数据库系统概论

An Introduction to Database System

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（续）

- 数据库的安全性是指保护数据库以防止不合法使用所造成的数据泄露、更改或破坏 。
- 系统安全保护措施是否有效是数据库系统主要的性能指标之一。

第四章 数据库安全性

4.1 数据库安全性概述

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 其他安全性保护

4.7 小结

4.1.1 数据库的不安全因素

1.非授权用户对数据库的恶意存取和破坏

- 一些黑客（Hacker）和犯罪分子在用户存取数据库时猎取用户名和用户口令，然后假冒合法用户偷取、修改甚至破坏用户数据。
- 数据库管理系统提供的安全措施主要包括用户身份鉴别、存取控制和视图等技术。

数据库的不安全因素（续）

2.数据库中重要或敏感的数据被泄露

- 黑客和敌对分子千方百计盗窃数据库中的重要数据，一些机密信息被暴露。
- 数据库管理系统提供的主要技术有强制存取控制、数据加密存储和加密传输等。
- 审计日志分析

数据库的不安全因素（续）

3.安全环境的脆弱性

- 数据库的安全性与计算机系统的安全性紧密联系
  - 计算机硬件、操作系统、网络系统等的安全性
- 建立一套可信（Trusted）计算机系统的概念和标准

4.1 数据库安全性概述

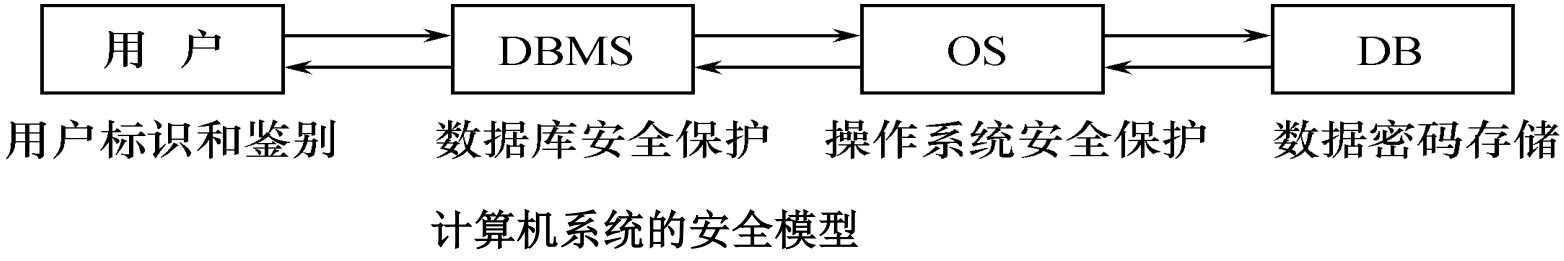
4.1.1 数据库的不安全因素

4.1.2 安全标准简介（自学）

第四章 数据库安全性

4.2 数据库安全性控制

❖ 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

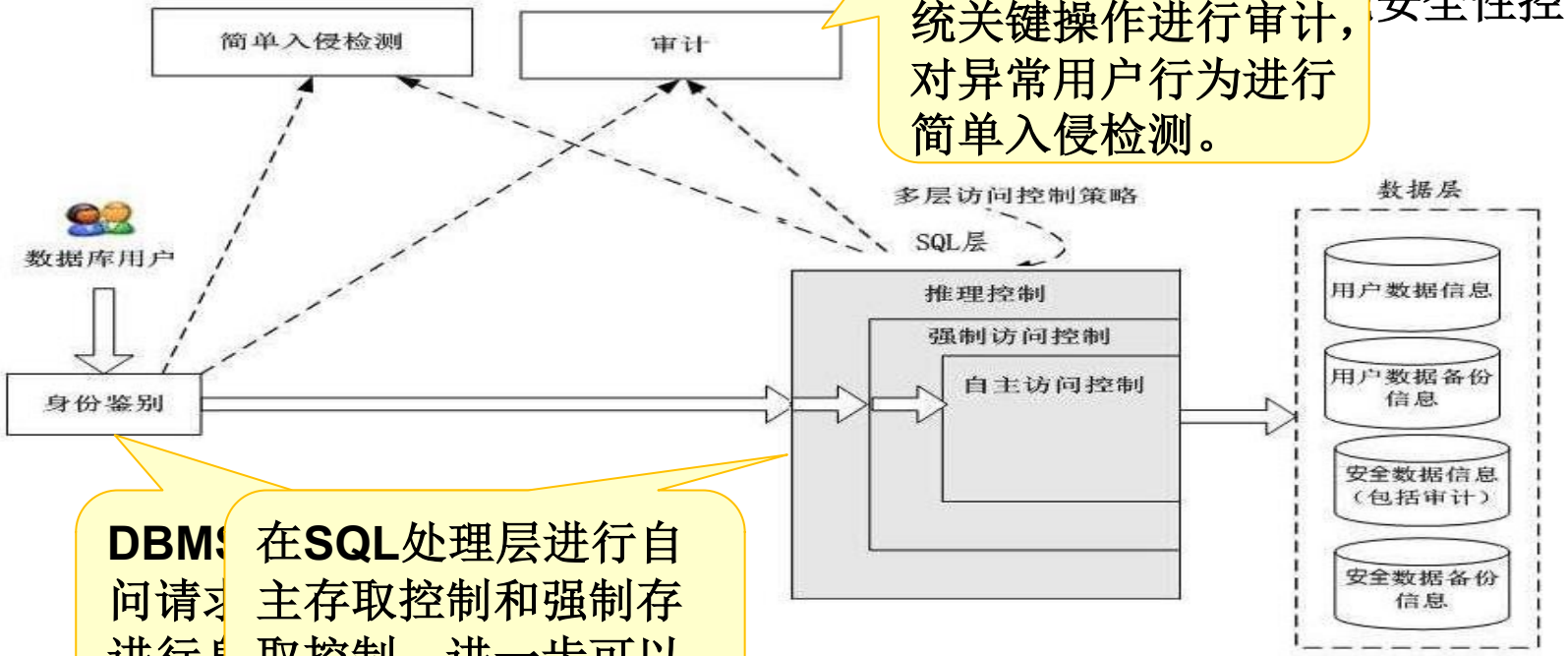
- 系统根据用户标识鉴定用户身份，合法用户才准许进入计算机系统
- 数据库管理系统进行存取控制，只允许用户执行合法操作
- 操作系统有自己的保护措施
- 数据以密码形式存储到数据库中

数据库安全性控制（续）

- 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
  - 数据加密

数据库安全性控制（续）

安全性控制模型



4.2 数据库安全性控制

4.2.1 用户身份鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权：授予与回收

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
  - 用户标识：由用户名和用户标识号组成
    （用户标识号在系统整个生命周期内唯一）

用户身份鉴别（续）

- 用户身份鉴别的方法

1.静态口令鉴别

- 静态口令一般由用户自己设定，这些口令是静态不变的

2.动态口令鉴别

- 口令是动态变化的，每次鉴别时均需使用动态产生的新口令登录数据库管理系统，即采用一次一密的方法

3.智能卡鉴别

- 智能卡是一种不可复制的硬件，内置集成电路的芯片，具有硬件加密功能

4.生物特征鉴别

- 通过生物特征进行认证的技术，生物特征如指纹、虹膜和掌纹等

4.2 数据库安全性控制

4.2.1 用户身份鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权：授予与回收

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，并将用户权限登记到数据字典中
    - 用户对某一数据对象的操作权力称为权限
    - DBMS提供适当的语言来定义用户权限，存放在数据字典中，称做安全规则或授权规则
  - 合法权限检查
    - 用户发出存取数据库操作请求
    - DBMS查找数据字典，进行合法权限检查

用户权限定义和合法权检查机制一起组成了DBMS的存取控制子系统

4.2 数据库安全性控制

4.2.1 用户身份鉴别

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权：授予与回收

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

- 自主存取控制（Discretionary Access Control，简称DAC）
  - 用户对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
  - 用户可‘自主’决定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

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	模式	CREATE SCHEMA
	基本表	CREATE TABLE，ALTER TABLE
	视图	CREATE VIEW
	索引	CREATE INDEX
数据	基本表和视图	SELECT，INSERT，UPDATE，DELETE，REFERENCES，ALL PRIVILEGES
	属性列	SELECT，INSERT，UPDATE， REFERENCES，ALL PRIVILEGES

关系数据库系统中的存取权限

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：授予与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

1. 权限授予：GRANT

- 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];

1. 权限授予：GRANT

❖ GRANT语句的一般格式

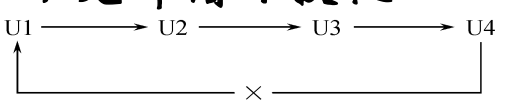
GRANT <权限>[,<权限>]...

ON <对象类型> <对象名>[,<对象类型> <对象名>]...

TO <用户>[,<用户>]...

[WITH GRANT OPTION];

不允许循环授权



权限

❖ 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
  - 数据库管理员
  - 数据库对象创建者（即属主Owner）
  - 拥有该权限、并能将该权限转授出去的用户
- 按受权限的用户
  - 一个或多个具体用户
  - PUBLIC（即全体用户）

例题

[例4.1] 把查询Student表权限授给用户U1

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

将一种权限授予一个用户。

例题（续）

[例4.2] 把对Student表和Course表的全部权限授予用户U2和U3

```
GRANT ALL PRIVILEGES
ON TABLE Student, Course
TO U2, U3;
```

一次向多个用户传播多种同类对象的权限。

例题（续）

[例4.3] 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4.4] 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

一次完成了对基本表和属性列这些不同对象的授权。

例题（续）

[例4.5] 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4.5后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限：

[例4.6] GRANT INSERT
ON TABLE SC
TO U6
WITH GRANT OPTION;

同样U6还可以将此权限授予U7

[例4.7] GRANT INSERT
ON TABLE SC
TO U7;

但U7不能再传播此权限。

GRANT（续）

执行了例4.1~例4.7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

2. 权限回收：REVOKE

- 授予的权限可以由数据库管理员或其他授权者用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>[,<对象类型><对象名>]…

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

2. 权限回收：REVOKE

- 授予的权限可以由数据库管理员或其他授权者用REVOKE语句收回

- REVOKE语句的一般格式为：

  REVOKE <权限>[,<权限>]...

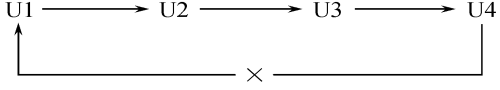
  ON <对象类型> <对象名>[,<对象类型><对象名>]...

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

CASCADE：级联回收

RESTRICT：受限回收

不允许循环授权



例题

[例4.8] 把用户U4修改学生学号的权限收回

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

例题（续）

[例4.9] 收回所有用户对表SC的查询权限

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

例题（续）

[例4.10] 把用户U5对SC表的INSERT权限收回

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

- 如果系统缺省值为RESTRICT，回收U5的INSERT权限时应该使用CASCADE短语，否则拒绝执行该语句
- 如果U6或U7还从其他用户处获得对SC表的INSERT权限，则他们仍具有此权限，系统只收回直接或间接从U5处获得的权限

REVOKE（续）

执行例4.8~4.10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

3.创建数据库模式的权限

❖关系数据库系统中的存取权限

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 INDEX
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.创建数据库模式的权限

❖关系数据库系统中的存取权限

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 INDEX
数据	基本表和视图	SELECT，INSERT，UPDATE，DELETE，REFERENCES，ALL PRIVILEGES
	属性列	SELECT，INSERT，UPDATE， REFERENCES，ALL PRIVILEGES

GRANT/REVOKE

3.创建数据库模式的权限

❖关系数据库系统中的存取权限

数据库管理员在创建用户时实现

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 INDEX
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.创建数据库模式的权限

❖ CREATE USER语句格式

CREATE USER <username>

[WITH][DBA|RESOURCE|CONNECT];

注：CREATE USER不是SQL标准，各个系统的实现相差甚远

创建数据库模式的权限（续）

拥有的权限	可否执行的操作			
	CREATE USER	CREATE SCHEMA	CREATE TABLE	登录数据库，执行数据查询和操纵
DBA	可以	可以	可以	可以
RESOURCE	不可以	不可以	可以	可以
CONNECT	不可以	不可以	不可以	可以，但必须拥有相应权限

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：授予与回收

4.2.5 数据库角色

4.2.6 强制存取控制方法

数据库角色的引入

例：

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U1, U2;
GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U1, U2;
GRANT SELECT, INSERT
ON TABLE SC
TO U1, U2;
```

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U3;
GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U3;
GRANT SELECT, INSERT
ON TABLE SC
TO U3;
```

数据库角色的引入

例：

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U4;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U4;

GRANT SELECT, INSERT
ON TABLE SC
TO U4;
```

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U3;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U3;

GRANT SELECT, INSERT
ON TABLE SC
TO U3;
```

数据库角色的引入

例：

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U4;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U4;

GRANT SELECT, INSERT
ON TABLE SC
TO U4;
```

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U5;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U5;

GRANT SELECT, INSERT
ON TABLE SC
TO U5;
```

什么是数据库角色

- 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

什么是数据库角色（续）

例：

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U4;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U4;

GRANT SELECT, INSERT
ON TABLE SC
TO U4;
```

```
GRANT SELECT, UPDATE, INSERT
ON TABLE Student
TO U5;

GRANT SELECT, UPDATE(Ccredit)
ON TABLE Course
TO U5;
```

使用角色来管理数据库权限，可以简化授权和回收的过程。

使用角色管理数据库权限

1.角色的创建

CREATE ROLE <角色名>

2.给角色授权

GRANT <权限>[,<权限>]…

ON <对象类型>对象名

TO <角色>[,<角色>]…

使用角色管理数据库权限（续）

3.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]

一个角色的权限：直接授予这

指定**WITH ADMIN OPTION**，则获得权限的角色或用户还可以把这种权限授予其他角色

- 授予者是角色的创建者或拥有在这个角色上的**ADMIN OPTION**

使用角色管理数据库权限（续）

4.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>]…

- 用户可以回收角色的权限，从而修改角色拥有的权限
- **REVOKE**执行者是
  - 角色的创建者
  - 拥有在这个（些）角色上的**ADMIN OPTION**

例题

[例4.11] 通过角色来实现权限管理。

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;
```

例题（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;
```

例题（续）

[例4.12] 增加角色的权限

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

使角色R1在原来的基础上增加了Student表的DELETE 权限

例题（续）

[例4.13]　减少角色的权限

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

使R1减少了SELECT权限

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”
- 例: 只有财务人员有权访问职工工资表EMP-Salary

```
CREATE TABLE Salary-copy
AS SELECT  Eno, Name, Salary
FROM EMP-Salary;

Grant SELECT
ON TABLE Salary-copy
TO PUBLIC;
```

自主存取控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

职工工资信息被泄露！

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）

- 在强制存取控制中，数据库管理系统所管理的全部实体被分为主体和客体两大类
- 主体是系统中的活动实体
  - 数据库管理系统所管理的实际用户
  - 代表用户的各进程
- 客体是系统中的被动实体，受主体操纵
  - 文件、基本表、索引、视图

强制存取控制方法（续）

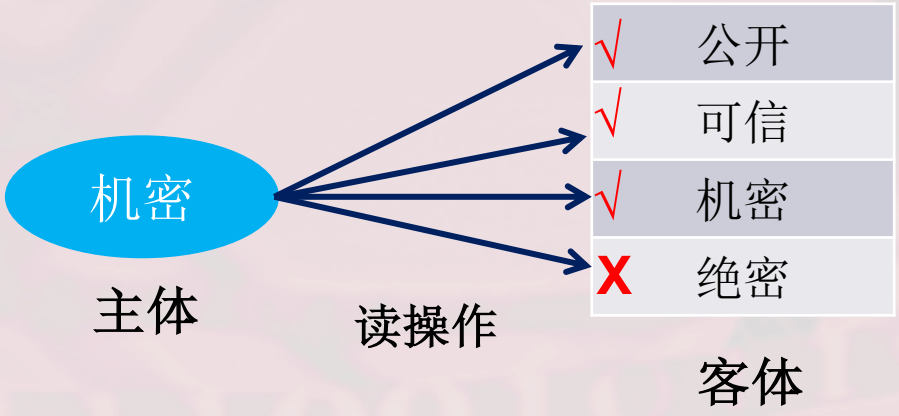
- 敏感度标记（Label）
  - 对于主体和客体，DBMS为它们每个实例（值）指派一个敏感度标记（Label）
  - 敏感度标记分成若干级别
    - 绝密（Top Secret，TS）
    - 机密（Secret，S）
    - 可信（Confidential，C）
    - 公开（Public，P）
    - TS>=S>=C>=P

- 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

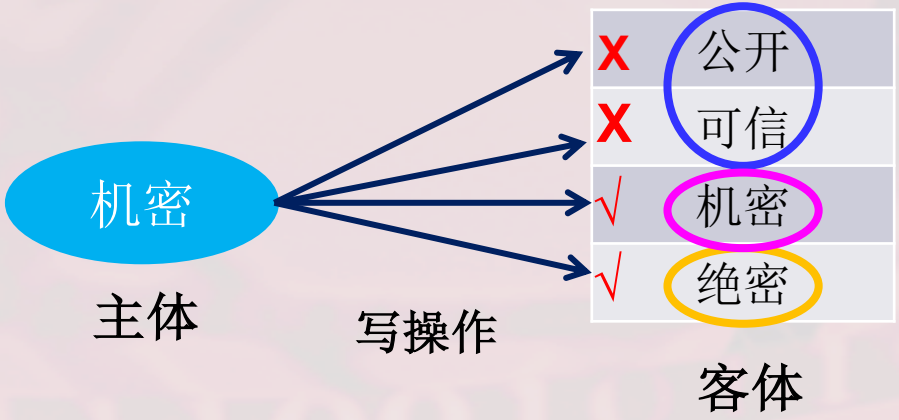
（1）仅当主体的许可证级别大于或等于客体的密级时，该主体才能读取相应的客体



强制存取控制方法（续）

❖ 强制存取控制规则

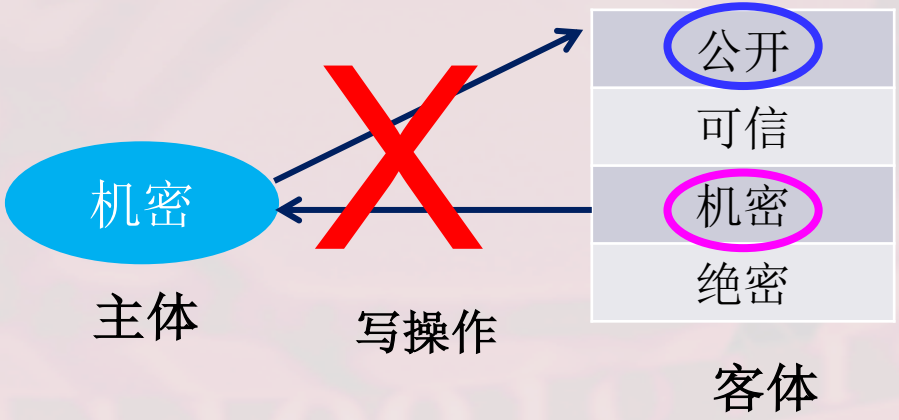
（2）仅当主体的许可证级别小于或等于客体的密级时，该主体才能写相应的客体



强制存取控制方法（续）

❖ 强制存取控制规则

（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



强制存取控制方法（续）

- 强制存取控制是对数据本身进行密级标记，无论数据如何复制，标记与数据是一个不可分的整体，只有符合密级标记要求的用户才可以操纵数据。

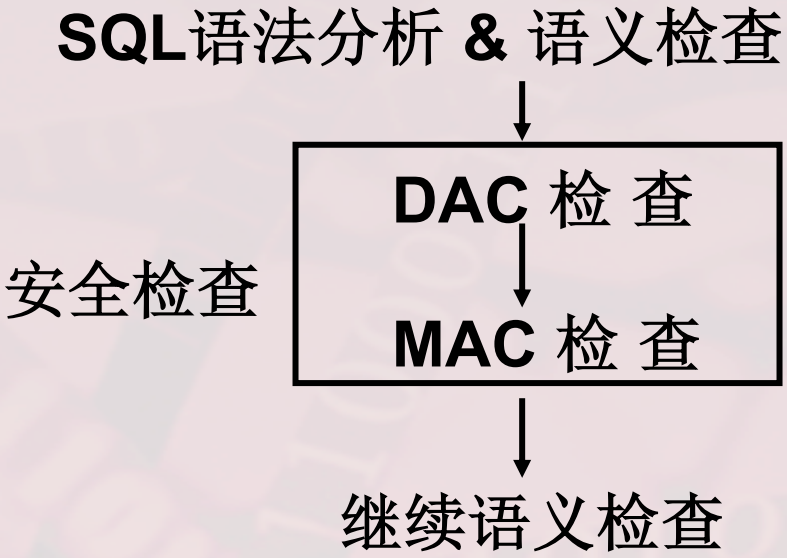
```
CREATE TABLE Salary-copy
AS SELECT  Eno, Name, Salary
FROM EMP-Salary;

Grant SELECT
ON TABLE Salary-copy
TO PUBLIC;
```

财务人员A：　绝密
EMP-Salary:　绝密数据
Salary-copy:　绝密数据

→ 许可证级别不是绝密的用户仍然不能访问Salary-copy表

DAC + MAC安全检查

- 实现强制存取控制MAC时要首先实现自主存取控制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
- 自主存取控制DAC与强制存取控制MAC共同构成数据库管理系统的安全机制



小结

- 数据库角色
  - 创建角色
  - 给角色授权
  - 将角色授予用户或其他角色
  - 角色权限的回收
- 强制存取控制
  - 为主体的每个实例指派一个许可证级别
  - 为客体的每个实例指派一个密级
  - 通过许可证级别与密级间匹配关系进行存取控制

第四章 数据库安全性

4.3 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

视图机制（续）

- 授予用户查询整个表的权限

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

- 授予用户查询某些列的权限

```
GRANT SELECT(Sno, Sname)
ON TABLE Student
TO U2;
```

- 授予用户查询某些行的权限？
  - 需要用存取谓词来定义用户权限
  - 无法直接用**GRANT**语句实现
  - 可以用视图机制间接地实现

视图机制（续）

[例4.14] 授权王平老师能查询计算机系学生的情况，授权系主任张明能对计算机系学生的信息进行所有操作。

(1) 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';
```

视图机制（续）

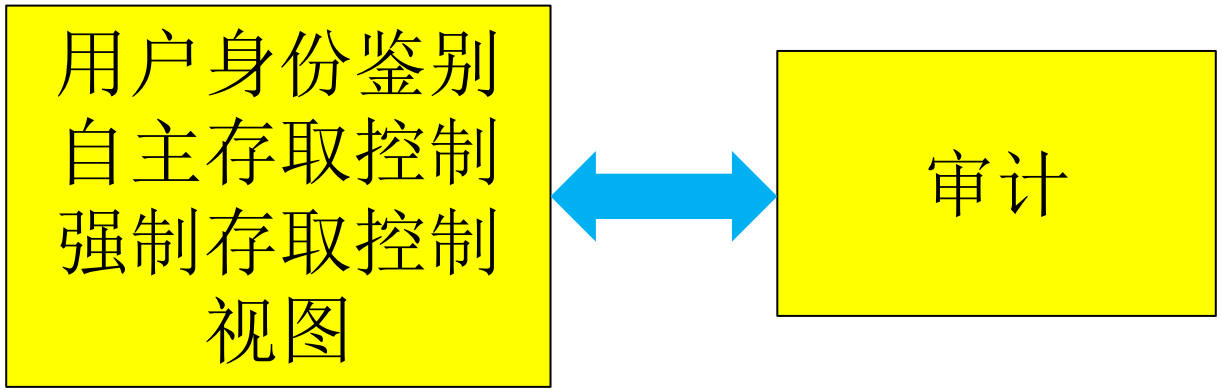
(2) 在视图上进一步定义存取权限

```
GRANT  SELECT
ON  CS_Student
TO 王平;

GRANT ALL PRIVILIGES
ON  CS_Student
TO  张明;
```

第四章 数据库安全性

4.1 数据库安全性概述
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 其他安全性保护
4.7 小结

数据库安全性控制措施



预防性措施　　监控措施

4.4 审计

❖什么是审计

- 启用一个专用的审计日志（Audit Log）
  将用户对数据库的所有操作记录在上面
- 审计员利用审计日志
  监控数据库中的各种行为
  发现非法存取，发现潜在威胁
- C2以上安全级别的DBMS必须具有审计功能

第四章 数据库安全性

4.7 小结

- 实现数据库系统安全性的技术和方法
  - 用户身份鉴别
  - 存取控制技术：自主存取控制和强制存取控制
  - 视图技术
  - 审计技术
  - 数据加密：加密存储和加密传输

小结（续）

- 本章目标
  - 掌握什么是数据库的安全性问题
  - 牢固掌握数据库管理系统实现数据库安全性控制的常用方法和技术
- 本章重点
  - 使用GRANT 语句和 REVOKE 语句实现自主存取控制功能
  - 使用CREATE ROLE语句创建角色，用GRANT 语句给角色授权
  - 掌握视图机制在数据库安全保护中的作用