数据库系统概论
An Introduction to Database System

第四章 数据库安全性

中国人民大学信息学院

第四章 数据库安全性

4.1 计算机安全性概述

4.2 数据库安全性控制

4.3 视图机制

4.4 审计（Audit）

4.5 数据加密

4.6 统计数据库安全性

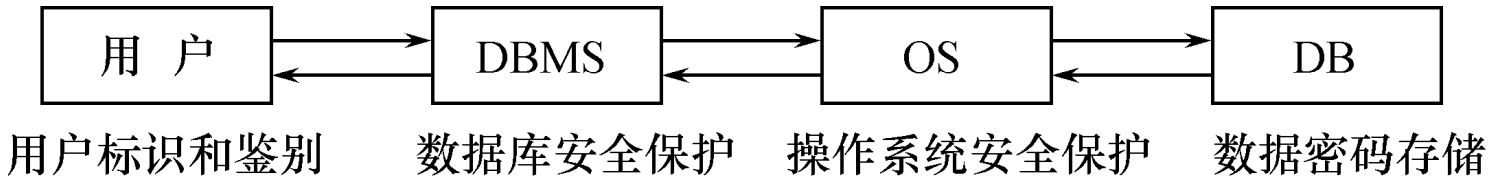
4.7 小结

4.2 数据库安全性控制概述

- 非法使用数据库的情况
  - 编写合法程序绕过 DBMS 及其授权机制
  - 直接或编写应用程序执行非授权操作
  - 通过多次合法查询数据库从中推导出一些保密数据

数据库安全性控制概述（续）

- 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

数据库安全性控制概述（续）

- 数据库安全性控制的常用方法
  - 用户标识和鉴定
  - 存取控制
  - 视图
  - 审计
  - 密码存储

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.1 用户标识与鉴别

- 用户标识与鉴别

  （Identification & Authentication）

  - 系统提供的最外层安全保护措施

用户标识与鉴别（续）

- 用户标识
- 口令
  - 系统核对口令以鉴别用户身份
- 用户名和口令易被窃取
  - 每个用户预先约定好一个计算过程或者函数

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.2 存取控制

- 存取控制机制组成
  - 定义用户权限
  - 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

存取控制（续）

- 常用存取控制方法
  - 自主存取控制（Discretionary Access Control ， 简称 DAC ）
    - C2 级
    - 灵活
  - 强制存取控制（Mandatory Access Control， 简称 MAC ）
    - B1 级
    - 严格

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.3 自主存取控制方法

- 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

自主存取控制方法（续）

- 关系数据库系统中存取控制对象

对象类型	对象	操 作 类 型
数据库	模式	CREATE SCHEMA
	基本表	CREATE TABLE， ALTER TABLE
模式	视图	CREATE VIEW
	索引	CREATE INDEX
数据	基本表和视图	SELECT， INSERT， UPDATE， DELETE， REFERENCES， ALL PRIVILEGES
数据	属性列	SELECT， INSERT， UPDATE， REFERENCES ALL PRIVILEGES

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.4 授权与回收

一、GRANT

- GRANT 语句的一般格式：

  GRANT < 权限 >[,< 权限 >]...

  [ON < 对象类型 > < 对象名 >]

  TO < 用户 >[,< 用户 >]...

  [WITH GRANT OPTION];

- 语义：将对指定操作对象的指定操作权限授予指定的用户

GRANT （续）

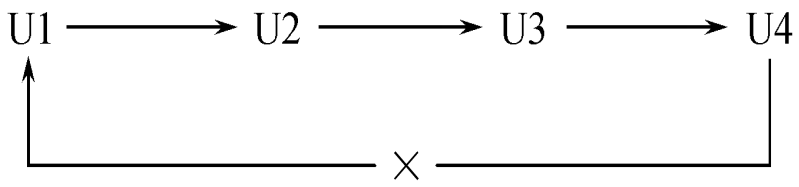
- 发出 GRANT：
  - DBA
  - 数据库对象创建者（即属主 Owner）
  - 拥有该权限的用户

- 按受权限的用户
  - 一个或多个具体用户
  - PUBLIC （全体用户）

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



例题

[例 1] 把查询 Student 表权限授给用户 U1

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

例题（续）

[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;
```

例题（续）

[例 3] 把对表 SC 的查询权限授予所有用户

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

例题（续）

[例 4]　把查询 Student 表和修改学生学号的权限授给用户 U4

```
GRANT UPDATE(Sno), SELECT
ON TABLE Student
TO U4;
```

- 对属性列的授权时必须明确指出相应属性列名

例题（续）

[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：

[例 6] GRANT INSERT ON TABLE SC TO U6
WITH GRANT OPTION;

同样， U6 还可以将此权限授予 U7 ：

[例 7] GRANT INSERT ON TABLE SC TO U7;

但 U7 不能再传播此权限。

传播权限（续）

下表是执行了［例 1 ］到［例 7 ］的语句后，学生 - 课程数据库中的用户权限定义表

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系 Student	SELECT	不能
DBA	U2	关系 Student	ALL	不能
DBA	U2	关系 Course	ALL	不能
DBA	U3	关系 Student	ALL	不能
DBA	U3	关系 Course	ALL	不能
DBA	PUBLIC	关系 SC	SELECT	不能
DBA	U4	关系 Student	SELECT	不能
DBA	U4	属性列 Student.Sno	UPDATE	不能
DBA	U5	关系 SC	INSERT	能
U5	U6	关系 SC	INSERT	能
U6	U7	关系 SC	INSERT	不能

授权与回收（续）

二、 REVOKE

- 授予的权限可以由 DBA 或其他授权者用 REVOKE 语句收回
- REVOKE 语句的一般格式为：

  REVOKE < 权限 >[,< 权限 >]...

  [ON < 对象类型 > < 对象名 >]

  FROM < 用户 >[,< 用户 >]...;

REVOKE （续）

[例 8] 把用户 U4 修改学生学号的权限收回

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

REVOKE （续）

[例 9] 收回所有用户对表 SC 的查询权限

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

REVOKE （续）

[例 10] 把用户 U5 对 SC 表的 INSERT 权限收回

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

- 将用户 U5 的 INSERT 权限收回的时候必须级联（ CASCADE ）收回
- 系统只收回直接或间接从 U5 处获得的权限

REVOKE （续）

执行［例 8 ］到［例 10 ］的语句后，学生 - 课程数据库中的用户权限定义表

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系 Student	SELECT	不能
DBA	U2	关系 Student	ALL	不能
DBA	U2	关系 Course	ALL	不能
DBA	U3	关系 Student	ALL	不能
DBA	U3	关系 Course	ALL	不能
DBA	U4	关系 Student	SELECT	不能

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

授权与回收（续）

三、创建数据库模式的权限

- DBA 在创建用户时实现

- CREATE USER 语句格式

  CREATE USER <username>

  [WITH] [DBA | RESOURCE | CONNECT]

授权与回收（续）

□□□□□	□□□□□□□			
	CREATE USER	CREATE SCHEMA	CREATE TABLE	□□□□□ □□□□□□□□□□
DBA	□□	□□	□□	□□
RESOURCE	□□□	□□□	□□□	□□□
CONNECT	□□□	□□□	□□□	□□□□□□□□□□□□

权限与可执行的操作对照表

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.5 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

数据库角色

- 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 >] …

数据库角色

- **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]

- **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 >] …

数据库角色（续）

[例 11]　通过角色来实现将一组权限授予一个用户。

步骤如下:

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 、 UPDATE 、 INSERT 权限

```
GRANT SELECT , UPDATE , INSERT
ON TABLE Student
TO R1 ;
```

数据库角色（续）

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. 可以一次性通过 R1 来回收王平的这 3 个权限

```
REVOKE  R1
FROM  王平;
```

数据库角色（续）

[例 12]　角色的权限修改

```
GRANT DELETE
ON TABLE Student
TO R1
```

数据库角色（续）

[例 13]

```
REVOKE SELECT
ON TABLE Student
FROM  R1 ;
```

4.2 数据库安全性控制

4.2.1 用户标识与鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

自主存取控制缺点

- 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

4.2.6 强制存取控制方法

- 强制存取控制（MAC)
  - 保证更高程度的安全性
  - 用户能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

强制存取控制方法（续）

- 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

强制存取控制方法（续）

- 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）

强制存取控制方法（续）

- 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等于客体的密级时，该主体才能写相应的客体

- 修正规则
  - 主体的许可证级别 <= 客体的密级 → 主体能写客体

强制存取控制方法（续）

- 规则的共同点

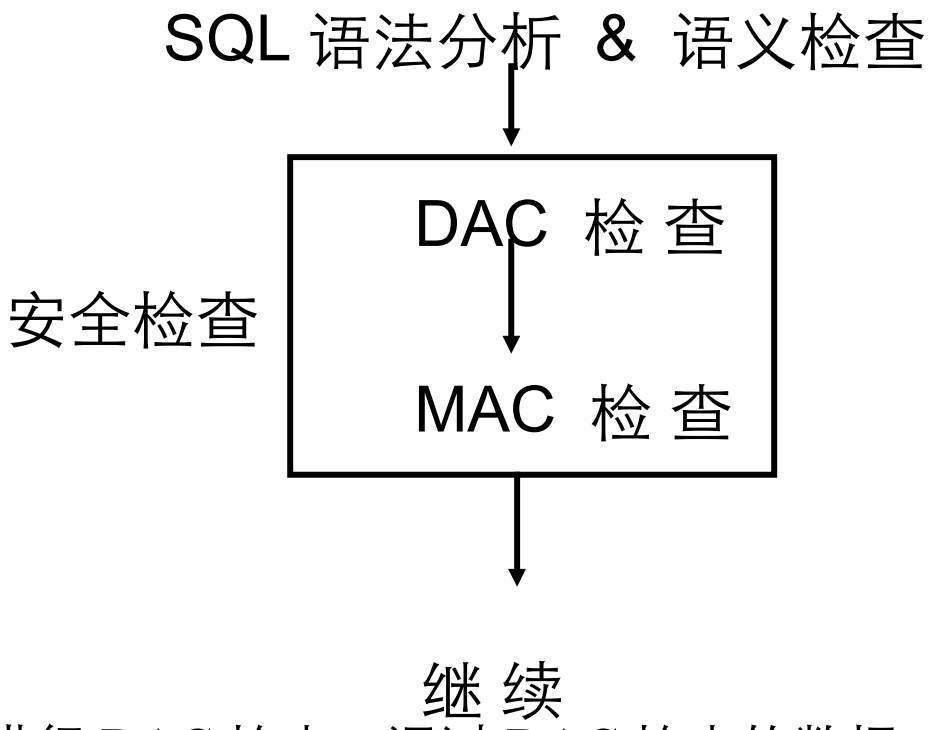
  禁止了拥有高许可证级别的主体更新低密级的数据对象

MAC 与 DAC

- DAC 与 MAC 共同构成 DBMS 的安全机制
- 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

强制存取控制方法（续）

DAC + MAC 安全检查示意图



- 先进行 DAC 检查，通过 DAC 检查的数据对象再由系统进行 MAC 检查，只有通过 MAC 检查的数据对象方可存取。

下课了。。。

认真

休息一会儿。。。