



加密列用户指南

**Adaptive Server<sup>®</sup> Enterprise**

15.7

文档 ID: DC00545-01-1570-01

最后修订日期: 2011 年 9 月

版权所有 © 2011 by Sybase, Inc. 保留所有权利。

本出版物适用于 Sybase 软件 and 任何后续版本, 除非在新版本或技术声明中另有说明。此文档中的信息如有更改, 恕不另行通知。此处说明的软件按许可协议提供, 其使用和复制必须符合该协议的条款。

若要订购附加文档, 美国和加拿大的客户请拨打客户服务部门电话 (800) 685-8225 或发传真至 (617) 229-9845。

持有美国许可协议的其它国家/地区的客户可通过上述传真号码与客户服务部门联系。所有其他国际客户请与 Sybase 子公司或当地分销商联系。仅在定期安排的软件发布日期提供升级。未经 Sybase, Inc. 的事先书面许可, 不得以任何形式、任何手段 (电子的、机械的、手工的、光学的或其它手段) 复制、传播或翻译本出版物的任何部分。

Sybase 商标可在位于 <http://www.sybase.com/detail?id=1011207> 的“Sybase 商标页”(Sybase trademarks page) 处进行查看。Sybase 和列出的标记均是 Sybase, Inc. 的商标。® 表示已在美国注册。

SAP 和此处提及的其它 SAP 产品与服务及其各自的徽标是 SAP AG 在德国和世界各地其它几个国家/地区的商标或注册商标。

Java 和所有基于 Java 的标记都是 Sun Microsystems, Inc. 在美国和其它国家/地区的商标或注册商标。

Unicode 和 Unicode 徽标是 Unicode, Inc. 的注册商标。

IBM 和 Tivoli 是 International Business Machines Corporation 在美国和/或其它国家/地区的注册商标。

提到的所有其它公司和产品名均可能是与之相关的相应公司的商标。

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

# 目录

<b>第 1 章</b>	<b>加密概述 .....</b>	<b>1</b>
<b>第 2 章</b>	<b>创建和管理列加密密钥 .....</b>	<b>5</b>
	使用列加密密钥保护数据 .....	5
	创建列加密密钥 .....	6
	密钥保护 .....	10
	删除加密密钥 .....	12
	密钥加密 .....	13
	使用主密钥保护列加密密钥 .....	14
	使用系统加密口令保护列加密密钥 .....	15
	使用用户指定的口令保护列加密密钥 .....	16
	使用双控制保护列加密密钥 .....	16
<b>第 3 章</b>	<b>使用数据库级主密钥和双主密钥 .....</b>	<b>17</b>
	创建主密钥和双主密钥 .....	18
	创建主密钥副本 .....	19
	为主密钥和双主密钥设置口令 .....	20
	更改主密钥副本的口令和密钥加密密钥 .....	21
	重新生成主密钥 .....	22
	删除主密钥和密钥副本 .....	23
	恢复主密钥和双主密钥 .....	24
	在无人值守启动模式下启动 Adaptive Server .....	24
	配置无人值守启动模式 .....	24
	主密钥启动文件 .....	25
	Adaptive Server 如何使用主密钥启动文件 .....	25
<b>第 4 章</b>	<b>保护外部口令和隐藏文本 .....</b>	<b>27</b>
	服务密钥 .....	28
	创建服务密钥 .....	29
	删除服务密钥 .....	29
	修改服务密钥 .....	30
	对外部口令使用服务密钥 .....	32
	在升级期间对隐藏文本使用服务密钥 .....	34

	使用主密钥加密的服务密钥的相关注意事项 .....	34
<b>第 5 章</b>	<b>加密数据 .....</b>	<b>37</b>
	指定新表加密 .....	38
	指定 select into 加密 .....	39
	加密现有表中的数据 .....	40
	创建加密列的索引和约束 .....	41
	创建加密列的域和访问规则 .....	42
	decrypt 权限 .....	42
	撤消解密权限 .....	43
	限制解密权限 .....	44
	分配受限解密权限的特权 .....	44
	返回缺省值而非解密数据 .....	44
	定义解密缺省值 .....	45
	权限和解密缺省值 .....	46
	具有解密缺省值的列 .....	47
	解密缺省列和查询限定 .....	48
	decrypt default 和隐式授予 .....	49
	decrypt default 和 insert、update 以及 delete 语句 .....	50
	删除解密缺省值 .....	51
	加密列长度 .....	51
<b>第 6 章</b>	<b>访问加密数据 .....</b>	<b>55</b>
	处理加密列 .....	55
	解密权限 .....	56
	删除加密 .....	57
<b>第 7 章</b>	<b>防止管理员查看数据隐私 .....</b>	<b>59</b>
	密钥管理者角色 .....	59
	用户、角色和数据访问 .....	61
	使用用户指定的口令保护密钥 .....	62
	更改密钥的保护方法 .....	63
	创建密钥副本 .....	65
	更改密钥副本的口令 .....	67
	使用用户口令访问加密数据 .....	68
	使用密钥副本登录口令的应用程序透明度 .....	70
	登录口令更改和密钥副本 .....	73
	删除密钥副本 .....	73

---

<b>第 8 章</b>	<b>从丢失的口令恢复密钥 .....</b>	<b>75</b>
	密钥副本的口令丢失 .....	75
	登录口令丢失 .....	76
	基本密钥口令丢失 .....	76
	密钥恢复命令 .....	77
	更改加密密钥的所有权 .....	79
<b>第 9 章</b>	<b>审计加密列 .....</b>	<b>81</b>
	审计选项 .....	81
	审计值 .....	81
	事件名和编号 .....	81
	在命令文本审计中屏蔽口令 .....	82
	审计密钥管理者的操作 .....	82
<b>第 10 章</b>	<b>性能注意事项 .....</b>	<b>83</b>
	为加密列编制索引 .....	83
	排序顺序和加密列 .....	84
	加密列的连接 .....	84
	搜索参数和加密列 .....	85
	将加密数据作为密文移动 .....	86
<b>索引</b> .....		<b>87</b>



# 加密概述

本指南描述 Adaptive Server<sup>®</sup> Enterprise 的加密列功能、外部口令的加密以及存储过程、视图和触发器的 SQL 文本。

Adaptive Server 鉴定和访问控制机制可确保只有正确识别和授权的用户能够访问数据。数据加密进一步保护敏感数据，以防失窃或出现安全问题。

使用 Adaptive Server 加密列功能可以对当前未使用的列级别数据进行加密，而无需更改应用程序。这一原生支持具有以下功能：

- 列级别粒度
- 使用国家标准技术研究所 (NIST) 认可的对称算法：高级加密标准 (AES)
- 性能优化
- 强制职责分离
- 完全集成并可自动执行主要管理功能
- 应用程序透明度：不需要更改应用程序
- 保护数据隐私，以防系统管理员访问私有数据

数据加密和解密以自动、透明的方式进行。如果您对某个表具有插入或更新权限，则会自动加密插入或修改的任何数据，然后存储这些数据。日常任务不会被中断。

除了需要 `select` 权限外，从加密列选择解密数据还需要 `decrypt` 权限。可以向特定数据库用户、组或角色授予 `Decrypt` 权限。Sybase<sup>®</sup> 通过提供对敏感数据的细化访问功能，为您提供了更多控制权。Sybase 还为具有 `decrypt` 权限的用户自动解密选定的数据。

与在中间层或客户端应用程序中使用加密相比，在 Adaptive Server 中加密列更为简便。您可以使用 SQL 语句来创建加密密钥并指定要加密的列，并且现有应用程序将继续运行而不会发生变化。

列加密密钥以加密形式存储在数据库中。您可以使用从以下项派生的密钥加密密钥 (KEK) 对列加密密钥进行加密：

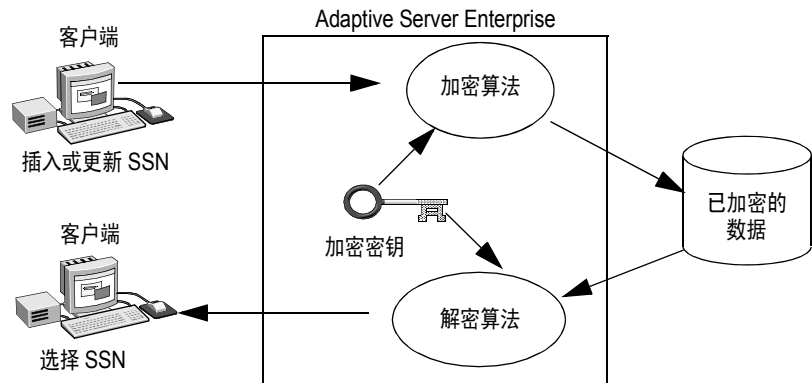
- 用户提供的系统级口令
- 从用户提供的口令（可以是用户的登录口令）派生的 KEK
- 单独创建的数据库级 KEK（主密钥或双主密钥）

您选择的口令将反映您保护数据隐私的能力（即使管理员也无法访问数据）。您可以选择使用双控制模式来保护您的列加密密钥，以提高安全性。请参见第 2 章“创建和管理列加密密钥”。

加密数据时，数据以称为“密文”的编码形式存储。密文会使加密列额外增加几个字节到 32 个字节的长度。请参见第 51 页的“加密列长度”。未加密的数据作为明文进行存储。

图 1-1 介绍了 Adaptive Server 中的加密和解密处理。在此示例中，客户端对社会保险号 (SSN) 进行更新和加密。

**图 1-1: Adaptive Server 中的加密和解密**



列加密使用对称加密算法，这意味着使用相同的密钥进行加密和解密。Adaptive Server 可跟踪加密给定列的密钥。



使用 `insert` 或 `update` 在加密列中插入或更新数据时，Adaptive Server 会在写入行之前透明地加密数据。使用 `select` 从加密列中选择数据时，Adaptive Server 在从行中读取数据后将其解密。对于所有平台，将按以下形式加密整数和浮点数据：

- 用于整数数据的最高有效位格式
- 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

可以在一个平台上加密数据，然后在其它平台上对该数据进行解密，但前提是两个平台使用相同的字符集。

通常，使用加密列需要执行以下步骤：

- 1 安装许可证选项 ASE\_ENCRYPTION。请参见 Adaptive Server Enterprise Installation Guide（《Adaptive Server Enterprise 安装指南》）。
- 2 系统安全员 (SSO) 在 Adaptive Server 中启用加密功能：

```
sp_configure 'enable encrypted columns', 1
```
- 3 根据您选择保护列加密密钥的方法，创建数据库级主密钥或设置系统加密口令。有关创建主密钥的信息，请参见第 18 页的“[创建主密钥和双主密钥](#)”。  
第 15 页的“[使用系统加密口令保护列加密密钥](#)”描述如何设置系统加密口令。
- 4 创建一个或多个命名加密密钥。请参见第 2 章“[创建和管理列加密密钥](#)”考虑使用口令来保护数据，以使数据甚至无法被数据库管理员访问。请参见第 7 章“[防止管理员查看数据隐私](#)”。
- 5 指定要加密的列。请参见第 38 页的“[指定新表加密](#)”和第 40 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 `decrypt` 权限。您可能会选择指定称为“解密缺省值”的缺省纯文本值。Adaptive Server 向没有解密权限的用户返回此缺省值，而不是受保护的数据。请参见第 56 页的“[解密权限](#)”。

---

执行这些步骤后，就可以立即针对现有表和列运行现有应用程序，但现在数据库中的数据已受到严格保护，以防失窃和误用。Adaptive Server 实用程序和其它 Sybase 产品可以采用加密形式处理数据，在整个企业内保护您的数据。例如，可以：

- 使用 Sybase Control Center 或 Sybase Central Adaptive Server 插件通过图形界面管理加密列。请参见联机帮助。
- 使用批量复制实用程序 (bcp) 安全地将加密数据从服务器拷入或拷出。请参见《实用程序指南》。
- 使用 Adaptive Server 迁移工具 sybmigrate 安全地将数据从一台服务器迁移到另一台服务器。请参见《Adaptive Server Enterprise 系统管理指南》。
- 使用 Sybase Replication Server 跨服务器和平台安全地分发加密密钥和数据。有关在复制时进行加密的信息，请参见《Replication Server 管理指南》。

有关对外部口令和隐藏的 SQL 文本进行强加密的信息，请参见[第 4 章“保护外部口令和隐藏文本”](#)。

## 创建和管理列加密密钥

主题	页码
<a href="#">使用列加密密钥保护数据</a>	5
<a href="#">密钥加密</a>	13

Adaptive Server 包含用于创建列加密密钥、更改列加密密钥的属性以及删除未使用的列加密密钥的命令。密钥所有者必须将权限授予表所有者，后者才能使用单个或多个特定密钥在列级配置加密。

### 使用列加密密钥保护数据

在密钥未被使用时，Adaptive Server 会保持密钥的加密状态。用户必须首先对列加密密钥 (CEK) 具有访问权限，然后才能访问加密数据；但必须先加密 CEK，然后才能将其存储在磁盘上或内存中。

Adaptive Server 使用密钥加密密钥 (KEK) 对 CEK 进行加密，并将其以加密形式存储在 `sysencryptkeys` 中。KEK 还用于解密 CEK，从而使您可以访问解密数据。请参见第 13 页的“密钥加密”。

CEK 管理包括创建、删除和修改列加密密钥，分发口令，创建密钥副本，以及在丢失口令时提供密钥恢复。

## 创建列加密密钥

在表所有者可以在新表或现有表中标记要加密的列之前，列加密密钥必须已存在。初次设置密钥时，请考虑：

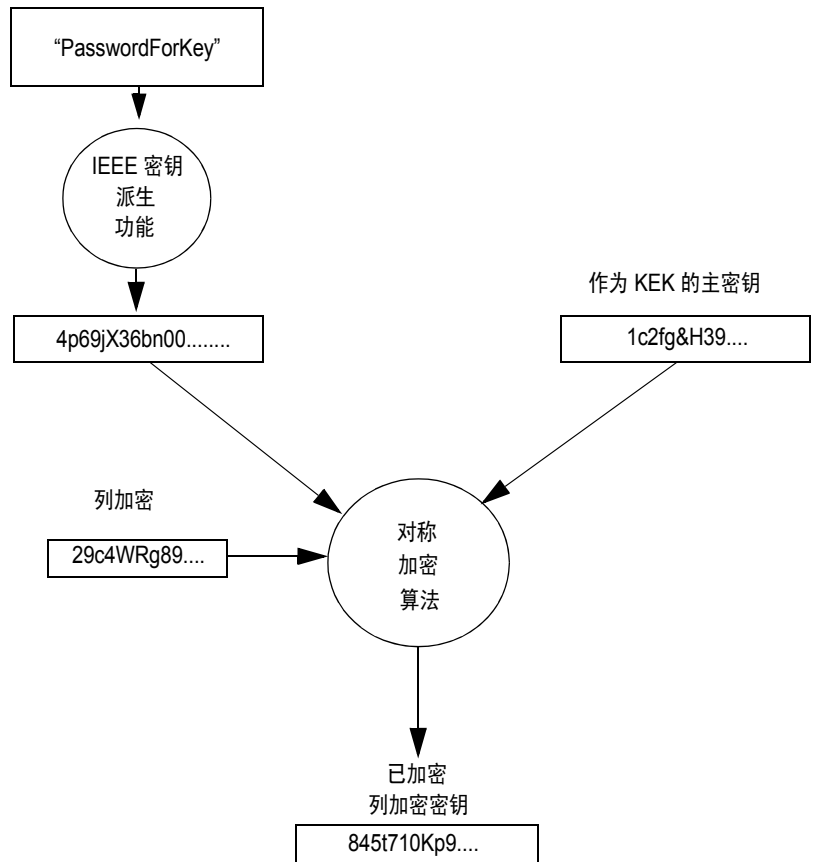
- 密钥所有者或管理者的指派 — 必须由系统安全员 (SSO) 授予 `create encryption key` 权限才能创建密钥。缺省情况下，`sso_role` 和 `keycustodian_role` 具有 `create encryption key` 权限。请参见第 59 页的“[密钥管理者角色](#)”。
- 是否应在单独的密钥数据库中创建密钥 — Sybase 建议您使用单独的数据库存储密钥，尤其是在使用系统加密口令对密钥进行加密时。
- 所需密钥数目 — 可以为每个加密列创建单独的密钥，也可以使用同一密钥跨多个表来加密列。从性能角度而言，通过不同表中的等值列连接的加密列应共享同一密钥。为了安全起见，不相关的列应使用不同的密钥。

Adaptive Server 中的列加密使用高级加密标准 (AES) 对称密钥加密算法，其可用密钥大小为 128、192 和 256 位。随机密钥生成和加密功能是由符合 FIPS 140-2 标准的模块提供的。

为对密钥值提供安全保护，Adaptive Server 使用从系统加密口令或用户指定口令派生的 256 位密钥加密密钥 (KEK)（可能是主密钥或内部密钥）。请参见第 3 章“[使用数据库级主密钥和双主密钥](#)”。Adaptive Server 可对新密钥（列加密密钥）进行加密并将结果存储在 `sysencryptkeys` 中。

图 2-1：使用 KEK 对列加密密钥进行加密

从口令中派生的 KEK



缺省情况下，Adaptive Server 创建 256 位密钥加密密钥。为了与 15.7 之前的版本兼容，它使用 128 位密钥（如果 KEK 派生自系统加密口令）。

创建列加密密钥的语法

创建列加密密钥的语法为：

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

其中：

- *keyname* — 在当前数据库中的用户表、视图和过程名称空间中必须是唯一的。如果该密钥位于另一数据库中，请指定数据库名；如果数据库中有多个具有该名称的密钥，请指定所有者的名称。*owner* 的缺省值是当前用户，而 *database* 的缺省值是当前数据库。只有系统安全员才能为其他用户创建密钥。

---

**注释** 不能创建以 “#” 开头的临时密钥名称。

---

- *as default* — 允许系统安全员或密钥管理者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。
- *for 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、192 和 256 位，支持的块大小为 16 个字节。在加密单元中，块大小即为字节数。大型数据将拆分以进行加密。
- *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 *keylength* 为 128 位。
- *passwd password\_phrase* — 指示 ASE 使用用户口令 *password\_phrase*（可以是带引号的字母数字字符串，其长度最多为 255 个字节）保护 CEK。
- *passwd system\_encr\_passwd* — 指示 ASE 使用系统加密口令保护 CEK。
- *master key* — 指示 ASE 使用主密钥保护 CEK。缺省情况下，Adaptive Server 使用主密钥（如果存在）保护列加密密钥。请参见第 10 页的“密钥保护”。
- *init\_vector*
  - *random* — 指定在加密过程中使用初始化矢量。如果加密算法使用初始化矢量，则两个相同的明文部分的密文是不同的，这可防止检测数据模式。通过使用初始化矢量，可以提高数据的安全性。

使用初始化矢量意味着使用密码块链 (CBC) 加密模式，在此模式下，每个数据块在加密前与上一个块合并在一起，而第一个块与初始化矢量合并在一起。

但是，初始化矢量会对性能造成一定的影响。对于加密密钥没有指定初始化矢量的列，您可以仅为其创建索引并优化连接和搜索。请参见第 10 章“性能注意事项”。

- **null** — 在加密时不使用初始化矢量。这样可使列支持索引。

缺省值是使用初始化矢量，即 `init_vector random`。

设置 `init_vector null` 意味着使用电码本 (ECB) 模式（在此模式下，将单独加密每个数据块）。

要使用初始化矢量加密一个列，而不使用初始化矢量加密另一个列，请创建两个不同的密钥：一个密钥指定使用初始化矢量；另一个密钥指定不使用初始化矢量。

- **pad**

- **null** — 缺省值，它将忽略数据随机填充。

如果列必须支持索引，则不能使用填充。

- **random** — 在加密前，自动使用随机字节来填充数据。通过使用填充代替初始化矢量，可以使密文随机化。填充仅适用于明文长度小于块长度一半的列。对于 AES 算法，块的长度为 16 字节。

- **dual control** — 指示是否必须使用双控制加密新密钥。缺省情况下不配置双控制。请参见第 3 章 “使用数据库级主密钥和双主密钥”。

#### create encryption key 示例

以下示例在创建列加密密钥时使用各种加密属性，并且许多示例假定您已创建主密钥或已设置系统加密口令（请参见第 10 页的“密钥保护”）。

- 示例 1 — 将名为 “safe\_key” 的 256 位密钥指定为数据库缺省密钥。由于该密钥不指定口令，因此，Adaptive Server 使用数据库级主密钥作为 safe\_key 的 KEK。如果没有主密钥，Adaptive Server 会使用系统加密口令：

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

只有系统安全员或具有 `keycustodian_role` 的用户才能创建缺省密钥。

- 示例 2 — 创建一个名为 “salary\_key” 的 128 位密钥，以使用随机填充来加密列：

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

- 示例 3 — 创建一个名为 “mykey” 的 192 位密钥，以使用初始化矢量来加密列：

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

- 示例 4 — 创建一个受用户指定口令保护的密钥：

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

如果密钥受用户指定的口令保护，则必须先输入该口令，然后才能访问已使用该密钥加密的列。请参见第 7 章“防止管理员查看数据隐私”，以了解有关将密钥与显式口令一起使用的信息。

- 示例 5 — 创建一个受双控制保护的密钥：

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

密钥“dualprotectedkey”受主密钥和用户口令（在双控制中）保护。若要访问该密钥，必须同时输入该密钥的用户口令和主密钥的口令。

*create encryption key*  
权限

sso\_role 和 keycustodian\_role 隐式具有创建加密密钥的权限。系统安全员或密钥管理者使用以下语法将 create encryption key 权限授予其他用户：

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

例如：

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

若要撤消密钥创建权限，请使用：

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

---

**注释** grant all 不会将 create encryption key 权限授予用户。该权限必须由系统安全员显式授予。

---

## 密钥保护

密钥管理员必须决定存储密钥的位置、应该续订密钥的时间，以及哪些表所有者可以使用给定密钥加密其表中的列。



## 授予对密钥的访问权

在其他用户可以在 `create table`、`alter table` 和 `select into` 语句中指定密钥之前，密钥所有者或具有 `sso_role` 的用户必须授予对密钥的 `select` 权限。密钥所有者可以是系统安全员或密钥管理者，对于非缺省密钥，密钥所有者也可以是具有 `create encryption key` 权限的任何用户。密钥所有者应根据需要授予对密钥的 `select` 权限。

在以下示例中，具有 `db_admin_role` 的用户可以在 `create table`、`alter table` 和 `select into` 语句中指定加密时使用名为 “`safe_key`” 的加密密钥：

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

---

**注释** 通过 `insert`、`update`、`delete` 和 `select` 处理加密列的用户不需要加密密钥的 `select` 权限。

---

## 更改密钥

定期更改用于对列进行加密的密钥。可使用 `create encryption key` 创建一个新密钥，然后使用 `alter table...modify` 通过新密钥来加密列。

以下示例假定 “`creditcard`” 列已经过加密。`alter table` 命令使用 `cc_key_new` 对每行中客户的信用卡值进行解密和重新加密。

```
create encryption key cc_key_new for AES
```

```
alter table customer modify creditcard encrypt with  
cc_key_new
```

## 将密钥与数据分开

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。

使用不同数据库中的密钥进行加密具有安全优势，这是因为一旦数据库转储失窃，这可防止同时访问密钥和加密数据。管理员也可以使用不同的口令来保护每个数据库转储，从而使未经授权的访问变得更加困难。

使用不同数据库中的密钥进行加密时需要特别小心，以避免在分布式系统中出现数据和密钥完整性问题。应仔细协调数据库转储和装载。

Sybase 建议，如果使用不同数据库中的命名密钥，则：

- 在转储包含加密列的数据库时，也应转储在其中创建密钥的数据库。如果新密钥是在上次转储后添加的，则必须执行此操作。
- 在转储包含加密密钥的数据库时，应转储所有包含使用该密钥加密的列的数据库。这使加密数据与可用密钥保持同步。

系统安全员或密钥管理者可以使用 `sp_encryption` 标识使用给定密钥加密的列。

## 删除加密密钥

要删除加密密钥，请使用：

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

例如，以下语句将删除名为 `cc_key` 的加密密钥：

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

密钥所有者可以删除自己的密钥。系统安全员可以删除任何密钥。仅当任何数据库中没有使用某个密钥的加密列时，才能删除该密钥。

执行 `drop encryption key` 时，Adaptive Server 不会检查处于可疑、已存档、脱机、未恢复状态或当前正在装载的数据库中是否存在加密列。在上述任何一种情况下，该命令都会发出一条警告消息，指出不可用数据库的名称，但该命令不会失败。在该数据库联机后，包含使用已删除密钥加密的列的表都将无法使用。若要恢复密钥，系统管理员必须装载包含已删除密钥的数据库的转储（该转储是在删除此密钥之前生成的）。

系统安全员可以使用 `sp_encryption` 标识使用给定密钥加密的所有列。

## 密钥加密

在用户和数据之间实际存在两个密钥：列加密密钥 (CEK) 和密钥加密密钥 (KEK)。CEK 可对数据进行加密，用户必须对 CEK 具有访问权限，然后才能访问加密数据。不能以未加密的形式将该密钥存储在磁盘上。相反，当您创建或更改加密密钥时，Adaptive Server 会使用 1 个 KEK 或 2 个 KEK（在双控制中）来加密 CEK。在您可以访问解密数据之前，KEK 还会解密 CEK。CEK 以加密形式存储在 `sysencryptkeys` 中。

KEK 是由系统安全员或密钥管理者单独创建的主密钥，它是从系统加密口令、用户指定的口令或登录口令内部派生的，具体取决于您如何使用 `create encryption key` 和 `alter encryption key` 语句指定该密钥的加密。系统加密口令和主密钥均以加密形式存储。

第 13 页的图 2-2 说明了如何为 `create encryption key` 语句创建和存储列加密密钥。KEK 派生自口令，KEK 和原始 CEK 将提供给加密函数以生成加密 CEK。

图 2-2: 创建加密密钥的步骤

创建加密密钥 ...

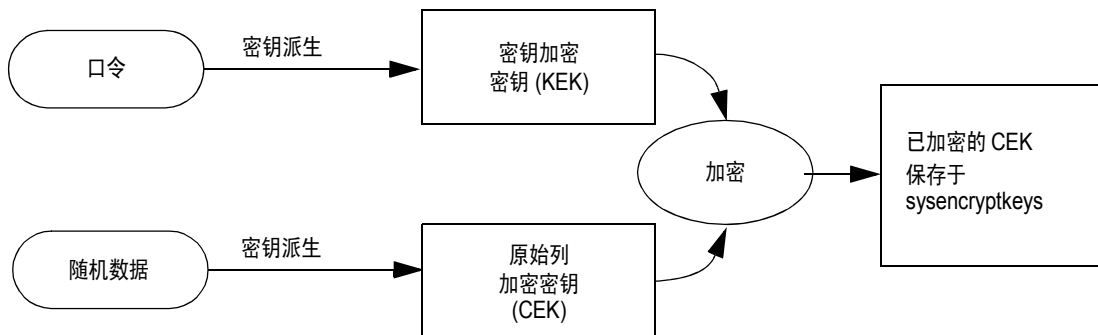
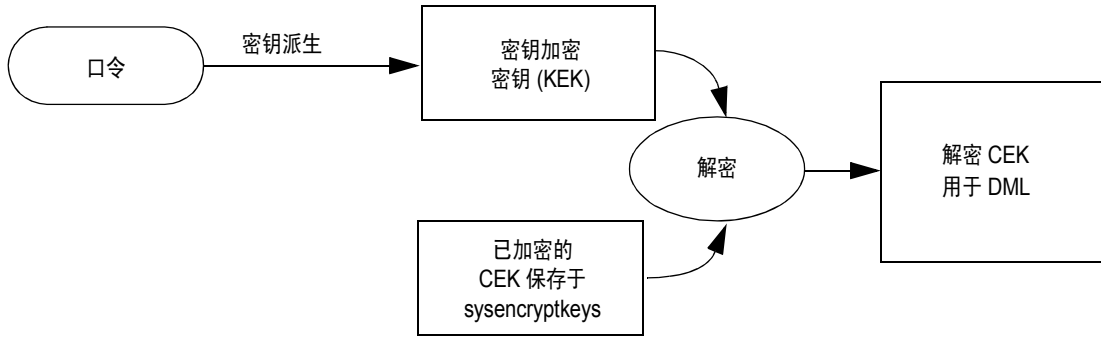


图 2-3 说明了如何在 DML 操作期间使用 KEK 来解密 CEK。然后，原始 CEK 会用于加密或解密数据。

图 2-3：在 DML 语句中访问 CEK 以加密或解密



## 使用主密钥保护列加密密钥

主密钥是具有 `sso_role` 或 `keycustodian_role` 的用户创建的数据库级密钥，可用作用户创建的加密密钥的 KEK。创建主密钥后，它将替代系统加密口令作为用户创建的密钥的缺省 KEK。虽然 Adaptive Server 支持系统加密口令以便与 15.7 之前的版本兼容，但 Sybase 建议您使用主密钥。

您可以将主密钥与双主密钥一起使用，以便为用户创建的所有密钥创建可提供双控制和分离知识的组合密钥。您还可以通过使用包含 CEK 显式口令的主密钥创建组合密钥。

由于以下原因，使用主密钥可简化对加密数据的管理：

- 管理密钥口令只限于设置主密钥的口令。
- 您无需在 `create encryption key` 和 `alter encryption key` 语句中指定口令。
- 允许从丢失的列加密密钥口令分发和恢复口令。
- 对加密数据的访问控制是通过列的解密权限强制实施的。请参见第 44 页的“限制解密权限”。
- 不需要对应用程序进行任何更改。

创建主密钥的语法为：

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

请参见《参考手册：命令》。

## 使用系统加密口令保护列加密密钥

系统加密口令是特定于数据库的口令，并且是 CEK 的辅助缺省加密方法。也就是说，您不需要为数据库创建主密钥。Adaptive Server 使用系统加密口令加密在指定数据库中创建的密钥，而无需显式口令子句。在系统安全员或密钥管理者设置了系统加密口令之后，您无需指定该口令便可处理加密列。Adaptive Server 在需要对列加密密钥进行加密或解密时会在内部访问系统加密口令。

系统安全员或密钥管理者可使用 `sp_encryption` 设置系统加密口令。系统口令针对的是使用 `sp_encryption` 的数据库，其加密值存储在该数据库的 `sysattributes` 系统表中。

```
sp_encryption system_encr_passwd, password
```

`password` 的长度最多可以为 255 个字节。

仅在创建加密密钥的数据库中设置系统加密口令。

系统加密口令可保护您的加密密钥。请选择较长且复杂的系统加密口令。较长的口令也较难通过穷举法猜到或破译。请在系统加密口令中包含大小写字母、数字和特殊字符。Sybase 建议，系统加密口令的长度应至少为 16 个字节。

Adaptive Server 会强制系统加密口令符合 `minimum password length` 和 `check password for digit` 配置参数的要求。

通过使用 `sp_encryption` 并提供旧口令可更改系统口令：

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

需定期更改系统加密口令，尤其是在了解系统加密口令的管理人员离开公司的情况下。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。加密数据在系统口令发生更改时不受影响，换句话说，它不会被解密和重新加密。

通过提供“null”作为 `password` 的参数并提供 `old_password` 的值，可取消设置系统加密口令。仅当删除了该数据库中已使用系统加密口令加密的所有加密密钥后，才能取消设置系统口令。

## 使用用户指定的口令保护列加密密钥

当您使用 `create encryption key` 或 `alter encryption key` 对密钥指定口令时，可以将系统管理员或数据库所有者的权限限制为只能访问私有数据。如果密钥具有显式口令，则用户必须具有以下两项才能解密数据：

- 列的 `decrypt` 权限
- 加密密钥的口令

用户还必须知道口令，才能运行对数据进行加密的 DML 命令。

请参见第 62 页的 [“使用用户指定的口令保护密钥”](#)。

## 使用双控制保护列加密密钥

您可以使用 `create encryption key` 命令通过双控制来保护列加密密钥。

如果您指定 `create encryption key with dual_control`，但不指定用户口令，则列加密密钥受主密钥和双主密钥保护。

如果您指定 `with dual_control` 并包括特定于用户的口令，则列加密密钥受主密钥和用户口令保护。

- 示例 1 - 同时使用主密钥和双主密钥保护 CEK “Reallysecret” 并失败，除非这两个密钥均位于数据库中：

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

- 示例 2 - 同时使用主密钥和用户口令 “Whybother” 加密 CEK “k3”：

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

若要将现有密钥更改为使用双控制，请参见第 63 页的 [“更改密钥的保护方法”](#)。

## 使用数据库级主密钥和双主密钥

主题	页码
<a href="#">创建主密钥和双主密钥</a>	18
<a href="#">为主密钥和双主密钥设置口令</a>	20
<a href="#">更改主密钥副本的口令和密钥加密密钥</a>	21
<a href="#">重新生成主密钥</a>	22
<a href="#">删除主密钥和密钥副本</a>	23
<a href="#">恢复主密钥和双主密钥</a>	24
<a href="#">在无人值守启动模式下启动 Adaptive Server</a>	24

Adaptive Server 允许用户创建称为主密钥和双主密钥的数据库级别的加密密钥。这两个密钥均用作密钥加密密钥，并用于保护其它密钥，例如列加密密钥和服务密钥。创建主密钥后，这些密钥即会成为列加密密钥的缺省保护方法。只有列加密密钥的双控制才需要双主密钥。

只有具有 `sso_role` 或 `keycustodian_role` 的用户才能创建主密钥和双主密钥。每个数据库只能有一个主密钥和一个双主密钥。

主密钥和双主密钥必须具有不同的所有者。您可以使用 `isql` 或通过只有通过 Adaptive Server 才能访问的服务器专用文件为主密钥提供口令。这些密钥的口令不存储在数据库中。

## 创建主密钥和双主密钥

若要创建主密钥和双主密钥，请使用以下命令：

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

其中：

- **master** 和 **dual master** 是指用于在定义其它密钥的数据库中加密这些密钥的数据库级密钥。这些密钥不用于加密数据。主密钥在内部指定为 **sysobjects** 中的 **sybencrmasterkey**，而双主密钥指定为 **sysobjects** 中的 **sybencrdualmasterkey**。
- **with passwd** 必须后跟一个遵循 **sp\_passwordpolicy** 的字符串口令。

请参见《参考手册：命令》。

- 示例 1 — 在数据库 **tdb1** 中创建主密钥：

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

- 示例 2 — 在数据库 **tdb1** 中创建双主密钥：

```
use database tdb1
create encryption key dual master with passwd
'dualunforgettable'
```

- 示例 3 — 因无法将主密钥用作列加密密钥而生成错误：

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

若要更改主密钥或双主密钥的口令，请使用以下命令：

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



## 创建主密钥副本

具有 `sso_role` 或 `keycustodian_role` 的用户或主密钥所有者可以为该主密钥创建副本，为此，您可能需要：

- 提供对主密钥或双主密钥的访问权，以便以无人值守的方式启动 Adaptive Server。这种密钥副本称为 `automatic_startup` 副本。
- 支持在其口令丢失时恢复主密钥。这种密钥副本称为恢复副本。请参见第8章“从丢失的口令恢复密钥”。
- 允许基本密钥所有者以外的用户为主密钥或双主密钥设置加密口令。这种密钥副本称为常规副本。

若要在数据库中添加主密钥副本，请使用以下命令：

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

其中：

- *char\_string* —（第一个引用）指定当前加密主密钥或双主密钥的基本副本的口令。
- *char\_string* —（第二个引用）指定常规副本或恢复副本的口令。不能将其用于 `automatic_startup` 副本。
- *for user* — 表示必须向其分配常规副本或恢复副本的用户。不要使用此参数为 `automatic_startup` 副本输入口令。
- *for recovery* — 表示将在口令丢失时使用密钥副本恢复主密钥。
- *for automatic\_startup* — 表示将在重新启动服务器后且启用 `automatic master key access` 的情况下使用密钥副本访问主密钥或双主密钥。
- 示例 1 — 主密钥所有者为 Mary 创建密钥副本：

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

- 示例 2 — 双主密钥所有者 Smith 使用以下命令为 `automatic_startup` 创建密钥副本：

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

## 为主密钥和双主密钥设置口令

基本密钥所有者（即，拥有常规密钥副本的用户）可以为主密钥或双主密钥设置口令。必须先设置口令，然后才能使用主密钥。若要为主密钥设置口令，可以使用以下任一方法：

- `set encryption passwd` 命令
- 使用无人值守启动功能
- （仅主密钥）`dataserver` 命令

`set encryption` 命令为：

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

其中：

- *char\_literal* — 如果用户是密钥所有者，则为当前加密主密钥或双主密钥的基本副本的口令。如果用户不是密钥所有者，则为当前加密用户的密钥副本的口令。

示例 — 在数据库 `tdb1` 中为主密钥设置口令 “MasterSecret”：

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

Adaptive Server 会在在其中定义主密钥或双主密钥的数据库的服务器内存中设置口令，还会记录设置该口令的用户的身份。设置该口令后，即可将其用于对数据库中主密钥的所有访问。

## 更改主密钥副本的口令和密钥加密密钥

拥有主密钥副本的用户可以使用以下命令更改其密钥副本的口令：

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

其中：

- *char\_string* —（第一个引用）如果用户是密钥所有者，则为当前加密主密钥或双主密钥的基本副本的口令。如果用户不是密钥所有者，则为当前加密用户的密钥副本的口令。
- *char\_string* —（第二个引用）指定常规副本或恢复副本的新口令。不要使用此参数为 *automatic\_startup* 副本输入口令。
- *for automatic\_startup* — 生成一个新 KEK 并用其创建新的 *automatic\_startup* 密钥副本。

如果未指定 *for recovery* 或 *for automatic startup*，且命令由密钥所有者发出，则 Adaptive Server 会更改基本密钥副本的口令。如果命令不是由密钥所有者发出的，则 Adaptive Server 仅在当前用户具有 *sso\_role* 或 *keycustodian\_role* 时，才会更改基本密钥副本的口令。

- 示例 1 — 主密钥所有者 “Jones” 使用以下命令为 “Mary” 创建密钥副本：

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

- 示例 2 — “Mary” 使用以下命令更改她的副本的口令：

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

- 示例 3 — 主密钥所有者 “John” 使用以下命令更改基本密钥的口令：

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

具有 `sso_role` 或 `keycustodian_role` 的用户可以修改 `automatic_startup` 密钥副本，以更改其密钥加密密钥。例如，了解主密钥口令的用户可以使用以下命令更改 `automatic_startup` 密钥副本的密钥加密密钥：

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

Adaptive Server:

- 使用从口令派生的密钥加密密钥对基本主密钥进行解密。
- 创建新的主密钥加密密钥，并使用此新密钥替换主密钥启动文件中的旧密钥。
- 通过使用新主密钥加密密钥对主密钥进行加密，创建一个新的 `automatic_startup` 密钥副本，并使用此新副本替换 `sysencryptkeys` 中的旧 `automatic_startup` 密钥副本。

## 重新生成主密钥

可定期更改主密钥和双主密钥。但是，每次更改主密钥和双主密钥时，还必须使用新的主密钥和双主密钥对所有列加密密钥进行加密。为自动完成此过程，Adaptive Server 会使用 `regenerate key` 选项（该选项可使用新值替换主密钥或双主密钥的值），并重新加密当前通过要重新生成的主密钥或双主密钥进行加密的所有列加密密钥：

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

在执行 `regenerate key` 命令时，Adaptive Server 将：

- 验证提供的口令是否对基本主密钥或双主密钥进行解密。
- 创建新的主密钥或双主密钥。
- 对由主密钥或双主密钥完全或部分加密的所有列加密密钥进行解密。Adaptive Server 会使用新的主密钥或双主密钥对其进行重新加密。
- 使用由第二口令加密的新密钥替换基本主密钥或双主密钥。如果未提供第二口令，Adaptive Server 将使用当前配置的口令对新密钥进行加密。
- 删除常规密钥副本。主密钥所有者必须使用 `alter encryption key` 为指定用户重新创建常规密钥副本。

- 删除密钥恢复副本。主密钥所有者必须使用 `alter encryption key` 添加新的恢复密钥副本，并向恢复密钥所有者告知新口令。
- 使用新密钥副本（通过使用随机生成的新主密钥加密密钥对新主密钥进行加密而创建）替换 `automatic_startup` 副本。Adaptive Server 会将新的主密钥加密密钥写入主密钥启动文件。

## 删除主密钥和密钥副本

具有 `sso_role` 或 `keycustodian_role` 的用户可以删除主密钥或双主密钥，前提条件是没有当前使用该主密钥或双主密钥进行加密的任何其它列加密密钥。使用：

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

删除主密钥或双主密钥时，Adaptive Server 将：

- 删除该主密钥或双主密钥及其密钥副本。从数据库中删除所有常规密钥副本、`automatic_startup` 密钥副本和恢复密钥副本。
- 从主密钥启动文件中删除主密钥加密密钥（如果 `automatic_startup` 密钥副本存在）。

若要只删除常规密钥副本，请使用：

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

若要只删除恢复密钥副本，请使用：

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

若要只删除 `automatic_startup` 密钥副本，请使用：

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

## 恢复主密钥和双主密钥

具有 `sso_role` 或 `keycustodian_role` 的用户可以使用以下命令恢复主密钥或双主密钥：

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

其中，对 `passwd` 的第一个引用是用于恢复密钥副本的口令，而对 `passwd` 的第二个引用是基本密钥的新口令。

## 在无人值守启动模式下启动 Adaptive Server

使用无人值守启动模式可以在联系不到口令所有者时允许访问主密钥。

通常情况下，使用无人值守启动模式需要执行以下操作：

- 1 启用 `automatic master key access` 配置参数。
- 2 （可选）设置主密钥启动文件的路径和名称。如果不进行设置，Adaptive Server 将使用缺省文件路径和名称。
- 3 为打算以无人值守方式启动的数据库的主密钥或双主密钥添加 `automatic_startup` 副本。

## 配置无人值守启动模式

具有 `sso_role` 的用户可以通过设置以下内容将 Adaptive Server 配置为使用无人值守启动模式：

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

在此模式下，Adaptive Server 可通过读取和加密主密钥启动文件中的主密钥加密密钥来访问主密钥，即使其口令不在服务器内存中也是如此。若要使用无人值守启动模式，您还必须为数据库中的 `master key` 和 `dual master key` 创建 `automatic_startup` 密钥副本。

## 主密钥启动文件

启用 `automatic master key access` 时，Adaptive Server 会读取主密钥启动文件（如果存在）中的密钥加密密钥。如果它不存在，则 Adaptive Server 会创建主密钥启动文件，但不会将密钥加密密钥值写入文件，直到创建主密钥或双主密钥的 `automatic_startup` 密钥副本。

禁用 `automatic master key access` 时，Adaptive Server 会从服务器内存中删除主密钥和双主密钥的密钥加密密钥。Adaptive Server 不会从主密钥启动文件中清除密钥加密密钥值。

### 创建主密钥启动文件

具有 `sso_role` 的用户可以使用以下命令指定主密钥启动文件的路径和名称：

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

其中：

- `new_path` — 指定主密钥启动文件的位置和名称。Adaptive Server Cluster Edition 独立安装不支持 `new_path`。
- `default location` — 将主密钥启动文件设置为缺省路径和名称：  
`$$SYBASE_ASE/security/ase_encrcols_mk_<servername>.dat`。  
Adaptive Server Cluster Edition 独立安装不支持 `default location`。
- `null` — 显示当前主密钥启动文件的路径和名称。
- `sync_with_mem` — 如果启用配置选项 `automatic master key access`，则将服务器内存中的主密钥加密密钥写入主密钥启动文件。Adaptive Server Cluster Edition 独立安装不支持 `sync_with_mem`。
- `sync_with_qrm` — （仅适用于 Cluster Edition 独立安装）使用仲裁设备中的副本更新本地主密钥启动文件中的密钥副本。

## Adaptive Server 如何使用主密钥启动文件

如果出现以下情况，Adaptive Server 会将主密钥启动文件中的主密钥和双主密钥加密密钥读入服务器内存：

- 服务器启动时 `automatic master key access` 处于启用状态，或者
- 服务器运行时 `automatic master key access` 处于启用状态。

如果：

- 创建了主密钥或双主密钥的 `automatic_startup` 密钥副本，则 Adaptive Server 会将主密钥或双主密钥加密密钥写入该文件。
- 更改了主密钥或双主密钥的 `automatic_startup` 密钥副本的密钥加密密钥，则 Adaptive Server 会将新的主密钥或双主密钥加密密钥写入该文件。
- 删除了 `automatic_startup` 密钥副本， Adaptive Server 将删除该文件中的相应记录。
- 删除了数据库， Adaptive Server 将删除属于已删除数据库的所有记录。
- 删除了主密钥或双主密钥， Adaptive Server 将删除相应记录。
- 使用 `sp_encryption mkey_startup_file` 指定新的主密钥启动文件， Adaptive Server 会将服务器内存与新文件的内容进行同步。

如果主密钥加密密钥位于内存中，可通过 `automatic_startup` 副本访问该主密钥，即使未设置主密钥口令也是如此。



# 保护外部口令和隐藏文本

主题	页码
服务密钥	28
使用主密钥加密的服务密钥的相关注意事项	34

Adaptive Server 可使用 AES-256 对称加密算法对外部登录口令和隐藏文本进行强加密。可选择对以下对象的外部口令进行强加密：

- Replication Agent — 复制型数据库。
- CIS — 远程描述符和登录名。
- Job Scheduler — Job Scheduler 代理。
- RTMS — 实时消息传送。
- 安全套接字层 (SSL) 和轻量目录访问协议 (LDAP) — SSL 和 LDAP 访问帐户。可以保护使用 `sp_ldapadmin` 和 `sp_ssladmin` 存储过程管理的口令。

可以选择使用 `sp_hidetext` 对在 `syscomments` 中存储 SQL 文本的对象（例如存储过程、用户定义的函数和计算列）进行强加密。

---

**注释** 加密外部口令和隐藏文本需要 ASE\_ENCRYPTION 许可证。

---

## 服务密钥

服务密钥是 256 位持久性加密密钥，用于对外部登录口令和隐藏文本进行强加密，它们存储在 `sysencryptkeys` 中。

可使用以下任一项对服务密钥进行加密：

- 静态密钥 — 是用于服务密钥的缺省密钥加密密钥，如果尚未在当前数据库中创建主密钥，则可以使用该密钥。借助此方法，Adaptive Server 可以在进行无人值守启动后使用服务密钥。
- 主密钥 — 所提供的保护强于静态密钥。Adaptive Server 需要口令来解密特定于数据库的主密钥。

描述这些服务密钥的数据库对象包括：

- `syb_extpasswdkey` — 标识用于对 `sysattributes` 中的外部登录口令进行加密的服务密钥。任何数据库都只有一个 `syb_extpasswdkey`。当 `syb_extpasswdkey` 更改时，将使用新密钥对使用该密钥加密的所有数据重新进行加密。

尽管外部登录口令通常存储在主数据库中，但 RepAgent 将此信息存储在复制数据库中。

- `syb_syscommkey_ddd` — 标识用于对 `syscomments` 中的隐藏文本进行加密的服务密钥，其中，“ddd”是 Adaptive Server 生成的、用于唯一标识该密钥的全局标识符。如果有许多与同一对象关联的 `syb_syscommkey` 密钥，则名称中将包含全局标识符以区分各个名称。全局标识符用于区分本地数据库和复制数据库中的密钥。

对隐藏文本进行强加密需要在其中执行 `sp_hidetext` 以隐藏 SQL 文本的每个数据库中的服务密钥。在创建新服务密钥后，数据库中的任何现有服务密钥仍将保留，直到被显式删除，并且在您重新发出 `sp_hidetext` 之前，不会对任何隐藏文本重新进行加密。

---

**注释** 系统加密口令不加密服务密钥。

---

## 创建服务密钥

具有 `sso_role` 或 `keycustodian_role` 的用户可以使用以下命令创建服务密钥：

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

缺省情况下，使用静态密钥加密这些密钥。若要使用主密钥，可使用 `with master key` 参数。

创建服务密钥的用户将成为该密钥的所有者。

在创建 `syb_extpasswdkey` 时，将通过使用强加密的新密钥对 `sysattributes` 中的所有外部口令重新进行加密。

在创建 `syb_syscommkey` 后，`sp_hidetext` 的任何后续执行将使用具有强加密的新密钥。必须对现有数据库对象执行 `sp_hidetext`，才能使用新密钥加密该对象。因为重新加密隐藏文本涉及的数据量可能非常大，所以数据库管理员应该将 `sp_hidetext` 推迟到系统需求较低时执行。

要创建服务密钥，必须满足以下条件：

- 需要 `ASE_ENCRYPTION` 许可证。
- 必须设置 `enable encrypted columns` 配置参数。
- 创建服务密钥的用户必须具有 `sso_role` 或 `keycustodian_role`。
- 如果要使用主密钥保护服务密钥，则必须在创建服务密钥之前创建主密钥。

---

**注释** 不能对服务密钥使用双控制。

---

## 删除服务密钥

`drop encryption key` 可确保不存在对加密密钥的引用，并随后将其删除。您无法删除不存在的 `syb_extpasswdkey` 或 `syb_syscommkey_ddd`。要确保删除所有隐藏文本密钥，可使用 `sp_encryption` 标识所有现有密钥。

---

**注释** 如果您的 `ASE_ENCRYPTION` 许可证已过期，则加密数据不再可用，并且您无法执行 `drop encryption key` 命令。请与 Sybase 技术支持部门联系，以获取临时许可证。

---

具有 `sso_role` 或 `keycustodian_role` 的用户可使用以下命令删除未使用的外部登录服务密钥：

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

当指定 `with password encryption downgrade` 时，Adaptive Server 将用 15.7 之前的版本中使用的算法来重置外部登录口令。Replication Agent 口令以及 CIS 和 RTMS 外部登录口令将重置为无效值。管理员必须在删除密钥后手动重新输入这些口令，才能继续使用相应的服务。

具有 `sso_role` 或 `keycustodian_role` 的用户可使用以下命令删除未使用的隐藏文本服务密钥：

- 该命令指示您删除单个密钥：

```
drop encryption key syb_syscommkey_ddddd
```

Adaptive Server 会检查是否对指定密钥 `_dddddd` 进行了任何引用，并在未发现引用时删除该密钥。

因为 `syb_syscommkey_ddddd` 指示单个密钥，所以您无法使用 `with text encryption downgrade` 参数指定 `syb_syscommkey_ddddd`。

- 该命令指示您删除多个密钥：

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

- 如果您指定 `with text encryption downgrade`，则无法使用 `syb_syscommkey_ddddd` 指定单个服务密钥，而只能使用 `syb_syscommkey`。
- 如果 `syb_syscommkey` 不带 “`dddddd`” 后缀，Adaptive Server 将用 15.7 之前的版本中使用的算法重新加密 `syscomments` 中的所有隐藏文本，并删除所有 `syb_syscommkey_ddddd` 密钥。

## 修改服务密钥

您可以重新生成 `syb_extpasswdkey` 或将其保护加密从主密钥更改为静态密钥，反之亦然。但您无法重新生成 `syb_syscommkey`。

## 更改 `syb_extpasswdkey`

可使用以下命令更改 `syb_extpasswdkey`：

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

其中：

- `with {static key | master key}` 的第一个实例是可选的，它表示当前加密 `syb_extpasswdkey` 的方式。
- `with {static key | master key}` 的第二个实例允许管理员将对重新生成的密钥的加密从静态更改为动态，反之亦然。如果省略该参数，重新生成的密钥将保持发出此命令之前的加密状态。
- `with {static key | master key}` 的第三个实例更改对现有密钥的保护，以便根据指定使用静态密钥或主密钥。如果省略该参数，缺省情况下将使用静态密钥。

重新生成 `syb_extpasswdkey` 是单个事务：

- 1 为外部登录口令创建新的服务密钥。
- 2 使用新密钥重新加密 `sysattributes` 中的口令。
- 3 删除旧密钥。

例如：

- 为外部登录口令创建一个服务密钥，然后使用受静态密钥保护的的服务密钥加密所有外部登录口令。

```
create encryption key syb_extpasswdkey
```

- 为外部登录口令重新生成服务密钥，以保留受静态密钥保护的新服务密钥并重新加密由旧服务密钥加密的所有外部口令：

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

- 更改对要由主密钥加密的服务密钥的保护。该服务密钥不发生更改，并且不重新加密外部登录口令：

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

---

**注释** 在发出该命令之前，请确保主密钥所有者已输入主密钥口令。

---

## 更改 syb\_syscommkey

若要更改 syb\_syscommkey，可创建一个新密钥并使用 sp\_hidetext 通过该新密钥重新进行加密。

例如：

- 示例 1 — 创建一个新的隐藏文本加密密钥，并使用新创建的密钥加密 syscomments 表中的所有 SQL 文本对象：

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

---

**注释** 创建新 syb\_syscommkey 后，它将成为该数据库中供 sp\_hidetext 使用的缺省密钥。

---

- 示例 2 — 创建一个新的隐藏文本加密密钥，使用新创建的密钥加密 syscomments 中特定存储过程的文本，并使用主密钥保护该密钥：

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

在本例中，syscomments 中的所有其它隐藏文本行仍然使用上面的加密密钥进行加密。

## 对外部口令使用服务密钥

服务密钥可加密使用 SSL 的网络监听器的私钥口令。私钥口令可初始化 SSL 证书。

## SSL 口令

如果通过主密钥加密服务密钥，而该主密钥不可用：

- 当仅在接口文件中指定 SSL 监听器时，没有用户可以登录来输入主密钥或双主密钥口令。Adaptive Server 将关闭，因为它无法启动任何监听器。

- 当在接口文件中同时指定 SSL 和非 SSL 监听器时，非 SSL 监听器可以接受登录请求。SSL 监听器将被阻止，直至授权用户在连接到非 SSL 监听器端口上的 Adaptive Server 后使用以下命令手动输入主密钥口令：

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

在正确输入主密钥口令后，Adaptive Server 将唤醒 SSL 监听器进程，这些进程将开始接受传入的登录请求。

## LDAP 口令

当 Adaptive Server 在进行 LDAP 用户身份验证过程期间验证用户身份时，需要使用服务密钥对 LDAP 管理帐户的口令进行解密。在完成验证过程之前，用户无法使用 LDAP 进行登录。可以使用 Adaptive Server 鉴定对其进行本地鉴定的授权用户可以使用以下命令手动输入主密钥口令：

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

有关配置 LDAP 用户身份验证的详细信息，请参见《安全性管理指南》中有关外部验证的章节。

## Replication Agent 口令

服务密钥可解密 Replication Agent 用于对用户数据库启动连接的口令。如果通过主密钥加密服务密钥，则配置为自动启动的代理将被阻止，直至授权用户手动输入主密钥口令。

如果服务密钥位于复制的用户数据库中，则该服务密钥还可用于复制数据库，因为也会复制存储加密密钥的 `sysencryptkeys` 表。主密钥也存储在复制的 `sysencryptkeys` 表中，并且也可用于复制数据库。因为已加密服务密钥，所以在复制过程中这些服务密钥仍受保护。

Adaptive Server 启动后，授权用户可使用以下命令进行连接并设置每个数据库的主密钥口令：

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

可以通过状态值 “passwd sleep” 识别等待主密钥口令的 Replication Agent:

```
sp_who
go

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

## 在升级期间对隐藏文本使用服务密钥

在升级到 15.7 版本期间，将根据源代码重新编译过程对象。在为对隐藏文本启用强加密的数据库输入主密钥口令，以及通过主密钥保护服务密钥之前，已连接用户可以执行的操作会受到限制。授权用户必须使用以下命令对此类数据库设置主密钥口令：

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

## 使用主密钥加密的服务密钥的相关注意事项

如果通过主密钥加密服务密钥，则必须根据您的指定主密钥的方式，将主密钥的口令自动或手动输入 Adaptive Server 中。

如果您不使用自动主密钥访问，则通常可以使用 `set encryption passwd` 输入主密钥口令。但是，如果在启动期间需要使用服务密钥解密网络监听器的私钥口令，则可以在命令行中或通过命令行提示提供主密钥。

使用 `dataserver ... --master_key_password` 参数将在启动 Adaptive Server 期间提示您输入主密钥口令。发出 `--master_key_passwd` 参数的用户必须知道主数据库的主密钥口令，并对控制台和用于输入该口令的键盘具有物理访问权限。



如果您不提供口令，`--master_key_password` 将提示您在命令行中输入该口令。例如：

```
dataserver --master_key_passwd -dd_master -eerrorlog
master_key_passwd:_
```

不会显示口令字符，也不会验证该口令，直至稍后进入 Adaptive Server 启动序列。

如果在 `--master_key_passwd` 参数中包括口令：

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

在读取和使用 `mysecret` 口令后，将从内存中清空该口令。但在清空内存之前会显示明文口令。

如果您输入的口令不正确，则使用服务密钥的尝试将失败，需要服务密钥的 Adaptive Server 服务仍然无法使用。当该服务器启动后，授权用户可使用以下命令进行连接并在主数据库中设置主密钥口令：

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

如果您仅配置了 SSL 监听器并输入错误的口令，Adaptive Server 将关闭，因为它无法启动任何监听器。

Sybase 建议您不要在命令行中输入口令，因为会在以下位置显示这些口令：

- 在可以使用 UNIX `ps` 命令查看的内存中
- 在内存中，在无人值守的终端屏幕上，或者磁盘上的命令历史记录缓冲区和文件中
- 屏幕上

Sybase 建议客户网站提供口令输入提示，以避免在使用有人值守启动时出现这些漏洞。



# 加密数据

主题	页码
<a href="#">指定新表加密</a>	38
<a href="#">加密现有表中的数据</a>	40
<a href="#">创建加密列的索引和约束</a>	41
<a href="#">创建加密列的域和访问规则</a>	42
<a href="#">decrypt 权限</a>	42
<a href="#">限制解密权限</a>	44
<a href="#">返回缺省值而非解密数据</a>	44
<a href="#">加密列长度</a>	51

可以对以下数据类型加密：

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

## 指定新表加密

要加密新表中的列，请在 `create table` 语句中使用 `encrypt column` 限定符。`create table` 的以下部分语法仅包括特定于加密的子句。请参见《参考手册：命令》。

```
create table table_name
(column_name
...

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

*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。

---

**注释** 不能加密计算列，并且加密列不能出现在定义计算列的表达式中。您不能在表的 *partition\_clause* 中指定加密列。

---

以下示例创建两个密钥：数据库缺省密钥，以及另一密钥 (`cc_key`)，该密钥必须在 `create table` 命令中指定。两个密钥均使用长度和初始化矢量的缺省值。`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

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

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

该示例创建了密钥 `k1`，此密钥使用初始化矢量和随机填充的非缺省值。`employee` `esalary` 列在加密前已使用随机数据进行了填充：

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

## 指定 *select into* 加密

缺省情况下，*select into* 创建一个没有加密的目标表，即使源表具有一个或多个加密列。若要加密目标表中的任何列，必须使用 *encrypt* 子句对目标列进行限定，如下所示：

```
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
```

即使在源表中没有加密数据，您也可以加密目标表中的特定列。如果源表中的列的加密密钥与为目标列指定的密钥相同，Adaptive Server 可通过绕过源表的解密步骤和目标表的加密步骤来优化处理。

用于指定对目标表加密的规则与 *create table* 上指定的加密规则在以下几个方面是相同的：

- 要加密的列允许的数据类型
- 省略 *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。

下面的示例选择了 *daily\_xacts* 表中的加密列 *creditcard*，并以加密形式将其存储在 *#bigspenders* 临时表中：

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

---

**注释** *select into* 需要源表的列级权限（包括 *decrypt*）。

---

## 加密现有表中的数据

要加密现有表中的列，请在包含 `encrypt` 子句的 `alter table` 语句中使用 `modify column` 选项：

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

其中，*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。

请参见《参考手册：命令》。

修改加密列存在以下限制：

- 不能对创建了触发器的列进行修改以将其加密或解密。必须：
  - a 删除触发器。
  - b 加密或解密列。
  - c 重新创建触发器。
- 不能更改现有加密列，不能修改表中的某个列以进行加密或解密，也不能修改加密列的类型（如果该列是聚簇索引或位置索引中的键）。必须：
  - a 删除索引。
  - b 改变表/修改列的类型。
  - c 重新创建索引。

您可以在改变列的其它属性的同时更改加密属性。也可以使用 `alter table` 添加加密列。

例如：

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

## 创建加密列的索引和约束

如果已指定加密密钥而未使用任何初始化矢量或随机填充，则可以创建加密列的索引。如果对具有初始化矢量或随机填充的加密列执行 `create index`，则会发生错误。加密列的索引通常对等同性和非等同性匹配非常有用。不过，索引既不能用来匹配区分大小写的的数据，也不能用来对任何数据执行范围搜索。

---

**注释** 不能在函数索引表达式中使用加密列。

---

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
  with init_vector null

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

create index cust_idx on customer(creditcard)
```

可以对已声明为主键或唯一键的列进行加密。

在以下情况下，可以定义加密列的参照完整性约束：

- 引用的列和被引用的列是使用相同密钥进行加密的。
- 用于对列进行加密的密钥将 `init_vector` 指定为 `null`，但尚未指定 `pad random`。

由于参照完整性检查是针对密文值执行的，因此非常有效。

在下面的示例中，`ssn_key` 对主表和外表中的 `ssn` 列进行了加密：

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

## 创建加密列的域和访问规则

可以创建加密列的域规则、检查约束或访问规则。但是，如果将加密列用于目标列表、`where` 子句等，则需要该加密列的解密权限。下面的示例创建了 `creditcard` 列的 `rule_creditcard` 规则，该列已定义了一个域规则：

```
create encryption key cc_key
    with init_vector null

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

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

```
sp_bindrule rule_creditcard, creditcard
```

`bcp in -C` 会绕过加密列的域规则或检查约束，因为 Adaptive Server 使用包含 `bcp in -C` 的快速 `bcp`。如果加密列包含访问规则，`bcp out -C` 会生成错误号 2929。当复制带有域规则或检查约束的加密列时，Adaptive Server 会绕过 `insert` 和 `update` 语句的规则或约束。当您为 `update`、`delete` 或 `select` 语句复制包含访问规则的加密列时，Adaptive Server 也会生成错误号 2929。

## decrypt 权限

要从加密列中选择明文数据或者搜索或连接加密列，用户必须具有解密权限。

表所有者或具有 `sso_role` 的用户可以使用 `grant decrypt` 向其他用户、组和角色授予解密表中的一个或多个列的显式权限。在过程或视图所有者授予以下权限时，可能会隐式地授予解密权限：

- 对存储过程或用户定义函数的 `exec` 权限，以便从加密列（过程或函数的所有者也拥有包含该加密列的表）中选择数据
- 对视图列的 `decrypt` 权限，以便从视图所有者也拥有该表的加密列中选择数据

在这两种情况下，不需要对基表中的加密列授予解密权限。



语法为：

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

在表或视图级授予解密权限时，将对表中的所有加密列授予解密权限。

要授予 `customer` 表中所有加密列的解密权限，请输入：

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

以下示例显示了 `user2` 在基表 “`employee`” 的 `ssn` 列上的隐式解密权限。`user1` 设置 `employee` 表和 `employee_view`，如下所示：

```
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` 在从 `emp_salary` 视图中选择数据时，访问解密的社会保险号：

```
select * from emp_salary
```

---

**注释** `grant all` 不授予解密权限。解密权限必须单独授予。

---

将 `Adaptive Server` 配置为使用受限解密权限，以限制用户的隐式解密权限。请参见第 44 页的“限制解密权限”。

只具有加密列的 `select` 权限的用户仍可以通过 `bulk copy` 命令将加密列作为密文处理。此外，如果加密列指定了解密缺省值，则无权解密数据的用户可以在 `select` 目标列表或 `where` 子句中指定该列。请参见第 44 页的“返回缺省值而非解密数据”。

## 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
```

## 限制解密权限

即使您使用主密钥或系统加密口令来保护密钥， Adaptive Server 也可保护数据隐私，以防管理员访问私有数据。如果您不希望进行口令管理，并使用主密钥或系统加密口令来保护加密密钥，则可以通过设置 `restricted decrypt permission` 配置参数来限制数据库所有者对私有数据的访问。系统安全员 (SSO) 可以使用此参数来控制哪些用户拥有解密权限。一旦启用 `restricted decrypt permission`，SSO 即是接收隐式解密权限以及具有将该权限授予其他人的隐式特权的唯一用户。SSO 将决定哪些用户接收到解密权限，或者通过使用 `with grant` 选项授予解密权限将此工作委派给另一个用户。表所有者不会自动拥有对其表的解密权限。

对存储过程或用户定义的功能拥有执行权限的用户不具有解密由过程或功能选定的数据的隐式权限。对视图列拥有解密权限的用户不具有解密由视图选定的数据的隐式权限。

---

**注释** 具有别名的用户会继续继承其别名所指向的用户的所有解密权限。`set proxy/set user` 语句会继续允许管理员或数据库所有者设置其标识已被该命令采用的用户的解密权限。

---

## 分配受限解密权限的特权

如果使用受限解密权限，则可以分配特权，以创建任务模式以及管理密钥，如下所示：

- 系统安全员 — 配置受限解密权限、创建加密密钥并为数据库所有者授予密钥的 `select` 权限，以及为最终用户授予解密权限。
- 数据库所有者 — 创建模式并装载数据。

## 返回缺省值而非解密数据

本节说明如何将解密缺省值用于加密列。未被允许查看机密数据的用户在针对加密列运行查询时，将看到解密缺省值而非解密数据。解密缺省值使旧式应用程序和报告可以在运行时不出错，即使对于未被允许查看机密数据的用户也是如此。

## 定义解密缺省值

当没有解密权限的用户尝试从加密列中选择信息时，`create table` 和 `alter table` 的 `decrypt_default` 参数允许加密列返回用户定义的值，从而避免显示错误消息 10330：

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

使用加密列的解密缺省值允许现有报告运行完成而没有错误，并允许用户继续查看未加密的信息。例如，如果 `customer` 表包含加密列 `creditcard`，则可以设计表模式以便获得以下结果：

```
select * from customer
```

对没有解密权限的用户返回值 “\*\*\*\*\*”，而不是返回信用卡数据。

### 添加和删除解密缺省值

使用 `create table` 指定新列的解密缺省值。`create table` 的部分语法为：

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

- `decrypt_default` — 指定此列对没有解密权限的用户发出的 `select` 语句返回缺省值。
- `value` — 是 Adaptive Server 对 `select` 语句返回的值，而不是解密值。常量值表达式无法引用数据库列，但可以包含自身引用表和列的用户定义函数。该值只能在可空列中 NULL，且必须能够转换为该列的数据类型。

例如，当没有解密权限的用户选择表 `t2` 的 `ssnum` 列时，它会返回 “?????????”：

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

要向以前未加密的现有列添加加密和解密缺省值，请使用：

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

以下示例修改了 `emp` 表以对 `ssn` 列进行加密，并指定了解密缺省值：

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

要向现有加密列添加解密缺省值或更改已具有解密缺省值的列上的解密缺省值，请使用：

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

以下示例向已经加密的 `salary` 列添加了解密缺省值：

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

以下示例使用新值替换以前的 `decrypt_default` 值并使用用户定义的函数 (UDF) 生成缺省值：

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

要在不删除加密属性的情况下从加密列删除解密缺省值，请使用：

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

以下示例删除了 `salary` 的解密缺省值，但并没有删除加密属性：

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

## 权限和解密缺省值

必须先对加密列授予解密权限，然后用户或角色才能选择或搜索这些列中的加密数据。如果加密列具有解密缺省属性，则没有解密权限的用户可以运行在这些列上进行选择或搜索的查询，但明文数据既不会显示，也不会用于搜索。

在此示例中，表 `emp` 的所有者允许具有 `hr_role` 的用户查看 `emp.ssn`。因为 `ssn` 列具有解密缺省值，所以对 `emp` 只具有 `select` 权限的用户以及不具有 `hr_role` 的用户将只看到 `decrypt_default` 值，而不是实际的解密数据。

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

如果您具有 `hr_role`，并且从此表执行 `select`，则会看到 `ssn` 的值：

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

如果您不具有 `hr_role`，并且从此表执行 `select`，则将看到解密缺省值：

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

如果您不具有此表的 `hr_role`，则 `order by` 子句不会影响结果集。

## 具有解密缺省值的列

对于在查询中如何使用具有解密缺省属性的列没有限制。您可以在目标列表表达式、`where` 子句、`order by`、`group by` 或子查询中使用它们。虽然解密缺省常量值的表达式可能没有实际用途，但是放置列的解密缺省值对于 `Transact-SQL™` 语句中列的使用不会施加任何语法限制。

以下示例对目标列表中具有解密缺省值的列使用 `select` 语句：

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

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

如果您对此表执行 `select` 语句，但您没有解密权限，则将看到以下结果：

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

当 `Adaptive Server` 对 `select into` 命令返回列的解密缺省值时，此解密缺省值会插入到目标表中。但是，目标列不会继承解密缺省属性。必须使用 `alter table` 对目标表指定解密缺省值。

## 解密缺省列和查询限定

如果在 `where` 子句中使用具有解密缺省属性的列，则当您不具有 `decrypt` 权限时，限定的求值结果为 `false`。这些示例使用上述 `emp` 表。只有具有 `hr_role` 的用户才拥有 `ssn` 的解密权限。

- 1 如果您具有 `hr_role` 并发出以下查询，则 Adaptive Server 会返回一行。

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

```
名
```

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

- 2 如果您不具有 `hr_role`，则 Adaptive Server 不会返回任何行：

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

```
名
```

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

- 3 如果您具有 `hr_role`，并且在非加密列中包含 `or` 语句，则 Adaptive Server 会返回相应的行：

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

```
名
```

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

- 4 如果您不具有 `hr_role`，并且发出相同的命令，则 Adaptive Server 只返回一行：

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

```
名
```

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

在这种情况下，对具有解密缺省属性的加密列的限定的求值结果为 `false`，而对非加密列的限定会成功。

如果对加密列没有 `decrypt` 权限，并且对具有解密缺省值的此列发出 `group by` 语句，则 Adaptive Server 会按解密缺省常量值分组。

## decrypt default 和隐式授予

如果您对表不具有显式或隐式权限，则 Adaptive Server 会返回 decrypt default 值。

在此示例中（使用上述 emp 表），数据库所有者会创建 p\_emp 过程，该过程会从数据库所有者拥有的 emp 表中进行选择：

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

因为您具有 corp\_role，所以您对 emp 拥有隐式 select 和 decrypt 权限：

```
exec p_emp

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

如果 emp 表和 p\_emp 存储过程已由其他用户创建，则您必须对 emp 具有 select 权限，以避免权限错误。如果您具有 select 权限，但没有 decrypt 权限，则 Adaptive Server 会返回 emp.ssn 的 decrypt default 值。

在下一个示例中，“joe”（该用户不具有数据库）会创建 v\_emp 视图，该视图将从 emp 表中选择数据。对该视图授予的任何权限都不会隐式应用到基表。

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

虽然您具有 emp\_role，但是当您发出以下命令时：

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

因为对 dbo.emp.ssn 的 decrypt 权限尚未授予 emp\_role，并且 dbo.emp.ssn 上没有对 emp\_role 执行的隐式 grant，所以 Adaptive Server 会返回以下结果：

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

## decrypt default 和 insert、update 以及 delete 语句

decrypt default 参数不会影响 insert 和 update 语句的目标列表。

如果在 update 或 delete 语句的 where 子句中使用具有解密缺省值的列，则 Adaptive Server 不能更新或删除任何行。例如，使用上述示例中的 emp 表和权限时，如果您不具有 hr\_role 并且发出以下查询，则 Adaptive Server 不会删除用户名称：

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

解密缺省属性可能会间接影响应用程序插入和更新数据，尤其是具有图形用户界面 (GUI) 进程的应用程序：

- 1 选择数据
- 2 允许用户更新任何数据
- 3 将更改行应用回相同的或不同的表

如果用户对加密列没有 decrypt 权限，则应用程序会检索解密缺省值，并且可能会将未更改的解密缺省值自动写回表中。要避免使用解密缺省值覆盖有效数据，请使用检查约束来防止自动应用这些值。例如：

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

如果用户没有 cc\_num 的 decrypt 权限，并且选择了 customer 表中的数据，则会显示以下数据：

name	cc_num
Paul Jones	-1
Mick Watts	-1

但是，如果用户更改名称并更新了数据库，则应用程序会尝试更新所示值中的所有字段，cc\_num 的缺省值会导致 Adaptive Server 发出错误 548：

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

设置检查约束可保护数据的完整性。要获得更好的解决方法，您可以在编写应用程序逻辑时过滤这些更新。



## 删除解密缺省值

可以使用以下任何命令来删除解密缺省值：

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

例如，要从 `ssn` 列删除解密缺省属性，请输入：

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

如果您不具有 `hr_role`，并且在表所有者删除了解密缺省值之后从 `emp` 表中选择数据，则 Adaptive Server 会返回错误消息 10330。

## 加密列长度

在执行 `create table`、`alter table` 和 `select into` 操作的过程中，Adaptive Server 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。若要忽略初始化矢量，请在 `create encryption key` 中指定 `init_vector null`。
- 明文数据的长度。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。这 2 个字节表示明文数据长度。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。

在表 5-1 中，最大加密数据长度列中的长度反映了 `sycolumns.enclen` 中表示指定类型和长度的列的值。

**表 5-1: 加密列的数据类型长度**

用户指定的列类型	输入数据长度	加密列类型	最大加密数据长度 (无 init vector)	实际加密数据长度 (无 init vector)	最大加密数据长度 (有 init vector)	实际加密数据长度 (有 init vector)
bigint	8	varbinary	17	17	33	33
unsigned bigint	8	varbinary	17	17	33	33
tinyint、smallint、int (有符号或无符号)	1、2 或 4	varbinary	17	17	33	33
tinyint、smallint、int (有符号或无符号)	0 (空)	varbinary	17	0	33	0
float、float(4)、real	4	varbinary	17	17	33	33
float、float(4)、real	0 (空)	varbinary	17	0	33	0
float(8)、double	8	varbinary	17	17	33	33
float(8)、double	0 (空)	varbinary	17	0	33	0
numeric(10,2)	3	varbinary	17	17	33	33
numeric(38,2)	18	varbinary	33	33	49	49
numeric(38,2)	0 (空)	varbinary	33	0	49	0
char、varchar(100)	1	varbinary	113	17	129	33
char、varchar(100)	14	varbinary	113	17	129	33
char、varchar(100)	15	varbinary	113	33	129	49
char、varchar(100)	31	varbinary	113	49	129	65
char、varchar(100)	0 (空)	varbinary	113	0	129	0
binary、 varbinary(100)	1	varbinary	113	17	129	33
binary、 varbinary(100)	14	varbinary	113	17	129	33
binary、 varbinary(100)	15	varbinary	113	33	129	49
binary、 varbinary(100)	31	varbinary	113	49	129	65
binary、 varbinary(100)	0 (空)	varbinary	113	0	65	0
unicar(10)	2 (1 个 unicar 字符)	varbinary	33	17	49	33
unicar(10)	20 (10 个 unicar 字符)	varbinary	33	33	49	49

用户指定的列类型	输入数据长度	加密列类型	最大加密数据长度 (无 init vector)	实际加密数据长度 (无 init vector)	最大加密数据长度 (有 init vector)	实际加密数据长度 (有 init vector)
univarchar(20)	20 (10 个 unichar 字符)	varbinary	49	33	65	49
date	4	varbinary	17	17	33	33
time	4	varbinary	17	17	33	33
time	null	varbinary	17	0	33	0
smalldatetime	4	varbinary	17	17	33	33
datetime	8	varbinary	17	17	33	33
smallmoney	4	varbinary	17	17	33	33
money	8	varbinary	17	17	33	33
money	null	varbinary	17	0	33	0
bit	1	varbinary	17	17	33	33

### 注释

- Adaptive Server 不支持 timestamp 数据类型。
- 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。
- 对于加密列，磁盘上的列长度将会增加，但这种增加对工具和命令不可见。例如，sp\_help 只显示原始大小。



## 访问加密数据

主题	页码
<a href="#">处理加密列</a>	55
<a href="#">解密权限</a>	56
<a href="#">删除加密</a>	57

在加密列中处理数据时， Adaptive Server 会自动执行加密和解密。在加密列中更新或插入数据时， Adaptive Server 会对数据进行加密，而在 `where` 子句中选择或使用数据时， Adaptive Server 会对该数据进行解密。

### 处理加密列

针对加密列发出 `select`、`insert`、`update` 或 `delete` 命令时， Adaptive Server 会使用与加密列关联的加密密钥自动加密或解密数据。

- 对加密列发出 `insert` 或 `update` 时：
  - 如果您对加密列没有 `insert` 或 `update` 权限，该命令将失败。
  - 如果列是使用用户指定的口令进行加密的， Adaptive Server 要求该口令可用。如果尚未设置用户指定的口令，命令将失败。请参见第 68 页的“使用用户口令访问加密数据”。
  - Adaptive Server 会对加密密钥进行解密。
  - Adaptive Server 会使用列的加密密钥对数据进行加密。
  - Adaptive Server 会将 `varbinary` 密文数据插入表中。
  - 完成插入或更新后， Adaptive Server 会清除保存明文内容的内存。完成语句后，则会清除保存原始加密密钥的内存。

- 针对加密列中的数据发出 `select` 命令时：
  - 如果您对加密列没有 `select` 权限，该命令将失败。
  - 如果加密密钥与使用用户指定口令加密的列相关联，Adaptive Server 要求该口令可用。如果尚未设置用户指定的口令，`select` 语句将失败。请参见第 68 页的“使用用户口令访问加密数据”。否则，Adaptive Server 会对加密密钥进行解密。
  - 如果您对该列具有 `decrypt` 权限，便可成功解密所选数据，Adaptive Server 会将明文数据返回给用户。
  - 如果已对加密列声明某个解密缺省值，并且您不具有该列的 `decrypt` 权限，Adaptive Server 将返回解密缺省值。
- 将加密列包含在 `where` 子句中时：
  - 如果您对该列没有 `decrypt` 权限，而该列包括解密缺省值，则 `where` 子句谓词的求值结果为 `false`。请参见第 48 页的“解密缺省列和查询限定”。
  - 如果出现下述情况，Adaptive Server 会尽可能不解密数据就进行比较：
    - `where` 子句将加密列与使用同一密钥加密的另一列进行连接，而未使用初始化矢量或随机填充
    - 正在使用等于或不等于条件将列数据与常数值进行匹配请参见第 83 页的“性能注意事项”。

## 解密权限

要查看或处理解密数据，用户必须具有：

- 对目标列表以及 `where`、`having`、`order by`、`group by` 等其它此类子句中使用的列具有 `select` 和 `decrypt` 权限
- 在将 `passwd password_phrase` 子句与 `create` 或 `alter encryption key` 命令一起使用时用来对密钥进行加密的口令。请参见第 7 章“防止管理员查看数据隐私”。

将 Adaptive Server 配置为具有受限 `decrypt` 权限会限制隐式解密权限。必须向表所有者显式授予 `decrypt` 权限才能使表所有者可以从其拥有的表的加密列中执行选择操作。用户可能预料不到存储过程的 `execute` 权限或视图的 `select` 权限不会针对基础表显式授予用户 `decrypt` 权限。用户还必须对基表具有显式 `decrypt` 权限。

## 删除加密

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

例如，要删除 `customer` 表中的 `creditcard` 列的加密，请输入：

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

如果 `creditcard` 列是通过一个使用显式用户口令的密钥加密的，则需要首先设置该口令。



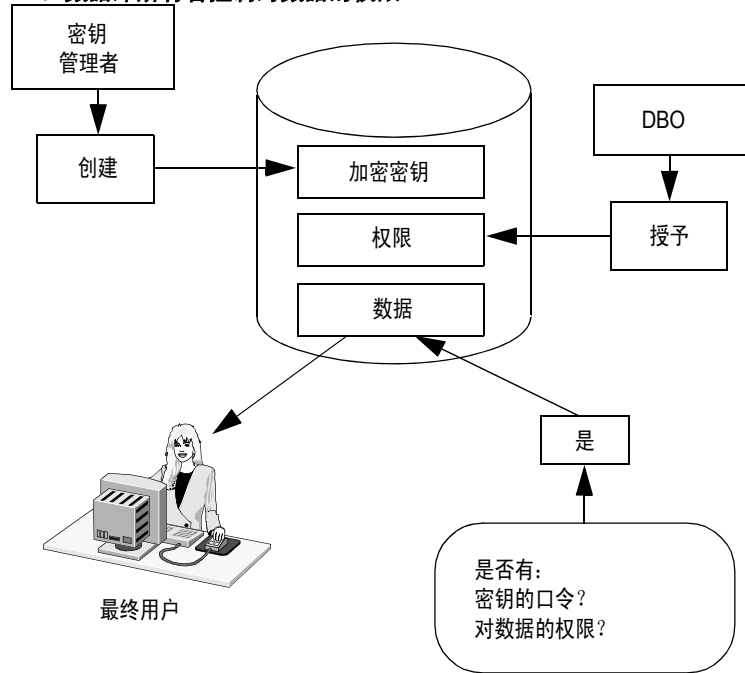


主题	页码
密钥管理者角色	59
使用用户指定的口令保护密钥	62

## 密钥管理者角色

必须为其分配 `keycustodian_role` 的密钥管理者负责维护加密密钥。使用 `keycustodian_role` 角色可以通过确保没有任何管理员具有对数据的隐式访问权，来分离管理机密数据的职责。图 7-1 说明了作为模式所有者的数据库所有者可以控制访问数据的权限，但在不知道密钥口令的情况下不具有任何访问权。密钥管理者管理密钥及其口令，但对数据不具有任何权限。只有对数据具有权限并且知道加密密钥口令的限定最终用户才可能访问该数据。

图 7-1：数据库所有者控制对数据的权限



系统管理员和数据库所有者不具有隐式密钥管理职责。Adaptive Server 可提供系统角色 `keycustodian_role`，以便 SSO 不需要承担全部加密职责。密钥管理者拥有加密密钥，但是对数据不应具有显式或隐式权限。数据库所有者可通过列权限授予用户对数据的访问权，并且密钥管理者允许用户访问密钥口令。`keycustodian_role` 将被自动授予 `sso_role`，并可以由具有 `sso_role` 的用户授予。

密钥管理者可以执行以下操作：

- 创建和变更加密密钥。
- 将其拥有的密钥指定为数据库缺省密钥，只要该管理者同时还拥有当前缺省密钥（如果有）。
- 为指定用户设置密钥副本，使每个用户都可以通过选定的口令或登录口令访问密钥。
- 与最终用户共享密钥加密口令。
- 授予模式所有者对密钥管理者所拥有密钥的加密密钥的选择访问权。
- 创建主密钥或设置系统加密口令。
- 恢复加密密钥。

- 删除其所拥有的加密密钥。
- 更改其所拥有密钥的所有权。

您可以有多个密钥管理者，每人拥有一组密钥。密钥管理者会授予模式所有者使用 `create table`、`alter table` 和 `select into` 密钥的权限，还可能对具有特权的用户公开密钥口令，或者允许用户将密钥副本与个人口令或登录口令进行关联。密钥管理者可以与“密钥恢复者”一起工作，以便在丢失口令或出现灾难时恢复密钥。如果密钥管理者离开公司，则 SSO 可以使用 `alter encryption key` 命令将密钥所有权赋予新的密钥管理者。

## 用户、角色和数据访问

加密密钥的用户指定口令可确保数据隐私受到保护，以使数据无法被系统管理员使用。表 7-1 说明如何实现以下目标：

- 密钥管理者可以拥有密钥，但看不到数据。
- 数据库所有者可以拥有模式，但不拥有数据。
- 由于具有以下权限，用户可以看到并处理数据：
  - 密钥访问权，由密钥管理者授予
  - 数据访问权，由表所有者授予

**表 7-1: 用户和角色对加密列的权限**

角色	是否可以创建加密密钥？	是否可以在模式定义中使用密钥？	是否可以解密已加密的数据？
sso_role	可以	不可以，需要 <code>create table</code> 权限	不可以。具有角色的用户可能知道口令，但是需要对表的 <code>select</code> 权限（SSO 具有隐式解密权限）。
sa_role	不可以，需要 <code>create encryption key</code> 权限	可以，但必须向其授予对密钥的 <code>select</code> 权限	不可以，需要知道口令
keycustodian_role	可以	不可以，需要 <code>create table</code> 权限	不可以。具有相应角色的用户可能知道口令，但需要对表或列具有 <code>decrypt</code> 和 <code>select</code> 权限。
数据库所有者或模式所有者	不可以，需要 <code>create encryption key</code> 权限	可以，但必须向其授予对密钥的 <code>select</code> 权限	不可以，需要知道口令。
用户	不可以	不可以	可以，但必须向其授予 <code>decrypt</code> 或 <code>select</code> 权限，并且用户必须知道密钥的口令。

## 使用用户指定的口令保护密钥

使用 `create encryption key` 将口令与密钥进行关联：

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

其中，`password_phrase` 是带引号的字母数字字符串，其长度不超过 255 个字节，Adaptive Server 将使用它来生成密钥加密密钥 (KEK)。

Adaptive Server 不会保存用户指定的口令。它会将称为 “salt” 的一组验证字节保存在 `sysencryptkeys.eksalt` 中，这使 Adaptive Server 能够识别在后续加密或解密操作中所使用的口令对密钥是否合法。您必须向 Adaptive Server 提供口令，然后才能访问由 `keyname` 加密的任何列。

当您创建加密密钥时，它在 `sysencryptkeys` 表中的条目称为基本密钥。对于某些用户和应用程序，由主密钥、系统加密口令或显式口令加密的**基本密钥**已足够。任何显式口令均在需要访问密钥的用户之间共享。此外，还可以为其他用户和应用程序创建**密钥副本**。每个密钥副本可以由单个口令加密，并作为单个行存储在 `sysencryptkeys` 中。加密密钥始终由一个基本密钥和零个或多个密钥副本来表示。

此示例说明如何对密钥使用口令，以及密钥管理者在设置加密时的作用。密钥的口令将在有处理加密数据业务需要的所有用户之间共享。

- 1 密钥管理者 “razi” 创建了加密密钥：

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

- 2 “razi” 对所有需要访问加密数据的用户分发了口令。
- 3 每个用户都需要输入口令后才能处理包含加密列的表：

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

- 4 如果由于未经授权的用户获得对口令的访问权而使密钥的安全性受到威胁，“razi” 会改变密钥以更改口令。

## 更改密钥的保护方法

可以使用 `alter encryption key` 命令来更改加密密钥的保护方法：

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

其中：

- *keyname* — 标识列加密密钥。
- `with passwd 'old_password'` — 指定以前使用 `create encryption key` 或 `alter encryption key` 语句指定以用于加密基本密钥或密钥副本的用户定义口令。该口令长度最多可以是 255 个字节。如果不对基本密钥指定 `with passwd`，则缺省值为主密钥或系统加密口令。
- `with passwd 'new_password'` — 指定 Adaptive Server 用于加密列加密密钥或密钥副本的新口令。该口令长度最多可以是 255 个字节。如果不指定 `with passwd`，并且您要加密基本密钥，则缺省值为 `system_encr_passwd`。
- `system_encr_passwd` — 是缺省加密口令。如果已经存在一个或多个密钥副本，则无法修改基本密钥以使用系统加密口令对它进行加密。在密钥管理者已经设置了加密密钥以使单个用户使用受限制之后，此限制可防止密钥管理者无意中将加密密钥暴露给访问的管理员。您无法修改密钥副本以使用系统加密口令进行加密。
- `login_passwd` — 是当前会话的登录口令。您无法修改基本密钥，以使用 `login_password` 进行加密。用户可以修改自己的密钥副本，以使用自己的登录口令进行加密。

有关使用用户登录口令加密密钥副本并且不要求密钥副本被授予人的用户执行 `alter encryption key` 的替代方法，请参见第 70 页的“[使用密钥副本登录口令的应用程序透明度](#)”。

- `master key` — 在第一个实例中，它表示当前加密使用主密钥。在第二个实例中，它表示必须使用主密钥对 CEK 重新加密。

示例 1: 在此示例中, 由于口令受到安全威胁, 或者由于知道口令的用户离开了公司, 密钥管理者变更了基本密钥。

- 1 密钥管理者 “razi” 创建了加密密钥:

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

- 2 “razi” 与需要处理加密数据 (不涉及密钥副本) 的 “joe” 和 “bill” 共享基本密钥的口令。

- 3 “joe” 离开了公司。

- 4 “razi” 更改了加密密钥的口令, 然后与 “bill” 和 “pete” (“joe” 的继任者) 共享该口令。因为基本密钥并未更改, 而是只更改了密钥的保护方式, 所以不需要重新加密数据。以下语句使用旧口令解密 key1, 并使用新口令对它进行重新加密:

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

示例 2: 使用主密钥来加密现有 CEK “k2”:

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

示例 3: 重新加密现有的当前用主密钥加密的 CEK “k3”, 以便使用双控制:

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

---

**注释** 在本例中, 您可以省略 `with master key` 以实现相同加密。

---

示例 4: 重新加密当前用主密钥和口令 “k4\_password” 加密的现有 CEK “k4”, 以便删除双控制。CEK 及其所有密钥副本均由从 “k4\_new\_password” 派生的单个密钥控制:

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

示例 5: 加密当前用主密钥加密的现有 CEK “k5”，以便使用通过主密钥和口令 “k5\_password” 加密的双控制:

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

示例 6: 加密 CEK 以便通过主密钥和口令 “k6\_password” 进行双控制:

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

为用户 “ned” 加密其 CEK “k6” 的现有密钥副本（当前已使用通过主密钥和口令 “k6\_password” 进行的双控制对该副本进行加密），以便通过主密钥和口令 “k6\_ned\_password” 进行双控制:

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

---

**注释** 用户 “ned” 无法更改他的密钥副本的双控制属性。

---

示例 7: 加密当前用主密钥和双主密钥加密的 CEK “k7”，以便使用系统加密口令:

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

## 创建密钥副本

密钥管理者可能需要使密钥的副本对必须将数据装载到加密列中的管理员或操作员暂时可用。由于该操作员在其它情况下不具有访问加密数据的权限，因此他不应具有对密钥的永久访问权。

可以使密钥副本对单个用户可用，如下所示:

- 密钥管理者可使用 `create encryption key` 创建带用户定义口令的密钥。此密钥称为基本密钥。

- 密钥管理者使用 `alter encryption key` 将基本密钥的副本指派给具有单独口令的单个用户。

此语法说明如何为指定用户添加使用显式口令加密的密钥：

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

其中：

- `base_key_password` — 是用于加密基本密钥的口令，可能只有密钥管理者知道。该口令长度最多可以是 255 个字节。Adaptive Server 使用第一个口令来解密基本列加密密钥。
- `key_copy_password` — 用于加密密钥副本的口令。该口令长度不能大于 255 个字节。Adaptive Server 会创建解密基本密钥的副本，使用从 `key_copy_password` 派生的密钥加密密钥对它进行加密，并在 `sysencryptkeys` 中将加密基本密钥副本保存为新行。
- `user_name` — 标识为其创建密钥副本的用户。对于给定密钥，拥有密钥副本的每个用户在 `sysencryptkeys` 中都包含一行，该行由其用户 ID (uid) 标识。
- 密钥管理者会根据需要通过专用口令访问的用户的数量添加密钥副本。
- 用户可以改变其加密密钥副本，以使用不同的口令对它进行加密。

以下示例说明如何使用加密列来设置和使用密钥副本。

- 1 密钥管理者 “razi” 使用用户指定口令创建基本加密密钥：

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

- 2 “razi” 将 `key1` 的 `select` 权限授予数据库所有者以创建模式：

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

- 3 数据库所有者创建模式，并将表和列级访问权授予 “bill”：

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



- 4 密钥管理者为 “bill” 创建密钥副本，并将其密钥副本的口令提供给 “bill”。只有密钥管理者和 “bill” 才知道此口令。

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

- 5 当 “bill” 访问 `employee.emp_salary` 时，他会先提供其口令：

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

当 Adaptive Server 访问用户的密钥时，它会查找该用户的密钥副本。如果给定用户不存在副本，则 Adaptive Server 会假定用户想访问基本密钥。

## 更改密钥副本的口令

一旦为用户分配了密钥副本，该用户即可使用 `alter encryption key` 来修改密钥副本的口令。

此示例显示已分配有密钥副本的用户如何变更副本，以通过其个人口令来访问数据：

- 密钥管理者 “razi” 为 “bill” 设置现有密钥的密钥副本，并使用临时口令对它进行加密：

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

- “razi” 通过 `key1` 将自己的数据访问口令发送给 “bill”。
- “bill” 为其密钥副本分配专用口令：

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

只有 “bill” 才能更改其密钥副本的口令。当 “bill” 输入上述命令后，Adaptive Server 会验证 “bill” 的密钥副本是否存在。如果 “bill” 的密钥副本不存在，Adaptive Server 会假定用户要尝试修改基本密钥的口令，并发出错误消息：

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

不能为用户 “guest” 创建密钥副本以用于登录关联。使用登录口令加密密钥副本需要两个步骤。

## 使用用户口令访问加密数据

您必须提供加密密钥的口令才能在 `insert`、`update`、`delete`、`select`、`alter table` 或 `select into` 语句中加密或解密数据。如果系统加密口令保护了加密密钥，则您无需提供系统加密口令，因为 Adaptive Server 已经可以访问该口令。同样，如果使用您的登录口令加密了您的密钥副本，则在您保持登录到服务器上的同时，Adaptive Server 可以访问此口令（请参见第 70 页的“使用密钥副本登录口令的应用程序透明度”）。对于使用显式口令加密的密钥，必须先为您的会话中设置该口令，然后才能使用以下语法执行加密或解密列的任何命令：

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

其中：

- `password_phrase` — 是使用 `create encryption key` 或 `alter encryption key` 命令指定的显式口令，用于保护密钥。
- `key` — 指示当 Adaptive Server 访问由命名密钥加密的任何列时，将使用此口令解密密钥
- `keyname` — 可提供为完全限定名。例如：  
`[[database.][owner].]keyname`
- `column` — 指定 Adaptive Server 只在加密或解密命名列的上下文中使用此口令。最终用户不必知道加密给定列的密钥的名称。
- `column_name` — 对其设置加密口令的列的名称。将以下列形式提供 `column_name`：  
`[[ database.][ owner ].]table_name.column_name`

需要访问由显式口令加密的密钥的每个用户都必须提供该口令。Adaptive Server 会以加密形式将口令保存在用户会话的内部上下文中。Adaptive Server 会通过用零覆盖内存，在会话结束时从内存中删除密钥。

此示例说明当 Adaptive Server 必须加密或解密数据时，如何确定口令。它假定 `employee` 和 `payroll` 表中的 `ssn` 列是使用 `key1` 加密的，如以下简化的模式创建语句中所示：

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

- 1 密钥管理者与“susan”共享访问 `employee.ssn` 所需要的口令。他不需要公开密钥的名称即可实现此目的。
- 2 如果“susan”对 `employee` 具有 `select` 和 `decrypt` 权限，则她可以使用提供给她的 `employee.ssn` 口令选择员工数据：

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

ename
-----
Priscilla Kramnik
```

- 3 如果“susan”在没有指定 `payroll.ssn` 口令的情况下尝试从 `payroll` 中选择数据，则以下 `select` 将失败（即使“susan”对 `payroll` 具有 `select` 和 `decrypt` 权限）：

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

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

要避免此错误，“susan”必须先输入以下命令：

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

密钥管理者可能会选择在列名称的基础上而不是在密钥名称的基础上共享口令，以避免在应用程序代码中出现用户硬编码密钥名称，而这会使数据库所有者很难更改用于加密数据的密钥。但是，如果一个密钥用于加密若干列，则一次性输入口令会使得加密很方便。例如：

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

如果一个密钥用于加密若干列，并且用户要为列设置口令，则该用户需要为所有要处理的列设置口令。例如：

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

如果为列设置了口令，然后在密钥级别为加密列的密钥设置口令，则 Adaptive Server 会放弃与该列关联的口令，并保留密钥级别的口令。如果输入了相同密钥或列的两个连续条目，则 Adaptive Server 只保留最新条目。例如：

- 1 如果列 `employee.ssn` 的正确口令为 “Ynot387”，而用户错误地键入了 “Unot387”：

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

- 2 然后用户重新输入了正确的口令，则 Adaptive Server 仅保留第二个条目：

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

- 3 如果现在用户在密钥级别输入了相同的口令，则 Adaptive Server 仅保留最后一个条目：

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

- 4 如果现在用户在列级别输入了相同的口令，但由于 Adaptive Server 已经在密钥级别具有此口令，因此它会放弃此条目：

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

如果存储过程或触发器引用了由用户指定口令加密的列，则必须先设置加密口令，然后才能执行过程或引发触发器的语句。

---

**注释** Sybase 建议不要在触发器或过程中放置 `set encryption passwd` 语句；这可能会导致通过 `sp_helptext` 意外暴露口令。此外，当更改口令时，硬编码口令还需要更改过程或触发器。

---

## 使用密钥副本登录口令的应用程序透明度

密钥管理者可以使用用户登录口令来设置用于加密的密钥副本，从而提供以下好处：

- 使用方便 — 其登录口令与密钥相关联的用户可以在不提供口令的情况下访问加密数据。
- 安全性更高 — 用户需要跟踪的口令更少，减小了写下口令的可能性。

- 降低了密钥管理者的管理开销 — 密钥管理者无需对每个需要通过专用口令访问密钥的用户手动分发临时口令。
- 应用程序透明度 — 应用程序不需要提示输入口令来处理加密数据。现有应用程序可以利用措施来保护数据隐私，以使数据无法被管理员使用。

要使用用户的登录口令加密密钥副本，请使用以下命令：

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

其中，`login_association` 会提示 Adaptive Server 为指定用户创建密钥副本，它会在稍后使用该用户的登录口令加密此副本。使用登录口令加密密钥副本需要执行以下两个步骤：

- 1 通过使用 `alter encryption key`，密钥管理者会为每个需要通过登录口令访问密钥的用户创建密钥副本。Adaptive Server 将信息添加到密钥副本，以将密钥副本与给定用户安全地进行关联。将使用从主密钥或系统加密口令（不存在主密钥时）派生的密钥临时加密标识信息和密钥。密钥副本会保存在 `sysencryptkeys` 中。
- 2 当用户处理需要密钥查找的列时，Adaptive Server 会说明为此用户标识的加密密钥的副本已经准备好进行登录口令关联。通过使用主密钥或系统加密口令来解密密钥副本中的信息，Adaptive Server 会验证与密钥副本关联的用户信息是否符合用户登录凭据，并使用从已提供给会话的用户登录口令派生的 KEK 来加密该密钥副本。

当使用 `alter encryption key` 密钥为 `login_association` 添加密钥副本时，必须能够使用主密钥或系统加密口令加密该密钥副本。系统加密口令必须仍可供 Adaptive Server 用来在用户登录时解密密钥副本。在 Adaptive Server 使用用户的登录口令重新加密密钥副本之后，将不再需要系统加密口令。

以下示例使用用户的登录口令来加密用户加密密钥 `key1` 的副本：

- 1 密钥管理者 “razi” 创建了加密密钥：

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

- 2 “razi” 为用户 “bill” 创建 `key1` 的副本，最初会使用主密钥或系统加密口令加密该副本，但最终由 “bill” 的登录口令对该副本进行加密：

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

- 3 Adaptive Server 会使用主密钥或系统加密口令来加密密钥和标识 “bill” 密钥副本的信息的组合，并将结果存储在 `sysencryptkeys` 中。
- 4 “bill” 登录到 Adaptive Server 并处理数据，需要使用 `key1`。例如，如果 `emp.ssn` 由 `key1` 加密：

```
select * from emp
```

Adaptive Server 认识到它必须使用 “bill” 的登录口令来加密其 `key1` 的副本。Adaptive Server 使用主密钥或系统加密口令来解密步骤 4 中保存的密钥值数据。它会验证该信息是否符合当前登录凭据，然后使用从 “bill” 的登录口令生成的 KEK 来加密 `key1` 的密钥值。

- 5 在后续登录过程中，当 “bill” 处理由 `key1` 加密的列时，Adaptive Server 会通过使用 “bill” 的登录口令（Adaptive Server 可通过 “bill” 的内部会话上下文获得该口令）解密 `key1` 来直接访问该密钥。化名为 “bill” 的用户无法访问由 `key1` 加密的数据，因为他们自己的登录口令无法解密 `key1`。
- 6 当 “bill” 失去处理机密数据的权限时，密钥管理者会删除 “bill” 对密钥的访问权限：

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

用户可以使用带有 `with passwd login_passwd` 子句的 `alter encryption key` 直接使用登录口令加密密钥副本。但是，使用此方法对于登录关联具有以下缺点：

- 密钥管理者必须向用户提供密钥副本首次分配的口令。
- 用户必须发出 `alter encryption key` 才能使用登录口令重新加密密钥副本。

例如：

- “razi” 为用户 “bill” 添加由显式口令加密的密钥副本：

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

- “razi” 会与 “bill” 共享密钥副本的口令。

- “bill”为了其自身方便，决定使用他的登录口令来加密其密钥副本：
 

```
alter encryption key key1 with passwd "just4bill"
modify encryption with passwd login_passwd
```
- 现在，当“bill”处理加密列时，Adaptive Server会通过“bill”的登录口令访问他的密钥副本。

## 登录口令更改和密钥副本

如果您拥有由一个或多个密钥的登录口令加密的密钥副本，则不需要在更改登录口令后修改密钥副本。在更改登录口令期间，`sp_password`会使用您的旧登录口令解密您的密钥副本，然后使用新登录口令对它们进行重新加密。

如果SSO在不提供旧口令的情况下使用`sp_password`来更改您的口令，则`sp_password`会删除您的密钥副本。这可防止管理员通过已知口令获得对密钥的访问权。在此类强制口令更改之后，密钥管理者必须使用`alter encryption key`为更改了口令的用户添加`login_association`的密钥副本。`sp_password`会忽略脱机数据库，并且对于存储在脱机数据库中的密钥，当数据库恢复到联机时，密钥管理者会按照步骤来恢复丢失的密钥副本口令。请参见第76页的“登录口令丢失”。

在使用`-p`标志启动服务器之后，如果更改用户口令，则密钥管理者可能还需要执行这些步骤。如果使用`-p`标志的SSO还可以通过使用其登录口令进行加密的密钥副本访问密钥，则密钥管理者必须删除并重新创建SSO的密钥副本。

## 删除密钥副本

当用户更换工作或离开公司后，密钥管理者应删除该用户的密钥副本：

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

例如，如果用户“bill”离开了公司，则密钥所有者可以通过删除其密钥副本来防止“bill”访问key1：

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

因为不需要密钥解密，所以对于此命令，Adaptive Server不需要口令。`drop encryption key`将删除基本密钥及其所有副本。





主题	页码
密钥副本的口令丢失	75
登录口令丢失	76
基本密钥口令丢失	76
密钥恢复命令	77
更改加密密钥的所有权	79

## 密钥副本的口令丢失

如果用户丢失了加密密钥的口令，则密钥管理者必须先删除该用户的加密密钥副本，然后将该加密密钥的另一个副本随新口令发送给该用户。

在此示例中，密钥管理者将 `key1` 的副本分配给 “bill”，然后 “bill” 将其 `key1` 的口令更改为只有他自己知道的口令。在丢失口令之后，“bill” 从密钥管理者处请求新的密钥副本。

- 1 密钥管理者删除了 Bill 的密钥副本：

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

- 2 密钥管理者为用户 “bill” 创建了新的 `key1` 副本，并为 “bill” 提供了口令：

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

- 3 “bill” 将自动具有可更改其 `key1` 的副本的权限：

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

## 登录口令丢失

如果“bill”具有由其登录口令加密的密钥副本，但丢失了他的登录口令，您可以使用以下步骤恢复他对加密密钥的访问权限：

- 1 SSO 使用 `sp_password` 向“bill”发出一个新登录口令。Adaptive Server 会删除分配给“bill”以用于登录关联的任何密钥副本，或者已经由“bill”的登录口令加密的密钥副本。
- 2 密钥管理者会遵循常规过程，按照登录关联设置密钥加密。他必须确保已设置主密钥或系统加密口令，然后创建“bill”的密钥副本：

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

此步骤假定密钥管理者仍然知道基本密钥的口令。如果不知道密钥的加密口令，则密钥管理者必须先执行密钥恢复过程。有关详细信息，请参见第 76 页的“基本密钥口令丢失”。

- 3 “bill”下次访问由 k1 加密的数据时，Adaptive Server 会使用“bill”的新登录口令重新加密“bill”的密钥副本。例如，如果 `emp_salary` 由密钥 k1 加密，则以下语句会自动使用“bill”的登录口令重新加密他的密钥副本：

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

## 基本密钥口令丢失

如果基本密钥口令丢失，则密钥管理者可以使用密钥恢复。密钥恢复很重要，因为如果没有口令，密钥管理者就无法更改密钥的口令或者添加密钥副本。

如果所有用户共享通过基本密钥对数据的访问权，并且一个用户忘记了口令，则该用户可以从其他用户或密钥管理者处获得口令。如果没有人记住口令，则对数据的所有访问权都将丢失。因此，Adaptive Server 建议您通过创建可用于恢复的基本密钥副本来备份密钥。此副本称为“密钥恢复副本”。

密钥管理者应该执行以下操作：

- 1 指定一个用户作为密钥恢复者。密钥恢复者的职责是记住密钥恢复副本的口令。
- 2 为密钥恢复者建立基本密钥的副本。在灾难之后需要恢复的每个密钥都必须具有密钥恢复副本。

## 密钥恢复命令

Adaptive Server 不允许通过恢复密钥副本访问数据。密钥恢复副本的存在只是为了提供用于访问基本密钥的备份。

使用以下命令设置恢复密钥副本：

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

其中：

- *base\_key\_passwd* — 是密钥管理者分配给基本密钥的口令。
- *recovery\_passwd* — 是用于保护密钥恢复副本的口令。
- *key\_recovery\_user* — 对其分配了记住密钥恢复口令职责的用户。

在设置密钥恢复副本之后，密钥管理者会与密钥恢复用户共享口令，密钥恢复用户可以使用以下命令改变口令：

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

在密钥恢复过程中，密钥恢复用户会告诉密钥管理者关于密钥恢复副本的口令。密钥管理者会使用以下命令来恢复对基本密钥的访问权：

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

其中：

- *recovery\_key\_passwd* — 是与密钥恢复副本相关联的口令，由恢复密钥用户与密钥管理者共享。Adaptive Server 会使用 *recovery\_key\_passwd* 来解密密钥恢复副本以访问原始密钥。
- *new\_base\_key\_passwd* — 是用于加密原始密钥的口令。Adaptive Server 会使用结果来更新 `sysencryptkeys` 中的基本密钥行。

可能会需要将密钥所有权更改为另一个密钥管理者。请参见第 79 页的“更改加密密钥的所有权”。

以下示例显示了如何设置恢复密钥副本，并在丢失口令之后将它用于密钥恢复：

- 1 密钥管理者将创建由口令保护的新加密密钥。

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

- 2 密钥管理者会为用户 “charlie” 添加 key1 的加密密钥恢复副本。

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

- 3 “Charlie” 为该恢复副本分配不同的口令，并将此口令保存在锁住的抽屉中：

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

- 4 如果密钥管理者丢失基本密钥的口令，则他可以从 “charlie” 处获得该口令，并使用以下命令从恢复副本恢复基本密钥：

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

现在，密钥管理者可通过共享基本密钥口令，或者通过删除和添加密钥副本（在发生人员变动的情况下），与其他用户共享对 key1 的访问权。

## 更改加密密钥的所有权

更改所有权可能会发生在正常业务期间，或者发生在密钥恢复过程中。当此命令由 SSO 执行时，会将密钥所有权移交给命名用户：

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

其中，`user_name` 是新密钥所有者的名称。此用户必须是已在其中创建密钥的数据库中的用户。

例如，如果“razi”是密钥管理者且拥有密钥 `encr_key`，但他将被名为“tinnap”的新密钥管理者取代，则可以使用以下命令更改密钥所有权：

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

只有 SSO 或密钥所有者可以运行此命令。

如果新所有者已具有密钥副本，则将显示：

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

已具有常规密钥副本或恢复密钥副本的用户无法成为该密钥的新所有者。Adaptive Server 不允许为密钥所有者建立密钥副本。

如果以前的密钥所有者已授予对密钥的任何权限，则 `sysprotects` 系统表中的授权者 `uid` 将更改为该密钥的新所有者的 `uid`。所有者更改会立即生效；新所有者无需重新登录便可使更改生效。



主题	页码
审计选项	81
审计值	81
事件名和编号	81
在命令文本审计中屏蔽口令	82
审计密钥管理者的操作	82

## 审计选项

请参见《安全性管理指南》中的“审计”，以了解加密列审计信息（尤其是表 18-5，该表列出了 `event` 和 `extrainfo` 列中的值）。

## 审计值

请参见《安全性管理指南》中的“审计”，以了解 `sysaudits` 的 `event` 列中所显示的值（尤其是表 18-2，该表列出了审计选项、要求和示例）。

## 事件名和编号

可以查询特定审计事件的审计追踪。请使用 `audit_event_name` 并将 `event id` 作为参数。

```
audit_event_name(event_id)
```

请参见《安全性管理指南》中的“审计”，以了解 `sysaudits` 的 `event` 列中所显示的值（尤其是表 18-6，该表列出了审计事件值）。

## 在命令文本审计中屏蔽口令

审计记录中的口令将被屏蔽。例如，如果 SSO 已对数据库 db1 中的用户“alan”启用命令文本审计（即，审计特定用户的所有操作）：

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

并且“alan”发出以下命令：

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

Adaptive Server 会将以下 SQL 文本写入 audit 表的 extrainfo 列：

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

## 审计密钥管理者的操作

要审计 keycustodian\_role 在其中处于活动状态的所有操作，请使用：

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



主题	页码
<a href="#">为加密列编制索引</a>	83
<a href="#">排序顺序和加密列</a>	84
<a href="#">加密列的连接</a>	84
<a href="#">搜索参数和加密列</a>	85
<a href="#">将加密数据作为密文移动</a>	86

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。开销的数量取决于 CPU 和 Adaptive Server 引擎数量、系统上的负载、访问加密数据的并发会话数量以及查询中引用的加密列的数量。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，数据越宽，加密操作中的 CPU 使用率就会越高。

占用时间取决于 Adaptive Server 优化程序能否使用加密列。

## 为加密列编制索引

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，则可以创建该加密列的索引。使用初始化矢量或随机填充可以对不同模式的密文进行相同的数据加密，这样可防止索引强制执行唯一性，并可防止对密文格式的数据执行等同性匹配。

加密数据的索引可用于数据的等同性匹配和非等同性匹配，但不能用于数据排序、范围搜索或查找最大和最小值。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。相反，必须解密每个行中的加密列，然后再进行搜索。这会减慢数据处理的速度。

## 排序顺序和加密列

如果使用不区分大小写的排序顺序，则在 Adaptive Server 执行与另一列的连接操作或基于常量值的搜索时，无法使用加密 char 或 varchar 列的索引。对于不区分变音的排序顺序也是如此。

例如，在不区分大小写的搜索中，字符串 abc 与以下范围内的所有字符串都匹配：abc、Abc、ABc、ABC、AbC、aBC、aBc、abC。Adaptive Server 必须将 abc 与此范围内的值进行比较。相反，如果将字符串 abc 与列数据进行区分大小写的比较，则比较结果仅与相同的列值相匹配，即包含 abc 的列。不区分大小写和区分大小写的列查找之间的主要区别是：不区分大小写匹配要求 Adaptive Server 执行范围搜索，而区分大小写匹配则要求 Adaptive Server 执行等同性搜索。

未加密字符列的索引按照定义的排序顺序对数据进行排序。对于加密列，该索引按照密文值对数据进行排序，与明文值的顺序无关。因此，加密列的索引仅用于等同性和非等同性匹配，而不用于搜索某范围内的值。abc 和 Abc 加密为不同的密文值，且在索引中的存储位置不相邻。

当 Adaptive Server 使用加密列的索引时，它会按密文格式比较列数据。对于区分大小写的数据，您不希望 abc 与 Abc 匹配，且基于等同性匹配的密文连接或搜索最有效。Adaptive Server 可以基于密文值连接列，并可有效匹配 where 子句值。以下示例将加密 maidenname 列：

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

如果在加密 maidenname 时没有使用初始化矢量或随机填充，则 Adaptive Server 将加密 McCarthy，并对 maidenname 执行密文搜索。如果 maidenname 有索引，则搜索将使用该索引。

## 加密列的连接

Adaptive Server 可通过执行密文比较来优化两个加密列的连接，但前提是：

- 连接的列具有相同的数据类型。对于密文比较，char 和 varchar 被视为相同的数据类型，binary 和 varbinary 也是如此。
- 对于 int 和 float 类型，列具有相同的长度。对于 numeric 和 decimal 类型，各列的精度和标度必须相同。
- 相连接的列使用同一密钥加密。

- 连接的列不是表达式的一部分。例如，不能对满足以下条件的连接执行密文连接：t.encyr\_col1 = s.encyr\_col1 +1。
- 该加密密钥是通过将 init\_vector 和 pad 设置为 NULL 而创建的。
- 连接运算符为 “=” 或 “<”。
- 数据使用缺省的排序顺序。

以下示例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 执行以下 select 语句以计算客户每天使用信用卡花费的总金额，而无需解密 customer 或 daily\_xacts 表中的 creditcard 列。

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

## 搜索参数和加密列

对于加密列与常量值的等同性比较和非等同性比较，Adaptive Server 通过仅对常量值加密一次（而不是对表中每一行的加密列都进行解密）来优化列扫描。第 84 页的“加密列的连接”中列出的相同限制在此处适用。

例如：

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

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。如果查询包含其它谓词，Adaptive Server 将选择最小数据集的最有效连接顺序，它通常将针对加密列的搜索放在最后。

如果您的查询包含多个范围搜索，但又没有有用的索引，请在编写查询时，使它最后对加密列执行范围搜索。以下示例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，将 `zipcode` 列定位在对“调整的总收入”加密列的范围搜索之前：

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

### 参照完整性搜索

如果以下两个条件都满足，则参照完整性探查在密文级别匹配：

- 按照前面介绍的规则，主键和外键的数据类型匹配。
- 主键和外键的加密满足连接列的键要求。

## 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。这适用于 `select into` 命令、批量复制和复制。

# 索引

## 英文

- alter encryption key 13
- alter encryption key 命令 16
- alter table, 创建加密 11
- as default 8
- cc\_key
  - 用于构建索引 41
- cc\_key\_new
  - 用于创建加密密钥 11
- CEK, 列加密密钥 13
- create encryption key
  - 权限 10
  - 示例 9
- create encryption key 7, 11, 13
- create encryption key 的参数
  - keylength *num\_bits* 8
  - keylength *num\_bits* 8
  - keyname* 8
- create encryption key 命令 16
- create encryption key 语法 16, 62
- create index 41
- decrypt 权限
  - grant decrypt 42
- decrypt 权限 1
- decrypt\_default 参数 45
- event
  - 编号 81
  - 名称, 语法 81
- exec 命令 42
- for algorithm 8
- grant all 命令, 不授予解密权限 43
- grant decrypt on, 语法 43
- init\_vector 8
- insert 2
- int\_vector 8
- KEK, 密钥加密密钥 13
- key\_length *num\_bits* 8
- keycustodian\_role 59
- keylength 8
- null 8
- pad 8
  - 参数 8
- password\_phrase* 8
- select into 39
  - 加密 39
  - 需要 decrypt 39
- select 命令 56
- sentinel 字节, 附加到密文的后面 51
- set encryption passwd
  - 不要在触发器或过程中放置 70
- sp\_encryption 15
- sp\_encryption, 语法 15
- sp\_help 53
- sysencryptkeys 62
  - 列加密密钥 (CEK) 的存储 13
- update, 透明加密 2
- where 子句, 对加密列中的数据发出命令 56

## B

- 编号, 事件 81
- 名称, 语法 81
- 变量
  - 部分子句 38
- 表
  - 对新表加密 38
- 步骤, 管理, 使用加密列 3
- 部分子句, 变量 38

## C

- 参数
  - key\_length 8
  - keyname* 8
  - null 8
  - password\_phrase* 8

## 索引

- 解密缺省值 45
  - 参照完整性搜索 86
  - 长度
    - 明文数据的 51
    - 最大,加密列的 51
  - 撤消解密权限 43
  - 初始化矢量 51
  - 创建
    - 加密列的索引 41
    - 加密密钥 6
    - 口令,操作说明 15
    - 密钥副本 65
- ## D
- 登录口令
    - 丢失 76
    - 更改 73
  - 丢失
    - 登录口令 76
    - 加密密钥的口令 75
    - 口令,恢复密钥 75
  - 对称加密算法 2
  - 对加密列发出语句,要求 55,56
  - 对密钥使用口令 62
  - 对修改加密列的限制 40

## F

- 返回缺省值而非解密数据 44
- 访问加密数据 55
  - 语法 68
- 浮点数据,加密形式 3
- 副本
  - 创建密钥 65
  - 更改密钥的口令 67
  - 密钥,登录口令更改 73

## G

- 更改密钥的口令 63

## H

- 恢复,密钥命令 77

## J

- 基本密钥 62
  - 口令丢失 76
- 计算列
  - 加密列不能出现在定义中 38
  - 无法加密 38
- 加密
  - select into 39
  - 创建系统加密口令 15
  - 更改密钥 11
  - 加密列 1
    - 列 51
  - 缺省密钥 11
  - 删除 12,57
  - 删除密钥 12
  - 授予对密钥的权限 11
  - 现有表 40
  - 新表 38
- 加密列
  - create index 41
  - 包含在 where 子句中 56
  - 处理 55
  - 对修改的限制 40
  - 连接 84
  - 排序顺序 84
  - 审计 81
  - 使用步骤 3
  - 搜索参数 85
  - 索引 83
  - 增加长度 2
  - 最大内部长度 51
  - 加密列的内部长度,最大 51
  - 加密列的排序顺序 84
  - 加密列支持的功能 1
- 加密密钥
  - 创建 6
  - 创建,创建前的注意事项 6
  - 创建和管理,章 5

- 存储加密 2
- 更改所有权 79
- 更改所有权语法 79
- 加密 2
- 口令 2
- 来自另一数据库 11, 12
- 加密密钥的所有权, 更改语法 79
- 加密数据
  - 访问 55
  - 使用用户口令访问 68
  - 语法用于 40
  - 作为密文移动 86
- 角色
  - 数据访问 61
- 解密
  - 权限 56
- 解密缺省列
  - 查询限定 48
- 解密缺省值
  - insert 和 delete 50
  - 定义 44
  - 权限 46
  - 删除 51
  - 添加和删除 45
  - 隐式授予 49
- 解密缺省值, 列 47
- 解密数据, 返回缺省值而非 44

## K

- 可加密数据类型 37
- 口令
  - alter encryption key 63
  - 登录更改 73
  - 丢失 76
  - 对密钥使用 62
  - 更改密钥副本 67
  - 恢复密钥 75
  - 基本密钥丢失 76
  - 加密密钥丢失 75
  - 使用用户口令访问数据 68
  - 系统加密, 密钥保护 15

- 用户指定 16
- 在命令文本审计中屏蔽 82
- 口令, 变量, 长度 15

## L

- 连接, 加密列 84
- 列
  - 处理加密 55
  - 带有查询限定 48
  - 加密 51
  - 加密, 语法 40
  - 具有解密缺省值 47

## M

- 密文
  - 附加的 sentinel 字节 51
  - 加密数据的移动 86
  - 数据的编码形式 2
  - 增加加密列的长度 2
- 密钥
  - 创建加密 6
  - 从丢失的口令恢复 75
  - 更改 11
  - 删除加密 12
  - 使用口令 62
  - 授予权限 11
  - 与数据分开 11, 12
- 密钥保护 13
- 密钥副本 62
  - 创建 65
  - 登录更改 73
  - 更改口令 67
  - 删除 73
- 密钥管理者 62
  - 管理者, 密钥, 活动 60
  - 角色 59
  - 审计操作 82
- 密钥恢复命令 77
- 密钥加密密钥 (KEK) 13

## 索引

### 明文

- 数据, 长度 51
  - 未加密的数据 2
- 名称, 事件 81
- 命令
- 41
  - alter encryption key 16
  - create encryption key 16
  - exec 42
  - grant all 43
  - select 56
  - select into, 需要列级权限 39
  - 共享口令的语法 77
  - 密钥恢复 77
  - 密钥恢复的语法 77
  - 用于删除解密缺省值 51
- 命令文本审计, 屏蔽口令 82

## P

### 平台

- 适用于所有平台的加密形式 3

## Q

### 权限

- 撤销解密 43
  - 分配受限解密的特权 44
  - 解密 56
  - 解密缺省值 46
  - 限制解密 44
- 缺省加密密钥
- 创建 11
- 缺省值, 返回 44

## S

### 删除

- 加密 12, 57
- 密钥副本 73

- 删除解密缺省值 45, 51
- 命令 51

### 审计

- 加密列 81
- 密钥管理者的操作 82
- 选件 81
- 在命令文本中屏蔽口令 82
- 值 81

- 矢量, 初始化 51

- 授予, 隐式 49

- 数据, 加密, 作为密文移动 86

### 数据访问

- 用户和角色 61

### 数据库

- 不同, 加密密钥 12

- 加密密钥 11

- 数据类型, 可加密 37

### 搜索

- 参照完整性 86

- 搜索参数, 加密列 85

- 随机 8

### 索引

- 加密列 41, 83

## T

- 特权, 分配 44

- 添加解密缺省值 45

### 透明度

- 应用程序 70

- 透明加密 2

## X

- 系统加密口令 15

- 用于创建的操作说明 15

### 现有表

- 加密数据 40

- 限制解密权限 44

- 分配特权 44

- 性能注意事项 83



选件  
    审计 81

## Y

要求

    发出 56  
    发出 insert 55  
    发出 select 56  
    发出 update 55

隐式授予和 decrypt default 49, 50

应用程序透明度 70

用户

    数据访问 61

用户口令

    访问加密数据 68

用户指定的口令 16

用于加密的 create table 部分语法 38

用于密钥保护的系统加密口令 15

语法

    alter encryption key 16, 62

    grant decrypt on 43

    set encryption password 68

    部分, 用于加密 38

    对列进行加密 40

    加密密钥, 更改所有权 79

    密钥副本恢复 77

    删除加密密钥 12

    事件名和编号 81

    与密钥恢复用户共享口令的命令 77

源表, 需要列级权限 39

## Z

整数数据, 加密形式 3

值

    缺省值 47

    审计 81

