第四章　数据库安全性

数据库安全性

- 问题的提出
  - 数据库的一大特点是数据可以共享
  - 数据共享必然带来数据库的安全性问题
  - 数据库系统中的数据共享不能是无条件的共享

例： 军事秘密、国家机密、新产品实验数据、
市场需求分析、市场营销策略、销售计划、
客户档案、医疗档案、银行储蓄数据

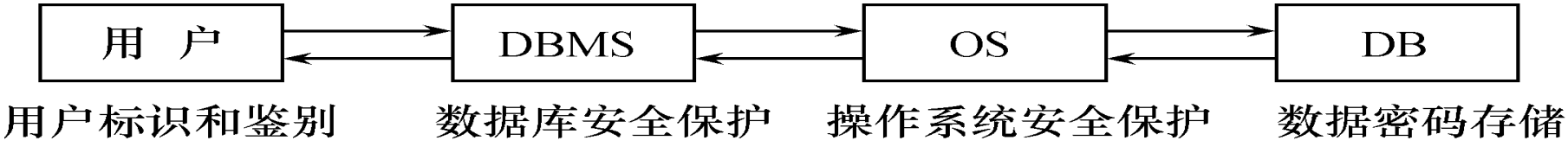
数据库安全性

数据库安全性：

数据库的安全性是指保护数据库系统和数据，以**防止非法使用**所造成的数据泄漏、更改或破坏。

- 对数据库安全性产生威胁的因素主要有：
  - 非授权用户对数据库的恶意存取和破坏
  - 数据库中重要或敏感的数据被泄漏
  - 安全环境的脆弱性

- 计算机系统中，安全措施是一级一级层层设置



计算机系统的安全模型

数据库安全性控制的常用方法

- 用户身份鉴别
- 存取控制
- 视图
- 审计
- 密码存储

4.1 数据库安全性控制

4.1.1 用户身份鉴别

4.1.2 存取控制

4.1.3 数据库角色

4.1.1 用户身份鉴别

- 用户身份鉴别

  (Identification & Authentication)

  - 系统提供的最外层安全保护措施

常用的用户身份鉴别方法：

- 静态口令鉴别。
  - 方式简单，但容易被攻击，安全性较低。
- 动态口令鉴别。
  - 口令动态变化，一次一密，常用的方式如短信密码和动态令牌方式。
  - 增加了口令被窃取或破解的难度，目前较为安全的鉴别方式。

常用的用户身份鉴别方法：

- 生物特征鉴别。
  - 通过生物特征进行认证。如指纹、虹膜、掌纹等。
- 智能卡鉴别。
  - 智能卡是一种不可复制的硬件。由用户随身携带。
  - 每次从智能卡中读取的数据是静态的。

4.1.2 存取控制

- 存取控制机制组成
  - 定义用户权限（权限即用户对某一数据对象的操作权力）
  - 合法权限检查
- 用户权限定义和合法权检查机制一起组成了 DBMS的安全子系统

- 常用存取控制方法
  - 自主存取控制（Discretionary Access Control，简称DAC)
    - 灵活
  - 强制存取控制（Mandatory Access Control, 简称 MAC)
    - 严格

一、自主存取控制方法（DAC）

- 同一用户对于不同的数据对象有不同的存取权限
- 不同的用户对同一对象也有不同的权限
- 用户还可将其拥有的存取权限转授给其他用户

- 通过 SQL 的 GRANT 语句和 REVOKE 语句实现
- 用户权限组成
  - 数据对象
  - 操作类型
- 定义用户存取权限：定义用户可以在哪些数据库对象上进行哪些类型的操作
- 定义存取权限称为授权（Authorization）

- 定义存取权限
  - 用户
- 检查存取权限
  - DBMS
- 授权粒度
  - 数据对象粒度：数据库、表、属性列、行
- 数据值粒度：存取谓词
  - 授权粒度越细，授权子系统就越灵活，能够提供的安全性就越完善。但另一方面，因数据字典变大变复杂，系统定义与检查权限的开销也会相应地增大。

1、SQL的授权功能

- GRANT语句的一般格式：

GRANT <权限>[,<权限>]...

ON <对象类型><对象名>[, <对象类型><对象名>]…

TO <用户>[,<用户>]...

[WITH GRANT OPTION];

- 将对指定操作对象的指定操作权限授予指定的用户。

(1) 用户的权限

- 基本表或视图的属主拥有对该表或视图的一切操作权限。
- 存取控制的对象不仅有数据本身，还有数据库模式。

(2) 接受权限的用户

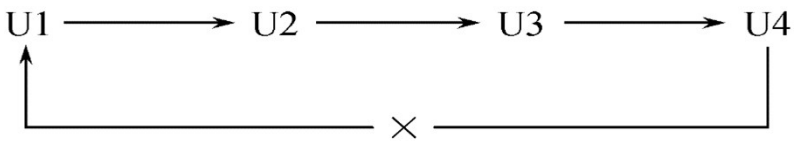
- 一个或多个具体用户
- PUBLIC（全体用户）

关系数据库系统中的存取权限表

对象类型	对象	操作类型
数据库模式	模式	CREATE SCHEMA
	基本表	CREATE TABLE,ALTER TABLE
	视图	CREATE VIEW
	索引	CREATE VIEW
数据	基本表和视图	SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALL PRIVILEGES
	属性列	SELECT, INSERT, UPDATE, REFERENCES, ALL PRIVILEGES

(3) WITH GRANT OPTION子句

- WITH GRANT OPTION子句:
  - 指定：可以再授予
  - 没有指定：不能传播
- 不允许循环授权，即被授权者不能把权限再授回给授权者或其祖先。



例1 把查询Student表权限授给用户U1

```
GRANT   SELECT
ON   TABLE   Student
TO   U1;
```

SQLSever中语法：
Grant select on student to U1；
MySQL中对象类型是可选项：
object_type:
{ TABLE | FUNCTION | PROCEDURE }

例2 把对Student表和Course表的全部权限授予用户U2和U3

```
GRANT ALL [PRIVILIGES]
ON TABLE Student, Course
TO U2, U3;
```

例3 把对表SC的查询权限授予所有用户

```
GRANT SELECT
ON [TABLE] SC
    TO PUBLIC;
```

例4 把查询Student表和修改学生姓名的权限授给用户U4

```
GRANT UPDATE(Sname), SELECT
ON [TABLE] Student
TO U4;
```

例5 把对表SC的INSERT权限授予U5用户，并允许他再将此权限授予其他用户

```
GRANT INSERT
ON [TABLE] SC
TO U5
WITH GRANT OPTION;
```

传播权限

执行例5后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限：

```
GRANT INSERT ON [TABLE] SC TO U6
WITH GRANT OPTION;
```

同样，U6还可以将此权限授予U7：

```
GRANT INSERT ON [TABLE] SC TO U7;
```

但U7不能再传播此权限。

GRANT语句小结

- 一次向一个用户授权（例1）这是最简单的一种授权操作;
- 一次向多个用户授权（例2、例3）；
- 一次传播多个同类对象的权限(例2）；
- 一次可以完成对基本表、视图和属性列这些不同对象的授权（例4）；

2、SQL收回权限的功能

- REVOKE语句的一般格式为：

  REVOKE <权限>[,<权限>]...

  ON <对象类型><对象名>[, <对象类型><对象名>]…

  FROM <用户>[,<用户>]... [CASCADE|RESTRICT] ;

- 功能：从指定用户那里收回对指定对象的指定权限

例6　把用户U4修改学生学号的权限收回

```
REVOKE UPDATE(Sno)
ON TABLE Student
FROM U4;
```

例7　收回所有用户对表SC的查询权限

```
REVOKE SELECT
ON TABLE SC
FROM PUBLIC;
```

例8 把用户U5对SC表的INSERT权限收回

```
REVOKE INSERT
ON TABLE SC
FROM U5;
```

小结: SQL灵活的授权机制

- DBA：拥有所有对象的所有权限
  - 不同的权限授予不同的用户
- 用户：拥有自己建立的对象的全部的操作权限
  - GRANT：授予其他用户
- 被授权的用户
  - “继续授权”许可：再授予
- 所有授予出去的权力在必要时又都可用REVOKE语句收回

- 优点
  - 能够通过授权机制有效地控制其他用户对敏感数据的存取
- 缺点
  - 可能存在数据的“无意泄露”
  - 原因：这种机制仅仅通过对数据的存取权限来进行安全控制，而数据本身并无安全性标记。
  - 解决：对系统控制下的所有主客体实施强制存取控制策略

二、强制存取控制方法 （MAC）

- 保证更高程度的安全性
- 用户不能**直接**感知或进行控制
- 适用于对数据有严格而固定密级分类的部门
  - 军事部门
  - 政府部门
- 在通用数据库系统中不十分有用，只是某些专用系统中才有用

- 主体是系统中的活动实体
  - DBMS所管理的实际用户
  - 代表用户的各进程
- 客体是系统中的被动实体，是受主体操纵的
  - 文件
  - 基表
  - 索引
  - 视图

- 敏感度标记（Label）
  - 绝密（Top Secret）
  - 机密（Secret）
  - 可信（Confidential）
  - 公开（Public）
- 主体的敏感度标记称为许可证级别（Clearance Level）
- 客体的敏感度标记称为密级（Classification Level）

- 强制存取控制规则

  (1)仅当主体的许可证级别大于或等于客体的密级时，该主体才能读取相应的客体

  (2)仅当主体的许可证级别小于或等于客体的密级时，该主体才能写相应的客体

规则的共同点：

- 禁止了拥有高许可证级别的主体更新低密级的数据对象
- 防止数据的密级从高流到低，造成数据泄漏

MAC与DAC

- **实现MAC时要首先实现DAC**
  - 原因：较高安全性级别提供的安全保护要包含较低级别的所有保护
- **DAC与MAC共同构成DBMS的安全机制**

4.1 数据库安全性控制

4.1.1 用户标识与鉴别

4.1.2 存取控制

4.1.3 数据库角色

- 数据库角色：被命名的一组与数据库操作相关的权限
  - 角色是权限的集合
  - 可以为一组具有相同权限的用户创建一个角色
  - 简化授权的过程

(一) 角色的创建

CREATE ROLE <角色名>

(二) 给角色授权

GRANT <权限> [, <权限>] ...

ON <对象类型>对象名

TO <角色> [, <角色>] ...

(三）将一个角色授予其他的角色或用户

GRANT <角色1>［，<角色2>］...

TO <角色3>［，<用户1>］...

[WITH ADMIN OPTION] -- 传播授权

(四）角色权限的收回

REVOKE <权限>［，<权限>］...

ON <对象类型> <对象名>

FROM <角色>［，<角色>］...

例9 通过角色来实现将一组权限授予一组用户。

1. 首先，创建一个角色R1

```
Create role R1;
```

2. 然后，使用Grant语句，使角色R1拥有Student表的Select、Insert和Delete权限。

```
Grant Select, Insert ,Delete On Student to R1;
```

3. 将这个角色授予王平，张明，赵玲。使他们具有角色R1所包含的全部权限。

Grant R1 to 王平，张明，赵玲；

4. 可以一次性地回收王平的这三个权限

Revoke R1 from 王平；

第四章　数据库安全性

4.1　数据库安全性控制

4.2　视图机制（略）

4.3　审计（Audit）

4.4　数据加密

4.2 视图机制

- 把要保密的数据对无权存取这些数据的用户隐藏起来，对数据提供一定程度的安全保护
  - 主要功能是提供数据独立性，无法完全满足要求
  - 间接实现了支持存取谓词的用户权限定义

例10 建立计算机系学生的视图，把对该视图的SELECT权限授于王平，把该视图上的所有操作权限授于张明

先建立计算机系学生的视图CS_Student

```
CREATE VIEW CS_Student
 AS
    SELECT  *
    FROM   Student
    WHERE  Sdept='CS';
```

在视图上进一步定义存取权限

```
GRANT  SELECT
ON  CS_Student
TO 王平；

GRANT ALL PRIVILIGES
ON  CS_Student
TO  张明;
```

4.3 审 计

- 什么是审计
  - 审计日志（Audit Log）
    将用户对数据库的所有操作记录在上面
  - DBA利用审计日志
    找出非法存取数据的人、时间和内容
  - C2以上安全级别的DBMS必须具有
  - 审计在SQLServer2005中没有

审计分为：

- 用户级审计
  - 针对自己创建的数据库表或视图进行审计
  - 记录所有用户对这些表或视图的一切成功和（或）不成功的访问要求以及各种类型的SQL操作
- 系统级审计
  - DBA设置
  - 监测成功或失败的登录要求
  - 监测GRANT和REVOKE操作以及其他数据库级权限下的操作

- AUDIT语句：设置审计功能
- NOAUDIT语句：取消审计功能

例11 对修改SC表结构或修改SC表数据的操作进行审计

```
AUDIT ALTER，UPDATE
ON  SC；
```

例12 取消对SC表的一切审计

```
NOAUDIT  ALTER，UPDATE
ON  SC；
```

注：SQLServer2005/2008不支持此语句。

4.4 数据加密

- 防止数据库中数据在存储和传输中失密的有效手段
- 加密的基本思想
  - 根据一定的算法将原始数据（术语为明文，Plain text）变换为不可直接识别的格式（术语为密文，Cipher text）
  - 不知道解密算法的人无法获知数据的内容
- 分类
  - 存储加密和传输加密

- 加密方法
  - 替换方法
    - 使用密钥（Encryption Key）将明文中的每一个字符转换为密文中的一个字符
  - 置换方法
    - 将明文的字符按不同的顺序重新排列
  - 混合方法
    - 美国1977年制定的官方加密标准：数据加密标准（Data Encryption Standard，简称DES）

- DBMS中的数据加密
  - 有些数据库产品提供了数据加密例行程序
  - 有些数据库产品本身未提供加密程序，但提供了接口
- 数据加密功能通常也作为可选特征，允许用户自由选择
  - 数据加密与解密是比较费时的操作
  - 数据加密与解密程序会占用大量系统资源
  - 应该只对高度机密的数据加密

本章小结

- 数据的共享日益加强，数据的安全保密越来越重要
- DBMS是管理数据的核心，因而其自身必须具有一整套完整而有效的安全性机制

- 实现数据库系统安全性的技术和方法
  - 存取控制技术
  - 视图技术
  - 审计技术
- 自主存取控制功能
  - 通过SQL 的GRANT语句和REVOKE语句实现
- 角色
  - 使用角色来管理数据库权限可以简化授权过程
  - CREATE ROLE语句创建角色
  - GRANT 语句给角色授权