**数据库实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **实验人：** | **凌国明** | | **学号：** | **21307077** | **日 期：** | **2023年10月19号** |
| **院（系）：** | **计算机学院** | | | **专业（班级）：** | **计算机科学与技术** | |
| **联系：** |  | | |  |  | |
| **实验题目：** | | **3.1 实体完整性实验** | | | | |

* + - 1. **实验目的**

掌握实体完整性的定义和维护方法。

* + - 1. **实验内容和要求**

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。

* + - 1. **实验重点和难点**

实验重点：创建表时定义实体完整性。

实验难点：有多个候选码时实体完整性的定义。

* + - 1. **实验工具**

MySQL、SQL Server、Navicat

* + - 1. **实验过程**
* 在SQL Server环境下，执行以下操作

1. 创建表时定义实体完整性（列级实体完整性）

定义供应商表的实体完整性。

CREATE TABLE Test\_Supplier1(

suppkey INTEGER CONSTRAINT PK\_supplier PRIMARY KEY,

name CHAR(25) ,

address VARCHAR(40) ,

nationkey INTEGER,

phone CHAR(15) ,

acctbal REAL,

comment VARCHAR(101)

) ;

截图如下：



2. 创建表时定义实体完整性（表级实体完整性）

定义供应商表的实体完整性。

CREATE TABLE Test\_Supplier2(

suppkey INTEGER ,

name CHAR(25) ,

address VARCHAR(40) ,

nationkey INTEGER,

phone CHAR(15) ,

acctbal REAL,

comment VARCHAR(101),

CONSTRAINT PK\_supplier2 PRIMARY KEY (suppkey)

) ;

截图如下：

3. 创建表后定义实体完整性

定义供应商表的实体完整性。

CREATE TABLE Test\_Supplier3(

suppkey INTEGER not NULL,

name CHAR(25) ,

address VARCHAR(40) ,

nationkey INTEGER,

phone CHAR(15) ,

acctbal REAL,

comment VARCHAR(101) );

ALTER TABLE Test\_Supplier3 /\*再修改供应商表,增加实体完整性\*/

ADD CONSTRAINT PK\_Supplir3 PRIMARY KEY(suppkey);

在SQL Server中，只能将NOT NULL的列设置为主码，所以在创建表时suppkey列不允许空值。

截图如下：



4. 定义实体完整性（主码由多个属性组成）

定义供应关系表的实体完整性。主码由partkey和suppkey组成，实体完整性需定义在表级。

CREATE TABLE Test\_PartSupp(

partkey INTEGER,

suppkey INTEGER,

availqty INTEGER,

supplycost REAL,

comment VARCHAR(199) ,

PRIMARY KEY(partkey, suppkey) );

/\*主码由多个属性组成,实体完整性必须定义在表级\*/

截图如下：



5. 有多个候选码时定义实体完整性

定义国家表的实体完整性，其中nationkey和name都是候选码，选择nationkey作为主码，name上定义唯一性约束。

CREATE TABLE Test\_nation(

nationkey INTEGER CONSTRAINT PK\_nation PRIMARY KEY,

name CHAR(25) UNIQUE,

regionkey INTEGER,

comment VARCHAR(152) ) ;

截图如下：



6. 删除实体完整性

删除国家实体的主码。

ALTER TABLE Test\_nation DROP CONSTRAINT PK\_nation;

截图如下：



7. 增加两条相同记录，验证实体完整性是否起作用。

/\*插入两条主码相同的记录就会违反实体完整性约束\*/

插入第一条记录，主码值为11。

INSERT INTO Test\_Supplier1(suppkey, name, address, nationkey, phone, acctbal, comment)

VALUES(11, 'test 1', 'test 1', 101, '12345678', 0.0, 'test 1');

插入结果如下：



插入两条主码相同的记录会违反实体完整性约束，接下来插入第二条记录，主码值为11。

INSERT INTO Test\_Supplier1(suppkey, name, address, nationkey, phone, acctbal, comment)

VALUES(11, 'test 2', 'test 2', 102, '23456789', 0.0, 'test 2');

违反实体完整性约束，拒绝执行。

截图如下：





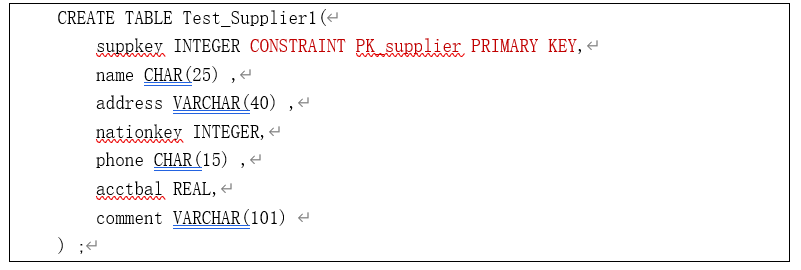
* + - 1. **与实验结果相关的文件**

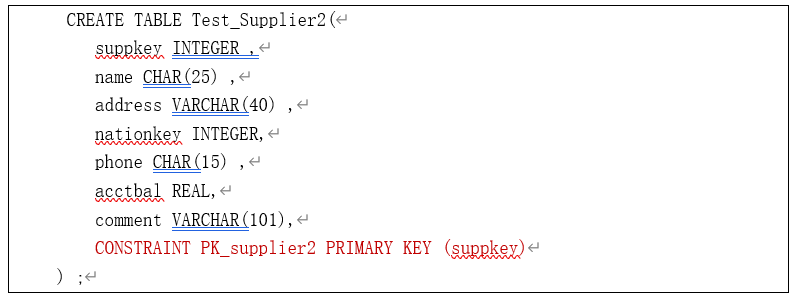
无

* + - 1. **思考题**

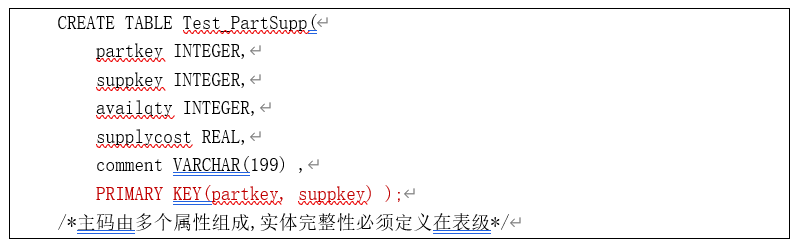
（1）所有列级完整性约束都可以改写为表级完整性约束，而表级完整性约束不一定能改写成列级完整性约束。请举例说明。

答：列级完整性约束可以改为表级完整性约束；只涉及一个列的表级完整性约束可以改为列级完整性约束。以下例子中，两种完整性约束是等效的。



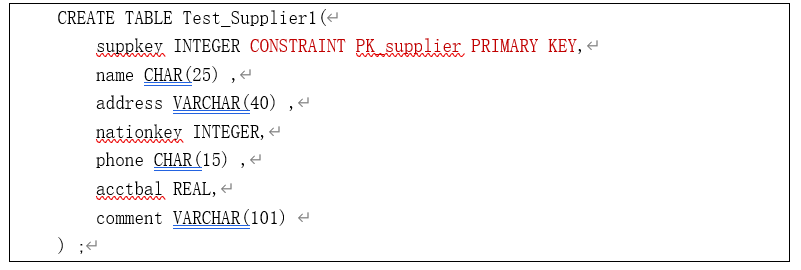


当表级完整性约束涉及多个列时，不可以改写成列级完整性约束。这是因为表级完整性约束可以涵盖整个表的数据，而列级完整性约束只能涵盖特定的列

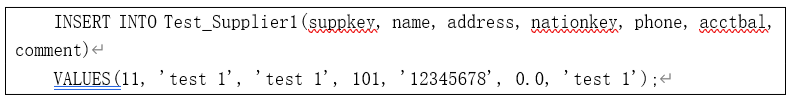


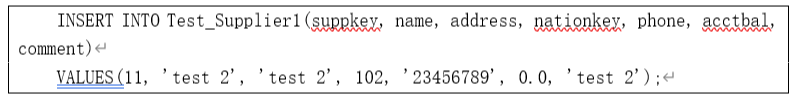
（2）什么情况下会违反实体完整性约束，DBMS将做何种违约处理？请用实验验证。

答：当DBMS发现插入、更新或删除操作导致违反实体完整性约束时，它会采取相应的违约处理操作。如果插入、更新或删除操作导致违反某个完整性约束，DBMS 会阻止该操作，并引发错误或异常。以下是一个例子



设置supplykey为主键，这是一个主键约束。





现进行两个插入操作，插入的supplykey都是一样的，这违反了主键约束



DBMS 阻止了第二次插入操作，并报出错误信息。

* + - 1. **实验总结**

在实验中，我学会了如何创建表时定义实体完整性约束，通过在CREATE TABLE语句中添加CONSTRAINT关键字来定义主键约束（PRIMARY），以及如何定义其他约束，如NOT NULL约束和CHECK约束。此外，我还学会了如何创建表后定义实体完整性约束，通过使用ALTER TABLE语句来添加或删除约束。

实验中的重点是创建表时定义实体完整性，这是因为在表创建阶段定义约束可以更好地保证数据的完整性。然而，当涉及到多个候选码时，定义实体完整性可能会变得复杂，需要仔细考虑每个候选码的作用和限制条件，以确保数据的一致性和完整性。

思考题中提到了列级完整性约束和表级完整性约束的关系，以及实体完整性约束违约处理的问题。列级完整性约束可以改写为表级完整性约束，只涉及一个列的表级完整性约束可以改写为列级完整性约束，这取决于约束的复杂性和约束的作用范围。实验中的例子展示了这一点。

最后，我还通过实验验证了实体完整性约束的违约处理，当插入、更新或删除操作导致违反实体完整性约束时，DBMS会阻止该操作并引发错误或异常，确保数据的完整性得到维护。

总的来说，这次实验让我更深入地了解了实体完整性的概念和维护方法，以及如何在数据库中定义和验证实体完整性约束。这对于数据库管理和数据的保护至关重要。