第一篇 基础篇

第四章 数据库安全性

福州大学数计学院

程 烨

chengye@fzu.edu.cn

2013年4月22日

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

DBMS的数据控制功能

数据库系统中的数据是由DBMS统一管理和控制的，为了适应数据并发共享的环境，DBMS必须提供数据控制能力，以保证数据库中数据的安全可靠和正确有效。

数据控制功能：

- 安全性--防止非法使用数据
- 完整性--保证数据正确、有效、相容
- 并发控制—对并发操作的协调与控制
- 数据库恢复—发生故障后对数据库的恢复

数据库的安全性

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。

数据库系统安全保护措施是否有效是数据库系统主要技术指标之一。

数据库的安全性与计算机系统的安全性是紧密联系、互相支持的。

第四章 数据库安全性

4.1 计算机安全性概述

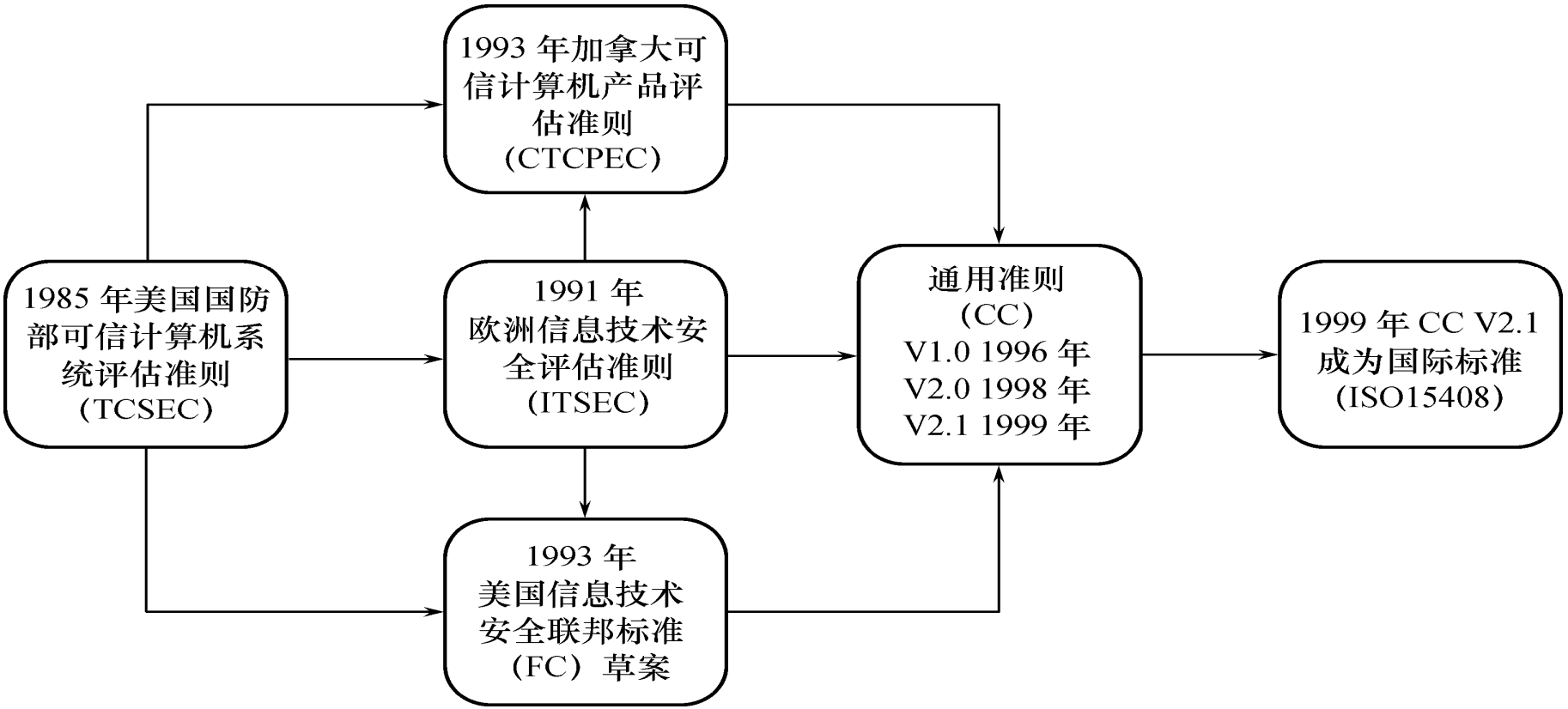
- 计算机系统安全性
  - 为计算机系统建立和采取的各种安全保护措施，以保护计算机系统中的硬件、软件及数据，防止其因偶然或恶意的原因使系统遭到破坏，数据遭到更改或泄露等。

4.1.1 计算机系统的三类安全性问题

- 技术安全类—采用技术手段保证计算机系统具有一定的安全性。需要建立一套可信计算机系统的概念和标准。
- 管理安全类—
- 政策法律类—

4.1.2 安全标准简介

计算机以及信息安全技术方面的一系列安全标准中，最有影响的是：**TCSEC**和**CC**标准。



TCSEC/TDI安全级别划分

根据计算机系统对上述各项指标的支持情况，TCSEC/TDI将系统划分为四组，七个等级，依次是：

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）

实现数据库系统安全性的技术和方法有多种，最重要的是存取控制技术和审计技术。

C2级的DBMS必须具有自主存取控制和初步的审计功能。

B1的DBMS必须具有强制存取控制和增强的审计功能。

目前许多大型DBMS达到了C2级，其安全版本达到了B1。

TCSEC/TDI安全级别划分（续）

- **B2以上的系统**
  - 还处于理论研究阶段
  - 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标准

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 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）

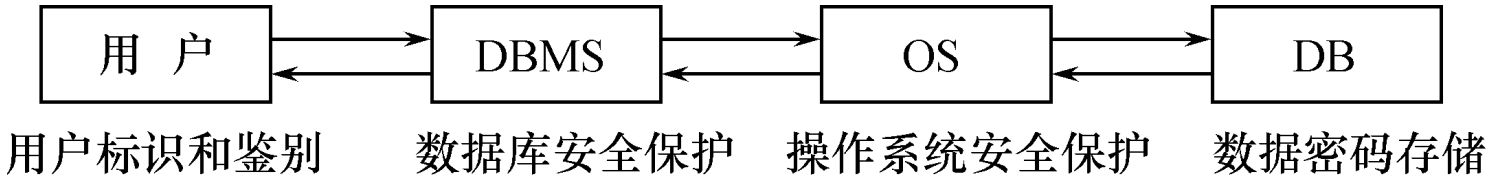
4.5 数据加密

4.6 统计数据库安全性

4.7 小结

数据库安全性控制概述

- 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---

  系统提供的最外层安全保护措施。

  为防止非授权用户使用系统，通过创建用户账号和口令来实现。

4.2.2 存取控制

存取控制机制主要包括两部分：

1、定义用户权限:用户权限是指不同的用户对于不同的数据对象允许执行的操作权限。

2、合法权限检查:每当用户发出存取数据库的操作请求后，DBMS查找数据字典，进行合法权限检查，若用户的操作请求超出了定义的权限，系统将拒绝执行此操作。

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

4.2.3 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

关系数据库系统中的存取权限

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）

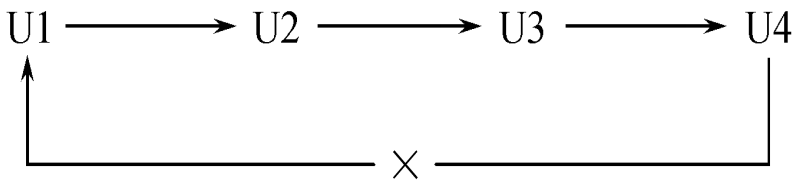
- 发出**GRANT**：
  - **DBA**
  - 数据库对象创建者（即属主**Owner**）
  - 拥有该权限的用户

- 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（续）

[例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;
```

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

传播权限

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

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。

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

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）

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

4.2.5 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>］…

SQL Server的预定义角色

- public角色
- 系统预定义角色
  - 使用**sp_helpsrvrole**获得各种系统管理员角色的描述
  - 使用**sp_srvrolepermission**得到每种系统管理员角色的特定权限（可以执行的命令、系统存储过程或说明）
- 数据库预定义角色
  - 使用**sp_helpdbfixedrole**获得数据库上各种预定义角色的描述
  - 使用**sp_dbfixedrolepermission**得到每种数据库预定义角色的特定权限（可以执行的命令、系统存储过程或说明）

public角色

- public角色是一个特殊的数据库角色，每个数据库用户都是该角色的成员。public角色具有如下特点：
  - public角色自动获得数据库中用户的所有默认权限；
  - 不需要、也无法将用户指派给public角色，因为默认情况下所有用户都属于该角色；
  - 每个数据库（包括所有系统数据库和所有用户数据库）都有public角色；
  - 不可以删除public角色。

SQL Server系统预定义角色

- **sysadmin**：具有系统管理员全部权限的角色。
- **serveradmin**：负责配置数据库服务器的设置。
- **setupadmin**：负责添加和删除链接的服务器。
- **securityadmin**：负责管理服务器的登录。
- **processadmin**：负责管理在**SQL Server**实例中运行的进程。
- **dbcreator**：负责创建和改变数据库。
- **bulkadmin**：可以执行**BULK INSERT**语句（数据库数据的装载）。

SQL Server数据库预定义角色

- **db_owner**：在数据库中有全部权限，即具有数据库管理员全部权限的角色。
- **db_accessadmin**：负责数据库用户的管理。
- **db_securityadmin**：负责数据库的安全管理，如负责权限管理、角色和角色成员资格管理等。
- **db_ddladmin**：主要负责数据库的完整性和一致性检查及管理。
- **db_backupoperator**：主要负责数据库的备份。
- **db_datareader**：可以查询数据库中任何用户表中的所有数据。
- **db_datawriter**：可以更改数据库中任何用户表中的所有数据。
- **db_denydatareader**：不能查询数据库中任何用户表中的任何数据。
- **db_denydatawriter**：不能更改数据库中任何用户表中的任何数据。

自主存取控制特点

- 同一用户对于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

由于用户对数据的存取权限是“自主”的，用户可以自由地决定将数据的存取权限授予何人、决定是否可将“授权”的权限授予别人，而系统对此无法控制。

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

强制存取控制方法（MAC）

- 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

强制存取控制方法（续）

在**MAC**中，**DBMS**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，既包括**DBMS**所管理的实际用户，也包括代表用户的各进程。

客体是系统中的被动实体，是受主体操纵的，包括文件、基表、索引、视图等。

强制存取控制方法（续）

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**），并分若干级别，如：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，

客体的敏感度标记称为密级。

MAC机制就是通过对比主体的**LABEL**和客体的**LABEL**，最终确定主体是否能够存取客体。

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

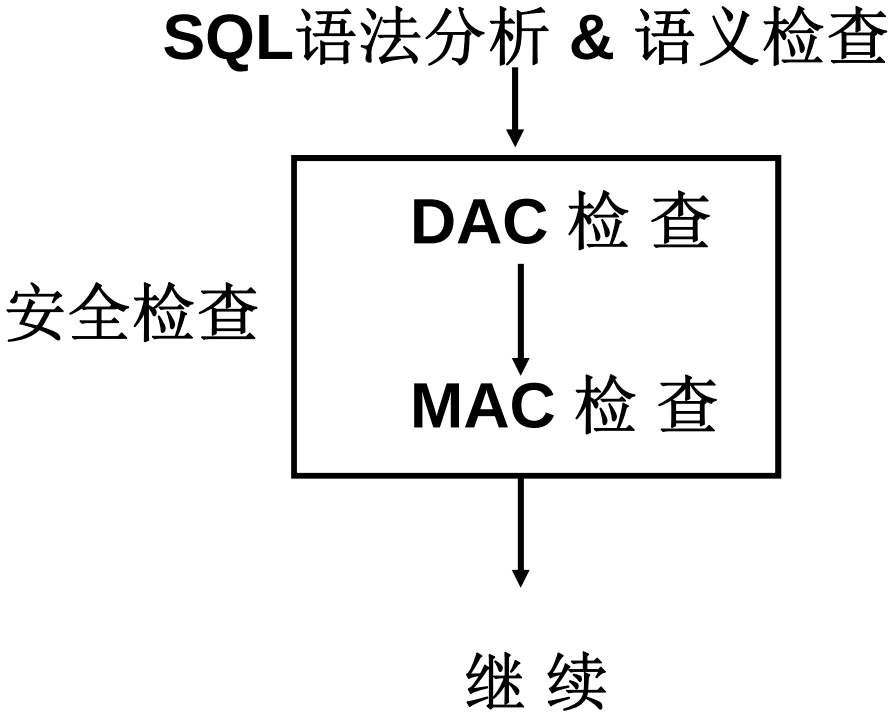
MAC与DAC

- DAC与MAC共同构成DBMS的安全机制
- 实现MAC时要首先实现DAC

系统首先进行DAC检查，对通过DAC检查的允许存取的数据对象再由系统自动进行MAC检查，只有通过MAC检查的数据对象方可存取。

强制存取控制方法（续）

DAC + MAC安全检查示意图



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

4.3 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
- 间接实现了支持存取谓词的用户权限定义

视图机制（续）

[例14]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';
```

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；

4.2 数据库安全性控制

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

4.4 审计

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（续）

- 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（续）

- AUDIT语句：设置审计功能
- NOAUDIT语句：取消审计功能

  - 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（续）

［例15］对修改SC表结构和修改SC表数据的操作进行审计

AUDIT ALTER，UPDATE

ON SC；

［例16］取消对SC表的一切审计

NOAUDIT ALTER，UPDATE

ON SC；

4.2 数据库安全性控制

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

4.5 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）变换为不可直接识别的格式（密文），从而使得不知道解密算法的人无法获知数据的内容。

- 加密方法---替换方法，置换方法，混合方法

- DBMS中的数据加密

第四章 数据库安全性

4.6 统计数据库安全性

统计数据库中的数据分：

微数据：是描述现实世界的实体，概念或事件的数据。

统计或综合数据：对微数据进行综合处理而得到的结果数据。

DBMS必须防止用户访问或推导出统计DB的微数据。

统计数据库安全性（续）

- 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？”

统计数据库安全性（续）

防止用户推导出统计数据库的微数据，可采用如下三种方法：

A、对统计查询结果的记录数加以控制，使之不小于一定范围。

B、禁止在相同元组集上重复执行一系列统计查询。

C、在统计查询结果中加入噪声。

统计数据库安全性（续）

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

用户定义的安全性措施

- 除了利用数据库管理系统提供的安全功能外，还可以使用触发器定义一些用户级的安全性措施。例如，最典型的用户定义安全性控制是，可以规定用户只在指定的时间允许对表进行更新操作。

用户定义的安全性措施

```
CREATE TRIGGER secure_wh
ON 仓库
FOR INSERT,DELETE,UPDATE
AS
IF DATENAME(weekday, getdate())='星期六'
   OR DATENAME(weekday, getdate())='星期日'
   OR    (convert(INT,DATENAME(hour,    getdate()))    NOT
    BETWEEN 9 AND 17)
BEGIN
RAISERROR ('只许在工作时间操作！', 16, 1)
ROLLBACK TRANSACTION
END
```

第四章 数据库安全性

4.7 小结

- 数据的共享日益加强，数据的安全保密越来越重要
- **DBMS**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
- **TCSEC**和**CC**

小结（续）

- 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