完整性语言实验

16337113
劳马东
计算机科学与技术（超算方向）

一、 实验目的

1. 掌握实体完整性的定义和维护方法；
2. 掌握参照完整性的定义和维护方法。

二、 实验内容和要求

1. 定义实体完整性，删除实体完整性。能够写出两种方式定义实体完整性的 SQL 语句：创建表时定义实体完整性、创建表后定义实体完整性。设计 SQL 语句验证完整性约束是否起作用。
2. 定义参照完整性，定义参照完整性的违约处理，删除参照完整性。写出两种方式定义参照完整性的 SQL 语句：创建表时定义参完整性、创建表后定义参照完整性。

三、 实验环境

系统	Windows 10
SQL	MySQL 8.0
工具	MySQL workbench

四、 实验过程

(一) 实体完整性实验

1. 创建表时定义实体完整性（列级实体完整性）

定义供应商表实体完整性。MySQL 定义列级实体完整性定义不需要加 CONSTRAINT 关键字。

```
CREATE TABLE Supplier(
    suppkey INTEGER PRIMARY KEY,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
```

```
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101)
);
```

如下图，suppkey 被加上 PRIMARY KEY 约束后不能为 NULL，其他没有加约束默认可以为 NULL。

Column	Type	Default Value	Nullable
suppkey	int(11)		NO
name	char(25)		YES
address	varchar(40)		YES
nationkey	int(11)		YES
phone	char(15)		YES
acctbal	double		YES
comment	varchar(101)		YES

2. 创建表时定义实体完整性（表级实体完整性）

删除步骤 1 创建的表，重新以表级约束创建供应商表。表级约束是在 所有属性定义之后的，在 MySQL 中 CONSTRAINT 关键字可选，其后的 PK_supplier 是主键的别名。

```
CREATE TABLE Supplier(
    suppkey INTEGER,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101),
    CONSTRAINT PK_supplier PRIMARY KEY(suppkey)
);
```

结果如下图，同样 suppkey 属性也不能为 NULL。

Column	Type	Default Value	Nullable
suppkey	int(11)		NO
name	char(25)		YES
address	varchar(40)		YES
nationkey	int(11)		YES
phone	char(15)		YES
acctbal	double		YES
comment	varchar(101)		YES

3. 创建表后定义实体完整性

首先创建供应商表。

```
CREATE TABLE Supplier(
    suppkey INTEGER,
    name CHAR(25),
```

```
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101)
);
```

查看各个属性的情况，可以看到这时 suppkey 属性可以为 NULL，因为它还只是一个普通属性。

Column	Type	Default Value	Nullable
suppkey	int(11)		YES
name	char(25)		YES
address	varchar(40)		YES
nationkey	int(11)		YES
phone	char(15)		YES
acctbal	double		YES
comment	varchar(101)		YES

然后修改供应商表，增加实体完整性。

```
ALTER TABLE supplier
ADD CONSTRAINT PK_supplier PRIMARY KEY(suppkey);
```

此时再查看各属性的情况，suppkey 属性此时不能为 NULL，可知成功给其加上了主键约束。

Column	Type	Default Value	Nullable
suppkey	int(11)		NO
name	char(25)		YES
address	varchar(40)		YES
nationkey	int(11)		YES
phone	char(15)		YES
acctbal	double		YES
comment	varchar(101)		YES

4. 定义实体完整性（主键由多个属性组成）

定义供应商关系表，它的主键由两个属性组成。当主键由多个属性组成时，实体完整性不能定义在列级，而应该定义在表级。下面的代码省略了 CONSTRAINT 关键字。

```
CREATE TABLE PartSupp(
    partkey INTEGER,
    suppkey INTEGER,
    availqty INTEGER,
    supplycost REAL,
    comment VARCHAR(199),
    PRIMARY KEY(partkey, suppkey)
);
```

查看属性的情况，partkey 和 suppkey 都不能为 NULL，即组成主键的任何一个属性都不能为 NULL。

Column	Type	Default Value	Nullable
partkey	int(11)		NO
suppkey	int(11)		NO
availqty	int(11)		YES
supplycost	double		YES
comment	varchar(199)		YES

5. 有多个候选键时定义实体完整性

定义国家表的实体完整性，其中 nationkey 和 name 都是候选码，选择 nationkey 作为主键，name 作为候选键。由于候选键不能重复，需要在 name 上定义唯一性约束。

```
CREATE TABLE nation(
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25) UNIQUE,
    regionkey INTEGER,
    comment VARCHAR(152)
);
```

查看结果，nationkey 不能为 NULL，这是显然的。但是 name 可以为 NULL，因为任何东西和 NULL 比较的结果都是 false，也就不存在“重复”了。

Column	Type	Default Value	Nullable
nationkey	int(11)		NO
name	char(25)		YES
regionkey	int(11)		YES
comment	varchar(152)		YES

然而，除了 nationkey 之外，name 也被作为一个唯一索引，即任何候选键都是一个索引，如下图。

Visible	Key	Type	Uni	Columns
☑	PRIMARY	BTREE	YES	nationkey
☑	name	BTREE	YES	name

6. 删除实体完整性

删除国家实体的主键。在未删除之前，状态如下。nationkey 不可为 NULL，同时也是一个唯一索引。

Column	Type	Default Value	Nullable
nationkey	int(11)		NO
name	char(25)		YES
regionkey	int(11)		YES
comment	varchar(152)		YES

```
ALTER TABLE nation DROP PRIMARY KEY;
```

删除主键约束之后，nationkey 可以为 NULL，同时不再是一个索引，如下图。

Visible	Key	Type	Uni	Columns
☑	name	BTREE	YES	name

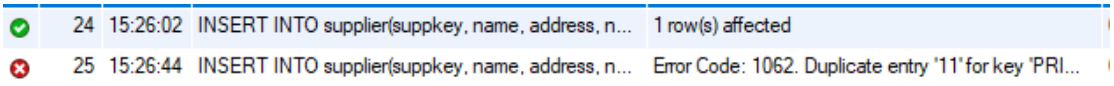
7. 增加两条相同记录，验证实体完整性是否起作用

插入两条主键相同的记录就会违反实体完整性约束，因为主键都是 UNIQUE 的。

```
INSERT INTO supplier
VALUES(11, 'test1', 'test1', 101, '12345678', 0.0, 'test1');

INSERT INTO supplier
VALUES(11, 'test2', 'test2', 102, '23456789', 0.0, 'test2');
```

如下图，第一次插入记录时是成功的，因为 supplier 表为空。第二次尝试插入一条 suppkey 也为 11 的记录时发生错误：Error Code: 1062. Duplicate entry '11' for key 'PRIMARY'。因为表中已经有一条主键为 11 的记录，而主键约束着各条记录之间在 suppkey 上不能相同，及时其他属性的值都是不同的。



(二) 参照完整性实验

1. 创建表时定义参照完整性

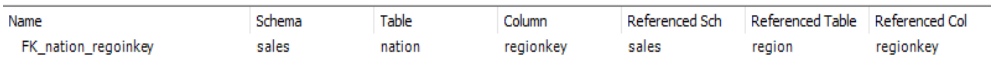
先定义地区表的实体完整性。由于被引用的属性必须要是主键，因此需要给 regionkey 加上主键约束。

```
CREATE TABLE region(
    regionkey INTEGER PRIMARY KEY,
    name CHAR(25),
    comment VARCHAR(152)
);
```

再定义国家表上的参照完整性，将 region 表的 regionkey 属性作为外键。外键约束的作用是任何出现在 nation 表中 regionkey 的值都必须在 region 表中出现。

```
CREATE TABLE nation(
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25),
    regionkey INTEGER,
    comment VARCHAR(152),
    CONSTRAINT FK_nation_regoinkey FOREIGN KEY (regionkey)
                REFERENCES region(regionkey)
);
```

查看 nation 表的外键情况，多出了一个名为 FK_nation_regionkey 的外键，这正是我们在代码中起的名字。被引用的表为 region，被引用的属性为 regionkey。



2. 创建表后定义参照完整性

大体过程和定义实体完整性相同。首先定义 nation 表。

```
CREATE TABLE nation(
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25),
    regionkey INTEGER,
    comment VARCHAR(152)
);
```

此时查看 nation 表的外键情况，发现是空的。然后给 nation 表加上外键约束。

```
ALTER TABLE nation
ADD CONSTRAINT FK_Nation_regionkey
FOREIGN KEY(regionkey) REFERENCES Region(regionkey);
```

结果如下，约束表中多出了 FK_Nation_key 外键：

Name	Schema	Table	Column	Referenced Sch	Referenced Table	Referenced Col
FK_Nation_regionkey	sales	nation	regionkey	sales	region	regionkey

3. 定义参照完整性（外键由多个属性组成）

在 MySQL 中，列级参照完整性（即在属性定义之后加上 REFERENCE 语句）不起作用，因此需要放在表级。也就是说，不管外键由几个属性组成，都需要放在表级定义。

```
CREATE TABLE PartSupp (
    partkey INTEGER,
    suppkey INTEGER,
    availqty INTEGER,
    supplycost REAL,
    comment VARCHAR(199),
    PRIMARY KEY (partkey, suppkey),
    FOREIGN KEY (partkey) REFERENCES Part (partkey),
    FOREIGN KEY (suppkey) REFERENCES Supplier (suppkey)
);
CREATE TABLE Lineitem (
    orderkey INTEGER,
    partkey INTEGER REFERENCES Part (partkey),
    suppkey INTEGER REFERENCES Supplier (suppkey),
    linenumber INTEGER,
    quantity REAL,
    extendedprice REAL,
    discount REAL,
    tax REAL,
    returnflag CHAR(1),
    linestatus CHAR(1),
    shipdate DATE,
    commitdate DATE,
```

```
    receipdate DATE,
    shipinstruct CHAR(25),
    shipmode CHAR(10),
    comment VARCHAR(44),
    PRIMARY KEY (orderkey, linenumber),
    FOREIGN KEY (partkey)
        REFERENCES Part(partkey),
    FOREIGN KEY (suppkey)
        REFERENCES Supplier(suppkey),
    FOREIGN KEY (orderkey)
        REFERENCES Orders(orderkey),
    FOREIGN KEY (partkey, suppkey)
        REFERENCES PartSupp(partkey, suppkey)
);
```

结果如下，lineitem 表多出了五个外键，分别引用 orders 表的 orderkey、part 表的 partkey、partsupp 表的 partkey 与 suppkey、supplier 表的 suppkey。

Name	Schema	Table	Column	Referenced Sch	Referenced Table	Referenced Col
lineitem_ibfk_4	sales	lineitem	orderkey	sales	orders	orderkey
lineitem_ibfk_1	sales	lineitem	partkey	sales	part	partkey
lineitem_ibfk_3	sales	lineitem	partkey	sales	partsupp	partkey
lineitem_ibfk_2	sales	lineitem	suppkey	sales	supplier	suppkey
lineitem_ibfk_3	sales	lineitem	suppkey	sales	partsupp	suppkey

4. 定义参照完整性的违约处理

定义国家表的参照完整性，当删除或修改被参照表记录时，设置参照表中相应记录的值为空值。

```
CREATE TABLE nation(
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25),
    regionkey INTEGER,
    comment VARCHAR(152),
    CONSTRAINT FK_Nation_regionkey FOREIGN KEY(regionkey)
        REFERENCES region(regionkey)
        ON DELETE SET NULL ON UPDATE SET NULL
);
```

结果如下：

Name	Schema	Table	Column	Referenced Sch	Referenced Table	Referenced Col
FK_Nation_regionkey	sales	nation	regionkey	sales	region	regionkey

在 region 和 nation 表中插入两条记录，如下：

```
INSERT INTO region VALUES(0, 'region1', 'region1');
INSERT INTO nation VALUES(0, 'nation1', 0, 'nation1');
INSERT INTO region VALUES(1, 'region2', 'region2');
INSERT INTO nation VALUES(1, 'nation2', 1, 'nation2');
```

查询 nation 表，此时两条记录如下：

nationkey	name	regionkey	comment
0	nation1	0	nation1

nationkey	name	regionkey	comment
1	nation2	1	nation2

此时删除 region 表 regionkey 为 0 的记录，再查看 nation 表，发现对应的 regionkey 变成了 NULL。

nationkey	name	regionkey	comment
0	nation1	NULL	nation1

如果修改 region 表中 regionkey 为 1 的记录的 regionkey 为 2，那么 nation 表中值为 1 的 regionkey 的假设不成立，于是也被设置为 NULL。

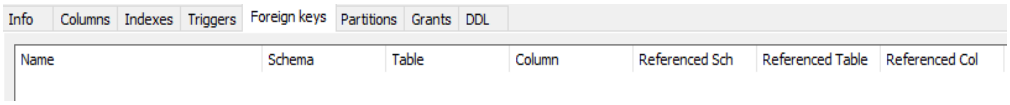
nationkey	name	regionkey	comment
1	nation2	NULL	nation2

5. 删除参照完整性

删除国家表的外键。

```
ALTER TABLE nation DROP FOREIGN KEY FK_Nation_regionkey;
```

查看 nation 表的外键表发现为空，即 FK_Nation_regionkey 已经不再是外键。



6. 插入一条国家记录，验证参照完整性是否起作用

插入一条国家记录，如果 1001 号地区记录不存在，违反参照完整性约束。

```
INSERT INTO nation VALUES (3, 'nation1', 1001, 'comment1');
```

由于 region 表中没有 regionkey 为 1001 的记录，不符合外键约束的假设，于是报错：

```
Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`sales`.`nation`, CONSTRAINT `FK_Nation_regionkey` FOREIGN KEY (`regionkey`) REFERENCES `region` (`regionkey`) ON DELETE SET NULL ON UPDATE SET NULL)
```

五、 实验总结

这次实验总体来说比较顺利，在很短时间内就完成了。一方面是因为经过前两次实验的折磨，已经清楚了 MySQL 语法和教材语法的大部分差异，也知道语法不一致时应该在什么地方找到正确语法；另一方面是因为最近在开发一个应用，需要用 MySQL 数据库来存储后台数据，经历了从建表、添加各种约束、授予权限、用程序语言操作数据库等等过程，大多数麻烦都在这个过程中遇到并解决了，因此在做这个实验的时候能迅速遇到的问题。不过，重温数据库约束的过程也给我带来了一些新的思考。

许多系统的设计都会有各种各样的约束。事务处理系统有原子性约束，一个事物的操作要么全都完成，要么全都不完成；分布式文件系统有一致性约束，一份数据的多个副本之间需要在某种程度上是相同的；在数据库系统的设计中，表内、各个表之间也需

要有一定的约束。

约束的作用，在我看来，主要有三点。

一是增强系统内部各组件之间的联系。如一致性约束将一个事物的各个操作的结果捆绑在一起；数据库主键约束使表内记录之间互相区分，外键约束使表之间的记录有某种依赖关系。

二是增强系统的可用性。试想一个转钱的事物只成功了一半，钱被扣了，但是却没有存到收账人的账户中，这样的系统是有问题的；如果给不同的学生分配了相同的 netID，但是却成功存到了数据库中，这将会带来很多麻烦。

三是减轻数据库维护人员的负担。设计人员在创建表前梳理各数据表的特征，建立正确的约束，之后维护人员对数据表的更改都将在数据库处理系统的“监督”之下，而不至于说插入了不合约束的记录，等到出现问题了让维护人员按一些文件之类的“约束”规则来查阅记录，这无疑是一件无聊且低效的事情。