



第五章 数据库完整性



■ 数据库的完整性

□ 数据的正确性、（有效性）和相容性。

- 正确性是指数据是符合现实世界语义、反应当前实际状况的。
- 相容性是指数据库同一对象在不同关系表中的数据是复合逻辑的。
- 有效性是指数据是否属于所定义的有效范围。



■ 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



为维护数据库的完整性，DBMS必须：

- 1.提供定义完整性约束条件的机制
- 2.提供完整性检查的方法
- 3.进行违约处理

第五章 数据库完整性

5.1 实体完整性

5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名子句

5.5 触发器（自学）





5.1 实体完整性

5.1.1 实体完整性定义

5.1.2 实体完整性检查和违约处理



5.1.1 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



[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) );
```




(2) 在表级定义主码

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



[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*/
```

```
);
```



5.1.2 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：
 1. 检查主码值是否唯一，如果不唯一则拒绝插入或修改
 2. 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

检查记录中主码值是否唯一的一种方法是进行全表扫描

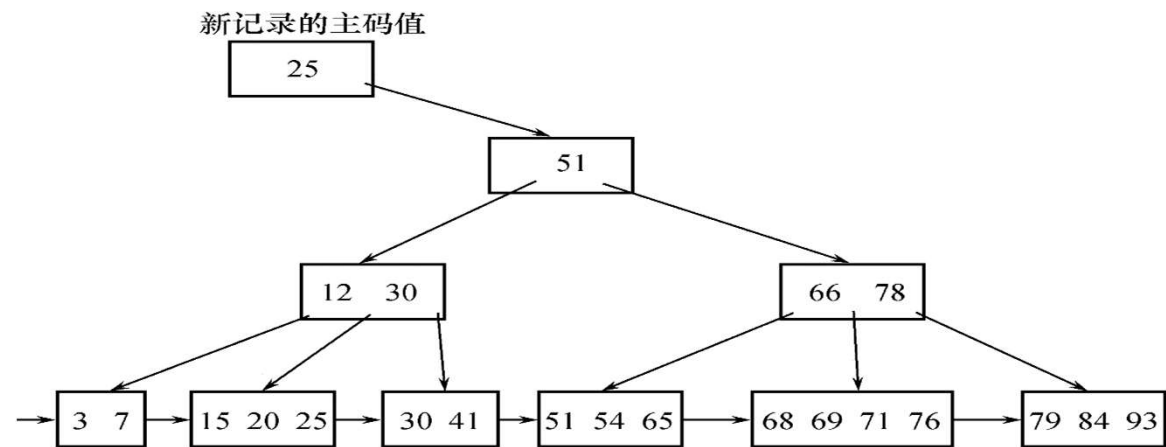
待插入记录

Key _i	F2 _i	F3 _i	F4 _i	F5 _i
------------------	-----------------	-----------------	-----------------	-----------------

基本表

Key1	F21	F31	F41	F51
Key2	F22	F32	F42	F52
Key3	F23	F33	F43	F53
⋮				

索引





5.2 参照完整性


5.2.1 参照完整性定义

5.2.2 参照完整性检查和违约处理



5.2.1 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



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) );
```




5.2 参照完整性

5.2.1 参照完整性定义

5.2.2 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



■ 参照完整性违约处理

1. 拒绝(NO ACTION)执行

- 默认策略

2. 级联(CASCADE)操作

3. 设置为空值 (SET NULL)

- 对于参照完整性，除了应该定义外码，还应定义外码列是否允许空值

[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*/);



5.3 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



5.3 用户定义的完整性

5.3.1 属性上的约束条件的定义

5.3.2 属性上的约束条件检查和违约处理

5.3.3 元组上的约束条件的定义

5.3.4元组上的约束条件检查和违约处理



5.3.1 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)



1. 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 */



2. 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)
);
```



3. 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)
);
```



5.3.2 属性上的约束条件检查和违约处理

- 插入元组或修改**属性的值**时，RDBMS检查属性上的约束条件是否被满足
- 如果不满足则操作被**拒绝执行**



5.3 用户定义的完整性

5.3.1 属性上的约束条件的定义

5.3.2 属性上的约束条件检查和违约处理

5.3.3 元组上的约束条件的定义

5.3.4 元组上的约束条件检查和违约处理



5.3.3 元组上的约束条件的定义

- 在CREATE TABLE时可以用CHECK短语定义元组上的约束条件，即元组级的限制
- 同属性值限制相比，元组级的限制可以设置不同属性之间的取值的相互约束条件



[例8] 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(8) NOT NULL,
```

```
Ssex CHAR(2), Sage SMALLINT,
```

```
Sdept CHAR(20), PRIMARY KEY (Sno),
```

```
CHECK (Ssex='女' OR Sname NOT LIKE 'Ms.%')
```

```
/*定义了元组中Sname和 Ssex两个属性值之间的约束条件*/
```

```
);
```

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



5.3.4 元组上的约束条件检查和违约处理

- 插入元组或修改属性的值时，RDBMS检查元组上的约束条件是否被满足
- 如果不满足则操作被**拒绝执行**



5.4 完整性约束命名子句

■ CONSTRAINT 约束格式

CONSTRAINT <完整性约束条件名>

[PRIMARY KEY短语

|FOREIGN KEY短语

|CHECK短语]




[例9] 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)
);
```



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



[例10] 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);
```

第五章 数据库完整性

5.1 实体完整性


5.2 参照完整性

5.3 用户定义的完整性

5.4 完整性约束命名子句

5.5 触发器（自学）





触发器是一种特殊类型的存储过程，不同于一般存储过程，它是通过**事件**（常见的触发事件就是对数据表的insert、delete、update操作）**进行触发而被执行的**，而存储过程可以通过存储过程名称而被直接调用。

触发器是一个功能强大的工具，它使每个站点可以在有数据修改时自动强制执行其业务规则。如：

- 可在写入数据前，强制检验或者转换数据(保证护数据安全)
- 触发器发生错误时，前面用户已经执行成功的操作会被撤销，类似事务的回滚



MYSQL存储过程定义:

```
CREATE PROCEDURE sp_name  
  ([[ IN | OUT | INOUT ] param_name  
  datatype [...]])  
  Valid SQL routine statement
```



存储过程示例:

```
mysql> delimiter //  
mysql> CREATE PROCEDURE Sno_Count_Cno (IN p_sno CHAR(8), OUT  
Count_Cno INT)  
    BEGIN  
        SELECT COUNT(cno) INTO Count_Cno FROM SC WHERE Sno=  
p_sno;  
    END //  
mysql> delimiter ;  
mysql> CALL Sno_Count_Cno('2014001', @n); --存放Count_Cno  
mysql> SELECT @n; ---结果显示
```



定义触发器包括：

- 触发器名
- 所属表或者视图
- 触发事件（insert, delete, update）
- 触发时机（before/After）
- 触发器类型（行级/语句级）




触发器定义

```
CREATE [DEFINER = user] TRIGGER trigger_name  
    trigger_time trigger_event  
    ON tbl_name FOR EACH ROW
```

```
    trigger_body
```

```
trigger_time: { BEFORE | AFTER }
```

```
trigger_event: { INSERT | UPDATE | DELETE }
```

- 
- you can refer to columns in **the subject table** (the table associated with the trigger) by using the aliases **OLD** and **NEW**.
 - **OLD.col_name** refers to a column of an existing row before it is updated or deleted.
 - **NEW.col_name** refers to the column of a new row to be inserted or an existing row after it is updated.



触发器类型	new和old的使用
INSERT型触发器	没有 old, 只有 new, new 表示将要(插入前)或者已经增加(插入后)的数据
UPDATE型触发器	既有 old 也有 new, old 表示更新之前的数据, new 表示更新之后的数据
DELETE型触发器	没有 new, 只有 old, old 表示将要(删除前)或者已经被删除(删除后)的数据

触发器示例

```
mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
-> //
Query OK, 0 rows affected (1.05 sec)

mysql> delimiter ;
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
->      FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.10 sec)

mysql> SET @sum = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO account VALUES(137,14.98), (141,1937.50), (97,-100.00);
Query OK, 3 rows affected (0.08 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> SELECT @sum AS 'Total amount inserted';
+-----+
| Total amount inserted |
+-----+
|                1852.48 |
+-----+
1 row in set (0.00 sec)
```



课堂练习

- 在学生基本信息表Student上创建一个触发器Trigger_student,该触发器被Update操作触发,当用户在Student表中修改一条学生记录的学号时,同时自动更新学生选课表SC中相应的学号。
- --需要删除外键约束



```
mysql> delimiter //  
mysql> Create Trigger Trigger_student before Update On Student  
    -> For each row  
    -> Begin  
    ->   Update SC Set sno=New.sno where sno =OLD.sno;  
    -> End //  
Query OK, 0 rows affected (0.10 sec)  
  
mysql> Delimiter ;
```

```
mysql> Select * from student;
```

Sno	Sname	Ssex	Sage	Sdept
2014001	李勇	男	20	CS
2014002	刘晨	女	19	CS
2014004	张立	男	19	IS
2014005	王敏	女	18	MA

```
4 rows in set (0.00 sec)
```

```
mysql> Select * from sc;
```

Sno	Cno	Grade
2014001	1	92
2014001	2	85
2014001	3	88
2014002	2	90
2014002	3	80

```
5 rows in set (0.00 sec)
```



```
mysql> Update student set sno='2014003' where sno='2014002';  
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails ('testml`.`sc`, CONSTRAINT `sc_ibfk_1` FOREIGN KEY (`sno`) REFERENCES `student` (`sno`))  
mysql> done go
```



```
mysql> Alter table sc drop constraint sc_ibfk_1;
```

```
Query OK, 0 rows affected (0.64 sec)
```

```
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> Update student set sno='2014003' where sno='2014002';
```

```
Query OK, 1 row affected (0.12 sec)
```

```
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> Select * from sc;
```

Sno	Cno	Grade
2014001	1	92
2014001	2	85
2014001	3	88
2014003	2	90
2014003	3	80



删除触发器

```
DROP TRIGGER [IF EXISTS]  
[schema_name.]trigger_name
```



本章小结

- 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
- RDBMS完整性实现的机制
 - 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
- 触发器的定义