

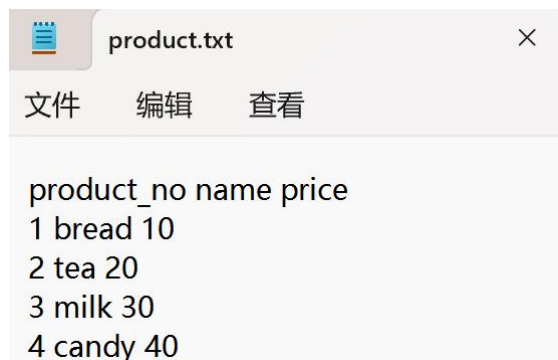
数据库原理与应用第五次作业

42233098 陈胤卿

题目一：在数据库中创建 product 表格：

```
3 ✓ CREATE TABLE product (  
4     product_no numeric(8) PRIMARY KEY,  
5     name varchar(12),  
6     price numeric(8)  
7 );
```

并自建一个符合上面关系的 txt 文件：



```
product_no name price  
1 bread 10  
2 tea 20  
3 milk 30  
4 candy 40
```

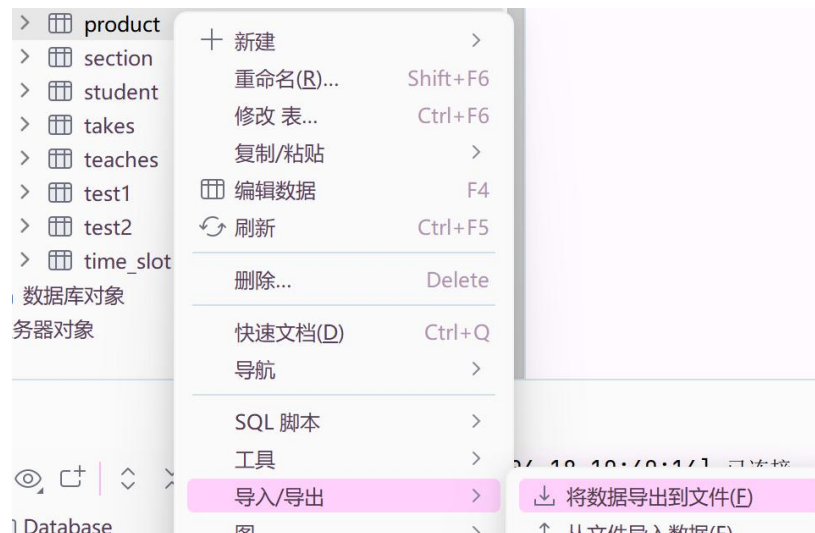
1.因为无法在 DataGrip 中使用 COPY 命令，所以在 psql 中使用\COPY 导入 txt 文件中的数据：

```
university=# \COPY product (product_no, name, price) FROM 'C:\\User  
s\\lenovo\\Desktop\\product.txt' DELIMITER ' ' CSV HEADER;  
COPY 4
```

返回“COPY 4”显示导入数据成功，进入 DataGrip 中查看：

	product_no	name	price
1	1	bread	10
2	2	tea	20
3	3	milk	30
4	4	candy	40

2.在 DataGrip 中右键单击 product 数据库，在菜单中依次选择【导入/导出】-【将数据导出到文件】：



选定文件输出路径后，点击导出为 csv 文件，显示导出成功：



题目二：

1.插入新数据：

```
16 ✓ INSERT INTO product(product_no, name, price)
17     VALUES ( product_no 666, name 'cake', price null);
```

Product 发生变化：

	product_no	name	price
1	1	bread	10
2	2	tea	20
3	3	milk	30
4	4	candy	40
5	666	cake	<null>

2.同时添加 3 个商品：

```
19 ✓ INSERT INTO product VALUES
20     ( product_no 777, name 'sugar', price 70),
21     ( product_no 888, name 'meat', price 80),
22     ( product_no 999, name 'slat', price 90);
```

Product 发生变化:

	product_no	name	price
1	1	bread	10
2	2	tea	20
3	3	milk	30
4	4	candy	40
5	666	cake	<null>
6	777	sugar	70
7	888	meat	80
8	999	slat	90

3.统一将所有商品价格打 8 折:

```
24 UPDATE product
25 SET price = price * 0.8;
26
```

不安全的查询: 不带 'where' 的 'Update' 语句会立刻更新所有表行

强制执行后 product 发生变化:

	product_no	name	price
1	1	bread	8
2	2	tea	16
3	3	milk	24
4	4	candy	32
5	666	cake	<null>
6	777	sugar	56
7	888	meat	64
8	999	slat	72

4.强制将价格大于 100 的上涨 2%，其余 4%:

```
27 UPDATE product
28 SET price = CASE
29     WHEN price > 100 THEN price * 1.02
30     ELSE price * 1.04
31 END;
```

Product 发生变化:

	product_no	name	price
1	1	bread	8
2	2	tea	17
3	3	milk	25
4	4	candy	33
5	666	cake	<null>
6	777	sugar	58
7	888	meat	67
8	999	slat	75

5.删除名字包含‘cake’的商品:

```
33 ✓ v DELETE FROM product
34      WHERE name LIKE '%cake%';
```

有 1 行受到影响, cake 被删除:

	product_no	name	price
1	1	bread	8
2	2	tea	17
3	3	milk	25
4	4	candy	33
5	777	sugar	58
6	888	meat	67
7	999	slat	75

6.删除价格高于平均价格的商品:

```
36 ✓ v DELETE FROM product
37      WHERE price > (SELECT avg(price) FROM product);
```

有 3 行受到影响:

	product_no	name	price
1	1	bread	8
2	2	tea	17
3	3	milk	25
4	4	candy	33

题目三:

1.因为原本设置的主码是 product_no, 需要先删除,再重新设置主码为(product_no, name), 接着修改设置的 name 的字段长度为 20, 运行题目给出的语句:

```
41 ALTER TABLE product DROP CONSTRAINT product_pkey;
42 ALTER TABLE product ADD CONSTRAINT product_name_pkey PRIMARY KEY (product_no, name);
43 ALTER TABLE product
44     ALTER COLUMN name TYPE VARCHAR(20);
45 INSERT INTO product (product_no, name, price)
46 SELECT
47     product_no random()*10000,
48     name 'Product' || generate_series, -- 生成名称 Product1, Product2, ...
49     price ROUND((random() * 1000)::numeric, 2) -- 生成0到1000之间的随机价格, 保留2位小数
50 FROM generate_series(1, 100000);
```

2.分别使用 DELETE 和 TRUNCATE 删除数据

DELETE 会出发是否强制执行的询问, 而 TRUNCATE 不会:

```
52 -- 使用DELETE删除数据
53 ⚠ EXPLAIN ANALYZE DELETE FROM product;
54 -- 使用TRUNCATE删除数据
55 TRUNCATE product;
56
```

不安全的查询: 不带 'where' 的 'Delete' 语句会清除表中的所有数据

[执行\(E\)](#)

且 DELETE 需要反应时间和执行时间, 而 TRUNCATE 在数据库中直接清空表中的所有数据:

QUERY PLAN	
1	Delete on product (cost=0.00..1637.04 rows=0 width=0) (actual time=350.926..350.927 rows=0 loops=1)
2	→ Seq Scan on product (cost=0.00..1637.04 rows=100004 width=6) (actual time=0.018..16.420 rows=)
3	Planning Time: 0.750 ms
4	Execution Time: 350.994 ms