第四章 数据库安全性

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

4.5 数据加密

4.6 其他安全性保护

数据库安全性

- 什么是数据库的安全性
  - 数据库的安全性是指保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。
- 什么是数据的保密
  - 数据保密是指用户合法地访问到机密数据后能否对这些数据保密。
  - 通过制订法律道德准则和政策法规来保证。

数据库的不安全因素

- 非授权用户对数据库的恶意存取和破坏

  解决措施：用户身份鉴别、自主存取控制、视图

- 数据库中重要或敏感数据被泄漏

  解决措施：强制存取控制、数据加密存储、加密传输、审计

- 安全环境的脆弱性

数据库安全的基本概念

- 主体、客体与主客体分离

主体是数据的访问者，包括用户、应用程序、进程以及线程等。客体是数据及其载体，包括表、视图、数据文件、磁盘区域、内存区域等。

- 自主访问控制

定义和控制系统中主体对客体的访问，阻止非授权主体访问客体。

数据库安全的基本概念

- **审计**

记录可疑主体访问客体的轨迹，以防止非法主体对客体的访问。记录的内容包括：访问时间、用户、操作类型、是否成功等。

- **标记与强制访问控制**

对主体与客体加上标记，规定了主客体的安全级别，只有主体的标记大于或相等于客体的标记时才允许访问。

信息安全标准的发展

1967年美国国防部(DOD)成立研究组，针对当时计算机使用环境中的安全策略进行研究。

1969年，C.Weisman发表有关Adept-50的安全控制研究成果；

1970年，W.H.Ware推出的研究报告对多渠道访问的资源共享的计算机系统引起的安全问题进行了研究；

1972年,J.P.Anderson提出引用监控机、引用验证机制和安全核等根本思想，揭示安全规则的严格模型化的重要性,并提出了独立的安全评价方法问题。

之后,D.E.Bell和L. J.LaPadula提出著名Bell & LaPadula模型，该模型在Multics系统中得到了成功实现。

信息安全标准的发展

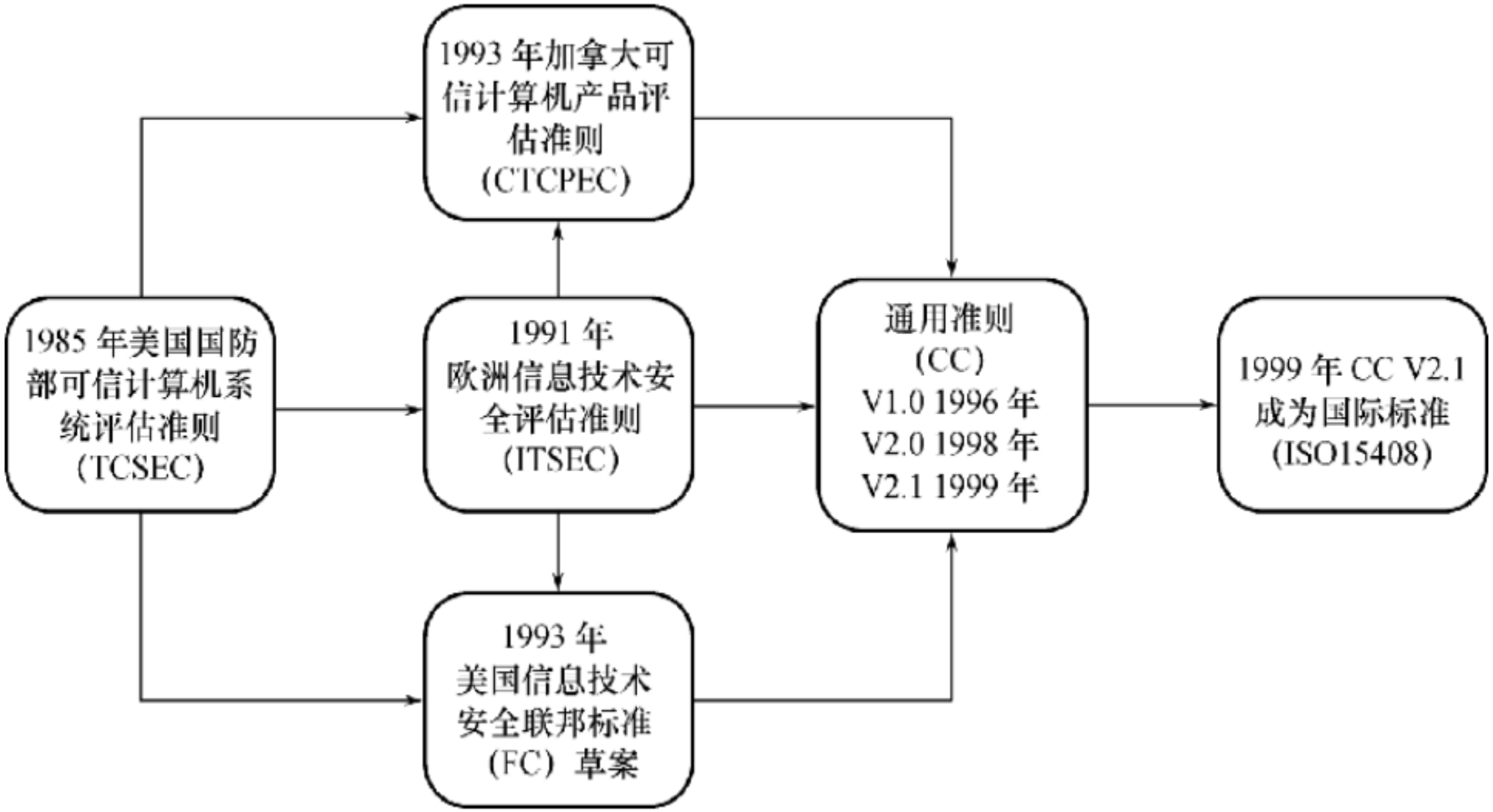
1983年美国国防部首次公布《可信计算机系统评估准则》(TCSEC) 以用于对操作系统的评估，这是IT历史上的第一个安全评估标准，1985年公布第二版。

TCSEC 为业界所熟知的名字“桔皮书”则是因其封面的颜色而得来。

后来DOD又发布了可信数据库解释(TDI)、可信网络解释(TNI)等一系列与TCSEC相关的说明和指南，由于这些文档发行时封面均为不同的颜色，因此常被人称为“彩虹计划”。

这一时期进行了大量的基础研究工作，并最终形成第一个美国国内的信息安全评估标准TCSEC。

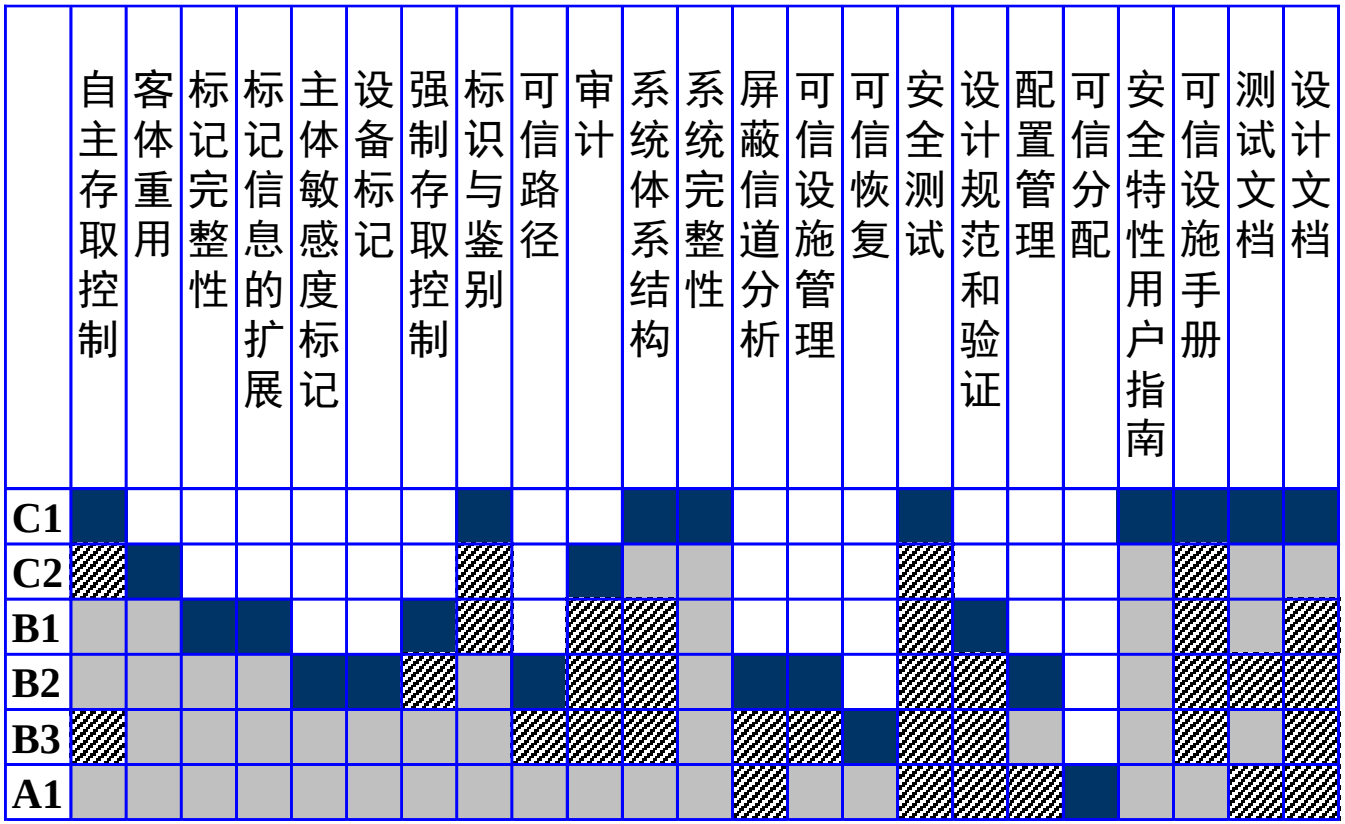
信息安全标准的发展


1985 年美国国防部可信计算机系统评估准则（TCSEC）
1993 年加拿大可信计算机产品评估准则（CTCPEC）
1991 年欧洲信息技术安全评估准则（ITSEC）
1993 年美国信息技术安全联邦标准（FC）草案
通用准则（CC）
V1.0 1996 年
V2.0 1998 年
V2.1 1999 年
1999 年 CC V2.1 成为国际标准（ISO15408）

TCSEC信息安全级别划分

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）

不同安全级别对安全指标的支持



不支持

新增支持

支持同相邻低一级

支持较下一级有增加或改动

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

我国标准

为提高我国计算机信息系统安全保护水平，1999年9月国家质量技术监督局发布了国家标准 GB17859-1999《计算机信息安全保护等级划分准则》，它是建立安全等级保护制度、实施安全等级管理的重要基础性标准。该标准的制定参照了美国的TCSEC。

2001年3月，国家质量技术监督局正式颁布了援引CC的国家标准 GB/T18336-2001《信息技术 安全技术 信息技术安全性评估准则》。

该标准共分为五级与美国TCSEC标准的对应

第一级：用户自主保护级C1级

第二级：系统审计保护级C2级

第三级：安全标记保护级B1级

第四级：结构化保护级B2级

第五级：访问验证保护级B3级

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

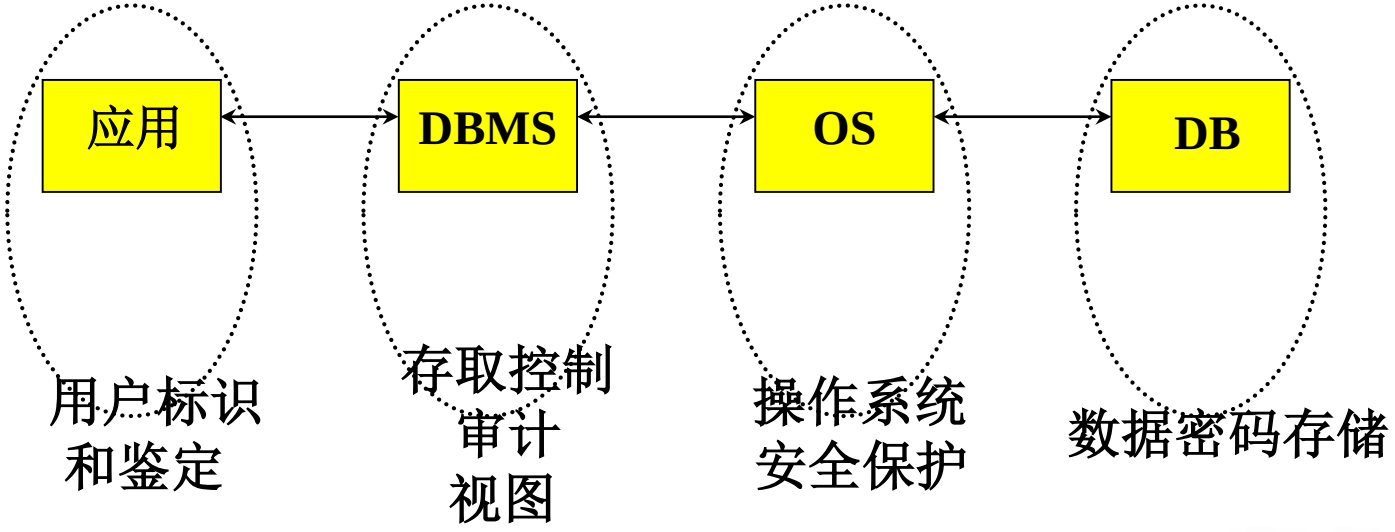
4.3 视图机制

4.4 审计（Audit）

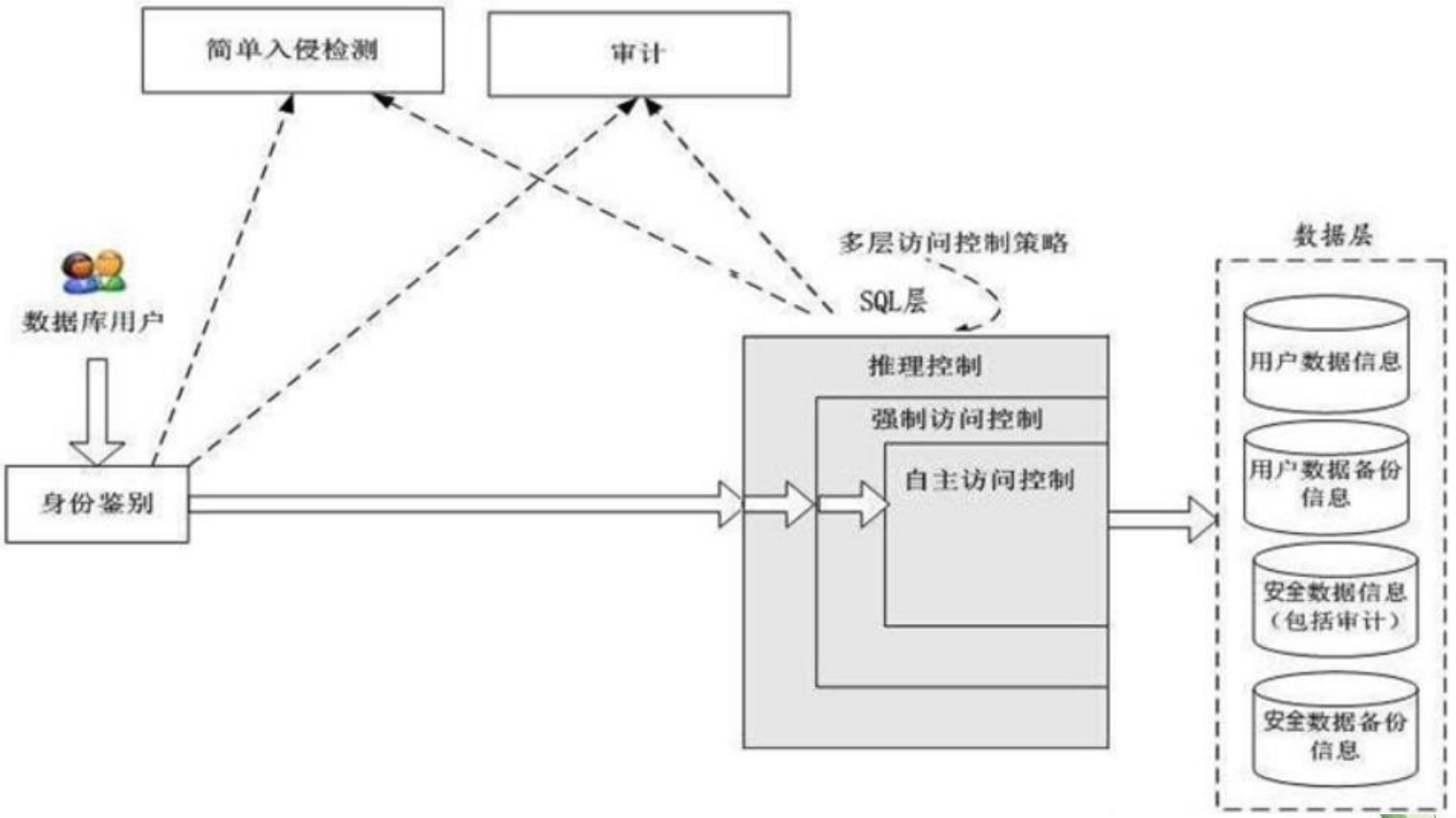
4.5 数据加密

4.6 其他安全性保护

计算机系统中的安全模型



数据库管理系统安全性控制模型



4.2.1 用户标识与鉴别

- 静态口令鉴别
- 动态口令鉴别
- 生物特征鉴别
- 智能卡鉴别

4.2.2存取控制

- 预先为每个用户定义存取权限（授权），经编译后存在数据字典中；
- 存取权限的两个要素：数据对象和操作类型
- 授权：定义存取权限
- 存取控制：对于取得上机权的后又进一步发出存取数据库操作的用户，DBMS查找数据字典，根据其存取权限对操作的合法性进行检查，若用户的请求超出了定义的权限，系统将拒绝执行此操作。

存取控制

- 存取控制机制组成
  - 定义用户权限
  - 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

- 授权粒度：是衡量授权机制是否灵活的一个重要指标（即可以定义的数据对象的范围）。
- 授权定义中数据对象的粒度越细，即可以定义的数据对象的范围越小，授权子系统就越灵活。
- 在关系数据库中，授权的数据对象粒度包括表、属性列、行(记录)。

存取控制

- 一个授权粒度很粗的表，它只能对整个关系授权 ， 如表2;
- 授权表也可以精细到对属性列授权，如表3;

表2 授权表

用户	数据对象名	允许的操作类型
张明	关系Student	SELECT
李青	关系Student	ALL
李青	关系Cource	ALL
李青	关系SC	UPDATE
王楠	关系SC	SELECT
王楠	关系SC	INSERT
……	……	……

表3 授权表

用户	数据对象名	允许的操作类型
张明	关系Student	SELECT
李青	关系Student	ALL
李青	关系Cource	ALL
李青	关系SC	SELECT
王楠	列SC.Grade	UPDATE
王楠	列SC.Sno	SELECT
王楠	列SC.Cno	SELECT
……	……	……

表4 扩充后的授权表

用户名	数据对象名	允许的操作类型	存取谓词
王平	关系Student	SELECT	Sdept=′CS′
张明霞	关系Student	UPDATE	Sname=′张明霞′
张明霞	关系 Course	ALL	空

存取控制

- 表2和表3中的授权定义均独立于数据值，用户能否执行某个操作与数据内容无关。
- 而表4中的授权表则不但可以对属性列授权，还可以提供与数值有关的授权，即可以对关系中的一组记录授权。
- 提供与数据值有关的授权，要求系统必须能支持存取谓词。还可以在存取谓词中引用系统变量。

实现与数据值有关的授权

- 实现与数据值有关的授权
  - 利用存取谓词
  - 存取谓词可以很复杂
    - 可以引用系统变量，如终端设备号，系统时钟等，实现与时间地点有关的存取权限，这样用户只能在某段时间内，某台终端上存取有关数据

例：规定“教师只能在每年1月份和7月份星期一至星期五上午8点到下午5点处理学生成绩数据”。

存取控制

- 授权粒度越细，授权子系统就越灵活，能够提供的安全性就越完善。
- 但另一方面，因数据字典变大变复杂，系统定义与检查权限的开销也会相应地增大。

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
    - 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

- 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

自主存取控制方法

- 自主存取控制
  - DAC （Discretionary Access Control ）， C2级， 灵活
  - 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

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

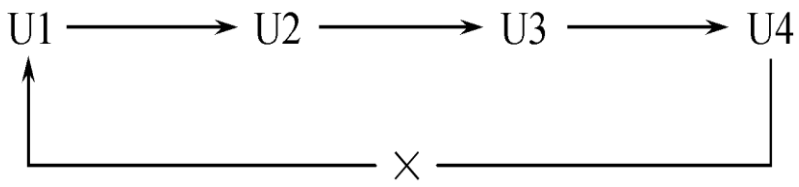
GRANT（续）

- 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

[例1] 把查询Student表权限授给用户U1

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

[例2] 把对Student表和Course表的全部权限授予用户U2和U3

```
GRANT ALL PRIVILIGES
ON Student, Course
TO U2, U3;
```

例题（续）

[例3] 把对表SC的查询权限授予所有用户

```
GRANT SELECT
ON SC
 TO PUBLIC;
```

[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 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 SC
TO U5
WITH GRANT OPTION;
```

执行例5后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限。

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

同样，U6还可以将此权限授予U7：

[例7] GRANT INSERT ON SC TO U7;

但U7不能再传播此权限。

传播权限（续）

下表是执行了［例1］到［例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8] 把用户U4修改学生学号的权限收回

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

[例9] 收回所有用户对表SC的查询权限

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

REVOKE（续）

[例10] 把用户U5对SC表的INSERT权限收回

```
REVOKE INSERT
ON 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

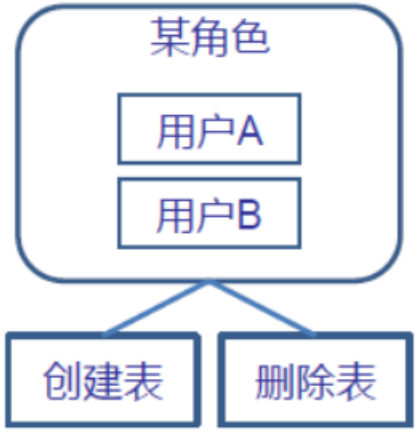
REVOKE（续）

执行［例8］到［例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

数据库角色

- 角色是数据库管理系统为方便权限管理而设置的管理单位，是一组权限的集合。
- 通过角色可将用户分为不同的类型，对相同类用户赋予相同的操作权限。



数据库角色

角色的类型

- 固定角色
- 用户定义数据库角色

数据库角色

固定角色是指其权限已被SQL Server定义，且SQL Server管理者不能对其权限进行修改的角色。

- 固定服务器角色
- 固定数据库角色

数据库角色

- 固定服务器角色

  独立于各个数据库，具有固定的权限，作用范围是整个服务器。

角色名称	说 明	角色名称	说 明
sysadmin	系统管理员角色	dbcreator	数据库创建者角色
serveradmin	服务器管理员角色	diskadmin	可管理磁盘文件
setupadmin	设置管理员角色	bulkadmin	可执行大容量的插入操作
securityadmin	安全管理员角色	public	查看任何数据库
processadmin	进程管理员角色		

数据库角色

- 固定数据库角色

  指数据库的管理、访问权限已被SQL Server固定的那些角色。

角色名称	说 明	角色名称	说 明
db_owner	数据库所有者	db_backupoperator	数据库备份操作员
public	特殊数据库角色	db_datareader	数据库数据读取者
db_accessadmin	数据库访问权限管理者	db_datawriter	数据库数据写入者
db_securityadmin	数据库安全管理者	db_denydatareader	数据库拒绝数据读取者
db_ddladmin	数据库DDL管理员	db_denydatawriter	数据库拒绝数据写入者

数据库角色

用户定义数据库角色

在创建数据库角色时将某些权限授予该角色，然后将数据库用户指定为该角色的成员，用户将继承这个角色的所有权限。

- 使用CREATE ROLE命令创建数据库角色

  语法格式：

  CREATE ROLE 角色名 [AUTHORIZATION 所有者名]

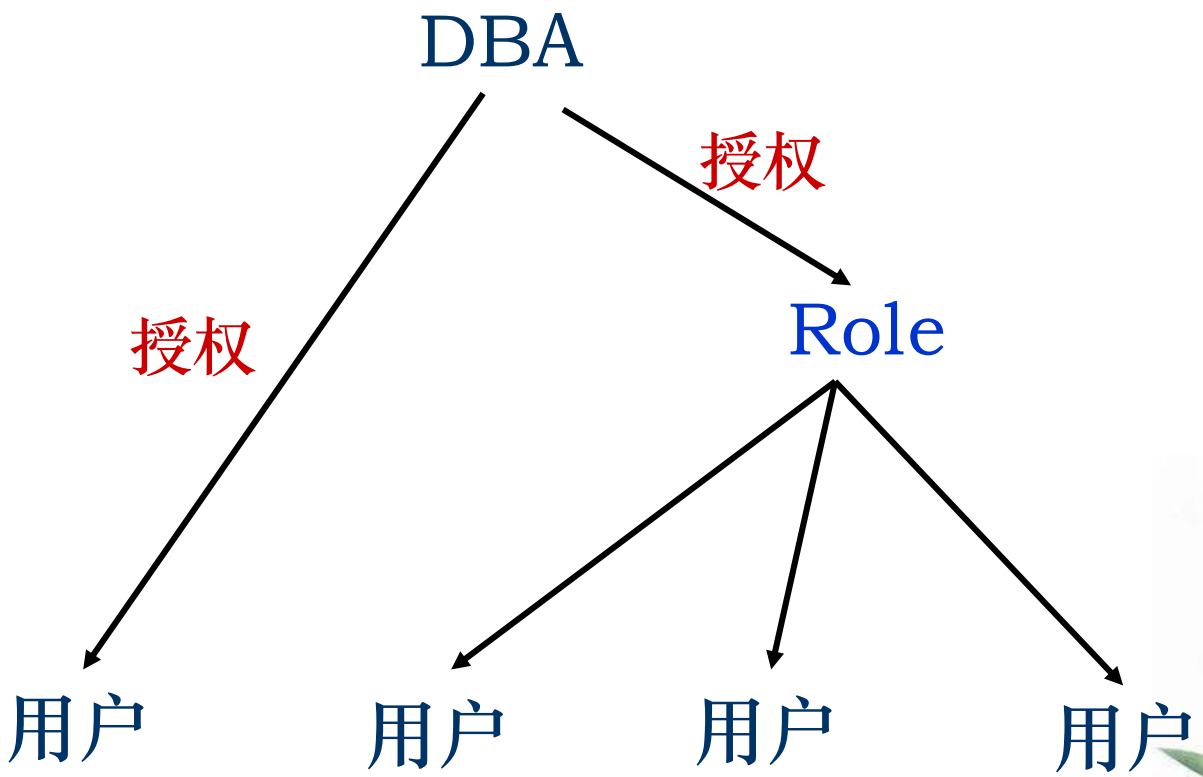
- 使用DROP ROLE命令删除数据库角色

  语法格式：

  DROP ROLE 角色名

数据库角色

授权管理



- **给角色授权**

  GRANT <权限>［，<权限>］…

  ON <对象类型>对象名

  TO <角色>［，<角色>］…

❖ **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］

❖ **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>］…

[例11]　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

```
sp_addrole　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 Student
TO R1；
```

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12]　角色的权限修改

GRANT DELETE

ON Student

TO R1

[例13]

REVOKE SELECT

ON Student

FROM R1；

自主存取控制小结

- 机制：
  - 定义存取权限：用户
  - 检查存取权限：DBMS
- 优点：能够通过授权机制有效地控制其他用户对敏感数据的存取
- 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

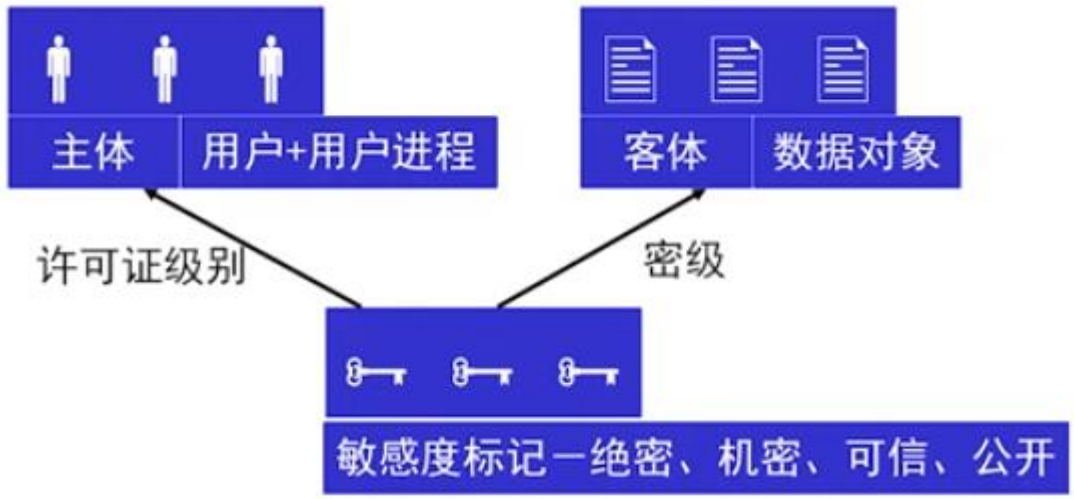
强制存取控制方法

- 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）
  - 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（Clearance Level）
- 客体的敏感度标记称为密级（Classification Level）



强制存取控制规则

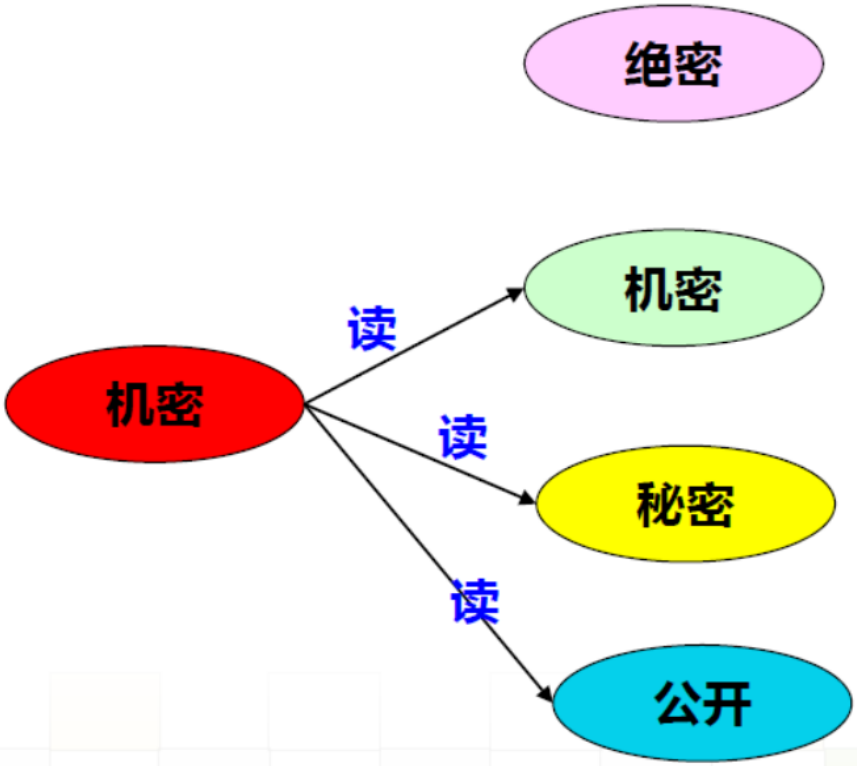
- 当某一用户（或某一主体）以标记label注册入系统时，系统要求他对任何客体的存取必须遵循下面两条规则：

  （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。

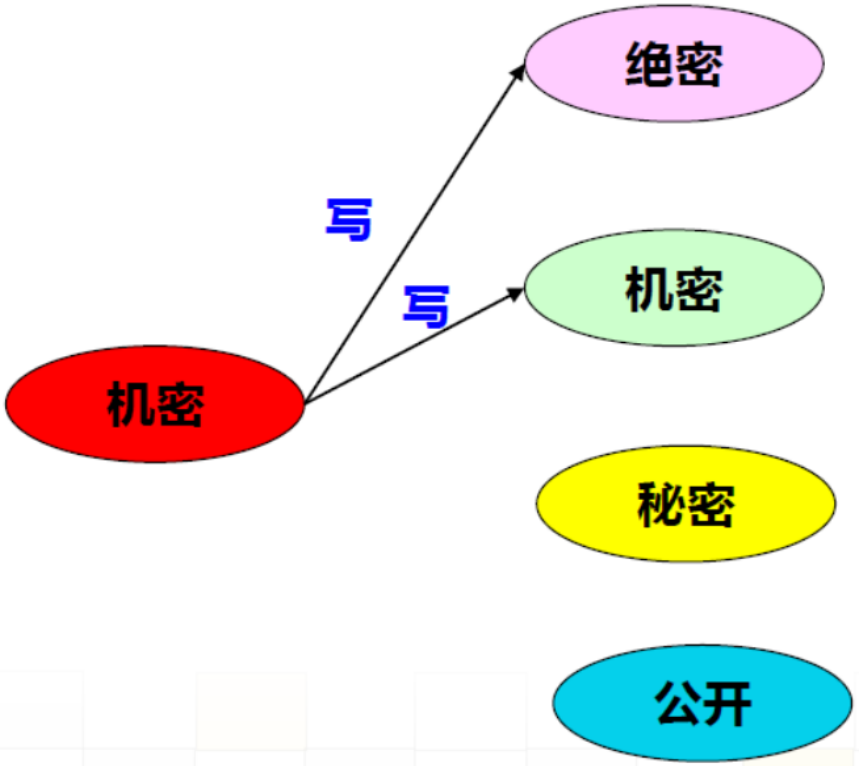
  – 用户可为写入的数据对象赋予高于自己的许可证级别的密级
  – 一旦数据被写入，该用户自己也不能再读该数据对象

强制存取控制规则



读操作

下读



写操作

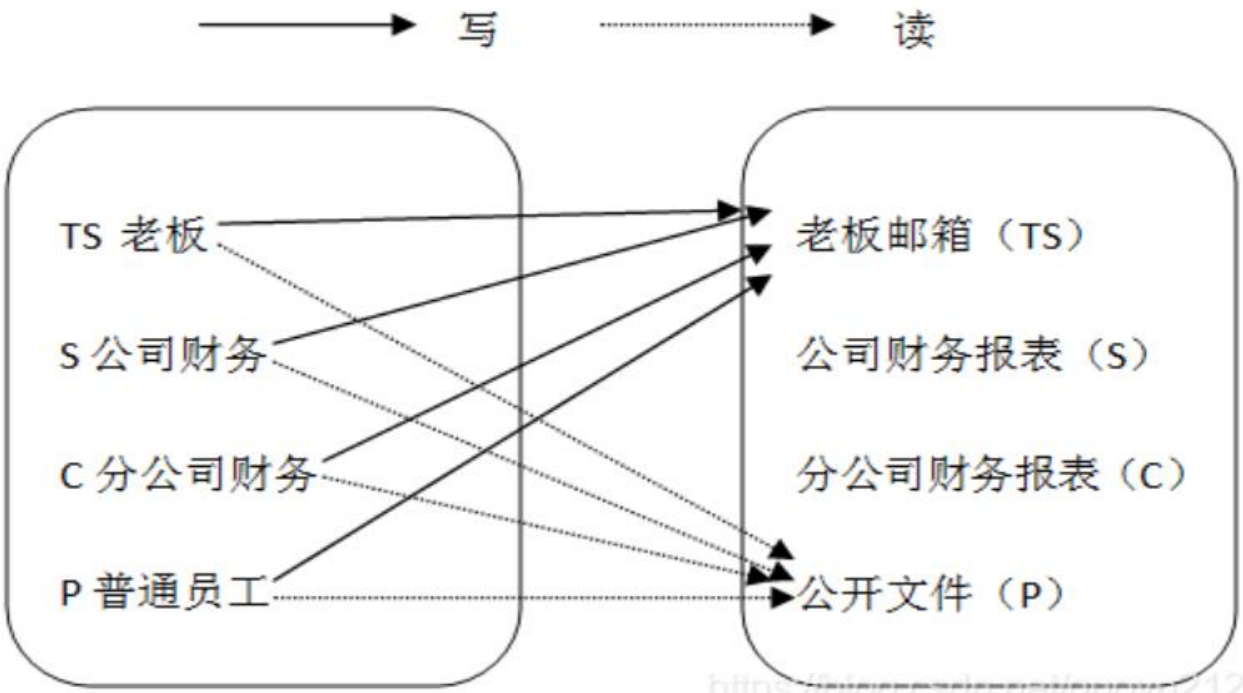
上写

举例

标签级别： TS>=S>=C>=P (绝密、机密、可信、公开)

读取规则：主体级别>=客体级别 可读；主体级别<=客体级别 可写

示例： 老板TS、S财务、C分公司财务、P普通员工



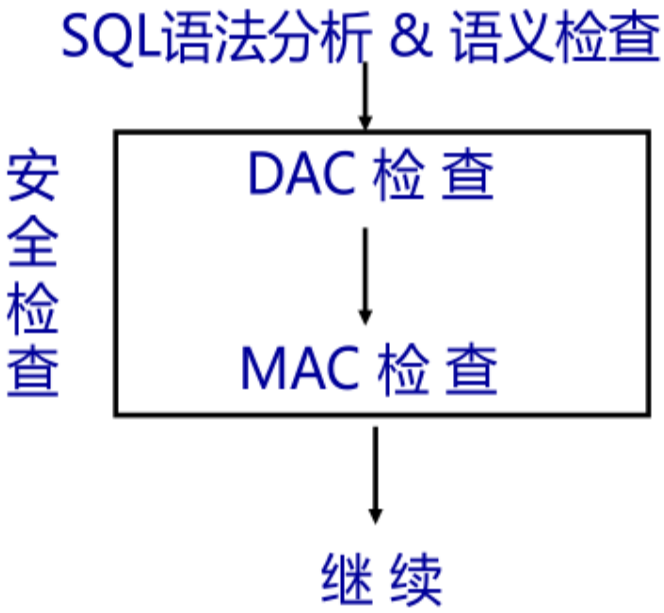
强制存取控制的优点

- 禁止拥有高许可证级别的主体更新低密级的数据对象，从而保证了敏感数据的可靠性；
- 禁止低许可证级别的主体浏览高密级的数据，避免了敏感数据的泄漏；
- MAC对数据本身进行密级标记，无论数据如何复制，标记与数据是不可分割的整体。只有符合密级标记要求的用户才可以操作相应数据，提高了安全性级别。

强制存取控制方法

- **DAC与MAC共同构成DBMS的安全机制**
- **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

❖ **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。**



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

4.5 数据加密

4.6 其他安全性保护

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

视图机制与授权机制配合使用

- 方法
  - 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

例：王平只能检索计算机系学生的信息：先建立计算机系学生的视图CS_Student，在视图上进一步定义存取权限

```
CREATE VIEW CS_Student
AS
SELECT
FROM   Student
WHERE  Sdept='CS';
```

```
GRANT  SELECT
ON  CS_Student
TO 王平 ;
```

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

4.5 数据加密

4.6 其他安全性保护

审计

- 当数据相当敏感，或者对数据的处理极为重要时，就必须以审计技术作为预防手段，监测可能的不合法行为。
- 审计追踪使用的是一个专用文件，系统自动将用户对数据库的所有操作记录在上面，利用审计追踪的信息，就能重现导致数据库现有状况的一系列事件，以找出非法存取数据的人。
- 审计功能一般主要用于安全性要求较高的部门。
- 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（续）

审计功能可以记录正在使用哪些系统权限，使用频率是多少，多少用户正在登录，会话平均持续多长时间，正在特殊表上使用哪些命令，以及许多其他有关事实。

审计功能把用户对数据库的所有操作自动记录下来放入**审计日志**（Audit Log）中。审计日志一般包括下列内容：

(1) 操作类型（如修改、查询等）。

(2) 操作终端标识与操作人员标识。

(3) 操作日期和时间。

(4) 操作的数据对象（如表、视图、记录、属性等）。

(5) 数据修改前后的值。

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

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

4.5 数据加密

4.6 其他安全性保护

◆ 基于文件的加密

把数据库文件作为整体，用加密算法对整个数据库文件加密来保证信息的真实性和完整性。

这一方法的实际应用受到多方面的限制：

（1）数据修改的工作将变得十分困难，需要进行解密、修改、复制和加密四个操作，极大地增加了系统的时空开销；

（2）即使用户只是需要查看某一条记录，也必须将整个数据库文件解密，这样无法实现对文件中不需要让用户知道的信息的控制。

◆ 字段加密

原理：将重要的字段内容进行加密，每个字段可以使用不同的密钥，也可以使用共同的密钥。

字段加密具有较高的安全性，但是由于对一个记录进行存取时需要多次的加/解密处理，影响数据库的访问速度。

通常情况下，以下几种字段不宜加密：

（1）索引字段不能加密；

（2）关系运算的比较字段不能加密；

（3）表间的连接码字段不能加密。

◆ 秘密同态技术

上述数据库加密方法存在一个共同的问题：对于所形成的密文数据库无法进行操作。而且在实际应用中，对于某些重要或敏感数据，无法满足用户对其进行操作但又不让用户了解其中信息的需要。

秘密同态 (Private homomorphism) 技术：能对密文数据库进行常规的数据库操作。

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

4.5 数据加密

4.6 其他安全性保护

其他安全性保护

- 推理控制
- 隐蔽通道
- 数据隐私

其他安全性保护

- 推理控制
  - 统计数据库中隐蔽的信息通道
  - 能从合法的查询中推导出不合法的信息

推理控制

许多场合，统计数据可以公开，而个别数据需要保密。如：人均工资可以知道，个人工资保密。有些数据库以统计应用为主，称为统计数据库，而个别值可以通过统计数据推断出来。如何堵塞这个漏洞，还未完全解决。

下面举例说明泄密的途径和可以采取的一些预防措施。

下表为一统计表S。

统计表 s

姓名	性别	家庭成员数	职业	工资
王	男	2	程序员	120
常	女	2	经理	240
陈	女	0	程序员	140
李	女	2	工程师	160
刘	男	2	前台	110
朱	女	1	培训	80
赵	男	0	教授	180
孙	男	1	培训	110
徐	女	2	程序员	130
马	女	1	程序员	150

推理控制

设除工资外，其它均可显示。那【王】的工资如何推断出来？

Q1：SELECT COUNT(*) FROM S

WHERE 性别='男' AND 职业='程序员'

结果： 1

Q2： SELECT sum(工资) FROM S

WHERE 性别='男' AND 职业='程序员'

结果：120

改进：系统限制查询结果的元组个数c不得小于一个下限b，即c≥b。 可否？

姓名	性别	家庭成员数	职业	工资
王	男	2	程序员	120
常	女	2	经理	240
陈	女	0	程序员	140
李	女	2	工程师	160
刘	男	2	前台	110
朱	女	1	培训	80
赵	男	0	教授	180
孙	男	1	培训	110
徐	女	2	程序员	130
马	女	1	程序员	150

推理控制

姓名	性别	家庭成员数	职业	工资
王	男	2	程序员	120
常	女	2	经理	240
陈	女	0	程序员	140
李	女	2	工程师	160
刘	男	2	前台	110
朱	女	1	培训	80
赵	男	0	教授	180
孙	男	1	培训	110
徐	女	2	程序员	130
马	女	1	程序员	150

本例中令b=2，下面的查询Q3～Q6均满足c≥2，但仍可以推断出【王】的工资。

Q3：SELECT COUNT(*) FROM S

结果： 10

Q4：SELECT COUNT(*) FROM S
WHERE NOT（性别='男' and 职业='程序员'）

结果： 9　推断出：男程序员1人

Q5： SELECT sum(工资) FROM S

结果：1420

Q6： SELECT sum(工资) FROM S
WHERE NOT（性别='男' and 职业='程序员'）

结果：1300 推断出：【王】的工资120

改进：系统再限制查询结果的元组个数c，即（n-b）≥ c≥b。

本例中，2<=c<=8,以上方法失效，但仍无法堵住漏洞。

以下Q7～Q10，满足条件，仍然能推断出【王】的工资。

推理控制

姓名	性别	家庭成员数	职业	工资
王	男	2	程序员	120
常	女	2	经理	240
陈	女	0	程序员	140
李	女	2	工程师	160
刘	男	2	前台	110
朱	女	1	培训	80
赵	男	0	教授	180
孙	男	1	培训	110
徐	女	2	程序员	130
马	女	1	程序员	150

Q7：SELECT COUNT(*) FROM S WHERE 性别='男'

结果： 4

Q8：SELECT COUNT(*) FROM S

WHERE 性别='男' and not 职业='程序员'

结果： 3 推断出：男程序员1人

Q9： SELECT sum(工资) FROM S WHERE 性别='男'

结果：520

Q10： SELECT sum(工资) FROM S

WHERE 性别='男' and not 职业='程序员'

结果：400 推断出：【王】的工资120

Q10中的条件谓词：性别='男' and not 职业='程序员'

在推断中起了很重要的作用，称为“个体追踪器”。

可以普遍化。

推理控制

对数据库中的表，几乎总可以找到它们的追踪器。

仅仅限制查询结果的元组数，是不够的。必须使所有查询结果经过各种组合的集合运算后，所得元组不小于某一下限，这样做，实现困难：

1）开销太大，难以检查；

2）对正常的使用也有限制；

3）为防止断续查询方式的窃密，不仅要考虑当前查询，还要考虑历史查询；

4）为防止多人合作窃密，不仅要考虑当前用户查询，还要考虑其它用户查询；

所有都考虑到检查到，这种方法不现实。

推理控制

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。

该约束下想获取用户B的工资至少查询次数:

$$1 + （N - 2）/ M$$

规则3：任一用户的查询次数不能超过1+(N-2)/M

其他安全性保护

➢数据脱敏

- ✓ 是指对某些敏感信息，通过脱敏规则进行数据的变形，实现敏感隐私数据的可靠保护。
- ✓ 在涉及客户安全数据或者一些商业性敏感数据的情况下，在不违反系统规则条件下，对真实数据进行改造并提供测试使用，如身份证号、手机号、卡号、客户号等个人信息都需要进行数据脱敏。

➢数据库漏扫

- ✓ 用户可以通过自动扫描和手动输入发现数据库，经授权扫描、非授权扫描、弱口令、渗透攻击等检测方式发现数据库安全隐患，形成修复建议报告提供给用户。

其他安全性保护

- 隐秘信道
- 数据隐私
- …