

《数据库系统原理》课程实验指导

OceanBase 完整性约束



2023年10月

目录

| | |
|---|----|
| 前 言 | 2 |
| 实验环境说明 | 2 |
| 1 完整性约束实验 | 3 |
| 1.1 实验目的 | 3 |
| 1.2 实验内容 | 3 |
| 1.3 实验要求 | 3 |
| 1.4 实验总结 | 3 |
| 2 实验示例 | 4 |
| 2.1 利用 Create table/Alter table 语句建立完整性约束 | 4 |
| 2.2 主键/候选键/空值/check/默认值约束验证 | 6 |
| 2.2.1 主键/候选键约束 | 6 |
| 2.2.2 空值 | 8 |
| 2.3 外键/参照完整性约束验证 | 9 |
| 2.3.1 参照完整性约束验证 | 9 |
| 2.3.2 级联/非级联外键关联下数据访问 | 11 |
| 2.4 函数依赖分析验证 | 14 |
| 2.5 触发器约束 | 15 |

前 言

实验环境说明

本实验环境为任意受支持 Linux 系统上的 OceanBase v4.2.1 数据库，实验数据采用电商数据库的八张表。

1 完整性约束实验

1.1 实验目的

了解 SQL 语言和 OceanBase 数据库提供的完整性（integrity）机制，通过实验掌握面向实际数据库建立实体完整性、参照完整性、断言、函数依赖等各种完整性约束的方法，验证各类完整性保障措施。

1.2 实验内容

在前面完成的实验中已建立了本实验所需的 8 张表。本实验将针对这 8 张表，采用 create table、alter table 等语句，添加主键、候选键、外键、check 约束、默认/缺省值约束，并观察当用户对数据库进行增、删、改操作时，DBMS 如何维护完整性约束。

1. 建立完整性约束
2. 主键/候选键/空值/check/默认值约束验证
3. 外键/参照完整性验证分析
4. 函数依赖
5. 触发器

1.3 实验要求

1. 罗列的实验内容比较多，不必都做。类似实验内容选做有代表性的，例如，
 - 1) 主键验证、候选键验证只做一个；
 - 2) 在一个实验中同时验证空值、默认值、主键、check 等约束；
 - 3) 级联、非级联外键约束实验二选一

参照下面所给示例，选择电商数据库中不同的关系表，完成各个实验内容。

1.4 实验总结

在实验中有哪重要问题或者事件？你如何处理的？你的收获是什么？有何建议和意见等等。

2 实验示例

2.1 利用 Create table/Alter table 语句建立完整性约束

2.1.1.1 实验要求

选择电商数据库中的一张表，如订单明细表 LINEITEM，分析识别该表上所有约束；采用 create table 语句，建立该表的副本 LINEITEMcopy，将数据导入 LINEITEMcopy，后续实验可在上进行。

注意：后面的实验会使数据发生改动，可以在完成每项实验后，将数据发生改动的表清空，然后重新导入数据，从而使数据恢复原样，避免发生不必要的麻烦

| 属性名称 | 字段中文名称 | 数据类型 | 数据取值范围，完整性/约束说明 |
|-----------------|---------|-------------------|-----------------|
| L_ORDERKEY | 订单 key | INTEGER | NOT NULL 主键 外键 |
| L_PARTKEY | 零件 key | INTEGER | NOT NULL 外键 |
| L_SUPPKEY | 供应商 key | INTEGER | NOT NULL 外键 |
| L_LINENUMBER | 流水号 | INTEGER | NOT NULL 主键 |
| L_QUANTITY | 数量 | DECIMAL (15,2) | NOT NULL |
| L_EXTENDEDPRICE | 价格 | DECIMAL (15,2) | NOT NULL |
| L_DISCOUNT | 折扣 | DECIMAL (15,2) | NOT NULL |
| L_TAX | 税 | DECIMAL (15,2) | NOT NULL |
| L_RETURNFLAG | 退货标志 | CHAR(1) | NOT NULL |
| L_LINESTATUS | 明细状态 | CHAR(1) | NOT NULL |
| L_SHIPDATE | 发货日期 | DATE | NOT NULL |
| L_COMMITDATE | 预计到达日期 | DATE | NOT NULL |
| L_RECEIPTDATE | 实际到达日期 | DATE | NOT NULL |

| | | | |
|--------------------|--------|-----------------|----------|
| L_SHIPINSTRU CT | 运单处理策略 | CHAR(25) | NOT NULL |
| L_SHIPMODE | 运送方式 | CHAR(10) | NOT NULL |
| L_COMMENT | 备注 | VARCHAR (44) | NOT NULL |

2.1.1.2 实验过程

步骤 1. 使用 create table 在该表相关属性上，添加主键、非空。示例：

```
CREATE TABLE LINEITEMcopy1(
  L_ORDERKEY integer NOT NULL,
  L_PARTKEY integer NOT NULL,
  L_SUPPKEY integer NOT NULL,
  L_LINENUMBER integer NOT NULL,
  L_QUANTITY DECIMAL(15,2) NOT NULL,
  L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
  L_DISCOUNT DECIMAL(15,2) NOT NULL,
  L_TAX DECIMAL(15,2) NOT NULL,
  L_RETURNFLAG CHAR(1) NOT NULL,
  L_LINESTATUS CHAR(1) NOT NULL,
  L_SHIPDATE DATE NOT NULL,
  L_COMMITDATE DATE NOT NULL,
  L_RECEIPTDATE DATE NOT NULL,
  L_SHIPINSTRUCT CHAR(25) NOT NULL,
  L_SHIPMODE CHAR(10) NOT NULL,
  L_COMMENT VARCHAR(44) NOT NULL,
  PRIMARY KEY (L_ORDERKEY, L_LINENUMBER),
  FOREIGN KEY (L_PARTKEY) REFERENCES PART(P_PARTKEY),
  FOREIGN KEY (L_SUPPKEY) REFERENCES SUPPLIER(S_SUPPKEY)
);
```

创建成功

步骤 2. 创建一个不带约束的关系表，使用 alter table 语句，在该表上添加约束。

```
CREATE TABLE LINEITEMcopy2(
  L_ORDERKEY integer NOT NULL,
  L_PARTKEY integer NOT NULL,
  L_SUPPKEY integer NOT NULL,
  L_LINENUMBER integer NOT NULL,
  L_QUANTITY DECIMAL(15,2) NOT NULL,
  L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
  L_DISCOUNT DECIMAL(15,2) NOT NULL,
  L_TAX DECIMAL(15,2) NOT NULL,
  L_RETURNFLAG CHAR(1) NOT NULL,
  L_LINESTATUS CHAR(1) NOT NULL,
  L_SHIPDATE DATE NOT NULL,
  L_COMMITDATE DATE NOT NULL,
  L_RECEIPTDATE DATE NOT NULL,
  L_SHIPINSTRUCT CHAR(25) NOT NULL,
```

```
L_SHIPMODE CHAR(10) NOT NULL,  
L_COMMENT VARCHAR(44) NOT NULL  
);
```

新建主键，如果表中没有建立主键，利用下面的语句添加主键

```
alter table 表名  
add constraint 约束名  
primary key (字段名)
```

说明：字段名为要在其上创建主键的字段名

示例:

```
alter table LINEITEMcopy2 add constraint LINEITEMcopy2_PK primary key (L_ORDERKEY, L_LINENUMBER);
```

候选键，如果表中没有建立候选键，利用下面的语句添加候选键

```
alter table 表名  
add constraint 约束名  
unique (字段名)
```

示例:

```
alter table LINEITEMcopy2 add constraint LINEITEMcopy2_UK unique (L_ORDERKEY, L_LINENUMBER,  
L_PARTKEY);
```

非空约束，利用下面的语句添加非空约束

```
alter table 表名  
alter column 字段名  
set not null
```

示例:

```
alter table LINEITEMcopy2 modify L_EXTENDEDPRICE DECIMAL(15,2) not null;
```

将 LINEITEM 表的数据复制到 LINEITEMcopy 表中，保证 LINEITEM 表与 LINEITEMcopy 表内容一致。

```
insert into lineitemcopy1  
select *  
from lineitem;
```

2.2 主键/候选键/空值/check/默认值约束验证

2.2.1 主键/候选键约束

2.2.1.1 实验要求

对于主键约束，选取定义了主键的关系表，如 lineitemcopy1，

(1) 使用分组聚集运算语句，判断是否满足主键约束

(2) 向该表插入在主属性上取值为空的元组，观察 DBMS 反应；

(3) 选取表中某些或某个元组，修改这些元组在主属性上的取值，或向表中插入新元组，使这些元组与表中已有其它元组的主属性取值相同，或者将选定的元组在主属性上的取值修改为 null，观察 DBMS 反应；

对于候选键约束

(1) 选取定义了候选键的关系表，如 lineitemcopy1，使用分组聚集运算语句，判断是否满足候选键约束

(2) 向该表插入在候选键属性上取值为空的元组，观察 DBMS 的反应；

(3) 选取表中某些或某个元组，修改这些元组在候选键属性上的取值，或插入新元组，使这些元组与表中已有其它元组的候选键属性取值相同，或者将选定的元组在候选键属性上的取值修改为 null，观察系统反应；

2.2.1.2 实验过程

这里以主键约束验证为例，选取上面实验创建并导入数据的表 lineitemcopy1。

使用分组聚集运算语句，判断是否满足主键约束，可以看出没有重复主键的数据行。

执行如下 SQL 语句：

```
Select l_orderkey, count(*)  
From lineitemcopy1  
Group by l_orderkey, l_linenum  
Having count(*)>1;
```

结果返回 0 行

表中并无重复主键的数据。

判断是否有主键为空的数据

```
select *  
from lineitemcopy1  
where l_orderkey is null  
and l_linenum is null;
```

结果返回 0 行

表中并无主键为空的数据。

插入主键为空数据，报错：

比如，插入值为(null,0,0,null,0,0,0,0,'a','b','2020-01-01',
'2020-01-12', '2020-01-15','name3', 'name4', 'name5');


```
INSERT INTO lineitemcopy1
values(null,0,0,null,0,0,0,0, 'a', 'b', '2020-01-01',
'2020-01-12', '2020-01-15','name3', 'name4', 'name5');
```

修改原有数据行 l_orderkey, l_linenumbr 字段为空，报错：

比如，将 l_orderkey=1, l_linenumbr=5 的数据行的 l_orderkey 和 l_linenumbr 修改为空

```
UPDATE lineitemcopy1
SET l_orderkey=null, l_linenumbr=null
WHERE l_orderkey =1 and l_linenumbr=5;
```

修改原有数据行 l_orderkey 字段和 l_linenumbr 字段与表中已有其它元组的主属性取值相同，报错：

比如，更新表中 l_orderkey=1, l_linenumbr=1 的数据行，将其 l_linenumbr 字段值改为 2，可以看到由于表中已经存在 l_orderkey=1, l_linenumbr=2 的数据行，l_orderkey, l_linenumbr 作为主键，不允许重复值，因此执行失败。

```
UPDATE lineitemcopy1
SET l_linenumbr =2
WHERE l_orderkey=1 and l_linenumbr=1;
```

同样地，插入主键重复的数据，报错：

比如，插入值为(1,0,0,2,0,0,0,0,'a','b','2020-01-01',
'2020-01-01', '2020-01-01','name3', 'name4', 'name5');

```
INSERT INTO lineitemcopy1
values(1,0,0,2,0,0,0,0, 'a', 'b', '2020-01-01',
'2020-01-01', '2020-01-01','name3', 'name4', 'name5');
```

候选键约束与上述类似。

2.2.2 空值

2.2.2.1 实验要求

选取定义了 not null 属性约束的关系表，如 lineitemcopy1 及其属性 l_extendedprice，观察（1）向表中插入新元组，或（2）修改表中已有元组时，如果导致该属性上取值为空，DBMS 的反应和处理方式。

2.2.2.2 实验过程

插入一数据行，其 l_extendedprice 字段为空，报错：

比如，插入值为(1,0,0,2,0,null,0,0,'a','b','2020-01-01', '2020-01-01', '2020-01-01','name3', 'name4', 'name5');

```
INSERT INTO lineitemcopy1
```

```
values(1,0,0,2,0,null,0,0, '1', '2', '2020-01-01',
'2020-01-01', '2020-01-01','name3', 'name4', 'name5');
```

修改原有数据行 l_extendedprice 字段为空，报错：

比如，将 l_orderkey=2 and l_linenum=1 的数据行的 l_extendedprice 修改为空

```
UPDATE lineitemcopy1
SET l_extendedprice=null
WHERE l_orderkey=2 and l_linenum=1;
```

2.3 外键/参照完整性约束验证

2.3.1 参照完整性约束验证

2.3.1.1 实验要求

表 1 可能在外键关联的表

| 参照关系 r1 | | 被参照关系 r2 | |
|----------|---|--------------------|---|
| 表 | 外键属性 | 表 | 主键 |
| ORDERS | O_CUSTKEY | CUSTOMER | C_CUSTKEY |
| SUPPLIER | S_NATIONKEY | NATION | N_NATIONKEY |
| NATION | N_REGIONKEY | REGION | R_REGIONKEY |
| PARTSUPP | PS_PARTKEY 、 PS_SUPPKEY | PART 、 SUPPLIER | P_PARTKEY 、 S_SUPPKEY |
| CUSTOMER | C_NATIONKEY | NATION | N_NATIONKEY |
| LINEITEM | L_ORDERKEY、 L_PARTKEY 、 L_SUPPKEY | ORDERS 、 PART | O_ORDERKEY、 P_PARTKEY 、 S_SUPPKEY |

实验步骤

步骤 1：判断参照完整性约束是否满足

从表 1 中选定一组表 r_1 和 r_2 ，编写 SQL 语句，判断两表间是否满足参照完整性约束。例如， $r_1 = \text{ORDERS}$ ， $r_2 = \text{CUSTOMER}$ ，判断 ORDERS 在属性 O_CUSTKEY 上的取值是否都出现在 ORDERS 表的 C_CUSTKEY 列中。

步骤 2：改造参照关系表，满足完整性要求

如果两张表间不满足参照完整性约束。则使用 delete 语句，去除参照关系表中相关元组，使得两表间参照完整性约束关系成立。

步骤 3：在改造后的参照表 r_1 和被参照 r_2 上，建立非级联外键关联。

2.3.1.2 实验过程

为方便起见，分别创建关系表 orders 和 customer 的副本 orderscopy 和 customercopy，并将数据导入进去

```
CREATE TABLE customercopy1(  
  c_custkey integer,  
  c_name varchar(25),  
  c_address varchar(40),  
  c_nationkey integer,  
  c_phone char(15),  
  c_acctbal decimal(15,2),  
  c_mktsegment char(10),  
  c_comment varchar(117),  
  PRIMARY KEY (c_custkey),  
  FOREIGN KEY (c_nationkey) REFERENCES nation(n_nationkey)  
);  
INSERT INTO customercopy1  
SELECT *  
FROM customer;
```

```
CREATE TABLE orderscopy1(  
  o_orderkey integer,  
  o_custkey integer,  
  o_orderstatus char(1),  
  o_totalprice decimal(15,2),  
  o_orderdate date,  
  o_orderpriority char(15),  
  o_clerk char(15),  
  o_shippriority integer,  
  o_comment varchar(79),  
  PRIMARY KEY (o_orderkey),  
  FOREIGN KEY (o_custkey) REFERENCES customercopy1(c_custkey)  
);  
INSERT INTO orderscopy1  
SELECT *
```

```
FROM orders;
```

判断两表间是否满足参照完整性约束

```
select count(O_CUSTKEY)
from orderscopy1
where O_CUSTKEY not in (
select C_CUSTKEY
from customercopy1
);
```

若结果不为 0，则不满足参照完整性约束

若结果为 0，则两表已经满足参照完整性约束。

(由于数据量较大，执行时间可能较长)

定义 orderscopy 和 customercopy 之间的级联关联如下：

```
alter table orderscopy1
add constraint FK_O_CUSTKEY
foreign key(O_CUSTKEY) references customercopy1(C_CUSTKEY)
on delete cascade
on update cascade;
```

或者定义 orderscopy 和 customercopy 之间的非级联关联如下：

```
alter table orderscopy1
add constraint FK_O_CUSTKEY
foreign key(O_CUSTKEY) references customercopy1(C_CUSTKEY);
```

定义成功

2.3.2 级联/非级联外键关联下数据访问

2.3.1.3 实验要求

非级联：

选取相互间定义了非级联外键关联的一组表 r1 和 r2，分别在参照关系 r1、被参照关系 r2 上，对表的主属性/外键属性作插入 insert、删除 delete、更新 update 操作，观察当其中 1 个表（如参照关系表 r1、被参照关系表 r2）在外键属性或主属性上的取值发生变化时，DBMS 对这些操作的反应，以及另外一个表（如被参照关系表、参照关系表）在主属性或外键属性上的取值的变化，并记录实验结果。

上述插入、删除、更新操作操作分为违反约束和不违反约束两种情况。

级联：

步骤 1: 使用 Alter table 中的 Drop constraint 参数, 删除前面定义的参照关系 r1 和被参照关系 r2 间的非级联关联, 再重新定义级联关联:

```
alter table orderscopy1
drop constraint FK_O_CUSTKEY;
```

步骤 2: 分别在参照关系 r1、被参照关系 r2 上, 对表的主属性/外键属性作插入 insert、删除 delete、更新 update 操作, 观察当其中 1 个表 (如参照关系表 r1、被参照关系表 r2) 在外键属性或主属性上的取值发生变化时, DBMS 对这些操作的反应, 以及另外一个表 (如被参照关系表、参照关系表) 在主属性或外键属性上的取值的变化, 并记录实验结果。

上述插入、删除、更新操作操作分为违反约束和不违反约束两种情况。

2.3.1.4 实验过程

建立好外键关联后, 向 orderscopy 表中插入一行数据, 其 O_CUSTKEY 值设为 0, 由于 customercopy 表中不存在 C_CUSTKEY 值为 0 的数据行, 违反了外键约束, 因此插入失败:

```
insert into orderscopy1
values(1200001,0,'O',181580,'2019-01-02','5-LOW','Clerk#000000406',0,'special f');
```

再向 orderscopy 表中插入一行数据, 其 O_CUSTKEY 值设为 25519, 由于 customercopy 表中存在 C_CUSTKEY 值为 25519 的数据行, 不违反外键约束, 因此插入成功:

```
insert into orderscopy1
values(1200002,25519,'O',181580,'2019-01-02','5-LOW','Clerk#000000406',0,'special f');
```

向 customercopy 中插入一行 C_CUSTKEY 为 30001 的数据, 而 customercopy 中不存在 C_CUSTKEY 值为 30001 的数据, 插入成功:

```
INSERT INTO customercopy1
values(30001,'Customer#000030001','a',0,'10-396-325-3144',100,'b','x')
```

将 orderscopy 表中一行 O_CUSTKEY 值为 7828 的数据的 O_CUSTKEY 字段值修改为 31000, 而 customercopy 表中并没有 C_CUSTKEY 值为 31000 的数据行, 因此违反了外键约束, 更新失败:

```
UPDATE orderscopy1
SET O_CUSTKEY=31000
WHERE O_CUSTKEY=7828;
```

将 orderscopy 表中一行 O_CUSTKEY 值为 8897 的数据的 O_CUSTKEY 字段值修改为 29980, customercopy 表中已有 C_CUSTKEY 值为 29980 的数据行, 不违反外键约束, 因此执行成功:

```
UPDATE orderscopy1
SET O_CUSTKEY=29980
WHERE O_CUSTKEY =8897;
```

将 customercopy 表中一行 C_CUSTKEY 值为 12 的数据的 C_CUSTKEY 字段值修改为 31001，表 orderscopy 中不存在 O_CUSTKEY 值为 12 的数据行,不违反外键约束，执行成功：

```
UPDATE customercopy1
SET C_CUSTKEY=31001
WHERE C_CUSTKEY=12;
```

删除 orderscopy 表中 O_CUSTKEY 字段值为 17768 的数据行，执行成功：

```
delete from orderscopy1
where O_CUSTKEY=17768;
```

从 customercopy 表中删除 C_CUSTKEY 值为 31001 的数据行，表 orderscopy 中不存在 C_CUSTKEY 值为 31001 的数据行，不违反外键约束，执行成功：

```
delete from customercopy1
where C_CUSTKEY=31001;
```

以上步骤在级联外键关联和非级联外键关联下都一样，没有本质区别。

级联/非级联区别

查看 orderscopy 表中 O_CUSTKEY=8890 的数据项

```
select *
from orderscopy1
where O_CUSTKEY=8890;
```

共有 24 条

非级联外键关联：

将 customercopy 表中一行 C_CUSTKEY 值为 8890 的数据的 C_CUSTKEY 字段值修改为 30005，表 orderscopy 中存在 O_CUSTKEY 值为 8890 的数据行,因此违反了外键约束，执行失败：

```
UPDATE customercopy1
SET C_CUSTKEY=30005
WHERE C_CUSTKEY=8890;
```

从 customercopy 表中删除 C_CUSTKEY 值为 8890 的数据行，表 orderscopy 中存在 O_CUSTKEY 值为 8890 的数据行，因此违反了外键约束，执行失败：

```
delete from customercopy1
where C_CUSTKEY =8890;
```

级联外键关联：

将 customercopy 表中一行 C_CUSTKEY 值为 8890 的数据的 C_CUSTKEY 字段值修改为 30005，表 orderscopy 中存在 O_CUSTKEY 值为 8890 的数据行,虽然违反了外键约束，但执行成功：

```
UPDATE customercopy1
SET C_CUSTKEY=30005
WHERE C_CUSTKEY=8890;
```

因为 orderscopy 表中 O_CUSTKEY=8890 的数据项的 N_SECTOR_ID 的值会跟着改成 30005

```
select *
from orderscopy1
where O_CUSTKEY=8890;
```

查询结果为 0 行

```
select *
from orderscopy1
where O_CUSTKEY =30005;
```

查询结果为 24 行

从 customercopy 表中删除 C_CUSTKEY 值为 30005 的数据行，表 orderscopy 中存在 O_CUSTKEY 值为 30005 的数据行，虽然违反了外键约束，但执行成功：

```
delete from customercopy1
where C_CUSTKEY=30005;
```

因为 orderscopy 表中 O_CUSTKEY=30005 的数据项会跟着被删除

```
select *
from orderscopy1
where O_CUSTKEY=30005;
```

查询结果为 0 行

非级联外键关联下的操作，只要违法了参照完整性约束，便无法执行。而级联外键关联下，当被参照关系中的主键发生修改，删除时，参照关系中的外键会跟着进行相应地修改，删除。

2.4 函数依赖分析验证

2.4.1.1 实验要求

函数依赖反映了关系表中属性间的依赖关系。主键、候选键、外键约束都属于函数依赖，对于这三类函数依赖的验证参见上面的实验。下面考虑验证非主属性间的函数依赖关系。

在零部件表 PART 中，一种零件品牌只能由一个零件厂商生产，因此零件品牌与零件厂商间存在函数依赖：

$P_BRAND \rightarrow P_MFGR$

要求：

(1) 用 SQL 语句判断 P_BRAND 与 P_MFGR 间是否存在函数依赖关系。

(2) 如果 P_BRAND 与 P_MFGR 间函数依赖不存在，用 SQL 语句找出导致该函数依赖不存在的元组。

E.g.

```
select P_BRAND, T1. P_MFGR, T2. P_MFGR
  from part as T1, part as T2
  where T1. P_BRAND = T2. P_BRAND
 and T1. P_MFGR <> T2. P_MFGR
```

2.4.1.2 实验过程

步骤 1：判断函数依赖 $P_BRAND \rightarrow P_MFGR$ 是否满足。

首先对 P_BRAND 进行分组，对于有相同 P_BRAND 的元组，统计去重之后的零件厂商数量，对于每组结果，只要不同的零件厂商数量大于 1，则说明相同的 P_BRAND 对应不同的 P_MFGR ，不满足函数依赖，若所有分组结果的最大值都为 1，则满足函数依赖。

```
select max(a) as a_MFGR
  from(
select count(DISTINCT P_MFGR) as a
  from part
  group by P_BRAND
);
```

结果满足依赖。

步骤 2：对于其他不满足依赖关系的例子，可以尝试找出导致该函数依赖不存在的元组。

2.5 触发器约束

2.5.1.1 实验要求

实验 1：开发一个数据插入查重触发器，实现：

向一张表中插入一行新数据时，如果新数据的主键与表中已有其它元组的主键不相同，则直接插入；如果新数据的主键与表中已有元组的主键相同，则根据新插入元组的属性值修改已有元组的属性值，或者：先删除主键相同的已有元组，再插入新元组。

实验 2：开发一个日期校对触发器，实现：

当向订单明细表 LINEITEM 中插入一行，或者修改现有订单的发货日期时，判断新插入的、或修改后的发货日期是否合法，即发货日期必须在预计到达日期和实际到达日期之前。如果不合法，回滚。

实验 3：开发一个触发器，实现：

当客户账户余额小于 50 时，不允许向订单表中插入来自该客户的新订单。

2.5.1.2 实验过程

以实验二为示例。

由于触发器中禁止增删改操作的嵌套使用，因此为了完成实验需求，再对 lineitemcopy 表进行一个备份，新表为 lineitemcopy_new。为了验证触发器正确性，删除新表上的相关约束。在实际应用中需保持两表的数据一致性，本实验仅验证触发器效果。

```
CREATE TABLE lineitemcopy1_new(  
  L_ORDERKEY integer NOT NULL,  
  L_PARTKEY integer NOT NULL,  
  L_SUPPKEY integer NOT NULL,  
  L_LINENUMBER integer NOT NULL,  
  L_QUANTITY DECIMAL(15,2) NOT NULL,  
  L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,  
  L_DISCOUNT DECIMAL(15,2) NOT NULL,  
  L_TAX DECIMAL(15,2) NOT NULL,  
  L_RETURNFLAG CHAR(1) NOT NULL,  
  L_LINESTATUS CHAR(1) NOT NULL,  
  L_SHIPDATE DATE NOT NULL,  
  L_COMMITDATE DATE NOT NULL,  
  L_RECEIPTDATE DATE NOT NULL,  
  L_SHIPINSTRUCT CHAR(25) NOT NULL,  
  L_SHIPMODE CHAR(10) NOT NULL,  
  L_COMMENT VARCHAR(44) NOT NULL  
);  
insert into lineitemcopy1_new  
select *  
from lineitemcopy1;
```

创建触发器函数，将新数据插入到 lineitemcopy 中。在 lineitemcopy_new 上定义插入触发器，如果发货日期满足插入条件，则插入到 lineitemcopy 中，若不满足条件，则不进行插入操作。

```
DELIMITER //  
  
CREATE TRIGGER insert_trig_before BEFORE INSERT ON lineitemcopy1_new  
FOR EACH ROW  
BEGIN  
  IF (NEW.l_shipdate <= NEW.l_commitdate AND NEW.l_shipdate <= NEW.l_receiptdate) THEN  
    INSERT INTO lineitemcopy1 (  
      L_ORDERKEY, L_PARTKEY, L_SUPPKEY, L_LINENUMBER, L_QUANTITY, L_EXTENDEDPRICE,
```

```

        L_DISCOUNT, L_TAX, L_RETURNFLAG, L_LINESTATUS, L_SHIPDATE, L_COMMITDATE,
        L_RECEIPTDATE, L_SHIPINSTRUCT, L_SHIPMODE, L_COMMENT
    ) VALUES (
        NEW.L_ORDERKEY, NEW.L_PARTKEY, NEW.L_SUPPKEY, NEW.L_LINENUMBER,
NEW.L_QUANTITY,
        NEW.L_EXTENDEDPRICE, NEW.L_DISCOUNT, NEW.L_TAX, NEW.L_RETURNFLAG,
NEW.L_LINESTATUS,
        NEW.L_SHIPDATE, NEW.L_COMMITDATE, NEW.L_RECEIPTDATE, NEW.L_SHIPINSTRUCT,
NEW.L_SHIPMODE,
        NEW.L_COMMENT
    );
END IF;
END //

DELIMITER ;

```

通过 show triggers 命令查看创建的触发器函数

通过 information_schema.TRIGGERS 获取触发器定义信息。其中，`your_trigger_name`应该替换为实际触发器名。

```

SELECT TRIGGER_SCHEMA, TRIGGER_NAME, EVENT_MANIPULATION, EVENT_OBJECT_TABLE,
ACTION_STATEMENT, ACTION_TIMING
FROM information_schema.TRIGGERS
WHERE TRIGGER_NAME = 'your_trigger_name';

```

向 lineitemcopy_new 插入一行数据，发货日期为 2020 年 1 月 2 日，预计到达日期和实际到达日期均为 2020 年 1 月 1 日，不满足约束条件。

```

INSERT INTO lineitemcopy1_new
values(4,0,0,2,0,0,0,0, 'a', 'b', '2020-01-02',
'2020-01-01', '2020-01-01','name3', 'name4', 'name5');

```

查看订单 key 为 4，流水号为 2 的数据行，发现数据插入到了 lineitemcopy_new 表中

但没有插入进 lineitemcopy 表

```

select *
from lineitemcopy1_new(lineitemcopy1)
where l_orderkey=4 and l_linenumber=2;

```

插入一行发货日期为 2020 年 1 月 2 日，预计到达日期为 2020 年 1 月 3 日，实际到达日期均为 2020 年 1 月 4 日的数据，满足约束，插入成功。

```

INSERT INTO lineitemcopy1_new
values(4,1,1,3,0,0,0,0, 'a', 'b', '2020-01-02',
'2020-01-03', '2020-01-04','name3', 'name4', 'name5');

```

查看订单 key 为 4，流水号为 3 的数据行，在 lineitemcopy_new 中查看刚才插入的数据，发现插入成功。

在 lineitemcopy 中查看刚才插入的数据，发现插入也成功了。

创建触发器函数，将新数据插入到 lineitemcopy 中。在 lineitemcopy_new 上定义插入触发器，如果发货日期满足插入条件，则插入到 lineitemcopy 中，若不满足条件，则不进行插入操作。

```
DELIMITER //

CREATE TRIGGER updata_trig_before BEFORE UPDATE ON lineitemcopy1_new
FOR EACH ROW
BEGIN
    IF (NEW.l_shipdate <= NEW.l_commitdate AND NEW.l_shipdate <= NEW.l_receiptdate) THEN
        UPDATE lineitemcopy1
        SET L_SHIPDATE = NEW.L_SHIPDATE
        WHERE L_ORDERKEY = NEW.L_ORDERKEY AND L_LINENUMBER = NEW.L_LINENUMBER;
    END IF;
END;
//
```

DELIMITER ;

通过与上面相同的命令查看创建的触发器函数和获取触发器定义信息。

将刚才插入的订单 key 为 4，流水号为 3 的数据进行修改，发货日期新值为 2019-02-01，满足约束条件。

```
UPDATE lineitemcopy1_new
SET l_shipdate='2019-02-01'
WHERE l_orderkey=4 and l_linenumber=3;
```

查看订单 key 为 4，流水号为 3 的数据行，在 lineitemcopy_new 中查看更新的数据，发现更新成功。

在 lineitemcopy 中查看更新的数据，发现更新也成功了。

同样的，将订单 key 为 4，流水号为 3 的数据再进行修改，PCI 新值为 2020-02-01，不满足约束条件。

```
UPDATE lineitemcopy1_new
SET l_shipdate='2020-02-01'
WHERE l_orderkey=4 and l_linenumber=3;
```

查看订单 key 为 4，流水号为 3 的数据行，在 lineitemcopy_new 中查看更新的数据，发现更新成功，

在 lineitemcopy 中查看更新的数据，发现更新失败，表中维持原数据不变。