课程内容

- 数据库系统基本概念（数据模型，体系结构）
- 关系数据库
- 关系数据库标准语言SQL
- 数据库保护
- 关系数据理论
- 数据库设计
- 存储管理与索引
- 查询处理和查询优化
- 事务处理技术
- 数据库技术新发展

第一部分 基础理论

第二部分 设计理论

第三部分 实现技术

第四部分 新技术

第四章　数据库保护

- 本章主要内容
  - 数据库安全性控制
  - 数据库完整性控制
- 本章要阐述的问题
  - 1. 数据库中的数据会有哪些安全威胁，如何进行安全保护？
  - 2. 如何保证数据库中数据的正确、有效和一致性？

第四章　数据库保护

- 数据库安全性控制
- 数据库完整性控制

数据库安全性控制

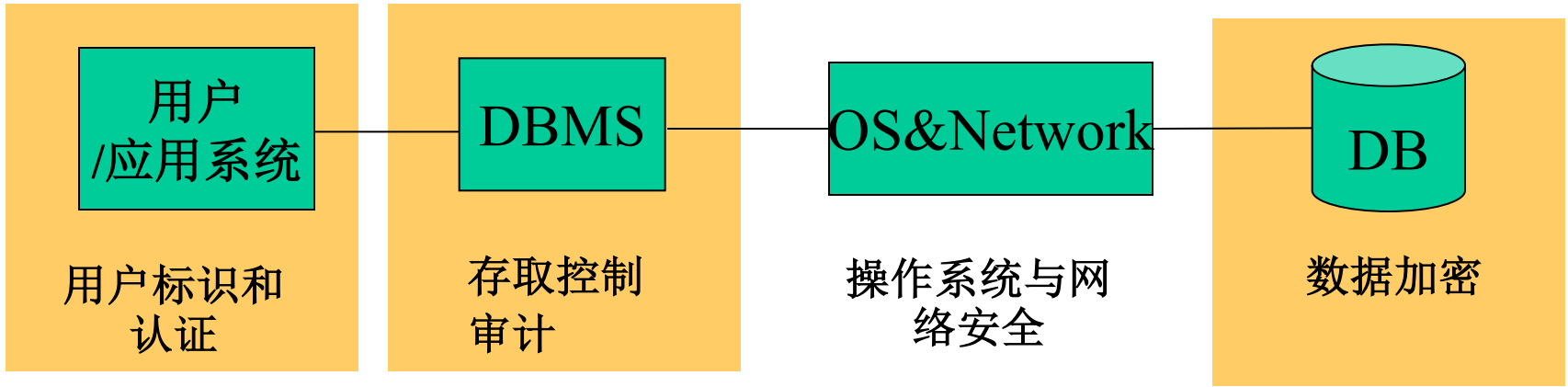
- 数据库安全性含义
- 数据库安全性控制
  - 用户标识与鉴别
  - 自主存取控制
  - 强制存取方法
  - 其它方法
- 可信计算机系统评测标准

数据库系统的安全性

- 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漏、更改和破坏。它包括两个方面的含义：
  - 向授权用户提供可靠的信息服务
  - 拒绝对数据的非授权存取访问请求，保证数据的可用性、完整性和一致性，进而保护数据库所有者和使用者的合法权益

数据库安全性控制

- 包含数据库系统的计算机系统安全模型：



- 用户标识与认证
- 访问控制（存取控制）
- 审计
- 加密技术
- 推理的控制
- 隐通道分析技术

用户标识与鉴别

- 用户标识和认证是系统提供的最外层安全保护措施。
  - 标识是指系统采用一定的方式标识其用户或应用程序的名字或身份。
  - 认证是指系统在用户或应用程序登录时判断其是否为合法的授权用户。
  - 常用的方法是采用用户名和口令。

存取控制

- 存取控制确保合法用户按照指定的权限使用DBMS和访问数据，而非法用户或不具有相关权限的用户则不能。
- 存取控制机制主要包括两个部分：
  - 用户权限定义：将用户权限记录到数据字典中，形成安全规则或授权规则。
  - 合法权限检查，每当用户发出数据库操作请求后，DBMS根据数据字典中的安全规则进行合法权限检查，决定是否接受用户的操作请求。
  - 用户权限定义和合法权限检查机制一起组成了DBMS的安全子系统。

存取控制方法分类

- 自主存取控制(discretionary access control，简称DAC)
  - 用户对于不同的数据对象拥有不同的存取权限，不同的用户对同一对象也有不同的权限，而且用户还可以将其拥有的权限转授给其他用户。
- 强制存取控制(mandatory access control，简称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。对于任一个对象，只有具有合法许可证的用户才可以存取。

自主存取控制

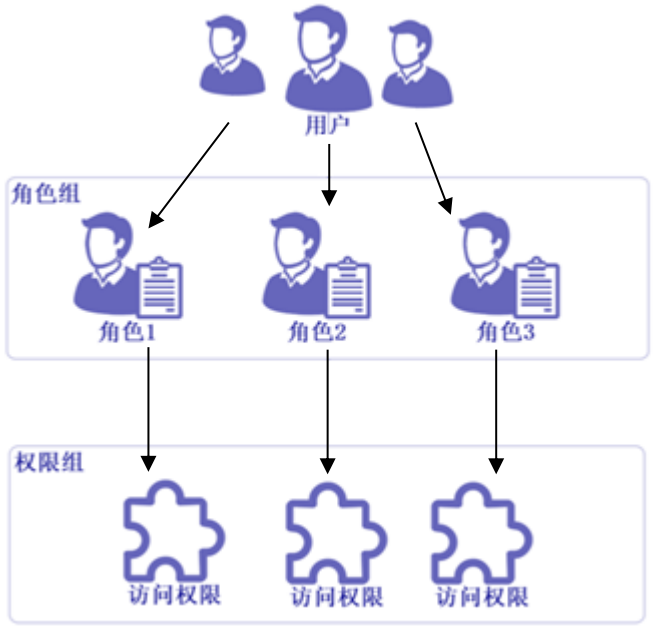
- 根据预先定义的用户权限进行存取控制。用户权限是指用户对数据对象允许执行的操作类型，由数据对象和操作类型两个要素组成。

	数据对象	操作类型
模式	模式	建立、修改、检索
	外模式	建立、修改、检索
	内模式	建立、修改、检索
数据	表	检索、插入、删除、修改
	属性列	检索、插入、删除、修改

- 对于用户存取权限的定义称为授权。在授权中应指明：用户名，数据对象名，允许的操作类型。

基于角色存取控制

- 角色是一组相关权限的集合，将角色授予用户就可以实现权限分配，用户访问时根据所拥有的角色权限进行存取控制——RBAC
- 使用角色控制用户对数据对象的权限，可以大大简化权限的管理



SQL的数据安全性控制

- 在SQL中可以授予用户两类权限：
  - 用户级权限
    - 是数据库管理员为每个用户授予的特定权限，是对用户使用整个数据库权限的限定。与整个数据库相关，与数据库中具体的关系无关。
  - 关系级权限
    - 是数据库管理员或数据库对象的拥有者为用户授予的与关系或视图有关的权限。这种权限是对用户使用关系和视图权限的限定。

用户级权限与角色的授予与收回

- 在SQL语言中，通过Grant语句授予用户用户级权限或角色，其语法格式为：

Grant <用户级权限>|<角色> [{,<用户级权限>|<角色>}]

To <用户名>|<角色>|public [{,<用户名>|<角色>}]

[With Grant Option]

数据库中的全部用户

允许被授权的用户将指定的用户级权限或角色授予其他用户

- 当要取消一个用户或角色的权限时，可以使用Revoke语句将其收回：

Revoke <用户级权限>|<角色> [{,<用户级权限>|<角色>}]

From <用户名>|<角色>|public [{,<用户名>|<角色>}]

示例

- 为用户授予用户级权限
  - Grant Create Session to SCOTT;
- 将权限授予角色
  - Grant Create table to Student_role;
- 取消用户SCOTT的Create Table权限。
  - Revoke Create Table From SCOTT;

关系级权限的授予与收回

- DBA和数据库对象所有者将这些数据库对象上的部分或全部权限授予其他用户。语法格式为：

  Grant ALL| <权限> [{,<权限>}]
  On <表名> | <视图名> [{,<表名> | <视图名>}]
  To {<用户> [{, <用户>}] | public}
  [With Grant Option]

- 回收权限
  - Revoke ALL|<表级权限> [{,<表级权限>}]
    On <表名>|<视图名> [{,<表名> | <视图名>}]
    From {<用户>[{,<用户>}]|PUBLIC}
  - 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。

示例

- 授予用户Liming在Student表上的Select和Insert权限。
  - Grant Select , Update On Student
    To Liming
    With Grant Option;
- 将Student表上的全部权限授予全体用户。
  - Grant ALL On Student To PUBLIC;
- 收回Liming对Student表的全部权限
  Revoke ALL On Student From Liming;

强制存取方法

- 在MAC中，DBMS所管理的全部实体被分为主体和客体两类。
  - 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、基本表、索引、视图等。
- 对于主体和客体，DBMS为他们每个实例指定一个敏感度标记(Label)。敏感度标记被分为若干级别，如绝密、机密、秘密、公开等。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。

强制存取方法

- MAC机制通过对比主体的Label和客体的Label，最终确定主体是否能够存取客体。
- 当某一主体以某一许可证级别注册入系统时，系统要求他对任何客体的存取必须遵循如下规则：
  - 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；
  - 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。

数据安全性控制的其它方法（1）

- 视图机制
  - 为不同的用户定义不同的视图，可以将用户对数据的访问限制在一定的范围内。
  - 例：限制王平只能检索Student表中计算机系学生的学号和姓名。

```
Create View CS_Student
        As Select Sno, Sname From Student
        Where Sdept = 'CS';
Grant Select On CS_Student To Wangping;
```

数据安全性控制的其它方法（2）

- 审计
  - 把用户对数据库的所有操作都自动记录下来放入审计日志中。DBA可以利用审计跟踪的信息，重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等；
- 数据加密
  - 防止数据库中数据在存储和传输中失密。加密的基本思想是根据一定的算法将原始数据（明文）变换为不可识别的格式（密文），从而使得不知道解密算法的人无法获知数据的内容。

可信计算机系统评测标准

- TCSEC（Trusted Computer System Evaluation Criteria)
  - 1985年，美国国防部制定了可信计算机评估标准TCSEC
- TDI/TCSEC
  - 1991年4月，美国国家计算机安全中心NCSC发布《可信计算机系统评估标准关于数据库系统的解释TDI（Trusted Database Interpretation）》,将TCSEC扩展到数据库管理系统;
- TDI与TCSEC从安全策略、责任、保证、文档四个方面描述了安全级别划分的指标。

可信计算机系统评测标准

安全性等级					主要特征
1	D	最低保护等级	D		非安全保护
2	C	自主保护等级	C1	自主安全保护	自主存取控制、审计功能
			C2	可控存取保护	比C1级更强的 自主存取控制、审计功能
3	B	强制保护等级	B1	标记安全保护	强制存取控制，敏感度标记
			B2	可结构化保护	形式化模型，隐蔽通道约束
			B3	安全区域保护	安全内核，高抗渗透能力
4	A	验证保护等级	A1	可验证保护	形式化安全验证，隐蔽通道分析
			超A1		

数据完整性控制

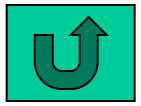
- 数据完整性含义
- 完整性约束条件
- 完整性控制

数据完整性控制

- 数据完整性是指数据的正确性和相容性。
  - 正确性是指数据应具有合法的类型，并在有效的取值范围之内。
  - 相容性是指同一对象在不同关系中的数据是符合逻辑的。
- 数据库能否保持完整性关系到数据库系统是否能够真实的反映现实世界，因此维护数据库的完整性十分重要。

数据完整性与数据安全性

- **数据完整性控制**是为了防止数据库中存在不符合语义的数据，防止错误信息的输入和输出；
- **数据安全性控制**是保护数据库防止恶意的破坏和非法存取；
- 安全性防范的是非法用户和非法操作，完整性措施的防范对象是不合语义的数据。



完整性约束条件

- 施加在数据库数据之上的语义约束条件称为数据库完整性约束条件。数据库系统依据完整性约束条件进行完整性检查。
- 完整性约束条件作用的对象可以是列、元组、关系三种
  - 列约束主要是列的类型、取值范围、精度等约束条件；
  - 元组约束是元组中各个字段间联系的约束；
  - 关系约束是若干元组间、关系之间的联系的约束。

完整性约束条件

- 关系模型中的三种完整性约束条件
  - 三种类型：实体完整性，参照完整性，用户自定义完整性

完整性约束条件类型	约束条件作用的对象	SQL支持方式
实体完整性	关系约束：一个关系元组之间的	PRIMARY KEY
参照完整性	关系约束：关系之间联系的约束	FOREIGN KEY
用户自定义完整性	列约束，元组约束	UNIQUE,NOT NULL,CHECK

完整性约束条件分类

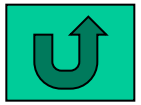
- 完整性约束可分为静态约束和动态约束。
  - 静态约束是指数据库在每一确定状态数据对象所应满足的约束条件，它是反映数据库状态合理性的约束。
  - 动态约束是指数据库从一种状态转变为另一种状态时，新、旧值之间所应满足的约束条件，它是反映数据库状态变迁的约束。

静态约束

- **静态列级约束**是对一个列的取值域的说明，包括对数据类型（包括数据类型、长度、单位、精度等）、数据格式、取值范围或取值集合、空值等的约束。
- **静态元组约束**规定了组成一个元组的各个列之间的约束关系。
- **静态关系约束**规定了一个关系的若干元组或者若干关系之间常常存在的各种联系或约束。包括：实体完整性约束、参照完整性约束、函数依赖、统计约束等。

动态约束

- 动态列级约束是修改列定义或列值时应满足的约束条件；
- 动态元组约束指修改元组值时元组中各个字段间需要满足的约束；
- 动态关系约束是加在关系变化前后状态上的限制条件。



完整性控制

- 数据库完整性控制应包括三个方面的功能：
  - 定义功能，提供定义完整性约束条件的机制。
  - 检查功能，检查用户发出的操作请求是否违背了完整性约束条件。
  - 违约响应，若违背了完整性约束条件，则采取一定措施来保证数据的完整性。

完整性检查的时机

- 完整性约束条件按照完整性检查的时机分为立即执行约束和延迟执行约束。
  - 立即执行约束是指在执行用户事务的过程中，在一条语句执行完后立即进行完整性约束的检查。若违背了完整性约束，系统将拒绝该操作。
  - 延迟执行约束是指在整个用户事务执行完毕后，再进行完整性约束的检查，结果正确方能提交。否则系统将拒绝整个事务。

完整性规则的表示

- 一条完整性规则可以用一个五元组（D，O，A，C，P）来描述，其中：
  - D（Data）约束所作用的数据对象
  - O（Operation）触发完整性检查的数据库操作。
  - A（Assertion）数据对象必须满足的断言或语义约束。
  - C（Condition）选择A作用的数据对象值的谓词。
  - P（Procedure）违反完整性规则时触发的过程。

示例

- 若有表TEACHER(编号，姓名，职称，工资，...)，表达：教授的工资不得低于5000元。
  - D： 工资
  - O： 插入或修改
  - A： 工资>=5000
  - C： 职称='教授'
  - P： 拒绝执行该操作

SQL的数据完整性支持-CREATE TABLE

- **CREATE TABLE 语句**

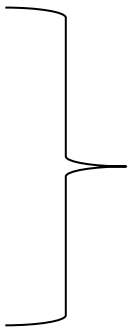
  Create Table <表名>

  (<列名><数据类型>[<列级完整性约束>]

  [{,<列名><数据类型>[<列级完整性约束>]}]

  [{, [<表级完整性约束>]}]);

  – 完整性约束
    - NULL/NOT NULL
    - UNIQUE
    - PRIMARY KEY
    - FOREIGN KEY
    - CHECK



- 列级约束
- 元组级约束
- 关系级约束

SQL的数据完整性支持-CREATE TABLE

- 示例

```
Create table S
    ( S#  char(5) not null unique,
      SN char(20) not null,
      SS char(2) check ( SS in ( 'M' ,  'F' ) ),
      SA int check (SA >=18 and SA <=45),
      SD char(15),
      Primary key(S#),
      Check ( SS =  'F'  or  SN  not like 'Ms.%')
    );
```

SQL的数据完整性支持-断言

- CREATE ASSERTION

  CREATE ASSERTION <断言名> <CHECK子句>

  – 定义涉及多个表或统计操作的比较复杂约束

  – 任何对断言中所涉及关系的操作都会出发断言检查，使断言不为真的操作将被拒绝

- 示例:每一门课程最多60人选修

```
CREATE ASSERTION ASSE-SC-SNUM1
    CHECK ( 60>= ALL( SELECT COUNT(*)
                      FROM SC
                      GROUP BY C#) );
```

SQL的数据完整性支持-触发器

- 触发器（Trigger)是用户定义在关系上的一类由事件驱动的特殊过程
- 对于用户对表的更新操作，系统自动激活相应触发器，执行完整性控制
- 定义触发器
  - CREATE TRIGGER <触发器名称>
    {BEFORE|AFTER}<触发器事件> ON <表名>
    REFERENCING NEW | OLD ROW AS <变量>
    FOR EACH {ROW|STATEMENT}
    [WHEN <触发条件>]
    <触发动作体>

小 结

- 数据库安全性控制与完整性控制的基本概念
- 安全性控制常用技术，包括用户标识与鉴别，自主存取控制，强制存取控制，以及视图、审计等
- 完整性约束条件的种类、表示及实现机制