福州大学

《数据库应用实践》课程报告

学号：102102145

姓名：胡嘉鑫

年级：大三

学院：计算机与大数据学院

专业：数据科学与大数据技术

本组其它成员：学号_____姓名_____

学号_____姓名_____

实验时间：2023－2024 学年第一学期

任课教师：程烨

数据库应用实践

胡嘉鑫　　102102145

2023 年 12 月 16 日

目录

实验一：数据库管理系统及其应用开发环境的创建使用

1 实验目的

了解数据库应用开发环境的建立与使用；掌握 SQL 语言的使用；通过实践理解关系数据模型的相关概念；掌握数据库应用开发环境的使用；掌握创建、删除数据库的方法；掌握创建基本表、查看表属性、修改属性的方法；掌握向表中添加、删除以及修改数据的方法；掌握查询分析器的使用方法；掌握查询语句在单表查询中的应用；掌握复杂查询、多表查询的方法；掌握视图的使用方法；巩固数据库的基础知识.

2 实验预备内容

1. 阅读教材《数据库系统概论》第三章关系数据库标准语言 SQL.
2. 阅读实验使用的数据库管理系统的相关参考资料.

3 实验环境

- OS: Linux
- DBMS: OpenGauss DataBase

4 实验内容

4.1 所设计的数据库和表的情况简介

4.1.1 应用场景

商家用来管理产品、顾客、订单、供应商等的数据库 (commerce).

1. 产品数据;
2. 顾客数据;
3. 订单数据;
4. 供应商数据.

4.1.2 vendors 表

vendors 表存储销售产品的供应商. 每个供应商在这个表中有一个记录,供应商 ID(vend_id) 列用来匹配产品和供应商. 这个表用 vend_id 作为主键.vend_id 为自动增量字段.

表 1: vendors

列	说明
vend_id	唯一的供应商 ID
vend_name	供应商名
vend_address	供应商的地址
vend_city	供应商的城市
vend_state	供应商的州
vend_zip	供应商的邮政编码
vend_country	供应商的国家

4.1.3 products 表

products 表包含产品列表，每行一个产品. 每个产品有唯一的 ID (prod_id 列)，通过 vend_id(供应商的唯一 ID) 关联到它的供应商. 这个表用 prod_id 作为其主键. 同时在 vend_id 上定义一个外键，关联到 vendors 的 vend_id.

表 2: products

列	说明
prod_id	唯一的产品 ID
vend_id	产品供应商 ID(关联到 vendors 表中的 vend_id)
prod_name	产品名
prod_price	产品价格
prod_desc	产品描述

4.1.4 customers 表

customers 表存储所有顾客信息. 每个顾客有唯一 ID(cust_id). 这个表用 cust_id 作为主键.cust_id 是自动增量字段.

表 3: customers

列	说明
cust_id	唯一的顾客 ID
cust_name	顾客名
cust_address	顾客的地址
cust_city	顾客的城市
cust_state	顾客的州
cust_zip	顾客的邮政编码
cust_country	顾客的国家
cust_contact	顾客的联系名
cust_email	顾客的联系 email 地址

4.1.5 orders 表

orders 表存储顾客订单. 每个订单编号唯一 (order_num). 订单用 cust_id 列 (关联到 customer 表的顾客唯一 ID) 与相应顾客关联. 这个表用 order_num 作为主键.order_num 是自动增量字段. 同时在 cust_id 上定义一个外键，关联到 customers 的 cust_id.

表 4: orders

列	说明
order_num	唯一订单号
order_date	订单日期
cust_id	订单顾客 ID (关系到 customers 表的 cust_id)

4.1.6 orderitems 表

orderitems 表存储每个订单中的实际物品，每个订单的每个物品占一行. 对 orders 中的每一行，orderitems 中有一行或多行. 每个订单物品由订单号加订单物品 (第一个物品、第二个物品等) 唯一标识. 订单物品通过 order_num 列 (关联到 orders 中订单的唯一 ID) 与它们相应的订单相关联.

表 5: orderitems

列	说明
order_num	订单号 (关联到 orders 表的 order_num)
order_item	订单物品号 (在某个订单中的顺序)
prod_id	产品 ID(关联到 products 表的 prod_id)
quantity	物品数量
item_price	物品价格

4.1.7 productnotes 表

productnotes 表存储与特定产品有关的注释. 并非所有产品都有相关的注释，而有的产品可能有许多相关的注释. 这个表用 note_id 作为其主键.

表 6: productnotes

列	说明
note_id	唯一注释 ID
prod_id	产品 ID(对应于 products 表中的 prod_id)
note_date	增加注释的日期
note_text	注释文本

4.2 熟悉 DBMS 实验环境

了解华为 openGauss 数据库的安装过程及相关工具的使用方法.

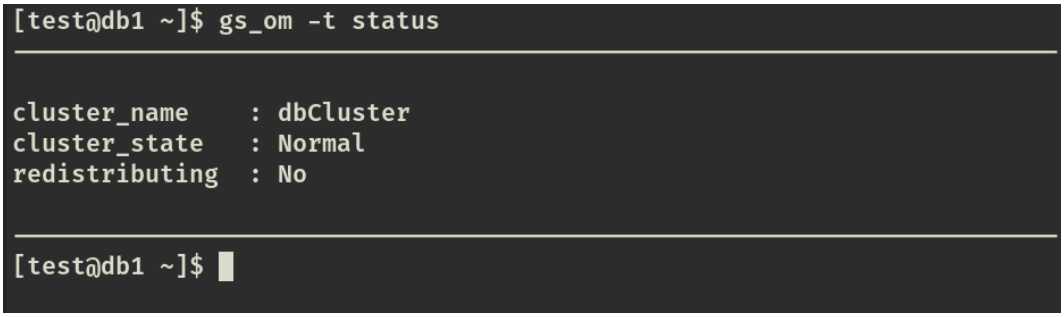


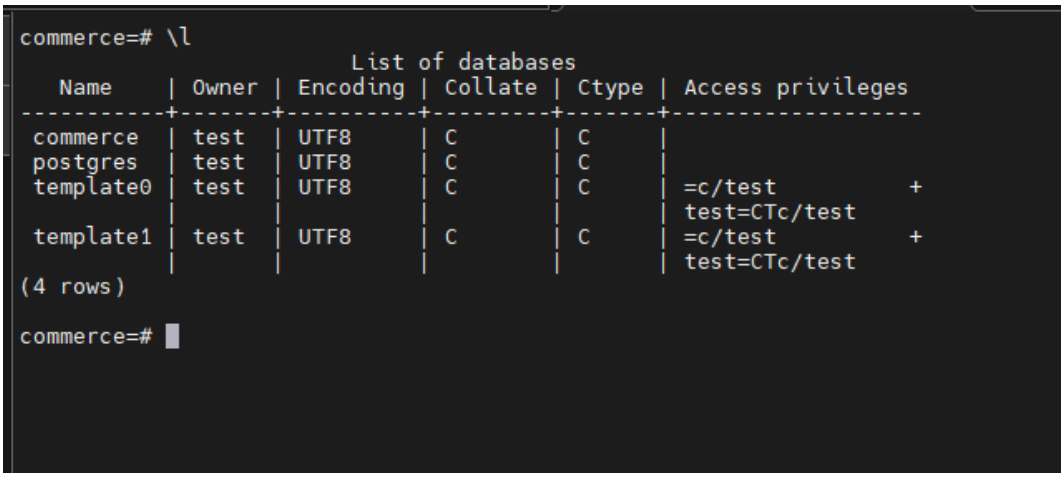
图 1: openGauss 数据库相关工具的使用和安装情况

4.3 创建数据库，创建并维护基本表的结构和数据

以下内容使用 SQL 语句完成:

1. 设计一个应用场景，创建符合该应用需求的应用数据库.

```sql
-- 应用需求如 4.1 节所述
create database commerce;
```



```
commerce=# \l
                              List of databases
   Name    | Owner | Encoding | Collate | Ctype | Access privileges
-----------+-------+----------+---------+-------+-------------------
 commerce  | test  | UTF8     | C       | C     |
 postgres  | test  | UTF8     | C       | C     |
 template0 | test  | UTF8     | C       | C     | =c/test          +
           |       |          |         |       | test=CTc/test
 template1 | test  | UTF8     | C       | C     | =c/test          +
           |       |          |         |       | test=CTc/test
(4 rows)

commerce=#
```

图 2: 创建数据库 commerce

2. 在该数据库中创建至少 4 个相互关联的基本表，并设置主键、外键、自定义完整性约束 (非空、唯一、默认值、check).

```sql
-- Create customers table
CREATE SEQUENCE customers_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
CREATE TABLE customers
(
  cust_id      int        PRIMARY KEY default
↪  nextval('customers_id_seq'::regclass),
  cust_name    char(50)   NOT NULL ,
```

```sql
  cust_address char(50)  NULL ,
  cust_city    char(50)  NULL ,
  cust_state   char(5)   NULL ,
  cust_zip     char(10)  NULL ,
  cust_country char(50)  NULL ,
  cust_contact char(50)  NULL ,
  cust_email   char(255) NULL
);

-- Create orderitems table
CREATE TABLE orderitems
(
  order_num  int          NOT NULL ,
  order_item int          NOT NULL ,
  prod_id    char(10)     NOT NULL ,
  quantity   int          NOT NULL ,
  item_price decimal(8,2) NOT NULL ,
  PRIMARY KEY (order_num, order_item)
);

-- Create orders table
CREATE SEQUENCE orders_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
CREATE TABLE orders
(
  order_num  int      NOT NULL PRIMARY KEY default
↪  nextval('orders_id_seq'::regclass),
  order_date date NOT NULL ,
  cust_id    int      NOT NULL
);

-- Create products table
```

```sql
CREATE TABLE products
(
  prod_id    char(10)      NOT NULL,
  vend_id    int           NOT NULL ,
  prod_name  char(255)     NOT NULL ,
  prod_price decimal(8,2)  NOT NULL ,
  prod_desc  text          NULL ,
  PRIMARY KEY(prod_id)
);

-- Create vendors table
CREATE SEQUENCE vendors_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
CREATE TABLE vendors
(
  vend_id      int      NOT NULL PRIMARY KEY default
↪  nextval('vendors_id_seq'::regclass),
  vend_name    char(50) NOT NULL ,
  vend_address char(50) NULL ,
  vend_city    char(50) NULL ,
  vend_state   char(5)  NULL ,
  vend_zip     char(10) NULL ,
  vend_country char(50) NULL
);

-- Create productnotes table
CREATE SEQUENCE productnotes_id_seq
START WITH 1
INCREMENT BY 1
NO MINVALUE
NO MAXVALUE
CACHE 1;
```

```sql
CREATE TABLE productnotes
(
  note_id    int           NOT NULL PRIMARY KEY default
↪  nextval('productnotes_id_seq'::regclass),
  prod_id    char(10)      NOT NULL,
  note_date date       NOT NULL,
  note_text  text          NULL
);
```

```
commerce=# drop table customers;
DROP TABLE
commerce=# CREATE SEQUENCE customers_id_seq
commerce-# START WITH 1
commerce-# INCREMENT BY 1
commerce-# NO MINVALUE
commerce-# NO MAXVALUE
commerce-# CACHE 1;
CREATE SEQUENCE
commerce=# CREATE TABLE customers
commerce-# (
commerce(#   cust_id      int       PRIMARY KEY default nextval('customers_id_seq'::regclass),
commerce(#   cust_name    char(50)  NOT NULL ,
commerce(#   cust_address char(50)  NULL ,
commerce(#   cust_city    char(50)  NULL ,
commerce(#   cust_state   char(5)   NULL ,
commerce(#   cust_zip     char(10)  NULL ,
commerce(#   cust_country char(50)  NULL ,
commerce(#   cust_contact char(50)  NULL ,
commerce(#   cust_email   char(255) NULL
commerce(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "customers_pkey" for table "customers"
CREATE TABLE
commerce=# \d customers;
                              Table "public.customers"
    Column    |       Type      |                       Modifiers
--------------+-----------------+-------------------------------------------------------
 cust_id      | integer         | not null default nextval('customers_id_seq'::regclass)
 cust_name    | character(50)   | not null
 cust_address | character(50)   |
 cust_city    | character(50)   |
 cust_state   | character(5)    |
 cust_zip     | character(10)   |
 cust_country | character(50)   |
 cust_contact | character(50)   |
 cust_email   | character(255)  |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (cust_id) TABLESPACE pg_default

commerce=#
```

图 3: 创建 customers 表

```
commerce=# CREATE SEQUENCE orders_id_seq
commerce-# START WITH 1
commerce-# INCREMENT BY 1
commerce-# NO MINVALUE
commerce-# NO MAXVALUE
commerce-# CACHE 1;
  order_num  int      NOT NULL PRIMARY KEY default nextval('orders_id_seq'::regclass),
  order_date datetime NOT NULL ,
  cust_id    int      NOT NULL
);
CREATE SEQUENCE
commerce=# CREATE TABLE orders
commerce-# (
commerce(#   order_num  int      NOT NULL PRIMARY KEY default nextval('orders_id_seq'::regclass),
commerce(#   order_date datetime NOT NULL ,
commerce(#   cust_id    int      NOT NULL
commerce(# );
ERROR:  type "datetime" does not exist
LINE 4:   order_date datetime NOT NULL ,
                      ^
commerce=# CREATE TABLE orders
(
  order_num  int      NOT NULL PRIMARY KEY default nextval('orders_id_seq'::regclass),
  order_date date NOT NULL ,
  cust_id    int      NOT NULL
);
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "orders_pkey" for table "orders"
CREATE TABLE
commerce=# \d orders;
                                    Table "public.orders"
   Column   |              Type              |                     Modifiers
------------+--------------------------------+-----------------------------------------------
 order_num  | integer                        | not null default nextval('orders_id_seq'::regclass)
 order_date | timestamp(0) without time zone | not null
 cust_id    | integer                        | not null
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_num) TABLESPACE pg_default

commerce=#
```

图 4: 创建 orders 表

```
commerce=# CREATE SEQUENCE vendors_id_seq
commerce-# START WITH 1
commerce-# INCREMENT BY 1
commerce-# NO MINVALUE
commerce-# NO MAXVALUE
commerce-# CACHE 1;
CREATE SEQUENCE
commerce=# CREATE TABLE vendors
commerce-# (
commerce(#   vend_id      int      NOT NULL PRIMARY KEY default nextval('vendors_id_seq'::regclass),
commerce(#   vend_name    char(50) NOT NULL ,
commerce(#   vend_address char(50) NULL ,
commerce(#   vend_city    char(50) NULL ,
commerce(#   vend_state   char(5)  NULL ,
commerce(#   vend_zip     char(10) NULL ,
commerce(#   vend_country char(50) NULL
commerce(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "vendors_pkey" for table "vendors"
CREATE TABLE
commerce=# \d vendors;
                                Table "public.vendors"
    Column    |      Type      |                      Modifiers
--------------+----------------+-----------------------------------------------------
 vend_id      | integer        | not null default nextval('vendors_id_seq'::regclass)
 vend_name    | character(50)  | not null
 vend_address | character(50)  |
 vend_city    | character(50)  |
 vend_state   | character(5)   |
 vend_zip     | character(10)  |
 vend_country | character(50)  |
Indexes:
    "vendors_pkey" PRIMARY KEY, btree (vend_id) TABLESPACE pg_default

commerce=#
```

图 5: 创建 vendors 表

```
commerce=# CREATE TABLE orderitems
commerce-# (
commerce(#   order_num  int          NOT NULL ,
commerce(#   order_item int          NOT NULL ,
commerce(#   prod_id    char(10)     NOT NULL ,
commerce(#   quantity   int          NOT NULL ,
commerce(#   item_price decimal(8,2) NOT NULL ,
commerce(#   PRIMARY KEY (order_num, order_item)
commerce(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "orderitems_pkey" for table "orderitems"
CREATE TABLE
commerce=# \d orderitems
       Table "public.orderitems"
   Column    |      Type      | Modifiers
-------------+----------------+-----------
 order_num   | integer        | not null
 order_item  | integer        | not null
 prod_id     | character(10)  | not null
 quantity    | integer        | not null
 item_price  | numeric(8,2)   | not null
Indexes:
    "orderitems_pkey" PRIMARY KEY, btree (order_num, order_item) TABLESPACE pg_default

commerce=#
```

图 6: 创建 orderitems 表

```
commerce=# CREATE TABLE products
commerce-# (
commerce(#   prod_id    char(10)      NOT NULL,
commerce(#   vend_id    int           NOT NULL ,
commerce(#   prod_name  char(255)     NOT NULL ,
commerce(#   prod_price decimal(8,2)  NOT NULL ,
commerce(#   prod_desc  text          NULL ,
commerce(#   PRIMARY KEY(prod_id)
commerce(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "products_pkey" for table "products"
CREATE TABLE
commerce=# \d products;
         Table "public.products"
   Column   |      Type      | Modifiers
------------+----------------+-----------
 prod_id    | character(10)  | not null
 vend_id    | integer        | not null
 prod_name  | character(255) | not null
 prod_price | numeric(8,2)   | not null
 prod_desc  | text           |
Indexes:
    "products_pkey" PRIMARY KEY, btree (prod_id) TABLESPACE pg_default

commerce=#
```

图 7: 创建 products 表

```
commerce=# CREATE SEQUENCE productnotes_id_seq
commerce-# START WITH 1
commerce-# INCREMENT BY 1
commerce-# NO MINVALUE
commerce-# NO MAXVALUE
commerce-# CACHE 1;
CREATE SEQUENCE
commerce=# CREATE TABLE productnotes
commerce-# (
commerce(#   note_id    int           NOT NULL PRIMARY KEY default nextval('productnotes_id_seq'::regclass),
commerce(#   prod_id    char(10)      NOT NULL,
commerce(#   note_date date       NOT NULL,
commerce(#   note_text  text          NULL
commerce(# );
NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index "productnotes_pkey" for table "productnotes"
CREATE TABLE
commerce=# \d productnotes;
                                  Table "public.productnotes"
  Column   |              Type              |                         Modifiers
-----------+--------------------------------+-----------------------------------------------------
 note_id   | integer                        | not null default nextval('productnotes_id_seq'::regclass)
 prod_id   | character(10)                  | not null
 note_date | timestamp(0) without time zone | not null
 note_text | text                           |
Indexes:
    "productnotes_pkey" PRIMARY KEY, btree (note_id) TABLESPACE pg_default

commerce=#
```

图 8: 创建 productnotes 表

3. 对表的结构进行修改操作.

```
ALTER TABLE orderitems ADD CONSTRAINT fk_orderitems_orders
  FOREIGN KEY (order_num) REFERENCES orders (order_num);
```

```sql
ALTER TABLE orderitems ADD CONSTRAINT fk_orderitems_products
  FOREIGN KEY (prod_id) REFERENCES products (prod_id);
ALTER TABLE orders ADD CONSTRAINT fk_orders_customers
  FOREIGN KEY (cust_id) REFERENCES customers (cust_id);
ALTER TABLE products ADD CONSTRAINT fk_products_vendors
  FOREIGN KEY (vend_id) REFERENCES vendors (vend_id);
```

```
commerce=# ALTER TABLE orderitems ADD CONSTRAINT fk_orderitems_orders
commerce-#   FOREIGN KEY (order_num) REFERENCES orders (order_num);
ALTER TABLE
commerce=# ALTER TABLE orderitems ADD CONSTRAINT fk_orderitems_products
commerce-#   FOREIGN KEY (prod_id) REFERENCES products (prod_id);
ALTER TABLE
commerce=# ALTER TABLE orders ADD CONSTRAINT fk_orders_customers
commerce-#   FOREIGN KEY (cust_id) REFERENCES customers (cust_id);
ALTER TABLE
commerce=# ALTER TABLE products ADD CONSTRAINT fk_products_vendors
commerce-#   FOREIGN KEY (vend_id) REFERENCES vendors (vend_id);
ALTER TABLE
commerce=# \d customers;
                                Table "public.customers"
    Column    |      Type      |                        Modifiers
--------------+----------------+-------------------------------------------------------
 cust_id      | integer        | not null default nextval('customers_id_seq'::regclass)
 cust_name    | character(50)  | not null
 cust_address | character(50)  |
 cust_city    | character(50)  |
 cust_state   | character(5)   |
 cust_zip     | character(10)  |
 cust_country | character(50)  |
 cust_contact | character(50)  |
 cust_email   | character(255) |
Indexes:
    "customers_pkey" PRIMARY KEY, btree (cust_id) TABLESPACE pg_default
Referenced by:
    TABLE "orders" CONSTRAINT "fk_orders_customers" FOREIGN KEY (cust_id) REFERENCES customers(cust_id)

commerce=# \d vendors;
                               Table "public.vendors"
    Column    |      Type      |                       Modifiers
--------------+----------------+-------------------------------------------------------
 vend_id      | integer        | not null default nextval('vendors_id_seq'::regclass)
 vend_name    | character(50)  | not null
 vend_address | character(50)  |
 vend_city    | character(50)  |
 vend_state   | character(5)   |
 vend_zip     | character(10)  |
 vend_country | character(50)  |
Indexes:
    "vendors_pkey" PRIMARY KEY, btree (vend_id) TABLESPACE pg_default
Referenced by:
    TABLE "products" CONSTRAINT "fk_products_vendors" FOREIGN KEY (vend_id) REFERENCES vendors(vend_id)

commerce=# \d orders;
                                     Table "public.orders"
   Column   |              Type              |                      Modifiers
------------+--------------------------------+-------------------------------------------------
 order_num  | integer                        | not null default nextval('orders_id_seq'::regclass)
 order_date | timestamp(0) without time zone | not null
 cust_id    | integer                        | not null
Indexes:
    "orders_pkey" PRIMARY KEY, btree (order_num) TABLESPACE pg_default
Foreign-key constraints:
    "fk_orders_customers" FOREIGN KEY (cust_id) REFERENCES customers(cust_id)
Referenced by:
    TABLE "orderitems" CONSTRAINT "fk_orderitems_orders" FOREIGN KEY (order_num) REFERENCES orders(order_num)

commerce=#
```

图 9: 对表进行修改操作

4. 创建索引及删除索引.

```sql
CREATE INDEX product_index ON products(prod_name);
DROP INDEX product_index;
```

```
commerce=# CREATE INDEX product_index ON products(prod_name);
CREATE INDEX
commerce=# DROP INDEX product_index;
DROP INDEX
commerce=#
```

图 10: 创建及删除索引

5. 向表中录入若干数据，对表中数据进行修改和删除操作.

```sql
-- Customers table
INSERT INTO customers(cust_id, cust_name, cust_address, cust_city,
↪  cust_state, cust_zip, cust_country, cust_contact, cust_email)
VALUES(10001, 'Coyote Inc.', '200 Maple Lane', 'Detroit', 'MI',
↪  '44444', 'USA', 'Y Lee', 'ylee@coyote.com');
INSERT INTO customers(cust_id, cust_name, cust_address, cust_city,
↪  cust_state, cust_zip, cust_country, cust_contact)
VALUES(10002, 'Mouse House', '333 Fromage Lane', 'Columbus', 'OH',
↪  '43333', 'USA', 'Jerry Mouse');
INSERT INTO customers(cust_id, cust_name, cust_address, cust_city,
↪  cust_state, cust_zip, cust_country, cust_contact, cust_email)
VALUES(10003, 'Wascals', '1 Sunny Place', 'Muncie', 'IN', '42222',
↪  'USA', 'Jim Jones', 'rabbit@wascally.com');
INSERT INTO customers(cust_id, cust_name, cust_address, cust_city,
↪  cust_state, cust_zip, cust_country, cust_contact, cust_email)
VALUES(10004, 'Yosemite Place', '829 Riverside Drive', 'Phoenix',
↪  'AZ', '88888', 'USA', 'Y Sam', 'sam@yosemite.com');
INSERT INTO customers(cust_id, cust_name, cust_address, cust_city,
↪  cust_state, cust_zip, cust_country, cust_contact)
VALUES(10005, 'E Fudd', '4545 53rd Street', 'Chicago', 'IL',
↪  '54545', 'USA', 'E Fudd');
```

```sql
-- Vendors table
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1001,'Anvils R Us','123 Main
↪  Street','Southfield','MI','48075', 'USA');
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1002,'LT Supplies','500 Park Street','Anytown','OH','44333',
↪  'USA');
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1003,'ACME','555 High Street','Los Angeles','CA','90046',
↪  'USA');
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1004,'Furball Inc.','1000 5th Avenue','New
↪  York','NY','11111', 'USA');
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1005,'Jet Set','42 Galaxy Road','London', NULL,'N16 6PS',
↪  'England');
INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city,
↪  vend_state, vend_zip, vend_country)
VALUES(1006,'Jouets Et Ours','1 Rue Amusement','Paris',
↪  NULL,'45678', 'France');

-- Products table
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('ANV01', 1001, '.5 ton anvil', 5.99, '.5 ton anvil, black,
↪  complete with handy hook');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('ANV02', 1001, '1 ton anvil', 9.99, '1 ton anvil, black,
↪  complete with handy hook and carrying case');
```

```sql
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('ANV03', 1001, '2 ton anvil', 14.99, '2 ton anvil, black,
↪  complete with handy hook and carrying case');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('OL1', 1002, 'Oil can', 8.99, 'Oil can, red');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('FU1', 1002, 'Fuses', 3.42, '1 dozen, extra long');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('SLING', 1003, 'Sling', 4.49, 'Sling, one size fits all');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('TNT1', 1003, 'TNT (1 stick)', 2.50, 'TNT, red, single
↪  stick');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('TNT2', 1003, 'TNT (5 sticks)', 10, 'TNT, red, pack of 10
↪  sticks');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('FB', 1003, 'Bird seed', 10, 'Large bag (suitable for road
↪  runners)');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('FC', 1003, 'Carrots', 2.50, 'Carrots (rabbit hunting season
↪  only)');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('SAFE', 1003, 'Safe', 50, 'Safe with combination lock');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('DTNTR', 1003, 'Detonator', 13, 'Detonator (plunger
↪  powered), fuses not included');
```

```sql
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('JP1000', 1005, 'JetPack 1000', 35, 'JetPack 1000, intended
↪  for single use');
INSERT INTO products(prod_id, vend_id, prod_name, prod_price,
↪  prod_desc)
VALUES('JP2000', 1005, 'JetPack 2000', 55, 'JetPack 2000,
↪  multi-use');

-- Orders table
INSERT INTO orders(order_num, order_date, cust_id)
VALUES(20005, '2005-09-01', 10001);
INSERT INTO orders(order_num, order_date, cust_id)
VALUES(20006, '2005-09-12', 10003);
INSERT INTO orders(order_num, order_date, cust_id)
VALUES(20007, '2005-09-30', 10004);
INSERT INTO orders(order_num, order_date, cust_id)
VALUES(20008, '2005-10-03', 10005);
INSERT INTO orders(order_num, order_date, cust_id)
VALUES(20009, '2005-10-08', 10001);

-- Orderitems table
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20005, 1, 'ANV01', 10, 5.99);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20005, 2, 'ANV02', 3, 9.99);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20005, 3, 'TNT2', 5, 10);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20005, 4, 'FB', 1, 10);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
```

```sql
VALUES(20006, 1, 'JP2000', 1, 55);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20007, 1, 'TNT2', 100, 10);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20008, 1, 'FC', 50, 2.50);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20009, 1, 'FB', 1, 10);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20009, 2, 'OL1', 1, 8.99);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20009, 3, 'SLING', 1, 4.49);
INSERT INTO orderitems(order_num, order_item, prod_id, quantity,
↪  item_price)
VALUES(20009, 4, 'ANV03', 1, 14.99);

-- Productnotes table
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(101, 'TNT2', '2005-08-17',
'Customer complaint:
Sticks not individually wrapped, too easy to mistakenly detonate
↪  all at once.
Recommend individual wrapping.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(102, 'OL1', '2005-08-18',
'Can shipped full, refills not available.
Need to order new can if refill needed.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(103, 'SAFE', '2005-08-18',
'Safe is combination locked, combination not provided with safe.
```

```
This is rarely a problem as safes are typically blown up or dropped
↪  by customers.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(104, 'FC', '2005-08-19',
'Quantity varies, sold by the sack load.
All guaranteed to be bright and orange, and suitable for use as
↪  rabbit bait.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(105, 'TNT2', '2005-08-20',
'Included fuses are short and have been known to detonate too
↪  quickly for some customers.
Longer fuses are available (item FU1) and should be recommended.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(106, 'TNT2', '2005-08-22',
'Matches not included, recommend purchase of matches or detonator
↪  (item DTNTR).'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(107, 'SAFE', '2005-08-23',
'Please note that no returns will be accepted if safe opened using
↪  explosives.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(108, 'ANV01', '2005-08-25',
'Multiple customer returns, anvils failing to drop fast enough or
↪  falling backwards on purchaser. Recommend that customer
↪  considers using heavier anvils.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(109, 'ANV03', '2005-09-01',
'Item is extremely heavy. Designed for dropping, not recommended
↪  for use with slings, ropes, pulleys, or tightropes.'
);
```

```sql
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(110, 'FC', '2005-09-01',
'Customer complaint: rabbit has been able to detect trap, food
↪  apparently less effective now.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(111, 'SLING', '2005-09-02',
'Shipped unassembled, requires common tools (including oversized
↪  hammer).'
);
iNSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(112, 'SAFE', '2005-09-02',
'Customer complaint:
Circular hole in safe floor can apparently be easily cut with
↪  handsaw.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(113, 'ANV01', '2005-09-05',
'Customer complaint:
Not heavy enough to generate flying stars around head of victim. If
↪  being purchased for dropping, recommend ANV02 or ANV03 instead.'
);
INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
VALUES(114, 'SAFE', '2005-09-07',
'Call from individual trapped in safe plummeting to the ground,
↪  suggests an escape hatch be added.
Comment forwarded to vendor.'
);

-- 更新数据
UPDATE customers
SET cust_name = 'The Fudds',
    cust_email = 'elmer@fudd.com'
WHERE cust_id = 10005;

-- 删除数据
```

```sql
DELETE FROM customers WHERE cust_id = 10006;
```

```
commerce=# INSERT INTO customers(cust_id, cust_name, cust_address, cust_city, cust_state, cust_zip, cust_country
, cust_contact, cust_email)
commerce-# VALUES(10001, 'Coyote Inc.', '200 Maple Lane', 'Detroit', 'MI', '44444', 'USA', 'Y Lee', 'ylee@coyote
.com');
INSERT 0 1
commerce=# INSERT INTO customers(cust_id, cust_name, cust_address, cust_city, cust_state, cust_zip, cust_country
, cust_contact)
commerce-# VALUES(10002, 'Mouse House', '333 Fromage Lane', 'Columbus', 'OH', '43333', 'USA', 'Jerry Mouse');
INSERT 0 1
commerce=# INSERT INTO customers(cust_id, cust_name, cust_address, cust_city, cust_state, cust_zip, cust_country
, cust_contact, cust_email)
commerce-# VALUES(10003, 'Wascals', '1 Sunny Place', 'Muncie', 'IN', '42222', 'USA', 'Jim Jones', 'rabbit@wascal
ly.com');
INSERT 0 1
commerce=# INSERT INTO customers(cust_id, cust_name, cust_address, cust_city, cust_state, cust_zip, cust_country
, cust_contact, cust_email)
commerce-# VALUES(10004, 'Yosemite Place', '829 Riverside Drive', 'Phoenix', 'AZ', '88888', 'USA', 'Y Sam', 'sam
@yosemite.com');
INSERT 0 1
commerce=# INSERT INTO customers(cust_id, cust_name, cust_address, cust_city, cust_state, cust_zip, cust_country
, cust_contact)
commerce-# VALUES(10005, 'E Fudd', '4545 53rd Street', 'Chicago', 'IL', '54545', 'USA', 'E Fudd');
INSERT 0 1
commerce=# select * from customers;
 cust_id |                cust_name                |                cust_address
   |                cust_city                | cust_state |  cust_zip  |                cust_coun
try                |                cust_contact                |
                                                                                                cust_email

---------+-----------------------------------------+-----------------------------------------
---+-----------------------------------------+------------+------------+-----------------------------
----------------------+-----------------------------------------+-----------------------------------
-----------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------
   10001 | Coyote Inc.                             | 200 Maple Lane
   | Detroit                                 | MI         | 44444      | USA
                      | Y Lee                                   | ylee@coyote.com


   10002 | Mouse House                             | 333 Fromage Lane
   | Columbus                                | OH         | 43333      | USA
                      | Jerry Mouse                             |
   10003 | Wascals                                 | 1 Sunny Place
   | Muncie                                  | IN         | 42222      | USA
                      | Jim Jones                               | rabbit@wascally.com


   10004 | Yosemite Place                          | 829 Riverside Drive
   | Phoenix                                 | AZ         | 88888      | USA
                      | Y Sam                                   | sam@yosemite.com


   10005 | E Fudd                                  | 4545 53rd Street
   | Chicago                                 | IL         | 54545      | USA
                      | E Fudd                                  |
(5 rows)

commerce=#
```

图 11: 向 customers 表插入数据

```
commerce=# INSERT INTO orders(order_num, order_date, cust_id)
commerce-# VALUES(20005, '2005-09-01', 10001);
INSERT 0 1
commerce=# INSERT INTO orders(order_num, order_date, cust_id)
commerce-# VALUES(20006, '2005-09-12', 10003);
INSERT 0 1
commerce=# INSERT INTO orders(order_num, order_date, cust_id)
commerce-# VALUES(20007, '2005-09-30', 10004);
INSERT 0 1
commerce=# INSERT INTO orders(order_num, order_date, cust_id)
commerce-# VALUES(20008, '2005-10-03', 10005);
INSERT 0 1
commerce=# INSERT INTO orders(order_num, order_date, cust_id)
commerce-# VALUES(20009, '2005-10-08', 10001);
INSERT 0 1
commerce=# select * from orders;
 order_num |     order_date      | cust_id
-----------+---------------------+---------
     20005 | 2005-09-01 00:00:00 |   10001
     20006 | 2005-09-12 00:00:00 |   10003
     20007 | 2005-09-30 00:00:00 |   10004
     20008 | 2005-10-03 00:00:00 |   10005
     20009 | 2005-10-08 00:00:00 |   10001
(5 rows)

commerce=#
```

图 12: 向 orders 表插入数据

```
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1001,'Anvils R Us','123 Main Street','Southfield','MI','48075', 'USA');
INSERT 0 1
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1002,'LT Supplies','500 Park Street','Anytown','OH','44333', 'USA');
INSERT 0 1
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1003,'ACME','555 High Street','Los Angeles','CA','90046', 'USA');
INSERT 0 1
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1004,'Furball Inc.','1000 5th Avenue','New York','NY','11111', 'USA');
INSERT 0 1
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1005,'Jet Set','42 Galaxy Road','London', NULL,'N16 6PS', 'England');
INSERT 0 1
commerce=# INSERT INTO vendors(vend_id, vend_name, vend_address, vend_city, vend_state, vend_zip, vend_country)
commerce-# VALUES(1006,'Jouets Et Ours','1 Rue Amusement','Paris', NULL,'45678', 'France');
INSERT 0 1
commerce=# select * from vendors;
 vend_id |                    vend_name                     |                   vend_address
   |                    vend_city                     | vend_state |  vend_zip  |                    vend_coun
try
---------+--------------------------------------------------+-----------------------------------------------
---+--------------------------------------------------+------------+------------+----------------------------
----------------------
    1001 | Anvils R Us                                      | 123 Main Street
   | Southfield                                       | MI         | 48075      | USA

    1002 | LT Supplies                                      | 500 Park Street
   | Anytown                                          | OH         | 44333      | USA

    1003 | ACME                                             | 555 High Street
   | Los Angeles                                      | CA         | 90046      | USA

    1004 | Furball Inc.                                     | 1000 5th Avenue
   | New York                                         | NY         | 11111      | USA

    1005 | Jet Set                                          | 42 Galaxy Road
   | London                                           |            | N16 6PS    | England

    1006 | Jouets Et Ours                                   | 1 Rue Amusement
   | Paris                                            |            | 45678      | France

(6 rows)

commerce=#
```

图 13: 向 vendors 表插入数据

```
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('ANV01', 1001, '.5 ton anvil', 5.99, '.5 ton anvil, black, complete with handy hook');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('ANV02', 1001, '1 ton anvil', 9.99, '1 ton anvil, black, complete with handy hook and carrying
 case');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('ANV03', 1001, '2 ton anvil', 14.99, '2 ton anvil, black, complete with handy hook and carryin
g case');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('OL1', 1002, 'Oil can', 8.99, 'Oil can, red');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('FU1', 1002, 'Fuses', 3.42, '1 dozen, extra long');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('SLING', 1003, 'Sling', 4.49, 'Sling, one size fits all');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('TNT1', 1003, 'TNT (1 stick)', 2.50, 'TNT, red, single stick');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('TNT2', 1003, 'TNT (5 sticks)', 10, 'TNT, red, pack of 10 sticks');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('FB', 1003, 'Bird seed', 10, 'Large bag (suitable for road runners)');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('FC', 1003, 'Carrots', 2.50, 'Carrots (rabbit hunting season only)');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('SAFE', 1003, 'Safe', 50, 'Safe with combination lock');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('DTNTR', 1003, 'Detonator', 13, 'Detonator (plunger powered), fuses not included');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('JP1000', 1005, 'JetPack 1000', 35, 'JetPack 1000, intended for single use');
INSERT 0 1
commerce=# INSERT INTO products(prod_id, vend_id, prod_name, prod_price, prod_desc)
commerce-# VALUES('JP2000', 1005, 'JetPack 2000', 55, 'JetPack 2000, multi-use');
INSERT 0 1
commerce=#
```

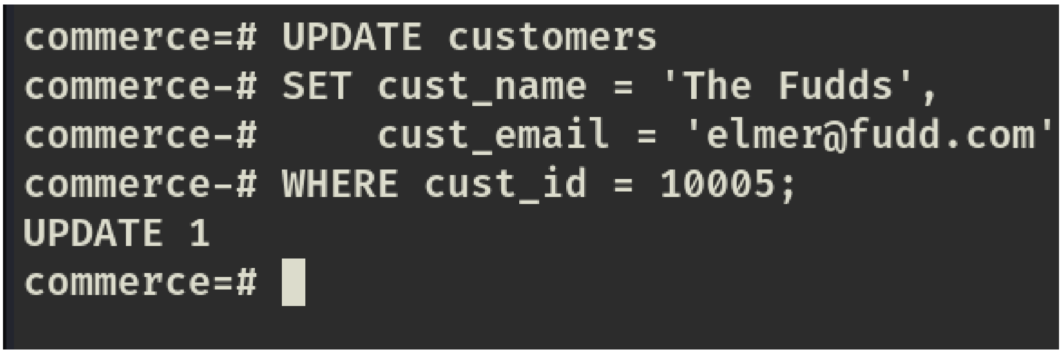
图 14: 向 products 表插入数据

```
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20005, 1, 'ANV01', 10, 5.99);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20005, 2, 'ANV02', 3, 9.99);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20005, 3, 'TNT2', 5, 10);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20005, 4, 'FB', 1, 10);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20006, 1, 'JP2000', 1, 55);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20007, 1, 'TNT2', 100, 10);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20008, 1, 'FC', 50, 2.50);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20009, 1, 'FB', 1, 10);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20009, 2, 'OL1', 1, 8.99);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20009, 3, 'SLING', 1, 4.49);
INSERT 0 1
commerce=# INSERT INTO orderitems(order_num, order_item, prod_id, quantity, item_price)
commerce-# VALUES(20009, 4, 'ANV03', 1, 14.99);
INSERT 0 1
commerce=#
commerce=# select * from orderitems;
 order_num | order_item |  prod_id   | quantity | item_price
-----------+------------+------------+----------+------------
     20005 |          1 | ANV01      |       10 |       5.99
     20005 |          2 | ANV02      |        3 |       9.99
     20005 |          3 | TNT2       |        5 |      10.00
     20005 |          4 | FB         |        1 |      10.00
     20006 |          1 | JP2000     |        1 |      55.00
     20007 |          1 | TNT2       |      100 |      10.00
     20008 |          1 | FC         |       50 |       2.50
     20009 |          1 | FB         |        1 |      10.00
     20009 |          2 | OL1        |        1 |       8.99
     20009 |          3 | SLING      |        1 |       4.49
     20009 |          4 | ANV03      |        1 |      14.99
(11 rows)

commerce=#
```

图 15: 向 orderitems 表插入数据

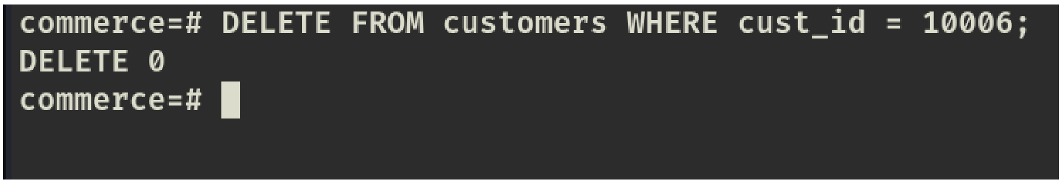
```
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(104, 'FC', '2005-08-19',
commerce(# 'Quantity varies, sold by the sack load.
commerce'# All guaranteed to be bright and orange, and suitable for use as rabbit bait.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(105, 'TNT2', '2005-08-20',
commerce(# 'Included fuses are short and have been known to detonate too quickly for some customers.
commerce'# Longer fuses are available (item FU1) and should be recommended.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(106, 'TNT2', '2005-08-22',
commerce(# 'Matches not included, recommend purchase of matches or detonator (item DTNTR).'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(107, 'SAFE', '2005-08-23',
commerce(# 'Please note that no returns will be accepted if safe opened using explosives.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(108, 'ANV01', '2005-08-25',
commerce(# 'Multiple customer returns, anvils failing to drop fast enough or falling backwards on purchaser. Rec
ommend that customer considers using heavier anvils.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(109, 'ANV03', '2005-09-01',
commerce(# 'Item is extremely heavy. Designed for dropping, not recommended for use with slings, ropes, pulleys,
 or tightropes.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(110, 'FC', '2005-09-01',
commerce(# 'Customer complaint: rabbit has been able to detect trap, food apparently less effective now.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(111, 'SLING', '2005-09-02',
commerce(# 'Shipped unassembled, requires common tools (including oversized hammer).'
commerce(# );
INSERT 0 1
commerce=# iNSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(112, 'SAFE', '2005-09-02',
commerce(# 'Customer complaint:
commerce'# Circular hole in safe floor can apparently be easily cut with handsaw.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(113, 'ANV01', '2005-09-05',
commerce(# 'Customer complaint:
commerce'# Not heavy enough to generate flying stars around head of victim. If being purchased for dropping, rec
ommend ANV02 or ANV03 instead.'
commerce(# );
INSERT 0 1
commerce=# INSERT INTO productnotes(note_id, prod_id, note_date, note_text)
commerce-# VALUES(114, 'SAFE', '2005-09-07',
commerce(# 'Call from individual trapped in safe plummeting to the ground, suggests an escape hatch be added.
commerce'# Comment forwarded to vendor.'
commerce(# );
INSERT 0 1
commerce=#
```

图 16: 向 productnotes 表插入数据



```
commerce=# UPDATE customers
commerce-# SET cust_name = 'The Fudds',
commerce-#     cust_email = 'elmer@fudd.com'
commerce-# WHERE cust_id = 10005;
UPDATE 1
commerce=# 
```

图 17: 更新 customers 表中的数据



```
commerce=# DELETE FROM customers WHERE cust_id = 10006;
DELETE 0
commerce=# 
```

图 18: 删除 customers 表中的数据

4.4 数据库查询，视图使用

在创建的表中自行设计实现以下查询：

1. 单表查询.

```
SELECT prod_id, prod_name, prod_price FROM products;
```

```
commerce=# SELECT prod_id, prod_name, prod_price FROM products limit 3;
  prod_id   |                                                                                              prod_name
                                                                                                | prod_price
────────────+───────────────────────────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────────────────────────────────────────────────────────+────────────
 ANV01      | .5 ton anvil
                                                                                                |       5.99
 ANV02      | 1 ton anvil
                                                                                                |       9.99
 ANV03      | 2 ton anvil
                                                                                                |      14.99
(3 rows)

commerce=# 
```

图 19: 单表查询

2. 多表连接查询并排序输出.

```sql
SELECT vend_name, prod_name, prod_price
FROM vendors, products
WHERE vendors.vend_id = products.vend_id
ORDER BY vend_name, prod_name;
```

```
commerce=# SELECT vend_name, prod_name, prod_price
FROM vendors, products
WHERE vendors.vend_id = products.vend_id
ORDER BY vend_name, prod_name;
                    vend_name                       |
                               prod_name
                  | prod_price
————————————————————————————————————————————————————+————————————————————————————————————————————————————
————————————————————————————————————————————————————————————————————————————————————————————————————————
——————————————————+————————————
 ACME                                               | Bird seed

                  |      10.00
 ACME                                               | Carrots

                  |       2.50
 ACME                                               | Detonator

                  |      13.00
 ACME                                               | Safe

                  |      50.00
 ACME                                               | Sling

                  |       4.49
 ACME                                               | TNT (1 stick)

                  |       2.50
 ACME                                               | TNT (5 sticks)

                  |      10.00
 Anvils R Us                                        | .5 ton anvil

                  |       5.99
```

图 20: 多表连接查询并排序输出

3. 使用聚集函数的查询.

```sql
SELECT COUNT(*) AS num_cust FROM customers;
```

```
commerce=# SELECT COUNT(*) AS num_cust FROM customers;
 num_cust
----------
        5
(1 row)

commerce=#
```

图 21: 使用聚集函数的查询

4. 分组查询.

```
SELECT vend_id, COUNT(*) AS num_prods FROM products
GROUP BY vend_id;
```

```
commerce=# SELECT vend_id, COUNT(*) AS num_prods FROM products
commerce-# GROUP BY vend_id;
 vend_id | num_prods
---------+-----------
    1001 |         3
    1003 |         7
    1005 |         2
    1002 |         2
(4 rows)

commerce=#
```

图 22: 分组查询

5. 嵌套查询.

```
SELECT cust_name, cust_contact FROM customers
WHERE cust_id IN (SELECT cust_id FROM orders
  WHERE order_num IN (SELECT order_num FROM orderitems
    WHERE prod_id = 'TNT2'));
```

```
commerce=# SELECT cust_name, cust_contact FROM customers
commerce-# WHERE cust_id IN (SELECT cust_id FROM orders
commerce(#   WHERE order_num IN (SELECT order_num FROM orderitems
commerce(#     WHERE prod_id = 'TNT2'));
                    cust_name                     |                    cust_contact
--------------------------------------------------+--------------------------------------------------
 Coyote Inc.                                      | Y Lee
 Yosemite Place                                   | Y Sam
(2 rows)

commerce=#
```

图 23: 嵌套查询

6. 创建并使用视图查询.

```
-- 创建视图
CREATE VIEW productcustomers AS
SELECT cust_name, cust_contact, prod_id
FROM customers, orders, orderitems
WHERE customers.cust_id = orders.cust_id
AND orderitems.order_num = orders.order_num;

-- 查询
SELECT cust_name, cust_contact FROM productcustomers WHERE prod_id
↪  = 'TNT2';
```

```
commerce=#
commerce=# CREATE VIEW productcustomers AS
commerce-# SELECT cust_name, cust_contact, prod_id
commerce-# FROM customers, orders, orderitems
commerce-# WHERE customers.cust_id = orders.cust_id
commerce-# AND orderitems.order_num = orders.order_num;
CREATE VIEW
commerce=# SELECT cust_name, cust_contact FROM productcustomers WHERE prod_id = 'TNT2';
                    cust_name                     |
  cust_contact
--------------------------------------------------+------------------
----------------------------------
 Coyote Inc.                                      | Y Lee

 Yosemite Place                                   | Y Sam

(2 rows)

commerce=#
```

图 24: 创建并使用视图查询

4.5 实验总结

4.5.1 实验涉及的相关知识

- Linux 命令的使用.
- openGauss 数据库的安装和基本使用.
- 数据库的创建.
- 表的创建和修改.
- 索引的创建和删除.
- 视图的创建和使用.
- 表的完整性及其约束的设计.
- 表的数据的插入、更新、删除.
- 单表查询.
- 多表连接查询.
- 嵌套查询.
- 聚合查询.

4.5.2 实验遇到的问题及其解决

没有问题.

实验二：数据库管理系统的维护与管理

5 实验目的

掌握 DBMS 提供的数据库用户和权限管理机制；理解存储过程概念，掌握存储过程与触发器的使用；掌握数据库备份与恢复方法。

6 实验预备内容

1. 阅读教材《数据库系统概论》相关章节。
2. 阅读实验使用的数据库管理系统的相关帮助文档。

7 实验环境

- OS: Linux
- DBMS: OpenGauss DataBase

8 实验内容

8.1 数据库安全性

1. 数据库账户的添加、删除

```sql
-- 账户添加
CREATE USER JIM PASSWORD 'Bigdata@123';

-- 账户查看
SELECT * FROM pg_user;

-- 账户删除
DROP USER jim CASCADE;
```

图 25: 数据库账户的添加, 删除

2. 对账户进行授予权限、收回权限。

```sql
CREATE USER joe PASSWORD 'Bigdata@123';

-- 授予权限
 GRANT ALL PRIVILEGES TO joe;

-- 收回权限
REVOKE ALL PRIVILEGES FROM joe;
```

图 26: 对账户进行授予权限、收回权限

8.2 触发器，存储过程的使用

1. 创建存储过程并执行.

```sql
CREATE TABLE t_test(c1 INT, c2 INT);

-- 创建存储过程
CREATE OR REPLACE procedure insert_data
IS
a INT;
b INT;
BEGIN
a=1;
b=2;
```

```sql
INSERT INTO t_test VALUES(a,b);
INSERT INTO t_test VALUES(b,a);
END;
/

-- 执行存储过程
CALL insert_data();
SELECT * FROM t_test;
```

```
commerce=# CREATE TABLE t_test(c1 INT, c2 INT);
CREATE TABLE
commerce=# CREATE OR REPLACE procedure insert_data
commerce-# IS
commerce$# a INT;
commerce$# b INT;
commerce$# BEGIN
commerce$# a=1;
commerce$# b=2;
commerce$# INSERT INTO t_test VALUES(a,b);
commerce$# INSERT INTO t_test VALUES(b,a);
commerce$# END;
commerce$# /
CREATE PROCEDURE
commerce=#
commerce=# CALL insert_data();
 insert_data
-------------

(1 row)

commerce=# SELECT * FROM t_test;
 c1 | c2
----+----
  1 |  2
  2 |  1
(2 rows)

commerce=#
```

图 27: 创建存储过程并执行

2. 创建触发器并测试效果。

```
--创建源表及触发表
CREATE TABLE test_trigger_src_tbl(id1 INT, id2 INT, id3 INT);
CREATE TABLE test_trigger_des_tbl(id1 INT, id2 INT, id3 INT);

--创建触发器函数
CREATE OR REPLACE FUNCTION tri_insert_func() RETURNS TRIGGER AS
  $$
  DECLARE
  BEGIN
    INSERT INTO test_trigger_des_tbl VALUES(NEW.id1, NEW.id2,
↪   NEW.id3);
    RETURN NEW;
  END
  $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION tri_update_func() RETURNS TRIGGER AS
  $$
  DECLARE
  BEGIN
    UPDATE test_trigger_des_tbl SET id3 = NEW.id3 WHERE id1=OLD.id1;
    RETURN OLD;
  END
  $$ LANGUAGE PLPGSQL;

CREATE OR REPLACE FUNCTION TRI_DELETE_FUNC() RETURNS TRIGGER AS
  $$
  DECLARE
  BEGIN
          DELETE FROM test_trigger_des_tbl WHERE id1=OLD.id1;
          RETURN OLD;
  END
  $$ LANGUAGE PLPGSQL;

-- 创建INSERT触发器
```

```sql
CREATE TRIGGER insert_trigger
  BEFORE INSERT ON test_trigger_src_tbl
  FOR EACH ROW
  EXECUTE PROCEDURE tri_insert_func();

-- 创建UPDATE触发器
CREATE TRIGGER update_trigger
  AFTER UPDATE ON test_trigger_src_tbl
  FOR EACH ROW
  EXECUTE PROCEDURE tri_update_func();

-- 创建DELETE触发器
CREATE TRIGGER delete_trigger
  BEFORE DELETE ON test_trigger_src_tbl
  FOR EACH ROW
  EXECUTE PROCEDURE tri_delete_func();

-- 执行INSERT触发事件并检查触发结果
INSERT INTO test_trigger_src_tbl VALUES(100,200,300);
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl;

-- 执行UPDATE触发事件并检查触发结果
UPDATE test_trigger_src_tbl SET id3=400 WHERE id1=100;
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl;

-- 执行DELETE触发事件并检查触发结果
DELETE FROM test_trigger_src_tbl WHERE id1=100;
SELECT * FROM test_trigger_src_tbl;
SELECT * FROM test_trigger_des_tbl;

-- 修改触发器
ALTER TRIGGER delete_trigger ON test_trigger_src_tbl RENAME TO
↪  delete_trigger_renamed;
```

```sql
-- 禁用insert_trigger触发器
ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER insert_trigger;

-- 禁用当前表上所有触发器
ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER ALL;

--删除触发器
DROP TRIGGER insert_trigger ON test_trigger_src_tbl;
DROP TRIGGER update_trigger ON test_trigger_src_tbl;
DROP TRIGGER delete_trigger_renamed ON test_trigger_src_tbl;
```

```
commerce=# CREATE TABLE test_trigger_des_tbl(id1 INT, id2 INT, id3 INT);
ERROR:  relation "test_trigger_des_tbl" already exists
commerce=#
commerce=# --创建触发器函数
commerce=# CREATE OR REPLACE FUNCTION tri_insert_func() RETURNS TRIGGER AS
commerce-#   $$
commerce$#   DECLARE
commerce$#   BEGIN
commerce$#     INSERT INTO test_trigger_des_tbl VALUES(NEW.id1, NEW.id2, NEW.id3);
commerce$#     RETURN NEW;
commerce$#   END
commerce$#   $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
commerce=#
commerce=# CREATE OR REPLACE FUNCTION tri_update_func() RETURNS TRIGGER AS
commerce-#   $$
commerce$#   DECLARE
commerce$#   BEGIN
commerce$#     UPDATE test_trigger_des_tbl SET id3 = NEW.id3 WHERE id1=OLD.id1;
commerce$#     RETURN OLD;
commerce$#   END
commerce$#   $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
commerce=#
commerce=# CREATE OR REPLACE FUNCTION TRI_DELETE_FUNC() RETURNS TRIGGER AS
commerce-#   $$
commerce$#   DECLARE
commerce$#   BEGIN
commerce$#           DELETE FROM test_trigger_des_tbl WHERE id1=OLD.id1;
commerce$#           RETURN OLD;
commerce$#   END
commerce$#   $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
commerce=#
commerce=# -- 创建INSERT触发器
commerce=# CREATE TRIGGER insert_trigger
commerce-#   BEFORE INSERT ON test_trigger_src_tbl
commerce-#   FOR EACH ROW
commerce-#   EXECUTE PROCEDURE tri_insert_func();
CREATE TRIGGER
commerce=#
commerce=# -- 创建UPDATE触发器
commerce=# CREATE TRIGGER update_trigger
commerce-#   AFTER UPDATE ON test_trigger_src_tbl
commerce-#   FOR EACH ROW
commerce-#   EXECUTE PROCEDURE tri_update_func();
CREATE TRIGGER
commerce=#
commerce=# -- 创建DELETE触发器
commerce=# CREATE TRIGGER delete_trigger
commerce-#   BEFORE DELETE ON test_trigger_src_tbl
commerce-#   FOR EACH ROW
commerce-#   EXECUTE PROCEDURE tri_delete_func();
CREATE TRIGGER
commerce=#
```

图 28: 创建触发器并测试效果

```
 id1 | id2 | id3
-----+-----+-----
 100 | 200 | 300
(1 row)

commerce=#
commerce=# -- 执行UPDATE触发事件并检查触发结果
commerce=# UPDATE test_trigger_src_tbl SET id3=400 WHERE id1=100;
UPDATE 1
commerce=# SELECT * FROM test_trigger_src_tbl;
 id1 | id2 | id3
-----+-----+-----
 100 | 200 | 400
(1 row)

commerce=# SELECT * FROM test_trigger_des_tbl;
 id1 | id2 | id3
-----+-----+-----
 100 | 200 | 400
(1 row)

commerce=#
commerce=# -- 执行DELETE触发事件并检查触发结果
commerce=# DELETE FROM test_trigger_src_tbl WHERE id1=100;
DELETE 1
commerce=# SELECT * FROM test_trigger_src_tbl;
 id1 | id2 | id3
-----+-----+-----
(0 rows)

commerce=# SELECT * FROM test_trigger_des_tbl;
 id1 | id2 | id3
-----+-----+-----
(0 rows)

commerce=#
commerce=# -- 修改触发器
commerce=# ALTER TRIGGER delete_trigger ON test_trigger_src_tbl RENAME TO delete_trigger_renamed;
ALTER TRIGGER
commerce=#
commerce=# -- 禁用insert_trigger触发器
commerce=# ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER insert_trigger;
ALTER TABLE
commerce=#
commerce=# -- 禁用当前表上所有触发器
commerce=# ALTER TABLE test_trigger_src_tbl DISABLE TRIGGER ALL;
ALTER TABLE
commerce=#
commerce=# --删除触发器
commerce=# DROP TRIGGER insert_trigger ON test_trigger_src_tbl;
DROP TRIGGER
commerce=# DROP TRIGGER update_trigger ON test_trigger_src_tbl;
DROP TRIGGER
commerce=# DROP TRIGGER delete_trigger_renamed ON test_trigger_src_tbl;
DROP TRIGGER
commerce=#
```

图 29: 创建触发器并测试效果

8.3 数据库备份与恢复

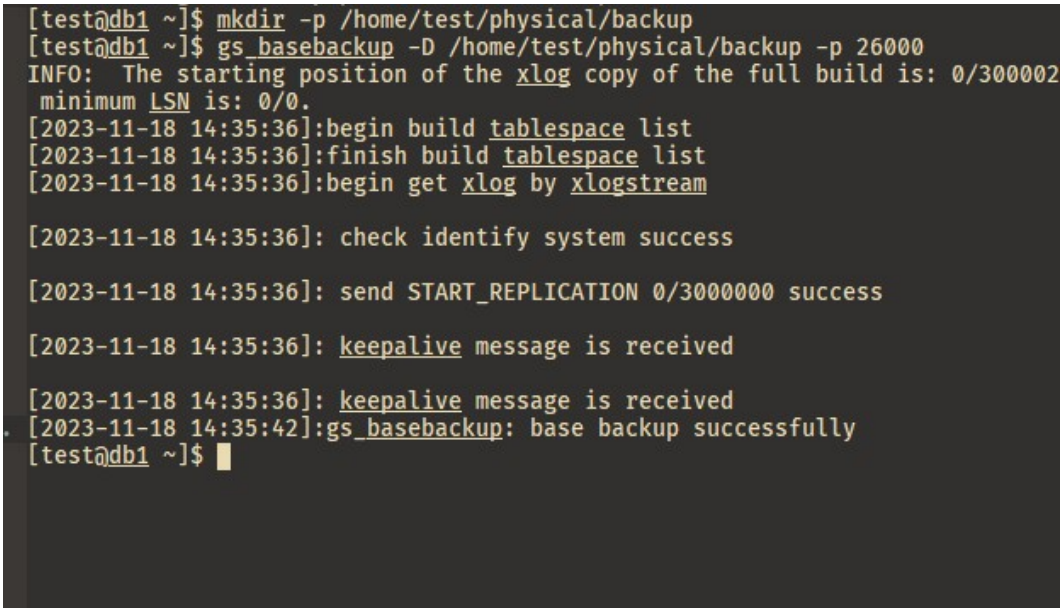
1. 对所创建的数据库进行备份

```
DROP TABLE IF EXISTS customer_t1;
CREATE TABLE customer_t1
(
 c_customer_sk integer,
```

```
 c_customer_id char(5),
 c_first_name char(6),
 c_last_name char(8)
);
INSERT INTO customer_t1 (c_customer_sk, c_customer_id,
↪  c_first_name) VALUES
(3769, 'hello', DEFAULT) ,
(6885, 'maps', 'Joes'),
(4321, 'tpcds', 'Lily'),
(9527, 'world', 'James');

DROP TABLE IF EXISTS customer_t2;
CREATE TABLE customer_t2
(
 c_customer_sk integer,
 c_customer_id char(5),
 c_first_name char(6),
 c_last_name char(8)
);
INSERT INTO customer_t2 (c_customer_sk, c_customer_id,
↪  c_first_name) VALUES
(3769, 'hello', DEFAULT) ,
(6885, 'maps', 'Joes'),
(9527, 'world', 'James');

DROP user IF EXISTS lucy;
CREATE USER lucy WITH PASSWORD "Bigdata@123";
\c - lucy
DROP TABLE IF EXISTS lucy.mytable;
CREATE TABLE mytable (firstcol int);
INSERT INTO mytable values (100);

mkdir -p /home/test/physical/backup
gs_basebackup -D /home/test/physical/backup -p 26000
```

```
. INSERT INTO customer_t2 (c_customer_sk, c_customer_id, c_first_name) VALUES
  (3769, 'hello', DEFAULT) ,
  (6885, 'maps', 'Joes'),
  (9527, 'world', 'James');

  DROP user IF EXISTS lucy;
  CREATE USER lucy WITH PASSWORD "Bigdata@123";
  \c - lucyDROP TABLE
  commerce=# CREATE TABLE customer_t1
  commerce-# (
  commerce(#  c_customer_sk integer,
  commerce(#  c_customer_id char(5),
  commerce(#  c_first_name char(6),
  commerce(#  c_last_name char(8)
  commerce(# );
  CREATE TABLE
  commerce=# INSERT INTO customer_t1 (c_customer_sk, c_customer_id, c_first_name) VALUES
  commerce-# (3769, 'hello', DEFAULT) ,
  commerce-# (6885, 'maps', 'Joes'),
  commerce-# (4321, 'tpcds', 'Lily'),
  commerce-# (9527, 'world', 'James');
  INSERT 0 4
  commerce=#
  commerce=# DROP TABLE IF EXISTS customer_t2;
  DROP TABLE
  commerce=# CREATE TABLE customer_t2
  commerce-# (
  commerce(#  c_customer_sk integer,
  commerce(#  c_customer_id char(5),
  commerce(#  c_first_name char(6),
  commerce(#  c_last_name char(8)
> commerce(# );
  CREATE TABLE
  commerce=# INSERT INTO customer_t2 (c_customer_sk, c_customer_id, c_first_name) VALUES
  commerce-# (3769, 'hello', DEFAULT) ,
  commerce-# (6885, 'maps', 'Joes'),
  commerce-# (9527, 'world', 'James');
  INSERT 0 3
  commerce=#
  commerce=# DROP user IF EXISTS lucy;
  NOTICE:  role "lucy" does not exist, skipping
  DROP ROLE
  commerce=# CREATE USER lucy WITH PASSWORD "Bigdata@123";
  CREATE ROLE
  commerce=# \c - lucy
  Password for user lucy:
  Non-SSL connection (SSL connection is recommended when requiring high-security)
  You are now connected to database "commerce" as user "lucy".
  commerce=> DROP TABLE IF EXISTS lucy.mytable;
  NOTICE:  table "mytable" does not exist, skipping
  DROP TABLE
  commerce=> CREATE TABLE mytable (firstcol int);
  CREATE TABLE
  commerce=> INSERT INTO mytable values (100);
  INSERT 0 1
  commerce=>
```

图 30: 对所创建的数据库进行备份



```
[test@db1 ~]$ mkdir -p /home/test/physical/backup
[test@db1 ~]$ gs_basebackup -D /home/test/physical/backup -p 26000
INFO:  The starting position of the xlog copy of the full build is: 0/300002
 minimum LSN is: 0/0.
[2023-11-18 14:35:36]:begin build tablespace list
[2023-11-18 14:35:36]:finish build tablespace list
[2023-11-18 14:35:36]:begin get xlog by xlogstream

[2023-11-18 14:35:36]: check identify system success

[2023-11-18 14:35:36]: send START_REPLICATION 0/3000000 success

[2023-11-18 14:35:36]: keepalive message is received

[2023-11-18 14:35:36]: keepalive message is received
[2023-11-18 14:35:42]:gs_basebackup: base backup successfully
[test@db1 ~]$
```

图 31: 对所创建的数据库进行备份

2. 利用备份进行数据库恢复

```
gs_om -t stop
cd /gaussdb/data/db1
rm -rf *
cp -r /home/test/physical/backup/. /gaussdb/data/db1
gs_om -t start
```

```
Stopping cluster.
=========================================
Successfully stopped cluster.
=========================================
End stop cluster.
[test@db1 ~]$ cd /gaussdb/data/db1
[test@db1 db1]$ rm -rf *
[test@db1 db1]$ cp -r /home/test/physical/backup/. /gaussdb/data/db1
[test@db1 db1]$ gs_om -t start
Starting cluster.
=========================================
[SUCCESS] db1
2023-11-18 14:39:00.288 65585c04.1 [unknown] 140231253587712 [unknown] 0 dn_6001 01000  0 [BACKEND] WARNING:
could not create any HA TCP/IP sockets
2023-11-18 14:39:00.291 65585c04.1 [unknown] 140231253587712 [unknown] 0 dn_6001 01000  0 [BACKEND] WARNING:
Failed to initialize the memory protect for g_instance.attr.attr_storage.cstore_buffers (16 Mbytes) or shared
memory (1496 Mbytes) is larger.
=========================================
Successfully started.
[test@db1 db1]$ gsql -d commerce -p 26000 -r
gsql ((openGauss 2.0.0 build 78689da9) compiled at 2021-03-31 21:04:03 commit 0 last mr  )
Non-SSL connection (SSL connection is recommended when requiring high-security)
Type "help" for help.

commerce=# \l
                              List of databases
   Name    | Owner | Encoding | Collate | Ctype | Access privileges
-----------+-------+----------+---------+-------+-------------------
 commerce  | test  | UTF8     | C       | C     |
 postgres  | test  | UTF8     | C       | C     |
 template0 | test  | UTF8     | C       | C     | =c/test          +
           |       |          |         |       | test=CTc/test
 template1 | test  | UTF8     | C       | C     | =c/test          +
           |       |          |         |       | test=CTc/test
(4 rows)

commerce=# use commerce;
ERROR:  syntax error at or near "use"
LINE 1: use commerce;
        ^
commerce=# \d
                                 List of relations
 Schema |          Name          |   Type   | Owner |              Storage
--------+------------------------+----------+-------+-----------------------------------
 public | customer_t1            | table    | test  | {orientation=row,compression=no}
 public | customer_t2            | table    | test  | {orientation=row,compression=no}
 public | customers              | table    | test  | {orientation=row,compression=no}
 public | customers_id_seq       | sequence | test  |
 public | id_seq                 | sequence | test  |
 public | orderitems             | table    | test  | {orientation=row,compression=no}
 public | orders                 | table    | test  | {orientation=row,compression=no}
 public | orders_id_seq          | sequence | test  |
 public | productcustomers       | view     | test  |
 public | productnotes           | table    | test  | {orientation=row,compression=no}
 public | productnotes_id_seq    | sequence | test  |
 public | products               | table    | test  | {orientation=row,compression=no}
```

图 32: 利用备份进行数据库恢复

9 实验总结

9.1 实验涉及到的相关知识

- 数据库账户的添加和删除;
- 对账户进行授予权限, 收回权限;
- 创建并执行存储过程;

- 创建触发器并测试效果;
- 删除触发器;
- 数据库的备份和恢复.

9.2 实验遇到的问题及其解决

没有问题.