数据库系统原理

教程：数据库系统概论（第5版）

结合：CMU 15-445/645 INTRO TO DATABASE SYSTEMS

华中科技大学 计算机学院

左琼

第五章 数据库完整性

Principles of Database Systems

第五章 数据库完整性

5.1 数据库完整性概述

5.2 实体完整性

5.3 参照完整性

5.4 用户定义的完整性

5.5 完整性约束命名字句

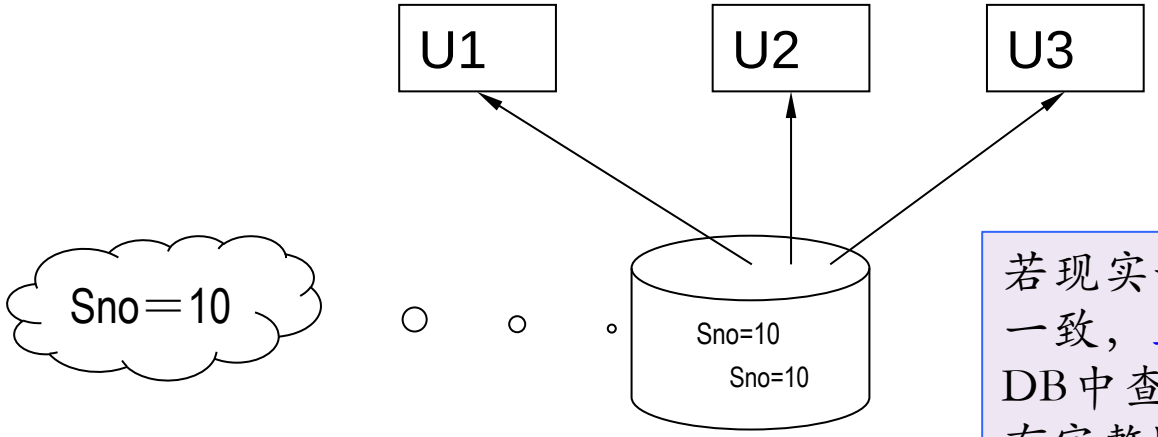
*5.6 域中的完整性限制

5.7 触发器

5.8 小结

5.1 数据库完整性概述

- 数据库的完整性：
  - 数据的正确性和相容性。
    - 正确性指数据库中数据与现实世界的实际情况是相符的。
    - 相容性指数据库中数据自身不存在自相矛盾的现象。



若现实世界的Sno与DB中的Sno保持一致，且U1、U2、U3等所有用户从DB中查询的结果均为10，则称A具有完整性，否则就是不完整的。

之所以要引入数据完整性是为了在数据的添加、删除、修改等操作中不出现数据的破坏或多个表数据不一致。

数据库完整性

- 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

DBMS的完整性控制功能:

1. 为用户提供定义完整性约束条件的机制

   DBMS将这些约束条件作为模式的一部分存入数据库中。

2. 监督系统中执行的操作是否违反完整性限制条件

   应进行完整性检查的操作有INSERT/DELETE/UPDATE。

3. 对违反完整性约束的情况进行相应处理

   拒绝执行 / 级联操作

思考：完整性控制由DBMS实现和由应用程序实现有什么区别？哪种方式更好？

5.2 实体完整性

- 实体是客观存在且相互区别的事物。

1. 定义实体完整性

实体完整性在关系模型中通过关系的主码（PRIMARY KEY）来体现。

PRIMARY KEY在CREATE TABLE语句中定义。

- 【方法一】列级约束

```
CREATE  TABLE   COURSE (
    cno  CHAR(8)  PRIMARY KEY,
    cname  CHAR(30),
    credit   SMALLINT
    ) ;
```

- 【方法二】表级约束

```
CREATE  TABLE   COURSE (
        cno  CHAR(8),
        cname  CHAR(30),
        credit   SMALLINT,
        PRIMARY KEY (cno)
    ) ;
```

5.2 实体完整性

- 如果主码由多个列组成，则只能用第二种方法定义主码。

```
CREATE TABLE SC (
        sno CHAR(8),
        cno CHAR(8),
        grade SMALLINT,
        PRIMARY KEY(sno, cno)
);
```

2. 如何检查实体完整性

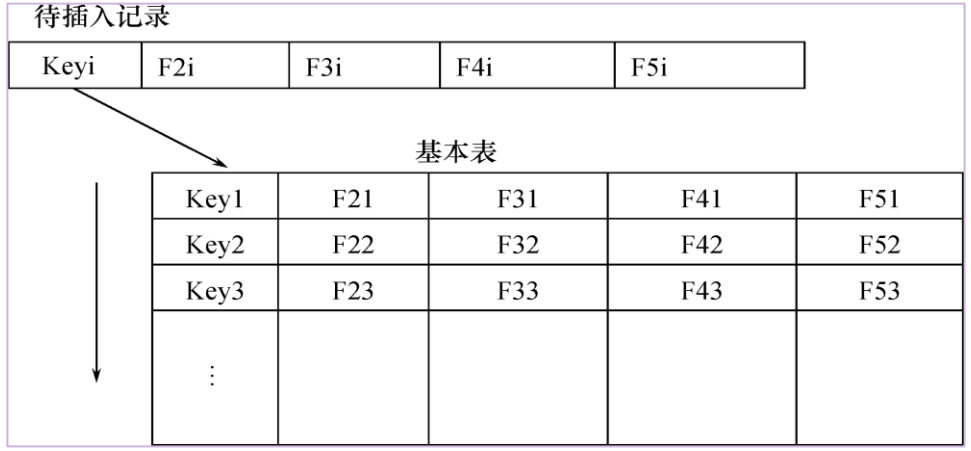
对基表插入数据或对主码列进行更新时，DBMS将自动对该操作进行实体完整性检查。包括：

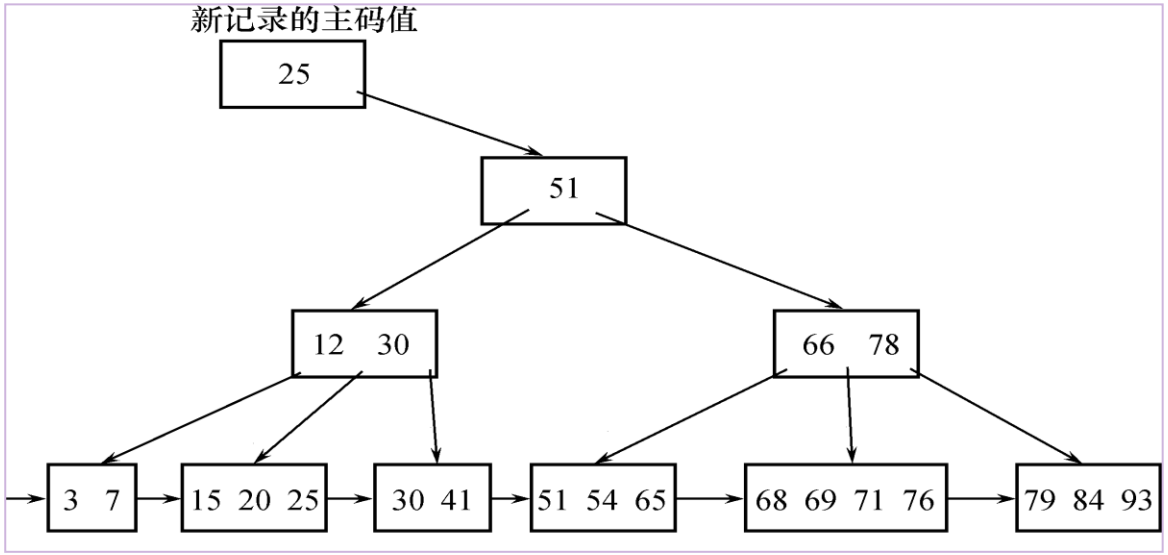
- 主码值是否唯一，否则拒绝执行该操作；
- 各主码列的值是否为空，是则拒绝执行该操作。

5.2 实体完整性

2. 如何检查实体完整性

检查记录中主码值是否唯一的一种方法是进行全表扫描，可以利用索引。





5.3 参照完整性

引用数据与被引用数据应该是一致的。

1. 如何定义参照完整性

参照完整性在关系模型中通过关系的外码（FOREIGN KEY）来体现。

FOREIGN KEY同样在CREATE TABLE语句中定义。

□ 方法一：列级约束

```
CREATE TABLE SC (
    sno CHAR(8) REFERENCES STUDENT(sno),
    cno CHAR(8) REFERENCES COURSE(cno),
    grade SMALLINT,
    PRIMARY KEY(sno, cno)
    ) ;
```

□ 方法二：表级约束

```
CREATE TABLE SC (
        sno CHAR(8),
        cno CHAR(8),
        grade SMALLINT,
        PRIMARY KEY(sno, cno),
        FOREIGN KEY(sno) REFERENCES STUDENT(sno),
        FOREIGN KEY(cno) REFERENCES COURSE(cno)
    ) ;
```

5.3 参照完整性

2. 如何检查参照完整性

外码的取值限制：

- NULL（思考：如果该外码同时又是子表的主属性呢？）
- 引用父表中的某值

□ 可能破坏参照完整性的情况及违约处理：

被参照表（如Stu）	参照表（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

No Action

Cascade delete

Cascade update

Set Null

5.3 参照完整性

1）在子表中插入元组

仅当父表中存在与插入元组相应的元组时，系统执行在子表中插入元组的操作，否则拒绝此插入操作。

【例】SC表的sno引用了Student表的sno。

如图： 在SC表中插入一条sno = '4'的学生选课记录：

Student

Sno	Sname	Ssex	Sage	Sdept
1	李勇	男	19	CS
2	刘晨	女	18	CS
3	王敏	女	18	MA

4 ?

NO ACTION

SC

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
3	1	80
4	**1**	**89**

INSERT INTO SC
VALUES('4','1',89);

5.3 参照完整性

2）修改子表外码

如果父表中存在待修改值，则执行；否则不允许执行此类修改操作。

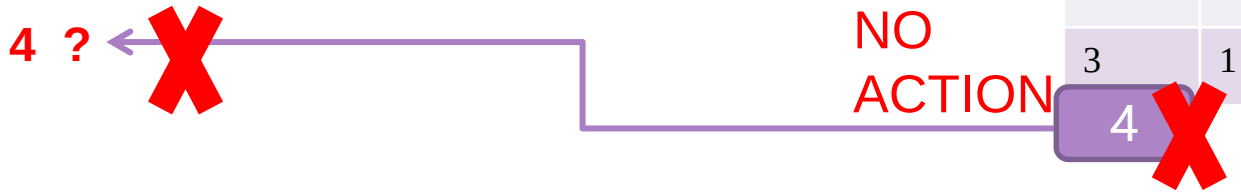
【例】修改SC表中的外键sno，需要检查父表中是否存在该值:

Student

Sno	Sname	Ssex	Sage	Sdept
1	李勇	男	19	CS
2	刘晨	女	18	CS
3	王敏	女	18	MA

SC

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
3	1	80




```
UPDATE SC
SET Sno=4
WHERE Sno=3;
```

5.3 参照完整性

3）修改父表主码（若子表中有相关参照记录）

- 拒绝修改：拒绝执行此类操作。
- 级联修改：将子表中相关记录在外码上的值一起自动修改。
- 置空修改：将子表中相关记录在外码上的值全部置为NULL。

Student

Sno	Sname	Ssex	Sage	Sdept
1	李勇	男	19	CS
2	刘晨	女	18	CS
3	王敏	女	18	MA
4				

CASCADE UPDATE

SC

Sno	Cno	Grade
1	1	92
1	2	85
2	1	88
2	2	90
4	1	80

```
UPDATE Student
SET Sno=4
WHERE Sno=3;
```

5.3 参照完整性

4) 删除父表元组（若子表中有相关参照记录）

- 拒绝删除: 发出警告，拒绝执行此类操作。
- 级联删除: 删除父表中元组的同时，自动删除子表中的相关元组。
- 置空删除: 删除父表中元组的同时，自动将子表中的相关元组的外码置空。

Course

Cno	Cname	Cpno	Ccredit
1	数据库	NULL	3
2	高等数学		4
3	信息系统	1	3
4	操作系统	5	3
5	数据结构		4
6	数据处理		2

SET NULL

```
DELETE FROM Course
WHERE Cno='6';
```

5.3 参照完整性

- 如何指定修改或删除父表时的违约处理动作?

在定义参照完整性约束时，可同时指定删除或修改时的违约处理动作，如NOT ACTION、SET NULL或CASCADE。缺省动作为NOT ACTION。

形如：

```
CREATE  TABLE   SC (
……
FOREIGN  KEY(sno) REFERENCES STUDENT(sno)
ON DELETE SET NULL          ×
ON UPDATE CASCADE,          √
……
);
```

语法正确；
但是此处sno为主属性，不可为空，
因此，这个处理的设置不正确

违约处理

【例】 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 */
);
```

数据库完整性应用实例

- 以疫情期间在超市发现冻品表面新冠检测呈阳性为例。
- 在设计超市数据库时，要考虑在若干关键表之间建立关联性：

会员表

会员卡编号	姓名	身份证号	电话号码	会员等级	。。。。

商品

商品编号	商品名	单价	数量	。。。。

销售单据

会员卡编号	商品编号	。。。。		

- 若某超市在售的某个冻品被发现表面新冠检测呈阳性。
- 应急处理方式：

1）该商品下架；

2）通过销售单据表找到买此类商品的会员卡编号，排查与此关联的会员表，找到具体的购物人。

3）疾控人员可及时对经超市数据库排查出的密接人员进行核酸检测、隔离疑似人员等措施，尽可能减少对社会危害。

5.4 用户定义的完整性约束

- 在一个具体应用中，数据需满足一些特定的语义要求。

1）空值约束（列级）

指定属性的值不允许为空值（NOT NULL）

```
CREATE  TABLE   COURSE (
        cno  CHAR(8),
        cname  CHAR(30) NOT NULL,
        credit   SMALLINT,
        PRIMARY KEY (cno)
);
```

5.4 用户定义的完整性约束

2）唯一性约束（列级/表级）

指定属性或属性组的值必须唯一（UNIQUE）

方法一：

```
CREATE TABLE COURSE (
cno CHAR(8),
cname CHAR(30) NOT NULL UNIQUE,
credit SMALLINT,
PRIMARY KEY (cno)
);   /* 列级约束 */
```

方法二：

```
CREATE TABLE COURSE (
  cno CHAR(8),
  cname CHAR(30) NOT NULL,
  credit SMALLINT,
  PRIMARY KEY (cno),
  UNIQUE(cname)
);  /* 表级约束 */
```

5.4 用户定义的完整性约束

3）CHECK约束（列级/表级）

检查一个属性或多个属性的取值是否满足一个布尔表达式。

方法一：

```
CREATE  TABLE   SC (
  sno  CHAR(8),
  cno  CHAR(8),
  grade  SMALLINT CHECK(grade <=100),
  ……) ;          /* 列级约束 */
```

方法二：

```
  CREATE TABLE EMPLOYEE(
                ……
        age SMALLINT,
        work_year SMALLINT,
        CHECK(age>work_year),
        ……
    );   /* 表级约束 */
```

5.5 完整性约束命名子句

- 完整性约束与列一样是关系模式的构成元素。
- 关系中的列是必须命名的，但是约束却不一定要命名。

思考：为什么列必须命名而约束可以不命名？约束不命名可能带来什么问题?

- 如何给完整性约束命名?

约束命名子句:

CONSTRAINT <约束名> <约束说明>

其中，<约束说明> 可以是PRIMARY KEY子句，FOREING KEY 子句，NOT NULL子句，UNIQUE子句和CHECK子句。

5.5 完整性约束命名子句

在CREATE TABLE语句中使用约束命名子句为约束命名：

```
CREATE  TABLE   SC (
        sno  CHAR(8),
        cno  CHAR(8),
        grade  SMALLINT
        CONSTRAINT C1 CHECK(grade<=100),
        CONSTRANT PK PRIMARY KEY(sno, cno),
        CONSTRANT FK1 FOREIGN KEY(sno) REFERENCES STUDENT(sno),
        CONSTRANT FK2 FOREIGN KEY(cno) REFERENCES COURSE(cno)
  ) ;
```

5.5 完整性约束命名子句

- 在ALTER TABLE语句中通过约束名删除约束：

```
ALTER TABLE SC DROP CONSTRANT C1;
```

- 在ALTER TABLE语句中使用约束命名子句增加新的约束：

```
ALTER  TABLE   S
    ADD CONSTRAINT C2
        CHECK(age>=15 AND age<=30) ;
```

提问

不会！

- 思考：增加新约束，数据库会不会对已经在库里的数据检查一遍约束条件呢?

【例】 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。

```
//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);
```

5.6 域中的完整性限制*

- SQL支持域的概念，用CREATE DOMAIN语句建立一个域以及该域应该满足的完整性约束条件。

【例】创建一个名为GenderDomain的域，它是值为‘男’‘女’字符

```
CREATE DOMAIN GenderDomain CHAR(2) CHECK (VALUE IN ('男'，'女') );
// 对Ssex的说明可以改写为：  Ssex  GenderDomain
```

【例】建立一个性别域GenderDomain，并对其中的限制命名为GD

```
CREATE DOMAIN GenderDomain CHAR(2)
        CONSTRAINT GD CHECK ( VALUE IN ('男'，'女') );
```

【例】删除域GenderDomain的限制条件GD。

```
ALTER  DOMAIN  GenderDomain  DROP CONSTRAINT GD;
```

【例】在域GenderDomain上增加限制条件GDD。

```
ALTER  DOMAIN  GenderDomain  ADD CONSTRAINT GDD CHECK (VALUE IN ( ‘1’，‘0’) );
//上两步将性别的取值范围由('男'，'女')改为 ( '1'，'0')
```

断言*

- 申明性断言（Declarative assertions）：指定更具一般性约束，可涉及多表或聚集操作的较复杂的完整性约束。
- 创建断言的语句格式：

```
CREATE ASSERTION <断言名> <CHECK 子句>;
```

其中：<CHECK 子句>中的约束条件与where子句的条件表达式类似。

【例】限制每门课程最多60名学生选修。

```
CREATE ASSERTION ASSE_SC1
    CHECK ( 60>=ALL ( SELECT  Count(*)
                      FROM  SC
                      GROUP BY Cno )
          );
```

【例】限制DB课程最多60名学生选修。

```
CREATE ASSERTION ASSE_DB_NUM
    CHECK ( 60>=(SELECT  Count(*)
                 FROM  Course, SC
                 WHERE Course.cno=SC.cno
                       AND Cname='DB' );
```

5.7 触发器

什么是触发器?

触发器是定义在基表上的一种数据库对象，它指定：在执行对表的某些操作的时候，另一些操作也同时被执行。

又叫做：事件-条件-动作规则（ECA规则）。

- 定义一个触发器应包含哪些要素?
  - 触发器的名字
  - 触发器关联的表名
  - 哪些操作执行时将引发其它操作被执行（触发事件）
  - 被触发后执行的操作（触发动作）
  - 执行触发动作的时机（触发条件）

5.7.1 定义触发器

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>;

说明：

- <触发事件>可以是INSERT、DELETE或UPDATE，或这些事件的组合，UPDATE后面可以带OF <触发列，…>，指明只当哪些列被修改时才激活触发器。BEFORE或AFTER表示触发器是在触发事件之前还是之后被激活。
- FOR EACH ROW表示该触发器为行级触发器，触发事件每影响一条元组都将激活一次触发器。
- FOR EACH STATEMENT表示该触发器为语句级触发器，触发事件只激活一次触发器。

5.7.1 定义触发器

DBMS如何执行触发器：

- 触发器被激活后，先检查<触发条件>，当其为真时，<触发动作体>才执行；如果没有指定<触发条件>，则<触发动作体> 在触发器被激活后立即执行。
- <触发动作体>是一段程序，如果触发器是行级触发器，则这段程序中还可以使用NEW和OLD分别引用触发事件发生前后的元组值。

【例1】保留表BOOKS中被删除的书的记录。

```
CREATE TRIGGER BOOKS_DELETE
AFTER DELETE ON BOOKS
FOR EACH ROW
INSERT INTO BOOKS_DELETED_LOG
        VALUES(OLD.TITLE);
```

5.7.1 定义触发器

【例2】假设在Teacher表上创建了一个触发条件为AFTER UPDATE的触发器，假设表Teacher有1000行。

执行语句：UPDATE Teacher SET Deptno=5;

则：

- 如果该触发器为语句级触发器FOR EACH STATEMENT，那么执行完该语句后，触发动作只发生一次；
- 如果是行级触发器FOR EACH ROW，触发动作将执行1000次。

- 提问：
  - 如果是个after触发器，sql执行完了然后触发器失败了，会撤销掉sql的执行结果吗？
  - 如果是个行级的触发器，一行的触发器失败了，其它行还会运行吗？

5.7.1 定义触发器

【例3】教授的工资不得低于10000元，否则自动改为10000元。

```
CREATE  TRIGGER  Update_Sal
    BEFORE INSERT OR UPDATE OF Sal, Pos
    ON Teacher
    FOR EACH ROW
    WHEN ( NEW.Pos = '教授'   AND   NEW.Sal<10000 )
      NEW.Sal := 10000;
```

下列语句执行时将产生什么结果？

A. INSERT INTO Teachar(Sal, Pos) VALUES(8000, '讲师')

B. INSERT INTO Teachar(Sal, Pos) VALUES(10000, '教授')

C. UPDATE教师A的职称为教授，且教师A的工资为8000

D. UPDATE教师A的工资为10500，且该教师的职称为教授

5.7.1 定义触发器示例

【例4】定义AFTER行级触发器，当教师表Teacher的工资发生变化后，就自动在工资变化表Sal_log中增加一条相应记录。

```
1）CREATE TRIGGER Insert_Sal
    AFTER INSERT ON Teacher        /*触发事件是INSERT*/
      FOR EACH ROW
       AS BEGIN
          INSERT INTO Sal_log VALUES(
                        new.Eno，new.Sal，CURRENT_USER，CURRENT_TIMESTAMP);
       END;
2）CREATE TRIGGER Update_Sal
    AFTER UPDATE ON Teacher      /*触发事件是UPDATE */
    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;
```

```
//工资变化表Sal_log
      CREATE TABLE Sal_log
      (Eno   NUMERIC(4)  references teacher(eno)，
       Sal    NUMERIC(7，2)，
       Username  char(10)，
       Date   TIMESTAMP )
```

5.6.2 激活触发器

- 触发器的执行，是由触发事件激活的，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

  同一个表上的多个触发器激活时遵循如下的执行顺序：

  (1) 执行该表上的BEFORE触发器；

  (2) 激活触发器的SQL语句；

  (3) 执行该表上的AFTER触发器。

  - 有多个触发器时，这些触发器按创建时间顺序依次执行。
  - 任一触发器的执行失败都将中止整个操作。

【例】执行修改某个教师工资的SQL语句，激活上述定义的触发器。

```
UPDATE Teacher SET Sal=800 WHERE Ename='陈平';
```

执行顺序是：

- 执行触发器Insert_Or_Update_Sal
- 执行SQL语句
- 执行触发器Insert_Sal
- 执行触发器Update_Sal

3. 触发器的撤销

- 语法格式：DROP TRIGGER <触发器名> ON <表名>;
- 说明：触发器必须是一个已经创建的触发器，并且只能由具有相应权限的用户删除。

【例】 DROP TRIGGER Insert_Sal ON Teacher;

- 注：触发器是与表相关联的，因此，表的撤销将引起该表上的所有触发器同时被撤销。

不同的DBMS对于触发器的支持方式和语法是有所区别的。这是因为SQL标准直到SQL99才将触发器纳入标准，而在此之前几乎所有主流的商用DBMS都已经实现了对触发器的支持。

Mysql的触发器*

- 触发器语法：

```
CREATE    [DEFINER { 'user' | CURRENT_USER }]
  TRIGGER trigger_name
  trigger_time trigger_event     // BEFORE| AFTER  ;  INSRET | UPDATE | DELETE
  ON table_name
  FOR EACH ROW
  [trigger_order]
  trigger_body
```

- 查看数据库中所有的触发器：

```
SHOW TRIGGERS [from db_name]
```

- 查看触发器结构：查information_schema.TRIGGERS表

```
select * from information_shema.Triggers where TRIGGER_NAME= `...'  ;
```

触发器的作用

- 强制实现由主键和外键所不能保证的参照完整性和数据的一致性；
- 实现比CHECK语句更为复杂的约束（涉及多表）；
- 找到数据修改前后表状态的差异，并基于此差异采取行动；
- 级联运行修改数据库中相关表，自动触发相关操作；
- 跟踪变化，禁止/回滚不合规则的操作，防止非法修改数据；
- 返回自定义的错误消息（约束做不到）；
- 自动调用存储过程。

小结

本章作业：P166　7

- 数据库的完整性是为了保证数据库中存储的数据是正确的。
- 为此，RDBMS提供了一套完整性机制，即：
  - 完整性约束定义机制
  - 完整性检查机制
  - 违背完整性约束条件时RDBMS应采取的动作。
- 关系数据库中的三类完整性约束为：
  - 实体完整性约束（由主键约束来实现）
  - 参照完整性约束（由外键约束来实现）
  - 用户定义的完整性约束（NOT NULL，UNIQUE，CHECK，触发器）
- 触发器是由事件驱动的特殊过程，它的功能非常强大，并不仅限于完整性控制，目前被广泛应用于数据库应用中。

前5章总结

基础篇

- DB,DBS,DBMS
- 数据模型
- 关系数据库模型
- 关系代数
- SQL语言*
- DB安全性
- DB完整性

细化：

- ☐ 概念：DB, DBMS, DBS 与 数据库系统组成
- ☐ 数据独立性与数据库系统结构（三级模式，两层映像）
- ☐ 数据模型
  - 数据模型的三要素：数据结构、数据操作、约束
  - 概念模型：E-R 模型
  - 逻辑模型：层次、网状、关系
- ☐ 关系模型
  - 关系数据结构及定义（关系，候选码，主码，外码，关系模式，关系数据库）
  - 关系的完整性约束（实体完整性，参照完整性，用户定义的完整性）
- ☐ 关系代数（5个基本运算（并，差，笛卡儿积，选择，投影）以及交，连接，自然连接，除）
- ☐ SQL语言4大功能：
  - 数据定义、数据查询、数据更新、数据控制
- ☐ 除了基本表之外，索引 与 视图 的概念及其作用。
- ☐ 数据库安全控制：自主存取控制、强制存取控制
- ☐ 关系数据库中的3类完整性约束为：
  - 实体完整性约束（主键）
  - 参照完整性约束（外键）
  - 用户定义的完整性约束（NOT NULL，UNIQUE，CHECK，断言、触发器）