数据库系统原理

李瑞轩

华中科技大学计算机学院

第四章
数据库安全性

Database security refers to protecting a database from data leakage, alteration, or destruction caused by illegal use

- 学习内容

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

4.1 计算机安全性概述

- 1. 定义
  - 数据库的安全性是指保护数据库，以防止不合法的使用所造成的数据泄露、更改或破坏

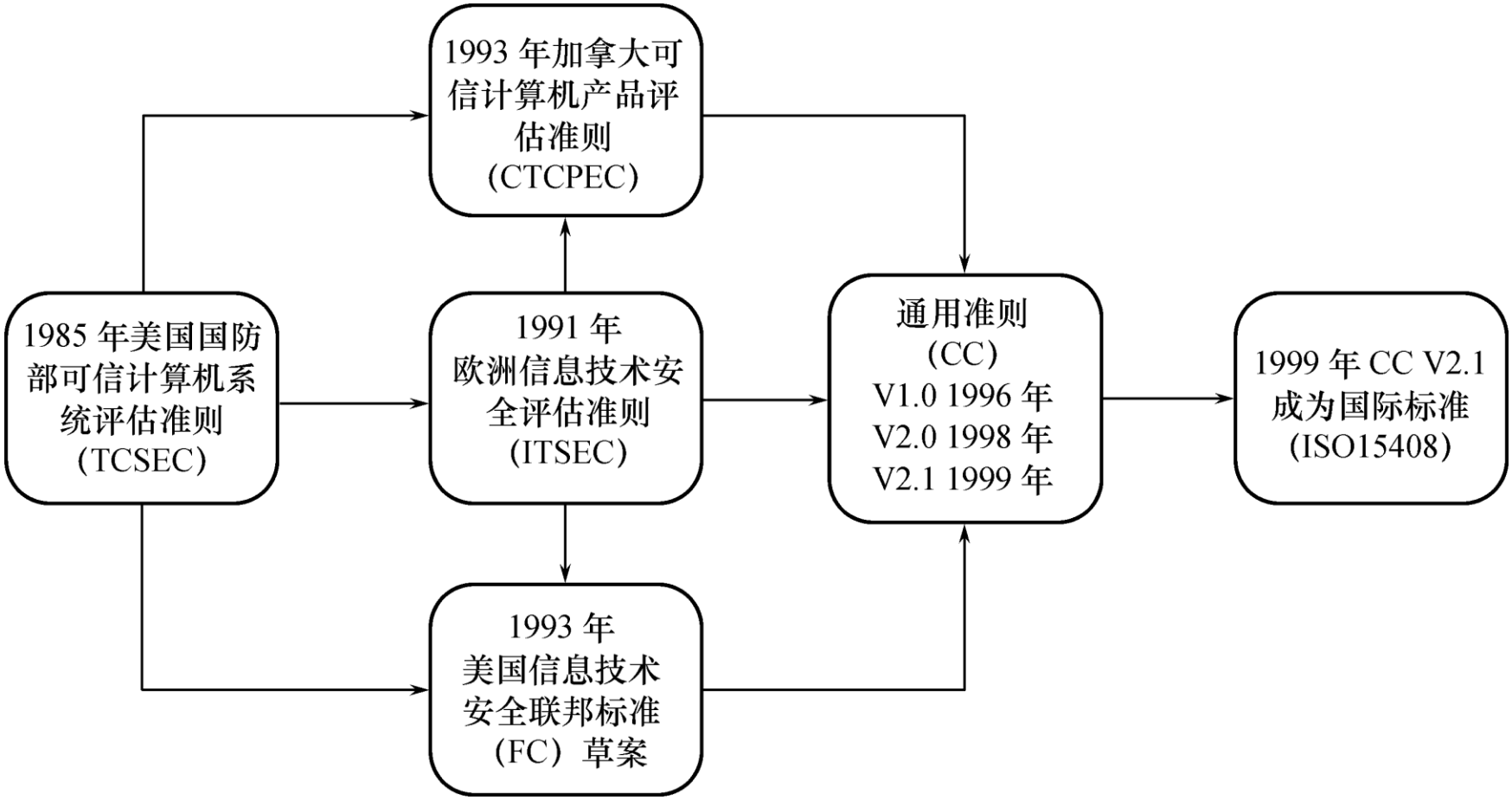
- 2. 重要性
  - 数据库系统中大量数据集中存放，许多用户直接共享
  - 系统安全保护措施是否有效是数据库系统的主要性能指标之一
  - 自然安全性＋系统安全性

4.1 计算机安全性概述(续)

- 3. 计算机系统三类安全性问题
  - 技术安全
  - 管理安全
  - 政策法律类

- 4. 计算机系统安全标准
  - 与信息安全标准有关的组织 (DoD, NCSC)
  - 国际信息系统安全标准 (TCSEC, CC)
  - 国内信息系统安全标准

安全标准简介



信息安全标准的发展历史

TCSEC/TDI安全级别划分

安全级别	定　　义
A1	验证设计（Verified Design）
B3	安全域（Security Domains）
B2	结构化保护（Structural Protection）
B1	标记安全保护（Labeled Security Protection）
C2	受控的存取保护（Controlled Access Protection）
C1	自主安全保护（Discretionary Security Protection）
D	最小保护（Minimal Protection）

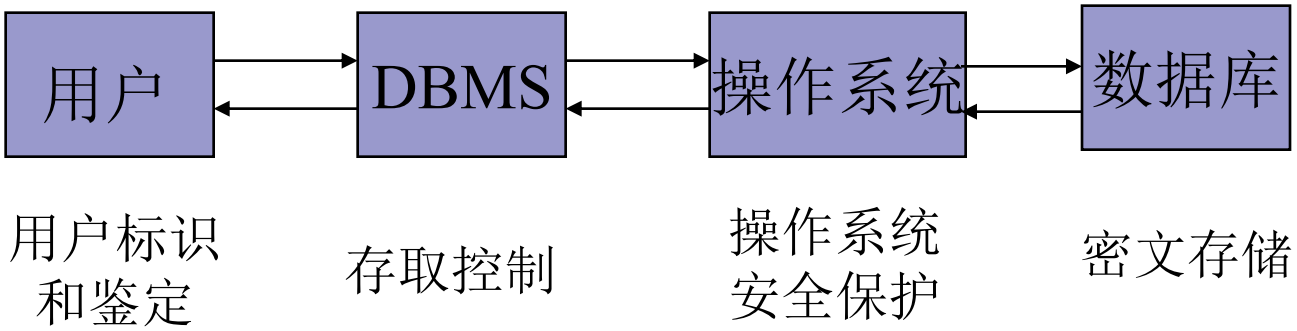
CC (Common Criteria)

- CC评估保证级划分

评估保证级	定　　义	TCSEC安全级别（近似相当）
EAL1	功能测试（functionally tested）	
EAL2	结构测试（structurally tested）	C1
EAL3	系统地测试和检查（methodically tested and checked）	C2
EAL4	系统地设计、测试和复查（methodically designed， tested， and reviewed）	B1
EAL5	半形式化设计和测试（semiformally designed and tested）	B2
EAL6	半形式化验证的设计和测试（semiformally verified design and tested）	B3
EAL7	形式化验证的设计和测试（formally verified design and tested）	A1

4.2 数据库安全性控制

- 4.2.1 用户标识和鉴别
- 4.2.2 存取控制
- 4.2.3 自主存取控制(DAC)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(MAC)方法



4.2.1 用户标识和鉴别 (Identification & Authentication)

- 定义
  - 系统提供一定的方式让用户标识自己的名字和身份，系统进行核实，通过鉴定后才提供系统使用权

- 常用方法
  - 用户标识证实(用户名/口令)
  - 过程(或函数)识别
  - 上机密码卡
  - 指纹、声音、照片(刷脸)、虹膜等识别
  - 回答问题

4.2.2 存取控制 (Access Control)

- 1. 存取控制概述
  - 对于获得上机权的用户还要根据系统预先定义好的外模式（视图）或用户权限进行存取控制，保证用户只能存取他有权存取的数据 (也叫访问控制)

- 2. 方法
  - 定义用户权限
  - 合法权限检查

4.2.2 存取控制 (Access Control)

- 3. 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

4.2.3 自主存取控制(DAC)方法

- 自主存取控制(DAC)
  - 用户权限
    - 数据对象
    - 操作类型
  - 授权语句
    - GRANT
    - REVOKE

4.2.3 自主存取控制(DAC)方法

- 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	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

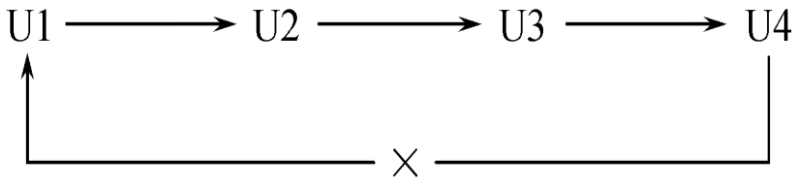
4.2.4 授权(Authorization)与回收

- 1. GRANT命令格式

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>|PUBLIC]...
[WITH GRANT OPTION];

WITH GRANT OPTION表示获得权限的用户可以把权限再授予其他用户

- 指定：可以再授予
- 没有指定：不能传播
- 不允许循环授权



GRANT（续）

- 发出GRANT授权
  - 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）

GRANT（续）

Ex 1: 把查询Student表权限授给用户A

GRANT SELECT ON TABLE Student TO A;

Ex2: 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;

Ex3: 把对表SC的查询权限授予所有用户

GRANT SELECT ON TABLE SC TO PUBLIC;

GRANT（续）

Ex 4: 把查询SC表和修改学生成绩的权限授给用户U4

```
GRANT UPDATE(Score), SELECT
ON TABLE SC
TO U4;
```

注：对属性列的授权时必须明确指出相应属性列名

Ex 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;
```

2. REVOKE

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>| public] …

[**CASCADE** | **RESTRICT**]…

CASCADE：收回权限时，若该用户已将权限授予其它用户，则也一并收回；否则，拒绝执行（RESTRICT）。

- Ex 6:把用户U4修改学生学号的权限收回

REVOKE *UPDATE(sno)* ***ON*** *TABLE Student* ***FROM*** *U4*

参见教材P141-142例8，9，10以及相关说明

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

3. 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]

创建数据库模式的权限（续）

权限与可执行的操作对照表

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

4.2.5 数据库角色 (Role)

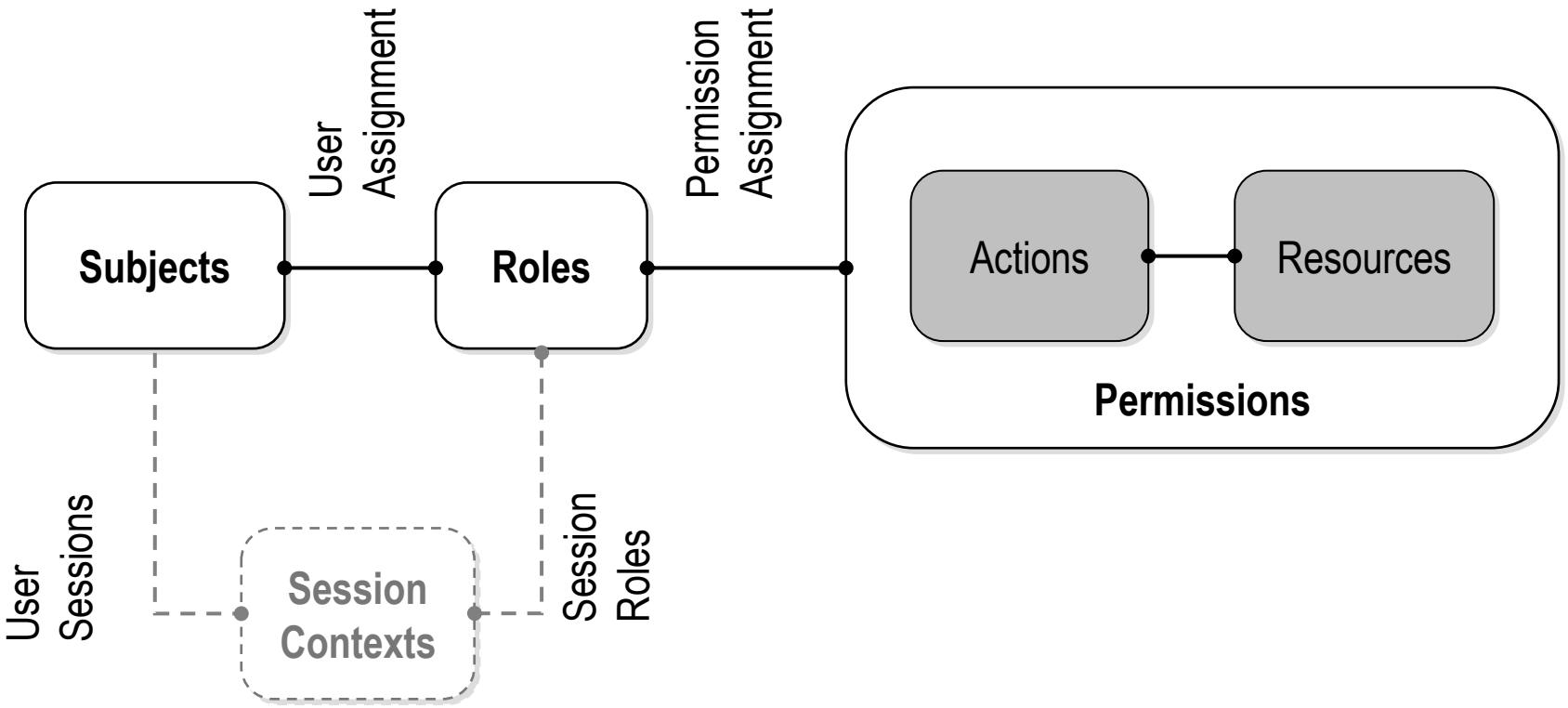
- 数据库角色：被命名的一组与数据库操作相关的权限
  - 角色是权限的集合
  - 可以为一组具有相同权限的用户创建一个角色
  - 简化授权的过程

基于角色的访问控制（Role Based Access Control，RBAC）

- 起因：Intranet的广泛应用使网上信息的完整性要求超过了机密性，而传统的DAC和MAC策略难以提供这方面的支持，于是基于角色的访问控制被提出并被广泛接受。
- 驱动：RBAC 发展的动力是在简化安全策略管理的同时，允许灵活地定义安全策略。

  From O(mn) to O(m+n), where m is the number of users and n is the number of permissions.
- 应用：目前，RBAC 被应用在各个企业领域，包括操作系统、数据库管理系统、PKI、工作流管理系统和Web服务等领域。

基于角色的访问控制（Role Based Access Control，RBAC）



数据库角色

- **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>]…

数据库角色

- **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］

  获得了某种权限的角色或用户还可以将这种权限再授予其他角色。

- **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>］…

数据库角色（续）

[例1]　通过角色来实现将一组权限授予一个用户。

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、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2]　角色的权限修改

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

[例3]

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

WITH ADMIN OPTION和 WITH GRANT OPTION 的区别

- WITH ADMIN OPTION 用于系统权限或角色授权， WITH GRANT OPTION 用于对象授权。
- 给一个用户授予系统权限带上WITH ADMIN OPTION时，此用户可把此系统权限授予其他用户或角色，但收回这个用户的系统权限时，这个用户已经授予其他用户或角色的此系统权限不会因传播无效。
  - 如授予A系统权限create session with admin option, 然后A又把create session权限授予B, 但管理员收回A的create session权限时，B依然拥有create session的权限，但管理员可以显式收回B create session的权限，即直接Revoke create session from B.
- 通过GRANT语句可以将系统权限和角色授予用户或者角色。但是必须满足如下的条件：
  - 对于系统权限，执行用户需要具有 GRANT ANY PRIVILEGE权限或者具有被授予的权限（WITH ADMIN OPTION）
  - 对于角色，执行用户需要具有GRANT ANY ROLE系统权限或者具有被授予角色（ WITH ADMIN OPTION ）

WITH ADMIN OPTION和 WITH GRANT OPTION 的区别

- 给一个用户授予对象权限带上WITH GRANT OPTION时，被授予的用户也可把此对象权限授予其他用户或角色，不同的是但管理员收回用WITH GRANT OPTION授权的用户对象权限时，权限会因传播而失效。
  - 如：grant select on 表名 to A with grant option;，A用户把此权限授予B，但管理员收回A的权限时，B的权限也会失效，但管理员不可以直接收回B的SELECT ON TABLE 权限。
- 使用WITH GRANT OPTION时
  - 必须是对象的拥有者，或者具有grant any object privilege系统权限，或者具有相应权限（with grant option）
  - 如果用户具有某表的对象权限（with grant option），并依附该表创建视图，那么该用户可以授权视图的权限给其他用户
  - 当我们将对象权限（with grant option）赋予某角色，拥有该角色的用户不会继承或者传播WITH GRANT OPTION

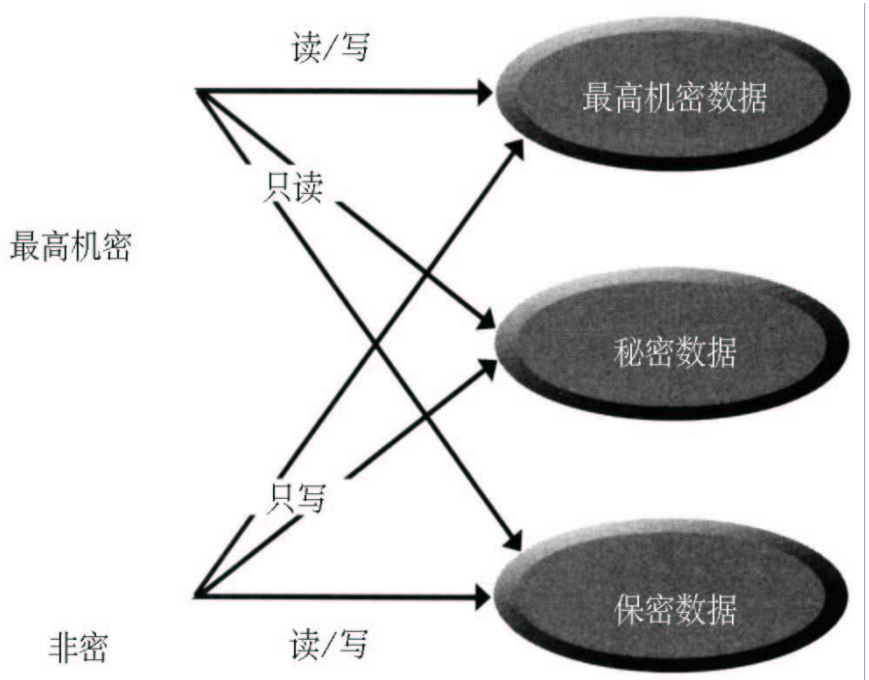
4.2.6 强制存取控制(MAC)方法

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

强制存取控制(MAC)

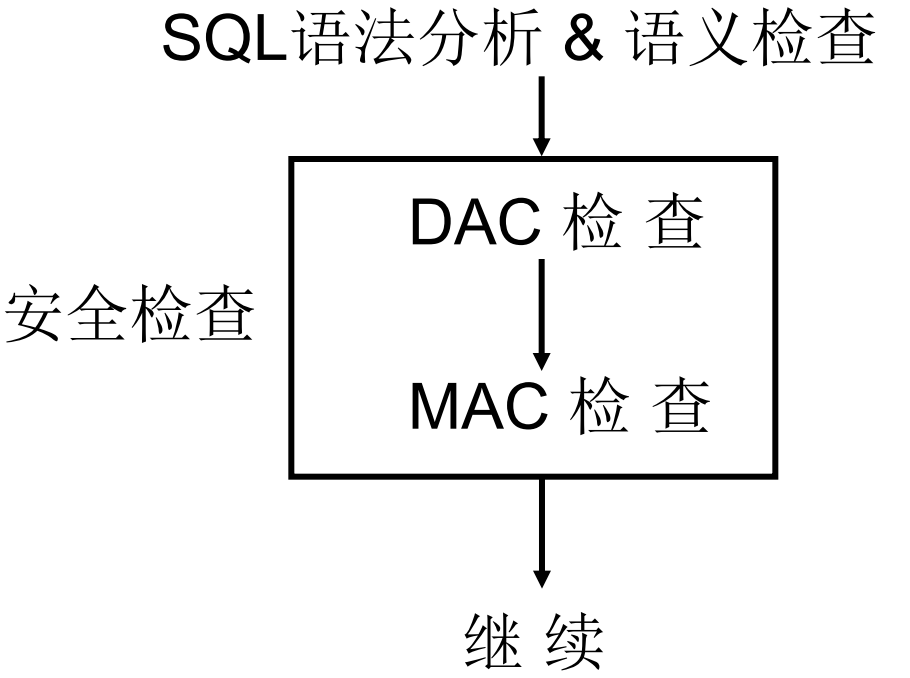
- 实体类别
  - 主体
  - 客体
- 敏感度标记
  - 绝密、机密、可信、公开
  - 许可证级别(主体)
  - 密级(客体)



向上写，向下读

强制存取控制(MAC)

DAC与MAC共同构成DBMS的安全机制



- 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。

4.3 视图机制

- 视图就象架设在用户与底层表之间的一道桥梁，用户只能对视图进行操作，而无法直接访问底层基本表。

```
CREATE VIEW VIEW_1
        AS SELECT COLUMN_1 FROM TABLE_1;

GRANT ALL ON VIEW_1 TO USER_2;
```

- USER_2通过VIEW_1可以访问COLUMN_1而无法访问COLUMN_2，这就是VIEW_1的屏障作用。

4.4 审计 (Audit)

- 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 (Audit)

- 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

审计 (Audit)

- AUDIT语句：设置审计功能
- NOAUDIT语句：取消审计功能

［例1］对修改SC表结构或修改SC表数据的操作进行审计

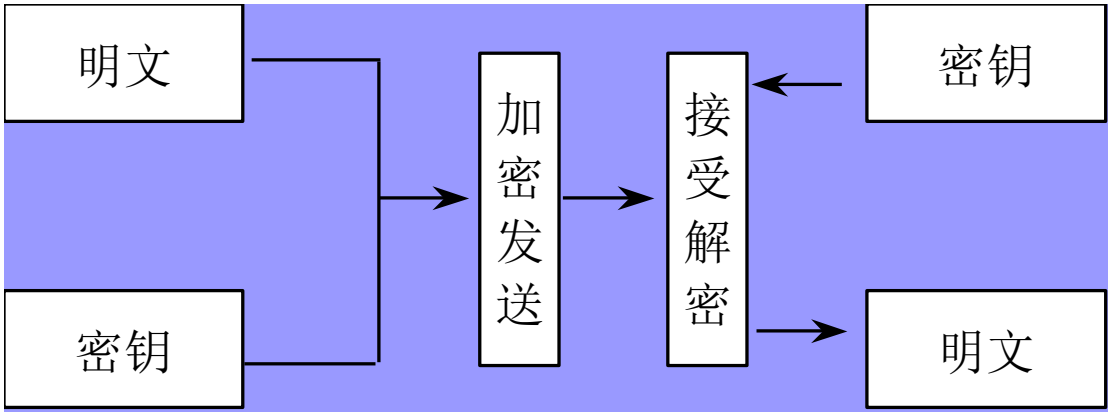
```
AUDIT ALTER，UPDATE
ON SC；
```

［例2］取消对SC表结构或修改SC表数据的操作进行审计

```
NOAUDIT ALTER，UPDATE
ON SC；
```

4.5 数据加密

- 思想
  - 数据库中的数据以密文形式存放，使用时由用户设计的解码程序将其转换成用户可读的数据。这样，数据库中的数据即使被窃取，也只能是一些无法辨认的代码。
- 加密算法
  - DES
  - AES



- 数据库密码系统的基本流程

4.6 统计数据库安全性

- 统计数据库
  - 允许用户查询聚集类型的信息（如合计、平均值等）
  - 不允许查询单个记录信息

- 统计数据库中特殊的安全性问题
  - 隐蔽的信息通道
  - 能从合法的查询中推导出不合法的信息

统计数据库安全性

规则1：任何查询至少要涉及N(N足够大)个以上的记录

规则2：任意两个查询的相交数据项不能超过M个

规则3：任一用户的查询次数不能超过1+(N-2)/M

数据库安全机制的设计目标：

试图破坏安全的人所花费的代价 >> 得到的利益

4.7 小结

- 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

下课了。。。

认真

休息一会儿。。。