上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

数 据 库 技 术

Database System Technology

郭 捷

(guojie@sjtu.edu.cn)

饮 水 思 源 • 爱 国 荣 校

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目 录

什么是数据库的完整性?

- 数据库的完整性（integrity）是指数据的正确性（correctness）和相容性（compatability）；
- 数据的正确性：是指数据符合现实世界语义、反映当前实际状况；
- 数据的相容性：是指数据库同一对象在不同关系表中的数据是符合逻辑的；

数据库完整性举例

- 例：　学生的年龄必须是整数，取值范围为14--29；

  学生的性别只能是男或女；

  学生的学号一定是唯一的；

  学生所在的院系必须是学校已成立的院系；

数据库完整性和数据库安全性的区别

- 数据库完整性
  - 防止数据库中存在不符合语义的数据和不正确的数据；
  - 检查和控制的防范对象：不合语义的、不正确的数据，防止它们进入数据库；
- 数据库安全性
  - 保护数据库，防止恶意破坏和非法存取；
  - 检查和控制的防范对象：非法用户和非法操作，防止他们对数据库数据的非法存取；

DBMS的完整性控制机制

- 定义完整性约束条件的机制
- 提供完整性检查的方法
- 进行违约处理

一、完整性约束条件的定义

- 完整性约束条件，又称为完整性规则，是数据库中数据必须满足的语义约束条件；
- 表达了给定数据模型中数据及其联系所具有的制约和依存规则，用以限定符合数据模型的数据库状态及状态变化，保证数据的正确、有效和相容；
- DBMS应提供定义数据库完整性约束条件的机制，包括实体完整性、参照完整性、用户定义完整性，并把它们作为模式的一部分，存入数据字典中；

二、完整性检查的方法

- 完整性检查，是指在DBMS中，检查数据是否满足完整性约束条件的机制；
- 一般在INSERT、UPDATE、DELETE语句执行后开始检查，也可在事务提交时检查；
- 检查用户发出的操作请求，是否违背了完整性约束条件；

三、违约反应

- DBMS如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性；
- 例如拒绝（NO ACTION）执行该操作，或级联（CASCADE）执行其他操作，进行违约处理以保证数据的完整性；

完整性约束条件

- 静态约束
  - 对静态对象的约束是反映数据库状态合理性的约束；
- 动态约束
  - 对动态对象的约束是反映数据库状态变迁的约束；

完整性约束条件小结

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	元组的属性之间应满足的条件	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

DBMS的完整性控制机制

完整性规则五元组表示：

(D, O, A, C, P)

- D（Data） 约束作用的数据对象；
- O（Operation） 触发完整性检查的数据库操作

  当用户发出什么操作请求时需要检查该完整性规则，是立即检查还是延迟检查；
- A（Assertion） 数据对象必须满足的断言或语义约束，是规则的主体；
- 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。

DBMS的完整性控制机制

例：在“教授工资不得低于1000元”的约束中

- D（Data）　　约束作用的对象为工资Sal属性；
- O（Operation）　插入或修改职工元组时；
- A（Assertion）　Sal不能小于1000；
- C（Condition）　职称=‘教授’（A仅作用于职称=‘教授’的记录）；
- P（Procedure）　拒绝执行该操作

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目 录

数据库完整性

- 完整性定义和检查控制由关系数据库管理系统实现，不必由应用程序来完成；
- 关系数据库管理系统，使得完整性控制成为其核心支持功能，从而为所有用户和应用提供一致的数据库完整性；

实体完整性的定义

- CREATE TABLE语句中提供了PRIMARY KEY子句，供用户在建表时指定关系的主码列。
- 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

实体完整性的定义

例1：在学生选课数据库中，要定义Student表的Sno属性为主码

```
CREATE TABLE Student
    (Sno   NUMBER(8),
     Sname VARCHAR(20) NOT NULL,
     Sage  NUMBER(20),
     PRIMARY KEY (Sno));                    /* 在表级定义主码*/
```

或：

```
CREATE TABLE Student
    (Sno   NUMBER(8)  PRIMARY KEY ,         /* 在列级定义主码*/
     Sname VARCHAR(20) NOT NULL,
     Sage  NUMBER(20));
```

实体完整性的定义

例2：要在SC表中定义(Sno, Cno)为主码

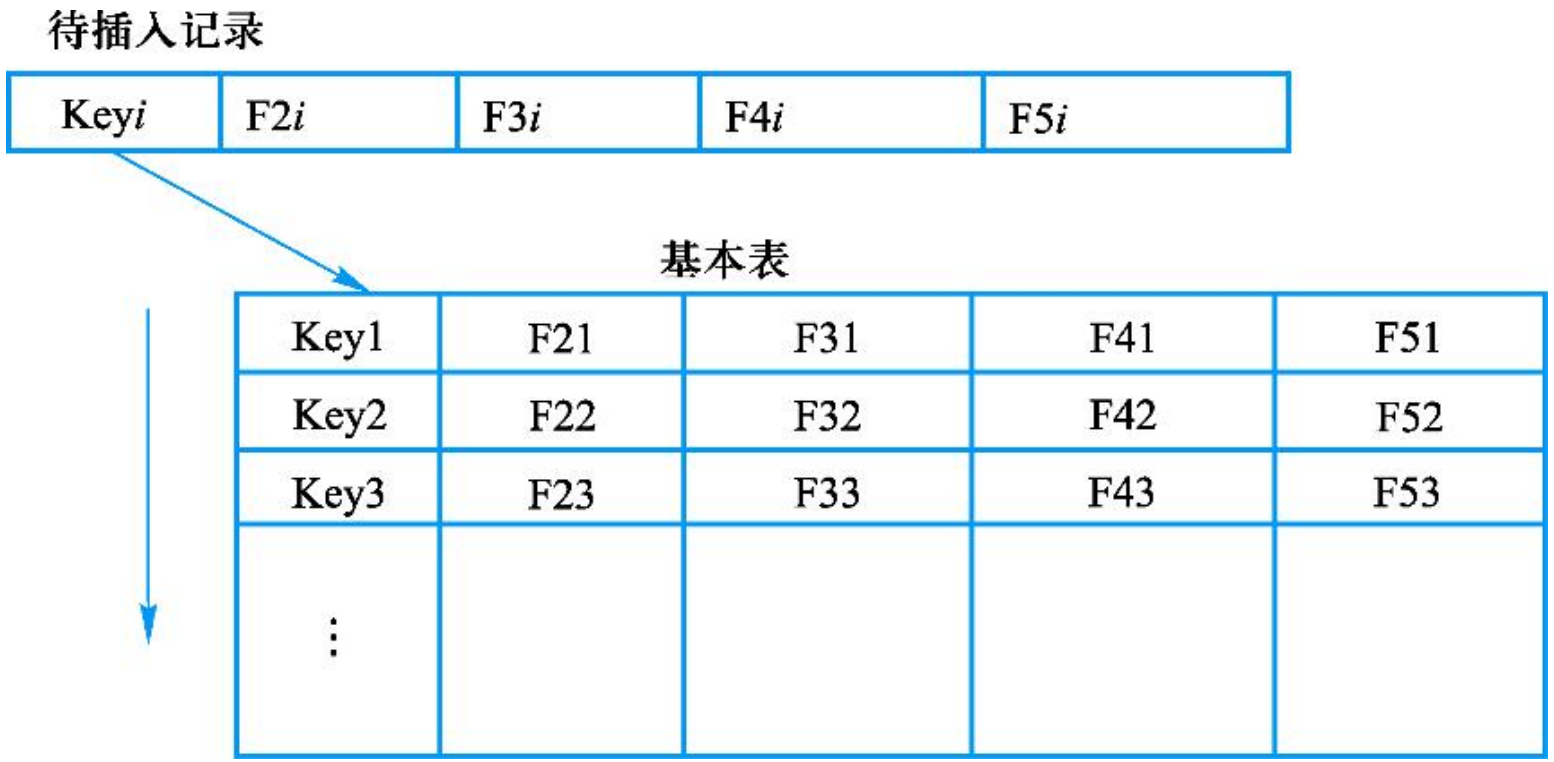
```
CREATE TABLE SC
      (Sno   NUMBER(8)  NOT NULL,
       Cno   NUMBER(2)  NOT NULL,
       Grade NUMBER(2),
       PRIMARY KEY (Sno, Cno)          /* 只能在表级定义主码*/
FOREIGN KEY (Sno) REFERENCES Student(Sno)   /* 在表级定义参照完整性 */
FOREIGN KEY (Cno) REFERENCES Course(Cno)   /* 在表级定义参照完整性 */
      );
```

实体完整性检查和违约处理

- 用户程序对主码列进行更新操作时，系统自动进行实体完整性检查；
- 实体完整性检查：
  - 检查主码值是否唯一，否则拒绝插入或修改；
  - 检查主码的各个属性是否为空，否则拒绝插入或修改；
- 违约反应
  - 系统拒绝此操作，从而保证了实体完整性；

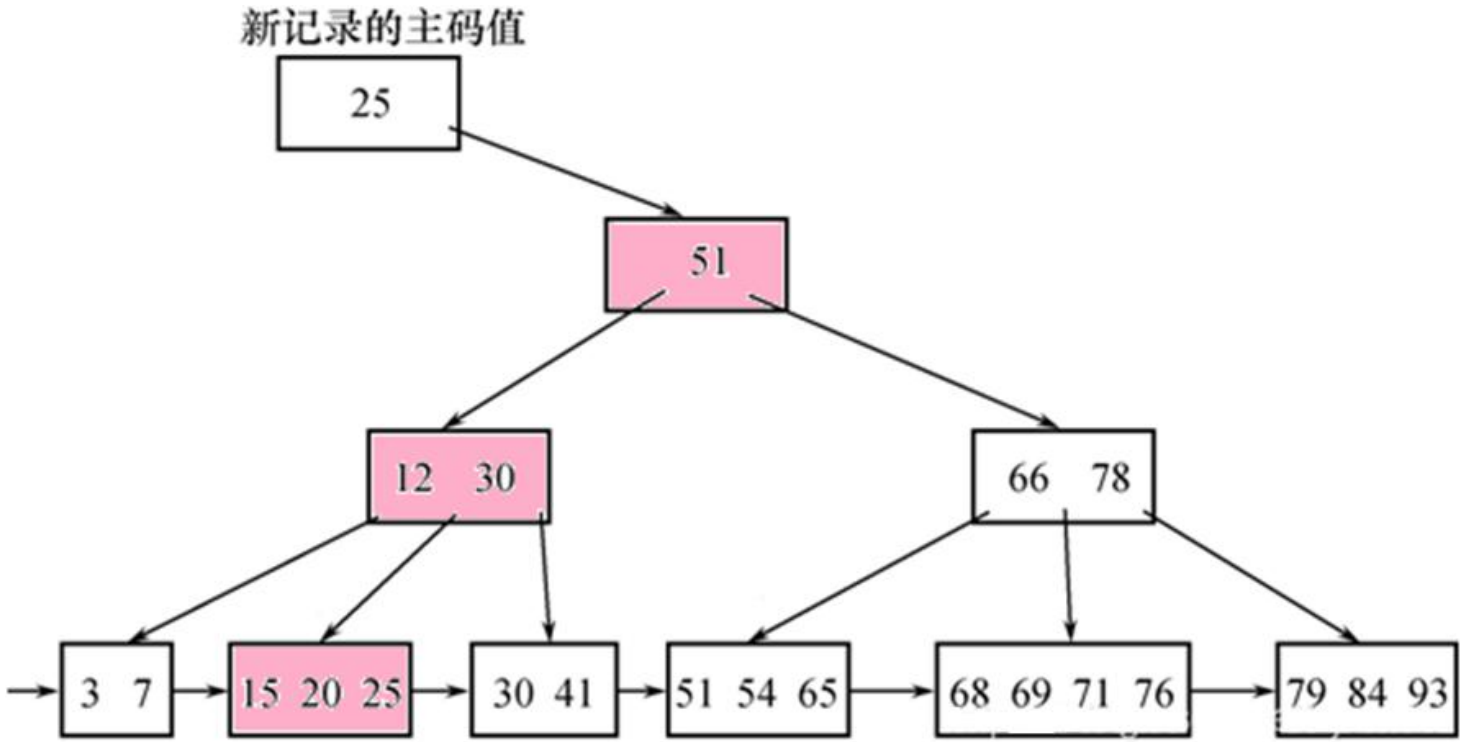
实体完整性检查和违约处理

- 检查记录中主码值是否唯一的方法是进行全表扫描；
  - 依次判断表中每一条记录的主码值与将插入记录上的主码值（或者修改的新主码值）是否相同 ；



检查记录中主码值是否唯一

- 全表扫描十分耗时；
- B+树索引；



- 例：新插入记录的主码值是25
  - 通过主码索引，从B+树的根结点开始查找
  - 读取3个结点：根结点（51）、中间结点（12 30）、叶结点（15 20 25）
  - 该主码值已经存在，不能插入这条记录

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目 录

参照完整性的定义

- 定义参照完整性
  - FOREIGN KEY子句：定义外码列
  - REFERENCES子句：外码相应于哪个表的主码
  - 级联（CASCADE）操作：在删除或修改被参照关系的元组时，同时删除或修改参照关系所有导致不一致的元组

参照完整性的定义

例：建立表SC

```
CREATE TABLE SC
    (Sno CHAR(9)    NOT NULL,
     Cno CHAR(4)    NOT NULL,
     Grade SMALLINT,
     PRIMARY KEY (Sno, Cno),                          /* 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

- 一个参照完整性将两个表中的相应元组联系起来；
- 对被参照表和参照表进行增删改操作时有可能破坏参照完整性，必须进行检查；

参照完整性检查和违约处理

- 例如，对表SC和Student有四种可能破坏参照完整性的情况：

①在参照关系中插入元组：SC表中增加一个元组，该元组的Sno属性值在表Student中找不到一个元组，其Sno属性值与之相等。

②在参照关系中修改外码值：修改SC表中的一个元组，修改后该元组的Sno属性值在表Student中找不到一个元组，其Sno属性值与之相等。

③在被参照关系中删除元组：从Student表中删除一个元组，造成SC表中某些元组的Sno属性值在表Student中找不到一个元组，其Sno属性值与之相等。

④在被参照关系中修改主码：修改Student表中一个元组的Sno属性，造成SC表中某些元组的Sno属性值在表Student中找不到一个元组，其Sno属性值与之相等 。

可能破坏参照完整性的情况

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/级联删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级联删除/设置为空值

参照完整性检查和违约处理

违约处理

- **拒绝（NO ACTION）**：不允许该操作执行（默认策略）；
- **级联（CASCADE）**：当删除或修改被参照表的一个元组导致与参照表不一致时，删除或修改参照表中所有导致不一致的元组；
- **设置为空值**：当删除或修改被参照表的一个元组导致与参照表不一致时，将参照表中所有造成不一致的元组的对应属性设置为空值；

外码是否可以接受空值的问题

- 外码是否能够取空值：依赖于应用环境的语义
- 实现参照完整性：
  - 系统提供定义外码的机制
  - 定义外码列是否允许空值的机制

外码是否可以接受空值的问题

例1:职工—部门数据库包含职工表EMP和部门表DEPT

1　DEPT关系的主码为部门号Deptno

2　EMP关系的主码为职工号Empno，外码为部门号Deptno

称DEPT为被参照关系或目标关系，EMP为参照关系

- 参照关系EMP的某元组的这一列若为空值，表示这个职工尚未分配到任何具体的部门工作,和应用环境的语义是相符;

外码是否可以接受空值的问题

例2：学生—选课数据库，Student关系为被参照关系，其主码为Sno。SC为参照关系，外码为Sno。

- 若SC的Sno为空值：表明尚不存在的某个学生，或者某个不知学号的学生，选修了某门课程，其成绩记录在Grade中，与学校的应用环境是不相符的，因此SC的Sno列不能取空值；

参照完整性的违约处理示例

例：建立表SC

```
CREATE TABLE SC
    (Sno CHAR(9)    NOT NULL,
     Cno CHAR(4)    NOT NULL,
     Grade SMALLINT,
     PRIMARY KEY (Sno, Cno),        /*表级定义实体完整性，Sno、Con都不能取空值*/
     FOREIGN KEY (Sno) REFERENCES Student(Sno)   /* 在表级定义参照完整性*/
         ON DELETE CASCADE          /* 删除Student元组时，级联删除SC中相应元组*/
         ON UPDATE CASCADE,         /* 更新Student表中Sno时，级联更新SC中相应元组*/
     FOREIGN KEY (Cno) REFERENCES Course(Cno)   /* 在表级定义参照完整性*/
         ON DELETE NO ACTION        /* 删除Course元组造成与SC表不一致时，拒绝删除*/
         ON UPDATE CASCADE,         /* 更新Course表中Cno时，级联更新SC中相应元组*/）；
```

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目 录

用户定义的完整性

- 用户定义的完整性是：针对某一具体应用的数据必须满足的语义要求。
- 用户定义的完整性的两类方法：
  - 用CREATE TABLE语句在建表时定义用户完整性约束；
  - 通过触发器来定义用户的完整性规则；

一、属性上的约束条件

- 用CREATE TABLE语句在建表时定义用户完整性约束，可定义三类属性上的完整性约束：
  - 列值非空（NOT NULL短语）
  - 列值唯一（UNIQUE短语）
  - 检查列值是否满足一个条件表达式（CHECK短语）

1、不允许取空值

例：建立表SC，说明Sno、Con、Grade属性不允许取空值

```
CREATE TABLE SC
    (Sno CHAR(9)    NOT NULL,                          /* Sno属性不允许取空值*/
     Cno CHAR(4)    NOT NULL,                          /* Cno属性不允许取空值*/
     Grade SMALLINT    NOT NULL,                       /* Grade属性不允许取空值*/
     PRIMARY KEY (Sno, Cno),                           /* 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 */
    );
```

2、列值唯一

例：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  NOT  NULL,  /* 要求Dname列值唯一，且不能取空值*/
         Loc    VARCHAR(10),
         PRIMARY KEY (Deptno));
```

3、CHECK短语检查

例： 建立学生登记表Student，要求年龄<29，性别只能是‘男’或‘女’，姓名非空

```
CREATE TABLE Student
        (Sno  NUMBER(5)    PRIMARY KEY,        /* 在列级定义主码*/
         Sname  CHAR(20)   NOT NULL,           /* Sname属性不允许取空值*/
         Sage  SMALLINT    CHECK (Sage < 29),          /* Sage属性小于２９*/
         Ssex   CHAR(2)    CHECK (Ssex IN (‘男’, ‘女’)    /* Ssex属性只允许取‘男’或 ‘女’*/
         );
```

3、CHECK短语检查

例：建立表SC，Grade的值在０和１００之间

```
CREATE TABLE SC
    (Sno CHAR(9),
     Cno CHAR(4),
     Grade SMALLINT    CHECK(Grade>=0 AND Grade <=100),    /* Grade取值范围０-１００*/
     PRIMARY KEY (Sno, Cno),
     FOREIGN KEY (Sno) REFERENCES Student(Sno),
     FOREIGN KEY (Cno) REFERENCES Course(Cno) );
```

二、元组上的约束条件

- 用CREATE TABLE语句在建表时定义用户完整性约束，可定义元组上的完整性约束：
  - 设置不同属性之间取值的相互约束条件（CHECK短语）

元组上的约束条件

- 当学生的性别是男时，其名字不能以Ms.打头：

```
CREATE TABLE Student
    (Sno CHAR(9),
     Sname CHAR(20) NOT NULL,          /* Sname非空值*/
     Ssex CHAR(2),
     Sage SMALLINT,
     Sdept CHAR(20),
     PRIMARY KEY (Sno),
     CHECK (Ssex='女' OR Sname NOT LIKE 'MS.%')
    );
```

  - 性别是女性的元组都能通过该项CHECK检查;
  - 当性别是男性时，要通过检查则名字一定不能以'Ms.'打头

完整性小结

- 提供定义完整性约束条件
  - CREATE TABLE语句
  - CREATE TRIGGER语句
  - 可以定义很复杂的完整性约束条件
- 自动执行相应的完整性检查
  - 对于违反完整性约束条件的操作：拒绝执行或者执行事先定义的操作

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录

一、完整性约束命名子句

- SQL在CREATE TABLE语句中提供了CONSTRAINT子句，用来对完整性约束条件命名。
  - 可以灵活增加、删除一个完整性约束条件；
- 完整性约束命名子句

  CONSTRAINT <完整性约束条件名> <完整性约束条件>
  - <完整性约束条件>包括NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK短语等；

完整性约束命名子句

例： 建立学生登记表Student，要求学号在90000～99999之间，年龄＜30，性别只能是‘男’或‘女’，姓名非空

```
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));
```

完整性约束命名子句

例 ：建立职工表EMP，要求每个职工的应发工资不低于3000元，

应发工资实际上就是实发工资列Sal与扣除项Deduct之和。

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

二、修改表中的完整性限制

- SQL提供了ALTER TABLE子句，用来修改表中的完整性限制。

[例] 去掉Student表中对性别的限制。

```
ALTER TABLE Student
    DROP CONSTRAINT C4;
```

修改表中的完整性限制

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

```
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);
```

第八章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录

一、创建断言的语句格式

- SQL提供了CREATE ASSERTION语句，通过声明性断言(declarative assertions)来指定更具一般性的约束。
  - 可以定义涉及多个表或聚集操作比较复杂的完整性约束；
  - 断言创建后，任何对断言中所涉及关系的操作都会触发RDBMS对断言的检查，任何使断言不为真值的操作都会被拒绝执行；

创建断言的语句格式

- 创建断言的语句格式

  CREATE ASSERTION <断言名> <CHECK 子句>

  - 每个断言都被赋予一个名字，<CHECK 子句>中的约束条件与WHERE子句的条件表达式类似；

[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 (*) /* 此断言的谓词涉及聚集操作count*/
                  FROM Course, SC
                  WHERE SC.CNO=COURSE.CNO AND COURSE.CNAME =‘数据库’));
```

创建断言的语句格式

[例] 限制每一门课程最多60名学生选修。

```
CREATE ASSERTION ASSE_SC_CNUM1
    CHECK (60 >= ALL(SELECT count (*)   /* 此断言的谓词涉及聚集操作count*/
                     FROM SC            /* 和分组函数group by的SQL语句*/
                     GROUP by Cno));
```

创建断言的语句格式

[例] 限制每个学期每一门课程最多60名学生选修。

1）首先修改SC表的模式,增加一个‘学期(TERM)’的属性

```
ALTER TABLE SC                          /* 先修改SC表，增加TERM属性，类型是DATE*/
        ADD TERM DATE;
```

2）然后定义断言

```
CREATE ASSERTION ASSE_SC_CNUM2
      CHECK (60 >= ALL(SELECT count (*)   /* 此断言的谓词涉及聚集操作count*/
                       FROM SC            /* 和分组函数group by的SQL语句*/
                       GROUP by Cno, TERM));
```

二、删除断言的语句格式

- 删除断言的语句格式：

  DROP ASSERTION <断言名>

- 如果断言很复杂，系统在检测和维护断言上开销很高；

第七章 数据库完整性

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目 录

触发器

- 触发器(trigger)是用户定义在关系表上的一类由事件驱动的特殊过程。
- 通过触发器来定义用户的完整性规则：
  - 定义其它的完整性约束时，需要用数据库触发器（Trigger）来实现。
  - 一旦由某个用户定义，触发器将被保存在数据库服务器中。
  - 任何用户对该数据的增、删、改操作均由服务器自动激活相应的触发器，在DBMS核心层进行集中的完整性控制。
  - 触发器类似于约束，但比约束更灵活，可以实施更为复杂的检查和操作，具有更精细和更强大的数据控制能力；

一、定义触发器

- 触发器又叫做事件-条件-动作（event-condition-action）规则。
- SQL使用CREATE TRIGGER命令建立触发器，一般格式为：

```
CREATE TRIGGER <触发器名>                /* 每当触发事件发生时，该触发器被激活*/
{BEFORE | AFTER} <触发事件> ON <表名>     /* 指明触发器激活时间是在执行触发事件前或后*/
REFERENCING NEW | OLD ROW AS <变量>      /* REFERENCING 指出引用的变量*/
FOR EACH {ROW | STATEMENT}               /* 定义触发器的类型，指明动作体执行的频率*/
[WHEN <触发条件> ]<触发动作>              /*仅当触发条件为真时，才执行触发动作体*/
```

定义触发器的语法说明

- 创建触发器
  - 只有表的拥有者，即创建表的用户，才可以在表上创建触发器；
  - 一个表上只能创建一定数量的触发器；
- 触发器名
  - 触发器名可以包含模式名，也可以不包含模式名；
  - 同一模式下，触发器名必须是唯一的，而且触发器名和表名必须在同一模式下；

定义触发器的语法说明

- 表名
  - 触发器只能定义在基本表上，不能定义在视图上；
  - 基本表数据发生变化，将激活触发器，该表称为触发器的目标表；
- 触发事件
  - 触发事件可以是INSERT、DELETE、UPDATE；
  - 也可以是几个事件的组合，如INSERT OR DELETE，或UPDATE OF <触发列，…>，进一步指明修改哪些列时激活触发器；
  - AFTER/BEFORE是触发时机，AFTER表示在触发事件的操作执行后激活触发器，BEFORE表示在触发事件的操作执行之前激活触发器。

定义触发器的语法说明

- 触发器类型
  - 触发器按照所触发动作的间隔尺寸可分为行级触发器（FOR EACH ROW）和语句触发器（FOR EACH STATEMENT）；
  - 默认的触发器是语句级触发器；

例：在TEACHER表上创建一个AFTER UPDATE触发器，触发事件是UPDATE语句：

```
UPDATE TEACHER SET Deptno = 5;
```

- 假设表TEACHER有1000行，如果定义的触发器为语句触发器，那么执行完UPDATE语句后触发动作体执行一次；
- 如果是行级触发器，触发动作体将执行1000次；

定义触发器的语法说明

- 触发条件
  - 触发器被激活时，只有当触发条件为真时，触发动作体才执行，否则触发动作体不执行；
  - 如果省略WHEN触发条件，则触发动作体在触发器激活后立即执行；
- 触发动作体
  - 触发动作体即可以是一个匿名PL/SQL（Procedural Language/SQL，过程化SQL语言）过程块，也可以是对已创建存储过程的调用；
  - 如果是行级触发器，用户可在过程体中使用NEW和OLD引用UPDATE/INSERT事件之后的新值和UPDATE/DELETE事件之前的旧值；
  - 如果是语句级触发器，则不能在触发动作体中使用NEW或OLD进行引用；

定义触发器的示例

例：当对表SC的Grade属性进行修改时，若分数增加了10%，则将此次操作记录到另一个表SC_U（Sno、Cno、Oldgrade、Newgrade）中，其中Oldgrade是修改前的分数，Newgrade是修改后的分数。

```
CREATE TRIGGER SC_T                          /* SC_T是触发器的名字*/
AFTER UPDATE OF Grade ON SC                  /* UPDATE OF Grade ON SC是触发事件*/
                /* AFTER是触发的时机，表示当对SC的Grade属性修改完后再触发下面的规则*/
REFERENCING
      OLDROW AS OldTuple
      NEWROW AS NewTuple
 FOR EACH ROW            /* 行级触发器，每执行一次Grade更新，下面的规则就执行一次*/
 WHEN (NewTuple.Grade>=1.1*OldTuple.Grade)    /* 触发条件，只有该条件为真时才执行*/
  BEGIN
      INSERT INTO SC_U (Sno, Cno, OldGrade, NewGrade)          /* 下面的insert操作*/
      VALUES(OldTuple.Sno, OldTuple.Cno, OldTuple.Grade, NewTuple.Grade)
  END
```

定义触发器的示例

例：将每次对表Student的插入操作所增加的学生个数记录到表Student-InsertLog中。

```
CREATE TRIGGER Student_Count                    /* Student_Count是触发器的名字*/
AFTER INSERT ON Student          /* 指明触发器激活的时间是在执行INSERT之后*/
REFERENCING
        NEW TABLE AS DELTA
FOR EACH STATEMENT
        /* 语句级触发器，即执行完INSERT语句后下面的触发动作体才执行一次*/
 BEGIN
        INSERT INTO StudentInsertLog (Numbers)
        SELECT COUNT(*) FROM DELTA
END
```

定义触发器的示例

例： 定义一个BEFORE行级触发器，为教师表Teacher定义完整性规则

“教授的工资不得低于4000元，如果低于4000元，自动改为4000元”

```
CREATE TRIGGER Insert_OR_Update_Sal   /* 对教师表插入或更新时激活触发器*/
BEFORE INSERT OR UPDATE ON Teacher    /* BEFORE触发事件*/
REFERENCING
        NEW row AS new Tuple
FOR EACH ROW                          /* 行级触发器*/
BEGIN                                 /* 定义触发动作，这是一个PL/SQL过程块*/
        IF (newtuple.Job = '教授' AND  (newtuple.Sal < 4000)
            THEN newtuple.Sal = 4000;
                /* 因为是行级触发器，可在过程体使用插入或更新操作后的新值*/
        END IF;
END;                                  /* 触发动作体结束*/
```

二、执行触发器

- 触发器的执行是由触发事件激活，并由数据库服务器自动执行；
- 一个数据表上可能定义了多个触发器，如多个BEFORE、AFTER触发器；
- 同一个表上的多个触发器激活时遵循如下执行顺序：
  - 执行该表上的BEFORE触发器；
  - 激活触发器的SQL语句；
  - 执行该表上的AFTER触发器；
- 同一个表上的多个BEFORE(AFTER)触发器，遵循“谁先创建谁先执行”的原则，即按触发器创建的时间先后顺序执行。

三、删除触发器

- 删除触发器的SQL语法如下：

DROP TRIGGER <触发器> ON <表名>;

- 触发器必须是一个已经创建的触发器，只能由具有相应权限的用户删除；

数据库完整性小结

- 完整性机制的实施会极大地影响系统性能；
- 不同的数据库产品对完整性的支持策略和支持程度是不同的
  - 许多数据库管理系统对完整性机制的支持比对安全性的支持要晚得多也弱得多；
  - 数据库厂商对完整性的支持越来越好，不仅在能保证实体完整性和参照完整性，而且能在DBMS核心定义、检查和保证用户定义的完整性约束条件；

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

THANK YOU !

饮水思源 爱国荣校