Nouvelles fonctionnalités Adaptive Server® Enterprise 12.5.3a

Réf. du document : DC00533-01-1253-01

Dernière mise à jour : octobre 2005

Ce document décrit les nouvelles fonctionnalités disponibles dans Adaptive Server® Enterprise 12.5.3a.

1. Cryptage des colonnes

Les mécanismes d'authentification et de contrôle d'accès d'Adaptive Server® garantissent que seuls les utilisateurs correctement identifiés et dûment habilités aient accès aux données. En outre, les fonctionnalités de cryptage protègent les données sensibles stockées sur disque ou archivées contre le vol ou les failles de sécurité.

Le cryptage des données dans Adaptive Server permet de crypter les données au niveau colonne. Seules les données sensibles sont cryptées, ce qui réduit la charge de traitement.

La fonctionnalité Colonnes cryptées d'Adaptive Server est plus facile à utiliser que le cryptage de niveau intermédiaire ou au niveau de l'application cliente. Vous utilisez les instructions sql pour créer les clés de cryptage et spécifier les colonnes à crypter. Adaptive Server gère la génération et le stockage des clés. Le cryptage et le décryptage des données s'effectuent de manière automatique et transparente à mesure que vous lisez et écrivez les données des colonnes cryptées. Aucune modification n'est requise au niveau de l'application, et il n'est pas nécessaire d'acheter de logiciels tiers.

Lorsque les données sont cryptées, elles sont stockées sous la forme de texte chiffré. Dans le cas contraire, elles sont stockées sous la forme de texte simple.

La fonctionnalité de cryptage est également assurée par :

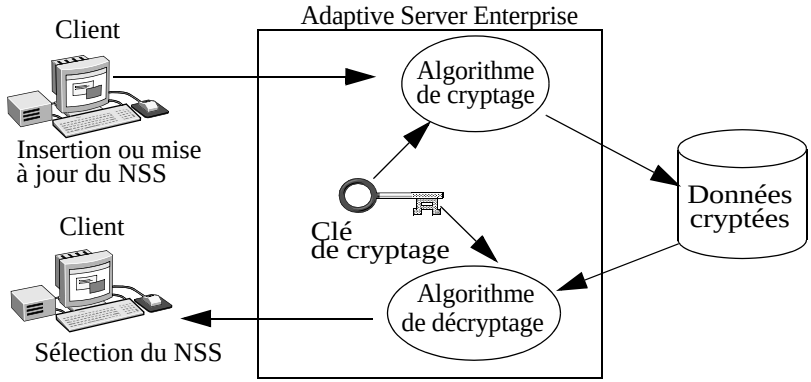
- Sybase Central et le module externe d'Adaptive Server. Reportez-vous au *Guide de l'utilisateur de Sybase Central* pour obtenir de plus amples informations.

- sybmigrate (l'outil de migration), bulk copy et CIS, qui sont documentés dans le manuel *Adaptive Server Enterprise - Guide d'administration système*.
- Replication Server. Veuillez consulter le *Guide d'administration de Replication Server* pour obtenir de plus amples informations sur le cryptage dans le contexte de la réplication.

1.1 Présentation

La Figure 1 offre une vue de haut niveau du cryptage et du décryptage dans Adaptive Server. Dans cet exemple, le numéro de sécurité sociale (NSS) est mis à jour et crypté :

Figure 1 : Cryptage et décryptage dans Adaptive Server



Pour créer des clés de cryptage, utilisez la commande create encryption key, qui :

- crée en interne une clé symétrique pour la longueur de clé spécifiée, à l'aide de l'API Security Builder Crypto™ ;
- stocke la clé sous une forme cryptée dans le catalogue système *sysencryptkeys*.

Adaptive Server conserve une trace de la clé utilisée pour crypter une colonne donnée. Le cryptage des colonnes utilise un algorithme de cryptage symétrique, ce qui signifie que la même clé est utilisée pour le cryptage et le décryptage.

Lorsque vous insérez (commande insert) ou mettez à jour (commande update) des données dans une colonne cryptée, Adaptive Server crypte les données de manière transparente, immédiatement avant d'écrire la ligne. Lorsque vous effectuez une sélection (commande select) dans une colonne cryptée, Adaptive Server décrypte les données après les avoir lues dans la ligne. Les données entières et en virgule flottante sont cryptées sous une forme canonique :

- Format "MSN" (Most Significant Bit, octet le plus significatif) pour les données entières.
- Norme de virgule flottante IEEE (Institute of Electrical and Electronics Engineers) avec format MSB pour les données en virgule flottante.

Les données cryptées sur une plate-forme peuvent être décryptées sur une autre, pour autant que toutes deux utilisent le même jeu de caractères.

Pour utiliser les colonnes cryptées dans Adaptive Server :

1 Installez l'option de licence ASE_ENCRYPTION. Consultez le *Guide d'installation d'Adaptive Server Enterprise* pour obtenir de plus amples informations.

2 Activez le cryptage d'Adaptive Server Enterprise :

```
sp_configure 'enable encrypted columns', 0|1
```

0 – cryptage désactivé.

1 – cryptage activé.

Redémarrez le serveur une fois cette option paramétrée.

Si vous désactivez cette option dans un serveur qui contient des colonnes cryptées, toute commande appliquée à ces colonnes échoue avec un message d'erreur. Tant le paramètre de configuration que l'option de licence sont nécessaires pour utiliser les colonnes cryptées. Seul le responsable de la sécurité du système peut activer les colonnes cryptées.

3 Définissez le mot de passe de cryptage du système pour une base de données à l'aide de la commande sp_encryption. Pour plus d'informations, reportez-vous à la section « Définition du mot de passe de cryptage du système », page 5.

4 Créez la clé de cryptage des colonnes. Pour plus d'informations, reportez-vous à la section « Création des clés de cryptage », page 5.

5 Spécifiez les colonnes à crypter. Reportez-vous aux sections « Spécification du cryptage pour les nouvelles tables », page 11 et « Cryptage de données dans des tables existantes », page 13.

6 Accordez une autorisation de décryptage aux utilisateurs qui doivent voir les données. Reportez-vous à la section « Autorisations de décryptage », page 13.

1.2 Définition du mot de passe de cryptage du système

Le responsable de la sécurité du système utilise sp_encryption pour définir le mot de passe de cryptage du système. Le mot de passe système est spécifique à la base de données où sp_encryption est exécuté, et sa valeur cryptée est stockée dans la table système sysattributes de cette base de données.

sp_encryption system_encr_passwd, *mot_de_passe*

Mot_de_passe peut comporter jusqu'à 64 octets et est utilisé par Adaptive Server pour crypter toutes les clés dans cette base de données. Une fois le mot de passe de cryptage du système défini, il n'est plus nécessaire de le spécifier pour accéder aux clés ou aux données.

Le mot de passe de cryptage du système doit être défini dans toutes les bases de données où des clés de cryptage sont créées.

Le responsable de la sécurité du système peut modifier le mot de passe système à l'aide de la commande sp_encryption et en fournissant l'ancien mot de passe.

sp_encryption system_encr_passwd, *mot_de_passe* [, *ancien_mot_de_passe*]

Lorsque le mot de passe système change, Adaptive Server recrypte automatiquement toutes les clés de la base de données avec le nouveau mot de passe.

1.3 Création et gestion des clés de cryptage

Adaptive Server génère les clés de cryptage et les stocke dans la base de données sous une forme cryptée. Les propriétaires des clés accordent aux propriétaires des tables l'autorisation de crypter des colonnes dans la base de données en cours avec une clé nommée.

1.3.1 Création des clés de cryptage

Toutes les informations liées aux clés et au cryptage sont encapsulées par la commande create encryption key, qui vous permet de spécifier l'algorithme de cryptage et la taille de la clé, la propriété par défaut de la clé, ainsi que le recours éventuel à un vecteur d'initialisation ou au remplissage au cours du processus de cryptage.

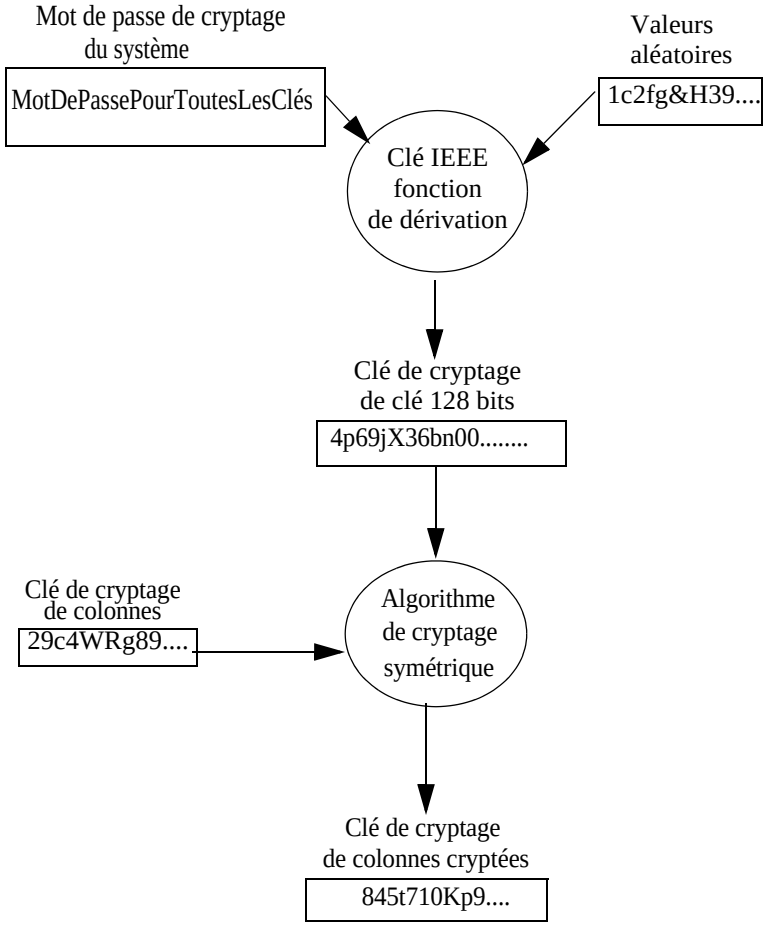
Dans Adaptive Server, le cryptage des colonnes utilise l'algorithme de cryptage de clé symétrique AES (Advanced Encryption Standard), disponible avec des clés d'une taille de 128, 192 et 256 bits. La génération de clé aléatoire et la fonctionnalité de cryptographie sont assurées par l'API Security Builder Crypto™.

Vous pouvez créer des clés distinctes pour chaque colonne cryptée. Les clés peuvent être partagées par plusieurs colonnes, mais chaque colonne ne peut être associée qu'à une seule clé. Pour crypter une colonne avec un vecteur d'initialisation et une autre sans celui-ci, créez deux clés distinctes, la première spécifiant l'utilisation d'un vecteur d'initialisation, l'autre n'en spécifiant aucun.

Le responsable de la sécurité du système utilise la clause as default dans la commande create encryption key pour définir une clé de cryptage par défaut pour la base de données. La clé par défaut est utilisée lorsque le qualifieur encrypt est employé sans nom de clé avec la commande create table ou alter table.

Pour sécuriser les valeurs de clé, Adaptive Server utilise le mot de passe de cryptage du système afin de générer une clé de cryptage de clé de 128 bits, servant elle-même à crypter la nouvelle clé. La clé de cryptage de colonnes est stockée sous une forme cryptée dans la table système *sysencryptkeys*.

Figure 2 : Cryptage des clés utilisateur



La syntaxe de la commande create encryption key est la suivante :

```
create encryption key nom_clé [as default] for algorithme
[with [keylength nbre_bits]
[init_vector [null | random]]
[pad [null | random]]]
```

où :

- *nom_clé* doit être unique dans la table de l'utilisateur, la vue et l'espace de nom de la procédure dans la base de données actuelle.

- as default permet au responsable de la sécurité du système de créer une clé de base de données par défaut pour le cryptage. Le créateur de la table peut ainsi spécifier le cryptage sans utiliser un nom de clé pour la commande create table, alter table et select into. Adaptive Server utilise la clé par défaut de la base de données. La clé par défaut peut être modifiée. Pour plus d'informations, reportez-vous à la section « alter encryption key », page 39.
- *algorithme* : Advanced Encryption Standard (AES) est le seul algorithme reconnu. AES prend en charge des longueurs de clé de 128, 192 et 256 bits et une taille de bloc de 16 octets.
- keylength *nbre_bits* : taille, en bits, de la clé à créer. Pour AES, les longueurs de clé valables sont de 128, 192 et 256 bits. La longueur de clé par défaut est de 128 bits.
- init_vector random spécifie l'utilisation d'un vecteur d'initialisation pendant le cryptage. Lorsqu'un vecteur d'initialisation est utilisé par l'algorithme de cryptage, le texte crypté de deux portions identiques de texte simple est différent, ce qui empêche un cryptologue de détecter des schémas de données. Le recours à un vecteur d'initialisation peut accroître la sécurité de vos données.

  Un vecteur d'initialisation a certaines implications en termes de performances. La création d'index ainsi que des jointures et des recherches optimisées ne peuvent être effectuées que sur une colonne dont la clé de cryptage ne spécifie pas de vecteur d'initialisation. Pour plus d'informations, reportez-vous à la section « Notes sur les performances », page 21.
- init_vector null empêche l'utilisation d'un vecteur d'initialisation lors du cryptage. Ceci permet à la colonne de prendre en charge un index.

  La valeur par défaut consiste à utiliser un vecteur d'initialisation, à savoir init_vector random. L'utilisation d'un vecteur implique l'application d'un mode de cryptage CBC (cipher block chaining) ; la définition de init_vector null suppose le mode ECB (electronic code book).
- pad null est la valeur par défaut. Elle empêche le remplissage aléatoire de données.

  Vous pouvez utiliser le remplissage si la colonne doit prévoir un index.

- pad random : les données sont automatiquement remplies avec des octets aléatoires avant le cryptage. Vous pouvez utiliser le remplissage au lieu d'un vecteur d'initialisation pour randomiser le texte chiffré. Le remplissage ne convient que pour les colonnes dont la longueur en texte simple est inférieure à la moitié de la longueur du bloc. Pour l'algorithme AES, la longueur du bloc est de 16 octets.

Par exemple, pour spécifier une clé de 256 bits appelée "safe_key" comme clé par défaut pour la base de données, le responsable de la sécurité du système entre :

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

L'exemple suivant crée une clé de 128 bits appelée "salary_index" pour crypter les colonnes à l'aide du remplissage :

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

L'exemple suivant crée une clé de 192 bits appelée "mykey" pour crypter les colonnes à l'aide d'un vecteur d'initialisation :

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

Le responsable de la sécurité du système détient par défaut l'autorisation de créer des clés de cryptage et peut accorder cette même autorisation à d'autres utilisateurs.

Par exemple :

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

1.3.2 Utilisation de clés de cryptage

Lorsque vous spécifiez une colonne de cryptage, vous pouvez utiliser une clé nommée de la base de données en cours ou d'une autre base de données. Si vous ne spécifiez pas de clé nommée, la colonne est automatiquement cryptée à l'aide de la clé par défaut de la base de données concernée.

Le cryptage à l'aide d'une clé provenant d'une autre base de données présente un net avantage en termes de sécurité, car il empêche de disposer à la fois des clés et des données cryptées en cas de vol d'une sauvegarde de base de données. Pour accéder aux données, il est nécessaire d'avoir accès à la fois à l'archive de la base de données contenant les données et à l'archive contenant les clés de cryptage. Les administrateurs peuvent également protéger les sauvegardes de base de données à l'aide de mots de passe différents, de manière à compliquer davantage tout accès non autorisé.

Le cryptage à l'aide d'une clé provenant d'une autre base de données exige une attention toute particulière, car il convient de prévenir tout problème d'intégrité des données et des clés dans les systèmes distribués. Veillez à coordonner soigneusement sauvegardes et restaurations de bases de données. Si vous utilisez une clé nommée issue d'une autre base de données, Sybase vous recommande de respecter les instructions suivantes :

- Lorsque vous sauvegardez la base de données contenant les colonnes cryptées, sauvegardez également la base de données dans laquelle la clé a été créée. Cette opération est nécessaire si de nouvelles clés ont été ajoutées depuis la dernière sauvegarde.
- Lorsque vous sauvegardez la base de données contenant une clé de cryptage, sauvegardez toutes les bases de données contenant des colonnes cryptées avec cette clé. Ceci garantit la synchronisation des données cryptées avec les clés disponibles.

Le responsable de la sécurité du système peut identifier toutes les colonnes cryptées avec une clé donnée à l'aide de la commande sp_encryption. Pour plus d'informations, reportez-vous à la section « sp_encryption », page 41.

1.3.3 Octroi d'autorisations sur les clés

Le propriétaire de la clé doit accorder l'autorisation select sur la clé avant qu'un autre utilisateur puisse spécifier la clé dans une instruction create table, alter table et select into. Le propriétaire de la clé par défaut de la base de données est le responsable de la sécurité du système. N'accordez l'autorisation select sur les clés qu'en fonction des besoins.

L'exemple suivant permet aux utilisateurs possédant un rôle db_admin_role d'utiliser la clé de cryptage "safe_key" lorsqu'ils spécifient le cryptage dans les instructions "create table" et "alter table" :

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

Les utilisateurs qui appliquent aux colonnes cryptées les commandes insert, update, delete et select ne doivent pas posséder l'autorisation select sur la clé de cryptage.

1.3.4 Modification des clés

Modifiez régulièrement les clés utilisées pour crypter les colonnes dans le cadre de votre stratégie de sécurité de l'information. Créez une nouvelle clé à l'aide de la commande create encryption key, puis utilisez alter table...modify pour crypter la colonne avec la nouvelle clé.

Dans l'exemple suivant, supposons que la colonne "creditcard" soit déjà cryptée. La commande alter table décrypte et recrypte la valeur "creditcard" pour chaque ligne de "customer" avec cc_key_new.

```
create encryption key cc_key_new for AES

alter table customer modify creditcard encrypt with
cc_key_new
```

Pour plus d'informations, reportez-vous à la section « alter table », page 42.

1.4 Cryptage des données

Vous pouvez crypter les types de données suivants :

- int, smallint, tinyint
- float4 et float8
- decimal et numeric
- char et varchar
- binary et varbinary

Le type sous-jacent des données cryptées sur disque est varbinary. Reportez-vous à la section « Longueur des colonnes cryptées », page 16 pour en savoir plus sur la longueur des données varbinary.

Les valeurs nulles ne sont pas cryptées.

1.4.1 Spécification du cryptage pour les nouvelles tables

Pour crypter les colonnes d'une nouvelle table, utilisez cette option de colonne dans l'instruction create table :

[encrypt [with [*base_de_données*.[*propriétaire*].]*nom_clé*]]

nom_clé identifie une clé créée à l'aide de la commande create encryption key. Le créateur de la table doit posséder l'autorisation select sur *nom_clé*. Si *nom_clé* n'est pas renseigné, Adaptive Server recherche une clé par défaut créée à l'aide de la clause as default de la commande create encryption key. Pour connaître la syntaxe complète de create table, reportez-vous à la section « Création d'une table », page 45.

L'exemple suivant crée deux clés : une clé de base de données par défaut, qui utilise les valeurs par défaut de init_vector, pad et keylength, et une clé nommée, cc_key, avec des valeurs différentes des valeurs par défaut. La colonne ssn de la table "employee" est cryptée à l'aide de la clé par défaut, et la colonne creditcard de la table "customer" est cryptée à l'aide de la clé cc_key:

```
create encryption key new_key as default for AES
create encryption key cc_key for AES with
          keylength 256
          init_vector null
          pad random

create table employee_table (ssn char(15) encrypt)

create table customer (creditcard char(20)
          encrypt with cc_key)
```

1.4.2 Création d'index pour les colonnes cryptées

Vous pouvez créer un index pour une colonne cryptée si la clé de cryptage a été spécifiée sans vecteur d'initialisation ni remplissage aléatoire. Une erreur se produit si vous exécutez la commande create index sur une colonne cryptée possédant un vecteur d'initialisation ou un remplissage aléatoire. Les index des colonnes cryptées sont utiles pour des recherches d'égalité et d'inégalité, mais pas pour les recherches séquentielles ou les agencements.

Dans l'exemple suivant, cc_key spécifie un cryptage sans recours à un vecteur d'initialisation ni à un remplissage. Ceci permet de générer un index sur toute colonne cryptée avec cc_key:

```
create encryption key cc_key for AES
          with init_vector null

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

create index cust_idx on customer(creditcard)
```

1.4.3 Cryptage de données dans des tables existantes

Pour crypter les colonnes d'une table existante, utilisez cette option de colonne dans l'instruction alter table :

[encrypt [with [*base_de_données*.[*propriétaire*].]*nom_clé*

nom_clé identifie une clé créée à l'aide de la commande create encryption key. Le créateur de la table doit posséder l'autorisation select sur *nom_clé*. Si *nom_clé* n'est pas renseigné, Adaptive Server recherche une clé par défaut créée à l'aide de la clause as default de la commande create encryption key. Pour connaître la syntaxe complète de alter table, reportez-vous au *Adaptive Server Enterprise – Manuel de référence* .

Remarque Si vous cryptez une colonne d'une table existante dans laquelle un trigger a été créé, la commande alter table échoue et renvoie un message d'erreur. Vous devez supprimer le trigger avant de modifier la table pour pouvoir la crypter. Régénérez le trigger après avoir exécuté la commande alter table .. encrypt.

Vous pouvez modifier la propriété de cryptage d'une colonne tout en modifiant d'autres attributs, tels que le type de données et le caractère NULL. Vous pouvez également ajouter une colonne cryptée en utilisant alter table.

Par exemple :

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

Reportez-vous à la section « alter table », page 44 pour connaître la syntaxe complète.

1.5 Décryptage des données

1.5.1 Autorisations de décryptage

Vous devez posséder ces deux autorisations pour sélectionner des données en texte clair dans une colonne cryptée ou pour effectuer une recherche ou une jointure sur une colonne cryptée :

- Autorisation select sur la colonne
- Autorisation decrypt sur la colonne utilisée dans la liste cible et dans where, having, order by, update et autres clauses du même type.

Le propriétaire de la table utilise la commande grant decrypt pour accorder, à d'autres utilisateurs, groupes et rôles, une autorisation explicite de décrypter une ou plusieurs colonnes d'une table. L'autorisation de cryptage peut être accordée implicitement lorsqu'un propriétaire de procédure ou de vue octroie :

- l'autorisation exec sur une procédure stockée qui effectue une sélection dans une colonne cryptée si le propriétaire de la procédure possède également la table contenant la colonne cryptée ;
- l'autorisation decrypt sur une colonne de la vue qui effectue une sélection dans une colonne cryptée si le propriétaire de la procédure possède également la vue contenant la colonne cryptée.

Dans les deux cas, il n'est pas nécessaire d'octroyer l'autorisation "decrypt" sur la colonne cryptée de la table sous-jacente.

La syntaxe est la suivante :

grant decrypt on [*propriétaire*.] *table*[(*colonne[{ ,colonne*}])] to *utilisateur | groupe | rôle*

L'octroi de l'autorisation "decrypt" au niveau de la table s'applique à toutes les colonnes cryptées de la table.

Pour octroyer l'autorisation "decrypt" sur toutes les colonnes cryptées de la table customer, entrez :

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

L'exemple suivant illustre l'autorisation "decrypt" implicite de user2 sur la colonne ssn de la table sous-jacente "employee". user1 configure la table "employee" et la vue employee_view de la manière suivante :

```
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 a accès aux numéros de sécurité sociale décryptés lorsqu'il effectue une sélection dans la vue emp_salary :

```
select * from emp_salary
```

1.5.2 Révocation des autorisations de décryptage

Vous pouvez révoquer l'autorisation de décryptage d'un utilisateur à l'aide de :

revoke decrypt on [*propriétaire*.] *table*[(*colonne*[{,*colonne*}])] from *utilisateur* | *groupe* | *rôle*

Par exemple :

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

1.6 Suppression du cryptage et des clés

1.6.1 Suppression du cryptage

Si vous êtes propriétaire d'une table, vous pouvez supprimer le cryptage d'une colonne en utilisant la commande alter table avec l'option decrypt.

La syntaxe est la suivante :

drop encrypton key [*base_de_données*.[*propriétaire*].]*nom_clé*

Par exemple, pour supprimer le cryptage de la colonne creditcard de la table customer, entrez :

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

1.6.2 Suppression de clés

Les clés peuvent être supprimées par leurs propriétaires et par le responsable de la sécurité du système. Cette procédure n'est possible que si aucune colonne cryptée de l'une des bases de données n'utilise la clé concernée. Vous ne pouvez pas vérifier la présence de colonnes cryptées à l'aide de la clé dans les bases de données suspectes ou hors ligne. La commande émet un message d'avertissement mentionnant la base de données indisponible, mais elle n'échoue pas. Lorsque la base de données est mise en ligne, les tables dont les colonnes ont été cryptées à l'aide de la clé supprimée ne sont plus utilisables. Pour restaurer la clé, l'administrateur système doit charger une sauvegarde de la base de données de la clé supprimée datant d'avant la suppression de la clé.

Pour supprimer une clé de cryptage, entrez :

drop encryption key [*base_de_données*.[*propriétaire*].]*nom_clé*

Par exemple :

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

1.7 Commande *select into*

Par défaut, la commande select into crée une table cible sans cryptage, même si la table source contient une ou plusieurs colonnes cryptées. La colonne select into exige des autorisations au niveau colonne, dont decrypt, sur la table source.

Pour crypter des colonnes dans la nouvelle table, entrez :

```
select [all|distinct] < liste_colonnes>
        into nom_table
        [(colname encrypt [with [[basededonnées.[propriétaire].]nom_clé]
                [, colname encrypt
                [with [[ basededonnées.]propriétaire].]nom_clé]])]
         from nom_table | nom_vue
```

Vous pouvez crypter une colonne spécifique de la table cible, même si ces données n'étaient pas cryptées dans la table source. Si la colonne de la table source est cryptée avec la même clé que celle spécifiée pour la colonne cible, Adaptive Server passe l'étape de décryptage de la table source et l'étape de cryptage de la table cible.

Les règles de cryptage dans une table cible sont identiques à celles de la spécification encrypt de create table pour la table source concernant :

- Types de données autorisés dans les colonnes à crypter
- Utilisation de la clé par défaut de la base de données en l'absence d'un nom de clé
- Nécessité de disposer de l'autorisation select sur la clé utilisée pour crypter les colonnes cibles

Par exemple, pour crypter la colonne creditcard :

```
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
```

1.8 Longueur des colonnes cryptées

Au cours des opérations create table et alter table, Adaptive Server calcule la longueur interne maximale de la colonne cryptée. Pour prendre des décisions quant aux organisations de schémas et aux tailles de pages, le propriétaire de la base de données doit connaître la longueur maximale des colonnes cryptées.

AES est un algorithme de chiffrage au niveau du bloc. La longueur des données cryptées pour ces algorithmes est un multiple de la taille du bloc correspondant. Pour AES, la taille du bloc est de 128 bits, soit 16 octets. Par conséquent, les colonnes cryptées occupent au minimum 16 octets, avec de l'espace supplémentaire pour :

- Le vecteur d'initialisation. S'il est utilisé, celui-ci requiert 16 octets supplémentaires par colonne cryptée. Par défaut, le processus de cryptage utilise un vecteur d'initialisation. Spécifiez init_vector null dans create encryption key pour omettre le vecteur d'initialisation.
- La longueur des données en texte simple. Si le type de colonne est char, varchar, binary ou varbinary, un préfixe de 2 octets est ajouté aux données avant le cryptage. Aucun espace supplémentaire n'est occupé par la colonne cryptée, à moins que les 2 octets supplémentaires de texte chiffré n'entraînent l'utilisation d'un bloc supplémentaire.
- Un octet "sentinelle", à savoir un octet joint au texte chiffré afin de le protéger contre la suppression des zéros en fin de chaîne par le système de base de données.

Tableau 1 : Longueurs du texte chiffré

Type de colonne défini par l'utilisateur	Longueur des données saisies	Vecteur d'initialisation	Type de colonne interne	Longueur des données cryptées
tinyint, smallint ou int	1, 2 ou 4	Non	varbinary(17)	17
tinyint, smallint ou int	1, 2 ou 4	Oui	varbinary(33)	33
tinyint, smallint ou int	0 (zéro)	Non	varbinary(17)	0
float, float(4), real	4	Non	varbinary(17)	17
float, float(4), real	4	Oui	varbinary(33)	33
float, float(4), real	0 (zéro)	Non	varbinary(17)	0
float(8), double	8	Non	varbinary(17)	17
float(8), double	8	Oui	varbinary(33)	33
float(8), double	0 (zéro)	Non	varbinary(17)	0
numeric(10,2)	3	Non	varbinary(17)	17
numeric (10,2)	3	Oui	varbinary(33)	33
numeric (38,2)	18	Non	varbinary(33)	33
numeric (38,2)	18	Oui	varbinary(49)	49
numeric (38,2)	0 (zéro)	Non	varbinary(33)	0
char, varchar (100)	1	Non	varbinary(113)	17
char, varchar(100)	14	Non	varbinary(113)	17
char, varchar(100)	15	Non	varbinary(113)	33
char, varchar(100)	15	Oui	varbinary(117)	49
char, varchar(100)	31	Oui	varbinary(117)	65
char, varchar(100)	0 (zéro)	Oui	varbinary(117)	0
binary, varbinary(100)	1	Non	varbinary(113)	17
binary, varbinary(100)	14	Non	varbinary(113)	17
binary, varbinary(100)	15	Non	varbinary(113)	33
binary, varbinary(100)	15	Oui	varbinary(117)	49
binary, varbinary(100)	31	Oui	varbinary(117)	65
binary, varbinary(100)	0 (zéro)	Oui	varbinary(117)	0

char et binary sont traités comme des types de données à longueur variable et sont vidés des éventuels blancs et remplissages nuls avant le cryptage. Les blancs et remplissages nuls sont appliqués lorsque les données sont décryptées.

Remarque La longueur des colonnes sur disque augmente pour les colonnes cryptées, mais ces accroissements sont invisibles. Par exemple, sp_help ne présente que la taille originale.

1.9 Audit des colonnes cryptées

Vous pouvez auditer les commandes DDL associées à des colonnes cryptées, telles que la création ou la suppression d'une clé de cryptage. Par ailleurs, lorsque vous créez une table, l'enregistrement d'audit contient le nom de la colonne cryptée et la clé de cryptage correspondante. Une option d'audit à l'échelle de la base de données vous permet de regrouper et de gérer les enregistrements d'audit des colonnes cryptées et des clés.

1.9.1 Options d'audit

La table suivante est extraite du Guide d'administration système d'Adaptive Server et illustre les nouvelles commandes auditées à l'aide des options d'événement existantes et nouvelles.

Tableau 2: Options d'audit, conditions et exemples

Options	*nom_login*	*nom_objet*	Base de données à partir de laquelle définir l'option	Commande ou accès à auditer
clé_de_cryptage (spécifique à la base)	all	Base de données à auditer	Quelconque	alter encryption key create encryption key drop encryption key create table drop table alter table sp_encryption
	Exemple `sp_audit "encryption_key", "all", "pubs2", "on"` (Audite toutes les commandes susmentionnées dans la base de données pubs2.)			

1.9.2 Valeurs d'audit

La Tableau 3 répertorie les valeurs qui apparaissent dans la colonne event, organisée par l'option sp_audit. La colonne "Informations dans extrainfo" décrit les données susceptibles de figurer dans la colonne extrainfo d'une table d'audit, en fonction des catégories mentionnées dans la Tableau 3.

Tableau 3 : Valeurs des colonnes event et extrainfo

Option d'audit	Commande à auditer	event	Informations dans la sortie de extrainfo

alter	alter table	3	*Mots-clés ou options :*
			`ADD/DROP/MODIFY COLUMNS`
			`REPLACE COLUMN`
			`ADD CONSTRAINT`
			`DROP CONSTRAINT`
			Si une ou plusieurs colonnes cryptées sont ajoutées, les *mots-clés* comprennent :
			`ADD/DROP/MODIFY COLUMNS column1/keyname1, [,column2/keyname2]`
			où *keyname* est le nom qualifié de la clé.

create	create table	10	Pour les colonnes cryptées, les mots-clés contiennent les noms de colonnes et de clés. `EK column1/keyname1[,column2 keyname2]` où EK est un préfixe indiquant que les informations suivantes font référence aux clés de cryptage et *keyname* est le nom qualifié de la clé.
clé_de_cryptage	sp_encryption	106	*Les mots-clés* comprennent `ENCR_ADMIN system_encr_passwd password` ******** si le mot de passe est défini pour la première fois et contient `ENCR_ADMIN system_encr_passwd password` ******** ******** si le mot de passe est modifié par la suite
	create encryption key	107	*Les mots de passe contiennent :* Nom algorithme-longueur en bits/IV [RANDOM|NULL]/PAD [RANDOM|NULL] Par exemple : `AES-128/IV RANDOM/PAD NULL`

	alter encryption key	108	*Les mots de passe contiennent :*
			`NOT DEFAULT`
			si la clé n'est plus la clé par défaut.
			`DEFAULT`
			si la clé devient la clé par défaut.

	drop encryption key	109	

1.9.3 Nouveaux noms et numéros d'événements

Vous pouvez rechercher des événements d'audit spécifiques dans la piste d'audit. Utilisez audit_event_name avec le paramètre *event id* .

audit_event_name(*event_id*)

Dans la Tableau 4, vous trouverez une liste des nouveaux numéros et noms d'événements.

Tableau 4 : Nouveaux numéros d'événements

Numéro d'événement	Sortie des noms d'événements
106	`Encrypted Column Administration`
107	`Create Encryption Key`
108	`Alter Encryption Key`
109	`Drop Encryption Key`

1.10 Notes sur les performances

Le cryptage utilise l'UC de manière intensive, ce qui risque de ralentir les performances de vos applications et de prolonger le délai d'exécution des commandes qui utilisent les colonnes cryptées. L'overhead dépend du nombre d'UC et de moteurs Adaptive Server, de la charge du système, du nombre de sessions concurrentes qui accèdent aux données cryptées et du nombre de colonnes cryptées référencées dans la requête. La taille de la clé de cryptage et la longueur des données cryptées jouent également un rôle. En général, plus la clé est grande et les données volumineuses, et plus l'UC est utilisée au cours du processus de cryptage.

Cette section aborde les conséquences sur les performances des recherches dans les colonnes cryptées, ainsi que la manière dont Adaptive Server Enterprise optimise le traitement des données cryptées afin de minimiser le nombre d'opérations de cryptage et de décryptage.

1.10.1 Index des colonnes cryptées

Vous pouvez créer un index pour une colonne cryptée, pour autant que la clé de cryptage de la colonne n'ait pas été spécifiée avec un vecteur d'initialisation ou un remplissage aléatoire. L'utilisation d'un vecteur d'initialisation ou d'un remplissage aléatoire résulte dans le cryptage de données identiques selon différents schémas de texte chiffré, ce qui empêche l'index d'imposer l'unicité requise.

Les index des données cryptées sont utiles pour des recherches d'égalité et d'inégalité, mais pas pour l'agencement des données, les recherches séquentielles ou la recherche de valeurs minimales et maximales. Si Adaptive Server effectue une recherche dépendant de l'ordre sur une colonne cryptée, il ne peut pas exécuter de recherche indexée sur les données cryptées. Au lieu de cela, la colonne cryptée doit être décryptée au niveau de chaque ligne et faire l'objet d'une recherche. Ce processus ralentit le traitement des données.

1.10.2 Jointures des colonnes cryptées

Adaptive Server optimise la jointure de deux colonnes cryptées en effectuant des comparaisons de texte chiffré si les conditions suivantes s'appliquent :

- Les colonnes à joindre ont le même type de données. char et varchar sont considérés comme identiques, tout comme binary et varbinary.
- Pour les types int et float, les colonnes ont la même longueur. Pour les types numeric et decimal, les colonnes ont la même précision et la même échelle.
- Les colonnes à joindre sont cryptées avec la même clé.
- Les colonnes à joindre n'appartiennent pas à une expression. Par exemple, il est impossible d'effectuer une jointure de texte chiffré si t.encr_col1 = s.encr_col1 +1.
- La clé de cryptage a été créée avec une valeur init_vector et pad de NULL.
- L'opérateur de jointure est "=" ou "<>".
- Les données présentent l'ordre de tri par défaut.

Par exemple, ceci définit un schéma de jointure pour du texte chiffré :

```
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........)
```

Vous pouvez également configurer des index sur les colonnes à joindre :

```
create index cust_cc on customer(creditcard)

create index daily_cc on daily_xacts(creditcard)
```

Adaptive Server exécute l'instruction "select" suivante pour calculer le total des débits quotidiens d'une carte de crédit donnée sans décrypter la colonne creditcard de la table customer ou daily_xacts.

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

1.10.3 Arguments de recherche à valeur constante et colonnes cryptées

Pour les comparaisons d'égalité et d'inégalité d'une colonne cryptée avec une valeur constante, Adaptive Server optimise le balayage de colonne en cryptant la valeur constante une seule fois, et non en décryptant la colonne cryptée dans chaque ligne de la table. Les restrictions énumérées dans la section « Jointures des colonnes cryptées », page 22 s'appliquent également.

Adaptive Server ne peut pas utiliser un index pour effectuer une recherche séquentielle. Il doit décrypter chaque ligne avant d'effectuer des comparaisons de données. Si une requête contient d'autres prédicats, Adaptive Server sélectionne l'ordre de jointure le plus efficace, faisant généralement passer les recherches dans les colonnes cryptées en dernier lieu, sur le jeu de données le plus petit.

Si votre requête comporte plus d'une recherche séquentielle sans index utile, écrivez la requête de manière à ce que la recherche séquentielle par rapport à la colonne cryptée soit la dernière. Par exemple, la requête suivante recherche les numéros de sécurité sociale des contribuables de Rhode Island avec des revenus de plus de 100 000 dollars. La recherche séquentielle de la colonne zipcode passe avant celle de la colonne cryptée "gross income" :

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

1.10.4 Transfert de données cryptées sous forme de texte chiffré

Dans la mesure du possible, Adaptive Server optimise la copie des données cryptées en copiant le texte chiffré, sans décrypter et recrypter les données. Ce principe s'applique à select into, au bulk copy et à la réplication.

1.11 Tables système

1.11.1 *syscolumns*

Dans la table système syscolumns, ces colonnes décrivent les propriétés du cryptage :

Champ	Type	Valeurs	Description
encrtype	int	null	Type de données sous forme cryptée
encrlen	int	null	Longueur des données cryptées
encrkeyid	int	null	id d'objet de la clé.
encrkeydb	varchar (30)	null	Nom de la base de données dans laquelle réside la clé de cryptage. NULL si la clé réside dans la même base de données que la colonne cryptée.
encrdate	datetime	null	Date de création, copiée depuis sysobjects.crdate.

1.11.2 *sysobjects*

sysobjects possède une entrée pour chaque clé du type EK (clé de cryptage).

Pour les références de clé entre plusieurs bases de données, syscolumns.encrdate correspond à sysobjects.crdate.

encrkeyid dans *sysencryptkeys* correspond à la colonne id dans *sysobjects*.

1.11.3 *sysencryptkeys*

Chaque clé créée dans une base de données, y compris la clé par défaut, possède une entrée dans le catalogue système spécifique à la base de données sysencryptkeys.

Tableau 5 : sysencryptkeys

Champ	Type	Description
id	int	ID de clé de cryptage
ekalgorithm	int	Algorithme de cryptage
type	smallint	Identifie le type de clé. Les valeurs sont EK_SYMMETRIC et EK_DEFAULT.
status	int	Informations d'état internes
eklen	smallint	Longueur de clé définie par l'utilisateur
value	varbinary(1282)	La valeur cryptée d'une clé contient un cryptage symétrique de la clé. Pour crypter les clés, Adaptive Server utilise AES avec une clé de 128 bits issue du mot de clé de cryptage du système.
uid	int null	Pas utilisé
eksalt	varbinary(20)	Contient un salt aléatoire utilisé pour valider le décryptage de la clé de cryptage.
ekpairid	int null	Pas utilisé
pwdate	datetime null	Pas utilisé
expdate	int null	Pas utilisé
ekpwdwarn	int null	Pas utilisé

1.12 Modifications de l'utilitaire *ddlgen*

ddlgen prend en charge la génération d'instructions DDL pour les clés cryptées. Pour spécifier une clé, utilisez :

<nombase>.<propriétaire>.<nomclé>

Le nouveau type EK (clé de cryptage) est destiné à générer le DDL servant à créer une clé de cryptage et à accorder des autorisations y afférentes. ddlgen génère des informations sur la colonne cryptée et une instruction grant decrypt, avec le DDL d'une table.

Cet exemple génère le DDL de toutes les clés cryptées d'une base de données "accounts" sur une machine appelée "HARBOR" en passant par le port 1955 :

```
ddlgen -Uroy -Proy123 -SHARBOR:1955 -TEK
-Naccounts.dbo.%
```

Vous pouvez également spécifier le nom de la base de données à l'aide de l'option -D :

```
ddlgen -Uroy -Proy134 -SHARBOR:1955 -TEK -Ndbo.%
  -Daccounts
```

```
------------------------------------------------------
-----------------------
-- DDL for EncryptedKey 'ssn_key'
------------------------------------------------------
-----------------------
      print 'ssn_key'

create encryption key accounts.dbo.ssn_key
      for AES
      with keylength 128
      init vector random
go


------------------------------------------------------
-----------------------
-- DDL for EncryptedKey 'ek1'
------------------------------------------------------
-----------------------
print 'ek1'

create encryption key accounts.dbo.ek1 as default
     for AES
     with keylength 192
     init vector NULL
go

use accounts
go

grant select on accounts.dbo.ek1 to acctmgr_role
go
```

ddlgen possède également une option étendue pour générer la commande create encryption key qui spécifie la valeur cryptée de la clé telle que représentée dans *sysencryptkeys*. Cette option s'appelle -XOD et elle peut être utilisée si vous devez synchroniser des clés de cryptage entre plusieurs serveurs dans l'optique d'un transfert de données. Par exemple, pour mettre la clé cc_key du serveur "PACIFIC" à disposition sur le serveur "ATLANTIC", exécutez ddlgen avec -XOD sur "PACIFIC", de la manière suivante :

```
ddlgen -Sfred -Pget2work -SPACIFIC:8532 -TEK -Nsales.dbo.cc_key -XOD
```

Résultat de ddlgen :

```
------------------------------------------------------------------------
-- DDL for EncryptedKey 'cc_key'
```

```
------------------------------------------------------------------------------
print 'cc_key'

create encryption key sales.dbo.cc_key
      for AES
with keylength 128
passwd 0x0000E1D8235FEBEB118901
init_vector NULL
keyvalue 0xF772B99CE547D2932A12E0A83F2114848BD93F38016C068D720DDEBAC4DF8AA001
keystatus 32
go
```

Ensuite, modifiez la clé générée par create encryption key pour spécifier la base de données cible sur "ATLANTIC" et exécuter la commande sur le serveur cible. cc_key est désormais disponible sur le serveur "ATLANTIC" pour décrypter les données transférées de "PACIFIC" vers "ATLANTIC".

Reportez-vous au guide *Utilitaires d'Adaptive Server Enterprise* pour obtenir de plus amples informations sur les options de syntaxe ddlgen et au *Guide d'administration de Replication Server* pour des exemples d'utilisation de ddlgen avec des bases de données répliquées.

1.13 Réplication des données cryptées

Si votre site réplique les modifications de schémas, les instructions DDL suivantes sont répliquées :

- alter encryption key
- create table et alter table avec des extensions pour le cryptage
- create encryption key
- grant et revoke create encryption key
- grant et revoke select sur la clé
- grant et revoke decrypt sur la colonne
- sp_encryption system_encr_passwd
- drop encryption key

Les clés sont répliquées sous une forme cryptée.

Si votre système ne réplique pas le DDL, vous devez synchroniser manuellement les clés de cryptage sur le site répliqué. ddlgen prend en charge une forme spéciale de create encryption key pour répliquer la valeur de la clé.

Insérez (commande insert) et mettez à jour (commande update) les colonnes cryptées répliquées sous forme cryptée. De cette manière, les données répliquées seront protégées pendant que Replication Server les traite dans des files d'attente permanentes sur disque.

Reportez-vous au *Guide d'administration de Replication Server* pour de plus amples informations sur le cryptage en cours de réplication.

1.14 Bulk copy (*bcp*)

bcp transfère des données cryptées à destination et au départ de bases de données sous la forme de texte simple ou de texte chiffré. Par défaut, bcp copie les données en texte simple. bcp traite les fichiers de données en texte simple de la manière suivante :

- Les données sont automatiquement cryptées par Adaptive Server avant l'insertion lorsque vous exécutez bcp in. Une commande bcp lente est utilisée. L'utilisateur doit détenir les autorisations insert et select sur toutes les colonnes.
- Les données sont automatiquement décryptées par Adaptive Server lorsque vous exécutez bcp out. L'autorisation select est nécessaire sur toutes les colonnes. En outre, l'autorisation decrypt doit être octroyée pour les colonnes cryptées.

Cet exemple copie la table "customer" sous la forme de données en texte simple au format machine natif :

```
bcp uksales.dbo.customer out uk_customers -n -Uroy
-Proy123
```

Utilisez l'option -C de bcp pour copier les données sous la forme de texte chiffré. Dans ce dernier cas, vous pouvez copier des données de plusieurs systèmes d'exploitation différents. Si vous copiez des données de type caractères sous la forme de texte chiffré, les deux plates-formes doivent prendre en charge le même jeu de caractères.

L'option -C de bcp permet aux administrateurs d'exécuter bcp lorsqu'ils ne disposent pas de l'autorisation decrypt sur les données. Lorsque l'option -C est utilisée, bcp traite les données comme suit :

- Les données sont supposées être au format texte chiffré pendant l'exécution de la commande bcp in, et Adaptive Server n'effectue aucun cryptage. N'utilisez l'option -C avec bcp in que si le fichier copié dans Adaptive Server a été créé avec l'option -C de bcp out. Le texte chiffré doit avoir été copié d'une colonne possédant exactement les mêmes attributs et la même clé de cryptage que la colonne vers laquelle les données sont copiées. Une commande bcp rapide est utilisée. L'utilisateur doit détenir les autorisations insert et select.
- Les données sont copiées d'Adaptive Server sans décryptage lors de l'exécution de bcp out. Les données en texte chiffré se trouvent au format hexadécimal. L'utilisateur doit détenir l'autorisation select sur toutes les colonnes. Pour copier du texte chiffré, decrypt n'est pas nécessaire pour les colonnes cryptées.
- Les données char ou varchar cryptées conservent le jeu de caractères utilisé par Adaptive Server au moment du cryptage. Si les données sont copiées vers un autre serveur sous la forme de texte chiffré, le jeu de caractères utilisé sur le serveur cible doit correspondre à celui des données cryptées copiées de la source. Le jeu de caractères associé aux données du serveur source au moment du cryptage n'est pas stocké avec les données cryptées et n'est ni connu ni converti sur le serveur de données.

  L'administrateur système doit vérifier que les jeux de caractères sont identiques sur les serveurs source et cible. Vous pouvez également exécuter la commande bcp sans l'option -C, de manière à éviter tout problème lié aux jeux de caractères.

  L'option -J pour la conversion des jeux de caractères ne peut pas être utilisée avec l'option -C.

L'exemple suivant copie la table "customer". La colonne cc_card est copiée sous la forme de texte chiffré. Les autres colonnes sont copiées sous la forme de caractères. L'utilisateur "roy" ne doit pas nécessairement disposer de l'autorisation "decrypt" sur customer cc_card.

```
 bcp uksales.dbo.customer out uk_customers -C -c -Uroy
-Proy123
```

Avertissement ! Vous pouvez utiliser l'indicateur -C avec bcp out sur une vue pour autant que la qualification de la vue ne recherche pas des colonnes cryptées.

1.15 Component Integration Services (CIS)

Par défaut, le cryptage et le décryptage sont gérés par le serveur Adaptive Server distant. CIS effectue une recherche unique de colonnes cryptées sur le serveur Adaptive Server distant. Si le serveur Adaptive Server distant prend en charge le cryptage, CIS met à jour le catalogue *syscolumns* local en y reprenant les métadonnées associées aux colonnes cryptées.

- create proxy_table met automatiquement à jour *syscolumns* avec toute information relative aux colonnes cryptées émanant des tables distantes.
- create existing table met automatiquement à jour *syscolumns* en y incluant les éventuelles métadonnées associées aux colonnes cryptées émanant des tables distantes. Le mot-clé "encrypt" n'est pas autorisé dans *columnlist* pour la commande create existing table. CIS marque les colonnes automatiquement comme cryptées s'il détecte des colonnes cryptées dans la table distante.
- La commande create table à l'emplacement avec des colonnes cryptées n'est pas autorisée.
- La commande alter table n'est pas autorisée sur des colonnes cryptées pour les tables proxy.
- select into existing extrait le texte simple de la source et l'insère dans la table de destination. Dans ce cas, le serveur Adaptive Server local crypte le texte simple avant de l'insérer dans une colonne cryptée.

Les colonnes suivantes sont mises à jour depuis le catalogue *syscolumns* du serveur distant :

- encrtype : type de données sur le disque.
- encrlen : longueur des données cryptées.
- status2 : bits d'état indiquant que la colonne est cryptée.

1.16 Chargement et sauvegarde de bases de données (*load* et *dump*)

dump et load agissent sur le texte chiffré des colonnes cryptées. Ce comportement garantit que les données des colonnes cryptées restent cryptées lorsqu'elles se trouvent sur disque. dump et load s'appliquent à la base de données dans son ensemble. Les clés par défaut et clés créées dans la même base de données sont sauvegardées et chargées en même temps que les données y afférentes.

Si la base de données en chargement contient des clés de cryptage utilisées dans d'autres bases de données, la commande load échoue, à moins d'utiliser la nouvelle syntaxe with override .

```
load database key_db from "/tmp/key_db.dat" with override
```

Si vos clés se trouvent dans une base de données autre que celle des colonnes cryptées, Sybase vous recommande de suivre les instructions suivantes :

- Lorsque vous sauvegardez la base de données contenant les colonnes cryptées, sauvegardez également la base de données dans laquelle la clé a été créée. Cette opération est nécessaire si de nouvelles clés ont été ajoutées depuis la dernière sauvegarde.
- Lorsque vous sauvegardez la base de données contenant une clé de cryptage, sauvegardez toutes les bases de données contenant des colonnes cryptées avec cette clé. Ceci garantit la synchronisation des données cryptées avec les clés disponibles.
- Après avoir chargé la base de données contenant les clés de cryptage et celle contenant les colonnes cryptées, mettez-les toutes deux en ligne simultanément.

Si vous chargez la base de données contenant les clés dans une base de données portant un nom différent, des erreurs se produiront lorsque vous accéderez aux colonnes cryptées des autres bases de données. Pour modifier le nom de la base de données des clés, procédez comme suit :

- Avant de sauvegarder la base de données contenant les colonnes cryptées, utilisez la commande alter table pour décrypter les données.
- Sauvegardez les bases de données contenant les clés et les colonnes cryptées.
- Après avoir chargé les bases de données, utilisez la commande alter table pour recrypter les données avec les clés de la base de données renommée.

Les problèmes d'intégrité entre les clés de cryptage et les colonnes cryptées sont analogues à ceux qui se posent pour l'intégrité référentielle entre plusieurs bases de données. Reportez-vous à la section "Contraintes entre bases de données et chargement de bases de données" dans le *Guide d'administration système d'Adaptive Server Enterprise*.

Ne tentez pas de charger des sauvegardes contenant des données cryptées dans des versions antérieures d'Adaptive Server. Chargez la base de données dans un serveur Adaptive Server version 12.5.3a et supprimez-en tout cryptage. Effectuez une sauvegarde, puis chargez-la dans un serveur Adaptive Server antérieur. Pour plus d'informations, reportez-vous à la section « Procédure de mise à niveau descendante », page 35.

Pour plus d'informations sur les clés, reportez-vous à la section « Création et gestion des clés de cryptage », page 5.

1.17 *unmount database*

Lorsque des colonnes sont cryptées à l'aide de clés provenant d'autres bases de données, démontez toutes les bases de données associées comme s'il s'agissait d'un ensemble. L'interdépendance des bases de données contenant les colonnes cryptées et des bases de données contenant les clés est analogue à celle qui existe entre plusieurs bases de données utilisant l'intégrité référentielle.

Utilisez l'option override pour démonter (unmount) une base de données contenant des colonnes cryptées par une clé provenant d'une autre base de données.

Grâce aux commandes suivantes, la clé de cryptage créée dans key_db a été utilisée pour crypter les colonnes de col_db. Ces commandes démontent avec succès la base de données nommée :

```
unmount database key_db, col_db
unmount database key_db with override
unmount database col_db with override
```

Ces commandes échouent et renvoient un message d'erreur si elles sont exécutées sans l'option override :

```
unmount database key_db

unmount database col_db
```

1.18 *quiesce database*

Vous pouvez utiliser quiesce database lorsque la base de données contient la clé de cryptage.

Vous devez utiliser l'option with override pour mettre en sommeil (quiesce) une base de données contenant des colonnes cryptées par des clés provenant d'autres bases de données.

`quiesce database` ***`key_db, col_db`*** est autorisé. Ici, *key_db* est la base de données contenant la clé de cryptage et *col_db* la base de données contenant une colonne cryptée à l'aide de la clé de *key_db*.

Par exemple, les commandes suivantes s'exécutent avec succès. Ici, key_db contient la clé de cryptage utilisée pour crypter des colonnes de col_db :

```
quiesce database key_tag hold key_db for external
         dump to "/tmp/keydb.dat"

quiesce database encr_tag hold col_db for external dump
      to "/tmp/col.dat" with overridewith override

quiesce database col_tag hold key_db, col_db for
    external dump to "/tmp/col.dat"
```

1.19 Suppression des bases de données

Pour éviter toute perte accidentelle de clés, Adaptive Server met un terme à la commande drop database si elle contient des clés actuellement utilisées pour crypter des colonnes dans d'autres bases de données. Avant de supprimer la base de données contenant les clés de cryptage, vous devez d'abord supprimer le cryptage ou la base de données contenant les colonnes cryptées.

Dans l'exemple suivant, key_db est la base de données dans laquelle réside la clé de cryptage et col_db celle qui contient les colonnes cryptées :

```
drop database key_db, col_db
```

Adaptive Server génère une erreur et annule la suppression de key_db. La suppression de col_db réussit. Pour supprimer les deux bases de données, commencez par supprimer col_db :

```
drop database col_db, key_db
```

1.20 *sybmigrate*

sybmigrate est l'outil de migration utilisé pour transférer des données d'un serveur à un autre.

Par défaut, sybmigrate assure la migration des colonnes cryptées sous la forme de texte chiffré. Ceci évite l'overhead que supposent le décryptage des données à la source et leur cryptage à la cible. Dans certains cas, sybmigrate choisit la méthode de migration reencrypt, décryptant ainsi les données à la source et les cryptant à la cible.

Pour les bases de données comportant des colonnes cryptées, sybmigrate :

1 Transfère le mot de passe de cryptage du système. Si vous choisissez de ne pas transférer le mot de passe de cryptage du système, sybmigrate assure la migration des colonnes cryptées à l'aide de la méthode reencrypt au lieu de transférer directement le texte chiffré.

2 Transfère les clés de cryptage. Vous pouvez sélectionner les clés devant migrer. sybmigrate sélectionne automatiquement les clés de la base de données en cours utilisées pour crypter les colonnes de la même base de données. Si vous avez choisi de transférer le mot de passe de cryptage du système, sybmigrate assure la migration des clés de cryptage en utilisant leurs valeurs réelles. Les valeurs des clés émanant de la table système *sysencryptkeys* ont été cryptées à l'aide du mot de passe de cryptage du système, et ce sont ces valeurs qui migrent. Si vous n'avez pas transféré le mot de passe de cryptage du système, sybmigrate assure la migration des clés en fonction de leur nom, de manière à éviter de transférer des clés qui ne seraient pas décryptées correctement sur la cible. Dans ce cas, la clé créée au niveau de la cible possède une autre valeur que celle présente au niveau de la source.

3 Transfère les données. Par défaut, les données sont transférées sous forme de texte chiffré. Ces données peuvent migrer vers un autre système d'exploitation. Les données de type caractères exigent que le serveur cible utilise le même jeu de caractères que le serveur source.

sybmigrate utilise la base de données comme unité de travail. Si la base de données présente sur le serveur source comporte des données cryptées par une clé d'une autre base de données, commencez par transférer la base de données de la clé.

sybmigrate choisit de recrypter les données migrées dans les situations suivantes :

- L'une des clés de la base de données en cours n'est spécifiquement pas sélectionnée aux fins de la migration ou existe déjà sur le serveur cible. Il n'existe aucune garantie de ce que les clés de la cible sont identiques à celles de la source. Par conséquent, les données migrées doivent être recryptées.
- Le mot de passe système n'a pas été sélectionné aux fins de la migration. Lorsque le mot de passe système de la cible est différent de celui de la source, il est impossible de transférer les clés en fonction de leur valeur. De ce fait, les données ne peuvent pas migrer sous la forme de texte chiffré.

- L'utilisateur emploie l'indicateur suivant :

```
sybmigrate -T 'ALWAYS_REENCRYPT'
```

Le recryptage des données ralentit les performances. Un message dans ce sens s'inscrit dans le fichier journal de la migration lorsque vous effectuez le transfert avec un mode de recryptage.

Pour migrer des colonnes cryptées, vous devez posséder à la fois les rôles sa_role et sso_role.

1.21 Procédure de mise à niveau descendante

Si vous n'avez jamais configuré enable encrypted columns sur votre serveur, il n'est pas nécessaire de prendre une quelconque mesure avant d'utiliser une version antérieure d'Adaptive Server avec des bases de données 12.5.3a. Une manière de vous assurer que vous n'avez jamais configuré de colonnes cryptées consiste à vérifier que la table système *sysencryptkeys* n'existe pas dans l'une des bases de données.

Nous vous conseillons vivement de sauvegarder toutes les bases de données avant de procéder à la mise à niveau descendante.

Avant d'effectuer une mise à niveau descendante sur un serveur configuré pour les colonnes cryptées, vous devez supprimer ou modifier toute table comportant des colonnes cryptées, de manière à désactiver le cryptage. Ensuite, exécutez la commande sp_encryption remove_catalog, qui vérifie l'absence de colonnes cryptées dans chaque base de données, puis supprimez la table système *sysencryptkeys*. Les nouvelles colonnes ajoutées dans *syscolumns* pour 12.5.3a sont ignorées par les binaires antérieurs et doivent être supprimées.

Pour revenir d'un serveur 12.5.3a à une version antérieure de 12.5.x :

1 Si les colonnes cryptées ne sont actuellement pas activées, le responsable de la sécurité du système exécute :

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

2 Utilisez drop ou alter pour décrypter toutes les tables comportant des colonnes cryptées dans toutes les bases de données. Le responsable de la sécurité du système exécute la commande suivante dans chaque base de données dans laquelle des clés de cryptage ont été créées afin de lister toutes les clés de cryptage générées dans cette base de données :

```
sp_encryption help
```

Pour chaque clé listée, le responsable de la sécurité du système exécute les commandes suivantes pour visualiser une liste des colonnes cryptées avec une clé donnée :

```
sp_encryption help, <keyname>, 'display_cols'
```

Pour chaque colonne cryptée, l'une des étapes suivantes doit être exécutée :

- alter table pour décrypter les colonnes cryptées
- alter table pour supprimer les colonnes cryptées
- drop pour supprimer la table contenant les colonnes cryptées
- drop the encryption key

3 Redémarrez le serveur en mode "single user" pour vous assurer qu'aucun autre utilisateur ne pourra y accéder pendant la suppression d'une table système. Reportez-vous au guide *Utilitaires d'Adaptive Server Enterprise* .

4 Un utilisateur avec un rôle sso_role et sa_role doit exécuter la procédure stockée système suivante, qui supprime le catalogue *sysencryptkeys* de chaque base de données :

```
sp_encryption remove_catalog
```

Si une base de données n'est pas disponible, la commande renvoie une erreur et se ferme. S'il existe des colonnes cryptées par l'une des clés de *sysencryptkeys*, la commande ne supprime pas *sysencryptkeys*, mais émet un message d'erreur ou d'avertissement et passe à la base de données suivante.

Si sp_encryption réussit à supprimer *sysencryptkeys*, il élimine aussi ces lignes de *sysattributes* dans chaque base de données :

- l'enregistrement de l'élément de mise à niveau qui a ajouté *sysencryptkeys* ;
- le mot de passe de cryptage du système pour la base de données.

5 Supprimez la procédure stockée système sp_encryption de la base de données *sybsystemprocs*.

6 Éteignez le serveur. Vous pouvez à présent utiliser un binaire d'Adaptive Server 12.5.x d'une version antérieure à 12.5.3a.

Pour réactiver les colonnes cryptées, lorsque vous revenez à la version 12.5.3a depuis un serveur 12.5.3a passé à une version antérieure, configurez enable encrypted columns. Lors du redémarrage du serveur 12.5.3a, la table système *sysencryptkeys* est installée dans chaque base de données.

1.21.1 Problèmes de réplication avec la mise à niveau descendante

Lorsque vous effectuez une mise à niveau descendante sur un serveur sur lequel la réplication est activée pour les bases de données contenant des données cryptées, vous devez effectuer l'une des opérations suivantes avant de lancer la procédure de mise à niveau descendante :

1 Assurez-vous que toutes les données répliquées dans le journal de transactions de la base de données primaire ont été transférées avec succès vers la base de données en veille ou répliquée. Le processus permettant de le faire dépend de l'application.

2 Tronquez le journal de transactions dans la base de données primaire et remettez à zéro le releveur de coordonnées de RS pour cette base de données dans Replication Server. Utilisez les commandes suivantes :

Dans la base de données primaire, exécutez :

```
sp_stop_rep_agent primary_dbname
       dbcc settrunc ('ltm', 'ignore')
       dump tran  primary_dbname with truncate_only
       dbcc settruc ('ltm', 'valid')
```

Quittez Replication Server. Dans le RSSD pour Replication Server, exécutez :

```
rs_zeroltm primary_servername, primary_dbname
```

1.22 Nouvelles commandes

1.22.1 *create encryption key*

Toutes les informations associées aux clés et au cryptage sont encapsulées par la commande create encryption key, qui vous permet de spécifier l'algorithme de cryptage et la taille de la clé, la propriété par défaut de la clé, ainsi que le recours éventuel à un vecteur d'initialisation ou au remplissage au cours du processus de cryptage.

Adaptive Server utilise l'API Security Builder Crypto™ pour générer et crypter des clés.

Le responsable de la sécurité du système détient par défaut l'autorisation de créer des clés de cryptage et peut accorder cette même autorisation à d'autres utilisateurs.

Syntaxe

create encryption key [[*basededonnées.[propriétaire].]nomclé* [as default] for *algorithme*
[with [keylength *nbre_bits*]
[init_vector [NULL | random]]
[pad [NULL | random]]]

- *nomclé* doit être unique dans la table de l'utilisateur, la vue et l'espace de nom de la procédure dans la base de données actuelle.
- as default permet au responsable de la sécurité du système de créer une clé de base de données par défaut pour le cryptage. Le créateur de la table peut ainsi spécifier le cryptage sans utiliser un nom de clé pour la commande create table, alter table et select into. Adaptive Server utilise la clé par défaut de la base de données. La clé par défaut peut être modifiée. Pour plus d'informations, reportez-vous à la section « alter encryption key », page 39.
- *algorithme* : Advanced Encryption Standard (AES) est le seul algorithme reconnu. AES prend en charge des longueurs de clé de 128, 192 et 256 bits et une taille de bloc de 16 octets.
- keylength *nbre_bits* : taille, en bits, de la clé à créer. Pour AES, les longueurs de clé valables sont de 128, 192 et 256 bits. La longueur de clé par défaut est de 128 bits.
- init_vector random spécifie l'utilisation d'un vecteur d'initialisation pendant le cryptage. Lorsqu'un vecteur d'initialisation est utilisé par l'algorithme de cryptage, le texte crypté de deux portions identiques de texte simple est différent, ce qui empêche un cryptologue de détecter des schémas de données. Le recours à un vecteur d'initialisation peut accroître la sécurité de vos données.

  Un vecteur d'initialisation a certaines conséquences en termes de performances. La création d'index ainsi que des jointures et des recherches optimisées ne peuvent être effectuées que sur une colonne dont la clé de cryptage ne spécifie pas de vecteur d'initialisation. Pour plus d'informations, reportez-vous à la section « Notes sur les performances », page 21.
- init_vector null empêche l'utilisation d'un vecteur d'initialisation lors du cryptage. Ceci permet à la colonne de prendre en charge un index.

La valeur par défaut consiste à utiliser un vecteur d'initialisation, à savoir init_vector random. L'utilisation d'un vecteur implique l'application d'un mode de cryptage CBC (cipher block chaining) ; la définition de init_vector null suppose le mode ECB (electronic code book).

- pad null est la valeur par défaut. Elle empêche le remplissage aléatoire de données.

  Vous pouvez utiliser le remplissage si la colonne doit prévoir un index.

- pad random : les données sont automatiquement remplies avec des octets aléatoires avant le cryptage. Vous pouvez utiliser le remplissage au lieu d'un vecteur d'initialisation pour randomiser le texte chiffré. Le remplissage ne convient que pour les colonnes dont la longueur en texte simple est inférieure à la moitié de la longueur du bloc. Pour l'algorithme AES, la longueur du bloc est de 16 octets.

Par exemple, pour spécifier une clé de 256 bits appelée "safe_key" comme clé par défaut pour la base de données, le responsable de la sécurité du système entre :

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

L'exemple suivant crée une clé de 128 bits appelée "salary_index" pour crypter les colonnes à l'aide du remplissage :

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

L'exemple suivant crée une clé de 192 bits appelée "mykey" pour crypter les colonnes à l'aide d'un vecteur d'initialisation :

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

1.22.2 *alter encryption key*

Pour modifier la clé de cryptage par défaut, entrez :

alter encryption key *clé1* as default

Si une clé par défaut existe déjà, elle n'a plus la propriété par défaut. *clé* devient la clé par défaut.

Si *clé1* est la clé par défaut, vous pouvez supprimer la désignation par défaut pour *clé1*, comme suit :

alter encryption key *clé1* as not default

Si *clé1* n'est pas la clé par défaut, la commande renvoie une erreur.

alter encryption key as default ou not default ne peuvent être exécutés que par le responsable de la sécurité du système et ne peuvent pas être octroyés à d'autres utilisateurs.

1.22.3 *drop encryption key*

Le propriétaire des clés et le responsable de la sécurité du système peuvent supprimer les clés de cryptage. La commande échoue si une colonne d'une base de données est cryptée à l'aide de la clé.

Syntaxe drop encryption key [*basededonnées*.[*propriétaire*].]*nom_clé*

1.22.4 *grant create encryption key*

Le responsable de la sécurité du système accorde l'autorisation de créer des clés de cryptage.

Syntaxe grant create encryption key to *utilisateur* | *rôle* | *groupe*

1.22.5 *revoke create encryption key*

Le responsable de la sécurité du système peut révoquer l'autorisation d'autres utilisateurs, groupes et rôles de créer des clés de cryptage.

Syntaxe revoke create encryption key from *utilisateur* | *rôle* | *groupe*

1.22.6 *grant decrypt*

Le propriétaire de la table ou le responsable de la sécurité du système accorde l'autorisation decrypt sur une table ou une liste de colonnes dans une table.

Syntaxe grant decrypt on [*owner*.]*nom_table*[(*nom_colonne* [{,*nom_colonne*}])]
to *utilisateur* | *groupe* | *rôle*

Remarque grant all sur une table ou une colonne n'accorde pas d'autorisation "decrypt".

1.22.7 *revoke decrypt*

Le propriétaire de la table ou le responsable de la sécurité du système révoque l'autorisation decrypt sur une table ou une liste de colonnes dans une table.

Syntaxe revoke decrypt on [owner.] *nom_table*[(*nom_table* [{,*nom_table*}])] from *utilisateur* | *groupe* | *rôle*

1.23 *sp_encryption*

Le responsable de la sécurité du système définit le mot de passe de cryptage du système à l'aide de sp_encryption. Le mot de passe système est spécifique à la base de données où sp_encryption est exécuté, et sa valeur cryptée est stockée dans la table système sysattributes de cette base de données.

sp_encryption system_encr_passwd, '*mot_de_passe'*

Le mot de passe spécifié à l'aide de sp_encryption peut avoir une longueur maximale de 64 octets, et il est utilisé par Adaptive Server pour crypter toutes les clés de cette base de données. Vous ne devez pas spécifier ce mot de passe pour accéder aux clés ou aux données.

Le mot de passe de cryptage du système doit être défini dans toutes les bases de données où des clés de cryptage sont créées.

Le responsable de la sécurité du système peut modifier le mot de passe système à l'aide de la commande sp_encryption et en fournissant l'ancien mot de passe.

sp_encryption system_encr_passwd, *'mot_de_passe* '[, *'ancien_mot_de_passe*']

Lorsque le mot de passe système change, Adaptive Server recrypte automatiquement toutes les clés de la base de données avec le nouveau mot de passe.

1.23.1 *sp_encryption help*

sp_encryption help affiche le nom, le propriétaire, la taille et l'algorithme de cryptage de la clé. Il indique également si la clé a été désignée en tant que clé par défaut de la base de données et si le cryptage avec cette clé utilise un remplissage aléatoire ou un vecteur d'initialisation.

sp_encryption help [, *nom_clé* [, display_cols]]

Lorsque la commande sp_encryption help est exécutée par un utilisateur avec le rôle sso_role, les propriétés de toutes les clés de la base de données s'affichent. Lorsqu'elle est exécutée par un utilisateur sans le rôle sso_role, les propriétés clés ne s'affichent que pour les clés pour lesquelles l'utilisateur dispose de l'autorisation select dans cette base de données.

sp_encryption help, *nom_clé* affiche les propriétés de "nom_clé". Si la commande est exécutée par un utilisateur sans le rôle sso_role, l'utilisateur doit disposer de l'autorisation "select" sur la clé.

sp_encryption help, nom_clé, display_cols ne peut être exécuté que par un utilisateur avec un rôle sso_role. Cette commande liste les colonnes cryptées par "nom_clé".

1.24 Modifications apportées à la syntaxe des commandes

L'ajout de la fonctionnalité "Colonnes cryptées" a modifié la syntaxe des commandes de cette section ou y a ajouté des éléments.

1.24.1 alter table

Utilisez alter table pour crypter ou décrypter des données existantes ou ajouter une colonne cryptée à une table.

Syntaxe

Pour crypter une colonne :

alter table *nom_table* add *nom_colonne*
encrypt [with [*basededonnées*.[*propriétaire*].]*nom_clé*

Pour décrypter une colonne existante :

[decrypt [with [*basededonnées*.[*propriétaire*].]*nom_clé*]]

nom_clé identifie une clé créée à l'aide de la commande create encryption key. Le créateur de la table doit posséder l'autorisation select sur *nom_clé*. Si *nom_clé* n'est pas défini, Adaptive Server recherche une clé par défaut créée à l'aide de create encryption key ou alter encryption key.

Exemple

Pour créer une clé de cryptage et crypter la colonne ssn dans la table "employee" existante.

```
alter table employee modify ssn
         encrypt with ssn_key
grant decrypt on employee(ssn) to hr_manager_role,
         hr_director_role
```

Utilisez alter table pour modifier une clé de cryptage. Lorsque le qualifieur encrypt est utilisé sur une colonne déjà cryptée, Adaptive Server décrypte la colonne et la recrypte avec la nouvelle clé. Cette opération prend un temps considérable si la table contient un grand nombre de lignes.

1.24.2 Création d'une table

Utilisez le qualifieur encrypt pour configurer le cryptage sur une colonne de table.

Syntaxe

create table *nom_table* (*colname* datatype [default_clause]
[encrypt [with [*basededonnées*.[*propriétaire*].]*nom_clé*]])

nom_clé identifie une clé créée à l'aide de la commande create encryption key. Le créateur de la table doit posséder l'autorisation select sur *nom_clé*. Si *nom_clé* n'est pas mentionné, Adaptive Server recherche une clé par défaut créée à l'aide de la clause as default de la commande create encryption key ou alter encryption key.

Exemple

Pour créer une table "employee" avec cryptage :

```
create table employee_table (ssn char(15) null encrypt)
```

1.24.3 Paramètre de configuration *enable encrypted columns*

Le paramètre de configuration enable encrypted columns doit avoir la valeur 1 pour utiliser la fonctionnalité de cryptage. Si vous désactivez cette option de configuration dans un serveur qui contient des colonnes cryptées, toute commande appliquée à ces colonnes échoue avec un message d'erreur. Tant le paramètre de configuration que l'option de licence sont nécessaires pour utiliser le cryptage.

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

1.24.4 *load database*

Si la base de données en chargement contient des clés de cryptage utilisées dans d'autres bases de données, la commande load échoue, à moins d'utiliser with override .

```
load database key_db from "/tmp/key_db.dat" with override
```

1.24.5 *select into*

select into requiert des autorisations au niveau colonne, dont decrypt, sur la table source.

Syntaxe

Pour crypter des colonnes dans la nouvelle table, utilisez cette syntaxe :

```
select [all|distinct] < liste_colonnes>
        into table_cible
        [(colname encrypt [with [basededonnées.[propriétaire].]nom_clé]
        [,colname encrypt
        [with [basededonnées.[propriétaire].]nom_clé])]
    from nomtab | nomvue
```

Exemple

Cryptage de la colonne creditcard dans la table.

```
select creditcard, custid, sum(amount) into
       #bigspenders
       (creditcard encrypt with
```

```
    cust.database.new_cc_key)
from daily_xacts group by creditcard
having sum(amount) > $5000
```

1.24.6 *dbcc*

dbcc checkcatalog inclut les contrôles d'homogénéité supplémentaires suivants :

1 Pour chaque ligne de clé de cryptage dans *sysobjects*, l'existence d'une ligne définissant cette clé dans *sysencryptkeys* est vérifiée.

2 Pour chaque colonne de *syscolumns* marquée aux fins du cryptage, l'existence de la clé est vérifiée dans *sysobjects* et *sysencryptkeys*.

1.25 Syntaxe complète des commandes

Les informations suivantes illustrent la syntaxe complète pour les commandes couvertes par ce bulletin.

1.25.1 *alter encryption key*

alter encryption key *clé1* as default | not default

1.25.2 alter table

alter table [[*basededonnées*.[*propriétaire*].*nom_table*
{ add *nom_colonne type_données*
[default {*expression_constante* | user | null}]
{identity | null | not null}
[off row | in row]
[[constraint *nom_contrainte*]
{ { unique | primary key }
[clustered | nonclustered]
[asc | desc]
[with { fillfactor = *pct*,
max_rows_per_page = *nbre_lignes*,
reservepagegap = *nbre_pages* }]
[on *nom_segment*]
| references [[*basededonnées*.]*propriétaire*.]*table_réf*
[(*colonne_réf*)]
[match full]
| check (*condition_recherche*)] ... }
[encrypt [with [*basededonnées* .] *propriétaire*] .]*nom_clé*
[, *colonne_suivante*]...
| add {[constraint *nom_contrainte*]
{ unique | primary key}
[clustered | nonclustered]

(*nom_colonne* [asc | desc]
[, *nom_colonne* [asc | desc]...])
[with { fillfactor = *pct*,
max_rows_per_page = *nbre_lignes*,
reservepagegap = *nbre_pages*}]
[on *nom_segment*]
| foreign key (*nom_colonne* [{, *nom_colonne*}...])
references [[*basededonnées*.]*owner*.]*table_réf*
[(*colonne_réf* [{, *colonne_réf*}...])]
[match full]
| check (*condition_recherche*)}
| drop {*nom_colonne* [, *nom_colonne*]...
| constraint *nom_contrainte* }
| modify *nom_colonne type_données* [null | not null]
[encrypt | [with [*basededonnées* .]*propriétaire*].]
nom_clé
| modify column_name datatype [null | not null
[encrypt | [with [*basededonnées* .]*propriétaire*].]
***nom_clé*]**
|decrypt
[, *colonne_suivante*]...
| replace *nom_colonne*
default { *expression_constante* | user | null}
| partition *nombre_partitions*
| unpartition| { enable | disable } trigger
| lock {allpages | datarows | datapages } }
| with exp_row_size=num_bytes
| [alter_partition_clause]
[partition_clause]

1.25.3 *Création d'une table*

create table [*basededonnées* .[*propriétaire*].]*nom_table* (*nom_colonne type_données)*
[default {*expression_constante* | user | null}]
{[{identity | null | not null}]
[off row | [in row [(*taille_en_octets*)]]
[[constraint *nom_contrainte*]
{{unique | primary key}
[clustered | nonclustered] [asc | desc]
[with { fillfactor = *pct,*
max_rows_per_page = *nbre_rows*, }
reservepagegap = nbre_pages }]
[on *nom_segment*]
| references [[*basededonnées* .]*propriétaire* .]*table_réf*
[(*colonne_réf*)]
[match full]
| check (*condition_recherche*)}]}
[match full]...
[encrypt [with [[*basededonnées*.] *propriétaire] .*] *nom_clé*]

```
| [constraint nom_contrainte]
    {{unique | primary key}
        [clustered | nonclustered]
        (nom_colonne [asc | desc]
            [{, nom_colonne [asc | desc]}...])
        [with { fillfactor = pct
            max_rows_per_page = nbre_lignes,
            reservepagegap = nbre_pages } ]
        [on nom_segment]
    |foreign key (nom_colonne [{,nom_colonne}...])
        references [[basededonnées.]propriétaire.]table_réf
            [(colonne_réf [{, colonne_réf}...])]
            [match full]
    | check (condition_recherche) ... }
[{, {colonne_suivante | contrainte_suivante}}...])
[lock {datarows | datapages | allpages }]
[with { max_rows_per_page = nbre_lignes,
        exp_row_size = nbre_octets,
        reservepagegap = nbre_pages,
        identity_gap = valeur } ]
[ table_lob_clause ]
[ on nom_segment ]
[ [ external table ] at chemin_accès ]
[ partition_clause ]
```

1.25.4 *select*

```
clause_into ::=
   into [[basededonnées.]propriétaire.]nom_table
   [(colname encrypt [with [basededonnées.] propriétaire].] nom_clé
            [, colname encrypt [with [database. ]owner] . ]
               keyname])]
   [ lock {datarows | datapages | allpages } ]
   [ with option_into [, option_into] ...]

      option_into ::=
          | max_rows_per_page = nbre_lignes
          | exp_row_size = nbre_octets
          | reservepagegap = nbre_pages
          | identity_gap = intervalle
          [existing table nom_table]
          [[external type] at "chemin_accès"
          [column delimiter délimiteur]]
```

2. Internet protocol version 6

Pour qu'Adaptive Server prenne en charge IPv6, vous devez démarrer Adaptive Server avec l'indicateur trace 7841, qui permet à Adaptive Server de déterminer la disponibilité de IPv6 et de le prendre en charge.

IPv6 est disponible sur les plates-formes Sun Solaris 32 et 64 bits.

- Plates-formes Sun Solaris 32 bits et 64 bits
- Windows
- Plates-formes HP 32 bits et 64 bits

Pour plus d'informations sur la configuration et l'administration de la mise en réseau IPv6 sur votre plate-forme, reportez-vous à la documentation de votre système d'exploitation.

3. Messagerie en temps réel

Real Time Messaging Services pour Adaptive Server version 12.5.3a prend en charge TIBCO JMS et IBM WebSphere MQSeries.

4. Prise en charge du module PAM dans Adaptive Server 64 bits sur AIX

Adaptive Server version 12.5.3a sur AIX 64 bits prend en charge l'authentification utilisateur basée sur le module PAM (Pluggable Authentication Module). Bien que la version 12.5.3a d'Adaptive Server 64 bits soit sortie sur AIX 5.1, IBM ne prend en charge PAM dans les applications 64 bits que sous AIX 5.2. Sybase vous recommande de mettre votre système à niveau vers AIX 5.2 pour utiliser cette fonctionnalité. Veuillez contacter votre représentant de support SE afin de vous assurer que vous disposez du dernier patch PAM disponible pour votre hôte IBM.

Les librairies PAM 64 bits fournies avec AIX 5.1 ne permettent pas de charger automatiquement des bibliothèques 64 bits depuis `/usr/lib/security/64`. Pour activer l'AU PAM dans un serveur ASE 64 bits version 12.5.3a sur AIX5.1, vous devez indiquer le chemin d'accès complet du module PAM dans le fichier */etc/pam.conf*. L'exemple suivant illustre la manière de spécifier le module SE pam_aix sur une machine AIX 5.1.

Autorisation Adaptive Server requise `/usr/lib/security/64/pam_aix`

5. Tableau des fonctionnalités

Ce tableau illustre les diverses fonctionnalités disponibles dans Adaptive Server Enterprise version 12.5.3a pour différentes plates-formes :

Figure 3 : Tableau

Operating System	Sol32	Sol64	HP32	HP64	AIX64	Linux x86	Windows x86
Options							
Encrypted Columns	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	new for 12.5.3a
High Availability	*	*	*	*	*	*	*
Distributed Transaction	*	*	*	*	*	*	*
XML Management	*	*	*	*	*	*	*
Java Option	*	*	*	*	*	*	*
Native XML	*	*	*	*	*	*	*
Java Based XML	*	*	*	*	*	*	*
Web Services	*	*	*	*	*	*	*
Security & Dir Services	*	*	*	*	*	*	*
LDAP Server Directory	*	*	*	*	new for 12.5.3a	*	*
LDAP User Authentication	*	*	*	*	new for 12.5.3a	*	*
Secure Socket Layer	*	*	*	*	*	*	*
Cybersafe Kerberos	*	*					*
MIT Kerberos	*	*		new for 12.5.3a	new for 12.5.3a	*	
Platform Native Kerberos	*	*					
Fine Grained Access Control	*	*	*	*	*	*	*
Pluggable Authentication Modu	*	*			new for 12.5.3a	*	
Content Management	*	*	*	*	*	*	*
Enhanced Full Text Search	*	*	*	*	*	*	*
Real Time Messaging	*	*		*	*	*	*
JMS support	*	*		*	*	*	*
Websphere MQ support	new for 12.5.3a	new for 12.5.3a		new for 12.5.3a	new for 12.5.3a	new for 12.5.3a	
Disaster Recovery	*		*			*	*
Features Included with ASE							
IPv6	*	*	new for 12.5.3a	new for 12.5.3a			new for 12.5.3a
Cross Platform Dump and Loa	*	*	*	*	*	*	*
Job Scheduler	*	*	*	*	*	*	*
ASE Replicator	*	*	*	*	*	*	*

- * Indique que cette fonctionnalité est prise en charge. Une case vide signifie que cette fonctionnalité n'est pas disponible sur cette plate-forme.

- Une prise en charge haute disponibilité indique un support actif-actif.

  Le support actif-passif pour la haute disponibilité est uniquement limité à la plate-forme Solaris/SPARC.
- La disponibilité des options varie d'une édition à l'autre. Veuillez vous reporter à la fiche technique d'Adaptive Server Enterprise pour plus d'informations sur les différences entre les éditions.