

## 第4章

# 数据库的安全性与完整性保护

## □ 为什么要提供数据库的安全性保护和完整性保护功能？

### ➤ 常用的保护措施

- 计算机系统外部的

- ❖ 环境的保护

- ❖ 社会的保护

- ❖ 设备的保护

- 计算机系统内部的

- ❖ 计算机

- ❖ 操作系统

- ❖ 数据库

- ❖ 网络

- ❖ 应用系统

- 在上述这些保护措施中，数据库系统中的数据保护是至关重要的。在本章中我们主要讨论在数据库管理系统内部设置一些必要的软件以达到数据保护的  
目的，这就是数据库的保护
- 数据库管理系统提供的保护措施主要有两类：
  - 数据库的安全性（**security**）保护
  - 数据库的完整性（**integrity**）保护

# 4.1 数据库的安全性

## 4.1.1 数据库的安全与安全数据库

## 4.1.2 数据库安全的基本概念与内容

## 4.1.3 数据库的安全标准

## 4.1.4 SQL对数据库安全的支持

## 4.1.1 数据库的安全与安全数据库

### ➤ 数据库的安全（database security）

- 防止非法使用数据库。即要求数据库的用户：

- ❖ 通过规定的访问途径

- ❖ 按照规定的访问规则（规范）

来访问数据库中的数据，并接受来自DBMS的各种检查

- 访问数据库的规范有多种，但是：

- ❖ 不同的规范适用于不同的应用

- ❖ 可以根据应用的不同需求来选择不同的数据库访问规范，即选择具有不同安全级别的数据库

- ❖ 那些能适应网络环境下安全要求级别的数据库称为安全数据库（secure database），或称为可信数据库（trusted database）

# 4.1 数据库的安全性

## 4.1.1 数据库的安全与安全数据库

## 4.1.2 数据库安全的基本概念与内容

## 4.1.3 数据库的安全标准

## 4.1.4 SQL对数据库安全的支持

## 4.1.2 数据库安全的基本概念与内容

### ❑ 数据库安全的基本概念与内容

- ▶ 可信计算基TCB (**trusted computing base**)
- ▶ 主体 (**subject**)、客体 (**object**) 与主客体分离
- ▶ 身份标识与鉴别 (**identification & authentication**)
- ▶ 自主访问控制 (**discretionary access control**)
- ▶ 强制访问控制 (**mandatory access control**)
- ▶ 数据完整性 (**data integrity**)
- ▶ 隐蔽通道 (**hiding canal**)
- ▶ 数据库安全的形式化模型 (**formulization of database security**)
- ▶ 审计 (**audit**)
- ▶ 访问监控器 (**access monitor**)



## ❑ 可信计算基TCB

- 它是为实现数据库安全所采用的所有实施策略与机制的集合
- 它是实施、检查、监督数据库安全的机构



# 数据库安全的基本概念与内容(续)

## ➤ 主体、客体与主客体分离

### ■ 客体

❖ 数据库中的数据及其载体

❖ 如：表、视图、快照、存储过程、数据文件等

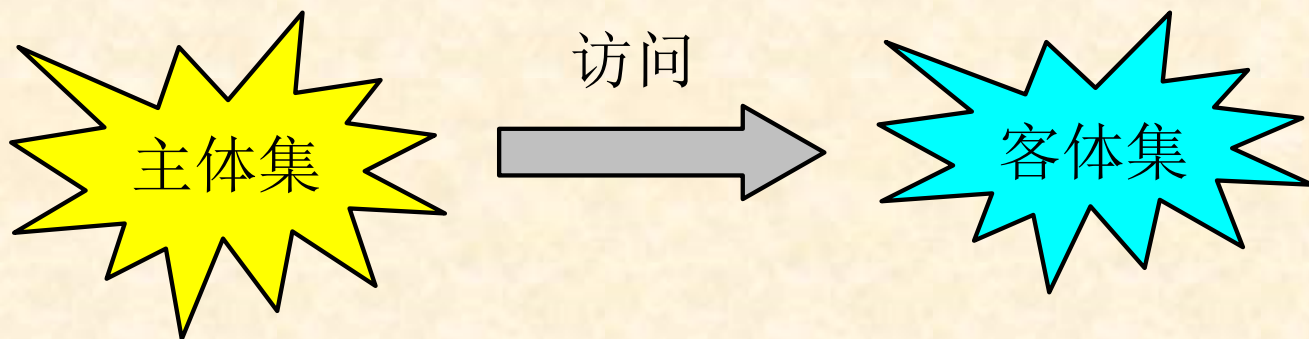
### ■ 主体

❖ 数据库中数据的访问者

# 数据库安全的基本概念与内容(续)



- 与数据库安全有关的所有实体都可以被抽象为**主体**或**客体**。由所有主体构成的主体集合与由所有客体构成的客体集合之间存在着一种单向访问关系（如图所示）



- 数据库安全就是研究有关实体的主/客体划分以及主/客体之间的访问关系的控制

# 数据库安全的基本概念与内容(续)

## □ 身份标识与鉴别

- 在数据库安全中，主体访问客体时需进行一定的安全控制与检查，目前存在三种控制方式：
  - 身份标识与鉴别
  - 自主访问控制
  - 强制访问控制
- 其中身份标识与鉴别是系统提供的最外层安全保护措施
- 每个主体必须有一个标志自己身份的标识符（以区别不同的主体）以及一个用以验证其身份的访问口令。当主体访问客体时，**TCB** 将对主体所提交的身份标识符和口令进行鉴别，以阻止非法访问

# 数据库安全的基本概念与内容(续)

## ➤ 常用的控制措施

### — 登录控制

- 对用户标识符和口令的鉴别
- 限制重复性登录失败次数
- 保证登录的可信路径（本地或远程）
- 限制在一天当中允许访问的时间

### — 口令字选取

- 控制用户口令字选取
  - » 如：最小长度，组合，历史等
- 某些口令可由系统生成再分配给用户
- 强制实施口令字有效期机制

# 数据库安全的基本概念与内容(续)

## — 对鉴别数据的保护

- 输入口令字时不许回显
- 避免未授权的访问与修改
- 防止重播攻击
- 防止伪造或拷贝
- 防止重用
  - » 如：只允许单次使用的口令字
- 提供口令字修改的可信路径

## — 会话连接挂起

- 在用户停止操作一段时间后挂起
- 根据用户要求挂起
- 在用户停止操作一段时间后终止

## — 用户帐号和属性文件

- 控制对用户帐号的生成、删除、激活或停用
- 定义用户属性文件中所包含的安全属性
- 控制对用户属性文件的修改



# 数据库安全的基本概念与内容(续)

## ❑ 自主访问控制 (DAC)

- 是一种基于存取矩阵的安全控制模型
- 此模型由主体、客体和存/取操作三部分内容构成了一个矩阵 (如图所示)

	主体1	主体2	.....	主体i	.....
客体1	.....	.....	.....	读/写	.....
客体2	.....	.....	.....	读/修改	.....
.....	.....	.....	.....	.....	.....
客体j	插入	修改/删除	.....	读/插入/删除	.....
.....	.....	.....	.....	.....	.....

存取矩阵模型图



# 数据库安全的基本概念与内容(续)

## ➤ 基于存取矩阵的自主访问控制功能

### — 安全政策范围验证

- 主体、客体以及操作

### — 控制主体访问客体的规则验证

- 主体按存取矩阵模型的要求访问客体
- 凡不符合存取矩阵要求的访问均属非法访问

### — 超越DAC的特权验证

- 如数据库管理员、客体的属主等

## ➤ 对**DAC**属性的控制

- 改变客体的许可权限
- 对新建客体的缺省保护
- 改变客体的属主
- 改变用户组的隶属关系

## ➤ **DAC**的特点

- 存取矩阵中的元素是可以随意改变的
- 主体可以通过授权（**Grant**）/回收（**Revoke**）操作变更某些操作权限
- 适合于单机方式下的访问控制

# 数据库安全的基本概念与内容(续)

## ❑ 强制访问控制 (MAC)

- 是主体访问客体的一种强制性的安全控制方式，主要用于网络环境，对网络中的数据库安全实体作统一的、强制性的访问管理
- 主/客体标记 (label)
  - 安全级别标记 (label of security level)
    - 规定了主/客体的安全级别
  - 安全范围标记 (label of security category)
    - 规定了主体可以访问的范围 (客体所处的范围)
- 在主体访问客体的过程中，主体与客体的标记必须满足系统所采用的强制访问控制策略的要求，否则将被视为非法访问

# 数据库安全的基本概念与内容(续)

- 强制访问控制的实施机制: **Bell-Lapadula**模型
  - 每个标记由一个整数的分层密级和一个非分层范畴的集合构成。当主体访问客体时必须满足以下条件:
    - (下读) 仅当主体的分层密级大于或等于客体的分层密级, 且主体的非分层范畴集合包含或等于客体的非分层范畴集合时, 主体才能读客体
    - (上写) 仅当主体的分层密级小于或等于客体的分层密级, 且主体的非分层范畴集合被包含或等于客体的非分层范畴集合时, 主体才能写客体
- 强制访问控制中的主、客体标记由专门的安全管理员设置, 任何主体均无权设置与授权, 它体现了在网上对数据库安全的强制性与统一性

- 强制性的访问控制措施
  - 安全政策范围验证
    - 主体、客体以及操作
  - 验证控制访问和信息流动的规则
  - 验证可超越**MAC**特权
  
- 对**MAC**属性的控制
  - 改变客体标记
  - 对新建客体的缺省标记
  - 改变主体安全属性



## □ 数据完整性

- 防止非法使用插入 (**insert**)、删除 (**delete**)、修改 (**update**) 等影响数据完整性的操作
- 控制手段
  - 对存储数据错误的检测
  - 事务回卷功能
- 常用的控制手段：三类数据完整性
  - 实体完整性
  - 关联完整性
  - 用户定义完整性约束

## □ 隐蔽通道

- 在主体对客体的访问过程中，可通过自主及强制访问控制措施来提供安全保护
- 公开通道
  - 正规的、接受**TCB**的（自主/强制）访问控制检查的访问通道
- 隐蔽通道
  - 非正规的、不受 **TCB** 控制的访问通道
- 为了真正保证数据的安全性，就必须分析、发现隐蔽通道，防止隐蔽通道的产生



## ❑ 数据库安全的形式化模型

- 用数学形式对数据安全模型的安全策略作形式化描述、验证与证明，形成严格的形式化体系
- 建立一个数据库安全的形式化模型，有利于发现并填补安全漏洞，防止隐蔽通道，并为数据安全的进一步研究提供理论基础

# 数据库安全的基本概念与内容(续)

## □ 审计

- 跟踪记录用户对数据的访问操作：
  - 访问时间/访问内容/用户名/终端名/操作类型/操作结果
  - 并可根据审计结果给出报警信息
- 由于执行审计操作需要额外的时间和空间开销，因此在**DBMS**中，‘审计’通常是一个可选择的安全保护手段，主要用于安全性要求较高的部门

# 数据库安全的基本概念与内容(续)

## ➤ 与审计有关的保护措施有：

### — 审计事件控制

- 对可审计事件和需记录信息的说明
- 对需审计事件选择的控制
- 对单个身份审计

### — 入侵检测和应对

- 生成报警信息，并对即将发生的安全违规行为采取应对措施
- 定义用于指示潜在或即将发生的安全违规行为的那些规则、事件、事件序列或系统使用模式

## — 审计记录的保护

- 避免数据丢失

- » 如： 审计记录饱和、操作中中断等

- 避免未授权的修改 / 访问

## — 审计记录的分析 / 查阅

- 提供相应的分析 / 查阅工具

## □ 访问监控器

- 上述的安全策略须有一个网络中的实体来完成，即访问监控器
  - 访问监控器是一个独立的（既非主体，亦非客体）、最小的、抗篡改的、自主机构，用来监控主体和客体之间的授权访问关系
- **TCB**是一个抽象的功能/策略集合，而访问监控器则是一个客观存在的实体，是**TCB**在网络中的实现

# 4.1 数据库的安全性

## 4.1.1 数据库的安全与安全数据库

## 4.1.2 数据库安全的基本概念与内容

## 4.1.3 数据库的安全标准

## 4.1.4 SQL对数据库安全的支持



## 4.1.3 数据库的安全标准

### ❑ 安全标准的制订历史

- 美国(早期): 可信计算机系统评估标准TCSEC  
(Trusted Computer System Evaluation Criteria)

❖ 1970年由美国国防科学委员会提出, 1985年公布为国防部标准, 后扩展至民用

- 安全级别从低到高分为四类七级 (D, C<sub>1</sub>, C<sub>2</sub>, B<sub>1</sub>, B<sub>2</sub>, B<sub>3</sub>, A)



## 4.1.3 数据库的安全标准

### ❑ 安全标准的制订历史（续）

#### ➤ 中国

– **1999年：计算机信息系统安全保护等级划分准则（GB 17859-1999）**

▪ 参照美国的**TCSEC**标准

GB 17859-1999 分级	TCSEC 分级
-	D
第 1 级：自主安全保护级	C1
第 2 级：系统审计保护级	C2
第 3 级：安全标记保护级	B1
第 4 级：结构化保护级	B2
第 5 级：访问验证保护级	B3
-	A

## 4.1.3 数据库的安全标准

### ❑ 计算机信息系统安全保护等级划分准则（GB 17859-1999）

- ▶ 第1级：用户自主保护级
- ▶ 第2级：系统审计保护级
- ▶ 第3级：安全标记保护级
- ▶ 第4级：结构化保护级
- ▶ 第5级：访问验证保护级

# 4.1 数据库的安全性

## 4.1.1 数据库的安全与安全数据库

## 4.1.2 数据库安全的基本概念与内容

## 4.1.3 数据库的安全标准

## 4.1.4 SQL对数据库安全的支持

## 4.1.4 SQL对数据库安全的支持

□ 在SQL'92中提供了C<sub>1</sub>级数据库安全的支持，它们是：

- 主体、客体及主/客体分离
- 身份标识与鉴别
- 数据完整性
- 自主访问控制与授权功能

## 4.1.4 SQL对数据库安全的支持

### ❑ 自主访问控制与授权功能

➤ **SQL**中的自主访问控制是通过（用户，操作对象，操作权限）这样的三元组来定义用户对于数据的访问权限的，并可通过授权（**Grant**）和回收（**Revoke**）语句来改变用户的访问权限

#### ➤ 操作权限

❖ **SELECT**权

❖ **INSERT**权

❖ **DELETE**权

❖ **UPDATE**权

❖ **REFERENCE**权

❖ **EXECUTE**权

❖ **USAGE**权

## 4.1.4 SQL对数据库安全的支持

### ➤操作对象

- 表，视图
- 属性
- 域（**type**），**UDT**（用户定义数据类型）
- 存储过程/函数，触发器

### ➤用户

- 数据库用户

## 4.1.4 SQL对数据库安全的支持

### ➤授权语句

**GRANT** <操作权限列表> **ON** <操作对象>  
**TO** <用户名列表> [**WITH GRANT OPTION**]

—例:

- **grant SELECT, UPDATE on S**  
**to XULIN with grant option**

- **grant UPDATE (G) on SC to XULIN**



## 4.1.4 SQL对数据库安全的支持

### ➤ 回收语句

**REVOKE** <操作权限列表> **ON** <操作对象>  
**FROM** <用户名列表> [**RESTRICT** | **CASCADE**]

- **CASCADE**: 连锁回收
- **RESTRICT**: 在不存在连锁回收问题时才能回收权限, 否则拒绝回收

### – 例:

▪ **revoke UPDATE on S from XULIN cascade**

## 4.2 数据库的完整性

## 4.2 数据库的完整性

### □ 数据库的完整性

- 指数据库中数据的正确性和一致性，包括：
  - 正确性：数据的有效性、有意义
  - 一致性：在多用户（多程序）并发访问数据库的情况下，保证对数据的更新不会出现与实际不一致的情况
- 我们一方面要避免在数据库中出现错误的数据，即防止数据的完整性受到破坏。同时，如果因为某些不可抗拒的原因而导致数据库中的数据被破坏，也要能够及时发现并采取一定的措施将数据库中的数据恢复到正确的状态下去

## 4.2 数据库的完整性

### □ 完整性保护

- 对数据库中数据的正确性和一致性的维护，包括：
  - 在执行更新操作时，检查是否违反完整性约束条件，并且在证明其无效后作出适当的反应
  - 防止有存取权的合法用户的误操作

### □ 完整性保护的目

- 及时发现错误
- 能够采取措施防止错误的进一步蔓延
- 最终将数据库恢复到正确状态

### □ 完整性保护的实现措施

- 完整性约束条件的定义及检查
- 触发器
- 并发控制技术

## 4.2 数据库的完整性

### 4.2.1 数据库完整性保护的功能

### 4.2.2 完整性规则的三个内容

### 4.2.3 完整性约束的设置、检查与处理

### 4.2.4 触发器

## 4.2.1 数据库完整性保护的功能

### □ 三个基本功能

#### ➤ 设置功能

- 系统及用户对数据库完整性的基本要求

#### ➤ 检查功能

- 有能力检查数据库中的数据是否有违反约束条件的现象出现

#### ➤ 处理功能

- 出现违反约束条件时，有及时处理的能力



## 4.2 数据库的完整性

### 4.2.1 数据库完整性保护的功能

### 4.2.2 完整性规则的三个内容

### 4.2.3 完整性约束的设置、检查与处理

### 4.2.4 触发器

## 4.2.2 完整性规则的三个内容

- ❑ 在关系数据库系统中提供了下述三类数据完整性约束：
  - 实体完整性规则
    - 在一个基表的主关键字（主码）中，其属性的取值不能为空值
  - 参照完整性规则
  - 用户定义的完整性规则
    - 由用户来定义的数据完整性要求
- 由数据库管理系统来提供数据完整性约束条件的定义和检查的优点？

## 4.2.2 完整性规则的三个内容

### ➤ 参照完整性规则

- 若关系**R**中有属性集**F**与另一个关系**S**的主关键字**K<sub>s</sub>**相对应（即具有相同的语义和取值范围），则称关系**R**引用关系**S**中的元组
  - 这样的属性集**F**被称为关系**R**的外关键字（关系**R**和关系**S**可以是同一个关系）
- 参照完整性规则要求
  - 关系**R**中的每个元组在外关键字**F**上的值或者是空值（**NULL**），或必须引用在关系**S**中存在的元组，即不能引用不存在的实体

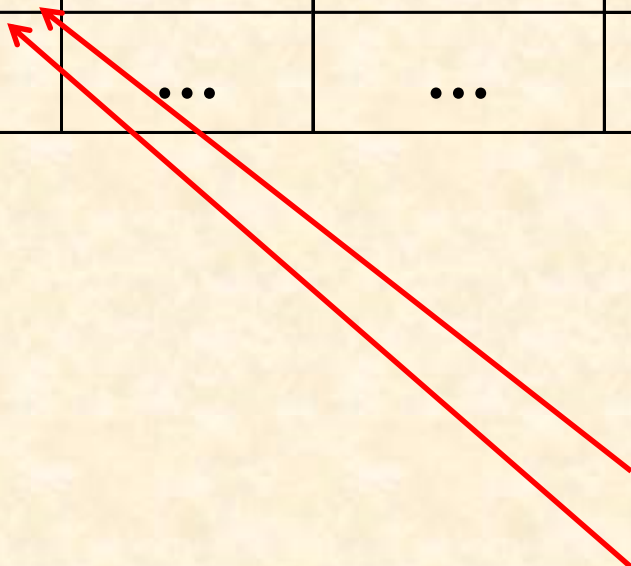
## 4.2.2 完整性规则的三个内容

学生关系

学 号	姓 名	系 别	年 龄
...	...	...	...
<b>993501</b>	<b>刘英</b>	<b>计算机</b>	<b>18</b>
...	...	...	...

选课关系

学 号	课程号	成绩
...	...	...
<b>993501</b>	<b>CS101</b>	<b>85</b>
<b>993501</b>	<b>EN103</b>	<b>90</b>
...	...	...



## 4.2.2 完整性规则的三个内容

职工关系

工作证号	姓名	职务	年龄	工资	上级领导的工作证号
E012	李 强	车间主任	53	1200	E001
E025	王 英	技 工	27	800	E012
E001	张伟一	厂 长	49	3000	NULL

其中：‘工作证号’是主码，‘上级领导的工作证号’是外码

# 外键完整性约束示例



学生关系

学 号	姓 名	系 别	年 龄
...	...	...	...
993501	刘英	计算机	18
...	...	...	...

选课关系

学 号	课程号	成绩
...	...	...
993501	CS101	85
993501	EN103	90
...	...	...

对‘选课’关系中的外键‘学号’的引用完整性约束定义：

Foreign key(学号) References 学生 On Delete **Cascade**;

Foreign key(学号) References 学生 On Delete **Restrict**;

Foreign key(学号) References 学生 On Delete **Set Null**;

Foreign key(学号) References 学生 On Delete **No Action**;




# 外键完整性约束

## □ “restrict”和 “no action” 的区别

PAR

PID	NAME	CID	CDESC	PID
1	PAR1	51	51DESC	1
2	PAR2	52	52DESC	2
3	PAR3			



UPDATE PAR SET PID=PID-1

- “restrict”报错
- “no action”可以执行

## 4.2 数据库的完整性

### 4.2.1 数据库完整性保护的功能

### 4.2.2 完整性规则的三个内容

### 4.2.3 完整性约束的设置、检查与处理

### 4.2.4 触发器

## 4.2.3 完整性约束的设置、检查与处理

- 一条完整性约束规则一般有三个组成部分
  - 完整性约束条件的设置
  - 完整性约束条件的检查
    - 在**DBMS**内部设置专门的软件检查模块
  - 完整性约束条件的处理
    - 在用户的操作会破坏数据的完整性（即违反完整性约束条件的要求）时，系统将：
      - 拒绝执行，并报警或报错；
      - 调用相应的函数（例程）进行处理，如：
        - » 在外键定义子句中给出的处理方法
        - » 在触发器中给出的处理过程

## 4.2.3 完整性约束的设置、检查与处理

### □ 完整性约束条件设置

- 属性级的约束（域约束）
  - 数据类型的约束，非空值约束，取值范围的约束
- 元组级的约束（表约束）
  - 主码定义，候选码（唯一键）定义
  - 外码定义
  - 基于元组的检查子句：属性间关系的定义
- 全局约束（断言**assertion**）
  - 单个关系中涉及到统计操作的约束条件
  - 多个关系之间复杂的约束条件

### □ 对约束命名

- **CONSTRAINT** <约束名> <完整性约束定义子句>

# SQL语言对完整性约束规则的支持

## ❑ 创建基表命令 (CREATE TABLE)

### ➤ 需要定义的内容

— 模式名 & 表名

— 属性的定义

- 属性名 & 数据类型

- 属性的缺省值定义

**DEFAULT { default\_constant | NULL }**

- 属性级的数据约束定义

— 表级的数据约束定义

# SQL语言对完整性约束规则的支持

## ❑ 属性级的约束

➤ 数据类型的约束，取值范围的约束，非空值约束

```
{ NOT NULL |  
[ CONSTRAINT constraint_name ]  
    UNIQUE  
    | PRIMARY KEY  
    | CHECK ( search_condition )  
    | REFERENCES table_name [ ( column_name ) ]  
        [ ON DELETE CASCADE | RESTRICT | SET NULL ]  
        [ ON UPDATE CASCADE | RESTRICT | SET NULL ] }
```

基于单个属性的取值约束



# SQL语言对完整性约束规则的支持

[ CONSTRAINT **constraint\_name** ]

{ UNIQUE ( colname { , colname ... } )

| PRIMARY KEY ( colname { , colname ... } )

| CHECK ( **search\_condition** )

| FOREIGN KEY ( colname { , colname ... } )

REFERENCES **table\_name** [ ( colname { , colname ... } ) ]

[ ON DELETE CASCADE | RESTRICT | SET NULL ]

[ ON UPDATE CASCADE | RESTRICT | SET NULL ] }

基于多个属性的取值约束

# SQL语言对完整性约束规则的支持

## ❑ 在定义属性级的数据约束时需要考虑的内容

### ➤ NOT NULL vs. DEFAULT NULL

### ➤ Constraint name

- 对某个数据约束条件进行命名（可选项）
- 以利于以后使用**ALTER TABLE**命令来修改表中的数据约束定义

### ➤ UNIQUE vs. NOT NULL

- **UNIQUE**属性可以取空值
- 候选键（**candidate key**）：

**UNIQUE + NOT NULL**

# SQL语言对完整性约束规则的支持

## ❑ 在定义属性级的数据约束时需要考虑的内容（cont.）

### ➤ PRIMARY KEY vs. NOT NULL

### ➤ REFERENCES

- FOREIGN KEY（外键） vs. PRIMARY KEY（主键）
- 外键上的取值约束及其一致性的保证措施
  - CASCADE | RESTRICT | SET NULL

### ➤ CHECK

# SQL语言对完整性约束规则的支持

## □ 元组级的约束

➤ 主码定义: **PRIMARY KEY( <column-list> )**

➤ 唯一键定义: **UNIQUE( <column-list> )**

➤ 外码定义:

**FOREIGN KEY ( <fk-column-list> )**

**REFERENCES <table-name> ( <pk-column-list> )**

**[ ON UPDATE [ RESTRICT | CASCADE | SET NULL ] ]**

**[ ON DELETE [ RESTRICT | CASCADE | SET NULL ] ]**

➤ 属性间关系的定义: **CHECK( <condition> )**

# SQL语言对完整性约束规则的支持

## □ 定义属性级数据约束的例子

### ➤ 例1

```
CREATE TABLE Student (
```

```
    sno NUMBER(5)
```

```
        CONSTRAINT C1 CHECK (sno BETWEEN  
            90000 AND 99999),
```

```
    sname VARCHAR (20)
```

```
        CONSTRAINT C2 NOT NULL,
```

```
    sage NUMBER(3)
```

```
        CONSTRAINT C3 CHECK (sage<29) );
```

# SQL语言对完整性约束规则的支持

## ➤ 例2

**CREATE TABLE EMP (**

**Empno NUMBER (4),**

**Ename VARCHAR (10),**

**PersonId VARCHAR(15),**

**Job VARCHAR (9),**

**Mgr NUMBER (4),**

**Sal NUMBER (7, 2),**

**Deptno NUMBER (2),**

**CONSTRAINT pk PRIMARY KEY( Empno ),**

**CONSTRAINT uni\_name UNIQUE ( PersonId ),**



# SQL语言对完整性约束规则的支持

**CONSTRAINT fk\_dept**

**FOREIGN KEY( Deptno )**

**REFERENCES DEPT( Deptno )**

**ON UPDATE CASCADE**

**ON DELETE RESTRICT,**

# SQL语言对完整性约束规则的支持

**CONSTRAINT fk\_mgr**

**FOREIGN KEY( Mgr )**

**REFERENCES EMP( Empno )**

**ON UPDATE CASCADE**

**ON DELETE SET NULL,**

# SQL语言对完整性约束规则的支持

```
CONSTRAINT chk_1 CHECK (  
    (Job='总经理' AND Sal>15000) OR  
    (Job='部门经理' AND Sal between 5000  
        and 10000) OR  
    (Job<>'总经理' AND Job<>'部门经理'  
        AND  
        Sal between 1000 and 5000)  
);
```

# SQL语言对完整性约束规则的支持

- 在基表的定义命令中，可以定义很复杂的、基于属性或元组的完整性约束条件，并且可以在约束条件中使用到其它的关系

# SQL语言对完整性约束规则的支持

- 例如：不允许‘计算机系’（CS）的学生选修‘PASCAL程序设计’（PAS）课程，则在创建‘学习’关系时可以定义一个完整性约束：

```
CHECK ( NOT (  
    (S# IN (SELECT S#  
            FROM S  
            WHERE sd='CS'))  
    AND  
    (C# IN (SELECT C#  
            FROM C  
            WHERE cn='PAS'))  
));
```

# SQL语言对完整性约束规则的支持

**？问题：**上述完整性约束只对定义它的‘学习’关系起作用，而对‘学生’关系则不起作用。如果一个正在选修PASCAL课程的学生转到计算机系（将‘系别’属性的值修改为‘计算机’），这无疑会破坏‘学习’关系的数据完整性



# SQL语言对完整性约束规则的支持

## □ 全局约束：断言

### ➤ 定义断言

```
CREATE ASSERTION <name> CHECK( <condition> )
```

### ➤ 撤消断言

```
DROP ASSERTION <assertion-name-list>
```

# SQL语言对完整性约束规则的支持

- 例：每门课程的选修人数不少于**20**人

```
CREATE ASSERTION ass_1 CHECK (  
    20 <= ALL ( SELECT COUNT(*)  
                FROM SC  
                GROUP BY C# )  
);
```

# SQL语言对完整性约束规则的支持

- ❑ 例：学生在选修‘数据结构’（DS）课程之前必需先选修过‘PASCAL程序设计’（PAS）课

```
CREATE ASSERTION ass_2 CHECK (  
    NOT EXISTS (  
        SELECT * FROM SC WHERE  
            C# IN ( SELECT C#  
                    FROM C  
                    WHERE cn = 'DS' )  
        AND  
        S# NOT IN ( SELECT SCx.S#  
                    FROM SC SCx, C  
                    WHERE SCx.C# = C.C#  
                      AND C.cn='PAS' )  
    ) );
```

## 4.2 数据库的完整性

### 4.2.1 数据库完整性保护的功能

### 4.2.2 完整性规则的三个内容

### 4.2.3 完整性约束的设置、检查与处理

### 4.2.4 触发器

## 4.2.4 触发器

### □ 触发器

- 在数据库系统中，一个事件的发生会导致另外一些事件的发生，这样的功能被称为触发器
- 触发器的功能：某个事件的发生会导致另外一些事件的执行，以消除前一个事件对数据完整性所起的影响
- 触发器最初是用于数据的完整性保护，但现在已经远远超出了此范围，也被应用于其它的目的，如：
  - 数据的安全性保护
  - 用户的应用逻辑处理
  - 数据库系统的主动功能

## 4.2.4 触发器

### □ 触发器的组成

#### ➤ 触发事件（由用户定义）

- 通常为某个完整性约束条件的否定或某种数据操纵事件
- 如：用户登录，数据的增、删、改等

#### ➤ 结果事件（由用户定义）

- 当触发事件发生时，用以消除触发事件所引起的负面影响的程序
- 通常是一组由用户书写的**SQL**命令

#### ➤ 触发过程

- 当**DBMS**检测到触发事件的发生时，自动调用并执行结果事件的过程



## 4.2.4 触发器

### □ 触发器的定义命令

```
CREATE TRIGGER trigger_name { BEFORE | AFTER }  
    { INSERT | DELETE  
      | UPDATE [ OF colname { , colname ... } ] }  
    ON table_name  
    [ REFERENCING corr_name_def { , ..... } ]  
    [ FOR EACH ROW | FOR EACH STATEMENT ]  
    [ WHEN ( search_condition ) ]  
        { statement  
          | BEGIN ATOMIC statement; { statement; ... } END
```

## 4.2.4 触发器

### ❑ 触发器的定义命令（续）

The corr\_name\_def that defines a correlation name follows:

```
{    OLD [ ROW ] [ AS ] old_row_corr_name  
    | NEW [ ROW ] [ AS ] new_row_corr_name  
  
    | OLD TABLE [ AS ] old_table_corr_name  
    | NEW TABLE [ AS ] new_table_corr_name }
```

## 4.2.4 触发器

### ❑ 触发器的删除命令

```
DROP TRIGGER trigger_name ;
```

## 4.2.4 触发器

### ❑ 例4.12

- **触发事件**：修改或增添教师的工资和职称
- **结果事件**：在插入新的教师元组或教师的职称晋升为教授时，若教授工资低于**1000**元，则将其自动转为**1000**元。

```
CREATE TRIGGER update_sal
    BEFORE INSERT or UPDATE(Sal, Pos) ON Teach
    FOR EACH ROW
    WHEN (:new.Pos = '教授' ) /*某教师的职称为教授
        */
    BEGIN IF      :new.sal < 1000
        THEN :new.sal := 1000;
    END IF;
END;
```