新增功能
Adaptive Server® Enterprise 12.5.3a

文档 ID：DC00532-01-1253-02

最后修订日期：2005 年 10 月

本文档介绍 Adaptive Server® Enterprise 12.5.3a 中提供的新增功能。

1. 列加密

Adaptive Server® 鉴定和访问控制机制可确保只有经过正确标识和授权的用户才能访问数据。数据加密可进一步保护磁盘或存档中的敏感数据，防止出现数据被窃或安全问题。

Adaptive Server 中的数据加密允许在列级加密数据。应仅对敏感数据进行加密，以最大程度地降低处理开销。

与在中间层或客户端应用程序中进行加密相比， Adaptive Server 中的加密列功能更易于使用。使用 sql 语句可以创建加密密钥并指定要加密的列。 Adaptive Server 处理密钥生成和存储。当在加密列中写入和读取数据时，会以透明的方式自动对数据进行加密和解密。无需更改应用程序，亦无需购买第三方软件。

加密的数据以密文形式进行存储。非加密数据则存储为纯文本。

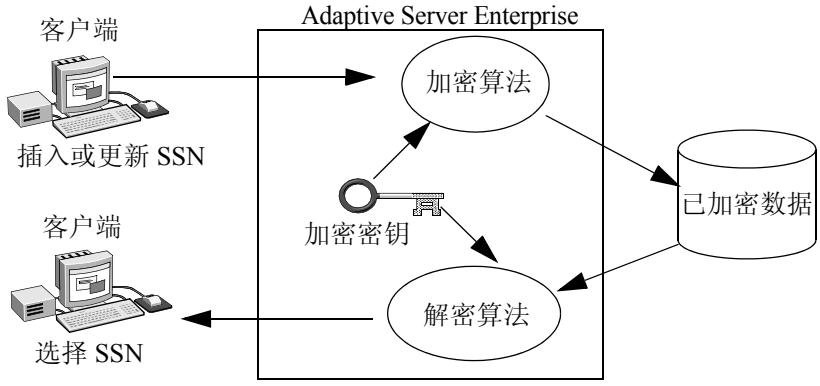
加密功能还包括在以下工具中：

- Sybase Central 和 Adaptive Server 插件。有关详细信息，请参见 *Sybase Central User's Manual* （*Sybase Central 用户手册*）。
- sybmigrate （迁移工具）、 bulk copy 和 CIS，这些内容在 *Adaptive Server Enterprise System Administration Guide* （*Adaptive Server Enterprise 系统管理指南*）中做了介绍。
- Replication Server。有关在复制时进行加密的信息，请参见 *Replication Server Administration Guide* （*Replication Server 管理指南*）。

1.1 概述

图 1 从一个较高的层次显示了 Adaptive Server 中的加密和解密处理。在此示例中，对社会保险号 (SSN) 进行更新和加密：

图 1：Adaptive Server 中的加密和解密



若要创建加密密钥，请使用 create encryption key，它将：

- 使用 Security Builder Crypto™ API 在内部创建具有指定密钥长度的对称密钥。
- 在系统目录 *sysencryptkeys* 中以加密的形式存储密钥。

Adaptive Server 会跟踪用于加密给定列的密钥。列加密使用对称加密算法，这意味着，将使用同一密钥进行加密和解密。

当对加密列中的数据执行 insert 或 update 操作时， Adaptive Server 会在写入行之前立即对数据进行透明加密。当您在加密列中执行 select 操作时， Adaptive Server 会在读取行中的数据后对数据进行解密。整数数据和浮点数据以规范形式进行加密：

- 对于整数数据，使用最高有效位 (MSB) 格式。
- 对于浮点数据，使用电气与电子工程师协会 (IEEE) 浮点标准和 MSB 格式。

如果两个平台使用相同的字符集，则在一个平台上加密的数据可以在另一个平台上解密。

要在 Adaptive Server 中使用加密列，请执行下列操作：

1 安装许可证选项 ASE_ENCRYPTION。有关信息，请参见 *Adaptive Server Enterprise Installation Guide （Adaptive Server Enterprise 安装指南）*。

2 在 Adaptive Server Enterprise 中启用加密：

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

0 — 表示禁用加密。

1 — 表示启用加密。

设置此选项后，重新启动服务器。

如果在包含加密列的服务器中关闭此选项，则对这些列执行的任何命令都将失败，并会显示错误消息。需要同时设置配置参数和许可证选项，才能使用加密列。只有系统安全员才能启用加密列。

3 使用 sp_encryption 设置数据库的系统加密口令。请参见第 4 页的 “设置系统加密口令”。

4 创建用于加密列的密钥。请参见第 5 页的 “创建加密密钥”。

5 指定要加密的列。请参见第 10 页的 “指定对新表加密”和第 11 页的 “加密现有表中的数据”。

6 向必须查看该数据的用户授予解密权限。请参见第 11 页的 “解密权限”。

1.2 设置系统加密口令

系统安全员使用 sp_encryption 设置系统加密口令。系统口令将针对在其中执行 sp_encryption 的数据库，并且其加密值存储在该数据库的 sysattributes 系统表中。

```
sp_encryption system_encr_passwd, password
```

Password 可以长达 64 个字节，Adaptive Server 将使用它加密该数据库中的所有密钥。设置系统加密口令后，您在访问密钥或数据时不必再指定该口令。

必须在创建加密密钥的每个数据库中设置系统加密口令。

系统安全员可以通过使用 sp_encryption 并提供旧口令来更改系统口令。

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

更改系统口令后， Adaptive Server 会自动使用新口令对数据库中的所有密钥重新加密。

1.3 创建和管理加密密钥

Adaptive Server 生成加密密钥并将其以加密形式存储在数据库中。密钥所有者可向表所有者授予使用指定密钥对当前数据库中的列进行加密的权限。

1.3.1 创建加密密钥

所有与密钥和加密相关的信息都由 create encryption key 封装，从而允许您在加密过程中指定加密算法和密钥大小、密钥的缺省属性以及是使用初始化矢量还是填充。

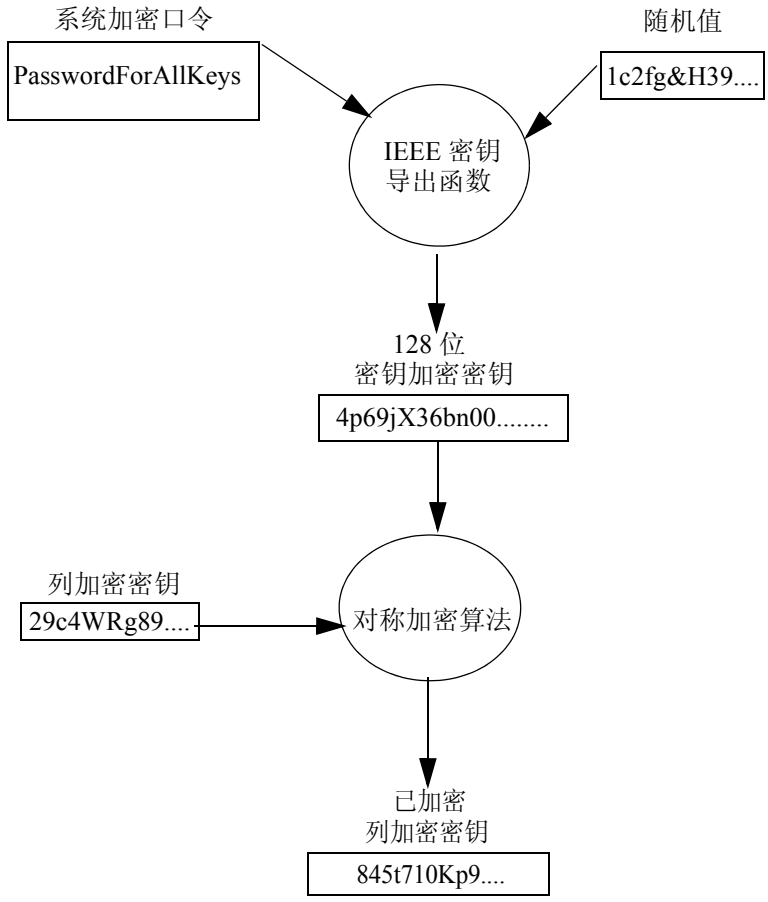
Adaptive Server 中的列加密功能使用高级加密标准 (AES) 对称密钥加密算法，可用的密钥大小为 128、 192 和 256 位。随机密钥生成和密码功能由 Security Builder Crypto™ API 提供。

可以为每个加密列创建单独的密钥。密钥可在多个列间共享，但每个列只能拥有一个密钥。在加密列时，若要对一个列使用初始化矢量，而对另一个列不使用初始化矢量，则可创建两个单独的密钥，一个指定使用初始化矢量，而另一个则不指定。

系统安全员在 create encryption key 中使用 as default 子句设置数据库的缺省加密密钥。在使用 encrypt 限定符时，如果 create table 或 alter table 缺少密钥名，则始终使用缺省密钥。

为对密钥值提供安全保护， Adaptive Server 使用系统加密口令生成 128 位的密钥加密密钥，而此密钥又反过来用于加密新创建的密钥。列加密密钥以加密的形式存储在 *sysencryptkeys* 系统表中。

图 2：加密用户密钥



create encryption key 的语法为：

```
create encryption key keyname [as default] for algorithm
[with [keylength num_bits]
[init_vector [null | random]]
[pad [null | random]]]
```

其中：

- *keyname* — 在当前数据库的用户表、视图和过程名称空间中必须是唯一的。
- as default — 允许系统安全员创建用于加密的数据库缺省密钥。它使表创建者无需在 create table、alter table 和 select into 中使用 keyname 便可指定加密。 Adaptive Server 使用同一数据库中的缺省密钥。可以更改缺省密钥。请参见第 32 页的 “alter encryption key”。
- *algorithm* — 高级加密标准 (AES) 是唯一受支持的算法。AES 支持的密钥大小为 128 位、 192 位和 256 位，支持的块大小为 16 字节。
- keylength *num_bits* — 要创建的密钥的大小 （以位为单位）。对于 AES，有效的密钥长度为 128、 192 和 256 位。缺省密钥长度为 128 位。
- init_vector random — 指定在加密过程中使用初始化矢量。当加密算法使用初始化矢量时，两段相同纯文本的密文是不同的，这样可防止 cryptanalyst 检测数据模式。使用初始化矢量可提高数据的安全性。

  初始化矢量对性能还有一些影响。只有对加密密钥未指定初始化矢量的列才能执行索引创建、优化连接和搜索。请参见第 18 页的 “性能考虑事项”。
- init_vector null — 加密时忽略使用初始化矢量。这样可使列支持索引。

  缺省设置为使用初始化矢量 （即 init_vector random）。使用初始化矢量意味着使用加密的密码块链接 (CBC) 模式；设置 init_vector null 则意味着将使用电子代码簿 (ECB) 模式。
- pad null — 缺省设置。它将忽略随机数据填充。

  如果列必须支持索引，则不能使用填充。
- pad random — 在加密前自动填充随机字节大小的数据。通过使用填充代替初始化矢量，可以使密文随机化。只有纯文本长度小于块长度的一半的列才适合填充。对于 AES 算法，块的长度为 16 字节。

例如，若要将名为 “safe_key” 的 256 位密钥指定为数据库的缺省密钥，系统安全员应输入：

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

以下示例将创建一个称为 “salary_key” 的 128 位密钥，以使用随机填充对列进行加密：

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

以下示例将创建一个名为 “mykey”的 192 位密钥，以使用初始化矢量对列进行加密：

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

系统安全员具有创建加密密钥的缺省权限，并可以将该权限授予其他用户。

例如：

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

1.3.2 使用加密密钥

指定要加密的列时，可以使用同一数据库或其它数据库中的命名密钥。如果不指定命名密钥，将自动使用同一数据库中的缺省密钥对列进行加密。

使用其它数据库中的密钥进行加密可提供特有的安全优势，它可以在数据被窃取或数据库转储时，防止对密钥和加密数据进行访问。若要访问数据，需要同时访问包含数据的数据库存档以及包含加密密钥的数据库存档。管理员还可以使用不同的口令来保护数据库转储，使未经授权的访问更难以进行。

使用其它数据库中的密钥进行加密时需要特别小心，以避免出现分布式系统中的数据和密钥完整性问题。应仔细协调数据库转储和装载。如果使用其它数据库中的命名密钥， Sybase 建议：

- 当转储包含加密列的数据库时，还应转储创建密钥的数据库。如果自上次转储以后添加了新的密钥，则必须执行此操作。
- 当转储包含加密密钥的数据库时，应转储包含用该密钥加密的列的所有数据库。这样可保持加密的数据与可用密钥同步。

系统安全员可以使用 sp_encryption 来标识用给定密钥加密的所有列。请参见第 33 页的 “sp_encryption”。

1.3.3 授予对密钥的权限

密钥所有者必须向其他用户授予密钥的 select 权限，然后该用户才能在 create table、 alter table 和 select into 语句中指定该密钥。对于数据库的缺省密钥，其所有者为系统安全员。仅应 “根据需要”授予对密钥的 select 权限。

以下示例允许具有 db_admin_role 的用户，在 create table 和 alter table 语句中指定加密时使用加密密钥 “safe_key”：

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

通过 insert、 update、 delete 和 select 处理加密列的用户，不需要对加密密钥具有 select 权限。

1.3.4 更改密钥

作为信息安全策略的一部分，应定期更改用于加密列的密钥。使用 create encryption key 创建一个新密钥，然后通过 alter table...modify 来用新密钥对列加密。

以下示例假定 creditcard 列已经过加密。 alter table 命令会使用 cc_key_new 对每个 customer 行的 creditcard 值进行解密并重新加密。

```
create encryption key cc_key_new for AES

alter table customer modify creditcard encrypt with
cc_key_new
```

请参见第 34 页的 “alter table”。

1.4 加密数据

可以对以下数据类型加密：

- int、 smallint、 tinyint
- float4 和 float8
- decimal 和 numeric
- char 和 varchar
- binary 和 varbinary

磁盘上加密的数据的基本类型为 varbinary。有关 varbinary 数据长度的详细信息，请参见第 14 页的 “加密列的长度”。

不会对空值加密。

1.4.1 指定对新表加密

若要对新表中的列加密，请在 create table 语句中使用以下列选项：

[encrypt [with [*database*.[*owner*].]*keyname*]]

keyname — 标识使用 create encryption key 创建的密钥。表的创建者必须对 *keyname* 具有 select 权限。如果未提供 *keyname*， Adaptive Server 将查找使用 create encryption key 中的 as default 子句创建的缺省密钥。有关 create table 的完整语法，请参见第 38 页的 “create table”。

以下示例创建两个密钥：一个是数据库缺省密钥，它使用 init_vector、pad 和 keylength 的缺省值；另一个是使用非缺省值的命名密钥 cc_key。employee 表中的 ssn 列使用缺省密钥加密，customer 表中的 creditcard 列使用 cc_key 加密：

```
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 创建加密列的索引

如果指定的加密密钥不使用初始化矢量或随机填充，则可以对其加密的列创建索引。如果加密列具有初始化矢量或随机填充，则对其执行 create index 命令时，将发生错误。加密列的索引对等于和不等于匹配非常有用，但不能用于范围搜索或排序。

在以下示例中， cc_key 指定不使用初始化矢量或填充进行加密。它允许对任何用 cc_key 加密的列创建索引：

```
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 加密现有表中的数据

若要对现有表中的列加密，请在 alter table 语句中使用以下列选项：

[encrypt [with [*database*.[*owner*].]*keyname*

keyname — 标识使用 create encryption key 创建的密钥。表的创建者必须对 *keyname* 具有 select 权限。如果未提供 *keyname*， Adaptive Server 将查找使用 create encryption key 中的 as default 子句创建的缺省密钥。有关 alter table 的完整语法，请参见 *Adaptive Server Enterprise Reference Manual*（*Adaptive Server Enterprise 参考手册*）。

注释 对现有表中的列加密时，如果表中创建有触发器，则会导致 alter table 因错误而失败。在变更表以进行加密之前，必须删除触发器，并在执行 alter table .. encrypt 后重新创建该触发器。

您可以在更改列的加密属性的同时，更改列的其它属性（如数据类型和可为空性）。还可以使用 alter table 添加加密列。

例如：

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

有关完整语法，请参见第 36 页的 “alter table”。

1.5 解密数据

1.5.1 解密权限

您必须具有以下两种权限，才能从加密列中选择明文数据，或者对加密列进行搜索或连接：

- 对列的 select 权限
- 对列的 decrypt 权限，用于目标列表以及 where、 having、 order by、 update 等其它类似子句

表所有者使用 grant decrypt 向其他用户、组和角色显式授予对表中的一个或多个列解密的权限。当过程或视图所有者授予以下权限时，可以隐式授予解密权限：

- 针对存储过程的 exec 权限，用于从具有以下特征的加密列中进行选择：过程的所有者同时拥有包含该加密列的表。

- 针对视图列的 decrypt 权限，用于从具有以下特征的加密列中进行选择：视图的所有者同时拥有该表。

在这两种情况下，均无需对基表中的加密列授予解密权限。

其语法为：

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

在表级授予解密权限时，会对表中所有加密列授予解密权限。

对 customer 表中所有加密列授予解密权限：

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

以下示例显示 user2 对基表 “employee” 中 ssn 列的隐式解密权限。user1 按以下方式设置 employee 表和 employee_view：

```
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
```

当从 emp_salary 视图进行选择时，user2 有权访问已解密的社会保险号：

```
select * from emp_salary
```

1.5.2 撤消解密权限

可以使用以下语法撤消用户的解密权限：

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

例如：

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

1.6 删除加密和密钥

1.6.1 删除加密

如果您是表所有者，则可以使用带有 decrypt 选项的 alter table 删除对列的加密。

其语法为：

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

例如，删除对 customer 表中 creditcard 列的加密：

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

1.6.2 删除密钥

系统安全员和密钥所有者可以删除密钥。仅当所有数据库中没有使用密钥的加密列时，才能删除该密钥。无法在可疑数据库和脱机数据库中检查是否存在由该密钥加密的列。该命令会发出一条警告消息来指出不可用的数据库，但它不会失败。将该数据库联机后，所有包含用已删除密钥加密的列的表都将变为不可用。若要恢复该密钥，系统管理员必须装载在删除密钥之前转储的已删除密钥的数据库。

使用以下语法删除加密密钥：

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

例如：

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

1.7 *select into* 命令

缺省情况下，即使源表包含一个或多个加密列， select into 也会创建不带加密的目标表。使用 select into 需要拥有对源表的列级权限，包括 decrypt。

使用以下语法对新表中的列加密：

```
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
```

即使数据在源表中未加密，也可以对目标表中的特定列加密。如果用于加密源表中的列的密钥与为目标列指定的密钥相同， Adaptive Server 将绕过源表的解密步骤以及目标表的加密步骤。

用于对目标表加密的规则，与用于对源表执行的 create table 中的 encrypt 限定符的规则，存在以下相同点：

- 允许加密列中的数据类型
- 忽略 keyname 时使用数据库的缺省密钥
- 需要对用于加密目标列的密钥拥有 select 权限

例如，加密 creditcard 列：

```
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 加密列的长度

在 create table 和 alter table 操作期间，Adaptive Server 将计算加密列的最大内部长度。若要做出有关模式安排和页大小的决策，数据库所有者必须知道加密列的最大长度。

AES 是一种块密码算法。在块密码算法中，加密数据的长度是该加密算法中块大小的倍数。对于 AES，块大小为 128 位 （即 16 字节）。因此，加密列至少占用 16 字节的空间，额外空间用于：

- 初始化矢量。如果使用初始化矢量，它会向每个加密列添加 16 字节。缺省情况下，加密过程会使用初始化矢量。在 create encryption key 中指定 init_vector null 可忽略初始化矢量。
- 纯文本数据的长度。如果列类型是 char、 varchar、 binary 或 varbinary，则其数据在加密之前会添加 2 个字节的前缀。除非附加的 2 个字节导致密文占用额外的块，否则，加密列不会再占用额外的空间。
- 一个 sentinel 字节。该字节附加到密文的后面，可防止数据库系统删除尾随零。

表 1：密文长度

用户指定的列类型	输入数据长度	是否使用初始化矢量？	内部列类型	加密数据长度
tinyint、smallint 或 int	1、2 或 4	否	varbinary(17)	17
tinyint、smallint 或 int	1、2 或 4	是	varbinary(33)	33
tinyint、smallint 或 int	0（空）	否	varbinary(17)	0
float、float(4)、real	4	否	varbinary(17)	17
float、float(4)、real	4	是	varbinary(33)	33
float、float(4)、real	0（空）	否	varbinary(17)	0
float(8)、double	8	否	varbinary(17)	17
float(8)、double	8	是	varbinary(33)	33
float(8)、double	0（空）	否	varbinary(17)	0
numeric(10,2)	3	否	varbinary(17)	17
numeric(10,2)	3	是	varbinary(33)	33
numeric(38,2)	18	否	varbinary(33)	33
numeric(38,2)	18	是	varbinary(49)	49
numeric(38,2)	0（空）	否	varbinary(33)	0
char、varchar(100)	1	否	varbinary(113)	17
char、varchar(100)	14	否	varbinary(113)	17
char、varchar(100)	15	否	varbinary(113)	33
char、varchar(100)	15	是	varbinary(117)	49
char、varchar(100)	31	是	varbinary(117)	65
char、varchar(100)	0（空）	是	varbinary(117)	0
binary、varbinary(100)	1	否	varbinary(113)	17
binary、varbinary(100)	14	否	varbinary(113)	17
binary、varbinary(100)	15	否	varbinary(113)	33
binary、varbinary(100)	15	是	varbinary(117)	49
binary、varbinary(100)	31	是	varbinary(117)	65
binary、varbinary(100)	0（空）	是	varbinary(117)	0

char 和 binary 被视为可变长度数据类型，在加密之前会去掉空白和零填充。解密数据时，将应用所有空白或零填充。

注释 加密列在磁盘中的列长度将增加，但这种增加是不可见的。例如， sp_help 仅显示原始大小。

1.9 审计加密列

您可以审计与加密列相关的 DDL 命令 （例如，用于创建或删除加密密钥的命令）。此外，当您创建表时，审计记录将包含加密列的名称及其相应的加密密钥。通过数据库范围的审计选项，可以对加密列和密钥的审计记录进行分组和管理。

1.9.1 审计选项

下表摘自 Adaptive Server System Administration Guide （Adaptive Server 系统管理指南），其中显示了使用现有事件选项和新增事件选项审计的新命令。

表 2：审计选项、要求及示例

选项	*login_name*	*object_name*	设置选项时应在的数据库	要审计的命令或访问
encryption_key （数据库特定）	all	要审计的数据库	任何	alter encryption key create encryption key drop encryption key create table drop table alter table sp_encryption
	示例 `sp_audit "encryption_key", "all", "pubs2", "on"` （在 pubs2 数据库中审计所有上述命令。）			

1.9.2 审计值

表 3 列出了 event 列中显示的值 （由 sp_audit 选项排列）。“extrainfo 中的信息”列根据表 3 中介绍的种类描述了可能显示在审计表的 extrainfo 列中的信息。

表 3：event 和 extrainfo 列中的值

审计选项	要审计的命令	event	extrainfo 中的信息输出

alter	alter table	3	*关键字或选项：*
			`ADD/DROP/MODIFY COLUMNS`
			`REPLACE COLUMN`
			`ADD CONSTRAINT`
			`DROP CONSTRAINT`
			如果添加了一个或多个加密列，则关键字将包含：
			`ADD/DROP/MODIFY COLUMNS column1/keyname1, [,column2/keyname2]`
			其中， *keyname* 是密钥的全限定名。

create	create table	10	对于加密列，关键字将包含列名和密钥名。 `EK column1/keyname1[,column2 keyname2]` 其中，EK 为前缀，用于指示后续信息将引用加密密钥；而 *keyname* 则是密钥的全限定名。
encryption_key	sp_encryption	106	*关键字* 包含 `ENCR_ADMIN system_encr_passwd password ********` 如果是首次设置口令，以及 `ENCR_ADMIN system_encr_passwd password ********` ******** 如果后来更改口令
	create encryption key	107	*关键字包含*： algorithm Name-bitlength/IV [RANDOM|NULL]/PAD [RANDOM|NULL] 例如：`AES-128/IV RANDOM/PAD NULL`

	alter encryption key	108	*关键字包含*：
			`NOT DEFAULT`
			如果密钥不再是缺省密钥。
			`DEFAULT`
			如果将密钥设置为缺省密钥。

	drop encryption key	109	

1.9.3 新事件名和编号

可以查询特定审计事件的审计追踪。使用带有 *event id* 参数的 audit_event_name。

audit_event_name(*event_id*)

表 4 中列出了新事件的编号和名称。

表 4：新事件编号

事件编号	事件名输出
106	`Encrypted Column Administration`
107	`Create Encryption Key`
108	`Alter Encryption Key`
109	`Drop Encryption Key`

1.10 性能考虑事项

加密是一种 CPU 密集型操作，可能会给应用程序带来性能开销，这种开销可通过 CPU 使用率以及使用加密列的命令所占用的时间来衡量。开销的大小则取决于 CPU 和 Adaptive Server 引擎的数目、系统的负荷、访问加密数据的并发会话数以及查询中引用的加密列数。加密密钥的大小和加密数据的长度同样会影响开销。通常，密钥大小越大，数据越长，加密操作的 CPU 使用率也就越高。

本节讨论搜索加密列对性能的影响，以及 Adaptive Server Enterprise 如何优化对加密数据的处理，以尽可能减少加密和解密操作的数目。

1.10.1 加密列上的索引

如果加密列的加密密钥未指定使用初始化矢量或随机填充，则可以在该列上创建索引。使用初始化矢量或随机填充会导致将相同数据加密为不同模式的密文，从而使索引无法强制实施唯一性。

在对数据进行等于和不等于匹配时，加密数据上的索引将非常有用，但它无法用于数据排序、范围搜索或查找最小值和最大值。如果 Adaptive Server 对加密列执行与顺序有关的搜索，它将无法对加密数据进行索引查找。而必须将每一行中的加密列解密，然后再进行搜索。此过程会降低数据的处理速度。

1.10.2 加密列上的连接

Adaptive Server 通过执行密文比较来优化两个加密列间的连接，但必须满足下列条件：

- 相连接的列具有相同的数据类型。与 binary 和 varbinary 一样， char 和 varchar 也被视为相同的数据类型。
- 对于 int 和 float 类型，各列的长度应相同。对于 numeric 和 decimal 类型，各列的精度和标度应相同。
- 相连接的列使用同一密钥加密。
- 相连接的列不是表达式的组成部分。例如，无法对 t.encr_col1 = s.encr_col1 +1 这样的连接执行密文连接。
- 创建加密密钥时使用 init_vector，而将 pad 设置为 NULL。
- 连接运算符为 “=” 或 “<>”。
- 数据使用缺省的排序顺序。

例如，以下语句会将模式设置为执行密文连接：

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

还可以在相连接的列上设置索引：

```
create index cust_cc on customer(creditcard)

create index daily_cc on daily_xacts(creditcard)
```

Adaptive Server 可执行以下 select 语句，以计算客户在给定信用卡中的日常支出总和，而无需对 customer 或 daily_xacts 表中的 creditcard 列解密。

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

1.10.3 常数值搜索参数和加密列

如果对加密列的等于或不等于比较得出一个常数值，则 Adaptive Server 会仅对该常数值加密一次，而不是对表中每一行的加密列都进行解密，从而可以优化列扫描。第 19 页的 “加密列上的连接”中列出的限制也同样适用。

Adaptive Server 无法利用索引对加密列执行范围搜索；在执行数据比较前，它必须先对每一行解密。如果查询中包含其它谓词， Adaptive Server 将选择最有效的连接顺序，这样通常会最后搜索包含最小数据集的加密列。

如果您的查询包含多个范围搜索，但其中又没有有用索引，请在编写查询时，使它最后对加密列执行范围搜索。例如，以下查询将搜索罗德岛州中收入超过 $100, 000 的纳税人的社会保险号。其中，对 zipcode 列的范围搜索在对 “调整的总收入”加密列的范围搜索之前进行：

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

1.10.4 将加密数据作为密文移动

Adaptive Server 通过复制密文而不是先解密后重新加密数据的方式来复制加密数据，以尽可能实现优化。这一方式适用于 select into、批量复制和复制。

1.11 系统表

1.11.1 *syscolumns*

在 syscolumns 系统表中，以下列描述加密属性：

字段	类型	值	说明
encrtype	int	空	采用加密形式的数据的类型。
encrlen	int	空	加密数据的长度。
encrkeyid	int	空	密钥的对象 id。
encrkeydb	varchar(30)	空	加密密钥所在的数据库名。如果密钥与加密列位于同一数据库中，则为 NULL。
encrdate	datetime	空	从 sysobjects.crdate 复制的创建日期。

1.11.2 *sysobjects*

对于每个密钥， sysobjects 都有一个 EK （加密密钥）类型的条目。

对于跨数据库密钥引用， syscolumns.encrdate 与 sysobjects.crdate 匹配。

sysencryptkeys 中的 encrkeyid 与 *sysobjects* 中的 id 列匹配。

1.11.3 *sysencryptkeys*

数据库中创建的每个密钥 （包括缺省密钥）在数据库特定的系统目录 sysencryptkeys 中都有相应的条目。

表 5：sysencryptkeys

字段	类型	说明
id	int	加密密钥 ID。
ekalgorithm	int	加密算法。
type	smallint	标识密钥类型。值为 EK_SYMMETRIC 和 EK_DEFAULT。
status	int	内部状态信息。
eklen	smallint	用户指定的密钥长度。
value	varbinary(1282)	密钥的加密值。包含密钥的对称加密。为加密密钥， Adaptive Server 对来自系统加密口令的 128 位密钥应用 AES 标准。
uid	int null	未使用。
eksalt	varbinary(20)	包含用于对加密密钥验证解密的随机 salt。
ekpairid	int null	未使用。
pwdate	datetime null	未使用。
expdate	int null	未使用。
ekpwdwarn	int null	未使用。

1.12 *ddlgen* 实用程序更改

ddlgen 支持为加密的密钥生成 DDL 语句。若要指定密钥，请使用：

<dbName>.<owner>.<keyName>

加密密钥的新类型 EK 用于生成 DDL 以创建加密密钥并向其授予相应权限。 ddlgen 生成加密列的信息、 grant decrypt 语句以及表的 DDL。

以下示例将在名为 “HARBOR”的计算机 （使用端口 1955）上，为数据库 “accounts”中的所有加密的密钥生成 DDL：

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

或者，可以使用 -D 选项来指定数据库名：

```
    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 还具有一个扩展的选项，可生成用于指定密钥加密值（如 *sysencryptkeys* 中所示）的 create encryption key。该选项便是 -XOD，如果在移动数据时必须跨服务器同步加密密钥，则可以使用它。例如，若要使服务器 “PACIFIC”中的 cc_key 在服务器 “ATLANTIC”中可用，则可以在 “PACIFIC”上执行带 -XOD 的 ddlgen，如下所示：

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

ddlgen 输出为：

```
------------------------------------------------------------------------------
-- 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
```

接下来更改由 create encryption key 生成的密钥，以指定 “ATLANTIC”上的目标数据库，并在目标服务器上运行该命令。现在便可以在服务器 “ATLANTIC”上使用 cc_key 对从 “PACIFIC”移到 “ATLANTIC”的数据进行解密。

有关 ddlgen 语法选项的详细信息，请参见 *Adaptive Server Enterprise Utility Guide （Adaptive Server Enterprise 实用程序指南）*；有关将 ddlgen 用于复制型数据库的示例，请参见 *Replication Server Administration Guide （Replication Server 管理指南）*。

1.13 复制加密的数据

如果您的站点复制模式更改，则会复制以下 DDL 语句：

- alter encryption key
- 带加密扩展的 create table 和 alter table
- create encryption key
- grant 和 revoke create encryption key
- 针对密钥的 grant 和 revoke select
- 针对列的 grant 和 revoke decrypt

- sp_encryption system_encr_passwd
- drop encryption key

密钥以加密的形式复制。

如果您的系统不复制 DDL，则必须在复制站点手动同步加密密钥。ddlgen 支持使用特殊形式的 create encryption key 来复制密钥值。

insert 和 update 以加密的形式复制加密列，这样当 Replication Server 在磁盘的稳定队列中处理复制的数据时，可以对这些数据提供保护。

有关在复制时进行加密的信息，请参见 *Replication Server Administration Guide （Replication Server 管理指南）*。

1.14 批量复制 (*bcp*)

bcp 可以采用纯文本或密文的形式在数据库间传送加密的数据。缺省情况下， bcp 复制纯文本数据。 bcp 按照以下方式处理纯文本数据文件：

- 在执行 bcp in 时，Adaptive Server 会在插入数据前自动对数据加密。使用慢速 bcp。用户必须对所有列都具有 insert 和 select 权限。
- 执行 bcp out 时， Adaptive Server 将自动对数据解密。用户需要对所有列都具有 select 权限；此外，还需要对加密列具有 decrypt 权限。

以下示例将 “customer”表复制出为采用本机格式的纯文本数据：

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

使用 bcp 的 -C 选项可将数据作为密文复制。复制密文时，可以跨不同的操作系统复制数据。如果将字符数据复制为密文，则两个平台必须支持相同的字符集。

bcp 的 -C 选项允许管理员在缺少对数据的 decrypt 权限时运行 bcp。使用 -C 选项时， bcp 将按以下方式处理数据：

- 假定在执行 bcp in 期间数据采用密文格式，并且 Adaptive Server 不执行加密。仅当要复制到 Adaptive Server 的文件是使用 bcp out 的 -C 选项创建时，才可以在 bcp in 中使用 -C 选项。已复制密文所在的源列与数据复制到的列必须具有完全相同的列属性，并使用同一密钥加密。使用快速 bcp。用户必须具有 insert 和 select 权限。
- 执行 bcp out 时，不对从 Adaptive Server 复制出的数据解密。密文数据采用十六进制格式。用户必须对所有列都具有 select 权限。复制密文时，不需要对加密列执行 decrypt 操作。

- 加密的 char 或 varchar 数据将保留 Adaptive Server 在加密数据时使用的字符集。如果将数据以密文格式复制到另一台服务器，则目标服务器上使用的字符集必须与从源服务器复制的加密数据的字符集相匹配。源服务器在加密数据时使用的字符集与加密数据并未存储在一起，且该字符集在目标服务器上是未知的或并未进行转换。

  系统管理员必须检验源服务器和目标服务器上的字符集是否匹配。还可以在不使用 -C 选项的情况下执行 bcp，这样便不会遇到字符集问题。

  用于字符集转换的 -J 选项不能与 -C 选项一起使用。

以下示例复制 “customer”表。将以密文形式复制出 cc_card 列。其它列则以字符格式复制。用户 “roy”无需对 customer cc_card 具有解密权限。

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

警告！ 仅当视图限定不搜索加密的列时，才可以对视图执行带有 -C 标志的 bcp out。

1.15 组件集成服务 (CIS)

缺省情况下，加密和解密由远程 Adaptive Server 处理。 CIS 会对远程 Adaptive Server 上的加密列进行一次性检查。如果远程 Adaptive Server 支持加密，则 CIS 会使用与加密列相关的元数据更新本地 *syscolumns* 目录。

- create proxy_table 可自动使用远程表中的任何加密列信息更新 *syscolumns*。
- create existing table 可自动使用远程表中的任何加密列元数据更新 *syscolumns*。不允许在 create existing table 的 *columnlist* 中使用加密口令。如果 CIS 在远程表中找到任何加密的列，它会自动将其标记为已加密。
- 不允许在包含加密列的位置执行 create table。
- 不允许对代理表的加密列执行 alter table。
- select into existing 可选择源表中的纯文本并将其插入目标表中。然后，本地 Adaptive Server 会在将该纯文本插入任何加密的列中之前对其进行加密。

以下列从远程服务器的 *syscolumns* 目录更新：

- encrtype — 磁盘上数据的类型。
- encrlen — 加密数据的长度。
- status2 — 指示列已被加密的状态位。

1.16 *load* 和 *dump* 数据库

dump 和 load 用于处理加密列的密文。这一行为可确保加密列中的数据在磁盘上仍保持加密状态。 dump 和 load 与整个数据库有关。缺省密钥和在同一数据库中创建的密钥将同与它们相关的数据一起转储和装载。

如果装载数据库包含其它数据库中使用的加密密钥，则除非使用新语法 with override，否则 load 将不会成功。

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

如果密钥与它们加密的列位于单独的数据库中， Sybase 建议：

- 当转储包含加密列的数据库时，还应转储创建密钥的数据库。如果自上次转储以后添加了新的密钥，则必须执行此操作。
- 当转储包含加密密钥的数据库时，应转储包含用该密钥加密的列的所有数据库。这样可保持加密的数据与可用密钥同步。
- 在装载包含加密密钥的数据库和包含加密列的数据库之后，请将这两个数据库同时联机。

如果将包含密钥的数据库装载到具有不同名称的数据库中，则当您访问其它数据库中的加密列时，将导致发生错误。若要更改密钥数据库的数据库名，请执行下列步骤：

- 在转储包含加密列的数据库之前，请使用 alter table 解密数据。
- 转储包含密钥和加密列的数据库。
- 装载数据库之后，通过 alter table 用新命名数据库中的密钥对数据重新加密。

加密密钥和加密列间的一致性问题与跨数据库参照完整性的一致性问题相似。请参见 *Adaptive Server Enterprise System Administration Guide（Adaptive Server Enterprise 系统管理指南）*中的 “Cross-database constraints and loading databases”（跨数据库约束和装载数据库）。

请勿尝试将任何包含加密的数据的转储装载到 Adaptive Server 的早期版本中。将数据库装载到 Adaptive Server 12.5.3a 版中并删除其中的所有加密。执行转储，然后将数据库装载到 Adaptive Server 的某个早期版本中。请参见第 29 页的 “降级过程”。

有关密钥的详细信息，请参见第 5 页的 “创建和管理加密密钥”。

1.17 *unmount database*

当使用其它数据库中的密钥加密列时，应将所有相关的数据库作为一个集合卸下。包含加密列的数据库与包含密钥的数据库之间的相互依赖性，类似于使用参照完整性的数据库间的相互依赖性。

如果数据库中包含的列是使用其它数据库中的密钥加密的，则应使用 override 选项对该数据库执行 unmount。

通过以下命令，已将在 key_db 中创建的加密密钥用于加密 col_db 中的列。以下命令可成功卸下指定的数据库：

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

以下命令未使用 override，将会失败，并显示一条错误消息：

```
unmount database key_db

unmount database col_db
```

1.18 *quiesce database*

如果数据库包含加密密钥，则可以使用 quiesce database。

如果数据库中的列是使用其它数据库中的密钥加密的，则必须使用 with override 对该数据库执行 quiesce。

允许 `quiesce database` ***key_db, col_db***，其中 *key_db* 是包含加密密钥的数据库， *col_db* 是表中含有用 *key_db* 中的密钥加密的列的数据库。

例如，以下命令将成功执行，其中 key_db 包含用于加密 col_db 中的列的加密密钥：

```
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 删除数据库

如果数据库中包含的密钥当前用于加密其它数据库中的列，则为避免密钥意外丢失， Adaptive Server 将使 drop database 操作失败。在删除包含加密密钥的数据库之前，必须先删除加密或删除包含加密列的数据库。

在以下示例中， key_db 是加密密钥所驻留的数据库， col_db 是包含加密列的数据库：

```
drop database key_db, col_db
```

Adaptive Server 将引发错误，并使删除 key_db 的操作失败。但将成功删除 col_db。若要删除这两个数据库，请先删除 col_db：

```
drop database col_db, key_db
```

1.20 *sybmigrate*

sybmigrate 是一个迁移工具，可用于将数据从一台服务器迁移到另一台服务器。

缺省情况下， sybmigrate 以密文格式迁移加密的列。这样可避免在源上解密数据以及在目标上加密数据所带来的开销。在某些情况下， sybmigrate 可选择 reencrypt 迁移方法，即在源上解密数据并在目标上加密数据。

对于包含加密列的数据库， sybmigrate 将：

1 迁移系统加密口令。如果指定不迁移系统加密口令， sybmigrate 将使用 reencrypt 方法迁移加密的列，而不是直接迁移密文。

2 迁移加密密钥。您可以选择要迁移的密钥。 sybmigrate 会自动在当前数据库中选择用于对同一数据库中的列进行加密的密钥。如果已选择迁移系统加密口令，则 sybmigrate 将使用加密密钥的实际值迁移加密密钥。 *sysencryptkeys* 系统表中的密钥值已使用系统加密口令加密，它们便是要迁移的值。如果尚未迁移系统加密口令， sybmigrate 将通过名称迁移密钥，以避免迁移的密钥无法在目标上正确解密。通过名称迁移密钥时，将导致在目标上创建的密钥的密钥值不同于它在源上的密钥值。

3 迁移数据。缺省情况下，数据以密文形式传送。可以将密文数据迁移到不同的操作系统上。字符数据要求目标服务器使用与源服务器相同的字符集。

sybmigrate 将数据库作为一个工作单元处理。如果源服务器上的数据库含有用其它数据库中的密钥加密的数据，则应先迁移密钥数据库。

在以下情况中， sybmigrate 将选择对已迁移的数据重新加密：

- 当前数据库中的任何密钥都未被专门选择进行迁移，或已经存在于目标服务器上。由于无法保证目标中的密钥与源中的密钥一致，因此必须对迁移数据重新加密。
- 未选择迁移系统口令。如果目标与源中的系统口令有别，则无法通过值迁移密钥。反过来，数据也无法以密文形式迁移。
- 用户使用以下标志：

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

重新加密数据会降低性能。当您在重新加密模式下执行迁移时，迁移日志文件中将写入一条有关此影响的消息。

若要迁移加密的列，必须同时启用 sa_role 和 sso_role。

1.21 降级过程

如果您从未在服务器中配置 enable encrypted columns，则在旧版本的 Adaptive Server 中使用 12.5.3a 数据库之前，无需执行任何操作。若要检验您是否从未配置过加密列，一种方式是检查是否所有数据库中都不存在 *sysencryptkeys* 系统表。

在执行降级过程之前，应备份所有数据库。

如果服务器已配置为使用加密的列，则在降级该服务器之前，必须删除或修改任何包含加密列的表以删除加密。然后运行 sp_encryption remove_catalog，它将检验每个数据库中是否已不存在加密的列，然后删除系统表 *sysencryptkeys*。 *syscolumns* 中针对 12.5.3a 添加的新列将被旧版二进制忽略，因此无需删除这些列

要从 12.5.3a 服务器降级到早期版本的 12.5.x，请执行下列操作：

1 如果当前未启用加密的列，则系统安全员可以执行以下操作：

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

2 使用 drop 或 alter 对所有数据库中包含加密列的所有表解密。系统安全员可在创建加密密钥的每个数据库中运行以下命令，以列出该数据库中创建的所有加密密钥：

```
sp_encryption help
```

对于列出的每个密钥，系统安全员可运行以下命令，以查看使用特定密钥加密的列的列表：

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

对于每个加密的列，必须执行下列步骤之一：

- 执行 alter table 解密加密的列
- 执行 alter table 删除加密的列
- 执行 drop 删除包含加密列的表
- 删除加密密钥

3 若要保证在删除系统表时没有其他用户能够访问 Adaptive Server，请以单用户模式重新启动服务器。请参见 *Adaptive Server Enterprise Utility Guide （Adaptive Server Enterprise 实用程序指南）*。

4 具有 sso_role 和 sa_role 角色的用户必须执行以下系统存储过程，该存储过程将删除每个数据库中的 *sysencryptkeys* 目录：

```
sp_encryption remove_catalog
```

如果数据库不可用，则此命令将输出错误并退出。如果存在由 *sysencryptkeys* 中的任何密钥加密的列，则该命令不会删除 *sysencryptkeys*，但会输出一则错误或警告并继续处理下一个数据库。

如果 sp_encryption 成功删除了 *sysencryptkeys*，它还会从每个数据库的 *sysattributes* 中删除以下行：

- 添加到 *sysencryptkeys* 的升级项的记录
- 数据库的系统加密口令

5 将系统存储过程 sp_encryption 从 *sybsystemprocs* 数据库中删除。

6 关闭服务器。现在，您可以使用 12.5.3a 之前版本中的 12.5.x Adaptive Server 二进制文件。

若要在从已降级的 12.5.3a 服务器向前滚动回 12.5.3a 时重新启用加密列，请配置 enable encrypted columns。重新启动 12.5.3a 服务器时，将在每个数据库中安装 *sysencryptkeys* 系统表。

1.21.1 降级中的复制问题

如果要降级的服务器对包含加密数据的数据库启用了复制，则在对此类服务器开始执行降级过程之前，必须执行以下操作之一：

1 确保已将主数据库事务日志中的所有已复制数据成功传送到备份数据库或复制数据库中。执行此操作的过程与应用程序相关。

2 截断主数据库中的事务日志，并在 Replication Server 中将该数据库的 RS 定位符重置为零。使用以下命令：

在主数据库中运行：

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

关闭 Replication Server。在 Replication Server 的 RSSD 中运行：

```
rs_zeroltm primary_servername, primary_dbname
```

1.22 新命令

1.22.1 *create encryption key*

所有与密钥和加密相关的信息都由 create encryption key 封装，从而允许您在加密过程中指定加密算法和密钥大小、密钥的缺省属性以及是使用初始化矢量还是填充。

Adaptive Server 使用 Security Builder Crypto™ 来生成密钥并进行加密。

系统安全员具有创建加密密钥的缺省权限，并可以将该权限授予其他用户。

语法

create encryption key [[*database.[owner].]keyname* [as default] for *algorithm*
[with [keylength *num_bits*]
[init_vector [NULL | random]]
[pad [NULL | random]]]

- *keyname* — 在当前数据库的用户表、视图和过程名称空间中必须是唯一的。
- as default — 允许系统安全员创建用于加密的数据库缺省密钥。它使表创建者无需在 create table、alter table 和 select into 中使用 keyname 便可指定加密。 Adaptive Server 使用同一数据库中的缺省密钥。可以更改缺省密钥。请参见第 32 页的 “alter encryption key”。
- *algorithm* — 高级加密标准 (AES) 是唯一受支持的算法。AES 支持的密钥大小为 128 位、 192 位和 256 位，支持的块大小为 16 字节。
- keylength *num_bits* — 要创建的密钥的大小 （以位为单位）。对于 AES，有效的密钥长度为 128、 192 和 256 位。缺省密钥长度为 128 位。
- init_vector random — 指定在加密过程中使用初始化矢量。当加密算法使用初始化矢量时，两段相同纯文本的密文是不同的，这样可防止 cryptanalyst 检测数据模式。使用初始化矢量可提高数据的安全性。

初始化矢量对性能还有一些影响。只有对加密密钥未指定初始化矢量的列才能执行索引创建、优化连接和搜索。请参见第 18 页的“性能考虑事项”。

- init_vector null — 加密时忽略使用初始化矢量。这样可使列支持索引。

  缺省设置为使用初始化矢量（即 init_vector random）。使用初始化矢量意味着使用加密的密码块链接 (CBC) 模式；设置 init_vector null 则意味着将使用电子代码簿 (ECB) 模式。

- pad null — 缺省设置。它将忽略随机数据填充。

  如果列必须支持索引，则不能使用填充。

- pad random — 在加密前自动填充随机字节大小的数据。通过使用填充代替初始化矢量，可以使密文随机化。只有纯文本长度小于块长度的一半的列才适合填充。对于 AES 算法，块的长度为 16 字节。

例如，若要将名为“safe_key”的 256 位密钥指定为数据库的缺省密钥，系统安全员应输入：

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

以下示例将创建一个称为 “salary_key”的 128 位密钥，以使用随机填充对列进行加密：

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

以下示例将创建一个名为 “mykey”的 192 位密钥，以使用初始化矢量对列进行加密：

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

1.22.2 *alter encryption key*

若要更改缺省加密密钥，请输入：

alter encryption key *key1* as default

如果缺省密钥已经存在，它将不再具有缺省属性。 *key1* 将成为缺省密钥。

如果 *key1* 是缺省密钥，则可以按照以下方式删除 *key1* 的缺省名称：

alter encryption key *key1* as not default

如果 *key1* 不是缺省密钥，则此命令将返回错误。

alter encryption key as default 或 not default 只能由系统安全员执行，且不能授予其他用户。

1.22.3 *drop encryption key*

密钥所有者和系统安全员可以删除加密密钥。如果任何数据库中有任意列使用该密钥加密，则此命令将失败。

语法

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

1.22.4 *grant create encryption key*

系统安全员可授予创建加密密钥的权限。

语法

grant create encryption key to *user* | *role*| *group*

1.22.5 *revoke create encryption key*

系统安全员可以撤消其他用户、组和角色创建加密密钥的权限。

语法

revoke create encryption key from *user* | *role* | *group*

1.22.6 *grant decrypt*

表所有者或系统安全员可授予对表或表中某些列的列表的 decrypt 权限。

语法

grant decrypt on [*owner*.]*tablename*[(*columnname* [{,*columname*}])]
to *user* | *group* | *role*

注释 对表或列执行 grant all 并不授予解密权限。

1.22.7 *revoke decrypt*

表所有者或系统安全员可撤消对表或表中某些列的列表的 decrypt 权限。

语法

revoke decrypt on [owner.] *tablename*[(*columnname* [{,*columname*}])]
from *user* | *group* | *role*

1.23 *sp_encryption*

系统安全员使用 sp_encryption 设置系统加密口令。系统口令将针对在其中执行 sp_encryption 的数据库，并且其加密值存储在该数据库的 sysattributes 系统表中。

sp_encryption system_encr_passwd, '*password*'

使用 sp_encryption 指定的口令的长度最多可为 64 字节，并将由 Adaptive Server 用于加密该数据库中的所有密钥。无需指定此口令便可访问密钥或数据。

必须在创建加密密钥的每个数据库中设置系统加密口令。

系统安全员可以通过使用 sp_encryption 并提供旧口令来更改系统口令。

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

更改系统口令后， Adaptive Server 会自动使用新口令对数据库中的所有密钥重新加密。

1.23.1 *sp_encryption help*

sp_encryption help 显示密钥的名称、所有者、大小和加密算法。它还可以指示是否已将该密钥指定为数据库缺省密钥，以及在使用该密钥加密时是使用随机填充还是使用初始化矢量。

```
sp_encryption help [, keyname [, display_cols]]
```

当 sp_encryption help 由具有 sso_role 角色的用户运行时，将显示数据库中所有密钥的密钥属性。当它由不具有 sso_role 角色的用户运行时，将仅显示该用户在该数据库中拥有 select 权限的密钥的密钥属性。

sp_encryption help, *keyname* 显示密钥名的属性。如果该命令由不具有 sso_role 角色的用户运行，则该用户必须对密钥具有选择权限。

sp_encryption help, keyname, display_cols 只能由具有 sso_role 角色的用户运行。它将列出由密钥名加密的列。

1.24 对命令语法的更改

本节中所述命令的语法因增加的加密列功能而受到更改或增添。

1.24.1 alter table

使用 alter table 可对现有数据进行加密或解密，也可以向表中添加加密的列。

语法

对列加密：

```
alter table tablename add column_name
        encrypt [with [database.[owner].]keyname
```

对现有列解密：

```
[decrypt [with [database.[owner].]keyname]]
```

keyname — 标识使用 create encryption key 创建的密钥。表的创建者必须对 *keyname* 具有 select 权限。如果未提供 *keyname*， Adaptive Server 将查找使用 create encryption key 或 alter encryption key as default 创建的缺省密钥。

示例

在现有表 “employee”中创建加密密钥并对 ssn 列加密。

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

使用 alter table 更改加密密钥。当在已加密的列上使用 encrypt 限定符时，Adaptive Server 将对该列解密，并使用新密钥对其重新加密。如果表中包含大量的行，则此操作可能会占用相当长的时间。

1.24.2 create table

使用 encrypt 限定符对表列设置加密。

语法

create table *tablename* (*colname* datatype [default_clause]
[encrypt [with [*database*.[*owner*].]*keyname*]])

keyname — 标识使用 create encryption key 创建的密钥。表的创建者必须对 *keyname* 具有 select 权限。如果未提供 *keyname*，Adaptive Server 将查找通过在 create encryption key 或 alter encryption key 中使用 as default 子句创建的缺省密钥。

示例

创建带加密的 employee 表：

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

1.24.3 *enable encrypted columns* 配置参数

必须将配置参数 enable encrypted columns 设置为 1 才能使用加密功能。如果在包含加密列的服务器中关闭此配置选项，则对这些列执行的任何命令都将失败，并出现错误。需要同时设置此配置参数和许可证选项，才能启用加密。

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

1.24.4 *load database*

如果装载数据库包含其它数据库中使用的加密密钥，则除非使用 with override，否则 load 将不会成功。

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

1.24.5 *select into*

使用 select into 需要拥有对源表的列级权限，包括 decrypt。

语法

使用以下语法对新表中的列加密：

```
select [all|distinct] < column_list>
        into target_table
        [(colname encrypt [with [database.[owner].]keyname]
        [,colname encrypt
        [with [database.[owner].]keyname]])]
    from tabname | viewname
```

示例

对表中的 creditcard 列加密。

```
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 包括以下附加一致性检查：

1 对于 *sysobjects* 中的每个加密密钥行，都将在 *sysencryptkeys* 中检查定义该密钥的行是否存在。

2 对于标记为加密的 *syscolumns* 中的每个列，都将在 *sysobjects* 和 *sysencryptkeys* 中检查密钥是否存在。

1.25 命令的完整语法

以下信息显示本公告中涉及的命令的完整语法。

1.25.1 *alter encryption key*

```
alter encryption key key1 as default | not default
```

1.25.2 alter table

```
alter table [[database.[owner].table_name
    { add column_name datatype
        [default {constant_expression | user | null}]
        {identity | null | not null}
        [off row | in row]
        [ [constraint constraint_name]
```

```
        { { unique | primary key }
              [clustered | nonclustered]
              [asc | desc]
              [with { fillfactor = pct,
                    max_rows_per_page = num_rows,
                    reservepagegap = num_pages }]
              [on segment_name]
        | references [[database.]owner.]ref_table
              [(ref_column)]
              [match full]
        | check (search_condition) ] ... }
        [encrypt [with [database .] owner ] .]keyname
        [, next_column]...
| add {[constraint constraint_name]
{  unique | primary key}
              [clustered | nonclustered]
              (column_name [asc | desc]
        [, column_name [asc | desc]...])
        [with { fillfactor = pct,
                    max_rows_per_page = num_rows,
                    reservepagegap = num_pages}]
        [on segment_name]
| foreign key (column_name [{, column_name}...])
        references [[database.]owner.]ref_table
        [(ref_column [{, ref_column}...])]
        [match full]
| check (search_condition)}
| drop {column_name [, column_name]...
        | constraint constraint_name }
| modify column_name datatype [null | not null]
              [encrypt | [with [database .]owner].] keyname
| modify column_name datatype [null | not null
              [encrypt | [with [database .]owner].] keyname]
                    |decrypt
              [, next_column]...
| replace column_name
        default { constant_expression | user | null}
        | partition number_of_partitions
| unpartition| { enable | disable } trigger
| lock {allpages | datarows | datapages } }
| with exp_row_size=num_bytes
| [ alter_partition_clause ]
[ partition_clause ]
```

1.25.3 *create table*

```
create table [database .[owner ].]table_name (column_name datatype)
    [default {constant_expression  | user | null}]
    {[{identity | null | not null}]
        [off row | [ in row [ (size_in_bytes) ] ]
    [[constraint constraint_name ]
        {{unique | primary key}
        [clustered | nonclustered] [asc | desc]
        [with { fillfactor = pct,
                max_rows_per_page = num_rows, }
                reservepagegap = num_pages }]
        [on segment_name]
        | references [[database .]owner .]ref_table
            [(ref_column )]
            [match full]
            | check (search_condition)}]}
            [match full]...
    [encrypt [with [[ database.] owner] . ] keyname]
    | [constraint constraint_name]
        {{unique | primary key}
            [clustered | nonclustered]
            (column_name [asc | desc]
                [{, column_name [asc | desc]}...])
            [with { fillfactor = pct
                max_rows_per_page = num_rows ,
                reservepagegap = num_pages } ]
            [on segment_name]
        |foreign key (column_name [{,column_name}...])
            references [[database.]owner.]ref_table
                [(ref_column [{, ref_column}...])]
                [match full]
        | check (search_condition) ... }
    [{, {next_column | next_constraint}}...])
    [lock {datarows | datapages | allpages }]
    [with { max_rows_per_page = num_rows,
            exp_row_size = num_bytes,
            reservepagegap = num_pages,
            identity_gap = value } ]
    [ table_lob_clause ]
    [ on segment_name ]
    [ [ external table ] at pathname ]
    [ partition_clause ]
```

1.25.4 *select*

```
into_clause ::=
    into [[database.]owner.]table_name
    [(colname encrypt [with [database.] owner].] keyname
                [, colname encrypt [with [database. ]owner] . ]
                    keyname])]
    [ lock {datarows | datapages | allpages } ]
    [ with into_option [, into_option] ...]

    into_option ::=
        | max_rows_per_page = num_rows
        | exp_row_size = num_bytes
        | reservepagegap = num_pages
        | identity_gap = gap
        [existing table table_name]
        [[external type] at ìpath_nameî
        [column delimiter delimiter]]
```

2. Internet 协议版本 6

若要使 Adaptive Server 可识别 IPv6，必须使用跟踪标志 7841 启动 Adaptive Server。通过这种方法， Adaptive Server 就可以确定 IPv6 的可用性，并可以识别 IPv6。

IPv6 可用于 Sun Solaris 32 位和 64 位平台。

- Sun Solaris 32 位和 64 位平台
- Windows
- HP 32 位和 64 位平台

有关如何在所用平台上设置和管理启用 IPv6 的网络的详细信息，请参见所用的操作系统文档。

3. 实时消息传送

Adaptive Server 12.5.3a 版的实时消息传送服务可同时提供对 TIBCO JMS 和 IBM WebSphere MQSeries 的支持。

4. 用于 AIX 的 64 位 Adaptive Server 中支持 PAM

用于 AIX 的 64 位 Adaptive Server 12.5.3a 版支持基于可插入鉴定模块的用户鉴定 (PAMUA)。尽管 12.5.3a 64 位 Adaptive Server 是在 AIX 5.1 上发布的，但 IBM 仅在用于 AIX 5.2 的 64 位应用程序上支持 PAM。Sybase 建议您升级到 AIX 5.2 以使用此功能。请与您的操作系统支持代表联系，以确保您的 IBM 主机上有可用于 PAM 的最新修补程序。

AIX 5.1 附带的 64 位 PAM 库不具有从 `/usr/lib/security/64` 自动装载 64 位库的能力。若要在 AIX5.1 上的 64 位 ASE 12.5.3a 中启用 PAMUA，需要在 */etc/pam.conf* 文件中提供 PAM 模块的完整路径名。以下示例说明如何在 AIX 5.1 计算机上指定操作系统模块 pam_aix。

需要 Adaptive Server 授权 `/usr/lib/security/64/pam_aix`

5. 功能表

此表显示了 Adaptive Server Enterprise 12.5.3a 版中可用于不同平台的各种功能。

图 3：表

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	*	*	*	*	*	*	*

- * 表示受支持的功能。空单元格表示该功能在对应平台上不可用。
- 高可用性支持表示主动 — 主动式支持。

  高可用性的主动 — 被动式支持仅限 Solaris/SPARC 平台。
- 选项可用性因版本的不同而异。有关不同版本间的差异的详细信息，请检查 Adaptive Server Enterprise 数据表。