2025年春季学期

数据库系统概论

An Introduction to Database Systems

第四章 数据库安全性

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

计算机系统的安全性

- 计算机系统安全性
  - 为计算机系统建立和采取的各种安全保护措施，以保护计算机系统中的硬件、软件及数据，防止其因偶然或恶意的原因使系统遭到破坏，数据遭到更改或泄露等。
    - 技术安全类
    - 管理安全类
    - 政策法律类

复习：数据由DBMS统一管理和控制

- **DBMS提供的数据控制功能**
  - **1. 数据的安全性（Security）保护 （第4章）**

    保护数据，以防止不合法的使用造成的数据的泄密和破坏
  - **2. 数据的完整性（Integrity）检查 （第5章）**

    将数据控制在有效的范围内，或保证数据之间满足一定的关系
  - **3. 数据库恢复（Recovery） （第10章）**

    将数据库从错误状态恢复到某一已知的正确状态
  - **4. 并发（Concurrency）控制 （第11章）**

    对多用户的并发操作加以控制和协调，防止相互干扰而得到错误的结果

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
    - 共享 与 安全
  - 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

第四章 数据库安全性

4.1 数据库安全性概述

4.1.1 数据库不安全因素

4.1.2 安全标准简介

4.1.1 数据库不安全因素

- 对数据库安全性产生威胁的主要因素
  - 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露
  - 安全环境的脆弱
    - 计算机硬件、操作系统、网络系统等

4.1.1 数据库不安全因素

- 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。

4.1.1 数据库不安全因素

- 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

数据中的隐私泄露

- 大数据比赛
  - 在线电影推荐
  - 数据匿名化处理

4.1 数据库安全性概述

4.1.1 数据库不安全因素

4.1.2 安全标准简介

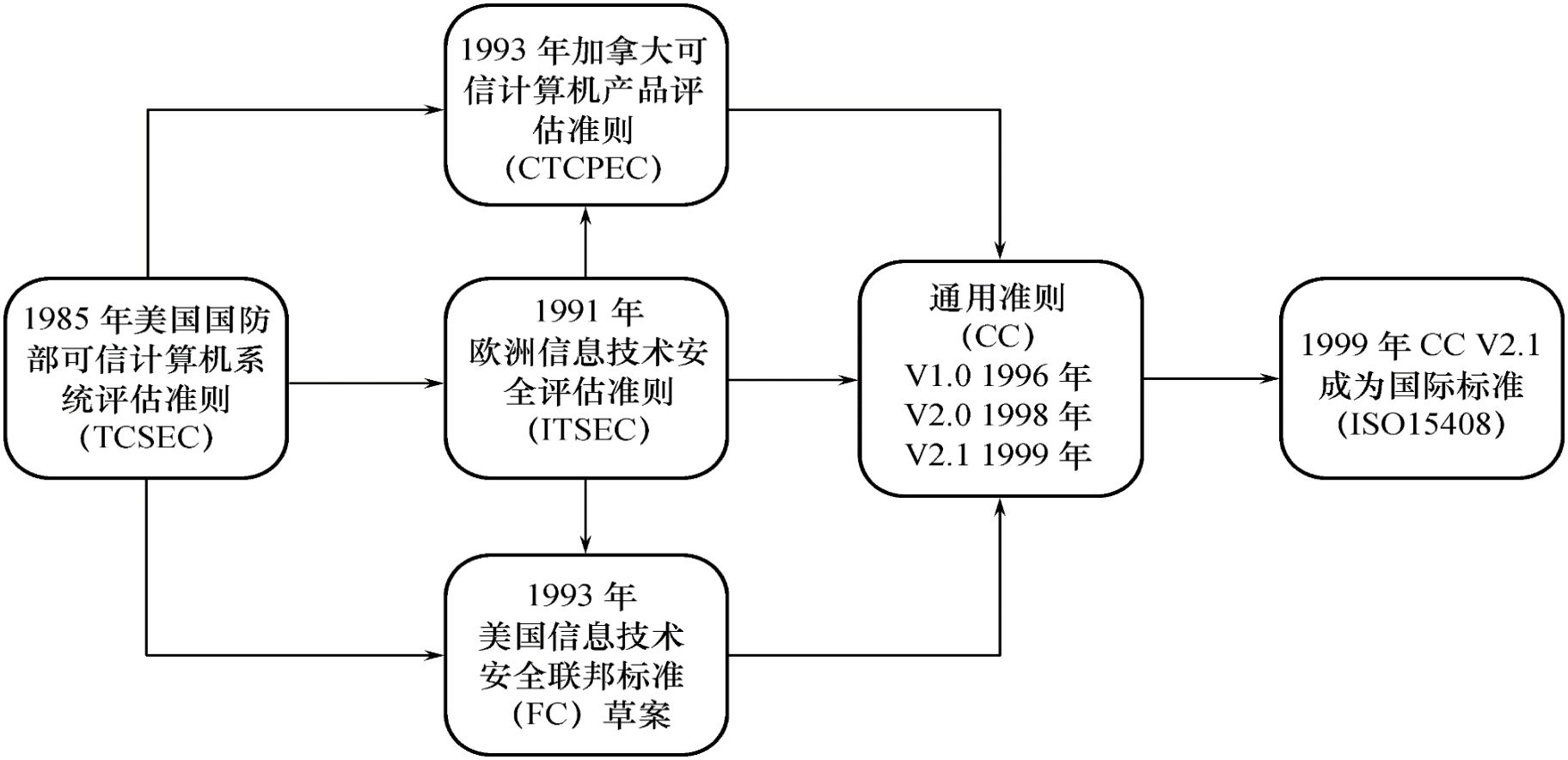
4.1.2 安全标准简介

- 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）

4.1.2 安全标准简介

- 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



信息安全标准的发展历史

安全标准简介

- **TCSEC标准**
  - 美国国防部可信计算机系统评估准则
  - 1985年颁布
- **CC标准**
  - 通用准则
  - 2001年成为我国标准

安全标准简介（续）

- **TCSEC/TDI** (Trusted Database Interpretation)**标准的基本内容**
  - 可信计算机系统评估准则关于数据库系统的解释
  - **TCSEC/TDI，从四个方面来描述安全性级别划分的指标**
    - **安全策略**
    - **责任**
    - **保证**
    - **文档**

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）

按系统可靠或可信程度逐渐增高

各安全级别之间：偏序向下兼容

TCSEC/TDI安全级别划分（续）

- 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

TCSEC/TDI安全级别划分（续）

- 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

TCSEC/TDI安全级别划分（续）

- B1级：真正意义上的安全产品
  - 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

TCSEC/TDI安全级别划分（续）

- B2以上的系统
  - 处于理论研究阶段
  - 应用多限于一些特殊的部门，如军队等
  - 美国正在大力发展安全产品，试图将目前仅限于少数领域应用的B2安全级别下放到商业应用中来，并逐步成为新的商业标准

CC

- **CC (Common Criteria)**
  - **提出国际公认的表述信息技术安全性的结构**
    - **结构开放、表达方式通用**
  - **把信息产品的安全要求分为**
    - **安全功能要求**
      - 信息技术的安全机制所要达到的功能和目的
    - **安全保证要求**
      - 确保安全功能有效并正确实现的措施与手段

CC（续）

- CC文本组成
  - 简介和一般模型
    - 介绍CC中有关的术语、基本概念和一般模型以及与评估有关的框架
  - 安全功能要求
    - 列出了一系列类（11个）、子类（66个）和组件（135个）。
  - 安全保证要求
    - 列出了保证类（11个）、子类（26个）和组件（74个），提出了评估保证级(EAL)

CC（续）

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

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 其它安全性保护

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
  - 大数据安全：从数据模型中推导保密数据

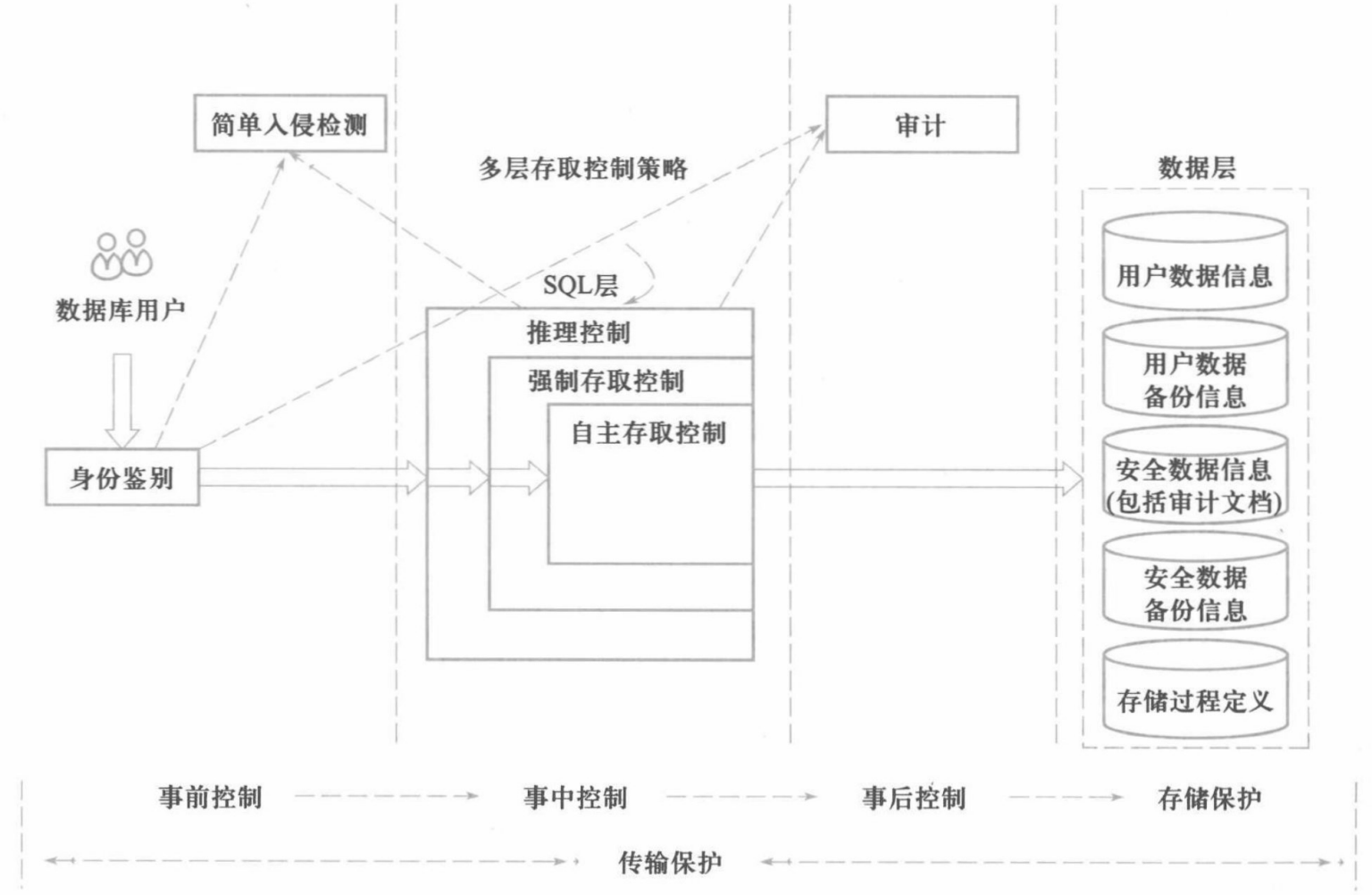
数据库安全性控制概述（续）

- 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

数据库安全性控制概述（续）



数据库管理系统安全性控制模型

数据库安全性控制概述（续）

- 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和强制存取控制，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。

数据库安全性控制概述（续）

- 数据库安全性控制的常用方法
  - 用户身份鉴定
  - 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 用户身份鉴定

4.2.2 存取控制

4.2.3 自主存取控制方法

4.2.4 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.1 用户身份鉴定

- 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）

用户身份鉴定（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.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

用户身份鉴定（补充）

- **SQL注入：Web应用最常见的攻击方式之一**

SQL Injection

学校：你好，这里是你儿子的学校，我们遇到了一些计算机问题

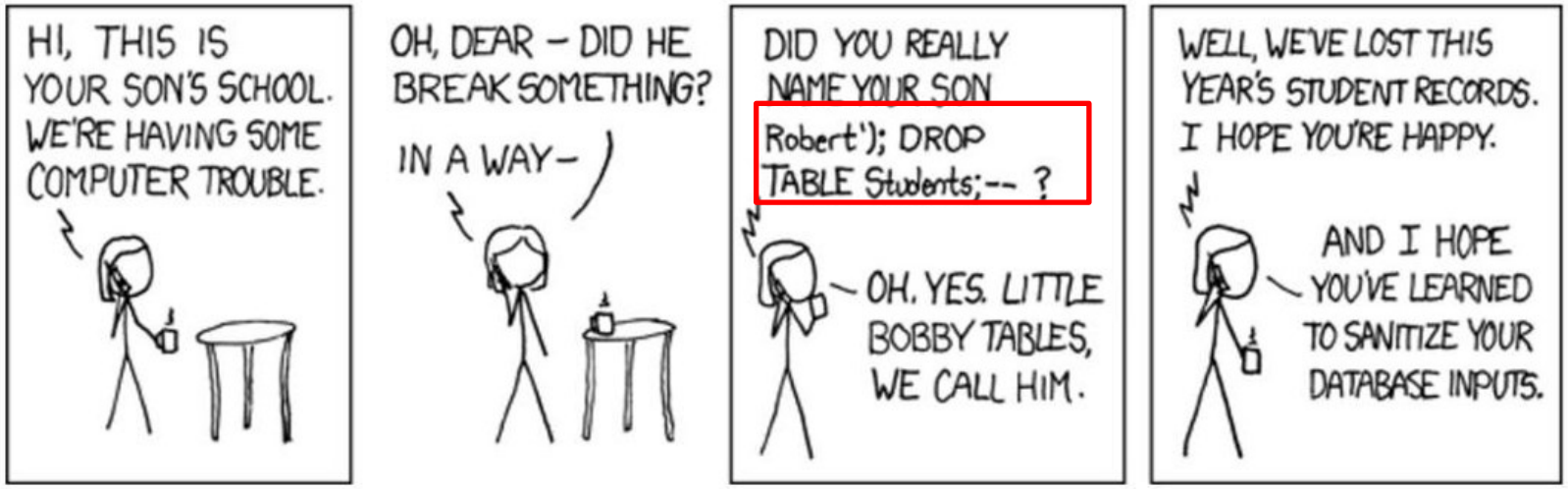
家长：啊这？他搞坏了什么嘛？

学校：你真的给你儿子起名字叫”Robert'); DROP TABLE Student;--”

家长：啊对对对，我们都叫他波比桌子。

学校：好吧，我们丢失了所有的学生数据

家长：希望你们能学会清洗数据库输入



用户身份鉴定（补充）

- 发生什么事了？

绑定$name

Username: Robert Submit

原SQL语句：
SELECT * FROM users WHERE name='{$name}'

$name = "Robert"
SELECT * FROM users WHERE name='Robert' → 正常

$name = "Robert'; DROP TABLE Student;--"
SELECT * FROM users WHERE name='Robert' ;
DROP TABLE Student;--' → SQL注入：删库

用户身份鉴定（补充）

- **SQL注入：Web应用最常见的攻击方式之一**
  - **1.思想：**
    - **利用Web应用对用户输入数据的合法性没有判断或过滤不严，在预定义好的查询语句后添加一段数据库查询代码，获得想得知的数据**
  - **2.漏洞在哪？**
    - **Robert’ ; DROP TABLE Student;--**
      - ’使得原本的‘闭合，--注释了后面的’
  - 分类
    - **联合注入、布尔注入、报错注入、时间注入、堆叠注入、二次注入、宽字节注入、cookie注入**

用户身份鉴定（续）

- **3.防御**
  - **最基本：用户口令的要求**
  - **检查变量数据类型和格式**
    - 日期、时间、邮箱、数字型等严格按照固定格式检查
  - **过滤特殊符号**
    - 过滤’ ” \字符中添加反斜杠转义
      - Robert’; DROP TABLE Student;--
      - Robert\’; DROP TABLE Student;--(SQL语句中’{$name}’两个’无法闭合)
  - **绑定变量，预编译语句**
    - 将传入的特殊SQL语句视为字符串执行(不会再编译SQL)
  - **严格管理数据库帐号权限**
    - 避免普通用户增删改查其他用户的资源

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 授权与收回

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

- 存取控制机制的组成
  - 定义用户权限
    - 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，DAC）
    - C2级
    - 灵活：用户自主（用户）
      - 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
  - 强制存取控制（Mandatory Access Control，MAC）
    - B1级
    - 严格：系统强制（数据）
      - 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 授权与收回

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	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

在对用户授权列**INSERT**权限时，一定要包含对主码的**INSERT**权限，否则用户的插入会因为控制而被拒绝。除了授权的列，其他列的值或者取空，或者取默认值。

在对用户授权列**UPDATE**一列的权限时，用户修改该列仍然要遵守创建时定义的主码和其他约束。

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 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法

4.2.4 授权与收回

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）

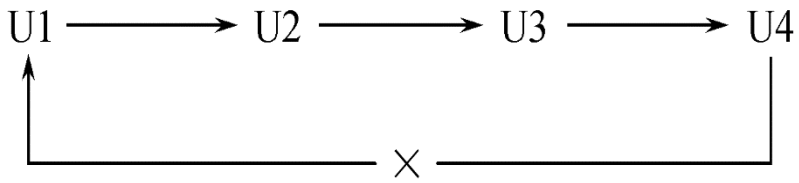
- 发出GRANT：
  - DBA(数据库管理员，mysql中的root)
  - 数据库对象创建者（即属主Owner）
  - 拥有该权限的用户

- 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4. 1] 把查询Student表权限授给用户U1

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

[例4.2] 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

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。

传播权限（续）

执行了［例4.1］到［例4.7］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

授权与收回（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>]...[CASCADE|RESTRICT];

REVOKE（续）

[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4.9] 收回所有用户对表SC的查询权限

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

REVOKE（续）

[例4.10] 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
  - U5—U6—U7，Ux—U6—U7

REVOKE（续）

执行［例4.8］到［例4.10］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

课后尝试MYSQL的授权方式

授权与收回（续）

*三、创建数据库的权限

- 创建数据库也需要进行安全控制，也是要授权的
- 但是在SQL标准中没有创建数据库的标准定义，因此各个数据库系统实现的语句不同

  下面仅以金仓数据库KingabseES为例做简要说明

  - CREATE USER语句创建超级用户，再由超级用户创建具有CREATE数据库权限的用户
  - CREATE USER语句格式

    CREATE USER <username> [WITH]

    [SUPERUSER | CREATEDB] | PASSWORD 'password';

授权与收回（续）

- **CREATE USER语句格式**
  - **具有SUPERUSER权限的用户是系统的超级用户，在系统中跳过权限检查，可以执行任何操作。**
  - **具有CREATEDB权限的用户可以创建数据库，成为数据库的属主，具有在数据库上创建模式的权限，并可以把这些权限授予其他用户**
  - **注意：超级用户是系统初始化时指定的。**
    - **例如，在安装系统时可以指定一个名称为system的超级用户。**

授权与收回（续）

[例4.11] 创建超级用户system2

首先以超级用户system登录，然后创建system2:

```
CREATE USER system2
WITH SUPERUSER PASSWORD '123456';
```

[例4.12] 创建具有CREATEDB权限的用户U1和普通用户U2

以超级用户system登录，创建用户U1, U2如下：

```
CREATE USER U1 WITH CREATEDB PASSWORD '123456'
/* U1 具有创建数据库的权限了 */
CREATE USER U2 PASSWORD '123456';    /* U2 是普通用户*/
```

授权与收回（续）

[例4.13] 创建数据库U1DB

以U1用户登录，创建数据库U1DB:

```
CREATE DATABASE U1DB;        / *  U1创建了数据库 * /
```

U1成为数据库U1DB的属主，它可以在U1DB数据库上创建SCHEMA

课后尝试MYSQL的权限管理

- **MySQL 默认有个root用户，权限太大，在管理数据库时候才用**
- **为 MySQL 创建一个新用户：**
  - **CREATE USER username IDENTIFIED BY 'password';**
- **为这个用户分配相应权限**
  - **GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost' IDENTIFIED BY 'password';**
- **授予它在某个数据库上的权限，切换到root 用户撤销刚才的权限，重新授权：**
  - EVOKE ALL PRIVILEGES ON *.* FROM 'username'@'localhost';
  - GRANT ALL PRIVILEGES ON wordpress.* TO 'username'@'localhost' IDENTIFIED BY 'password';
- **定该用户只能执行 select 和 update 命令**
  - GRANT SELECT, UPDATE ON wordpress.* TO 'username'@'localhost' IDENTIFIED BY 'password';

课后尝试MYSQL的权限管理

- 查询某个用户的权限
  - Show grants for USERNAME;
  - **select * from mysql.user where user=** USERNAME**;**
- 查询所有用户
  - **select * from mysql.user** # myysql数据库中的用户表
- 查询针对不同对象具有操作权限的用户
  - **数据库级别的权限信息是mysql.db表**
  - **表对象的授权信息记录是mysql.tables_priv表**
  - **列级权限记录在mysql.column_priv表**

查询权限

- 查询某个用户的权限

```
1 •  select * from mysql.user;
```

-

- Result Grid | Filter Rows: | Edit: | Export/Import:

Host	User	Select_priv	Insert_priv
localhost	mysql.infoschema	Y	N
localhost	mysql.session	N	N
localhost	mysql.sys	N	N
localhost	root	Y	Y
NULL	NULL	NULL	NULL

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 授权与收回

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]

  - 该语句把角色授予某用户，或授予另一个角色
  - 授予者是角色的创建者或拥有在这个角色上的ADMIN OPTION
  - 指定了WITH ADMIN OPTION则获得某种权限的角色或用户还可以把这种权限授予其他角色

一个角色的权限：直接授予这个角色的全部权限加上其他角色授予这个角色的全部权限

数据库角色

- 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>］…

  - 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**

数据库角色（续）

[例4.14]　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4.15]　角色的权限修改

```
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

数据库角色（续）

[例4.16]

```
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 授权与收回

4.2.5 数据库角色

4.2.6 强制存取控制方法

存取控制

- 常用存取控制方法
  - 自主存取控制（Discretionary Access Control，简称DAC）
    - 用户可“自主”地决定将数据的存取权限授予何人、决定是否也将“授予”的权限授予别人
      - C2级
      - 灵活
  - 强制存取控制（Mandatory Access Control，简称MAC）
    - 系统“强制”地给用户和数据标记安全等级
      - B1级
      - 严格

自主存取控制缺点

- 可能存在数据的“无意泄露”
  - 用户可以授权，用户可以备份数据
- 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）
  - 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）
    - 绝密（Top Secret, TS）
    - 机密（Secret, S）
    - 可信（Confidential, C）
    - 公开（Public, 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S的许可证级别大于或等于客体O的密级时，该主体才能读取相应的客体

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

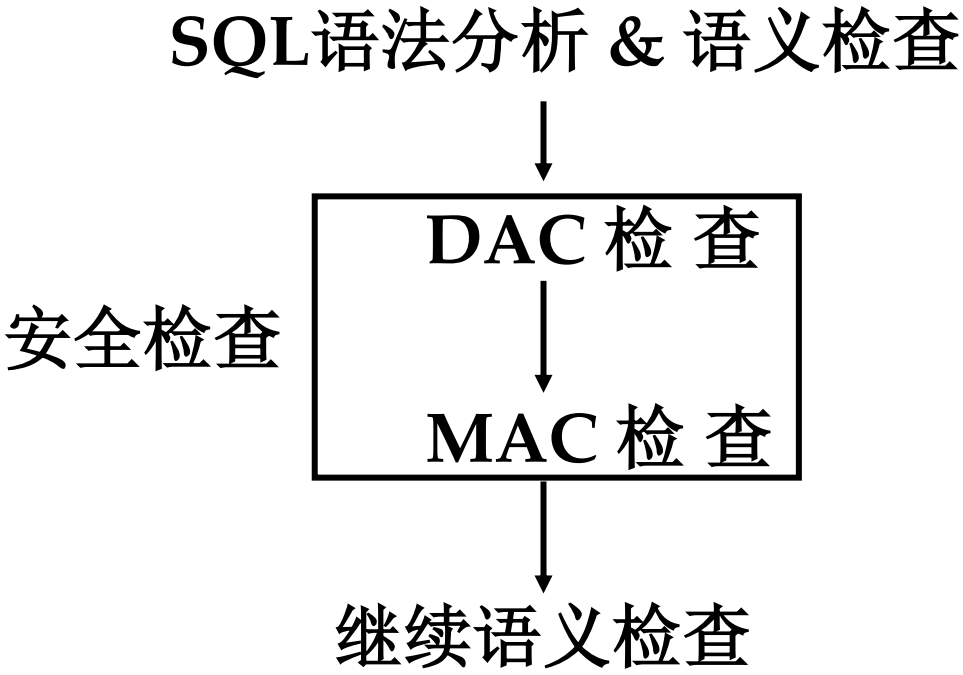
- 修正（即）规则
  - 主体的许可证级别 =客体的密级 → 主体能写客体

强制存取控制方法（续）

- 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共同构成数据库管理系统的安全机制

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。

第四章 数据库安全性

4.3 视图机制

- 视图机制与授权机制配合使用
- 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

视图机制（续）

[例4.17]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';
WITH  CHECK OPTION;
```

视图机制（续）

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;
```