キー管理者は最初にキーのリカバリ手順を実行する必要があります。詳細については、「ベース・キーのパスワードが失われた場合」(78 ページ) を参照してください。

3 次回 “bill” が k1 によって暗号化されたデータにアクセスすると、“bill” の新しいログイン・パスワードを使用して、“bill” のキー・コピーが再暗号化されます。たとえば、emp_salary がキー k1 によって暗号化されている場合、次の文を使用すると、“bill” のキー・コピーが自動的に “bill” のログイン・パスワードで再暗号化されます。

```
select emp_salary from emp
     where name like 'Prisicilla%'
```

ベース・キーのパスワードが失われた場合

ベース・キーのパスワードが失われた場合、キー管理者は、キー・リカバリを使用できます。パスワードがないとキー管理者はキーのパスワードの変更やキー・コピーの追加を行うことができないので、キー・リカバリは重要です。

すべてのユーザがベース・キーによるデータへのアクセスを共有し、1 人のユーザがパスワードを忘れた場合、このユーザは、別のユーザまたはキー管理者からパスワードを取得できます。パスワードを覚えているユーザがいない場合は、データへのアクセスが失われます。このため、リカバリ用に使用できるベース・キーのコピーを作成し、キーをバックアップしておくことを推奨します。このコピーを、キー・リカバリ・コピーと呼びます。

キー管理者には、以下のことが推奨されます。

1　1 人のユーザをキー・リカバリ・ユーザに指定します。キー・リカバリ・ユーザの責任は、キー・リカバリ・コピーのパスワードを覚えておくことです。

2　キー・リカバリ・ユーザ用にベース・キーのコピーを作成します。災害時にリカバリが必要な各キーには、キー・リカバリ・コピーが必要です。

キー・リカバリ・コマンド

Adaptive Server では、リカバリ・キー・コピーでデータにアクセスすることはできません。キー・リカバリ・コピーは、ベース・キーにアクセスするためのバックアップを提供するためにのみ存在します。

リカバリ・キー・コピーを設定するには、次の構文を使用します。

```
alter encryption key keyname with passwd base_key_passwd
add encryption with passwd recovery_passwd
for user key_recovery_user for recovery
```

それぞれの意味は、次のとおりです。

- *base_key_passwd* − キー管理者がベース・キーに割り当てたパスワードです。
- *recovery_passwd* − キー・リカバリ・コピーを保護するために使用するパスワードです。
- *key_recovery_user* − キー・リカバリのパスワードを記憶する責任が割り当てられているユーザです。

キー・リカバリ・コピーを設定した後、キー管理者は、キー・リカバリ・ユーザとパスワードを共有します。キー・リカバリ・ユーザは、次のコマンドを使用してパスワードを自分だけが知っているパスワードに変更できます。

```
alter encryption key keyname with passwd old_recovery_passwd
    modify encryption with passwd new_recovery_passwd for recovery
```

キー・リカバリの際、キー・リカバリ・ユーザは、キー管理者にキー・リカバリ・コピーのパスワードを伝えます。キー管理者は、次の構文を使用してベース・キーへのアクセスを復元します。

```
alter encryption key keyname with passwd recovery_key_passwd
    recover encryption with passwd new_base_key_passwd
```

それぞれの意味は、次のとおりです。

- *recovery_key_passwd* – キー・リカバリ・コピーに割り当てられ、キー・リカバリ・ユーザがキー管理者と共有するパスワードです。Adaptive Server は、*recovery_key_passwd* を使用して、ロー・キーにアクセスするためのキー・リカバリ・コピーを復号化します。
- *new_base_key_passwd* – ロー・キーの暗号化に使用するパスワードです。sysencryptkeys のベース・キー・ローが結果で更新されます。

また、キーの所有権を別のキー管理者に変更する必要がある場合があります。「暗号化キーの所有権の変更」(81 ページ) を参照してください。

次の例では、リカバリ・キー・コピーを設定して、パスワードが失われたときにキー・リカバリに使用する方法を示します。

1 キー管理者が、パスワードで保護された新しい暗号化キーを作成します。

```
create encryption key key1 for AES
     passwd 'loseitl8ter'
```

2 キー管理者は、“charlie” 用に key1 の暗号化キー・リカバリ・コピーを追加します。

```
alter encryption key key1 with passwd 'loseitl8ter'
     add encryption
     with passwd 'temppasswd'
     for user charlie
     for recovery
```

3 “charlie” は、リカバリ・コピーに別のパスワードを割り当て、このパスワードを安全な場所に保管します。

```
alter encryption key key1
     with passwd 'temppasswd'
     modify encryption
     with passwd 'finditl8ter'
     for recovery
```

4 キー管理者がベース・キーのパスワードを失った場合、“charlie” からパスワードを取得し、次のコマンドを使用してリカバリ・コピーからベース・キーをリカバリできます。

```
alter encryption key key1
     with passwd 'finditl8ter'
     recover encryption
     with passwd 'newpasswd'
```

人事異動が発生した場合、キー管理者は、ベース・キーのパスワードを共有するか、キー・コピーの削除と追加を行うことによって、key1 へのアクセスをその他のユーザと共有します。

暗号化キーの所有権の変更

所有権の変更は、通常の業務で発生する場合や、キー・リカバリの一部として発生する場合があります。このコマンドが SSO によって実行された場合、指定されたユーザにキーの所有権が変更されます。

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

ここで、*user_name* は、新しいキーの所有者となるユーザの名前です。このユーザはすでに、キーが作成されたデータベース内のユーザである必要があります。

たとえば、キー encr_key を所有しているキー管理者 “razi” の後任として “tinnap” という新しいキー管理者が就任した場合、次のコマンドを使用してキーの所有権を変更します。

```
alter encryption key encr_key modify owner tinnap
```

このコマンドを実行できるのは、SSO またはキー所有者だけです。

新しい所有者がすでにキー・コピーを持っている場合は、次のようなメッセージが表示されます。

```
A copy of key encr_key already exists for user tinnap
```

キーの通常のキー・コピーまたはリカバリ・キー・コピーをすでに持っているユーザは、キーの新しい所有者になることができません。Adaptive Server では、キーの所有者に対してキー・コピーを作成できません。

前のキーの所有者にキーのパーミッションが付与されていた場合は、sysprotects のシステム・テーブルの付与者 uid が、キーの新しい所有者の uid に変更されます。所有者の変更はすぐに有効になります。新しい所有者は、変更を有効にするためにもう一度ログインする必要はありません。

第 9 章

暗号化カラムの監査

監査オプション

暗号化カラムの監査については、『セキュリティ管理ガイド』の「監査」(特に、event カラムと extrainfo カラムの値がリストされている表 18-5) を参照してください。

監査値

sysaudits の event カラムに表示される値については、『セキュリティ管理ガイド』の「監査」(特に、監査オプション、要件、および例がリストされている表 18-2) を参照してください。

イベントの名前と番号

特定の監査イベントの監査証跡を問い合わせることができます。audit_event_name を使用し、*event id* をパラメータとして指定します。

```
audit_event_name(event_id)
```

sysaudits の event カラムに表示される値については、『セキュリティ管理ガイド』の「監査」(特に、監査イベント値がリストされている表 18-6) を参照してください。

コマンド・テキストの監査におけるパスワードのマスキング

監査レコードではパスワードがマスキングされます。たとえば、SSO がデータベース db1 のユーザ “alan” に対してコマンド・テキストの監査 (特定のユーザのすべてのアクションの監査) を有効にしている場合は、次のようになります。

```
sp_audit "cmdtext", "alan", "db1", "on"
```

次に、“alan” が次のコマンドを発行します。

```
create encryption key key1 with passwd "bigsecret"
```

Adaptive Server は、次のような SQL テキストを audit テーブルの extrainfo カラムに書き込みます。

```
"create encryption key key1 with passwd "xxxxxx"
```

キー管理者のアクションの監視

keycustodian_role がアクティブなすべてのアクションを監視するには、次の構文を使用します。

```
sp_audit "all", "keycustodian_role", "all", "on"
```

第 10 章

パフォーマンスの考慮事項

暗号化は CPU 集約操作であるため、CPU 使用率や暗号化カラムを使用するコマンドの実行時間の面で、アプリケーションにパフォーマンス・オーバヘッドをもたらす場合があります。オーバヘッドの量は、CPU と Adaptive Server エンジンの数、システムへの負荷、暗号化データに同時にアクセスするセッションの数、クエリで参照されている暗号化カラムの数によって異なります。暗号化キーのサイズと暗号化データ長も要因になります。一般的に、キー・サイズが大きくデータが長いほど、暗号化オペレーションにおける CPU 使用率が高くなります。

経過時間は、Adaptive Server オプティマイザが暗号化カラムを使用できるかどうかで異なります。

暗号化カラムのインデックス

カラムの暗号化キーで初期化ベクトルまたはランダム埋め込みが指定されていない場合、暗号化カラムにインデックスを作成できます。初期化ベクトルまたはランダム埋め込みを使用すると、同一のデータが異なるパターンの暗号テキストに暗号化され、インデックスでユニーク性を強制することも、暗号テキストのデータに等号を使用する一致を実行することもできなくなります。

暗号化データのインデックスは、等号および不等号を使用したデータの一致には役立ちますが、データの順序付け、範囲検索、または最小値と最大値の検索には役立ちません。Adaptive Server が暗号化カラムで順序に依存する検索を実行している場合、暗号化データに対してインデックス・ルックアップを実行できません。代わりに、各ローの暗号化カラムは復号化されてから検索されます。このため、データ処理が低速になります。

ソート順と暗号化カラム

大文字と小文字を区別しないソート順を使用する場合、Adaptive Server では、別のカラムとのジョインまたは定数値に基づく検索を実行するときに、暗号化された char または varchar カラムでインデックスを使用できません。これは、アクセント記号を区別しないソート順でも同様です。

たとえば、大文字と小文字を区別しない検索では、文字列 abc は次の範囲のすべての文字列に一致します。abc、Abc、ABc、ABC、AbC、aBC、aBc、abC。Adaptive Server は、abc をこの範囲の値と比較する必要があります。一方、文字列 abc とカラム・データとの大文字と小文字を区別した比較は、同一のカラム値 (abc を含むカラム) のみと一致します。大文字と小文字を区別しない検索と大文字と小文字を区別する検索の大きな違いは、大文字と小文字を区別しない一致の場合は Adaptive Server が範囲検索を実行する必要があり、大文字と小文字を区別する一致では等価探索を実行する必要があります。

非暗号化文字カラムのインデックスでは定義されたソート順に従ってデータの順序が決まります。暗号化カラムの場合、インデックスでは暗号テキスト値に従ってデータの順序が決まります。暗号テキスト値の順序は、プレーン・テキスト値の順序とは関係ありません。したがって、暗号化カラムのインデックスは、等しいか等しくないかを照合するときにのみ役立ち、値の範囲検索には役立ちません。abc および Abc は、異なる暗号テキスト値に暗号化され、インデックス内では隣接する位置に格納されません。

Adaptive Server が暗号化カラムでインデックスを使用している場合、カラム・データは暗号テキスト・フォームで比較されます。大文字と小文字を区別するデータの場合、abc を Abc と一致させないようにし、等号を使用する一致に基づく暗号テキストのジョインまたは検索も実行されないようにする必要があります。Adaptive Server は暗号テキスト値に基づいてカラムをジョインし、where 句の値を効率的に一致させることができます。次の例では、maidenname カラムが暗号化されています。

```
select account_id from customer
     where cname = 'Peter Jones'
     and maidenname = 'McCarthy'
```

maidenname が初期化ベクトルやランダム埋め込みを使用せずに暗号化されているため、Adaptive Server は McCarthy を暗号化して、maidenname の暗号テキスト検索を実行します。maidenname にインデックスがある場合、検索ではそのインデックスを利用します。

暗号化カラムのジョイン

Adaptive Server は、次の場合に、暗号テキストを比較して 2 つの暗号化カラムのジョインを最適化します。

- ジョインするカラムのデータ型が同じ場合。暗号テキストの比較の場合、char および varchar は binary および varbinary と同じデータ型とみなされます。
- int 型と float 型でカラムの長さが同じ場合。numeric 型と decimal 型でカラムの精度と位取りが同じであることが必要です。
- ジョインするカラムが同じキーで暗号化されている場合。
- ジョインするカラムが式に使用されていない場合。たとえば、t.encr_col1 = s.encr_col1 +1 というジョインでは、暗号テキストのジョインを実行できません。
- 暗号化キーが、init_vector と pad を NULL に設定して作成されている場合。
- ジョイン演算子が '=' または '<>' である場合。
- データがデフォルトのソート順を使用している場合。

次の例では、暗号テキストに対してジョインを行うスキーマが設定されます。

```
create encryption key new_cc_key for AES
        with init_vector NULL
create table customer
        (custid int,
        creditcard char(16) encrypt with new_cc_key)
create table daily_xacts
       (cust_id int, creditcard char(16) encrypt with
        new_cc_key, amount money........)
```

次のように、ジョインするカラムにインデックスも設定できます。

```
create index cust_cc on customer(creditcard)

create index daily_cc on daily_xacts(creditcard)
```

Adaptive Server は、次の select 文を実行して、特定のクレジット・カードで顧客の毎日の請求額を集計します。このとき、customer テーブルまたは daily_xacts テーブルの creditcard カラムは復号化されません。

```
select sum(d.amount) from daily_xacts d, customer c
     where d.creditcard = c.creditcard and
     c.custid = 17936
```

探索引数と暗号化カラム

暗号化カラムと定数値が等しいか等しくないかを比較する場合、Adaptive Server はカラムのスキャンを最適化するために、テーブルの各ローの暗号化カラムを復号化するのではなく、定数値を 1 回暗号化します。「暗号化カラムのジョイン」(87 ページ) と同じ制約が適用されます。

次に例を示します。

```
select sum(d.amount) from daily_xacts d
     where creditcard = '123-456-7890'
```

Adaptive Server は、暗号化カラムの範囲検索を実行するときにはインデックスを使用できません。各ローを復号化してからデータ比較を実行する必要があります。クエリに他の述語が含まれている場合、Adaptive Server によって最も適切なジョイン順が選択されます。多くの場合、暗号化カラムに対する検索は、最後に、最も小さいデータ・セットに対して行われます。

クエリに複数の範囲検索があり、有効なインデックスがない場合は、暗号化カラムに対する範囲検索が最後になるようにクエリを作成します。ロード・アイランドの納税者で所得が $100, 000 を超える人の社会保障番号を検索する次の例では、zipcode カラムを、暗号化された調整総所得のカラムに対する範囲検索よりも前に指定します。

```
select ss_num from taxpayers
       where zipcode like '02%' and
       agi_enc > 100000
```

参照整合性検索

参照整合性では、次の両方に該当する場合、暗号テキスト・レベルで一致を検査します。

- プライマリ・キーと外部キーのデータ型が上記で説明したルールに従って一致している。
- プライマリ・キーと外部キーの暗号化が、ジョインするカラムのキー要件を満たしている。

暗号テキストとしての暗号化データの移動

Adaptive Server が暗号化データをコピーするときは、データを復号化してから再暗号化するのではなく、できるかぎり暗号テキストをコピーすることで処理を最適化します。これは、select into コマンド、バルク・コピー、複写に適用されます。

索引

R

S

U

W

あ

い

う

お

か

き

け

こ

さ

し

そ

た

て

と

な

は

ふ

へ

や

ゆ

り

ろ