# 第四章 数据库安全性

# 4.1 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

# 4.1 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。

# 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

# 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。

## 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

## 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

# 4.1.2　安全标准简介

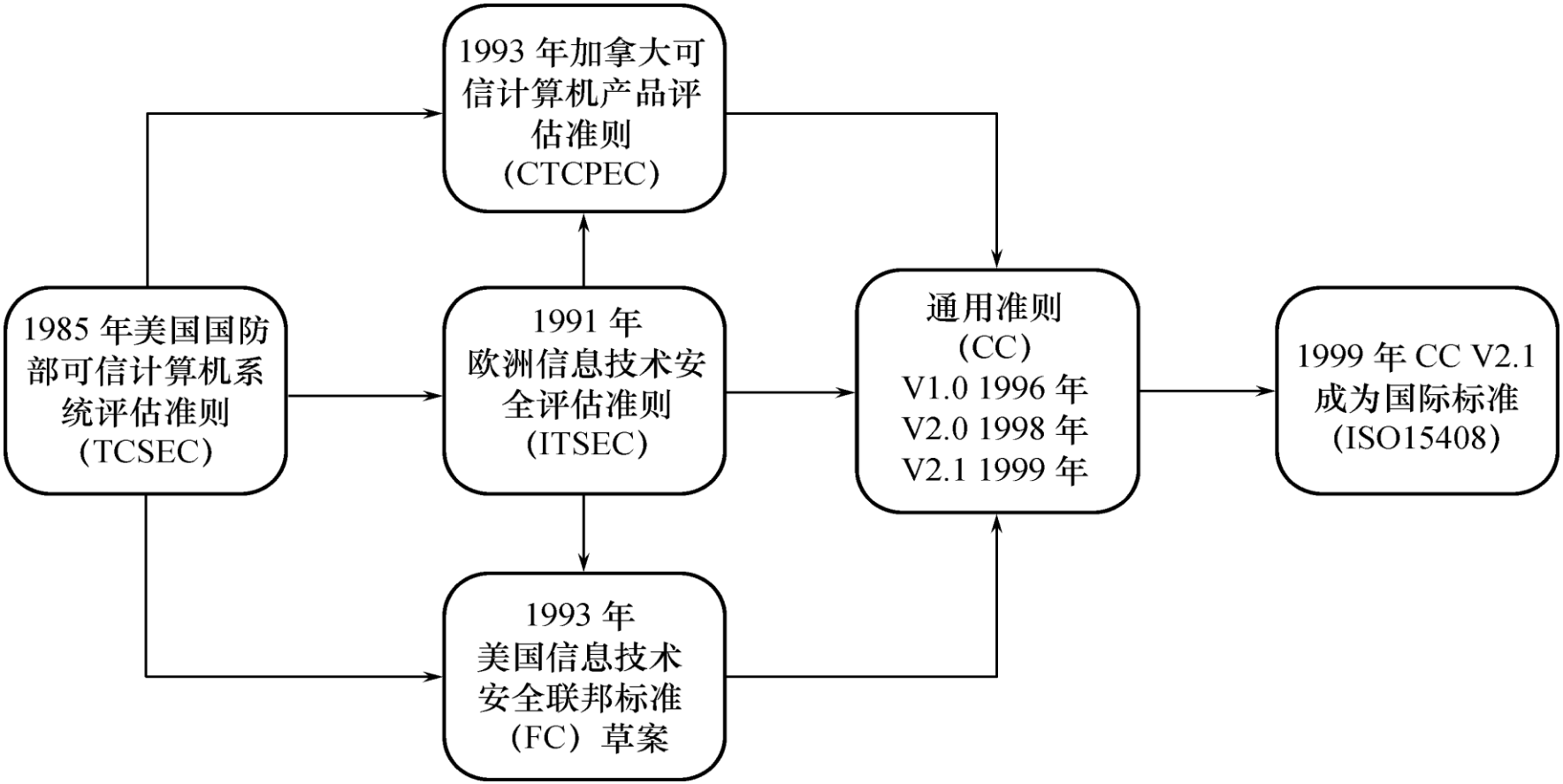
1985年美国国防部（DoD）正式颁布《DoD可信计算机系统评估准则》（简称TCSEC或DoD85）

不同国家建立在TCSEC概念上的评估准则

- 欧洲的信息技术安全评估准则（ITSEC）
- 加拿大的可信计算机产品评估准则（CTCPEC）
- 美国的信息技术安全联邦标准（FC）

- 1993年，CTCPEC、FC、TCSEC和ITSEC联合行动，解决原标准中概念和技术上的差异，称为CC（Common Criteria）项目
- 1999年 CC V2.1版被ISO采用为国际标准
- 2001年 CC V2.1版被我国采用为国家标准
- 目前CC已基本取代了TCSEC，成为评估信息产品安全性的主要标准。

# 安全标准简介（续）




信息安全标准的发展历史

# 安全标准简介（续）

**1. TCSEC标准**

**2. CC标准(自学)**

# TCSEC标准

1991年4月美国NCSC（国家计算机安全中心）颁布了《可信计算机系统评估标准关于可信数据库系统的解释》（ Trusted Database Interpretation 简称TDI）

- TDI又称紫皮书。它将TCSEC扩展到数据库管理系统
- TDI中定义了数据库管理系统的设计与实现中需满足和用以进行安全性级别评估的标准

# TCSEC/TDI安全级别划分

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）** |

# TCSEC/TDI安全级别划分（续）

- 四组（division）七个等级

  D

  C（C1，C2）

  B（B1，B2，B3）

  A（A1）

- 按系统可靠或可信程度逐渐增高
- 各安全级别之间具有一种偏序向下兼容的关系，即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要求，同时提供更多或更完善的保护能力

## D级

- 将一切不符合更高标准的系统均归于D组
- 典型例子：DOS是安全标准为D的操作系统

  DOS在安全性方面几乎没有什么专门的机制来保障

## C1级

- 非常初级的自主安全保护
- 能够实现对用户和数据的分离，进行自主存取控制（DAC），保护或限制用户权限的传播。
- 现有的商业系统稍作改进即可满足

## C2级

- 安全产品的最低档次
- 提供受控的存取保护，将C1级的DAC进一步细化，以个人身份注册负责，并实施审计和资源隔离
- 达到C2级的产品在其名称中往往不突出“安全”（Security）这一特色
- 典型例子
  - ✓Windows 2000
  - ✓ Oracle 7

- B1级
  - 标记安全保护。“安全”（Security）或“可信的”（Trusted）产品。
  - 对系统的数据加以标记，对标记的主体和客体实施强制存取控制（MAC）、审计等安全机制
  - B1级典型例子
    - ✓ 操作系统
      - 惠普公司的HP-UX BLS release 9.09+
    - ✓ 数据库
      - Oracle公司的Trusted Oracle 7
      - Sybase公司的Secure SQL Server version 11.0.6

- B2级
  - 结构化保护
  - 建立形式化的安全策略模型并对系统内的所有主体和客体实施DAC和MAC
- B3级
  - 安全域
  - 该级的TCB必须满足访问监控器的要求，审计跟踪能力更强，并提供系统恢复过程
- A1级
  - 验证设计，即提供B3级保护的同时给出系统的形式化设计说明和验证以确信各安全保护真正实现。

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

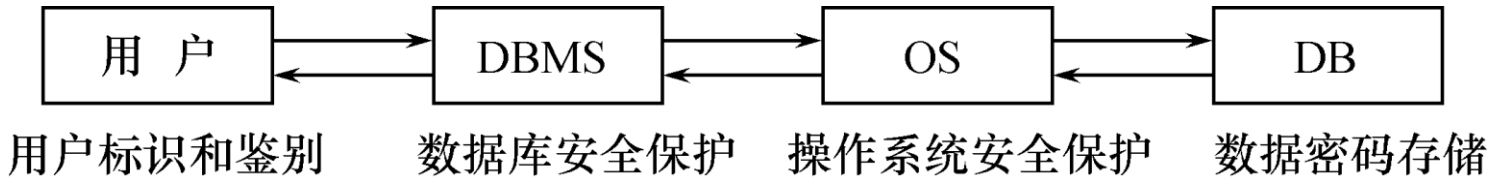
# 4.2 数据库安全性控制

## 非法使用数据库的情况

- 编写合法程序绕过数据库管理系统及其授权机制
- 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（续）

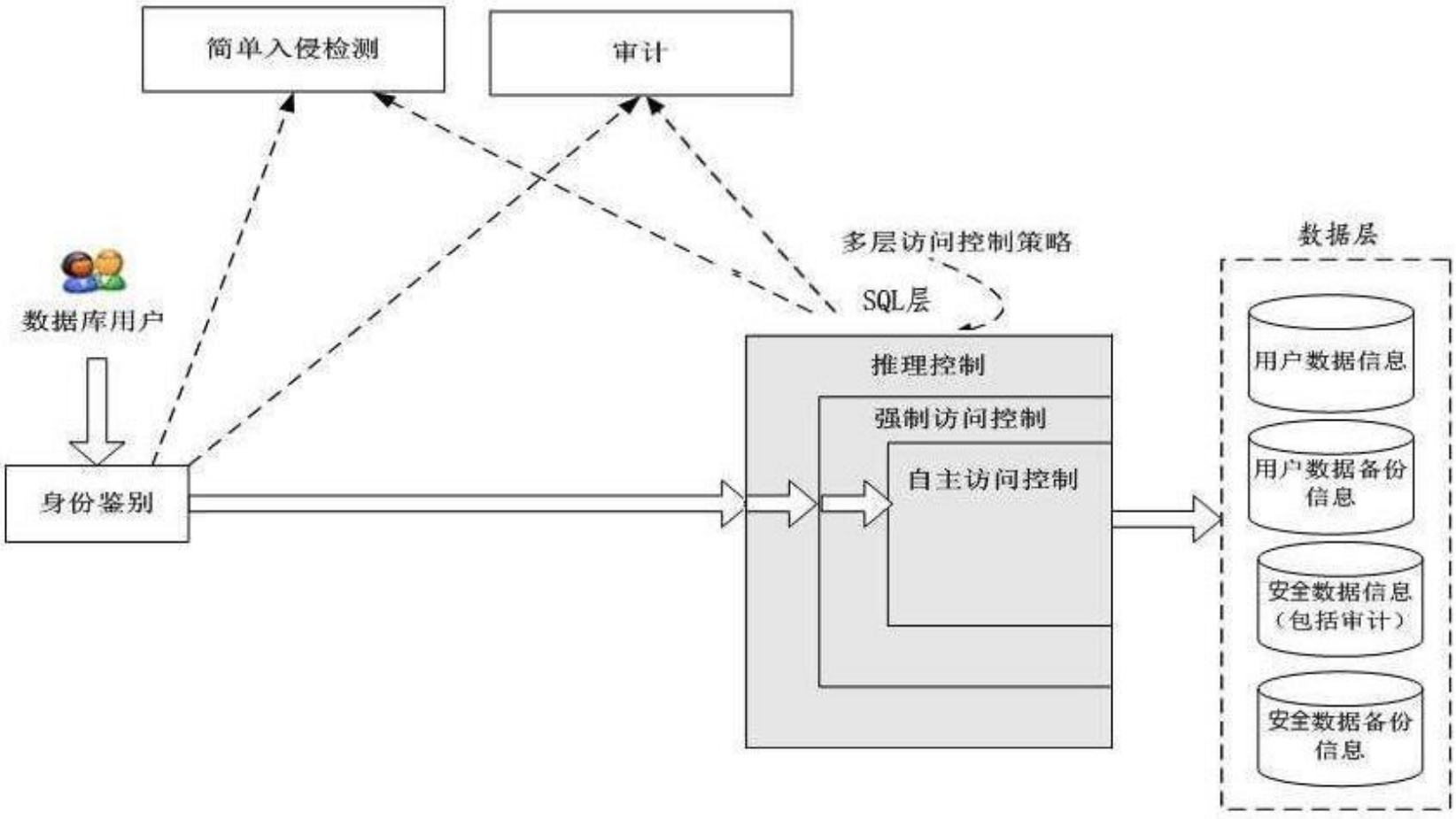
- 计算机系统中，安全措施是一级一级层层设置




计算机系统的安全模型

# 数据库安全性控制（续）

- 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
- 数据库管理系统还要进行存取控制，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




数据库管理系统安全性控制模型

## ❖存取控制流程

- 首先，数据库管理系统对提出SQL访问请求的数据库用户进行身份鉴别，防止不可信用户使用系统。
- 然后，在SQL处理层进行自主存取控制(DAC)和强制存取控制(MAC)，进一步可以进行推理控制。
- 还可以对用户访问行为和系统关键操作进行审计，对异常用户行为进行简单入侵检测。

# 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 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）


国家自然科学基金委员会
National Natural Science Foundation of China
科学基金网络信息系统
Internet-based Science Information System
NSFC首页 | 关于ISIS | 常见问题 | 相关软件下载
单位查询
项目检索
意见与建议
新时代科学基金资助导向
鼓励探索、突出原创；聚焦前沿、独辟蹊径
需求牵引、突破瓶颈；共性导向、交叉融通
系统登录
用户名：
用户名/邮箱
密　码：
密码区分大小写
验证码：
2agp
登录
找回用户名/密码？

# 验证码有什么用？

# 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.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.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

# 4.2 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 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数据库管理系统的存取控制子系统

# 4.2.2 存取控制

常用存取控制方法:

- 自主存取控制（Discretionary Access Control， 简称DAC）
  - C2级
  - 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
  - 用户还可将其拥有的存取权限转授给其他用户

根据DAC的控制要求,每一个用户要想访问数据库都需要以合法的用户注册数据库并拥有一定的权限。

为什么我们在一些系统(比如携程网)进行查询的时候,并不需要进行注册?

- 强制存取控制（Mandatory Access Control，简称 MAC）
  - B1级
  - 每一个数据对象被标以一定的密级
  - 每一个用户也被授予某一个级别的许可证
  - 对于任意一个对象，只有具有合法许可证的用户才可以存取

# 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 强制存取控制方法

# 4.2.3 自主存取控制方法

1.通过 SQL 的相应语句实现（4.3节中详细讲，如GRANT\REVOKE等）

2.用户权限组成

- 数据对象
- 操作类型

3.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

4.定义存取权限称为授权

# 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 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

# 4.2.4 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

# 4.2.4 强制存取控制方法

- 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

## 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

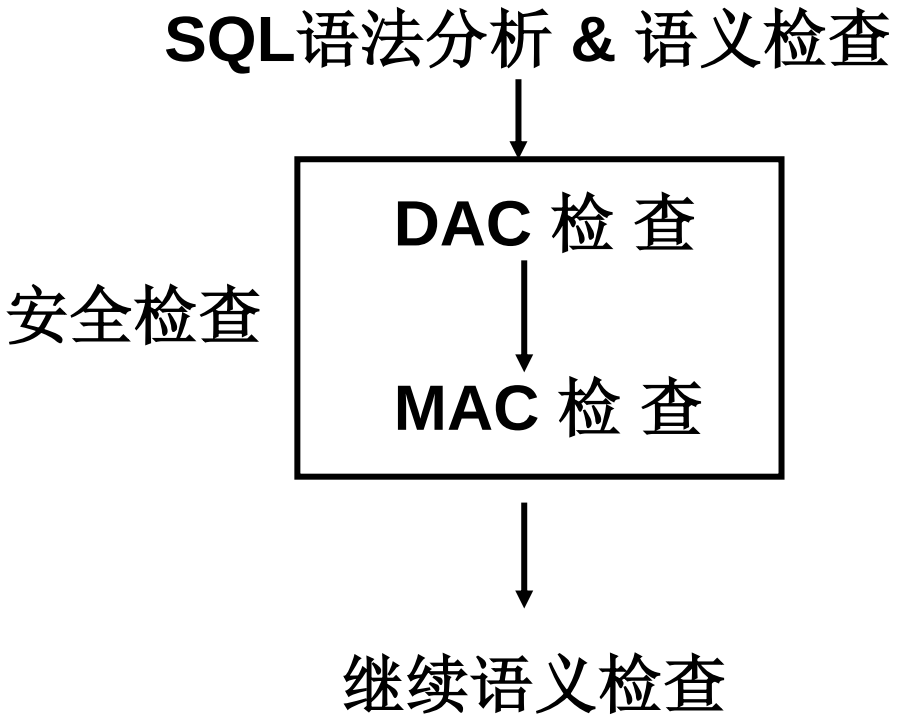
（2）仅当主体的许可证级别小于或等于客体的密级时，该主体才能写相应的客体

**问题：用户只能读密级比自己低的数据，但是可以写密级比自己高的数据，这意味着存在写进去的数据自己看不见的情况，这合理吗？**

# 强制存取控制方法（续）

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

# DAC + MAC安全检查




- 先进行自主存取控制检查，通过自主存取控制检查的数据对象再由系统进行强制存取控制检查，只有通过强制存取控制检查的数据对象方可存取。

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

# 4.3 SQL Server的安全控制

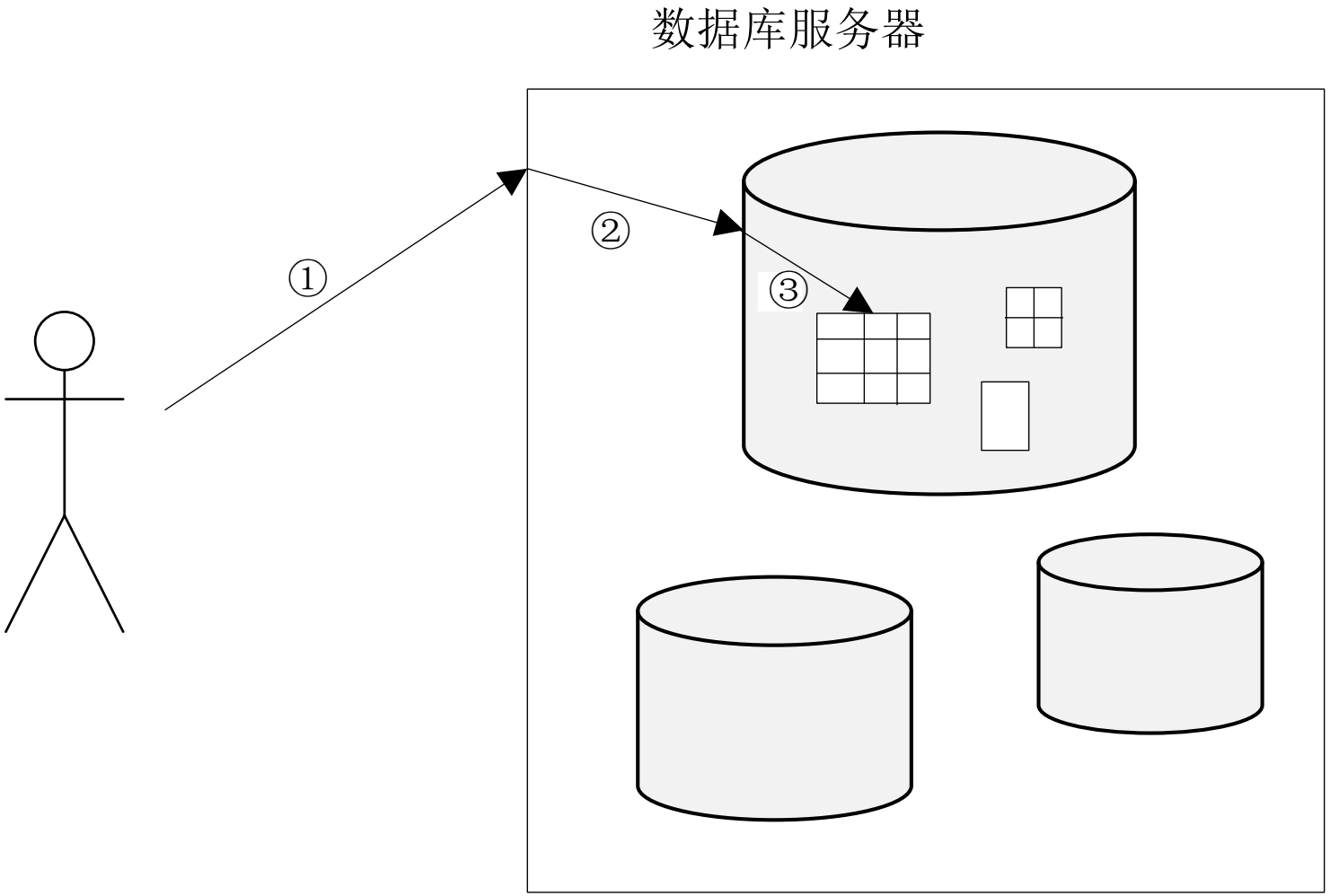
用户访问数据库数据都要经过三个安全认证过程：

1）第一个过程，确认用户是否是数据库服务器的合法账户（验证连接权）；

2）第二个过程，确认用户是否是要访问的数据库的合法用户（验证数据库访问权）；

3）第三过程，确认用户是否具有合适的操作权限（验证操作权）。

# 三个安全认证过程图示



# 4.3 SQL Server的安全控制

**4.3.1 登录名**

**4.3.2 数据库用户**

**4.3.3 数据库角色**

**4.3.4 权限管理**

# 4.3.1 登录名

**（1）身份认证模式**

**（2）建立登录名**

**（3）删除登录名**

# （1）身份验证模式

Windows身份验证模式

- 仅允许授权的Windows用户访问SQL Server

混合身份验证模式

- 仅允许授权的Windows用户和非Windows用户访问SQL Server

# （2）用T-SQL语句建立登录名

SQL作为结构化查询语言，是标准的关系型数据库通用的标准语言；T-SQL是在SQL基础上扩展的SQL Server中使用的语言。

```
CREATE LOGIN login_name { WITH
 <option_list1> | FROM <sources> }


<sources> ::=
   WINDOWS [ WITH <windows_options> [ ,...
 ] ]
<option_list1> ::=
   PASSWORD = 'password' [ , <option_list2> [
 ,... ] ]
```

# （2）用T-SQL语句建立登录名(续)

<option_list2> ::=
SID = sid
| DEFAULT_DATABASE = database
| DEFAULT_LANGUAGE = language

<windows_options> ::=
DEFAULT_DATABASE = database
| DEFAULT_LANGUAGE = language

例1．创建SQL Server身份验证的登录帐户。登录名为：SQL_User2，密码为：a1b2c3XY。

（windows连接级别下）

```
CREATE LOGIN SQL_User2
  WITH PASSWORD = 'a1b2c3XY'
```

# （3）删除登录帐户的T-SQL语句

DROP LOGIN login_name

不能删除正在使用的登录帐户，也不能删除拥有任何数据库对象（如：SA）、服务器级别（如：**WINDDOWS**的账户）对象的登录帐户。

例2．删除SQL_User2登录帐户。

DROP LOGIN SQL_User2

# 4.3.2 数据库用户

**（1）建立数据库用户**

**（2）删除数据库用户**

# 注 意

服务器登录名与数据库用户是两个完全不同的概念。

具有登录名的用户可以登录到SQL Server实例上，而且只局限在实例上进行操作。

数据库用户是数据库级别上的主体。用户在具有了登录名之后，只能连接到SQL SERVER 数据库服务器上。并不具有访问任何用户数据库的权限，只有成为了数据库的合法用户后，才能访问该数据库。

数据库用户是登录名以什么样的身份在该数据库中进行操作，是登录名在具体数据库中的映射。默认情况下，新建立的数据库含有一个用户——dbo,它是数据库的拥有者。

# （1）用T-SQL语句建立数据库用户

CREATE USER user_name [ { { FOR | FROM }
{ LOGIN login_name } ]

user_name：指定在此数据库中用于识别该用户的名称。

LOGIN login_name：指定要映射为数据库用户的SQL Server登录名。

如果省略FOR LOGIN，则新的数据库用户将被映射到同名的SQL Server登录名。

例3.本示例首先创建名为SQL_JWC且具有密码的SQL Server身份验证的服务器登录名，然后在students数据库中创建与此登录帐户对应的数据库用户JWC。

```
CREATE LOGIN SQL_JWC
WITH PASSWORD = 'jKJl3$nN09jsK84';
GO
USE students;
GO
CREATE USER JWC FOR LOGIN SQL_JWC;
```

# （2）删除数据库用户的T-SQL语句

DROP USER user_name

例4．删除SQL_User2用户。

DROP USER SQL_User2

# 4.3.3 数据库角色

- 为便于对用户及权限的管理，可以将一组具有相同权限的用户组织在一起，这一组具有相同权限的用户就称为角色(Role)。
- 用户可以根据实际的工作职能情况定义自己的一系列角色，并给每个角色授予合适的权限。
- 简化授权的过程
- 用户拥有角色或者角色拥有角色。

# T-SQL 建立用户定义的角色

CREATE ROLE role_name

[ AUTHORIZATION owner_name ]

role_name：待创建角色的名称。

owner_name：将拥有新角色的数据库用户或角色。

如果未指定用户，则执行CREATE ROLE的用户将拥有该角色。

例5.创建用户自定义角色：CompDept，拥有者为创建该角色的用户。（在windows身份验证连接的模式下执行）

```
CREATE ROLE CompDept;
```

例6.创建用户自定义角色：InfoDept，拥有者为SQL_User1。

```
CREATE ROLE InfoDept
 AUTHORIZATION SQL_User1;
```

# 4.3.4 管理权限

**(1) 权限种类**

**(2) 权限的管理**

# (1)权限种类

## 对象权限

- 是对表、视图等对象中数据的操作权。

## 语句权限

- 创建对象的权限。

## 隐含权限

- 指由SQL Server预定义的服务器角色、数据库角色、数据库拥有者和数据库对象拥有者所具有的权限。

# （2） 权限管理

- 授予权限：允许用户或角色具有某种操作权。

- 收回权限：不允许用户或角色具有某种操作权，或者收回曾经授予的权限。

- 拒绝权限：拒绝某用户或角色具有某种操作权，即使用户或角色由于继承而获得这种操作权，也不允许执行相应的操作。

# 用SQL语句管理对象权限

GRANT：授予权限；

REVOKE：收回权限；

DENY：拒绝权限。

# 用SQL语句管理对象权限(续)

GRANT 对象权限名 [ ， … ]
ON {表名| 视图名| 存储过程名}
TO {数据库用户名| 用户角色名 }
[ ， … ]
[WITH GRANT OPTION];

REVOKE 对象权限名 [ ， … ]
ON { 表名|视图名|存储过程名 }
FROM { 数据库用户名|用户角色名 }
[ ， … ]

DENY 对象权限名 [ ， … ]
ON {表名|视图名|存储过程名}
TO {数据库用户名|用户角色名 }
[ ， … ]

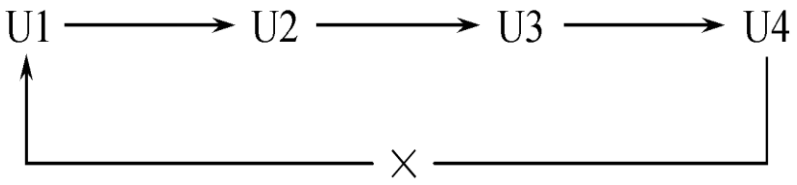
语句中数据库用户名、用户角色名为已存在的

# 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7．为用户user1授予Student表的查询权。

GRANT SELECT ON Student TO user1

例8．为用户user1授予SC表的查询权、插入、修改学号的权限。

GRANT SELECT，INSERT， UPDATE(Sno) ON SC TO user1

例9．收回用户user1对Student表的查询权。

REVOKE SELECT ON Student FROM user1

例10．拒绝user1用户具有SC表的更改权。

DENY UPDATE ON SC TO user1

例11．授予user1具有创建表的权限。

GRANT CREATE TABLE TO user1

例12．授予u1和u2具有创建表和视图的权限。

GRANT CREATE TABLE, CREATE VIEW TO u1, u2

例13．收回授予user1创建表的权限。

REVOKE CREATE TABLE FROM user1

例14．拒绝user1创建视图的权限。

DENY CREATE VIEW TO user1

例15．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 SC
TO U5
WITH GRANT OPTION;
```

执行例15后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限：

当以U5用户所在的登录名登录后，可执行例16,为U6用户授权。

例16. GRANT INSERT ON SC TO U6;

但是以U6用户所在的登录名登录后，不可再传播授权，即为其他用户授权。

使用SQL语句实现对用户角色的授权与为数据库用户授权完全一样。

例17.为Software角色授予Student表的查询权。

```
GRANT SELECT ON Student TO Software
```

例18.为Admin角色授予Student表的增、删、改、查权。

```
GRANT SELECT,INSERT,DELETE,
  UPDATE ON Student TO Admin
```

# SQL灵活的授权机制

数据库管理员：

- 拥有所有对象的所有权限
- 根据实际情况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
- 可以使用GRANT，把权限授予其他用户

被授权的用户:

- 如果具有“继续授权”的许可，可以把获得的权限再授予其他用户

所有授予出去的权力在必要时又都可用REVOKE语句收回

# 4.4 视图机制

- 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。
- 视图机制更主要的功能在于提供数据独立性，其安全保护功能太不精细，往往远不能达到应用系统的要求。

# 4.4 视图机制

- 视图机制与授权机制配合使用：
  - ✓首先用视图机制屏蔽掉一部分保密数据
  - ✓视图上面再进一步定义存取权限
  - ✓间接地实现支持存取谓词的用户权限定义

[例] 建立计算机系学生的视图，把对该视图的SELECT权限授于王平。

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;
```

如何让学生只能看自己所在系的学生数据(假定学生帐户名就是学生的名字,可以通过系统内置函数USER获得)?

```
create view classmate as
 select * from student  s1
 where dept = (select dept from student s2
          where sname=USER)
```

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

# 4.5 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，找出非法存取数据的人、时间和内容
- C2以上安全级别的DBMS必须具有审计功能

# 审计（续）

## 审计功能的可选性

- 审计很费时间和空间
- DBA可以根据应用对安全性的要求，灵活地打开或关闭审计功能
- 审计功能主要用于安全性要求较高的部门

# 审计（续）

- AUDIT语句和NOAUDIT语句
  - AUDIT语句：设置审计功能
  - NOAUDIT语句：取消审计功能
- 用户级审计
  - 任何用户可设置的审计
  - 主要是用户针对自己创建的数据库表和视图进行审计
- 系统级审计
  - 只能由数据库管理员设置
  - 监测成功或失败的登录要求、监测授权和收回操作以及其他数据库级权限下的操作

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

# 4.6 数据加密

- **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—明文（Plain text）变换为不可直接识别的格式—密文（Cipher text）

# 4.6 数据加密

- 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**

    如美国1977年制定的官方加密标准：数据加密标准（Data Encryption Standard，简称DES）

# 4.6 数据加密

- **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

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

# 4.7 其他安全性保护

- **推理控制**
  - 处理强制存取控制未解决的问题
  - 避免用户利用能够访问的数据推知更高密级的数据
  - 常用方法
    - ✓基于函数依赖的推理控制
    - ✓基于敏感关联的推理控制
- **隐蔽信道**
  - 处理强制存取控制未解决的问题
- **数据隐私保护**
  - 描述个人控制其不愿他人知道或他人不便知道的个人数据的能力
  - 范围很广：数据收集、数据存储、数据处理和数据发布等各个阶段

# 统计数据库安全性

统计数据库的特点

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

例：允许查询“程序员的平均工资是多少？”

不允许查询“程序员张勇的工资？”

统计数据库中特殊的安全性问题

- 隐蔽的信息通道
- 从合法的查询中推导出不合法的信息

规则1：任何查询至少要涉及N(N足够大)个以上的记录

例1：下面两个查询都是合法的：

1. 本公司共有多少女高级程序员？
2. 本公司女高级程序员的工资总额是多少？

如果第一个查询的结果是”1”，

那么第二个查询的结果显然就是这个程序员的工资数。

规则2：任意两个查询的相交数据项不能超过M个

例2：用户A发出下面两个合法查询：

1．用户A和其他N个程序员的工资总额是多少？

2．用户B和其他N个程序员的工资总额是多少？

若第一个查询的结果是X，第二个查询的结果是Y，由于用户A知道自己的工资是Z，那么他可以计算出用户B的工资=Y-(X-Z)。

原因：两个查询之间有很多重复的数据项

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

# 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 SQL Server的安全控制**

**4.4 视图机制**

**4.5 审计（Audit）**

**4.6 数据加密**

**4.7 其他安全性**

**4.8 小结**

# 4.8 小结

数据的共享日益加强，数据的安全保密越来越重要。

数据库管理系统是管理数据的核心，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

- 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
- 数据加密存储和加密传输

5版教材 P155

第7大题的第（7）小题

用户杨兰具有从每个部门职工中SELECT最高工资、最低工资、

平均工资的权限，但是他不能查看每个人的工资。

请用SQL语言实现它。