数据库安全性 – 回顾

- 数据库数据安全问题的重要性
- 计算机安全问题 – 安全标准简介
  - TCSEC，TDI ➔ 安全级别
- 如何实现数据库的安全性控制
  - 用户标识与鉴别
  - 存储控制（DAC，MAC）
  - 视图机制
  - 审计
  - 数据加密
- 统计数据库的安全性

数据库完整性

2020/4/23

问题的提出

* 应用环境中存在很多规则，而这些规则反映为属性的取值域、属性之间的取值约束，这是数据库必须遵从的，否则，数据库中的数据会出现不一致或错误。
* 如何保证数据之间的约束关系的？
* 完整性约束的检测对系统性能的影响？
* DBMS能提供什么帮助？

数据库完整性 – 基本概念

安全性：防范非法用户和非法操作，防止对库内数据的非法存取。

* 数据库完整性 - integrity
  * 数据的正确性和相容性
  * 防止不符合语义、不正确的数据进入数据库。

例：期末成绩必须是整数，取值范围为[0，100]；
学生的性别只能是男或女；学生的学号一定是唯一的；
学生所在的系必须是学校开设的系；等等…

* 完整性控制是DBMS核心功能之一
  * 减轻应用程序员的负担
  * 为所有用户和所有应用提供一致的数据库完整性

数据库完整性 – 基本概念

- DBMS如何维护数据库的完整性
  - 提供定义完整性约束条件的机制
    - DDL语句描述完整性（存入数据字典）
  - 提供完整性检查的方法
    - 检查操作执行后数据是否违背完整性约束条件
    - 检查时机：立即执行检查、延迟执行检查（如，事务提交时）
      - 关于事务：DB内不可分割的工作单位。如，借贷平衡。
  - 违约处理
    - 对违背完整性约束条件的操作，采取
      - 拒绝 NO ACTION
      - 级联执行其他操作 CASCADE
      - ……

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

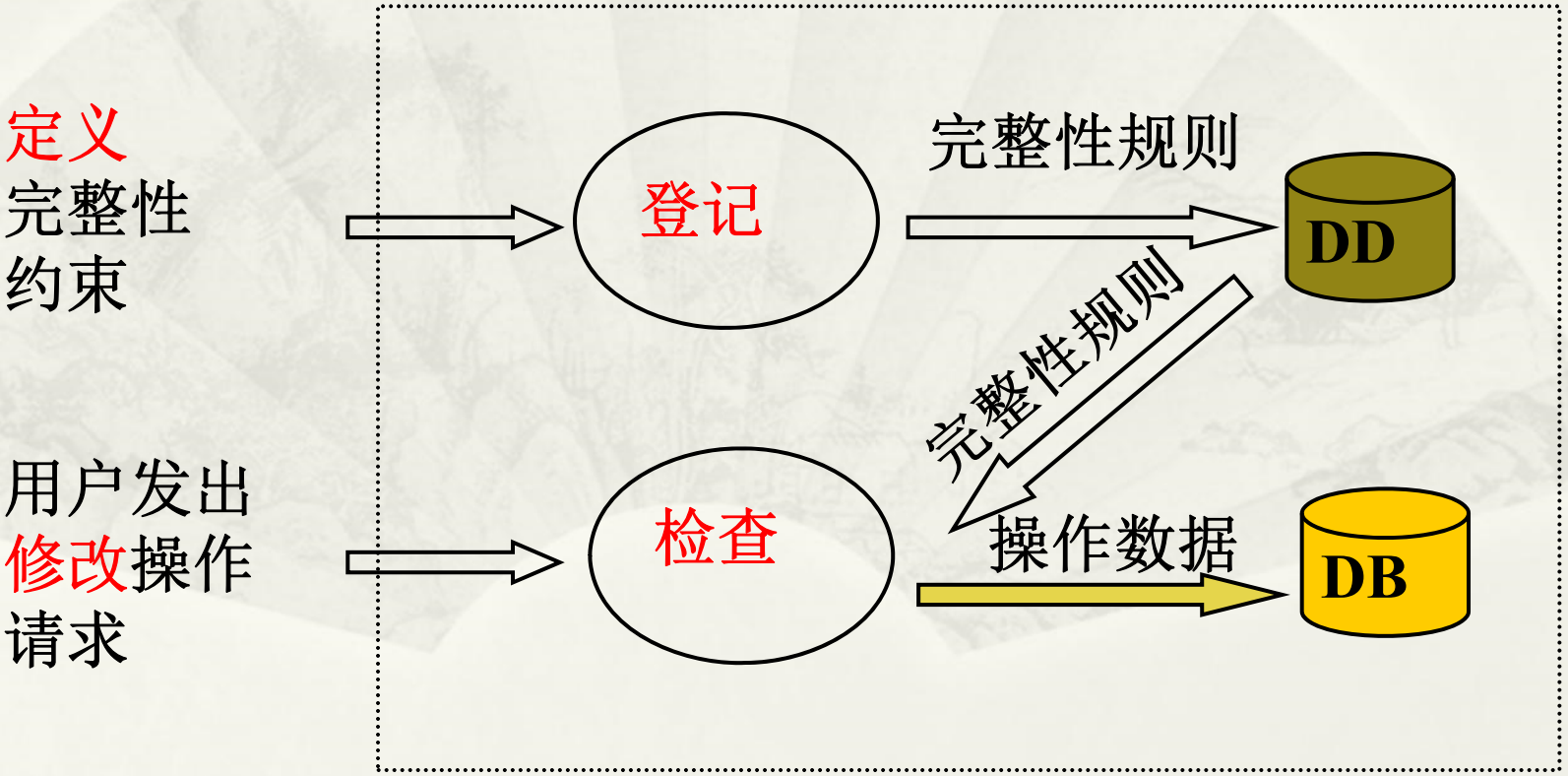
五. 用户自定义完整性

- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

1 完整性控制的实现原理
- 完整性控制子系统

* 定义、检查、采取行动、……



主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

五. 用户自定义完整性

- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

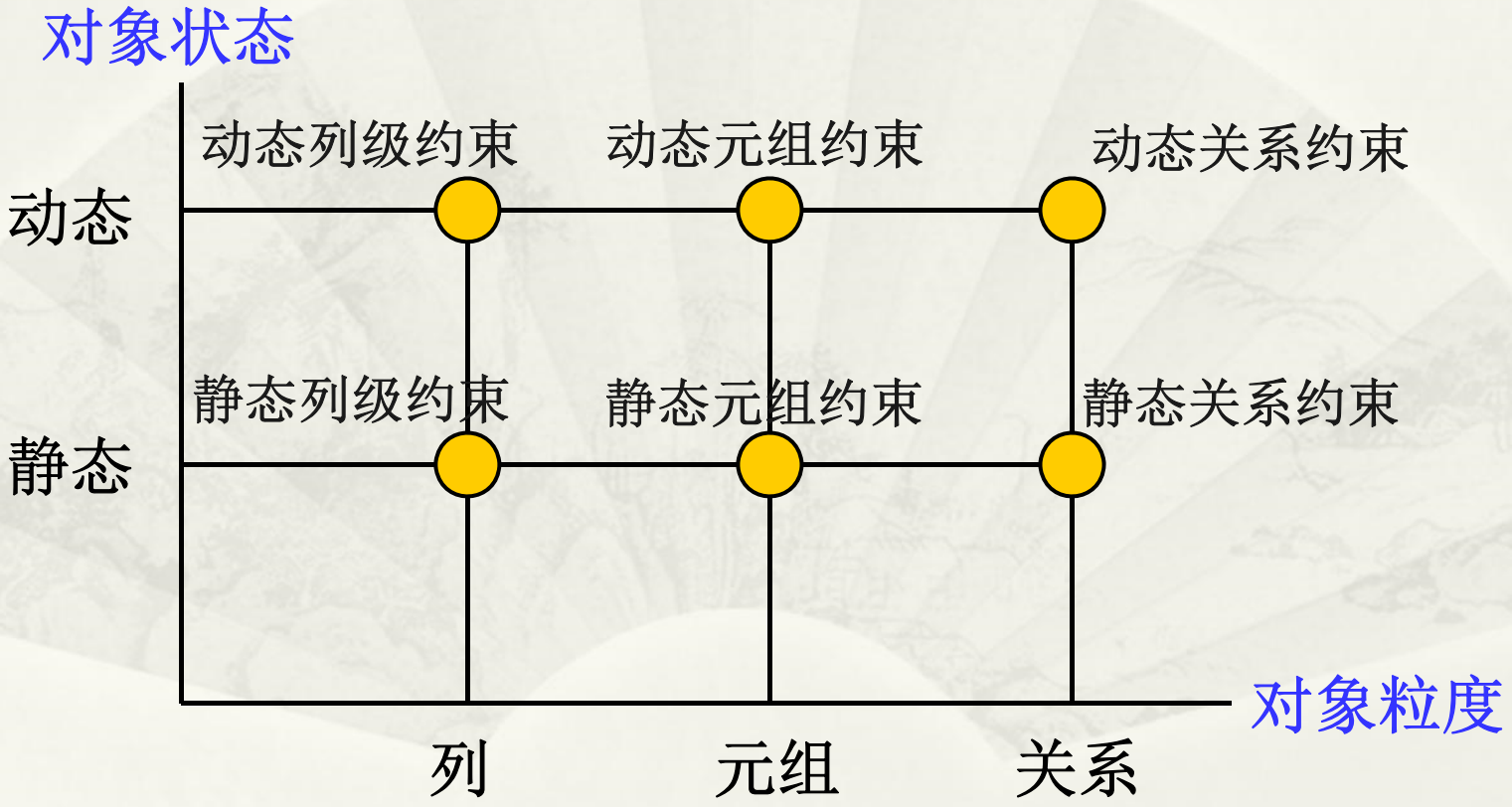
2.1 完整性约束条件分类 – 对象粒度

- 完整性约束条件的对象粒度
  - 关系约束
    - 若干元组间、关系集合上以及关系之间的联系的约束。
  - 元组约束
    - 元组中各个字段间的联系的约束。
  - 列/属性约束
    - 列的类型、取值范围、精度、排序等约束。

2.1 完整性约束条件分类 – 对象状态

* 静态约束
  * 指数据库每一确定状态时的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
  * 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。

2.2 完整性约束条件分类 – 小结



2.3 完整性约束 – 静态列级

- 最常见、最简单、最容易实现的一类完整性约束。在列定义时给出。
  - 类型：Sno CHAR(6)
  - 格式：入学日期 YY.MM.DD
  - 值域：取值范围或取值集合
    - （0~100）、[男、女]
  - 空值：是否允许为空
  - 其他约束：关于列的排序说明、组合列等

2.3 完整性约束 – 静态元组级

- 规定了元组的各个列之间的约束关系
  - 订货关系中，“发货量”<=“订货量”
  - 教师关系中，“教授”的“工资”>=4000
  - 学生关系中，“性别”为男，“名字”不能以“Ms.”打头。
    - CHECK (Sgender='女' OR Sname NOT LIKE 'Ms.%')
  - 函数依赖约束
    - 学生课程教师关系SJT中，（S，J）-> T

2.3 完整性约束 – 静态关系级

- 关系的若干关系之间、关系中或者若干元组间存在的各种联系或约束
  - 参照完整性约束
  - 实体完整性约束
  - 元组之间，如：统计约束
    - 定义某个字段值一个关系多个元组的统计值之间的约束关系
    - 如，部门经理的工资 [2倍职工平均工资，5倍…]

2.3 完整性约束 – 动态列级

- 修改列定义或列值时应满足的约束条件
  - 修改列定义时
    - 将原来允许为空的，改为不允许为空
      - 若目前已存在空值，则拒绝修改
  - 修改列值，新旧值之间需要满足的约束条件
    - 如，职工调整后的工资 >= 之前的工资

2.3 完整性约束 – 动态元组、关系级

- 修改元组时，各个字段之间要满足的约束关系
  - 职工“工资”调整后，应大于等于“原来工资”+“工龄”*1000
- 动态关系约束
  - 关系变化前后状态所需满足的限制条件
    - 事务一致性、原子性等约束条件

2.4 完整性约束条件分类 – 回顾

状态 粒度	列级	元组级	关系级
静态	列定义 ◆类型 ◆格式 ◆值域 ◆空值	元组值应满足的条件	实体完整性约束 参照完整性约束 函数依赖约束 统计约束
动态	改变列定义或列值	元组新旧值之间应满足的约束条件	关系新旧状态间应满足的约束条件

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

五. 用户自定义完整性

- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

3 主码和实体完整性 - 概念回顾

- 候选码：该属性组值能唯一标识关系中的一个元组。 学号、姓名、性别、年龄、手机号、邮箱、...
- 主属性：候选码的诸属性
- 非主属性：不包含在任何候选码中的属性
- 主码/主键：从候选码中挑一个作为该关系的PRIMARY KEY
- 实体完整性：基本关系的主属性均不能为空

3.1 主码和实体完整性 – 定义

* **CREATE TABLE中的PRIMARY KEY**

```
CREATE TABLE Student
        ( Sno CHAR(6) PRIMARY KEY,
        Sname CHAR(20) NOT NULL,
        Sgender CHAR(2),
        Sage SMALLINT,
        Sdept CHAR(20) );
```

列级约束

```
CREATE TABLE Student
        ( Sno CHAR(6),
        Sname CHAR(20) NOT NULL,
        Sgender CHAR(2),
        Sage SMALLINT,
        Sdept CHAR(20) ,
        PRIMARY KEY (Sno));
```

表级约束

3.2 主码和实体完整性 – 检查与违约处理

- 可能破坏完整性的操作 - 数据修改
  - 对基本表插入一条记录或对主码列进行修改
  - 删除不会破坏实体完整性
- DBMS按照实体完整性规则
  - 检查内容：
    - 主码值是否唯一，若不唯一，拒绝
    - 主码各属性值是否为空，若一个为空，拒绝
  - 检查方法：全表扫描
    - DBMS在主码上自动建立索引以提高速度
      - B+树
  - 实验课中问题：在已有数据的关系上建立主码？

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. **外码和参照完整性**

五. 用户自定义完整性

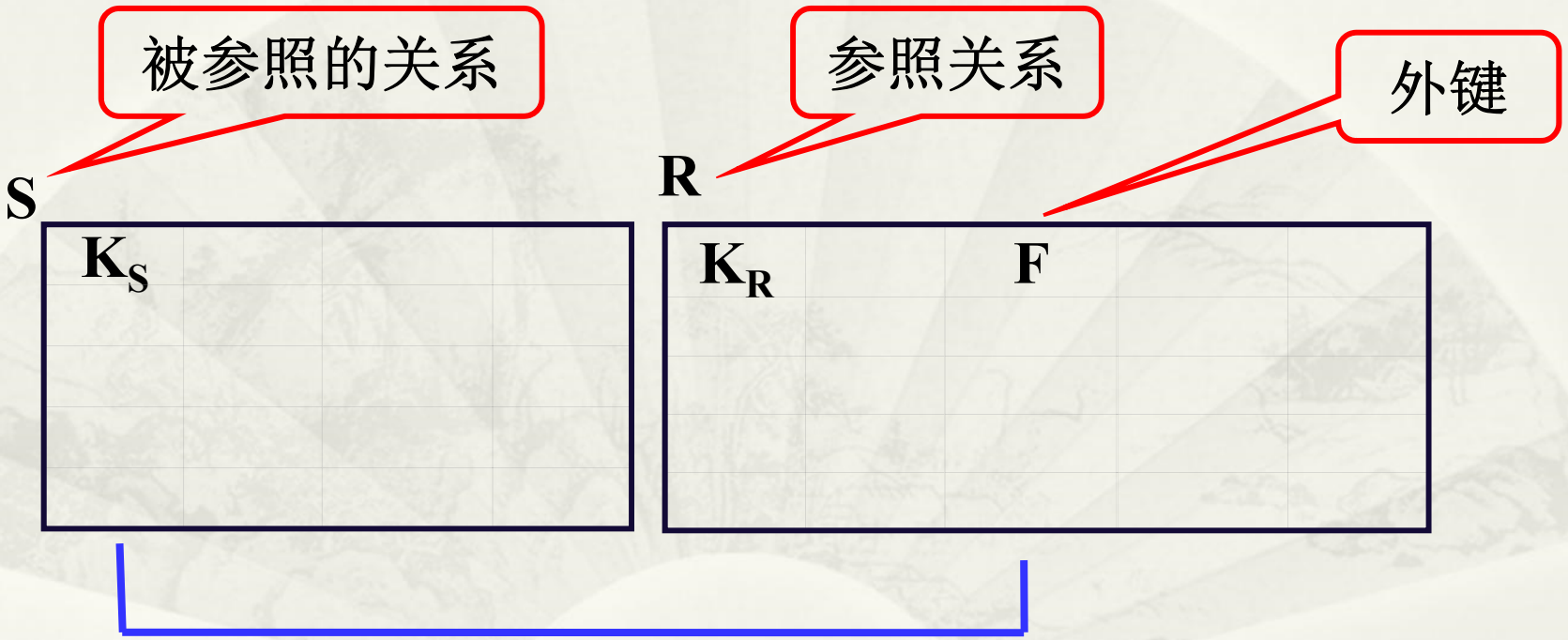
- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

4 外码和参照完整性 – 概念回顾

* 外码
  * 设F是基本关系R的一个/组属性，但不是关系R的码。K_S是基本关系S的主码。若F与K_S相对应，则称F是R的FOREIGN KEY。
* 参照完整性规则
  * 若属性（属性组）F是基本关系R的外码，它与基本关系S的主码K_S相对应，则对于R中每个元组在F上的值必须为
    * 或者取空值
    * 或者等于S中某个元组的主码值

4 外码和参照完整性 - 图示



4 外码和参照完整性 – 例

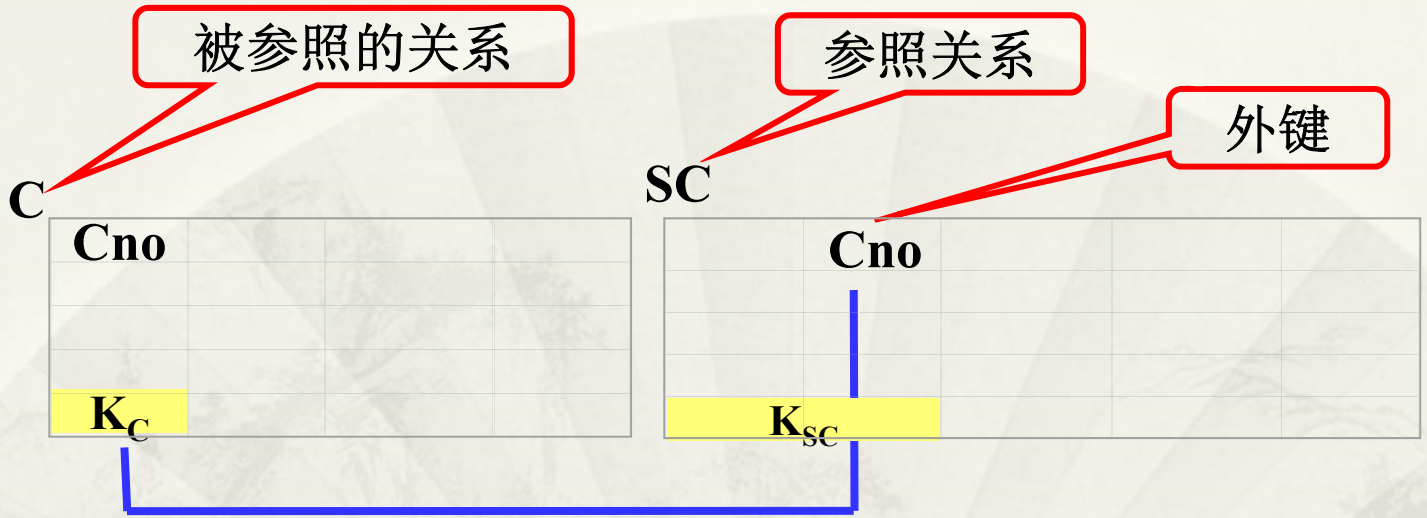
* 两个关系间
  学生（学号，姓名，性别，专业号，年龄）
  专业（专业号，专业名）
* 两个以上的关系间
  学生（学号，姓名，性别，专业号，年龄）
  课程（课程号，课程名，学分）
  选修（学号，课程号，成绩）
* 同一关系
  学生2（学号，姓名，性别，专业号，年龄，班长）

4.1 外码和参照完整性 – 定义

* **CREATE TEBLE中的FOREIGN KEY**

```
CREATE TABLE SC
        ( Sno CHAR(6) NOT NULL,
        Cno CHAR(4) NOT NULL,
        Grade SMALLINT,
        PRIMARY KEY (Sno, Cno),
        FOREIGN KEY  (Sno) REFERENCES Student(Sno),
        FOREIGN KEY  (Cno) REFERENCES Course(Cno)
        );
```

4.2 外码和参照完整性 – 哪些操作有可能破坏参照完整性



	C (Cno)	SC (Cno)
插入元组		X
删除元组	X	
修改元组	X	X

4.2 外码和参照完整性 – 检查与违约处理

- 当发生不一致时，采取措施
  - 拒绝 – NO ACTION（默认）
  - 级连操作 – CASCADE
    - 被参照表S元组的删除和修改，导致参照表SC的不一致，则依次删除或修改所有不一致的元组。
  - 设置为空值
    - 学生（学号，姓名，性别，年龄，专业号）
    - 专业（专业号，专业名）
    - 外码能否取空值？
      - 在定义时需说明“外码列能否允许空值”

4.2 外码和参照完整性 – 违约处理

* 系统默认策略：拒绝执行。其他策略在创建表时显式说明

```
CREATE TABLE SC
 ( Sno CHAR(6) NOT NULL,
   Cno CHAR(4) NOT NULL,
   Grade SMALLINT,
   PRIMARY KEY ( Sno, Cno ),
   FOREIGN KEY (Sno) REFERENCES Student(Sno)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   FOREIGN KEY (Cno) REFERENCES Course(Cno)
      ON DELETE NO ACTION    /*不允许删除学生已选修的课*/
      ON UPDATE CASCADE
);
```

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

五. **用户自定义完整性**

- ß **非过程性约束的实现 – 定义中约束**
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

5.1.1 非过程性约束 – CREATE TABLE

* 基本CREATE TABLE语句的语法：

```
CREATE  TABLE [schema.]  tablename
( {columnname  datatype  [DEFAULT  {default_constant | null}]
     [col_constr{col_constr …}] | table_constr}
  {,{columnname datatype [DEFAULT {default_constant| null}]
     [col_constr{col_constr …}] | table_constr}
…});
```

5.1.1 CREATE TABLE语句中的列约束

* 约束单个列的 col_constr 形式如下

```
{NOT NULL |
 [CONSTRAINT constrainname]
   UNIQUE
  | PRIMARY KEY （< columnname >[，< columnname >] …）
  | CHECK ( search_cond )
  | FOREIGN KEY （< columnname >[，< columnname >] …）
                  REFERENCES       <tablename>
     [ON DELETE {CASCADE | SET DEFAULT | SET NULL
     | NO ACTION} ]
     [ON UPDATE {CASCADE | SET DEFAULT | SET NULL
     |NO ACTION} ]
     }
```

5.1.1 CREATE TABLE - 列约束 – 例

* 创建表:学生（学号，姓名，性别，专业，年龄）
  其中,性别只能取 ‘男’和 ‘女’这两个值。

```
CREATE TABLE Student
( Sno   CHAR(6),
  Sname  CHAR(20) UNIQUE,
  Sgender CHAR(2) CONSTRAINT S_Gender
                  CHECK(Sgender IN (‘男’, ‘女’)),
  Sage  SMALLINT,
  Sdept  CHAR(20),
  PRIMARY KEY（Sno）
 );
```

5.1.2 完整性约束命名子句

- **CONSTRAINT <完整性约束条件名> <PRIMARY KEY | NOT NULL | UNIQUE | FOREIGN KEY | CHECK >**

- **优点：可以灵活地增加、删除一个完整性约束条件**

5.1.2 完整性约束命名子句 – 命名子句

* 建立学生登记表Student，要求学号在90000~99999之间，姓名不能为空值，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 SMALLINT
  CONSTRAINT C3 CHECK (Sage < 30),
 Sgender CHAR(2)
  CONSTRAINT C4 CHECK (Sgender IN (‘男’, ‘女’)),
 CONSTRAINT StudentKey PRIMARY KEY(Sno) );
```

5.1.2 完整性约束命名子句 – 命名子句

* 建立教师表TEACHER，要求每个教师的应发工资不低于3000元。

```
CREATE TABLE TEACHER
(Eno NUMERIC(4) PRIMARY KEY,              /*列级定义主码 */
  Ename CHAR(10),
  Job CHAR(8),
  Sal NUMERIC(7, 2),
  Deduct NUMERIC(7, 2),
  Deptno NUMERIC(2),
  CONSTRAINT EMPFKey FOREIGN KEY (Deptno)
                          REFERENCES DEPT(Deptno),
  CONSTRAINT C1 CHECK (Sal+Deduct >= 3000)
);
```

5.1.2 完整性约束命名子句 – 修改

* 可用ALTER TABLE来修改表中的完整性限制

  • 去掉Stuedent表中对年龄的限制

```
ALTER TABLE Student
        DROP CONSTRAINT C3;
```

  • 将Student表中年龄的限制改为小于40岁。

```
ALTER TABLE Student
        DROP CONSTRAINT C3;
ALTER TABLE Student
        ADD CONSTRAINT C3 CHECK (Sage < 40);
```

5.1.3 CREATE TABLE语句中的元组约束

- 元组约束的CHECK (search_cond)实例
  - 创建表:学生（学号，姓名，性别，专业，年龄），其中，如果性别为男，姓名不能以Ms.开头。

```
CREATE TABLE Student
（ Sno        CHAR（6），
  Sname       CHAR（20） UNIQUE，
  Sgender     CHAR（2） CONSTRAINT S_SE
                    CHECK(Sgender IN (‘男’, ‘女’))，
  Sage        SMALLINT，
  Sdept       CHAR（20），
 PRIMARY KEY（Sno),
  CHECK (Sgender = ‘女’ OR Sname NOT LIKE
    ‘Ms.%’)
）；
```

5.1.4 非过程性约束 – ALTER TABLE

* ALTER TABLE语句允许DBA改变原先在CREATE TABLE语句中定义的表的结构，加入或改变列、加入或删除各种约束等。

* 给约束命名的用处：如果表约束被命了名，表所有者可以用这个名字DROP该约束。

5.1.5 非过程性约束的实现

- 用CREATE TABLE及ALTER TABLE实现非过程性约束
  - NOT NULL
  - UNIQUE
  - CHECK
  - PRIMARY KEY
  - FOREIGN KEY

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

五. 用户自定义完整性

- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

5.2.1 断言的定义

- 断言：
  - 通过声明性断言来指定更具一般性的约束
  - 涉及多个表或聚集操作
  - 定义
    - CREATE ASSERTION <断言名> <CHECK 子句>
    - CHECK 子句中的约束条件类似WHERE子句
  - 检查
    - 使断言为假的操作将被拒绝执行
  - 删除
    - DROP ASSERTION <断言名>
  - 注意：检测和维护断言的开销较高！

5.2.2 断言- 示例

* 限制数据库课程最多60名学生选修

```
CREATE ASSERTION ASSE_SC_DB_NUM
    CHECK ( 60 >= ( SELECT count(*)
                FROM Course, SC
                WHERE SC.CNO=Course.CNO AND
                        Course.CNAME='数据库' ));
```

  * 随着一条条记录插入，当选修该课程的人数超过60时，CHECK返回“假”，操作遂被拒绝。

主要内容

一. 完整性控制的实现原理

二. 完整性约束条件分类

三. 主码和实体完整性

四. 外码和参照完整性

五. **用户自定义完整性**

- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß **过程性约束与触发器**

六. **小结**

5.3 过程性约束与触发器 - 概念

* 触发器（Trigger）：用户定义在关系表上的一类由事件驱动的特殊过程。一旦定义，任何关联该触发器的事件发生时，均由服务器(DBMS)自动激活相应的触发器。
* 特点
  * 类似约束，但比约束更加灵活
  * 可实现更为复杂的检查和操作
  * 具有更精细、更强大的数据控制能力

5.3.1 过程性约束与触发器 - 定义

* 用CREATE TRIGGER命令建立触发器

CREATE TRIGGER <触发器名>

{ BEFORE | AFTER } <触发事件> ON <表名>

FOR EACH { ROW | STATEMENT }

[WHEN <触发条件>]

<触发动作体>

INSERT, UPDATE, DELETE 或 组合

条件为真,方执行动作体

匿名PL/SQL过程块 或
已创建的存储过程

5.3.1 过程性约束与触发器 – 定义 – 例

* 定义一个BEFORE行级触发器，为教师表TEACHER定义完整性规则“教授的工资不能低于4000元，若低于，则自动改为4000元”

```
CREATE TRIGGER Insert_Or_Update_Sal
   BEFORE INSERT OR UPDATE ON TEACHER
   FOR EACH ROW
   AS BEGIN
      IF (new.Job=‘教授’) AND (new.Sal < 4000 ) THEN
            new.Sal := 4000;
      END IF;
   END;
```

5.3.1 过程性约束与触发器 – 定义 – 例

* 定义AFTER行级触发器，当工资变化时自动在“工资日志表”中增加一条记录。

```
CREATE TABLE Sal_log
 ( Eno NUMERIC(4) REFERENCES TEACHER(Eno),
   Sal NUMERIC(7, 2),
   Username CHAR(10),
   Date TIMESTAMP ) ;

CREATE TRIGGER Insert_Sal    /* 建立触发事件INSERT的触发器 */
  AFTER INSERT ON TEACHER
  FOR EACH ROW
    AS BEGIN
       INSERT INTO Sal_log VALUES ( new.Eno, new.Sal, CURRENT_USER,
                                    CURRENT_TIMESTAMP);
    END;
```

5.3.1 过程性约束与触发器 – 定义 – 例

```
/* 建立触发事件UPDATE的触发器 */

CREATE TRIGGER Update_Sal
   AFTER UPDATE ON TEACHER
   FOR EACH ROW
      AS BEGIN
         IF (new.Sal<>old.Sal) THEN
               INSERT INTO Sal_log VALUES ( new.Eno, new.Sal,
                                    CURRENT_USER, CURRENT_TIMESTAMP);
         END IF;
      END;
```

5.3.2 过程性约束与触发器 – 激活

- 由规定事件激活触发器，并由DBMS自动执行。
- 表上所定义的多个触发器的执行顺序
  1. BEFORE触发器
  2. 激活该触发器的SQL语句
  3. AFTER触发器
  - 注：多个BEFORE/AFTER，按照创建的先后顺序执行（或字母顺序）

5.3.2 过程性约束与触发器 – 激活 – 例

- 教师表上的触发器
  - Insert_Or_Update_Sal (BEFORE)
  - Insert_Sal (AFTER)
  - Update_Sal (AFTER)

- 执行语句

  UPDATE TEACHER SET Sal=800 WHERE Ename = ‘陈平’ ;

5.3.3 过程性约束与触发器 – 删除

* 删除的SQL语法如下

DROP TRIGGER <触发器名> ON <表名>;

* 删除教师表上的触发器Insert_Sal。

DROP TRIGGER Insert_Sal ON TEACHER;

5.3.4 非过程性约束与过程性约束

- CREATE TABLE语句中的非过程性约束的种类是有限的（CHECK约束，UNIQUE约束，PRIMARY KEY约束，FOREIGN KEY约束），因而能力是有限的，但容易理解；
- 非过程性约束直接为系统所知；
- 非过程性约束难以给出约束不满足时的相应动作；
- 过程性约束的作用更大
  - 给出约束不满足时的相应动作
  - 保证事务的一致性(可以使一个更新操作在另一个执行之后自动发生)

主要内容

一. 完整性控制的实现原理
二. 完整性约束条件分类
三. 主码和实体完整性
四. 外码和参照完整性
五. 用户自定义完整性
- ß 非过程性约束的实现 – 定义中约束
- ß 断言- Assertion子句
- ß 过程性约束与触发器

六. 小结

小结

1. DBMS的完整性控制的基本原理
2. 完整性机制的实施会极大地影响系统性能
3. 实现完整性约束的方法有非过程性的方法和过程性的方法，利用CREATE TABLE语句的列约束和表约束实现有限的非过程性地约束；利用触发器过程性地实现动态的约束。
4. 实体完整性和参照完整性的内容和使用
5. 关于“断言”

作业

* P.173 2、3、6

作业 P.164 2、3、6

2、数据库的完整性概念与数据库的安全性概念有什么区别和联系？

3、什么是数据库的完整性约束条件？

6、假定有下面两个关系模式

职工（职工号，姓名，年龄，职务，工资，部门号），其中职工号为主码；

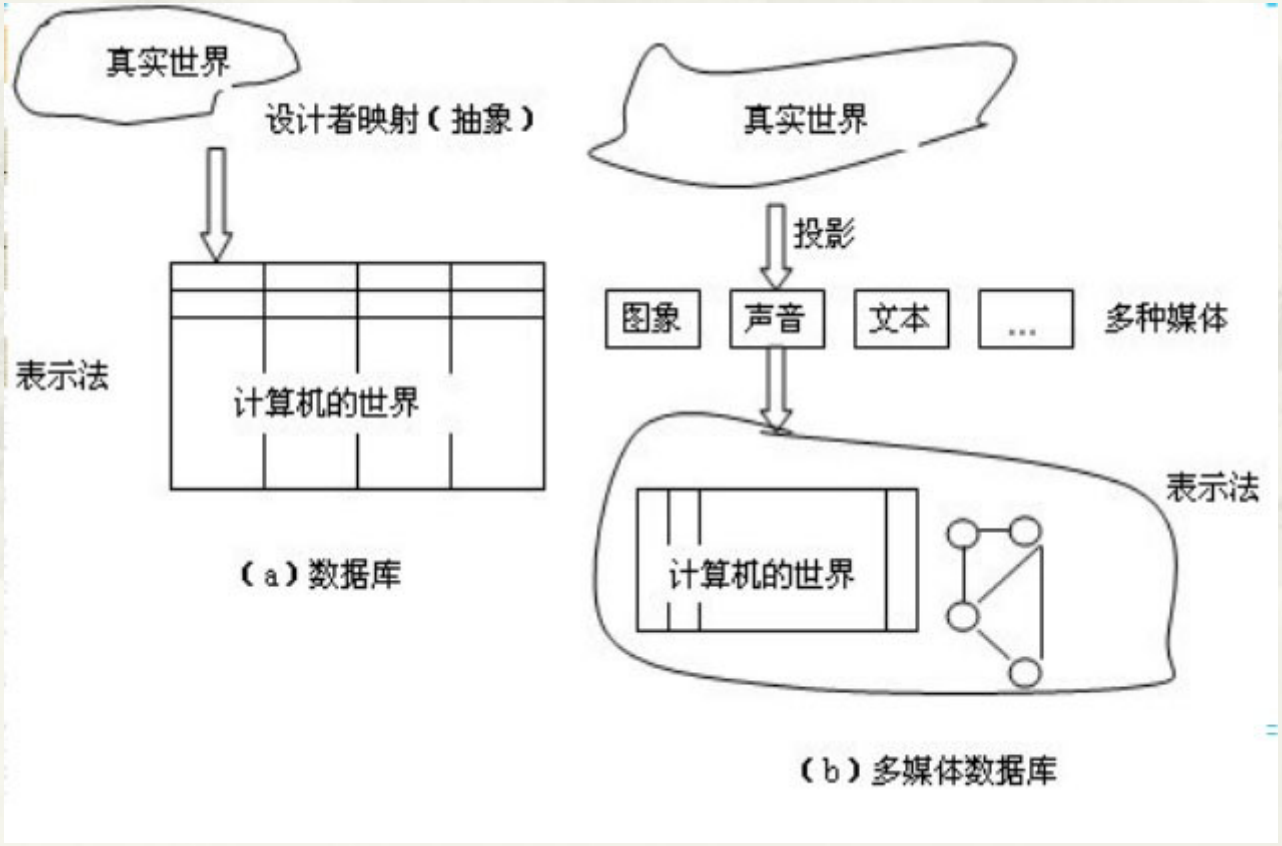
部门（部门号，名称，经理名，电话），其中部门号为主码；

用SQL语言定义这两个关系模式，并完成如下完整性约束条件的定义：

（1）定义主码；（2）定义参照完整性；（3）职工年龄不得超过60岁。

关于多媒体数据库

* MMDB：最早由Tsichritzis和Christodoulakis于1983年在第九届VLDB会议上提出。



多媒体数据库所面临的问题

- 数据的组织和存储
  - 数据量大、数据本身冗余大
- 媒体种类的增加
- 查询问题 – 信息检索
- 用户接口
- 信息的分布对体系结构的影响
- 长事务增多
- 服务质量
- 版本控制、等

多媒体数据类型及编码

- 多媒体数据类型
  - 文本、图像、图形、声音/音频、视频、动画、3D、VR/AR/MR、元数据、.....
- 压缩编码技术
  - 必要性和有效性
  - 压缩标准
  - 对应的文件后缀名
    - 如图像对应的：jpg，bmp，gif，png，tiff，.....

多媒体数据库 - 数据模型

- 扩充的关系模型
  - NF^2数据模型 – Non First Normal Form
  - 表中可嵌套表；BLOB
- 面向对象数据模型
  - 引入面向对象概念，描述复杂对象
- 对象-关系数据模型
  - 用户定义新的数据类型和操作，并结合关系数据库提供的存储管理功能
- 面向具体应用 – 如，超文本数据模型
  - 定义文本的概念结构，实现高效管理

多媒体数据库 - 体系结构

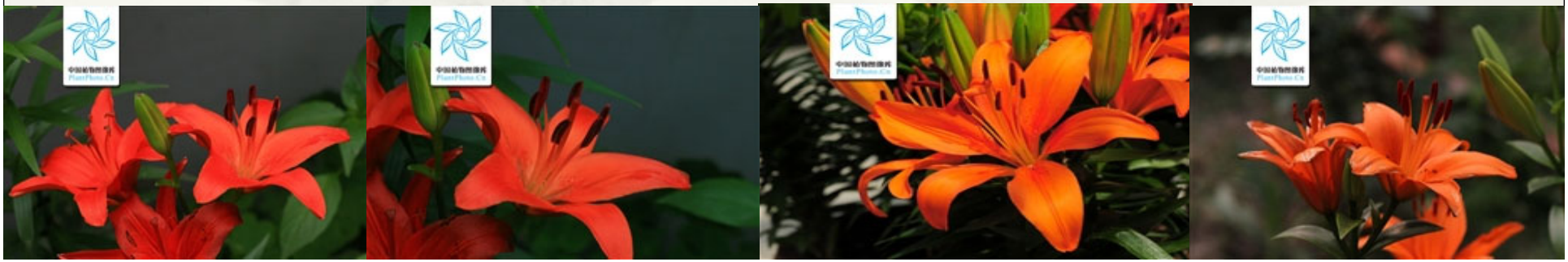
- 联邦型
  - 多个独立媒体数据库整合在一起，各自具有独立的DBMS，
- 集中统一型
  - 多种媒体被统一存储在一个库内，由MMDBMS统一管理。
- 客户/服务器型
  - 多用于网络环境下。多种媒体数据库相对独立，通过专用服务器和一个多媒体管理服务器相连。
- 超媒体型
  - 多种媒体数据库分散存储在有网络连接的不同存储空间。重点在于对数据时空索引的组织。

多媒体数据库 – 查询语言SQL/MM

- 1992年提出标准，命名SQL/MM。
- SQL/MM的基本结构
  - 第一部分，框架 FrameWork
  - 第二部分，全文 FullText
  - 第三部分，空间 Spatial
  - 第五部分，静态图像 StillImage
  - 第六部分，数据挖掘 DataMining
- 部分数据库系统提供支持

多媒体数据库 – SQL/MM - StillImage

- **Oracle10g的SI_StillImage类型数据**
  - ***myimage SI_StillImage;***
  - ***myAvgColor:=si_findavgclr(myimage);--颜色平均***
  - ***myColorHist:=si_findclrhstgr(myimage);--颜色直方图***
  - ***myPosColor:=si_findpstnlclr(myimage);--颜色位置***
  - ***myTexture:=si_findtexture(myimage);--纹理***
  - **求解**
    - ***myFeatureList := new SI_FeatureList(myAvgColor, 0.25, myColorHist, 0.35, myPosColor, 0.10, myTexture, 0.5);***



多媒体数据库 –多媒体数据库设计

- 体系结构 – 灵活和可扩展
- 界面设计 – 可用性
- 信息空间表示 – 逻辑信息空间和物理信息空间
- 可视推理 – 基于视觉表现线索的推测和制作推理的过程
- 相关反馈技术 – 与信息系统的人机交互
- 高维特征索引技术研究
- ……

研究问题 - 用户查询界面

* 用户对内容的感知
* 用户与系统的交互方式
* 查询界面

* 现代多媒体信息系统的一个重要特征就是信息获取过程的可交互性，人在系统中是主动的。

研究问题 - 高维索引技术

- 高维特征向量的多维索引结构
  - 如 k-d树和R-树，及改进的索引树结构
  - 仍需有效的高维索引方法，以支持多特征、异构特征、权重、主键特征方面的查询要求。
- 相似度匹配
  - 聚类和神经网络方法适于解决这类问题。
- 时间序列媒体
  - 考虑它们的时间索引结构

研究问题 - 高层概念和底层特征的关联

* 高层：人们倾向于使用概念表达事物。
* 底层：目前特征提取和检索都是基于底层特征方面的研究。
* 特定应用领域可实现：例如人脸识别和指纹识别，做到了一定程度的底层的特征与高层语义概念的关联
* 对于一般性的特征，建立起这种关联是非常困难的。

研究问题 - 多媒体信息安全

* 数字媒体内容的安全问题是制约多媒体内容市场的瓶颈问题，包括安全传递、访问控制和版权保护
* 数字版权管理技术：数字水印