基于关系 DBMS 的一种数据库加密实现方法

A Method of Database Encryption Basing on RDBMS

王 洪 吕述望 刘振华 （中国科学院研究生院 100039）

摘要：本文提出了一种在 DBMS 内实现数据库加密的方法，并对各相关功能的实现和操作进行了阐述。该加密方法在尽可能不影响原有系统功能的基础上，在方便用户使用的同时，提高了数据库系统的安全。

关键词：数据库加密 数据库管理系统

1 引言

数据库加密是利用现有的数据库和加密技术，来研究如何对数据库中的数据加、解密，从而提高数据库系统的安全。

数据库加密可以在 OS、DBMS 内层、DBMS 外层上实现。OS 层上的数据库加密由于存在太多的问题和缺陷，已经不被人们考虑。已有的数据库加密产品都是在 DBMS 外层实现的，这种实现方式虽说做到了数据和密钥的分离，可以有效地抵御外部的非法入侵，但对企业内部的攻击却没有很好的防范措施；另外，这种方式还存在着：使数据和应用程序紧密结合起来、在 DBMS 之外又增加了其他的数据管理工具、硬盘存储空间浪费大等缺陷。

其实，数据的加密与否只是数据的一个属性，数据的加密存储是数据的一种保存方式，关乎数据的物理存储，这些都是 DBMS 现有功能的一部分。另外从数据和应用程序的分离、数据的维护和安全来看，数据库加密都应当在 DBMS 内实现。本文就是基于关系 DBMS，提出了一种数据库加密的实现方法。

2 系统设计

在 DBMS 中实现数据库加密，不可避免地要考虑 DBMS 的实现方式、操作过程等。我们认为数据库加密的 DBMS 实现应当做到以下几点：

（1）不要改变现有数据库系统，尽可能减少对现有系统的影响，属性列被加密后不影响在 DDL、DML 中的使用。

（2）数据库加密是在现有数据库安全基础上对系统更深一层次的安全保护，它基于访问控制，而不能依赖于访问控制。

（3）在实现数据库加密后，特权用户（如 DBA、系统安全员）的权限应受到限制，没有授权就无法看到被加密的数据。

（4）由于数据库中需要加密的属性列会很多，加、解密时需要给定加密算法、密钥、密钥长度等加密属性，用户又不可能记住所有的密钥，另外基于安全的考虑，加密属性需要定期修改，这些都要求数据库加密要有一个合理、安全的密钥管理体制，既便于系统管理，又方便用户的使用。

基于以上的考虑，本文的设计思路是在 DBMS 中引入一个加密子系统，该子系统提供和软、硬件加密模块的接口，完成加密定义、操作、维护以及密钥的管理、使用等各项功能，所有和加密有关的操作都需要在加密子系统中完成。用户连接到数据库或进入 DBMS 交互式处理环境，可以执行原有的各项功能。但如果要执行和加密有关的操作，就需要通过命令、输入密码进入加密子系统。

需要说明的是：在实现数据库加密后，用户对表中加密属性列的访问，既需要加密子系统中的访问授权，也需要数据库访问控制中的授权。

下面就加密子系统的设置以及其他相关功能的实现及操作分别进行阐述。

2.1 为用户设置加密子系统

用户能够或被取消访问加密子系统，这需要由系统安全员在 DBMS 的交互式处理环境下使用以下命令来完成：

enable encryption for {user_id} {password}

disable encryption for {user_id}

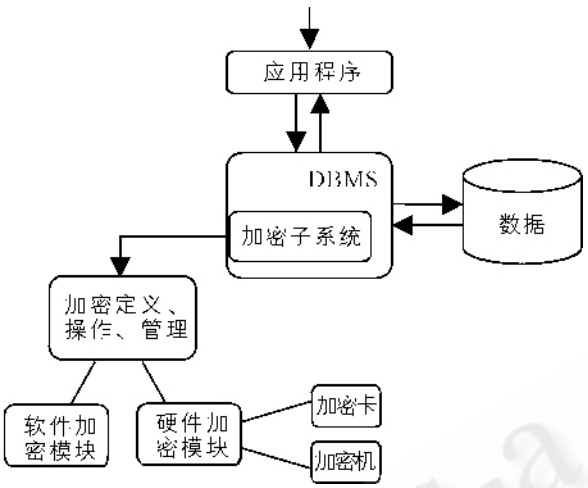

图 1

用户要访问加密子系统,必须先连接到数据库或进入 DBMS 交互式处理环境。用户可以使用下列命令进入和退出加密子系统:

set encryption on using {password}

set encryption off

在加密子系统中,用户、系统安全员使用下面的命令更改用户加密子系统密码:

set password {new_password} {password_confirm}

set password {new_password} {password_confirm} for user_id

2.2 加密属性及密钥管理

在数据库系统中引入一个新的对象 cryptograph,用来指定加密需要的加密算法、密钥、密钥长度等加密属性。用户根据要求创建自己需要的 cryptograph 对象,从而在创建表时可以为需要加密的属性列指定一个对象,确定加密时所需要的加密属性。

(1) 对象创建。使用下面的命令创建 cryptograph 对象:

create cryptograph {cryptograph_name} with {algorithm_name} {length} [interval]

参数 length、interval 指定密钥的长度、密钥有效期信息,对象的密钥由系统自动生成、加密后存储,用户无法直接获取。创建对象的用户就成为它的所有者,有关对象的所有信息都存放在数据字典中。

(2) 对象授权管理。具有加密子系统访问能力的用户,能够在加密子系统中创建 cryptograph 对象,该用户就成为新建对象的所有者。只有对象的所有者才能够修改、删除 cryptograph 对象,才能将对象的使用权限授予其他的用户。

用户只有在获得 cryptograph 对象的使用权限之后,才能在属性列的定义中使用该对象,才能操作由该对象定义的属性列。

使用下列命令授予、撤消用户对数据库对象 cryptograph 的使用权限:

grand use on {cryptograph_name} to {user_id}

revoke use on {cryptograph_name} from {user_id}

要成功地撤消用户 cryptograph 对象的使用权限,必须保证该用户没有使用该对象定义任何表的属性列。

对象的使用授权被单独存放在数据字典的授权表中,系统在对象创建时自动为对象所有者在授权表中添加一条记录,对象所有者给其他用户的授权也是在授权表中增加一条和该对象有关的授权记录。授权表中所有记录都包含了相应对象的密钥,对对象所有者来说,记录中对象的密钥是在创建对象时,系统将自动生成的密钥使用所有者的加密子系统密码加密后存储的;对其他的用户来说,记录中对象的密钥是在所有者授权时,系统将经所有者密码解密后的对象密钥,再使用被授权用户的密码加密后存储的。这样,用户在成功地登录进加密子系统之后,由于系统能够使用用户的加密子系统密码获得对象的密钥,从而就可以操作和该对象有关的所有加密属性列了。

用户在修改了自己的加密子系统密码后,由于加密子系统不会自动对授权表进行更新,用户现有密码和加密对象密钥的密码不一致,用户由于无法获取正确的对象密钥,进而无法正常操作和该对象有关的加密属性列。这时就需要对象所有者首先收回授权,然后重新给用户授权。

如果对象所有者忘记或修改了加密子系统密码,他就无法正常操作和该对象相关的加密属性列,也无法再向其他的用户进行授权。这时尽管其他已经被授权的用户仍旧可以正常操作,但最好是对数据进行明

文备份,重建对象、表后进行恢复。

这种对象授权及密钥保存方式,可以有效地限制特权用户的权限,他们在没有得到对象所有者授权的情况下,是没有办法正常操作和该对象有关的加密属性列。如果数据库管理员更改用户的帐号密码,由于无法得到正确的加密子系统密码,他无法以该用户的名义进入加密子系统;如果系统安全员更改了用户的加密子系统密码并以该用户的名义进入加密子系统,系统利用这个密码不会得到正确的对象密钥,因而对有关加密属性列的操作也不会成功。

(3) 对象删除。使用下面的命令删除 cryptograph 对象:

drop cryptograph {cryptograph_name}

要成功地删除 cryptograph 对象,需要满足下面的两个条件:(A) 没有一个加密属性列在定义中使用了这个对象;(B) 除了对象所有者,没有一个用户被赋予该对象的使用权限。

(4) 对象更改。使用下面的命令修改 cryptograph 对象的加密算法、密钥长度、密钥生存期等信息:

alter cryptograph {cryptograph_name} with {algorithm_name} {length} [interval]

如果修改了对象的加密算法、密钥长度,系统将会重新生成新的密钥,还将自动执行下面的操作:(A) 对所有使用这个对象的加密属性列,首先使用旧的密钥解密数据,然后再用新的密钥加密后存储;(B) 对所有被赋予该对象使用权限的用户,用他们的加密子系统密码加密新的密钥后更新授权表。

另外加密子系统提供系统故障恢复能力,保证这些操作的正常完成并能够从中断中恢复。

2.3 加密属性列的定义

在数据库系统中引入新的数据类型 encryption,将所有需要加密的属性列都一律定义为 encryption 类型。在创建表时可以使用下面的格式定义加密属性列:

column_name encryption with {cryptograph_name} {type_name} [length]

其中:参数 cryptograph_name 指定了属性列使用的 cryptograph 对象,type_name 和 length 指定明文数值的类型、长度。

给定属性列的明文数据类型和长度,系统就可以在操作加密属性列时进行数据类型及取值的一致性检查;另外系统会自动根据明文数据的类型、长度以及加密算法、密钥长度计算出该属性列所需存储空间,然后作为这个 encryption 类型的长度。(由于加密属性列明文数据类型、长度可变,系统无法为 encryption 类型设定固定的长度)

如果要修改加密属性列的加密属性定义,需要将表中数据以明文的方式导出,在修改了加密属性后,再将明文数据加密后装入表中。

2.4 对加密属性列的操作

对加密属性列的操作是通过两个系统函数 Fun_encryption、Fun_decryption 来实现,它们的定义如下:

Encryption Fun_encryption (column_name, column_value)

Void Fun_decryption (column_name)

Fun_encryption 函数调用加密模块,对输入的数据加密后返回 encryption 类型的数据。在执行加密操作前,该函数会根据 column_name 值在数据字典中查找属性列定义,然后执行明文数据的类型匹配检查、数据取值的一致性检查。

Fun_decryption 函数对属性列的数据进行解密,根据属性列明文数据类型返回数据。

向加密属性列中插入数据时,需要使用 Fun_encryption 函数加密数据并转换为 encryption 类型;对加密属性列的修改,首先通过 Fun_decryption 函数获得明文数据,在更改完成后再使用 Fun_encryption 函数加密后存储。

对加密属性列的查询,可以使用 Fun_encryption 函数加密要查询的数据,然后在属性列上寻找满足条件的记录;也可以使用 Fun_decryption 函数解密属性列上的每一个数值后再进行比较,比较而言,前者的执行效率肯定会更高一些。

2.5 加密数据的备份和恢复

为保证数据的可用性、可恢复性,在实现数据库加密后,系统应提供加密属性列在密文、明文两种形式下的备份和恢复、数据的导出和装入。密文方式的操作可以直接在 DBMS 的交互处理环境下完成,而明文方式则需要进入加密子系统并在命令中指定加、解密选项。具体的备份和恢复、数据导出和装入命令如下:

backup table {table_name} [with decryption] to

{device_name}

restore table {table _name} [with encryption] from {device_name}

export table {table _name} [with decryption] to {device_name}

import table {table _name} [with encryption] from {device_name}

2.6 加密属性列存在的限制

属性列被加密后,仍旧可以在查询语句中实现表间连接、数据的分组和排序,仍旧可以实现表间的参照完整性(这时需要两个表的加密属性列使用相同的 cryptograph 对象),但不可避免地要影响系统性能。另外需要指出的是:在实现加密后,对加密属性列的操作会存在以下两个限制:

(1) 不能在加密属性列上建立索引。加密后属性的值已经改变。如果建立索引,索引的排序顺序和明文数据的顺序并不一致,这不但不会给查询带来好处,反而使敏感数据多处存放。

(2) 不能对加密属性列进行信息统计。在属性列被加密后,由于数值的改变,数据的统计信息对优化器的优化算法已经失去意义。

虽然存在以上的限制和性能影响,但需要加密的数据一般是企业的敏感数据,只有极少数人可以访问,使用频度低,一般也不会出现在查询条件中,因此属性列在加密后不会给数据库系统带来太大的影响。

3 结束语

在使用这种方法实现数据库加密后,数据的管理和维护可以完全在 DBMS 中实现,用户只需要掌握两个密码,特权用户的权限受到限制,数据库系统的安全进一步得到提高,该方法对数据库加密在关系 DBMS 中的实现有比较强的实用价值。

参考文献

1 高品均、陈荣良,数据库加密技术综述。
http://www.tongji.edu.cn/~yangdy/computer/DataBase/paper2.htm

2 APPLICATION SECURITY, INC.。DBEncryption for User Guide。
http://www.appsecinc.com

3 RSA SECURITY, INC.。Developing a Database Encryption Strategy。
http://www.rsasecurity.com.