数据库技术与应用

北京邮电大学计算机学院 肖达

xiaoda99@gmail.com

An Introduction to Database System

数据库保护

- **数据库安全性**
- 数据库完整性
- 数据库恢复技术

数据库安全性

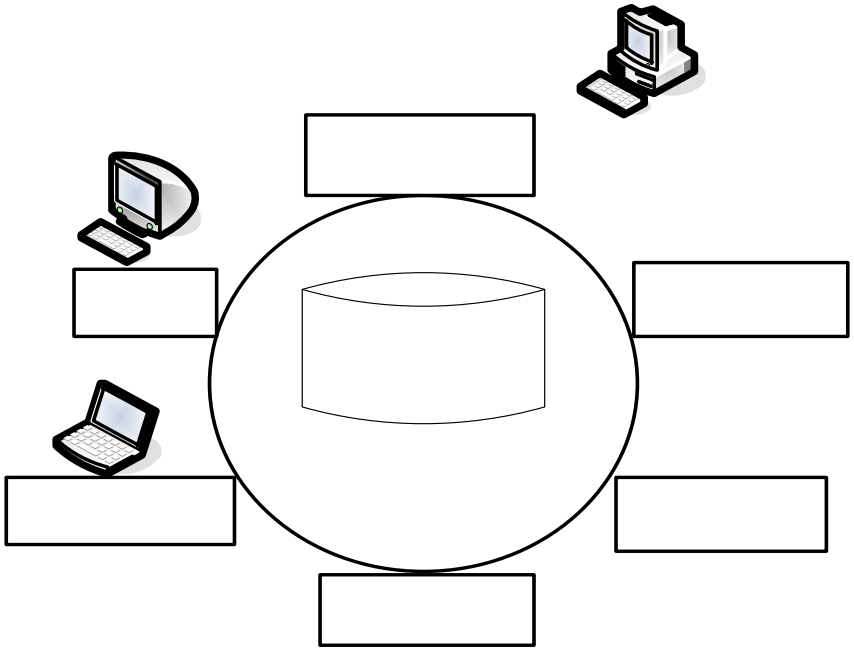
- 问题的提出
  - 数据库的一大特点是数据可以共享
  - 数据共享必然带来数据库的安全性问题
  - 数据库系统中的数据共享不能是无条件的共享

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

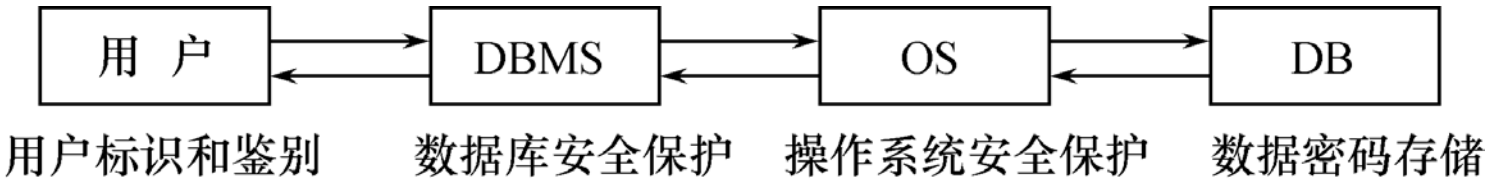
数据库安全性

对数据库安全的威胁



数据库安全性控制概述（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

用户标识与鉴别

存取控制

授权与回收

数据库角色

用户标识与鉴别

- 用户标识与鉴别（Identification & Authentication）
  - 系统提供的最外层安全保护措施
- 用户标识
  - 用户名、用户标识号
- 口令
  - 系统核对口令以鉴别用户身份
- 用户名和口令易被窃取
  - 每个用户预先约定好一个计算过程或者函数

数据库安全性控制

用户标识与鉴别

存取控制

授权与回收

数据库角色

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（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

自主存取控制方法

- 用户权限组成
  - 数据对象
  - 操作类型
- 定义用户存取权限：定义哪些用户可以在哪些数据库对象上进行哪些类型的操作
  - 是策略问题，DBMS应提供机制
- 数据库权限 vs 文件系统权限

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

数据库安全性控制

用户标识与鉴别

存取控制

授权与回收

数据库角色

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的用户
  - 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:
  - 指定：可以再授予
  - 没有指定：不能传播

例题

[例1] 把查询Student表权限授给用户U1

```
GRANT   SELECT
ON   TABLE   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 TABLE 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 TABLE 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 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 TABLE SC TO U6
WITH GRANT OPTION;

同样，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。

传播权限（续）

执行了［例1］到［例7］后，学生-课程数据库的用户权限定义表

授权用户名	被授权用户名	数据库对象名	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 TABLE Student
FROM U4;
```

[例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

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	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

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）

权限与可执行的操作对照表

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

数据库安全性控制

用户标识与鉴别

存取控制

授权与回收

数据库角色

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
  - 类比：Windows操作系统中的用户组

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>]…

数据库角色（续）

[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

```
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12]　角色的权限修改

```
GRANT DELETE
ON TABLE Student
TO R1
```

[例13]

```
REVOKE SELECT
ON TABLE Student
FROM  R1;
```

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（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）
  - 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
- 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

SQL Server的安全性管理

- 安全体系结构
- 权限管理
- 用户管理
- 角色管理

SQL Server的安全体系结构

(1) Windows NT操作系统的安全防线:网络管理员负责建立用户组，设置帐号并注册，同时决定不同的用户对不同系统资源的访问级别。

(2) SQL Server的运行安全防线: 通过另一种帐号设置来创建附加安全层。

(3) SQL Server数据库的安全防线: 特定数据库都有自己的用户和角色，该数据库只能由它的用户或角色访问，其他用户无权访问其数据。

(4) SQL Server数据库对象的安全防线: 对权限进行管理，TSQL的DCL功能保证合法用户既使进入了数据库也不能有超越权限的数据存取操作，即合法用户必须在自己的权限范围内进行数据操作。

用户认证

- 数据库管理系统级安全性控制—SQL Server 的登录认证
  - SQL Server支持两种认证方式
    - Windows认证模式
    - 混合认证模式

用户认证

- Windows认证模式
  - 利用Windows本身具有的管理登录、验证用户合法性的能力，以Windows的安全性来确定用户登录的账号是否正确
  - 连接SQL Server时不需在输入账号和密码
  - 在安装SQL Server后，系统中预设了两个个账户，分别为
    - BUILTIN\Administrators
    - sa

用户认证

- **混合认证模式**
  - 由SQL Server自身来执行验证工作
  - Windows认证和SQL Server 认证两种认证模式都可用
  - **SQL Server 认证模式**
    - SQL Server 自己执行认证处理
    - 用户在连接SQL Server 时必须提供登录名和密码
    - 登录信息存储在系统表syslogins中
    - 若输入的登录信息与系统表syslogins 中的某条记录相匹配则表明登录成功

SQL Server的权限管理

- **权限分类**
  - 对象权限和语句权限
- **对象权限**
  - 针对表、视图、存储过程
  - 决定了能对表、视图、存储过程执行哪些操作，包括UPDATE、DELETE、INSERT、EXECUTE
- **语句权限**
  - 用户是否具有权限来执行某一语句
  - 这些语句通常是一些具有管理性的操作，如创建数据库、表、存储过程等

SQL Server的权限管理

- 对象权限

对象	操作类型
表	SELECT、INSERT、UPDATE、DELETE、REFERENCES
视图	SELECT、UPDATE、INSERT、DELETE
存储过程	EXECUTE
列	SELECT、UPDATE

- 语句权限

CREATE DATABASE	创建数据库
CREATE TABLE	创建表
CREATE VIEW	创建视图
CREATE RULE	创建规则
CREATE DEFAULT	创建默认值
CREATE PROCEDURE	创建存储过程
BACKUP DATABASE	备份数据库
BACKUP LOG	备份日志

数据库用户的管理

(1) dbo用户

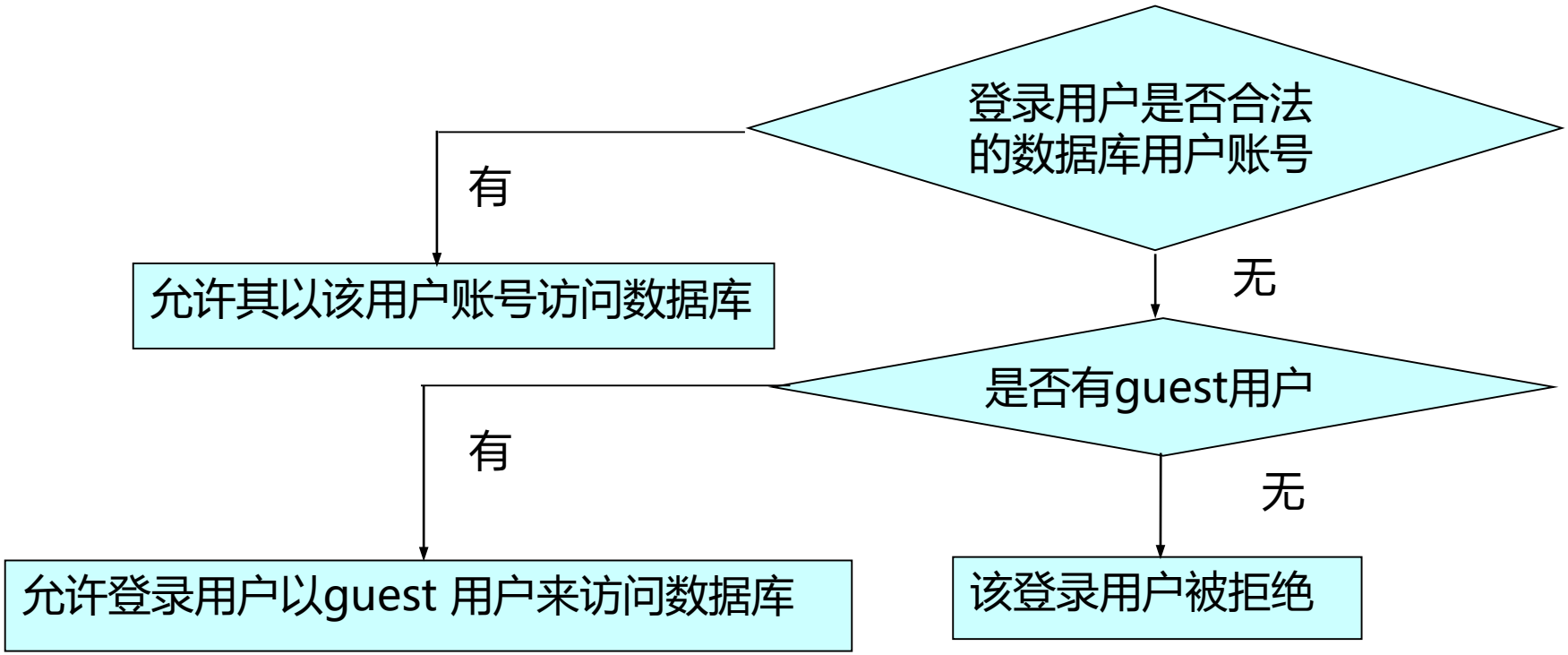
dbo用户即数据库拥有者或数据库创建者，dbo在其所拥有的数据库中拥有所有的操作权限。dbo的身份可被重新分配给另一个用户，系统管理员Sa可以作为他所管理系统的任何数据库的dbo用户。

(2) guest用户

如果guest用户在数据库存在，则允许任意一个登录用户作为guest用户访问数据库，其中包括那些不是数据库用户的SQL服务器用户。除系统数据库master和临时数据库tempdb的guest用户不能被删除外，其他数据库都可以将自己guest用户删除，以防止非数据库用户的登录用户对数据库进行访问。

数据库用户的管理

- guest用户



角色管理

- 用角色统一授予和管理权限
  - 服务器角色

固定服务器角色名	角色的权限
Sysadmin	可在SQL Server 中执行任何操作
Serveradmin	SQL Server服务器范围内的配置
setupadmin	增加、删除连接服务器，并执行某些系统存储过程
Securityadmin	管理数据库登录
processadmin	管理SQL Server进程
dbcreator	创建数据库并修改数据库
diskadmin	管理磁盘文件
Bulkadmin	执行BULK INSERT（大容量插入）操作

角色管理

- 使用存储过程管理服务器角色的成员

sp_addsrvrolemember	功能	将某一登录者加入到服务器角色中成为该角色的成员
	格式	sp_addsrvrolemember '登录者名', '服务器角色'
	举例	将登录者FirstUser加入到sysadmin 角色中 EXEC sp_addsrvrolemember 'FirstUser','sysadmin'
sp_dropsrvrrolemember	功能	将某一登录者从某一服务器角色中删除使之不再具有该服务器角色所赋予的权限
	格式	sp_dropsrvrolemember '登录者名','服务器角色'
	举例	EXEC sp_dropsrvrolemember 'FirstUser', 'sysadmin'

角色管理

固定数据库角色名	角色的权限
db_owner	数据库的所有者，可以执行任何数据库管理工作，可以对数据库内的任何对象进行任何操作，如删除、创建对象、将对象权限指定给其它用户。该角色包含下面各角色的所有权限
db_accessadmin	可增加或删除Windows认证模式下Windows用户或Windows用户组登录者以及SQL Server用户
db_datareader	能且仅能对数据库中所有表执行SELECT操作，以读取所有表数据
db_datawriter	能对数据库中所有表执行INSERT、UPDATE、DELETE操作，但不能进行SELECT操作
db_addladmin	可以新建、删除、修改数据库中任何对象
db_securityadmin	管理数据库内权限的GRANT、DENY和REVOKE，主要包括语句和对象权限，也包括对角色权限的管理
db_backupoperator	可以备份数据库
db_denydatareader	不能对数据库中任何表执行SELECT 操作
db_denydatawriter	不能对数据库中任何表执行UPDATE、DELETE和INSERT

数据库技术与应用

数据库完整性

数据库完整性

- 数据库的完整性
  - 数据的正确性和相容性
- 数据的完整性和安全性是两个不同概念
  - 数据的完整性
    - 防止数据库中存在不符合语义的数据，也就是防止数据库中存在不正确的数据
    - 防范对象：不合语义的、不正确的数据
  - 数据的安全性
    - 保护数据库防止恶意的破坏和非法的存取
    - 防范对象：非法用户和非法操作

数据库完整性(续)

- **例： 一条完整性规则**
  - 在更新“学生成绩”关系中的“成绩”字段时的完整性规则
  - 当更新“学生成绩.成绩”之后，判断“成绩是否>=0”，若不是，则进行相关的“规则违反”处理
  - 该规则规定：更新“学生成绩.成绩”时进行完整性检查，“成绩是否>=0”为检查条件，最后给出出错后的处理。
- 完整性规则也称完整性约束条件，用有关的语言进行描述，系统加以编译，放入数据库中，可以进行修改和删除

数据库完整性(续)

为维护数据库的完整性，DBMS必须：

- 提供定义完整性约束条件的机制
- 提供完整性检查的方法
- 违约处理

数据库完整性

实体完整性

参照完整性

用户定义的完整性

完整性约束命名子句

实体完整性定义

- 关系模型的实体完整性
  - CREATE TABLE中用PRIMARY KEY定义
- 单属性构成的码有两种说明方法
  - 定义为列级约束条件
  - 定义为表级约束条件
- 对多个属性构成的码只有一种说明方法
  - 定义为表级约束条件

实体完整性定义(续)

［例1］ 将Student表中的Sno属性定义为码

(1)在列级定义主码

```
CREATE TABLE Student
(Sno  CHAR(9)  PRIMARY KEY，
 Sname  CHAR(20) NOT NULL，
 Ssex  CHAR(2) ，
 Sage  SMALLINT，
 Sdept  CHAR(20)
);
```

实体完整性定义(续)

［例2］将SC表中的Sno，Cno属性组定义为码

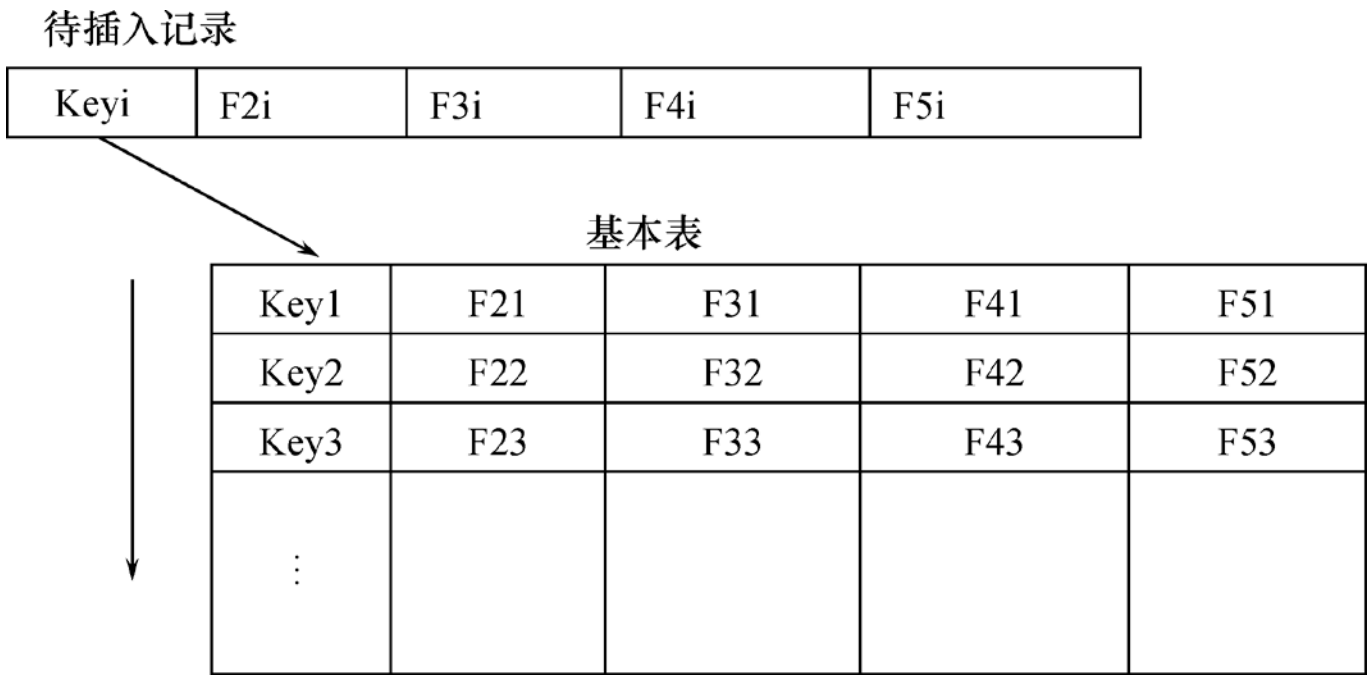
```
CREATE TABLE SC
    (Sno   CHAR(9)  NOT NULL，
     Cno  CHAR(4)  NOT NULL，
     Grade    SMALLINT，
     PRIMARY KEY (Sno，Cno)     /*只能在表级定义主码*/
    );
```

实体完整性检查和违约处理

- 插入或对主码列进行更新操作时，RDBMS按照实体完整性规则自动进行检查。包括：
  - 检查主码值是否唯一，如果不唯一则拒绝插入或修改
  - 检查主码的各个属性是否为空，只要有一个为空就拒绝插入或修改

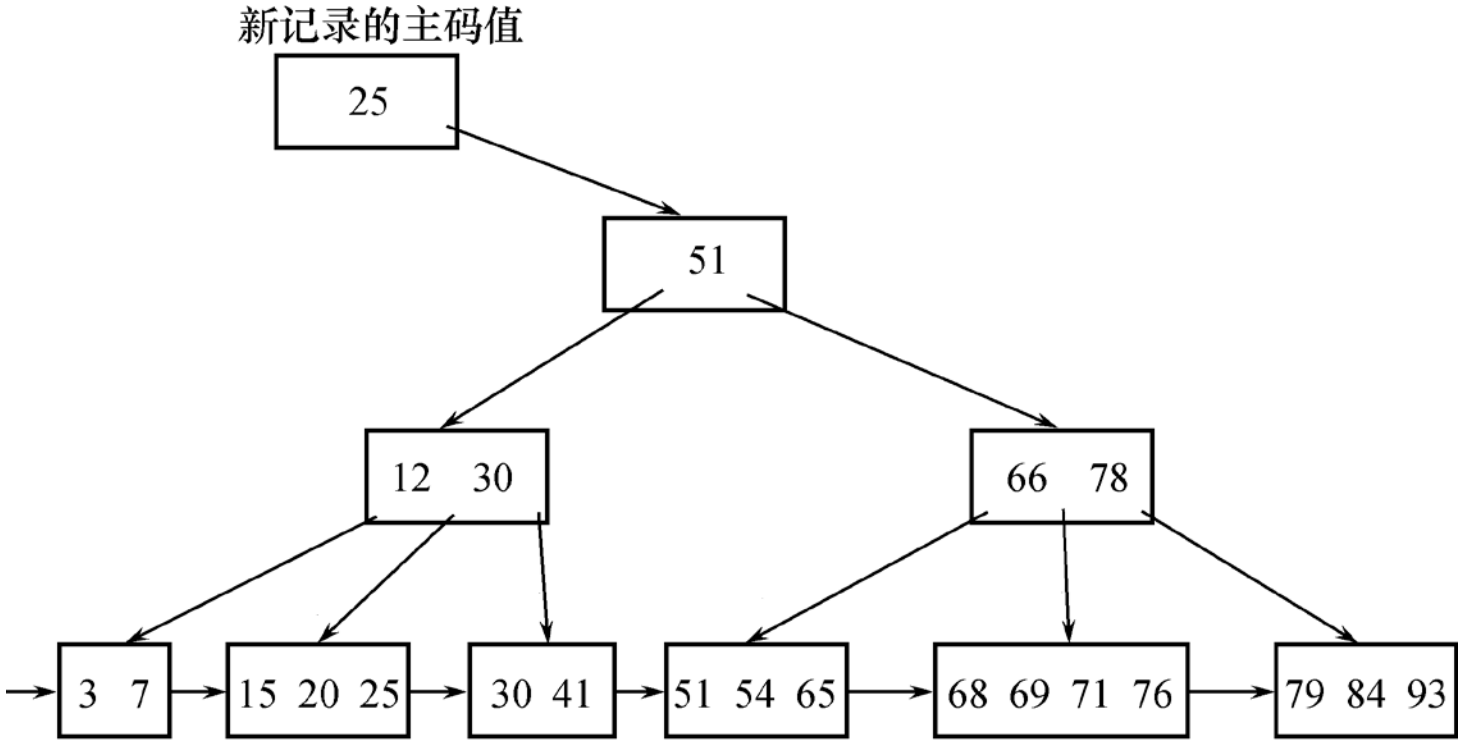
实体完整性检查和违约处理(续)

- 检查记录中主码值是否唯一的一种方法是进行全表扫描



实体完整性检查和违约处理(续)

- 另一个是索引



数据库完整性

实体完整性

参照完整性

用户定义的完整性

完整性约束命名子句

参照完整性定义

- 关系模型的参照完整性定义
  - 在CREATE TABLE中用FOREIGN KEY短语定义哪些列为外码
  - 用REFERENCES短语指明这些外码参照哪些表的主码

参照完整性定义(续)

例如，关系SC中一个元组表示一个学生选修的某门课程的成绩，（Sno，Cno）是主码。Sno，Cno分别参照引用Student表的主码和Course表的主码

［例3］ 定义SC中的参照完整性

```
CREATE TABLE SC
   (Sno    CHAR(9)  NOT NULL，
    Cno     CHAR(4)  NOT NULL，
    Grade    SMALLINT，
    PRIMARY KEY (Sno， Cno)，    /*在表级定义实体完整性*/
    FOREIGN KEY (Sno) REFERENCES Student(Sno)，
     /*在表级定义参照完整性*/
    FOREIGN KEY (Cno) REFERENCES Course(Cno)
    /*在表级定义参照完整性*/
);
```

参照完整性检查和违约处理

可能破坏参照完整性的情况及违约处理

被参照表（例如Student）	参照表（例如SC）	违约处理
可能破坏参照完整性 ←	插入元组	拒绝
可能破坏参照完整性 ←	修改外码值	拒绝
删除元组 →	可能破坏参照完整性	拒绝/级连删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级连修改/设置为空值

违约处理(续)

［例4］ 显式说明参照完整性的违约处理示例

```
CREATE TABLE SC
   (Sno   CHAR(9)  NOT NULL，
    Cno   CHAR(4)  NOT NULL，
    Grade  SMALLINT，
    PRIMARY KEY（Sno，Cno），
    FOREIGN KEY (Sno) REFERENCES Student(Sno)
        ON DELETE CASCADE     /*级联删除SC表中相应的元组*/
         ON UPDATE CASCADE，  /*级联更新SC表中相应的元组*/
    FOREIGN KEY (Cno) REFERENCES Course(Cno)
        ON DELETE NO ACTION
        /*当删除course 表中的元组造成了与SC表不一致时拒绝删除*/
        ON UPDATE CASCADE
       /*当更新course表中的cno时，级联更新SC表中相应的元组*/
   );
```

数据库完整性

实体完整性

参照完整性

用户定义的完整性

完整性约束命名子句

用户定义的完整性

- 用户定义的完整性就是针对某一具体应用的数据必须满足的语义要求
- RDBMS提供，而不必由应用程序承担

属性上的约束条件的定义

- CREATE TABLE时定义
  - 列值非空（NOT NULL）
  - 列值唯一（UNIQUE）
  - 检查列值是否满足一个布尔表达式（CHECK）

属性上的约束条件的定义(续)

- 不允许取空值

［例5］ 在定义SC表时，说明Sno、Cno、Grade属性不允许取空值。

```
CREATE TABLE SC
  （Sno  CHAR(9)  NOT NULL，
    Cno  CHAR(4)  NOT NULL，
    Grade  SMALLINT NOT NULL，
    PRIMARY KEY (Sno， Cno)，
    /* 如果在表级定义实体完整性，隐含了Sno，Cno不允许取空值，则在列级不允许取空值的定义就不必写了 * /
  ）；
```

属性上的约束条件的定义(续)

- 列值唯一

［例6］ 建立部门表DEPT，要求部门名称Dname列取值唯一，部门编号Deptno列为主码

```
CREATE TABLE DEPT
   (Deptno  NUMERIC(2)，
    Dname  CHAR(9)  UNIQUE，/*要求Dname列值唯一*/
    Location  CHAR(10)，
    PRIMARY KEY (Deptno)
   )；
```

属性上的约束条件的定义(续)

- 用CHECK短语指定列值应该满足的条件

［例7］ Student表的Ssex只允许取“男”或“女”。

```
CREATE TABLE Student
    (Sno  CHAR(9) PRIMARY KEY，
     Sname CHAR(8) NOT NULL，
     Ssex  CHAR(2)  CHECK (Ssex IN (‘男’，‘女’))，
        /*性别属性Ssex只允许取'男'或'女' */
     Sage  SMALLINT，
     Sdept  CHAR(20)
    );
```

数据库完整性

实体完整性

参照完整性

用户定义的完整性

完整性约束命名子句

完整性约束命名子句

- CONSTRAINT 约束

CONSTRAINT <完整性约束条件名>

［PRIMARY KEY短语　| FOREIGN KEY短语　| CHECK短语 ］

完整性约束命名子句(续)

［例10］ 建立学生登记表Student，要求学号在90000~99999之间，姓名不能取空值，年龄小于30，性别只能是“男”或“女”。

```
CREATE TABLE Student
    (Sno  NUMERIC(6)
      CONSTRAINT C1 CHECK (Sno BETWEEN 90000 AND 99999),
      Sname  CHAR(20)
      CONSTRAINT C2 NOT NULL,
      Sage  NUMERIC(3)
      CONSTRAINT C3 CHECK (Sage < 30),
      Ssex  CHAR(2)
      CONSTRAINT C4 CHECK (Ssex IN ( '男'，'女')),
      CONSTRAINT StudentKey PRIMARY KEY(Sno)
    );
```

- ✓ 在Student表上建立了5个约束条件，包括主码约束（命名为StudentKey）以及C1、C2、C3、C4四个列级约束。

完整性约束命名子句(续)

- 修改表中的完整性限制
  - 使用ALTER TABLE语句修改表中的完整性限制

完整性约束命名子句(续)

［例13］ 修改表Student中的约束条件，要求学号改为在900000~999999之间，年龄由小于30改为小于40

- 可以先删除原来的约束条件，再增加新的约束条件

```
ALTER TABLE Student
    DROP CONSTRAINT C1;
ALTER TABLE Student
    ADD CONSTRAINT C1 CHECK (Sno BETWEEN 900000 AND 999999);
ALTER TABLE Student
    DROP CONSTRAINT C3;
ALTER TABLE Student
    ADD CONSTRAINT C3 CHECK (Sage < 40)；
```

实验3 数据控制

- 实验目的
  - 熟悉通过SQL对数据库进行数据控制，包括安全性、完整性和数据库恢复
- 实验工具
  - SQL Server提供的交互查询工具
- 实验数据库
  - 自行设计（至少三个表）

实验内容1：安全性部分

- 授权与回收
  - 在自行设计数据库中由DBA创建若干用户，权限全部选择为CONNECT（SQL Server中的db_accessadmin角色）
  - 仿照教材4.2.4 [例1]~[例10]，在DBA与这些用户之间进行授权和回收，**并查看效果（4用例）**
    - ⑩注意SQL Server中登录名与用户的区别
- 数据库角色（MySQL可不做）
  - 仿照教材4.2.5 [例11]，创建一个角色（sp_addrole）并赋予权限，将角色授予用户（sp_addrolemember），**并查看效果（1用例）**

实验内容2：完整性部分

- 使用SQL对数据进行完整性控制，**并用实验证实，当操作违反了完整性约束条件时，系统是如何处理的（各1用例）**
  - 实体完整性(仿照[例1]、[例2])
  - 参照完整性(仿照[例3])
  - 用户定义完整性(仿照[例5]、 [例6])
  - CHECK短语(仿照[例7]或 [例8] 、[例9])
  - CONSTRAINT子句(仿照[例10]、 [例13])