IBM DB2 10.1
for Linux, UNIX, and Windows

数据库安全性指南

更新时间 2013 年 1 月

S151-1753-02

IBM DB2 10.1
for Linux, UNIX, and Windows

数据库安全性指南

更新时间 2013 年 1 月

S151-1753-02

注意

使用此信息及其支持的产品前，请先阅读第 337 页的附录 B，『声明』下的常规信息。

修订版声明

此文档包含 IBM 的所有权信息。它在许可协议中提供，且受版权法的保护。本出版物中包含的信息不包括对任何产品的保证，且提供的任何语句都不需要如此解释。

您可在线或通过当地的 IBM 代表处订购 IBM 出版物。

- 要在线订购出版物，请转至 IBM 出版物中心，网址为: http://www.ibm.com/shop/publications/order
- 要查找当地的 IBM 代表处，请转至 IBM 全球联系人目录，网址为: http://www.ibm.com/planetwide/

要从美国或加拿大的 DB2 市场和销售部订购 DB2 出版物，请致电 1-800-IBM-4YOU（426-4968）。

您发送信息给 IBM 后，即授予 IBM 非独占权限，IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

目录

关于本书

本*数据库安全性指南*描述如何使用 DB2® 安全性功能部件来实现和管理安装数据库所必需的安全性级别。

*数据库安全性指南*提供以下内容的详细信息:

- 管理能访问DB2 数据库的用户的权限
- 设置权限以控制用户访问数据库对象和数据

第 1 章 DB2 安全性模型

安全性采用两种方式来控制对 DB2 数据库系统数据和函数的访问。对 DB2 数据库系统的访问由位于 DB2 数据库系统外部的工具来管理（认证），而 DB2 数据库系统内的访问由数据库管理器管理（授权）。

认证

认证就是系统验证用户身份的过程。用户认证是由 DB2 数据库系统外部的安全性工具通过认证安全插件模块来完成的。当您安装 DB2 数据库系统时就包括了依赖于基于操作系统的认证的缺省认证安全插件模块。为方便起见，DB2 数据库管理器还提供了用于 Kerberos 和轻量级目录访问协议（LDAP）的认证插件模块。为了更灵活地适应您特定的认证需要，可以构建您自己的认证安全插件模块。

认证过程将生成一个 DB2 授权标识。用户的组成员资格信息也是在认证期间获得的。缺省情况下获得的组信息依赖于当您安装 DB2 数据库系统时包括的依赖于基于操作系统的组成员资格插件模块。如果愿意，可以通过使用特定的组成员资格插件模块（如 LDAP）来获取组成员资格信息。

权限

对用户进行认证之后，数据库管理器就会确定是否允许该用户访问 DB2 数据或资源。授权是这样一个过程：DB2 数据库管理器通过此过程来获取有关已认证的用户的信息，指示该用户可以执行哪些数据库操作，该用户可以访问哪些数据对象。

以下是可用于授权标识的许可权的不同来源:

1. 主要许可权：直接授予授权标识的许可权。
2. 辅助许可权：授予该授权标识作为其成员的组和角色的许可权。
3. 公用许可权：授予 PUBLIC 的许可权。
4. 上下文敏感许可权：授予可信上下文角色的那些许可权。

可以按下列类别将权限授予用户:

- 系统级别权限

  系统管理员（SYSADM）、系统控制（SYSCTRL）、系统维护（SYSMAINT）和系统监视（SYSMON）权限提供了不同程度的对实例级别函数的控制权。权限提供一种方法来对特权分组和控制实例、数据库和数据库对象的维护和实用程序操作。
- 数据库级别权限

  安全性管理员 (SECADM)、数据库管理员 (DBADM)、访问控制 (ACCESSCTRL)、数据访问 (DATAACCESS)、SQL 管理员 (SQLADM)、工作负载管理管理员 (WLMADM) 以及说明（EXPLAIN）权限提供了数据库内的控制权。其他数据库权限包括 LOAD（能够将数据装入到表中）和 CONNECT（能够连接至数据库）。
- 对象级别权限

  对象级别权限涉及对对象执行操作时检查特权。例如，要从表中进行选择，用户就必须至少对该表具有 SELECT 特权。

- 基于内容的权限

  通过视图可以控制特定用户可以读取一个表中的哪些列或行。基于标号的访问控制 (LBAC) 确定哪些用户对各行和各列具有读写访问权。

可以将这些功能和 DB2 审计设施一起用来监视访问，定义和管理数据库安装需要的安全级别。

认证

用户认证是通过使用 DB2 数据库系统之外的安全性工具来完成的。此安全性工具可以是操作系统的一部分，也可以是一个独立的产品。

安全性工具需要两项来认证用户：用户标识和密码。用户标识向安全性工具标识用户。通过提供正确的密码（只有该用户和安全性工具才知道的信息）来验证该用户的身份（与该用户标识相对应）。

注: 在非 root 用户安装中，必须通过运行 **db2rfe** 命令来启用基于操作系统的认证。

在认证之后:

- 必须使用 SQL 权限名或 *authid* 来向 DB2 标识该用户。此名称可以与用户标识或一个映射值相同。例如，在 UNIX 操作系统上，当您使用缺省安全插件模块时，可将符合 DB2 命名约定的一个 UNIX 用户标识变换为大写字母来获得一个 DB2 *authid*。
- 获取用户所属的组的列表。在授权用户时，可使用组成员资格。组是安全性工具实体，它们也必须映射至 DB2 授权名。执行此映射的方法与映射用户标识的方法类似。

DB2 数据库管理器使用安全性工具以下面两种方式之一来认证用户:

- 将成功的安全性系统登录作为身份证明，并允许:
  - 使用本地命令访问本地数据。
  - 当服务器信赖客户机认证时使用远程连接。
- 将安全性工具成功地对用户标识和密码进行验证来作为身份证明，并允许:
  - 使用远程连接，在此情况下服务器需要认证的证明。
  - 使用操作，在此情况下用户希望以某个不同于登录时所用的标识来运行命令。

注: 在某些 UNIX 系统上，DB2 数据库管理器可将操作系统的失败密码尝试次数记入日志，并检测客户机何时超出允许的登录尝试次数，该值由 LOGINRETRIES 参数指定。

权限

使用 DB2 工具来执行授权。DB2 表和配置文件用于记录与每个授权名关联的许可权。

当已认证的用户尝试访问数据时，会将这些已记录的许可权与下列各项的许可权进行比较:

- 用户的授权名
- 用户所属的组
- 直接或通过组或角色间接授予用户的角色
- 通过可信上下文获取的许可权

根据此比较，DB2 服务器会确定是否允许执行所请求的访问。

记录的许可权类型是特权、权限级别和 LBAC 凭证。

*特权*为授权名定义单个许可权，它使用户能够创建或访问数据库资源。特权存储在数据库目录中。

*权限级别*提供了一种对特权进行分组的方法以及对数据库管理器操作的控制权。特定于数据库的权限存储在数据库目录中；系统权限与组成员关系相关联，并且与权限级别相关联的组名存储在给定实例的数据库管理器配置文件中。

*LBAC 凭证*是 LBAC 安全标号和 LBAC 规则免除权，它们允许访问受基于标号的访问控制 (LBAC) 保护的数据。LBAC 凭证存储在数据库目录中。

组提供了一个简便的方法来对一组用户执行授权，而不必单独对每个用户授予或撤销特权。除非另有指定，否则，组授权名可以用在为了授权而使用授权名的任何地方。通常，对于动态 SQL 和非数据库对象授权（如实例级别命令和实用程序），考虑使用组成员资格，但对于静态 SQL，那么不考虑使用。在授予 PUBLIC 特权时则例外：在处理静态 SQL 时要考虑使用组成员资格。DB2 文档中的适当地方提到了组成员资格不适用的特殊情况。

角色是将一项或多项特权集中在一起的数据库对象，可以使用 GRANT 语句将角色指定给用户、组、PUBLIC 或其他角色，也可以使用 CREATE TRUSTED CONTEXT 或 ALTER TRUSTED CONTEXT 语句将它指定给可信上下文。可以对工作负载定义中的 SESSION_USER ROLE 连接属性指定角色。使用角色时，将对数据库对象的访问许可权与角色关联。然后，作为这些角色的成员的用户将具有对角色定义的特权，通过这些特权可访问数据库对象。

角色提供类似于组的功能；它们对一组用户执行授权，而不必单独地对每个用户授予或撤销特权。角色的一个优点是它们由 DB2 数据库系统管理。在视图、触发器、具体化查询表 (MQT)、程序包和 SQL 例程的授权过程中将考虑授予角色的许可权，这些许可权与授予组的许可权不同。在视图、触发器、MQT、程序包和 SQL 例程的授权过程中不考虑授予组的许可权，这是因为 DB2 数据库系统无法发现组中的成员资格何时更改，因此它无法在适当时使先前提到的对象无效。

注：在视图、触发器、MQT、程序包和 SQL 例程的授权过程中不考虑授予角色的许可权，这些角色已授予组。

在 SQL 语句处理期间，DB2 授权模型考虑的许可权是下列许可权的并集：

1. 授予与 SQL 语句关联的主授权标识的许可权
2. 授予与 SQL 语句关联的辅助授权标识（组或角色）的许可权
3. 授予 PUBLIC 的许可权，包括直接或通过其他角色间接授予 PUBLIC 的角色。
4. 授予可信上下文角色的许可权（如果适用）。

安装和使用 DB2 数据库管理器时的安全性注意事项

从安装产品的那刻起，安全性注意事项对于 DB2 管理员而言是十分重要的。

要完成 DB2 数据库管理器的安装，需要用户标识、组名和密码。基于 GUI 的 DB2 数据库管理器安装程序为不同的用户标识和组创建缺省值。根据是在 Linux 和 UNIX 操作系统上还是在 Windows 操作系统上进行安装，将创建不同的缺省值：

- 在 UNIX 和 Linux 操作系统上，如果您在“实例设置”窗口中选择创建 DB2 实例，那么在缺省情况下，DB2 数据库安装程序将为 DAS (dasusr)、实例所有者 (db2inst) 和受防护用户 (db2fenc) 创建不同的用户。（可选）可以指定不同用户名。

  DB2 数据库安装程序将 1 至 99 的数字追加到缺省用户名后面，直到可以创建尚未存在的用户标识为止。例如，如果用户 db2inst1 和 db2inst2 已存在，那么 DB2 数据库安装程序会创建用户 db2inst3。如果使用大于 10 的数字，那么该名称的字符部分在缺省用户标识中将被截断。例如，如果用户标识 db2fenc9 已存在，DB2 数据库安装程序截断用户标识中的 c，然后追加 10（即 db2fen10）。在将数字值追加到缺省 DAS 用户（例如 dasusr24）时，不发生截断。
- 在 Windows 操作系统上，缺省情况下，DB2 数据库安装程序将为 DAS 用户、实例所有者和受防护用户创建用户 db2admin（只要您愿意，在安装期间也可以指定另一个用户名）。与 Linux 和 UNIX 操作系统不同，不会将任何数字值追加至该用户标识。

除管理员以外的用户可能会知道缺省值，并且在数据库和实例中以不适当的方式来使用这些缺省值，要将这种风险降到最低，请您在安装期间将缺省值更改为您选择的新用户标识或现有用户标识。

注: 响应文件安装不对用户标识或组名使用缺省值。这些值必须在响应文件中指定。

认证用户时，密码非常重要。如果在操作系统级别上未设置认证要求，且数据库正在使用该操作系统来认证用户，那么将允许用户连接。例如，在 Linux 和 UNIX 操作系统上，将未定义的密码视为 NULL。在此情况下，任何不具备已定义密码的用户将被视为具有 NULL 密码。从操作系统的角度来看，这是一种匹配，用户得到验证，并且能够连接到数据库。如果要使操作系统为您的数据库执行用户认证，请使用操作系统级别的密码。

在 Linux 和 UNIX 操作系统上使用分区数据库环境时，缺省情况下，DB2 数据库管理器使用 rsh 实用程序（在 HP-UX 上为 remsh 实用程序）来对远程成员运行一些命令。rsh 实用程序通过网络以明文的方式传送密码，这样，如果 DB2 服务器不是在安全的网络中，那么这种方式可能会导致安全性漏洞。可以使用 **DB2RSHCMD** 注册表变量将远程 shell 程序设置为更安全的备用实用程序以避免此漏洞。更安全的其他方法的一个示例就是 ssh。请参阅 **DB2RSHCMD** 注册表变量文档，以了解远程 shell 配置的限制。

在安装 DB2 数据库管理器之后，还可查看和更改（如果需要）已经授予用户的缺省特权。缺省情况下，安装过程在每种操作系统上均为以下用户授予系统管理（SYSADM）特权:

Linux 和 UNIX 操作系统

属于实例所有者的主组的有效 DB2 数据库用户名。

Windows 环境

- 本地 Administrators 组的成员。
- 域控制器中 Administrators 组的成员（当 DB2 数据库管理器配置为在定义用户的位置上枚举这些用户的组时）。在 Windows 操作系统上，使用 **DB2_GRP_LOOKUP** 环境变量来配置组枚举。
- DB2ADMNS 组的成员（当启用了 Windows 扩展安全性时）。DB2ADMNS 组的位置在安装期间确定。
- LocalSystem 帐户。

通过更新数据库管理器配置参数 **sysadm_group**，管理员可以控制由哪个用户组拥有 SYSADM 特权。您必须遵循以下准则来满足 DB2 数据库安装和后续实例及数据库创建的安全性要求。

任何定义为系统管理组的组（通过更新 **sysadm_group**）都必须存在。此组的名称应该能够让人轻松地识别出是为实例所有者创建的组。属于此组的用户标识和组对相应的实例均具有系统管理员权限。

管理员应该考虑创建可轻松识别为与特定实例相关联的实例所有者用户标识。作为其中一个组成员，此用户标识应该具有先前创建的 SYSADM 组的名称。另一项建议是只将此实例所有者用户标识用作实例所有者组的成员，而且不在任何其他组中使用该标识。这应该控制可以修改该实例的用户标识和组的增加。

创建的用户标识必须与密码相关联，以便在被允许输入实例内的数据和数据库之前提供认证。创建密码时，建议遵循您所在组织的密码命名准则。

注：为了避免意外删除或覆盖实例配置或其他文件，管理员应考虑对直接在服务器上执行的日常管理任务使用另一个用户帐户，而该用户帐户与实例所有者不属于同一个主组。

实例和数据库目录的文件许可权要求

DB2 数据库系统要求实例和数据库目录具有最低级别许可权。

注：如果实例和数据库目录是由 DB2 数据库管理器创建的，那么许可权是正确的，不应更改。

在 UNIX 和 Linux 机器上，实例目录和 NODE000*x*/sqldbdir 目录必须至少具有下列许可权：u=rwx 和 go=rx。下表中说明了这些字母代表的含义：

字符	代表的含义：
u	用户（所有者）
g	组
o	其他用户
r	读
w	写
x	执行

例如，/home 目录中的 db2inst1 实例的许可权为：

```
drwxr-xr-x  36 db2inst1  db2grp1         4096 Jun 15 11:13 db2inst1
```

对于包含数据库的目录，直到并且包括 NODE000*x* 的每个目录级别都需要下列许可权：

```
drwxrwxr-x  11 db2inst1  db2grp1         4096 Jun 14 15:53 NODE0000/
```

例如，如果数据库位于 /db2/data/db2inst1/db2inst1/NODE0000 目录中，那么 /db2、/db2/data、/db2/data/db2inst1、/db2/data/db2inst1/db2inst1 和 /db2/data/db2inst1/db2inst1/NODE0000 这些目录都需要 drwxrwxr-x 许可权。

例如，在 NODE000x 目录中，sqldbdir 目录需要 drwxrwxr-x 许可权：

```
drwx------   5 db2inst1  db2grp1         256 Jun 14 14:17 SAMPLE/
drwxr-x---   7 db2inst1  db2grp1        4096 Jun 14 13:26 SQL00001/
drwxrwxr-x   2 db2inst1  db2grp1         256 Jun 14 13:02 sqldbdir/
```

注意:
为了维护文件的安全性，请不要将 *DBNAME* 目录（例如 SAMPLE）和 SQL*xxxx* 目录的许可权更改为不是 DB2 数据库管理器创建这些目录时所指定的许可权。

认证详细信息

服务器的认证方法

访问实例或数据库首先要求*认证*用户。每个实例的*认证类型*确定对用户进行验证的方式和位置。

认证类型存储在服务器上的配置文件中。它是在创建实例时进行的初始设置。每个实例都有一个认证类型，该类型控制对数据库服务器和该服务器控制下的所有数据库的访问。

如果打算从联合数据库访问数据源，必须考虑数据源认证处理和联合认证类型的定义。

注: 可以访问以下 Web 站点以获取有关 DB2 数据库管理系统用来对用户标识和密码（使用 SERVER_ENCRYPT 认证时）或者用户标识、密码和用户数据（使用 DATA_ENCRYPT 认证时）执行加密的加密例程的认证信息: http://www.ibm.com/security/standards/st_evaluations.shtml。

在显式可信连接上切换用户

对于 CLI/ODBC 和 XA CLI/ODBC 应用程序，在处理需要认证的切换用户请求时使用的认证机制与最初建立可信连接本身时使用的机制相同。因此，对于显式可信连接上的切换用户请求所需的任何认证来说，在建立该可信连接时使用的任何其他协商安全属性（例如，加密算法、加密密钥和插件名称）都相同。通过使用数据源属性，Java™ 应用程序允许对切换用户请求更改认证方法。

因为可以定义可信上下文对象以便在可信连接上切换用户不需要认证，所以为了充分利用显式可信连接上的切换用户功能，用户编写的安全插件必须能够:

- 接受仅用户标识标记
- 对该用户标识返回有效的 DB2 授权标识

注: 如果 CLIENT 类型的认证有效，那么不能建立显式可信连接。

提供的认证类型

提供了下列认证类型:

SERVER

指定在服务器上通过对于该配置有效的安全性机制（例如，通过安全插件模块）进行认证。缺省安全性机制是: 如果在尝试连接期间指定了用户标识和密码，那么会将它们发送至服务器并在服务器上将它们与有效用户标识和密码组合比较，以确定是否允许该用户访问实例。

注: 服务器代码检测一个连接是本地连接还是远程连接。对于本地连接，当认证类型是 SERVER 时，不需用户标识和密码就可认证成功。

SERVER_ENCRYPT

指定服务器接受加密的 SERVER 认证方案。如果未指定客户机认证，使用在服务器中选择的方法认证客户机。当用户标识和密码通过网络从客户机发送至服务器时，它们处于已加密状态。

当客户机与服务器之间协商产生的认证方法为 SERVER_ENCRYPT 时，可以选择通过使用 AES（高级加密标准）256 位算法来对用户标识和密码进行加密。为此，请设置数据库管理器配置参数 **alternate_auth_enc**。此配置参数具有以下三项设置:

- NOT_SPECIFIED（缺省值）表示服务器接受客户机建议的加密算法，其中包括 AES 256 位算法。
- AES_CMP 意味着如果连接客户机建议使用 DES 但支持 AES 加密，那么服务器会针对 AES 加密重新协商。
- AES_ONLY 意味着服务器仅接受 AES 加密。如果客户机不支持 AES 加密，那么连接会被拒绝。

仅当客户机与服务器之间协商的认证方法为 SERVER_ENCRYPT 时，才能使用 AES 加密。

CLIENT

指定使用操作系统安全性在调用应用程序所在的数据库分区上执行认证。在客户机节点上，将在连接期间或尝试连接期间指定的用户标识和密码与有效的用户标识和密码的组合比较，以确定是否允许此用户标识访问该实例。不在数据库服务器上执行其他认证。这有时称为单点登录。

如果用户执行本地登录或客户机登录，那么只有该本地客户机工作站识别该用户。

如果远程实例具有 CLIENT 认证，那么另两个参数确定最终的认证类型: **trust_allclnts** 和 **trust_clntauth**。

仅用于可信客户机的 CLIENT 级安全性:

可信的客户机是具有可靠的、本地安全性系统的客户机。

当已选择认证类型 CLIENT 时，可选择一个附加选项来阻止其操作环境没有固有安全性的客户机访问系统。

要阻止不安全的客户机访问系统，管理员可将 **trust_allclnts** 参数设置为 NO 来选择“可信客户机认证”。这意味着所有可信平台都可以代表服务器认证用户。不可信的客户机是在服务器上进行认证，并且认证时必须提供用户标识和密码。使用 **trust_allclnts** 配置参数来指示您是否信赖客户机。此参数的缺省值是 YES。

注: 可以信赖所有客户机（**trust_allclnts** 为 YES），但其中的某些客户机可以没有用于认证的本机保密安全性系统。

甚至对于可信的客户机，您也可能希望在服务器上完成认证。使用 **trust_clntauth** 配置参数来指示对可信客户机进行验证的位置。此参数的缺省值是 CLIENT。

注: 仅对于可信的客户机，如果在试图 CONNECT 或 **ATTACH** 时没有显式提供用户标识或密码，那么对用户的验证在客户机上进行。**trust_clntauth** 参数仅用于确定对 USER 或 USING 子句上提供的信息进行验证的位置。

为了防止所有客户机（其中包括 z/OS® 和 System i® 上的 JCC 4 类客户机，但不包括 z/OS、OS/390®、VM、VSE 和 System i 上的本机 DB2 客户机）进行未授权的访问，请将 **trust_allclnts** 参数设置为 DRDAONLY。只有这些客户机可信赖，才能执行客户端认证。所有其他客户机必须提供用户标识和密码，以供服务器认证。

trust_clntauth 参数用于确定认证先前提到的客户机的位置: 如果 **trust_clntauth** 是 CLIENT，那么在客户机上进行认证。如果 **trust_clntauth** 是 SERVER，那么未提供用户标识和密码时在客户机上进行认证，提供了用户标识和密码时在服务器上进行认证。

表 1. 使用 TRUST_ALLCLNTS 和 TRUST_CLNTAUTH 参数组合的认证方式。

trust_ allclnts	trust_ clntauth	不可信非 DRDA® 客户机认证（没有用户标识和密码）	不可信非 DRDA 客户机认证（具有用户标识和密码）	可信非 DRDA 客户机认证（没有用户标识和密码）	可信非 DRDA 客户机认证（具有用户标识和密码）	DRDA 客户机认证（没有用户标识和密码）	DRDA 客户机认证（具有用户标识和密码）
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

DATA_ENCRYPT

服务器接受加密的 SERVER 认证方案和用户数据的加密。该认证与 SERVER_ENCRYPT 所示的工作方式相同。当用户标识和密码通过网络从客户机发送至服务器时，它们处于已加密状态。

使用此认证类型时，加密以下用户数据:

- SQL 和 XQuery 语句。
- SQL 程序变量数据。
- 从处理 SQL 或 XQuery 语句和包括数据描述的服务器中输出的数据。
- 从查询获得的某些或所有答案集数据。
- 大对象 (LOB) 数据流动。
- SQLDA 描述符。

DATA_ENCRYPT_CMP

服务器接受加密的 SERVER 认证方案和用户数据的加密。另外，此认证类型允许与不支持 DATA_ENCRYPT 认证类型的下层产品兼容。这些产品允许使用 SERVER_ENCRYPT 认证类型来进行连接，并且不对用户数据进行加密。支持新认证类型的产品必须使用该认证类型。此认证类型仅在服务器的数据库管理器配置文件中有效，而在 **CATALOG DATABASE** 命令上使用该认证类型无效。

KERBEROS

当 DB2 客户机和服务器均位于支持 Kerberos 安全协议的操作系统上时，使用此项。通过使用传统密码术来创建共享密钥，Kerberos 安全性协议作为第三方认证服务执行认证。此密钥成为用户的凭证，在所有请求本地或网络服务的场合中，都使用它来验证用户的身份。此密钥消除了将用户名和密码以明文的方式通过网络传送这一需要。通过使用 Kerberos 安全协议，您能够对远程 DB2 数据库服务器使用单点登录功能。KERBEROS 认证类型在各种操作系统上受支持，请参阅相关信息部分以了解更多信息。

Kerberos 认证工作原理如下所示:

1. 用户对域控制器上的 Kerberos 密钥分发中心 (KDC) 使用域帐户认证登录客户机。密钥分发中心将授予凭单的凭单 (TGT) 发送至客户机。
2. 在连接的第一阶段，服务器将目标主体名称发送至客户机，该主体名称是 DB2 数据库服务器服务的服务帐户名。通过使用服务器的目标主体名称和授予目标的凭证，客户机向授予凭证的服务 (TGS) 请求服务凭单，该凭证也在域控制器中。如果客户机的授予凭单的凭单和服务器的目标主体名称都有效，那么 TGS 向客户机发出服务凭单。记录在数据库目录中的主体名称可能被指定为 name/instance@REALM。（除了 DOMAIN\userID 和 userID@xxx.xxx.xxx.com 格式以外，Windows 还支持此格式。）
3. 客户机使用通信信道（例如，它可能是 TCP/IP）将此服务凭单发送至服务器。
4. 服务器验证客户机的服务凭单。如果客户机的服务凭单有效，那么认证完成。

可能会对客户机上的数据库进行编目，并对服务器的目标主体名称显式指定 Kerberos 认证类型。使用此方法，可忽略连接的第一个阶段。

如果指定了用户标识和密码，那么客户机将请求该用户帐户的授予凭单的凭单并将其用于认证。

KRB_SERVER_ENCRYPT

指定服务器接受 KERBEROS 认证或加密的 SERVER 认证方案。如果客户机认证类型是 KERBEROS，那么使用 Kerberos 安全性系统认证客户机。如果客户机认证类型是 SERVER_ENCRYPT，那么使用用户标识和加密密码认证客户机。如果未指定客户机认证类型，如有可能，客户机将使用 Kerberos，否则它将使用密码加密。对于其他客户机认证类型，将返回一个认证错误。不能将客户机的认证类型指定为 KRB_SERVER_ENCRYPT。

注: Kerberos 认证类型在特定操作系统上运行的客户机和服务器上受支持，请参阅相关信息部分以了解更多信息。对于 Windows 操作系统，客户机和服务器都必须属于同一个 Windows 域或属于可信域。在服务器支持 Kerberos 且某些（但并非全部）客户机支持 Kerberos 认证的情况下，应使用此认证类型。

GSSPLUGIN

指定服务器使用 GSS-API 插件来执行认证。如果未指定客户机认证，服务器将服务器支持的插件列表，包括在数据库管理器配置参数 **srvcon_gssplugin_list** 中列示的任何 Kerberos 插件返回至客户机。客户机从列表中选择在客户机插件目录中找到的第一个插件。如果客户机不支持列表中的任何插件，那么使用 Kerberos 认证方案（如果返回）认证客户机。如果客户机认证是 GSSPLUGIN 认证方案，那么使用列表中第一个支持的插件来认证客户机。

GSS_SERVER_ENCRYPT

指定服务器接受插件认证或加密的服务器认证方案。如果通过插件执行客户机认证，那么使用服务器支持的插件列表中第一个客户机支持的插件来认证客户机。

如果未指定客户机认证且在执行隐式连接（即，生成连接时，客户机不提供用户标识和密码），那么服务器返回服务器支持的插件列表、Kerberos 认证方案（如果列表中的某个插件是基于 Kerberos 的）和加密的服务器认证方案。使用客户机插件目录中找到的第一个支持的插件来认证客户机。如果客户机不支持列表中的任何插件，那么使用 Kerberos 认证方案认证客户机。如果客户机不支持 Kerberos 认证方案，使用加密的服务器认证方案来认证客户机，且连接将因为丢失密码而失败。如果 DB2 提供的 Kerberos 插件适用于操作系统或者为数据库管理器配置参数 **srvcon_gssplugin_list** 指定基于 Kerberos 的插件，那么客户机支持 Kerberos 认证方案。

如果未指定客户机认证且在执行显式连接（即，同时提供用户标识和密码），那么认证类型等价于 SERVER_ENCRYPT。在此情况下，选择使用哪种加密算法来加密用户标识和密码取决于数据库管理器配置参数 **alternate_auth_enc** 的设置。

注:

1. 因为对配置文件本身的访问受到配置文件中信息的保护，所以在更改认证信息时，不要无意中将自己锁定在实例之外。下列数据库管理器配置文件参数控制对实例的访问:
   - **authentication** *
   - **sysadm_group** *
   - **trust_allclnts**
   - **trust_clntauth**
   - **sysctrl_group**
   - **sysmaint_group**

   * 指示两个最重要的参数。

   可以采取一些措施来确保这种情况不会发生: 如果意外将自己锁定在 DB2 数据库系统之外，所有平台上都提供了一个故障保险选项，它将允许您使用具有较高特权的本地操作系统安全性用户，重设通常的 DB2 数据库安全性检查来更新数据库管理器配置文件。此用户*始终*具有更新数据库管理器配置文件并校正该问题的特权。但是，这种绕过安全性检查的做法只限于对数据库管理器配置文件进行本地更新。不能以远程方式或对任何其他 DB2 数据库命令使用故障保险用户。此特殊用户被标识为:
   - UNIX 平台: 实例所有者
   - Windows 平台: 属于本地 『Administrators』 组的人员
   - 其他平台: 由于在其他平台上没有本地安全性，因此所有用户无论如何都要通过本地安全性检查

远程客户机的认证注意事项

当对数据库编目以便进行远程访问时，可在数据库目录条目中指定认证类型。

该认证类型不是必需的认证类型。如果未指定该认证类型，那么客户机将首先尝试使用 SERVER_ENCRYPT 认证类型来建立连接。如果服务器不支持 SERVER_ENCRYPT，那么服务器将返回它支持的认证类型的列表。客户机将使用所列示的第一种认证类型来连接至服务器。在未指定认证类型时，使用 LIST DATABASE DIRECTORY 命令所列示的数据库目录将不会显示认证类型。如果在数据库目录条目中未指定认证类型，那么客户机可能要花更长时间进行连接。如果指定了认证类型，且指定的值与服务器上的值相匹配，认证会立即开始。如果检测出不匹配，那么 DB2 数据库会尝试恢复。恢复可能导致更多的流来协调差异或导致错误（如果 DB2 数据库不能恢复）。在不匹配的情况下，认为服务器上的值是正确的。

认证类型 DATA_ENCRYPT_CMP 旨在允许某些客户机使用 SERVER_ENCRYPT 认证而不是使用 DATA_ENCRYPT 来连接至服务器，这些客户机来自不支持数据加密的前发行版。这种认证在下列情况下不工作:

- 客户机级别为 V7.2。
- 网关级别为 V8 FP7 或更高版本。
- 服务器为 V8 FP7 或更高版本。

在这些情况下，客户机无法连接至服务器。要允许进行连接，必须将客户机升级到 V8 或更高版本，或者使网关级别为 V8 FP6 或更低版本。

通过指定适当的认证类型作为网关处的数据库目录条目来确定连接时使用的认证类型。此方法对于 DB2 Connect™ 方案和分区数据库环境中的客户机和服务器（其中客户机设置了 **DB2NODE** 注册表变量）来说都是可行的。将对目录分区中的认证类型进行编目，以便“跳跃”至适当的分区。在此方案中，将不使用在网关处编目的认证类型，这是因为只在客户机与服务器之间进行协商。

如果需要具有使用不同认证类型的客户机，那么可能需要使用不同的认证类型在网关处对多个数据库别名进行编目。决定了要在网关处编目的认证类型时，可以使认证类型与在客户机和服务器中使用的认证类型相同；或者，可以使用 NOTSPEC 认证类型，但需要了解 NOTSPEC 缺省为 SERVER。

分区数据库认证注意事项

在一个分区数据库中，必须为数据库的每个分区定义同一组用户和组。如果这些定义不相同，那么用户也许是被授权在不同的分区上做不同的事情。

建议所有分区保持一致。

Kerberos 认证

Kerberos 是第三方网络认证协议，它使用共享密钥系统，在不安全的网络环境中安全地认证用户。DB2 数据库系统提供对 AIX®、HP-UX、Solaris、Linux IA32 和 AMD64 及 Windows 操作系统上的 Kerberos 认证协议的支持。

简介

Kerberos 认证由包含三层的系统管理，在该系统中，应用程序服务器与客户机之间交换的是加密服务凭单（而不是明文用户标识和密码对）。这些加密服务凭单称为*凭证*，由称为 Kerberos 密钥分发中心 (KDC) 的单独服务器提供。凭证的生存期有限，并且只有客户机和服务器才能识别凭证。这些功能可减少安全风险，即使凭单在网络上被拦

截也是如此。每个用户（在 Kerberos 术语中称为*主体*）拥有与 KDC 共享的专用加密密钥。总的来说，向 KDC 注册的主体和计算机称为*域*。

Kerberos 的一个关键特征是它提供单点登录环境；用户只需验证一次身份即可访问 Kerberos 域中的资源。此单点登录环境意味着用户可连接至 DB2 数据库服务器而不提供用户标识或密码。另一个优点是简化了用户标识的管理，因为 Kerberos 使用中央存储库来存储主体。最后，Kerberos 支持相互认证，相互认证允许客户机验证服务器的标识。

设置

必须先在所有计算机上安装并配置 Kerberos 层，才能将 Kerberos 与 DB2 数据库系统配合使用。对于典型配置，必须符合以下需求:

- 创建适当主体。
- 确保客户机和服务器计算机及主体属于同一个域或多个可信域。在 Windows 术语中，可信域 (Trusted realm) 称为可信域 (trusted domain)。
- 适当时创建服务器密钥表文件。
- 使所有计算机上的时钟同步。Kerberos 通常允许 5 分钟的时间偏差；如果时间偏差超过 5 分钟，那么尝试获取凭证时会发生提前认证错误。

为 DB2 服务器设置 Kerberos

必须先在所有计算机上安装并配置 Kerberos 层，才能将 Kerberos 认证与 DB2 数据库系统配合使用。对于典型配置，必须遵循此页面上的指示信息。

开始之前

如果正在使用 Linux、Sun Solaris 或 HP-UX 操作系统，请确保系统上未安装 `krb5` 库以外的 Kerberos 库。否则，Kerberos 认证会失败，并且会在 `db2diag` 日志文件中记录一条消息。

如果正在使用 Linux 或 Sun Solaris 操作系统，请卸载 IBM® Network Authentication Service (NAS) Toolkit 的任何实例，并从 **PATH** 系统变量中除去对 NAS 安装路径位置的任何引用。

关于此任务

DB2 数据库是否使用 Kerberos 认证取决于是否使用连接应用程序所提供的凭证成功创建了安全性凭证。而且，只要可用，Kerberos 相互认证就受支持，此时客户机和服务器必须同时证明其身份才能使用 Kerberos。然而，其他 Kerberos 功能（如消息签名或加密）将不可用。

有关在系统上安装和配置 Kerberos 产品的其他详细信息，请参阅 http://www.ibm.com/developerworks/data/library/techarticle/dm-0603see/index.html 或随 Kerberos 产品提供的文档。

DB2 数据库系统的 Kerberos 支持是通过 IBMkrb5 GSS-API 安全插件提供的。此插件用于服务器认证和客户机认证。插件库是在以下位置安装 DB2 期间安装的。

- 在 UNIX 和 Linux 32 位操作系统上: `sqllib/security32/plugin/IBM/client` 和 `sqllib/security32/plugin/IBM/server` 目录

- 在 UNIX 和 Linux 64 位操作系统上: sqllib/security64/plugin/IBM/client 和 sqllib/security64/plugin/IBM/server 目录
- 在 Windows 操作系统上: sqllib\security\plugin\IBM\client 和 sqllib\security\plugin\IBM\server 目录

sqllib/samples/security/plugins 目录中提供了 UNIX 和 Linux 插件 IBMkrb5.C 的源代码。对于 64 位 Windows 操作系统，插件库称为 IBMkrb564.dll。

Kerberos 和组

Kerberos 没有组的概念。因此，DB2 数据库实例依赖本地操作系统来获取 kerberos 主体的组列表。对于 UNIX 和 Linux 操作系统，此依赖需要每个主体的等价系统帐户。例如，对于主体 *name@REALM*，DB2 数据库产品通过查询本地操作系统以获取操作系统用户 *name* 所属的全部组名来收集组信息。如果操作系统用户 *name* 不存在，那么 AUTHID 仅属于 PUBLIC 组。

在 Windows 操作系统上，域帐户与 Kerberos 主体自动关联。无需执行其他步骤来创建单独的操作系统帐户。

Kerberos 密钥表文件

要接受安全上下文请求，UNIX 或 Linux 操作系统上的每个 Kerberos 服务必须将其凭证放置在*密钥表*文件中。此需求适用于 DB2 数据库实例用作服务器主体的那些主体。系统仅在缺省密钥表文件中搜索服务器密钥。有关向密钥表文件添加密钥的指示信息，请参阅随 Kerberos 产品提供的文档。

Windows 操作系统上没有密钥表文件的概念；系统会自动存储和获取主体的凭证。

可使用 **KRB5_KTNAME** 环境变量来指定缺省密钥表文件名。但是，因为该服务器插件在 DB2 数据库引擎进程内运行，所以此环境变量可能不可访问。为了避免这种情况，请使用 **db2set** 命令将 **KRB5_KTNAME** 环境变量添加至 **DB2ENVLIST** 注册表变量:

```
db2set DB2ENVLIST=KRB5_KTNAME
```

因为 Kerberos 未对 Windows 使用密钥表文件，所以此选项仅对 Linux 或 UNIX 服务器可用。

过程

要为 DB2 服务器设置 Kerberos，请执行以下操作:

1. 通过执行下列其中一个步骤来安装 Kerberos:
   - 对于 AIX 操作系统，请在 AIX 上为 DB2 安装 NAS (Network Authentication Services) Toolkit V1.4 或更高版本。可从 https://www.ibm.com/services/forms/preLogin.do?source=dm-nas 下载 NAS 程序包。
   - 对于 Linux 和 HP-UX（仅 64 位）操作系统，请安装操作系统安装介质上包括的 Kerberos 程序包 krb5。
   - 对于 Sun Solaris 操作系统，Kerberos 服务包含在 Solaris R10 中。不需要其他安装。
   - 对于 Windows 操作系统，请在域控制器上启用 Active Directory。
2. 将 DB2 产品配置为使用 Kerberos 插件。请参阅第 196 页的『部署 Kerberos 插件』。

3. 重新启动 DB2 服务器。

Kerberos 的命名和映射

必须先确保客户机和服务器计算机和主体属于同一个域或多个可信域，才能将 Kerberos 与 DB2 数据库系统配合使用。

客户机主体

任何可以接收 Kerberos 凭单进行认证的唯一标识都称为*主体*。Kerberos 主体身份通过由两个部分或多个部分组成的格式（*name@REALM* 或 *name/instance@REALM*）定义。因为在授权标识 (AUTHID) 映射中使用了 *name* 部分，所以 *name* 必须遵循 DB2 数据库命名规则。这些规则将名称限制为 128 个字符并限制字符选择。

注: Windows 操作系统直接将 Kerberos 主体身份与域用户相关联。这暗示 Kerberos 认证不可用于未与域相关联的 Windows 操作系统。而且，Windows 操作系统仅支持使用由两个部分组成的格式来定义主体身份，即，*name@domain*。

授权标识映射

与存在范围通常限于单台计算机的操作系统用户标识不同，Kerberos 主体可以在除它们自己的域之外的域中获得认证。通过使用域名来完全限定主体名称以避免出现重复的主体名称这样的潜在问题。在 Kerberos 中，标准主体名称采用以下格式:

name/instance@REALM

其中 *instance* 可以是用正斜杠 (/) 分隔的多个实例名称，例如，*name/instance1/instance2@REALM*。或者，可省略 *instance* 字段。

域名在网络内定义的所有域中必须唯一。授权标识与主体名称（即，标准主体中的 *name* 字段）之间需要一对一映射。需要此简单映射是因为授权标识被 DB2 数据库管理器用作缺省模式，并且应该轻松地按逻辑派生。注意以下映射导致的潜在问题:

- 名称相同但来自不同域的主体将映射至相同的授权标识。例如，以下两个主体名称都映射至授权标识 gregor1x:
  - gregor1x@EXAMPLE.COM
  - gregor1x@WWW.COM
- 名称相同但在不同实例上的主体将映射至相同的授权标识。例如，以下两个主体名称都映射至授权标识 gregor1x:
  - gregor1x/bigmachine@EXAMPLE.COM
  - gregor1x/littlemachine@EXAMPLE.COM

因此，应遵循以下准则:

- 对于访问 DB2 数据库服务器的所有可信域中的名称，使其名称空间保持唯一。
- 无论实例如何，应使所有具有相同 *name* 字段的主体属于同一用户。

服务器主体

在 UNIX 和 Linux 操作系统上，假设 DB2 数据库实例的服务器主体名称为 *instance name/fully qualified hostname@REALM*。此主体必须能够接受 Kerberos 安全上下文，并且在您启动 DB2 数据库实例之前必须存在，因为插件会在初始化时将服务器名称报告给 DB2 数据库实例。

在 Windows 操作系统上，服务器主体通常由用于启动 DB2 数据库服务的域帐户标识。此情况的一个例外是实例由 LocalSystem 帐户启动时。在此情况下，服务器主体名称报告为 host/*hostname*。仅当客户机和服务器都属于 Windows 域时，此标识才有效。

Windows 操作系统不支持由超过两个部分组成的名称。例如：*component/component@REALM*。Windows 客户机尝试连接至 UNIX 服务器时，这会导致问题。因此，如果需要与 UNIX Kerberos 进行互操作，那么必须在 Kerberos 主体与 Windows 域中的 Windows 帐户之间创建映射。有关指示信息，请参阅适当的 Windows 文档。

通过将 **DB2_KRB5_PRINCIPAL** 环境变量设置为标准服务器主体名称，可覆盖 DB2 服务器在 UNIX 和 Linux 操作系统上使用的 Kerberos 服务器主体名称。仅当发出 **db2start** 命令以重新启动实例后，DB2 数据库系统才识别替换服务器主体名称。

启用 Kerberos 认证

必须先启用 Kerberos 认证，才能将 Kerberos 与 DB2 数据库系统配合使用。

在客户机上启用 Kerberos 认证

要在客户机上启用 Kerberos 认证，请将 **clnt_krb_plugin** 数据库管理器配置参数设置为您要使用的 Kerberos 插件的名称。

对于本地授权，如果 **authentication** 配置参数设置为 KERBEROS 或 KRB_SERVER_ENCRYPT，那么客户机将使用 Kerberos。否则，不会提供客户端 Kerberos 支持。

要点：不会执行任何检查以验证 Kerberos 支持是否可用。

编目数据库时，要对与 DB2 服务器的出站连接启用 Kerberos 认证，应改为将 Kerberos 指定为认证类型，如以下示例所示：

```
CATALOG DATABASE testdb AT NODE testnode
   AUTHENTICATION KERBEROS TARGET PRINCIPAL
   service/host@REALM
```

但是，如果未提供认证信息，那么服务器会向客户机发送服务器主体的名称。

在服务器上启用 Kerberos 认证

要在服务器上启用 Kerberos 认证，请在您对服务器上的 **srvcon_gssplugin_list** 数据库管理器配置参数指定的插件列表中包括特定 Kerberos 插件名称。将该 Kerberos 插件名称包括在此列表中允许客户机扫描服务器并在进行连接时选择 Kerberos 认证方法。

如果此配置参数留为空并且您将 **authentication** 配置参数设置为 KERBEROS 或 KRB_SERVER_ENCRYPT，那么系统改用缺省 Kerberos 插件 IBMkrb5。只能指定一个 Kerberos 插件。

最后，要仅对入局连接认证使用 Kerberos，请将 **svrcon_auth** 参数设置为下列两个选项的其中一个:

- KERBEROS 以仅使用 Kerberos 认证；或
- KRB_SERVER_ENCRYPT 以使用 Kerberos 和 SERVER_ENCRYPT 认证。

如果要对入局连接和本地授权使用 Kerberos，请将 **svrcon_auth** 配置参数留为空并将 **authentication** 配置参数的值设置为其中一个 Kerberos 选项。

创建 Kerberos 插件

要在 DB2 数据库系统上定制 Kerberos 认证的行为，可开发您自己的 Kerberos 认证插件。

创建 Kerberos 插件时考虑以下几点:

- 将 Kerberos 插件编写为 GSS-API 插件，但在初始化函数中，对于返回至 DB2 数据库实例的函数指针数组，将 *plugintype* 变量设置为 DB2SEC_PLUGIN_TYPE_KERBEROS。
- 在某些情况下，服务器将服务器主体名称报告给客户机。Kerberos 插件必须以 GSS_C_NT_USER_NAME 格式（即，*server/host@REALM*）指定主体。GSS_C_NT_HOSTBASED_SERVICE 格式（即，*service@host*）不受支持。

Kerberos 兼容性

DB2 Kerberos 认证与 IBM System z®、IBM i 和 Windows 系统兼容。

IBM System z 和 IBM i 兼容性

要连接至 IBM System z 或 IBM i 系统上的数据库，必须使用 **CATALOG DATABASE** 命令的 **AUTHENTICATION** 和 **KERBEROS TARGET PRINCIPAL** 参数来编目数据库。

IBM System z 和 IBM i 操作系统都不支持 Kerberos 的相互认证安全功能。

Windows 问题

在 Windows 操作系统上使用 Kerberos 时，应注意以下问题:

- 因为 Windows 操作系统检测和报告某些错误的方式，所以以下情况会导致客户机安全插件错误。
  - 帐户到期
  - 密码无效
  - 密码到期
  - 管理员强制更改了密码
  - 帐户被禁用

  此外，在所有情况下，DB2 管理日志或 **db2diag** 日志文件都包含登录失败或登录被拒绝消息。
- 如果域帐户名也是在本地定义的，那么显式指定域名和密码的连接将失败，并且出现下列错误: 无法与本地安全机构联系。该错误是由于 Windows 操作系统先查找本地用户造成的。解决方案是在连接字符串中对用户进行完全限定，例如，name@DOMAIN.IBM.COM。
- Windows 帐户的名称中不能包括 at 字符 (@)，因为 DB2 Kerberos 插件假定该字符是域名分隔符。

- 如果客户机和服务器都在 Windows 操作系统上，那么可使用 LocalSystem 帐户启动 DB2 服务。但是，如果客户机和服务器在不同的域中，那么连接可能失败，并且出现目标主体名称无效这一错误。为避免此错误，应通过 **CATALOG DATABASE** 命令（使用标准服务器主机名和标准域名）在客户机上显式编目目标主体。使用以下格式：*host/server hostname@server domain name*。例如，host248/server34.toronto.ibm.com@TORONTO.IBM.COM。使用 LocalSystem 帐户的替代方法是使用有效域帐户。

在服务器上维护密码

您可能需要执行密码维护任务。因为在服务器上通常需要这样的任务，并且许多用户无法使用或不能很好地使用服务器环境，所以执行这些任务可能会比较困难。DB2 数据库系统提供了一种不必在服务器上就能更新和验证密码的方法。

当连接至下列服务器上的数据库时，可以为所指示（和更高）的发行版指定新密码：AIX 和 Windows 操作系统上的 DB2 Universal Database™ V8、Linux 操作系统上的 DB2 V9.1 FP3 或更高版本、DB2 for z/OS V7 和 DB2 for i V6R1。

例如，如果接收到错误消息 SQL1404N“密码过期”或 SQL30082N“安全处理失败，原因为 1（密码过期）”，那么使用 CONNECT 语句来按如下所示更改密码：

```
CONNECT TO database USER userid USING password NEW new_password CONFIRM new_password
```

权限、特权和对象所有权

仅当用户（由授权标识标识）具有执行指定函数的权限时，他们才能成功执行操作。要创建表，必须授权用户创建表；要改变表，必须授权用户改变表；等等。

数据库管理器要求对每个用户特别授权，以使用执行特定任务所需的每个数据库函数。用户可以获取必需权限的方式如下：通过对其用户标识授予该权限，或借助拥有该权限的角色或组的成员资格。

存在三种形式的权限：*管理权限*、*特权*和 *LBAC 凭证*。此外，对象的所有权会带给它对所创建对象的某种程度的权限。下一节中讨论了这些形式的权限。

管理权限

拥有管理权限的人管理控制数据库管理器的任务并负责数据的安全性和完整性。

系统级别权限

系统级别权限提供了不同程度的对实例级别函数的控制权：

- SYSADM（系统管理员）权限

  SYSADM（系统管理员）权限提供了对数据库管理器所创建和维护的全部资源的控制权。系统管理员拥有下列全部权限：SYSCTRL、SYSMAINT 和 SYSMON 权限。具有 SYSADM 权限的用户负责控制数据库管理器并确保数据的安全和完整性。

- SYSCTRL 权限

SYSCTRL 权限提供了对影响系统资源的操作的控制权。例如，具有 SYSCTRL 权限的用户可以创建、更新、启动、停止或删除数据库。此用户还可以启动或停止实例，但不能访问表数据。具有 SYSCTRL 权限的用户还具有 SYSMON 权限。

- SYSMAINT 权限

  SYSMAINT 权限提供在所有与实例关联的数据库上执行维护操作所需的权限。具有 SYSMAINT 权限的用户可以更新数据库配置、备份数据库或表空间、复原现有数据库并监视数据库。类似于 SYSCTRL，SYSMAINT 不提供对表数据的访问。具有 SYSMAINT 权限的用户也具有 SYSMON 权限。

- SYSMON（系统监视）权限

  SYSMON（系统监视）权限提供使用数据库系统监视器所需的权限。

数据库级别权限

数据库级别权限提供了数据库内的控制权:

- DBADM（数据库管理员）

  DBADM 权限级别提供对单个数据库的管理权限。此数据库管理员拥有创建对象和发出数据库命令所需的特权。

  DBADM 权限只能由具有 SECADM 权限的用户授予。不能将 DBADM 权限授予 PUBLIC。

- SECADM（安全性管理员）

  SECADM 权限级别针对安全性提供对单个数据库的管理权限。安全性管理员权限能够管理数据库安全性对象（数据库角色、审计策略、可信上下文、安全标号组件和安全标号）以及授予和撤销所有数据库特权和权限。具有 SECADM 权限的用户可以转移不属于他们的对象的所有权。他们可以使用 AUDIT 语句将审计策略与服务器中的特定数据库或数据库对象关联。

  SECADM 权限没有访问存储在表中的数据的固有特权。它只能由具有 SECADM 权限的用户授予。不能将 SECADM 权限授予 PUBLIC。

- SQLADM（SQL 管理员）

  SQLADM 权限级别提供在单个数据库内监视和调整 SQL 语句的管理权限。它可由具有 ACCESSCTRL 或 SECADM 权限的用户授予。

- WLMADM（工作负载管理管理员）

  WLMADM 权限提供管理工作负载管理对象（如服务类、工作操作集、工作类集以及工作负载）的管理权限。它可由具有 ACCESSCTRL 或 SECADM 权限的用户授予。

- EXPLAIN（说明权限）

  EXPLAIN 权限级别提供在没有获得数据访问权的情况下说明查询方案的管理权限。它只能由具有 ACCESSCTRL 或 SECADM 权限的用户授予。

- ACCESSCTRL（访问控制权限）

ACCESSCTRL 权限级别提供发出以下 GRANT（和 REVOKE）语句的管理权限。

– GRANT（数据库权限）

  ACCESSCTRL 权限不会使拥有者能够授予 ACCESSCTRL、DATAACCESS、DBADM 或 SECADM 权限。只有具有 SECADM 权限的用户才能授予这些权限。

– GRANT（全局变量特权）
– GRANT（索引特权）
– GRANT（模块特权）
– GRANT（程序包特权）
– GRANT（例程特权）
– GRANT（模式特权）
– GRANT（序列特权）
– GRANT（服务器特权）
– GRANT（表、视图或昵称特权）
– GRANT（表空间特权）
– GRANT（工作负载特权）
– GRANT（XSR 对象特权）

ACCESSCTRL 权限只能由具有 SECADM 权限的用户授予。不能将 ACCESSCTRL 权限授予 PUBLIC。

• DATAACCESS（数据访问权限）

  DATAACCESS 权限级别提供下列特权和权限。

  – LOAD 权限
  – 对表、视图、昵称和具体化查询表的 SELECT、INSERT、UPDATE 和 DELETE 特权
  – 对程序包的 EXECUTE 特权
  – 对模块的 EXECUTE 特权
  – 对例程的 EXECUTE 特权

    对下列审计例程除外：AUDIT_ARCHIVE、AUDIT_LIST_LOGS 和 AUDIT_DELIM_EXTRACT。

  – 对所有全局变量的 READ 特权和对除只读变量外的所有全局变量的 WRITE 特权
  – 对所有 XSR 对象的 USAGE 特权
  – 对所有序列的 USAGE 特权

  它只能由拥有 SECADM 权限的用户授予。不能将 DATAACCESS 权限授予 PUBLIC。

• 数据库权限（非管理）

  要执行诸如创建表或例程，或者用于将数据装入表等的活动，需要特定数据库权限。例如，使用 **load** 实用程序将数据装入到表中需要 LOAD 数据库权限（用户还必须具有对表的 INSERT 特权）。

特权

特权是执行操作或任务的许可权。授权用户可以创建对象、有权访问他们拥有的对象并可以使用 GRANT 语句将对他们自己的对象的特权传递给其他用户。

可以对单个用户、组或 PUBLIC 授予特权。PUBLIC 是一个由所有用户（包括将来的用户）组成的特殊的组。如果支持组，属于组成员的用户将间接利用授予组的特权。

CONTROL 特权：拥有对对象的 CONTROL 特权允许用户访问该数据库对象，并授予和撤销其他用户对该对象的特权。

注： CONTROL 特权只应用于表、视图、昵称、索引和程序包。

如果其他用户需要对该对象的 CONTROL 特权，那么具有 SECADM 或 ACCESSCTRL 权限的用户可授予对该对象的 CONTROL 特权。但是，无法撤销对象所有者的 CONTROL 特权，可以使用 TRANSFER OWNERSHIP 语句来更改对象所有者。

个别特权：可以授予个别特权以允许用户对特定对象执行特定任务。具有管理权限 ACCESSCTRL 或 SECADM 或者具有 CONTROL 特权的用户可以对用户授予和撤销该特权。

个别特权和数据库权限允许特定功能，但不包括授予其他用户相同特权或权限的权限。授予其他人表、视图、模式、程序包、例程和序列特权的权限可以通过 GRANT 语句上的 WITH GRANT OPTION 扩展至其他用户。但是，WITH GRANT OPTION 不允许授予该特权的人在授权后撤销该特权。必须具有 SECADM 权限、ACCESSCTRL 权限或 CONTROL 特权才能撤销该特权。

对程序包或例程中的对象的特权：当用户具有执行程序包或例程的特权时，他们不必拥有对该程序包或例程中使用的对象的特定特权。如果程序包或例程包含静态 SQL 或 XQuery 语句，那么该程序包所有者的特权用于这些语句。如果程序包或例程包含动态 SQL 或 XQuery 语句，那么用于特权检查的授权标识取决于发出动态查询语句的程序包的 **DYNAMICRULES BIND** 选项设置，以及发出这些语句时是否正在例程的上下文中使用该程序包（在以下审计例程上例外：AUDIT_ARCHIVE、AUDIT_LIST_LOGS 和 AUDIT_DELIM_EXTRACT）。

可以将个别特权或权限的任何组合授予一个用户或组。当特权与对象关联时，对象必须已存在。例如，除非先前已创建一个表，否则不能授予用户对该表的 SELECT 特权。

注： 当授予一个表示用户或组的权限名权限和特权且没有使用该权限名创建的用户或组时，必须小心。稍后，可以使用该权限名创建一个用户或组，并且该用户或组自动接收与该权限名关联的所有权限和特权。

REVOKE 语句用于撤销先前授予的特权。撤销权限名的特权会撤销所有权限名授予的特权。

撤销权限名称的特权不会撤销任何其他权限名称的相同特权，此特权由该权限名称授予。例如，假定 CLAIRE 将 SELECT WITH GRANT OPTION 授予 RICK，然后 RICK 将 SELECT 授予 BOBBY 和 CHRIS。如果 CLAIRE 撤销 RICK 的 SELECT 特权，那么 BOBBY 和 CHRIS 仍保留 SELECT 特权。

LBAC 凭证

基于标签的访问控制 (LBAC) 使安全性管理员能够准确地确定对于各行各列具有写访问权的用户和具有读访问权的用户。安全性管理员通过创建安全策略来配置 LBAC 系统。安全策略描述的是用来确定哪些用户能够访问哪些数据的条件。对于任何一个表，只能使用一个安全策略来保护它，但不同的表可以由不同的安全策略保护。

创建安全策略之后，安全性管理员将创建称为安全标号和免除权的数据库对象，这些对象是安全策略的组成部分。安全标号描述一组安全条件。免除权遵循一个规则，即，在拥有免除权的用户访问受安全策略保护的数据时，不需要强制对该用户比较安全标号。

一旦创建安全标号，就可以使其与各个表列和表行相关联以保护存放在那些位置中的数据。受安全标号保护的数据称为受保护数据。安全性管理员通过将安全标号授予用户来允许该用户访问受保护数据。当用户尝试访问受保护数据时，该用户的安全标号将与用于保护该数据的安全标号进行比较。用于进行保护的标号将阻塞一部分安全标号。

对象所有权

创建一个对象时，可将该对象的*所有权*分配给一个授权标识。所有权是指用户有权在任何适用的 SQL 或 XQuery 语句中引用此对象。

在模式内创建对象时，此语句的授权标识必须具有在隐式或显式指定模式中创建对象所需的特权。即，权限名称必须是模式的所有者或对模式拥有 CREATEIN 特权。

注: 创建表空间、缓冲池或数据库分区组时，此要求不适用。这些对象并非在模式中创建。

创建对象时，语句的授权标识是此对象的定义者；缺省情况下，在创建此对象之后，语句的授权标识是此对象的所有者。

注: 例外情况是: 如果对 CREATE SCHEMA 语句指定 AUTHORIZATION 选项，那么作为 CREATE SCHEMA 操作部分创建的其他任何对象由 AUTHORIZATION 选项指定的授权标识所有。但是，初始 CREATE SCHEMA 操作后在模式中创建的任何对象由与特定 CREATE 语句关联的授权标识拥有。

例如，语句 `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C1 INT)` 创建模式 `SCOTTSTUFF` 和表 `SCOTTSTUFF.T1`，这两者均属 SCOTT 所有。假设用户 BOBBY 对 `SCOTTSTUFF` 模式拥有 CREATEIN 特权，并在 `SCOTTSTUFF.T1` 表上创建索引。因为索引在模式之后创建，因此 BOBBY 在 `SCOTTSTUFF.T1` 上拥有索引。

根据要创建的对象类型向对象所有者分配特权:

- 对新创建的表、索引和程序包隐式授予 CONTROL 特权。此特权允许对象创建程序访问数据库对象，并授予和撤销其他用户对此对象的特权。如果其他用户需要对该对象的 CONTROL 特权，那么具有 ACCESSCTRL 或 SECADM 权限的用户必须授予对该对象的 CONTROL 特权。对象所有者无法撤销 CONTROL 特权。
- 如果对象所有者对视图定义引用的所有表、视图和昵称具有 CONTROL 特权，那么对新创建的视图隐式授予 CONTROL 特权。
- 其他对象（如触发器、例程、序列、表空间和缓冲池）没有与其关联的 CONTROL 特权。但是，对象所有者自动会接收到与对象关联的各项特权，并且这些特权带有

WITH GRANT OPTION（如果受支持）。因此，对象所有者可以通过使用 GRANT 语句来向其他用户提供这些特权。例如，如果 USER1 创建了表空间，那么 USER1 会自动对此表空间具有带 WITH GRANT OPTION 的 USEAUTH 特权，并且可以将 USEAUTH 特权授予其他用户。此外，对象所有者可以改变或删除对象，或为对象添加注释。这些权限对于对象所有者是隐式的且不能撤销。

所有者可以授予对对象的某些特权（例如，改变表），并且具有 ACCESSCTRL 或 SECADM 权限的用户可以撤销所有者对对象的这些特权。所有者不能授予对对象的某些特权（例如，注释表），并且不能撤销所有者对对象的这些特权。使用 TRANSFER OWNERSHIP 语句将这些特权转移给另一个用户。创建对象时，语句的授权标识是此对象的定义者；缺省情况下，在创建此对象之后，语句的授权标识是此对象的所有者。但是，当使用 **BIND** 命令来创建程序包并指定 **OWNER** *authorization id* 选项时，由程序包中静态 SQL 语句创建的对象的所有者是 *authorization id* 的值。此外，如果在 CREATE SCHEMA 语句中指定了 AUTHORIZATION 子句，那么在 AUTHORIZATION 关键字后面指定的权限名是模式的所有者。

安全性管理员或对象所有者可以使用 TRANSFER OWNERSHIP 语句来更改数据库对象的所有权。因此，管理员可为授权标识创建一个对象，方法是将授权标识用作限定词来创建对象，然后使用 TRANSFER OWNERSHIP 语句将管理员对该对象的所有权转移给授权标识。

权限概述

在实例级别和数据库级别上存在各种管理权限。这些管理权限分组为某些特权和权限，以便您能够将它们授予在数据库安装过程中负责这些任务的用户。

实例级别权限

实例级别权限使您能够执行实例范围的函数，例如，创建和升级数据库、管理表空间以及监视实例上的活动和性能。任何实例级别权限都不提供对数据库表中数据的访问权。下图概述了每个实例级别管理权限提供的功能:

- SYSADM - 供用户将实例作为整体来管理
- SYSCTRL - 供用户管理数据库管理器实例
- SYSMAINT - 供用户在实例内维护数据库
- SYSMON - 供用户监视实例及其数据库

具有较高级别权限的用户还能够执行较低级别权限提供的功能。例如，具有 SYSCTRL 权限的用户还可以执行具有 SYSMAINT 和 SYSMON 权限的用户的函数。

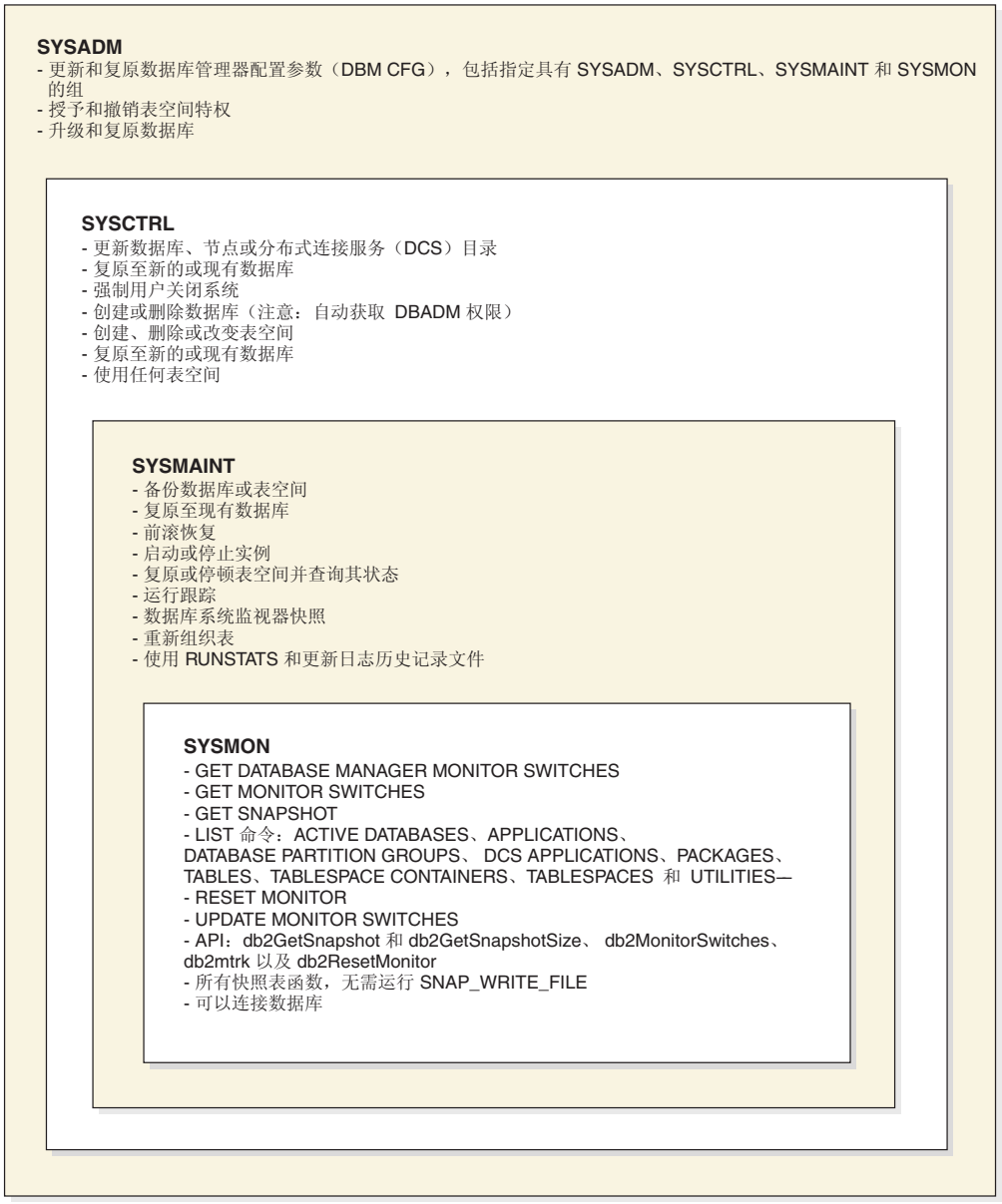


图 1. 实例级别权限

数据库级别权限

数据库级别权限使您能够在特定数据库内执行函数，例如授予和撤销特权，插入、选择、删除和更新数据以及管理工作负载。下图概述了每个数据库级别权限提供的功能。管理数据库权限包括:

- SECADM - 供用户在数据库内管理安全性
- DBADM - 供用户管理数据库
- ACCESSCTRL - 供需要授予和撤销权限及特权的用户使用（除 SECADM、DBADM、ACCESSCTRL 和 DATAACCESS 权限之外，需要 SECADM 权限才能授予和撤销这些权限）
- DATAACCESS - 供需要访问数据的用户使用
- SQLADM - 供监视和调整 SQL 查询的用户使用

- WLMADM - 供管理工作负载的用户使用
- EXPLAIN - 供需要说明查询方案的用户使用（EXPLAIN 权限本身不会提供对数据的访问权）

下图显示了适当时哪些较高级别权限包括较低级别权限提供的功能。例如，具有 DBADM 权限的用户可以执行具有 SQLADM 和 EXPLAIN 权限的用户的函数以及具有 WLMADM 权限的用户的所有函数（授予对工作负载的 USAGE 特权除外）。

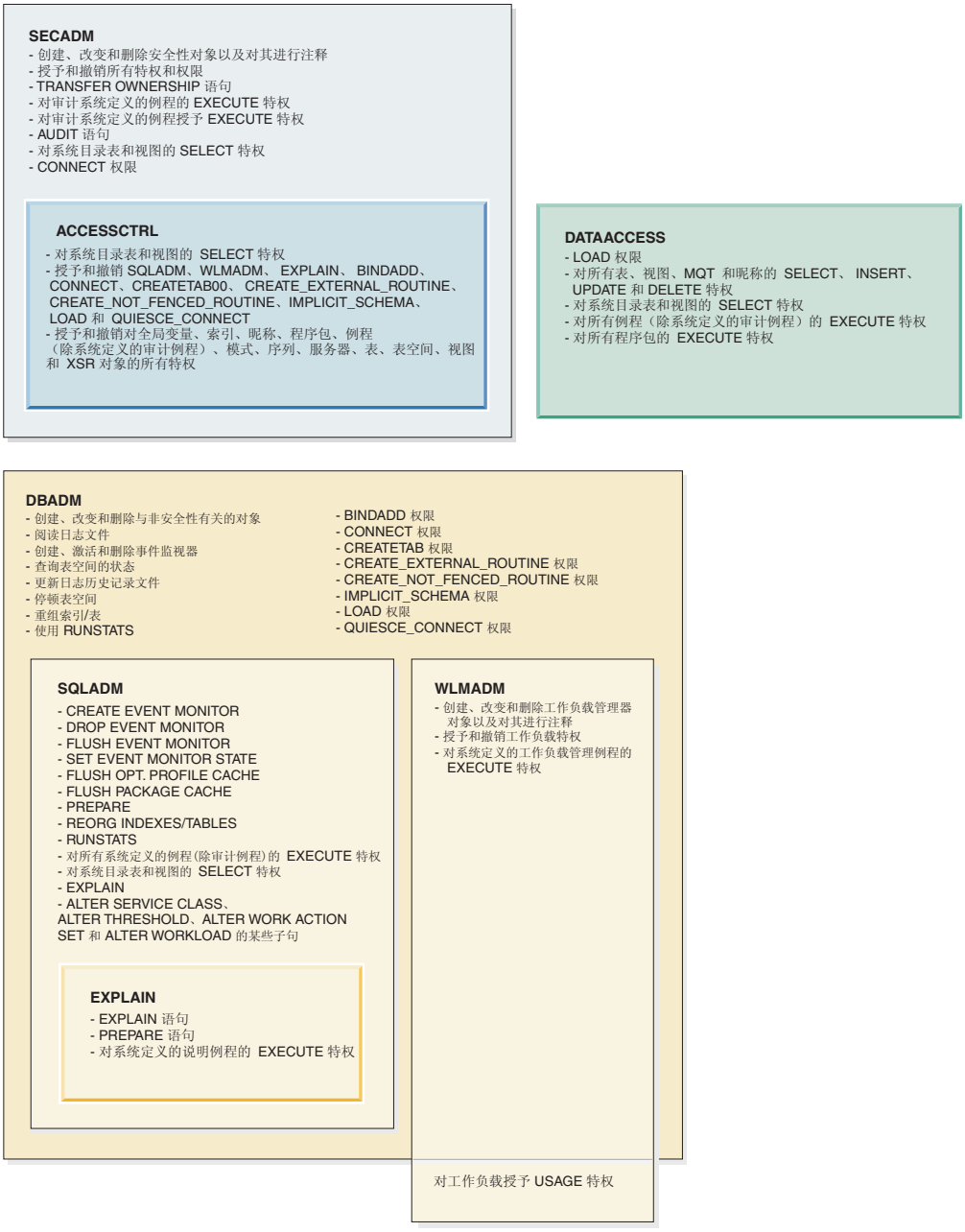


图 2. 数据库级别权限

实例级别权限

系统管理权限（SYSADM）

SYSADM 权限级别是实例级别上最高级别的管理权限。具有 SYSADM 权限的用户可以在实例内运行一些实用程序以及发出一些数据库和数据库管理器命令。

对 **sysadm_group** 配置参数指定的组指定 SYSADM 权限。通过平台上使用的安全性工具来从数据库管理器外部控制该组的成员资格。

只有具有 SYSADM 权限的用户才可以执行下列功能:

- 升级数据库
- 复原数据库
- 更改数据库管理器配置文件（其中包括指定具有 SYSADM、SYSCTRL、SYSMAINT 或 SYSMON 权限的组）

具有 SYSADM 权限的用户可以授予和撤销表空间特权，还可以使用任何表空间。

注: 当具有 SYSADM 权限的用户创建数据库时，会自动授予该用户对该数据库的 ACCESSCTRL、DATAACCESS、DBADM 和 SECADM 权限。如果要防止该用户以数据库管理员或安全性管理员身份访问该数据库，那么必须显式地撤销该用户的这些数据库权限。

在 V9.7 之前的发行版中，SYSADM 权限包括了隐式 DBADM 权限并且还提供了授予和撤销所有权限和特权的功能。在 V9.7 中，DB2 授权模型已更新为明确地区分系统管理员、数据库管理员和安全性管理员的职责。作为此增强的一部分，由 SYSADM 权限提供的功能已减少。

在 V9.7 中，仅 SECADM 权限提供授予和撤销所有权限和特权的功能。

为了让拥有 SYSADM 权限的用户获取 V9.5 中的功能（除了授予 SECADM 权限的功能），安全性管理员必须显式授予该用户 DBADM 权限并且授予该用户新的 DATAACCESS 和 ACCESSCTRL 权限。可以通过将 GRANT DBADM ON DATABASE 语句与该语句的缺省选项 WITH DATAACCESS 和 WITH ACCESSCTRL 配合使用来授予这些新权限。DATAACCESS 权限是允许对特定数据库中的数据进行访问的权限，而 ACCESSCTRL 权限是允许用户在特定数据库中授予和撤销特权以及非管理权限的权限。

有关 Windows LocalSystem 帐户的注意事项

在 Windows 系统上，当未指定数据库管理器配置参数 **sysadm_group** 时，LocalSystem 帐户被认为是系统管理员（拥有 SYSADM 权限）。在 V9.7 中，SYSADM 权限作用域中的更改会影响由 LocalSystem 运行的任何 DB2 应用程序。这些应用程序通常是以 Windows 服务形式编写的，并且使用作为服务登录帐户的 LocalSystem 帐户来运行。如果需要这些应用程序执行不再属于 SYSADM 作用域内的数据库操作，那么必须将必需的数据库特权或权限授予 LocalSystem 帐户。例如，如果应用程序需要数据库管理员功能，请使用 GRANT（数据库权限）语句将 DBADM 权限授予 LocalSystem 帐户。请注意，LocalSystem 帐户的授权标识是 SYSTEM。

系统控制权限 (SYSCTRL)

SYSCTRL 权限是最高级别的系统控制权限。此权限提供对数据库管理器实例及其数据库执行维护和实用程序操作的功能。这些操作可以影响系统资源，但是它们并不允许对数据库中数据的直接访问。

系统控制权限旨在供用户管理包含敏感数据的数据库管理器实例户。

对 **sysctrl_group** 配置参数指定的组指定 SYSCTRL 权限。如果指定了一个组，那么通过平台上使用的安全性工具从数据库管理器外部控制该组的成员资格。

只有具备 SYSCTRL 权限或更高权限的用户才能执行下列操作:

- 更新数据库、节点或分布式连接服务 (DCS) 目录
- 强制用户从系统注销
- 创建或删除一个数据库
- 删除、创建或改变一个表空间
- 使用任何表空间
- 复原到现有或新的数据库。

另外，具有 SYSCTRL 权限的用户可以执行具有系统维护权限 (SYSMAINT) 和系统监视权限 (SYSMON) 的用户的函数。

具有 SYSCTRL 权限的用户也有与一个数据库连接的隐式特权。

注: 当具有 SYSCTRL 权限的用户创建数据库时，会自动授予他们对数据库的显式 ACCESSCTRL、DATAACCESS、DBADM 和 SECADM 权限。如果从 SYSCTRL 组中除去了数据库创建者，而且还要防止其以管理员身份访问该数据库，那么必须显式撤销先前提到的四项管理权限。

系统维护权限（SYSMAINT）

SYSMAINT 权限是第二级别的系统控制权限。此权限提供对数据库管理器实例及其数据库执行维护和实用程序操作的功能。这些操作可以影响系统资源，但是它们并不允许对数据库中数据的直接访问。

系统维护权限是为某些用户设计的，这些用户维护包含敏感数据的数据库管理器实例中的数据库。

对 **sysmaint_group** 配置参数指定的组指定 SYSMAINT 权限。如果指定了一个组，那么通过平台上使用的安全性工具从数据库管理器外部控制该组的成员资格。

只有具有 SYSMAINT 或更高系统权限的用户才可以执行下列操作:

- 备份数据库或表空间
- 复原为现有的数据库
- 执行前滚恢复
- 启动或停止实例
- 复原表空间
- 使用 **db2trc** 命令来运行跟踪
- 生成数据库管理器实例或其数据库的数据库系统监视器快照。

具有 SYSMAINT 权限的用户可以执行下列操作:

- 查询表空间的状态
- 更新日志历史记录文件
- 停顿表空间
- 重组表
- 使用 **RUNSTATS** 实用程序收集目录统计信息。

具有 SYSMAINT 权限的用户也有与数据库连接的隐式特权，并且可以执行具有系统监视权限（SYSMON）的用户的函数。

系统监视权限（SYSMON）

SYSMON 权限提供生成数据库管理器实例或其数据库的数据库系统监视器快照的功能。

SYSMON 权限已分配给 **sysmon_group** 配置参数指定的组。如果指定了一个组，那么通过平台上使用的安全性工具从数据库管理器外部控制该组的成员资格。

SYSMON 权限使用户可以运行下列命令:

- **GET DATABASE MANAGER MONITOR SWITCHES**
- **GET MONITOR SWITCHES**
- **GET SNAPSHOT**
- LIST（一些命令）:
  - **LIST ACTIVE DATABASES**
  - **LIST APPLICATIONS**
  - **LIST DATABASE PARTITION GROUPS**
  - **LIST DCS APPLICATIONS**
  - **LIST PACKAGES**
  - **LIST TABLES**
  - **LIST TABLESPACE CONTAINERS**
  - **LIST TABLESPACES**
  - **LIST UTILITIES**
- **RESET MONITOR**
- **UPDATE MONITOR SWITCHES**

SYSMON 权限使用户可以使用下列 API:

- db2GetSnapshot - 获取快照
- db2GetSnapshotSize - 估计 db2GetSnapshot() 输出缓冲区所需的大小
- db2MonitorSwitches - 获取/更新监视开关
- db2mtrk - 内存跟踪程序
- db2ResetMonitor - 重置监视器

SYSMON 权限使用户可以使用下列 SQL 表函数:

- 无需预先运行 SYSPROC.SNAP_WRITE_FILE 的所有快照表函数

  SYSPROC.SNAP_WRITE_FILE 生成快照并将其内容保存到文件中。如果通过空输入参数调用任何快照表函数，那么返回文件内容，而不是实时系统快照。

数据库权限

每个数据库权限都允许拥有该权限的授权标识对整个数据库执行某种特定类型的操作。数据库权限与特权不同，后者允许对特定数据库对象（例如表或索引）执行特定操作。

这些是数据库权限。

ACCESSCTRL
允许拥有者授予和撤销所有对象特权和数据库权限（对审计例程的特权除外）以及 ACCESSCTRL、DATAACCESS、DBADM 和 SECADM 权限。

BINDADD
允许拥有者在数据库中创建新包。

CONNECT
允许拥有者连接到数据库。

CREATETAB
允许拥有者在数据库中创建新表。

CREATE_EXTERNAL_ROUTINE
允许拥有者创建过程以供数据库的应用程序和其他用户使用。

CREATE_NOT_FENCED_ROUTINE
允许拥有者创建*未受防护的*用户定义的函数 (UDF) 或过程。将把 CREATE_EXTERNAL_ROUTINE 自动授予任何已被授予 CREATE_NOT_FENCED_ROUTINE 权限的用户。

注意: 数据库管理器不会阻止未受防护的 UDF 或过程访问它的存储器或控制块。因此，具有此权限的用户必须非常仔细地测试他们的 UDF，以使之特别严密，然后再将其注册为未受防护的 UDF。

DATAACCESS
允许拥有者访问存储在数据库表中的数据。

DBADM
允许拥有者充当数据库管理员。特别是，它授予拥有者除 ACCESSCTRL、DATAACCESS 和 SECADM 之外的所有其他数据库权限。

EXPLAIN
允许拥有者说明查询方案，而不要求他们拥有访问这些查询方案所引用的表中数据的特权。

IMPLICIT_SCHEMA
允许任何用户隐式地创建模式（使用 CREATE 语句创建对象，并指定尚不存在的模式名）。SYSIBM 成为隐式创建的模式的所有者，并且授予 PUBLIC 在此模式中创建对象的特权。

LOAD 允许拥有者将数据装入到表中。

QUIESCE_CONNECT
允许拥有者在数据库处于停顿状态时访问该数据库。

SECADM
允许拥有者充当数据库的安全性管理员。

SQLADM

允许拥有者监视和调整 SQL 语句。

WLMADM

允许拥有者充当工作负载管理员。特别是，WLMADM 权限的拥有者可以创建和删除工作负载管理器对象、授予和撤销工作负载管理器特权以及执行工作负载管理器例程。

只有具有 SECADM 权限的授权标识才能授予 ACCESSCTRL、DATAACCESS、DBADM 和 SECADM 权限。所有其他权限都可以由具有 ACCESSCTRL 或 SECADM 权限的授权标识授予。

要从 PUBLIC 除去任何数据库权限，具有 ACCESSCTRL 或 SECADM 权限的授权标识必须显式地撤销该权限。

安全性管理权限 (SECADM)

SECADM 权限是对特定数据库的安全性管理权限。此权限允许您创建和管理与安全性相关的数据库对象以及授予和撤销所有数据库权限和特权。此外，安全性管理员可以执行审计系统例程以及管理还有哪些用户可以执行审计系统例程。

SECADM 权限能够从目录表和目录视图中进行选择，但是无法访问存储在用户表中的数据。

SECADM 权限只能由安全性管理员（拥有 SECADM 权限）授予，并且可授予用户、组或角色。PUBLIC 无法直接或间接获取 SECADM 权限。

数据库必须至少有一个具备 SECADM 权限的 USER 类型的授权标识。无法从 USER 类型的每个授权标识撤销 SECADM 权限。

SECADM 权限使用户能够执行下列操作:

- 创建、改变、注释和删除:
  - 审计策略
  - 安全标号组件
  - 安全策略
  - 可信上下文
- 创建、注释和删除:
  - 角色
  - 安全标号
- 授予和撤销数据库特权和权限
- 执行下列审计例程以执行指定的任务:
  - SYSPROC.AUDIT_ARCHIVE 存储过程和表函数对审计日志进行归档。
  - SYSPROC.AUDIT_LIST_LOGS 表函数允许您查找关注的日志。
  - SYSPROC.AUDIT_DELIM_EXTRACT 存储过程将数据抽取到定界文件中，以便进行分析。

安全性管理员可以授予或撤销对这些例程的 EXECUTE 特权，从而在需要时使安全性管理员能够委派这些任务。只有安全性管理员才能授予对这些例程的 EXECUTE 特权。对于这些例程，不能授予 EXECUTE 特权 WITH GRANT OPTION (SQLSTATE 42501)。

- 使用 AUDIT 语句将审计策略与服务器中的特定数据库或数据库对象关联
- 使用 TRANSFER OWNERSHIP 语句来传输该语句的授权标识未拥有的对象

没有其他权限提供这些功能。

只有安全性管理员才能将 ACCESSCTRL、DATAACCESS、DBADM 和 SECADM 权限授予其他用户、组或角色。

在 V9.7 中，DB2 授权模型已更新为明确地区分系统管理员、数据库管理员和安全性管理员的职责。作为此增强的一部分，由 SECADM 权限提供的功能已扩展。在 V9.7 之前的发行版中，SECADM 权限未提供授予和撤销所有权限和特权的功能。并且，SECADM 权限只能授予用户，而不能授予角色或组。此外，对于审计内置过程和表函数，SECADM 权限未提供将 EXECUTE 特权授予其他用户的功能。

数据库管理权限 (DBADM)

DBADM 权限是对特定数据库的管理权限。数据库管理员拥有创建对象和发出数据库命令所需的特权。此外，具有 DBADM 权限的用户还对系统目录表和视图具有 SELECT 特权，可以执行所有内置 DB2 例程（审计例程除外）。

DBADM 权限只能由安全性管理员（拥有 SECADM 权限）授予或撤销，并且可授予用户、组或角色。PUBLIC 无法直接或间接获取 DBADM 权限。

对数据库拥有 DBADM 权限允许用户对该数据库执行下列操作:

- 创建、改变和删除与安全性不相关的数据库对象
- 读取日志文件
- 创建、激活和删除事件监视器
- 查询表空间的状态
- 更新日志历史记录文件
- 停顿表空间
- 重组表
- 使用 **RUNSTATS** 实用程序收集目录统计信息

SQLADM 权限和 WLMADM 权限包含在 DBADM 权限中。WLMADM 权限还能够授予对工作负载的 USAGE 特权。

利用 DBADM 权限来授予 DATAACCESS 权限

安全性管理员可以指定数据库管理员是否能够访问数据库中的数据。DATAACCESS 权限是允许对特定数据库中的数据进行访问的权限。安全性管理员可以使用 GRANT DBADM ON DATABASE 语句的 WITH DATAACCESS 选项来为数据库管理员提供此功能。如果既未指定 WITH DATAACCCESS 选项也未指定 WITHOUT DATAACCCESS 选项，那么缺省情况下会授予 DATAACCESS 权限。

要授予不带 DATAACCESS 权限的数据库管理员权限，请在 SQL 语句中使用 GRANT DBADM WITHOUT DATAACCESS。

利用 DBADM 权限来授予 ACCESSCTRL 权限

安全性管理员可以指定数据库管理员是否能够在数据库中授予和撤销特权。ACCESSCTRL 权限是允许用户在特定数据库中授予和撤销特权以及非管理权限的权限。安全性管理员可以使用 GRANT DBADM ON DATABASE 语句的 WITH ACCESSCTRL 选项来为数据库管理员提供此功能。如果既未指定 WITH ACCCESSCTRL 选项也未指定 WITHOUT ACCCESSCTRL 选项，那么缺省情况下会授予 ACCESSCTRL 权限。

要授予不带 ACCESSCTRL 权限的数据库管理员权限，请在 SQL 语句中使用 GRANT DBADM WITHOUT ACCESSCTRL。

撤销 DBADM 权限

如果安全性管理员已授予包括 DATAACCESS 或 ACCESSCTRL 权限的 DBADM 权限，那么要撤销这些权限，安全性管理员必须显式撤销 DATAACCESS 或 ACCESSCTRL 权限。例如，当安全性管理员将 DBADM 权限授予用户时:

```
GRANT DBADM ON DATABASE TO user1
```

缺省情况下，还会将 DATAACCESS 和 ACCESSCTRL 权限授予 user1。

稍后，安全性管理员从 user1 撤销 DBADM 权限:

```
REVOKE DBADM ON DATABASE FROM user1
```

现在，user1 不再拥有 DBADM 权限，但是仍然拥有 DATAACCESS 和 ACCESSCTRL 权限。

要撤销这些仍有的权限，安全性管理员需要显式地对它们进行撤销:

```
REVOKE ACCESSCTRL, DATAACCESS ON DATABASE FROM user1
```

DBADM 权限在先前发行版中的差别

在 V9.7 中，DB2 授权模型已更新为明确地区分系统管理员、数据库管理员和安全性管理员的职责。作为此增强的一部分，由 DBADM 权限提供的功能已更改。在 V9.7 之前的发行版中，DBADM 权限自动包括了访问数据以及授予和撤销对数据库的特权的功能。在 V9.7 中，这些功能由新权限 DATAACCESS 和 ACCESSCTRL 提供，如前面所说明的那样。

此外，在 V9.7 之前的发行版中，授予 DBADM 权限时也自动授予了下列权限:

- BINDADD
- CONNECT
- CREATETAB
- CREATE_EXTERNAL_ROUTINE
- CREATE_NOT_FENCED_ROUTINE
- IMPLICIT_SCHEMA
- QUIESCE_CONNECT
- LOAD

在 V9.7 之前，当撤销 DBADM 权限时，并不会撤销这些权限。

在 V9.7 中，这些权限现在包含在 DBADM 权限中。当在 V9.7 中撤销 DBADM 权限时，这些权限会丢失。

但是，如果当升级至 V9.7 之前用户已拥有 DBADM 权限，那么撤销 DBADM 权限后，这些权限不会丢失。仅当用户通过拥有在 V9.7 中所授予的 DBADM 权限获取了这些权限时，在 V9.7 中撤销 DBADM 权限才会导致用户丢失这些权限。

访问控制管理权限 (ACCESSCTRL)

ACCESSCTRL 权限是授予和撤销对特定数据库中对象的特权所需的权限。除目录表和视图之外，ACCESSCTRL 权限没有访问存储在表中的数据的固有特权。

ACCESSCTRL 权限只能由安全性管理员（拥有 SECADM 权限）授予。可以将该权限授予用户、组或角色。PUBLIC 无法直接或间接获取 ACCESSCTRL 权限。ACCESSCTRL 权限使用户能够执行下列操作:

- 授予和撤销下列管理权限:
  - EXPLAIN
  - SQLADM
  - WLMADM
- 授予和撤销下列数据库权限:
  - BINDADD
  - CONNECT
  - CREATETAB
  - CREATE_EXTERNAL_ROUTINE
  - CREATE_NOT_FENCED_ROUTINE
  - IMPLICIT_SCHEMA
  - LOAD
  - QUIESCE_CONNECT
- 授予和撤销对下列对象的所有特权（无论特权由谁授予）:
  - 全局变量
  - 索引
  - 昵称
  - 程序包
  - 例程（审计例程除外）
  - 模式
  - 序列
  - 服务器
  - 表
  - 表空间
  - 视图
  - XSR 对象
- 对系统目录表和视图的 SELECT 特权

此权限包含在安全性管理员 (SECADM) 权限中。

数据访问管理权限 (DATAACCESS)

DATAACCESS 是允许对特定数据库中的数据进行访问的权限。

DATAACCESS 权限只能由安全性管理员（拥有 SECADM 权限）授予。可以将该权限授予用户、组或角色。PUBLIC 无法直接或间接获取 DATAACCESS 权限。

对于所有表、视图、具体化查询表和昵称，它会提供下列权限和特权:

- 对数据库的 LOAD 权限
- SELECT 特权（其中包括对系统目录表和视图的 SELECT 特权）
- INSERT 特权
- UPDATE 特权
- DELETE 特权

此外，DATAACCESS 权限还提供下列特权:

- 对所有程序包的 EXECUTE 特权
- 对所有例程（审计例程除外）的 EXECUTE 特权
- 对所有模块的 EXECUTE 特权
- 对所有全局变量的 READ 特权和对除只读变量外的所有全局变量的 WRITE 特权
- 对所有 XSR 对象的 USAGE 特权
- 对所有序列的 USAGE 特权

SQL 管理权限 (SQLADM)

SQLADM 权限是监视和调整 SQL 语句所需的权限。

SQLADM 权限可由安全性管理员（拥有 SECADM 权限）或具有 ACCESSCTRL 权限的用户授予。可以将 SQLADM 权限授予用户、组、角色或 PUBLIC。SQLADM 权限使用户能够执行下列函数:

- 执行下列 SQL 语句:
  - CREATE EVENT MONITOR
  - DROP EVENT MONITOR
  - EXPLAIN
  - FLUSH EVENT MONITOR
  - FLUSH OPTIMIZATION PROFILE CACHE
  - FLUSH PACKAGE CACHE
  - PREPARE
  - REORG INDEXES/TABLE
  - RUNSTATS
  - SET EVENT MONITOR STATE

  注: 如果 **DB2AUTH** 注册表变量已设置为 SQLADM_NO_RUNSTATS_REORG，那么具有 SQLADM 权限的用户将无法执行重组或 runstats 操作。
- 执行下列工作负载管理器 SQL 语句的某些子句:
  - ALTER SERVICE CLASS 语句的下列子句:
    - COLLECT AGGREGATE ACTIVITY DATA

- COLLECT AGGREGATE REQUEST DATA
- COLLECT REQUEST METRICS

– ALTER THRESHOLD 语句的以下子句
- WHEN EXCEEDED COLLECT ACTIVITY DATA

.

– ALTER WORK ACTION SET 语句的允许您改变工作操作的下列子句:
- ALTER WORK ACTION ... COLLECT ACTIVITY DATA
- ALTER WORK ACTION ... COLLECT AGGREGATE ACTIVITY DATA
- ALTER WORK ACTION ... WHEN EXCEEDED COLLECT ACTIVITY DATA

– ALTER WORKLOAD 语句的下列子句:
- COLLECT ACTIVITY METRICS
- COLLECT AGGREGATE ACTIVITY DATA
- COLLECT LOCK TIMEOUT DATA
- COLLECT LOCK WAIT DATA
- COLLECT UNIT OF WORK DATA

- 对系统目录表和视图的 SELECT 特权
- 对所有内置 DB2 例程的 EXECUTE 特权（审计例程除外）

SQLADM 权限包含在数据库管理员 (DBADM) 权限中。

EXPLAIN 权限包含在 SQLADM 权限中。

工作负载管理权限 (WLMADM)

WLMADM 权限是管理特定数据库的工作负载对象所需的权限。此权限允许您创建、改变、删除和注释工作负载管理器对象以及授予和撤销对其的访问权。

WLMADM 权限可由安全性管理员（拥有 SECADM 权限）或具有 ACCESSCTRL 权限的用户授予。可以将 WLMADM 权限授予用户、组、角色或 PUBLIC。WLMADM 权限使用户能够执行下列操作:

- 创建、改变、注释和删除下列工作负载管理器对象:
  – 直方图模板
  – 服务类
  – 阈值
  – 工作操作集
  – 工作类集
  – 工作负载
- 授予和撤销工作负载特权
- 执行内置工作负载管理例程。

WLMADM 权限包含在数据库管理员权限 DBADM 中。

说明管理权限（EXPLAIN）

EXPLAIN 权限是在没有获得特定数据库数据的访问权的情况说明查询方案所需的权限。此权限包含在数据库管理员权限中，没有访问存储在表中的数据的固有特权。

EXPLAIN 权限可由安全性管理员（拥有 SECADM 权限）或具有 ACCESSCTRL 权限的用户授予。可以将 EXPLAIN 权限授予用户、组、角色或 PUBLIC。该权限使您能够执行下列 SQL 语句:

- EXPLAIN
- PREPARE
- DESCRIBE（对于 SELECT 语句或 XQuery 语句的输出）

EXPLAIN 权限还提供对内置说明例程的 EXECUTE 特权。

EXPLAIN 权限包含在 SQLADM 权限中。

LOAD 权限

在数据库级别具有 LOAD 权限以及对表具有 INSERT 特权的用户可以使用 **LOAD** 命令将数据装入到表中。

注: 具有 DATAACCESS 权限的用户对 LOAD 命令具有完全访问权。

如果先前的装入操作是用来装入插入数据的操作，那么在数据库级别具有 LOAD 权限且对表具有 INSERT 特权的用户可以执行 **LOAD RESTART** 或 **LOAD TERMINATE** 操作。

在数据库级别具有 LOAD 权限同时对表具有 INSERT 和 DELETE 特权的用户可以使用 **LOAD REPLACE** 命令。

如果先前的装入操作是装入替换，那么还必须对该用户授予 DELETE 特权，该用户才能执行 **LOAD RESTART** 或 **LOAD TERMINATE** 操作。

如果将异常表用作装入操作的一部分，那么用户对异常表必须具有 INSERT 特权。

具有此权限的用户可以执行 **QUIESCE TABLESPACES FOR TABLE**、**RUNSTATS** 和 **LIST TABLESPACES** 命令。

隐式模式权限 (IMPLICIT_SCHEMA) 注意事项

当创建新数据库时，除非在 **CREATE DATABASE** 命令中指定了 **RESTRICTIVE** 选项，否则 PUBLIC 会被授予 IMPLICIT_SCHEMA 数据库权限。

具有 IMPLICIT_SCHEMA 权限的用户可通过创建对象并指定不存在的模式名称来创建模式。SYSIBM 成为隐式创建的模式的所有者，并且授予 PUBLIC 在此模式中创建对象的特权。如果数据库具有限制性，那么 PUBLIC 没有对该模式的 CREATEIN 特权。隐式创建该模式的用户具有对该模式的 CREATEIN 特权。

如果数据库需要控制隐式创建模式对象的用户，那么创建该数据库时必须指定 RESTRICTIVE 选项。如果数据库不是限制性的，那么必须撤销 PUBLIC 的 IMPLICIT_SCHEMA 数据库权限。在此场景中，只有三种方法可用来创建模式对象:

- 任何用户都可使用他们自己在 CREATE SCHEMA 语句上的授权名称创建模式。
- 具有 DBADM 权限的任何用户都可显式创建不存在的任何模式，并且可选择将另一用户指定为该模式的所有者。
- 具有 DBADM 权限的任何用户都具有 IMPLICIT_SCHEMA 数据库权限，所以他们可在创建其他数据库对象时以任何名称隐式创建模式。

特权

授权标识特权: SETSESSIONUSER

授权标识特权包括对授权标识执行的操作。目前，只有一个这样的特权：SETSESSIONUSER 特权。

可以将 SETSESSIONUSER 特权授予用户或组并允许拥有者将身份切换为任何授予了该特权的授权标识。身份切换是通过使用 SQL 语句 SET SESSION AUTHORIZATION 实现的。SETSESSIONUSER 特权只能由拥有 SECADM 权限的用户授予。

注: 在将 V8 的数据库升级到 V9.1 或更高版本时，对该数据库拥有显式 DBADM 权限的授权标识将被自动授予对 PUBLIC 的 SETSESSIONUSER 特权。这将防止基于具有 DBADM 权限的授权标识的应用程序无法将会话授权标识设置为任何授权标识。当授权标识具有 SYSADM 权限但尚未被显式地授予 DBADM 权限时，不会发生这种情况。

模式特权

模式特权属于对象特权类别。

对象特权显示在图 3 中。

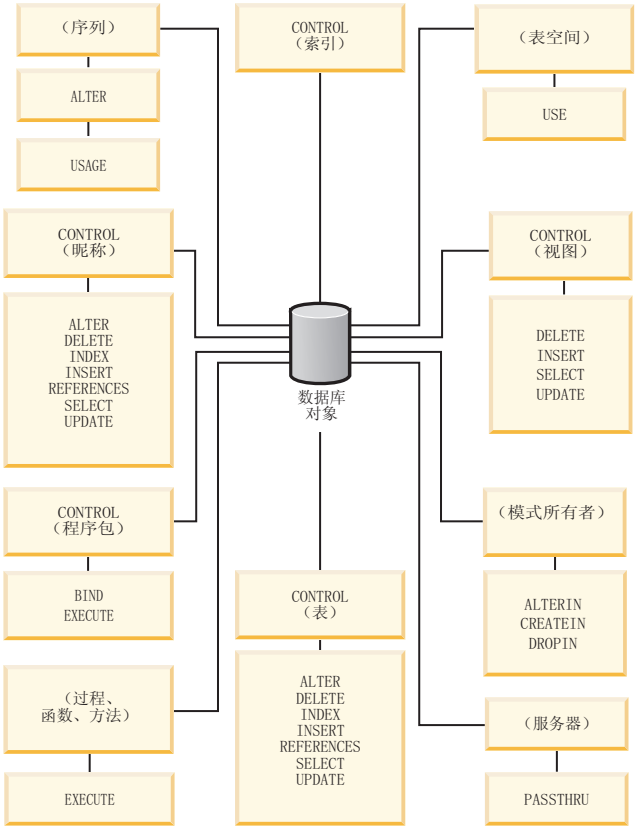


图 3. 对象特权

模式特权涉及到对一个数据库中的模式所执行的操作。可将下列任何特权授予用户、组、角色或 PUBLIC:

- CREATEIN 允许用户在模式中创建对象。

- ALTERIN 允许用户在模式中改变对象。
- DROPIN 允许用户在模式中删除对象。

模式所有者具有所有这些特权，并且有将这些特权授予其他用户的功能。在模式对象中操纵的对象包括：表、视图、索引、程序包、数据类型、函数、触发器、过程和别名。

表空间特权

表空间特权涉及对数据库中表空间的操作。可将对表空间的 USE 特权授予用户，这允许用户在该表空间中创建表。

创建表空间时，其所有者被授予对该表空间的 USE 特权（带 WITH GRANT OPTION）。此外，拥有 SECADM 或 ACCESSCTRL 权限的用户能够授予对表空间的 USE 特权。

拥有 SYSADM 或 SYSCTRL 权限的用户能够使用任何表空间。

在缺省情况下，创建数据库时，会将对表空间 USERSPACE1 的 USE 特权授予 PUBLIC（虽然可以撤销此特权）。

USE 特权不能配合 SYSCATSPACE 或任何系统临时表空间使用。

表和视图特权

表和视图特权涉及到对一个数据库中的表或视图所执行的操作。

用户必须对数据库具有 CONNECT 权限，才可使用下列任何一个特权:

- CONTROL 给用户提供对表或视图的所有特权，包括删除它以及授予和撤销各个表特权的功能。必须具有 ACCESSCTRL 或 SECADM 权限，才能授予 CONTROL。一个表的创建者自动接收表的 CONTROL 特权。仅当视图的创建者对视图定义中引用的所有表、视图和昵称都具有 CONTROL 特权时，该创建者才自动接收 CONTROL 特权。
- ALTER 允许用户修改表，例如，为表添加列或唯一约束。具有 ALTER 特权用户还可以 COMMENT ON 一个表，或者表的一列。有关可能对表执行的修改的信息，请参阅 ALTER TABLE 和 COMMENT 语句。
- DELETE 允许用户从表或视图中删除行。
- INDEX 允许用户对表创建一个索引。索引创建者自动具有索引的 CONTROL 特权。
- INSERT 允许用户将行插入表或视图以及运行 **IMPORT** 实用程序。
- REFERENCES 允许用户创建和删除一个外键，并指定该表为关系中的父表。用户可能只对特定的列拥有此特权。
- SELECT 允许用户检索表或视图中的行、对表创建视图以及运行 **EXPORT** 实用程序。
- UPDATE 允许用户更改表或视图中的条目，或表或视图中的一个或多个特定列的条目。用户只能对特定的列拥有此特权。

将这些特权授予其他用户的特权也可在 GRANT 语句上使用 WITH GRANT OPTION 来授予。

注: 当授予一个用户或组对某个表的 CONTROL 特权时，将使用 WITH GRANT OPTION 自动授予对该表的所有其他特权。如果接着从某个用户撤销了对该表的 CON-

TROL 特权，该用户将仍然保留自动授予的其他特权。要撤销使用 CONTROL 特权授予的所有特权，必须显式撤销个别特权，或者在 REVOKE 语句上指定 ALL 关键字，例如:

```
REVOKE ALL
       ON EMPLOYEE FROM USER HERON
```

当使用类型表时，表和视图的特权有特别的意义。

注: 可在一个表层次结构的每一层单独授予特权。因此，对于类型表的层次结构中的一个超表，被授予对该超表的特权的用户也可间接影响任何子表。但是，如果对一个子表持有需要的特权，那么用户只能直接对该子表操作。

在一个表层次结构中，表之间的“超表/子表”关系表示像 SELECT、UPDATE 和 DELETE 这样的操作将影响该操作的目标表和其所有子表（若有）中的行。这种行为称为*可替换性*。例如，假设创建了一个类型为 Employee_t 的 Employee 表，它具有类型为 Manager_t 的子表 Manager。经理是一种特殊的职员，这由结构化类型 Employee_t 与 Manager_t 之间的“类型/子类型”关系，和对应的在表 Employee 与 Manager 之间的“表/子表”关系来指示。由于这种关系，SQL 查询:

```
SELECT * FROM Employee
```

将返回职员和经理的对象标识和 Employee_t 属性。类似地，更新操作:

```
UPDATE Employee SET Salary = Salary + 1000
```

将给经理和正式职员加薪一千元。

对 Employee 具有 SELECT 特权的用户可以执行这个 SELECT 操作，即使他们对 Manager 没有显式 SELECT 特权。但是，将不允许这类用户直接对 Manager 子表执行 SELECT 操作，因此，这类用户将不能访问 Manager 表的任何非继承列。

类似地，对 Employee 具有 UPDATE 特权的用户将能够对 Manager 执行 UPDATE 操作，从而影响正规的职员和经理，即使该用户对 Manager 表不具有显式的 UPDATE 特权。但是，将不允许这类用户直接对 Manager 子表执行 UPDATE 操作，因而，这类用户将不能更新 Manager 表的任何非继承列。

程序包特权

程序包是一个数据库对象，它包含数据库管理器以适合于特定应用程序的最有效方式访问数据所需的信息。程序包特权使用户能够创建和操纵程序包。

用户必须对数据库具有 CONNECT 权限，才可使用下列任何特权:

- CONTROL 给用户提供重新绑定、删除或执行程序包的功能，以及将那些特权授予其他用户的功能。程序包的创建者自动接收此特权。具有 CONTROL 特权的用户被授予 BIND 和 EXECUTE 特权，还可以使用 GRANT 语句将这些特权授予其他用户。（如果使用 WITH GRANT OPTION 授予特权，那么接收 BIND 或 EXECUTE 特权的用户可以依次将此特权授予其他用户。）要授予 CONTROL 特权，用户必须具有 ACCESSCTRL 或 SECADM 权限。
- 对程序包的 BIND 特权允许用户重新绑定或绑定该程序包以及添加具有相同程序包名和创建者的新程序包版本。
- EXECUTE 允许用户执行或运行程序包。

注: 所有程序包特权适用于共享相同程序包名和创建者的所有 VERSION。

除这些程序包特权外，BINDADD 数据库权限还允许用户创建新程序包或重新绑定数据库中的现有程序包。

按昵称引用的对象需要对包含该对象的数据源遍历认证检查。另外，程序包用户必须对数据源中的数据源对象拥有适当特权或权限级别。

包含昵称的程序包可能需要其他授权步骤，因为与 DB2 系列数据源通信时，DB2 数据库使用动态查询。在数据源运行程序包的授权标识必须有恰当的权限，才可在该数据源动态执行此程序包。

索引特权

索引或索引规范的创建者自动接收该索引的 CONTROL 特权。索引的 CONTROL 特权实际是删除此索引的功能。要授予对索引的 CONTROL 特权，用户必须具有 ACCESSCTRL 或 SECADM 权限。

表级别 INDEX 特权允许用户对该表创建索引。

昵称级 INDEX 特权允许用户对该昵称创建索引规范。

序列特权

序列的创建者自动接收对序列的 USAGE 和 ALTER 特权。要使用序列的 NEXT VALUE 和 PREVIOUS VALUE 表达式，需要具有 USAGE 特权。

要允许其他用户使用 NEXT VALUE 和 PREVIOUS VALUE 表达式，必须将序列特权授予 PUBLIC。这就允许所有用户使用具有指定序列的表达式。

序列上的 ALTER 特权允许用户执行诸如重新启动序列或更改将来序列值增量之类的任务。序列的创建者可以授予其他用户 ALTER 特权，并且如果使用 WITH GRANT OPTION，那么这些用户可以依次将这些特权授予其他用户。

例程特权

EXECUTE 特权涉及对所有类型的例程（如数据库中的函数、过程和方法）执行的操作。一旦具有 EXECUTE 特权，用户就可以调用例程、创建源于该例程序（仅应用于函数）的函数以及在任何 DDL 语句（如 CREATE VIEW 和 CREATE TRIGGER）中引用例程。

定义外部存储过程、函数或方法的用户接收 EXECUTE WITH GRANT 特权。如果通过 WITH GRANT OPTION 将 EXECUTE 特权授予另一个用户，那么该用户可以依次将 EXECUTE 特权授予其他用户。

对工作负载的 USAGE 特权

要能够使用某一工作负载，拥有 ACCESSCTRL、SECADM 或 WLMADM 权限的用户可以使用 GRANT USAGE ON WORKLOAD 语句来为用户、组或角色授予对该工作负载的 USAGE 特权。

当 DB2 数据库系统发现相匹配的工作负载时，它就会检查会话用户是否对该工作负载具有 USAGE 特权。如果会话用户对该工作负载没有 USAGE 特权，那么 DB2 数据库系统将在有序列表中搜索下一个相匹配的工作负载。换句话说，会话用户对其不具备 USAGE 特权的工作负载将被当作不存在一样来对待。

USAGE 特权信息存储在目录中，可以通过 SYSCAT.WORKLOADAUTH 视图来查看此特权信息。

可以使用 REVOKE USAGE ON WORKLOAD 语句来撤销 USAGE 特权。

具有 ACCESSCTRL、DATAACCESS、DBADM、SECADM 或 WLMADM 权限的用户隐式地具有对所有工作负载的 USAGE 特权。

SYSDEFAULTUSERWORKLOAD 工作负载和 USAGE 特权

如果在未使用 RESTRICT 选项的情况下创建数据库，那么创建数据库时就会将对于 SYSDEFAULTUSERWORKLOAD 的 USAGE 特权授予 PUBLIC。否则，必须由具有 ACCESSCTRL、WLMADM 或 SECADM 权限的用户显式授予 USAGE 特权。

如果会话用户对任何工作负载（包括 SYSDEFAULTUSERWORKLOAD）都没有 USAGE 特权，那么就会返回 SQL 错误。

SYSDEFAULTADMWORKLOAD 工作负载和 USAGE 特权

不能将对 SYSDEFAULTADMWORKLOAD 的 USAGE 特权显示授予任何用户。只允许发出了 **SET WORKLOAD TO SYSDEFAULTADMWORKLOAD** 命令并且其会话授权标识具有 ACCESSCTRL、DATAACCESS、DBADM、WLMADM 或 SECADM 权限的用户使用此工作负载。

GRANT USAGE ON WORKLOAD 和 REVOKE USAGE ON WORKLOAD 语句对 SYSDEFAULTADMWORKLOAD 没有任何影响。

不同上下文中的授权标识

使用授权标识有两个目的：标识和授权检查。例如，会话授权标识用于初始授权检查。

在特定上下文中使用授权标识时，将限定对该授权标识的引用以标识上下文，如下一节所示。

对授权标识的上下文引用

定义

系统授权标识

用于执行任何初始授权检查的授权标识，例如，在连接处理期间检查 CONNECT 特权。在连接处理期间的认证过程中，将产生符合 DB2 命名要求的授权标识，该标识表示 DB2 数据库系统内的外部用户标识。系统授权标识表示创建连接的用户。使用 SYSTEM_USER 专用寄存器来查看系统授权标识的当前值。不能更改连接的系统授权标识。

会话授权标识

用于任何会话授权检查的授权标识，会话授权检查在连接处理期间执行完初始检查后执行。会话授权标识的缺省值是系统授权标识的值。使用 SESSION_USER 专用寄存器来查看会话授权标识的当前值。USER 专用寄存器是 SESSION_USER 专用寄存器的同义词。可以使用 SET SESSION AUTHORIZATION 语句更改会话授权标识。

程序包授权标识

用于将程序包绑定至数据库的授权标识。从 **BIND** 命令的 **OWNER** *authorization id* 选项的值中获取此授权标识。程序包授权标识有时称为程序包绑定程序或程序包所有者。

例程所有者授权标识

列示在系统目录中作为已调用 SQL 例程所有者的授权标识。

例程调用程序授权标识

调用 SQL 例程的语句的语句授权标识。

语句授权标识

与特定 SQL 语句关联的授权标识，该语句用于任何权限要求并在适当时用于确定对象所有权。它根据 SQL 语句类型从相应的源授权标识中获取其值:

- 静态 SQL

  使用程序包授权标识。

- 动态 SQL（在非例程上下文中）

  下表显示了每种情况下使用的授权标识:

用于发出程序包的 **DYNAMICRULES** 选项的值	使用的授权标识
RUN	会话授权标识
BIND	程序包授权标识
DEFINERUN 和 INVOKERUN	会话授权标识
DEFINEBIND 和 INVOKEBIND	程序包授权标识

- 动态 SQL（在例程上下文中）

  下表显示了每种情况下使用的授权标识:

用于发出程序包的 **DYNAMICRULES** 选项的值	使用的授权标识
DEFINERUN 和 DEFINEBIND	例程所有者授权标识
INVOKERUN 和 INVOKEBIND	例程调用程序授权标识

使用 CURRENT_USER 专用寄存器来查看语句授权标识的当前值。不能直接更改语句授权标识；DB2 数据库系统将自动更改该标识以反映每个 SQL 语句的性质。

创建数据库时授予的缺省特权

创建数据库时，会在该数据库内授予您缺省数据库级别权限和缺省对象级别特权。

按照将权限和特权记录到其中的系统目录视图，列示了授予您的权限和特权:

1. SYSCAT.DBAUTH
   - 数据库创建者被授予下列权限:
     - ACCESSCTRL
     - DATAACCESS
     - DBADM
     - SECADM

- 在非限定数据库中，特殊组 PUBLIC 被授予下列权限:
  - CREATETAB
  - BINDADD
  - CONNECT
  - IMPLICIT_SCHEMA

2. SYSCAT.TABAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予下列特权:
   - 对所有 SYSCAT 和 SYSIBM 表的 SELECT 特权
   - 对所有 SYSSTAT 表的 SELECT 和 UPDATE 特权
   - 对模式 SYSIBMADM 中下列视图的 SELECT 特权:
     - ALL_*
     - USER_*
     - ROLE_*
     - SESSION_*
     - DICTIONARY
     - TAB

3. SYSCAT.ROUTINEAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予下列特权:
   - 对模式 SQLJ 中的所有过程的 EXECUTE with GRANT 特权
   - 对模式 SYSFUN 中所有函数和过程的 EXECUTE with GRANT 特权
   - 对模式 SYSPROC 中所有函数和过程（审计例程除外）的 EXECUTE with GRANT 特权
   - 对模式 SYSIBM 中所有表函数的 EXECUTE 特权
   - 对模式 SYSIBM 中所有其他过程的 EXECUTE 特权

4. SYSCAT.MODULEAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予下列特权:
   - 对模式 SYSIBMADM 中下列模块的 EXECUTE 特权:
     - DBMS_DDL
     - DBMS_JOB
     - DBMS_LOB
     - DBMS_OUTPUT
     - DBMS_SQL
     - DBMS_STANDARD
     - DBMS_UTILITY

5. SYSCAT.PACKAGEAUTH
   - 数据库创建者被授予下列特权:
     - 对 NULLID 模式中创建的所有包的 CONTROL 特权
     - 对 NULLID 模式中创建的所有包的 BIND with GRANT 特权
     - 对 NULLID 模式中创建的所有包的 EXECUTE with GRANT 特权

- 在非限定数据库中，特殊组 PUBLIC 被授予下列特权:
  - 对 NULLID 模式中创建的所有包的 BIND 特权
  - 对 NULLID 模式中创建的所有包的 EXECUTE 特权

6. SYSCAT.SCHEMAAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予下列特权:
   - 对模式 SQLJ 的 CREATEIN 特权
   - 对模式 NULLID 的 CREATEIN 特权

7. SYSCAT.TBSPACEAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予对表空间 USERSPACE1 的 USE 特权。

8. SYSCAT.WORKLOADAUTH

   在非限定数据库中，特殊组 PUBLIC 被授予对 SYSDEFAULTUSERWORKLOAD 的 USAGE 特权。

9. SYSCAT.VARIABLEAUTH

   在非限定数据库中，对特殊组 PUBLIC 授予了对于 SYSIBM 模式中的模式全局变量（下列变量除外）的 READ 特权:
   - SYSIBM.CLIENT_ORIGUSERID
   - SYSIBM.CLIENT_USRSECTOKEN

非限定数据库是使用不带 RESTRICTIVE 选项的 CREATE DATABASE 命令创建的数据库。

授予和撤销访问权

授予特权

要授予对大多数数据库对象的特权，必须对该对象具有 ACCESSCTRL 权限、SECADM 权限或 CONTROL 特权；或者，必须拥有特权 WITH GRANT OPTION。此外，具有 SYSADM 或 SYSCTRL 权限的用户可以授予表空间特权。只能授予对现有对象的特权。

关于此任务

要将 CONTROL 特权授予其他用户，必须具有 ACCESSCTRL 或 SECADM 权限。要授予 ACCESSCTRL、DATAACCESS、DBADM 或 SECADM 权限，必须具有 SECADM 权限。

GRANT 语句允许授权用户授予特权。可以在一条语句中将一个特权授予一个或多个授权名；或授予 PUBLIC，这使该特权可供所有用户使用。注意授权名可以是个别用户，也可以是组。

在存在具有相同名称的用户和组的操作系统上，应当指定是将该特权授予用户还是授予组。GRANT 和 REVOKE 语句都支持关键字 USER、GROUP 和 ROLE。如果未使用这些可选的关键字，那么数据库管理器会检查操作系统安全性工具，以确定授权名是标识用户还是组；它还检查是否存在同名且类型为角色的授权标识。如果数据库管理器无法确定授权名是否引用用户、组或角色，那么会返回错误。以下示例将 EMPLOYEE 表的 SELECT 特权授予用户 HERON:

```
GRANT SELECT
       ON EMPLOYEE TO USER HERON
```

以下示例将 EMPLOYEE 表的 SELECT 特权授予组 HERON:

```
GRANT SELECT
        ON EMPLOYEE TO GROUP HERON
```

撤销特权

REVOKE 语句允许授权用户撤销先前已授予其他用户的特权。

关于此任务

要撤销对数据库对象的特权，必须对该对象具有 ACCESSCTRL 权限、SECADM 权限或 CONTROL 特权。表空间特权还可以由具有 SYSADM 和 SYSCTRL 权限的用户撤销。注意，持有使用 WITH GRANT OPTION 授予的特权并不足以撤销该特权。要撤销另一个用户的 CONTROL 特权，必须具有 ACCESSCTRL 或 SECADM 权限。要撤销 ACCESSCTRL、DATAACCESS、DBADM 或 SECADM 权限，必须具有 SECADM 权限。表空间特权只能由拥有 SYSADM 或 SYSCTRL 权限的用户撤销。只能撤销对现有对象的特权。

注: 不具有 ACCESSCTRL 权限、SECADM 权限或 CONTROL 特权的用户不能撤销他们使用 WITH GRANT OPTION 授予的特权。另外，由被撤销特权的人授予特权的那些人不会被撤销特权。
如果撤销用户（具有 DBADM 权限）的显式授予的表（或视图）特权，那么将不会从在该表上定义的其他视图撤销特权。这是因为视图特权可通过 DBADM 权限得到，并不依赖于基础表上的显式特权。

如果已将特权授予名称相同的用户、组或角色，那么当撤销此特权时必须指定 GROUP、USER 或 ROLE 关键字。以下示例从用户 HERON 撤销对 EMPLOYEE 表的 SELECT 特权:

```
REVOKE SELECT
        ON EMPLOYEE FROM USER HERON
```

以下示例从组 HERON 撤销对 EMPLOYEE 表的 SELECT 特权:

```
REVOKE SELECT
        ON EMPLOYEE FROM GROUP HERON
```

注意从一个组中撤销特权并不能从该组的所有成员中撤销该特权。如果个别名称已被直接授予一个特权，那么此名称将保留它，直到被直接撤销该特权为止。

如果从一个用户撤销了表特权，那么也撤销对该用户创建的任何视图的特权，这取决于已撤销的表特权。但是，只撤销系统隐式授予的那些特权。如果该视图的一个特权是另一个用户直接授予的，那么此特权仍然会被保留。

如果从一个用户撤销了表特权，那么也撤销对该用户创建的任何视图的特权，这取决于已撤销的表特权。但是，只撤销系统隐式授予的那些特权。如果该视图的一个特权是另一个用户直接授予的，那么此特权仍然会被保留。

您可能会遇到这样的情况，您想要将一种特权授予一个组，然后仅从组中的其中一个成员撤销该特权。仅有两种方式执行该操作而不会接收到错误消息 SQL0556N:

- 您可以从组中除去该成员；或者创建具有较少成员的新组，并将特权授予新组。

- 可以从组中撤销特权，然后将特权授予个别用户（授权标识）。

注: 当从一个用户撤销对一个表或视图的 CONTROL 特权时，此用户仍然能够将特权授予其他用户。当授予 CONTROL 特权时，用户也可接收使用 WITH GRANT OPTION 提供的所有其他特权。一旦撤销了 CONTROL，那么使用 WITH GRANT OPTION 提供的所有其他特权会保留，一直到显式撤销它们为止。

取决于被撤销的特权的所有程序包都标记为无效，但是如果一个具有合适权限的用户重新绑定它们，那么可以变得有效。如果随后又将特权授予应用程序的绑定者，也可以重建程序包；运行此应用程序将触发一个成功的隐式重新绑定。如果从 PUBLIC 撤销了特权，那么所有由只能根据 PUBLIC 特权绑定的用户绑定的程序包都无效。如果从用户撤销了 DBADM 权限，那么该用户绑定的所有程序包全都无效，包括与数据库实用程序相关的那些程序包。试图使用标记为无效的程序包将引起系统试图重新绑定此程序包。如果此重新绑定尝试失败，那么发生错误（SQLCODE -727）。在此情况下，必须由具有如下权限的用户显式地重新绑定程序包:

- 重新绑定程序包的权限
- 对程序包中使用的对象的适当权限

应当在撤销特权时重新绑定这些程序包。

如果根据一个或多个特权定义触发器或 SQL 函数，并且失去了这些特权中的一个或多个特权，那么不能使用该触发器或 SQL 函数。

通过创建和删除对象来管理隐式权限

数据库管理器将某些特权隐式地授予创建数据库对象（如表或程序包）的用户。当具有 DBADM 权限的用户创建对象时，也会授予特权。类似地，当删除一个对象时，就除去了特权。

关于此任务

当创建的对象是表、昵称、索引或程序包时，用户会接收到对该对象的 CONTROL 特权。当对象是视图时，只有在用户对该视图定义中引用的所有表、视图和昵称具有 CONTROL 特权时，才隐式授予此视图 CONTROL 特权。

当显式创建的对象是一个模式时，将使用 WITH GRANT OPTION 授予此模式所有者 ALTERIN、CREATEIN 和 DROPIN 特权。隐式创建的模式具有授予 PUBLIC 的 CREATEIN。

建立程序包的所有权

BIND 和 **PRECOMPILE** 命令创建或更改应用程序包。在其中任何一个命令中，使用 **OWNER** 选项来命名所生成程序包的所有者。

关于此任务

命名程序包的所有者有简单规则:

- 任何用户可命名自己为所有者。如果未指定 **OWNER** 选项，那么这是缺省值。
- 具有 DBADM 权限的用户标识可使用 **OWNER** 选项将任何授权标识命名为所有者。

并非所有可使用 DB2 数据库产品绑定程序包的操作系统都支持 **OWNER** 选项。

程序包中的隐式特权

对一个数据库中数据的访问可以由应用程序请求，也可由参与交互式工作站会话的人请求。程序包包含允许用户对许多数据库对象执行不同操作的语句。其中每个操作需要一个或多个特权。

授予绑定程序包的个人、PUBLIC 和角色（这些角色已授予个人和 PUBLIC）的特权用于在绑定静态 SQL 和 XQuery 语句时检查权限。通过组授予的特权以及授予组的角色不用于在绑定静态 SQL 和 XQuery 语句时检查权限。

除非绑定程序包时指定了 VALIDATE RUN，否则具有有效授权标识、绑定程序包的用户必须满足以下任一条件:

- 已被授予执行程序包中静态 SQL 或 XQuery 语句所需的所有特权。
- 已通过下列一项或多项的成员资格获取必需特权:
  - PUBLIC
  - 授予 PUBLIC 的角色
  - 授予用户的角色

如果执行 BIND 时指定了 VALIDATE RUN，那么并非此程序包中任何静态 SQL 或 XQuery 语句的所有授权失败都将导致 BIND 失败，但会在运行时会重新验证这些 SQL 或 XQuery 语句。当进行检查以确保用户具有相应的权限（BIND 或 BINDADD 特权）来绑定程序包时，PUBLIC、组、角色和用户特权全部都会用到。

程序包可包含静态 SQL 和 XQuery 语句以及动态 SQL 和 XQuery 语句。要处理包含静态查询的程序包，用户只需要对该程序包具有 EXECUTE 特权。然后，对于该程序包中的任何静态查询，此用户可以隐式获得程序包绑定者的特权，但只在此程序包所施加的限制内。

如果程序包中有动态 SQL 或 XQuery 语句，那么在预编译或绑定程序包时，所需特权取决于为 **DYNAMICRULES** 指定的值。有关更多信息，请参阅描述有关动态查询的 **DYNAMICRULES** 效果的主题。

对包含昵称的程序包的间接特权

当程序包包含对昵称的引用时，对程序包创建者和程序包用户的授权处理比较复杂。

当程序包创建者成功绑定包含昵称的程序包时，程序包创建者不必通过在数据源处对昵称引用的表和视图所作的认证检查或特权检查。但是，此程序包的执行者必须通过数据源的认证和权限检查。

例如，假定程序包创建者的 .SQC 文件包含几条 SQL 或 XQuery 语句。一条静态语句引用了本地表。另一条动态语句引用了昵称。当绑定此程序包时，使用程序包创建者的授权标识来验证对本地表和昵称的特权，但不对昵称标识的数据源对象执行检查。当另一个用户执行此程序包时，假设他对该程序包具有 EXECUTE 特权，那么该用户不必通过对引用此表的语句所做的任何附加的特权检查。但是，对于引用昵称的语句，执行此程序包的用户必须通过数据源的认证检查和特权检查。

当 .SQC 文件仅包含动态 SQL 和 XQuery 语句以及表与昵称的混合引用时，对本地对象和昵称的 DB2 数据库授权检查类似。程序包用户必须通过在语句内设置的任何本地

对象（表和视图）的特权检查，还要通过昵称对象的特权检查（程序包用户必须通过在包含昵称标识的对象的数据源处的认证检查和特权检查）。在这两种情况下，程序包的用户都必须具有 EXECUTE 特权。

程序包执行者的授权标识和密码用于所有数据源认证和特权处理。可通过创建用户映射更改此信息。

注: 不能在静态 SQL 和 XQuery 语句中指定昵称。请不要对包含昵称的程序包使用 **DYNAMICRULES** 选项（设置为 BIND）。

包含昵称的程序包可能需要其他授权步骤，因为与 DB2 系列数据源通信时，DB2 数据库使用动态 SQL。在数据源运行程序包的授权标识必须有恰当的权限，才可在该数据源动态执行此程序包。

使用视图控制对数据的访问

视图提供了一种方法来控制对表的访问或扩展对表的特权。

使用视图时可以对表的访问进行下列控制:

- 只访问表的指定列。

  对于要求只访问一个表的特定列的用户和应用程序，授权用户可以创建一个视图来限制这些列只被需要它们的那些人访问。
- 只访问表的所有行的一个子集。

  通过在一个视图定义的子查询中指定 WHERE 子句，授权用户可以限制通过一个视图访问的行。
- 只访问数据源表或视图中的行或列的一个子集。如果通过昵称访问数据源，那么可以创建引用昵称的本地 DB2 数据库视图。这些视图可从一个或多个数据源引用昵称。

  注: 因为可以创建一个视图来包含对多个数据源的昵称引用，因此用户可从一个视图访问多个数据源中的数据。这些视图称为*多位置视图*。当将一个分布式环境中各敏感表的列信息连接在一起时，或当个别用户缺少在数据源需要的特定对象的特权时，这类视图可以发挥作用。

要创建视图，用户必须对视图定义中引用的每个表、视图或昵称都具有 DATAACCESS 权限或者 CONTROL 或 SELECT 特权。用户还必须能够在为此视图指定的模式中创建对象。即，用户必须对现有模式具有 DBADM 权限或 CREATEIN 特权，或者，当此模式尚未存在时，用户必须对数据库具有 IMPLICIT_SCHEMA 权限。

如果要创建引用昵称的视图，不需要对视图中昵称引用的数据源对象（表和视图）有其他权限；但是，当视图的用户访问该视图时，必须对基础数据源对象具有 SELECT 权限或等价的权限级别。

如果用户在数据源处对基础对象（表和视图）没有合适的权限，可执行下列操作:

1. 以用户可访问的数据源表中的那些列为基础，创建一个数据源视图
2. 授予用户对此视图的 SELECT 特权
3. 创建一个引用此视图的昵称

然后用户可发出引用新昵称的 SELECT 语句来访问那些列。

以下方案提供了如何使用视图来限制访问信息的一个更详细的示例。

因种种原因，许多人可能需要访问 STAFF 表中的信息。例如:

- 人事部门需要能更新和查看整个表。

  通过授予组 PERSONNL 对 STAFF 表的 SELECT 和 UPDATE 特权，可以很容易地满足此要求:

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- 个别部门的经理需要查看他们职员的工资信息。

  可以通过为每个部门经理创建一个视图来满足此要求。例如，可以为部门号为 51 的经理创建如下视图:

  ```
  CREATE VIEW EMP051 AS
         SELECT NAME,SALARY,JOB FROM STAFF
         WHERE DEPT=51
    GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  具有授权名 JANE 的经理将像查询 STAFF 表一样查询 EMP051 视图。当访问 STAFF 表的 EMP051 视图时，此经理会看到如下信息:

名称	工资	职位
Fraye	45150.0	Mgr
Williams	37156.5	Sales
Smith	35654.5	Sales
Lundquist	26369.8	Clerk
Wheeler	22460.0	Clerk

- 所有用户都需要能够找到其他职员。可以根据 STAFF 表的 NAME 列和 ORG 表的 LOCATION 列创建一个视图，并通过两个表相应的 DEPT 和 DEPTNUMB 列将这两个表连接，来满足此要求:

  ```
  CREATE VIEW EMPLOCS AS
          SELECT NAME, LOCATION FROM STAFF, ORG
          WHERE STAFF.DEPT=ORG.DEPTNUMB
    GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
  ```

  访问职员位置视图的用户将看到如下信息:

名称	所在地
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington

名称	所在地
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

控制数据库管理员（DBA）进行的访问

可能要监视、控制或防止数据库管理员（拥有 DBADM 权限的用户）对数据进行的访问。

监视对数据的访问

可以使用 DB2 审计设施来监视数据库管理员进行的访问。为此，请遵循下列步骤:

1. 创建审计策略，用来监视要为拥有 DBADM 权限的用户捕获的事件。
2. 使此审计策略与 DBADM 权限相关联。

控制对数据的访问

可将可信上下文与角色配合使用来控制数据库管理员进行的访问。为此，请遵循下列步骤:

1. 创建一个角色，并对该角色授予 DBADM 权限。
2. 定义一个可信上下文，并使该角色成为此可信上下文的缺省角色。

请不要对任何授权标识显式授予该角色中的成员资格。这样，该角色只有通过此可信上下文才可用，并且用户只有位于该可信上下文范围内时才能获得 DBADM 功能。

3. 可以使用两种方法来控制用户如何访问可信上下文:
   - 隐式访问: 为每个用户创建唯一的可信上下文。当用户建立与可信上下文的属性相匹配的常规连接时，它们是隐式可信的，并且获得对角色的访问权。
   - 显式访问: 使用 WITH USE FOR 子句创建一个可信上下文，以定义可以访问此可信上下文的所有用户。创建一个应用程序，这些用户可以通过此应用程序来发出数据库请求。该应用程序建立显式可信连接，当用户发出请求时，该应用程序就切换至该用户标识，并代表该用户对数据库执行请求。

如果要监视此可信上下文的使用，那么可以创建审计策略，用来为此可信上下文的用户捕获您关注的事件。使此审计策略与可信上下文相关联。

防止对数据的访问

要防止对表中数据的访问，请选择下列其中一个选项:

- 要防止对所有表中数据的访问，从 DBADM 用户、角色或组撤销 DATAACCESS。或者，可在不使用 DATAACCESS 选项的情况下对关注的用户、角色或组授予 DBADM
- 要防止对一个特定表中数据的访问，请遵循下列步骤:
  - 将安全标号指定给该表中的每列。
  - 将该安全标号授予角色。
  - 对具有访问该表的合法需要的所有用户（或角色）授予该角色。

  除非用户是该角色的成员，否则无论用户的权限如何，用户都将不能够访问该表中的数据。

通过间接方法来访问数据

要成功地管理安全性，需要了解用户可以通过哪些间接方法来访问数据。

以下列表表示用户可以用来获取他们可能无权访问的数据的访问权的间接方法:

- **目录视图**: DB2 数据库系统目录视图存储数据库对象的元数据和统计信息。对目录视图拥有 SELECT 访问权的用户可以在一定程度上了解他们无法访问的数据。为了提高安全性，应该确保只有有资格的用户才能访问目录视图。

  注: 在 DB2 通用数据库 V8 或更早版本中，缺省情况下会将目录视图的 SELECT 访问权授予 PUBLIC。在 DB2 V9.1 或更高版本的数据库系统中，通过使用 **CREATE DATABASE** 命令的新选项 **RESTRICTIVE**，用户可以选择是否将目录视图的 SELECT 访问权授予 PUBLIC。
- **说明快照**: 说明快照是说明 SQL 或 XQuery 语句时收集的压缩信息。此信息作为二进制大对象（BLOB）存储在 EXPLAIN_STATEMENT 表中，它包含列统计信息，而列统计信息可能会透露关于表数据的信息。为了提高安全性，只应该将说明表的访问权授予有资格的用户。
- **节说明**: 节说明过程（EXPLAIN_FROM_SECTION、EXPLAIN_FROM_CATALOG、EXPLAIN_FROM_ACTIVITY 和 EXPLAIN_FROM_DATA）可使用位于程序包高速

缓存的任何节中的信息来填充说明表。此信息包括可能包含输入数据值的语句文本。为了提高安全性，只应该将节说明过程和说明表的访问权授予有资格的用户。

- **日志阅读器函数**：如果用户有权运行读取日志的函数，并且能够理解日志记录的格式，他们就能访问可能无权访问的数据。下列函数读取日志：

函数	执行该函数所需的权限
db2ReadLog	SYSADM 或 DBADM
db2ReadLogNoConn	无。

- **复制**：复制数据时，即使受保护数据也会在目标位置再现。为了提高安全性，应该确保目标位置至少与源位置同样安全。
- **异常表**：如果在将数据装入表中时指定了异常表，有权访问异常表的用户就会获得他们可能无权访问的信息。为了提高安全性，只应该将异常表的访问权授予授权用户，并且，使用异常表完毕后应立即将其删除。
- **备份表空间或数据库**：有权运行 **BACKUP DATABASE** 命令的用户能够创建数据库或表空间的备份（包含任何受保护数据）并将该数据复原到别处。备份可能包含用户无法以其他方式访问的数据。

  拥有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户可以执行 **BACKUP DATABASE** 命令。
- **设置会话权限**：在 DB2 通用数据库 V8 或更早版本中，具有 DBADM 权限的用户可以使用 SET SESSION AUTHORIZATION SQL 语句来设置任何数据库用户的会话授权标识。在 DB2 V9.1 或更高版本的数据库系统中，必须通过 GRANT SETSESSIONUSER 语句显式地对用户授权，这样他们才能设置会话授权标识。

  但是，在将现有 V8 数据库升级到 DB2 V9.1 或更高版本的数据库系统时，拥有现有显式 DBADM 权限（例如，在 SYSCAT.DBAUTH 中授予了此权限）的用户仍能够将会话授权标识设置为任何数据库用户标识。允许这样做的目的是使现有应用程序仍能够正常运行。由于能够设置会话授权标识，因此潜在地允许用户访问所有受保护数据。为了提高安全性，可以通过执行 REVOKE SETSESSIONUSER SQL 语句来覆盖此设置。
- **锁定监视**：在 DB2 数据库管理系统的锁定监视活动中，如果指定了 HIST_AND_VALUES 收集级别，就会将与参数标记关联的值写至监视输出。值也将嵌入到锁定事件监视器所捕获的语句文本中。于是，能够访问监视输出的用户就能访问他们可能无权访问的信息。
- **活动监视**：在使用活动事件监视器的 DB2 数据库管理系统的监视活动中，如果指定了 VALUE 子句，就会将与参数标记关联的值写至监视输出；如果指定了 WITH DETAILS 子句，就会将语句文本（其中可能包含输入数据值）写至监视输出。于是，能够访问监视输出的用户就能访问他们可能无权访问的信息。为了提高安全性，只应该将 CREATE EVENT MONITOR 语句以及任何事件监视器表的访问权授予有资格的用户。
- **程序包高速缓存监视**：在使用程序包高速缓存事件监视器监视 DB2 数据库管理系统中的程序包高速缓存时，只要从程序包高速缓存中弹出了一个节，就会将语句文本（其中可能包含输入数据值）写至监视输出。为了提高安全性，只应该将 CREATE EVENT MONITOR 语句以及任何事件监视器表的访问权授予有资格的用户。
- **监视器表函数、视图和报告**：以下监视器表函数、视图和报告会显示当前执行语句或程序包高速缓存中语句的语句文本。

- SYSPROC.MON_GET_ACTIVITY_DETALS
- SYSPROC.MON_GET_PKG_CACHE_STMT
- SYSPROC.MON_GET_PKG_CACHE_STMT_DETALS
- SYSIBMADM.MON_PKG_CACHE_SUMMARY
- SYSIBMADM.MON_CURRENT_SQL
- SYSIBMADM.MON_LOCKWAITS
- SYSIBMADM.MONREPORT.LOCKWAIT
- SYSIBMADM.MONREPORT.CURRENTSQL
- SYSIBMADM.MONREPORT.PKGCACHE

语句文本可能包含输入数据值。为了提高安全性，只应该将这些表函数和报告的 EXECUTE 特权以及这些视图的 SELECT 特权授予有资格的用户。

- **跟踪**：跟踪可能会包含表数据。于是，能够访问此类跟踪的用户就能访问他们可能无权访问的信息。
- **转储文件**：为了便于调试某些问题，DB2 数据库产品可能会在 `sqllib\db2dump` 目录中生成内存转储文件。这些内存转储文件可能包含表数据。在此情况下，能够访问这些文件的用户就能访问他们可能无权访问的信息。为了提高安全性，应该限制对 `sqllib\db2dump` 目录的访问。
- **db2dart**：**db2dart** 工具检查数据库并报告它找到的任何体系结构错误。此工具可以访问表数据，且 DB2 不对该访问强制实施访问控制。于是，有权运行 **db2dart** 工具或有权访问 **db2dart** 输出的用户就能访问他们可能无权访问的信息。
- **REOPT 绑定选项**：指定了 **REOPT** 绑定选项时，在运行时，每个可重新优化的增量绑定 SQL 语句的说明快照信息都保存在说明表中。说明快照还会显示输入数据值。
- **db2cat**：**db2cat** 工具用来转储表的压缩描述符。表的压缩描述符包含统计信息，这些统计信息可能会透露有关表内容的信息。于是，运行 **db2cat** 工具或者有权访问输出的用户就能访问他们可能无权访问的信息。

数据加密

DB2 数据库系统提供了若干方法来对存储器中的数据和通过网络传输的数据进行加密。

对存储器中的数据进行加密

可以选择下列方法来对存储器中的数据进行加密:

- 可以使用加密和解密内置函数 ENCRYPT、DECRYPT_BIN、DECRYPT_CHAR 和 GETHINT 来对数据库表中的数据进行加密。
- 可以使用 IBM Database Encryption Expert 来对底层操作系统数据和备份文件进行加密。
- 如果在 AIX 操作系统上运行 DB2 Enterprise Server Edition 系统，并且仅关注文件级别加密，那么可使用加密文件系统 (EFS) 来对操作系统数据和备份文件进行加密。

对传输的数据进行加密

要对在客户机与 DB2 数据库之间传输的数据进行加密，可以使用 DATA_ENCRYPT 认证类型，或者使用 DB2 数据库系统对“安全套接字层”(SSL) 的支持。

使用 ENCRYPT、DECRYPT_BIN、DECRYPT_CHAR 和 GETHINT 函数

ENCRYPT 内置函数使用基于密码的加密方法对数据进行加密。这些函数还允许您封装密码提示。密码提示嵌入在加密数据中。一旦加密，对数据进行解密的唯一方式是通过使用正确的密码来解密。选择使用这些函数的开发者应该对忘记的密码和不能用的数据如何管理进行计划。

ENCRYPT 函数的结果是 VARCHAR FOR BIT DATA（最大长度为 32631 字节）。

只能加密 CHAR、VARCHAR 和 FOR BIT DATA。

DECRYPT_BIN 和 DECRYPT_CHAR 函数使用基于密码的解密对数据进行解密。

DECRYPT_BIN 始终返回 VARCHAR FOR BIT DATA，而 DECRYPT_CHAR 始终返回 VARCHAR。因为第一个自变量可能是 CHAR FOR BIT DATA 或 VARCHAR FOR BIT DATA，所以存在结果与第一个自变量不同的情况。

结果的长度取决于到下一个 8 字节边界的字节数。当指定可选提示参数时，结果的长度可能是数据自变量的长度加上 40 再加上到下一个 8 字节边界的字节数。或者，如果未指定可选提示参数，结果的长度可能是数据自变量的长度加上 8 再加上到下一个 8 字节边界的字节数。

GETHINT 函数返回封装的密码提示。密码提示是将帮助数据所有者回忆起密码的短语。例如，可以将“大海”这个单词用作回忆密码“太平洋”的提示。

以下列两种方式之一确定用于对数据加密的密码:

- 密码自变量。密码是当调用 ENCRYPT 函数时显式传送的字符串。使用给出的密码对数据进行加密和解密。
- 加密密码专用寄存器。SET ENCRYPTION PASSWORD 语句对密码值进行加密，并将加密后的密码发送至数据库管理器以存储在专用寄存器中。未使用密码参数调用的 ENCRYPT、DECRYPT_BIN 和 DECRYPT_CHAR 函数使用 ENCRYPTION PASSWOED 专用寄存器中的值。ENCRYPTION PASSWORD 专用寄存器只以加密格式存储。

  专用寄存器的初始或缺省值是一个空字符串。

密码的有效长度在 6 和 127 之间，包括 6 和 127。提示的有效长度在 0 和 32 之间，包括 0 和 32。

配置 DB2 实例中的安全套接字层 (SSL) 支持

DB2 数据库系统支持 SSL，这意味着也支持 SSL 的 DB2 客户机应用程序可以使用 SSL 套接字连接至 DB2 数据库。CLI、CLP 和 .Net Data Provider 客户机应用程序和使用 IBM 数据服务器 JDBC 和 SQLJ 驱动程序（4 类连接）的应用程序支持 SSL。

开始之前

在配置 SSL 支持之前，请执行下列步骤:

- 在 Windows 平台上，确保 IBM Global Security Kit (GSKit) 库的路径出现在 **PATH** 环境变量中；在 Linux 和 UNIX 平台上，确保该路径出现在 **LIBPATH**、**SHLIB_PATH** 或 **LD_LIBRARY_PATH** 环境变量中。当安装 DB2 数据库系统时，会自动包括 GSKit。

  在 Windows 32 位平台上，GSKit 库位于 `C:\Program Files\IBM\GSK8\lib` 中。在此情况下，系统 **PATH** 必须包括 `C:\Program Files\IBM\GSK8\lib`。在 Windows 64 位平台上，64 位 GSKit 库位于 `C:\Program Files\IBM\GSK8\lib64` 中，而 32 位 GSKit 库位于 `C:\Program Files (x86)\IBM\GSK8\lib` 中。

  在 UNIX 和 Linux 平台上，GSKit 库位于 `sqllib/lib/gskit` 中。

  在非 Windows 平台上，DB2 数据库管理器以本地方式安装 GSKit，对于给定实例，GSKit 库将位于 `sqllib/lib/gskit` 或 `sqllib/lib64/gskit` 中。没有必要在全局位置安装 GSKit 的另一个副本来启用实例。如果存在 GSKit 的全局副本，请使全局 GSKit 与局部 GSKit 处于同一版本。
- 确保未激活连接集中器。如果正在运行连接集中器，将不会在 DB2 实例中启用 SSL 支持。

  要确定是否激活了连接集中器，请发出 **GET DATABASE MANAGER CONFIGURATION** 命令。如果将配置参数 **max_connections** 的值设置为大于 **max_coordagents** 的值，那么会激活连接集中器。

关于此任务

SSL 通信将始终为 FIPS 方式。

DB2 Connect 的 SSL 支持

如果在中间服务器计算机上使用 DB2 Connect for System i、DB2 Connect for System z 或 DB2 企业服务器版将 DB2 客户机连接至主机或 System i 数据库，那么 SSL 支持在下列任何配置可用:

- 在客户机与 DB2 Connect 服务器之间
- 在 DB2 Connect 服务器与服务器之间
- 同时在客户机与 DB2 Connect 服务器之间以及 DB2 Connect 服务器与服务器之间

注: 为了在配置中所有路径上启用 SSL 支持，每台客户机或服务器都必须满足 SSL 支持的所有要求。例如，如果 DB2 Connect 连接集中器处于打开状态，那么对 DB2 Connect 服务器的入站请求不能使用 SSL。但是，对目标服务器的出站请求可以使用 SSL。

高可用性灾难恢复 (HADR) 系统的 SSL 支持

在客户机与 HADR 主服务器之间支持 SSL。连接至使用 SSL 的 HADR 主服务器的客户机能够重新路由至使用 SSL 的 HADR 备用数据库。但是，在 HADR 主服务器与 HADR 备用服务器之间不支持 SSL。

GSKit 工具 GSKCapiCmd 的文档

有关 GSKit 工具 GSKCapiCmd 的信息，请参阅以下网址提供的 *GSKCapiCmd User's Guide*: ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf。

配置 SSL 支持

为了配置 SSL 支持，您首先创建密钥数据库来管理数字证书。这些证书和加密密钥用于建立 SSL 连接。其次，DB2 实例所有者必须为 SSL 支持配置 DB2 实例。

过程

1. 创建密钥数据库并设置数字证书。
   a. 使用 GSKCapiCmd 工具来创建密钥数据库。它必须为证书管理系统 (CMS) 类型的密钥数据库。 GSKCapiCmd 为非基于 Java 的命令行工具，不需要在系统上安装 Java 就能使用此工具。

      您使用 **gskcapicmd** 命令来调用 GSKCapiCmd，如 *GSKCapiCmd User's Guide* 中所述。在 Linux 和 UNIX 平台上，该命令的路径为 sqllib/gskit/bin，在 32 位和 64 位 Windows 平台上，则为 C:\Program Files\IBM\GSK8\bin。（在 64 位平台上，还存在 32 位 GSKit 可执行文件和库；在此情况下，该命令的路径为 C:\Program Files (x86)\IBM\GSK8\bin。）请确保 PATH（在 Windows 平台上）包括正确的 GSKit 库路径；LIBPATH、SHLIB_PATH 或 LD_LIBRARY_PATH（在 UNIX 或 Linux 平台上）包括正确的 GSKit 库路径，例如，sqllib/lib64/gskit。

      例如，以下命令创建称为 mydbserver.kdb 的密钥数据库以及称为 mydbserver.sth 的隐藏文件:

      ```
      gsk8capicmd_64 -keydb -create -db "mydbserver.kdb" -pw "myServerPassw0rdpw0"
            -stash
      ```

      -stash 选项会在密钥数据库所在的路径上创建隐藏文件，其文件扩展名为 .sth。实例启动时，GSKit 会使用隐藏文件来获取密钥数据库的密码。

      注: 应该对隐藏文件使用强文件系统保护。缺省情况下，只有实例所有者才具有访问此文件的权限（读写访问权）。

      当创建密钥数据库时，会自动使用来自一些诸如 Verisign 之类的认证中心 (CA) 的签署者证书对它进行填充。

   b. 将服务器的证书添加至密钥数据库。在 SSL 握手期间，服务器会将此证书发送至客户机来为服务器提供认证。 要获取证书，可以使用 GSKCapiCmd 来创建新的证书请求并将它提交至 CA 以便签署，也可以创建自签名证书以用于测试。

      例如，要创建标签为 myselfsigned 的自签名证书，请按以下示例中所示的方式使用 **GSKCapiCmd** 命令:

      ```
      gsk8capicmd_64 -cert -create -db "mydbserver.kdb" -pw "myServerPassw0rdpw0"
             -label "myselfsigned" -dn "CN=myhost.mycompany.com,O=myOrganization,
             OU=myOrganizationUnit,L=myLocation,ST=ON,C=CA"
      ```

   c. 将刚才创建的证书抽取至文件，以便可将它分发给运行客户机（将与 DB2 服务器建立 SSL 连接）的计算机。

      例如，以下 GSKCapiCmd 命令将证书抽取至称为 mydbserver.arm 的文件:

      ```
      gsk8capicmd_64 -cert -extract -db "mydbserver.kdb" -pw "myServerPassw0rdpw0"
            -label "myselfsigned" -target "mydbserver.arm" -format ascii -fips
      ```

2. 要针对 SSL 支持设置 DB2 服务器，以 DB2 实例所有者身份登录并设置下列配置参数和 **DB2COMM** 注册表变量。

a. 将 **ssl_svr_keydb** 配置参数设置为密钥数据库文件的标准路径。 例如:

```
db2 update dbm cfg using SSL_SVR_KEYDB
         /home/test/sqllib/security/keystore/key.kdb
```

如果 **ssl_svr_keydb** 为 NULL（未设置），那么不会启用 SSL 支持。

b. 将 **ssl_svr_stash** 配置参数设置为隐藏文件的标准路径。 例如:

```
db2 update dbm cfg using SSL_SVR_STASH
        /home/test/sqllib/security/keystore/mydbserver.sth
```

如果 **ssl_svr_stash** 为 NULL（未设置），那么不会启用 SSL 支持。

c. 将 **ssl_svr_label** 配置参数设置为服务器的数字证书的标签，在步骤 1 中已添加该数字证书。 如果未设置 **ssl_svr_label**，那么会使用密钥数据库中的缺省证书。如果密钥数据库中不存在任何缺省证书，那么不会启用 SSL。 例如: `db2 update dbm cfg using SSL_SVR_LABEL myselfsigned`，其中 *myselfsigned* 是样本标签。

d. 将 **ssl_svcename** 配置参数设置为 DB2 数据库系统应该进行侦听以获取 SSL 连接的端口。 如果同时启用了 TCP/IP 和 SSL（**DB2COMM** 注册表变量设置为“TCPIP, SSL”），那么必须将 **ssl_svcename** 为与为 **svcename** 设置的端口不同的端口。 **svcename** 配置参数设置 DB2 数据库系统进行侦听以获取 TCP/IP 连接的端口。 如果将 **ssl_svcename** 与 **svcename** 设置为同一端口，那么将不会启用 TCP/IP 和 SSL 之中的任何一项。 如果 **ssl_svcename** 为 NULL（未设置），那么不会启用 SSL 支持。

注: 在 HADR 环境中，请不要对主或备用数据库系统将 **hadr_local_svc** 设置为对 **ssl_svcename** 设置的值。另外，请不要将 **hadr_local_svc** 设置为 **svcename** 的值或 **svcename** 的值加一。

注: 当 **DB2COMM** 注册表变量设置为“TCPIP,SSL”时，如果未正确启用 TCPIP 支持（例如，由于 **svcename** 配置参数设置为 NULL），那么会返回错误 SQL5043N 并且不会启用 SSL 支持。

e. （可选）如果要指定服务器可以使用哪些密码套件，那么设置 **ssl_cipherspecs** 配置参数。 如果将 **ssl_cipherspecs** 保留为 NULL（未设置），那么这允许 GSKit 使用同时受客户机和服务器支持的最强可用密码套件。 请参阅第 66 页的『受支持的密码套件』，以获取有关哪些密码套件可用的信息。

f. 将值 SSL 添加至 **DB2COMM** 注册表变量。 例如:

```
db2set -i db2inst1 DB2COMM=SSL
```

其中 *db2inst1* 是 DB2 实例名称。 数据库管理器可以同时支持多个协议。例如，要同时启用 TCP/IP 和 SSL 通信协议:

```
db2set -i db2inst1 DB2COMM=SSL,TCPIP
```

g. 重新启动 DB2 实例。 例如:

```
db2stop
db2start
```

示例

以下示例演示了如何显示证书。此示例使用由以下命令创建的自签名证书:

```
gsk8capicmd_64 -cert -create -db "mydbserver.kdb" -pw "mydbserverpw0"
  -label "myselfsigned" -dn "CN=myhost.mycompany.com,O=myOrganization,
        OU=myOrganizationUnit,L=myLocation,ST=ON,C=CA"
```

要显示证书，请发出以下命令:

```
gsk8capicmd_64 -cert -details -db "mydbserver.kdb" -pw "mydbserverpw0"
  -label "myselfsigned"
```

输出显示如下:

```
label : myselfsigned
key size : 1024
version : X509 V3
serial : 96c2db8fa769a09d
issue:CN=myhost.mycompany.com,O=myOrganization,OU=myOrganizationUnit,
   L=myLocation,ST=ON,C=CA
subject:CN=myhost.mycompany.com,O=myOrganization,OU=myOrganizationUnit,
   L=myLocation,ST=ON,C=CA
not before : Tuesday, 24 February 2009 17:11:50 PM
not after : Thursday, 25 February 2010 17:11:50 PM
public Key
    30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01
    05 00 03 81 8D 00 30 81 89 02 81 81 00 B6 B8 DC
    79 69 62 C9 A5 C1 5C 38 31 53 AB 27 BE 63 C0 DB
    DE C6 BC 2E A4 0D 37 45 95 22 0E 83 32 FE 67 A9
    2F D7 51 FF 40 A3 76 68 B9 E3 34 CB 7D 4A D8 38
    CA B1 6B 32 66 74 8F E2 B8 DA 8F D0 F3 62 04 BE
    C4 FE 80 2A D0 FF 27 72 37 9A 36 1D DB D3 A1 33
    A1 A6 48 33 E9 64 B9 9B 6B DB 08 60 7D 5E 0E 20
    0A 26 AA 62 3A DF D3 78 56 DC 15 DE 9F 0B 91 DD
    3B 1B 2B E2 82 FA 24 FF 81 A3 F7 3F C1 02 03 01
    00 01
public key type : RSA : 1.2.840.113549.1.1.1
finger print : SHA1 :
    2D C1 93 F8 AC A0 8F E2 C2 05 D8 23 D7 5D 87 E6
    82 3C 47 EC
signature algorithm : SHA1WithRSASignature : 1.2.840.113549.1.1.5
value
    0E 80 24 98 F6 6E 89 43 76 57 76 7F 82 95 18 6A
    43 A5 81 EC F4 82 1F 1F F2 3F E5 61 67 48 C0 59
    94 17 8E 8F DE 4F 7C 35 0C 5D A7 98 73 2A 34 7D
    1E BA 53 78 A5 E4 31 45 D1 08 86 BE 5E 57 C6 9D
    B5 E7 A7 01 3F 54 01 5E 8F 8B 2F 66 19 24 1E A4
    94 58 B0 D4 40 95 AB 98 C2 EF 1C 5C 4A 29 48 EC
    8C C0 A2 B1 AC 2A E9 3C 14 E5 77 B2 A6 55 A8 21
    CB 59 81 86 79 F0 46 35 F8 FC 99 2D EC D4 B9 EB
Trusted : enabled
```

要为您的服务器获得 CA 签名证书（代替自签名证书），需要生成证书签名请求并向知名 CA（如 VeriSign）支付费用以获得证书签名。在您获得已签名的证书后，需要将其接收至服务器密钥数据库。以下示例演示了如何请求和接收证书。该示例中使用了证书的试用版本。

1. 首先，为 mydbserver.kdb 创建证书签名请求 (CSR)。以下命令用于在指定密钥数据库中创建新 RSA 私人/公用密钥对和 PKCS10 证书请求。对于 CMS 密钥数据库，证书请求信息将保存在扩展名为".rdb"的文件中。由 **-file** 选项指定的文件为需要发送至 CA 的文件。

   ```
   gsk8capicmd_64 -certreq -create -db "mydbserver.kdb" -pw "mydbserverpw0"
      -label "mycert" -dn "CN=myhost.mycompany.com,
       O=myOrganization,OU=myOrganizationUnit,L=myLocation,ST=ON,C=CA",
      -file "mycertRequestNew"
   ```

   以下命令将列出 my db 服务器的证书请求的详细信息:

```
gsk8capicmd_64 -certreq -details -showOID -db "mydbserver.kdb"
   -pw "mydbserverpw0" -label "mycert"
```

输出将显示如下:

```
label : mycert
key size : 1024
subject :    Common Name (CN):
    Type : 2.5.4.3
    Value: myhost.mycompany.com
   Organization (O):
    Type : 2.5.4.10
    Value: myOrganization
   Organizational Unit (OU):
    Type : 2.5.4.11
    Value: myOrganizationUnit
   Locality (L):
    Type : 2.5.6.3
    Value: myLocation
   State (ST):
    Type : ?
    Value: Ontario
   Country or region (C):
    Type : 2.5.4.6
    Value: CA
public Key
    30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01
    05 00 03 81 8D 00 30 81 89 02 81 81 00 9C B4 62
    3C 89 02 4E B0 D8 EA 0B B8 CC 70 63 4A 59 1F 0F
    FD 98 9A 1A 39 94 E3 43 C1 63 7A CD 21 47 57 D9
    86 6F 11 B8 91 08 AC E3 E2 21 32 FE 43 1F 07 C9
    F5 40 6B 3E 4D 56 35 05 62 D6 78 0B E3 97 28 F7
    27 31 A4 05 BE F2 3A 44 6B D8 D1 FF 1E DA 59 63
    E6 49 52 39 45 9C 1E 8E CC DA A1 D9 0F 3A 96 09
    66 5C 89 23 2E EE 31 65 8D 87 8E B9 61 C6 69 BC
    A5 DB EB 03 16 E6 33 85 14 68 BC DD F1 02 03 01
    00 01
finger print :
e0dcde10ded3a46a53c0190e84cc994e
5d7e4bad
attributes
signature algorithm1.2.840.113549.1.1.5
value
    4F 06 B4 E3 1F 00 B4 81 90 CC A2 99 4A 02 68 D0
    84 B5 7F 33 0B F0 04 D5 7D 4C 5C CB 5C D3 37 77
    E2 6D 10 17 50 19 D0 7F 61 C7 C8 54 7B DB CD 6F
    47 9F 7E 7E 5A CC 64 20 85 95 A8 5E C7 7D FB F4
    8A 7F 4B 74 6F 0A C6 EF 09 E7 0A 15 17 CC 1D D2
    5D ED 02 A1 BE 1D FC F2 65 EB 0D E2 93 BC 88 4C
    4C 73 76 16 9F 1B 12 3B 7A 01 CF E0 63 97 E8 38
    02 FB 47 EE F2 17 54 66 4D F7 7F 9E 13 DA 76 A2
```

要显示证书请求文件:

```
$ cat mycertRequestNew

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCQ0ExEDAOBgNVBAgTB09udGFyaW8xEDAO
BgNVBAcTB01hcmtoYW0xDDAKBgNVBAoTA0lCTTEMMAoGA1UECxMDREIyMR8wHQYD
VQQDExZnaWxlcmEudG9yb2xhYi5pYm0uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQCctGI8iQJOsNjqC7jMcGNKWR8P/ZiaGjmU40PBY3rNIUdX2YZvEbiR
CKzj4iEy/kMfB8n1QGs+TVY1BWLWeAvjlyj3JzGkBb7yOkRr2NH/HtpZY+ZJUjlF
nB6OzNqh2Q86lglmXIkjLu4xZY2Hjrlhxmm8pdvrAxbmM4UUaLzd8QIDAQABoAAw
DQYJKoZIhvcNAQEFBQADgYEATwa04x8AtIGQzKKZSgJo0IS1fzML8ATVfUxcy1zT
```

```
N3fibRAXUBnQf2HHyFR7281vR59+flrMZCCFlahex3379Ip/S3RvCsbvCecKFRfM
HdJd7QKhvh388mXrDeKTvIhMTHN2Fp8bEjt6Ac/gY5foOAL7R+7yF1RmTfd/nhPa
dqI=
-----END NEW CERTIFICATE REQUEST-----
```

如果您需要删除证书请求，请使用类似以下示例的命令:

```
gsk8capicmd_64 -certreq -delete -db "mydbserver.kdb" -pw "mydbserverpw0"
   -label "mycert"
```

2. 然后，访问 VeriSign Web 站点并进行注册，此站点将要求您剪切并粘贴请求文件以提交请求。对于试用版本，您将收到一封包含已签名的证书的电子邮件。该电子邮件还包含用于下载试用根 CA 证书及试用中间 CA 证书的链接。使用记事本或 vi 将所有三个证书都保存至文件中:

   - RootCert.arm
   - IntermediateCert.arm
   - MyCertificate.arm

   这三个证书组成一个信任链。

   使用以下命令将试用根 CA 证书添加至 mydbserver.kdb:

   ```
   gsk8capicmd_64 -cert -add -db "mydbserver.kdb" -pw "mydbserverpw0"
      -label "trialRootCACert"  -file RootCert.arm -format ascii
   ```

   使用以下命令将试用中间 CA 证书添加至 mydbserver.kdb:

   ```
   gsk8capicmd_64 -cert -add -db "mydbserver.kdb" -pw "mydbserverpw0"
      -label "trialIntermediateCACert" -file IntermediateCert.arm -format ascii
   ```

   使用以下命令将试用证书接收至 mydbserver.kdb:

   ```
   $ cat SSLCertificate.cer2

   -----BEGIN CERTIFICATE-----
   MIIFVjCCBD6gAwIBAgIQdOydrySM+J4uUPNzbPHhVjANBgkqhkiG9w0BAQUFADCB
   yzELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMTAwLgYDVQQL
   EydGb3IgVGVzdCBQdXJwb3NlcyBPbmx5LiAgTm8gYXNzdXJhbmNlcy4xQjBABgNV
   BAsTOVRlcm1zIG9mIHVzZSBhdCBodHRwczovL3d3dy52ZXJpc2lnbi5jb20vY3Bz
   L3Rlc3RjYSAoYykwNTEtMCsGA1UEAxMkVmVyaVNpZ24gVHJpYWwgU2VjdXJlIFNl
   cnZlciBUZXN0IENBMB4XDTA5MDIyMzAwMDAwMFoXDTA5MDMwOTIzNTk1OVowgaox
   CzAJBgNVBAYTAkNBMRAwDgYDVQQIEwdPbnRhcmlvMRAwDgYDVQQHFAdNYXJraGFt
   MQwwCgYDVQQKFANJQk0xDDAKBgNVBAsUA0RCMjE6MDgGA1UECxQxVGVybXMgb2Yg
   dXNlIGF0IHd3dy52ZXJpc2lnbi5jb20vY3BzL3Rlc3RjYSAoYykwNTEfMB0GA1UE
   AxQWZ2lsZXJhLnRvcm9sYWIuaWJtLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAw
   gYkCgYEAnLRiPIkCTrDY6gu4zHBjSlkfD/2Ymho5lONDwWN6zSFHV9mGbxG4kQis
   4+IhMv5DHwfJ9UBrPk1WNQVi1ngL45co9ycxpAW+8jpEa9jR/x7aWWPmSVI5RZwe
   jszaodkPOpYJZlyJIy7uMWWNh465YcZpvKXb6wMW5jOFFGi83fECAwEAAaOCAdcw
   ggHTMAkGA1UdEwQCMAAwCwYDVR0PBAQDAgWgMEMGA1UdHwQ8MDowOKA2oDSGMmh0
   dHA6Ly9TVlJTZWN1cmUtY3JsLnZlcmlzaWduLmNvbS9TVlJUcmlhbDIwMDUuY3Js
   MEoGA1UdIARDMEEwPwYKYIZIAYb4RQEHFTAxMC8GCCsGAQUFBwIBFiNodHRwczov
   L3d3dy52ZXJpc2lnbi5jb20vY3BzL3Rlc3RjYTAdBgNVHSUEFjAUBggrBgEFBQcD
   AQYIKwYBBQUHAwIwHwYDVR0jBBgwFoAUZiKOgeAxWd0qf6tGxTYCBnAnh1oweAYI
   KwYBBQUHAQEEbDBqMCQGCCsGAQUFBzABhhhodHRwOi8vb2NzcC52ZXJpc2lnbi5j
   b20wQgYIKwYBBQUHMAKGNmh0dHA6Ly9TVlJTZWN1cmUtYWlhLnZlcmlzaWduLmNv
   bS9TVlJUcmlhbDIwMDUtYWlhLmNlcjBuBggrBgEFBQcBDARiMGChXqBcMFowWDBW
   FglpbWFnZS9naWYwITAfMAcGBSsOAwIaBBRLa7kolgYMu9BSOJsprEsHiyEFGDAm
   FiRodHRwOi8vbG9nby52ZXJpc2lnbi5jb20vdnNsb2dvMS5naWYwDQYJKoZIhvcN
   AQEFBQADggEBAKs1YpIeOAL6mTryIXpYfokkzRdwP5ooDutHhVbRYcPwq9ynOrHM
   3gZolv8th5PpSkZAGTPr3HJZG6HnxRiQjPT88PAADR3SEzVMzQEESHfYToF1qBPZ
   svigphI9eIHcg5IWwv7dyuXtkFGbTCqcvEqJiT3UHhubgMfoTuTGayhNoGt75FGU
   h4kSJz3af6MNuGmQLs4wzJTepU7srlhGV1C1ujTCydax2BiWfWwO4YaFcckvHxbR
   6I7vVj1PTC2RO8n5qcWJYmGU0PG3d58hJETD4E8tAReh21ShBWDgn4+e0k1XtQ8K
   lB66QpsFYGTLtGyd/4w4BAgq/QLmcs+mpjc=
   ```

```
-----END CERTIFICATE-----

gsk8capicmd_64 -cert -receive -file MyCertificate.arm -db "mydbserver.kdb"
   -pw "mydbserverp -format ascii
```

使用以下命令列出 mydbserver.kdb 中的所有证书:

```
gsk8capicmd_64 -cert -list all -db "mydbserver.kdb" -pw "mydbserverpw0"

certificates found
* default, - personal, ! trusted
-!      mycert
!       trialIntermediateCACert
!       trialRootCACert
-!      myselfsigned
db2 update dbm cfg using SSL_SVR_LABEL mycert
```

在非 Java DB2 客户机中配置安全套接字层 (SSL) 支持

可以将诸如 CLI、CLP 和 .Net Data Provider 客户机之类的 DB2 数据库客户机配置为支持安全套接字层 (SSL) 以便与 DB2 服务器进行通信。

开始之前

注: 如果 V9.7 的 DB2 客户机或 DB2 Connect 服务器与 z/OS V1.8、V1.9 或 V1.10 系统上 DB2 z/OS 版服务器建立 SSL 连接，那么必须将 APAR PK72201 的相应 PTF 应用于 Communication Server for z/OS IP Services。

注: 由于 GSKit V8 与 GSKit 7.0.4.20 之前的 GSKit V7d 不兼容，与 IDS 数据服务器（使用 GSKit 7.0.4.20 之前的 GSKit V7d）的 CLI 应用程序连接将失败。要纠正该问题，请将 IDS 数据服务器上的 GSKit 库升级到 GSKit 7.0.4.20 或更高版本

在为客户机配置 SSL 支持之前，请执行下列步骤:

- 如果客户机和服务器位于同一物理计算机上，那么不需要安装 GSKit，因为 GSKit 已自动随 DB2 服务器一起进行了安装。

  自 V9.7 FP1 起，当您安装 DB2 服务器的 64 位版本时，安装中将自动包括 32 位 GSKit 库。要使用这些库，在 Linux 和 UNIX 操作系统上，必须确保已正确设置 **LD_LIBRARY_PATH**、**LIBPATH** 或 **SHLIB_PATH** 环境变量。在 Windows 操作系统上，请确保已正确设置 **PATH** 环境变量，如下表中所示。

应用程序	操作系统	GSKit 库的位置	环境变量设置
32 位	Linux 和 UNIX 64 位	*$INSTHOME*/sqllib/lib32/gskit	在 *LD_LIBRARY_PATH*、*LIBPATH* 或 *SHLIB_PATH* 环境变量中包括 *$INSTHOME*/sqllib/lib32/gskit。
64 位	Linux 和 UNIX 64 位	*$INSTHOME*/sqllib/lib64/gskit	在 *LD_LIBRARY_PATH*、*LIBPATH* 或 *SHLIB_PATH* 环境变量中包括 *$INSTHOME*/sqllib/lib64/gskit。
32 位	Windows 64 位	C:\Program Files (x86)\IBM\GSK8\lib	在 *PATH* 环境变量中包括 C:\Program Files (x86)\IBM\GSK8\lib
64 位	Windows 64 位	C:\Program Files\IBM\GSK8\lib64	在 *PATH* 环境变量中包括 C:\Program Files\IBM\GSK8\lib64

SSL 通信将始终为 FIPS 方式。

在非 Windows 平台上，DB2 数据库管理器以本地方式安装 GSKit，对于给定实例，GSKit 库将位于 sqllib/lib/gskit 或 sqllib/lib64/gskit 中。没有必要在全局位置安装 GSKit 的另一个副本。如果存在 GSKit 的全局副本，请使全局 GSKit 与局部 GSKit 处于同一版本。

- 当将客户机安装在另一台计算机上时，如果基于“C”的客户机使用 SSL 来与服务器通信，那么对于这些客户机，必须安装 GSKit。可以从“IBM DB2 Support Files for SSL Functionality DVD”安装 GSKit 库。或者，可以通过已从 Passport Advantage® 下载的映像进行安装。
  - 在 Windows 上，确保 IBM Global Security Kit (GSKit) 库的路径出现在 **PATH** 环境变量中；在 Linux 和 UNIX 上，确保该路径出现在 **LIBPATH**、**SHLIB_PATH** 或 **LD_LIBRARY_PATH** 环境变量中。例如，在 Windows 上，将 GSKit bin 和 lib 目录添加至 **PATH** 环境变量:

    ```
    set PATH="C:\Program Files\ibm\gsk8\bin";%PATH%
    set PATH="C:\Program Files\ibm\gsk8\lib";%PATH%
    ```

GSKit 工具 GSKCapiCmd 的文档

有关 GSKit 工具 GSKCapiCmd 的信息，请参阅以下网址提供的 *GSKCapiCmd User's Guide*: ftp://ftp.software.ibm.com/software/webserver/appserv/library/v80/GSK_CapiCmd_UserGuide.pdf。

关于此任务

SSL 通信将始终为 FIPS 方式。

过程

要在 DB2 客户机中配置 SSL 支持:

1. 获取客户机上服务器数字证书的签署者证书。服务器证书可以是自签名证书或由认证中心 (CA) 签署的证书。
   - 如果服务器证书是自签名证书，那么必须将其签署者证书抽取至服务器计算机上的文件，然后将它分发给运行客户机（将与该服务器建立 SSL 连接）的计算机。请参阅第 53 页的『配置 DB2 实例中的安全套接字层 (SSL) 支持』，以获取有关如何将该证书抽取至文件的信息。
   - 如果服务器证书由熟悉的 CA 签署，那么客户机密钥数据库可能已包含签署了服务器证书的 CA 证书。如果没有包含，那么必须获取该 CA 证书，通常通过访问该 CA 的 Web 站点来完成此任务。
2. 在 DB2 客户机上，使用 GSKCapiCmd 工具来创建类型为 CMS 的密钥数据库。GSKCapiCmd 工具为非基于 Java 的命令行工具（不需要在系统上安装 Java 就能使用此工具）。

   您使用 **gskcapicmd** 命令来调用 GSKCapiCmd，如 *GSKCapiCmd User's Guide* 中所述。在 Linux 和 UNIX 操作系统上，该命令的路径为 sqllib/gskit/bin，在 32 位和 64 位 Windows 操作系统上，此路径为 C:\Program Files\IBM\GSK8\bin。（在 64 位操作系统上，还存在 32 位 GSKit 可执行文件和库；在此情况下，该命令的路径为 C:\Program Files (x86)\IBM\GSK8\bin。）

例如，以下命令创建称为 mydbclient.kdb 的密钥数据库以及称为 mydbclient.sth 的隐藏文件:

```
gsk8capicmd_64 -keydb -create -db "mydbclient.kdb" -pw "myClientPassw0rdpw0"
      -stash
```

-stash 选项会在密钥数据库所在的路径上创建隐藏文件，其文件扩展名为 .sth。在连接时，GSKit 会使用隐藏文件来获取密钥数据库的密码。

3. 将签署者证书添加到客户机密钥数据库中

   例如，以下 **gsk8capicmd** 命令会将该证书从文件 mydbserver.arm 导入到称为 mydbclient.kdb 的密钥数据库中:

   ```
   gsk8capicmd_64 -cert -add -db "mydbclient.kdb" -pw "myClientPassw0rdpw0"
         -label "dbselfsigned" -file "mydbserver.arm" -format ascii -fips
   ```

4. 对于客户机应用程序，设置适当的连接字符串或配置参数，如客户机的适用示例中所示。

示例

CLP 和嵌入式 SQL 客户机

CLP 客户机和嵌入式 SQL 客户机可以连接至远程主机上的数据库，已使用 **CATALOG TCPIP NODE** 命令将该远程主机添加至节点目录。发出 **CATALOG TCPIP NODE** 命令，**SECURITY** 关键字设置为 SSL 以对该连接指定 SSL。

以下示例演示了如何编目节点和数据库，以便 CLP 客户机可以使用 SSL 连接来与它们建立连接。

首先，编目节点和数据库，以便客户机应用程序可与它们建立 SSL 连接:

```
catalog TCPIP NODE mynode  REMOTE  127.0.0.1  SERVER 50001  SECURITY SSL

catalog DATABASE sample  AS myssldb  AT NODE mynode  AUTHENTICATION SERVER
```

接着，使用 **ssl_clnt_keydb** 和 **ssl_clnt_stash** 配置参数来指定客户机密钥数据库和隐藏文件。您将 **ssl_clnt_keydb** 配置参数设置为密钥数据库文件 (.kdb) 的标准路径并将 **ssl_clnt_stash** 配置参数设置为隐藏文件的标准路径。

```
db2 update dbm cfg using
      SSL_CLNT_KEYDB /home/test1/sqllib/security/keystore/clientkey.kdb

SSL_CLNT_STASH /home/test1/sqllib/security/keystore/clientstore.sth
```

如果 **ssl_clnt_keydb** 或 **ssl_clnt_stash** 配置参数为 NULL（未设置），连接将失败并返回错误 SQL10013N，标记为 GSKit Error: *GSKit_return_code*。

然后，从 CLP 客户机连接至服务器:

```
db2 connect to myssldb user user1 using password
```

或者，嵌入式 SQL 应用程序可使用以下语句来进行连接:

```
Strcpy(dbAlias,"myssldb");
 EXEC SQL CONNECT TO :dbAlias USER :user USING :pswd;
```

CLI/ODBC 客户机应用程序

根据运行 CLI 应用程序的环境，您使用连接字符串参数（ssl_client_keystoredb 和 ssl_client_keystash）或 DB2 配置参数（**ssl_clnt_keydb** 和 **ssl_clnt_stash**）来指定客户机密钥数据库路径和隐藏文件路径。

- 如果使用 IBM Data Server Driver for ODBC and CLI，那么使用连接字符串参数，如以下示例中所示:

  通过包含 SECURITY=SSL 关键字的连接字符串来调用 SQLDriverConnect。例如:

  ```
  "Database=sampledb; Protocol=tcpip; Hostname= myhost; Servicename=50001;
  Security=ssl; Ssl_client_keystoredb=/home/test1/keystore/clientstore.kdb;
  Ssl_client_keystash=/home/test1/keystore/clientstore.sth;"
  ```

  在此情况下，因为指定了 Security=ssl，所以必须设置 ssl_client_keystoredb 和 ssl_client_keystash 连接字符串参数，否则，连接将失败。

- 如果使用 IBM 数据服务器客户机或 IBM Data Server Runtime Client，那么可使用连接字符串参数或 DB2 配置参数来设置客户机密钥数据库路径和存储文件路径。如果设置了 **ssl_client_keystoredb** 和 **ssl_client_keystash** 连接字符串参数，那么它们会覆盖由 **ssl_clnt_keydb** 或 **ssl_clnt_stash** 配置参数设置的任何值。

  此示例使用 db2cli.ini 文件来设置连接字符串参数:

  ```
  [sampledb]
  Database=sampledb
  Protocol=tcpip
  Hostname=myhost
  Servicename=50001
  Security=ssl
  SSL_client_keystoredb=/home/test1/keystore/clientstore.kdb
  SSL_client_keystash=/home/test1/keystore/clientstore.sth
  ```

  此示例使用 **FileDSN** CLI/ODBC 关键字来标识包含数据库连接信息的 DSN 文件，该文件设置连接字符串参数。例如，该 DSN 文件看起来可能与下面的内容相似:

  ```
  [ODBC]
  DRIVER=IBM DB2 ODBC DRIVER - DB2COPY1
  UID=user1
  AUTHENTICATION=SERVER
  PORT=50001
  HOSTNAME=myhost
  PROTOCOL=TCPIP
  DATABASE=SAMPLEDB
  SECURITY=SSL
  SSL_client_keystoredb=/home/test1/keystore/clientstore.kdb
  SSL_client_keystash=/home/test1/keystore/clientstore.sth
  ```

  在这些情况下，因为指定了 Security=ssl，所以如果没有设置 **ssl_client_keystoredb** 和 **ssl_client_keystash** 连接字符串参数并且也没有设置 **ssl_clnt_keydb** 和 **ssl_clnt_stash** 配置参数，那么连接将失败。

基于证书的认证

从 DB2 V9.7 FP6 开始，在 db2dsdriver.cfg 认证参数中引入了新的认证类型“CERTIFICATE”，如以下语法中所示:

```
<parameter name="Authentication" value="CERTIFICATE | SERVER | SERVER_ENCRYPT |
   SERVER_ENCRYPT_AES | DATA_ENCRYPT | KERBEROS | GSSPLUGIN"/>
```

基于证书的认证允许您使用 SSL 客户机认证，而不需要在数据库客户机上提供数据库密码。配置基于证书的认证以提供认证信息时，不能以任何其他方式指定密码（如在 db2dsdriver.cfg 配置文件、db2cli.ini 配置文件或连接字符串中）。由于认证参数需要指定标签，所以还引入了新的数据服务器驱动程序配

置参数 **SSLClientLabel**。如果指定了 CERTIFICATE，那么还必须在 CLI 配置文件 db2cli.ini 中或在数据服务器驱动程序配置文件 db2dsdriver.cfg 中指定新的标签参数 **SSLCLientLabel**。

从 DB2 V9.7 FP6 开始，引入了新的关键字 **SSLClientKeyStoreDBPassword** 来为通过 **SSLClientKeystoredb** 关键字指定的密钥库数据库设置密码。配置参数 **SSLClientKeystash** 和 **SSLClientKeyStoreDBPassword** 互斥。同时在 CLI 配置文件或数据服务器驱动程序配置文件中指定了 **SSLClientKeystash** 配置参数和 **SSLClientKeyStoreDBPassword** 配置参数时，会返回错误 CLI0220E。因此，要成功地完成基于证书的认证，建议仅指定其中一个关键字而不是同时指定这两个关键字。

DB2 .Net Data Provider 应用程序

借助以下方法，DB2 .Net Data Provider 应用程序可与数据库建立 SSL 连接：通过定义连接字符串参数 SSLClientKeystoredb 和 SSLClientKeystash 来指定客户机密钥数据库路径和隐藏文件路径。连接字符串还必须包含 Security=SSL。例如：

```
String connectString = "Server=myhost:50001;Database=sampledb;Security=ssl;
SSLClientKeystoredb=/home/test1/keystore/clientstore.kdb;
SSLClientKeystash=/home/test1/keystore/clientstore.sth";
```

这样，如以下 C# 代码片段中所示，要与数据库建立连接，请将此 **connectString** 传递至 **DB2Connection** 构造函数并使用 **DB2Connection** 对象的 **Open** 方法来与 **connectString** 中标识的数据库建立连接：

```
DB2Connection conn = new DB2Connection(connectString);
Conn.Open();
Return conn;
```

如果 **SSLClientKeystoredb** 或 **SSLClientKeystash** 连接字符串参数为 NULL（未设置），那么连接将失败并返回错误 SQL10013N（标记为 GSKit Error: *GSKit_return_code*）。

安全套接字层 (SSL)

DB2 数据库系统支持使用安全套接字层 (SSL) 及其后继者传输层安全性（TLS），以使客户机能够认证服务器和通过使用加密来提供客户机与服务器之间的专用通信。认证是通过交换数字证书来执行的。

注: 当本主题提到 SSL 时，除非另有说明，否则，相同信息适用于 TLS。

在没有加密的情况下，信息包通过网络时，具有访问权的任何用户都对其一览无余。可以使用 SSL 来保护在使用 TCP/IP（可将 SSL 连接视为安全 TCP/IP 连接）的所有网络中传输的数据。

客户机与服务器通过执行“SSL 握手”来建立安全 SSL 连接。

SSL 握手概述

在 *SSL 握手*期间，公用密钥算法（通常为 RSA）用来在客户机与服务器之间安全地交换数字签名和加密密钥。此标识和密钥信息用来为客户机与服务器之间的会话建立安全连接。在建立安全会话之后，会使用对称算法（例如 AES）来对客户机与服务器之间的数据传输进行加密。

在 SSL 握手期间，客户机和服务器执行下列步骤:

1. 客户机请求 SSL 连接并列示它的受支持密码套件。
2. 服务器以所选密码套件进行响应。
3. 服务器将它的数字证书发送至客户机。
4. 客户机验证服务器证书的有效性，以用于认证目的。它可通过与发出服务器证书的可信认证中心进行核对或通过检入其自己的密钥数据库来完成此步骤。
5. 客户机与服务器安全地协商会话密钥和消息认证代码（MAC）。
6. 客户机与服务器使用所选密钥和 MAC 来安全地交换信息。

注: DB2 数据库系统不支持 SSL 握手期间对客户机进行（可选）认证。

将 SSL 加密与 DB2 认证配合使用

可以将 SSL 加密与诸如 KERBEROS 或 SERVER 之类的全部现有 DB2 认证方法配合使用。您通过在 DBM 配置参数中将实例的认证类型设置为所选认证方法来照常完成此任务。

数字证书和认证中心

数字证书由可信方（称为认证中心）发出，以验证诸如客户机或服务器之类的实体的身份。

数字证书的使用有两个目的: 验证所有者的身份以及使所有者的公用密钥可用。证书发出时带有截止日期，在此日期之后，它不再由认证中心（CA）保证。

为了获取数字证书，您将请求发送至所选 CA，例如 Verisign 或 RSA。该请求包括您的专有名称、公用密钥和签名。专有名称 (DN) 是您为其申请证书的每个用户或主机的唯一标识。CA 会使用公用密钥检查您的签名并对您的身份执行某些级别的验证（这随 CA 的不同而变化）。在验证之后，CA 将已签名数字证书发送给您，该数字证书包含您的专有名称、公用密钥、该 CA 的专有名称以及该认证中心的签名。您将此已签名证书存储在密钥数据库中。

当将此证书发送给接收方时，接收方会执行以下两个步骤来验证您的身份:

1. 使用随证书提供的公用密钥来检查您的数字签名。
2. 验证发出证书的 CA 是否合法且可信。为此，接收方需要该 CA 的公用密钥。接收方可能已将该 CA 的公用密钥的受保证副本保留在其密钥数据库中，但是，如果没有，那么接收方必须另外取得数字证书来获取该 CA 的公用密钥。此证书可能又依赖于另一个 CA 的数字证书；可能存在由多个 CA 发出的证书的层次结构，每个都依赖于下一个的有效性。但是，最终，接收方需要*根 CA* 的公用密钥。根 CA 是位于该层次结构顶部的 CA。为了信任根 CA 的数字证书的有效性，公用密钥用户必须以安全方式接收该数字证书，例如，通过从已认证服务器下载、借助接收自可信源的预装入软件或使用安全交付的软盘。

将数字证书发送至接收方的许多应用程序并不仅发送其自己的证书，而且发送验证证书层次结构直至根 CA 证书所必需的全部 CA 数字证书。

要使数字证书完全可信，该数字证书的所有者必须已仔细保护其专用密钥，例如，通过在其计算机的硬盘驱动器上对该数字证书进行加密。如果其专用密钥已损坏，那么冒名顶替者可能滥用其数字证书。

可以将自签名数字证书用于测试目的。自签名数字证书包含您的专有名称、公用密钥和签名。

公用密钥密码术

SSL 使用公用密钥算法来为认证交换加密密钥信息和数字证书信息。公用密钥密码术（也称为非对称密码术）使用两种不同的加密密钥: 用于加密数据的公用密钥以及用于对其进行解密的关联专用密钥。

反之，对称密钥密码术仅使用一个密钥，安全通信中涉及的所有各方都共享该密钥。此密钥既用于对信息进行加密，也用于对信息进行解密。该密钥必须安全地分发给所有各方并由其存储，这很难保证。通过公用密钥密码术，公用密钥不保密，但是它加密的消息只能通过使用其关联的专用密钥来解密。专用密钥必须安全地存储（例如，在密钥数据库中）或在计算机的硬盘驱动器上加密。

仅有公用密钥算法不会保证通信安全，还需要对与您通信的任何用户的标识进行验证。为了执行此认证，SSL 使用数字证书。当将数字证书发送至某用户时，该证书会向其提供公用密钥。您已使用专用密钥来对证书进行数字签名，因此通信的接收方可以使用公用密钥来验证您的签名。数字证书本身的有效性由发出它的认证中心（CA）保证。

受支持的密码套件

在 SSL 握手期间，客户机与服务器协商哪个密码套件将用来交换数据。密码套件是一组用来提供认证、加密和数据完整性的算法。

DB2 数据库系统使用以 FIPS 方式运行的 GSKit 来提供 SSL 支持。GSKit 支持下列密码套件:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA

每个密码套件的名称都指定它用于认证、加密和数据完整性检查的算法。例如，密码套件 TLS_RSA_WITH_AES_256_CBC_SHA 将 RSA 用于认证、将 AES 256 位和 CBC 用作加密算法以及将 SHA-1 用作检查数据完整性的散列函数。

在 SSL 握手期间，DB2 数据库系统自动选择同时受客户机和服务器支持的最强密码套件。如果要服务器仅接受一个或多个特定密码套件，那么可以将 **ssl_cipherspecs** 配置参数设置为下列任何值:

- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_3DES_EDE_CBC_SHA
- 以上三个值的任何组合。要设置多个值，请用逗号分隔每个值，但不要在这些值之间留空格。
- Null。在此情况下，会自动选择最强的可用算法。

不能对选择哪个密码套件进行优先级设置。如果设置了 **ssl_cipherspecs** 配置参数，那么 DB2 数据库系统会选择最强的可用密码套件；此选择不取决于您在设置 **ssl_cipherspecs** 时指定密码套件的顺序。

捆绑库和处理规则

当 DB2 for Linux, UNIX, and Windows 捆绑需要 GSKit 的供应商软件时，或者当需要 GSKit 的供应商软件捆绑 DB2 for Linux, UNIX, and Windows 时，必须遵循某些规则。

库规则

当 DB2 for Linux, UNIX, and Windows 捆绑需要 GSKit 的供应商软件时，此供应商软件将提供与 DB2 for Linux, UNIX, and Windows 链接的库。这些库必须遵循某种规则。此规则称为库规则。

库规则：使用短名称

动态装入 GSKit 库时，调用者必须为装入器函数仅传递 GSKit 库的基本文件名，而不传递路径。

例如，`dlopen("libgsk8ssl_64.so", RTLD_NOW | RTLD_GLOBAL)` 正确，而 `dlopen("/usr/opt/ibm/gsk8_64/lib/libgsk8ssl_64.so", RTLD_NOW | RTLD_GLOBAL)` 不正确。

处理规则

当需要 GSKit 的供应商软件捆绑 DB2 for Linux, UNIX, and Windows 时，供应商软件会与 IBM 数据服务器客户机链接。供应商软件必须遵循某种规则。此规则称为处理规则。

处理规则：设置环境搜索路径

进程必须设置它要在其中查找 GSKit 库的环境搜索路径。进程必须执行此设置，以便所包括的库可从同一位置装入 GSKit 库。

在 AIX 上，进程可以将可执行文件的 LIBPATH 或 RPATH 设置为 GSKit 库的路径。在 **setuid** 和 **setgid** 情况下，进程可以使用 db2chglibpath 将 GSKit 的搜索路径包括在可执行文件的 RPATH 中。只有执行此操作之后，才能使用位于该位置的 GSKit 库。在 Linux、Sun 和 HP-UX 上，进程可以将 LD_LIBRARY_PATH 设置为 GSKit 库的路径。在 **setuid** 和 **setgid** 情况下，进程可以使用 db2chglibpath 将 GSKit 的搜索路径包含在 IBM 数据服务器客户机库的 RPATH 中。只有执行此操作之后，才能使用位于该位置的 GSKit 库。例如，当进程需要在服务器实例中使用全局 GSKit，或者需要在客户机或服务器实例中使用它自己的局部 GSKit 时，它可以使用 db2chglibpath 来更改 RPATH。

符号链接方法和限制

当您在 UNIX 和 Linux 平台上安装 DB2 for Linux, UNIX, and Windows 时，也会安装局部 GSKit 库。这些库位于 `<db2_install_path>/lib64/gskit_db2` 或 `<db2_install_path>/lib32/gskit_db2`。

在安装 IBM 的其他产品期间，可能会安装 GSKit 库的另一个副本。这些库可能是局部 GSKit 库，也可能是全局 GSKit 库，视所安装的产品而定。当 DB2 for Linux, UNIX, and Windows 和 IBM 提供的另一个包括 GSKit 库的产品都安装在同一台机器上时，

可能会产生某些互操作性问题。因为 GSKit 仅允许任何单个进程中存在单个 GSKit 源中的库，所以可能会产生这些互操作性问题。这些互操作性问题可能会导致不可预测的行为和运行时错误。

要确保同一台机器上存在所安装的多个 GSKit 的情况下使用 GSKit 库的单个源，可以使用符号链接方法。在初始安装 DB2 for Linux, UNIX, and Windows 期间，安装程序将创建从 <db2_install_path>/lib64/gskit 或 <db2_install_path>/lib32/gskit 到 <db2_install_path>/lib64/gskit_db2 或 <db2_install_path>/lib32/gskit_db2 的符号链接。此位置是从其中装入 GSKit 库的缺省位置。用于捆绑 DB2 for Linux, UNIX, and Windows 并更改从所提到的缺省目录到 GSKit 的另一个副本的库目录的符号链接的产品必须确保新安装的 GSKit 处于同一级别或更高级别。无论库是全局库还是局部库，都存在此限制。在更新 DB2 for Linux, UNIX, and Windows 或者对其进行升级期间，会保留此符号链接。如果新安装的副本具有指向缺省位置的符号链接，那么会保留与旧的安装副本相关联的符号链接。如果新安装的副本具有不指向缺省位置的符号链接，那么在更新的安装副本中会使用与更新的安装副本相关联的符号链接。由于符号链接 <db2_install_path>/lib64/gskit 或 <db2_install_path>/lib32/gskit 位于 DB2 for Linux, UNIX, and Windows 安装副本的路径中，因此存在某些局限性。例如，如果为任何 DB2 副本创建了两个或两个以上的实例，那么符号链接更改会影响所有实例。

在 Solaris x64 上，DB2 for Linux, UNIX, and Windows 附带包括的 GSKit 版本为 8.0.14.14 或 8.0.15.1。

示例

DB2 for Linux, UNIX, and Windows 将捆绑 LDAP 客户机。DB2 for Linux, UNIX, and Windows 进程遵循处理规则。要遵循处理规则，经由 RPATH 的环境搜索路径设置为其 GSKit 的本地副本。LDAP 客户机库将从同一位置装入 GSKit 库。设置 GSKIT_LOCAL_INSTALL_MODE 时，LDAP 客户机库（它们遵循库规则）将按 GSKit 库的基本文件名来装入 GSKit 库。

LDAP 服务器将捆绑 DB2 for Linux, UNIX, and Windows。LDAP 进程遵循处理规则。环境搜索路径设置为 GSKit 的全局副本，IBM 数据服务器客户机库将从同一位置装入 GSKit 库。IBM 数据服务器客户机库（它们遵循库规则）将按 GSKit 库的基本文件名来装入 GSKit 库。

GSKit 返回码

一些 DB2 数据库管理器消息可能会显示 IBM Global Security Kit (GSKit) 的返回码。

常规 GSKit 返回码

表 2. GSKit 常规返回码

返回码（十六进制）	返回码（十进制）	常量	说明
0x00000000	0	GSK_OK	任务已成功完成。已通过每个成功完成的函数调用发出。
0x00000001	1	GSK_INVALID_HANDLE	环境或 SSL 句柄无效。所指定句柄不是成功 open 函数调用的结果。
0x00000002	2	GSK_API_NOT_AVAILABLE	动态链接库（DLL）已卸载，不可用。（仅对于 Windows。）

表 2. GSKit 常规返回码 (续)

返回码（十六进制）	返回码（十进制）	常量	说明
0x00000003	3	GSK_INTERNAL_ERROR	内部错误。向服务报告此错误。
0x00000004	4	GSK_INSUFFICIENT_STORAGE	没有足够内存用于执行操作。
0x00000005	5	GSK_INVALID_STATE	句柄的状态对于操作无效，例如，对某个句柄执行初始化操作两次。
0x00000006	6	GSK_KEY_LABEL_NOT_FOUND	在密钥文件中找不到指定的密钥标签。
0x00000007	7	GSK_CERTIFICATE_NOT_AVAILABLE	没有从伙伴接收到证书。
0x00000008	8	GSK_ERROR_CERT_VALIDATION	证书验证错误。
0x00000009	9	GSK_ERROR_CRYPTO	处理密码术时出错。
0x0000000a	10	GSK_ERROR_ASN	验证证书中的 ASN 字段时出错。
0x0000000b	11	GSK_ERROR_LDAP	连接至 LDAP 服务器时出错。
0x0000000c	12	GSK_ERROR_UNKNOWN_ERROR	内部错误。向服务报告此错误。
0x00000065	101	GSK_OPEN_CIPHER_ERROR	内部错误。向服务报告此错误。
0x00000066	102	GSK_KEYFILE_IO_ERROR	读取密钥文件时发生 I/O 错误。
0x00000067	103	GSK_KEYFILE_INVALID_FORMAT	密钥文件具有无效内部格式。请重新创建密钥文件。
0x00000068	104	GSK_KEYFILE_DUPLICATE_KEY	密钥文件包含两个具有同一密钥的条目。请使用 iKeyman 实用程序来除去重复的密钥。
0x00000069	105	GSK_KEYFILE_DUPLICATE_LABEL	密钥文件包含两个具有同一标签的条目。请使用 iKeyman 实用程序来除去重复的标签。
0x0000006a	106	GSK_BAD_FORMAT_OR_ INVALID_PASSWORD	密钥文件密码用于完整性检查。密钥文件已损坏或密码标识不正确。
0x0000006b	107	GSK_KEYFILE_CERT_EXPIRED	密钥文件中的缺省密钥具有已到期证书。请使用 iKeyman 实用程序来除去已到期证书。
0x0000006c	108	GSK_ERROR_LOAD_GSKLIB	装入其中一个 GSKit 动态链接库时发生错误。请确保 GSKit 已正确安装。
0x0000006d	109	GSK_PENDING_CLOSE_ERROR	指出当将 GSK_ENVIRONMENT_ CLOSE_OPTIONS 设置为 GSK_DELAYED_ ENVIRONMENT_CLOSE 并且调用 gsk_environment_close() 函数之后在 GSKit 环境中尝试建立连接。
0x000000c9	201	GSK_NO_KEYFILE_PASSWORD	因为既未指定密码也未指定隐藏文件名称，所以未能初始化密钥文件。
0x000000ca	202	GSK_KEYRING_OPEN_ERROR	无法打开密钥文件或 Microsoft Certificate Store。未正确指定路径、文件许可权未允许打开文件或文件格式不正确。

表 2. GSKit 常规返回码 (续)

返回码（十六进制）	返回码（十进制）	常量	说明
0x000000cb	203	GSK_RSA_TEMP_KEY_PAIR	无法生成临时密钥对。向服务报告此错误。
0x000000cc	204	GSK_ERROR_LDAP_NO_SUCH_OBJECT	已指定找不到的“用户名”对象。
0x000000cd	205	GSK_ERROR_LDAP_INVALID_ CREDENTIALS	用于 LDAP 查询的密码不正确。
0x000000ce	206	GSK_ERROR_BAD_INDEX	LDAP 服务器的“故障转移”列表中的某个索引不正确。
0x000000cd	207	GSK_ERROR_FIPS_NOT_SUPPORTED	尝试将 GSKit 置于 FIPS 方式失败。
0x0000012d	301	GSK_CLOSE_FAILED	指出未正确处理 GSKit 环境关闭请求。此错误的原因很可能是在 gsk_close_environment() 调用之后尝试 gsk_secure_socket*() 命令。
0x00000191	401	GSK_ERROR_BAD_DATE	系统日期已设置为无效值。
0x00000192	402	GSK_ERROR_NO_CIPHERS	既未启用 SSLV2 又未启用 SSLV3。
0x00000193	403	GSK_ERROR_NO_CERTIFICATE	没有从伙伴接收到所需证书。
0x00000194	404	GSK_ERROR_BAD_CERTIFICATE	接收到的证书的格式不正确。
0x00000195	405	GSK_ERROR_UNSUPPORTED_ CERTIFICATE_TYPE	不支持接收到的证书的类型。
0x00000196	406	GSK_ERROR_IO	执行数据读或写操作时发生 I/O 错误。
0x00000197	407	GSK_ERROR_BAD_KEYFILE_LABEL	找不到密钥文件中的指定标签。
0x00000198	408	GSK_ERROR_BAD_KEYFILE_ PASSWORD	所指定密钥文件密码不正确。未能使用密钥文件。密钥文件还可能已损坏。
0x00000199	409	GSK_ERROR_BAD_KEY_LEN_ FOR_EXPORT	在受限密码术环境中，密钥大小太大，无法支持。
0x0000019a	410	GSK_ERROR_BAD_MESSAGE	从伙伴接收到格式不正确的 SSL 消息。
0x0000019b	411	GSK_ERROR_BAD_MAC	未成功验证消息认证代码（MAC）。
0x0000019c	412	GSK_ERROR_UNSUPPORTED	不支持 SSL 协议或证书类型。
0x0000019d	413	GSK_ERROR_BAD_CERT_SIG	接收到的证书包含了不正确的签名。
0x0000019e	414	GSK_ERROR_BAD_CERT	从伙伴接收到的证书的格式不正确。
0x0000019f	415	GSK_ERROR_BAD_PEER	从伙伴接收到的 SSL 协议无效。
0x000001a0	416	GSK_ERROR_PERMISSION_DENIED	向服务报告此内部错误。
0x000001a1	417	GSK_ERROR_SELF_SIGNED	自签名证书无效。
0x000001a2	418	GSK_ERROR_NO_READ_FUNCTION	读操作失败。向服务报告此内部错误。
0x000001a3	419	GSK_ERROR_NO_WRITE_FUNCTION	写操作失败。向服务报告此内部错误。

表 2. GSKit 常规返回码 (续)

返回码（十六进制）	返回码（十进制）	常量	说明
0x000001a4	420	GSK_ERROR_SOCKET_CLOSED	在协议完成之前伙伴已关闭套接字。
0x000001a5	421	GSK_ERROR_BAD_V2_CIPHER	指定的 V2 密码无效。
0x000001a6	422	GSK_ERROR_BAD_V3_CIPHER	指定的 V3 密码无效。
0x000001a7	423	GSK_ERROR_BAD_SEC_TYPE	向服务报告此内部错误。
0x000001a8	424	GSK_ERROR_BAD_SEC_ TYPE_COMBINATION	向服务报告此内部错误。
0x000001a9	425	GSK_ERROR_HANDLE_ CREATION_FAILED	未能创建句柄。向服务报告此内部错误。
0x000001aa	426	GSK_ERROR_INITIALIZATION_ FAILED	初始化失败。向服务报告此内部错误。
0x000001ab	427	GSK_ERROR_LDAP_NOT_AVAILABLE	当验证证书时，无法访问指定的 LDAP 目录。
0x000001ac	428	GSK_ERROR_NO_PRIVATE_KEY	所指定密钥未包含专用密钥。
0x000001ad	429	GSK_ERROR_PKCS11_ LIBRARY_NOTLOADED	尝试装入指定的 PKCS11 共享库失败。
0x000001ae	430	GSK_ERROR_PKCS11_TOKEN_ LABELMISMATCH	PKCS #11 驱动程序找不到调用者指定的标记。
0x000001af	431	GSK_ERROR_PKCS11_TOKEN_ NOTPRESENT	插槽中不存在 PKCS #11 标记。
0x000001b0	432	GSK_ERROR_PKCS11_TOKEN_ BADPASSWORD	用于访问 PKCS #11 标记的密码/pin 无效。
0x000001b1	433	GSK_ERROR_INVALID_V2_HEADER	接收到的 SSL 头不是格式正确的 SSLV2 头。
0x000001b2	434	GSK_CSP_OPEN_ERROR	无法访问基于硬件的密码服务提供者（CSP）。系统中未注册给定的 CSP 名称，或所指定 CSP 名称已注册但证书库未能打开。
0x000001b3	435	GSK_CONFLICTING_ATTRIBUTE_ SETTING	PKCS11、CMS 密钥数据库与 Microsoft Crypto API 之间存在属性设置冲突。
0x000001b4	436	GSK_UNSUPPORTED_PLATFORM	在运行应用程序的平台上不支持所请求函数。例如，在除了 Windows 2000 之外的平台上不支持 Microsoft Crypto API。
0x000001b5	437	GSK_ERROR_INCORRECT_ SESSION_TYPE	从重置会话类型回调函数返回了不正确的值。 仅允许 GSKit GSK_SERVER_SESSION 或 GSK_SERVER_SESSION_ WITH_CL_AUTH。
0x000001f5	501	GSK_INVALID_BUFFER_SIZE	缓冲区大小为负数或零。
0x000001f6	502	GSK_WOULD_BLOCK	与非分块的 I/O 配合使用。

表 2. GSKit 常规返回码 (续)

返回码（十六进制）	返回码（十进制）	常量	说明
0x00000259	601	GSK_ERROR_NOT_SSLV3	SSLV3 对于 reset_cipher 是必需的，但连接使用的是 SSLV2。
0x0000025a	602	GSK_MISC_INVALID_ID	为 gsk_secure_soc_misc 函数调用指定了无效标识。
0x000002bd	701	GSK_ATTRIBUTE_INVALID_ID	函数调用具有无效标识。导致此错误的原因还可能是在应该使用 SSL 连接的句柄时指定了环境句柄。
0x000002be	702	GSK_ATTRIBUTE_INVALID_LENGTH	属性的长度为负数，这无效。
0x000002bf	703	GSK_ATTRIBUTE_INVALID_ ENUMERATION	枚举值对于指定的枚举类型无效。
0x000002c0	704	GSK_ATTRIBUTE_INVALID_ SID_CACHE	用于替换“会话标识”（SID）高速缓存例程的参数列表无效。
0x000002c1	705	GSK_ATTRIBUTE_INVALID_ NUMERIC_VALUE	当设置数字属性时，所指定值对于要设置的特定属性无效。
0x000002c2	706	GSK_CONFLICTING_VALIDATION_ SETTING	为其他证书验证设置的参数存在冲突。
0x000002c3	707	GSK_AES_UNSUPPORTED	指定的密码包括了在执行系统上不受支持的 AES 密码。
0x000002c4	708	GSK_PEERID_LENGTH_ERROR	对等标识的长度不正确。它必须小于或等于 16 个字节。
0x000002c5	709	GSK_CIPHER_INVALID_WHEN_ FIPS_MODE_OFF	当 FIPS 方式处于关闭状态时，不允许使用给定密码。
0x000002c6	710	GSK_CIPHER_INVALID_WHEN_ FIPS_MODE_ON	在 FIPS 方式下未选择任何由 FIPS 核准的密码。
0x00000641	1601	GSK_TRACE_STARTED	跟踪已成功启动。
0x00000642	1602	GSK_TRACE_STOPPED	跟踪已成功停止。
0x00000643	1603	GSK_TRACE_NOT_STARTED	因为先前未启动任何跟踪文件，所以无法使其停止。
0x00000644	1604	GSK_TRACE_ALREADY_STARTED	因为跟踪文件已启动，所以无法使其再次启动。
0x00000645	1605	GSK_TRACE_OPEN_FAILED	无法打开跟踪文件。gsk_start_trace() 的第一个参数必须为有效完整路径文件名。

密钥管理返回码

表 3. 密钥管理返回码

返回码（十六进制）	返回码（十进制）	常量
0x00000000	0	GSKKM_ERR_OK
0x00000000	0	GSKKM_ERR_SUCCESS
0x00000001	1	GSKKM_ERR_UNKNOWN
0x00000002	2	GSKKM_ERR_ASN

表 3. 密钥管理返回码 (续)

返回码（十六进制）	返回码（十进制）	常量
0x00000003	3	GSKKM_ERR_ASN_INITIALIZATION
0x00000004	4	GSKKM_ERR_ASN_PARAMETER
0x00000005	5	GSKKM_ERR_DATABASE
0x00000006	6	GSKKM_ERR_DATABASE_OPEN
0x00000007	7	GSKKM_ERR_DATABASE_RE_OPEN
0x00000008	8	GSKKM_ERR_DATABASE_CREATE
0x00000009	9	GSKKM_ERR_DATABASE_ALREADY_EXISTS
0x0000000a	10	GSKKM_ERR_DATABASE_DELETE
0x0000000b	11	GSKKM_ERR_DATABASE_NOT_OPENED
0x0000000c	12	GSKKM_ERR_DATABASE_READ
0x0000000d	13	GSKKM_ERR_DATABASE_WRITE
0x0000000e	14	GSKKM_ERR_DATABASE_VALIDATION
0x0000000f	15	GSKKM_ERR_DATABASE_INVALID_VERSION
0x00000010	16	GSKKM_ERR_DATABASE_INVALID_PASSWORD
0x00000011	17	GSKKM_ERR_DATABASE_INVALID_FILE_TYPE
0x00000012	18	GSKKM_ERR_DATABASE_CORRUPTION
0x00000013	19	GSKKM_ERR_DATABASE_PASSWORD_ CORRUPTION
0x00000014	20	GSKKM_ERR_DATABASE_KEY_INTEGRITY
0x00000015	21	GSKKM_ERR_DATABASE_DUPLICATE_KEY
0x00000016	22	GSKKM_ERR_DATABASE_DUPLICATE_ KEY_RECORD_ID
0x00000017	23	GSKKM_ERR_DATABASE_DUPLICATE_ KEY_LABEL
0x00000018	24	GSKKM_ERR_DATABASE_DUPLICATE_ KEY_SIGNATURE
0x00000019	25	GSKKM_ERR_DATABASE_DUPLICATE_ KEY_UNSIGNED_CERTIFICATE
0x0000001a	26	GSKKM_ERR_DATABASE_DUPLICATE_KEY_ ISSUER_AND_SERIAL_NUMBER
0x0000001b	27	GSKKM_ERR_DATABASE_DUPLICATE_KEY_ SUBJECT_PUBLIC_KEY_INFO
0x0000001c	28	GSKKM_ERR_DATABASE_DUPLICATE_KEY_ UNSIGNED_CRL
0x0000001d	29	GSKKM_ERR_DATABASE_DUPLICATE_LABEL
0x0000001e	30	GSKKM_ERR_DATABASE_PASSWORD_ ENCRYPTION
0x0000001f	31	GSKKM_ERR_DATABASE_LDAP
0x00000020	32	GSKKM_ERR_CRYPTO
0x00000021	33	GSKKM_ERR_CRYPTO_ENGINE
0x00000022	34	GSKKM_ERR_CRYPTO_ALGORITHM

表 3. 密钥管理返回码 (续)

返回码（十六进制）	返回码（十进制）	常量
0x00000023	35	GSKKM_ERR_CRYPTO_SIGN
0x00000024	36	GSKKM_ERR_CRYPTO_VERIFY
0x00000025	37	GSKKM_ERR_CRYPTO_DIGEST
0x00000026	38	GSKKM_ERR_CRYPTO_PARAMETER
0x00000027	39	GSKKM_ERR_CRYPTO_UNSUPPORTED_ ALGORITHM
0x00000028	40	GSKKM_ERR_CRYPTO_INPUT_GREATER_ THAN_MODULUS
0x00000029	41	GSKKM_ERR_CRYPTO_UNSUPPORTED_ MODULUS_SIZE
0x0000002a	42	GSKKM_ERR_VALIDATION
0x0000002b	43	GSKKM_ERR_VALIDATION_KEY
0x0000002c	44	GSKKM_ERR_VALIDATION_DUPLICATE_ EXTENSIONS
0x0000002d	45	GSKKM_ERR_VALIDATION_KEY_WRONG_ VERSION
0x0000002e	46	GSKKM_ERR_VALIDATION_KEY_ EXTENSIONS_REQUIRED
0x0000002f	47	GSKKM_ERR_VALIDATION_KEY_VALIDITY
0x00000030	48	GSKKM_ERR_VALIDATION_KEY_VALIDITY_ PERIOD
0x00000031	49	GSKKM_ERR_VALIDATION_KEY_VALIDITY_ PRIVATE_KEY_USAGE
0x00000032	50	GSKKM_ERR_VALIDATION_KEY_ISSUER_ NOT_FOUND
0x00000033	51	GSKKM_ERR_VALIDATION_KEY_MISSING_ REQUIRED_EXTENSIONS
0x00000034	52	GSKKM_ERR_VALIDATION_KEY_BASIC_ CONSTRAINTS
0x00000035	53	GSKKM_ERR_VALIDATION_KEY_SIGNATURE
0x00000036	54	GSKKM_ERR_VALIDATION_KEY_ROOT_KEY_ NOT_TRUSTED
0x00000037	55	GSKKM_ERR_VALIDATION_KEY_IS_REVOKED
0x00000038	56	GSKKM_ERR_VALIDATION_KEY_AUTHORITY_ KEY_IDENTIFIER
0x00000039	57	GSKKM_ERR_VALIDATION_KEY_PRIVATE_KEY_ USAGE_PERIOD
0x0000003a	58	GSKKM_ERR_VALIDATION_SUBJECT_ ALTERNATIVE_NAME
0x0000003b	59	GSKKM_ERR_VALIDATION_ISSUER_ ALTERNATIVE_NAME
0x0000003c	60	GSKKM_ERR_VALIDATION_KEY_USAGE

表 3. 密钥管理返回码 (续)

返回码（十六进制）	返回码（十进制）	常量
0x0000003d	61	GSKKM_ERR_VALIDATION_KEY_ UNKNOWN_CRITICAL_EXTENSION
0x0000003e	62	GSKKM_ERR_VALIDATION_KEY_PAIR
0x0000003f	63	GSKKM_ERR_VALIDATION_CRL
0x00000040	64	GSKKM_ERR_MUTEX
0x00000041	65	GSKKM_ERR_PARAMETER
0x00000042	66	GSKKM_ERR_NULL_PARAMETER
0x00000043	67	GSKKM_ERR_NUMBER_SIZE
0x00000044	68	GSKKM_ERR_OLD_PASSWORD
0x00000045	69	GSKKM_ERR_NEW_PASSWORD
0x00000046	70	GSKKM_ERR_PASSWORD_EXPIRATION_TIME
0x00000047	71	GSKKM_ERR_THREAD
0x00000048	72	GSKKM_ERR_THREAD_CREATE
0x00000049	73	GSKKM_ERR_THREAD_WAIT_FOR_EXIT
0x0000004a	74	GSKKM_ERR_IO
0x0000004b	75	GSKKM_ERR_LOAD
0x0000004c	76	GSKKM_ERR_PKCS11
0x0000004d	77	GSKKM_ERR_NOT_INITIALIZED
0x0000004e	78	GSKKM_ERR_DB_TABLE_CORRUPTED
0x0000004f	79	GSKKM_ERR_MEMORY_ALLOCATE
0x00000050	80	GSKKM_ERR_UNSUPPORTED_OPTION
0x00000051	81	GSKKM_ERR_GET_TIME
0x00000052	82	GSKKM_ERR_CREATE_MUTEX
0x00000053	83	GSKKM_ERR_CMDCAT_OPEN
0x00000054	84	GSKKM_ERR_ERRCAT_OPEN
0x00000055	85	GSKKM_ERR_FILENAME_NULL
0x00000056	86	GSKKM_ERR_FILE_OPEN
0x00000057	87	GSKKM_ERR_FILE_OPEN_TO_READ
0x00000058	88	GSKKM_ERR_FILE_OPEN_TO_WRITE
0x00000059	89	GSKKM_ERR_FILE_OPEN_NOT_EXIST
0x0000005a	90	GSKKM_ERR_FILE_OPEN_NOT_ALLOWED
0x0000005b	91	GSKKM_ERR_FILE_WRITE
0x0000005c	92	GSKKM_ERR_FILE_REMOVE
0x0000005d	93	GSKKM_ERR_BASE64_INVALID_DATA
0x0000005e	94	GSKKM_ERR_BASE64_INVALID_MSGTYPE
0x0000005f	95	GSKKM_ERR_BASE64_ENCODING
0x00000060	96	GSKKM_ERR_BASE64_DECODING
0x00000061	97	GSKKM_ERR_DN_TAG_NULL
0x00000062	98	GSKKM_ERR_DN_CN_NULL

表 3. 密钥管理返回码 (续)

返回码（十六进制）	返回码（十进制）	常量
0x00000063	99	GSKKM_ERR_DN_C_NULL
0x00000064	100	GSKKM_ERR_INVALID_DB_HANDLE
0x00000065	101	GSKKM_ERR_KEYDB_NOT_EXIST
0x00000066	102	GSKKM_ERR_KEYPAIRDB_NOT_EXIST
0x00000067	103	GSKKM_ERR_PWDFILE_NOT_EXIST
0x00000068	104	GSKKM_ERR_PASSWORD_CHANGE_MATCH
0x00000069	105	GSKKM_ERR_KEYDB_NULL
0x0000006a	106	GSKKM_ERR_REQKEYDB_NULL
0x0000006b	107	GSKKM_ERR_KEYDB_TRUSTCA_NULL
0x0000006c	108	GSKKM_ERR_REQKEY_FOR_CERT_NULL
0x0000006d	109	GSKKM_ERR_KEYDB_PRIVATE_KEY_NULL
0x0000006e	110	GSKKM_ERR_KEYDB_DEFAULT_KEY_NULL
0x0000006f	111	GSKKM_ERR_KEYREC_PRIVATE_KEY_NULL
0x00000070	112	GSKKM_ERR_KEYREC_CERTIFICATE_NULL
0x00000071	113	GSKKM_ERR_CRLS_NULL
0x00000072	114	GSKKM_ERR_INVALID_KEYDB_NAME
0x00000073	115	GSKKM_ERR_UNDEFINED_KEY_TYPE
0x00000074	116	GSKKM_ERR_INVALID_DN_INPUT
0x00000075	117	GSKKM_ERR_KEY_GET_BY_LABEL
0x00000076	118	GSKKM_ERR_LABEL_LIST_CORRUPT
0x00000077	119	GSKKM_ERR_INVALID_PKCS12_DATA
0x00000078	120	GSKKM_ERR_PKCS12_PWD_CORRUPTION
0x00000079	121	GSKKM_ERR_EXPORT_TYPE
0x0000007a	122	GSKKM_ERR_PBE_ALG_UNSUPPORT
0x0000007b	123	GSKKM_ERR_KYR2KDB
0x0000007c	124	GSKKM_ERR_KDB2KYR
0x0000007d	125	GSKKM_ERR_ISSUING_CERTIFICATE
0x0000007e	126	GSKKM_ERR_FIND_ISSUER_CHAIN
0x0000007f	127	GSKKM_ERR_WEBDB_DATA_BAD_FORMAT
0x00000080	128	GSKKM_ERR_WEBDB_NOTHING_TO_WRITE
0x00000081	129	GSKKM_ERR_EXPIRE_DAYS_TOO_LARGE
0x00000082	130	GSKKM_ERR_PWD_TOO_SHORT
0x00000083	131	GSKKM_ERR_PWD_NO_NUMBER
0x00000084	132	GSKKM_ERR_PWD_NO_CONTROL_KEY
0x00000085	133	GSKKM_ERR_SIGNATURE_ALGORITHM
0x00000086	134	GSKKM_ERR_INVALID_DATABASE_TYPE
0x00000087	135	GSKKM_ERR_SECONDARY_KEYDB_TO_OTHER
0x00000088	136	GSKKM_ERR_NO_SECONDARY_KEYDB

表 3. 密钥管理返回码 (续)

返回码（十六进制）	返回码（十进制）	常量
0x00000089	137	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_LABEL_NOT_EXIST
0x0000008a	138	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_PASSWORD_REQUIRED
0x0000008b	139	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_PASSWORD_NOT_REQUIRED
0x0000008c	140	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_LIBRARY_NOT_LOADED
0x0000008d	141	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_NOT_SUPPORT
0x0000008e	142	GSKKM_ERR_CRYPTOGRAPHIC_TOKEN_FUNCTION_FAILED
0x0000008f	143	GSKKM_ERR_LDAP_USER_NOT_FOUND
0x00000090	144	GSKKM_ERR_LDAP_INVALID_PASSWORD
0x00000091	145	GSKKM_ERR_LDAP_QUERY_ENTRY_FAILED
0x00000092	146	GSKKM_ERR_INVALID_CERT_CHAIN
0x00000093	147	GSKKM_ERR_CERT_ROOT_NOT_TRUSTED
0x00000094	148	GSKKM_ERR_CERT_REVOKED
0x00000095	149	GSKKM_ERR_CRYPTOGRAPHIC_OBJECT_FUNCTION_FAILED
0x00000096	150	GSKKM_ERR_NO_AVAILABLE_CRL_DATASOURCE
0x00000097	151	GSKKM_ERR_NO_TOKEN_PRESENT
0x00000098	152	GSKKM_ERR_FIPS_NOT_SUPPORTED
0x00000099	153	GSKKM_ERR_FIPS_CONFLICT_SETTING
0x0000009a	154	GSKKM_ERR_PASSWORD_STRENGTH_FAILED

用于加密静态数据的 IBM Database Encryption Expert

IBM Database Encryption Expert 是综合软件数据安全性解决方案，当与本机 DB2 安全性配合使用时，它会针对大量威胁高效地保护数据和数据库应用程序。

Database Encryption Expert 有助于组织确保在符合条例和立法机构法令的同时对专用和机密数据进行强保护。Database Encryption Expert 的主要优点如下:

- 对于 DB2 数据库系统，具有成熟的高数据安全性
- 保护实时文件、配置文件、日志文件和备份数据
- 对应用程序、数据库和存储环境透明
- 用于在联机环境和脱机环境中保护数据的策略和密钥管理统一
- 满足性能要求

Database Encryption Expert 使您能够对脱机数据库备份进行加密以及对联机（“实时”）数据库文件进行加密。这是磁盘上数据的加密，相对通过网络传输的“动态数据”而言，这类数据有时称为“静态数据”。

- 对于备份，数据的加密方式与它备份时的相同，因此，备份设备上的数据已加密。要是该数据需要恢复，恢复服务器就会识别该数据已加密并将对其进行解密。
- 对于数据库文件，包含 DB2 数据库中数据的操作系统数据文件已加密。这会防止尝试读取“原始”数据库文件的未授权用户对这些数据文件进行访问。

Database Encryption Expert 对用户、数据库、应用程序和存储器透明。不需要对现有基础结构进行任何代码更改或其他更改。Database Encryption Expert 可以保护任何存储环境中的数据，而用户继续以先前的方式对数据进行访问。

Database Encryption Expert 可以保护数据库应用程序，因为它可以防止对可执行文件、配置文件以及库之类的对象进行更改，从而防止对应用程序的攻击。

注: 不能在 DB2 pureScale®环境中使用 Database Encryption Expert。

Database Encryption Expert 的体系结构

Database Encryption Expert 是一组代理程序和服务器软件包，通过使用基于 Web 的用户界面和命令行实用程序来管理。Database Encryption Expert 管理员配置用于控制如何实现安全性和加密的安全策略。

根据定义这些安全策略的方式，Database Encryption Expert 备份代理程序会对 DB2 备份进行加密，而 Database Encryption Expert 文件系统代理程序则对 DB2 数据文件进行加密。

Encryption Expert Security Server 会存储安全策略、加密密钥和事件日志文件。安全策略包含数组安全规则，必须满足这些规则才能允许或拒绝访问。每条安全规则都对访问受保护数据的用户、内容、时间和方式进行评估，如果这些条件匹配，那么 Security Server 会允许或拒绝访问。

第 79 页的图 4 说明了 Database Encryption Expert 的体系结构。

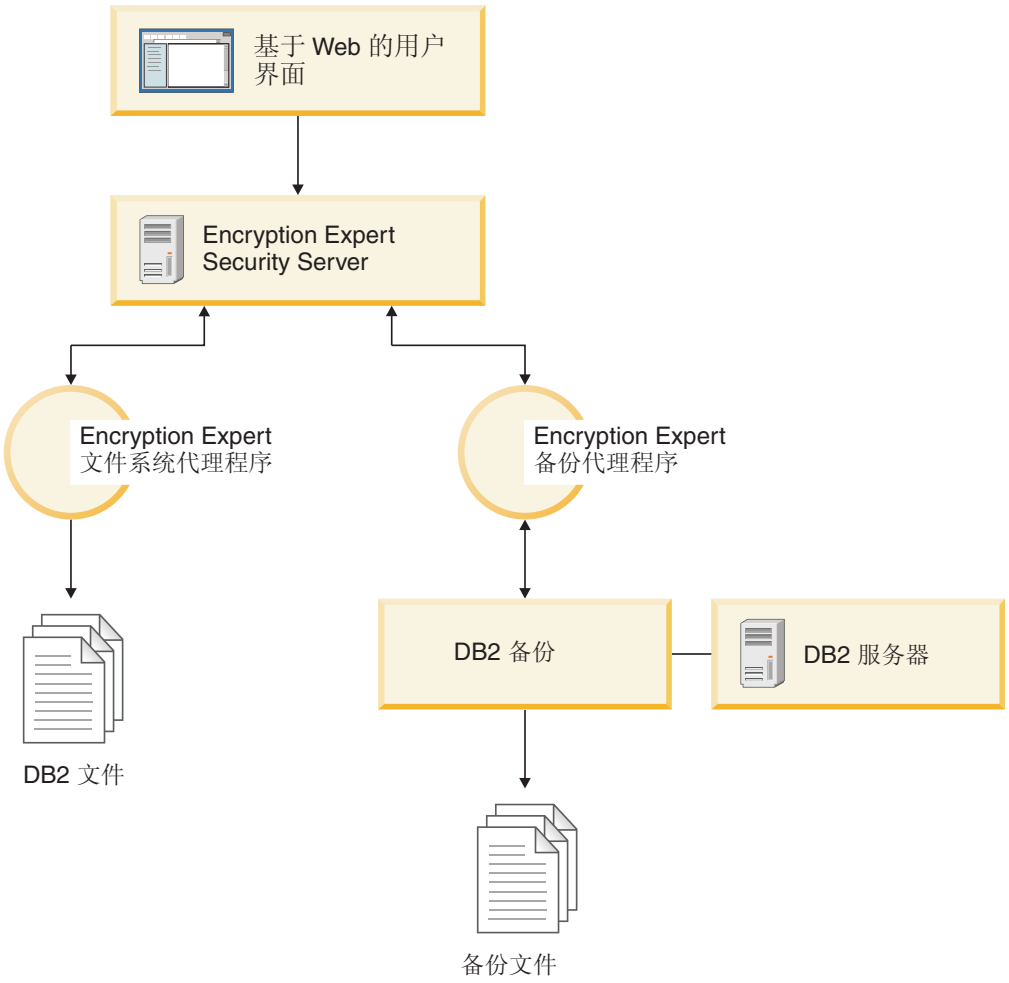


图 4. Database Encryption Expert 的体系结构

文件系统代理程序

Database Encryption Expert 文件系统代理程序进程始终在后台运行。代理程序会对任何访问所保护数据文件、目录或可执行文件的尝试进行拦截。Database Encryption Expert 文件系统代理程序会将访问尝试转发至 Security Server，根据所应用策略，Security Server 会允许或拒绝所尝试的访问。

Database Encryption Expert 保护不只是允许或拒绝对文件进行访问，您还可对文件进行加密。仅会加密文件内容，文件元数据保持原样。因此，仅为了查看已加密文件的名称、时间戳记以及文件类型等，不必对其进行解密。这允许数据管理应用程序执行其函数，而不必显示文件内容。例如，备份管理器可以在无法查看内容的情况下对特定数据进行备份。

如果已加密文件由未授权用户访问，那么在缺少相应 Security Server 核准和加密密钥的情况下，其内容毫无价值。但是，具有正确策略和许可权的用户不会意识到正在进行加密和解密。

备份代理程序

通常由 DB2 备份 API 系统执行的所有数据库备份函数都受 Database Encryption Expert 服务器支持，其中包括本机数据库压缩。除另有一个命令行自变量之外，DB2 备份操作员不会意识到 Database Encryption Expert 干预。Database Encryption Expert 会备份和复原静态数据以及活动的联机数据。

基本的备份和复原配置受支持。在基本配置中，数据通过一个服务器和多个代理程序进行加密和备份；数据的解密和复原是通过对先前用来生成备份的服务器配置的代理程序来完成的。

对于备份和复原，单站点和多站点配置也受支持。在单站点方案中，配置数据通过单个数据中心中多个 Security Server 进行镜像。在多站点方案中，备份是在不同数据中心中不同 Encryption Expert 服务器上复原的。

审计日志记录

通过集中式审计记录工具，会紧密监视和记录 Database Encryption Expert 代理程序活动。可以记录所有可审计事件，其中包括备份、复原和安全性管理操作。这包括 Database Encryption Expert 系统事件，例如，初始化、关闭和重新启动，以及不同 Database Encryption Expert 组件之间网络的连接与断开连接。

Database Encryption Expert 文档

有关 Database Encryption Expert 的更多信息，请转至以下 Web 页面：http://publib.boulder.ibm.com/infocenter/mptoolic/v1r0/topic/com.ibm.db2tools.eet.doc.ug/eetwelcome.htm。

使用 AIX 加密文件系统 (EFS) 来进行数据库加密

对于 AIX 操作系统上运行的 DB2 Enterprise Server Edition，可以选择通过使用 AIX 加密文件系统 (EFS) 来设置加密数据库。有关 EFS 的详细信息，请参阅 AIX 文档。

注： 在分区数据库环境中工作时，如果要使用 EFS，那么数据库应该位于单个数据库分区中。

通过将底层的 EFS 与 JFS2 文件系统配合使用，可以对包含数据库表中数据的操作系统文件进行加密。

设置加密的步骤如下所示：

1. 在系统上启用 EFS。
2. 为用来运行 DB2 数据库守护程序的用户帐户装入密钥库。
3. 在数据库文件系统上启用 EFS。
4. 确定要加密的操作系统文件。
5. 对包含需要 EFS 保护的数据库表的文件进行加密。

在系统上启用 EFS

在启用 EFS 之前，必须安装 `clic.rte` 文件集。可在扩展包 CD 上找到 `clic.rte` 安装映像。

以 root 用户身份运行下面的命令来在系统上启用 EFS:

```
% efsenable -a
```

仅需要运行 **efsenable** 命令一次。

装入密钥库

在下列配置示例中，用来运行 DB2 数据库守护程序的用户帐户称为 abst。用户 abst 必须具有密钥库，并且 abst 所属的任何组也必须具有密钥库。

1. 在启动 DB2 守护程序之前，所有密钥库都必须与 abst 进程相关联。

   可以通过使用 **efskeymgr -V** 命令来验证它们是否相关联，如以下示例中所示:

   ```
   # lsuser abst
   abst id=203 pgrp=abstgp groups=abstgp,staff ...

   # efskeymgr -V
   List of keys loaded in the current process:
     Key #0:
                       Kind ..................... User key
                       Id (uid / gid) ......... 203
                       Type ..................... Private
   key
                       Algorithm ................ RSA_1024
                       Validity ................. Key is
   valid
                       Fingerprint ..............
   24c88df2:d91cb6a2:c3e11b6a:4c13f8b4:666fabd8

     Key #1:
                       Kind ..................... Group
   key
                       Id (uid / gid) ......... 1
                       Type ..................... Private
   key
                       Algorithm ................ RSA_1024
                       Validity ................. Key is
   valid
                       Fingerprint ..............
   03fead42:57e7646e:a1715626:cfa56c8e:8abed1c1

     Key #2:
                       Kind ..................... Group
   key
                       Id (uid / gid) ......... 212
                       Type ..................... Private
   key
                       Algorithm ................ RSA_1024
                       Validity ................. Key is
   valid
                       Fingerprint ..............
   339dfb19:bc850f4c:5551c975:7fe4961b:2dddf3bc
   ```

2. 如果没有任何密钥库显示为与 abst 进程相关联，那么尝试使用以下命令来装入密钥库: `% efskeymgr -o ksh`

   此命令会提示用户提供密钥库密码，该密码最初设置为登录密码。

3. 通过重新运行以下命令来确认用户和组密钥是否已装入: `% efskeymgr -V`

   应该会列示用户和组密钥。如果仍然没有列示组密钥库，请继续执行步骤 4。

4. 根据创建组的方式的不同，组密钥库可能不存在。如果 **efskeymgr -V** 命令没有列示用户的组密钥库，那么必须创建组密钥库。

   请以 root 用户或 RBAC 角色 aix.efs_admin 身份创建组密钥库:

   ```
   % efskeymgr -C group_name
   ```

5. 将组密钥库访问权指定给每个合适的用户:

   ```
   % efskeymgr -k group /group_name -s user/user_name
   ```

   如果用户已登录，那么他们将不会立即对组密钥库具有访问权，他们应该使用 **efskeymgr -o ksh** 命令来重新装入其密钥库，或者重新登录。

在数据库文件系统上启用 EFS

EFS 仅在 JFS2 文件系统上运行，且必须专门启用。

如果数据库位于现有文件系统上，请运行 `% chfs -a efs=yes filesystem` 命令来启用 EFS，例如:

```
% chfs -a efs=yes /test01
```

如果在创建新的文件系统，那么可以通过将 **smit** 命令或 **crfs** 命令与 `-a efs=yes` 选项配合使用来启用 EFS。例如:

```
% crfs -v jfs2 -a efs=yes -m mount_point -d devide -A yes
```

现在，EFS 在文件系统上已启用但未打开。请仅为需要加密数据的特定数据库表打开 EFS。有关更多信息，请参阅关于 **efsmgr** 命令和继承的 AIX EFS 文档。

确定要加密的文件

为了确定包含要使用 EFS 加密来保护的特定数据库表的文件，请遵循将 EMPLOYEE 表用作示例的这些步骤。

1. 使用类似于以下示例的查询来查找表的 TBSPACEID:

   ```
   SELECT TABNAME, TBSPACEID FROM syscat.tables WHERE tabname='EMPLOYEE'
   ```

   假定此查询的结果如下所示:

TABNAME	TBSPACEID
EMPLOYEE	2

2. 使用类似于下面的示例的查询来在表空间中查找该 TBSPACEID:

   ```
   LIST TABLESPACE CONTAINERS FOR 2
   ```

   假定此查询的结果如下所示:

容器标识	名称	类型
0	/test01/abst/NODE0000/BAR/T0000002/C0000000.LRG	文件

   现在，您知道此表空间包含在称为 `/test01/abst/NODE0000/BAR/T0000002/C0000000.LRG` 的操作系统文件中。这是需要加密的文件。

加密文件

首先，请按对数据或数据库进行任何重大更改之前的操作来备份数据库。

遵循下列步骤以加密文件:

1. 列示文件，例如:

   ```
   # ls -U /test01/abst/NODE0000/BAR/T0000002/C0000000.LRG

   -rw-------- 1 abst abstgp 33554432 Jul 30 18:01
   /test01/abst/NODE0000/BAR/T0000002/C0000000.LRG
   ```

2. 使用 **efsmgr** 命令来对文件进行加密，例如:

   ```
   # efsmgr -e /test01/abst/NODE0000/BAR/T0000002/C0000000.LRG
   ```

   如果再次列示该文件，那么许可权字符串末尾将出现“e”，它指出该文件已加密。例如:

   ```
   # ls -U /test01/abst/NODE0000/BAR/T0000002/C0000000.LRG

   -rw-------e 1 abst abstgp 33554432 Jul 30 18:03
   /test01/abst/NODE0000/BAR/T0000002/C0000000.LRG
   ```

3. 按正常方式启动并使用 DB2 数据库管理器。在底层文件系统中，添加至 EMPLOYEE 表和此加密表空间的所有数据都将由 EFS 进行加密。每当检索到数据时，都将通过 DB2 数据库管理器按正常方式解密并显示该数据。

审计 DB2 活动

DB2 审计设施简介

要管理对敏感数据的访问，可以使用各种认证和访问控制机制来对可接受的数据访问建立规则和控制。但要防止未知或不可接受的行为以及要发现这类行为，可通过使用 DB2 审计设施来监视数据访问。

成功监视不需要的数据访问和后续分析，可改善对数据访问的控制，并最终防止对数据的恶意的未授权访问或因为粗心而导致的未授权访问。监视应用程序和单独的用户访问（包括系统管理操作）可提供有关数据库系统活动的历史记录。

DB2 审计设施为一系列预定义数据库事件生成审计跟踪，并允许您对其进行维护。将此设施生成的记录保存在审计日志文件中。对这些记录的分析可揭示会识别系统误用的使用模式。一旦识别，就可执行操作来减少或消除这类系统误用。

审计设施提供在实例级别和单个数据库级别进行审计的功能，并对每个活动使用不同日志独立地记录所有实例和数据库级别活动。系统管理员（拥有 SYSADM 权限）可以使用 **db2audit** 工具在实例级别上配置审计以及控制收集这种审计信息的时间。系统管理员可以使用 **db2audit** 工具来归档实例和数据库审计日志以及从任一类型的已归档日志中抽取审计数据。

安全性管理员（在数据库内拥有 SECADM 权限）可将审计策略与 SQL 语句 AUDIT 配合使用，以配置并控制单个数据库的审计要求。安全性管理员可以使用下列审计例程来执行指定的任务:

- SYSPROC.AUDIT_ARCHIVE 存储过程对审计日志进行归档。
- SYSPROC.AUDIT_LIST_LOGS 表函数允许您查找关注的日志。

- SYSPROC.AUDIT_DELIM_EXTRACT 存储过程将数据抽取到定界文件中，以便进行分析。

安全性管理员可以将对这些例程的 EXECUTE 特权授予另一个用户，因此在需要时使安全性管理员能够委派这些任务。

当在分区数据库环境中工作时，许多可审计的事件将在与用户连接的数据库分区（协调程序分区）或目录分区（若它们不是相同的数据库分区）中发生。这意味着审计记录可由多个数据库分区生成。每个审计记录的一部分包含用于标识协调程序分区和始发分区（产生审计记录的分区）的信息。

在实例级别，必须使用 **db2audit start** 和 **db2audit stop** 命令来显式停止和启动审计设施。在启动实例级别审计时，审计设施使用现有审计配置信息。因为审计设施与 DB2 数据库服务器无关，所以即使停止该实例，审计设施将仍然是活动的。事实上，当停止该实例时，可在审计日志中生成审计记录。为了在数据库级别上启动审计，首先需要创建审计策略，然后使此审计策略与要监视的对象（如授权标识、数据库权限、可信上下文或特定表）相关联。

审计记录的类别

可生成不同类别的审计记录。在下面关于可用于审计的事件类别描述中，您应注意每个类别的名称后面是一个单词的关键字，它用来标识该类别的类型。可用于审计的事件类别是:

- 审计 (AUDIT)。当更改审计设置或访问审计日志时会生成记录。
- 权限检查 (CHECKING)。对访问或操纵 DB2 数据库对象或函数的尝试进行权限检查期间会生成记录。
- 对象维护 (OBJMAINT)。当创建或删除数据对象以及改变某些对象时会生成记录。
- 安全维护 (SECMAINT)。在下列情况下会生成记录:
  - 授予或撤销对象特权或数据库权限
  - 授予或撤销安全标号或免除权
  - 改变组权限、角色权限或者覆盖或限制 LBAC 安全策略的属性
  - 授予或撤销 SETSESSIONUSER 特权
  - 修改下列任一配置参数: SYSADM_GROUP、SYSCTRL_GROUP、SYSMAINT_GROUP 或 SYSMON_GROUP。
- 系统管理 (SYSADMIN)。当执行需要 SYSADM、SYSMAINT 或 SYSCTRL 权限的操作时会生成记录。
- 用户验证 (VALIDATE)。当认证用户或检索系统安全性信息时会生成记录。
- 操作上下文 (CONTEXT)。当执行数据库操作时，生成记录以显示该操作上下文。此类别允许对审计日志文件进行更好的解释。当与该日志的事件相关因子字段配合使用时，可将一组事件重新与单个数据库操作关联。例如，动态查询的查询语句、静态查询的程序包标识或可执行的操作类型的指示符（如 CONNECT）均可提供分析审计结果时所需的上下文。

  注: 提供该操作上下文的 SQL 或 XQuery 语句可能很长，并可在 CONTEXT 记录内完全显示。这可能使 CONTEXT 记录变得很大。
- 执行 (EXECUTE)。在执行 SQL 语句期间生成记录。

对于先前列示的任何类别，您可以审计失败的操作和/或成功的操作。

在数据库服务器上执行的任何操作可能生成几个记录。审计日志中生成的实际记录数目取决于审计设施配置所指定的要记录的事件类别数。它还取决于是审计成功的操作，还是失败的操作，或二者兼有。由于此原因，对要审计的事件的选择十分重要。

审计策略

安全性管理员可以使用审计策略来配置审计设施，以仅收集关于所需数据和对象的信息。

安全性管理员可以创建审计策略来控制在单个数据库内审计的内容。下列对象可以具有关联的审计策略:

- 整个数据库

  根据审计策略，将审计该数据库内发生的所有可审计事件。

- 表

  审计所有数据操作语言 (DML) 以及对表（无类型）、MQT（具体化查询表）或昵称的 XQUERY 访问。在访问表时，只生成包含或不包含数据的 EXECUTE 类别审计事件，即使策略指示应审计其他类别亦如此。

- 可信上下文

  根据审计策略，将审计由特定可信上下文定义的可信连接内发生的所有可审计事件。

- 表示用户、组或角色的授权标识

  根据审计策略，将审计由指定用户启动的所有可审计事件。

  根据审计策略，将审计作为组或角色成员的用户所启动的所有可审计事件。还包括间接角色成员资格，例如，通过其他角色或组。

  通过为一个组定义工作负载并捕获活动详细信息，可以使用“工作负载管理”事件监视器来捕获类似数据。您应该了解，至工作负载的映射不仅仅涉及授权标识，还可能涉及其他属性，这可能会使您无法获得需要的审计粒度，或者在修改了这些其他属性时，连接可能会映射至不同（可能是未监视的）工作负载。审计解决方案保证审计用户、组或角色。

- 权限（SYSADM、SECADM、DBADM、SQLADM、WLMADM、ACCESSCTRL、DATAACCESS、SYSCTRL、SYSMAINT 和 SYSMON）

  根据审计策略，将审计由拥有指定权限的用户启动的所有可审计事件，即使该权限对事件来说不是必需的，情况亦如此。

安全性管理员可以创建多个审计策略。例如，您所在的公司可能需要一个策略来审计敏感数据，并需要一个策略来审计拥有 DBADM 权限的用户的活动。如果多个审计策略对一个语句有效，那么将审计每个审计策略要求审计的所有事件（但只审计一次）。例如，如果数据库的审计策略要求审计特定表的成功 EXECUTE 事件，并且用户的审计策略要求审计同一个表的失败 EXECUTE 事件，那么将审计访问该表时的成功和失败尝试。

对于特定对象，只能有一个审计策略有效。例如，不能同时有多个审计策略与同一个表关联。

审计策略不能与视图或类型表关联。根据基础表的策略，将审计访问具有关联审计策略的表的视图。

适用于表的审计策略不会自动适用于基于该表的 MQT。如果将审计策略与一个表关联，那么应将相同策略与基于该表的任何 MQT 关联。

事务期间执行的审计是根据事务开始时的审计策略及其关联进行的。例如，如果安全性管理员将一个审计策略与用户关联并且该用户当时处于事务中，那么该审计策略不会影响在该事务内执行的任何其余语句。此外，对审计策略所作的更改要在落实后才生效。如果安全性管理员发出 ALTER AUDIT POLICY 语句，那么该语句要在落实后才生效。

安全性管理员使用 CREATE AUDIT POLICY 语句来创建审计策略，并使用 ALTER AUDIT POLICY 语句来修改审计策略。这些语句可以指定:

- 要审计的事件的状态值: 无、成功、失败或两者。

  仅审计与指定状态值相匹配的可审计事件。
- 审计期间发生错误时的服务器行为。

安全性管理员使用 AUDIT 语句将审计策略与当前数据库或当前服务器中的数据库对象关联。只要该对象在使用中，就会根据此审计策略对它进行审计。

要删除审计策略，安全性管理员可以使用 DROP 语句。不能删除与任何对象关联的审计策略。使用 AUDIT REMOVE 语句除去与对象的任何其余关联。要将元数据添加至审计策略，安全性管理员可以使用 COMMENT 语句。

在建立完全连接之前生成的事件

对于在执行连接和切换用户操作期间生成的一些事件，唯一可用的审计策略信息是与数据库关联的策略。下表中显示了这些事件:

表 4. 连接事件

事件	审计类别	注释
CONNECT	CONTEXT	
CONNECT_RESET	CONTEXT	
AUTHENTICATION	VALIDATE	这包括在可信连接内连接和切换用户期间的认证。
CHECKING_FUNC	CHECKING	尝试的访问是 SWITCH_USER。

将只根据与数据库关联的审计策略审计这些事件，而不使用与任何其他对象（例如，用户、用户组或权限）关联的审计策略进行审计。对于在连接期间发生的 CONNECT 和 AUTHENTICATION 事件，将使用实例级别审计设置，直到数据库被激活为止。数据库在第一次连接期间或发出 ACTIVATE DATABASE 命令时被激活。

切换用户的影响

如果在可信连接内切换用户，那么不会留下原始用户的任何信息。在此情况下，将不再考虑与原始用户关联的审计策略，并且将针对新用户重新评估适用的审计策略。任何与可信连接关联的审计策略仍有效。

如果使用 SET SESSION USER 语句，那么只切换会话授权标识。原始用户的授权标识（系统授权标识）的审计策略保持有效，并且还使用新用户的审计策略。如果在会话内发出了多个 SET SESSION USER 语句，那么只考虑与原始用户（系统授权标识）和当前用户（会话授权标识）关联的审计策略。

数据定义语言限制

下列数据定义语言 (DDL) 语句称为 AUDIT 独占式 SQL 语句:

- AUDIT
- CREATE AUDIT POLICY、ALTER AUDIT POLICY 和 DROP AUDIT POLICY
- DROP ROLE 和 DROP TRUSTED CONTEXT（如果要删除的角色或可信上下文与审计策略关联）

AUDIT 独占式 SQL 语句在使用时具有一些限制:

- 每个语句后面必须跟有 COMMIT 或 ROLLBACK。
- 不能在全局事务内发出这些语句，例如，XA 事务。

所有分区中一次只允许有一个未落实的 AUDIT 独占式 DDL 语句。如果未落实的 AUDIT 独占式 DDL 语句正在执行，那么后续 AUDIT 独占式 DDL 语句将等待，直到当前 AUDIT 独占式 DDL 语句落实或回滚为止。

注: 更改将写入目录，但要在落实后才生效，即使对于发出该语句的连接亦如此。

审计对特定表的任何访问的示例

请考虑这样一家公司，其 EMPLOYEE 表包含非常敏感的信息，并且该公司希望审计对该表中的数据的任何和所有 SQL 访问。可以使用 EXECUTE 类别来跟踪对表的所有访问；该类别审计 SQL 语句，并可以选择审计在执行时为该语句提供的输入数据值。

跟踪该表上的活动需要执行两个步骤。首先，安全性管理员创建一个指定 EXECUTE 类别的审计策略，然后安全性管理员将该策略与表关联:

```
CREATE AUDIT POLICY SENSITIVEDATAPOLICY
    CATEGORIES EXECUTE STATUS BOTH ERROR TYPE AUDIT
COMMIT

AUDIT TABLE EMPLOYEE USING POLICY SENSITIVEDATAPOLICY
COMMIT
```

审计 SYSADM 或 DBADM 执行的任何操作的示例

为了完成安全合格证书，一家公司必须表明能够监视数据库内由拥有系统管理 (SYSADM) 或数据库管理 (DBADM) 权限的那些人执行的任何和所有活动。

要捕获数据库内的所有操作，应审计 EXECUTE 和 SYSADMIN 类别。安全性管理员创建一个审计这两种类别的审计策略。安全性管理员可以使用 AUDIT 语句将此审计策

略与 SYSADM 和 DBADM 权限关联。然后，拥有 SYSADM 或 DBADM 权限的任何用户将记录任何可审计事件。以下示例显示如何创建这种审计策略并将它与 SYSADM 和 DBADM 权限关联:

```
CREATE AUDIT POLICY ADMINSPOLICY CATEGORIES EXECUTE STATUS BOTH,
   SYSADMIN STATUS BOTH ERROR TYPE AUDIT
COMMIT
AUDIT SYSADM, DBADM USING POLICY ADMINSPOLICY
COMMIT
```

审计特定角色执行的任何访问的示例

一家公司允许对其企业数据库进行 Web 应用程序访问。使用 Web 应用程序的确切个人未知。只知道使用的角色，该角色用于管理数据库权限。该公司希望监视作为该角色成员的任何人的操作，以便检查他们提交给数据库的请求并确保他们只通过 Web 应用程序访问数据库。

EXECUTE 类别包含跟踪这种情况下的用户活动所需的审计级别。第一步是创建适当的审计策略并将它与 Web 应用程序所使用的角色关联（在本示例中，角色为 TELLER 和 CLERK）:

```
CREATE AUDIT POLICY WEBAPPPOLICY CATEGORIES EXECUTE WITH DATA
   STATUS BOTH ERROR TYPE AUDIT
COMMIT
AUDIT ROLE TELLER, ROLE CLERK USING POLICY WEBAPPPOLICY
COMMIT
```

对数据库启用审计的示例

某个公司想要确定谁在对名为 SAMPLE 的数据库进行 DDL 更改（示例: ALTER TABLE）。

```
CONNECT TO SAMPLE

CREATE AUDIT POLICY ALTPOLICY CATEGORIES AUDIT STATUS BOTH,
    OBJMAINT STATUS BOTH,  CHECKING STATUS BOTH,
    EXECUTE STATUS BOTH, ERROR TYPE NORMAL

AUDIT DATABASE USING POLICY ALTPOLICY
```

存储和分析审计日志

归档审计日志会将活动审计日志移至一个归档目录，而服务器开始写入新的活动审计日志。稍后，可以从归档日志将数据抽取到定界文件中，然后从这些文件将数据装入到 DB2 数据库表中，以便进行分析。

通过配置审计日志的位置，可以将审计日志放置在一个较大的高速磁盘中，并可以选择对分区数据库环境中的每个成员使用不同的磁盘。在分区数据库环境中，活动审计日志的路径可以是对每个成员唯一的目录。使每个成员具有唯一目录有助于避免文件争用，因为每个成员都写入不同磁盘。

在 Windows 操作系统上，审计日志的缺省路径为 *instance*\security\auditdata，而在 Linux 和 UNIX 操作系统上为 *instance*/security/auditdata。如果不想使用缺省位置，那么可以选择不同的目录（如果不存在备用位置，可以在系统上创建新目录以用作备用位置）。要设置活动审计日志位置和已归档审计日志位置的路径，请使用带有 **datapath** 和 **archivepath** 参数的 **db2audit configure** 命令，如以下示例中所示:

```
db2audit configure datapath /auditlog archivepath /auditarchive
```

使用 **db2audit** 设置的审计日志存储位置适用于实例中的所有数据库。

注: 如果服务器上有多个实例，那么每个实例都应该有不同的数据和归档路径。

分区数据库环境中的活动审计日志的路径 (datapath)

在分区数据库环境中，必须在每个分区上使用相同的活动审计日志位置（由 **datapath** 参数设置）。可使用两种方法来实现此目的:

1. 指定了 **datapath** 参数时，使用数据库分区表达式。使用数据库分区表达式允许将分区号包括在审计日志文件的路径中，并将结果包括在每个数据库分区上的不同路径中。
2. 使用在所有成员上相同的共享驱动器。

可以在对 **datapath** 参数指定的值中的任何位置使用数据库分区表达式。例如，在由三个成员组成的系统上（其中数据库分区号为 10），以下命令:

```
db2audit configure datapath '/pathForNode $N'
```

将使用以下路径:

- `/pathForNode10`
- `/pathForNode20`
- `/pathForNode30`

注: 不能使用数据库分区表达式来指定归档日志文件路径（**archivepath** 参数）。

归档活动审计日志

系统管理员可以使用 **db2audit** 工具来归档实例和数据库审计日志以及从任一类型的已归档日志中抽取审计数据。

安全性管理员或安全性管理员已向其授予对审计例程的 EXECUTE 特权的用户，可以通过运行 SYSPROC.AUDIT_ARCHIVE 存储过程来归档活动审计日志。要从日志中抽取数据并将该数据装入到定界文件中，他们可以使用 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程。

以下是使用审计例程来归档和抽取审计日志的步骤:

1. 调度应用程序以使用存储过程 SYSPROC.AUDIT_ARCHIVE 来执行活动审计日志的常规归档。
2. 确定感兴趣的已归档日志文件。使用 SYSPROC.AUDIT_LIST_LOGS 表函数来列示所有已归档审计日志。
3. 将文件名作为参数传递给 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程以从日志中抽取数据并将它们装入到定界文件中。
4. 将审计数据装入到 DB2 数据库表中以进行分析。

不需要立即将已归档日志文件装入到表中以进行分析；可以保存它们以在将来分析。例如，可能只需要在进行公司审计时查看这些文件。

如果归档期间出现问题（例如，用完归档路径中的磁盘空间，或者归档路径不存在），那么归档进程将失败并且在审计日志数据路径中生成文件扩展名为 `.bk` 的临时日志文件，例如，`db2audit.instance.log.0.20070508172043640941.bk`。在解决问题后（通过在归档路径中分配足够多的磁盘空间，或者通过创建归档路径），必须将此临

时日志移至归档路径。然后，可以像对待成功归档的日志一样对待该日志。

在分区数据库环境中归档活动审计日志

在分区数据库环境中，如果在实例正在运行时发出归档命令，那么归档进程将自动在每个成员上运行。所有成员上的已归档日志文件名中都使用相同的时间戳记。例如，在由三个成员组成的系统上（其中数据库分区号为 10），以下命令:

```
db2audit archive to /auditarchive
```

将创建下列文件:

- /auditarchive/db2audit.log.10.*timestamp*
- /auditarchive/db2audit.log.20.*timestamp*
- /auditarchive/db2audit.log.30.*timestamp*

如果在实例未运行时发出归档命令，那么可以通过下列其中一种方法控制归档命令在哪个成员上运行:

- 将 **node** 选项与 **db2audit** 命令配合使用以仅对当前成员执行归档命令。
- 使用 **db2_all** 命令对所有成员运行归档。

  例如:

  ```
  db2_all db2audit archive node to /auditarchive
  ```

  这将设置 **DB2NODE** 环境变量以指示在其上调用该命令的成员。

或者，可单独对每个成员发出一个归档命令。例如:

- 在成员 10 上:

  ```
  db2audit archive node 10 to /auditarchive
  ```

- 在成员 20 上:

  ```
  db2audit archive node 20 to /auditarchive
  ```

- 在成员 30 上:

  ```
  db2audit archive node 30 to /auditarchive
  ```

注: 当实例未在运行时，每个成员上的已归档审计日志文件名中的时间戳记不同。

注: 建议在所有成员间共享归档路径，但这不是必需的。

注: AUDIT_DELIM_EXTRACT 存储过程和 AUDIT_LIST_LOGS 表函数只能访问从当前（协调程序）成员可视的已归档日志文件。

归档日志并将数据抽取到表中的示例

一家公司为了确保能够捕获并存储其审计日志以供将来使用，需要每六个小时创建一个新的审计日志并将当前审计日志归档到 WORM 驱动器中。该公司安排安全性管理员或特定用户（安全性管理员已向该用户授予对 AUDIT_ARCHIVE 存储过程的 EXECUTE 特权）每 6 小时向 SYSPROC.AUDIT_ARCHIVE 存储过程发出下列调用一次。已归档日志的路径是缺省归档路径 /auditarchive，并且归档命令在所有成员上运行:

```
CALL SYSPROC.AUDIT_ARCHIVE( '/auditarchive', -2 )
```

作为安全过程的一部分，该公司标识并定义了一定数目的可疑行为或不允许的活动，需要在审计数据中监视这些行为或活动。他们希望抽取一个或多个审计日志中的所有数据，将这些数据放置在关系表中，然后使用 SQL 查询来查找这些活动。该公司已确定要审计的适当类别，并使必需的审计策略与数据库或其他数据库对象关联。

例如，他们可以调用 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程来从所有成员中抽取所有类别的已归档审计日志，这些审计日志是使用缺省定界符和时间戳记 2006 年 4 月创建的:

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT(
     '', '', '/auditarchive', 'db2audit.%.200604%', '' )
```

在另一个示例中，他们可以调用 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程来从 EXECUTE 类别中抽取成功事件的已归档审计记录、从 CHECKING 类别中抽取失败事件的已归档审计记录，并从具有感兴趣的时间戳记的文件中抽取已归档审计记录:

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT( '', '', '/auditarchive',
    'db2audit.%.20060419034937', 'category
     execute status success, checking status failure );
```

审计日志文件名:

审计日志文件的名称可区分它们是实例级别还是数据库级别日志，并标识它们源自分区数据库环境中的哪个分区。已归档审计日志的文件名后面追加了运行归档命令的时间戳记。

活动审计日志文件名

在分区数据库环境中，活动审计日志的路径可以是对每个分区唯一的目录，以便每个分区写入各自的文件。为了准确跟踪原始审计记录，将分区号包括在审计日志文件名中。例如，在分区 20 上，实例级别审计日志文件名为 db2audit.instance.log.20。对于此实例中名为 testdb 的数据库，审计日志文件为 db2audit.db.testdb.log.20。

在非分区数据库环境中，分区号将视为 0（零）。在此情况下，实例级别审计日志文件名为 db2audit.instance.log.0。对于此实例中名为 testdb 的数据库，审计日志文件为 db2audit.db.testdb.log.0。

已归档审计日志文件名

活动审计日志在进行归档之后，其文件名后面将追加以下格式的当前时间戳记: *YYYYMMDDHHMMSS*（其中 *YYYY* 是年份，*MM* 是月份，*DD* 是日，*HH* 是小时，*MM* 是分钟，而 *SS* 是秒）。

归档审计日志的文件名格式取决于审计日志的级别:

实例级别已归档审计日志
实例级别已归档审计日志的文件名为 db2audit.instance.log.*partition*.*YYYYMMDDHHMMSS*。

数据库级别已归档审计日志
数据库级别已归档审计日志的文件名为 db2audit.db*database*.log.*partition*.*YYYYMMDDHHMMSS*。

在非分区数据库环境中，*partition* 的值为 0（零）。

时间戳记表示运行归档命令的时间，因此它并非总是准确地反映日志中最后一条记录的时间。已归档审计日志文件可能包含一些记录，它们的时间戳记比日志文件名中的时间戳记要晚几秒钟，这是因为:

- 在发出归档命令时，审计设施将等到写入任何进程内记录完成后再创建已归档日志文件。
- 在多机器环境中，远程机器上的系统时间可能与发出归档命令的机器上的系统时间不同步。

在分区数据库环境中，如果运行归档命令时服务器正在运行，那么时间戳记在各个分区中一致并反映了在执行归档命令的分区中生成的时间戳记。

创建表来容纳 DB2 审计数据:

使用数据库表中的审计数据之前，需要创建表来容纳数据。应考虑在单独的模式中创建这些表，以将表中的数据与未授权用户相隔离。

开始之前

- 有关创建模式所需的权限和特权，请参阅 CREATE SCHEMA 语句。
- 有关创建表所需的权限和特权，请参阅 CREATE TABLE 语句。
- 确定想要用来容纳表的表空间。（本主题未描述如何创建表空间。）

注: 要创建用来容纳审计数据的表的格式在每个发行版中可能都不同。可能添加了新列，或者现有列的大小可能改变。脚本 db2audit.ddl 创建正确格式的表来包含审计记录。

关于此任务

下列示例说明如何创建表来容纳定界文件中的记录。如果愿意，可以创建单独模式来包含这些表。

如果不想使用这些文件中包含的所有数据，那么可以忽略表定义中的列或根据需要不创建某些表。如果忽略表定义的列，那么必须修改将数据装入这些表所用的命令。

过程

1. 发出 **db2** 命令打开 DB2 命令窗口。
2. 可选: 创建模式来容纳表。 对于此示例，模式名为 AUDIT:

   ```
   CREATE SCHEMA AUDIT
   ```

3. 可选: 如果创建了 AUDIT 模式，那么在创建任何表之前切换至该模式:

   ```
   SET CURRENT SCHEMA = 'AUDIT'
   ```

4. 运行脚本 db2audit.ddl 以创建将包含审计记录的表。

   脚本 db2audit.ddl 位于 sqllib/misc 目录（在 Windows 上为 sqllib\misc）中。该脚本假定数据库连接已存在并且 8K 表空间可用。用于运行该脚本的命令为: **db2 +o -tf sqllib/misc/db2audit.ddl**。该脚本创建的表有: AUDIT、CHECKING、OBJMAINT、SECMAINT、SYSADMIN、VALIDATE、CONTEXT 和 EXECUTE。

5. 创建表后，安全性管理员可以使用 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程或系统管理员可以使用 **db2audit extract** 命令将已归档审计日志文件中的审计记录抽取到定界文件中。 可以将这些定界文件中的审计数据装入到刚刚创建的数据库表中。

将 DB2 审计数据装入表中:

在已归档审计日志文件并将它抽取到定界文件中，并且创建了数据库表来保存审计数据后，可以将定界文件中的审计数据装入数据库表中以进行分析。

关于此任务

使用装入实用程序将审计数据装入表中。对每个表发出单独的装入命令。如果忽略表定义中的一个或多个列，那么必须修改使用的 **LOAD** 命令版本才能成功装入数据。此外，如果在抽取审计数据时指定了除缺省值外的定界字符，那么还必须修改使用的 **LOAD** 命令的版本。

过程

1. 发出 **db2** 命令打开 DB2 命令窗口。
2. 要装入 AUDIT 表，请发出下列命令:

   ```
   LOAD FROM audit.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.AUDIT
   ```

   注: 指定 **DELPRIORITYCHAR** 修饰符以确保正确解析二进制数据。

   注: 指定 **LOAD** 命令的 **LOBSINFILE** 选项（由于具有的限制，大对象的任何直接插入数据必须限于 32K）。在某些情况下，还可能需要使用 **LOBS FROM** 选项。

   注: 指定文件名时，请使用标准路径名。例如，如果将 DB2 数据库系统安装在 Windows 操作系统的 C: 盘上，那么应指定 `C:\Program Files\IBM\SQLLIB\instance\security\audit.del` 作为 `audit.del` 文件的标准路径名。
3. 要装入 CHECKING 表，请发出下列命令:

   ```
   LOAD FROM checking.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.CHECKING
   ```
4. 要装入 OBJMAINT 表，请发出下列命令:

   ```
   LOAD FROM objmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.OBJMAINT
   ```
5. 要装入 SECMAINT 表，请发出下列命令:

   ```
   LOAD FROM secmaint.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.SECMAINT
   ```
6. 要装入 SYSADMIN 表，请发出下列命令:

   ```
   LOAD FROM sysadmin.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.SYSADMIN
   ```
7. 要装入 VALIDATE 表，请发出下列命令:

   ```
   LOAD FROM validate.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.VALIDATE
   ```
8. 要装入 CONTEXT 表，请发出下列命令:

   ```
   LOAD FROM context.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
            INSERT INTO schema.CONTEXT
   ```

9. 要装入 EXECUTE 表，请发出下列命令:

```
LOAD FROM execute.del OF DEL MODIFIED BY DELPRIORITYCHAR LOBSINFILE
        INSERT INTO schema.EXECUTE
```

10. 将数据装入表之后，从 sqllib 目录的 security/auditdata 子目录中删除 .del 文件。
11. 将审计数据装入表之后，您就可以从这些表中选择数据以进行分析。

下一步做什么

如果初次填充表之后要再次执行此操作，则使用 **INSERT** 选项将新的表数据添加至现有表数据。如果要从表中除去以前 **db2audit extract** 操作的记录，那么使用 **REPLACE** 选项再次装入表。

审计归档和抽取存储过程:

安全性管理员可以使用 SYSPROC.AUDIT_ARCHIVE 存储过程及表函数、SYSPROC.AUDIT_DELIM_EXTRACT 存储过程和 SYSPROC.AUDIT_LIST_LOGS 表函数来归档审计日志并将数据抽取到定界文件中。

安全性管理员可以通过将对这些例程的 EXECUTE 特权授予另一个用户将这些例程的使用权委托给该用户。只有安全性管理员才能授予对这些例程的 EXECUTE 特权。对于这些例程，不能授予 EXECUTE 特权 WITH GRANT OPTION（SQLSTATE 42501）。

必须连接到数据库，才能使用这些存储过程和表函数来归档或列示该数据库的审计日志。

如果将已归档的文件复制到另一个数据库系统，并且要使用存储过程和表函数来对它们进行访问，请确保数据库名称相同，或者重命名文件以包括相同的数据库名称。

这些存储过程和表函数不会归档或列示实例级别审计日志。系统管理员必须使用 **db2audit** 命令来归档和抽取实例级别审计日志。

可以使用这些存储过程和表函数来执行下列操作:

表 5. 审计系统存储过程和表函数

存储过程和表函数	操作	注释
AUDIT_ARCHIVE	归档当前审计日志。	将归档路径用作输入。如果未提供归档路径，那么此存储过程采用审计配置文件中的归档路径。 将在每个成员上运行归档命令，并且将同步的时间戳记追加至审计日志文件名。
AUDIT_LIST_LOGS	在指定路径中返回当前数据库的已归档审计日志列表。	

表 5. 审计系统存储过程和表函数 (续)

存储过程和表函数	操作	注释
AUDIT_ DELIM_EXTRACT	从二进制已归档日志中抽取数据并将它们装入到定界文件中。	使用适合装入到 DB2 数据库表中的定界格式保存抽取的审计记录。输出将放在单独的文件中，每种类别一个文件。此外，将创建文件 **auditlobs**，以保存审计数据中包含的任何大对象。文件名为: • **audit.del** • **checking.del** • **objmaint.del** • **secmaint.del** • **sysadmin.del** • **validate.del** • **context.del** • **execute.del** • **auditlobs** 如果这些文件已存在，那么输出将追加到这些文件中。如果抽取 CONTEXT 或 EXECUTE 类别，那么将创建 **auditlobs** 文件。只能抽取当前数据库的已归档审计日志。仅抽取协调程序成员可视的那些文件。 只有实例所有者可以删除已归档的审计日志。

用于审计 SQL 语句的 EXECUTE 类别

EXECUTE 类别允许您准确地跟踪用户发出的 SQL 语句。在 V9.5 和更低发行版中，必须使用 CONTEXT 类别来查找此信息。

作为完整安全策略的一部分，公司可以要求回溯若干年并分析对某些数据库表发出的任何特定请求的影响的功能。为此，该公司必须制订一项策略，要求每周都归档备份和关联的日志文件，以便他们可以及时重新组成任何所选时刻的数据库。还需要捕获足够多的关于对数据库发出的每个请求的数据库审计信息，以便允许在将来任何时间都可以重放和分析对已复原的相关数据库发出的任何请求。此要求可同时涉及静态和动态 SQL 语句。

此 EXECUTE 类别捕获 SQL 语句文本以及将来重放该语句所需的编译环境和其他值。例如，重放语句可以向您准确地显示 SELECT 语句返回的行。要重新运行语句，首先必须将数据库表复原到发出该语句时它们所处的状态。

使用 EXECUTE 类别进行审计时，随着记录输入参数标记和主变量，将记录静态和动态 SQL 的语句文本。可使用输入值或不使用输入值配置要审计的 EXECUTE 类别。

注: 不审计全局变量。

对 EXECUTE 事件的审计在该事件完成时进行（对于 SELECT 语句，审计在游标关闭时进行）。还会存储事件完成时的状态。因为 EXECUTE 事件是在完成时审计的，所以长期运行的查询不会立即出现在审计日志中。

注: 预编译语句不视为执行的一部分。大多数授权检查都在预编译时执行（例如，SELECT 特权）。这意味着预编译期间由于授权错误而失败的语句不会生成 EXECUTE 事件。

“语句值索引”、“语句值类型”和“语句值数据”字段可能对给定执行记录重复。对于通过抽取生成的报告格式，每条记录将列示多个值。对于定界文件格式，将使用多行。第一行的事件类型为 STATEMENT 并且没有值。随后的行的事件类型为 DATA，其中一行表示与 SQL 语句关联的每个数据值。可以使用事件相关因子和应用程序标识字段将 STATEMENT 和 DATA 行链接在一起。“语句文本”、“语句隔离级别”和“编译环境描述”列不出现在 DATA 事件中。

将审计过的语句文本和输入数据值存储在磁盘上时，它们会转换到数据库代码页中（所有审计过的字段都存储在数据库代码页中）。如果输入数据的代码页与数据库代码页不兼容，那么不会返回错误；将改为记录未转换的数据。因为每个数据库都有自己的审计日志，所以具有不同代码页的数据库不会产生问题。

ROLLBACK 和 COMMIT 在应用程序执行它们时进行审计，并且在另一个命令（如 BIND）中隐式发出 ROLLBACK 和 COMMIT 时也会对它们进行审计。

在由于访问已审计的表而审计了 EXECUTE 事件后，将审计影响工作单元中执行哪些其他语句的所有语句。这些语句是 COMMIT、ROLLBACK、ROLLBACK TO SAVEPOINT 和 SAVEPOINT 语句。

保存点标识字段

可以使用“保存点标识”字段来跟踪受 ROLLBACK TO SAVEPOINT 语句影响的语句。普通的 DML 语句（如 SELECT、INSERT 等）将审计当前保存点标识。但是，对于 ROLLBACK TO SAVEPOINT 语句，将改为审计将回滚到的保存点标识。因此，保存点标识大于或等于该标识的每个语句都将回滚，如以下示例所示。表中显示了语句的运行顺序；保存点标识大于或等于 2 的所有事件都将回滚。只有值 3（来自第一个 INSERT 语句）将插入到表 T1 中。

表 6. 说明 ROLLBACK TO SAVEPOINT 语句的效果的语句顺序

语句	保存点标识
INSERT INTO T1 VALUES (3)	1
SAVEPOINT A	2
INSERT INTO T1 VALUES (5)	2
SAVEPOINT B	3
INSERT INTO T1 VALUES (6)	3
ROLLBACK TO SAVEPOINT A	2
COMMIT	

WITH DATA 选项

指定 WITH DATA 选项后，并不会审计所有输入值。LOB、LONG、XML 和结构化类型参数将显示为 NULL。

日期、时间和时间戳记字段记录为 ISO 格式。

如果在一个策略中指定了 WITH DATA，但在另一个策略中指定了 WITHOUT DATA 并且该策略与执行的 SQL 语句中所涉及的对象关联，那么 WITH DATA 将优先，并且将审计该特定语句的数据。例如，如果与用户关联的审计策略指定 WITHOUT DATA，但与表关联的策略指定 WITH DATA，那么在该用户访问该表时，将审计用于语句的输入数据。

您无法确定在定位更新或定位删除语句中修改了哪些行。仅记录执行的底层 SELECT 语句，而不记录单独的 FETCH。在发出语句时，根据 EXECUTE 记录不能确定游标所在的行。以后重放语句时，只能发出 SELECT 语句来了解可能受影响的行的范围。

重放过去的活动的示例

在此示例中，请考虑一家公司的完整安全策略的一部分，该公司要求他们保留最多回退 7 年以分析对某些数据库表发出的任何特定请求的效果的功能。为此，他们制订了一项策略，要求每周都归档备份和关联的日志文件，以便他们可以重新组成选择的任何时刻的数据库。他们要求数据库审计捕获足够多关于对数据库发出的每个请求的信息，以便允许重放和分析对已复原的相关数据库发出的任何请求。此要求同时涉及静态和动态 SQL 语句。

此示例显示在发出 SQL 语句时必须已存在的审计策略，以及用于归档审计日志并在以后抽取和分析这些日志的步骤。

1. 创建用于审计 EXECUTE 类别的审计策略并将此策略应用于数据库:

```
CREATE AUDIT POLICY STATEMENTS CATEGORIES EXECUTE WITH DATA
   STATUS BOTH ERROR TYPE AUDIT
COMMIT

AUDIT DATABASE USING POLICY STATEMENTS
COMMIT
```

2. 定期归档审计日志以创建归档副本。

   安全性管理员或被授予对 SYSPROC.AUDIT_ARCHIVE 存储过程的 EXECUTE 特权的用户应根据记录的数据量定期（例如，每周一次或每天一次）运行以下语句。可以根据需要将这些已归档的文件保留任意长时间。AUDIT_ARCHIVE 过程是使用两个输入参数（归档目录的路径和 -2）来调用的，以指示应在所有成员上运行归档:

```
CALL SYSPROC.AUDIT_ARCHIVE( '/auditarchive', -2 )
```

3. 安全性管理员或被授予对 SYSPROC.AUDIT_LIST_LOGS 表函数的 EXECUTE 特权的用户使用 AUDIT_LIST_LOGS 来检查自 2006 年 4 月以来的所有可用审计日志，以确定哪些日志可能包含必需的数据:

```
SELECT FILE FROM TABLE(SYSPROC.AUDIT_LIST_LOGS('/auditarchive'))
   AS T WHERE FILE LIKE 'db2audit.dbname.log.0.200604%'
FILE----------------------------------------
...
db2audit.dbname.log.0.20060418235612
db2audit.dbname.log.0.20060419234937
db2audit.dbname.log.0.20060420235128
```

4. 通过此输出，安全性管理员发现必需的日志应该在一个文件 db2audit.dbname.log.20060419234937 中。时间戳记显示此文件是在审计员想要查看的那天快结束时归档的。

   安全性管理员或被授予对 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程的 EXECUTE 特权的用户将此文件名用作 AUDIT_DELIM_EXTRACT 的输入，以将审计数据抽取到定界文件中。可以将这些文件中的审计数据装入到 DB2 数据库表中，然后在这些表中分析数据以查找审计员感兴趣的特定语句。即使审计员只对单个 SQL 语句感兴趣，也可能需要检查工作单元中的多个语句，以防这些语句对感兴趣的语句有任何影响。

5. 为了重放语句，安全性管理员必须执行下列操作:
   - 根据审计记录确定要发出的准确语句。
   - 根据审计记录确定发出语句的用户。
   - 重新创建用户在发出该语句时拥有的准确许可权，包括任何 LBAC 保护。
   - 通过使用审计记录中的编译环境列和 SET COMPILATION ENVIRONMENT 语句再现编译环境。
   - 将数据库复原到发出语句时它所处的准确状态。

   为了避免影响生产系统，应在另一个数据库系统上执行所有复原数据库和重放语句的操作。作为发出语句的用户运行的安全性管理员可以使用语句值数据元素中提供的任何输入变量重新发出在语句文本中找到的语句。

启用过去的活动的重放:

作为完整安全策略的一部分，公司可以要求回溯若干年并分析对某些数据库表发出的任何特定请求的影响的功能。

开始之前

公司必须制订一项策略，要求每周都归档备份和关联的日志文件，以便他们可以及时重新组成任何所选时刻的数据库。

关于此任务

要允许在将来任何时间都可以重放和分析对已复原的相关数据库发出的任何请求，必须捕获足够多的关于对数据库发出的每个请求的数据库审计信息。此要求可同时涉及静态和动态 SQL 语句。当记录的 WITH DATA 包含重放过去的 SQL 语句所必需的信息时，EXECUTE 类别假设已将数据库中的数据复原到发出该语句时该数据所处的状态。

限制

下列权限和特权是必需的:

- SECADM 权限是创建审计策略所必需的，
- EXECUTE 特权是所有审计例程和过程所必需的。

过程

要作为 SECADM 启用重放过去的活动，请执行以下操作:

1. 创建用于审计 EXECUTE 类别的审计策略并将此策略应用于数据库。

```
CREATE AUDIT POLICY STATEMENTS CATEGORIES EXECUTE WITH DATA
   STATUS BOTH ERROR TYPE AUDIT
COMMIT
AUDIT DATABASE USING POLICY STATEMENTS
COMMIT
```

2. 定期归档审计日志以创建归档副本。 要归档审计日志，请定期运行下列命令，指定归档目录的路径和 -2 以指示应在所有成员上运行归档:

   ```
   CALL SYSPROC.AUDIT_ARCHIVE( '/auditarchive', -2 )
   ```

3. 检查已创建的审计日志文件。 然后，这些归档文件将保存一定年数（该年数由公司的业务策略指定）。 要检查审计日志文件，请运行下列命令:

   ```
   SELECT FILE FROM SESSION.AUDIT_ARCHIVE_RESULTS
   ```

结果

现在，已设置了您的环境，这样会归档数据和信息以允许将来重放所记录的数据库活动。

重放过去的数据库活动:

如果所有必需的数据、日志和信息都可用，那么有可能重放过去的数据库活动。此参考主题通过示例显示了 SECADM 会如何重放过去的数据库活动。

描述

在某个时刻，公司审计员可能想要分析特定用户发生在过去的活动。SECADM 可以使用备份数据库映像（配合备份日志使用）和审计日志来重新组成有问题的数据库，并重放审计员想要分析的活动。假定特定用户发生在 2006 年 4 月 19 日的活动有问题，以下示例显示了 SECADM 可以如何帮助审计员执行其分析的流程。

示例

1. SECADM 将发出 AUDIT_LIST_LOGS 以查找自 2006 年 4 月起所有可用的审计日志。

   ```
   SELECT FILE FROM TABLE(SYSPROC.AUDIT_LIST_LOGS('/auditarchive'))
      AS T WHERE FILE LIKE 'db2audit.db.sample.log.0.200604%'
   FILENAME
   ----------------------------------------
   ...
   db2audit.db.sample.log.0.20060418235612
   db2audit.db.sample.log.0.20060419234937
   db2audit.db.sample.log.0.20060420235128
   ```

2. 通过此输出，SECADM 发现必要的日志应位于 db2audit.db.sample.log.20060419234937 文件中。该日志记录于 2006 年 4 月 19 日工作日结束时。
3. 这会用作 SYSPROC.AUDIT_DELIM_EXTRACT 存储过程的输入。传递给该过程的自变量有:
   - 字符定界符（缺省），
   - 输出路径，
   - 已归档的审计日志的路径，
   - 用于确定要从中提取哪些文件的文件名过滤器，
   - 要提取的每个类别的状态，在此示例中，只有一个类别 EXECUTE。

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT( '', '', '/auditarchive',
   'db2audit.db.sample.log.0.20060419234937',
   'category execute' )
```

4. 现在，审计数据已经位于定界文件中。SECADM 将把审计数据从 EXECUTE 类别装入 AUDITDATA.EXECUTE 表。该表可通过执行下列命令进行创建:

```
db2 CONNECT TO sample
db2 SET CURRENT SCHEMA AUDITDATA
db2 -tvf sqllib/misc/db2audit.ddl
```

5. 下一步，将数据从 execute.del 装入 AUDITDATA.EXECUTE 表。要执行此操作，请运行下列命令:

```
db2 LOAD FROM FILE execute.del OF DEL MODIFIED BY
LOBSINFILE INSERT INTO AUDITDATA.EXECUTE
```

6. 现在，SECADM 已将所有审计数据都放在位于 AUDITDATA 模式内的审计表中。现在，可以分析此数据以查找审计员感兴趣的特定语句。

   注: 即使审计员只对单个 SQL 语句感兴趣，也可能需要检查工作单元中的多个语句，以防这些语句对感兴趣的语句有任何影响。

7. 要重放语句，必须执行下列操作:
   - 必须根据审计记录来确定发出的准确语句。
   - 必须根据审计记录来确定发出语句的用户。
   - 必须重新创建用户在发出语句时拥有的准确许可权，包括任何 LBAC 保护。
   - 必须通过使用审计记录中的编译环境列和 SET COMPILATION ENVIRONMENT 语句再现编译环境。
   - 必须重新创建数据库在发出语句时所处的准确状态。

   注: 为了避免影响生产系统，应在辅助数据库系统上执行所有复原数据库和重放语句的操作。

8. SECADM 将需要前滚至该语句将要开始执行的时间。语句的本地开始时间 (local_start_time) 是 EXECUTE 审计记录的一部分。使用下列 EXECUTE 审计记录作为示例:

```
timestamp=2006-04-10-13.20.51.029203;
  category=EXECUTE;
  audit event=STATEMENT;
  event correlator=1;
  event status=0;
  database=SAMPLE;
  userid=smith;
  authid=SMITH;
  session authid=SMITH;
  application id=*LOCAL.prodrig.060410172044;
  application name=myapp;
  package schema=NULLID;
  package name=SQLC2F0A;
  package section=201;
  uow id=2;
  activity id=3;
  statement invocation id=0;
  statement nesting level=0;
  statement text=SELECT * FROM DEPARTMENT WHERE DEPTNO = ? AND DEPTNAME = ?;
  statement isolation level=CS;
  compilation environment=
    isolation level=CS
    query optimization=5
    min_dec_div_3=NO
```

```
    degree=1
    sqlrules=DB2
    refresh age=+00000000000000.000000
    schema=SMITH
    maintained table type=SYSTEM
    resolution timestamp=2006-04-10-13.20.51.000000
    federated asynchrony=0;
  value index=0;
  value type=CHAR;
  value data=C01;
  value index=1;
  value type=VARCHAR;
  value index=INFORMATION CENTER;  local_start_time=2006-04-10-13.20.51.021507;
```

前滚语句将与下列内容相似:

```
ROLLFORWARD DATABASE sample TO 2006-04-10-13.20.51.021507 USING
LOCAL TIME AND COMPLETE
```

9. 还需要设置编译环境。编译环境变量可通过 SET COMPILATION ENVIRONMENT 语句进行设置。作为发出语句的用户运行的 SECADM 现在可以使用语句值数据元素中提供的任何输入变量重放在语句文本中找到的语句。以下是使用 C 嵌入式 SQL 语言编写的一个样本程序，该程序将设置 COMPILATION ENVIRONMENT 并重放审计员想要分析的 SELECT 语句:

```
EXEC SQL INCLUDE SQLCA;

EXEC SQL  BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB(1M) hv_blob;
EXEC SQL  END DECLARE SECTION;

EXEC SQL DECLARE c1 CURSOR FOR SELECT COMPENVDESC FROM AUDITDATA.EXECUTE
TIMESAMP= '2006-04-10-13.20.51.029203';
EXEC SQL DECLARE c2 CURSOR FOR SELECT * FROM DEPARTMENT WHERE
DEPTNO = 'C01' AND DEPTNAME = 'INFORMATION CENTER';
EXEC SQL OPEN c1;

EXEC SQL FETCH c1 INTO :hv_blob;

EXEC SQL SET COMPILATION ENVIRONMENT :hv_blob;

EXEC SQL OPEN c2;

....

EXEC SQL CLOSE c1;
EXEC SQL CLOSE c2;
```

审计设施管理

审计设施行为

本主题提供了一些后台信息，它们有助于您了解将审计记录写入日志的时间如何影响数据库性能；如何管理审计设施中发生的错误；以及审计记录在不同情况下是如何生成的。

控制将审计记录写入活动日志的时间

将审计记录写入活动日志可与导致生成那些记录的事件同步或异步发生。**audit_buf_sz** 数据库管理器配置参数的值确定何时写入审计记录。

如果 **audit_buf_sz** 的值为零 (0)，那么异步写入记录。生成审计记录的事件将等到记录写入磁盘为止。与每个记录相关的等待将导致 DB2 数据库性能降级。

如果 **audit_buf_sz** 的值大于 0，那么异步写入该记录。**audit_buf_sz** 的值大于 0 时，该值是 4 KB 页的一个倍数，用来创建内部缓冲区。该内部缓冲区用来在将一组审计记录写入磁盘之前保存大量审计记录。作为审计事件的结果生成审计记录的语句将不会等到将该记录写入磁盘，它可继续其操作。

在异步情况下，审计记录可在未填写的缓冲区中保留一段时间。要防止这种情况的持续时间过长，数据库管理器将强制定期写入审计记录。审计设施的授权用户也可通过显式请求清空审计缓冲区。此外，缓冲区会在归档操作期间自动清空。

记录是同步写入还是异步写入，使发生错误时情况有些不同。在异步方式中，可能会有某些记录丢失，因为审计记录是在写入磁盘之前缓冲存储的。在同步方式中，可能有一个记录丢失，因为错误只能阻止最多一个审计记录写入。

管理审计设施错误

ERRORTYPE 审计设施参数的设置控制如何在 DB2 数据库系统和审计设施之间管理错误。当审计设施是活动的，并且 ERRORTYPE 审计设施参数的设置是 AUDIT 时，将审计设施作为 DB2 数据库的任何其他部件一样对待。对于与被视为成功的语句关联的一个审计事件，必须写入审计记录（在同步方式下，写至磁盘；在异步方式下，写至审计缓冲区）。当在此方式下运行时，不管何时遇到错误，都将一个负的 SQLCODE 返回至生成审计记录的语句的应用程序。

如果错误类型设置为 NORMAL，那么将忽略来自 **db2audit** 的任何错误，并返回该操作的 SQLCODE。

不同情况下生成的审计记录

根据 API 或查询语句以及审计设置，可为特定事件生成一个或几个审计记录，或不生成审计记录。例如，具有一个 SELECT 子查询的 SQL UPDATE 语句可生成两个审计记录，一个记录包含对一个表的 UPDATE 特权的权限检查结果，另一个记录包含对一个表的 SELECT 特权的权限检查结果。

对于动态数据操作语言（DML）语句，在准备该语句时会对所有权限检查生成审计记录。不会再次审计同一用户对那些语句的复用，因为那时不进行权限检查。但是，如果已更改包含特权信息的目录表之一，那么在下一个工作单元中，再次检查高速缓存的动态 SQL 或 XQuery 语句的语句特权，并创建一个或多个新的审计记录。

对于仅包含静态 DML 语句的程序包，可生成审计记录的唯一可审计的事件是权限检查，它查看用户是否具有执行该程序包的特权。在预编译或绑定该程序包时，执行该程序包中静态 SQL 或 XQuery 语句所需的权限检查和可能的审计记录创建。可使用 EXECUTE 类别审计在程序包内执行的静态 SQL 或 XQuery 语句。当用户显式地再次绑定程序包时，或系统隐式再次绑定程序包时，将为该静态 SQL 或 XQuery 语句所需的权限检查生成审计记录。

对于在执行语句时执行权限检查的语句（例如，数据定义语言（DDL）、GRANT 和 REVOKE 语句），不管何时使用这些语句，都将生成审计记录。

注: 当执行 DDL 时，无论该语句的实际节号可能是什么，在审计记录中为所有事件（除上下文事件外）记录的节号都将为零（0）。

审计设施技巧

用于管理审计的最佳实践包括: 定期归档审计日志，当创建审计策略时使用错误类型 AUDIT 以及下面所述的其他技巧。

将审计日志归档

应定期将审计日志归档。将审计日志归档时会将当前审计日志移至一个归档目录，而服务器开始写入新的活动审计日志。已归档的每个日志文件的名称中都包含一个时间戳记，可帮助您标识感兴趣的日志文件以便于将来进行分析。

为了长期存储，可能要对若干组已归档文件进行压缩。

对于您不再感兴趣的已归档审计日志，实例所有者只需从操作系统中删除这些文件即可。

错误处理

创建审计策略时，应该使用错误类型 AUDIT，除非您创建的只是一个测试审计策略。例如，如果错误类型设置为 AUDIT 并且发生了错误（例如，磁盘空间耗尽），那么将返回错误。必须更正错误情况之后才能继续执行任何其他可审计的操作。但是，如果错误类型设置为 NORMAL，那么记录将失败，但不会对用户返回错误。将像未发生错误一样继续执行操作。

如果归档期间出现问题（例如，用完归档路径中的磁盘空间，或者归档路径不存在），那么归档进程将失败并且在审计日志数据路径中生成文件扩展名为 `.bk` 的临时日志文件，例如，`db2audit.instance.log.0.20070508172043640941.bk`。在解决问题后（通过在归档路径中分配足够多的磁盘空间，或者通过创建归档路径），必须将此临时日志移至归档路径。然后，可以像对待成功归档的日志一样对待该日志。

DDL 语句限制

在进入下一个工作单元之前，某些数据定义语言 (DDL) 语句（称为 AUDIT 独占式 SQL 语句）不会生效。因此，建议在执行这些语句当中的每个语句之后就立即执行 COMMIT 语句。

AUDIT 独占式 SQL 语句包括:

- AUDIT
- CREATE AUDIT POLICY、ALTER AUDIT POLICY 和 DROP AUDIT POLICY
- DROP ROLE 和 DROP TRUSTED CONTEXT（如果要删除的角色或可信上下文与审计策略关联）

用于存放已归档数据的表格式可能会改变

安全性管理员可以使用 SYSPROC.AUDIT_DEL_EXTRACT 存储过程（系统管理员可以使用 **db2audit extract** 命令）将已归档审计日志文件中的审计记录解压缩到定界文件中。可以将这些定界文件中的审计数据装入到 DB2 数据库表中以进行分析。要创建用来容纳审计数据的表的格式在每个发行版中可能都不同。

要点: 脚本 db2audit.ddl 创建正确格式的表来包含审计记录。您将期望对每个发行版都运行 db2audit.ddl，因为可能添加了列，或者现有列的大小可能改变。

使用 CHECKING 事件

在大多数情况下，当使用 CHECKING 事件时，审计记录中的对象类型字段是要检查的对象，以了解试图访问该对象的用户标识是否拥有必需的特权或权限。例如，如果用户尝试通过添加列来改变一个表，那么 CHECKING 事件审计记录将指示尝试的访问是“ALTER”，且要检查的对象类型是“TABLE”（不是列，因为检查的是表特权）。

但是，当该检查要验证是否存在数据库权限来允许用户标识创建、绑定或删除对象时，虽然会针对数据库进行检查，但对象类型字段仍将指定要创建、绑定或删除的对象（而不是数据库本身）。

在表上创建一个索引时，需要创建索引的特权，因此，CHECKING 事件审计记录将具有访问尝试类型“索引”而不是“创建”。

创建用于绑定程序包的审计记录

当绑定一个已存在的程序包时，会为该程序包的 DROP 创建 OBJMAINT 事件审计记录，然后为该程序包新副本的 CREATE 创建另一个 OBJMAINT 事件审计记录。

回滚后使用 CONTEXT 事件信息

“数据定义语言”(DDL) 可生成记录为成功的 OBJMAINT 或 SECMAINT 事件。但是，在记录该事件后，后续发生的错误可能会导致进行回滚。这样将无法创建对象；或者 GRANT 或 REVOKE 操作不能完成。在此情况下，使用 CONTEXT 事件变得很重要。这类 CONTEXT 事件审计记录，特别是结束该事件的语句，将指示尝试的操作的完成性质。

装入定界符

抽取适合装入 DB2 数据库表中的定界格式的审计记录时，应清楚该语句文本字段内使用的定界符的有关情况。这可在抽取定界文件时使用下列语句来完成:

```
db2audit extract delasc delimiter load_delimiter
```

load_delimiter 可以是单个字符（例如，″）或者是表示十六进制值的四字节字符串（例如，『0x3b』）。有效命令的示例是:

```
db2audit extract delasc
   db2audit extract delasc delimiter !
db2audit extract delasc delimiter 0x3b
```

如果抽取时使用的定界符不是缺省装入定界符，那么应在 **LOAD** 命令中使用 **MODIFIED BY** 选项。下面是将 『0x3b』 用作定界符的 **LOAD** 命令的示例一部分:

```
db2 load from context.del of del modified by chardel0x3b replace into ...
```

这将覆盖缺省装入字符串定界符 ″（双引号）。

db2cluster 命令的安全性模型

db2cluster 命令是进入 DB2 集群服务的主接口，并以此身份充当为 IBM DB2 pureScale Feature 提供的集群管理器和共享文件系统集群。用户可用的 **db2cluster** 命令选项取决于用户的权限。

就 **db2cluster** 命令的安全性模型而言，一共有 3 个用户组（按每个用户组可能执行的任务类型划分）：

- 在系统上具有用户标识的任何人

  此组中的用户能够使用 **db2cluster** 命令来报告有关 DB2 pureScale实例的信息，但不能进行任何更改。

- SYSADM、SYSCTL 或 SYSMAINT 组

  此组中的用户能够使用 **db2cluster** 命令来使实例保持启动并运行并在集群管理器上执行一些管理任务。根据定义，此组中的用户是实例的用户标识、实例所有者的主组的成员或实例所有者的非主组的成员。DB2 建议使用具有实例所有者的非主组成员资格的用户标识来执行普通日常活动

- DB2 集群服务管理员

  此组中的用户不需要访问数据库中的数据；这是用于以下操作的管理角色:

  – 安装和配置 DB2 的 DB2 集群服务部分
  – 维护集群域中的集群实例及维护共享文件系统集群

  DB2 集群服务管理员角色是可访问操作系统的由 root 用户所有的用户标识的最终用户；例如，此角色是操作系统管理员。DB2 集群服务会影响所有集群环境，您是使用 DB2 pureScale Feature 还是具有集成 HA 的分区数据库环境。因此，数据库上充当 DBADM、SECADM、SQLADM、WLMADM、EXPLAIN、ACCESSCTRL 和 DATAACCESS 之类的角色未提供集群管理的适当权限级别。DB2 集群服务管理员与具有 SYSADM、SYSCTL 或 SYSMAINT 组中的用户标识的某人可能是同一人。

  注: 不能仅仅因为用户具有 SYSADM 特权就意味着此用户一定具有操作系统管理特权。

db2cluster 的集群管理器任务

- 在系统上具有用户标识的任何人都可使用 **-list** 和 **-verify** 选项来检索有关集群域的当前状态的信息。
- SYSADM、SYSMAINT 或 SYSCTL 组中的用户可使用 **-list** 和 **-set** 选项来查询和更改首选主集群高速缓存设施。因此，这些用户可使用 **-clear -alert** 选项来清除当前实例中的任何主机、成员和集群高速缓存设施的警报（按 DB2INSTANCE 注册表变量的定义）。此组中的用户还可创建和删除集群资源并修复集群管理器资源模型；强烈建议只有在 DB2 服务人员的建议下才执行这些任务。
- DB2 集群服务管理员可在集群域内的所有主机上执行影响所有集群实例中的整体 DB2 集群服务的管理任务。此用户可使用 **-set** 选项来执行配置任务（例如，设置仲裁设备和主机故障检测时间）。而且，DB2 集群服务管理员可执行与维护相关的任务，例如，使用 **-enter** 选项将主机置于维护方式，或者使用 **-commit** 选项将更改或更新落实至集群管理器。此用户还可在集群管理器对等域上执行高级维护操作，例如，创建、删除、启动或停止域以及添加或除去主机；但是，强烈建议只有在 DB2 服务人员建议时才执行这些任务。

db2cluster 的共享文件系统任务

- 在系统上具有用户标识的任何人都可使用 **-list** 和 **-verify** 选项来检索有关集群域的当前状态的信息。这些用户还可使用 **db2cluster** 命令选项来执行各种文件系统操作，但他们能够执行的操作受常规文件系统许可权约束。只要运行此命令的用户标识具有要使用的设备的读写所有权，那么该用户就可创建文件系统和添加磁盘。创建或安装文件系统后，对该文件系统的访问权被限制为创建它的用户标识以及 DB2 集群服务管理员，所以只有这些用户才能除去、删除或重新平衡文件系统。创建它的用户标识或 DB2 集群服务管理员可创建其他用户可访问的目录（与普通文件系统很像）。
- DB2 集群服务管理员可在集群域内的所有主机上执行影响所有集群实例中的整体 DB2 集群服务的管理任务。此用户可使用 **-set** 选项来对仲裁设备执行更改选项。而且，DB2 集群服务管理员可执行与维护相关的任务，例如，使用 **-enter** 选项将主机置于维护方式，或者使用 **-commit** 选项将更改或更新落实至共享文件系统。此用户还可在共享文件系统集群上执行高级维护操作，例如，创建、删除、启动或停止域以及添加或除去主机；但是，强烈建议只有在 DB2 服务人员建议时才执行这些任务。

第 2 章 角色

角色通过提供与组等价的功能但没有相同的限制，简化了特权的管理。

角色是将一项或多项特权集中在一起的数据库对象，可以使用 GRANT 语句将角色指定给用户、组、PUBLIC 或其他角色，也可以使用 CREATE TRUSTED CONTEXT 或 ALTER TRUSTED CONTEXT 语句将它指定给可信上下文。可以对工作负载定义中的 SESSION_USER ROLE 连接属性指定角色。

角色提供了一些优点，使得管理数据库系统中的特权变得更容易:

- 安全性管理员可以采用反映他们的组织结构的方式来控制对数据库的访问（他们可以在数据库中创建直接映射至组织中的工作职能的角色）。
- 授予用户在反映他们的工作职责的角色中的成员资格。当他们的工作职责变化时，可以很方便地授予和撤销他们在角色中的成员资格。
- 简化了特权的分配。管理员可以将一组特权授予表示特定工作职能的一个角色，然后将该角色授予该工作职能中的每个用户，而不用将相同的一组特权授予该工作职能中的每个单独用户。
- 可以更新角色的特权，并且授予了该角色的所有用户都将接收更新；管理员不需要逐个更新每个用户的特权。
- 在创建视图、触发器、具体化查询表 (MQT)、静态 SQL 和 SQL 例程时始终使用授予角色的特权和权限，但不使用直接或间接授予组的特权和权限。

  这是因为 DB2 数据库系统不能确定组中的成员资格何时更改，因为组由第三方软件（例如，操作系统或 LDAP 目录）管理。由于角色是在数据库内管理的，所以 DB2 数据库系统可以确定权限何时更改并相应地进行操作。将不考虑授予组的角色，其原因与不考虑组的原因相同。
- 指定给用户的所有角色将在该用户建立连接时启用，因此在用户连接时将考虑授予角色的所有特权和权限。不能显式启用或禁用角色。
- 安全性管理员可以将角色的管理权委托给其他人。

可以对角色授予能够在数据库内授予的所有 DB2 特权和权限。例如，可以授予角色下列任何权限和特权:

- DBADM、SECADM、DATAACCESS、ACCESSCTRL、SQLADM、WLMADM、LOAD 和 IMPLICIT_SCHEMA 数据库权限
- CONNECT、CREATETAB、CREATE_NOT_FENCED、BINDADD、CREATE_EXTERNAL_ROUTINE 和 QUIESCE_CONNECT 数据库权限
- 任何数据库对象特权（包括 CONTROL）

当用户连接至数据库时，将自动启用用户角色并考虑进行授权；不需要使用 SET ROLE 语句来激活角色。例如，在创建视图、具体化查询表 (MQT)、触发器、程序包或 SQL 例程时，通过角色获取的特权适用。但是，通过授予您所属的组的角色获取的特权将不适用。

角色没有所有者。安全性管理员可以使用 GRANT 语句的 WITH ADMIN OPTION 子句将角色的管理权委托给另一个用户，以便其他用户可以控制角色成员资格。

限制

在使用角色时有一些限制:

- 角色不能拥有数据库对象。
- 在创建下列数据库对象时，将不考虑授予组的权限和角色:
  - 包含静态 SQL 的程序包
  - 视图
  - 具体化查询表 (MQT)
  - 触发器
  - SQL 例程

  在创建这些对象时，只考虑直接或间接（例如，通过角色层次结构）授予创建对象的用户或 PUBLIC 的角色。

在角色中创建和授予成员资格

安全性管理员有权创建、删除、授权、撤销和注释角色。安全性管理员使用 GRANT (Role) 语句将角色中的成员资格授予一个授权标识并使用 REVOKE (Role) 语句撤销授权标识在角色中的成员资格。

通过使用 WITH ADMIN OPTION 授予授权标识在角色中的成员资格，安全性管理员可以将角色中成员资格的管理权委托给该授权标识。GRANT (Role) 语句的 WITH ADMIN OPTION 子句使其他用户可以:

- 将角色授予其他人。
- 撤销其他人的角色。
- 注释角色。

WITH ADMIN OPTION 子句不会使其他用户具有下列功能:

- 删除角色。
- 撤销授权标识对角色的 WITH ADMIN OPTION。
- 将 WITH ADMIN OPTION 授予其他人（如果您没有 SECADM 权限）。

在安全性管理员创建了角色后，数据库管理员可以使用 GRANT 语句将权限和特权指定给角色。可以对角色授予能够在数据库内授予的所有 DB2 特权和权限。不能将实例级别权限（例如，SYSADM 权限）指定给角色。

安全性管理员或安全性管理员使用 WITH ADMIN OPTION 为其授予了角色中的成员资格的任何用户都可以使用 GRANT (Role) 语句将该角色中的成员资格授予其他用户、组、PUBLIC 或角色。可能已使用 WITH ADMIN OPTION 并直接或通过 PUBLIC、组或角色间接授予用户在角色中的成员资格。

指定给用户的所有角色在该用户建立会话时启用。在 DB2 数据库系统检查权限时，将考虑与用户角色关联的所有特权和权限。某些数据库系统使用 SET ROLE 语句来激活

特定角色。DB2 数据库系统支持 SET ROLE 是为了与使用 SET ROLE 语句的其他产品兼容。在 DB2 数据库系统中，SET ROLE 语句检查会话用户是否是角色的成员，如果不是，那么它将返回错误。

要撤销用户在角色中的成员资格，安全性管理员或拥有对该角色的 WITH ADMIN OPTION 特权的用户可使用 REVOKE (Role) 语句。

示例

如果某个角色具有一组特权，那么授予了此角色中的成员资格的用户将继承这些特权。继承特权使得在将一个用户的特权重新指定给另一个用户时不必管理各个特权。使用角色时唯一需要的操作是撤销一个用户在角色中的成员资格并将角色中的成员资格授予其他用户。

例如，职员 BOB 和 ALICE 在部门 DEV 中工作，他们具有对表 SERVER、CLIENT 和 TOOLS 的 SELECT 特权。一天，管理人员决定将他们转到一个新部门 QA，因此数据库管理员必须撤销他们对表 SERVER、CLIENT 和 TOOLS 的 SELECT 特权。后来，部门 DEV 雇佣了一位新职员 TOM，数据库管理员必须将对表 SERVER、CLIENT 和 TOOLS 的 SELECT 特权授予 TOM。

使用角色时，将执行下列步骤:

1. 安全性管理员创建角色 DEVELOPER:

   ```
   CREATE ROLE DEVELOPER
   ```

2. 拥有 DBADM 权限的数据库管理员将对表 SERVER、CLIENT 和 TOOLS 的 SELECT 特权授予角色 DEVELOPER:

   ```
   GRANT  SELECT ON TABLE SERVER TO ROLE DEVELOPER
   GRANT  SELECT ON TABLE CLIENT TO ROLE DEVELOPER
   GRANT  SELECT ON TABLE TOOLS TO ROLE DEVELOPER
   ```

3. 安全性管理员将角色 DEVELOPER 授予部门 DEV 中的用户 BOB 和 ALICE:

   ```
   GRANT ROLE DEVELOPER TO USER BOB, USER ALICE
   ```

4. 当 BOB 和 ALICE 离开部门 DEV 后，安全性管理员撤销用户 BOB 和 ALICE 的角色 DEVELOPER:

   ```
   REVOKE ROLE DEVELOPER FROM USER BOB, USER ALICE
   ```

5. 当部门 DEV 中雇佣 TOM 时，安全性管理员将角色 DEVELOPER 授予用户 TOM:

   ```
   GRANT ROLE DEVELOPER TO USER TOM
   ```

角色层次结构

对一个角色授予另一个角色中的成员资格时，就形成了角色层次结构。

将一个角色授予另一个角色后，后一个角色将*包含*前一个角色。后一个角色将继承前一个角色的所有特权。例如，如果将角色 DOCTOR 授予角色 SURGEON，那么认为 SURGEON 包含 DOCTOR。角色 SURGEON 将继承角色 DOCTOR 的所有特权。

不允许角色层次结构中出现循环。如果以循环方式授予角色，以便将一个角色授予另一个角色，而又将后者授予原始角色，那么就会出现*循环*。例如，将角色 DOCTOR 授予角色 SURGEON，然后将角色 SURGEON 授予回角色 DOCTOR。如果在角色层次结构中形成循环，那么将返回错误（SQLSTATE 428GF）。

构建角色层次结构的示例

以下示例显示如何构建角色层次结构来表示医院中的医疗级别。

请考虑下列角色: DOCTOR、SPECIALIST 和 SURGEON。通过将一个角色授予另一个角色但不形成循环来构建角色层次结构。将角色 DOCTOR 授予角色 SPECIALIST，然后将角色 SPECIALIST 授予角色 SURGEON。

将角色 SURGEON 授予角色 DOCTOR 将形成循环，这是不允许的。

安全性管理员运行下列 SQL 语句来构建角色层次结构:

```
CREATE ROLE DOCTOR
CREATE ROLE SPECIALIST
CREATE ROLE SURGEON

GRANT ROLE DOCTOR TO ROLE SPECIALIST

GRANT ROLE SPECIALIST TO ROLE SURGEON
```

撤销角色的特权所产生的影响

撤销特权时，有时会导致从属数据库对象（例如，视图、程序包或触发器）变得无效或不可用。

下列示例说明在撤销授权标识的某些特权以及通过角色或其他方法拥有特权时对数据库对象产生的影响。

撤销角色的特权的示例

1. 安全性管理员创建角色 DEVELOPER 并对用户 BOB 授予此角色中的成员资格:

   ```
   CREATE ROLE DEVELOPER
   GRANT ROLE DEVELOPER TO USER BOB
   ```

2. 用户 ALICE 创建表 WORKITEM:

   ```
   CREATE TABLE WORKITEM (x int)
   ```

3. 数据库管理员将对表 WORKITEM 的 SELECT 和 INSERT 特权授予 PUBLIC 并同时授予角色 DEVELOPER:

   ```
   GRANT SELECT ON TABLE ALICE.WORKITEM TO PUBLIC
   GRANT INSERT ON TABLE ALICE.WORKITEM TO PUBLIC
   GRANT SELECT ON TABLE ALICE.WORKITEM TO ROLE DEVELOPER
   GRANT INSERT ON TABLE ALICE.WORKITEM TO ROLE DEVELOPER
   ```

4. 用户 BOB 创建一个使用表 WORKITEM 的视图 PROJECT 和基于表 WORKITEM 的程序包 PKG1:

   ```
   CREATE VIEW PROJECT AS SELECT * FROM ALICE.WORKITEM
   PREP emb001.sqc BINDFILE PACKAGE USING PKG1 VERSION 1
   ```

5. 如果数据库管理员撤销 PUBLIC 对表 ALICE.WORKITEM 的 SELECT 特权，由于视图定义者 BOB 通过其在角色 DEVELOPER 中的成员资格仍拥有必需的特权，所以视图 BOB.PROJECT 保持可用并且程序包 PKG1 仍有效:

   ```
   REVOKE SELECT ON TABLE ALICE.WORKITEM FROM PUBLIC
   ```

6. 如果数据库管理员撤销角色 DEVELOPER 对表 ALICE.WORKITEM 的 SELECT 特权，由于视图和程序包定义者 BOB 没有通过其他方法获得必需特权，所以视图 BOB.PROJECT 变得不可用并且程序包 PKG1 变得无效:

   ```
   REVOKE SELECT ON TABLE ALICE.WORKITEM FROM ROLE DEVELOPER
   ```

撤销 DBADM 权限的示例

在本示例中，角色 DEVELOPER 拥有 DBADM 权限并且已将该角色授予用户 BOB。

1. 安全性管理员创建角色 DEVELOPER:

   ```
   CREATE ROLE DEVELOPER
   ```

2. 系统管理员将 DBADM 权限授予角色 DEVELOPER:

   ```
   GRANT DBADM ON DATABASE TO ROLE DEVELOPER
   ```

3. 安全性管理员对用户 BOB 授予此角色中的成员资格:

   ```
   GRANT ROLE DEVELOPER TO USER BOB
   ```

4. 用户 ALICE 创建表 WORKITEM:

   ```
   CREATE TABLE WORKITEM (x int)
   ```

5. 用户 BOB 创建一个使用表 WORKITEM 的视图 PROJECT、基于表 WORKITEM 的程序包 PKG1 以及同样基于表 WORKITEM 的触发器 TRG1:

   ```
   CREATE VIEW PROJECT AS SELECT * FROM ALICE.WORKITEM
   PREP emb001.sqc BINDFILE PACKAGE USING PKG1 VERSION 1
   CREATE TRIGGER TRG1 AFTER DELETE ON ALICE.WORKITEM
                 FOR EACH STATEMENT MODE DB2SQL
          INSERT INTO ALICE.WORKITEM VALUES (1)
   ```

6. 安全性管理员撤销用户 BOB 的角色 DEVELOPER:

   ```
   REVOKE ROLE DEVELOPER FROM USER BOB
   ```

   撤销角色 DEVELOPER 会导致用户 BOB 失去 DBADM 权限，这是因为撤销了拥有该权限的角色。视图、程序包和触发器都将受到影响，如下所示:

   - 视图 BOB。PROJECT 仍有效。
   - 程序包 PKG1 变得无效。
   - 触发器 BOB.TRG1 仍有效。

   视图 BOB.PROJECT 和触发器 BOB.TRG1 可用，而程序包 PKG1 不可用。失去 DBADM 权限时，由拥有 DBADM 权限的授权标识创建的视图和触发器对象不受影响。

通过使用 WITH ADMIN OPTION 子句来委托角色维护

通过使用 GRANT（Role）SQL 语句的 WITH ADMIN OPTION 子句，安全性管理员可以将角色中成员资格的管理和控制权委托给其他人。

WITH ADMIN OPTION 子句使另一个用户有权将角色中的成员资格授予其他用户、撤销角色的其他成员在该角色中的成员资格以及注释但不删除该角色。

WITH ADMIN OPTION 子句不会使另一个用户有权将对角色的 WITH ADMIN OPTION 授予其他用户。它也不会使另一个用户有权撤销其他授权标识对角色的 WITH ADMIN OPTION。

说明 WITH ADMIN OPTION 子句的用法的示例

1. 安全性管理员创建角色 DEVELOPER 并使用 WITH ADMIN OPTION 子句将此新角色授予用户 BOB:

   ```
   CREATE ROLE DEVELOPER
   GRANT ROLE DEVELOPER TO USER BOB WITH ADMIN OPTION
   ```

2. 用户 BOB 可以将该角色中的成员资格授予其他用户（例如，ALICE）并撤销其他用户在该角色中的成员资格:

```
GRANT ROLE DEVELOPER TO USER ALICE
REVOKE ROLE DEVELOPER FROM USER ALICE
```

3. 用户 BOB 不能删除该角色或将 WITH ADMIN OPTION 授予另一个用户（只有安全性管理员可以执行这两个操作）。BOB 发出的这些命令将失败:

```
DROP ROLE DEVELOPER  - FAILURE!
                     - only a security administrator is allowed to drop the role
GRANT ROLE DEVELOPER TO USER ALICE WITH ADMIN OPTION - FAILURE!
                     - only a security administrator can grant WITH ADMIN OPTION
```

4. 由于用户 BOB 没有安全性管理员 (SECADM) 权限，所以他/她不能撤销角色 DEVELOPER 的用户的角色管理特权（由 WITH ADMIN OPTION 授予）。当 BOB 发出以下命令时，该命令将失败:

```
REVOKE ADMIN OPTION FOR ROLE DEVELOPER FROM USER SANJAY - FAILURE!
```

5. 安全性管理员可以撤销用户 BOB 对角色 DEVELOPER 的角色管理特权（由 WITH ADMIN OPTION 授予），并仍对用户 BOB 授予角色 DEVELOPER:

```
REVOKE ADMIN OPTION FOR ROLE DEVELOPER FROM USER BOB
```

此外，如果安全性管理员仅撤销用户 BOB 的角色 DEVELOPER，那么 BOB 将失去作为角色 DEVELOPER 的成员所拥有的所有特权以及通过 WITH ADMIN OPTION 子句拥有的对该角色的权限:

```
REVOKE ROLE DEVELOPER FROM USER BOB
```

比较角色与组

在创建视图、具体化查询表 (MQT)、SQL 例程、触发器和包含静态 SQL 的程序包时，将不考虑授予组的特权和权限。通过使用角色而不是组可避免此限制。

角色允许用户使用通过这些角色获取的特权来创建数据库对象，这些角色由 DB2 数据库系统控制。组和用户在 DB2 数据库系统外面控制，例如，由操作系统或 LDAP 服务器。

将使用的组替换为角色的示例

此示例显示如何使用角色来替换组。

假定有三个组 DEVELOPER_G、TESTER_G 和 SALES_G。用户 BOB、ALICE 和 TOM 是这些组的成员，如下表中所示:

表 7. 组和用户示例

组	属于此组的用户
DEVELOPER_G	BOB
TESTER_G	ALICE 和 TOM
SALES_G	ALICE 和 BOB

1. 安全性管理员创建要用来代替组的角色 DEVELOPER、TESTER 和 SALES。

```
CREATE ROLE DEVELOPER
CREATE ROLE TESTER
CREATE ROLE SALES
```

2. 安全性管理员将这些角色中的成员资格授予用户（设置组中的用户成员资格是系统管理员的职责）：

```
GRANT  ROLE DEVELOPER TO USER BOB
GRANT  ROLE TESTER TO USER ALICE, USER TOM
GRANT  ROLE SALES TO USER BOB, USER ALICE
```

3. 数据库管理员可以将组所拥有的类似特权或权限授予这些角色，例如:

```
GRANT privilege ON object TO ROLE DEVELOPER
```

然后，数据库管理员可以撤销组的这些特权，并要求系统管理员从系统中除去这些组。

使用通过角色获取的特权创建触发器的示例

此示例显示当用户 BOB 通过角色 DEVELOPER 拥有必需特权时，该用户可以成功创建触发器 TRG1。

1. 首先，用户 ALICE 创建表 WORKITEM:

```
CREATE TABLE WORKITEM (x int)
```

2. 然后，由数据库管理员将改变 ALICE 的表的特权授予角色 DEVELOPER:

```
GRANT ALTER ON ALICE.WORKITEM TO ROLE DEVELOPER
```

3. 由于用户 BOB 是角色 DEVELOPER 的成员，所以他/她成功创建触发器 TRG1。

```
CREATE TRIGGER TRG1 AFTER DELETE ON ALICE.WORKITEM
          FOR EACH STATEMENT MODE DB2SQL INSERT INTO ALICE.WORKITEM VALUES (1)
```

在从 IBM Informix Dynamic Server 迁移后使用角色

如果已从 IBM Informix® Dynamic Server 迁移至 DB2 数据库系统并且正在使用角色，那么您需要了解一些事项。

Informix Dynamic Server（IDS）SQL 语句 GRANT ROLE 提供子句 WITH GRANT OPTION。DB2 数据库系统的 GRANT ROLE 语句提供具有相同功能的子句 WITH ADMIN OPTION（该子句符合 SQL 标准）。在从 IDS 迁移至 DB2 数据库系统期间，在 **dbschema** 工具生成 CREATE ROLE 和 GRANT ROLE 语句后，**dbschema** 工具会将出现的任何 WITH GRANT OPTION 替换为 WITH ADMIN OPTION。

在 IDS 数据库系统中，SET ROLE 语句激活特定角色。DB2 数据库系统支持 SET ROLE 语句，但只是为了与使用该 SQL 语句的其他产品兼容。SET ROLE 语句检查会话用户是否是角色的成员，如果不是，那么它将返回错误。

dbschema 输出示例

假定 IDS 数据库包含角色 DEVELOPER、TESTER 和 SALES。为用户 BOB、ALICE 和 TOM 授予了不同的角色；将角色 DEVELOPER 授予 BOB、将角色 TESTER 授予 ALICE，并将角色 TESTER 和 SALES 授予 TOM。要迁移至 DB2 数据库系统，请使用 **dbschema** 工具为数据库生成 CREATE ROLE 和 GRANT ROLE 语句:

```
CREATE ROLE DEVELOPER
CREATE ROLE TESTER
CREATE ROLE SALES

GRANT  DEVELOPER TO BOB
GRANT  TESTER TO ALICE, TOM
GRANT  SALES TO TOM
```

必须先在 DB2 数据库系统中创建数据库，然后才可以在该数据库中运行上述语句，以重新创建角色并分配角色。

第 3 章 使用可信上下文和可信连接

在建立与 DB2 数据库的连接时，通过在应用程序内发出请求可以建立显式可信连接。安全性管理员先前必须已使用 CREATE TRUSTED CONTEXT 语句以及与要建立的连接的属性匹配的那些属性来定义可信上下文（请参阅后面的步骤 1）。

开始之前

建立连接时用来请求显式可信连接的 API 取决于使用的应用程序类型（请参阅步骤 2 中的表）。

建立显式可信连接之后，应用程序可以通过使用适用于该类型的应用程序的 API（请参阅步骤 3 中的表）将连接的用户标识切换至另一个用户标识。

过程

1. 安全性管理员通过使用 CREATE TRUSTED CONTEXT 语句在服务器中定义可信上下文。 例如:

```
CREATE TRUSTED CONTEXT MYTCX
       BASED UPON CONNECTION USING SYSTEM AUTHID NEWTON
       ATTRIBUTES (ADDRESS   '192.0.2.1')
       WITH USE FOR PUBLIC WITHOUT AUTHENTICATION
       ENABLE
```

2. 要建立可信连接，请使用应用程序中的下列其中一个 API:

选项	描述
应用程序	**API**
CLI/ODBC	SQLConnect 和 SQLSetConnectAttr
XA CLI/ODBC	Xa_open
JAVA	getDB2TrustedPooledConnection 和 getDB2TrustedXAConnection

3. 要切换至另一个经过认证或未经过认证的用户，请使用应用程序中的下列其中一个 API:

选项	描述
应用程序	**API**
CLI/ODBC	SQLSetConnectAttr
XA CLI/ODBC	SQLSetConnectAttr
JAVA	getDB2Connection 和 reuseDB2Connection
.NET	DB2Connection.ConnectionString 关键字: TrustedContextSystemUserID 和 TrustedContextSystemPassword

进行切换时是否对新用户标识进行认证取决于与该显式可信连接相关联的可信上下文对象的定义。例如，假设安全性管理员创建了以下可信上下文对象:

```
CREATE TRUSTED CONTEXT CTX1
 BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION
  ENABLE
```

进一步假定建立了显式可信连接。由于 USER3 已定义为不需要对其进行认证的可信上下文 CTX1 的用户，所以允许在不提供认证信息的情况下将可信连接上的用户标识切换至 USER3 的请求。但是，由于 USER2 已定义为必须提供其认证信息的可信上下文 CTX1 的用户，所以在未提供认证信息的情况下将可信连接上的用户标识切换至 USER2 的请求将失败。

建立显式可信连接并切换用户的示例

在以下示例中，中间层服务器需要代表最终用户发出一些数据库请求，但不必访问最终用户的凭证以代表该最终用户建立数据库连接。

可以在数据库服务器上创建一个可信上下文对象，该对象允许中间层服务器建立与数据库的显式可信连接。在建立显式可信连接后，中间层服务器可以将该连接的当前用户标识切换至新的用户标识，而不需要在数据库服务器中认证新的用户标识。以下 CLI 代码段说明了如何使用在前面的步骤 1 中定义的可信上下文 MYTCX 来建立可信连接，以及如何切换至可信连接上未经过认证的用户。

```
int main(int argc, char *argv[])
{
 SQLHANDLE henv;         /* environment handle */
 SQLHANDLE hdbc1;       /* connection handle */
 char origUserid[10] = "newton";
  char password[10] = "test";
   char switchUserid[10] = "zurbie";
 char dbName[10] = "testdb";

// Allocate the handles
SQLAllocHandle( SQL_HANDLE_ENV, &henv );
SQLAllocHandle( SQL_HANDLE_DBC, &hdbc1 );

// Set the trusted connection attribute
SQLSetConnectAttr( hdbc1, SQL_ATTR_USE_TRUSTED_CONTEXT,
SQL_TRUE, SQL_IS_INTEGER );

// Establish a trusted connection
SQLConnect( hdbc1, dbName, SQL_NTS, origUserid, SQL_NTS,
password, SQL_NTS );

//Perform some work under user ID "newton"
. . . . . .

 // Commit the work
SQLEndTran(SQL_HANDLE_DBC, hdbc1, SQL_COMMIT);


// Switch the user ID on the trusted connection
SQLSetConnectAttr( hdbc1,
SQL_ATTR_TRUSTED_CONTEXT_USERID, switchUserid,
SQL_IS_POINTER
);

//Perform new work using user ID "zurbie"
. . . . . .

//Commit the work
```

```
SQLEndTranSQL_HANDLE_DBC, hdbc1, SQL_COMMIT);

// Disconnect from database
SQLDisconnect( hdbc1 );

  return 0;

}  /* end of main */
```

下一步做什么

何时真正切换用户标识?

在发出用于切换可信连接上的用户的命令之后，并不会立即执行切换用户请求，要等到将下一个语句发送至服务器之后才会执行该请求。以下示例对这种情况进行了说明。要等到发出下一个语句之后，**list applications** 命令才会显示最初的用户标识。

1. 使用 USERID1 建立显式可信连接。
2. 对 USERID2 发出切换用户命令，例如 **getDB2Connection**。
3. 运行 db2 list applications。它仍将显示已连接 USERID1。
4. 在可信连接上发出一个语句（例如，executeQuery("values current sqlid")）以在服务器中执行切换用户请求。
5. 再次运行 db2 list applications。现在，此命令就会显示已连接 USERID2。

可信上下文和可信连接

可信上下文是一个数据库对象，它为数据库与外部实体（例如，应用程序服务器）之间的连接定义信任关系。

信任关系基于下列一组属性:

- 系统授权标识: 表示建立数据库连接的用户
- IP 地址（或域名）: 表示在其中建立数据库连接的主机
- 数据流加密: 表示用于数据库服务器与数据库客户机之间的数据通信的加密设置（如果此设置存在）

用户建立数据库连接时，DB2 数据库系统将检查该连接是否与数据库中的可信上下文对象的定义匹配。如果匹配，那么认为该数据库连接是可信连接。

可信连接允许此可信连接的发起方获取在可信连接作用域外可能不可用的其他功能。根据可信连接是显式连接还是隐式连接，这些功能会有所不同。

显式可信连接的发起方可以:

- 将连接上的当前用户标识切换至另一个经过认证或未经过认证的用户标识
- 通过可信上下文的角色继承功能获取其他特权

隐式可信连接是非显式请求的可信连接；隐式可信连接由正常连接请求而不是显式可信连接请求产生。获取隐式连接不需要更改应用程序代码。此外，您是否获得隐式可信连接不会影响连接返回码（请求显式可信连接时，连接返回码指示请求是否成功）。隐式可信连接的发起方只能通过可信上下文的角色继承功能获取其他特权；他们不能切换用户标识。

使用可信上下文如何增强安全性

三层应用程序模型通过在客户机应用程序与数据库服务器之间加入一个中间层来扩展标准两层客户机和服务器模型。该模型在最近几年非常流行，特别是在基于 Web 的技术和 Java 2 Enterprise Edition（J2EE）平台出现后。支持三层应用程序模型的软件产品示例有 IBM WebSphere® Application Server（WAS）。

在三层应用程序模型中，中间层负责认证运行客户机应用程序的用户并管理与数据库服务器的交互。传统上，所有与数据库服务器的交互都是通过中间层建立的数据库连接进行的，中间层在建立该连接时使用用户标识和凭证的组合向数据库服务器标识它自己。这意味着，数据库服务器使用与中间层用户标识关联的数据库特权来执行在进行任何数据库访问（包括中间层代表用户执行的访问）时都必须执行的所有授权检查和审计。

虽然三层应用程序模型具有许多优点，但让所有与数据库服务器的交互（例如，用户请求）都在中间层的授权标识基础上进行会产生一些安全问题，下面概述了这些问题:

- 失去用户标识

  某些企业更愿意知道访问数据库的实际用户的标识，以便进行访问控制。
- 减轻用户责任

  通过审计确保责任是数据库安全性中的一个基本原则。不知道用户标识很难将中间层为自己目的而执行的事务与代表用户执行的事务区分开来。
- 授予中间层授权标识过多特权

  中间层授权标识必须具有执行用户发出的所有请求所需的所有特权。这存在一个安全性问题，它使不需要访问某些信息的用户始终能够获取访问权。
- 降低安全性

  除上一点中提及的特权问题外，当前方法还要求必须授予中间层连接时使用的授权标识对用户请求可能访问的所有资源的特权。一旦该中间层授权标识泄露，那么所有这些资源都会暴露。
- 在同一连接的用户之间“溢出”

  先前用户所作的更改可能影响当前用户。

很明显，我们需要一种机制，通过该机制将实际用户的标识和数据库特权用于中间层代表该用户执行的数据库请求。实现此目标的最直接方法是让中间层使用用户的标识和密码建立新连接，然后通过该连接定向用户的请求。此方法虽然简单，但却具有下列缺点:

- 不适用于某些中间层。许多中间层服务器没有建立连接所需的用户认证凭证。
- 性能开销。在数据库服务器中创建新的物理连接并重新认证用户很显然会产生性能开销。
- 维护开销。在未使用集中安全性设置或未使用单点登录的情况下，具有两个用户定义（一个在中间层中，一个在服务器中）会产生维护开销。这需要在不同位置更改密码。

可信上下文功能可以解决此问题。安全性管理员可以在数据库中创建一个可信上下文对象，用来定义数据库与中间层之间的信任关系。然后，中间层可以建立与数据库的

显式可信连接，从而使中间层具有将该连接上的当前用户标识切换至另一个经过认证或未经过认证的用户标识的功能。除了解决最终用户身份声明问题外，可信上下文还具有另一个优点。即，控制数据库用户何时具有特权的功能。缺少这种控制功能可能会降低整体安全性。例如，特权可能会用于与最初意图不同的目的。安全性管理员可以为一个角色指定一种或多种特权，并将该角色指定给可信上下文对象。只有与该可信上下文定义匹配的显式或隐式可信数据库连接才能利用与该角色关联的特权。

提高性能

由于可信连接具有下列优点，在使用该连接时可以使性能最高:

- 切换连接的当前用户标识时不建立新连接。
- 如果可信上下文定义不需要认证切换至的用户标识，那么不会产生与在数据库服务器中认证新用户相关的开销。

创建可信上下文的示例

假定安全性管理员创建了以下可信上下文对象:

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER2
  ATTRIBUTES (ADDRESS   '192.0.2.1')
  DEFAULT ROLE managerRole
   ENABLE
```

如果用户 *user1* 从 IP 地址 192.0.2.1 请求可信连接，那么 DB2 数据库系统将返回一个警告（SQLSTATE 01679 和 SQLCODE +20360）以指示未能建立可信连接，该用户 *user1* 仅获得不可信连接。但是，如果用户 *user2* 从 IP 地址 192.0.2.1 请求可信连接，由于可信上下文 CTX1 满足连接属性，所以将实现该请求。既然用户 *user2* 建立了可信连接，那么他/她现在可以获取与可信上下文角色 managerRole 关联的所有特权和权限。这些特权和权限可能对此可信连接作用域外的用户 *user2* 不可用。

通过可信上下文继承角色成员资格

可信连接的当前用户可以通过可信上下文来自动继承角色以获取其他特权，但前提是此角色必须已由安全性管理员指定为相关可信上下文定义的一部分。

缺省情况下，可信连接的所有用户都可以继承角色。安全性管理员还可以使用可信上下文定义来指定供特定用户继承的角色。

在可信连接期间，会话授权标识可以拥有的活动角色包括:

- 其会话授权标识通常被视为成员的角色，以及
- 可信上下文缺省角色或可信上下文特定于用户的角色（如果定义了它们）

注:

- 如果构建了一个定制安全插件以便在成功连接时由该安全插件产生的系统授权标识和会话授权标识互不相同，并且使用该安全插件来配置用户认证，那么不能通过该连接继承可信上下文角色，即使该连接是可信连接亦如此。
- 通过角色获取的可信上下文特权仅对动态 DML 操作有效。它们对下列各项无效:
  - DDL 操作
  - 非动态 SQL（涉及静态 SQL 语句的操作，例如，BIND、REBIND、隐式重新绑定、增量绑定等）

获取可信上下文特定于用户的特权

安全性管理员可以使用可信上下文定义来使角色与可信上下文关联，以便:

- 缺省情况下可信连接的所有用户都可以继承指定角色
- 可信连接的特定用户可以继承指定角色

当可信连接上的用户切换至一个新的授权标识并且这个新授权标识存在可信上下文特定于用户的角色时，如果存在可信上下文缺省角色，那么该特定于用户的角色将覆盖缺省角色，如示例中所示。

创建指定缺省角色和特定于用户的角色的可信上下文示例

假定安全性管理员创建了以下可信上下文对象:

```
CREATE TRUSTED CONTEXT CTX1
  BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION
  DEFAULT ROLE AUDITOR
  ENABLE
```

当 USER1 建立可信连接时，授予角色 AUDITOR 的特权将由此授权标识继承。同样，当可信连接上的当前授权标识切换至 USER3 的用户标识时，这些特权还将由 USER3 继承。（如果该连接的用户标识在某个时间切换至 USER2，那么 USER2 也将继承可信上下文缺省角色 AUDITOR。）安全性管理员可以选择让 USER3 继承除可信上下文缺省角色外的另一个角色。他们可以通过将特定角色指定给此用户来实现此目的，如下所示:

```
CREATE TRUSTED CONTEXT CTX1
 BASED UPON CONNECTION USING SYSTEM AUTHID USER1
 ATTRIBUTES (ADDRESS   '192.0.2.1')
  WITH USE FOR USER2 WITH AUTHENTICATION,
               USER3 WITHOUT AUTHENTICATION ROLE OTHER_ROLE
  DEFAULT ROLE AUDITOR
  ENABLE
```

当可信连接上的当前用户标识切换至 USER3，此用户不再继承可信上下文缺省角色。相反，他们将继承由安全性管理员指定给他/她的特定角色 OTHER_ROLE。

关于在显式可信连接上切换用户标识的规则

在显式可信连接上，可以将连接的用户标识切换至另一个用户标识。但是应遵循某些规则。

1. 如果切换请求不是从显式可信连接发出的，并且该切换请求将发送至服务器以进行处理，那么连接将关闭并返回一条错误消息（SQLSTATE 08001 和原因码为 41 的 SQLCODE -30082）。
2. 如果切换请求不是在事务边界发出的、事务已回滚并且该切换请求将发送至服务器以进行处理，那么将使连接处于未连接状态并返回一条错误消息（SQLSTATE 58009 和 SQLCODE -30020）。
3. 如果切换请求是从存储过程发出的，那么将返回一条错误消息（SQLCODE -30090 和原因码 29），它指示此环境中的操作非法。将保持连接状态并且不会使连接处于未连接状态。可以处理后续请求。

4. 如果切换请求在实例连接（而不是数据库连接）中传递，那么该连接将关闭并返回一条错误消息 (SQLCODE -30005)。
5. 如果切换请求是使用可信连接上不允许的授权标识发出的，那么将返回错误（SQLSTATE 42517 和 SQLCODE -20361）并使连接处于未连接状态。
6. 如果切换请求是使用可信连接上允许的经过认证的授权标识发出的，但未提供相应的认证令牌，那么将返回错误（SQLSTATE 42517 和 SQLCODE -20361）并使连接处于未连接状态。
7. 如果与可信连接关联的可信上下文对象被禁用，并且对该可信连接发出了切换请求，那么将返回错误（SQLSTATE 42517 和 SQLCODE -20361）并使连接处于未连接状态。

   在此情况下，唯一接受的切换用户请求是指定建立了可信连接的用户标识或空用户标识的请求。如果切换至建立了可信连接的用户标识，那么此用户标识将不继承任何可信上下文角色（包括可信上下文缺省角色和可信上下文特定于用户的角色）。
8. 如果更改了与可信连接关联的可信上下文对象的系统授权标识属性，并且对该可信连接发出了切换请求，那么将返回错误（SQLSTATE 42517 和 SQLCODE -20361）并使连接处于未连接状态。

   在此情况下，唯一接受的切换用户请求是指定建立了可信连接的用户标识或空用户标识的请求。如果切换至建立了可信连接的用户标识，那么此用户标识将不继承任何可信上下文角色（包括可信上下文缺省角色和可信上下文特定于用户的角色）。
9. 如果删除了与可信连接关联的可信上下文对象，并且对该可信连接发出了切换请求，那么将返回错误（SQLSTATE 42517 和 SQLCODE -20361）并使连接处于未连接状态。

   在此情况下，唯一接受的切换用户请求是指定建立了可信连接的用户标识或空用户标识的请求。如果切换至建立了可信连接的用户标识，那么此用户标识将不继承任何可信上下文角色（包括可信上下文缺省角色和可信上下文特定于用户的角色）。
10. 如果切换请求是使用可信连接上允许的用户标识发出的，但该用户标识没有对数据库的 CONNECT 特权，那么将使连接处于未连接状态并返回一条错误消息（SQLSTATE 08004 和 SQLCODE -1060）。
11. 如果可信上下文系统授权标识出现在 WITH USE FOR 子句中，那么 DB2 数据库系统将使用切换用户请求中的系统授权标识的认证设置来切换回系统授权标识。如果可信上下文系统授权标识未出现在 WITH USE FOR 子句中，那么始终允许切换回系统授权标识的切换用户请求，即使该系统授权标识未经过认证亦如此。

注: 使连接处于未连接状态时，唯一接受并且不会导致返回错误“应用程序状态出错。数据库连接不存在。”(SQLCODE -900)的请求包括:

- 切换用户请求
- COMMIT 或 ROLLBACK 语句
- DISCONNECT、CONNECT RESET 或 CONNECT 请求

注: 当可信连接上的用户标识切换至新的用户标识时，旧用户在连接环境中的所有痕迹都将消失。也就是说，切换用户标识将产生全新的连接环境。例如，如果连接上的旧用户打开了任何临时表或 WITH HOLD 游标，那么当该连接上的用户标识切换至新用户标识时，这些对象将彻底丢失。

注: Java 可信连接不具有未连接状态。如果切换用户操作失败，那么 Java 将抛出异常并且此连接会断开。

可信上下文问题确定

显式可信连接是由可信连接的特定显式请求成功建立的连接。如果您请求显式可信连接，但您没有资格获得该连接，那么您将获得一般连接和一个警告（+20360）。

为了确定用户无法建立可信连接的原因，安全性管理员需要查看系统目录和连接属性中的可信上下文定义。特别是，要查看建立连接的 IP 地址、数据流或网络的加密级别以及建立连接的系统授权标识。**db2pd** 实用程序的 **-application** 选项将返回这些信息和下列附加信息:

- 连接信任类型: 指示连接是否受信任。连接受信任时，它还指出此连接是显式可信连接还是隐式可信连接。
- 可信上下文名称: 与可信连接关联的可信上下文的名称。
- 继承的角色: 通过可信连接继承的角色。

以下是未能获得显式可信连接的最常见原因:

- 客户机应用程序未使用 TCP/IP 来与 DB2 服务器通信。客户机应用程序与可用来建立显式或隐式可信连接的 DB2 服务器通信时，唯一受支持的协议是 TCP/IP。
- 数据库服务器认证类型设置为 CLIENT。
- 数据库服务器没有已启用的可信上下文对象。可信上下文对象的定义必须明确指出 ENABLE，以使该可信上下文被认为与入局连接的属性相匹配。
- 数据库服务器上的可信上下文对象与提供的可信属性不匹配。例如，可能存在下列情况之一:
  - 连接的系统授权标识与任何可信上下文对象的系统授权标识都不匹配。
  - 创建连接的 IP 地址与打算用于连接的可信上下文对象中的任何 IP 地址都不匹配。
  - 连接所使用的数据流加密方法与打算用于连接的可信上下文对象中 ENCRYPTION 属性的值不匹配。

  可以使用 **db2pd** 工具来了解建立连接的 IP 地址、连接所使用的数据流或网络的加密级别以及建立连接的系统授权标识。可以参考 SYSCAT.CONTEXTS 和 SYSCAT.CONTEXTATTRIBUTES 目录视图以了解特定可信上下文对象的定义，例如，了解它的系统授权标识、允许的 IP 地址的设置以及 ENCRYPTION 属性的值。

以下是发生切换用户故障的最常见原因:

- 要切换至的用户标识对数据库没有 CONNECT 特权。在此情况下，将返回 SQL1060N。
- 在与显式可信连接相关联的可信上下文对象的 WITH USE FOR 子句中未定义要切换至的用户标识或 PUBLIC。
- 允许在经过认证之后切换用户，但是用户未提供凭证或者提供了错误的凭证。
- 在事务边界上未发出切换用户请求。

- 与可信连接相关联的可信上下文已经被禁用、删除或改变。在此情况下，只允许切换至建立了可信连接的用户标识。

第 4 章 行和列访问控制 (RCAC) 概述

DB2 V10.1 引入了行和列访问控制 (RCAC) 作为数据安全性的其他层。行和列访问控制 (RCAC) 有时称为细颗粒度访问控制或 FGAC。RCAC 在行级别和/或列级别控制对表的访问。可以使用 RCAC 来补充表特权模型。

为了遵循各种政府法规，您可以实现过程和方法来确保信息受到充分的保护。组织中的个人只允许访问他们执行工作任务所需的数据子集。例如，您的领域的政府法规可能规定，医生有权查看其治疗的患者的病历卡，但不能查看其他患者的病历卡。相同的法规还可能规定，除非患者同意，否则不允许卫生保健提供者访问患者的个人信息（例如，患者的家庭电话号码）。

可以使用行和列访问控制来确保用户只能访问他们的工作所需的数据。例如，运行 DB2 for Linux, UNIX, and Windows 和 RCAC 的医院系统可以过滤患者信息和数据以仅包括特定医生所需的数据。在医生关心之前，其他患者不存在。同样，当患者服务代表查询同一医院的患者表时，他们能够查看患者姓名和电话号码列，但病历列对他们是隐藏的。如果数据是隐藏的，那么将显示 NULL 或备用值，而不是实际的病历。

行和列访问控制或 RCAC 具有下列优点:

- 不免除任何数据库用户是行和列访问控制规则固有的特性。

  这些规则不会免除甚至具有更高级别权限的用户（例如，具有 DATAACCESS 权限的用户）。仅具有安全性管理员 (SECADM) 权限的用户才能管理数据库中的行和列访问控制。因此，您可以使用 RCAC 来阻止具有 DATAACCESS 权限的用户自由访问数据库中的所有数据。
- 不管如何通过 SQL 来访问表，表数据都受保护。

  应用程序、临时提供的查询工具和报告生成工具都遵循 RCAC 规则。强制执行将以数据为中心。
- 任何应用程序更改都不需要使用数据安全性的这个附加层。

  即，将以现有应用程序不知道的方式建立和定义行和列级别访问控制。然而，RCAC 在范例方面有一个重要的改变，即重点不再是请求了什么内容，而在于谁在请求这些内容。相同查询的结果集将根据执行查询所在的上下文而更改，并且不会返回警告或错误。此行为是该解决方案的真正意图。这意味着应用程序设计人员和 DBA 必须知道，查询并非可以查找表中的所有数据，除非授予了特定许可权执行这样的操作。

行和列访问控制 (RCAC) 规则

行和列访问控制 (RCAC) 在表级别对数据本身进行访问控制。对行和列创建的 SQL 规则是实现此功能的基础。

行和列访问控制是安全性管理员用于管理隐私和安全性策略的访问控制模型。RCAC 允许所有用户访问同一个表，而不是表的备用视图。然而，RCAC 将根据与表相关联的策略所指定的各个用户许可权或规则来限制对表的访问。存在两组规则，一组适用于行，另一组适用于列。

- 行许可权
  - 行许可权是数据库对象，它表示特定表的行访问控制规则。
  - 行访问控制规则是 SQL 搜索条件，它描述用户可以访问的行集。
- 列掩码
  - 列掩码是数据库对象，它表示表中特定列的列访问控制规则。
  - 列访问控制规则是 SQL CASE 表达式，它描述用户有权查看的列值以及要满足的条件。

用于管理 RCAC 规则的 SQL 语句

通过使用下列 SQL 语句，可以创建、改变和删除 RCAC 规则。

用于管理 RCAC 许可权和掩码的内置函数

使用下列内置标量函数来表示许可权和掩码中的条件。例如，用户必须属于一个或多个角色或者属于一个或多个组才能访问特定行。

方案: 使用行和列访问控制的 ExampleHMO

此方案将 ExampleHMO（一个具有大量正在治疗的患者的国家组织）表示为行和列访问控制的用户。ExampleHMO 使用行和列访问控制来确保其数据库策略反映了政府法规对于隐私和安全性的要求以及管理业务目标。

处理患者健康状态信息及其个人信息的组织（如 ExampleHMO）必须遵守政府的隐私和数据保护法规（例如，健康保险可移植性和责任法案 (HIPAA)）。这些隐私和数据保护法规确保共享、查看和修改任何敏感的患者医疗信息或个人信息的操作只能由有特权执行这些操作的权力机构执行。对该法案的任何违反将导致巨大的处罚，包括民事和刑事诉讼。

ExampleHMO 必须确保其数据库系统中存储的数据是安全的并且只能特权用户才能访问这些数据。按照典型的隐私法规，某些患者信息只能由特权用户访问和修改。

方案: 使用行和列访问控制的 ExampleHMO - 安全策略

ExampleHMO 按照某些安全策略来实现对 DB2 数据库进行数据访问要遵循的安全策略。

这些安全策略符合政府的隐私和数据保护法规。第一列概述了这些策略和组织所面对的难题，第二列概述了解决难题的 DB2 行和列访问控制功能部件。

安全性难题	解决安全性难题的行和列访问控制功能部件
将列访问权限于仅对特权用户提供。 例如，Jane 是合作伙伴公司的药品研究员，将不允许她查看敏感的患者医疗信息或个人数据（如患者的保险号）。	可以使用列掩码来过滤敏感数据或对 Jane 隐藏这些数据。
将行访问权限于仅对特权用户提供。仅允许 Dr. Lee 查看他自己的患者的患者信息，并非允许他查看 ExampleHMO 系统中的所有患者。	可以实现行许可权以控制哪些用户可以查看任何特定行。

安全性难题	解决安全性难题的行和列访问控制功能部件
在“有需要才能知道”基础上限制数据。	行许可权加上在用户级别限制表级别数据有助于解决此难题。
限制其他数据库对象，如有关 RCAC 保密数据的 UDF、触发器和视图。	行和列访问控制在数据级别保护数据。正是行和列访问控制解决方案的这个以数据为中心的特性，来对甚至是 UDF、触发器和视图之类的数据库对象强制执行安全策略。

方案: 使用行和列访问控制的 ExampleHMO - 数据库用户和角色

在此方案中，许多不同的人创建、保护和使用 ExampleHMO 数据。这些人具有不同的用户权限和数据库权限。

ExampleHMO 实现了其安全策略以将从数据库访问数据的方式进行分类。对数据的内部和外部访问将基于访问数据的用户的职责分离以及这些用户的数据访问特权。ExampleHMO 创建了下列数据库角色来分离这些职责:

PCP
表示主治医生。

DRUG_RESEARCH
表示研究员。

ACCOUNTING
表示会计。

MEMBERSHIP
表示添加决定参加和决定不参加的患者的成员。

PATIENT
表示患者。

下列人员将创建、保护和使用 ExampleHMO 数据:

Alex
ExampleHMO 主安全性管理员。他拥有 SECADM 权限。

Peter
ExampleHMO 数据库管理员。他拥有 DBADM 权限。

Paul
ExampleHMO 数据库开发者。他具有创建触发器和用户定义的函数的特权。

Dr. Lee
ExampleHMO 医生。他属于 PCP 角色。

Jane
Innovative Pharmaceutical Company（ExampleHMO 的合作伙伴）的药品研究员。她属于 DRUG_RESEARCH 角色。

John
ExampleHMO 会计部。他属于 ACCOUNTING 角色。

Tom
ExampleHMO 成员资格管理人员。他属于 MEMBERSHIP 角色。

Bob
ExampleHMO 患者。他属于 PATIENT 角色。

如果您要尝试此方案中提供的任何示例 SQL 语句和命令，请创建这些用户标识并对他们授予所列示的权限。

下列示例 SQL 语句假定系统中已创建这些用户。这些 SQL 语句创建每个角色并给这些用户授予对 ExampleHMO 数据库中的各种表的 **SELECT** 和 **INSERT** 许可权:

```
--Creating roles and granting authority

CREATE ROLE PCP;

CREATE ROLE DRUG_RESEARCH;

CREATE ROLE ACCOUNTING;

CREATE ROLE MEMBERSHIP;

CREATE ROLE PATIENT;

GRANT ROLE PCP TO USER LEE;
GRANT ROLE DRUG_RESEARCH TO USER JANE;
GRANT ROLE ACCOUNTING  TO USER JOHN;
GRANT ROLE MEMBERSHIP TO USER TOM;
GRANT ROLE PATIENT TO USER BOB;
```

方案: 使用行和列访问控制的 ExampleHMO - 数据库表

此方案重点介绍 ExampleHMO 数据库中的两个表: PATIENT 表和 PATIENTCHOICE 表。

PATIENT 表存储基本的患者信息和健康状态信息。此方案将考虑 PATIENT 表中的以下各列:

SSN
患者的保险号。患者的保险号将认为是个人信息。

NAME
患者的姓名。患者的姓名将认为是个人信息。

ADDRESS
患者的地址。患者的地址将认为是个人信息。

USERID
患者的数据库标识。

PHARMACY
患者的医疗信息。

ACCT_BALANCE
患者的帐单信息。

PCP_ID
患者的主治医生数据库标识。

PATIENTCHOICE 表存储各个患者的决定参加和决定不参加信息，该信息确定患者是否同意对外人公开其健康状态信息以用于此表中的研究目的。此方案将考虑 PATIENTCHOICE 表中的以下各列:

SSN
患者的保险号用于使患者与其选项相匹配。

CHOICE
患者可以选择的某个选项的名称。

VALUE
患者作出的有关该选项的决定。

例如，行 123-45-6789, drug_research, opt-in 表示 SSN 为 123-45-6789 的患者同意公开其信息以用于医学研究。

下列示例 SQL 语句创建 PATIENT、PATIENTCHOICE 和 ACCT_HISTORY 表。将授予对表的权限并且将插入数据。

```
--Patient table storing information regarding patient
CREATE TABLE PATIENT (
 SSN CHAR(11),
 USERID VARCHAR(18),
 NAME VARCHAR(128),
 ADDRESS VARCHAR(128),
 PHARMACY VARCHAR(250),
 ACCT_BALANCE DECIMAL(12,2) WITH DEFAULT,
 PCP_ID VARCHAR(18)
 );


--Patientchoice table which stores what patient opts
--to expose regarding his health information

CREATE TABLE PATIENTCHOICE (
 SSN CHAR(11),
 CHOICE VARCHAR(128),
 VALUE VARCHAR(128)
);

--Log table to track account balance
CREATE TABLE ACCT_HISTORY(
 SSN   CHAR(11),
 BEFORE_BALANCE DECIMAL(12,2),
 AFTER_BALANCE DECIMAL(12,2),
 WHEN  DATE,
 BY_WHO VARCHAR(20)
);


--Grant authority

GRANT SELECT, UPDATE ON TABLE PATIENT TO ROLE PCP;

GRANT SELECT ON TABLE PATIENT TO ROLE DRUG_RESEARCH;

GRANT SELECT, UPDATE ON TABLE PATIENT TO ROLE ACCOUNTING;
GRANT SELECT ON TABLE ACCT_HISTORY TO ROLE ACCOUNTING;

GRANT SELECT, UPDATE, INSERT ON TABLE PATIENT TO ROLE MEMBERSHIP;
GRANT INSERT ON TABLE PATIENTCHOICE TO ROLE MEMBERSHIP;

GRANT SELECT ON TABLE PATIENT TO ROLE PATIENT;

GRANT SELECT, ALTER ON TABLE PATIENT TO USER ALEX;

GRANT ALTER, SELECT ON TABLE PATIENT TO USER PAUL;
GRANT INSERT ON TABLE ACCT_HISTORY TO USER PAUL;
```

```
--Insert patient data

INSERT INTO PATIENT VALUES('123-55-1234', 'MAX', 'Max', 'First Strt',
'hypertension', 89.70,'LEE');
INSERT INTO PATIENTCHOICE VALUES('123-55-1234', 'drug-research',
'opt-out');


INSERT INTO PATIENT VALUES('123-58-9812', 'MIKE', 'Mike', 'Long Strt',
null, 8.30,'JAMES');
INSERT INTO PATIENTCHOICE VALUES('123-58-9812', 'drug-research', 'opt-out');


INSERT INTO PATIENT VALUES('123-11-9856', 'SAM', 'Sam', 'Big Strt', null,
0.00,'LEE');
INSERT INTO PATIENTCHOICE VALUES('123-11-9856', 'drug-research', 'opt-in');


INSERT INTO PATIENT VALUES('123-19-1454', 'DUG', 'Dug', 'Good Strt', null,
0.00,'JAMES');
INSERT INTO PATIENTCHOICE VALUES('123-19-1454', 'drug-research', 'opt-in');
```

方案: 使用行和列访问控制的 ExampleHMO - 安全性管理

安全性管理和安全性管理员 (SECADM) 角色在保护 ExampleHMO 中的患者和公司数据方面有很重要的作用。在 ExampleHMO 中，管理决定了不同的人拥有数据库管理权限和安全性管理权限。

ExampleHMO 的管理团队决定创建角色来管理对其数据的访问。该团队还决定，缺省情况下，即使具有 DATAACCESS 权限的用户也无法查看受保护的健康状态数据和个人数据。

管理团队选择 Alex 作为 ExampleHMO 的唯一安全性管理员。从现在开始，Alex 将控制所有数据访问权限。借助此权限，Alex 定义安全性规则，例如，行许可权、列掩码以及函数和触发器是否安全。这些规则控制哪些用户可以有控制地访问任何给定数据。

在 Peter（数据库管理员）创建必需的表并建立必需的角色之后，将分离职责。将通过使 Alex 成为安全性管理员来分离数据库管理职责和安全性管理职责。

Peter 连接至数据库并对 Alex 授予 SECADM 权限。因为 Peter 当前拥有 DBADM、DATAACCESS 和 SECADM 权限，所以他可以授予 SECADM 权限。

```
-- To seperate duties of security administrator from system administrator,
-- the SECADMN Peter grants SECADM authority to user Alex.

GRANT SECADM ON DATABASE TO USER ALEX;
```

Alex 在获得 SECADM 权限之后连接至数据库并撤销 Peter 的安全性管理员特权。现在已分离这些职责，并且 Alex 成为唯一有权对 ExampleHMO 内部和外部的其他人授予数据访问权的人。以下 SQL 语句显示了 Alex 如何撤销 Peter 的 SECADM 权限:

```
--revokes the SECADMIN authority for Peter

REVOKE SECADM ON DATABASE FROM USER PETER;
```

方案: 使用行和列访问控制的 ExampleHMO - 行许可权

安全性管理员 Alex 通过使用行许可权（行和列访问控制的一个部件）来开始限制对 ExampleHMO 数据库的数据访问。行许可权按行对返回到用户的数据进行过滤。

允许患者查看自己的数据。允许医生查看其所有患者的数据，但不允许查看由其他医生治疗的患者的数据。属于 MEMBERSHIP、ACCOUNTING 或 DRUG_RESEARCH 角色的用户可以访问所有患者信息。要求安全性管理员 Alex 实现这些许可权，以在“有需要才能知道”基础上限制哪些用户可以查看任何给定行。

行许可权将根据已登录数据库的用户来限制或过滤行。在 ExampleHMO 中，行许可权将在名为 PATIENT 的表上创建一个横向数据限制。

Alex 将实现下列行许可权，以便将具有各自角色的用户限制为只能查看他们有特权查看的结果集:

```
CREATE PERMISSION ROW_ACCESS ON PATIENT
------------------------------------------------------
-- Accounting information:
-- ROLE PATIENT is allowed to access his or her own row
-- ROLE PCP is allowed to access his or her patients' rows
-- ROLE MEMBERSHIP, ACCOUNTING, and DRUG_RESEARCH are
-- allowed to access all rows
------------------------------------------------------------
FOR ROWS WHERE(VERIFY_ROLE_FOR_USER(SESSION_USER,'PATIENT') = 1
AND
PATIENT.USERID = SESSION_USER) OR
(VERIFY_ROLE_FOR_USER(SESSION_USER,'PCP') = 1
AND
PATIENT.PCP_ID = SESSION_USER) OR
 (VERIFY_ROLE_FOR_USER(SESSION_USER,'MEMBERSHIP') = 1 OR
 VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1 OR
 VERIFY_ROLE_FOR_USER(SESSION_USER, 'DRUG_RESEARCH') = 1)
ENFORCED FOR ALL ACCESS
ENABLE;
```

Alex 观察到，即使在创建行许可权之后，其他职员仍然可以查看所有数据。直到对定义了行许可权的表激活行许可权之后，才会应用行许可权。Alex 必须立即激活该许可权:

```
--Activate row access control to implement row permissions

ALTER TABLE PATIENT ACTIVATE ROW ACCESS CONTROL;
```

方案: 使用行和列访问控制的 ExampleHMO - 列掩码

安全性管理员 Alex 通过使用列掩码（行和列访问控制的一个部件）来进一步限制对 ExampleHMO 数据库的数据访问。除非允许用户查看返回给他们的数据，否则列掩码将按列来隐藏这些数据。

患者付款详细信息必须只能由会计部门的用户访问。帐户余额不能被任何其他数据库用户看到。要求 Alex 阻止任何不属于 ACCOUNTING 角色的用户进行访问。

Alex 将实现以下列掩码，以便将具有各自角色的用户限制为只能查看他们有特权查看的结果集:

```
--Create a Column MASK ON ACCT_BALANCE column on the PATIENT table

CREATE MASK ACCT_BALANCE_MASK ON PATIENT FOR
------------------------------------------------------------
-- Accounting information:
-- Role ACCOUNTING is allowed to access the full information
-- on column ACCT_BALANCE.
-- Other roles accessing this column will strictly view a
-- zero value.
------------------------------------------------------------
COLUMN ACCT_BALANCE RETURN
 CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1
```

```
   THEN ACCT_BALANCE
  ELSE 0.00
 END
ENABLE;
```

Alex 观察到，即使在创建列掩码之后，其他职员仍然可以查看该数据。直到对定义了列掩码的表激活列掩码之后，才会应用列掩码。Alex 必须立即激活该掩码:

```
--Activate column access control to implement column masks

ALTER TABLE PATIENT ACTIVATE COLUMN ACCESS CONTROL;
```

要求 Alex 通过管理来隐藏患者的保险号。只有患者、医生、会计或具有 MEMBERSHIP 角色的人才能查看 SSN 列。

此外，要保护患者的 PHARMACY 详细信息，PHARMACY 列中的信息必须只能由药品研究员或医生查看。仅当患者同意公开该信息时，药品研究员才能查看数据。

Alex 将实现以下列掩码，以便将具有各自角色的用户限制为只能查看他们有特权查看的结果集:

```
CREATE MASK SSN_MASK ON PATIENT FOR
-----------------------------------------------------
-- Personal contact information:
-- Roles PATIENT, PCP, MEMBERSHIP, and ACCOUNTING are allowed
-- to access the full information on columns SSN, USERID, NAME,
-- and ADDRESS. Other roles accessing these columns will
-- strictly view a masked value.
-----------------------------------------------------
COLUMN SSN RETURN
 CASE WHEN
  VERIFY_ROLE_FOR_USER(SESSION_USER,'PATIENT') = 1 OR
  VERIFY_ROLE_FOR_USER(SESSION_USER,'PCP') = 1 OR
  VERIFY_ROLE_FOR_USER(SESSION_USER,'MEMBERSHIP') = 1 OR
  VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1
 THEN SSN
  ELSE CHAR('XXX-XX-' || SUBSTR(SSN,8,4)) END
ENABLE;

CREATE MASK PHARMACY_MASK ON PATIENT FOR
-----------------------------------------------------------
-- Medical information:
-- Role PCP is allowed to access the full information on
-- column PHARMACY.
-- For the purposes of drug research, Role DRUG_RESEARCH can
-- conditionally see a patient's medical information
-- provided that the patient has opted-in.
-- In all other cases, null values are rendered as column
-- values.
-----------------------------------------------------
COLUMN PHARMACY RETURN
  CASE WHEN
  VERIFY_ROLE_FOR_USER(SESSION_USER,'PCP') = 1 OR
  (VERIFY_ROLE_FOR_USER(SESSION_USER,'DRUG_RESEARCH')=1
  AND
   EXISTS (SELECT 1 FROM PATIENTCHOICE C
 WHERE PATIENT.SSN = C.SSN AND C.CHOICE = 'drug-research' AND C.VALUE = 'opt-in'))
  THEN PHARMACY
  ELSE NULL
END
ENABLE;
```

Alex 观察到，在创建这两个列掩码之后，仅预期的用户才能看到该数据。PATIENT 表已激活列访问控制。

方案: 使用行和列访问控制的 ExampleHMO - 插入数据

接纳新患者进入医院进行治疗时，必须将新患者的记录添加到 ExampleHMO 数据库。

Bob 是一个新患者，必须将他的记录添加到 ExampleHMO 数据库。具有所需安全性权限的用户必须为 Bob 创建新记录。具有 MEMBERSHIP 角色的 Tom（来自 ExampleHMO 成员资格部门）将 Bob 登记为新成员。在连接至 ExampleHMO 数据库之后，Tom 运行下列 SQL 语句来将 Bob 添加到 ExampleHMO 数据库:

```
INSERT INTO PATIENT VALUES('123-45-6789', 'BOB', 'Bob', '123 Some St.',
'hypertension', 9.00,'LEE');
INSERT INTO PATIENTCHOICE VALUES('123-45-6789', 'drug-research', 'opt-in');
```

Tom 通过从 ExampleHMO 数据库的 PATIENT 表中查询到相同的信息确认已将 Bob 添加到数据库:

```
Select * FROM PATIENT WHERE NAME = 'Bob';

SSN         USERID      NAME       ADDRESS      PHARMACY     ACCT_BALANCE   PCP_ID

----------- --------- ------- ------------- ------------ ------------- ------
123-45-6789 BOB         Bob        123 Some St. XXXXXXXXXXX  0.00           LEE
```

方案: 使用行和列访问控制的 ExampleHMO - 更新数据

住院期间，Bob 的治疗方案会发生更改。因此，需要更新他在 ExampleHMO 数据库中的记录。

Bob 的医生 Dr. Lee 建议更改治疗方案并更改 Bob 的药物。必须更新 Bob 在 ExampleHMO 系统中的记录。ExampleHMO 数据库中设置的行许可权指定，不能查看行中数据的任何人都不能更新该行中的数据。因为 Bob 的 PCPID 包含 Dr. Lee 的标识并且设置了行许可权，所以 Dr. Lee 可以查看 Bob 的记录并可以使用以下示例 SQL 语句更新该记录:

```
UPDATE PATIENT SET PHARMACY = 'codeine' WHERE NAME = 'Bob';
```

Dr. Lee 检查该更新:

```
Select * FROM PATIENT WHERE NAME = 'Bob';

SSN         USERID      NAME       ADDRESS      PHARMACY     ACCT_BALANCE   PCP_ID

----------- --------- ------- ------------ ----------- -------------- ------
123-45-6789 BOB         Bob        123 Some St. codeine      0.00           LEE
```

Dug 是患者，他的主治医生是 Dr. Lee 的同事 Dr. James。Dr. Lee 尝试对 Dug 的记录进行相同的更新:

```
UPDATE PATIENT SET PHARMACY = 'codeine' WHERE NAME = 'Dug';
SQL0100W  No row was found for FETCH, UPDATE or DELETE; or the result of a
query is an empty table. SQLSTATE=02000
```

因为 Dug 的 PCPID 未包含 Dr. Lee 的标识并且设置了行许可权，所以 Dr. Lee 不能查看或更新 Dug 的记录。

方案: 使用行和列访问控制的 ExampleHMO - 数据查询

借助行和列访问控制，具有不同角色的人可以从相同的数据库查询中获得不同的结果集。例如，具有 DATAACCESS 权限的数据库管理员 Peter 无法查看 PATIENT 表中的任何数据。

Peter、Bob、Dr. Lee、Tom、Jane 和 John 各自连接至数据库并尝试执行以下 SQL 查询:

```
SELECT SSN, USERID, NAME, ADDRESS, PHARMACY, ACCT_BALANCE, PCP_ID FROM PATIENT;
```

查询结果将随运行该查询的用户而有所不同。由 Alex 创建的行和列访问控制将应用于这些查询。

以下是 Peter 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY    ACC_BALANCE PCP_ID

----------- --------- --------- ----------- ----------- ----------- -----------

  0 record(s) selected.
```

即使该表中有数据并且 Peter 是数据库管理员，但他并不具有查看所有数据的权限。

以下是 Bob 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY    ACC_BALANCE PCP_ID

----------- --------- --------- ----------- ----------- ----------- ------
123-45-6789 BOB         Bob         123 Some St.XXXXXXXXXX 0.00        LEE

1 record(s) selected.
```

Bob 作为患者只能看到他自己的数据。Bob 属于 PATIENT 角色。已对他隐藏 PHARMACY 和 ACC_BALANCE 列数据。

以下是 Dr. Lee 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY            ACC_BALANCE PCP_ID

----------- -------- -------- ----------- ------------------- ----------- ------
123-55-1234 MAX         Max         First Strt  hypertension        0.00        LEE
123-11-9856 SAM         Sam         Big Strt    High blood pressure 0.00        LEE
123-45-6789 BOB         Bob         123 Some St.codeine             0.00        LEE

3 record(s) selected.
```

Dr. Lee 只能看到他主治的患者的数据。Dr. Lee 属于 PCP 角色。对他隐藏了 ACC_BALANCE 列数据。

以下是 Tom 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY    ACC_BALANCE PCP_ID

----------- -------- -------- ----------- ----------- ----------- -----------
123-55-1234 MAX         Max         First Strt  XXXXXXXXXX 0.00        LEE
123-58-9812 MIKE        Mike        Long Strt   XXXXXXXXXX 0.00        JAMES
123-11-9856 SAM         Sam         Big Strt    XXXXXXXXXX 0.00        LEE
123-19-1454 DUG         Dug         Good Strt   XXXXXXXXXX 0.00        JAMES
123-45-6789 BOB         Bob         123 Some St.XXXXXXXXXX 0.00        LEE

5 record(s) selected.
```

Tom 可以看到所有成员。Tom 属于 MEMBERSHIP 角色。他不具有查看 PHARMACY 和 ACC_BALANCE 列中的任何数据的特权。

以下是 Jane 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY            ACC_BALANCE PCP_ID

----------- -------- -------- ----------- ------------------- ----------- -------
XXX-XX-1234 MAX         Max         First Strt  XXXXXXXXXX          0.00        LEE
XXX-XX-9812 MIKE        Mike        Long Strt   XXXXXXXXXX          0.00        JAMES
XXX-XX-9856 SAM         Sam         Big Strt    High blood pressure 0.00        LEE
XXX-XX-1454 DUG         Dug         Good Strt   Influenza           0.00        JAMES
XXX-XX-6789 BOB         Bob         123 Some St.codeine             0.00        LEE

5 record(s) selected.
```

Jane 可以看到所有成员。她属于 DRUG_RESEARCH 角色。对她隐藏了 SSN 和 ACC_BALANCE 列数据。仅当患者同意与药品研究公司共享他们的数据时，才会提供 PHARMACY 数据。

以下是 John 看到的结果集:

```
SSN         USERID      NAME        ADDRESS     PHARMACY    ACC_BALANCE PCP_ID

----------- -------- -------- ----------- ----------- ----------- --------
123-55-1234 MAX         Max         First Strt  XXXXXXXXXX 89.70       LEE
123-58-9812 MIKE        Mike        Long Strt   XXXXXXXXXX 8.30        JAMES
123-11-9856 SAM         Sam         Big Strt    XXXXXXXXXX 0.00        LEE
123-19-1454 DUG         Dug         Good Strt   XXXXXXXXXX 0.00        JAMES
123-45-6789 BOB         Bob         123 Some St.XXXXXXXXXX 9.00        LEE

5 record(s) selected.
```

John 可以看到所有成员。他属于 ACCOUNTING 角色。对他隐藏了 PHARMACY 列数据。

方案: 使用行和列访问控制的 ExampleHMO - 创建视图

可以在定义了行和列访问控制的表上创建视图。要求安全性管理员 Alex 在 PATIENT 表上创建医学研究员可以使用的视图。

与 ExampleHMO 具有合作关系的研究员可以在患者允许的情况下访问有限的患者数据。要求 Alex 和 IT 团队创建视图，以仅列示患者的与研究有关的特定信息。该报告必须包含患者保险号、患者的姓名和患者选择的泄露选项。

创建的视图将访存患者基本信息和健康状态泄露选项。此视图确保患者信息受保护并仅在患者许可的情况下受访存以用于任何其他目的。

Alex 和 IT 团队将实现以下视图:

```
CREATE VIEW PATIENT_INFO_VIEW AS
SELECT P.SSN, P.NAME FROM PATIENT P, PATIENTCHOICE C
WHERE P.SSN = C.SSN AND
   C.CHOICE = 'drug-research' AND
   C.VALUE = 'opt-in';
```

在 Alex 和他的团队创建该视图之后，用户可以查询该视图。用户将按照创建该视图时在基本表上定义的行和列访问控制规则来查看数据。

Alex 在对该视图执行以下查询之后会看到以下结果集:

```
SELECT SSN, NAME FROM PATIENT_INFO_VIEW;

SSN         NAME
----------- ----------

  0 record(s) selected.
```

Dr. Lee 在对该视图执行以下查询之后会看到以下结果集:

```
SELECT SSN, NAME FROM PATIENT_INFO_VIEW;

SSN         NAME
----------- ----------
123-11-9856 Sam
123-45-6789 Bob

  2 record(s) selected.
```

Bob 在对该视图执行以下查询之后会看到以下结果集:

```
SELECT SSN, NAME FROM PATIENT_INFO_VIEW;

SSN         NAME
----------- ----------
123-45-6789 Bob

  1 record(s) selected.
```

方案: 使用行和列访问控制的 ExampleHMO - 安全函数

必须认为函数是安全的，然后才能在行和列访问控制定义中调用这些函数。安全性管理员 Alex 与 ExampleHMO 的数据库开发者 Paul 讨论如何才能为他的新财务应用程序创建安全函数。

在 ExampleHMO 中的隐私和安全性策略生效之后，将通知 Alex，会计部已开发了功能强大的财务应用程序。ExampleHMOAccountingUDF 是 SQL 标量用户定义的函数 (UDF)，用在 PATIENT.ACCT_BALANCE 表和行的列掩码 ACCT_BALANCE_MASK 中。

在列掩码中只能调用安全的 UDF。Alex 首先与编写 UDF 的 Paul 讨论 UDF，以确保 UDF 中的操作是安全的。

当 Alex 认为该函数安全时，他会对 Paul 授予系统特权，以便 Paul 可以将 UDF 改变为安全 UDF:

```
GRANT CREATE_SECURE_OBJECT ON DATABASE TO USER PAUL;
```

要创建安全 UDF 或将 UDF 改变为安全 UDF，必须对开发者授予 CREATE_SECURE_OBJECT 权限。

Paul 创建以下函数:

```
CREATE FUNCTION EXAMPLEHMOACCOUNTINGUDF(X DECIMAL(12,2))
       RETURNS DECIMAL(12,2)
       LANGUAGE SQL
       CONTAINS SQL
       DETERMINISTIC
       NO EXTERNAL ACTION
       RETURN X*(1.0 + RAND(X));
```

Paul 改变该函数以使它成为安全函数:

```
ALTER FUNCTION EXAMPLEHMOACCOUNTINGUDF SECURED;
```

Alex 现在删除并重新创建掩码 ACC_BALANCE_MASK，以便使用新 UDF:

```
--Drop the mask to recreate

DROP MASK ACCT_BALANCE_MASK;



CREATE MASK EXAMPLEHMO.ACCT_BALANCE_MASK ONPATIENT FOR
-------------------------------------------------------------
-- Accounting information:
-- Role ACCOUNTING is allowed to invoke the secured UDF
-- ExampleHMOAccountingUDFL passing column ACCT_BALANCE as
-- the input argument
-- Other ROLEs accessing this column will strictly view a
-- zero value.
-------------------------------------------------------------
COLUMN ACCT_BALANCE RETURN
CASE WHEN VERIFY_ROLE_FOR_USER(SESSION_USER,'ACCOUNTING') = 1
THEN EXAMPLEHMOACCOUNTINGUDF(ACCT_BALANCE)
ELSE 0.00
  ENDENABLE;
```

具有 PCP 角色的 Dr. Lee 必须调用药品分析用户定义的函数。DrugUDF 返回患者的药品信息。在过去，Dr. Lee 发出调用 DrugUDF 的 SELECT 语句之后会很快接收到结果集。在使用行和列访问控制保护 PATIENT 表之后，相同的查询需要更长的时间才能返回结果集。

Dr. Lee 向 ExampleHMO IT 人员和安全性管理员 Alex 咨询有关此性能下降的问题。Alex 告诉 Dr. Lee，如果 UDF 不安全，那么无法优化查询，因此查询需要更长的时间才能返回结果集。

Alex 与 Dr. Lee 和所有者 Paul 检查 UDF，以确保 UDF 中的操作是安全的。当 Paul 仍然具有 Alex 授予的 CREATE_SECURE_OBJECT 特权时，Alex 要求 Paul 将 UDF 改变为安全 UDF:

```
--Function for ExampleHMO Pharmacy department

CREATE FUNCTION DRUGUDF(PHARMACY VARCHAR(5000))
      RETURNS VARCHAR(5000)
      NO EXTERNAL ACTION
      BEGIN ATOMIC
       IF PHARMACY IS NULL THEN
          RETURN NULL;
        ELSE
          RETURN 'Normal';
        END IF;
      END;

--Secure the UDF

ALTER FUNCTION DRUGUDF SECURED;

--Grant execute permissions to Dr.Lee

GRANT EXECUTE ON FUNCTION DRUGUDF TO USER LEE;
```

Dr. Lee 可以发出查询并且可以按预期优化该查询:

```
--Querying after the function is secured

SELECT PHARMACY FROM PATIENT
```

```
      WHERE DRUGUDF(PHARMACY) = 'Normal' AND SSN = '123-45-6789';

PHARMACY
--------
codeine

  1 record(s) selected.
```

方案: 使用行和列访问控制的 ExampleHMO - 安全触发器

在激活了行或列访问控制的表上定义的触发器必须是安全的。安全性管理员 Alex 与 ExampleHMO 的数据库开发者 Paul 讨论如何才能为他的新财务应用程序创建安全触发器。

Alex 与会计部交谈并了解到 PATIENT 表需要 AFTER UPDATE 触发器。此触发器将监视 ACCT_BALANCE 列的历史记录。

Alex 对 Paul 解释（Paul 具有创建此触发器的必需特权），必须将在行和列访问受保护的表中定义的任何触发器标记为安全。Paul 和 Alex 检查新触发器的操作并认为此触发器是安全的。

ExampleHMO_ACCT_BALANCE_TRIGGER 将监视 PATIENT 表中的 ACCT_BALANCE 列。每当更新该列时，该触发器会被触发并将当前帐户余额详细信息插入 ACCT_HISTORY 表中。

Paul 创建该触发器:

```
CREATE TRIGGER HOSPITAL.NETHMO_ACCT_BALANCE_TRIGGER
  AFTER UPDATE OF ACCT_BALANCE ON PATIENT
  REFERENCING OLD AS O NEW AS N
  FOR EACH ROW MODE DB2SQL SECURED
  BEGIN  ATOMIC
  INSERT INTO ACCT_HISTORY
  (SSN, BEFORE_BALANCE, AFTER_BALANCE, WHEN, BY_WHO)
  VALUES(O.SSN, O.ACCT_BALANCE, N.ACCT_BALANCE,
  CURRENT TIMESTAMP, SESSION_USER);
END;
```

会计部的 John 必须更新其 SSN 是 '123-45-6789' 的患者 Bob 的帐户余额。

John 在执行更新之前查看 Bob 的数据:

```
SELECT ACCT_BALANCE FROM PATIENT WHERE SSN = '123-45-6789';

ACCT_BALANCE
--------------
9.00

  1 record(s) selected.

SELECT * FROM ACCT_HISTORY WHERE SSN = '123-45-6789';

SSN         BEFORE_BALANCE AFTER_BALANCE  WHEN       BY_WHO
----------- -------------- -------------- ---------- --------------------

  0 record(s) selected.
```

然后 John 执行更新:

```
UPDATE PATIENT SET ACCT_BALANCE = ACCT_BALANCE * 0.9 WHERE SSN = '123-45-6789';
```

因为 PATIENT 表中定义了触发器，所以更新会触发此触发器。因为触发器定义了 SECURED，所以成功完成了更新。John 在执行更新之后查看 Bob 的数据:

```
SELECT ACCT_BALANCE FROM PATIENT WHERE SSN = '123-45-6789';

ACCT_BALANCE
--------------
8.10

  1 record(s) selected.

SELECT * FROM ACCT_HISTORY WHERE SSN = '123-45-6789';

SSN         BEFORE_BALANCE AFTER_BALANCE  WHEN       BY_WHO
----------- -------------- -------------- ---------- --------------------
123-45-6789 9.00           8.10           2010-10-10 JOHN
  1 record(s) selected.
```

方案: 使用行和列访问控制的 ExampleHMO - 撤销权限

作为安全性管理员，Alex 负责控制可以创建安全对象的人。当开发者创建了安全对象之后，Alex 将撤销他们对数据库的权限。

数据库开发者 Paul 已完成开发活动。Alex 立即撤销 Paul 的创建权限:

```
REVOKE CREATE_SECURE_OBJECT ON DATABASE FROM USER PAUL;
```

如果 Paul 在以后必须创建安全对象，那么他必须再次向 Alex 请求授予创建权限。

方案: 使用行和列访问控制的 ExampleBANK

此方案将 ExampleBANK（一个具有大量客户和许多支行的银行机构）作为行和列访问控制的用户进行介绍。ExampleBANK 使用行和列访问控制来确保其数据库策略反映了公司对于隐私和安全性的要求以及管理业务目标。

处理客户的投资、存款及其个人信息的组织（如 ExampleBANK）仅在必须知道的基础上共享其组织中的信息。此数据保护确保共享、查看和修改任何敏感的客户金融信息或个人信息的操作只能由有特权执行这些操作的职员执行。

方案: 使用行和列访问控制的 ExampleBANK - 安全策略

ExampleBANK 按照某些安全策略来实现对 DB2 数据库进行数据访问要遵循的安全策略。

这些安全策略符合 ExampleBANK 的隐私和数据保护规章。第一列概述了这些策略和 ExampleBANK 所面对的难题，第二列概述了解决难题的 DB2 行和列访问控制 (RCAC) 功能部件。

安全性难题	解决安全性难题的行和列访问控制功能部件
将行访问权限于仅对授权用户提供。出纳员仅被允许查看属于其支行的客户数据，而无法查看公司范围系统中 ExampleBANK 的所有客户。	可以实现行许可权以控制哪些用户可以查看任何特定行。

安全性难题	解决安全性难题的行和列访问控制功能部件
客户服务代表只有在使用帐户更新应用程序时才能访问帐号。通过存储过程 ACCOUNTS.ACCTUPDATE 来标识此应用程序。	如果客户服务代表在 ACCOUNTS.ACCTUPDATE 应用程序的外部查询数据，那么可以使用列掩码来过滤敏感数据或对他们隐藏这些数据。

方案: 使用行和列访问控制的 ExampleBANK - 数据库用户和角色

在此方案中，许多不同的人使用 ExampleBANK 数据。这些人具有不同的用户权限。

ExampleBANK 实现了其安全策略以将从数据库访问数据的方式进行分类。对数据的内部访问将基于访问数据的用户的职责分离以及这些用户的数据访问特权。ExampleBANK 创建了下列数据库角色来分离这些职责:

TELLER
表示分行位置的出纳员。

TELEMARKERTER
表示电话营销人员。

CSR
表示客户服务代表。

下列人员使用 ExampleBANK 数据:

ZURBIE
ExampleBANK 的客户服务代表。她属于 CSR 角色。

NEWTON
ExampleBANK 支行的出纳员。他属于 TELLER 角色。

PLATO
ExampleBANK 的电话营销人员。他属于 TELEMARKETER 角色。

如果您要尝试此方案中提供的任何示例 SQL 语句和命令，请创建这些用户标识并对他们授予所列示的权限。

下列示例 SQL 语句假定系统中已创建这些用户。这些 SQL 语句创建每个角色并给这些用户授予对 ExampleBANK 数据库中的各种表的 SELECT 许可权:

```
--Creating roles and granting authority

CREATE ROLE TELLER;

CREATE ROLE CSR;

CREATE ROLE TELEMARKERTER;

GRANT ROLE TELLER TO USER NEWTON;
GRANT ROLE CSR TO USER ZURBIE;
GRANT ROLE TELEMARKERTER TO USER PLATO;
```

方案: 使用行和列访问控制的 ExampleBANK - 数据库表

此方案重点介绍 ExampleBANK 数据库中的两个表: CUSTOMER 表和 INTERNAL_INFO 表。

INTERNAL_INFO 表存储有关为 ExampleBANK 工作的职员的信息。此方案将考虑 INTERNAL_INFO 表中的以下各列:

HOME_BRANCH
职员所在支行的标识。

EMP_ID
职员标识。

CUSTOMER 表存储各个客户的信息:

ACCOUNT
客户帐号。

NAME
客户名称。

INCOME
客户收入。

BRANCH
客户支行标识。

下列示例 SQL 语句创建 CUSTOMER 表和 INTERNAL_INFO 表。将授予对表的权限并且将插入数据。

```
--Client table storing information regarding client information
CREATE TABLE RCACTSPM.CUSTOMER (
   ACCOUNT VARCHAR(19),
   NAME VARCHAR(20),
   INCOME INTEGER,
   BRANCH CHAR(1)
);

--Internal_info table which stores employee information

CREATE TABLE RCACTSPM.INTERNAL_INFO (
   HOME_BRANCH CHAR(1),
   EMP_ID VARCHAR(10));

--Grant authority

GRANT SELECT ON RCACTSPM.CUSTOMER TO USER NEWTON, USER ZURBIE, USER PLATO;

--Insert data

INSERT INTO RCACTSPM.CUSTOMER VALUES ('1111-2222-3333-4444', 'Alice', 22000, 'A');
INSERT INTO RCACTSPM.CUSTOMER VALUES ('2222-3333-4444-5555', 'Bob', 71000, 'A');
INSERT INTO RCACTSPM.CUSTOMER VALUES ('3333-4444-5555-6666', 'Carl', 123000, 'B');
INSERT INTO RCACTSPM.CUSTOMER VALUES ('4444-5555-6666-7777', 'David', 172000, 'C');

INSERT INTO RCACTSPM.INTERNAL_INFO VALUES ('A', 'NEWTON');
INSERT INTO RCACTSPM.INTERNAL_INFO VALUES ('B', 'ZURBIE');
INSERT INTO RCACTSPM.INTERNAL_INFO VALUES ('C', 'PLATO');
```

方案: 使用行和列访问控制的 ExampleBANK - 行许可权

ExampleBANK 的安全性管理员通过使用行许可权（行和列访问控制的一个部件）来开始限制数据访问。行许可权按行对返回到用户的数据进行过滤。

只允许出纳员查看其所在支行中的客户数据。允许电话推销员和 CSR（客户服务代表）查看系统中的所有 ExampleBANK 客户，但电话推销员无法查看完整帐号。

行许可权将根据已登录数据库的用户来限制或过滤行。在 ExampleBANK 中，行许可权将在 CUSTOMER 表上创建一个横向数据限制。

安全性管理员将实现下列行许可权，以便将具有各自角色的用户限制为只能查看他们有特权查看的结果集:

```
CREATE PERMISSION TELLER_ROW_ACCESS ON RCACTSPM.CUSTOMER
-------------------------------------------------------
-- Teller information:
-- ROLE TELLER is allowed to access client data only
-- in their branch.
-------------------------------------------------------------
FOR ROWS WHERE VERIFY_ROLE_FOR_USER(USER, 'TELLER') = 1
AND
BRANCH = (SELECT HOME_BRANCH FROM RCACTSPM.INTERNAL_INFO WHERE EMP_ID = USER)
ENFORCED FOR ALL ACCESS
ENABLE;

CREATE PERMISSION CSR_ROW_ACCESS ON RCACTSPM.CUSTOMER
-----------------------------------------------------
-- CSR and telemarketer information:
-- ROLE TELEMARKETER and CSR are allowed to access all client
-- data rows in ExampleBANK.
-------------------------------------------------------------
FOR ROWS WHERE VERIFY_ROLE_FOR_USER (USER, 'CSR') = 1
OR
VERIFY_ROLE_FOR_USER (USER, 'TELEMARKETER') = 1
ENFORCED FOR ALL ACCESS
ENABLE;
```

安全性管理员观察到，即使在创建行许可权之后，职员仍然可以查看所有数据。直到对定义了行许可权的表激活行许可权之后，才会应用行许可权。安全性管理员必须立即激活该许可权:

```
--Activate row access control to implement row permissions

ALTER TABLE RCACTSPM.CUSTOMER ACTIVATE ROW ACCESS CONTROL;
```

方案: 使用行和列访问控制的 ExampleBANK - 列掩码

ExampleBANK 安全性管理员通过使用列掩码（行和列访问控制的一个部件）来进一步限制数据访问。除非允许用户或应用程序查看返回的数据，否则列掩码将按列来隐藏返回到用户或应用程序的数据。

客户服务代表可以查看 ExampleBANK 系统中的所有客户，但是不允许客户服务代表查看完整的帐号，除非他们正在使用特定的应用程序。

安全性管理员将实现以下列掩码，以便将客户服务代表限制为只能查看他们有特权查看的结果集:

```
CREATE MASK ACCOUNT_COL_MASK ON RCACTSPM.CUSTOMER FOR
-------------------------------------------------------------
-- Account number information:
-- Role customer service representative (CSR) is allowed to
-- access account number information only when they are using
-- the account update application. This application is
-- identified through stored procedure ACCOUNTS.ACCTUPDATE.
-- If a CSR queries this data outside of this application, the
-- account information is masked and the first 12 digits are
-- replaced with "x".
-------------------------------------------------------------
COLUMN ACCOUNT RETURN
  CASE WHEN (VERIFY_ROLE_FOR_USER (USER, 'CSR') = 1 AND
```

```
            ROUTINE_SPECIFIC_NAME = 'ACCTUPDATE' AND
            ROUTINE_SCHEMA = 'ACCOUNTS' AND
            ROUTINE_TYPE = 'P')
       THEN ACCOUNT
       ELSE 'xxxx-xxxx-xxxx-' || SUBSTR(ACCOUNT,16,4)
  END
ENABLE;
```

安全性管理员观察到，即使在创建列掩码之后，所有职员仍然可以查看该数据。直到对定义了列掩码的表激活列掩码之后，才会应用列掩码。安全性管理员必须立即激活该掩码:

```
--Activate column access control to implement column masks

ALTER TABLE RCACTSPM.CUSTOMER ACTIVATE COLUMN ACCESS CONTROL;
```

方案: 使用行和列访问控制的 ExampleBANK - 数据查询

借助行和列访问控制，具有不同角色的人可以从相同的数据库查询中获得不同的结果集。例如，Newton 是出纳员，他只能看到其支行外部的客户的任何数据。

Newton、Zurbie 和 Plato 各自连接至数据库并尝试执行以下 SQL 查询:

```
SELECT * FROM RCACTSPM.CUSTOMER;
```

查询结果将随运行该查询的用户而有所不同。由安全性管理员创建的行和列访问控制将应用于这些查询。

以下是 Newton 看到的结果集:

```
ACCOUNT             NAME                 INCOME      BRANCH
------------------- -------------------- ----------- ------
xxxx-xxxx-xxxx-4444 Alice                      22000 A
xxxx-xxxx-xxxx-5555 Bob                        71000 A

  2 record(s) selected.
```

Newton 作为支行 A 的出纳员只能看到属于该支行的 ExampleBANK 客户。

以下是 Zurbie 看到的结果集:

```
ACCOUNT             NAME                 INCOME      BRANCH
------------------- -------------------- ----------- ------
xxxx-xxxx-xxxx-4444 Alice                      22000 A
xxxx-xxxx-xxxx-5555 Bob                        71000 A
xxxx-xxxx-xxxx-6666 Carl                      123000 B
xxxx-xxxx-xxxx-7777 David                     172000 C

  4 record(s) selected.
```

Zurbie 作为客户服务代表可以看到系统中的所有 ExampleBANK 客户，但看不到这些客户的完整帐号，除非使用 ACCOUNTS.ACCTUPDATE 应用程序。因为是在 ACCOUNTS.ACCTUPDATE 的外部发出了此查询，所以将隐藏完整帐号。

以下是 Plato 看到的结果集:

```
ACCOUNT             NAME                 INCOME      BRANCH
------------------- -------------------- ----------- ------
xxxx-xxxx-xxxx-4444 Alice                      22000 A
xxxx-xxxx-xxxx-5555 Bob                        71000 A
```

```
xxxx-xxxx-xxxx-6666 Carl                      123000 B
xxxx-xxxx-xxxx-7777 David                     172000 C

  4 record(s) selected.
```

Plato 作为电话推销员可以看到系统中的所有 ExampleBANK 客户。

第 5 章 基于标签的访问控制 (LBAC)

基于标签的访问控制 (LBAC) 大大增强了您对哪些用户能够访问数据的控制。LBAC 使您能够准确地确定对于各行各列具有写访问权的用户和具有读访问权的用户。

LBAC 的功能

LBAC 功能的配置简单方便，可以针对特定的安全环境对其进行定制。所有 LBAC 配置工作都由*安全性管理员*（这是被授予 SECADM 权限的用户）执行。

安全性管理员通过创建安全标号组件来配置 LBAC 系统。*安全标号组件*是一种数据库对象，表示用于确定用户是否应该访问数据块的条件。例如，条件可以是用户是否在特定部门中或他们是否在完成特定项目。*安全策略*描述的是用来确定哪些用户能够访问哪些数据的条件。安全策略包含一个或多个安全标号组件。对于任何一个表，只能使用一个安全策略来保护它，但不同的表可以由不同的安全策略保护。

创建安全策略之后，安全性管理员将创建称为*安全标号*的对象，这些对象是安全策略的组成部分。安全标号包含安全标号组件。安全标号的具体内容由安全策略确定，可以将其配置成表示贵单位在确定哪些用户能访问特定数据项时所依据的条件。例如，如果您决定查看公司中某人的职务及其担任的项目，以便确定他们应该看到的数据，那么可以配置安全标号以使每个标号都可以包含该信息。LBAC 相当灵活，它使您能够设置非常复杂的条件，也能够建立非常简单的系统（每个标号都表示“高”或“低”信任级别）。

一旦创建安全标号，就可以使其与各个表列和表行相关联以保护存放在那些位置中的数据。受安全标号保护的数据称为*受保护数据*。安全性管理员通过将安全标号授予用户来允许该用户访问受保护数据。当用户尝试访问受保护数据时，该用户的安全标号将与用于保护该数据的安全标号进行比较。用于进行保护的标号将阻塞一部分安全标号。

允许一个用户、角色或组同时拥有多个安全策略的安全标号。但是，对于任何给定的安全策略，一个用户、角色或组最多可以拥有一个读访问权标号和一个写访问权标号。

安全性管理员还可以将免除权赋予用户。*免除权*允许用户访问安全标号可能不允许访问的受保护数据。安全标号和免除权统称为 *LBAC 凭证*。

如果尝试访问 LBAC 凭证不允许访问的受保护列，访问就会失败，并且会发出错误消息。

如果尝试读取 LBAC 凭证不允许读取的受保护行，那么 DB2 会将那些行视为不存在。在运行的任何 SQL 语句（包括 SELECT、UPDATE 或 DELETE）中都不能选择那些行。即使是聚集函数也会忽略 LBAC 凭证不允许读取的行。例如，COUNT(*) 函数将只返回您有权读取的行数。

视图和 LBAC

可以像对未受保护表定义视图那样对受保护表定义视图。当访问这样的视图时，将强制实施对基础表的 LBAC 保护。使用的 LBAC 凭证就是会话授权标识的那些 LBAC 凭

证。访问同一视图的两个用户可能会看到不同的行，这取决于他们的 LBAC 凭证。

引用完整性约束和 LBAC

下列规则说明了存在引用完整性约束时如何强制实施 LBAC 规则:

- **规则 1**: 不对内部生成的子表扫描应用 LBAC 读访问规则。这是为了避免出现孤立的子代。
- **规则 2**: 不对内部生成的父表扫描应用 LBAC 读访问规则。
- **规则 3**: 当对子表执行 CASCADE 操作时，应用 LBAC 写规则。例如，如果用户删除了父表，但由于违反 LBAC 写规则而无法删除任何子表，那么应该会回滚删除操作并发出错误。

使用 LBAC 时的存储器开销

使用 LBAC 在行级别来保护一个表时，增加的存储器成本就是行安全标号列所需的成本。此成本取决于所选择的安全标号的类型。例如，如果您创建具有两个组件的安全策略来保护表，那么该安全策略使用的安全标号将占用 16 个字节（每个组件占用 8 个字节）。因为行安全标号列被当作不可空的 VARCHAR 列来处理，所以这种情况下的总成本将为每行占用 20 个字节。通常，每行的总成本为 (*N**8 + 4) 个字节，其中 *N* 是组成用于保护表的安全策略的组件数。

使用 LBAC 在列级别来保护一个表时，列安全标号是元数据（即，它与该列的元数据一起存储在 SYSCOLUMNS 目录表中）。此元数据就是用于保护该列的安全标号的标识。在此情况下，用户表不会产生任何存储器开销。

LBAC 的不足之处

- LBAC 永远不允许访问被自主访问控制禁止访问的数据。

  示例: 如果您无权读取某个表，那么不会允许您读取该表的数据 - 即使是 LBAC 允许您访问的行和列亦如此。
- LBAC 凭证仅限制对受保护数据的访问。它们不影响对未受保护数据的访问。
- 删除表或数据库时，不会检查 LBAC 凭证，即使该表或数据库包含受保护数据亦如此。
- 备份数据时，不会检查 LBAC 凭证。如果您可以对某个表运行备份，那么应用于数据的 LBAC 保护功能不会以任何方式限制对各个表行的备份。并且，备份介质上的数据也不受 LBAC 保护。只有数据库中的数据才受保护。
- 不能使用 LBAC 来保护下列任何类型的表:
  - 登台表
  - 登台表依赖的表
  - 类型表
- 不能对昵称应用 LBAC 保护功能。

LBAC 教程

http://www.ibm.com/developerworks/data 上在线提供的教程为您导览了 LBAC 的基本用法，该教程称为 DB2 Label-Based Access Control, a practical guide。

LBAC 安全策略

安全性管理员使用安全策略来定义对表中的各行各列具有写访问权的用户和具有读访问权的用户。

安全策略包含以下信息:

- 在策略所包含的安全标号中使用的安全标号组件
- 在比较那些安全标号组件时使用的规则
- 在访问策略所保护的数据时使用的可选行为
- 在强制访问安全策略所保护的数据时要考虑哪些其他安全标号和免除权。例如，通过安全策略控制考虑或不考虑授予角色和组的安全标号的选项。

每个受保护的表都必须有且只有一个与其关联的安全策略。该表中的行和列只能由该安全策略所包含的安全标号保护，并且，对受保护数据的所有访问都遵循该策略的规则。在单个数据库中可以有多个安全策略，但不能同时使用多个安全策略来保护任何给定的表。

创建安全策略

您必须是安全性管理员才能创建安全策略。使用 SQL 语句 CREATE SECURITY POLICY 来创建安全策略。在执行 CREATE SECURITY POLICY 语句前，必须创建安全策略中列示的安全标号组件。创建安全策略时列示组件的顺序并未指示组件之间的任何优先顺序或其他关系，但在创建带有内置函数（例如 SECLABEL）的安全标号时，知道组件的顺序是非常重要的。

通过您创建的安全策略，可创建安全标号来保护数据。

改变安全策略

安全性管理员可以使用 ALTER SECURITY POLICY 语句来修改安全策略。

删除安全策略

您必须是安全性管理员才能删除安全策略。使用 SQL 语句 DROP 来删除安全策略。

如果安全策略与任何标相关联，（添加至任何表），那么不能删除此安全策略。

LBAC 安全标号组件概述

*安全标号组件*是组成基于标号的访问控制 (LBAC) 的数据库对象。安全标号组件用来建立贵单位的安全结构模型。

安全标号组件可以表示任何条件，您可以使用该条件来确定某个用户是否有权访问给定的数据。此类条件的典型示例包括:

- 该用户的可信程度
- 该用户所在的部门
- 该用户是否参与特定的项目

示例: 如果要让用户所在的部门对用户可以访问哪些数据产生影响，那么可以创建名为 dept 的组件，然后定义该组件的元素并让那些元素指定公司中的各个部门。然后，将组件 dept 包括在安全策略中。

安全标号组件的元素是该组件所允许的一个特定“设置”。

示例: 表示信任级别的安全标号组件可以有四个元素: Top Secret、Secret、Classified 和 Unclassified。

创建安全标号组件

您必须是安全性管理员才能创建安全标号组件。使用 SQL 语句 CREATE SECURITY LABEL COMPONENT 来创建安全标号组件。

创建安全标号组件时，您必须提供:

- 组件的名称
- 组件的类型（ARRAY、TREE 或 SET）
- 所允许元素的完整列表
- 对于类型 ARRAY 和 TREE，必须描述每个元素在组件结构中的位置

创建安全标号组件后，可根据这些组件来创建安全策略。通过此安全策略，可创建安全标号来保护数据。

组件的类型

有三种类型的安全标号组件:

- TREE: 每个元素表示树结构中的一个节点
- ARRAY: 每个元素表示线性刻度上的一个点
- SET: 每个元素表示集合中的一个成员

类型用来对元素相互之间的不同相关方式进行建模。例如，如果您要创建组件以描述公司中的一个或多个部门，那么可以使用 TREE 类型的组件，这是因为大部分企业结构都是树型的。如果您要创建组件以表示某人的信任级别，那么可以使用 ARRAY 类型的组件，这是因为，对于任何两个信任级别，总有一个高于另一个。

每种类型的详细信息（包括元素相互之间的关系的详细描述）都在它们各自的部分中描述。

改变安全标号组件

安全性管理员可以使用 ALTER SECURITY LABEL COMPONENT 语句来修改安全标号组件。

删除安全标号组件

您必须是安全性管理员才能删除安全标号组件。使用 SQL 语句 DROP 来删除安全标号组件。

LBAC 安全标号组件类型: SET

SET 是一种安全标号组件类型，它可以在基于标号的访问控制 (LBAC) 安全策略中使用。

SET 类型的组件是无序的元素列表。可以对此类组件的元素执行的唯一比较操作是“列表是否包含给定的元素”。

LBAC 安全标号组件类型: ARRAY

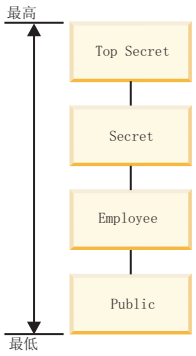
ARRAY 是一种安全标号组件类型。

在 ARRAY 类型的组件中，创建组件时列示元素的顺序定义了刻度，第一个列示的元素为最高值，最后一个列示的元素为最低值。

示例: 如果组件 mycomp 定义为:

```
CREATE SECURITY LABEL COMPONENT mycomp
    ARRAY [ 'Top Secret', 'Secret', 'Employee', 'Public' ]
```

则认为元素是按以下结构组织的:



在 ARRAY 类型的组件中，元素相互之间可以有下列类型的关系:

高于 如果元素 A 在 ARRAY 子句中出现在元素 B 之前，那么元素 A 高于元素 B。

低于 如果元素 A 在 ARRAY 子句中出现在元素 B 之后，那么元素 A 低于元素 B。

LBAC 安全标号组件类型: TREE

TREE 是一种安全标号组件类型，它可以在基于标号的访问控制 (LBAC) 安全策略中使用。

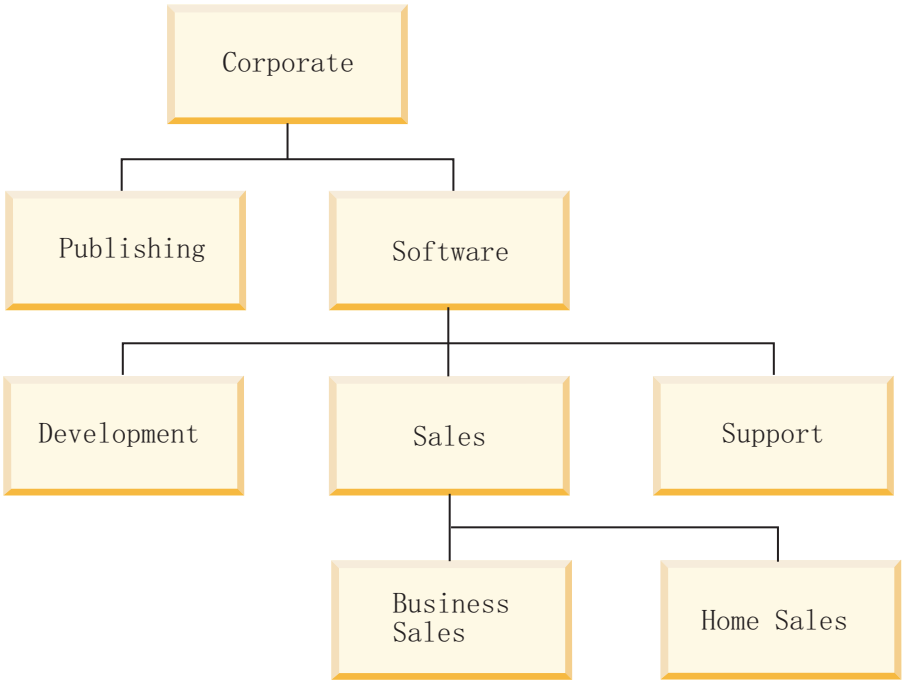
在 TREE 类型组件中，元素被视为以树结构方式排列。在指定 TREE 类型组件所包含的元素时，还必须指定它在哪个元素下面。唯一例外的是第一个元素，必须将其指定为树的 ROOT。这样，就能够以树结构的方式来组织元素。

示例: 如果组件 mycomp 定义为:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
        'Corporate'      ROOT,
        'Publishing'     UNDER 'Corporate',
        'Software'       UNDER 'Corporate',
        'Development'    UNDER 'Software',
        'Sales'          UNDER 'Software',
```

```
        'Support'        UNDER 'Software'
        'Business Sales' UNDER 'Sales'
        'Home Sales'     UNDER 'Sales'
)
```

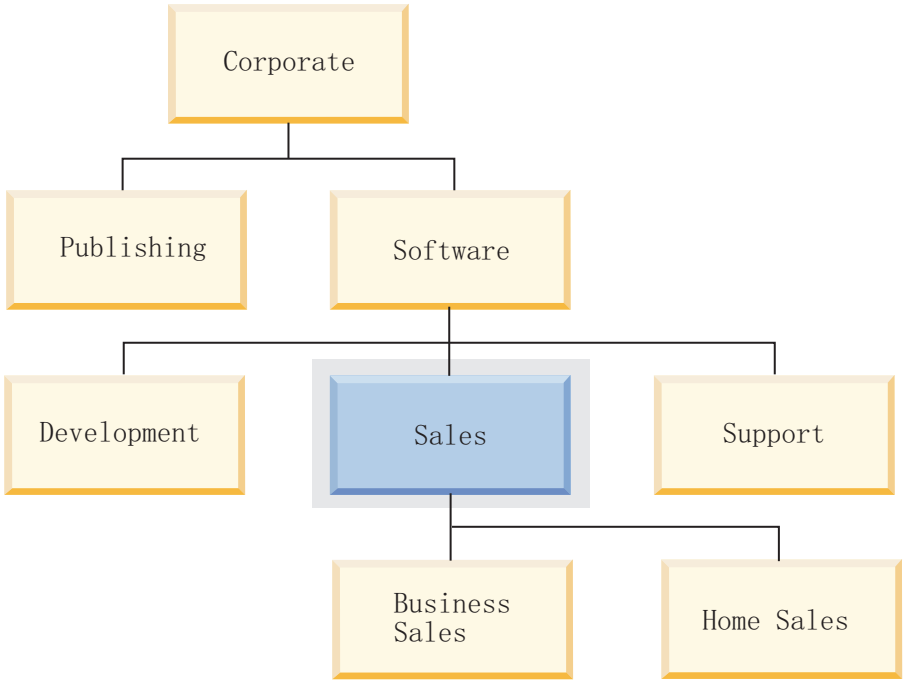
则认为元素是按以下树结构组织的:



在 TREE 类型的组件中，元素相互之间可以有下列类型的关系:

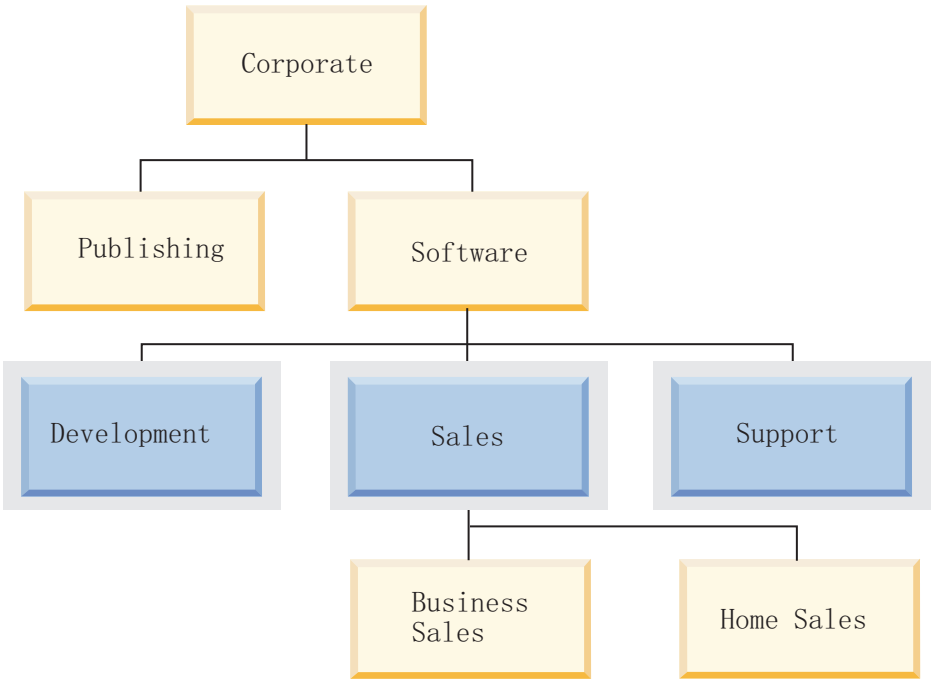
父代 如果元素 B 在元素 A 下面，那么元素 A 是元素 B 的父代。

示例: 此图显示 Business Sales 元素的父代:



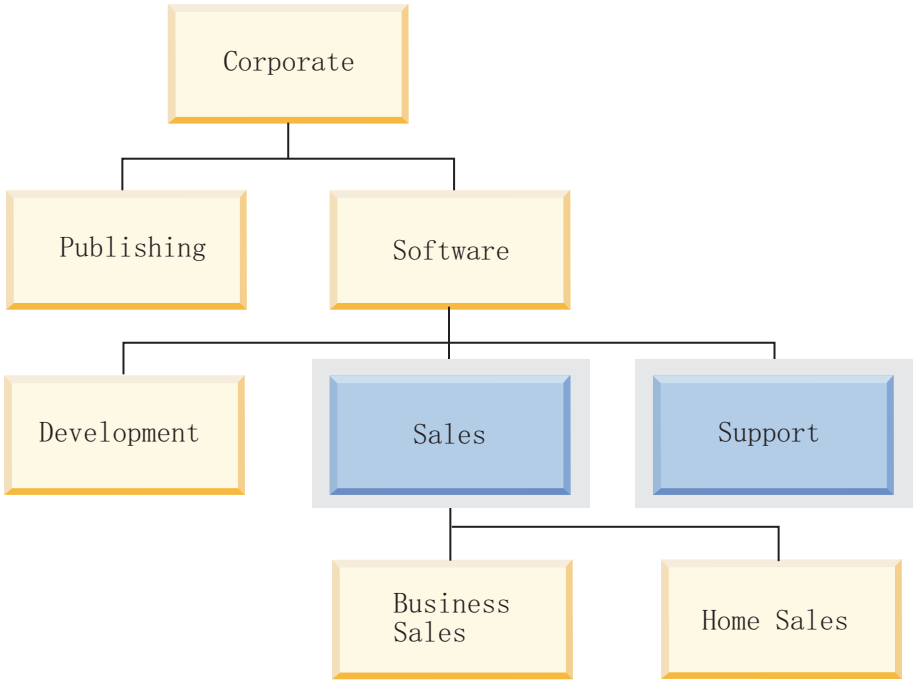
子代 如果元素 A 在元素 B 下面，那么元素 A 是元素 B 的子代。

示例: 此图显示 Software 元素的子代:



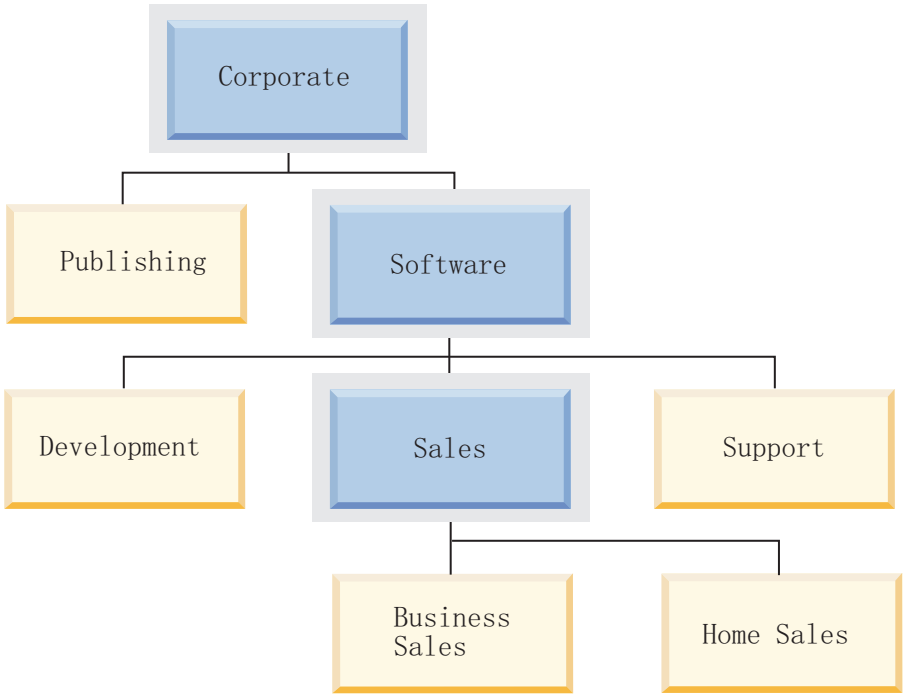
同代 如果两个元素有同一个父代，那么它们互为同代。

示例: 此图显示 Development 元素的同代:



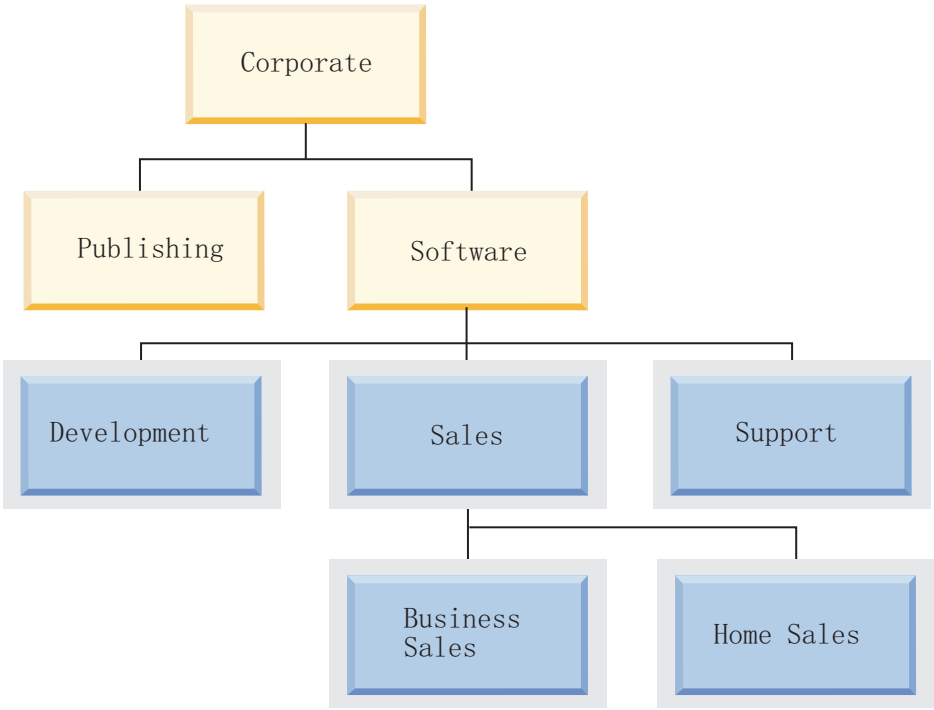
祖代 如果元素 A 是元素 B 的父代，或者元素 A 是 B 的父代的父代（依此类推），那么元素 A 是元素 B 的祖代。根元素是树中所有其他元素的祖代。

示例: 此图显示 Home Sales 元素的祖代:



后代 如果元素 A 是 B 的子代，或者如果元素 A 是 B 的子代的子代（依此类推），那么元素 A 是元素 B 的后代。

示例: 此图显示 Software 元素的后代:



LBAC 安全标号

在基于标签的访问控制 (LBAC) 中，安全标号是用于描述一组特定安全条件的数据库对象。安全标号被应用于数据以保护该数据。它们被授予用户以允许用户访问受保护数据。

当用户尝试访问受保护数据时，他们的安全标号将与用于保护该数据的安全标号进行比较。用于进行保护的安全标号将阻塞一部分安全标号。如果用户的安全标号被阻塞，该用户就无法访问该数据。

每个安全标号都刚好是一个安全策略的组成部分，它对于该安全策略中的每个组件都包含一个值。安全标号组件上下文中的*值*是指该组件所允许的 0 个或多个元素的列表。ARRAY 类型组件的值可以包含 0 个或一个元素，其他类型的值可以有 0 个或多个元素。未包含任何元素的值称为*空值*。

示例: 如果 TREE 类型组件有三个元素 Human Resources、Sales 和 Shipping，那么该组件的一些有效值是:

- Human Resources（或任何单个元素）
- Human Resources, Shipping（或者那些元素的任何其他组合，条件是不多次包括同一元素）
- *空值*

特定安全标号是否阻塞另一个安全标号，是由标号中每个组件值以及表的安全策略中指定的 LBAC 规则集确定的。在讨论如何比较 LBAC 安全标号的主题中提供了有关如何进行比较的详细信息。

将安全标号转换为文本字符串时，它们将使用在讨论安全标号值格式的主题中描述的格式。

创建安全标号

您必须是安全性管理员才能创建安全标号。使用 SQL 语句 CREATE SECURITY LABEL 来创建安全标号。创建安全标号时，需要提供:

- 标号的名称
- 包含该标号的安全策略
- 该安全策略中包含的一个或多个组件的值

对于任何未指定值的组件，假定它们具有空值。安全标号必须至少有一个非空值。

改变安全标号

不能改变安全标号。更改安全标号的唯一方法是删除它，然后重新创建。但是，安全性管理员可以使用 ALTER SECURITY LABEL COMPONENT 语句来修改安全标号的*组件*。

删除安全标号

您必须是安全性管理员才能删除安全标号。使用 SQL 语句 DROP 来删除安全标号。无法删除正被用来保护数据库中任何位置中的数据的安全标号或者当前正被一个或多个用户拥有的安全标号。

授予安全标号

您必须是安全性管理员才能将安全标号授予用户、组或角色。使用 SQL 语句 GRANT SECURITY LABEL 来授予安全标号。授予安全标号时，可以将其作为读访问安全标号、写访问安全标号或者读写访问安全标号授予。对于同一种访问类型，用户、组或角色不能拥有同一安全策略中的多个安全标号。

撤销安全标号

您必须是安全性管理员才能撤销用户、组或角色的安全标号。要撤销安全标号，请使用 SQL 语句 REVOKE SECURITY LABEL。

与安全标号兼容的数据类型

安全标号的数据类型为 SYSPROC.DB2SECURITYLABEL。支持在 SYSPROC.DB2SECURITYLABEL 与 VARCHAR(128) FOR BIT DATA 之间进行数据转换。

确定用户拥有的安全标号

可以使用以下查询来确定用户拥有的安全标号:

```
SELECT A.grantee, B.secpolicyname, c.seclabelname
FROM syscat.securitylabelaccess A, syscat.securitypolicies B, syscat.securitylabels C
WHERE A.seclabelid = C.seclabelid and B.secpolicyid = C.secpolicyid
```

安全标号值的格式

安全标号中的值有时是以字符串格式表示的，例如，使用内置函数 SECLABEL 时情况就是这样。

当安全标号中的值表示为字符串时，这些值的格式如下:

- 组件值是从左到右列示的，并且列示顺序与安全策略的 CREATE SECURITY POLICY 语句中的组件列示顺序相同
- 元素由该元素的名称表示
- 不同组件的元素由冒号（:）分隔
- 如果对同一个组件给出了多个元素，那么将那些元素括在括号（()）中并用逗号（,）分隔
- 空值由一对空括号（()）表示

示例: 安全策略包含一个安全标号，该安全策略由以下顺序的三个组件组成: Level、Department 和 Projects。该安全标号包含下列值:

表 8. 安全标号的示例值

组件	值
Level	Secret
Department	*空值*
Projects	• Epsilon 37 • Megaphone • Cloverleaf

这些安全标号值看上去就像以下字符串:

```
'Secret:():(Epsilon 37,Megaphone,Cloverleaf)'
```

如何比较 LBAC 安全标号

当您尝试访问受基于标签的访问控制 (LBAC) 保护的数据时，您的 LBAC 凭证将与一个或多个安全标号进行比较，以确定是否阻塞该访问。LBAC 凭证是您拥有的任何安全标号加上您拥有的任何免除权。

只能进行两种类型的比较。LBAC 凭证可以与单个读访问安全标号进行比较，LBAC 凭证也可以与单个写访问安全标号进行比较。更新和删除操作被认为是先读后写。当操作需要进行多次比较时，每次比较都是单独进行的。

使用的安全标号

即使您拥有多个安全标号，也只有一个安全标号会与用于保护的安全标号进行比较。使用的标号符合下列条件:

- 它包含在用于保护所访问的表的安全策略中。
- 它是针对该访问类型（读或写）授予的。

如果您没有符合这些条件的安全标号，那么将使用缺省安全标号，即所有组件都为空值。

如何进行比较

安全标号的比较是逐个组件进行的。如果安全标号的某个组件没有值，就会假定为空值。对每个组件进行检查时，将使用 LBAC 规则集中的适当规则来确定：该组件值中的元素是否应该被保护标号中同一组件值中的元素阻塞。如果任何值被阻塞，您的 LBAC 凭证就会被保护安全标号阻塞。

进行比较时使用的 LBAC 规则集是在安全策略中指定的。要了解规则内容以及每条规则的使用时间，请查看该规则集的描述。

免除权对比较的影响

如果您拥有用于比较两个值的规则的免除权，那么不会进行该比较，并且假定保护值不会阻塞您的安全标号值。

示例: LBAC 规则集是 DB2LBACRULES，安全策略有两个组件。一个组件的类型为 ARRAY，另一个组件的类型为 TREE。已授予用户对规则 DB2LBACREADTREE 的免除权，该规则是比较 TREE 类型组件值时使用的读访问规则。如果用户尝试读取受保护数据，那么无论该用户的 TREE 组件为何值（即使为空值），也不会阻塞访问，这是因为未使用该规则。用户是否可以读取数据完全取决于标号的 ARRAY 组件值。

LBAC 规则集概述

LBAC 规则集是比较安全标号时要使用的一组预定义规则。在对两个安全标号的值进行比较时，将使用规则集中的一个或多个规则来确定一个值是否阻塞另一个值。

每个 LBAC 规则集都由唯一的名称标识。在创建安全策略时，必须指定要与该策略配合使用的 LBAC 规则集。对该策略所包含的安全标号所作的任何比较都将使用该 LBAC 规则集。

规则集中的每个规则也由唯一的名称标识。在对该规则进行免除授权时，需要使用规则的名称。

规则集中的规则数以及使用每个规则的时间随规则集的不同而有所变化。

目前，只有一个受支持的 LBAC 规则集。该规则集的名称为 DB2LBACRULES。

LBAC 规则集: DB2LBACRULES

DB2LBACRULES LBAC 规则集提供了一组传统规则，这些规则用于对安全标号组件值进行比较。这个规则集能够预防上写和下写。

上写和下写的定义

上写和下写仅适用于 ARRAY 类型的组件，并且仅适用于写访问权。当用于保护所写数据的值大于您的值时，就会发生上写。当用于保护数据的值小于您的值时，就会发生下写。缺省情况下，既不允许上写也不允许下写，这意味着只能写您的值所保护的数据。

在对同一组件的两个值进行比较时，使用的规则取决于组件类型（ARRAY、SET 或 TREE）以及所尝试的访问类型（读或写）。下表列示了规则、指示了使用每个规则的时间，并描述了该规则确定是否阻塞访问的途径。

表 9. DB2LBACRULES 规则摘要

规则名	用来对此类型的组件的值进行比较	用于此类型的访问	当满足此条件时阻塞访问
DB2LBACREADARRAY	ARRAY	读	用户的值小于保护值。
DB2LBACREADSET	SET	读	存在一个或多个用户所没有的保护值。
DB2LBACREADTREE	TREE	读	用户所有的值都既不等于任何一个保护值也不是任何一个保护值的祖代。
DB2LBACWRITEARRAY	ARRAY	写	用户的值大于保护值或小于保护值。[1]
DB2LBACWRITESET	SET	写	存在一个或多个用户所没有的保护值。
DB2LBACWRITETREE	TREE	写	用户所有的值都既不等于任何一个保护值也不是任何一个保护值的祖代。

注:

1. 可以将 DB2LBACWRITEARRAY 规则看成两个不同规则的组合。其中一个规则防止写高于您的级别的数据（上写），另一个规则防止写低于您的级别的数据（下写）。在进行此规则的免除授权时，可以对用户免除这两个规则的其中一个或者全部。

规则如何处理空值

所有规则都以相同的方式处理空值。空值不会阻塞其他的值，但会被任何非空值阻塞。

DB2LBACREADSET 和 DB2LBACWRITESET 示例

这些示例对尝试读或尝试写受保护数据的用户有效。这些示例假定值属于类型为 SET 的组件，该组件包含下列元素: one two three four

表 10. DB2LBACREADSET 和 DB2LBACWRITESET 规则的应用示例

用户的值	保护值	是否阻塞访问?
“one”	“one”	不阻塞。值相同。
“(one,two,three)”	“one”	不阻塞。用户的值包含元素“one”。
“(one,two)”	“(one,two,four)”	阻塞。元素“four”包含在保护值中，但未包含在用户的值中。
’()’	“one”	阻塞。空值会被任何非空值阻塞。
“one”	’()’	不阻塞。空值不会阻塞任何值。
’()’	’()’	不阻塞。空值不会阻塞任何值。

DB2LBACREADTREE 和 DB2LBACWRITETREE

这些示例对同时进行读写访问的用户有效。这些示例假定值属于类型为 TREE 的组件，该组件是按以下方式定义的:

```
CREATE SECURITY LABEL COMPONENT mycomp
TREE (
        'Corporate'      ROOT,
        'Publishing'     UNDER 'Corporate',
        'Software'       UNDER 'Corporate',
```

```
        'Development'    UNDER 'Software',
        'Sales'          UNDER 'Software',
        'Support'        UNDER 'Software'
        'Business Sales' UNDER 'Sales'
        'Home Sales'     UNDER 'Sales'
)
```

这意味着元素是按此方式排列的:

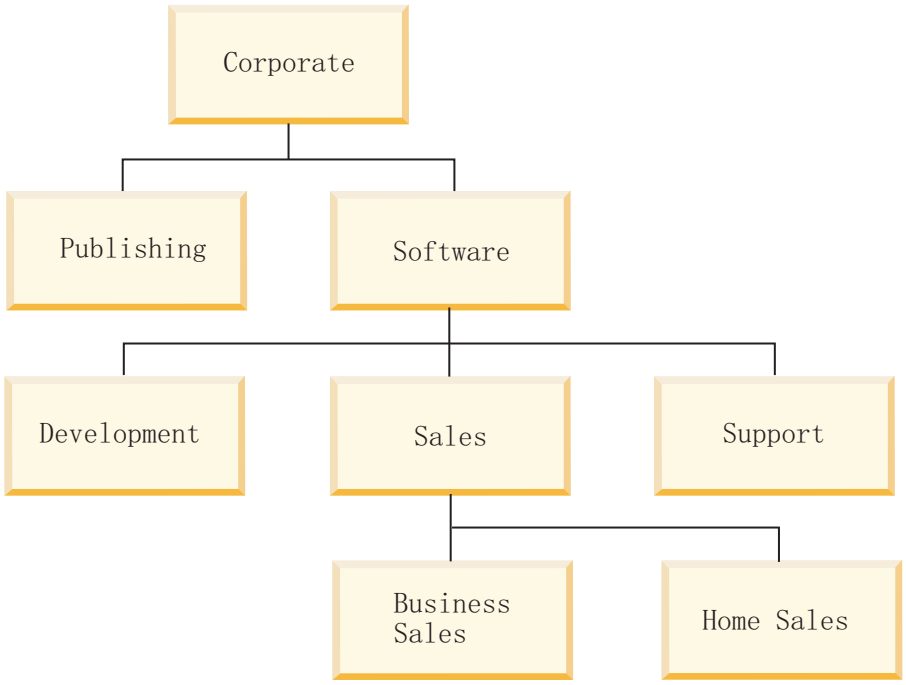


表 11. DB2LBACREADTREE 和 DB2LBACWRITETREE 规则的应用示例

用户的值	保护值	是否阻塞访问?
'(Support,Sales)'	'Development'	阻塞。元素“Development”不是用户的值，“Support”和“Sales”都不是“Development”的祖代。
'(Development,Software)'	'(Business Sales,Publishing)'	不阻塞。元素“Software”是“Business Sales”的祖代。
'(Publishing,Sales)'	'(Publishing,Support)'	不阻塞。两组值都包含元素“Publishing”。
'Corporate'	'Development'	不阻塞。根值是所有其他值的祖代。
'()'	'Sales'	阻塞。空值会被任何非空值阻塞。
'Home Sales'	'()'	不阻塞。空值不会阻塞任何值。
'()'	'()'	不阻塞。空值不会阻塞任何值。

DB2LBACREADARRAY 示例

这些示例仅适用于读访问。这些示例假定值属于类型为 ARRAY 的组件，该组件包含按以下方式排列的下列元素:

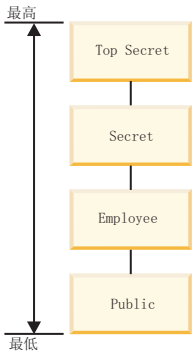


表 12. DB2LBACREADARRAY 规则的应用示例

用户的值	保护值	是否阻塞读访问?
“Secret”	“Employee”	不阻塞。元素“Secret”高于元素“Employee”。
“Secret”	“Secret”	不阻塞。值相同。
“Secret”	“Top Secret”	阻塞。元素“Top Secret”高于元素“Secret”。
’()’	’Public’	阻塞。空值会被任何非空值阻塞。
’Public’	’()’	不阻塞。空值不会阻塞任何值。
’()’	’()’	不阻塞。空值不会阻塞任何值。

DB2LBACWRITEARRAY 示例

这些示例仅适用于写访问。这些示例假定值属于类型为 ARRAY 的组件，该组件包含按以下方式排列的下列元素:

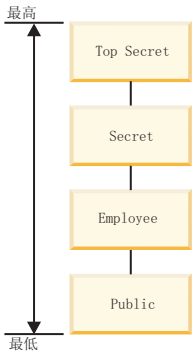


表 13. DB2LBACWRITEARRAY 规则的应用示例

用户的值	保护值	是否阻塞写访问?
“Secret”	“Employee”	阻塞。元素“Employee”低于元素“Secret”。
“Secret”	“Secret”	不阻塞。值相同。
“Secret”	“Top Secret”	阻塞。元素“Top Secret”高于元素“Secret”。
’()’	’Public’	阻塞。空值会被任何非空值阻塞。

表 13. DB2LBACWRITEARRAY 规则的应用示例 (续)

用户的值	保护值	是否阻塞写访问?
'Public'	'()'	不阻塞。空值不会阻塞任何值。
'()'	'()'	不阻塞。空值不会阻塞任何值。

LBAC 规则免除权

如果您拥有特定安全策略的特定规则的 LBAC 规则免除权，那么当您尝试访问受该安全策略保护的数据时，就不会强制实施该规则。

在比较任何安全策略的安全标号时，如果该安全策略不是授予免除权时所针对的安全策略，该免除权不起作用。

示例:

有两个表: T1 和 T2。T1 受安全策略 P1 保护，T2 是受安全策略 P2 保护。两个安全策略都有一个组件。它们的组件都具有 ARRAY 类型。T1 和 T2 都仅包含一行数据。在安全策略 P1 中，您拥有的读访问安全标号不允许您访问 T1 中的行。在安全策略 P2 中，您拥有的读访问安全标号不允许您对 T2 中的行执行读访问。

现在，在 P1 中，您被授予对 DB2LBACREADARRAY 的免除权。那么，您就可以读取 T1 中的行，但不能读取 T2 中的行，这是因为 T2 受另一个安全策略保护，而在该策略中，您没有对 DB2LBACREADARRAY 规则的免除权。

您可以拥有多个免除权。如果您对安全策略使用的每条规则都拥有免除权，那么您对该安全策略保护的所有数据拥有完全访问权。

授予 LBAC 规则免除权

您必须是安全性管理员才能授予 LBAC 规则免除权。要授予 LBAC 规则免除权，请使用 SQL 语句 GRANT EXEMPTION ON RULE。

授予 LBAC 规则免除权时，您要提供下列信息:

- 免除权所针对的规则
- 免除权所针对的安全策略
- 被授予免除权的用户、组或角色

重要事项: LBAC 规则免除权提供了非常强大的访问权。因此，授予免除权时务必十分谨慎。

撤销 LBAC 规则免除权

您必须是安全性管理员才能撤销 LBAC 规则免除权。要撤销 LBAC 规则免除权，请使用 SQL 语句 REVOKE EXEMPTION ON RULE。

确定用户拥有的规则免除权

可以使用以下查询来确定用户拥有的规则免除权:

```
SELECT A.grantee, A.accessrulename, B.secpolicyname
FROM syscat.securitypolicyexemptions A, syscat.securitypolicies B
WHERE A.secpolicyid = B.secpolicyid
```

用于管理 LBAC 安全标号的内置函数

提供了内置函数 SECLABEL、SECLABEL_BY_NAME 和 SECLABEL_TO_CHAR 来管理基于标签的访问控制 (LBAC) 安全标号。

下面对这三个函数作了简要描述，详细描述见 *SQL Reference*。

SECLABEL

通过指定安全策略和每个标号组件值，此内置函数用来构建安全标号。返回的值的数据类型为 DB2SECURITYLABEL，它是包含在指定安全策略中的安全标号，并且具有指定的组件值。具有指定值的安全标号不必已存在。

示例: 表 T1 有两列，第一列的数据类型为 DB2SECURITYLABEL，第二列的数据类型为 INTEGER。T1 受安全策略 P1 保护，此安全策略有三个安全标号组件: level、departments 和 groups。如果 UNCLASSIFIED 是组件 level 的元素，ALPHA 和 SIGMA 都是组件 departments 的元素，G2 是组件 groups 的元素，那么可以按如下方式插入安全标号:

```
INSERT INTO T1 VALUES
   ( SECLABEL( 'P1', 'UNCLASSIFIED:(ALPHA,SIGMA):G2' ), 22 )
```

SECLABEL_BY_NAME

此内置函数接受安全策略的名称以及该安全策略所包含的安全标号的名称。然后，它将指定的安全标号作为 DB2SECURITYLABEL 返回。在将现有安全标号插入到数据类型为 DB2SECURITYLABEL 的列时，必须使用此函数。

示例: 表 T1 有两列，第一列的数据类型为 DB2SECURITYLABEL，第二列的数据类型为 INTEGER。名为 L1 的安全标号是安全策略 P1 的一部分。以下 SQL 语句插入该安全标号:

```
INSERT INTO T1 VALUES ( SECLABEL_BY_NAME( 'P1', 'L1' ), 22 )
```

此 SQL 语句无法正常运行:

```
INSERT INTO T1 VALUES ( P1.L1, 22 )     // Syntax Error!
```

SECLABEL_TO_CHAR

此内置函数返回安全标号所包含的值的字符串表示。

示例: 表 T1 中的列 C1 的数据类型为 DB2SECURITYLABEL。T1 受安全策略 P1 保护，此安全策略有三个安全标号组件: level、departments 和 groups。T1 包含一行，对于每个组件，列 C1 中的值包含下列元素:

组件	元素
level	SECRET
departments	DELTA 和 SIGMA
groups	G3

拥有允许读取该行的 LBAC 凭证的用户执行以下 SQL 语句:

```
SELECT SECLABEL_TO_CHAR( 'P1', C1 ) AS C1 FROM T1
```

输出如下所示:

```
C1

'SECRET:(DELTA,SIGMA):G3'
```

使用 LBAC 来保护数据

基于标签的访问控制 (LBAC) 可以用于保护数据行和/或数据列。表中的数据只能由保护该表的安全策略所包含的安全标号保护。可以在创建表时进行数据保护工作（包括添加安全策略），以后也可以通过改变表来执行此工作。

可以在同一个 CREATE TABLE 或 ALTER TABLE 语句中对表添加安全策略以及保护该表中的数据。

作为惯例，不允许以当前 LBAC 凭证不允许写数据的方式来保护该数据。

对表添加安全策略

创建表时，可以使用 CREATE TABLE 语句的 SECURITY POLICY 子句来向表添加安全策略。可以使用 ALTER TABLE 语句的 ADD SECURITY POLICY 子句来向现有表添加安全策略。您不需要具有 SECADM 权限或拥有 LBAC 凭证就可以对表添加安全策略。

不能对 LBAC 无法保护的表类型添加安全策略。请参阅 LBAC 概述以获取 LBAC 无法保护的表类型的列表。

不能将一个以上安全策略添加给任何表。

保护行

在创建表时，可以通过包括数据类型为 DB2SECURITYLABEL 的列来允许新表中的行受保护。CREATE TABLE 语句还必须对该表添加安全策略。您不需要具有 SECADM 权限或拥有任何 LBAC 凭证就可以创建这样的表。

可以通过添加数据类型为 DB2SECURITYLABEL 的列来允许现有表中的行受保护。要添加这样的列，该表必须已受安全策略保护，否则添加列的 ALTER TABLE 语句还必须对该表添加安全策略。在添加列之后，将使用允许进行写访问的安全标号来保护所有已存在的行。如果用于保护该表的安全策略未包含允许进行写访问的安全标号，那么无法添加数据类型为 DB2SECURITYLABEL 的列。

在表包含 DB2SECURITYLABEL 类型的列之后，通过在该列中存储安全标号来保护每个新数据行。在有关插入和更新受 LBAC 保护的数据的主题中，描述了有关此操作工作方式的详细信息。您必须要有 LBAC 凭证才能将行插入到包含类型为 DB2SECURITYLABEL 的列的表中。

不能删除数据类型为 DB2SECURITYLABEL 的列，也不能将其更改为任何其他数据类型。

保护列

在创建表时，可以使用 CREATE TABLE 语句的 SECURED WITH 列选项来保护列。可以通过在 ALTER TABLE 语句中使用 SECURED WITH 选项来对现有的列添加保护。

要保护具有特定安全标号的列，您必须要有允许对该安全标号所保护的数据执行写操作的 LBAC 凭证。您不需要具有 SECADM 权限。

列只能由保护该表的安全策略所包含的安全标号保护。无法保护没有安全策略的表中的列。允许在同一个语句中保护带有安全策略的表并保护一列或多列。

可以保护表中任意数目的列，但是一个列只能由一个安全标号保护。

读取受 LBAC 保护的数据

当尝试读取受基于标号的访问控制 (LBAC) 保护的数据时，将把读操作 LBAC 凭证与保护该数据的安全标号作比较。如果保护数据的标号未阻塞您的凭证，就会允许您读取该数据。

对于受保护列来说，保护安全标号是在表的模式中定义的。对于表中的每一行，该列的保护安全标号都是相同的。对于受保护行来说，保护安全标号存储在行的类型为 DB2SECURITYLABEL 的列中。表中每一行的保护安全标号都可以不同。

在有关如何比较 LBAC 安全标号的主题中提供了有关 LBAC 凭证如何与安全标号进行比较的详细信息。

读取受保护的列

在尝试读取受保护的列时，LBAC 凭证将与用于保护该列的安全标号作比较。根据比较结果的不同，访问将被阻塞或允许。如果访问被阻塞，就会返回错误，并且语句将失败。否则，该语句正常地继续执行。

尝试读取 LBAC 凭证不允许读取的列将导致整个语句失败。

示例:

表 T1 有两个受保护的列。列 C1 受安全标号 L1 保护。列 C2 受安全标号 L2 保护。

假定用户 Jyoti 拥有读操作 LBAC 凭证，该凭证允许访问安全标号 L1，但不允许访问 L2。如果 Jyoti 发出以下 SQL 语句，此语句将失败:

```
SELECT * FROM T1
```

由于 SELECT 子句指定了通配符（*），因此它将包括 C2 列，所以此语句失败。

如果 Jyoti 发出以下 SQL 语句，此语句就会成功:

```
SELECT C1 FROM T1
```

在 SELECT 子句中，唯一受保护的列是 C1，Jyoti 的 LBAC 凭证允许她读取该列。

读取受保护的行

如果您没有允许读取某一行的 LBAC 凭证，那么该行对于您来说就好像不存在一样。

读取受保护的行时，将只返回 LBAC 凭证允许进行读访问的那些行。即使在 SELECT 子句中未指定类型为 DB2SECURITYLABEL 的列，情况亦如此。

根据 LBAC 凭证的不同，不同的用户在包含受保护行的表中可能会看到不同的行。例如，如果 T1 包含受保护行，但是有两个用户拥有不同的 LBAC 凭证，那么这两个执行 `SELECT COUNT(*) FROM T1` 语句的用户可能会获得不同的结果。

LBAC 凭证不仅影响 SELECT 语句，还影响诸如 UPDATE 和 DELETE 之类的其他 SQL 语句。如果您没有允许读取某一行的 LBAC 凭证，就无法影响该行。

示例:

表 T1 包含下列行和列。列 ROWSECURITYLABEL 的数据类型为 DB2SECURITYLABEL。

表 14. 表 T1 中的示例值

LASTNAME	DEPTNO	ROWSECURITYLABEL
Rjaibi	55	L2
Miller	77	L1
Fielding	11	L3
Bird	55	L2

假定用户 Dan 的 LBAC 凭证允许他读取受安全标号 L1 保护的数据，但不允许他读取受 L2 或 L3 保护的数据。

Dan 发出以下 SQL 语句:

```
SELECT * FROM T1
```

此 SELECT 语句将只返回 Miller 那一行。不会返回任何错误消息或警告。

Dan 看到的表 T1 是这样的:

表 15. 表 T1 的视图中的示例值

LASTNAME	DEPTNO	ROWSECURITYLABEL
Miller	77	L1

对于 Rjaibi、Fielding 和 Bird 这三行来说，由于它们的安全标号阻塞了读访问，所以不会返回这三行。Dan 无法删除或更新这些行。这些行也不会包括在任何聚集函数中。对于 Dan 来说，就好像是那些行不存在一样。

Dan 发出以下 SQL 语句:

```
SELECT COUNT(*) FROM T1
```

由于用户 Dan 只能读取 Miller 那一行，所以此语句将返回值 1。

读取包含受保护列的受保护行

先检查列访问权，后检查行访问权。如果读访问操作 LBAC 凭证被用来保护其中一个所选列的安全标号阻塞，那么整个语句将失败。否则，该语句将继续执行，并且只返回 LBAC 凭证允许进行读访问的安全标号所保护的行。

示例

表 T1 的列 LASTNAME 受安全标号 L1 保护。列 DEPTNO 受安全标号 L2 保护。列 ROWSECURITYLABEL 的数据类型为 DB2SECURITYLABEL。T1 及其数据如下所示:

表 16. 表 T1 中的示例值

LASTNAME *受 L1 保护*	**DEPTNO** *受 L2 保护*	**ROWSECURITYLABEL**
Rjaibi	55	L2
Miller	77	L1
Fielding	11	L3

假定用户 Sakari 的 LBAC 凭证允许读取受安全标号 L1 保护的数据，但不允许读取受 L2 或 L3 保护的数据。

Sakari 发出以下 SQL 语句:

```
SELECT * FROM T1
```

由于 SELECT 子句使用了通配符（*），这导致包括列 DEPTNO，所以此语句将失败。列 DEPTNO 受安全标号 L2 保护，Sakari 的 LBAC 凭证不允许她读取该安全标号。

Sakari 接着发出以下 SQL 语句:

```
SELECT LASTNAME, ROWSECURITYLABEL FROM T1
```

SELECT 子句未包括 Sakari 无法读取的任何列，因此此语句将继续执行。但是，只返回一行，这是因为其他每一行都受安全标号 L2 或 L3 保护。

表 17. 对表 T1 执行查询的示例输出

LASTNAME	**ROWSECURITYLABEL**
Miller	L1

插入受 LBAC 保护的数据

当尝试将数据插入到受保护列中或将新行插入到具有受保护行的表中时，LBAC 凭证会确定如何处理该 INSERT 语句。

插入到受保护的列中

当尝试将数据插入到受保护列中时，会将用于写操作的 LBAC 凭证与用于保护该列的安全标号进行比较。根据比较结果的不同，访问将被阻塞或允许。

在有关如何比较 LBAC 安全标号的主题中提供了有关如何比较两个安全标号的详细信息。

如果访问被允许，那么该语句正常地继续执行。如果访问被阻塞，插入就会失败，并且将返回错误。

如果正在插入一行，但未提供受保护列的值，那么将插入缺省值（如果有缺省值）。即使您的 LBAC 凭证不允许对该列进行写访问，也会发生这种情况。在下列情况下，有缺省值:

- 声明该列时指定了 WITH DEFAULT 选项
- 该列是生成列
- 该列具有通过前触发器生成的缺省值
- 该列的数据类型为 DB2SECURITYLABEL，在此情况下，您拥有的写访问安全标号是缺省值

插入到受保护的行中

向带有受保护行的表中插入新行时，不必为 DB2SECURITYLABEL 类型的列提供值。如果没有为该列提供值，那么该列将自动填充您拥有的写访问安全标号。如果您没有写访问安全标号，就会返回错误，并且插入操作失败。

通过使用内置函数（例如 SECLABEL），可以显式地提供安全标号以将其插入到 DB2SECURITYLABEL 类型的列中。但是，对于您正在尝试插入的安全标号所保护的数据，仅当您的 LBAC 凭证允许写该数据时，才会使用提供的安全标号。

如果您提供了无法写入的安全标号，那么结果将取决于用于保护表的安全策略。如果该安全策略具有 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么插入操作将失败并返回错误。如果该安全策略没有 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，或者它具有 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么将忽略您提供的安全标号，如果您具有写访问安全标号，那么将使用此安全标号。如果您没有写访问安全标号，就会返回错误。

示例

表 T1 受名为 P1 的安全策略保护，创建该安全策略时未指定 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项。表 T1 包含两列，但未包含任何行。这两列为 LASTNAME 和 LABEL。列 LABEL 的数据类型为 DB2SECURITYLABEL。

用户 Joe 拥有写访问安全标号 L2。假定安全标号 L2 允许他对受安全标号 L2 保护的数据执行写操作，但是不允许他对受安全标号 L1 或 L3 保护的数据执行写操作。

Joe 发出以下 SQL 语句:

```
INSERT INTO T1 (LASTNAME, DEPTNO) VALUES ('Rjaibi', 11)
```

因为 INSERT 语句未包含任何安全标号，所以将在 LABEL 行中插入 Joe 的写访问安全标号。

现在，表 T1 如下所示:

表 18. 示例表 T1 中第一个 INSERT 语句后的值

LASTNAME	LABEL
Rjaibi	L2

Joe 发出以下 SQL 语句，他在该 SQL 语句中显式地提供了要插入到列 LABEL 中的安全标号:

```
INSERT INTO T1 VALUES ('Miller', SECLABEL_BY_NAME('P1', 'L1') )
```

此语句中的 SECLABEL_BY_NAME 函数返回一个安全标号，该安全标号名为 L1，它是安全策略 P1 的组成部分。由于未允许 Joe 对 L1 保护的数据执行写操作，因此不允许他将 L1 插入列 LABEL 中。

因为用于保护 T1 的安全策略是在未指定 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项的情况下创建的，所以插入了 Joe 拥有的写访问安全标号。不会返回任何错误或消息。

现在，该表如下所示:

表 19. 示例表 T1 中第二个 INSERT 语句后的值

LASTNAME	LABEL
Rjaibi	L2
Miller	L2

如果用于保护该表的安全策略是在指定了 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项的情况下创建的，那么插入操作就会失败，并且将返回错误。

接着，Joe 被授予对其中一个 LBAC 规则的免除权。假定新的 LBAC 凭证允许他对受安全标号 L1 和 L2 保护的数据执行写操作。授予 Joe 的写访问安全标号不变，仍是 L2。

Joe 发出以下 SQL 语句:

```
INSERT INTO T1 VALUES ('Bird', SECLABEL_BY_NAME('P1', 'L1') )
```

新的 LBAC 凭证使 Joe 能够对受安全标号 L1 保护的数据执行写操作。因此，允许插入 L1。现在，该表如下所示:

表 20. 示例表 T1 中第三个 INSERT 语句后的值

LASTNAME	LABEL
Rjaibi	L2
Miller	L2
Bird	L1

更新受 LBAC 保护的数据

LBAC 凭证必须允许对数据进行写访问，这样才能更新该数据。如果要更新受保护的行，那么 LBAC 凭证还必须允许对该行进行读访问。

更新受保护的列

在尝试更新受保护列中的数据时，LBAC 凭证将与用于保护该列的安全标号作比较。此比较是针对写访问进行的。如果写访问被阻塞，就会返回错误，并且语句失败，否则继续进行更新。

在有关如何比较 LBAC 安全标号的主题中提供了有关 LBAC 凭证如何与安全标号进行比较的详细信息。

示例:

假定有一个表 T1，其中的列 DEPTNO 受安全标号 L2 保护，而列 PAYSCALE 受安全标号 L3 保护。T1 以及它的数据如下所示:

表 21. 表 T1

EMPNO	LASTNAME	DEPTNO *保护者* L2	PAYSCALE *保护者* L3
1	Rjaibi	11	4
2	Miller	11	7
3	Bird	11	9

用户 Lhakpa 没有 LBAC 凭证。他发出以下 SQL 语句:

```
UPDATE T1 SET EMPNO = 4
   WHERE LASTNAME = "Bird"
```

由于此语句未更新任何受保护的列，所以它的执行不会出错。现在，T1 如下所示:

表 22. 更新之后的表 T1

EMPNO	LASTNAME	DEPTNO *保护者* L2	PAYSCALE *保护者* L3
1	Rjaibi	11	4
2	Miller	11	7
4	Bird	11	9

Lhakpa 接着发出以下 SQL 语句:

```
UPDATE T1 SET DEPTNO = 55
   WHERE LASTNAME = "Miller"
```

由于 DEPTNO 受保护，并且 Lhakpa 没有 LBAC 凭证，所以此语句失败并返回错误。

假定 Lhakpa 被授予 LBAC 凭证，并且那些 LBAC 凭证允许进行下表概括的访问。对于本示例来说，有关那些凭证是什么以及安全标号所包含的元素之类的详细信息并不重要。

用于保护数据的安全标号	是否可读?	是否可写?
L2	否	是
L3	否	否

Lhakpa 再次发出以下 SQL 语句:

```
UPDATE T1 SET DEPTNO = 55
   WHERE LASTNAME = "Miller"
```

这次，由于 Lhakpa 的 LBAC 凭证允许他对用于保护 DEPTNO 列的安全标号所保护的数据执行写操作，所以该语句能够完成执行并且不会出错。他能否读取该列并不重要。现在，T1 中的数据是这样的:

表 23. 第二次更新之后的表 T1

EMPNO	**LASTNAME**	**DEPTNO** *保护者* **L2**	**PAYSCALE** *保护者* **L3**
1	Rjaibi	11	4
2	Miller	55	7
4	Bird	11	9

接着，Lhakpa 发出以下 SQL 语句:

```
UPDATE T1 SET DEPTNO = 55, PAYSCALE = 4
   WHERE LASTNAME = "Bird"
```

PAYSCALE 列受安全标号 L3 保护，Lhakpa 的 LBAC 凭证不允许他写该列。由于 Lhakpa 无法写该列，所以更新将失败，不会更改任何数据。

更新受保护的行

如果 LBAC 凭证不允许您读取某一行，那么该行对您来说就好像不存在一样，因此无法更新该行。对于能够读的行来说，您还必须能够写入该行，这样才能更新该行。

在尝试更新行时，写操作 LBAC 凭证将与保护该行的安全标号作比较。如果写访问被阻塞，那么更新就会失败，并且将返回错误。如果写访问未被阻塞，那么将继续进行更新。

除了处理数据类型为 DB2SECURITYLABEL 的列的方式有所不同以外，执行的更新与更新未受保护的行相同。如果未显式地设置该列的值，就会自动将它设置为受您拥有的写访问安全标号保护。如果您没有写访问安全标号，就会返回错误，并且语句将失败。

如果更新操作显式地设置数据类型为 DB2SECURITYLABEL 的列，那么将再次检查 LBAC 凭证。如果尝试执行的更新操作将创建当前 LBAC 凭证不允许写入的行，那么结果将取决于用于保护表的安全策略。如果该安全策略具有 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么更新操作将失败并返回错误。如果该安全策略没有 RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL 选项，或者它具有 OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL 选项，那么将忽略您提供的安全标号，如果您具有写访问安全标号，那么将使用此安全标号。如果您没有写访问安全标号，就会返回错误。

示例:

假定表 T1 受名为 P1 的安全策略保护并包含名为 LABEL 的列，该列的数据类型为 DB2SECURITYLABEL。

T1 以及它的数据如下所示:

表 24. 表 T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1
2	Miller	11	L2
3	Bird	11	L3

假定用户 Jenni 的 LBAC 凭证允许她读写受安全标号 L0 和 L1 保护的数据，但不允许她读写受任何其他安全标号保护的数据。允许她执行读写操作的安全标号是 L0。对于本示例来说，有关她的完全凭证是什么以及安全标号包含的元素之类的详细信息并不重要。

Jenni 发出以下 SQL 语句:

```
SELECT * FROM T1
```

Jenni 只看到表中的一行:

表 25. Jenni 的 SELECT 查询结果

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	11	L1

由于 Jenni 的 LBAC 凭证不允许她读取受标号 L2 和 L3 保护的行，所以那些行未包括在结果集中。对于 Jenni 来说，就好像是那些行不存在一样。

Jenni 发出下列 SQL 语句:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11;
SELECT * FROM T1;
```

查询返回的结果集如下所示:

表 26. Jenni 的 UPDATE 和 SELECT 查询结果

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0

表中的实际数据如下所示:

表 27. 表 T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L0
2	Miller	11	L2
3	Bird	11	L3

该语句的执行不会出错，但只影响第一行。由于 Jenni 无法读第二行和第三行，因此，尽管它们满足 WHERE 子句中的条件，该语句也不会选择它们来进行更新。

注意，尽管在 UPDATE 语句中未显式地设置 LABEL 列，但在更新后的行中，该列的值已更改。该列已被设置为 Jenni 拥有的写操作安全标号。

现在，Jenni 被授予 LBAC 凭证，此凭证允许她读取受任何安全标号保护的数据。她的写操作 LBAC 凭证未更改。她仍然只能够写受 L0 和 L1 保护的数据。

Jenni 再次发出以下 SQL 语句:

```
UPDATE T1 SET DEPTNO = 44 WHERE DEPTNO = 11
```

这次，更新操作由于第二行和第三行而失败。Jenni 能够读那些行，因此该语句选择了那些行以对其进行更新。但是，由于那些行受安全标号 L2 和 L3 保护，所以她无法写那些行。更新操作不执行，并且将返回错误。

现在，Jenni 发出以下 SQL 语句:

```
UPDATE T1
 SET DEPTNO = 55, LABEL = SECLABEL_BY_NAME( 'P1', 'L2' )
 WHERE LASTNAME = "Rjaibi"
```

该语句中的 SECLABEL_BY_NAME 函数返回了名为 L2 的安全标号。Jenni 尝试显式地设置用于保护第一行的安全标号。Jenni 的 LBAC 凭证允许她读取第一行，因此将选择该行以对其进行更新。她的 LBAC 凭证允许她写受安全标号 L0 保护的行，因此允许她更新该行。但是，她的 LBAC 凭证不允许她写受安全标号 L2 保护的行，因此不允许她将 LABEL 列设置为该值。该语句将失败，并且将返回错误。不会更新该行中的任何列。

现在，Jenni 发出以下 SQL 语句:

```
UPDATE T1 SET LABEL = SECLABEL_BY_NAME( 'P1', 'L1' ) WHERE LASTNAME = "Rjaibi"
```

由于她能够写受安全标号 L1 保护的行，所以该语句将成功。

现在，T1 如下所示:

表 28. 表 T1

EMPNO	LASTNAME	DEPTNO	LABEL
1	Rjaibi	44	L1
2	Miller	11	L2
3	Bird	11	L3

更新包含受保护列的受保护行

如果尝试更新包含受保护行的表中的受保护列，您的 LBAC 凭证就必须允许写所有受该更新操作影响的受保护列，否则更新操作将失败，并且将返回错误。在先前有关更新受保护的列一节中描述了这种情况。即使允许更新所有受更新操作影响的受保护列，也仍然只能够更新 LBAC 凭证既允许读也允许写的行。在先前有关更新受保护的行一节中描述了这种情况。无论更新操作是否影响受保护的列，对数据类型为 DB2SECURITYLABEL 的列的处理方式都是相同的。

如果数据类型为 DB2SECURITYLABEL 的列本身是受保护列，那么 LBAC 凭证必须允许您写该列，否则无法更新表中任何的行。

删除受 LBAC 保护的数据

能否删除受 LBAC 保护的表中的数据取决于 LBAC 凭证。

删除受保护的行

如果 LBAC 凭证不允许您读某一行，那么该行对您来说就好像不存在一样，因此无法删除该行。要删除您能够读取的行，您的 LBAC 凭证还必须允许您写该行。要删除表中任何带有受保护列的行，您必须拥有允许您写该表中的所有受保护列的 LBAC 凭证。

在尝试删除行时，写操作 LBAC 凭证将与保护该行的安全标号作比较。如果保护安全标号阻塞了 LBAC 凭证授予的写访问权，DELETE 语句就会失败，返回错误，并且不删除任何行。

示例

受保护的表 T1 包含下列各行:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2
Fielding	77	L3

假定用户 Pat 的 LBAC 凭证允许她进行下表所归纳的访问:

安全标号	是否允许读访问?	是否允许写访问?
L1	是	是
L2	是	否
L3	否	否

对于本示例，她的 LBAC 凭证以及安全标号的确切详细信息并不重要。

Pat 发出以下 SQL 语句:

```
SELECT * FROM T1 WHERE DEPTNO != 999
```

该语句执行并返回以下结果集:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2

由于 Pat 无权读取 T1 的最后一行，所以该行未包括在结果中。对于 Pat 来说，该行就像不存在一样。

Pat 发出以下 SQL 语句:

```
DELETE FROM T1 WHERE DEPTNO != 999
```

Pat 无权写第一行和第三行，这两行都受 L2 保护。因此，尽管她可以读取这些行，但她无法删除这些行。DELETE 语句将失败，不会删除任何行。

Pat 发出以下 SQL 语句:

```
DELETE FROM T1 WHERE DEPTNO = 77;
```

由于 Pat 能够写 LASTNAME 列包含 Miller 的那一行，所以此语句成功。这就是该语句选择的唯一一行。由于 Pat 的 LBAC 凭证不允许她读取 LASTNAME 列包含 Fielding 的那一行，所以未选择该行。由于决不会删除该行，所以不会发生错误。

现在，该表实际包含下列各行:

LASTNAME	DEPTNO	LABEL
Rjaibi	55	L2
Bird	55	L2
Fielding	77	L3

删除带有受保护列的行

要删除表中任何带有受保护列的行，您必须拥有允许您写该表中的所有受保护列的 LBAC 凭证。如果 LBAC 凭证不允许您写该表中的任何行，删除操作就会失败，并且将返回错误。

如果该表既包含受保护列也包含受保护行，那么为了删除特定的行，您必须拥有允许您写该表中的每个受保护列以及读写所要删除的行的 LBAC 凭证。

示例

在受保护的表 T1 中，列 DEPTNO 受安全标号 L2 保护。T1 包含下列各行:

LASTNAME	DEPTNO *受 L2 保护*	LABEL
Rjaibi	55	L2
Miller	77	L1
Bird	55	L2
Fielding	77	L3

假定用户 Benny 的 LBAC 凭证允许他进行下表中概括的访问:

安全标号	是否允许读访问?	是否允许写访问?
L1	是	是
L2	是	否
L3	否	否

对于本示例，他的 LBAC 凭证以及安全标号的确切详细信息并不重要。

Benny 发出以下 SQL 语句:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

由于 Benny 无权写 DEPTNO 列，所以此语句将失败。

现在，Benny 的 LBAC 凭证已更改，他可以进行下表所概括的访问:

安全标号	是否允许读访问?	是否允许写访问?
L1	是	是

安全标号	是否允许读访问?	是否允许写访问?
L2	是	是
L3	是	否

Benny 再次发出以下 SQL 语句:

```
DELETE FROM T1 WHERE DEPTNO = 77
```

这一次，Benny 有权写 DEPTNO 列，因此删除操作将继续执行。DELETE 语句将只选择 LASTNAME 列值为 Miller 的行。由于 Benny 的 LBAC 凭证不允许他读取 LASTNAME 列包含 Fielding 值的那一行，所以未选择该行。由于此语句未选择删除该行，因此，尽管 Benny 无法写该行，但不会出错。

选择的那一行受安全标号 L1 保护。Benny 的 LBAC 凭证允许他写受 L1 保护的数据，因此删除成功。

现在，T1 表实际包含下列各行:

LASTNAME	**DEPTNO** *受 L2 保护*	**LABEL**
Rjaibi	55	L2
Bird	55	L2
Fielding	77	L3

删除受保护的数据

除非 LBAC 凭证允许写受安全标号保护的列，否则无法删除该列。

无法从表中删除数据类型为 DB2SECURITYLABEL 的列。要除去它，首先必须从表中删除安全策略。删除安全策略后，该表不再受 LBAC 保护，并且列的数据类型会自动从 DB2SECURITYLABEL 更改为 VARCHAR(128) FOR BIT DATA。然后可以删除该列。

LBAC 凭证并不禁止删除整个包含受保护数据的表或数据库。如果您在正常情况下有权删除表或数据库，那么不需要任何 LBAC 凭证就可以执行该操作，即使该数据库包含受保护数据亦如此。

从数据中除去 LBAC 保护

您必须具有 SECADM 权限才能从表中除去安全策略。要从表中除去安全策略，可使用 ALTER TABLE 语句的 DROP SECURITY POLICY 子句。这还会自动除去对表的所有行和所有列的保护。

除去对行的保护

在包含受保护行的表中，每一行都必须由安全标号保护。因此，无法对各个行除去 LBAC 保护。

除非从表中除去安全策略，否则不能改变或除去类型为 DB2SECURITYLABEL 的列。

除去对列的保护

可以通过使用 SQL 语句 ALTER TABLE 的 DROP COLUMN SECURITY 子句来除去对列的保护。要除去对列的保护，除了改变表所需的一般特权和权限外，您还必须有允许您读写该列的 LBAC 凭证。

第 6 章 将系统目录用于安全性信息

有关每个数据库的信息自动在一个称为系统目录的视图集合中维护，系统目录是在生成数据库时创建的。此系统目录描述表、列、索引、程序、特权和其他对象。

下列视图和表函数列示关于用户拥有的特权、授予特权的用户标识和对象所有权的信息:

SYSCAT.COLAUTH
列示列的特权

SYSCAT.DBAUTH
列示数据库特权

SYSCAT.INDEXAUTH
列示索引特权

SYSCAT.MODULEAUTH
列示模块特权

SYSCAT.PACKAGEAUTH
列示程序包特权

SYSCAT.PASSTHRUAUTH
列示服务器特权

SYSCAT.ROLEAUTH
列示角色特权

SYSCAT.ROUTINEAUTH
列示例程（函数、方法和存储过程）特权

SYSCAT.SCHEMAAUTH
列示模式特权

SYSCAT.SEQUENCEAUTH
列示序列特权

SYSCAT.SURROGATEAUTHIDS
列示另一个授权标识可充当其代理的授权标识。

SYSCAT.TABAUTH
列示表和视图的特权

SYSCAT.TBSPACEAUTH
列示表空间特权

SYSCAT.VARIABLEAUTH
列示变量特权

SYSCAT.WORKLOADAUTH
列示工作负载特权

SYSCAT.XSROBJECTAUTH
列示 XSR 对象特权

系统授予用户的特权将让 SYSIBM 作为授权者。SYSADM、SYSMAINT SYSCTRL 和 SYSMON 未在系统目录中列示。

CREATE 和 GRANT 语句在系统目录中设置特权。具有 ACCESSCTRL 和 SECADM 权限的用户可以授予和撤销对系统目录视图的 SELECT 特权。

利用授予的特权来检索权限名

可以使用 PRIVILEGES 和其他管理视图来检索关于数据库中已授予特权的权限名的信息。

关于此任务

例如，以下查询从 PRIVILEGES 管理视图中检索所有显式特权和授予了这些特权的授权标识以及其他信息:

```
SELECT AUTHID, PRIVILEGE, OBJECTNAME, OBJECTSCHEMA, OBJECTTYPE
FROM SYSIBMADM.PRIVILEGES
```

以下查询使用 AUTHORIZATIONIDS 管理视图来查找所有已授予特权或权限的授权标识并显示他们的类型:

```
SELECT AUTHID, AUTHIDTYPE FROM SYSIBMADM.AUTHORIZATIONIDS
```

还可以使用 SYSIBMADM.OBJECTOWNERS 管理视图和 SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID 表函数来查找与安全性相关的信息。

在 V9.1 之前，没有单个系统目录视图包含关于全部特权的信息。对于 V9.1 之前的版本，以下语句检索具有特权的所有权限名:

```
 SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

应定期将此语句检索到的列表与系统安全性工具中定义的用户名和组名的列表比较。然后，可以标识不再有效的那些授权名。

注: 如果您支持远程数据库客户机，有可能只在远程客户机定义了此授权名，而没有在数据库服务器上定义。

利用 DBADM 权限来检索所有名称

如下语句检索被直接授予 DBADM 权限的所有授权名:

关于此任务

```
SELECT DISTINCT GRANTEE, GRANTEETYPE FROM SYSCAT.DBAUTH
   WHERE DBADMAUTH = 'Y'
```

检索有权访问表的名称

可以使用 PRIVILEGES 和其他管理视图来检索关于数据库中已授予特权的权限名的信息。

关于此任务

以下语句检索被直接授权访问具有限定符 JAMES 的表 EMPLOYEE 的所有权限名（及其类型）：

```
SELECT DISTINCT AUTHID, AUTHIDTYPE FROM SYSIBMADM.PRIVILEGES WHERE
OBJECTNAME = 'EMPLOYEE' AND OBJECTSCHEMA = 'JAMES'
```

对于 V9.1 之前的版本，以下查询检索相同的信息:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
        WHERE TABNAME = 'EMPLOYEE'
            AND TABSCHEMA = 'JAMES'
  UNION
  SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
        WHERE TABNAME = 'EMPLOYEE'
            AND TABSCHEMA = 'JAMES'
```

要了解谁可以更新具有限定符 JAME 的表 EMPLOYEE，发出如下语句:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
        WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
              (CONTROLAUTH  = 'Y' OR
            UPDATEAUTH IN ('G','Y'))        UNION
  SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
        WHERE DBADMAUTH = 'Y'
  UNION
  SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
        WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
        PRIVTYPE = 'U'
```

以上语句检索具有 DBADM 权限的任何授权名，以及被直接授予 CONTROL 或 UPDATE 特权的那些名称。

记住某些授权名可以是组，而不只是个别用户。

检索授予用户的所有特权

通过在系统目录视图上进行查询，用户可以检索他们持有的特权的列表和他们授予其他用户的特权的列表。

关于此任务

可以使用 PRIVILEGES 和其他管理视图来检索关于数据库中已授予特权的权限名的信息。例如，以下查询检索授予当前会话授权标识的所有特权:

```
SELECT * FROM SYSIBMADM.PRIVILEGES
WHERE AUTHID = SESSION_USER AND AUTHIDTYPE = 'U'
```

此语句中的关键字 SESSION_USER 是一个专用寄存器，它等于当前用户的权限名的值。

对于 V9.1 之前的版本，下列示例提供类似信息。例如，以下语句检索已直接授予个别权限名 JAMES 的数据库特权的列表:

```
SELECT * FROM SYSCAT.DBAUTH
  WHERE GRANTEE = 'JAMES' AND GRANTEETYPE = 'U'
```

以下语句检索由用户 JAMES 直接授予的表特权的列表:

```
SELECT * FROM SYSCAT.TABAUTH
  WHERE GRANTOR = 'JAMES'
```

以下语句检索由用户 JAMES 直接授予的个别列特权的列表:

```
SELECT * FROM SYSCAT.COLAUTH
    WHERE GRANTOR = 'JAMES'
```

保护系统目录视图

由于系统目录视图描述数据库中的每个对象，因此如果您具有敏感数据，您可能想要限制他们的访问。

关于此任务

下列权限对所有目录表都具有 SELECT 特权:

- ACCESSCTRL
- DATAACCESS
- DBADM
- SECADM
- SQLADM

此外，下列实例级别权限使您能够从 SYSCAT.BUFFERPOOLS、SYSCAT.DBPARTITIONGROUPS、SYSCAT.DBPARTITIONGROUPDEF、SYSCAT.PACKAGES 和 SYSCAT.TABLES 中进行选择:

- SYSADM
- SYSCTRL
- SYSMAINT
- SYSMON

可以使用 CREATE DATABASE ... RESTRICTIVE 命令来创建不自动将特权授予 PUBLIC 的数据库。在此情况下，将不会出现下列任何正常缺省授权操作:

- CREATETAB
- BINDADD
- CONNECT
- IMPLICIT_SCHEMA
- 对模式 SQLJ 中的所有过程的 EXECUTE with GRANT 特权
- 对模式 SYSPROC 中所有函数和过程的 EXECUTE with GRANT 特权
- 对 NULLID 模式中创建的所有包的 BIND 特权
- 对 NULLID 模式中创建的所有包的 EXECUTE 特权

- 对模式 SQLJ 的 CREATEIN 特权
- 对模式 NULLID 的 CREATEIN 特权
- 对表空间 USERSPACE1 的 USE 特权
- 对 SYSIBM 目录表的 SELECT 访问权
- 对 SYSCAT 目录视图的 SELECT 访问权
- 对 SYSIBMADM 管理视图的 SELECT 访问权
- 对 SYSSTAT 目录视图的 SELECT 访问权
- 对 SYSSTAT 目录视图的 UPDATE 访问权

如果创建数据库时使用了 RESTRICTIVE 选项，那么不会向 PUBLIC 授予任何许可权。可运行以下查询以验证 PUBLIC 能否访问任何模式:

```
SELECT DISTINCT OBJECTSCHEMA FROM SYSIBMADM.PRIVILEGES WHERE AUTHID='PUBLIC'

OBJECTSCHEMA
------------
```

对于 DB2 数据库管理器 V9.1 之前的版本，在创建数据库期间，会将对系统目录视图的 SELECT 特权授予 PUBLIC。大多数情况下，这样做不会引起任何安全性问题。但是，对于特别敏感的数据，这可能不恰当，因为这些表描述数据库中的每个对象。如果是这种情况，要考虑从 PUBLIC 撤销 SELECT 特权；然后按需要将 SELECT 特权授予特定用户。授予和撤销对系统目录视图的 SELECT 特权的方式与授予和撤销对任何视图的 SELECT 特权的方式相同，但是必须具有 ACCESSCTRL 或 SECADM 权限，才能完成此类操作。

至少，如果您不想任何用户知道其他用户有权访问哪些对象，应考虑限制对下列目录和管理视图的访问权:

- SYSCAT.COLAUTH
- SYSCAT.DBAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.ROUTINEAUTH
- SYSCAT.SCHEMAAUTH
- SYSCAT.SECURITYLABELACCESS
- SYSCAT.SECURITYPOLICYEXEMPTIONS
- SYSCAT.SEQUENCEAUTH
- SYSCAT.SURROGATEAUTHIDS
- SYSCAT.TABAUTH
- SYSCAT.TBSPACEAUTH
- SYSCAT.XSROBJECTAUTH
- SYSIBMADM.AUTHORIZATIONIDS
- SYSIBMADM.OBJECTOWNERS
- SYSIBMADM.PRIVILEGES

这将防止有关用户特权的信息对可访问该数据库的任何人可用。

还应检查对其收集统计信息的列。记录在系统目录中的某些统计信息包含可能是您环境中的敏感信息的数据值。如果这些统计信息包含敏感数据，可能希望从 PUBLIC 撤销对 SYSCAT.COLUMNS 和 SYSCAT.COLDIST 目录视图的 SELECT 特权。

如果希望限制对系统目录视图的访问，您可以定义视图，来让每个授权名检索有关它自己特权的信息。

例如，如下视图 MYSELECTS 包括每个特定的表的所有者和名称，已将该表的 SELECT 特权直接授予了一个用户的授权名:

```
CREATE VIEW MYSELECTS AS
         SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
         WHERE GRANTEETYPE = 'U'
             AND GRANTEE = USER
             AND SELECTAUTH = 'Y'
```

此语句中的关键字 USER 等于当前会话权限名的值。

如下语句使此视图可供每个授权名使用:

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

最后，应记住要通过发出下列两条语句来撤销对视图和基本表的 SELECT 特权:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
REVOKE SELECT ON TABLE SYSIBM.SYSTABAUTH FROM PUBLIC
```

第 7 章 防火墙支持

*防火墙*是一组相关的程序，位于网络网关服务器上，用来防止对系统或网关进行未授权的访问。

有四种类型的防火墙:

1. 网络级别、包过滤器或屏蔽路由器防火墙
2. 典型应用程序级别代理防火墙
3. 电路级别或透明代理防火墙
4. 有状态的多层检查 (SMLI) 防火墙

存在现有防火墙产品，它合并以上列示的其中一种类型的防火墙。有许多合并以上列示的类型组合的其他防火墙产品。

屏蔽路由器防火墙

屏蔽路由器防火墙也称为网络级别或信息包过滤器防火墙。这种防火墙的工作方式是根据协议属性屏蔽入局信息包。屏蔽的协议属性可能包括源或目标地址、协议类型、源或目标端口或某些其他特定于协议的属性。

对于所有防火墙解决方案（SOCKS 除外），您需要确保 DB2 数据库使用的所有端口对入局和出局信息包开放。DB2 数据库将端口 523 用于 DB2 管理服务器（DAS），此 DAS 由 DB2 数据库工具使用。通过使用 services 文件将服务器数据库管理器配置文件中的服务名称映射至其端口号来确定所有服务器实例使用的端口。

应用程序代理防火墙

代理或代理服务器是一种技术，它充当 Web 客户机与 Web 服务器之间的媒介。代理防火墙充当网关，用于接收来自客户机的请求。

防火墙接收到客户机请求时，由代理软件确定最终服务器目标地址。应用程序代理代表客户机转换地址，执行附加访问控制检查，登录（如果需要）并连接至服务器。

防火墙机器上的 DB2 Connect 产品可以充当目标服务器的代理。另外，在防火墙上充当最终目标服务器的中继段服务器的 DB2 数据库服务器充当应用程序代理的角色。

电路级别防火墙

电路级别防火墙也称为透明代理防火墙。

透明代理防火墙不会修改代理认证和标识所需之外的请求或响应。透明代理防火墙的一个示例是 SOCKS。

DB2 数据库系统支持 SOCKS V4。

有状态的多层检查 (SMLI) 防火墙

有状态的多层检查 (SMLI) 防火墙使用信息包过滤的完善形式来检查组成“开放式系统互连”（OSI）模型的所有七层。

检查每一个信息包并与友好的信息包的已知状态进行比较。屏蔽路由器防火墙只检查信息包头，而 SMLI 防火墙会检查整个信息包，包括数据。

第 8 章 安全插件

DB2 数据库系统的认证是使用安全*插件*来完成的。安全插件是可动态装入的库，该库提供认证安全服务。

组检索插件

检索特定用户的组成员资格信息。

用户标识/密码认证插件

下列认证类型是使用用户标识和密码认证插件实现的:

- CLIENT
- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- DATA_ENCRYPT_CMP

这些认证类型确定如何进行以及在何处进行用户认证。所使用的认证类型由以下方法确定:

- 对于连接操作，如果对 **srvcon_auth** 配置参数指定值，那么该值优先于 **authentication** 配置参数的值。
- 在所有其他情况下，将使用 **authentication** 配置参数的值。

GSS-API 认证插件

GSS-API 的正式名称为 Generic Security Service Application Program Interface V2 (IETF RFC2743) 和 Generic Security Service API V2: C-Bindings (IETF RFC2744)。Kerberos 协议是实现 GSS-API 认证机制的主要手段。下列认证类型是使用 GSS-API 认证插件实现的:

- KERBEROS
- GSSPLUGIN
- KRB_SERVER_ENCRYPT
- GSS_SERVER_ENCRYPT

KRB_SERVER_ENCRYPT 和 GSS_SERVER_ENCRYPT 都支持 GSS-API 认证和用户标识/密码认证。但是，GSS-API 认证是首选认证类型。客户端 Kerberos 支持在 Solaris、AIX、HP-UX（仅 64 位）、Windows 和 Linux 操作系统上可用。对于 Windows 操作系统，缺省情况下会启用 Kerberos 支持。

DB2 数据库管理器在客户机和服务器上支持这些插件。

注: 认证类型确定如何认证以及在何处认证用户。要使用特定认证类型，请设置 **authentication** 数据库管理器配置参数的值。

可独立使用每个插件或与其他插件配合使用。例如，可使用特定服务器端认证插件，但接受客户机和组认证的 DB2 缺省值。或者，您可能只有组检索或客户机认证插件，而没有服务器端插件。

如果要使用 GSS-API 认证，那么客户机和服务器上都需要插件。

认证的缺省行为是使用用于实现操作系统级别机制用户标识/密码插件进行认证。

DB2 数据库产品包括用于组检索、用户标识/密码认证和 GSS-API 认证的插件。可通过开发您自己的插件或向第三方购买插件来进一步定制 DB2 客户机和服务器认证行为。

在 DB2 客户机上部署安全插件

DB2 客户机可支持一个组检索插件和一个用户标识/密码认证插件。

或者，使用 GSS-API 认证插件的客户机通过扫描 DB2 服务器上的已实现 GSS-API 插件列表来确定要使用的插件。与客户机上实现的 GSS-API 认证插件相匹配的第一个认证插件名称就是选中插件。应使用 **srvcon_gssplugin_list** 数据库管理器配置参数来指定已实现的服务器 GSS-API 插件的列表。下图描绘了 DB2 客户机上安全插件的基础结构:

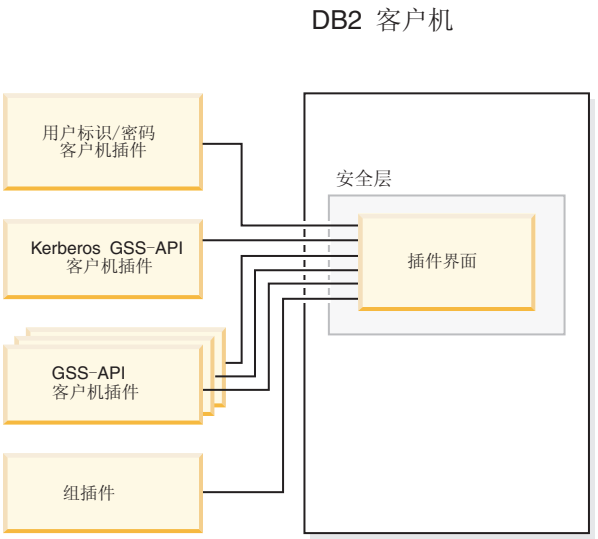


图 5. 在 DB2 客户机上部署安全插件

在 DB2 服务器上部署安全插件

DB2 服务器可支持一个组检索插件、一个用户标识/密码认证插件和多个 GSS-API 插件。可对 **srvcon_gssplugin_list** 数据库管理器配置参数指定值列表形式的可用 GSS-API 插件。但是，此列表中只有一个 GSS-API 插件可作为 Kerberos 插件。

除了在数据库服务器上部署服务器端安全插件外，可能还需要在数据库服务器上部署客户机认证插件。运行实例级别操作（例如，**db2start** 和 **db2trc** 命令）时，DB2 数据库管理器使用客户机认证插件对这些操作执行授权检查。因此，您可能需要安装与服务器上的认证插件相对应的客户机认证插件。此插件名称由服务器上的 **authentication** 数据库管理器配置参数指定。

可将 **authentication** 和 **srvcon_auth** 配置参数设置为不同值。此方案导致使用一种机制来认证数据库连接并使用另一种机制来进行本地授权。

实现此途径的最常见方法是:

- 将 **srvcon_auth** 配置参数设置为 GSSPLUGIN；并
- 将 **authentication** 配置参数设置为 SERVER。

srvcon_auth 配置参数是覆盖入局连接使用的认证类型的一种方法。这些连接使用 **srvcon_auth** 配置参数指定的认证方法，但如果此值留为空，那么会改用 **authentication** 参数的值。

如果未在数据库服务器上使用客户机认证插件，那么实例级别操作（例如，**db2start** 命令）失败。

下图概述了 DB2 服务器上安全认证插件的基础结构:

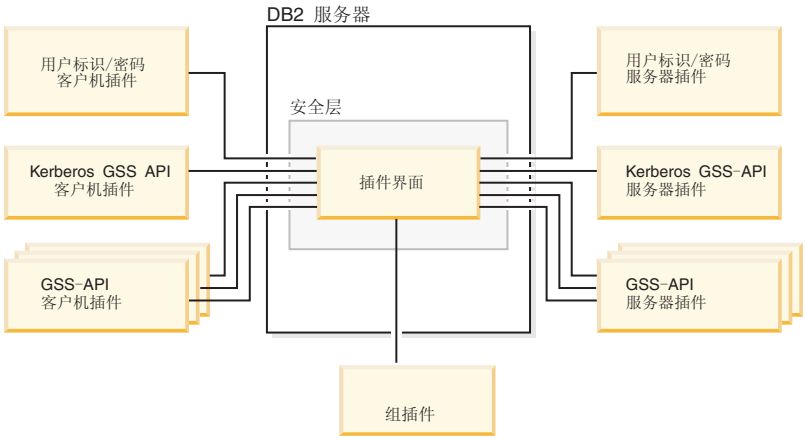


图 6. 在 DB2 服务器上部署安全插件

启用安全插件

可通过设置数据库管理器配置参数来指定要用于每个认证机制的插件。下表概述这些参数:

表 29. 安全认证插件的数据库管理器配置参数

描述	参数名称
客户机用户标识密码插件	CLNT_PW_PLUGIN
客户机 Kerberos 插件	CLNT_KRB_PLUGIN
组插件	GROUP_PLUGIN
本地授权的 GSS 插件	LOCAL_GSSPLUGIN
服务器插件方式	SRV_PLUGIN_MODE
GSS 插件的服务器列表	SRVCON_GSSPLUGIN_LIST
服务器用户标识密码插件	SRVCON_PW_PLUGIN
服务器连接认证	SRVCON_AUTH
数据库管理器认证	AUTHENTICATION

如果未设置这些参数的值，那么 DB2 产品提供的缺省插件用于组检索、用户标识/密码管理和 Kerberos 认证（如果 **authentication** 参数在服务器上设置为 KERBEROS）。但是，未提供缺省 GSS-API 插件。因此，如果对 **authentication** 参数指定认证类型 GSSPLUGIN，那么还必须对 **srvcon_gssplugin_list** 配置参数指定 GSS-API 认证插件。

装入安全插件

数据库管理器配置参数标识的所有受支持插件都是在数据库管理器启动时装入的。

在连接操作期间，DB2 客户机装入适用于客户机与服务器协商时使用的安全性机制的插件。客户机应用程序可导致并行装入和使用多个安全插件。例如，在一个并发连接至不同实例中的不同数据库的多线程程序中就会发生这种情况。在此方案中，客户机程序与使用 GSS-API 插件 (G1) 的服务器 A 建立初始连接。服务器 A 向客户机发送受支持插件列表，系统在客户机上装入匹配的 G1 插件。然后，客户机程序的另一线程连接至使用 GSS-API 插件 (G2) 的服务器 B。系统通知客户机有关 G2 的信息然后装入 G2，现在系统同时在客户机上使用 G1 和 G2 插件。

连接操作以外的操作（例如，更新数据库管理器配置、启动和停止数据库管理器或开启和关闭 DB2 跟踪）也需要授权机制。对于这类操作，DB2 客户机程序装入另一数据库管理器配置参数指定的插件:

- 如果将 **authentication** 配置参数设置为 GSSPLUGIN，那么 DB2 数据库管理器会使用 **local_gssplugin** 配置参数指定的插件。
- 如果将 **authentication** 配置参数设置为 KERBEROS，那么 DB2 数据库管理器会使用 **clnt_krb_plugin** 配置参数指定的插件。
- 否则，DB2 数据库管理器会使用 **clnt_pw_plugin** 配置参数指定的插件。

系统支持对通过 IPv4 和 IPv6 寻址协议与数据库服务器建立的连接使用安全插件。

开发安全插件

如果要开发安全认证插件，那么必须实现 DB2 数据库管理器使用的标准认证功能。您必须对以下三种类型的插件实现该功能:

组检索插件

- 查找并返回用户所属的组的列表

用户标识/密码认证插件

- 标识缺省安全上下文（仅适用于客户机插件）
- 验证并（可选）更改密码
- 确定特定字符串是否表示有效用户（仅适用于服务器插件）
- 修改客户机上提供的用户标识或密码，然后再将它发送至服务器（仅适用于客户机插件）
- 返回与特定用户相关联的 DB2 授权标识

GSS-API 认证插件

- 标识缺省安全上下文（仅适用于客户机插件）
- 实现必需 GSS-API 功能
- 生成基于用户标识和密码的初始凭证并（可选）更改密码（仅适用于客户机插件）
- 创建并接受安全凭单
- 返回与特定 GSS-API 安全上下文相关联的 DB2 授权标识

可对通过 CLP 或动态 SQL 语句发出的连接语句传递最多包含 255 个字符的用户标识。

要点: 如果安全插件未经过充分编码、复查和测试，那么可能会损害 DB2 数据库系统安装的完整性。DB2 数据库产品可预防许多常见类型的故障，但它不能保证部署用户编写的安全插件时的完整性。

安全插件库位置

在获取安全插件之后（无论是您自己开发的还是从第三方购买的），请将它们复制到数据库服务器上的特定位置。

DB2 客户机在以下目录中查找客户端用户认证插件:

- 32 位 UNIX: `$DB2PATH/security32/plugin/client`
- 64 位 UNIX: `$DB2PATH/security64/plugin/client`
- 32 位和 64 位 WINDOWS: `$DB2PATH\security\plugin\`*`instance name`*`\client`

注: 在基于 Windows 的平台上，不会自动创建 *instance name* 和 *client* 这两个子目录。实例所有者必须手动创建这两个目录。

DB2 数据库管理器在以下目录中查找服务器端用户认证插件:

- 32 位 UNIX: `$DB2PATH/security32/plugin/server`
- 64 位 UNIX: `$DB2PATH/security64/plugin/server`
- 32 位和 64 位 WINDOWS: `$DB2PATH\security\plugin\`*`instance name`*`\server`

注: 在基于 Windows 的平台上，不会自动创建 *instance name* 和 *server* 这两个子目录。实例所有者必须手动创建这两个目录。

DB2 数据库管理器在以下目录中查找组插件:

- 32 位 UNIX: `$DB2PATH/security32/plugin/group`
- 64 位 UNIX: `$DB2PATH/security64/plugin/group`
- 32 位和 64 位 WINDOWS: `$DB2PATH\security\plugin\`*`instance name`*`\group`

注: 在基于 Windows 的平台上，不会自动创建 *instance name* 和 *group* 这两个子目录。实例所有者必须手动创建这两个目录。

安全插件命名约定

安全插件库必须具有特定于平台的文件扩展名。使用 C 或 C++ 语言编写的安全插件库必须具有特定于平台的文件扩展名:

- Windows: `.dll`
- AIX: `.a` 或 `.so`; 如果这两个扩展名都存在，那么将使用 `.a` 扩展名。
- Linux、HP IPF 和 Solaris: `.so`

注: 用户还可以使用 DB2 通用 JDBC 驱动程序来开发安全插件。
例如，假定存在一个称为 `MyPlugin` 的安全插件库。对于每个受支持的操作系统，相应的库文件名为如下所示:

- 32 位 Windows: MyPlugin.dll
- 64 位 Windows: MyPlugin64.dll
- 32 位或 64 位 AIX: MyPlugin.a 或 MyPlugin.so
- 32 位或 64 位 SUN、32 位或 64 位 Linux、32 位或 64 位 HP on IPF: MyPlugin.so

注: 只有 64 位 Windows 安全插件的库名的后缀才需要是“64”。

当使用安全插件名称来更新数据库管理器配置时，使用库的全名但不带后缀“64”，并且省略该名称的文件扩展名和任何限定路径部分。无论是哪个操作系统，都将按如下所示注册称为 MyPlugin 的安全插件库:

```
UPDATE DBM CFG USING CLNT_PW_PLUGIN MyPlugin
```

安全插件名称是区分大小写的，并且必须与库名精确匹配。DB2 数据库系统使用相关数据库管理器配置参数的值来组合库路径，然后使用库路径来装入安全插件库。

为了避免安全插件名称发生冲突，那么应使用所用认证方法以及编写插件的公司的识别符号来命名该插件。例如，如果 Foo，Inc. 公司编写了一个用于实现 FOOsomemethod 认证方法的插件，那么就可以将该插件命名为 FOOsomemethod.dll。

插件名称的最大长度（不包括文件扩展名和后缀“64”）只能为 32 个字节。对于数据库服务器可以支持的最大插件数没有限制；但是，在数据库管理器配置中，用逗号分隔的插件列表的最大长度为 255 个字节。位于包含文件 sqlenv.h 中的两个 define 标识了这两个限制:

```
#define SQL_PLUGIN_NAME_SZ     32     /* plug-in name */
#define SQL_SRVCON_GSSPLUGIN_LIST_SZ 255 /* GSS API plug-in list */
```

安全插件库文件必须具有下列文件许可权:

- 归实例所有者所有。
- 系统上的所有用户都可读取该文件。
- 系统上的所有用户都可执行该文件。

“两部分”用户标识的安全插件支持

在 Windows 上，DB2 数据库管理器支持使用“两部分”用户标识，并将“两部分”用户标识映射至“两部分”授权标识。

考虑由域和用户标识组成的 Windows 操作系统“两部分”用户标识，例如: MEDWAY\pieter。在本示例中，MEDWAY 是域，pieter 是用户名。在 DB2 数据库系统中，可以指定应将此“两部分”用户标识映射至“一部分”授权标识还是“两部分”授权标识。

支持将“两部分”用户标识映射至“两部分”授权标识，但这不是缺省行为。缺省情况下，“一部分”用户标识和“两部分”用户标识都映射至“一部分”授权标识。支持将“两部分”用户标识映射至“两部分”授权标识，但这不是缺省行为。

将“两部分”用户标识映射至“一部分”用户标识这一缺省映射允许用户使用以下命令连接至数据库:

```
db2 connect to db user MEDWAY\pieter using pw
```

在此情况下，如果使用了缺省行为，那么用户标识 MEDWAY\pieter 将被解析为授权标识 PIETER。如果支持将“两部分”用户标识映射至“两部分”授权标识，那么授权标识将为 MEDWAY\PIETER。

要使 DB2 能够将“两部分”用户标识映射至“两部分”授权标识，那么 DB2 将提供两组认证插件:

- 一组插件只负责将“一部分”用户标识映射至“一部分”授权标识以及将“两部分”用户标识映射至“一部分”授权标识。

- 另一组插件负责将“一部分”用户标识或“两部分”用户标识都映射至“两部分”授权标识。

如果可以将您所在工作环境中的一个用户名映射至在不同位置定义的多个帐户（例如，本地帐户、域帐户和可信域帐户），那么可以指定支持“两部分”授权标识映射的插件。

一定要注意，“一部分”授权标识（例如 PIETER）与由域和用户标识组合而成的“两部分”授权标识（例如，MEDWAY\pieter）是功能截然不同的两种授权标识。与一个授权标识相关联的特权集可以与跟另一个授权标识相关联的特权集完全不同。使用“一部分”授权标识和“两部分”授权标识时应小心。

下表说明了 DB2 数据库系统提供的插件种类以及特定认证实现的插件名称。

表 30. DB2 安全插件

认证类型	“一部分”用户标识插件的名称	“两部分”用户标识插件的名称
用户标识/密码（客户机）	IBMOSauthclient	IBMOSauthclientTwoPart
用户标识/密码（服务器）	IBMOSauthserver	IBMOSauthserverTwoPart
Kerberos	IBMkrb5	IBMkrb5TwoPart

注: 在 64 位 Windows 平台上，会对此处列示的插件名称追加后缀“64”。

当指定需要“用户标识/密码”插件或 Kerberos 插件的认证类型时，缺省情况下就会使用上述表中““一部分”用户标识插件的名称”这一列中所列示的插件。

要将“两部分”用户标识映射至“两部分”授权标识，必须指定要使用“两部分”插件（不是缺省插件）。安全插件是在实例级别通过设置与安全性相关的数据库管理器配置参数来指定的，如下所示:

对于将“两部分”用户标识映射至“两部分”授权标识的服务器认证，必须进行下列设置:

- 将 srvcon_pw_plugin 设置为 IBMOSauthserverTwoPart
- 将 clnt_pw_plugin 设置为 IBMOSauthclientTwoPart

对于将“两部分”用户标识映射至“两部分”授权标识的客户机认证，必须进行下列设置:

- 将 srvcon_pw_plugin 设置为 IBMOSauthserverTwoPart
- 将 clnt_pw_plugin 设置为 IBMOSauthclientTwoPart

对于将“两部分”用户标识映射至“两部分”授权标识的 Kerberos 认证，必须进行下列设置:

- 将 srvcon_gssplugin_list 设置为 IBMOSkrb5TwoPart
- 将 clnt_krb_plugin 设置为 IBMkrb5TwoPart

安全插件库接受采用与 Microsoft Windows Security Account Manager 兼容的格式指定的“两部分”用户标识。例如，采用以下格式: *domain\user ID*。连接时，DB2 认证和授权进程将使用域和用户标识信息。

创建新的数据库时，应考虑实现“两部分”插件，以避免与现有数据库中的“一部分”授权标识发生冲突。必须在与使用“一部分”授权标识的数据库所在实例不同的实例中创建使用“两部分”授权标识的新数据库。

安全插件 API 版本控制

DB2 数据库系统支持对安全插件 API 的版本进行编号。对于 DB2 UDB V8.2，这些版本号是从 1 开始的整数。

DB2 传递给安全插件 API 的版本号是 DB2 可以支持的 API 的最高版本号，对应于结构的版本号。如果插件可以支持更高的 API 版本，那么它必须返回 DB2 已请求的版本的函数指针。如果插件仅支持更低版本的 API，那么插件应指定更低版本的函数指针。在任何一种情况下，安全插件 API 都应在函数结构的版本字段中返回此 API 支持的版本号。

对于 DB2，仅当需要时才会更改安全插件的版本号（例如，更改了 API 的参数时）。版本号不会随 DB2 发行版号一起自动更改。

32 位和 64 位安全插件的注意事项

通常，32 位 DB2 实例使用 32 位安全插件，而 64 位 DB2 实例使用 64 位安全插件。但是，在 64 位实例上，DB2 也支持 32 位应用程序，这些应用程序需要 32 位插件库。

既可以运行 32 位应用程序又可以运行 64 位应用程序的数据库实例称为混合实例。如果您具有混合实例并且打算运行 32 位应用程序，那么应确保 32 位插件目录中具有必需的 32 位安全插件。对于在 Linux 和 UNIX 操作系统（但不包括 Linux on IPF）上运行的 64 位 DB2 实例，将出现 `security32` 和 `security64` 这两个目录。对于在 Windows on x64 或 IPF 上运行的 64 位 DB2 实例，32 位和 64 位安全插件位于同一目录中，但是 64 位插件名称具有后缀“64”。

如果要从 32 位实例升级到 64 位实例，那么应获取针对 64 位实例重新编译的安全插件版本。

如果您从一个不提供 64 位插件库的供应商处获得了安全插件，那么您可以实现将执行 32 位应用程序的 64 位存根。在此情况下，安全插件是一个外部程序而不是一个库。

安全插件问题确定

安全插件发生的问题是通过以下两种方式来报告的：一种方式是通过 SQL 错误，另一种方式是通过管理通知日志。

以下是与安全插件相关的 SQLCODE 值：

- 如果在执行 **db2start** 或 **db2stop** 期间插件发生错误，那么将返回 SQLCODE -1365。
- 每当发生本地授权问题时就会返回 SQLCODE -1366。
- 插件发生了任何与连接相关的错误，都将返回 SQLCODE -30082。

在调试和管理安全插件时，管理通知日志很适用。在 UNIX 上，要查看管理通知日志文件，请检查 `sqllib/db2dump/`*`instance name`*`.N.nfy`。在 Windows 操作系统上，要查看管理通知日志，请使用“事件查看器”工具。可通过从 Windows 操作系统的“开始”按钮导航至设置 -> 控制面板 -> 管理工具 -> 事件查看器来找到“事件查看器”工具。以下是与安全插件相关的管理通知日志值：

- 13000，它指示因发生错误而调用 GSS-API 安全插件 API 失败，并且返回一条错误消息（可选）。

SQLT_ADMIN_GSS_API_ERROR (13000)
插件"*plug-in name*"从 GSS API"*gss api name*"中接收到错误代码"*error code*"，产生的错误消息为"*error message*"

- 13001，它指示因发生错误而调用 DB2 安全插件 API 失败，并且返回一条错误消息（可选）。

  SQLT_ADMIN_PLUGIN_API_ERROR(13001)
  插件"*plug-in name*"从 DB2 安全性插件 API"*gss api name*"接收到错误代码"*error code*"及错误消息"*error message*"

- 13002，它指示 DB2 未能卸载某个插件。

  SQLT_ADMIN_PLUGIN_UNLOAD_ERROR (13002)
  无法卸载插件"*plug-in name*"。不需要执行进一步的操作。

- 13003，它指示主体名称错误。

  SQLT_ADMIN_INVALID_PRIN_NAME (13003)
  用于"*plug-in name*"的主体名称"*principal name*"无效。请修正此主体名称。

- 13004，它指示插件名称无效。插件名称中不允许存在路径分隔符（在 UNIX 上为"/"，在 Windows 上为"\"）。

  SQLT_ADMIN_INVALID_PLGN_NAME (13004)
  插件名称"*plug-in name*"无效。请修正此插件名称。

- 13005，它指示未能装入安全插件。应确保插件位于正确的目录中，并且更新了相应的数据库管理器配置参数。

  SQLT_ADMIN_PLUGIN_LOAD_ERROR (13005)
  无法装入插件"*plug-in name*"。请验证该插件是否存在以及它所在的目录是否正确。

- 13006，它指示安全插件发生了意外错误。请收集所有 **db2support** 信息，如果有可能，还可以捕获 **db2trc**，然后致电 IBM 支持机构以获取进一步的帮助。

  SQLT_ADMIN_PLUGIN_UNEXP_ERROR (13006)
  插件发生了意外错误。请与 IBM 支持机构联系以获取进一步的帮助。

注：如果您正在 64 位 Windows 数据库服务器上使用一些安全插件，并且您发现某个安全插件发生了装入错误，请参阅有关 32 位和 64 位注意事项和安全插件命名约定的主题。64 位插件库要求库名中包含后缀"64"，但是安全插件数据库管理器配置参数中的条目不应指示此后缀。

启用插件

部署组检索插件

要定制 DB2 安全系统的组检索行为，您可以开发自己的组检索插件，也可以向第三方购买。

开始之前

在获得适合于您所在数据库管理系统的组检索插件之后，就可以部署此插件。

过程

- 要在数据库服务器上部署组检索插件，请执行下列步骤：
  1. 将该组检索插件库复制到服务器的组插件目录中。
  2. 将数据库管理器配置参数 **group_plugin** 更新为插件的名称。
- 要在数据库客户机上部署组检索插件，请执行下列步骤：
  1. 将该组检索插件库复制到客户机的组插件目录中。

2. 在数据库客户机上，将数据库管理器配置参数 **group_plugin** 更新为插件的名称。

部署“用户标识/密码”插件

要定制 DB2 安全系统的“用户标识/密码”认证行为，您可以开发自己的“用户标识/密码”认证插件，也可以向第三方购买。

开始之前

所有基于“用户标识/密码”的认证插件都必须放在客户机插件目录或服务器插件目录中，这取决于期望使用这些插件的方法。如果一个插件放在客户机插件目录中，那么此插件将用于本地授权检查，当客户机尝试连接至服务器时，此插件还将用于验证此客户机。如果此插件放在服务器插件目录中，那么它将用于处理与服务器的入局连接，并且，每当发出 GRANT 语句而没有指定关键字 USER 或 GROUP 时，就会将此插件用于检查授权标识是否存在并且有效。在大多数情况下，“用户标识/密码”认证只需要服务器端插件。尽管通常被认为不是很有用，但还是可以只具有客户机“用户标识/密码”插件。尽管要求客户机和服务器上具有相匹配的“用户标识/密码”插件非常少见，但是您还是可以这样做。

注: 必须停止使用插件的 DB2 服务器或任何应用程序之后才能部署*现有*插件的*新*版本。未定义的行为包括：如果一个插件已经具有了新版本（名称不变），而某个进程仍在使用此插件，那么就会进行限制。当您首次部署插件或者未使用该插件时，此限制将不起作用。

在获得适合于您所在数据库管理系统的“用户标识/密码”认证插件之后，就可以部署这些插件。

过程

- 要在数据库服务器上部署“用户标识/密码”认证插件，请在该数据库服务器上执行下列步骤:
  1. 复制服务器插件目录中的“用户标识/密码”认证插件库。
  2. 将数据库管理器配置参数 **srvcon_pw_plugin** 更新为服务器插件的名称。 当服务器处理 CONNECT 和 **ATTACH** 请求时就会使用此插件。
  3. 执行以下其中一个操作:
     - 将数据库管理器配置参数 **srvcon_auth** 设置为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP 认证类型。或者执行以下操作:
     - 将数据库管理器配置参数 **srvcon_auth** 设置为 NOT_SPECIFIED，并将 **authentication** 设置为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP 认证类型。
- 要在数据库客户机上部署“用户标识/密码”认证插件，请在每台客户机上执行下列步骤:
  1. 复制客户机插件目录中的“用户标识/密码”认证插件库。
  2. 将数据库管理器配置参数 **clnt_pw_plugin** 更新为客户机插件的名称。无论在哪里进行认证都会装入并调用此插件，而不是只有数据库配置参数 **authentication** 设置为 CLIENT 时才会这样做。
- 要在使用“用户标识/密码”认证插件的客户机、服务器或网关上进行本地授权，请在每个客户机、服务器或网关上执行下列步骤:

1. 复制该客户机、服务器或网关上的客户机插件目录中的“用户标识/密码”认证插件库。
2. 将数据库管理器配置参数 **clnt_pw_plugin** 更新为插件的名称。
3. 将 **authentication** 数据库管理器配置参数设置为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP。

部署 GSS-API 插件

要定制 DB2 安全系统的认证行为，您可以使用 GSS-API 来开发自己的认证插件，也可以向第三方购买。

开始之前

对于 Kerberos 之外的插件类型，在客户机和服务器上必须具有相匹配的插件名称和相同的插件类型。客户机和服务器上的插件不必来自同一个供应商，但是它们必须生成和使用兼容的 GSS-API 令牌。由于 Kerberos 插件都已实现标准化，因此，可以接受客户机和服务器上部署的 Kerberos 插件的任意组合。但是，不太标准的 GSS-API 机制的不同实现（例如 *x.509* 证书）可能只与 DB2 数据库系统部分兼容。所有 GSS-API 认证插件都必须放在客户机插件目录或服务器插件目录中，这取决于期望使用这些插件的方法。如果一个插件放在客户机插件目录中，那么此插件将用于本地授权检查，当客户机尝试连接至服务器时也是如此。如果此插件放在服务器插件目录中，那么它将用于处理与服务器的入局连接，并且，每当发出 GRANT 语句而没有指定关键字 USER 或 GROUP 时，就会将此插件用于检查授权标识是否存在并且有效。

注: 必须停止使用插件的 DB2 服务器或任何应用程序之后才能部署*现有*插件的*新*版本。未定义的行为包括：如果一个插件已经具有了新版本（名称不变），而某个进程仍在使用此插件，那么就会进行限制。当您首次部署插件或者未使用该插件时，此限制将不起作用。

在获得适合于您所在数据库管理系统的 GSS-API 认证插件之后，就可以部署这些插件。

过程

- 要在数据库服务器上部署 GSS-API 认证插件，请在服务器上执行下列步骤:
  1. 复制服务器插件目录中的 GSS-API 认证插件库。 可以将许多 GSS-API 插件复制到此目录中。
  2. 将数据库管理器配置参数 **srvcon_gssplugin_list** 更新为一个有序的、以逗号分隔的列表，此列表由安装在 GSS-API 插件目录中的各个插件的名称组成。
  3. 执行以下其中一个操作:
     - 将数据库管理器配置参数 **srvcon_auth** 设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT，这是使服务器能够使用 GSSAPI PLUGIN 认证方法的一种方式。或者执行以下操作:
     - 将数据库管理器配置参数 **srvcon_auth** 设置为 NOT_SPECIFIED，并将 **authentication** 设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT，这是使服务器能够使用 GSSAPI PLUGIN 认证方法的一种方式。
- 要在数据库客户机上部署 GSS-API 认证插件，请在每台客户机上执行下列步骤:

1. 复制客户机插件目录中的 GSS-API 认证插件库。 可以将许多 GSS-API 插件复制到此目录中。客户机通过选取客户机上提供的服务器插件列表中包含的第一个 GSS-API 插件来选择要在 CONNECT 或 ATTACH 操作期间认证的 GSS-API 插件。
2. 可选: 对客户机将访问的数据库进行编目，同时指示客户机将只接受 GSS-API 认证插件作为认证机制。 例如:

   ```
   CATALOG DB testdb AT NODE testnode AUTHENTICATION GSSPLUGIN
   ```

- 要在使用 GSS-API 认证插件的客户机、服务器或网关上进行本地授权，请执行下列步骤:
  1. 复制该客户机、服务器或网关上的客户机插件目录中的 GSS-API 认证插件库。
  2. 将数据库管理器配置参数 **local_gssplugin** 更新为插件的名称。
  3. 将数据库管理器配置参数 **authentication** 设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT。

部署 Kerberos 插件

要定制 DB2 安全系统的 Kerberos 认证行为，您可以开发自己的 Kerberos 认证插件，也可以向第三方购买。

开始之前

如果要部署现有插件的新版本，必须停止 DB2 服务器及使用该插件的任何应用程序。如果部署插件的新版本（名称相同）时某个进程正在使用该插件，那么会发生未定义行为，包括陷阱。

关于此任务

可在数据库服务器或数据库客户机上部署 Kerberos 认证插件。

过程

- 要在数据库服务器上部署 Kerberos 认证插件，请在服务器上执行下列步骤:
  1. 将 Kerberos 认证插件库复制到服务器插件目录。
  2. 更新 **srvcon_gssplugin_list** 数据库管理器配置参数的设置（它是一个有序的逗号定界列表）以包括 Kerberos 服务器插件名称。此列表中只能有一个插件是 Kerberos 插件。如果列表中没有 Kerberos 插件，那么会返回错误。如果列表中有多个 Kerberos 插件，那么会返回错误。如果该配置参数值为空白并且 **authentication** 配置参数设置为 KERBEROS 或 KRB_SVR_ENCRYPT，那么会使用缺省 DB2 Kerberos 插件 IBMkrb5。
  3. 必要时，请设置 **srvcon_auth** 数据库管理器配置参数的值。 如果要部署 Kerberos 插件，那么 **srvcon_auth** 数据库管理器配置参数的可接受值如下所示:
     - KERBEROS
     - KRB_SERVER_ENCRYPT
     - GSSPLUGIN
     - GSS_SERVER_ENCRYPT
     - 空白，但仅当 **authentication** 配置参数设置为此列表中的其中一个先前值时。
- 要在数据库客户机上部署 Kerberos 认证插件，请在客户机上执行以下步骤:
  1. 将 Kerberos 认证插件库复制到客户机插件目录。

2. 将 **clnt_krb_plugin** 数据库管理器配置参数设置为 Kerberos 插件的名称。如果 **clnt_krb_plugin** 配置参数的值为空白，那么客户机不能使用 Kerberos 认证。在 Windows 上，缺省值为 IBMkrb5。只需要对定制 Kerberos 插件改变此值。在 UNIX 上，必须设置此值，因为缺省值为空白。要在使用 Kerberos 认证插件的客户机、服务器或网关上进行本地授权，请执行下列步骤:
   a. 复制该客户机、服务器或网关上的客户机插件目录中的 Kerberos 认证插件库。
   b. 将 **clnt_krb_plugin** 数据库管理器配置参数设置为插件名称。
   c. 将 **authentication** 数据库管理器配置参数设置为 KERBEROS 或 KRB_SERVER_ENCRYPT。
3. 可选: 编目客户机将访问的数据库，指示客户机将仅使用 Kerberos 认证插件。以下示例编目 testdb 数据库:

```
CATALOG DB testdb AT NODE testnode AUTHENTICATION KERBEROS
        TARGET PRINCIPAL service/host@REALM
```

基于 LDAP 的认证和组查询支持

DB2 数据库管理器和 DB2 Connect 通过使用 LDAP 安全插件模块以及透明的 LDAP 来支持基于 LDAP 的认证和组查询功能

AIX 操作系统上已增强基于 LDAP 的认证支持。从 DB2 V9.7 FP1 开始，透明 LDAP 支持也已经扩展到 DB2 产品所支持的相同版本级别的 Linux、HP-UX 和 Solaris 操作系统。LDAP 现在允许使用透明的 LDAP 认证对用户认证和组成员资格进行集中管理。可以将 DB2 实例配置为通过操作系统来认证用户和获取他们的组。然后操作系统又可以通过 LDAP 服务器执行认证。要启用透明的 LDAP 认证，请将 **DB2AUTH** 杂项注册表变量设置为 OSAUTHDB。受支持的操作系统包括:

- AIX
- HP-UX
- Linux
- Solaris

用于实现基于 LDAP 的认证的另一选择是通过使用 LDAP 安全插件。LDAP 安全插件模块允许 DB2 数据库管理器对 LDAP 目录中定义的用户进行认证，以除去对 DB2 产品所支持的相同版本级别的操作系统定义的用户和组要求。受支持的操作系统包括:

- AIX
- HP-UX on Itanium-based HP Integrity Series systems (IA-64)
- Linux on IA32、x64 或 zSeries® 硬件
- Solaris
- Windows

可与安全插件模块配合使用的 LDAP 服务器包括:

- IBM Lotus® Domino® LDAP Server V8.0 及更高版本
- IBM Tivoli® Directory Server (ITDS) V6.2（带有 GSKit 7.0.4.20 及更高版本）及更高版本
- Microsoft Active Directory (MSAD) V2008 及更高版本
- Novell eDirectory V8.8 及更高版本

- OpenLDAP Server V2.4 及更高版本
- Sun Java System Directory Server Enterprise Edition V5.2 FP4 及更高版本
- z/OS Integrated Security Services LDAP Server V1R6 和更高版本

注: 使用 LDAP 插件模块时，必须在 LDAP 服务器上定义与数据库相关联的所有用户。这包括 DB2 实例所有者标识和受防护用户。（通常，在操作系统中定义了这些用户，但是还必须在 LDAP 中定义这些用户。）同样，如果使用 LDAP 组插件模块，那么必须在 LDAP 服务器上定义授权所需要的所有组。这些组包括在数据库管理器配置中定义的 SYSADM、SYSMAINT、SYSCTRL 和 SYSMON 组。

DB2 安全插件模块可用于服务器端认证、客户端认证和组查询（稍后将进行描述）。根据您所在的特定环境，可能需要使用一种、两种或所有这三种类型的插件。

要使用 DB2 安全插件模块，遵循下列步骤:

1. 决定您是需要服务器、客户机或组插件模块，还是需要这些模块的组合形式。
2. 通过设置 IBM LDAP 安全插件配置文件（缺省名称为 `IBMLDAPSecurity.ini`）中的值来配置插件模块。您需要咨询 LDAP 管理员以确定适当的值。
3. 启用插件模块
4. 使用各种 LDAP 用户标识来测试连接。

服务器认证插件

服务器认证插件模块对客户机在 CONNECT 和 ATTACH 语句中提供的用户标识和密码执行服务器验证。必要时，它还提供一种方法将 LDAP 用户标识映射至 DB2 授权标识。如果您想让用户使用他们的 LDAP 用户标识和密码向 DB2 数据库管理器进行认证，那么通常都需要服务器插件模块。

客户机认证插件

客户机认证插件模块用于客户机系统中进行用户标识和密码验证的位置；即，使用 SRVCON_AUTH 或 AUTHENTICATION 设置 CLIENT 来配置 DB2 服务器的位置。客户机验证 CONNECT 或 ATTACH 语句中提供的任何用户标识和密码，并将用户标识发送至 DB2 服务器。请注意，CLIENT 认证的安全很难保证，通常建议不要使用这种认证。

如果数据库服务器上的本地操作系统用户标识不同于与这些用户相关联的 DB2 授权标识，那么可能也需要客户机认证插件模块。在对数据库服务器上的本地命令（例如，`db2start`）执行授权检查之前，可以使用客户端插件将本地操作系统用户标识映射至 DB2 授权标识。

组查询插件

组查询插件模块从 LDAP 服务器中检索特定用户的组成员资格信息。如果要使用 LDAP 来存储您的组定义，那么此插件是必需的。最常见的情况是:

- 所有用户和组都是在 LDAP 服务器中定义的。
- 在数据库服务器上本地定义的任何用户（包括实例所有者和受防护用户）在 LDAP 服务器上也是以同一用户标识定义的。

- 在 DB2 服务器上进行密码验证（即，在服务器的 DBM 配置文件中将 AUTHENTICATION 或 SRVCON_AUTH 设置为 SERVER、 SERVER_ENCRYPT 或 DATA_ENCRYPT 的值）。

通常，仅将服务器认证插件模块和组查询插件模块安装在服务器上就足够了。DB2 客户机通常不需要安装 LDAP 插件模块。

可以仅将 LDAP 组查询插件模块与某些其他形式的认证插件（例如，Kerberos 插件）组合使用。在此情况下，LDAP 组查询插件模块将提供与某个用户相关联的 DB2 授权标识。插件模块将在 LDAP 目录中搜索具有相匹配的 AUTHID_ATTRIBUTE 的用户，然后检索与该用户对象相关联的组。

为认证和组查询配置透明 LDAP (AIX)

从 DB2 V9.7 开始，基于透明 LDAP 的认证和组查询在 AIX 操作系统上受支持。在启用此支持之前，需要先执行一些配置步骤。

开始之前

这些步骤假设 LDAP 服务器符合 RFC 2307 且已将 LDAP 服务器配置为存储用户和组信息。

过程

1. 要为 LDAP 配置 AIX 客户机系统，请执行下列步骤:
   a. 作为具有 root 用户权限的用户登录。
   b. 确保已将 LDAP 客户机文件集安装在 AIX 系统上。 AIX 可与所有三个版本的 LDAP 客户机配合使用: ITDS V5.2（随 AIX V5.3 提供）、ITDS V6.1（随 AIX V6.1 提供）和 ITDS V6.2（随 AIX 扩展包提供）。下列内容显示将 ITDS V5.2 文件集安装在 AIX V5.3 系统上:

```
$ lslpp -l "ldap*"
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  ldap.client.adt            5.2.0.0  COMMITTED  Directory Client SDK
  ldap.client.rte            5.2.0.0  COMMITTED  Directory Client Runtime (No
                                                 SSL)
  ldap.html.en_US.config     5.2.0.0  COMMITTED  Directory Install/Config
                                                 Gd-U.S. English
  ldap.html.en_US.man        5.2.0.0  COMMITTED  Directory Man Pages - U.S.
                                                 英语  ldap.msg.en_US            5.2.0.0  COMMITTED  Directory Messages - U.S.
                                                 英语
Path: /etc/objrepos
  ldap.client.rte            5.2.0.0  COMMITTED  Directory Client Runtime (No
                                                 SSL)
```

   c. 使用带 **-c** 选项的 **mksecldap** 命令来配置客户机。 有关 **mksecldap** 命令以及如何使用该命令来配置客户机的更多信息，请参阅http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.security/doc/security/setup_ldap_sec_info_server.htm
   d. 更新 /etc/security/user 文件中的缺省节。

      一旦您确定已正确配置 LDAP 且已经使用用户填充了 LDAP 目录后，就必须设置缺省用户来使用 LDAP。这将确保您可以使用 LDAP 目录中未受限制的任何用户登录 AIX 客户机。

      /etc/security/user 文件中的 **SYSTEM** 和 **REGISTRY** 属性用于指定认证方法以及用于用户管理的数据库。要启用 LDAP 认证和用户管理，请将缺省节中的 **SYSTEM** 和 **REGISTRY** 属性设置为 LDAP。例如:

```
chsec -f /etc/security/user -s default -a "SYSTEM=LDAP or files"
chsec -f /etc/security/user -s default -a "REGISTRY=LDAP"
```

DB2 支持下列 **SYSTEM** 属性:

- LDAP
- 文件

DB2 支持下列 **REGISTRY** 属性:

- LDAP
- KRB5LDAP
- KRB5ALDAP
- files
- KRB5files
- KRB5Afiles

使用其他 **SYSTEM** 属性或 **REGISTRY** 属性的配置可能有效，但是不受支持。

有关节的 **SYSTEM** 和 **REGISTRY** 属性的更多详细信息，请参阅http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=/com.ibm.aix.files/doc/aixfiles/user.htm?。

有关更多详细信息，请参阅标题为“将 AIX 整合至各种 LDAP 环境”的红皮书，该红皮书位于: http://www.redbooks.ibm.com/abstracts/sg247165.html

2. 要在 DB2 实例上配置透明 LDAP 认证，请执行以下操作:
   a. 将 **DB2AUTH** 杂项注册表变量设置为 OSAUTHDB。作为具有 SYSADM 权限的用户运行 **db2set DB2AUTH=OSAUTHDB**。
   b. 使用 **UPDATE DBM CFG** 命令将数据库服务器实例上的认证设置为下列任何一种:
      - SERVER
      - SERVER_ENCRYPT
      - DATA_ENCRYPT
   c. 确保使用的是缺省值 Client Userid-Password Plugin (clnt_pw_plugin)、Server Userid-Password Plugin (srvcon_pw_plugin) 和 Group Plugin (group_plugin)。
   d. 重新启动 DB2 实例。

使用各种认证方法时的注意事项

AIX 上基于透明 LDAP 的认证和组查询支持将扩展为支持 Kerberos 认证。

其他工作已在 AIX 上完成，以将 Kerberos 认证与透明 LDAP 配合使用。当需要在不同位置管理帐户和使用不同的认证方法（如 Kerberos）时，需要将下列内容包括在 /usr/lib/security/methods.cfg 和 /etc/security/users 中。

在 /usr/lib/security/methods.cfg 中，需要具有下列内容才能拥有文件、LDAP 和 Kerberos 认证。

注: KRB5A 用于将 Microsoft Active Directory 用作 Keberos 密钥分发中心 (KDC)。

对于 LDAP:

```
        program = /usr/lib/security/LDAP
        program_64 =/usr/lib/security/LDAP64
```

对于 KRB5A:

```
        program = /usr/lib/security/KRB5A
        program_64 = /usr/lib/security/KRB5A_64
        options = tgt_verify=no,authonly,is_kadmind_compat=no
```

对于 KRB5:

```
        program = /usr/lib/security/KRB5
        program_64 = /usr/lib/security/KRB5_64
        options = kadmind=no
```

对于 KRB5Afiles:

```
        options = db=BUILTIN,auth=KRB5A
```

对于 KRB5files:

```
        options = db=BUILTIN,auth=KRB5
```

对于 KRB5ALDAP:

```
        options = db=LDAP,auth=KRB5A
```

对于 KRB5LDAP:

```
        options = db=LDAP,auth=KRB5
```

示例

下列示例显示了以不同方式管理的四个帐户。每个都使用了不同的认证方法。

如果 Frank 的帐户存储在文件上并使用文件进行认证，那么 Frank 的节在 /etc/security/users 中类似于下列内容。

```
frank:
        SYSTEM = files
        registry = files
```

如果 Karen 的帐户存储在文件上并使用 Kerberos 进行认证，那么 Karen 的节在 /etc/security/users 中类似于下列内容。

```
karen:
        SYSTEM = KRB5files
        registry = KRB5files
```

如果 Luke 的帐户存储在 LDAP 上并使用 Kerberos 进行认证，那么 Luke 的节在 /etc/security/users 中类似于下列内容。

```
luke:
        SYSTEM = KRB5LDAP
        registry = KRB5LDAP
```

如果 Lucy 的帐户存储在 LDAP 上并使用 LDAP 进行认证，那么 Lucy 的节在 /etc/security/users 中类似于下列内容。

```
lucy:
        SYSTEM = LDAP
        registry = LDAP
```

要确定用户是否定义在 LDAP 上，可以使用下列命令来查询用户。

```
$ lsuser -R LDAP lucy
lucy id=1234 pgrp=staff groups=staff home=/home/lucy shell=/bin/ksh registry=LDAP
```

为认证和组查询配置透明 LDAP (Linux)

从 DB2 V9.7 修订包 1 及更高版本开始，要确保 DB2 数据库服务器在 Linux 操作系统上以透明方式使用基于 LDAP 的认证，请使用可插入认证模块 (PAM)。应已经配置 LDAP 服务器以存储用户和组信息。

开始之前

要在 DB2 数据库上启用对透明 LDAP 的支持，请完成以下任务:

1. 配置操作系统以使用 PAM 认证用户
2. 配置 DB2 实例

这些步骤假设 LDAP 服务器符合 RFC 2307。

过程

1. 要为 LDAP 和 PAM 配置操作系统，请执行以下步骤:
   a. 作为具有 root 用户权限的用户登录。
   b. 确保已安装 nss_ldap 和 pam_ldap 程序包。这两个程序包在 /lib(64) 或 /usr/lib(64) 目录中出现为 libnss_ldap.so 和 libpam_ldap.so。
   c. 通过修改 /etc/ldap.conf 文件以使操作系统能够与 LDAP 服务器绑定，设置操作系统以用作 LDAP 客户机。 以下是样本 /etc/ldap.conf 文件:

```
host <host>            # Address of ldap server
base <base>            # The DN of the search base.
rootbinddn <binddn>    # The bind DN to bind to LDAP
ldap_version 3            # LDAP version
pam_login_attribute uid    # user ID attribute for pam user lookups
nss_base_group <group> # nsswitch configuration pertaining to group
                  # search lookup
```

   d. 在 /etc/ldap.secret 文件中设置密码。应当只有 root 用户能够读取或写入此文件。
   e. 在 /etc/pam.d/db2 中创建或修改 PAM 配置文件。此文件只应由 root 用户读取或写入。您可能必须修改配置文件，这取决于要使用的操作系统的版本。以下是 SUSE Linux Enterprise Server 10 的样本配置文件:

```
auth     sufficient pam_unix2.so
auth     required   pam_ldap.so     use_first_pass
account  sufficient pam_unix2.so
account  required   pam_ldap.so
password required   pam_pwcheck.so
password sufficient pam_unix2.so    use_authtok use_first_pass
password required   pam_ldap.so     use_first_pass
session  required pam_unix2.so
```

   对于 Red Hat Enterprise Linux 5，请按如下所示修改配置文件:

```
#%PAM-1.0

auth    required    pam_env.so
auth    sufficient  pam_unix.so likeauth nullok
auth    sufficient  pam_ldap.so use_first_pass
auth    required    pam_deny.so

account  required   pam_unix.so
account  sufficient pam_succeed_if.so uid < 100 quiet
account  sufficient pam_ldap.so
account  required   pam_permit.so
```

```
password requisite  pam_cracklib.so retry=3 dcredit=-1 ucredit=-1
password sufficient pam_unix.so nullok use_authtok md5 shadowremember=3
password sufficient pam_ldap.so  use_first_pass
password required   pam_deny.so

session  required   pam_limits.so
session  required   pam_unix.so
```

DB2 支持使用 pam_ldap.so、pam_unix.so 和 pam_unix2.so 的 PAM 配置。使用其他 PAM 模块的配置可能有效，但是不受支持。

f. 设置 Linux 系统以通过 LDAP 执行组查询。在 /etc/nsswitch.conf 文件中找到 **group** 和 **passwd** 条目，并确保输入 ldap 作为查询方法。 以下是 **group** 和 **passwd** 条目的示例:

```
group:          files ldap
passwd:         files ldap
```

2. 要配置 DB2 实例以使用透明 LDAP 认证，请执行以下步骤:
   a. 将 **DB2AUTH** 杂项注册表变量设置为 OSAUTHDB。 作为具有 SYSADM 权限的用户发出以下命令:

      ```
      db2set DB2AUTH=OSAUTHDB
      ```

   b. 在服务器上将认证设置为下列任何一种:
      - SERVER
      - SERVER_ENCRYPT
      - DATA_ENCRYPT

   c. 确保使用的是缺省值 Client Userid-Password Plugin (clnt_pw_plugin)、Server Userid-Password Plugin (srvcon_pw_plugin) 和 Group Plugin (group_plugin)。
   d. 重新启动 DB2 实例。

为认证和组查询配置透明 LDAP (HP-UX)

从 DB2 V9.7 FP1 及更高版本开始，要确保 DB2 数据库服务器在 HP-UX 操作系统上以透明方式使用基于 LDAP 的认证，需要使用可插入认证模块 (PAM)。应已经配置 LDAP 服务器以存储用户和组信息。

开始之前

此过程假设 LDAP 服务器符合 RFC 2307。

过程

1. 如果使用的是 IBM Tivoli Directory Server (ITDS) V6.1，那么必须先设置 LDAP 服务器，HP-UX 系统才能与其连接。要在 HP-UX 操作系统上配置 LDAP 服务器，请执行以下步骤:
   a. 在 LDAP 服务器上作为具有 root 用户权限的用户登录。
   b. 发出 **idsldapadd** 命令:

      ```
      idsldapadd -D <root> -w <password> -h <hostname> -p <port> -c -i duaconfigschema.ldif

      where,
      <root> - the bind dn to bind to LDAP
      <password> - the password for bind dn
      <hostname> - hostname of the LDAP server
      <port> - the port LDAP server is running. Default is 389
      <schema.ldif> - LDIF file contains DUAConfigProfle Schema
      ```

如果使用 Netscape 或 Red Hat Directory Server，那么会使用 LDAP-UX 安装程序自动将 duaconfigschema.ldif 中列出的对象类添加至 LDAP 服务器。但是，如果使用 ITDS，那么在 HP-UX Client 上运行 LDAP-UX 安装程序之前，必须手动添加该对象类。

2. 要为 LDAP 和 PAM 配置操作系统，请执行以下步骤:
   a. 作为具有 root 用户权限的用户登录。
   b. 安装 LDAP-UX Client Service 并运行 LDAP-UX 安装程序。 将出现以下屏幕:

```
[ctrl-B]=Go Back screen 2
Hewlett-Packard Company
LDAP-UX Client Services Setup Program
-------------------------------------------------------------------------
Select which Directory Server you want to connect to:
1. Netscape or Red Hat Directory
2. Windows 2000/2003/2003 R2 Active Directory
To accept the default shown in brackets, press the Return key.
Directory Server: [1]:
```

   请选择选项 1，就像您连接至 Netscape 或 Red Hat Directory Server 一样并按照指示信息进行操作。

   有关安装 LDAP-UX 的详细信息，请参阅 *LDAP-UX Client Services B.04.15 Administrator's Guide*。

   c. 编辑 /etc/pam.conf 中的 PAM 配置文件。将以下文本添加至该文件:

```
db2 auth required       libpam_hpsec.so.1
db2 auth sufficient                  libpam_unix.so.1
db2 auth required       libpam_ldap.so.1 use_first_pass
```

   先前的配置首先针对本地文件系统检查用户标识和密码。如果找不到用户或向本地文件系统认证失败，那么它将只执行 LDAP 查询。

   DB2 支持使用 libpam_ldap.so 和 libpam_unix.so 的 PAM 配置。使用其他 PAM 模块的配置可能有效，但是不受支持。

   d. 设置 HP-UX 系统以通过 LDAP 执行组查询。在 /etc/nsswitch.conf 文件中找到 **group** 和 **passwd** 条目，并确保输入 ldap 作为查询方法。 以下是 **group** 和 **passwd** 条目的示例:

```
group:          files ldap
passwd:         files ldap
```

3. 要配置 DB2 实例以使用透明 LDAP 认证，请执行以下步骤:
   a. 将 **DB2AUTH** 杂项注册表变量设置为 OSAUTHDB。 作为具有 SYSADM 权限的用户发出以下命令:

```
db2set DB2AUTH=OSAUTHDB
```

   b. 使用 UPDATE DBM CFG 命令将数据库服务器实例上的认证设置为下列任何一种认证类型:
      - SERVER
      - SERVER_ENCRYPT
      - DATA_ENCRYPT
      - CLIENT
   c. 请确保您使用的是客户机用户标识/密码插件 (clnt_pw_plugin)、服务器用户标识/密码插件 (srvcon_pw_plugin) 和组插件 (group_plugin) 的缺省空值。缺省

插件为 IBMOSauthclient、IBMOSauthserver 和 IBMOSgroups；如果您将插件名称的值保留为空白，那么会暗指这些插件。

d. 重新启动 DB2 实例。

注: 透明 LDAP 未使用 IBMLDAPSecurity.ini。此文件仅供 LDAP 插件模块使用。

为认证和组查询配置透明 LDAP (Solaris)

从 DB2 V9.7 FP1 及更高版本开始，要确保 DB2 数据库服务器在 Solaris 操作系统上以透明方式使用基于 LDAP 的认证，需要使用可插入认证模块 (PAM)。应已经配置 LDAP 服务器以存储用户和组信息。

开始之前

此过程假设 LDAP 服务器符合 RFC 2307。

关于此任务

此任务描述了适用于 Solaris 10 的步骤。这些指示信息对于其他版本的 Solaris 操作系统可能略有不同。

过程

1. 执行以下步骤来为 LDAP 和 PAM 配置操作系统:
   a. 作为具有 root 用户权限的用户登录。
   b. 确保已安装 nss_ldap 和 pam_ldap 程序包。这两个程序包在 /usr/lib 和 /usr/lib/security 目录中显示为 nss_ldap.so 和 pam_ldap.so。
   c. 设置操作系统以用作 LDAP 客户机。 ldapclient(1M) 界面可用于发出 **ldapclient** 命令。以下是样本输出:

```
ldapclient manual -a credentialLevel=proxy \
-a authenticationMethod=simple \
-a proxyDN=<root> \
-a proxyPassword=<password> \
-a defaultSearchBase=<base> \
-a serviceSearchDescriptor=group:<group> \
-a domainName=<domain> \
-a defaultServerList=<IP>
```

其中,

<root>
要绑定到 LDAP 的绑定 dn。这是 LDAP 服务器中允许搜索 LDAP 服务器以取得用户帐户和组的用户条目的 dn

<password>
绑定 dn 的密码

<base>
搜索条件的 dn。此 dn 应比用户和组条目高一个级别

<group>
存储组信息的位置的基本 dn

<domain>
LDAP 服务器的域名

<IP> LDAP 服务器的 IP 地址

有关更多信息，请参阅 ldapclient(1M) 手册。

d. 编辑 /etc/pam.conf 中的 PAM 配置文件。将以下文本添加至该文件:

```
db2 auth requisite          pam_authtok_get.so.1
db2 auth required           pam_unix_cred.so.1
db2 auth sufficient         pam_unix_auth.so.1
db2 auth required           pam_ldap.so.1
```

先前的配置首先针对本地文件系统检查用户标识和密码。如果找不到用户或向本地文件系统认证失败，那么它将只执行 LDAP 查询。

DB2 支持使用 pam_ldap.so 和 pam_unix_auth.so 的 PAM 配置。使用其他 PAM 模块的配置可能有效，但是不受支持。

e. 设置 Solaris 系统以通过 LDAP 执行组查询。在 /etc/nsswitch.conf 文件中找到 **group** 和 **passwd** 条目，并确保输入 ldap 作为查询方法。 以下是 **group** 和 **passwd** 条目的示例:

```
group:          files ldap
passwd:         files ldap
```

2. 执行以下步骤来配置 DB2 实例以使用透明 LDAP 认证:

a. 将 **DB2AUTH** 杂项注册表变量设置为 OSAUTHDB。 作为具有 SYSADM 权限的用户发出以下命令:

```
db2set DB2AUTH=OSAUTHDB
```

b. 在服务器上将认证设置为下列任何一种:

- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT

c. 确保使用的是缺省值 Client Userid-Password Plugin (clnt_pw_plugin)、Server Userid-Password Plugin (srvcon_pw_plugin) 和 Group Plugin (group_plugin)。

d. 重新启动 DB2 实例。

注: 透明 LDAP 未使用 IBMLDAPSecurity.ini。此文件仅供 LDAP 插件模块使用。

配置 LDAP 插件模块

要配置 LDAP 插件模块，需要更新 IBM LDAP 安全插件配置文件以适合您的环境。大多数情况下，您需要咨询 LDAP 管理员以确定适当的配置值。

IBM LDAP 安全插件配置文件的缺省名称和位置是:

- 在 UNIX 上: INSTHOME/sqllib/cfg/IBMLDAPSecurity.ini
- 在 Windows 上: %DB2PATH%\cfg\IBMLDAPSecurity.ini

（可选）可以使用 DB2LDAPSecurityConfig 环境变量来指定此文件的位置。在 Windows 上，应在全局系统环境中设置 DB2LDAPSecurityConfig，以确保 DB2 服务已使用。

下列各表提供了信息来帮助您确定适当的配置值。

表 31. 与服务器相关的值

参数	描述
LDAP_HOST	LDAP 服务器的名称。 这是一个用空格分隔的 LDAP 服务器主机名或 IP 地址 （每个主机名或 IP 地址还可以附带一个端口号）的列表。 例如: `host1[:port] [host2:[port2] ... ]`。 缺省端口号是 389，如果启用了 SSL，那么缺省端口号为 636。
ENABLE_SSL	要启用 SSL 支持，应将 ENABLE_SSL 设置为 TRUE （必须安装 GSKit）。 这是一个可选参数; 它缺省为 FALSE（即，没有 SSL 支持）。
SSL_KEYFILE	SSL 密钥环的路径。 仅当 LDAP 服务器正在使用 GSKit 安装 不会自动信任的证书时才需要密钥文件。 例如: `SSL_KEYFILE = /home/db2inst1/IBMLDAPSecurity.kdb`
SSL_PW	SSL 密钥环密码。例如: `SSL_PW = keyfile-password`

表 32. 与用户相关的值

参数	描述
USER_ OBJECTCLASS	用于用户的 LDAP 对象类。 通常，将 USER_OBJECTCLASS 设置为 `inetOrgPerson`（Microsoft Active Directory 的用户）。 例如: `USER_OBJECTCLASS = inetOrgPerson`
USER_BASEDN	搜索用户时要使用的 LDAP 基本 DN。 如果未指定，那么将从 LDAP 目录的根目录开始搜索 用户。 某些 LDAP 服务器会要求您为此参数指定值。 例如: `USER_BASEDN = o=ibm`

USERID_
ATTRIBUTE
	用于表示用户标识的 LDAP 用户属性。
	USERID_ATTRIBUTE 属性将与 USER_OBJECTCLASS 和
	USER_BASEDN（如果指定了这两个属性）组合在一起，
	以在用户使用未限定的用户标识发出 DB2 CONNECT
	语句时构造 LDAP 搜索过滤器。
	
	例如，如果
	`USERID_ATTRIBUTE = uid`，那么发出以下语句:
	`db2 connect to MYDB user bob using bobpass`
	就会构造以下搜索过滤器:
	`&(objectClass=inetOrgPerson)(uid=bob)`

AUTHID_ ATTRIBUTE	用于表示 DB2 授权标识的 LDAP 用户属性。 通常，此参数的值与 USERID_ATTRIBUTE 的值相同。 例如: `AUTHID_ATTRIBUTE = uid`

表 33. 与组相关的值

参数	描述
GROUP_ OBJECTCLASS	用于组的 LDAP 对象类。 通常这是 groupOfNames 或 groupOfUniqueNames（对于 Microsoft Active Directory，它为 group）。 例如: GROUP_OBJECTCLASS = groupOfNames
GROUP_BASEDN	搜索组时要使用的 LDAP 基本 DN。 如果未指定，那么将从 LDAP 目录的根目录开始搜索组。 某些 LDAP 服务器会要求您为此参数指定值。 例如: GROUP_BASEDN = o=ibm
GROUPNAME_ ATTRIBUTE	用于表示组的名称的 LDAP 组属性。 例如: GROUPNAME_ATTRIBUTE = cn
GROUP_LOOKUP_ METHOD	确定用来查找用户的组成员资格的方法。可能的值包括: • SEARCH_BY_DN - 指示要搜索将用户列示为其成员的组。成员资格是由定义为 GROUP_LOOKUP_ATTRIBUTE 的组属性（通常为 member 或 uniqueMember）指示的。 • USER_ATTRIBUTE - 在此例中，用户的组是作为用户对象本身的属性来列示的。此设置指示要搜索定义为 GROUP_LOOKUP_ATTRIBUTE 的用户属性以获取用户的组（通常，对于 Microsoft Active Directory 为 memberOf，对于 IBM Tivoli Directory Server 为 ibm-allGroups）。 例如: GROUP_LOOKUP_METHOD = SEARCH_BY_DN GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
GROUP_LOOKUP_ ATTRIBUTE	用来确定组成员资格的属性的名称，如对于 GROUP_LOOKUP_METHOD 进行的描述。 例如: GROUP_LOOKUP_ATTRIBUTE = member GROUP_LOOKUP_ATTRIBUTE = ibm-allGroups
NESTED_GROUPS	如果 NESTED_GROUPS 为 TRUE，那么 DB2 数据库管理器将通过尝试查找所找到的每个组的组成员资格来递归搜索组成员资格。 正确处理了循环（例如，A 属于 B，而 B 又属于 A）。 此参数是可选的，其缺省值为 FALSE。

表 34. 其他值

参数	描述
SEARCH_DN 和 SEARCH_PW	如果 LDAP 服务器不支持匿名访问，或者在搜索用户或组时匿名访问不足，那么可以选择定义将用来执行搜索的 DN 和密码。 例如: SEARCH_DN = cn=root SEARCH_PW = rootpassword
DEBUG	将 DEBUG 设置为 TRUE，以将额外信息写入 **db2diag** 日志文件，从而帮助调试与 LDAP 相关的问题。 大多数附加信息都是在 DIAGLEVEL 4 (INFO) 记录的。 DEBUG 参数的缺省值为 false。

启用 LDAP 插件模块

在 DB2 实例目录中找到了已编译的二进制 LDAP 插件模块。

下列各表说明了各个 LDAP 插件模块在 DB2 实例中所处的位置。

表 35. 对于 64 位 UNIX 和 Linux 系统

插件模块类型	位置
服务器	/sqllib/security64/plugin/IBM/server
客户机	/sqllib/security64/plugin/IBM/client
组	/sqllib/security64/plugin/IBM/group

表 36. 对于 32 位 UNIX 和 Linux 系统

插件模块类型	位置
服务器	/sqllib/security32/plugin/IBM/server
客户机	/sqllib/security32/plugin/IBM/client
组	/sqllib/security32/plugin/IBM/group

表 37. 对于 Windows 系统（64 位和 32 位）

插件模块类型	位置
服务器	%DB2PATH%\security\plugin\IBM*instance-name*\server
客户机	%DB2PATH%\security\plugin\IBM*instance-name*\client
组	%DB2PATH%\security\plugin\IBM*instance-name*\group

注: 64 位 Windows 插件模块的文件名中包括数字 64。

使用 DB2 命令行处理器来更新数据库管理器配置以启用您需要的插件模块:

- 对于服务器插件模块:

  ```
  UPDATE DBM CFG USING SRVCON_PW_PLUGIN IBMLDAPauthserver
  ```

- 对于客户机插件模块:

  ```
  UPDATE DBM CFG USING CLNT_PW_PLUGIN IBMLDAPauthclient
  ```

- 对于组插件模块:

  ```
  UPDATE DBM CFG USING GROUP_PLUGIN IBMLDAPgroups
  ```

使用 **db2 terminate** 命令来终止所有正在运行的 DB2 命令行处理器后端进程，然后使用 **db2stop** 和 **db2start** 命令来停止并重新启动实例。

使用 LDAP 用户标识进行连接

在 DB2 实例中配置 LDAP 安全插件之后，用户可以使用各种不同的用户字符串来连接至数据库。

对象 LDAP 目录中的位置是由其专有名称 (DN) 定义的。DN 通常是一个由多部分组成的名称，它反映某种层次结构，例如:

```
cn=John Smith, ou=Sales, o=WidgetCorp
```

*用户标识*是由与用户对象相关联的属性（通常是 **uid** 属性）定义的。它可以是一个简单字符串（例如 jsmith）或者看起来像一个电子邮件地址（例如，jsmith@sales.widgetcorp.com），它反映组织层次结构的一部分。

一个用户的 DB2 *授权标识*是 DB2 数据库中与该用户相关联的名称。

以前，用户通常是在服务器的主机操作系统中定义的，并且用户标识与授权标识相同（尽管授权标识通常采用大写）。DB2 LDAP 插件模块使您能够将 LDAP 用户对象的不同属性与用户标识和授权标识相关联。大多数情况下，用户标识和授权标识可以是同一个字符串，并且可以对 USERID_ATTRIBUTE 和 AUTHID_ATTRIBUTE 使用相同的属性名。但是，在您所处的环境中，如果用户标识属性中通常包含您不想传递至授权标识的其他信息，那么可以在插件初始化文件中配置不同的 AUTHID_ATTRIBUTE。AUTHID_ATTRIBUTE 属性的值是从服务器中检索的，并用作该用户的内部 DB2 表示。

例如，如果您的 LDAP 用户标识看起来像电子邮件地址（例如，jsmith@sales.widgetcorp.com），但是您想只将用户部分（即，jsmith）用作 DB2 授权标识，那么您可以按如下所示来实现:

1. 使包含更短名称的新属性与 LDAP 服务器上的所有用户对象相关联
2. 使用此新属性的名称来配置 AUTHID_ATTRIBUTE

于是，用户通过指定他们的完整 LDAP 用户标识和密码就能够连接至 DB2 数据库，例如:

```
db2 connect to MYDB user 'jsmith@sales.widgetcorp.com' using 'pswd'
```

但是，DB2 数据库管理器在内部使用通过 AUTHID_ATTRIBUTE 检索到的短名称（在此例中为 jsmith）来表示该用户。

在启用和配置某个 LDAP 插件模块之后，用户可以使用多种不同的字符串连接至 DB2 数据库:

- 完整 DN。例如:

  ```
  connect to MYDB user 'cn=John Smith, ou=Sales, o=WidgetCorp'
  ```

- 部分 DN。如果使用部分 DN 和适当的搜索条件 DN（如果定义了）来搜索 LDAP 目录，那么将只搜索到一个匹配项。例如:

  ```
  connect to MYDB user 'cn=John Smith' connect to MYDB user uid=jsmith
  ```

- 简单字符串（不包含等号）。使用 USERID_ATTRIBUTE 来限定该字符串并当作部分 DN 来对待。例如:

  ```
  connect to MYDB user jsmith
  ```

注: 如果在 CONNECT 语句或 **ATTACH** 命令上提供的任何字符串中包含空格或特殊字符，那么必须使用单引号对该字符串进行定界。

组查询的注意事项

组成员资格信息通常是在 LDAP 服务器上作为用户对象属性或组对象属性来表示的:

- 作为用户对象属性

  每个用户对象都具有一个称为 GROUP_LOOKUP_ATTRIBUTE 的属性，可以查询此属性以检索该用户的所有组成员资格。

- 作为组对象属性

  每个组对象都具有一个也称为 GROUP_LOOKUP_ATTRIBUTE 的属性，可以使用此属性来列示作为该组的成员的所有用户对象。可以通过搜索将特定用户对象列示为成员的所有组来枚举该用户所属的组。

许多 LDAP 服务器都可以采用上述任一方法进行配置，某些 LDAP 服务器还支持同时采用上述两种方法进行配置。请咨询 LDAP 管理员以确定您的 LDAP 服务器是如何配置的。

配置 LDAP 插件模块时，可以使用 GROUP_LOOKUP_METHOD 参数来指定应该如何执行组查询:

- 如果需要使用用户对象的 GROUP_LOOKUP_ATTRIBUTE 属性来查找组成员资格，请设置 GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
- 如果需要使用组对象的 GROUP_LOOKUP_ATTRIBUTE 属性来查找组成员资格，请设置 GROUP_LOOKUP_METHOD = SEARCH_BY_DN

许多 LDAP 服务器使用组对象的 GROUP_LOOKUP_ATTRIBUTE 属性来确定成员资格。可以按以下示例中所示对它们进行配置:

```
GROUP_LOOKUP_METHOD = SEARCH_BY_DN
GROUP_LOOKUP_ATTRIBUTE = groupOfNames
```

Microsoft Active Directory 通常将组成员资格作为用户属性来存储，并且可以按以下示例中所示来对它们进行配置:

```
GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
GROUP_LOOKUP_ATTRIBUTE = memberOf
```

IBM Tivoli Directory Server 同时支持上述两种方法。要查询某个用户的组成员资格，可以利用特殊用户属性 **ibm-allGroups**，如以下示例中所示:

```
GROUP_LOOKUP_METHOD = USER_ATTRIBUTE
GROUP_LOOKUP_ATTRIBUTE = ibm-allGroups
```

其他 LDAP 服务器可能会提供相似的特殊属性来帮助检索组成员资格。通常，通过用户属性来检索成员资格比搜索将该用户列示为成员的组的速度更快。

对认证 LDAP 用户或检索组时产生的问题进行故障诊断

如果在认证 LDAP 用户或检索他们的组时遇到问题，那么 **db2diag** 日志文件和管理日志中的信息很适用于帮助进行故障诊断。

发生故障时，LDAP 插件模块通常将记录 LDAP 返回码、搜索过滤器以及其他有用数据。如果启用 LDAP 插件配置文件中的 DEBUG 选项，那么插件模块将在 **db2diag** 日志文件中记录更多信息。虽然这有助于故障诊断，但是建议不要在生产系统中使用此方法，这是因为将所有额外的数据写入单个文件中会增加开销。

确保将数据库管理器中的 **diaglevel** 配置参数设置为 4，以便将捕获 LDAP 插件模块中产生的所有消息。

编写安全插件

DB2 如何装入安全插件

为了使 DB2 数据库系统具有调用安全插件函数所必需的信息，安全插件必须具有正确设置的初始化函数。

每个插件库中必须包含具有由插件类型确定的特定名称的初始化函数:

- 服务器端认证插件: `db2secServerAuthPluginInit()`
- 客户端认证插件: `db2secClientAuthPluginInit()`
- 组插件: `db2secGroupPluginInit()`

此函数称为插件初始化函数。插件初始化函数将对指定插件进行初始化，并为 DB2 提供它调用插件的函数时所需要的信息。插件初始化函数接受下列参数:

- 调用该插件的 DB2 实例可以支持的最高版本号的函数指针结构
- 指向一个结构的指针，该结构包含指向所有需要实现的 API 的指针
- 指向函数的指针，该函数用于将日志消息添加至 **db2diag** 日志文件
- 指向错误消息字符串的指针
- 错误消息的长度

以下是一个组检索插件的初始化函数的函数特征符:

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit(
  db2int32 version,
  void *group_fns,
  db2secLogMessage *logMessage_fn,
  char **errormsg,
  db2int32  *errormsglen);
```

注: 如果将插件库编译为使用 C++ 语言，那么必须使用 `extern "C"` 来声明所有函数。DB2 依赖底层操作系统动态装入器来处理由 C++ 用户编写的插件库中所使用的 C++ 构造函数和析构函数。

初始化函数是插件库中使用规定函数名的唯一函数。其他插件函数是通过从初始化函数返回的函数指针来引用的。当 DB2 服务器启动时就会装入服务器插件。当客户机上需要客户机插件时就会装入这些插件。在 DB2 装入插件库之后，它就会立即解析此初始化函数所在的位置并调用此函数。此函数的特定任务为如下所示:

- 将指针的函数指针强制类型转换为适当的函数结构
- 指定指向库中其他函数的指针
- 指定正在返回的函数指针结构的版本号

DB2 可以潜在地多次调用插件初始化函数。当应用程序动态装入 DB2 客户机库，卸载它，再重新装入它，然后在重新装入之前和之后都在插件中执行认证函数时就会发生这种情况。在此情况下，可能不会先卸载然后重新装入插件库；但是，此行为会随着操作系统的不同而不同。

在执行存储过程或联合系统调用期间，DB2 也会对插件初始化函数发出多个调用，数据库服务器本身可以充当客户机。如果数据库服务器上的客户机和服务器插件位于同一个文件中，那么 DB2 可以调用插件初始化函数两次。

如果该插件检测到多次调用了 db2secGroupPluginInit，那么它应该像指示它终止并重新初始化插件库一样来处理此事件。同样，在再次返回函数指针集之前，插件初始化函数应执行调用 db2secPluginTerm 时将执行的整个清除任务。

在基于 UNIX 或 Linux 的操作系统上运行的 DB2 服务器上，DB2 可以在不同进程中多次潜在地装入并初始化插件库。

对于开发安全插件库的限制

存在某些影响插件库的开发方式的限制。

以下列表概述了开发插件库时存在的限制。

C 链接

插件库必须与 C 链接进行链接。头文件将提供原型和实现插件所需要的数据结构，并且仅对 C/C++ 提供错误代码定义。如果插件库是作为 C++ 代码来编译的，那么必须使用 extern ″C″ 来声明 DB2 在装入时将解析的函数。

不支持 .NET 公共语言运行时

.NET 公共语言运行时（CLR）不支持编译和链接插件库的源代码。

信号处理程序

插件库一定不能安装信号处理程序或者更改信号掩码，因为这样做将妨碍 DB2 的信号处理程序。妨碍 DB2 信号处理程序就会严重妨碍 DB2 的报告功能和从错误（其中包括插件代码本身包含的陷阱）进行恢复的功能。插件库还应该绝不抛出 C++ 异常，因为这也会妨碍 DB2 的错误处理功能。

线程安全

插件库必须保证线程安全并且可重入。插件初始化函数是唯一不需要重入的 API。可以在不同的进程中潜在地多次调用插件初始化函数；在此情况下，插件将清理所有已使用的资源并对它本身重新进行初始化。

出口处理程序和覆盖标准 C 库以及操作系统调用

插件库不应覆盖标准 C 库或操作系统调用。插件库还不应安装出口处理程序或 pthread_atfork 处理程序。不推荐使用出口处理程序，这是因为它们在程序退出之前就可能被卸载。

库依赖项

在 Linux 或 UNIX 上，装入插件库的进程可以是 setuid 或 setgid，这意味着它们将不能依赖 $LD_LIBRARY_PATH、$SHLIB_PATH 或 $LIBPATH 环境变量来查找被依赖库。因此，插件库不应依赖于其他库，除非可以通过其他方法（例如下列情况）来访问任何从属库:

- 位于 /lib 或 /usr/lib 目录中
- 在操作系统范围内指定它们所在的目录（例如，在 Linux 系统上的 ld.so.conf 文件中）
- 在插件库本身的 RPATH 中指定

此限制不适用于 Windows 操作系统。

符号冲突

应尽可能使用能降低发生符号冲突的可能性的任何可用选项（例如，可减少解除绑定外部符号引用的那些选项）来编译和链接插件库。例如，在 HP、Solaris

和 Linux 上使用 ″-Bsymbolic″ 链接程序选项有助于防止发生与符号冲突相关的问题。但是，对于在 AIX 上编写的插件，不要显式或隐式使用 "-brtl" 链接程序选项。

32 位和 64 位应用程序

32 位应用程序必须使用 32 位插件。64 位应用程序必须使用 64 位插件。请参阅有关 32 位和 64 位应用程序的注意事项的主题以了解更多详细信息。

文本字符串

并不能保证输入文本字符串一定以 null 结束，此外，输出字符串并不需要以 null 结束。但是，为所有输入字符串给定了整数长度，并为要返回的长度给定了指向整数的指针。

传递授权标识参数

DB2 传递到插件中的授权标识（authid）参数（这是一个输入 authid 参数）将包含一个大写的授权标识，并且会除去填充的空格。由插件返回给 DB2 的 authid 参数（这是一个输出 authid 参数）不需要任何特殊处理，但是 DB2 将按照内部 DB2 标准将 authid 转换为大写并使用空格将它填满。

对参数的大小限制

插件 API 对参数长度的限制如下:

```
#define DB2SEC_MAX_AUTHID_LENGTH 255
#define DB2SEC_MAX_USERID_LENGTH 255
#define DB2SEC_MAX_USERNAMESPACE_LENGTH 255
#define DB2SEC_MAX_PASSWORD_LENGTH 255
#define DB2SEC_MAX_DBNAME_LENGTH 128
```

特定的插件实现可能会要求或者强制授权标识、用户标识和密码使用更小的最大长度。特别是，在操作系统限制低于上述限制的情况下，随 DB2 数据库系统一起提供的操作系统认证插件将受由操作系统强制施加的最大用户、组和名称空间长度限制。

AIX 中的安全插件库扩展名

在 AIX 系统中，安全插件库的文件扩展名可以为 *.a* 或 *.so*。用来装入插件库的机制取决于所使用的扩展名:

- 文件扩展名为 *.a* 的插件库被认为是包含共享对象成员的归档。这些成员必须命名为 *shr.o*（32 位）或 *shr64.o*（64 位）。单个归档中可以同时包含 32 位和 64 位的成员，且允许将它部署在两种类型的平台上。

  例如，要构建 32 位归档形式的插件库:

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- 文件扩展名为 *.so* 的插件库被认为是可动态装入的共享对象。这种对象是 32 位或 64 位的，这取决于构建此对象时所使用的编译器和链接程序选项。例如，要构建 32 位的插件库:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

在除 AIX 之外的所有平台上，安全插件库始终被认为是可动态装入的共享对象。

派生（Fork）

不应派生插件库，因为文件描述符和套接字在子进程中将重复，这可能会导致

挂起或者不正确的行为。尤其是，当我们在该文件上具有打开的文件描述符时，如果派生子代，那么可能会导致错误的文件锁定冲突。派生还有可能继承许多其他资源（如信号量）。

对安全插件的限制

使用安全插件时存在某些限制。

DB2 数据库系列支持限制

不能使用 GSS-API 插件来认证 Linux、UNIX 和 Windows 上的 DB2 客户机与其他 DB2 系列服务器（例如，DB2 for z/OS）之间的连接。也不能认证从另一个充当客户机的 DB2 数据库系列产品与 Linux、UNIX 或 Windows 上的 DB2 服务器之间的连接。

如果使用 Linux、UNIX 或 Windows 上的 DB2 客户机来连接至其他 DB2 数据库系列服务器，那么可以使用客户端的“用户标识/密码”插件（例如，IBM 提供的操作系统认证插件），也可以编写您自己的“用户标识/密码”插件。还可以使用内置 Kerberos 插件，也可以实现您自己的 Kerberos 插件。

对于 Linux、UNIX 或 Windows 上的 DB2 客户机，不应使用 GSSPLUGIN 认证类型来编目数据库。

对 AUTHID 标识的限制: DB2 数据库系统的 V9.5 和更高版本允许您使用 128 个字节的授权标识，但是，当授权标识被解释为操作系统用户标识或组名时，应遵循操作系统命名限制（例如，用户标识的长度限制为 8 到 30 个字符，组名为 30 个字符）。因此，虽然您可以授予一个 128 字节的授权标识，但是作为一个具有该授权标识的用户，您却无法进行连接。如果您编写自己的安全插件，那么应该能够充分利用授权标识的扩展大小。例如，您可以为安全插件指定一个 30 字节的用户标识，并且在认证期间它可能会返回一个 128 字节的授权标识，您可以使用此授权标识进行连接。

InfoSphere® Federation Server 支持限制

DB2 II 不支持使用 GSS_API 插件中的委托凭证来建立与数据源的出站连接。与数据源的连接必须继续使用 CREATE USER MAPPING 命令。

数据库管理服务器支持限制

DB2 管理服务器 (DAS) 不支持安全插件。DAS 仅支持操作系统认证机制。

DB2 客户机的安全插件问题和限制 (Windows)

在开发将部署在 Windows 操作系统上的 DB2 客户机中的安全插件时，请不要卸载插件终止函数的任何辅助库。此限制适用于所有类型的客户机安全插件，包括组插件、“用户标识/密码”插件、Kerberos 插件和 GSS-API 插件。由于在任何 Windows 平台上都不会调用这些终止 API（例如，db2secPluginTerm、db2secClientAuthPluginTerm 和 db2secServerAuthPluginTerm），因此，您需要执行适当的资源清除。

此限制跟与卸载 Windows 上的 DLL 相关联的清除问题相关。

在 AIX 上装入扩展名为 .a 或 .so 的插件库

在 AIX 上，安全插件库的文件扩展名可以为 `.a` 或 `.so`。用来装入插件库的机制取决于所使用的扩展名:

- 文件扩展名为 .a 的插件库

  文件扩展名为 .a 的插件库被认为是包含共享对象成员的归档。这些成员必须命名为 shr.o（32 位）或 shr64.o（64 位）。单个归档中可以同时包含 32 位和 64 位的成员，且允许将它部署在两种类型的平台上。

  例如，要构建 32 位归档形式的插件库:

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- 文件扩展名为 .so 的插件库

  文件扩展名为 .so 的插件库被认为是可动态装入的共享对象。这种对象是 32 位或 64 位的，这取决于构建此对象时所使用的编译器和链接程序选项。例如，要构建 32 位的插件库:

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

在除 AIX 之外的所有平台上，安全插件库始终被认为是可动态装入的共享对象。

GSS-API 安全插件不支持消息加密和签名

消息加密和签名在 GSS-API 安全插件中不可用。

安全插件的返回码

所有安全插件 API 必须返回一个整数值以指示执行该 API 是成功还是失败。返回码值 0 指示已成功运行该 API。除 -3、-4 和 -5 之外的所有负数返回码都指示该 API 遇到了错误。

从安全插件 API 返回的所有负数返回码（-3、-4 和 -5 除外）都映射至 SQLCODE -1365、SQLCODE -1366 或 SQLCODE -30082。值 -3、-4 和 -5 用来指示授权标识是否表示有效的用户或组。

所有安全插件 API 返回码都是在 db2secPlugin.h 文件中定义的，可以在 DB2 包含目录 SQLLIB/include 中找到此文件。

下表中提供了与所有安全插件返回码有关的详细信息:

表 38. 安全插件返回码

返回码	定义值	含义	适用的 API
0	DB2SEC_PLUGIN_OK	成功执行了插件 API。	所有
-1	DB2SEC_PLUGIN_UNKNOWNERROR	插件 API 发生了意外错误。	所有
-2	DB2SEC_PLUGIN_BADUSER	未定义作为输入传入的用户标识。	db2secGenerateInitialCred db2secValidatePassword db2secRemapUserid db2secGetGroupsForUser
-3	DB2SEC_PLUGIN _INVALIDUSERORGROUP	没有这样的用户或组。	db2secDoesAuthIDExist db2secDoesGroupExist

表 38. 安全插件返回码 (续)

返回码	定义值	含义	适用的 API
-4	DB2SEC_PLUGIN_USERSTATUSNOTKNOWN	未知用户状态。DB2 并不认为这是一个错误；GRANT 语句使用它来确定 authid 是表示一个用户还是操作系统组。	db2secDoesAuthIDExist
-5	DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN	未知组状态。DB2 并不认为这是一个错误；GRANT 语句使用它来确定 authid 是表示一个用户还是操作系统组。	db2secDoesGroupExist
-6	DB2SEC_PLUGIN_UID_EXPIRED	用户标识到期。	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-7	DB2SEC_PLUGIN_PWD_EXPIRED	密码到期。	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-8	DB2SEC_PLUGIN_USER_REVOKED	撤销了用户。	db2secValidatePassword db2GetGroupsForUser
-9	DB2SEC_PLUGIN_USER_SUSPENDED	用户被暂挂。	db2secValidatePassword db2GetGroupsForUser
-10	DB2SEC_PLUGIN_BADPWD	密码错误。	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-11	DB2SEC_PLUGIN_BAD_NEWPASSWORD	新密码错误。	db2secValidatePassword db2secRemapUserid
-12	DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED	不支持更改密码。	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-13	DB2SEC_PLUGIN_NOMEM	由于内存不足，因此插件尝试分配内存失败。	所有
-14	DB2SEC_PLUGIN_DISKERROR	插件遇到了磁盘错误。	所有
-15	DB2SEC_PLUGIN_NOPERM	由于插件对某个文件的许可权错误，因此插件尝试访问该文件时失败。	所有
-16	DB2SEC_PLUGIN_NETWORKERROR	插件遇到了网络错误。	所有
-17	DB2SEC_PLUGIN_CANTLOADLIBRARY	插件无法装入必需的库。	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-18	DB2SEC_PLUGIN_CANT_OPEN_FILE	插件无法打开并读取某个文件，但并不是因为缺少该文件或者文件许可权不足。	所有

表 38. 安全插件返回码 (续)

返回码	定义值	含义	适用的 API
-19	DB2SEC_PLUGIN_FILENOTFOUND	插件无法打开并读取某个文件，因为文件系统中不存在该文件。	所有
-20	DB2SEC_PLUGIN _CONNECTION_DISALLOWED	插件拒绝连接，因为允许数据库进行连接时受到限制，或者 TCP/IP 地址不能连接至特定数据库。	所有服务器端插件 API。
-21	DB2SEC_PLUGIN_NO_CRED	仅对于 GSS API 插件：缺少初始客户机凭证。	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-22	DB2SEC_PLUGIN_CRED_EXPIRED	仅对于 GSS API 插件：客户机凭证到期。	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-23	DB2SEC_PLUGIN _BAD_PRINCIPAL_NAME	仅对于 GSS API 插件：主体名称无效。	db2secProcessServer PrincipalName
-24	DB2SEC_PLUGIN _NO_CON_DETAILS	此返回码由 db2secGetConDetails 回调返回（例如，从 DB2 返回给插件），以指示 DB2 无法确定客户机的 TCP/IP 地址。	db2secGetConDetails
-25	DB2SEC_PLUGIN _BAD_INPUT_PARAMETERS	调用插件 API 时，某些参数无效或者不存在。	所有
-26	DB2SEC_PLUGIN _INCOMPATIBLE_VER	插件报告的 API 的版本与 DB2 不兼容。	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-27	DB2SEC_PLUGIN_PROCESS_LIMIT	没有足够资源可用于插件来创建新的进程。	所有
-28	DB2SEC_PLUGIN_NO_LICENSES	插件遇到了用户许可证问题。可能底层机制许可证已达到限制。	所有
-29	DB2SEC_PLUGIN_ROOT_NEEDED	插件尝试运行需要 root 用户特权的应用程序。	所有
-30	DB2SEC_PLUGIN_UNEXPECTED_ SYSTEM_ERROR	插件遇到意外系统错误。当前系统配置可能不受支持。	所有

针对安全插件的错误消息处理

当安全插件 API 发生错误时，此 API 会在 errormsg 字段中返回一个 ASCII 文本字符串，它比返回码能对问题提供更具体的描述。

例如，errormsg 字符串中可以包含 "File /home/db2inst1/mypasswd.txt does not exist." DB2 将把此字符串完整地写入 DB2 管理通知日志，在某些 SQL 消息中还将包含此字符串的阶段版本作为一个标记。因为 SQL 消息中的标记仅长度受到限制，所以应使这些消息保持比较短，并且这些消息的可以改变的重要部分应位于该字符串的前端。为了帮助调试，可考虑将安全插件的名称添加至错误消息。

对于非紧急的错误（例如，密码到期错误），仅当数据库管理器配置参数 DIAGLEVEL 设置为 4 时才将转储 errormsg 字符串。

安全插件必须分配用于存放这些错误消息的内存。因此，插件还必须提供以下 API 来释放此内存：db2secFreeErrormsg。

仅当 API 返回非 0 值时，DB2 才会检查 errormsg 字段。因此，如果没有发生错误，插件就不应为返回的此错误消息分配内存。

在初始化时，会将消息记录函数指针 logMessage_fn 传递给组、客户机和服务器插件。这些插件可以使用此函数将所有调试信息记录到 **db2diag** 日志文件中。例如：

```
// Log an message indicate init successful
(*(logMessage_fn))(DB2SEC_LOG_CRITICAL,
                   "db2secGroupPluginInit successful",
                    strlen("db2secGroupPluginInit successful"));
```

有关 db2secLogMessage 函数的每个参数的更多详细信息，请参阅每种插件类型的初始化 API。

安全插件 API 的调用顺序

根据调用安全插件 API 时情况的不同，DB2 数据库管理器调用安全插件 API 的顺序会有变化。

以下是 DB2 数据库管理器在其中调用安全插件 API 的主要方案：

- 在（隐式和显式）数据库连接的客户机上
  - CLIENT
  - 基于服务器（SERVER、SERVER_ENCRYPT 和 DATA_ENCRYPT）
  - GSSAPI 和 Kerberos
- 在本地授权的客户机、服务器或网关上
- 在数据库连接的服务器上
- 在 GRANT 语句的服务器上
- 在要获取授权标识所属组的列表的服务器上

注：DB2 数据库服务器像客户机应用程序一样来对待需要本地授权的数据库操作，例如 **db2start**、**db2stop** 和 **db2trc**。

对于上述每一项操作，DB2 数据库管理器调用安全插件 API 的顺序是不同的。以下是在上述每种方案下 DB2 数据库管理器调用 API 的顺序。

CLIENT - 隐式

当用户配置的认证类型为 CLIENT 时，DB2 客户机应用程序将调用下列安全插件 API：

- db2secGetDefaultLoginContext();
- db2secValidatePassword();
- db2secFreetoken();

对于隐式认证，即，在未指定特定用户标识或密码的情况下进行连接时，如果您正在使用用户标识/密码插件，那么将调用 db2secValidatePassword API。此 API 允许插件开发者在必要时禁止隐式认证。

CLIENT - 显式

在显式认证时，即，在同时指定了用户标识和密码的情况下连接至数据库时，

如果 **authentication** 数据库管理器配置参数设置为 CLIENT，那么 DB2 客户机应用程序将多次调用下列安全插件 API（如果实现要求这样做）：

- db2secRemapUserid();
- db2secValidatePassword();
- db2secFreeToken();

基于服务器（SERVER、SERVER_ENCRYPT 和 DATA_ENCRYPT）- 隐式

在隐式认证时，当客户机和服务器协商用户标识/密码认证（例如，当服务器中的 **srvcon_auth** 参数设置为 SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP 时），客户机应用程序将调用下列安全插件 API:

- db2secGetDefaultLoginContext();
- db2secFreeToken();

基于服务器（SERVER、SERVER_ENCRYPT 和 DATA_ENCRYPT）- 显式

在显式认证时，当客户机和服务器协商用户标识/密码认证（例如，当服务器中的 **srvcon_auth** 参数设置为 SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP 时），客户机应用程序将调用下列安全插件 API:

- db2secRemapUserid();

GSSAPI 和 Kerberos - 隐式

在隐式认证时，当客户机和服务器协商 GSS-API 或 Kerberos 认证（例如，当服务器中的 **srvcon_auth** 参数设置为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时），客户机应用程序将调用下列安全插件 API。（调用 gss_init_sec_context() 时使用 GSS_C_NO_CREDENTIAL 作为输入凭证。）

- db2secGetDefaultLoginContext();
- db2secProcessServerPrincipalName();
- gss_init_sec_context();
- gss_release_buffer();
- gss_release_name();
- gss_delete_sec_context();
- db2secFreeToken();

借助多流 GSS-API 支持，可以多次调用 gss_init_sec_context()（如果实现要求这样做）。

GSSAPI 和 Kerberos - 显式

如果协商的认证类型是 GSS-API 或 Kerberos，那么客户机应用程序将按以下顺序调用 GSS-API 插件的下列安全插件 API。除非另有声明，否则这些 API 将同时用于隐式和显式认证。

- db2secProcessServerPrincipalName();
- db2secGenerateInitialCred();（仅适用于显式认证）
- gss_init_sec_context();
- gss_release_buffer ();
- gss_release_name();
- gss_release_cred();
- db2secFreeInitInfo();

- gss_delete_sec_context();
- db2secFreeToken();

如果从服务器返回了相互认证令牌并且实现也需要它，那么可能会多次调用 gss_init_sec_context() API。

在本地授权的客户机、服务器或网关上

对于本地授权，所使用的 DB2 命令会调用下列安全插件 API:

- db2secGetDefaultLoginContext();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

对于“用户标识/密码”和 GSS-API 认证机制都会调用这些 API。

在数据库连接的服务器上

对于数据库服务器上的数据库连接，DB2 代理进程或线程会对“用户标识/密码”认证机制调用下列安全插件 API:

- db2secValidatePassword();（仅当数据库配置参数 **authentication** 不是 CLIENT 时）
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

要连接 (CONNECT) 至数据库，DB2 代理进程或线程将对 GSS-API 认证机制调用下列安全插件 API:

- gss_accept_sec_context();
- gss_release_buffer();
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- gss_delete_sec_context();
- db2secFreeGroupListMemory();

在 GRANT 语句的服务器上

对于未指定 USER 或 GROUP 关键字的 GRANT 语句（例如，"GRANT CONNECT ON DATABASE TO user1"），DB2 代理进程或线程必须能够确定 user1 是一个用户和/或组。因此，DB2 代理进程或线程将调用下列安全插件 API:

- db2secDoesGroupExist();
- db2secDoesAuthIDExist();

在要获取授权标识所属组的列表的服务器上

在数据库服务器中，当您需要获取授权标识所属组的列表时，DB2 代理进程或线程会调用下列安全插件 API 并且只将授权标识作为输入:

- db2secGetGroupsForUser();

将没有来自其他安全插件的令牌。

第 9 章 安全插件 API

为了使您能够定制 DB2 数据库系统认证和组成员资格查询行为，DB2 数据库系统提供了一些 API，您可以使用这些 API 来修改现有插件模块或者构建新的安全插件模块。

开发安全插件模块时，需要实现 DB2 数据库管理器将调用的标准认证或组成员资格查询功能。对于三种可用的插件模块类型，您需要实现下列功能:

组检索 检索给定用户的组成员资格信息并确定给定的字符串是否表示一个有效组名。

用户标识/密码认证
这种认证将确定缺省安全上下文（仅适用于客户机）、验证密码和（可选）更改密码、确定给定的字符串是否表示一个有效用户（仅适用于服务器）、在将客户机上提供的用户标识或密码发送至服务器之前修改此用户标识或密码（仅适用于客户机）、返回与给定用户相关联的 DB2 授权标识。

GSS-API 认证
这种认证将实现必需的 GSS-API 功能、确定缺省安全上下文（仅适用于客户端）、根据用户标识和密码生成初始凭证、（可选）更改密码（仅适用于客户端）、创建和接受安全凭单以及返回与给定 GSS-API 安全上下文相关联的 DB2 授权标识。

以下列表显示描述插件 API 时使用的术语的定义。

插件 一个可动态装入的库，DB2 将装入该库以访问由用户编写的认证或组成员资格查询功能。

隐式认证
在未指定用户标识或密码的情况下连接至数据库。

显式认证
在同时指定了用户标识和密码的情况下连接至数据库。

Authid
一个内部标识，表示将数据库中的权限和特权授予的个人或组。在内部，DB2 authid 被转换为大写，其最小长度为 8 个字符（不足 8 个字符则填充空格）。当前，DB2 需要可以用 7 位 ASCII 表示的授权标识、用户标识、密码、组名、名称空间和域名。

本地授权
在实现授权的服务器或客户机本地进行的授权，将检查用户是否有权执行除了连接至数据库之外的操作，例如，启动和停止数据库管理器，打开和关闭 DB2 跟踪或者更新数据库管理器配置。

名称空间
由许多用户组成的集合或分组，其中的每个用户标识都必须是唯一的。名称空间的常见示例包括 Windows 域和 Kerberos 域。例如，在 Windows 域“usa.company.com”中，所有用户名都必须是唯一的。例如，“user1@usa.company.com”。但是，另一个域中的同一个用户标识（例如，“user1@canada.company.com”）却表示另一个用户。标准用户标识包括用户标识和名称空间对；例如，“user@domain.name”或“domain\user”。

输入　指示 DB2 将为安全插件 API 参数输入值。

输出　指示安全插件 API 将为 API 参数指定值。

组检索插件的 API

对于组检索插件模块，需要实现下列 API:

- db2secGroupPluginInit

  注: db2secGroupPluginInit API 通过以下原型将指向 API 的指针 *logMessage_fn 作为输入:

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
);
```

  db2secLogMessage API 允许插件将消息记录到 **db2diag** 日志文件中，以便进行调试或参考。此 API 是由 DB2 数据库系统提供的，因此您不需要实现此 API。

- db2secPluginTerm
- db2secGetGroupsForUser
- db2secDoesGroupExist
- db2secFreeGroupListMemory
- db2secFreeErrormsg
- 唯一需要在外部可解析的 API 是 db2secGroupPluginInit。此 API 将采用 void * 参数，应将它强制类型转换为以下类型:

```
typedef struct db2secGroupFunctions_1
{
db2int32 version;
db2int32 plugintype;
SQL_API_RC (SQL_API_FN * db2secGetGroupsForUser)
(
const char *authid,
db2int32    authidlen,
const char *userid,
db2int32    useridlen,
const char *usernamespace,
db2int32    usernamespacelen,
db2int32    usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
const void *token,
db2int32    tokentype,
db2int32    location,
const char *authpluginname,
db2int32    authpluginnamelen,
void      **grouplist,
db2int32   *numgroups,
char    **errormsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secDoesGroupExist)
(
const char *groupname,
db2int32    groupnamelen,
char      **errormsg,
```

```
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeGroupListMemory)
(
void     *ptr,
char    **errormsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *msgtobefree
);

SQL_API_RC (SQL_API_FN * db2secPluginTerm)
(
char    **errormsg,
db2int32 *errormsglen
);

} db2secGroupFunctions_1;
```

db2secGroupPluginInit API 指定了外部可用的其余函数的地址。

注: _1 指示这是与版本 1 的 API 相对应的结构。后续接口版本的扩展名将依次为 _2、_3 等等。

db2secDoesGroupExist API - 检查组是否存在

确定 authid 是否表示一个组。

如果 **groupname** 存在，那么此 API 必须能够返回值 DB2SEC_PLUGIN_OK 以指示成功。如果组名无效，那么它还必须能够返回值 DB2SEC_PLUGIN_INVALIDUSERORGROUP。如果不能确定输入是否是有效的组，那么允许此 API 返回值 DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN。如果返回了“组无效”(DB2SEC_PLUGIN_INVALIDUSERORGROUP) 或者“未知组”(DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN) 值，那么在不带关键字 USER 和 GROUP 的情况下发出 GRANT 语句时，DB2 for Linux, UNIX, and Windows 可能无法确定 authid 是一个组还是用户，这将导致对用户返回 SQLCODE -569 或 SQLSTATE 56092 错误。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secDoesGroupExist)
                       ( const char *groupname,
                         db2int32 groupnamelen,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secDoesGroupExist API 参数

groupname

输入。一个授权标识，采用大写，并且没有尾部空格。

groupnamelen

输入。**groupname** 参数值的长度（以字节计）。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secDoesGroupExist API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secFreeErrormsg API - 释放错误消息内存

释放用来存放上次调用 API 所产生的错误消息的内存。这是唯一不会返回错误消息的 API。如果此 API 返回了错误，那么 DB2 将记录此错误并继续运行。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secFreeErrormsg)
                      ( char *errormsg );
```

db2secFreeErrormsg API parameters

errormsg

输入。指向上次调用 API 时分配的错误消息的指针。

db2secFreeGroupListMemory API - 释放组列表内存

释放用来存放上次调用 db2secGetGroupsForUser API 时获得的组列表的内存。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secFreeGroupListMemory)
                      ( void *ptr,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secFreeGroupListMemory API 参数

ptr 输入。指向要释放的内存的指针。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secFreeGroupListMemory API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secGetGroupsForUser API - 获取用户的组列表

返回用户所属的组的列表。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secGetGroupsForUser)
                      ( const char *authid,
                        db2int32 authidlen,
                        const char *userid,
                        db2int32 useridlen,
                        const char *usernamespace,
                        db2int32 usernamespacelen,
                        db2int32 usernamespacetype,
                        const char *dbname,
```

```
db2int32 dbnamelen,
void *token,
db2int32 tokentype,
db2int32 location,
const char *authpluginname,
db2int32 authpluginnamelen,
void      **grouplist,
db2int32 *numgroups,
char      **errormsg,
db2int32 *errormsglen );
```

db2secGetGroupsForUser API 参数

authid 输入。此参数值是一个 SQL 授权标识，这意味着 DB2 for Linux, UNIX, and Windows 会将它转换为不带尾部空格的大写字符串。DB2 for Linux, UNIX, and Windows 始终都将为 **authid** 参数提供非空值。此 API 必须能够返回 authid 所属的组的列表而不依赖于其他输入参数。如果不能确定此列表，那么允许返回一个缩短的列表或空列表。

如果用户不存在，那么该 API 返回的返回码必须是 DB2SEC_PLUGIN_BADUSER。DB2 for Linux, UNIX, and Windows 并不认为“用户不存在”这种情况是一个错误，因为它允许 authid 没有任何相关联的组。例如，db2secGetAuthids API 就可以返回操作系统上不存在的 authid。此 authid 与任何组都不相关联，但是，仍然可以直接为它指定特权。

如果此 API 通过仅使用 authid 不能返回组的完整列表，那么对于与组支持相关的某些 SQL 函数将有一些限制。有关可能会发生的各种问题的列表，请参阅本主题中的“使用说明”部分。

authidlen
输入。**authid** 参数值的长度（以字节计）。DB2 数据库管理器始终为 **authidlen** 参数提供非零值。

userid 输入。这是与 **authid** 相对应的用户标识。当在服务器上在非连接方案中调用此 API 时，DB2 for Linux, UNIX, and Windows 将不会填充此参数。

useridlen
输入。**userid** 参数值的长度（以字节计）。

usernamespace
输入。从其中获得用户标识的名称空间。当未提供用户标识时，DB2 数据库管理器将不会填充此参数。

usernamespacelen
输入。**usernamespace** 参数值的长度（以字节计）。

usernamespacetype
输入。名称空间的类型。**usernamespacetype** 参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_NAMESPACE_SAM_COMPATIBLE 对应于一个样式类似于 domain\myname 的用户名
- DB2SEC_NAMESPACE_USER_PRINCIPAL 对应于一个样式类似于 myname@domain.ibm.com 的用户名

目前，DB2 数据库系统仅支持 DB2SEC_NAMESPACE_SAM_COMPATIBLE 值。当未提供用户标识时，**usernamespacetype** 参数设置为值 DB2SEC_USER_NAMESPACE_UNDEFINED（此值是在 db2secPlugin.h 中定义的）。

dbname

输入。要连接至的数据库的名称。在非连接方案中，此参数可以为 NULL。

dbnamelen

输入。**dbname** 参数值的长度（以字节计）。如果 **dbname** 参数在非连接方案中为 NULL，那么此参数设置为 0。

token 输入。指向由认证插件提供的数据的指针。DB2 for Linux, UNIX, and Windows 不使用此参数。它使插件编写者能够调整用户和组信息。可能不是在所有情况下都会提供此参数（例如，在非连接方案中就不会提供），在此情况下此参数将为 NULL。如果使用的认证插件基于 GSS-API，那么令牌将设置为 GSS-API 上下文句柄 (gss_ctx_id_t)。

tokentype

输入。指示由认证插件提供的数据的类型。如果使用的认证插件基于 GSS-API，那么令牌将设置为 GSS-API 上下文句柄 (gss_ctx_id_t)。如果使用的认证插件是基于“用户标识/密码”的插件，那么它将是通用类型。**tokentype** 参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_GENERIC: 指示令牌来自于基于用户标识/密码的插件。
- DB2SEC_GSSAPI_CTX_HANDLE: 指示令牌来自于基于 GSS-API（包括 Kerberos）的插件。

location

输入。指示 DB2 for Linux, UNIX, and Windows 是在客户端还是服务器端调用此 API。location 参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_SERVER_SIDE: 表示要在数据库服务器上调用此 API。
- DB2SEC_CLIENT_SIDE: 表示要在客户机上调用此 API。

authpluginname

输入。提供了令牌中的数据的认证插件的名称。db2secGetGroupsForUser API 可能会使用此信息来确定正确的组成员资格。如果未认证 **authid**（例如，在 **authid** 与当前连接的用户不匹配的情况下），那么 DB2 for Linux, UNIX, and Windows 就不会填充此参数。

authpluginnamelen

输入。**authpluginname** 参数值的长度（以字节计）。

grouplist

输出。用户所属的组的列表。组的列表必须作为一个指针返回，该指针指向由包含多个并置 varchar 的插件分配的内存片段（varchar 是一个字符数组，其中的第一个字节指示它后面具有的字节数）。长度是一个无符号数（1 个字节），并将 groupname 的最大长度限制为 255 个字符。例如，“\006GROUP1\007MYGROUP\008MYGROUP3”。每个组名都应该是一个有效的 DB2 授权标识。必须由插件为此数组分配内存。因此，插件必须提供一个 API（例如，db2secFreeGroupListMemory API）供 DB2 for Linux, UNIX, and Windows 调用以释放内存。

numgroups

输出。**grouplist** 参数中包含的组的数目。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secGetGroupsForUser API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

使用说明

以下列表说明当此 API 将不完整的组列表返回给 DB2 for Linux, UNIX, and Windows 时就会发生问题的各种情况:

- 在 CREATE SCHEMA 语句中提供了备用权限。如果 CREATE SCHEMA 语句中嵌套了 CREATE 语句，那么将对 AUTHORIZATION NAME 参数执行组查找。
- 在 MPP 环境中处理 JAR 文件。在 MPP 环境中，JAR 处理请求是通过会话授权标识从协调程序节点中发送的。目录节点接收到请求，并根据会话授权标识（执行 JAR 处理请求的用户）的特权来处理 JAR 文件。
  - 安装 JAR 文件。会话授权标识需要具有下列权限之一: DBADM 或 CREATEIN（对于 JAR 模式的隐式或显式权限）。如果上述权限已授予包含会话授权标识的组，但是未显式授予会话授权标识，那么该操作将失败。
  - 除去 JAR 文件。会话授权标识需要具有下列权限之一: DBADM 或 DROPIN（对于 JAR 模式的隐式或显式权限），或者是 JAR 文件的定义者。如果上述权限已授予包含会话授权标识的组，但是未显式授予会话授权标识，并且会话授权标识不是 JAR 文件的定义者，那么该操作将失败。
  - 替换 JAR 文件。这与除去该 JAR 文件，然后再重新安装该 JAR 文件的效果相同。上述两种情况都适用。
- 发出 SET SESSION_USER 语句时。后续的 DB2 操作将在由此语句指定的授权标识的上下文中运行。如果未将 SESSION_USER 的其中一个组所拥有的必需特权显式授予 SESSION_USER 授权标识，那么这些操作将失败。

db2secGroupPluginInit API - 初始化组插件

组检索插件的初始化 API，DB2 数据库管理器在装入该插件之后就立即调用此 API。

API 和数据结构语法

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit
                            ( db2int32 version,
                              void *group_fns,
                              db2secLogMessage *logMessage_fn,
                              char     **errormsg,
                              db2int32 *errormsglen );
```

db2secGroupPluginInit API 参数

version

输入。装入该插件的实例所支持的 API 的最高版本。db2secPlugin.h 中的 DB2SEC_API_VERSION 值包含 DB2 数据库管理器当前支持的 API 的最新版本号。

group_fns

输出。指向 db2secGroupFunctions_<version_number>（也称为 group_functions_<version_number>）结构的指针。

db2secGroupFunctions_<version_number> 结构包含指向为组检索插件实现的 API 的指针。将来可能有不同版本的 API（例如，db2secGroupFunctions_<version_number>），因此，**group_fns** 参数被强制类型转换为指向与插件已实现的版本相对应的 db2secGroupFunctions_<version_number> 结构的指针。group_functions_<version_number> 结构的第一个参数将对 DB2 for Linux, UNIX, and Windows 告知插件已实现的 API 的版本。注意：仅当 DB2 版本高于或等于插件已实现的 API 版本时才会进行强制类型转换。版本号表示由插件实现的 API 的版本，并且 **pluginType** 应设置为 DB2SEC_PLUGIN_TYPE_GROUP。

logMessage_fn

输入。指向由 DB2 数据库系统实现的 db2secLogMessage API 的指针。db2secGroupPluginInit API 可以调用 db2secLogMessage API 以将消息记录到 **db2diag** 日志文件中，以便进行调试或者供您参考。db2secLogMessage API 的第一个参数 (**level**) 指定将记录在 **db2diag** 日志文件中的诊断错误的类型，最后两个参数分别表示消息字符串及其长度。db2secLogMessage API 的第一个参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_LOG_NONE: (0) 不记录
- DB2SEC_LOG_CRITICAL: (1) 服务器发生错误
- DB2SEC_LOG_ERROR: (2) 发生错误
- DB2SEC_LOG_WARNING: (3) 警告
- DB2SEC_LOG_INFO: (4) 参考

仅当 db2secLogMessage API 的 **level** 参数的值小于或等于 **diaglevel** 数据库管理器配置参数的值时，消息文本才会出现在 db2diag 日志文件中。例如，如果使用 DB2SEC_LOG_INFO 值，那么仅当数据库管理器配置参数 **diaglevel** 设置为 4 时，**db2diag** 日志文件中才会出现消息文本。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secGroupPluginInit API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secPluginTerm - 清除组插件资源

释放组检索插件所使用的资源。

DB2 数据库管理器将在卸载组检索插件之前调用此 API。应该以这样一种方式来实现它：它将正确清除插件库拥有的任何资源，例如，释放由插件分配的任何内存，关闭仍处于打开状态的文件并关闭网络连接。插件负责跟踪这些资源以便释放这些资源。在任何 Windows 操作系统上都不会调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secPluginTerm)
                      ( char      **errormsg,
                        db2int32 *errormsglen );
```

db2secPluginTerm API 参数

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secPluginTerm API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

用户标识/密码认证插件的 API

对于用户标识/密码插件模块，需要实现下列客户端 API:

- db2secClientAuthPluginInit

  注: db2secClientAuthPluginInit API 通过以下原型将 API 的指针 *logMessage_fn 作为输入:

  ```
  SQL_API_RC (SQL_API_FN db2secLogMessage)
  (
  db2int32 level,
  void    *data,
  db2int32 length
  );
  ```

  db2secLogMessage API 允许插件将消息记录到 **db2diag** 日志文件中，以便进行调试或者供您参考。此 API 由 DB2 数据库系统提供，因此您不需要实现此 API。
- db2secClientAuthPluginTerm
- db2secGenerateInitialCred（仅用于 gssapi）
- db2secRemapUserid（可选）
- db2secGetDefaultLoginContext
- db2secValidatePassword
- db2secProcessServerPrincipalName（这仅适用于 GSS-API）
- db2secFreeToken（用于释放 DLL 占用的内存的函数）
- db2secFreeErrormsg
- db2secFreeInitInfo
- 唯一需要在外部可解析的 API 是 db2secClientAuthPluginInit。此 API 将采用 void * 参数，应将它强制类型转换为下列其中一项:

  ```
  typedef struct db2secUseridPasswordClientAuthFunctions_1
  {
  db2int32 version;
  db2int32 plugintype;

  SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
  (
  char        authid[DB2SEC_MAX_AUTHID_LENGTH],
  db2int32   *authidlen,
  char        userid[DB2SEC_MAX_USERID_LENGTH],
  db2int32   *useridlen,
  db2int32    useridtype,
  char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
  db2int32   *usernamespacelen,
  db2int32   *usernamespacetype,
  const char *dbname,
  ```

```
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32      *errormsglen
);
/* Optional */
SQL_API_RC (SQL_API_FN * db2secRemapUserid)
(
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
char        password[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *passwordlen,
char        newpassword[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
char      **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secValidatePassword)
(
const char *userid,
db2int32    useridlen,
const char *usernamespace,
db2int32    usernamespacelen,
db2int32    usernamespacetype,
const char *password,
db2int32       passwordlen,
const char *newpassword,
db2int32       newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
db2Uint32   connection_details,
void      **token,
char      **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
(
void     **token,
char     **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
char     **errormsg,
db2int32      *errormsglen
);
}
```

或者

```
typedef struct db2secGssapiClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;
```

```
SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
(
char        authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32   *authidlen,
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
db2int32    useridtype,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secProcessServerPrincipalName)
(
const void *data,
gss_name_t *gssName,
char      **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secGenerateInitialCred)
(
const char    *userid,
db2int32       useridlen,
const char    *usernamespace,
db2int32       usernamespacelen,
db2int32       usernamespacetype,
const char    *password,
db2int32       passwordlen,
const char    *newpassword,
db2int32       newpasswordlen,
const char    *dbname,
db2int32       dbnamelen,
gss_cred_id_t *pGSSCredHandle,
void         **initInfo,
char     **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
(
void      *token,
char     **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);


SQL_API_RC (SQL_API_FN * db2secFreeInitInfo)
(
void      *initInfo,
char     **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
```

```
char     **errormsg,
db2int32      *errormsglen
);

/* GSS-API specific functions -- refer to db2secPlugin.h
   for parameter list*/

  OM_uint32 (SQL_API_FN * gss_init_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);
  }
```

如果您正在编写用户标识/密码插件，那么应使用 db2secUseridPasswordClientAuthFunctions_1 结构。如果您正在编写 GSS-API（包括 Kerberos）插件，那么应使用 db2secGssapiClientAuthFunctions_1 结构。

对于用户标识/密码插件库，将需要实现下列服务器端 API:

- db2secServerAuthPluginInit

  db2secServerAuthPluginInit API 通过下列原型将 db2secLogMessage API 的指针 *logMessage_fn 以及 db2secGetConDetails API 的指针 *getConDetails_fn 作为输入:

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
);

SQL_API_RC (SQL_API_FN db2secGetConDetails)
(
db2int32    conDetailsVersion,
const void *pConDetails
);
```

  db2secLogMessage API 允许插件将消息记录到 **db2diag** 日志文件中，以便进行调试或者供您参考。db2secGetConDetails API 允许插件获取有关将尝试建立数据库连接的客户机的详细信息。db2secLogMessage API 和 db2secGetConDetails API 都是由 DB2 数据库系统提供的，因此您不需要实现这两个 API。而 db2secGetConDetails API 将它的第二个参数 **pConDetails** 作为指向下列其中一种结构的指针:

  db2sec_con_details_1:

```
typedef struct db2sec_con_details_1
{
  db2int32  clientProtocol;
  db2Uint32 clientIPAddress;
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char      dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
} db2sec_con_details_1;
```

  db2sec_con_details_2:

```
typedef struct db2sec_con_details_2
{
  db2int32  clientProtocol;     /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress;    /* Set if protocol is TCPIP4     */
```

```
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6     */
} db2sec_con_details_2;
```

db2sec_con_details_3:

```
typedef struct db2sec_con_details_3
{
  db2int32 clientProtocol; /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress; /* Set if protocol is TCPIP4 */
  db2Uint32 connect_info_bitmap;
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6 */
  db2Uint32 clientPlatform; /* SQLM_PLATFORM_* from sqlmon.h */
  db2Uint32 _reserved[16];
} db2sec_con_details_3;
```

conDetailsVersion 的值可以是 DB2SEC_CON_DETAILS_VERSION_1、DB2SEC_CON_DETAILS_VERSION_2 和 DB2SEC_CON_DETAILS_VERSION_3，它们表示 API 的版本。

注：使用 db2sec_con_details_1、db2sec_con_details_2 或 db2sec_con_details_3 时，应考虑下列事项：

- 当调用 db2GetConDetails API 时，使用 db2sec_con_details_1 结构和 DB2SEC_CON_DETAILS_VERSION_1 值的现有插件将继续像在 V8.2 中一样工作。如果在 IPv4 平台上调用了此 API，那么将在该结构的 **clientIPAddress** 字段中返回客户机 IP 地址。如果在 IPv6 平台上调用了此 API，那么在 **clientIPAddress** 字段中将返回值 0。要在 IPv6 平台上检索客户机 IP 地址，应将安全插件代码更改为使用 db2sec_con_details_2 结构和 DB2SEC_CON_DETAILS_VERSION_2 值，或者更改为使用 db2sec_con_details_3 结构和 DB2SEC_CON_DETAILS_VERSION_3 值。
- 新插件应使用 db2sec_con_details_3 结构和 DB2SEC_CON_DETAILS_VERSION_3 值。如果在 IPv4 平台上调用了 db2secGetConDetails API，那么将在 db2sec_con_details_3 结构的 **clientIPAddress** 字段中返回客户机 IP 地址；如果在 IPv6 平台上调用了此 API，那么将在 db2sec_con_details_3 结构的 **clientIP6Address** 字段中返回客户机 IP 地址。连接详细信息结构的 **clientProtocol** 字段将设置为 SQL_PROTOCOL_TCPIP（IPv4，具有 V1 的结构）、SQL_PROTOCOL_TCPIP4（IPv4，具有 V2 的结构）或 SQL_PROTOCOL_TCPIP6（IPv6，具有 V2 或 V3 的结构）的其中一项。
- 除了下面这一点不同之外，db2sec_con_details_3 结构与 db2sec_con_details_2 结构完全相同：db2sec_con_details_3 结构包含一个额外的字段 (**clientPlatform**)，此字段使用在 sqlmon.h 中定义的平台类型约束（例如，SQLM_PLATFORM_AIX）来标识客户机平台类型（与通信层报告的一样）。

- db2secServerAuthPluginTerm
- db2secValidatePassword
- db2secGetAuthIDs
- db2secDoesAuthIDExist
- db2secFreeToken
- db2secFreeErrormsg

- 唯一需要在外部可解析的 API 是 db2secServerAuthPluginInit。此 API 将采用 void * 参数，应将它强制类型转换为下列其中一项:

```
typedef struct db2secUseridPasswordServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

  /* parameter lists left blank for readability
    see above for parameters */
SQL_API_RC (SQL_API_FN * db2secValidatePassword)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeToken)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();
} userid_password_server_auth_functions;
```

或者

```
typedef struct db2secGssapiServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;
gss_buffer_desc serverPrincipalName;
gss_cred_id_t ServerCredHandle;
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();

/* GSS-API specific functions
refer to db2secPlugin.h for parameter list*/
OM_uint32 (SQL_API_FN * gss_accept_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_name )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);

} gssapi_server_auth_functions;
```

如果您正在编写用户标识/密码插件，那么应使用 db2secUseridPasswordServerAuthFunctions_1 结构。如果您正在编写 GSS-API（包括 Kerberos）插件，那么应使用 db2secGssapiServerAuthFunctions_1 结构。

db2secClientAuthPluginInit API - 初始化客户机认证插件

客户机认证插件的初始化 API，DB2 数据库管理器在装入该插件之后就立即调用此 API。

API 和数据结构语法

```
SQL_API_RC SQL_API_FN db2secClientAuthPluginInit
                            ( db2int32 version,
                              void *client_fns,
                              db2secLogMessage *logMessage_fn,
                              char    **errormsg,
                              db2int32 *errormsglen );
```

db2secClientAuthPluginInit API 参数

版本

输入。DB2 数据库管理器当前支持的 API 的最高版本号。db2secPlugin.h 中的 DB2SEC_API_VERSION 值包含 DB2 for Linux, UNIX, and Windows 当前支持的 API 的最新版本号。

client_fns

输出。如果使用了 GSS-API 认证，那么此参数表示指向 DB2 数据库管理器为 db2secGssapiClientAuthFunctions_<version_number> 结构（也称为 gssapi_client_auth_functions_<version_number>）提供的内存的指针；如果使用了“用户标识/密码”认证，那么此参数表示指向 DB2 数据库管理器为 db2secUseridPasswordClientAuthFunctions_<version_number> 结构（也称为 userid_password_client_auth_functions_<version_number>）提供的内存的指针。db2secGssapiClientAuthFunctions_<version_number> 结构和 db2secUseridPasswordClientAuthFunctions_<version_number> 结构分别包含指向为 GSS-API 认证插件和“用户标识/密码”认证插件实现的 API 的指针。在将来的 DB2 for Linux, UNIX, and Windows 版本中，可能会有不同版本的 API，因此 **client_fns** 参数被强制类型转换为指向与插件已实现的版本相对应的 gssapi_client_auth_functions_<version_number> 结构的指针。

gssapi_client_auth_functions_<version_number> 结构或 userid_password_client_auth_functions_<version_number> 结构的第一个参数将把插件已实现的 API 的版本告知给 DB2 数据库管理器。

注：仅当 DB2 版本高于或等于插件已实现的 API 版本时才会进行强制类型转换。

在 gssapi_server_auth_functions_<version_number> 或 userid_password_server_auth_functions_<version_number> 结构中，应将 **plugintype** 参数设置为 DB2SEC_PLUGIN_TYPE_USERID_PASSWORD、DB2SEC_PLUGIN_TYPE_GSSAPI 或 **DB2SEC_PLUGIN_TYPE_KERBEROS**。可以在将来的 API 版本中定义其他值。

logMessage_fn

输入。指向由 DB2 数据库管理器实现的 db2secLogMessage API 的指针。db2secClientAuthPluginInit API 可以调用 db2secLogMessage API 以将消息记录到 **db2diag** 日志文件中，以便进行调试或者供您参考。db2secLogMessage API 的第一个参数 (**level**) 指定将记录在 **db2diag** 日志文件中的诊断错误的类型，最后两个参数分别表示消息字符串及其长度。db2secLogMessage API 的第一个参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_LOG_NONE (0) 不记录
- DB2SEC_LOG_CRITICAL (1) 发生严重错误
- DB2SEC_LOG_ERROR (2) 发生错误
- DB2SEC_LOG_WARNING (3) 警告
- DB2SEC_LOG_INFO (4) 参考

仅当 db2secLogMessage API 的“level”参数的值小于或等于 **diaglevel** 数据库管理器配置参数的值时，消息文本才会出现在 **db2diag** 日志文件中。例如，如果使用 DB2SEC_LOG_INFO 值，那么仅当数据库管理器配置参数 **diaglevel** 设置为 4 时，**db2diag** 日志文件中才会出现消息文本。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secClientAuthPluginInit API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secClientAuthPluginTerm API - 清除客户机认证插件资源

释放客户机认证插件所使用的资源。

DB2 数据库管理器将在卸载客户机认证插件之前调用此 API。应该以这样一种方式来实现它：它将正确清除插件库拥有的任何资源，例如，释放由插件分配的任何内存，关闭仍处于打开状态的文件并关闭网络连接。插件负责跟踪这些资源以便释放这些资源。在任何 Windows 操作系统上都不会调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
                      ( char     **errormsg,
                        db2int32 *errormsglen);
```

db2secClientAuthPluginTerm API 参数

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secClientAuthPluginTerm API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secDoesAuthIDExist - 检查认证标识是否存在

确定 **authid** 是否表示单个用户（例如，API 可以将 **authid** 映射至外部用户标识）。

如果 **authid** 有效，那么此 API 应返回值 DB2SEC_PLUGIN_OK；如果 **authid** 无效，那么此 API 应返回 DB2SEC_PLUGIN_INVALID_USERORGROUP；如果不能确定 **authid** 是否存在，那么此 API 应返回 DB2SEC_PLUGIN_USERSTATUSNOTKNOWN。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secDoesAuthIDExist)
                      ( const char *authid,
                        db2int32 authidlen,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secDoesAuthIDExist API 参数

authid
输入。要验证的授权标识。此标识采用大写，并且没有尾部空格。

authidlen
输入。**authid** 参数值的长度（以字节计）。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secDoesAuthIDExist API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度的整数的指针。

db2secFreeInitInfo API - 清除由 db2secGenerateInitialCred 占用的资源

释放由 db2secGenerateInitialCred API 分配的任何资源。这些资源可以包括底层机制上下文的句柄或者为 GSS-API 凭证高速缓存创建的凭证高速缓存。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secFreeInitInfo)
                      ( void *initinfo,
                        char     **errormsg,
                        db2int32 *errormsglen);
```

db2secFreeInitInfo API 参数

initinfo
输入。指向 DB2 数据库管理器不知道的数据的指针。插件可以使用此内存来维护在生成凭证句柄过程中所分配的资源的列表。通过调用此 API 来释放这些资源。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secFreeInitInfo API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secFreeToken API - 释放由标记占用的内存

释放由标记占用的内存。当 DB2 数据库管理器不再需要 token 参数所占用的内存时就会调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secFreeToken)
                      ( void *token,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secFreeToken API 参数

token
输入。指向要释放的内存的指针。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secFreeToken API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secGenerateInitialCred API - 生成初始凭证

db2secGenerateInitialCred API 根据已传入的用户标识和密码来获取初始 GSS-API 凭证。

对于 Kerberos，这是授予凭单的凭单（TGT）。**pGSSCredHandle** 参数中返回的凭证句柄是与 gss_init_sec_context API 配合使用的句柄，并且必须是 INITIATE 或 BOTH 凭证。仅当提供了用户标识（还可能提供了密码）时才会调用 db2secGenerateInitialCred API。否则，调用 gss_init_sec_context API 时，DB2 数据库管理器将指定值 GSS_C_NO_CREDENTIAL，以表示将使用从当前登录上下文中获取的缺省凭证。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secGenerateInitialCred)
                      ( const char *userid,
                        db2int32 useridlen,
                        const char *usernamespace,
                        db2int32 usernamespacelen,
                        db2int32 usernamespacetype,
                        const char *password,
                        db2int32 passwordlen,
                        const char *newpassword,
                        db2int32 newpasswordlen,
                        const char *dbname,
                        db2int32 dbnamelen,
                        gss_cred_id_t *pGSSCredHandle,
                        void          **InitInfo,
                        char          **errormsg,
                        db2int32 *errormsglen );
```

db2secGenerateInitialCred API 参数

userid
输入。要在数据库服务器上验证其密码的用户标识。

useridlen
输入。**userid** 参数值的长度（以字节计）。

usernamespace
输入。从其中获得用户标识的名称空间。

usernamespacelen
输入。**usernamespace** 参数值的长度（以字节计）。

usernamespacetype
输入。名称空间的类型。

password
输入。要验证的密码。

passwordlen
输入。**password** 参数值的长度（以字节计）。

newpassword
输入。一个新密码（如果要更改密码）。如果未请求更改密码，那么 **newpassword** 参数将设置为 NULL。如果它不为 NULL，那么此 API 在将旧密码设置为新密码之前应验证旧密码。此 API 并非必须完成更改密码的请求，但是如果未完成此请求，那么应立即返回值 DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED 而不验证旧密码。

newpasswordlen

输入。**newpassword** 参数值的长度（以字节计）。

dbname

输入。要连接至的数据库的名称。此 API 可以忽略此参数，如果此 API 具有限制用户访问某些数据库的策略，那么它可以返回值 DB2SEC_PLUGIN_CONNECTION_DISALLOWED；否则这些用户将具有有效密码。

dbnamelen

输入。**dbname** 参数值的长度（以字节计）。

pGSSCredHandle

输出。指向 GSS-API 凭证句柄的指针。

InitInfo

输出。指向 DB2 for Linux, UNIX, and Windows 不知道的数据的指针。插件可以使用此内存来维护在生成凭证句柄过程中所分配的资源的列表。认证过程结束时，DB2 数据库管理器将调用 db2secFreeInitInfo API，此时将释放这些资源。如果 db2secGenerateInitialCred API 不需要维护这样一个列表，那么它应该返回 NULL。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secGenerateInitialCred API 不成功，就会返回此错误消息。

注：对于此 API，如果返回值指示错误的用户标识或密码，那么不应创建错误消息。仅当此 API 中发生了一个内部错误从而阻止它正确完成时，才应返回一条错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secGetAuthIDs API - 获取认证标识

返回已认证的用户的 SQL 授权标识。对于“用户标识/密码”和 GSS-API 认证方法，在建立数据库连接期间都将调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secGetAuthIDs)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *dbname,
                         db2int32 dbnamelen,
                         void      **token,
                         char SystemAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                         db2int32 *SystemAuthIDlen,
                         char InitialSessionAuthID[DB2SEC_MAX_AUTHID_LENGTH],
                         db2int32 *InitialSessionAuthIDlen,
                         char username[DB2SEC_MAX_USERID_LENGTH],
                         db2int32 *usernamelen,
                         db2int32 *initsessionidtype,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secGetAuthIDs API 参数

userid
输入。已认证的用户。除非定义了可信上下文以允许执行切换用户操作而无需进行认证，否则，通常不会将此参数用于 GSS-API 认证。在此情况下，会将为切换用户请求提供的用户名传递到此参数中。

useridlen
输入。**userid** 参数值的长度（以字节计）。

usernamespace
输入。从其中获得用户标识的名称空间。

usernamespacelen
输入。**usernamespace** 参数值的长度（以字节计）。

usernamespacetype
输入。名称空间类型值。目前，唯一受支持的名称空间类型值是 DB2SEC_NAMESPACE_SAM_COMPATIBLE（对应类似于 domain\myname 的用户名样式）。

dbname
输入。要连接至的数据库的名称。此 API 可以忽略此参数；当同一用户连接至不同数据库时，此 API 可以返回不同的授权标识。此参数可以为 NULL。

dbnamelen
输入。**dbname** 参数值的长度（以字节计）。如果 **dbname** 参数为 NULL，那么此参数设置为 0。

token
输入或输出。插件可以向 db2secGetGroupsForUser API 传递的数据。对于 GSS-API，这是上下文句柄 (gss_ctx_id_t)。通常，token 是一个“仅输入”参数，它的值是从 db2secValidatePassword API 中获取的。当在客户机上进行认证并因此而未调用 db2secValidatePassword API 时，此参数也可以是输出参数。在定义了可信上下文的环境中，如果该上下文允许执行切换用户操作而无需认证，那么 db2secGetAuthIDs API 必须能够接受此 token 参数的值为 NULL，并且能够根据先前提到的 **userid** 和 **useridlen** 输入参数来派生系统授权标识。

SystemAuthID
输出。与已认证的用户的标识相对应的系统授权标识。其大小为 255 个字节，但是 DB2 数据库管理器当前仅使用最长为 30 个字节的系统授权标识。

SystemAuthIDlen
输出。**SystemAuthID** 参数值的长度（以字节计）。

InitialSessionAuthID
输出。用于此连接会话的授权标识。它通常与 **SystemAuthID** 参数值相同，但是在某些情况下可以不相同，例如，发出 SET SESSION AUTHORIZATION 语句时就可以不相同。其大小为 255 个字节，但是 DB2 数据库管理器当前仅使用最长为 30 个字节的系统授权标识。

InitialSessionAuthIDlen
输出。**InitialSessionAuthID** 参数值的长度（以字节计）。

username
输出。与已认证的用户和授权标识相对应的用户名。这将仅用于审计，并且将记录

在 CONNECT 语句的审计记录中的“用户标识”字段中。如果此 API 指定 username 参数，那么 DB2 数据库管理器将从 userid 参数中复制此参数值。

usernamelen

输出。**username** 参数值的长度（以字节计）。

initsessionidtype

输出。会话授权标识类型指示 **InitialSessionAuthid** 参数是一个角色还是授权标识。此 API 应返回下列其中一个值（这些值是在 db2secPlugin.h 中定义的）：

- DB2SEC_ID_TYPE_AUTHID (0)
- DB2SEC_ID_TYPE_ROLE (1)

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secGetAuthIDs API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secGetDefaultLoginContext API - 获取缺省登录上下文

确定与缺省登录上下文相关联的用户。即，确定在没有显式指定用户标识（向数据库进行隐式认证，或者进行本地授权）的情况下调用 DB2 命令的用户的 DB2 授权标识。此 API 必须同时返回授权标识和用户标识。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
                       ( char authid[DB2SEC_MAX_AUTHID_LENGTH],
                         db2int32 *authidlen,
                         char userid[DB2SEC_MAX_USERID_LENGTH],
                         db2int32 *useridlen,
                         db2int32 useridtype,
                         char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                         db2int32 *usernamespacelen,
                         db2int32 *usernamespacetype,
                         const char *dbname,
                         db2int32 dbnamelen,
                         void      **token,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secGetDefaultLoginContext API 参数

authid

输出。应返回授权标识的参数。返回的值必须符合 DB2 授权标识的命名规则，否则将不会授权用户执行所请求的操作。

authidlen

输出。**authid** 参数值的长度（以字节计）。

userid

输出。应返回与缺省缺省登录上下文相关联的用户标识的参数。

useridlen

输出。**userid** 参数值的长度（以字节计）。

useridtype
输入。标识指定的是进程的实用户标识还是有效用户标识。在 Windows 上，只存在实用户标识。在 UNIX 和 Linux 上，如果应用程序的 uid 用户标识与执行该进程的用户的标识不同，那么实用户标识与有效用户标识可以不同。**userid** 参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

DB2SEC_PLUGIN_REAL_USER_NAME
指示指定的是实用户标识。

DB2SEC_PLUGIN_EFFECTIVE_USER_NAME
指示指定的是有效用户标识。

注: 某些插件实现可能并不区分实用户标识和有效用户标识。特别是，不使用用户的 UNIX 或 Linux 身份来建立 DB2 授权标识的插件可以安全地忽略此差别。

usernamespace
输出。用户标识的名称空间。

usernamespacelen
输出。**usernamespace** 参数值的长度（以字节计）。由于存在 **usernamespacetype** 参数必须设置为值 DB2SEC_NAMESPACE_SAM_COMPATIBLE（此值是在 db2secPlugin.h 中定义的）这一局限性，因此当前支持的最大长度为 15 字节。

usernamespacetype
输出。名称空间类型值。目前，唯一受支持的名称空间类型是 DB2SEC_NAMESPACE_SAM_COMPATIBLE（对应类似于 domain\myname 的用户名样式）。

dbname
输入。包含要连接至的数据库的名称（如果在数据库连接上下文中使用此调用）。对于本地授权操作或实例连接，此参数设置为 NULL。

dbnamelen
输入。**dbname** 参数值的长度（以字节计）。

token
输出。这是一个指向由插件分配的数据的指针，而此数据可能会被传递给该插件中的后续认证调用，也有可能被传递给组检索插件。此数据的结构由插件编写者确定。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secGetDefaultLoginContext API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secProcessServerPrincipalName API - 处理从服务器返回的服务主体名称

db2secProcessServerPrincipalName API 处理从服务器返回的服务主体名称，并按 gss_name_t 内部格式返回要对 gss_init_sec_context API 使用的主体名称。

使用 Kerberos 认证时，db2secProcessServerPrincipalName API 还会处理使用数据库目录编目的服务主体名称。通常，此转换使用 gss_import_name API。建立上下文之后，将通过调用 gss_release_name API 来释放 gss_name_t 对象。如果 **gssName** 参数指向有效的 GSS 名称，那么 db2secProcessServerPrincipalName API 将返回值 **DB2SEC_PLUGIN_OK**；如果主体名称无效，那么将返回 DB2SEC_PLUGIN_BAD_PRINCIPAL_NAME 错误代码。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secProcessServerPrincipalName)
                      ( const char *name,
                        db2int32 namelen,
                        gss_name_t *gssName,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secProcessServerPrincipalName API 参数

name
输入。采用 GSS_C_NT_USER_NAME 格式的服务主体的文本名称；例如，service/host@REALM。

namelen
输入。**name** 参数值的长度（以字节计）。

gssName
输出。指向采用 GSS-API 内部格式的输出服务主体名称的指针。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secProcessServerPrincipalName API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secRemapUserid API - 重新映射用户标识和密码

客户端的 DB2 数据库管理器调用此 API 将给定的用户标识和密码（可能是新密码和用户名称空间）重新映射至与连接时给定的那些值不相同的值。

仅当在连接时提供了用户标识和密码时，DB2 数据库管理器才会在客户端调用此 API。这将防止插件将用户标识本身重新映射至用户标识/密码对。此 API 是可选的。如果它不是由安全插件提供或实现的，那么就不会调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secRemapUserid)
                      ( char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32 *useridlen,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32 *usernamespacelen,
                        db2int32 *usernamespacetype,
                        char password[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32 *passwordlen,
                        char newpasswd[DB2SEC_MAX_PASSWORD_LENGTH],
                        db2int32 *newpasswdlen,
                        const char *dbname,
```

```
                                db2int32 dbnamelen,
                                char      **errormsg,
                                db2int32 *errormsglen);
```

db2secRemapUserid API 参数

userid

输入或输出。要重新映射的用户标识。如果有输入用户标识值，那么此 API 必须提供一个输出用户标识值，此输出用户标识值可与该输入用户标识值相同或不同。如果没有输入用户标识值，那么此 API 不应返回输出用户标识值。

useridlen

输入或输出。**userid** 参数值的长度（以字节计）。

usernamespace

输入或输出。用户标识的名称空间。（可选）可以重新映射此值。如果没有指定输入参数值，但是返回了输出值，那么 DB2 数据库管理器将只把 **usernamespace** 用于 CLIENT 类型认证，对于其他认证类型都会忽略 usernamespace 参数。

usernamespacelen

输入或输出。**usernamespace** 参数值的长度（以字节计）。由于存在 **usernamespacetype** 参数必须设置为值 DB2SEC_NAMESPACE_SAM_COMPATIBLE（此值是在 db2secPlugin.h 中定义的）这一局限性，因此当前支持的最大长度为 15 字节。

usernamespacetype

输入或输出。旧的和新的名称空间类型值。目前，唯一受支持的名称空间类型值是 DB2SEC_NAMESPACE_SAM_COMPATIBLE（对应类似于 domain\myname 的用户名样式）。

password

输入或输出。如果作为输入，那么它是要重新映射的密码。如果作为输出，那么它是已重新映射的密码。如果为此参数指定了输入值，那么此 API 必须能够返回与该输入值不同的输出值。如果没有指定输入值，那么此 API 一定不能返回输出密码值。

passwordlen

输入或输出。**password** 参数值的长度（以字节计）。

newpasswd

输入或输出。如果作为输入，那么它是要设置的新密码。如果作为输出，那么它是经过确认的新密码。

注: 这是 DB2 数据库管理器传递到客户机或服务器（取决于数据库管理器配置参数 **authentication** 的值）上的 db2secValidatePassword API 的 **newpassword** 参数的新密码。如果新密码是作为输入来传递的，那么此 API 必须能够返回输出值，并且输出值可以是不同的新密码。如果没有新密码是作为输入传递的，那么此 API 不应返回输出新密码。

newpasswdlen

输入或输出。**newpasswd** 参数值的长度（以字节计）。

dbname

输入。客户机要连接至的数据库的名称。

dbnamelen

输入。**dbname** 参数值的长度（以字节计）。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secRemapUserid API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secServerAuthPluginInit - 初始化服务器认证插件

db2secServerAuthPluginInit API 是服务器认证插件的初始化 API，DB2 数据库管理器在装入该插件之后就立即调用此 API。

对于 GSS-API，在初始化时，该插件负责在 gssapi_server_auth_functions 结构中的 **serverPrincipalName** 参数中填充服务器的主体名称，并在 gssapi_server_auth_functions 结构中的 **serverCredHandle** 参数中提供服务器的凭证句柄。必须由 db2secServerAuthPluginTerm API 通过调用 gss_release_name 和 gss_release_cred 这两个 API 来释放已分配的用于存放主体名称和凭证句柄的内存。

API 和数据结构语法

```
SQL_API_RC SQL_API_FN db2secServerAuthPluginInit
                            ( db2int32 version,
                              void *server_fns,
                              db2secGetConDetails *getConDetails_fn,
                              db2secLogMessage *logMessage_fn,
                              char               **errormsg,
                              db2int32 *errormsglen );
```

db2secServerAuthPluginInit API 参数

version
输入。DB2 数据库管理器当前支持的 API 的最高版本号。db2secPlugin.h 中的 DB2SEC_API_VERSION 值包含 DB2 数据库管理器当前支持的 API 的最新版本号。

server_fns
输出。如果使用了 GSS-API 认证，那么此参数表示指向 DB2 数据库管理器为 db2secGssapiServerAuthFunctions_<version_number> 结构（也称为 gssapi_server_auth_functions_<version_number>）提供的内存的指针；如果使用了“用户标识/密码”认证，那么此参数表示指向 DB2 数据库管理器为 db2secUseridPasswordServerAuthFunctions_<version_number> 结构（也称为 userid_password_server_auth_functions_<version_number>）提供的内存的指针。db2secGssapiServerAuthFunctions_<version_number> 结构和 db2secUseridPasswordServerAuthFunctions_<version_number> 结构分别包含指向为 GSS-API 认证插件和“用户标识/密码”认证插件实现的 API 的指针。

server_fns 参数被强制类型转换为指向与插件已实现的版本相对应的 gssapi_server_auth_functions_<version_number> 结构的指针。gssapi_server_auth_functions_<version_number> 结构或 userid_password_server_auth_functions_<version_number> 结构的第一个参数将把插件已实现的 API 的版本告知给 DB2 数据库管理器。

注: 仅当 DB2 版本高于或等于插件已实现的 API 版本时才会进行强制类型转换。

在 gssapi_server_auth_functions_<version_number> 或 userid_password_server_auth_functions_<version_number> 结构中，应将 **plugintype**

参数设置为 DB2SEC_PLUGIN_TYPE_USERID_PASSWORD、DB2SEC_PLUGIN_TYPE_GSSAPI 或 DB2SEC_PLUGIN_TYPE_KERBEROS。可以在将来的 API 版本中定义其他值。

getConDetails_fn

输入。指向由 DB2 实现的 db2secGetConDetails API 的指针。db2secServerAuthPluginInit API 可以调用其他任何一个认证 API 中的 db2secGetConDetails API，以获取与数据库连接相关的详细信息。这些详细信息包括有关与连接相关联的通信机制的信息（例如，在使用 TCP/IP 协议时的 IP 地址），插件编写者在作出认证决定时可能需要参考这些详细信息。例如，插件可以不接受来自特定用户的连接，除非该用户是从特定 IP 地址进行连接的。是否使用 db2secGetConDetails API 是可选的。

如果在不涉及到数据库连接的情况下调用 db2secGetConDetails API，那么它将返回值 DB2SEC_PLUGIN_NO_CON_DETAILS，否则它将在成功时返回 0。

db2secGetConDetails API 具有以下两个输入参数：一个是 **pConDetails**，它是指向 db2sec_con_details_<version_number> 结构的指针；另一个参数是 **conDetailsVersion**，它是一个版本号，用于指示要使用哪个 db2sec_con_details 结构。当使用 db2sec_con_details1 时，其值为 DB2SEC_CON_DETAILS_VERSION_1，当使用 db2sec_con_details2 时，其值为 DB2SEC_CON_DETAILS_VERSION_2。建议使用的版本号是 DB2SEC_CON_DETAILS_VERSION_2。

当成功返回时，db2sec_con_details 结构（db2sec_con_details1 或 db2sec_con_details2）将包含以下信息：

- 用于连接至服务器的协议。可以在位于包含 include 目录的 sqlenv.h 文件中找到协议定义列表 (SQL_PROTOCOL_*)。此信息是在 **clientProtocol** 参数中填充的。
- 如果 **clientProtocol** 为 SQL_PROTOCOL_TCPIP 或 SQL_PROTOCOL_TCPIP4，那么此参数表示与服务器的入站连接的 TCP/IP 地址。此信息是在 **clientIPAddress** 参数中填充的。
- 客户机在尝试连接至的数据库名称。进行实例连接时不会设置此信息。此信息是在 **dbname** 和 **dbnameLen** 参数中填充的。
- 连接信息位图，它包含与 db2secValidatePassword API 的 **connection_details** 参数中说明的相同的详细信息。此信息是在 **connect_info_bitmap** 参数中填充的。
- 如果 **clientProtocol** 为 SQL_PROTOCOL_TCPIP6，那么此参数表示与服务器的入站连接的 TCP/IP 地址。此信息是在 **clientIP6Address** 参数中填充的，仅当 DB2SEC_CON_DETAILS_VERSION_2 用于 db2secGetConDetails API 调用时才会提供此信息。

logMessage_fn

输入。指向由 DB2 数据库管理器实现的 db2secLogMessage API 的指针。db2secClientAuthPluginInit API 可以调用 db2secLogMessage API 以将消息记录到 **db2diag** 日志文件中，以便进行调试或者供您参考。db2secLogMessage API 的第一个参数 (**level**) 指定将记录在 **db2diag** 日志文件中的诊断错误的类型，最后两个参数分别表示消息字符串及其长度。db2secLogMessage API 的第一个参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_LOG_NONE (0)：不记录
- DB2SEC_LOG_CRITICAL (1)：发生严重错误
- DB2SEC_LOG_ERROR (2)：发生错误

- DB2SEC_LOG_WARNING (3): 警告
- DB2SEC_LOG_INFO (4): 参考

仅当 db2secLogMessage API 的 **level** 参数的值小于或等于 **diaglevel** 数据库管理器配置参数的值时，消息文本才会出现在 **db2diag** 日志文件中。

例如，如果使用 DB2SEC_LOG_INFO 值，那么仅当数据库管理器配置参数 **diaglevel** 设置为 4 时，**db2diag** 日志文件中才会出现消息文本。

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secServerAuthPluginInit API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secServerAuthPluginTerm API - 清除服务器认证插件资源

db2secServerAuthPluginTerm API 释放服务器认证插件所使用的资源。

DB2 数据库管理器将在卸载服务器认证插件之前调用此 API。应该以这样一种方式来实现它：它将正确清除插件库拥有的任何资源，例如，释放由插件分配的任何内存，关闭仍处于打开状态的文件并关闭网络连接。插件负责跟踪这些资源以便释放这些资源。在任何 Windows 操作系统上都不会调用此 API。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secServerAuthPluginTerm)
                        ( char     **errormsg,
                          db2int32 *errormsglen );
```

db2secServerAuthPluginTerm API 参数

errormsg
输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secServerAuthPluginTerm API 不成功，就会返回此错误消息。

errormsglen
输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2secValidatePassword API - 验证密码

提供一种在数据库连接操作期间执行用户标识和密码样式认证的方法。

注：在客户端运行此 API 时，将使用执行 CONNECT 语句的用户的特权来运行 API 代码。仅当 **authentication** 配置参数设置为 CLIENT 时才会在客户端调用此 API。

在服务器端运行此 API 时，将使用实例所有者的特权来运行 API 代码。

如果认证需要特殊特权（例如，UNIX 上的 root 用户级别系统访问权），那么插件编写者应考虑上述情况。

如果密码有效，那么此 API 必须返回值 DB2SEC_PLUGIN_OK（表示成功）；如果密码无效，那么此 API 必须返回错误代码（例如 DB2SEC_PLUGIN_BADPWD）。

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *password,
                         db2int32 passwordlen,
                         const char *newpasswd,
                         db2int32 newpasswdlen,
                         const char *dbname,
                         db2int32 dbnamelen,
                         db2Uint32 connection_details,
                         void      **token,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secValidatePassword API 参数

userid
输入。要验证其密码的用户标识。

useridlen
输入。**userid** 参数值的长度（以字节计）。

usernamespace
输入。从其中获得用户标识的名称空间。

usernamespacelen
输入。**usernamespace** 参数值的长度（以字节计）。

usernamespacetype
输入。名称空间的类型。**usernamespacetype** 参数具有下列有效值（它们是在 db2secPlugin.h 中定义的）：

- DB2SEC_NAMESPACE_SAM_COMPATIBLE 对应于一个样式类似于 domain\myname 的用户名
- DB2SEC_NAMESPACE_USER_PRINCIPAL 对应于一个样式类似于 myname@domain.ibm.com 的用户名

目前，DB2 数据库系统仅支持 DB2SEC_NAMESPACE_SAM_COMPATIBLE 值。当未提供用户标识时，**usernamespacetype** 参数设置为值 DB2SEC_USER_NAMESPACE_UNDEFINED（此值是在 db2secPlugin.h 中定义的）。

password
输入。要验证的密码。

passwordlen
输入。**password** 参数值的长度（以字节计）。

newpasswd
输入。一个新密码（如果要更改密码）。如果未请求更改密码，那么此参数设置为 NULL。如果此参数不为 NULL，那么此 API 在将旧密码更改为新密码之前应验证旧密码。此 API 并非必须完成更改密码的请求，但是如果未完成此请求，那么应立即返回值 DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED 而不验证旧密码。

newpasswdlen
输入。**newpasswd** 参数值的长度（以字节计）。

dbname

输入。要连接至的数据库的名称。API 可以忽略 **dbname** 参数，如果它具有限制用户访问某些数据库的策略，那么它可以返回值 DB2SEC_PLUGIN_CONNECTIONREFUSED；否则这些用户将具有有效密码。此参数可以为 NULL。

dbnamelen

输入。**dbname** 参数值的长度（以字节计）。如果 **dbname** 参数为 NULL，那么此参数设置为 0。

connection_details

输入。这是一个 32 位参数，当前使用其中 3 位来存储以下信息:

- 最右边的一位指示用户标识的源是 db2secGetDefaultLoginContext API 中的缺省值，还是在连接期间显式提供的。
- 右边的第二位指示连接是本地连接（使用“进程间通信” (IPC) 的连接或者从分区数据库环境中的 db2nodes.cfg 中的其中一个节点进行的连接）还是远程连接（通过网络或循环进行的连接）。这使 API 能够决定同一机器上的客户机是否无需提供密码就可以连接至 DB2 服务器。由于存在基于操作系统的缺省“用户标识/密码”插件，因此允许从同一机器上的客户机中进行本地连接而无需提供密码（假定用户具有连接特权）。
- 右边的第三位指示 DB2 数据库管理器是从服务器端还是客户端调用此 API。

位值是在 db2secPlugin.h 中定义的:

- DB2SEC_USERID_FROM_OS (0x00000001) 指示用户标识是从操作系统中获取的，而不是在 CONNECT 语句上显式给定的。
- DB2SEC_CONNECTION_ISLOCAL (0x00000002) 指示本地连接。
- DB2SEC_VALIDATING_ON_SERVER_SIDE (0x0000004) 指示 DB2 数据库管理器是从服务器端还是客户端调用此 API 以验证密码。如果设置了此位值，那么 DB2 数据库管理器将从服务器端进行调用；否则，它将从客户端进行调用。

对于隐式认证，DB2 数据库系统的缺省行为是允许在连接时不进行任何密码验证。但是，插件开发者可以不允许进行隐式认证，这种情况下将返回 DB2SEC_PLUGIN_BADPASSWORD 错误。

token

输入。指向满足以下条件的数据的指针: 在当前连接期间，可以将此数据作为参数传递给后续 API 调用。可以调用的 API 包括 db2secGetAuthIDs API 和 db2secGetGroupsForUser API。

errormsg

输出。指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行 db2secValidatePassword API 不成功，就会返回此错误消息。

errormsglen

输出。指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

GSS-API 认证插件的必需 API 和定义

下表是 DB2 安全插件接口必需的 GSS-API 的完整列表。

受支持的 API 遵循下列规范：*类属安全性服务应用程序编程接口版本 2*（IETF RFC2743）和*类属安全性服务 API 版本 2：C 绑定*（IETF RFC2744）。在实现基于 GSS-API 的插件之前，应彻底了解这些规范。

表 39. GSS-API 认证插件必需的 API 和定义

API 类型	API 名称	描述
客户端 API	gss_init_sec_context	使用同级应用程序来启动安全上下文。
服务器端 API	gss_accept_sec_context	接受由同级应用程序启动的安全上下文。
服务器端 API	gss_display_name	将内部格式名称转换为文本。
公共 API	gss_delete_sec_context	删除已建立的安全上下文。
公共 API	gss_display_status	获取与 GSS-API 状态码相关联的文本错误消息。
公共 API	gss_release_buffer	删除缓冲区。
公共 API	gss_release_cred	释放与 GSS-API 凭证相关联的本地数据结构。
公共 API	gss_release_name	删除内部格式名称。
必需的定义	GSS_C_DELEG_FLAG	请求授权。
必需的定义	GSS_C_EMPTY_BUFFER	表示 gss_buffer_desc 不包含任何数据。
必需的定义	GSS_C_GSS_CODE	指示 GSS 主要状态码。
必需的定义	GSS_C_INDEFINITE	指示机制不支持上下文到期。
必需的定义	GSS_C_MECH_CODE	指示 GSS 次要状态码。
必需的定义	GSS_C_MUTUAL_FLAG	请求了相互认证。
必需的定义	GSS_C_NO_BUFFER	表示 gss_buffer_t 变量并不指向有效的 gss_buffer_desc 结构。
必需的定义	GSS_C_NO_CHANNEL_BINDINGS	没有通信信道绑定。
必需的定义	GSS_C_NO_CONTEXT	表示 gss_ctx_id_t 变量并不指向有效的上下文。
必需的定义	GSS_C_NO_CREDENTIAL	表示 gss_cred_id_t 变量并不指向有效的凭证句柄。
必需的定义	GSS_C_NO_NAME	表示 gss_name_t 变量并不指向有效的内部名称。
必需的定义	GSS_C_NO_OID	使用缺省认证机制。
必需的定义	GSS_C_NULL_OID_SET	使用缺省机制。
必需的定义	GSS_S_COMPLETE	已成功完成 API。
必需的定义	GSS_S_CONTINUE_NEEDED	未完成处理，必须使用从同级接收到的应答令牌来再次调用 API。

GSS-API 认证插件的限制

以下列表说明 GSS-API 认证插件的限制。

- 始终假定采用缺省安全性机制；因此，不存在 OID 注意事项。
- gss_init_sec_context() 中请求的唯一 GSS 服务是相互认证和授权。DB2 数据库管理器始终请求凭单以便授权，但是不使用该凭单来生成新的凭单。
- 仅请求了缺省上下文时间。
- 未将 gss_delete_sec_context() 中的上下文标记从客户机发送至服务器，反之亦然。
- 不支持匿名。
- 不支持通道绑定。
- 如果初始凭证到期，那么 DB2 数据库管理器不会自动对它们进行刷新。

- GSS-API 规范规定，即使 gss_init_sec_context() 或 gss_accept_sec_context() 失败，任一函数也必须返回一个标记以发送至同级。但是，由于存在 DRDA 局限性，因此，仅当 gss_init_sec_context() 失败并且在首次调用生成标记时，DB2 数据库管理器才会发送标记。

第 10 章 通信缓冲区出口库

DB2 for Linux, UNIX, and Windows 数据库管理器使客户和供应商能够复查通信缓冲区。在发送和接收通信缓冲区之前，外部的可信共享库用来访问通信缓冲区（在客户机与数据库服务器之间传递通信缓冲区）。这些外部库称为通信缓冲区出口库。

借助通信缓冲区出口库，您可以检查通信缓冲区，以便根据缓冲区的内容来提供解决方案，例如，审计解决方案或者其他安全性解决方案。通过 DB2 for Linux, UNIX, and Windows 能够访问从客户机接收到的每个缓冲区以及要发送至客户机的每个缓冲区。提供缓冲区之后才使用 DATA_ENCRYPT 认证或者 SSL 对其进行加密。DB2 for Linux, UNIX, and Windows 使用 DRDA 协议在客户机与服务器之间进行通信。按照 DRDA 协议对传递到通信缓冲区出口库的通信缓冲区进行格式化。通信缓冲区出口库必须了解用于通信的 DRDA 协议。

无论使用哪种通信协议，DB2 for Linux, UNIX, and Windows 都会提供缓冲区。各个通信缓冲区出口库一致地使用 TCPIP（IPv4 和 IPv6）、SSL、进程间通信 (IPC) 和命名管道。

DB2 for Linux, UNIX, and Windows 除了使缓冲区可用以外，还会使身份信息（其中包括为数据库连接建立的用户名和会话授权标识）可用。此信息对于涉及到诸如 Kerberos 的 GSS-API 插件的方案很有用。在这种情况下，没有标准用户名，而是具有数据库管理器用来派生用户名的一般凭单。不是只有通过查看通信缓冲区才能获取此详细信息。

数据库管理器将确保仅装入可信库。必须将这些库安装在只能由实例所有者修改的特定位置。而且，只有具备 SYSADM 权限的用户才能启用此库。启用加密（DATA_ENCRYPT 或 SSL）也需要具备此权限级别。

如果所提供的任何缓冲区中包含库认为有害的数据，那么通信缓冲区出口库可以终止连接。这些数据既包括发送至服务器的数据，又包括返回给客户机的数据。例如，通信缓冲区出口库可能会检测到从 SELECT 语句返回的数据不适合于客户机接收。来自库的返回码向数据库管理器指出必须终止连接。数据库管理器将停止客户机的该通信缓冲区或者任何其他通信缓冲区，并且会终止连接。

注: 第三方供应商通常会提供这些通信缓冲区出口库。DB2 for Linux, UNIX, and Windows 在 `sqllib/samples/security/commexit` 目录中提供了库的样本。您可以选择使用这些样本作为指南来开发您自己的库。

通信缓冲区出口库部署

必须考虑与部署通信缓冲区出口库有关的某些注意事项。

在典型方案中，由供应商提供通信缓冲区出口库。在这些方案中，由供应商提供的安装脚本来处理通信缓冲区出口库部署操作。如果您选择部署自己的库，那么可以按照此处概述的部署步骤来部署。

通信缓冲区出口库位置

通信缓冲区出口库必须存在于特定目录中。

数据库管理器在以下目录中查找通信缓冲区出口库:

Linux 和 UNIX 32 位
$DB2PATH/security32/plugin/commexit

Linux 和 UNIX 64 位
$DB2PATH/security64/plugin/commexit

Windows 32 位和 64 位
$DB2PATH\security\plugin\instance_name\commexit

注: 在 Windows 平台上，不会自动创建子目录 instance_name 和 commexit。实例所有者必须手动创建这两个目录。

通信缓冲区出口库命名约定和许可权

通信缓冲区出口库必须符合平台特定命名规则和许可权规则。

通信缓冲区出口库名称的最大长度（不包括文件扩展名和后缀“64”）只能为 32 个字节。

以下列表概述了每个平台上的库文件扩展名的命名约定:

AIX
扩展名必须为 .a 或 .so

注: 如果 .a 和 .so 扩展名同时存在，那么使用 .a。

Linux、HP IPF 和 Solaris
扩展名必须为 .so

Windows
扩展名必须为 .dll

以下列表概述了每个平台上的库文件的许可权:

UNIX 和 Linux
归实例所有者所有，并仅对实例所有者可读并可执行。

Windows
归 DB2AMINS 组的一个成员所有，并对 DB2ADMINS 组的一个成员可读并可执行。

示例

以下示例显示所有平台上的名为 mycommexit 的库上的通信缓冲区出口库扩展名:

- AIX 64 位 mycommexit.a 或 mycommexit.so
- Solaris 64 位、Linux 32 位或 64 位、IPF 上的 HP 64 位: mycommexit.so
- Windows 32 位: mycommexit.dll
- Windows 64 位: mycommexit64.dll

注: 仅在 Windows 64 位的库名上必须带有文件名后缀“64”。

当使用通信缓冲区出口库名称来更新数据库管理器配置时，使用库的全名但不带后缀“64”。更新数据库管理器配置时，不得指定该文件扩展名和标准路径。

以下示例显示了更新 Windows 64 位系统上的数据库管理器配置，将 mycommexit64.dll 库设置为通信缓冲区出口库的过程。

```
UPDATE DBM CFG USING COMM_EXIT_LIST mycommexit
```

注： **COMM_EXIT_LIST** 名称是区分大小写的，并且必须与库名精确匹配。

在 DB2 pureScale 环境外部启用通信缓冲区出口库

通常由第三方提供的安装脚本来执行此任务中所概述的步骤。概述这些步骤以帮助您启用您开发的通信缓冲区出口库。

开始之前

您必须具备 SYSADM 权限才能执行此任务中的步骤。

限制

通信缓冲区出口库文件必须遵循严格的文件许可权准则。有关这些准则的更多信息，请参阅“相关概念”。

过程

要启用通信缓冲区出口库，请完成下列步骤:

1. 停止数据库管理器。要停止数据库管理器，请运行 **db2stop** 命令。
2. 将此通信缓冲区出口库文件复制到正确的目录中。 有关通信缓冲区出口库必需的位置的更多详细信息，请参阅“相关概念”。如果需要，此文件可以是与另一个位置建立的符号链接。
3. 将数据库管理器配置参数 **COMM_EXIT_LIST** 更新为此库的名称。 要更新此配置参数，请使用 **UPDATE DBM CFG** 命令。
4. 启动数据库管理器。要启动数据库管理器，请运行 **db2start** 命令。

结果

已装入此库并且已将其初始化。

在 DB2 pureScale 环境中启用通信缓冲区出口库

通常由第三方提供的安装脚本来执行此任务中所概述的步骤。概述这些步骤以帮助您启用您开发的通信缓冲区出口库。

关于此任务

通过对没有版本号的库利用文件名中包含版本号的通信缓冲区出口库以及与此文件建立的符号链接，可以逐个成员部署此库。在这种情况下，不必停止整个实例，只需停止各个成员。

限制

通信缓冲区出口库文件必须遵循严格的文件许可权准则。有关这些准则的更多信息，请参阅“相关概念”。

过程

要启用通信缓冲区出口库，请完成下列步骤:

1. 将文件名中包含版本号的通信缓冲区出口库复制到正确的目录中。 有关通信缓冲区出口库必需的位置的更多详细信息，请参阅“相关概念”。
2. 创建从没有版本的库到文件名中包含版本的库的符号链接。
3. 将数据库管理器配置参数 **comm_exit_list** 更新为库的名称。 要更新此配置参数，请使用 **UPDATE DBM CFG** 命令。
4. 逐个停止每个成员。 要停止每个成员，请对每个成员运行 **db2stop** 命令。
5. 重新启动已停止的成员。 要启动已停止的成员，请运行 **db2start** 命令。

结果

已装入此库并且已将其初始化。

通信缓冲区出口库问题确定

有一些选项可用于帮助诊断通信缓冲区出口库的问题。

通信缓冲区出口库并未作为 DB2 for Linux, UNIX, and Windows 的一部分来提供。而是作为供您安装的库。它可能已由您正在使用的工具或应用程序自动安装并配置，或者也可以由您来编写。

数据库管理器配置参数 **comm_exit_list** 中指定的库名称表明了库的来源。

如果您遇到有关该库的任何问题，那么必须参阅该工具或应用程序的文档以确定必须采取的问题确定步骤。

通信缓冲区出口库可使用一个接口来写入 db2diag 日志文件。如果对该库是否正常工作存在疑问，那么可以检查 db2diag 日志文件。

如果关注通信缓冲区出口库的性能，那么可使用监视等待时间来调查该库所花的时间。有关这些监视工具的更多信息，请参阅相关参考。

开发通信缓冲区出口库

必须考虑与开发通信缓冲区出口库有关的某些注意事项。

在典型方案中，由供应商提供通信缓冲区出口库。在这些方案中，由供应商来开发通信缓冲区出口库。如果您选择开发自己的库，那么可以进行开发。

通信缓冲区出口库的装入方式

在数据库管理器已启动的情况下，会动态装入和初始化通信缓冲区出口库。此库中必须包含初始化函数 db2commexitInit。此函数称为库初始化函数。

库初始化函数会初始化所指定的通信缓冲区出口库。进行初始化时，会为数据库管理器提供调用库函数所需要的信息。库初始化函数接受下列参数:

- 调用此库的数据库实例可以支持的函数指针结构的最高版本。
- 指向某个结构的指针，该结构中包含指向所有需要实现的 API 的指针。
- 指向用于将日志消息添加至 db2diag 日志文件的函数的指针。
- 指向错误消息字符串的指针。
- 错误消息的长度。

初始化函数的函数特征符为:

```
SQL_API_RC SQL_API_FN db2commexitInit
 ( db2int32 version,
   void *commexit_fns,
   db2commexitLogMessage *logMessage_fn,
   char     **errormsg,
   db2int32 *errormsglen );
```

初始化函数是库中唯一使用规定函数名的函数。通过从初始化函数返回的函数指针来引用其他库函数。

此函数的特定任务是:

- 将函数指针强制类型转换为适当函数结构的指针。
- 指定指向库中其他函数的指针。
- 指定所返回的函数指针结构的版本号。

如果此库是采用 C++ 语言进行编译的，那么必须将 db2commexitInit 函数声明为 extern "C"。

通信缓冲区出口库 API

在通信缓冲区出口库中实现的 API。

db2commexitInit API - 初始化

当使用 **db2start** 命令启动数据库管理器时，会装入通信缓冲区出口库。紧接着装入此库之后，就会调用此函数。此函数负责初始化通信缓冲区出口库。此函数还负责将所有已实现的函数返回给数据库管理器。

如果此库是采用 C++ 语言进行编译的，那么必须将此函数声明为 extern "C"。

由于仅调用此函数一次，因此不需要保证此函数线程安全。

API 头文件

db2commexit.h

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitInit )
(
      db2int32 version,
      void *commexit_fns,
      db2commexitLogMessage *logMessage_fn,
      char     **errormsg,
      db2int32 *errormsglen
);
```

db2commexiInit API 参数

version

输入。此参数表示装入该库的实例支持的 API 的最高版本。db2commexit.h 中的值 DB2COMMEXIT_API_VERSION 包含数据库管理器当前支持的 API 的最新版本号。

commexit_fns

输出。此参数表示指向 db2commexitFunctions_<version_number> 结构的指针，此结构中包含指向为通信缓冲区出口库实现的 API 的指针。可能具有不同版本的 API，因此会将 **commexit_fns** 参数强制类型转换为与此库所实现的版本相对应的 db2commexitFunctions_<version_number> 结构。db2commexitFunctions_<version_number> 结构的第一个参数指示插件已实现的 API 的版本。

logMessage_fn

输入。此参数表示指向由 DB2 数据库系统实现的 db2commexitLogMessage API 的指针。db2commexitInit API 可以调用 db2commexitLogMessage API 以将消息记录到 db2diag 日志文件中，以便进行调试或者供您参考。db2commexitLogMessage API 的第一个参数指定记录在 db2diag 日志文件中的诊断错误的类型，最后两个参数表示消息字符串及其长度。db2commexitLogMessage API 的第一个参数的有效值为（这些有效值是在 db2commexit.h 中定义的）：

- DB2COMMEXIT_LOG_NONE: (0) 不记录
- DB2COMMEXIT_LOG_CRITICAL: (1) 服务器发生错误
- DB2COMMEXIT_LOG_ERROR: (2) 发生错误
- DB2COMMEXIT_LOG_WARNING: (3) 警告
- DB2COMMEXIT_LOG_INFO: (4) 参考

仅当 db2commexitLogMessage API 的“level”参数的值小于或等于 **diaglevel** 数据库管理器配置参数的值时，才会将消息文本记录在 db2diag 日志文件中。例如，如果您使用值 DB2SEC_LOG_INFO，那么仅当 **diaglevel** 数据库管理器配置参数设置为 4 时才会记录消息文本。

errormsg

输出。此参数表示指向由插件分配的 ASCII 错误消息字符串地址的指针，如果执行此函数不成功，就会返回此错误消息。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen

输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitTerm API - 终止

此函数用于释放通信缓冲区出口库所使用的资源。

数据库管理器将在 **db2stop** 处理过程中上载通信缓冲区出口库之前调用此 API。必须以这样一种方式来实现该 API: 它将正确清除该库拥有的任何资源。例如，该 API 必须释放该库分配的任何内存，关闭仍处于打开状态的文件并关闭网络连接。该库负责跟踪这些资源以便释放这些资源。

由于仅调用此函数一次，因此不需要保证此函数线程安全。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitTerm )
(
   char     **errormsg,
   db2int32 *errormsglen
);
```

db2commexitTerm API 参数

errormsg
输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen
输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitRegister API - 注册

此函数向连接注册代理。

每当代理接受套接字并开始在该套接字上接收和发送数据时，数据库管理器就会调用此函数。此活动通常与到数据库或实例附件的新 SQL 连接相关联。

将空闲连接分派给代理程序以处理来自客户机的新请求时，也会调用此函数。

此函数不直接与到数据库的 SQL 连接关联。该函数的一个输入参数用于区分新套接字和分配给新代理的现有套接字。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitRegister )
(
   void                          ** pConnectionContext,
   const db2commexitCommInfo_v1   * pCommInfo,
   db2int32                         state,
   db2int64                       * pReservedFlags,
   char                          ** errormsg,
   db2int32                       * errormsglen
);
```

db2commexitRegister API Parameters

pConnectionContext
输入/输出。此参数表示指向特定于通信缓冲区出口库的数据的指针。此指针特定于入站连接。此参数将作为输出传递给为该连接的每次函数调用的输入。该库可以分配和存储特定于连接的信息并使其在每次函数调用中可用。在调用 db2commexitDeregister 时必须释放该参数的内存。数据库管理器无法访问由此参数指向的内存。

pCommInfo

输入。此参数表示指向结构的指针，该结构包含用于标识数据库服务器的信息和入站连接的协议特定信息。该结构中的某些字段未设置，直到与客户机交换多个缓冲区为止。在对 db2commexitRecv 和 db2commexitSend 的调用中可使用这些字段。此方案特定适用于 **inbound_appl_id**、**outbound_appl_id** 和 **connection_type**。一旦知道这些值之后，**connection_type** 参数将指示连接是本地数据库连接还是网关连接。

状态

输入。指示在什么条件下将调用该函数。可能的值包括:

- NEW_CONNECTION - 指示新的物理入站客户机连接。
- AGENT_ASSOCIATION - 指示重新成为活动连接并且与处理请求的代理相关联的现有空闲客户机连接。

pReservedFlags

输入/输出。保留以备将来使用。对于输出，必须将该值设为 0。

errormsg

输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen

输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitDeregister API - 取消注册

此函数用于将代理从关联的连接中释放出来。

每当代理停止处理连接上的请求时，数据库管理器就会调用此函数。当与客户机的物理连接终止，或客户机处于空闲状态且代理与其取消关联时，会发生此情况。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitDeregister )
(
   void                          * pConnectionContext,
   const db2commexitCommInfo_v1  * pCommInfo,
   db2int32                        state,
   db2int64                      * pReservedFlags,
   char                         ** errormsg,
   db2int32                      * errormsglen
);
```

db2commexitDeregister API 参数

pConnectionContext

输入。此参数表示指向特定于通信缓冲区出口的数据的指针。此指针特定于入站连接。此参数将作为输出传递给为该连接的每次函数调用的输入。数据库管理器无法访问由此参数指向的内存。必须使用此函数取消分配此内存。

pCommInfo
输入。此参数表示指向结构的指针，该结构包含用于标识数据库服务器的信息和入站连接的协议特定信息。

状态
输入。指示在什么条件下将调用该函数。可能的值包括:

- CONNECTION_TERM - 指示已终止与客户机的物理连接。
- AGENT_DISASSOCIATION - 指示客户机处于空闲状态且代理已与其取消关联。

pReservedFlags
输入/输出。保留以备将来使用。对于输出，必须将该值设为 0。

errormsg
输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen
输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitRecv API - 接收

将对数据库管理器从客户机接收的每个缓冲区调用此函数。

在从客户机接收通信缓冲区之后，数据库管理器将立即调用此函数。在对缓冲区解密之后将调用该函数，以便通信缓冲区出口库可以访问未加密的缓冲区。

API 头文件

db2commexit.h

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitRecv )
(
   void                          * pConnectionContext,
   const db2commexitCommInfo_v1  * pCommInfo,
   const db2commexitBuffer       * pBuffer,
   db2int64                      * pReservedFlags,
   char                         ** errormsg,
   db2int32                      * errormsglen
);
```

db2commexitRecv API 参数

pConnectionContext
输入。此参数表示指向特定于通信缓冲区出口的数据的指针。此指针特定于入站连接。此参数将作为输入传递给该连接的每次函数调用。数据库管理器无法访问由此参数指向的内存。必须使用此函数取消分配此内存。

pCommInfo
输入。此参数表示指向结构的指针，该结构包含用于标识数据库服务器的信息和入站连接的协议特定信息。该结构中的某些字段未设置，直到与客户机交换多个缓冲区为止。在对 db2commexitRecv 和 db2commexitSend 的调用中可使用这些字段。此方案特定适用于 **inbound_appl_id**、**outbound_appl_id** 和 **connection_type**。

pBuffer

输入。此参数表示指向结构的指针，该结构包含数据库管理器接收的缓冲区的长度以及指向该缓冲区的指针。如果该缓冲区已加密，那么在调用此函数前会对其解密。

pReservedFlags

输入/输出。设置位 DB2COMMEXIT_RECV_IN_FLAG_END_DECRYPT 以指示这是为已加密的 DSS 对此函数的最后一次调用。作为输入传递的 DSS 的长度指示已加密 DSS 的长度。但是，随后会对 DSS 解密并除去填充空格。因为始终存在填充空格，所以 DSS 的长度始终小于指示的长度。在 pBuffer 结构中指示的长度是 DSS 的最终数据。如果添加了完整块大小的填充空格，那么该长度可能为零。

对于输出，将保留此字段以供将来使用。对于输出，必须将该值设为 0。

errormsg

输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen

输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitSend API - 发送

将对数据库管理器发送给客户机的每个缓冲区调用此函数。

在将通信缓冲区发送给客户机之前，数据库管理器将调用此函数。在对缓冲区加密之前将调用该函数，以便通信缓冲区出口库可以访问未加密的缓冲区。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitSend )
(
   void                          * pConnectionContext,
   const db2commexitCommInfo_v1  * pCommInfo,
   const db2commexitBuffer       * pBuffer,
   db2int64                      * pReservedFlags,
   char                         ** errormsg,
   db2int32                      * errormsglen
);
```

db2commexitSend API 参数

pConnectionContext

输入。此参数表示指向特定于通信缓冲区出口的数据的指针。此指针特定于入站连接。此参数将作为输出传递给为该连接的每次函数调用的输入。数据库管理器无法访问由此参数指向的内存。

pCommInfo

输入。此参数表示指向结构的指针，该结构包含用于标识数据库服务器的信息和入站连接的协议特定信息。该结构中的某些字段未设置，直到与客户机交换多个缓冲

区为止。在对 db2commexitRecv 和 db2commexitSend 的调用中可使用这些字段。此方案特定适用于 **inbound_appl_id**、**outbound_appl_id** 和 **connection_type**。

pBuffer
输入。此参数表示指向结构的指针，该结构包含发送给客户机的缓冲区的长度以及指向该缓冲区的指针。如果该缓冲区已加密，那么在调用此函数前会对其解密。

pReservedFlags
输入/输出。如果数据库管理器遇到错误且必须清除准备发送给客户机的某些缓冲区，那么会设置位 DB2COMMEXIT_SEND_IN_FLAG_PURGE。这些缓冲区中的某些已作为输入传递给通信缓冲区出口库。

对于输出，将保留此字段以供将来使用。对于输出，必须将该值设为 0。

errormsg
输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen
输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitUserIdentity API - 用户身份

通过调用此函数可提供当前套接字的用户身份。

会调用此函数以通知通信缓冲区出口库，让其知道用来建立连接的用户名和会话授权标识。如果因为可信上下文切换用户或 SET SESSION AUTHORIZATION 而导致这些参数更改，那么也会调用此函数。直到数据库管理器验证用户身份之后，才会确定用户名和会话授权标识。直到调用 db2commexitRegister 并多次调用 db2commexitSend 和 db2commexitRecv 函数之后，才会调用此函数以执行认证。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitUserIdentity )
(
   void                          * pConnectionContext,
   const db2commexitCommInfo_v1  * pCommInfo,
   db2int32                        state,
   db2int32                        usernameLen,
   const char                    * pUserame,
   db2int32                        sessionAuthidLen,
   const char                    * pSessionAuthid,
   db2int64                      * pReservedFlags,
   char                         ** errormsg,
   db2int32                      * errormsglen
);
```

db2commexitUserIdentity API 参数

pConnectionContext
输入。此参数表示指向特定于通信缓冲区出口的数据的指针。此指针特定于入站连接。此参数将作为输出传递给为该连接的每次函数调用的输入。数据库管理器将无法访问由此参数指向的内存。

pCommInfo
输入。此参数表示指向结构的指针，该结构包含用于标识数据库服务器的信息和入站连接的协议特定信息。该结构中的某些字段未设置，直到与客户机交换多个缓冲区为止。在对 db2commexitRecv 和 db2commexitSend 的调用中可使用这些字段。此方案特定适用于 **inbound_appl_id**、**outbound_appl_id** 和 **connection_type**。

状态
输入。指示在什么条件下将调用该函数。可能的值包括:

- DB2COMMEXIT_USERIDENT_NEW_CONNECTION - 新连接。
- DB2COMMEXIT_USERIDENT_TC_SWITCH_USER - 发出了可信上下文切换用户请求。
- DB2COMMEXIT_USERIDENT_SET_SESSION_USER - 发出了 SET SESSION AUTHORIZATION SQL 语句以更改会话授权标识。

usernameLen
输入。**pUsername** 的长度。

pUsername
输入。用来建立连接的用户名。

sessionAuthidLen
输入。**pSessionAuthid** 的长度。

pSessionAuthid
输入。为此连接建立的会话授权标识。

pReservedFlags
输入/输出。保留以备将来使用。对于输出，必须将该值设为 0。

errormsg
输出。此参数表示指向由通信缓冲区出口库分配的 ASCII 错误消息字符串地址的指针。如果执行此函数不成功，那么可能返回此错误消息字符串。调用 db2commexitFreeErrorm–sg 无法释放此内存。

errormsglen
输出。此参数表示指向一个用于指示 **errormsg** 参数中的错误消息字符串长度（以字节计）的整数的指针。

db2commexitFreeErrormsg API - 释放错误消息内存

此函数释放用来存放上次调用 API 所产生的错误消息的内存。

API 头文件

```
db2commexit.h
```

API 和数据结构语法

```
SQL_API_RC ( SQL_API_FN * db2commexitFreeErrormsg )
 ( char * errormsg );
```

db2commexitFreeErrormsg API 参数

errormsg
输入。指向上次调用 API 时返回的错误消息的指针。

通信缓冲区出口库函数结构

db2commexitInit 函数采用 void * commexit_fns 参数。此参数会强制类型转换为特定于版本的结构，此结构中包含由通信缓冲区出口库实现的所有函数。db2commexitInit 函数必须指定函数指针，以便数据库管理器可以调用这些函数。

必须完成的结构（其中包括每个 API 的函数指针）如下。

```
struct db2commexitFunctions_v1
{
    db2int32 version;

    SQL_API_RC ( SQL_API_FN * db2commexitTerm )
    (
       char     **errormsg,
       db2int32 *errormsglen
    );

    SQL_API_RC ( SQL_API_FN * db2commexitRegister )
    (
       void                          ** ppConnectionContext,
       const db2commexitCommInfo_v1  * pCommInfo,
       db2int32                        state,
       db2int64                      * pReservedFlags,
       char                          ** errormsg,
       db2int32                      * errormsglen
    );

    SQL_API_RC ( SQL_API_FN * db2commexitDeregister )
    (
       void                          * pConnectionContext,
       const db2commexitCommInfo_v1  * pCommInfo,
       db2int32                        state,
       db2int64                      * pReservedFlags,
       char                          ** errormsg,
       db2int32                      * errormsglen
    );

    SQL_API_RC ( SQL_API_FN * db2commexitRecv )
    (
       void                          * pConnectionContext,
       const db2commexitCommInfo_v1  * pCommInfo,
       const db2commexitBuffer       * pBuffer,
       db2int64                      * pReservedFlags,
       char                          ** errormsg,
       db2int32                      * errormsglen
    );

    SQL_API_RC ( SQL_API_FN * db2commexitSend )
    (
       void                          * pConnectionContext,
       const db2commexitCommInfo_v1  * pCommInfo,
       const db2commexitBuffer       * pBuffer,
       db2int64                      * pReservedFlags,
       char                          ** errormsg,
       db2int32                      * errormsglen
    );

    SQL_API_RC ( SQL_API_FN * db2commexitUserIdentity )
    (
       void                          * pConnectionContext,
       const db2commexitCommInfo_v1  * pCommInfo,
       db2int32                        state,
       db2int32                        usernameLen,
       const char                    * pUserame,
       db2int32                        sessionAuthidLen,
```

```
      const char                   * pSessionAuthid,
      db2int64                     * pReservedFlags,
      char                        ** errormsg,
      db2int32                     * errormsglen
   );

   SQL_API_RC ( SQL_API_FN * db2commexitFreeErrormsg )
   (
      char * errormsg
   );

};
```

通信缓冲区出口库信息结构

信息结构指示当前物理连接的通信协议信息。

以下是传递给每个通信缓冲区出口库函数的 db2commexitCommInfo_v1 结构。此结构包括在 db2commexit.h 文件中。

```
struct db2commexitIPV4Info
{
   sockaddr_in client_sockaddr;
   sockaddr_in server_sockaddr;
};

struct db2commexitIPV6Info
{
   sockaddr_in6 client_sockaddr;
   sockaddr_in6 server_sockaddr;
};

struct db2commexitIPCInfo
{
   void * pSharedMemSegmentHandle;
};

struct db2commexitNamedPipeInfo
{
   void * handle;
};


struct db2commexitCommInfo_v1
{
   db2int32 clientProtocol;   // SQL_PROTOCOL_ ...
   db2int32 connectionType; // unknown, local or gateway

   db2int32 hostnameLen;
   db2int32 instanceLen;
   db2int32 dbnameLen;
   db2int32 dbaliasLen;
   db2int32 inbound_appl_id_len;
   db2int32 outbound_appl_id_len;

   db2int32 reserved1;
   db2int32 reserved2;

   db2NodeType member;

   char hostname[SQL_HOSTNAME_SZ+1];
   char instance[DB2COMMEXIT_INSTANCE_SZ + 1];
   char dbname[DB2COMMEXIT_DBNAME_SZ + 1];
   char dbalias[DB2COMMEXIT_DBNAME_SZ + 1];
   char inbound_appl_id[SQLM_APPLID_SZ + 1];
   char outbound_appl_id[SQLM_APPLID_SZ + 1];
```

```
   char reservedChar1[128];

   union
   {
      db2commexitIPV4Info ipv4Info;
      db2commexitIPV6Info ipv6Info;
      db2commexitIPCInfo  ipcInfo;
      db2commexitNamedPipeInfo namedPipeInfo;
   }
};
```

通信缓冲区出口库缓冲区结构

缓冲区结构是作为 db2commexitSend 和 db2commexitRecv 函数的输入来传递的一种结构。

缓冲区结构如下:

```
struct db2commexitBuffer
{
   const unsigned char * pBuffer;
   db2int64 buffer_len;

   db2int32 reserved1;
   db2int32 reserved2;
};
```

通信缓冲区出口库控制连接

通信缓冲区出口库随时都可以强制断开与客户机的连接。

对 db2commexitUserIdentity、db2commexitRegister、db2commexitDeregister、db2commexitRecv 或 db2commexitSend 进行任何调用时，如果通信缓冲区出口库返回适当的错误返回码，那么数据库管理器会立即关闭与客户机的连接。

此功能允许通信缓冲区出口库根据所复查的缓冲区来确定是否进行了某项不适当的活动。如果进行了这样一项确定，那么可以防止数据库管理器对该连接执行任何进一步的操作。

通信缓冲区出口库 API 版本

DB2 数据库系统支持对通信缓冲区出口库 API 版本进行编号。这些版本号是从 1 开始的整数。

数据库管理器传递给安全库初始化函数的版本号是受支持的 API 的最高版本号。如果该库可以支持更高的 API 版本，那么它必须返回数据库管理器请求的版本的函数指针。如果该库仅支持更低版本的 API，那么该库必须定义它支持的版本的函数指针。在任何一种情况下，库初始化函数都必须在函数结构的版本字段中返回此 API 支持的版本号。

通信缓冲区出口库 API 版本仅在必要时才会更改。例如，更改了该 API 的参数时。版本号不会随着数据库管理器的发行版号自动更改。

版本号允许引入新的或更改的 API。将会保留对旧版本的库支持。

通信缓冲区出口库错误处理和返回码

当通信缓冲区出口库 API 发生错误时，此 API 可以在 **errormsg** 字段中返回一个 ASCII 文本字符串。该 ASCII 文本字符串能对问题提供比返回码更具体的描述。数据库管理器会将整个该字符串写至 db2diag 日志文件。

通信缓冲区出口库必须分配用于存放这些错误消息的内存。因此，该出口库还必须提供以下 API 来释放此内存：db2commexitFreeErrormsg。

除了 **errormsg** 字段以外，在初始化时，还会将消息记录函数指针 logMessage_fn 传递给通信缓冲区出口库。该出口库可以使用此函数将所有调试信息记录到 db2diag 日志文件中。例如：

```
// Log an message indicate init successful
  (*(logMessage_fn))(DB2COMMEXIT_LOG_CRITICAL,
                     "comm exit initialization successful",
                      strlen("comm. exit initialization successful"));
```

有关 db2secLogMessage 函数的每个参数的更多详细信息，请参阅相关参考中的初始化 API db2commexitInit。

返回码

表 40. 通信缓冲区出口库可以返回给数据库管理器的返回码。

返回码	定义值	详细信息
0	DB2COMMEXIT_SUCCESS	成功执行
-1	DB2COMMEXIT_ERR_UNKNOWN	库发生了异常错误。
-2	DB2COMMEXIT_ERR_DROP_CONNECTION	该出口库确定必须终止为其调用该出口库的连接。

通信缓冲区出口库开发限制

开发通信缓冲区出口库时必须考虑某些限制和注意事项。

限制

C 链接

通信缓冲区出口库必须以 C/C++ 编写并使用 C-linkage 方式进行链接。头文件将提供原型和实现库所需要的数据结构，并且仅对 C/C++ 提供错误代码定义。如果此库是采用 C++ 语言进行编译的，那么必须将 db2commexitInit 函数声明为 extern "C"。

信号处理程序

通信缓冲区出口库不得安装信号处理程序或更改信号掩码。这样做会妨碍数据库管理器的信号处理程序。妨碍数据库管理器信号处理程序就会严重妨碍报告功能和从错误进行恢复的功能。

异常

通信缓冲区出口库 API 不得抛出 C++ 异常。此类异常会妨碍数据库管理器错误处理。

线程安全

通信缓冲区出口库函数必须是线程安全的。db2commexitInit 和 db2commexitTerm 函数是唯一没有此要求的 API。

出口处理程序

通信缓冲区出口库不得安装出口处理程序或 pthread_atfork 处理程序。不支持使用出口处理程序，因为通信缓冲区出口库是在数据库管理器处理出口前卸载的。

Fork/线程

通信缓冲区出口库不得调用 fork 或重建新线程，因为此情况会导致未定义行为（例如，数据库管理器中包含陷阱）。

库依赖项

在 Linux 和 UNIX 上，通信缓冲区出口库是从 setuid 或 setgid 进程装入的。它不能依赖于 **LD_LIBRARY_PATH**、**SHLIB_PATH** 或 **LIBPATH** 环境变量来查找从属库。因此，该出口库不得依赖于其他库，除非可以通过其他方法（例如下列情况）来访问任何从属库:

- 它们存在于 /lib 或 /usr/lib 中。
- 它们所在的目录是在操作系统范围内指定的（例如在 Linux 上的 ld.so.conf 文件中）。
- 它们是在出口库本身的 RPATH 中指定的。

符号冲突

应尽可能使用能降低发生符号冲突的可能性的任何可用选项（例如，可减少解除绑定外部符号引用的那些选项）来编译和链接通信缓冲区出口库。例如，在 HP、Solaris 和 Linux 上使用 ″-Bsymbolic″ 链接程序选项有助于防止发生与符号冲突相关的问题。但是，对于在 AIX 上编写的库，不要显式或隐式使用“-brtl”链接程序选项。

32 位与 64 位注意事项

数据库管理器有 32 位和 64 位版本，取决于平台。对于 32 位数据库管理器，必须启用 32 位通信缓冲区出口库，而对于 64 位数据库管理器，必须启用 64 位通信缓冲区出口库。不能混用两种库。

存储过程、触发器和其他内部 SQL

与服务器交互的存储过程被传递至通信缓冲区出口库。许多交互并非通过标准通信信道进行，并且也不适合用于出口库的模型。类似地，触发器和内部 SQL 的其他源不通过标准通信信道传递，也不会传递至通信缓冲区出口库。

不得操纵通信缓冲区

通信缓冲区出口库不应操纵或更改其传递的缓冲区。

滚动更新支持

DB2 for Linux, UNIX, and Windows 支持更新 DB2 pureScale 环境 中的个别成员的修订包级别但不停止其他成员。这称为滚动更新。类似地，正如通信缓冲区出口库的部署一节中概述，可以更新个别成员上使用的库级别。可能存在这样一种情况，两个不同版本的通信缓冲区出口库在两个不同成员上同时运行，每个成员处于不同的修订包级别。通信缓冲区出口库必须容忍此类情况而不发生错误。

通信缓冲区出口库 API 调用顺序

API 调用顺序可能会随特定方案而有所不同。

下列主题概述了您在开发通信缓冲区出口库时必须知道的特定方案。这些主题可以帮助您确定最适合于您所在环境的调用顺序。

API 调用顺序 - 单个代理程序中的正常连接

最典型的情况是客户机连接至数据库管理器，发出一些 SQL，然后断开连接。

在这种情况下，单个代理程序或线程将处理连接，并且进行下列调用:

1. 对于新的套接字连接，调用 db2commexitRegister。
2. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
3. 对于新连接，调用 db2commexitUserIdentity
4. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
5. 调用 db2commexitDeregister 以终止套接字连接。

API 调用顺序 - 不存在连接重置的连接

此方案说明了通过现有套接字建立的连接。客户机不必先发出连接重置即可启动另一个 SQL 连接。

一旦数据库管理器从客户机接收到 SQL CONNECT 语句，它就会隐式执行内部连接重置，然后再继续进行连接。不会更改套接字的状态，这些是常规请求和应答。在这种情况下，单个代理程序将处理所有请求。通过 db2commexitRecv 使包含来自客户机的连接请求的缓冲区可用时，通信缓冲区出口库能够确定解析此缓冲区时启动的新连接。进行了下列调用:

1. 对于新的套接字连接，调用 db2commexitRegister。
2. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
3. 对于新连接，调用 db2commexitUserIdentity
4. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
5. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
6. 对于新连接，调用 db2commexitUserIdentity。
7. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
8. 调用 db2commexitDeregister 以终止套接字连接。

注:
即使数据库管理器已处理两个 SQL 连接，也仅分别调用 db2commexitRegister 和 db2commexitDeregister 一次。

API 调用顺序 - 可信上下文和切换用户

此方案类似于不存在连接重置的连接。它们之间的区别在于客户机将请求可信上下文切换用户，而不发送新的 SQL 连接请求。

进行了下列调用:

1. 对于新的套接字连接，调用 db2commexitRegister。
2. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
3. 对于新连接，调用 db2commexitUserIdentity
4. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
5. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。

   在将来某个时间，客户机将向服务器发送可信上下文切换用户请求，以对连接切换用户。

6. 对于可信上下文切换用户，调用 db2commexitUserIdentity。
7. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
8. 调用 db2commexitDeregister 以终止套接字连接。

API 调用顺序 - 连接集中器

此方案说明了使用连接集中器时的 API 调用顺序。连接集中器功能部件允许数据库管理器处理的客户机数超过协调代理程序数或线程数。

一旦客户机达到工作单元边界，但是不立即发送另一个请求，就会将客户机套接字放入空闲池中。先前处理客户机请求的代理程序将处理另一个客户机。一旦空闲套接字具有要读取的数据时，分派器就会查找空闲代理程序以处理此数据。在 SQL 连接的生存期内，可能有多个用于处理客户机请求的代理程序。每当将套接字移入和移出空闲池时，就会调用 db2commexitDeregister 和 db2commexitRegister。进行了下列调用:

1. 对于新的套接字连接，调用 db2commexitRegister。
2. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
3. 对于新连接，调用 db2commexitUserIdentity
4. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。

   客户机不会立即发送另一个请求，并且会将套接字放入空闲池中。
5. 调用 db2commexitDeregister 以解除与代理程序的关联。

   在将来某个时间，客户机会在分派器选择空闲代理程序时发送另一个请求，此时间将可能与先前所使用的时间不同:
6. 调用 db2commexitRegister 以与代理程序相关联。
7. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
8. 调用 db2commexitDeregister 以终止套接字连接。

注: 对于单个 SQL 连接，会多次调用 db2commexitRegister 和 db2commexitDeregister。

API 调用顺序 - SET SESSION AUTHORIZATION 语句

此方案说明了使用 SET SESSION AUTHORIZATION 语句时的 API 调用顺序。

SET SESSION AUTHORIZATION 语句会更改正用于当前连接的会话授权标识。会调用 Db2commexitUserIdentity 以通知通信缓冲区出口库，让其知道已更改当前连接的身份信息。进行了下列调用:

1. 对于新的套接字连接，调用 db2commexitRegister。
2. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理认证。
3. 对于新连接，调用 db2commexitUserIdentity。
4. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。

   用户发出 SET SESSION AUTHORIZATION 语句。此请求会传递至 db2commexitRecv。它与其他 SQL 语句没有区别。
5. 对于 SET SESSION AUTHORIZATION，调用 db2commexitUserIdentity。
6. 有可能多次调用 db2commexitRecv 和 db2commexitSend 以处理客户机 SQL 请求。
7. 调用 db2commexitDeregister 以终止套接字连接。

有关设置目标逻辑节点的注意事项

使用 *DB2NODE* 变量或者使用 **SET CLIENT** 命令设置目标逻辑节点时必须考虑一些注意事项。

在分区数据库环境中，如果客户机通过 *DB2NODE* 变量指定的成员不是已将此客户机配置为要连接至的成员，那么数据库管理器会将连接切换为连接至此变量中指定的新成员。客户机连接会通过所连接的成员连接至远程成员。在这种情况下，在这两个成员中都会调用通信缓冲区出口库。要注意一些特点:

- 在所连接的成员中，客户机地址反映实际的客户机。
- 在远程成员中，客户机地址将反映所连接的成员。
- 所连接的成员中的出站应用程序标识与远程成员中的入站应用程序标识相同。

一旦已经建立了这些应用程序标识，就会将 db2commexitCommInfo_v1 结构中的 **connectionType** 设置为 GATEWAY。

有关连接网关的注意事项

当数据库管理器充当另一个 DRDA 数据库服务器的连接网关时，必须考虑一些注意事项。

当 DB2 for Linux, UNIX, and Windows 充当连接网关时，会按照与标准连接相同的方式来调用通信缓冲区出口库。一旦完成了验证并且建立了应用程序标识，就会将 db2commexitCommInfo_v1 结构中的 **connectionType** 设置为 GATEWAY。outbound_application_id 与 DRDA 数据库服务器中的连接的应用程序标识相匹配。

有关 DATA_ENCRYPT 的注意事项

使用 DATA_ENCRYPT 认证类型时，必须考虑一些注意事项。

处理使用 DATA_ENCRYPT 认证类型保护的通信时需要特别注意。与 SSL 不同，要支持 DATA_ENCRYPT，需要进行加密和解密；由数据库管理器在从客户机接收数据之后以及将应答发送给客户机之前进行加密和解密。

接收和 DATA_ENCRYPT

从客户机接收到已加密的 DSS 时，数据库管理器会根据需要对缓冲区进行解密。也就是说，未同时对整个缓冲区进行解密。将已加密的数据解密之后用来调用通信缓冲区出口库。

DSS 长度或者 DSS 继续长度（如果 DSS 长于逻辑记录）包含已加密的 DSS 的长度，而不包含已解密的缓冲区的长度。由于进行加密时始终会添加填充空格，因此，此长度始终大于纯文本的长度。最多可为 DSS 填充 8 个字节。

最后一次调用 db2CommexitRecv 时，会将 *DB2COMMEXIT_RECV_IN_FLAG_END_DECRYPT* 标志作为输入来传递以指示已加密的 DSS 的末尾。

注: 在这种情况下，长度可能为 0，表示已添加填充部分的整个块大小。

发送和 DATA_ENCRYPT

对客户机的 DSS 应答进行加密时，缓冲区中可能包含多个纯文本 DSS 和已加密的 DSS，这些 DSS 会发送至客户机。当这些 DSS 准备好时，会将它们作为输入传递给

db2commexitSend– 例程。由于在加密之前必须将纯文本数据用作输入，因此将按照一次完成一个传递的方式来完成这些传递。数据库管理器可能会遇到错误情况，这种错误情况要求数据库管理器清除先前已准备好、但是未发送的 DSS。通信缓冲区出口库可能已了解这些库。在长度为 0 且 DB2COMMEXIT_SEND_IN_FLAG_PURGE 标志指示已进行清除的情况下调用 db2CommexitSend 函数。

第 11 章 审计设施记录布局

从审计日志中抽取审计记录时，每个记录将具有下列各表中显示的其中一种格式。每个表前面都有一个样本记录。

该记录中每一项的描述显示在关联的表中，一次显示一行。表中每项的显示顺序与抽取操作后各项在定界文件中的输出顺序相同。

注:

1. 根据审计事件而定，并非审计记录中的所有字段都有值。如果字段中没有值，那么在审计输出中将不显示该字段。
2. 某些字段（如“尝试的访问”）以定界的 ASCII 格式存储为位图。但是，在此平面报告文件中，这些字段将显示为一组字符串，表示位图值。

审计记录对象类型

下表显示了对于每种审计记录对象类型，它是否可以生成 CHECKING、OBJMAINT 和 SECMAINT 事件。

表 41. 基于审计事件的审计记录对象类型

对象类型	CHECKING 事件	OBJMAINT 事件	SECMAINT 事件
ACCESS_RULE			X
ALIAS	X	X	
ALL	X		
AUDIT_POLICY	X	X	
BUFFERPOOL	X	X	
CHECK_CONSTRAINT		X	
DATABASE	X		X
DATA TYPE		X	
EVENT_MONITOR	X	X	
FOREIGN_KEY		X	
FUNCTION	X	X	X
FUNCTION MAPPING	X	X	
GLOBAL_VARIABLE	X	X	X
HISTOGRAM TEMPLATE	X	X	
INDEX	X	X	X
INDEX EXTENSION		X	
INSTANCE	X		
JAR_FILE		X	
METHOD_BODY	X	X	X
MODULE	X	X	X
NICKNAME	X	X	X
NODEGROUP	X	X	

表 41. 基于审计事件的审计记录对象类型 (续)

对象类型	CHECKING 事件	OBJMAINT 事件	SECMAINT 事件
NONE	X	X	X
OPTIMIZATION PROFILE	X		
PACKAGE	X	X	X
PACKAGE CACHE	X		
PRIMARY_KEY		X	
REOPT_VALUES	X		
ROLE	X	X	X
SCHEMA	X	X	X
SECURITY LABEL		X	X
SECURITY LABEL COMPONENT		X	
SECURITY POLICY		X	X
SEQUENCE	X	X	
SERVER	X	X	X
SERVER OPTION	X	X	
SERVICE CLASS	X	X	
STORED_PROCEDURE	X	X	X
SUMMARY TABLES	X	X	X
TABLE	X	X	X
TABLESPACE	X	X	X
THRESHOLD	X	X	
TRIGGER		X	
TRUSTED CONTEXT	X	X	X
TYPE MAPPING	X	X	
TYPE&TRANSFORM	X	X	
UNIQUE_CONSTRAINT		X	
USER MAPPING	X	X	
USER_TEMPORARY_TABLE	X	X	X
VIEW	X	X	X
WORK ACTION SET	X	X	
WORK CLASS SET	X	X	
WORKLOAD	X	X	X
WRAPPER	X	X	
XSR 对象	X	X	X

AUDIT 事件的审计记录布局

下表显示了 AUDIT 事件的审计记录布局。

样本审计记录:

```
timestamp=2007-04-10-08.29.52.000001;
category=AUDIT;
audit event=START;
event correlator=0;
event status=0;
userid=newton;
authid=NEWTON;
application id=*LOCAL_APPLICATION;
application name=db2audit.exe;
```

表 42. AUDIT 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: AUDIT
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 AUDIT 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名称。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节	SMALLINT	审计事件发生时正在使用的程序包中的节号。
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。

表 42. AUDIT 事件的审计记录布局 (续)

名称	格式	描述
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。
策略名称	VARCHAR(128)	审计策略名称。
策略关联对象类型	CHAR(1)	与审计策略关联的对象的类型。可能的值包括: • N = 昵称 • S = MQT • T = 表（无类型） • i = 授权标识 • g= 权限 • x = 可信上下文 • 空白 = 数据库
策略关联子对象类型	CHAR(1)	与审计策略关联的子对象的类型。如果对象类型是 ?（授权标识），那么可能的值包括: • U = 用户 • G = 组 • R = 角色
策略关联对象名	VARCHAR(128)	与审计策略关联的对象的名称。
策略关联对象模式	VARCHAR(128)	与审计策略关联的对象的模式名。如果“策略关联对象类型”标识模式不适用的对象，那么它将为 NULL。
审计状态	CHAR(1)	审计策略中 AUDIT 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
检查状态	CHAR(1)	审计策略中 CHECKING 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
上下文状态	CHAR(1)	审计策略中 CONTEXT 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功

表 42. AUDIT 事件的审计记录布局 (续)

名称	格式	描述
执行状态	CHAR(1)	审计策略中 EXECUTE 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
使用数据执行	CHAR(1)	审计策略中 EXECUTE 类别的 WITH DATA 选项。可能的值包括: • Y - 使用数据 • N - 不使用数据
Objmaint 状态	CHAR(1)	审计策略中 OBJMAINT 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
Secmaint 状态	CHAR(1)	审计策略中 SECMAINT 类别的状态。请参阅“审计状态”字段以了解可能的值。
Sysadmin 状态	CHAR(1)	审计策略中 SYSADMIN 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
Validate 状态	CHAR(1)	审计策略中 VALIDATE 类别的状态。可能的值包括: • B - 两者 • F - 失败 • N - 无 • S - 成功
错误类型	CHAR(8)	审计策略中的错误类型。可能的值包括: AUDIT 和 NORMAL。
数据路径	VARCHAR(1024)	**db2audit configure** 命令中所指定的活动审计日志的路径。
归档路径	VARCHAR(1024)	**db2audit configure** 命令中指定的已归档审计日志的路径

CHECKING 事件的审计记录布局

下表显示了 CHECKING 事件的审计记录格式。

样本审计记录:

```
timestamp=1998-06-24-08.42.11.622984;
category=CHECKING;
audit event=CHECKING_OBJECT;
event correlator=2;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
```

```
package schema=NULLID;
package name=SYSSH200;
package section=0;
object schema=GSTAGER;
object name=NONE;
object type=REOPT_VALUES;
access approval reason=DBADM;
access attempted=STORE;
```

表 43. CHECKING 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: CHECKING
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 CHECKING 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
对象模式	VARCHAR(128)	为其生成审计事件的对象的模式。
对象名	VARCHAR(128)	为其生成审计事件的对象的名称。
对象类型	VARCHAR(32)	为其生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。
访问批准原因	CHAR(34)	指示为此审计事件批准访问的原因。可能的值包括: 显示在标题为『可能的 CHECKING 访问批准原因的列表』的主题中的那些值。
尝试的访问	CHAR(34)	指定尝试的访问类型。可能的值包括: 显示在标题为『可能的 CHECKING 访问尝试类型的列表』的主题中的那些值。
程序包版本	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

表 43. CHECKING 事件的审计记录布局 (续)

名称	格式	描述
检查的授权标识	VARCHAR(128)	当授权标识与审计事件时的授权标识不同时，将检查授权标识。例如，它可以是 TRANSFER OWNERSHIP 语句中的目标所有者。 当审计事件为 SWITCH_USER 时，此字段表示切换至的授权标识。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(255)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。

CHECKING 访问批准原因

以下列表显示了可能的 CHECKING 访问批准原因。

请注意，审计记录可能包含多个访问批准原因，例如：`access approval reason=DATAACCESS,ACCESSCTRL;`。当存在多个访问批准原因时，用户必须具有已声明的所有权限和特权，才能通过对所尝试访问的授权检查。

0x00000000000000000000000000000001 ACCESS DENIED
未批准访问；确切地说，拒绝了访问。

0x00000000000000000000000000000002 SYSADM
已批准访问；该应用程序或用户具有 SYSADM 权限。

0x00000000000000000000000000000004 SYSCTRL
已批准访问；该应用程序或用户具有 SYSCTRL 权限。

0x00000000000000000000000000000008 SYSMAINT
已批准访问；该应用程序或用户具有 SYSMAINT 权限。

0x00000000000000000000000000000010 DBADM
已批准访问；该应用程序或用户具有 DBADM 权限。

0x00000000000000000000000000000020 DATABASE
已批准访问；该应用程序或用户具有使用该数据库的显式特权。

0x00000000000000000000000000000040 OBJECT
已批准访问；该应用程序或用户对该对象或功能具有特权。

0x00000000000000000000000000000080 DEFINER
已批准访问；该应用程序或用户是该对象或功能的定义者。

0x00000000000000000000000000000100 OWNER
已批准访问；该应用程序或用户是该对象或功能的所有者。

0x00000000000000000000000000000200 CONTROL
已批准访问；该应用程序或用户对该对象或功能具有 CONTROL 特权。

0x00000000000000000000000000000400 BIND
已批准访问；该应用程序或用户对该程序包具有绑定特权。

0x00000000000000000000000000000800 SYSQUIESCE
已批准访问；如果实例或数据库处于停顿方式，应用程序或用户可连接。

0x00000000000000000000000000001000 SYSMON
已批准访问；该应用程序或用户具有 SYSMON 权限。

0x00000000000000000000000000002000 SECADM
已批准访问；该应用程序或用户具有 SECADM 权限。

0x00000000000000000000000000004000 SETSESSIONUSER
已批准访问；该应用程序或用户具有 SETSESSIONUSER 权限。

0x00000000000000000000000000008000 TRUSTED_CONTEXT_MATCH
连接属性与在 DB2 服务器中定义的唯一可信上下文的属性匹配。

0x00000000000000000000000000010000 TRUSTED_CONTEXT_USE
已批准访问以便使用可信上下文。

0x00000000000000000000000000020000 SQLADM
已批准访问；该应用程序或用户具有 SQLADM 权限。

0x00000000000000000000000000040000 WLMADM
已批准访问；该应用程序或用户具有 WLMADM 权限。

0x00000000000000000000000000080000 EXPLAIN
已批准访问；该应用程序或用户具有 EXPLAIN 权限。

0x00000000000000000000000000100000 DATAACCESS
已批准访问；该应用程序或用户具有 DATAACCESS 权限。

0x00000000000000000000000000200000 ACCESSCTRL
已批准访问；该应用程序或用户具有 ACCESSSCTRL 权限。

CHECKING 访问尝试类型

以下列表显示了可能的 CHECKING 访问尝试类型。

如果审计事件是 CHECKING_TRANSFER，那么审计项将反映是否具有特权。

0x00000000000000000000000000000001 CONTROL
尝试验证是否具有 CONTROL 特权。

0x00000000000000000000000000000002 ALTER
尝试改变对象，或验证是否具有 ALTER 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000004 DELETE
尝试删除对象，或验证是否具有 DELETE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000008 INDEX
尝试使用索引，或验证是否具有 INDEX 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000010 INSERT
尝试插入到对象中，或验证是否具有 INSERT 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000020 SELECT
尝试查询表或视图，或验证是否具有 SELECT 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000040 UPDATE
尝试更新对象中的数据，或验证是否具有 UPDATE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000080 REFERENCE
尝试在对象间建立引用约束，或验证是否具有 REFERENCE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000000100 CREATE
尝试创建一个对象。

0x00000000000000000000000000000200 DROP
尝试删除一个对象。

0x00000000000000000000000000000400 CREATEIN
尝试在另一个模式内创建一个对象。

0x00000000000000000000000000000800 DROPIN
尝试删除在另一个模式内找到的对象。

0x00000000000000000000000000001000 ALTERIN
尝试改变或修改在另一个模式内找到的对象。

0x00000000000000000000000000002000 EXECUTE
尝试执行或运行应用程序或调用例程、创建源自例程的函数（仅适用于函数）或在任何 DDL 语句中引用例程，或验证是否具有 EXECUTE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000000004000 BIND
尝试绑定或准备一个应用程序。

0x00000000000000000000000000008000 SET_EVENT_MONITOR
尝试设置事件监视器开关。

0x00000000000000000000000000010000 SET_CONSTRAINTS
尝试设置对一个对象的约束。

0x00000000000000000000000000020000 COMMENT ON
尝试创建有关一个对象的注释。

0x00000000000000000000000000040000 GRANT
尝试将对一个对象的特权或角色授予给另一个授权标识。

0x00000000000000000000000000080000 REVOKE
尝试撤销对象授权标识的特权或角色。

0x00000000000000000000000000100000 LOCK
尝试锁定一个对象。

0x00000000000000000000000000200000 RENAME
尝试重命名一个对象。

0x00000000000000000000000000400000 CONNECT
尝试与一个数据库连接。

0x00000000000000000000000000800000 MEMBER_OF_SYS_GROUP
尝试访问或使用 SYS 组的成员。

0x00000000000000000000000001000000 ALL
尝试执行语句，对对象的所有必需特权被挂起（仅用于 DBADM/SYSADM）。

0x00000000000000000000000002000000 DROP ALL
尝试删除多个对象。

0x00000000000000000000000004000000 LOAD
尝试在表空间中装入表。

0x00000000000000000000000008000000 USE
尝试在表空间中创建表，或验证是否具有 USE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000000010000000 SET_SESSION_USER
尝试执行 SET SESSION_USER 语句。

0x00000000000000000000000020000000 FLUSH
尝试执行 FLUSH 语句。

0x00000000000000000000000040000000 STORE
尝试查看 EXPLAIN_PREDICATE 表中已重新优化的语句的值。

0x00000000000000000000000100000000 SET_OWNER
尝试设置绑定程序包时与当前用户不匹配的所有者。

0x00000000000000000000000200000000 SET_PASSTHRU
尝试发出 SET PASSTHRU 语句。

0x00000000000000000000000400000000 TRANSFER
尝试传送一个对象。

0x00000000000000000000000800000000 ALTER_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 ALTER 特权。

0x00000000000000000000001000000000 DELETE_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 DELETE 特权。

0x00000000000000000000002000000000 INDEX_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 INDEX 特权。

0x00000000000000000000004000000000 INSERT_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 INSERT 特权。

0x00000000000000000000008000000000 SELECT_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 SELECT 特权。

0x00000000000000000000010000000000 UPDATE_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 UPDATE 特权。

0x00000000000000000000020000000000 REFERENCE_WITH_GRANT
尝试验证是否具有带 GRANT 选项的 REFERENCE 特权。

0x00000000000000000000040000000000 USAGE
尝试使用序列或 XSR 对象，或验证是否具有 USAGE 特权（如果审计事件为 CHECKING_TRANSFER）。

0x00000000000000000000080000000000 SET ROLE
尝试设置角色。

0x00000000000000000000100000000000 EXPLICIT_TRUSTED_CONNECTION
尝试建立显式可信连接。

0x00000000000000000000200000000000 IMPLICIT_TRUSTED_CONNECTION
尝试建立隐式可信连接。

0x00000000000000000000400000000000 READ
尝试读取全局变量。

0x00000000000000000000800000000000 WRITE
尝试写入全局变量。

0x00000000000000000001000000000000 SWITCH_USER
尝试在显式可信连接上切换用户标识。

0x00000000000000000002000000000000 AUDIT_USING
尝试将审计策略与一个对象关联。

0x00000000000000000004000000000000 AUDIT_REPLACE
尝试替换与一个对象关联的审计策略。

0x00000000000000000008000000000000 AUDIT_REMOVE
尝试除去与一个对象关联的审计策略。

0x00000000000000000010000000000000 AUDIT_ARCHIVE
尝试归档审计日志。

0x00000000000000000020000000000000 AUDIT_EXTRACT
尝试抽取审计日志。

0x00000000000000000040000000000000 AUDIT_LIST_LOGS
尝试列示审计日志。

0x00000000000000000080000000000000 IGNORE_TRIGGERS
尝试忽略与数据库对象相关联的触发器。

0x00000000000000000100000000000000 PREPARE
尝试预编译 SQL 语句，并且用户不拥有必需的对象级别特权或 DATAACCESS 权限。

0x00000000000000000200000000000000 DESCRIBE
尝试描述语句，并且用户不拥有必需的对象级别特权或 DATAACCESS 权限。

0x00000000000000000400000000000000 SET_USAGELIST
尝试设置用法列表状态。

OBJMAINT 事件的审计记录布局

下表显示了 OBJMAINT 事件的审计记录格式。

样本审计记录:

```
timestamp=1998-06-24-08.42.41.957524;
category=OBJMAINT;
audit event=CREATE_OBJECT;
event correlator=3;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
package schema=NULLID;
package name=SQLC28A1;
package section=0;
object schema=BOSS;
object name=AUDIT;
object type=TABLE;
```

表 44. OBJMAINT 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: OBJMAINT
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 OBJMAINT 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(256)	审计事件发生时正在使用的程序包的名称。
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
对象模式	VARCHAR(128)	为其生成审计事件的对象的模式。
对象名	VARCHAR(128)	为其生成审计事件的对象的名称。

表 44. OBJMAINT 事件的审计记录布局 (续)

名称	格式	描述
对象类型	VARCHAR(32)	为其生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。
安全策略名称	VARCHAR(128)	安全策略的名称（如果对象类型是 TABLE 并且该表与安全策略相关联的话）。
改变操作	VARCHAR(32)	特定改变操作 可能的值包括: • ADD_PROTECTED_COLUMN • ADD_COLUMN_PROTECTION • DROP_COLUMN_PROTECTION • ADD_ROW_PROTECTION • ADD_SECURITY_POLICY • ADD_ELEMENT • ADD COMPONENT • USE GROUP AUTHORIZATIONS • IGNORE GROUP AUTHORIZATIONS • USE ROLE AUTHORIZATIONS • IGNORE ROLE AUTHORIZATIONS • OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL • RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
受保护的列名	VARCHAR(128)	如果改变操作是 ADD_COLUMN_PROTECTION 或 DROP_COLUMN_PROTECTION, 那么这是受影响的列名。
列安全标号	VARCHAR(128)	保护“列名”字段中指定的列的安全标号。
安全标号列名	VARCHAR(128)	包含保护行的安全标号的列名。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。
对象模块	VARCHAR(128)	对象所属的模块的名称。

SECMAINT 事件的审计记录布局

下表显示了 SECMAINT 事件的审计记录格式。

样本审计记录:

```
timestamp=1998-06-24-11.57.45.188101;
category=SECMAINT;
audit event=GRANT;
event correlator=4;
event status=0;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.boss.980624155728;
application name=db2bp;
package schema=NULLID;
package name=SQLC28A1;
package section=0;
object schema=BOSS;
object name=T1;
object type=TABLE;
grantor=BOSS;
grantee=WORKER;
grantee type=USER;
privilege=SELECT;
```

表 45. SECMAINT 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: SECMAINT
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 SECMAINT 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。

表 45. SECMAINT 事件的审计记录布局 (续)

名称	格式	描述
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
对象模式	VARCHAR(128)	为其生成审计事件的对象的模式。 如果对象类型字段为 ACCESS_RULE，那么此字段包含与规则关联的安全策略名。规则的名称存储在“对象名”字段中。 如果对象类型字段为 SECURITY_LABEL，那么此字段包含含有安全标号的安全策略的名称。安全标号的名称存储在“对象名”字段中。
对象名	VARCHAR(128)	为其生成审计事件的对象的名称。 当审计事件为下列任何一项时，表示角色名称: • ADD_DEFAULT_ROLE • DROP_DEFAULT_ROLE • ALTER_DEFAULT_ROLE • ADD_USER • DROP_USER • ALTER_USER_ADD_ROLE • ALTER_USER_DROP_ROLE • ALTER_USER_AUTHENTICATION 如果对象类型字段为 ACCESS_RULE，那么此字段包含规则的名称。与规则关联的安全策略名存储在“对象模式”字段中。 如果对象类型字段为 SECURITY_LABEL，那么此字段包含安全标号的名称。含有安全标号的安全策略的名称存储在“对象模式”字段中。
对象类型	VARCHAR(32)	为其生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。 当审计事件为下列任何一项时，值为 ROLE: • ADD_DEFAULT_ROLE • DROP_DEFAULT_ROLE • ALTER_DEFAULT_ROLE • ADD_USER • DROP_USER • ALTER_USER_ADD_ROLE • ALTER_USER_DROP_ROLE • ALTER_USER_AUTHENTICATION
授权者	VARCHAR(128)	特权或权限的授权者或撤销者的标识。

表 45. SECMAINT 事件的审计记录布局 (续)

名称	格式	描述
被授权者	VARCHAR(128)	被授予或撤销特权或权限的被授权者标识。 当审计事件为下列任何一项时，表示可信上下文对象: • ADD_DEFAULT_ROLE • DROP_DEFAULT_ROLE • ALTER_DEFAULT_ROLE • ADD_USER 或 DROP_USER • ALTER_USER_ADD_ROLE • ALTER_USER_DROP_ROLE • ALTER_USER_AUTHENTICATION
被授权者类型	VARCHAR(32)	被授予或撤销权限的被授权者的类型。当审计事件为下列任何一项时，可能的值包括 USER、GROUP、ROLE、AMBIGUOUS 和 TRUSTED_CONTEXT: • ADD_DEFAULT_ROLE • DROP_DEFAULT_ROLE • ALTER_DEFAULT_ROLE • ADD_USER • DROP_USER • ALTER_USER_ADD_ROLE • ALTER_USER_DROP_ROLE • ALTER_USER_AUTHENTICATION
特权或权限	CHAR(34)	指示授予或撤销的特权或权限的类型。可能的值包括: 显示在标题为『可能的 SECMAINT 特权或权限的列表』的主题中的那些值。 当审计事件为下列任何一项时，值为 ROLE MEMBERSHIP: • ADD_DEFAULT_ROLE 或 DROP_DEFAULT_ROLE • ALTER_DEFAULT_ROLE • ADD_USER • DROP_USER • ALTER_USER_ADD_ROLE • ALTER_USER_DROP_ROLE • ALTER_USER_AUTHENTICATION
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。

表 45. SECMAINT 事件的审计记录布局 (续)

名称	格式	描述
访问类型	VARCHAR(32)	授予对安全标号的访问类型。 可能的值为: • READ • WRITE • ALL 改变其安全策略的访问类型。可能的值为: • USE GROUP AUTHORIZATIONS • IGNORE GROUP AUTHORIZATIONS • USE ROLE AUTHORIZATIONS • IGNORE ROLE AUTHORIZATIONS • OVERRIDE NOT AUTHORIZED WRITE SECURITY LABEL • RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL
可采用的授权标识	VARCHAR(128)	当授予的特权为 SETSESSIONUSER 特权时，这是允许被授权者设置为会话用户的授权标识。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
授权者类型	VARCHAR(32)	授权者的类型。可能的值包括: USER。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文用户	VARCHAR(128)	标识当审计事件为 ADD_USER 或 DROP_USER 时的可信上下文用户。
可信上下文用户认证	INTEGER	指定当审计事件为 ADD_USER、DROP_USER 或 ALTER_USER_AUTHENTICATION 时的可信上下文用户的认证设置。 1 : 需要认证 0 : 不需要认证
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。

SECMAINT 特权或权限

以下列表显示了可能的 SECMAINT 特权或权限。

0x0000000000000000000000000000001 Control Table

授予的或撤销的对表或视图的控制特权。

0x00000000000000000000000000000002 ALTER
授予的或撤销的改变表或序列的特权。

0x00000000000000000000000000000004 ALTER with GRANT
对于允许授予特权的一个表或序列，授予的或撤销的改变该表或序列的特权。

0x00000000000000000000000000000008 DELETE TABLE
授予的或撤销的删除表或视图的特权。

0x00000000000000000000000000000010 DELETE TABLE with GRANT
对于允许授予特权的表，授予的或撤销的删除该表的特权。

0x00000000000000000000000000000020 Table Index
授予的或撤销的对索引的特权。

0x00000000000000000000000000000040 Table Index with GRANT
对于允许授予特权的索引，授予的或撤销的对该索引的特权。

0x00000000000000000000000000000080 Table INSERT
授予的或撤销的对表或视图进行插入的特权。

0x00000000000000000000000000000100 Table INSERT with GRANT
对于允许授予特权的表，授予的或撤销的对该表进行插入的特权。

0x00000000000000000000000000000200 Table SELECT
授予的或撤销的对表进行选择的特权。

0x00000000000000000000000000000400 Table SELECT with GRANT
对于允许授予特权的表，授予的或撤销的对该表进行选择的特权。

0x00000000000000000000000000000800 Table UPDATE
授予的或撤销的对表或视图进行更新的特权。

0x00000000000000000000000000001000 Table UPDATE with GRANT
对于允许授予特权的表或视图，授予的或撤销对该表或视图进行更新的特权。

0x00000000000000000000000000002000 Table REFERENCE
授予的或撤销的对表进行引用的特权。

0x00000000000000000000000000004000 Table REFERENCE with GRANT
对于允许授予特权的表，授予的或撤销的对该表进行引用的特权。

0x00000000000000000000000000020000 CREATEIN Schema
授予的或撤销的对模式的 CREATEIN 特权。

0x00000000000000000000000000040000 CREATEIN Schema with GRANT
对于允许授予特权的模式，授予的或撤销的对该模式的 CREATEIN 特权。

0x00000000000000000000000000080000 DROPIN Schema
授予的或撤销的对模式的 DROPIN 特权。

0x00000000000000000000000000100000 DROPIN Schema with GRANT
对于允许授予特权的模式，授予的或撤销的对该模式的 DROPIN 特权。

0x00000000000000000000000000200000 ALTERIN Schema
授予的或撤销的对模式的 ALTERIN 特权。

0x00000000000000000000000000400000 ALTERIN Schema with GRANT
对于允许授予特权的模式，授予的或撤销的对该模式的 ALTERIN 特权。

0x00000000000000000000000000800000 DBADM Authority
授予或撤销的 DBADM 权限。

0x00000000000000000000000001000000 CREATETAB Authority
授予或撤销的 Createtab 权限。

0x00000000000000000000000002000000 BINDADD Authority
授予或撤销的 Bindadd 权限。

0x00000000000000000000000004000000 CONNECT Authority
授予或撤销的 CONNECT 权限。

0x00000000000000000000000008000000 Create not fenced Authority
授予或撤销的 Create not fenced 权限。

0x00000000000000000000000010000000 Implicit Schema Authority
授予或撤销的 Implicit Schema 权限。

0x00000000000000000000000020000000 Server PASSTHRU
授予的或撤销的对此服务器（联合数据库数据源）使用传递（Pass-Through）工具的特权。

0x00000000000000000000000040000000 ESTABLISH TRUSTED CONNECTION
已创建可信连接。

0x00000000000000000000000100000000 Table Space USE
授予的或撤销的在表空间中创建表的特权。

0x00000000000000000000000200000000 Table Space USE with GRANT
授予的或撤销的在表空间中创建表并允许进行特权授权的特权。

0x00000000000000000000000400000000 Column UPDATE
授予的或撤销的对一个表的一个或多个特定列进行更新的特权。

0x00000000000000000000000800000000 Column UPDATE with GRANT
对于允许授予特权的表，授予的或撤销的对该表的一个或多个特定列进行更新的特权。

0x00000000000000000000001000000000 Column REFERENCE
授予的或撤销的对一个表的一个或多个特定列进行引用的特权。

0x00000000000000000000002000000000 Column REFERENCE with GRANT
对于允许授予特权的表，授予的或撤销的对该表的一个或多个特定列进行引用的特权。

0x00000000000000000000004000000000 LOAD Authority
授予或撤销的 LOAD 权限。

0x00000000000000000000008000000000 Package BIND
授予的或撤销的对程序包的 BIND 特权。

0x00000000000000000000010000000000 Package BIND with GRANT
对于允许授予特权的程序包，授予的或撤销的对该程序包的 BIND 特权。

0x00000000000000000000020000000000 EXECUTE
授予的或撤销的对程序包或例程的 EXECUTE 特权。

0x00000000000000000000040000000000 EXECUTE with GRANT
对于允许授予特权的程序包或例程，授予的或撤销的对该程序包或例程的 EXECUTE 特权。

0x00000000000000000000080000000000 EXECUTE IN SCHEMA
授予的或撤销的对模式中的所有例程的 EXECUTE 特权。

0x00000000000000000000100000000000 EXECUTE IN SCHEMA with GRANT
对于允许授予特权的模式中的所有例程，授予的或撤销的对这些例程的 EXECUTE 特权。

0x00000000000000000000200000000000 EXECUTE IN TYPE
授予的或撤销的对某个类型中的所有例程的 EXECUTE 特权。

0x00000000000000000000400000000000 EXECUTE IN TYPE with GRANT
对于允许授予特权的类型中的所有例程，授予的或撤销的对这些例程的 EXECUTE 特权。

0x00000000000000000000800000000000 CREATE EXTERNAL ROUTINE
授予或撤销的 CREATE EXTERNAL ROUTINE 特权。

0x00000000000000000001000000000000 QUIESCE_CONNECT
授予或撤销的 QUIESCE_CONNECT 特权。

0x00000000000000000004000000000000 SECADM Authority
授予或撤销的 SECADM 权限。

0x00000000000000000008000000000000 USAGE Authority
授予的或撤销的对序列的 USAGE 特权。

0x00000000000000000010000000000000 USAGE with GRANT Authority
对于允许授予特权的序列，授予的或撤销的对该序列的 USAGE 特权。

0x00000000000000000020000000000000 WITH ADMIN Option
授予的或撤销的对角色的 WITH ADMIN Option。

0x00000000000000000040000000000000 SETSESSIONUSER Privilege
授予或撤销的 SETSESSIONUSER。

0x00000000000000000080000000000000 Exemption
授予的或撤销的免除权。

0x00000000000000000100000000000000 Security label
授予的或撤销的安全标号。

0x00000000000000000200000000000000 WRITE with GRANT
对于允许授予特权的全局变量，授予的或撤销的写入该全局变量的特权。

0x00000000000000000400000000000000 Role Membership
授予的或撤销的角色成员资格。

0x00000000000000000800000000000000 Role Membership with ADMIN Option
授予的或撤销的带 ADMIN 选项的角色成员资格。

0x00000000000000001000000000000000 READ
授予的或撤销的读取全局变量的特权。

0x00000000000000002000000000000000 READ with GRANT
对于允许授予特权的全局变量，授予的或撤销的读取该全局变量的特权。

0x00000000000000004000000000000000 WRITE
授予的或撤销的写入全局变量的特权。

0x00000000000000001000000000000000 SQLADM
授予或撤销的 SQLADM 权限。

0x00000000000000002000000000000000 WLMADM
授予或撤销的 WLMADM 权限。

0x00000000000000004000000000000000 EXPLAIN
授予或撤销的 EXPLAIN 权限。

0x00000000000000008000000000000000 DATAACCESS
授予或撤销的 DATAACCESS 权限。

0x00000000000000010000000000000000 ACCESSCTRL
授予或撤销的 ACCESSCTRL 权限。

SYSADMIN 事件的审计记录布局

下表显示了 SYSADMIN 事件的审计记录布局。

样本审计记录:

```
timestamp=1998-06-24-11.54.04.129923;
category=SYSADMIN;
audit event=DB2AUDIT;
event correlator=1;
event status=0;
userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155404;
application name=db2audit;
```

表 46. SYSADMIN 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: SYSADMIN
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 SYSADMIN 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。

表 46. SYSADMIN 事件的审计记录布局 (续)

名称	格式	描述
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。

VALIDATE 事件的审计记录布局

下表显示了 VALIDATE 事件的审计记录格式。

样本审计记录:

```
timestamp=2007-05-07-10.30.51.585626;
category=VALIDATE;
audit event=AUTHENTICATION;
event correlator=1;
event status=0;
userid=newton;
authid=NEWTON;
execution id=gstager;
application id=*LOCAL.gstager.070507143051;
application name=db2bp;
auth type=SERVER;
plugin name=IBMOSauthserver;
```

表 47. VALIDATE 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: VALIDATE

表 47. VALIDATE 事件的审计记录布局 (续)

名称	格式	描述
审计事件	VARCHAR(32)	特定的审计事件。 可能的值包括: GET_GROUPS、GET_USERID、AUTHENTICATE_PASSWORD、VALIDATE_USER、AUTHENTICATION 和 GET_USERMAPPING_FROM_PLUGIN。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。
执行标识	VARCHAR(1024)	审计事件发生时正在使用的执行标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
认证类型	VARCHAR(32)	审计事件发生时的认证类型。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。
插件名称	VARCHAR(32)	审计事件发生时正在使用的插件的名称。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信上下文继承的角色的名称。

CONTEXT 事件的审计记录布局

下表显示了 CONTEXT 事件的审计记录布局。

样本审计记录:

```
timestamp=1998-06-24-08.42.41.476840;
category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;
event correlator=3;
database=FOO;
userid=boss;
authid=BOSS;
application id=*LOCAL.newton.980624124210;
application name=testapp;
package schema=NULLID;
package name=SQLC28A1;
package section=203;
text=create table audit(c1 char(10), c2 integer);
```

表 48. CONTEXT 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: CONTEXT
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 CONTEXT 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。 当审计事件为 SWITCH_USER 时，此字段表示切换至的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的授权标识。 当审计事件为 SWITCH_USER 时，此字段表示切换至的授权标识。
原始用户标识	VARCHAR(1024)	审计事件发生时 CLIENT_ORIGUSERID 全局变量的值。
原始节点号	SMALLINT	审计事件发生时所在的成员号。
协调程序节点号	SMALLINT	协调程序成员的成员号。
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
软件包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节号	SMALLINT	审计事件发生时正在使用的程序包中的节号。
语句文本	CLOB(8M)	SQL 或 XQuery 语句的文本（如果适用）。如果 SQL 或 XQuery 语句文本不可用，那么为 NULL。

表 48. CONTEXT 事件的审计记录布局 (续)

名称	格式	描述
程序包版本	VARCHAR(64)	审计事件发生时正在使用的程序包的版本。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_CONNECTION EXPLICIT_TRUSTED_CONNECTION
继承的角色	VARCHAR(128)	通过可信连接继承的角色。

EXECUTE 事件的审计记录布局

下表描述了作为 EXECUTE 类别的一部分审计的所有字段。

样本审计记录:

注: 与其他审计类别不同，在以表格式查看审计日志时，EXECUTE 类别可能会显示多个行描述一个事件。第一条记录描述主要事件，并且其事件列包含关键字 STATEMENT。其余行描述参数标记或主变量，每个参数占用一行，并且它们的事件列包含关键字 DATA。以报告格式查看审计日志时，有一条记录，但它的语句值具有多个条目。DATA 关键字只出现在表格式中。

```
timestamp=2006-04-10-13.20.51.029203;
  category=EXECUTE;
  audit event=STATEMENT;
  event correlator=1;
  event status=0;
  database=SAMPLE;
  userid=smith;
  authid=SMITH;
  session authid=SMITH;
  application id=*LOCAL.prodrig.060410172044;
  application name=myapp;
  package schema=NULLID;
  package name=SQLC2F0A;
  package section=201;
  uow id=2;
  activity id=3;
  statement invocation id=0;
  statement nesting level=0;
  statement text=SELECT * FROM DEPARTMENT WHERE DEPTNO = ? AND DEPTNAME = ?;
  statement isolation level=CS;
  compilation environment=
    isolation level=CS
    query optimization=5
    min_dec_div_3=NO
```

```
  degree=1
  sqlrules=DB2
  refresh age=+00000000000000.000000
  schema=SMITH
  maintained table type=SYSTEM
  resolution timestamp=2006-04-10-13.20.51.000000
  federated asynchrony=0;
value index=0;
value type=CHAR;
value data=C01;
value index=1;
value type=VARCHAR;
value extended indicator=-1;
value index=INFORMATION CENTER;  local_start_time=2006-04-10-13.20.51.021507
```

表 49. EXECUTE 事件的审计记录布局

名称	格式	描述
时间戳记	CHAR(26)	审计事件的日期和时间。
类别	CHAR(8)	审计事件的类别。可能的值包括: EXECUTE。
审计事件	VARCHAR(32)	特定的审计事件。 有关可能值的列表，请参阅第 305 页的『审计事件』中 EXECUTE 类别的部分。
事件相关因子	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件关联。
事件状态	INTEGER	审计事件的状态，由 SQLCODE 表示，其中成功的事件 > = 0，失败的事件 < 0。
数据库名称	CHAR(8)	为其生成了事件的数据库的名称。如果它是实例级别审计事件，那么为空白。
用户标识	VARCHAR(1024)	审计事件发生时的用户标识。
授权标识	VARCHAR(128)	审计事件发生时的语句授权标识。
会话授权标识	VARCHAR(128)	审计事件发生时的会话授权标识。
原始节点号	SMALLINT	审计事件发生时所在的成员号
协调程序节点号	SMALLINT	协调程序成员的成员号
应用程序标识	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
应用程序名称	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
客户机用户标识	VARCHAR(255)	审计事件发生时 CURRENT CLIENT USERID 专用寄存器的值。

表 49. EXECUTE 事件的审计记录布局 (续)

名称	格式	描述
客户机记帐字符串	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_ACCTNG 专用寄存器的值。
客户机工作站名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_WRKSTNNAME 专用寄存器的值。
客户机应用程序名称	VARCHAR(255)	审计事件发生时 CURRENT CLIENT_APPLNAME 专用寄存器的值。
可信上下文名称	VARCHAR(128)	与可信连接关联的可信上下文的名称。
连接信任类型	INTEGER	可能的值包括: IMPLICIT_TRUSTED_ CONNECTION 和 EXPLICIT_TRUSTED_ CONNECTION。
继承的角色	VARCHAR(128)	通过可信连接继承的角色。
程序包模式	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
程序包名称	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
程序包节	SMALLINT	审计事件发生时正在使用的程序包中的节号。
程序包版本	VARCHAR(164)	审计事件发生时正在使用的程序包的版本。
本地事务标识	VARCHAR(10) FOR BIT DATA	审计事件发生时正在使用的本地事务标识。这是作为事务日志一部分的 SQLU_TID 结构。
全局事务标识	VARCHAR(30) FOR BIT DATA	审计事件发生时正在使用的全局事务标识。这是作为事务日志一部分的 SQLP_GXID 结构中的数据字段。
UOW 标识	BIGINT	产生活动的工作单元标识。此值在每个工作单元的应用程序标识内是唯一的。
活动标识	BIGINT	工作单元内的唯一活动标识。
语句调用标识	BIGINT	一个标识，它可以区分工作单元中相同嵌套级别上的某次例程调用和其他例程调用。对于特定嵌套级别，它在工作单元中是唯一的。
语句嵌套级别	BIGINT	运行语句时有效的嵌套或递归级别；每个嵌套级别对应一个存储过程或用户定义的函数 (UDF) 的嵌套或递归调用。

表 49. EXECUTE 事件的审计记录布局 (续)

名称	格式	描述
活动类型	VARCHAR(32)	活动的类型。 可能的值包括: • READ_DML • WRITE_DML • DDL • CALL • NONE
语句文本	CLOB(8M)	SQL 或 XQuery 语句的文本（如果适用）。
语句隔离级别	CHAR(8)	运行语句时对该语句有效的隔离值。 可能的值包括: • NONE（未指定隔离级别） • UR（未落实的读） • CS（游标稳定性） • RS（读稳定性） • RR（可重复读）
编译环境描述	BLOB(8K)	编译 SQL 语句时使用的编译环境。可提供此元素作为 COMPILATION_ENV 表函数的输入，或者作为 SET COMPILATION ENVIRONMENT SQL 语句的输入。
修改的行数	INTEGER	包含由于下列操作而删除、插入或更新的总行数: • 在删除操作成功后强制执行约束 • 处理通过激活的直接插入触发器触发的 SQL 语句 如果调用了复合 SQL 语句，那么它包含所有子语句的这种行数的总和。在某些情况下，在遇到错误时，此字段包含一个负数值，它是内部错误指针。此值相当于 SQLCA 的 sqlerrd(5) 字段。
返回的行数	BIGINT	包含语句返回的总行数。

表 49. EXECUTE 事件的审计记录布局 (续)

名称	格式	描述
保存点标识	BIGINT	运行语句时对该语句有效的保存点标识。如果审计事件为 SAVEPOINT、RELEASE_SAVEPOINT 或 ROLLBACK_SAVEPOINT，那么保存点标识是正在设置、释放或回滚到的保存点。
语句值索引	INTEGER	SQL 语句中使用的输入参数标记或主变量的位置。
语句值类型	CHAR(16)	与 SQL 语句关联的数据值类型的字符串表示。可能的值示例有 INTEGER 或 CHAR。
语句值数据	CLOB(128K)	SQL 语句的数据值的字符串表示。LOB、LONG、XML 和结构化类型参数不存在。日期、时间和时间戳记字段记录为 ISO 格式。
语句值扩展指示符	INTEGER	对此语句值指定的扩展指示符的值。可能的值为: • 0 如果指定的语句值与指示符值分配的值相同， • -1 如果指示符值指定了 NULL 值， • -5 如果指示符值指定了 DEFAULT 值， • -7 如果指示符值指定了 UNASSIGNED 值。
本地开始时间	CHAR(26)	此活动在分区上开始工作的时间。当该活动不需要程序包时，此字段可以是空字符串，例如，对于 CONNECT、CONNECT RESET、COMMIT 和 ROLLBACK 时。值将以本地时间记录。

审计事件

对于每个审计类别，某些类型的事件可创建审计记录。

AUDIT 类别的事件

- ALTER_AUDIT_POLICY
- ARCHIVE
- AUDIT_REMOVE
- AUDIT_REPLACE

- AUDIT_USING
- CONFIGURE
- CREATE_AUDIT_POLICY
- DB2AUD
- DROP_AUDIT_POLICY
- EXTRACT
- FLUSH
- LIST_LOGS
- PRUNE（在 V9.5 和更高版本中不生成此值）。
- START
- STOP
- UPDATE_DBM_CFG

CHECKING 类别的事件

- CHECKING_FUNCTION
- CHECKING_MEMBERSHIP_IN_ROLES
- CHECKING_OBJECT
- CHECKING_TRANSFER

CONTEXT 类别的事件

- ADD_NODE
- ATTACH
- BACKUP_DB
- BIND
- CLOSE_CONTAINER_QUERY
- CLOSE_CURSOR
- CLOSE_HISTORY_FILE
- CLOSE_TABLESPACE_QUERY
- COMMIT
- CONNECT
- CONNECT_RESET
- CREATE_DATABASE
- DARI_START
- DARI_STOP
- DBM_CFG_OPERATION
- DESCRIBE
- DESCRIBE_DATABASE
- DETACH
- DISCOVER
- DROP_DATABASE
- ENABLE_MULTIPAGE

- ESTIMATE_SNAPSHOT_SIZE
- EXECUTE
- EXECUTE_IMMEDIATE
- EXTERNAL_CANCEL
- FETCH_CONTAINER_QUERY
- FETCH_CURSOR
- FETCH_HISTORY_FILE
- FETCH_TABLESPACE
- FORCE_APPLICATION
- GET_DB_CFG
- GET_DFLT_CFG
- GET_SNAPSHOT
- GET_TABLESPACE_STATISTIC
- IMPLICIT_REBIND
- LOAD_MSG_FILE
- LOAD_TABLE
- OPEN_CONTAINER_QUERY
- OPEN_CURSOR
- OPEN_HISTORY_FILE
- OPEN_TABLESPACE_QUERY
- PREPARE
- PRUNE_RECOVERY_HISTORY
- QUIESCE_TABLESPACE
- READ_ASYNC_LOG_RECORD
- REBIND
- REDISTRIBUTE
- REORG
- REQUEST_ROLLBACK
- RESET_DB_CFG
- RESET_MONITOR
- RESTORE_DB
- ROLLBACK
- ROLLFORWARD_DB
- RUNSTATS
- SET_APPL_PRIORITY
- SET_MONITOR
- SET_RUNTIME_DEGREE
- SET_TABLESPACE_CONTAINERS
- SINGLE_TABLESPACE_QUERY
- SWITCH_USER

- UNLOAD_TABLE
- UNQUIESCE_TABLESPACE
- UPDATE_AUDIT
- UPDATE_DBM_CFG
- UPDATE_RECOVERY_HISTORY

EXECUTE 类别的事件

- COMMIT 执行 COMMIT 语句
- CONNECT 建立数据库连接
- CONNECT RESET 终止数据库连接
- DATA 语句的主变量或参数标记数据值

  此事件将对语句中包括的每个主变量或参数标记重复。它只出现在定界抽取的审计日志中。
- GLOBAL COMMIT 在全局事务内执行落实
- GLOBAL ROLLBACK 在全局事务内执行回滚
- RELEASE SAVEPOINT 执行 RELEASE SAVEPOINT 语句
- ROLLBACK 执行 ROLLBACK 语句
- SAVEPOINT 执行 SAVEPOINT 语句
- STATEMENT 执行 SQL 语句
- SWITCH USER 在可信连接内切换用户

OBJMAINT 类别的事件

- ALTER_OBJECT（生成于改变受保护的表时和改变模块时）
- CREATE_OBJECT
- DROP_OBJECT
- RENAME_OBJECT

SECMAINT 类别的事件

- ADD_DEFAULT_ROLE
- ADD_USER
- ALTER_DEFAULT_ROLE
- ALTER SECURITY POLICY
- ALTER_USER_ADD_ROLE
- ALTER_USER_AUTHENTICATION
- ALTER_USER_DROP_ROLE
- DROP_DEFAULT_ROLE
- DROP_USER
- GRANT
- IMPLICIT_GRANT
- IMPLICIT_REVOKE
- REVOKE

- SET_SESSION_USER
- TRANSFER_OWNERSHIP
- UPDATE_DBM_CFG

SYSADMIN 类别的事件

- ACTIVATE_DB
- ADD_NODE
- ALTER_BUFFERPOOL
- ALTER_DATABASE
- ALTER_NODEGROUP
- ALTER_TABLESPACE
- ATTACH_DEBUGGER
- BACKUP_DB
- CATALOG_DB
- CATALOG_DCS_DB
- CATALOG_NODE
- CHANGE_DB_COMMENT
- CLOSE_CONTAINER_QUERY
- CLOSE_TABLESPACE_QUERY
- CREATE_BUFFERPOOL
- CREATE_DATABASE
- CREATE_DB_AT_NODE
- CREATE_EVENT_MONITOR
- CREATE_INSTANCE
- CREATE_NODEGROUP
- CREATE_TABLESPACE
- DB2AUD
- DB2AUDIT
- DB2REMOT
- DB2SET
- DB2TRC
- DEACTIVATE_DB
- DELETE_INSTANCE
- DESCRIBE_DATABASE
- DROP_BUFFERPOOL
- DROP_DATABASE
- DROP_EVENT_MONITOR
- DROP_NODEGROUP
- DROP_NODE_VERIFY
- DROP_TABLESPACE

- ENABLE_MULTIPAGE
- ESTIMATE_SNAPSHOT_SIZE
- FETCH_CONTAINER_QUERY
- FETCH_TABLESPACE
- FORCE_APPLICATION
- GET_SNAPSHOT
- GET_TABLESPACE_STATISTIC
- GRANT_DBADM（V97: 不再生成）
- GRANT_DB_AUTH（V97: 不再生成）
- KILLDBM
- LIST_DRDA_INDOUBT_TRANSACTIONS
- LOAD_TABLE
- MERGE_DBM_CONFIG_FILE
- MIGRATE_DB
- MIGRATE_DB_DIR
- MIGRATE_SYSTEM_DIRECTORY
- OPEN_CONTAINER_QUERY
- OPEN_TABLESPACE_QUERY
- PRUNE_RECOVERY_HISTORY
- QUIESCE_TABLESPACE
- READ_ASYNC_LOG_RECORD
- REDISTRIBUTE_NODEGROUP
- RENAME_TABLESPACE
- RESET_ADMIN_CFG
- RESET_DBM_CFG
- RESET_DB_CFG
- RESET_MONITOR
- RESTORE_DB
- REVOKE_DBADM（V97: 不再生成）
- REVOKE_DB_AUTH（V97: 不再生成）
- ROLLFORWARD_DB
- SET_APPL_PRIORITY
- SET_EVENT_MONITOR_STATE
- SET_RUNTIME_DEGREE
- SET_TABLESPACE_CONTAINERS
- SINGLE_TABLESPACE_QUERY
- START_DB2
- STOP_DB2
- UNCATALOG_DB
- UNCATALOG_DCS_DB

- UNCATALOG_NODE
- UNLOAD_TABLE
- UPDATE_ADMIN_CFG
- UPDATE_CLI_CONFIGURATION
- UPDATE_DB_VERSION
- UPDATE_DBM_CFG
- UPDATE_DB_CFG
- SET_MONITOR
- UPDATE_RECOVERY_HISTORY

VALIDATE 类别的事件

- AUTHENTICATE
- CHECK_GROUP_MEMBERSHIP（在 V9.5 和更高版本中不生成）
- GET_USERMAPPING_FROM_PLUGIN
- GET_GROUPS（在 V9.5 和更高版本中不生成）
- GET_USERID（在 V9.5 和更高版本中不生成）

第 12 章 使用操作系统安全性

操作系统提供安全性功能部件，您可以使用它们来支持数据库安装的安全性。

DB2 和 Windows 安全性

Windows 域是通过特定且唯一的名称引用的客户机和服务器的组合；且共享称为“安全访问管理器”（SAM）的单个用户帐户数据库。域中的其中一台计算机是域控制器。域控制器管理用户域交互作用的各个方面。

域控制器使用域用户帐户数据库中的信息来认证登录域帐户的用户。对于每个域，都有一个域控制器作为主域控制器（PDC）。在域中，还可以有备份域控制器（BDC），它在没有主域控制器或主域控制器不可用时认证用户帐户。备份域控制器拥有 Windows Security Account Manager（SAM）数据库的副本，会针对 PDC 上的主副本定期同步。

只需要在主域控制器中定义用户帐户、用户标识和密码就可以访问域资源。

注: CONNECT 语句和 ATTACH 命令支持由两部分组成的用户标识。与 SAM 兼容的用户标识的限定词是一个样式为“Domain\User”的名称，其最大长度为 15 个字符。

对于安装 Windows 服务器时的设置过程，可选择创建下列对象:

- 在新域中创建主域控制器
- 在已知域中创建备份域控制器
- 在已知域中创建独立服务器

在新域中选择“控制器”会使该服务器成为主域控制器。

用户可能要登录本地机器，或者当在“Windows 域”中安装机器时，用户可能要登录该域。要认证用户，DB2 首先检查本地机器，然后检查当前域的“域控制器”，最后检查“域控制器”认识的任何一个“可信域”。

为举例说明这是如何进行的，假设 DB2 实例需要服务器认证。该配置如下所示:

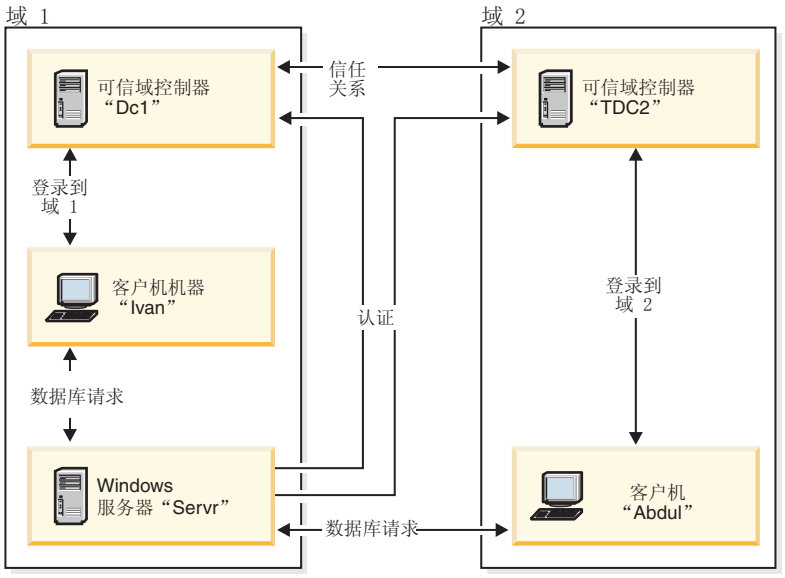


图 7. 使用 Windows 域的认证

每台机器都有一个安全性数据库，即“安全性访问管理”（SAM）。DC1 是域控制器，其中登记了客户机 Ivan 和 DB2 服务器 Servr。TDC2 是 DC1 的一个可信域，客户机 Abdul 是 TDC2 的域的一个成员。

认证方案

具有服务器认证的方案（Windows）

以下示例演示了服务器对用户的认证。

1. Abdul 登录 TDC2 域（即，它在 TDC2 SAM 数据库中是可识别的）。
2. Abdul 然后与一个 DB2 数据库连接，该数据库将被编目为位于 SRV3 上:

   ```
   db2 connect to remotedb user Abdul using fredpw
   ```

3. SRV3 确定可识别 Abdul 的位置。用于查找此信息的 API 首先搜索本地机器（SRV3），然后搜索域控制器（DC1），最后尝试搜索任何可信域。在 TDC2 上找到用户名 Abdul。此搜索顺序需要用户和组的单个名称空间。
4. 然后 SRV3:
   a. 在 TDC2 上验证用户名和密码。
   b. 通过询问 TDC2 来了解 Abdul 是否是管理员。
   c. 通过询问 TDC2 来列举所有 Abdul 的组。

具有客户机认证和 Windows 客户机的方案

以下示例演示了客户机对用户的认证。

1. 管理员 Dale 登录 SRV3，并将数据库实例的认证更改为“客户机”:

   ```
   db2 update dbm cfg using authentication client
   db2stop
   db2start
   ```

2. Ivan 在 Windows 客户机上登录 DC1 域（即，他在 DC1 SAM 数据库中是可识别的）。
3. Ivan 然后与一个 DB2 数据库连接，该数据库将被编目为位于 SRV3 上:

```
DB2 CONNECT to remotedb user Ivan using johnpw
```

4. Ivan 的机器验证用户名和密码。用于查找此信息的 API 首先搜索本地机器（Ivan），然后搜索域控制器（DC1），最后尝试搜索任何可信域。在 DC1 上找到用户名 Ivan。
5. 然后，Ivan 的机器用 DC1 来验证用户名和密码。
6. 然后 SRV3:
   a. 确定可识别 Ivan 的位置。
   b. 通过询问 DC1 来了解 Ivan 是否是管理员。
   c. 通过询问 DC1 来列举所有 Ivan 的组。

注: 在尝试连接至 DB2 数据库之前，确保已启动“DB2 安全服务”。“安全服务”是作为 Windows 安装的一部分安装的。然后，将安装 DB2 并将它“注册”为 Windows 服务，但是它不会自动启动。要启动“DB2 安全服务”，请输入 `NET START DB2NTSECSERVER` 命令。

对全局组的支持（Windows）

DB2 数据库系统支持全局组。

要使用全局组，必须将全局组包括在一个本地组中。当 DB2 数据库管理器枚举某个人所属的所有组时，它也列示用户间接所属的本地组（由于在一个全局组中，而该全局组本身是一个或多个本地组的成员）。

可在两种可能的方案中使用全局组:

- 包括在本地组中。必须对此本地组授予许可权。
- 包括在域控制器上。必须对此全局组授予许可权。

Windows 上的 DB2 用户认证和组信息

用户名和组名限制（Windows）

有一些特定于 Windows 环境的局限性。应知道常规 DB2 对象命名规则也适用。

- Windows 下的用户名不区分大小写；但是，密码区分大小写。
- 用户名和组名可以是大写字符与小写字符的组合。但是，当在 DB2 数据库中使用时，它们通常被转换为大写字符。例如，如果连接数据库并创建表 schema1.table1，那么此表作为 SCHEMA1.TABLE1 存储在数据库中。（如果要使用小写对象名，那么从命令行处理器发出命令并将对象名括在引号中，或者使用第三方 ODBC 前端工具。）
- DB2 数据库管理器支持单个名称空间。即，在可信域环境中运行时，相同名称的用户帐户不应在多个域中存在或在服务器的本地 SAM 和另一域中存在。
- 用户名不应该与组名相同。
- 本地组不应该与域级别组具有相同名称。

Windows 上的组和用户认证

在 Windows 上，通过使用称为“用户管理器”的 Windows 管理工具创建用户帐户来定义用户。包含其他帐户（又称为成员）的帐户是一个组。

组允许 Windows 管理员同时将权限和许可权授予组内的各个用户而不必分别维护每个用户。与用户帐户一样，组是在“安全性访问管理器”（SAM）数据库中定义并维护的。

有两种类型的组:

- 本地组。本地组可以包括在本地帐户数据库中创建的用户帐户。如果本地组在域中的某台机器上，那么本地组还可以包含 Windows 域中的域帐户和组。如果本地组是在工作站上创建的，那么它是特定于该工作站的。
- 全局组。全局组只存在于域控制器上，且包含域的 SAM 数据库中的用户帐户。即，全局组只包含在其上创建它的域中的用户帐户；它不能包含任何其他组作为成员。可在全局组自己的域中的服务器和工作站以及可信域中使用全局组。

Windows 上的域之间的信任关系

信任关系是两个域之间的管理和通信链路。两个域之间的信任关系允许在定义用户帐户的域之外的域中使用这些帐户和全局组。

共享帐户信息以验证可信域中未经认证的用户帐户和全局组的权限和许可权。信任关系通过将两个或多个域组合成单个管理单元来简化用户管理。

信任关系中有两个域:

- 信赖域。此域信赖另一个域以认证它们的用户。
- 可信域。此域代表（信赖）另一个域认证用户。

信任关系是不可传递的。这表示需要在两个域之间在每个方向上建立显式信任关系。例如，信赖域可能不一定是可信域。

使用组和域安全性的认证 (Windows)

授予特权或定义权限级别时，DB2 数据库系统允许您指定本地组或全局组。

关于此任务

如果用户的帐户是在本地或全局组中显式定义的，或者是作为定义为本地组成员的全局组成员隐式定义的，那么确定用户是组的成员。

DB2 数据库管理器支持下列类型的组:

- 本地组
- 全局组
- 作为本地组成员的全局组

  DB2 数据库管理器使用用户所在的安全性数据库来枚举该用户所属的本地组和全局组。DB2 数据库系统提供了一种覆盖，无论用户帐户的位置如何，都强制在安装了 DB2 数据库的本地 Windows 服务器上枚举组。可以使用下列命令来归档此覆盖:

  – 对于全局设置:

  ```
  db2set -g DB2_GRP_LOOKUP=local
  ```

  – 对于实例设置:

  ```
  db2set -i instance_name DB2_GRP_LOOKUP=local
  ```

发出此命令之后，必须停止 DB2 数据库实例，然后启动它才能使更改生效。然后，创建本地组并将域帐户或全局组包括在本地组中。

要查看设置的所有 DB2 概要文件注册表变量，输入

```
db2set -all
```

如果 **DB2_GRP_LOOKUP** 概要文件注册表变量设置为 local，那么 DB2 数据库管理器只尝试枚举用户在本地机器上的组。如果未将该用户定义为本地组的成员或嵌套在本地组中的全局组的成员，那么组枚举操作会失败。DB2 数据库管理器**不**会尝试在该域中另一机器上或在域控制器上枚举该用户的组。

如果 DB2 数据库管理器在作为资源域中主域控制器或备份域控制器的机器上运行，那么它能够找到任何可信域中的任何域控制器。这样的原因是：可信域中的备份域控制器的域的名称仅在域控制器上才能被识别。

使用访问令牌来获取用户的组信息（Windows）

访问令牌是描述进程或线程的安全上下文的对象。访问令牌中的信息包括与过程或线程关联的用户帐户的标识和特权。

登录时，系统通过比较密码与安全数据库中存储的信息来验证密码。如果密码得到认证，那么系统就会生成访问令牌。您运行的每个进程均使用此访问令牌的副本。

也可根据高速缓存凭证获取访问令牌。通过系统认证之后，操作系统就会高速缓存您的凭证。当不能连接域控制器时，可以在高速缓存中引用上一次登录的访问令牌。

访问令牌包括所属全部组的有关信息：本地组和各种域组（全局组、域本地组和通用组）。

注：使用远程连接时，即使启用了访问令牌支持，使用客户机认证的组查询也不受支持。

要启用访问令牌支持，必须使用 **db2set** 命令来更新 **DB2_GRP_LOOKUP** 注册表变量。**DB2_GRP_LOOKUP** 最多可以具有两个参数，它们以逗号分隔：

- 第一个参数用于传统的组查询，可以采用下列值：“ ”、“LOCAL”或“DOMAIN”。
- 第二个参数用于令牌样式组查询，可以采用下列值：“TOKEN”、“TOKENDOMAIN”或“TOKENLOCAL”。

如果指定了第二个参数（TOKEN、TOKENDOMAIN 或 TOKENLOCAL），那么它优先于传统的组枚举。如果令牌组枚举失败，那么在指定了 **DB2_GRP_LOOKUP** 的第一个参数的情况下，会进行传统的组查询。

值 TOKEN、TOKENDOMAIN 和 TOKENLOCAL 的含义如下所示：

- TOKENLOCAL

  令牌用来在本地机器上枚举组（这相当于传统的“LOCAL”组查询）。

- TOKENDOMAIN

  令牌用来在定义用户的位置（对于本地用户，为本地机器，对于域用户，则为域）枚举组。这相当于传统的“ ”或“DOMAIN”组查询。

- TOKEN

令牌用来在域和本地机器上枚举组。对于本地用户，返回的组将包含本地组。对于域用户，返回的组将包含域和本地组。TOKEN 参数的值不等于组查询参数值: “”、“LOCAL”或“DOMAIN”。

例如，**DB2_GRP_LOOKUP** 的以下设置启用访问令牌支持以便枚举本地组:

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

下一示例启用访问令牌支持，以便在本地机器以及定义用户标识的位置（如果帐户是在域中定义的）枚举组:

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

最后这个示例启用访问令牌支持，以便在定义用户标识的位置枚举域组:

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

注: 可以所有认证类型（CLIENT 认证除外）启用访问令牌支持。

DB2_GRP_LOOKUP 环境变量和 DB2 组枚举 (Windows)

在 Windows 上，用户可以属于域级别上定义的组和/或本地机器上定义的组。

DB2_GRP_LOOKUP 环境变量控制是在本地机器上还是在定义用户的位置（对于本地用户，为本地机器；对于域用户，则为域级别）枚举组。因此，当安全性管理员授予权限和特权时，必须注意按预期设置 **DB2_GRP_LOOKUP**，确保正确用户接收到预期的授权。

如果未设置 **DB2_GRP_LOOKUP** 概要文件注册表变量，那么:

1. DB2 数据库系统首先尝试在同一机器上查找用户。
2. 如果用户名是以本地方式定义的，那么以本地方式认证该用户。
3. 如果在本地找不到该用户，那么 DB2 数据库系统会尝试在它的域上查找该用户名，然后在可信域上进行查找。

例如，请考虑以下未设置 **DB2_GRP_LOOKUP** 的情况:

1. 域用户 DUSER1 属于本地组 GROUP1。
2. 安全性管理员（拥有 SECADM 权限）将 DBADM 权限授予组 GROUP1。

   ```
   GRANT DBADM ON database TO GROUP GROUP1
   ```

3. 因为未设置 **DB2_GRP_LOOKUP**，所以会在定义用户的位置枚举组。因此，会在域级别上枚举 DUSER1 的组。因为在域级别上 DUSER1 不属于组 GROUP1，所以 DUSER1 不会接收到 DBADM 权限。

此外，请考虑以下更复杂的情况，其中涉及 **UPGRADE DATABASE** 命令并且未设置 **DB2_GRP_LOOKUP**:

1. 域用户 DUSER2 属于本地 Administrators 组。
2. 因为未设置 **sysadm_group** 配置参数，所以本地 Administrators 组的成员会自动拥有 SYSADM 权限。
3. 用户 DUSER2 能够发出 **UPGRADE DATABASE** 命令（因为 DUSER2 拥有 SYSADM 权限）。**UPGRADE DATABASE** 命令将对要升级的数据库的 DBADM 权限授予 SYSADM 组，在此情况下，该组为 Administrators 组。
4. 因为未设置 **DB2_GRP_LOOKUP**，所以会在定义用户的位置枚举组。因此，会在域级别上枚举 DUSER2 的组。因为在域级别上 DUSER2 不属于 Administrators 组，所以 DUSER2 不会接收到 DBADM 权限。

对于这种情况，可能的解决方案是进行下列其中一项更改:

- 设置 **DB2_GRP_LOOKUP** = local
- 在域控制器中将应该具有 DBADM 权限的用户添加至 Administrators 或 GROUP1 组。

可以使用 SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID 表函数来验证用户拥有的权限，如针对 DUSER1 的以下示例中所示:

```
SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE
   FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('DUSER1', 'U') ) AS T
   ORDER BY AUTHORITY
```

可以使用 SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID 表函数来验证 DB2 数据库管理器已确定用户所属的组，如针对 DUSER1 的以下示例中所示:

```
SELECT * FROM TABLE (SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID ('DUSER1')) AS T
```

注: 如果在域级别和本地机器上使用相同组名，那么因为 DB2 数据库管理器没有完全限定这些组，所以这可导致混乱。

使用已排序的域列表来进行认证

可信域林中可多次定义用户标识。可信域林是通过网络相关的域集合。

关于此任务

某一域上的用户拥有的用户标识可能与不同域上另一用户的用户标识相同。在尝试执行以下操作时，这可能造成困难:

- 认证拥有相同用户标识但位于不同域中的多个用户。
- 组查找，以便根据组授予和撤销特权。
- 验证密码。
- 控制网络流量。

为了防止域林中因多个用户可能具有相同用户标识而引起麻烦，应使用通过 **db2set** 和注册表变量 **DB2DOMAINLIST** 定义的有序域列表。设置顺序时，用逗号隔开列表中将包含的域。认证用户时，必须就搜索域的顺序作出明智的决定。

如果要认证域列表下面域中出现的用户标识以便访问，那么必须重命名这些用户标识。

可通过域列表控制访问。例如，如果用户的域不在列表中，那么用户不能连接。

注: 仅当数据库管理器配置中设置了 CLIENT 认证时，**DB2DOMAINLIST** 注册表变量才有效，如果在 Windows 域环境中需要从 Windows 桌面单点登录，那么需要该注册表变量。**DB2DOMAINLIST** 受某些版本的 DB2 服务器支持，然而，如果客户机和服务器都不位于 Windows 环境中，那么将不会强制使用 **DB2DOMAINLIST**。

域安全性支持（Windows）

以下示例说明了 DB2 数据库管理系统可以如何支持 Windows 域安全性。因为用户名与本地组在同一域上，所以可以进行连接。

在以下方案中，因为用户名与本地或全局组在同一域上，所以可以进行连接。

注意，用户名与本地或全局组无需在运行数据库服务器的域上定义，但它们必须在同一个域上。

表 50. 使用域控制器的成功连接

Domain1	Domain2
存在与 Domain2 的信任关系。	• 存在与 Domain1 的信任关系。 • 定义了本地或全局组 grp2。 • 定义了用户名 id2。 • 用户名 id2 是 grp2 的一部分。

DB2 服务器在此域中运行。从此域中发出了下列 DB2 命令:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

扫描本地或全局域，但找不到 id2。扫描域安全性。	
	在此域中找到用户名 id2。DB2 获取关于此用户名的其他信息（即，它是组 grp2 的一部分）。
因为用户名与本地或全局组在同一域上，所以可以进行连接。	

定义哪些用户拥有 SYSADM 权限（Windows）

如果未设置数据库管理器配置参数 **sysadm_group**（即，它为 NULL），那么某些用户具有 SYSADM 权限。

这些用户为:

- 本地 Administrators 组的成员
- 域控制器中 Administrators 组的成员（当 DB2 数据库管理器配置为在定义用户的位置上枚举这些用户的组时；可使用 **DB2_GRP_LOOKUP** 环境变量来配置组枚举）
- DB2ADMNS 组的成员（当启用了 Windows 扩展安全性时）。DB2ADMNS 组的位置在安装期间确定。
- LocalSystem 帐户

在某些情况下，先前提到的缺省行为并不合适。可以通过下列其中一种方法使用数据库管理器配置参数 **sysadm_group** 来覆盖此行为:

- 在 DB2 服务器上创建本地组并向该组添加想要让其拥有 SYSADM 权限的用户（域用户或本地用户）。DB2 数据库管理器应配置为在本地机器上枚举用户的组。
- 创建域组并向该组添加想要让其拥有 SYSADM 权限的用户。DB2 数据库管理器应配置为在定义用户的位置上枚举这些用户的组。

然后，使用下列命令将数据库管理器配置参数 **sysadm_group** 更新为此组:

```
DB2 UPDATE DBM CFG USING SYSADM_GROUP group_name
DB2STOP
DB2START
```

Windows 本地系统帐户支持

在 Windows 平台上，DB2 数据库系统支持应用程序通过本地隐式连接在本地系统帐户 (LSA) 环境下运行。本地系统帐户的授权标识是 SYSTEM。

如果使用非英语版的 Windows 操作系统，那么需要检查本地系统帐户的授权标识是否包含无效字符。例如，如果使用法语版的 Windows 操作系统， 那么本地系统帐户为 Système，但您不能将此帐户用作授权标识，因为它具有无效字符 è。

当数据库管理器配置参数 **sysadm_group** 设置为 NULL 时，本地系统帐户会被认为是系统管理员（拥有 SYSADM 权限）。

如果需要应用程序在本地系统帐户环境下运行以执行不属于 SYSADM 作用域内的数据库操作，那么必须将必需的数据库特权或权限授予本地系统帐户。例如，如果应用程序需要数据库管理员功能，请使用 GRANT（数据库权限）语句将 DBADM 权限授予本地系统帐户。

对于编写要通过此帐户运行的应用程序的开发者，他们需要知道 DB2 数据库系统对模式名以“SYS”开头的对象有一些限制。因此，如果应用程序包含创建 DB2 数据库对象的 DDL 语句，那么应该按下列要求编写这些应用程序:

- 对于静态查询，它们应该与 QUALIFIER 选项的值（而不是缺省值 SYSTEM）绑定在一起。
- 对于动态查询，要创建的对象应使用 DB2 数据库管理器支持的模式名显式限定，或者必须将 CURRENT SCHEMA 寄存器设置为 DB2 数据库管理器支持的模式名。

DB2 数据库实例启动后，会在进行首次组查询请求时收集本地系统帐户的组信息，并且重新启动实例之前不会刷新此信息。

使用 DB2ADMNS 和 DB2USERS 组的扩展 Windows 安全性

缺省情况下，在 Windows 操作系统上，会在除 IBM Data Server Runtime Client 和 DB2 驱动程序之外的所有 DB2 数据库产品中启用扩展安全性。在 Windows 平台上，IBM Data Server Runtime Client 和 DB2 驱动程序不支持扩展安全性。

当安装 DB2 数据库产品时，**启用操作系统安全性**复选框会出现在**对 DB2 对象启用操作系统安全性**面板上。除非禁用此选项，否则，安装程序会创建两个新组: DB2ADMNS 和 DB2USERS。DB2ADMNS 和 DB2USERS 是缺省组名; 可以选择在安装时为这些组指定不同的名称（如果选择静默安装，那么可以在安装响应文件内更改这些名称）。如果选择使用系统上已存在的组，您必须知道这样做会修改这些组的特权。将根据需要对这些组授予下表中所列示的特权。必须了解这些组是用于在*操作系统*级进行保护，而与 DB2 权限级别（例如，SYSADM、SYSMAINT 和 SYSCTRL）没有任何关联。但是，数据库管理员可以根据安装者或管理员的要求对一个或所有 DB2 权限级别使用 DB2ADMNS 组，而不使用缺省 Administrator 组。建议: 如果要指定的是 SYSADM 组，那么所使用的组应该是 DB2ADMNS 组。该组可以在安装期间或安装以后由管理员建立。

注: 可以将 DB2 Administrators 组（DB2ADMNS 或者在安装期间选择的名称）和 DB2 用户组（DB2USERS 或者在安装期间选择的名称）指定为本地组或域组。这两个组必须属于同一类型，即，要么都是本地组，要么都是域组。

如果您更改计算机名称，并且计算机组 DB2ADMNS 和 DB2USERS 是本地计算机组，那么必须更新全局注册表 DB2_ADMINGROUP 和 DB2_USERSGROUP。要在重命名之后更新注册表变量并重新启动计算机，请运行以下命令:

1. 打开命令提示符。
2. 运行 **db2extsec** 命令以更新安全性设置:

   ```
   db2extsec -a new computer name\DB2ADMNS -u new computer name\DB2USERS
   ```

注: 如果在 Windows Vista 上的 DB2 数据库产品中启用了扩展安全性，那么只有属于 DB2ADMNS 组的用户才可以运行 DB2 图形管理工具。此外，DB2ADMNS 组的成员需要以完全管理员特权启动工具。可以通过右键单击快捷方式，然后选择“以管理员身份运行”来完成。

通过 DB2ADMNS 和 DB2USERS 组获得的功能

DB2ADMNS 和 DB2USERS 组的成员具有以下功能:

- DB2ADMNS

  完全控制所有 DB2 对象（请参阅下面的受保护对象列表）
- DB2USERS

  对位于安装和实例目录中的所有 DB2 对象具有读和执行访问权，但对数据库系统目录下的对象没有访问权，对 IPC 资源具有有限访问权

  对于某些对象，需要时可以提供其他特权（例如，写特权、添加或更新文件特权等等）。此组中的成员无权访问数据库系统目录下的对象。

  注: 执行访问权的含义取决于对象；例如，对于 **.dll** 或 **.exe** 文件，具有执行访问权表示您有权执行该文件，但是对于目录，它表示您有权浏览该目录。

理想情况下，所有 DB2 管理员都应该是 DB2ADMNS 组的成员（同时也是本地 Administrators 组的成员），但实际上对此并没有严格要求。需要访问 DB2 数据库系统的任何人都*必须*是 DB2USERS 组的成员。要将用户添加到其中一个组:

1. 启动“用户和密码管理器”工具。
2. 从列表中选择要添加的用户名。
3. 单击“属性”。在“属性”窗口中，单击“组成员”选项卡。
4. 选择“其他”单选按钮。
5. 从下拉列表中选择适当的组。

在安装后添加扩展安全性（db2extsec 命令）

如果在未启用扩展安全性的情况下安装了 DB2 数据库系统，那么可以通过执行 **db2extsec** 命令来对它进行启用。要执行 **db2extsec** 命令，您必须是本地 Administrators 组的成员，这样您才有权修改受保护对象的 ACL。

您可以根据需要多次运行 **db2extsec** 命令，但运行完之后，您将不能禁用扩展安全性，除非在每次执行 **db2extsec** 之后立即发出 **db2extsec -r** 命令。

除去扩展安全性

注意:
在启用扩展安全性之后，请不要除去扩展安全性，除非绝对需要这样做。

可以通过运行 **db2extsec -r** 命令来除去扩展安全性，但是，只有在启用扩展安全性后尚未执行其他数据库操作（例如，创建数据库、创建新的实例和添加表空间等等）时此命令才会成功。用于除去扩展安全性选项的最安全的方法是卸载 DB2 数据库系统，删除所有相关的 DB2 目录（包括数据库目录），然后在不启用扩展安全性的情况下重新安装 DB2 数据库系统。

受保护的对象

可以使用 DB2ADMNS 和 DB2USERS 组保护的*静态*对象为:

- 文件系统
  - 文件
  - 目录
- 服务
- 注册表键

可以使用 DB2ADMNS 和 DB2USERS 组保护的*动态*对象为:

- IPC 资源，这包括:
  - 管道
  - 信号
  - 事件
- 共享内存

DB2ADMNS 和 DB2USERS 组拥有的特权

下表列示了指定给 DB2ADMNS 和 DB2USERS 组的特权:

表 51. DB2ADMNS 和 DB2USERS 组的特权

特权	DB2ADMNS	DB2USERS	原因
创建标记对象 (SeCreateTokenPrivilege)	Y	N	标记处理（某些标记处理操作需要，用于认证和授权）
替换进程级别标记 (SeAssignPrimaryTokenPrivilege)	Y	N	以另一个用户身份创建进程
增加限额 (SeIncreaseQuotaPrivilege)	Y	N	以另一个用户身份创建进程
作为操作系统的部件 (SeTcbPrivilege)	Y	N	LogonUser（Windows XP 之前的版本为了进行认证而执行 LogonUser API 时需要）
生成安全性审计 (SeSecurityPrivilege)	Y	N	处理审计和安全性日志
拥有文件或其他对象的所有权 (SeTakeOwnershipPrivilege)	Y	N	修改对象 ACL
增大调度优先级 (SeIncreaseBasePriorityPrivilege)	Y	N	修改进程工作集

表 51. DB2ADMNS 和 DB2USERS 组的特权 (续)

特权	DB2ADMNS	DB2USERS	原因
备份文件和目录 (SeBackupPrivilege)	Y	N	概要文件/注册表处理（执行某些用户概要文件和注册表处理例程时需要: LoadUserProfile、RegSaveKey(Ex)、RegRestoreKey、RegReplaceKey 和 RegLoadKey(Ex)）
复原文件和目录 (SeRestorePrivilege)	Y	N	概要文件/注册表处理（执行某些用户概要文件和注册表处理例程时需要: LoadUserProfile、RegSaveKey(Ex)、RegRestoreKey、RegReplaceKey 和 RegLoadKey(Ex)）
调试程序 (SeDebugPrivilege)	Y	N	标记处理（某些标记处理操作需要，用于认证和授权）
管理审计和安全性日志 (SeAuditPrivilege)	Y	N	生成审计日志条目
作为服务登录 (SeServiceLogonRight)	Y	N	作为服务运行 DB2
从网络访问此计算机 (SeNetworkLogonRight)	Y	Y	允许网络凭证（允许 DB2 数据库管理器使用 LOGON32_LOGON_NETWORK 选项来认证，这会对性能产生影响）
在认证之后冒充客户机 (SeImpersonatePrivilege)	Y	N	客户机冒名（Windows 在允许使用某些 API 来冒充 DB2 客户机时需要: ImpersonateLoggedOnUser、ImpersonateSelf 和 RevertToSelf 等）
锁定内存中的页 (SeLockMemoryPrivilege)	Y	N	大页支持
创建全局对象 (SeCreateGlobalPrivilege)	Y	Y	Terminal Server 支持（Windows 上需要）

Windows 2008 和 Windows Vista 或更高版本的注意事项: 用户访问控制功能部件

Windows 2008、Windows Vista 和 Windows 7 的“用户访问控制”(UAC) 功能部件将以下列方式影响 DB2 数据库系统。

以所有管理特权启动应用程序

缺省情况下，在 Windows 2008、Windows Vista 和 Windows 7 上，仅以标准用户权限启动应用程序，即使用户是本地管理员也是如此。要以更多特权来启动应用程序，需要从以所有管理特权运行的命令窗口中启动命令。DB2 安装过程将特地为 Windows 2008、Windows Vista 和 Windows 7 用户创建一个称为“命令窗口 - 管理员”的快捷方式。如果您想运行管理命令，那么建议您启动此快捷方式。

当您没有所有管理特权时，如果尝试在 Windows 2008、Windows Vista 和 Windows 7 上通过命令提示符或图形工具来执行 DB2 管理任务，那么您会遇到各种错误消息，提示您的访问被拒绝并且将无法成功完成这些任务。

要确定执行的操作是否为管理任务，检查是否符合以下任何情况:

- 需要 SYSADM、SYSCTRL 或 SYSMAINT 权限

- 会修改注册表中 HKLM 分支下的注册表键
- 写入 Program Files 目录下的目录

例如，以下操作将全部被视为管理任务:

- 创建和删除 DB2 实例
- 启动和停止 DB2 实例
- 创建数据库
- 更新数据库管理器配置参数或 DB2 管理服务器（DAS）配置参数
- 更新 CLI 配置参数和配置系统数据源名称（DSN）
- 启动 DB2 跟踪工具
- 运行 **db2pd** 实用程序
- 更改 DB2 概要文件注册表变量

要解决此问题，必须使用完全管理员特权运行的命令提示符或图形工具执行 DB2 管理任务。要以完全管理员特权启动命令提示符或图形工具，右键单击快捷方式，然后选择**以管理员身份运行**。

注: 如果启用了扩展安全性，那么您还必须是 DB2ADMNS 组的成员才能启动图形管理工具（例如，IBM Data Studio)）。

用户数据位置

用户数据（例如，实例目录中的文件）存储在 ProgramData\IBM\DB2*copy_name* 目录中，其中 *copy_name* 是 DB2 副本的名称（缺省情况下，DB2COPY1 是安装的第一个副本的名称）。在除 Windows 2008、Windows Vista 和 Windows 7 之外的其他 Windows 版本上，用户数据存储在 Documents and Settings\All Users\Application Data\IBM\DB2*copy_name* 中。

DB2 和 UNIX 安全性

需要了解一些特定于 UNIX 平台的安全性注意事项。

DB2 数据库不支持 root 用户直接充当数据库管理员。应使用 **su - <instance owner>** 作为数据库管理员。

通常，为了安全起见，请不要使用实例名作为 Fenced ID。但是，如果打算不使用受防护的 UDF 或存储过程，那么可以将 Fenced ID 设置为实例名，而不用创建另一个用户标识。

建议创建一个被认为与此组相关联的用户标识。将受防护的 UDF 和存储过程的用户指定为实例创建脚本的参数（**db2icrt ... -u <FencedID>**）。如果安装了“DB2 客户机”或“DB2 软件开发者工具箱”，那么不需要这样做。

DB2 和 Linux 安全性

可能需要了解一些特定于 Linux 平台的安全性注意事项。

更改密码支持（Linux）

在 Linux 操作系统上，DB2 数据库产品支持更改密码。

此支持是通过使用称为 IBMOSchgpwdclient.so 和 IBMOSchgpwdserver.so 的安全插件库实现的。

要在 Linux 上启用密码更改支持，请将数据库管理器配置参数 **clnt_pw_plugin** 设置为 IBMOSchgpwdclient，将 **srvcon_pw_plugin** 设置为 IBMOSchgpwdserver。

同时还必须在 /etc/pam.d 目录下创建名称为“db2”的 PAM 配置文件。

部署更改密码插件（Linux）

在 Linux 上，要支持更改 DB2 数据库产品中的密码，必须配置 DB2 实例以使用安全插件 IBMOSchgpwdclient 和 IBMOSchgpwdserver。

开始之前

插件库位于下列目录中:

- *INSTHOME*/sqllib/security*XX*/plugin/IBM/client/IBMOSchgpwdclient.so
- *INSTHOME*/sqllib/security*XX*/plugin/IBM/server/IBMOSchgpwdserver.so

其中 *INSTHOME* 是实例所有者的主目录，*securityXX* 是 security32 或 security64（取决于实例的位宽）。

过程

要在 DB2 实例中部署安全插件，执行下列步骤:

1. 作为具有 root 用户权限的用户登录。
2. 创建 PAM 配置文件: /etc/pam.d/db2

   确保此文件中包含由系统管理员定义的一组适当的规则。例如，在 SLES 9 上，可以按如下所示使用此文件:

   ```
   auth     required pam_unix2.so    nullok
   account  required pam_unix2.so
   password required pam_pwcheck.so  nullok tries=1
   password required pam_unix2.so    nullok use_authtok use_first_pass
   session  required pam_unix2.so
   ```

   在 RHEL 上，可以按如下所示使用此文件:

   ```
   #%PAM-1.0
   auth     required   /lib/security/$ISA/pam_env.so
   auth     sufficient /lib/security/$ISA/pam_unix.so likeauth nullok
   auth     required   /lib/security/$ISA/pam_deny.so

   account  required   /lib/security/$ISA/pam_unix.so
   account  sufficient /lib/security/$ISA/pam_succeed_if.so uid < 100 quiet
   account  required   /lib/security/$ISA/pam_permit.so

   password requisite  /lib/security/$ISA/pam_cracklib.so retry=3 dcredit=-1
   ucredit=-1
   password sufficient /lib/security/$ISA/pam_unix.so nullok use_authtok md5
   shadow remember=3
   password required   /lib/security/$ISA/pam_deny.so

   session  required   /lib/security/$ISA/pam_limits.so
   session  required   /lib/security/$ISA/pam_unix.so
   ```

3. 启用 DB2 实例中的安全插件:

   a. 将数据库管理器配置参数 **SRVCON_PW_PLUGIN** 的值更新为 IBMOSchgpwdserver:

```
db2 update dbm cfg using srvcon_pw_plugin IBMOSchgpwdserver
```

b. 将数据库管理器配置参数 **CLNT_PW_PLUGIN** 的值更新为 IBMOSchgpwdclient:

```
db2 update dbm cfg using CLNT_PW_PLUGIN IBMOSchgpwdclient
```

c. 确保将数据库管理器配置参数 **SRVCON_AUTH** 的值设置为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP，或者将数据库管理器配置参数 **SRVCON_AUTH** 的值设置为 NOT_SPECIFIED，并将 **AUTHENTICATION** 的值设置为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP。

附录 A. DB2 技术信息概述

DB2 技术信息以多种可以通过多种方法访问的格式提供。

您可以通过下列工具和方法获得 DB2 技术信息:

- DB2 信息中心
  - 主题（任务、概念和参考主题）
  - 样本程序
  - 教程
- DB2 书籍
  - PDF 文件（可下载）
  - PDF 文件（在 DB2 PDF DVD 中）
  - 印刷版书籍
- 命令行帮助
  - 命令帮助
  - 消息帮助

注: DB2 信息中心主题的更新频率比 PDF 书籍或硬拷贝书籍的更新频率高。要获取最新信息，请安装可用的文档更新或者参阅 ibm.com 上的 DB2 信息中心。

您可以在线访问 ibm.com 上的其他 DB2 技术信息，例如技术说明、白皮书和 IBM Redbooks® 出版物。请访问以下网址处的 DB2 信息管理软件资料库站点: http://www.ibm.com/software/data/sw-library/。

文档反馈

我们非常重视您对 DB2 文档的反馈。如果您想就如何改善 DB2 文档提出建议，请向 db2docs@ca.ibm.com 发送电子邮件。DB2 文档小组将阅读您的所有反馈，但无法直接给您答复。请尽可能提供具体的示例，这样我们才能更好地了解您所关心的问题。如果您要提供有关具体主题或帮助文件的反馈，请加上标题和 URL。

请不要使用以上电子邮件地址与 DB2 客户支持机构联系。如果您遇到文档无法解决的 DB2 技术问题，请与您当地的 IBM 服务中心联系以获得帮助。

硬拷贝或 PDF 格式的 DB2 技术库

下列各表描述 IBM 出版物中心（网址为 www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss）所提供的 DB2 资料库。可从 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947 下载 PDF 格式的 DB2 V10.1 手册的英文版本和翻译版本。

尽管这些表标识书籍有印刷版，但可能未在您所在国家或地区提供。

每次更新手册时，表单号都会递增。确保您正在阅读下面列示的手册的最新版本。

注: *DB2 信息中心*的更新频率比 PDF 或硬拷贝书籍的更新频率高。

表 52. DB2 技术信息

书名	书号	是否提供印刷版	最近一次更新时间
Administrative API Reference	SC27-3864-00	是	2012 年 4 月
Administrative Routines and Views	SC27-3865-01	否	2013 年 1 月
Call Level Interface Guide and Reference Volume 1	SC27-3866-01	是	2013 年 1 月
Call Level Interface Guide and Reference Volume 2	SC27-3867-01	是	2013 年 1 月
Command Reference	SC27-3868-01	是	2013 年 1 月
数据库管理概念和配置参考	S151-1758-01	是	2013 年 1 月
数据移动实用程序指南和参考	S151-1756-01	是	2013 年 1 月
数据库监视指南和参考	S151-1759-01	是	2013 年 1 月
数据恢复及高可用性指南与参考	S151-1755-01	是	2013 年 1 月
数据库安全性指南	S151-1753-02	是	2013 年 1 月
DB2 Workload Management Guide and Reference	SC27-3891-01	是	2013 年 1 月
开发 ADO.NET 和 OLE DB 应用程序	S151-1765-01	是	2013 年 1 月
开发嵌入式 SQL 应用程序	S151-1763-01	是	2013 年 1 月
Developing Java Applications	SC27-3875-01	是	2013 年 1 月
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	否	2012 年 4 月
Developing RDF Applications for IBM Data Servers	SC27-4462-00	是	2013 年 1 月
开发用户定义的例程（SQL 和外部例程）	S151-1761-01	是	2013 年 1 月
数据库应用程序开发入门	G151-1764-01	是	2013 年 1 月
Linux 和 Windows 上的 DB2 安装和管理入门	G151-1769-00	是	2012 年 4 月
全球化指南	S151-1757-00	是	2012 年 4 月
安装 DB2 服务器	G151-1768-01	是	2013 年 1 月
安装 IBM Data Server Client	G151-1751-00	否	2012 年 4 月
消息参考第 1 卷	S151-1767-01	否	2013 年 1 月

表 52. DB2 技术信息 (续)

书名	书号	是否提供印刷版	最近一次更新时间
消息参考第 2 卷	S151-1766-01	否	2013 年 1 月
Net Search Extender 管理和用户指南	S151-1905-01	否	2013 年 1 月
分区和集群指南	S151-1754-01	是	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	否	2013 年 1 月
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	否	2013 年 1 月
pureXML 指南	S151-1775-01	是	2013 年 1 月
Spatial Extender User's Guide and Reference	SC27-3894-00	否	2012 年 4 月
SQL 过程语言: 应用程序启用和支持	S151-1762-01	是	2013 年 1 月
SQL Reference Volume 1	SC27-3885-01	是	2013 年 1 月
SQL Reference Volume 2	SC27-3886-01	是	2013 年 1 月
Text Search Guide	SC27-3888-01	是	2013 年 1 月
故障诊断和调整数据库性能	S151-1760-01	是	2013 年 1 月
升级到 DB2 V10.1	S151-1770-01	是	2013 年 1 月
DB2 V10.1 新增内容	S151-1752-01	是	2013 年 1 月
XQuery 参考	S151-1774-01	否	2013 年 1 月

表 53. 特定于 DB2 Connect 的技术信息

书名	书号	是否提供印刷版	最近一次更新时间
DB2 Connect 安装和配置 DB2 Connect Personal Edition	S151-1773-00	是	2012 年 4 月
DB2 Connect 安装和配置 DB2 Connect 服务器	S151-1772-01	是	2013 年 1 月
DB2 Connect 用户指南	S151-1771-01	是	2013 年 1 月

从命令行处理器显示 SQL 状态帮助

DB2 产品针对可能充当 SQL 语句结果的条件返回 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

过程

要启动 SQL 状态帮助，请打开命令行处理器并输入:

```
? sqlstate or ? class code
```

其中，*sqlstate* 表示有效的 5 位 SQL 状态，*class code* 表示该 SQL 状态的前 2 位。例如，`? 08003` 显示 08003 SQL 状态的帮助，而 `? 08` 显示 08 类代码的帮助。

访问不同版本的 DB2 信息中心

您可以在 ibm.com® 上的不同信息中心中找到其他版本 DB2 产品的文档。

关于此任务

对于 DB2 V10.1 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v10r1。

对于 DB2 V9.8 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/。

对于 DB2 V9.7 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/。

对于 DB2 V9.5 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

对于 DB2 V9.1 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9/。

对于 DB2 V8 主题，请转至 *DB2 信息中心* URL: http://publib.boulder.ibm.com/infocenter/db2luw/v8/。

更新安装在计算机或内部网服务器上的 DB2 信息中心

安装在本地的 DB2 信息中心必须定期进行更新。

开始之前

必须已安装 DB2 V10.1 信息中心。有关详细信息，请参阅安装 DB2 服务器中的“使用 DB2 安装向导来安装 DB2 信息中心”主题。所有适用于安装信息中心的先决条件和限制同样适用于更新信息中心。

关于此任务

可以自动或手动更新现有的 DB2 信息中心:

- 自动更新将更新现有的信息中心功能部件和语言。自动更新的一个优点是，与手动更新相比，信息中心的不可用时间较短。另外，自动更新可设置为作为定期运行的其他批处理作业的一部分运行。
- 可以使用手动更新方法来更新现有的信息中心功能部件和语言。自动更新可以缩短更新过程中的停机时间，但如果您想添加功能部件或语言，那么必须执行手动过程。例如，如果本地信息中心最初安装的是英语和法语版，而现在还要安装德语版；那么手动更新将安装德语版，并更新现有信息中心的功能和语言。但是，手动更新要求您手动停止、更新和重新启动信息中心。在整个更新过程期间信息中心不可用。在自动更新过程中，信息中心仅在更新完成后停止工作以重新启动信息中心。

此主题详细说明了自动更新的过程。有关手动更新的指示信息，请参阅“手动更新安装在您的计算机或内部网服务器上的 DB2 信息中心”主题。

过程

要自动更新安装在计算机或内部网服务器上的 DB2 信息中心:

1. 在 Linux 操作系统上,
   a. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 /opt/ibm/db2ic/V10.1 目录中。
   b. 从安装目录浏览至 doc/bin 目录。
   c. 运行 update-ic 脚本:

      ```
      update-ic
      ```
2. 在 Windows 操作系统上,
   a. 打开命令窗口。
   b. 浏览至信息中心的安装位置。缺省情况下，DB2 信息中心安装在 <Program Files>\IBM\DB2 Information Center\V10.1 目录中，其中 <Program Files> 表示 Program Files 目录的位置。
   c. 从安装目录浏览至 doc\bin 目录。
   d. 运行 update-ic.bat 文件:

      ```
      update-ic.bat
      ```

结果

DB2 信息中心将自动重新启动。如果更新可用，那么信息中心会显示新的以及更新后的主题。如果信息中心更新不可用，那么会在日志中添加消息。日志文件位于 doc\eclipse\configuration 目录中。日志文件名称是随机生成的编号。例如，1239053440785.log。

手动更新安装在计算机或内部网服务器上的 DB2 信息中心

如果您已在本地安装 DB2 信息中心，那么可从 IBM 获取文档更新并进行安装。

关于此任务

手动更新安装在本地的 *DB2 信息中心*要求您:

1. 停止计算机上的 *DB2 信息中心*，然后以独立方式重新启动信息中心。如果以独立方式运行信息中心，那么网络上的其他用户将无法访问信息中心，因而您可以应用更新。DB2 信息中心的工作站版本总是以独立方式运行。
2. 使用“更新”功能部件来查看可用的更新。如果有您必须安装的更新，那么请使用“更新”功能部件来获取并安装这些更新。

   注: 如果您的环境要求在一台未连接至因特网的机器上安装 *DB2 信息中心*更新，请使用一台已连接至因特网并已安装 *DB2 信息中心*的机器将更新站点镜像至本地文件系统。如果网络中有许多用户将安装文档更新，那么可以通过在本地也为更新站点制作镜像并为更新站点创建代理来缩短每个人执行更新所需要的时间。
   如果提供了更新包，请使用“更新”功能部件来获取这些更新包。但是，只有在单机方式下才能使用“更新”功能部件。

3. 停止独立信息中心，然后在计算机上重新启动 *DB2 信息中心*。

注: 在 Windows 2008、Windows Vista 和更高版本上，稍后列示在此部分的命令必须作为管理员运行。要打开具有全面管理员特权的命令提示符或图形工具，请右键单击快捷方式，然后选择**以管理员身份运行**。

过程

要更新安装在您的计算机或内部网服务器上的 *DB2 信息中心*:

1. 停止 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 > 控制面板 > 管理工具 > 服务**。右键单击 **DB2 信息中心**服务，并选择**停止**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv10 stop
     ```
2. 以独立方式启动信息中心。
   - 在 Windows 上:
     a. 打开命令窗口。
     b. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 *Program_Files*\IBM\DB2 Information Center\V10.1 目录中，其中 *Program Files* 表示 Program Files 目录的位置。
     c. 从安装目录浏览至 doc\bin 目录。
     d. 运行 help_start.bat 文件:

        ```
        help_start.bat
        ```
   - 在 Linux 上:
     a. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 /opt/ibm/db2ic/V10.1 目录中。
     b. 从安装目录浏览至 doc/bin 目录。
     c. 运行 help_start 脚本:

        ```
        help_start
        ```

   系统缺省 Web 浏览器将打开以显示独立信息中心。
3. 单击**更新**按钮（  ）。（必须在浏览器中启用 JavaScript。）在信息中心的右边面板上，单击**查找更新**。将显示现有文档的更新列表。
4. 要启动安装过程，请检查您要安装的选项，然后单击**安装更新**。
5. 在安装进程完成后，请单击**完成**。
6. 要停止独立信息中心，请执行下列操作:
   - 在 Windows 上，浏览至安装目录中的 doc\bin 目录并运行 help_end.bat 文件:

     ```
     help_end.bat
     ```

     注: help_end 批处理文件包含安全地停止使用 help_start 批处理文件启动的进程所需的命令。不要使用 Ctrl-C 或任何其他方法来停止 help_start.bat。
   - 在 Linux 上，浏览至安装目录中的 doc/bin 目录并运行 help_end 脚本:

     ```
     help_end
     ```

注: help_end 脚本包含安全地停止使用 help_start 脚本启动的进程所需的命令。不要使用任何其他方法来停止 help_start 脚本。

7. 重新启动 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 > 控制面板 > 管理工具 > 服务**。右键单击 **DB2 信息中心**服务，并选择**启动**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv10 start
     ```

结果

更新后的 *DB2 信息中心*将显示新的以及更新后的主题。

DB2 教程

DB2 教程帮助您了解 DB2 数据库产品的各个方面。这些课程提供了逐步指示信息。

开始之前

您可以在信息中心中查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/。

某些课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述，请参阅教程。

DB2 教程

要查看教程，请单击标题。

*pureXML 指南*中的『**pureXML®**』
: 设置 DB2 数据库以存储 XML 数据以及对本机 XML 数据存储器执行基本操作。

DB2 故障诊断信息

我们提供了各种各样的故障诊断和问题确定信息来帮助您使用 DB2 数据库产品。

DB2 文档
: 您可以在《*故障诊断和调整数据库性能*》或者 *DB2 信息中心*的“数据库基础”部分中找到故障诊断信息，这些信息包含以下内容:
  - 有关如何使用 DB2 诊断工具和实用程序来隔离和确定问题的信息。
  - 一些最常见问题的解决方案。
  - 旨在帮助您解决 DB2 数据库产品使用过程中可能会遇到的其他问题的建议。

IBM Support Portal
: 如果您遇到问题并且希望得到帮助以查找可能的原因和解决方案，请访问 IBM Support Portal 。这个技术支持站点提供了指向最新 DB2 出版物、技术说明、授权程序分析报告（APAR 或错误修订）、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

  访问 IBM 支持门户网站网址为: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

信息中心条款和条件

如果符合以下条款和条件，那么授予您使用这些出版物的许可权。

适用性: 用户需要遵循 IBM Web 站点的使用条款及以下条款和条件。

个人使用: 只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

商业使用: 只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作这些出版物的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

权利: 除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论是明示的还是暗含的。

IBM 保留根据自身的判断，认为对出版物的使用损害了 IBM 的权益（由 IBM 自身确定）或未正确遵循以上指示信息时，撤回此处所授予权限的权利。

只有您完全遵循所有适用的法律和法规，包括所有的美国出口法律和法规，您才可以下载、出口或再出口该信息。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的关于适销和适用于某种特定用途的保证。

IBM 商标: IBM、 IBM 徽标和 ibm.com 是 International Business Machines Corp., 在全球许多管辖区域注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。当前的 IBM 商标列表，可从 Web 站点 www.ibm.com/legal/copytrade.shtml获得

附录 B. 声明

本信息是为在美国提供的产品和服务编写的。有关非 IBM 产品的信息是基于首次出版此文档时的可获信息且会随时更新。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节字符集 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是此 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: (i) 允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及 (ii) 允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口 (API) 进行应用程序的开发、使用、经销或分发，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。此样本程序“按现状”提供，且不附有任何种类的保证。对于使用此样本程序所引起的任何损坏，IBM 将不承担责任。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

© (*your company name*) (*year*). 此部分代码是根据 IBM 公司的样本程序衍生出来的。
© Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

商标

IBM 商标: IBM、 IBM 徽标和 ibm.com 是 International Business Machines Corp., 在全球许多管辖区域注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公

司的商标。当前的 IBM 商标列表，可从 Web 站点 www.ibm.com/legal/copytrade.shtml 上“版权和商标信息”部分获取。

下列各项是其他公司的商标或注册商标

- Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。
- Java 和所有基于 Java 的商标和徽标是 Oracle 和/或其子公司的商标或注册商标。
- UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。
- Intel、Intel 徽标、Intel Inside、Intel Inside 徽标、Celeron、Intel SpeedStep、Itanium 和 Pentium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。
- Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

索引

[A]

[B]

[C]

[D]

[F]

[G]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

K

L

N

O

P

Q

R

S

T

U

V

W

X

[特别字符]

Printed in China

S151-1753-02

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

数据库安全性指南