# 数据库系统及安全

## 实验手册

学号：20373108

姓名：张耀东

网络空间安全学院

2022 年秋季

# 实验二、SQL 高级查询

## 一、 实验目的

1、 掌握多表连接、子查询和分组统计的 SQL 语法；

2、 熟悉 SQL 内、外连接的语法、使用场景和用法；

3、 了解 in、exist、any、all 等谓词的使用方法；

4、 了解触发器的基本原理和编写方法。

## 二、 实验要求

1、 作业模板可基于本实验手册，在实验内容后面直接填写实验报告。

2、 作业提交方式：电子版(rar/zip 打包)，文件命名格式：[学号]-[姓名]-实验[序号 1 位数，1--9].rar/zip；作业文件建议使用 word 或者 pdf 格式，不接受拍照图片版本。

3、 团队项目：[项目名称]-[小组序号 2 位数, 01--99].rar/zip, 压缩文件内须包含一个组员学号、姓名和具体分工说明的 txt 文件[文件名：小组组成与任务分工.txt]。

4、 作业提交到北航网盘：共享目录“DBMS-2022-作业”上传
https://bhpan.buaa.edu.cn:443/link/53C3AF672185198F3381B8E0F83229DF
密码：z0sD

5、 每一个实验内容，根据要求，进行实际操作，并把具体的步骤记录下来，如给出数据查询/修改等的 SQL 语句，查询结果截图后附在后面（如果数据很多，可只截第一页页面，不需要把全部的查询结果截图）；安装配置类的，可以只截重要的配置页面，以及测试成功的页面。
或者可以参考本实验提供的 sqlplus 的 spool.sql 文件，将结果输出到外部的一个 html 文件，随作业一块提交。

6、 实验作业期限：自本实验日期起，3 周之内，下一次实验课之前。

## 三、 实验内容

根据 oracle 的 sample db, 进行如下的 SQL 练习：

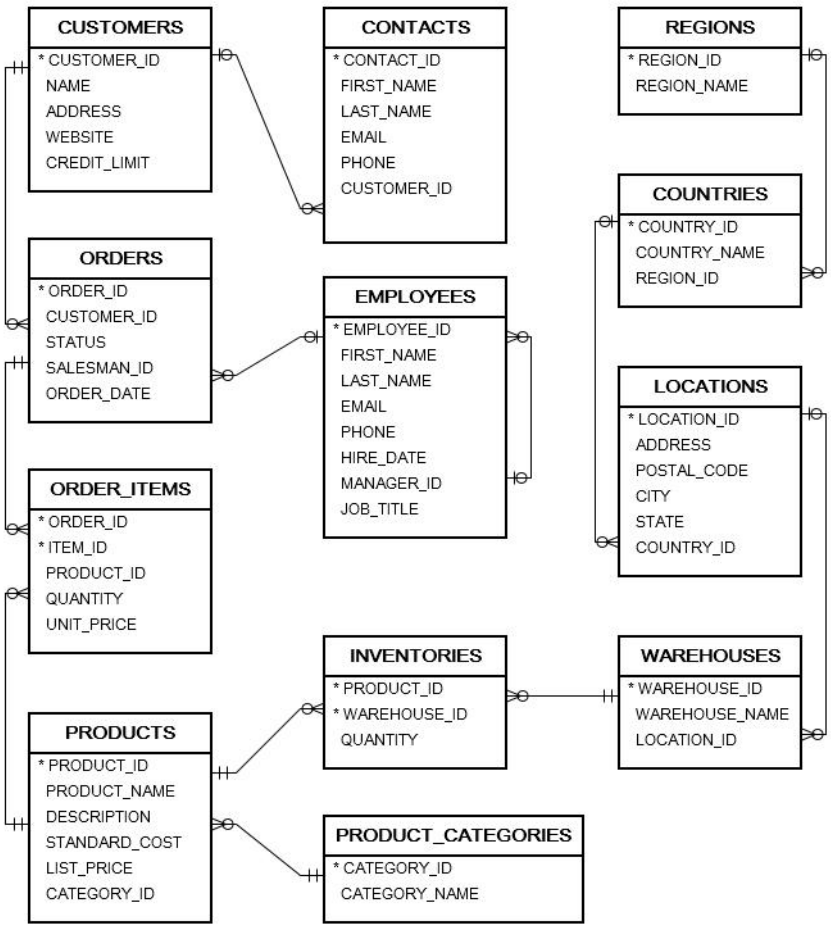



图 1 测试数据库 ERD

1、 查询所有仓库的仓库名称、地址、邮政编码、城市、州名、国家名称和区域名称。

```
SELECT DISTINCT
warehouses.WAREHOUSE_NAME,locations.ADDRESS,locations.postal_code,loca
tions.city,locations.state,countries.country_name,regions.region_name
FROM warehouses NATURAL JOIN locations NATURAL JOIN countries NATURAL JOIN
regions;
```

| WAREHOUSE_NAME | ADDRESS | postal_code | city | state | country_name | region_name |
|---|---|---|---|---|---|---|
| Toronto | 147 Spadina Ave | M5V 2L7 | Toronto | Ontario | Canada | Americas |
| Mexico City | Mariano Escobedo 9991 | 11932 | Mexico City | Distrito Federal, | Mexico | Americas |
| Southlake, Texas | 2014 Jabberwocky Rd | 26192 | Southlake | Texas | United States of America | Americas |
| San Francisco | 2011 Interiors Blvd | 99236 | South San Francisco | California | United States of America | Americas |
| New Jersey | 2007 Zagora St | 50090 | South Brunswick | New Jersey | United States of America | Americas |
| Seattle, Washington | 2004 Charade Rd | 98199 | Seattle | Washington | United States of America | Americas |
| Sydney | 12-98 Victoria Street | 2901 | Sydney | New South Wales | Australia | Asia |
| Beijing | 40-5-12 Laogianggen | 190518 | Beijing | (NULL) | China | Asia |
| Bombay | 1298 Vileparle (E) | 490231 | Bombay | Maharashtra | India | Asia |

2、 查询所有客户联系人（contacts）和雇员的信息，按照以下格式显示：

姓名（first_name 空格 last_name）、email、人员类别（客户联系人显示“contact”，雇员显示“employee”），并按照姓名升序排序；

查询所有客户联系人（contacts）和雇员的姓氏信息，如果有重复的话，如何去重。

```
SELECT CONCAT(first_name,' ',LAST_name) 姓名,email,"employee" 类别
FROM employees
UNION
SELECT CONCAT(first_name,' ',LAST_name) 姓名,email,"contact" 类别
FROM contacts
ORDER BY 姓名;
```

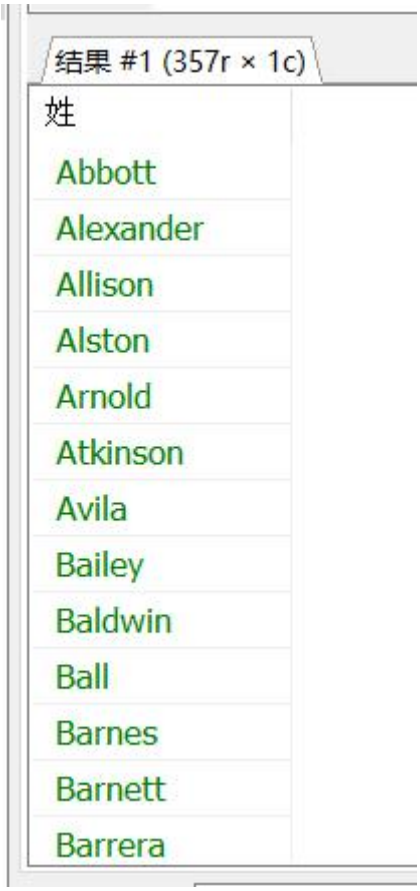
employees (426r × 3c)

| 姓名 | email | 类别 |
|---|---|---|
| Aaron Holder | aaron.holder@gilead.com | contact |
| Aaron Patterson | aaron.patterson@example.com | employee |
| Abigail Palmer | abigail.palmer@example.com | employee |
| Adah Myers | adah.myers@dom.com | contact |
| Adam Jacobs | adam.jacobs@univar.com | contact |
| Adrienne Lang | adrienne.lang@qualcomm.com | contact |
| Agustina Conner | agustina.conner@dollartree.com | contact |
| Al Schultz | al.schultz@altria.com | contact |
| Albert Watson | albert.watson@example.com | employee |
| Aleshia Reese | aleshia.reese@adp.com | contact |

employees (426r × 3c)

| 姓名 | email | 类别 |
|---|---|---|
| Vernia Hayes | vernia.hayes@drhorton.com | contact |
| Vida Kline | vida.kline@lfg.com | contact |
| Vincenza Walton | vincenza.walton@intlfcstone.com | contact |
| Violeta Stokes | violeta.stokes@honeywell.com | contact |
| Virgie Mays | virgie.mays@wholefoodsmarket.com | contact |
| Wallace Dillard | wallace.dillard@pnc.com | contact |
| Weldon Robinson | weldon.robinson@nationwide.com | contact |
| Wendell Massey | wendell.massey@bbt.com | contact |
| Willette Rodgers | willette.rodgers@pfgc.com | contact |
| Willie Barrera | willie.barrera@starbucks.com | contact |
| Willow Reyes | willow.reyes@example.com | employee |
| Yolanda Ball | yolanda.ball@gamestopcorp.com | contact |
| Yolando Wilkerson | yolando.wilkerson@oracle.com | contact |

```
SELECT DISTINCT 姓
FROM (
SELECT LAST_name 姓
    FROM employees
    UNION
    SELECT LAST_name 姓
    FROM contacts
    )AS peoples
ORDER BY 姓;
```




3、 查询有销售人员参与成交的订单客户信息，结果显示客户联系人姓名、客户公司名称和销售代表姓名。

```
SELECT DISTINCT CONCAT(contacts.first_name,' ',contacts.last_name) 客户联系人姓名,customers.name 客户公司名称,CONCAT(employees.first_name,' ',employees.last_name) 销售代表名称
FROM contacts,customers,orders,employees
WHERE
contacts.customer_id = customers.customer_id AND
customers.customer_id=orders.customer_id and orders.salesman_id = employees.employee_id
ORDER BY CONCAT(contacts.first_name,' ',contacts.last_name);
-- AND orders.`status`="Shipped" 应该加上这一项，但是答案却没有加
```

customers (66r × 3c)

| 客户联系人姓名 | 客户公司名称 | 销售代表名称 |
|---|---|---|
| Shyla Ortiz | AbbVie | Evie Harrison |
| Shyla Ortiz | AbbVie | Grace Ellis |
| Shyla Ortiz | AbbVie | Daisy Ortiz |
| Geraldine Martin | Aflac | Lily Fisher |
| Geraldine Martin | Aflac | Freya Gomez |
| Matthias Cruise | Alcoa | Grace Ellis |
| Matthias Cruise | Alcoa | Scarlett Gibson |
| Matthias Cruise | Alcoa | Daisy Ortiz |
| Guillaume Edwards | AutoNation | Florence Freeman |
| Guillaume Edwards | AutoNation | Freya Gomez |

4、 查询 2017 年销售额高于 100 万的销售员，显示 ID 和姓名。

```
SELECT
employees.employee_id,CONCAT(employees.first_name,employees.last_name)
,SUM(salesman_price.price) sum_price
FROM (
    SELECT orders.salesman_id salesman_id,order_items.quantity *
order_items.unit_price price
    FROM orders,order_items
    WHERE orders.`STATUS` <> 'Canceled' AND year(orders.ORDER_DATE) =
'2017' AND orders.ORDER_ID = order_items.order_id
) as salesman_price,employees
WHERE salesman_price.salesman_id = employees.employee_id
GROUP BY employees.employee_id
HAVING sum_price>1000000.0
ORDER BY employees.employee_id;
```

employees (4r × 3c)

| employee_id | CONCAT(employees.first_name,emp... | sum_price |
|---|---|---|
| 55 | GraceEllis | 1,990,776.9499999997 |
| 56 | EvieHarrison | 1,143,716.87 |
| 60 | IsabelleMarshall | 2,092,044.24 |
| 62 | FreyaGomez | 3,768,957.0300000003 |

5、 查询没有任何人购买的商品名称和商品描述。

```
SELECT products.product_name 商品名称,products.description
FROM products
WHERE products.product_id NOT IN (
    SELECT DISTINCT order_items.product_id
    FROM orders , order_items
    WHERE orders.order_id = order_items.order_id
    );
```

products (32r × 2c)

| 商品名称 | description |
| --- | --- |
| Seagate ST3000DM008 | Series:Barracuda,Type:7200RPM,Ca... |
| Samsung MZ-75E500B/AM | Series:850 EVO-Series,Type:SSD,Ca... |
| Corsair Dominator Platinum | Speed:DDR4-2666,Type:288-pin DI... |
| Intel Xeon E5-2695 V3 (OEM/Tray) | Speed:2.3GHz,Cores:14,TDP:120W |

6、 查询 45 号客户 46 号客户都订购过的产品信息（产品 ID 和名称）(提示：可以采用集合运算或者 outer join)。

```
SELECT products1.product_id id,products1.product_name name
FROM orders orders1,orders orders2,order_items order_items1,order_items
order_items2,products products1,products products2
WHERE
    orders1.customer_id=45 AND
    orders2.customer_id=46 AND
    orders1.order_id=order_items1.order_id AND
    orders2.order_id=order_items2.order_id AND
    order_items1.product_id=products1.product_id AND
    order_items2.product_id=products2.product_id AND
    products1.product_id=products2.product_id;
```

products (3r × 2c)

| id | name |
| --- | --- |
| 96 | SanDisk SDSSDHII-480G-G25 |
| 200 | Intel Core i7-4790K |
| 260 | Crucial CT1050MX300SSD1 |

7、 查询所有客户订单状态为”Shipped”的如下统计信息：每一位客户的 ID、

姓名、订单数量、订单总金额和订单平均金额。

```
SELECT customers.customer_id 客户 ID,customers.name 姓名,COUNT(*) 订单数量,SUM(order_items.quantity*order_items.unit_price) 订单总金额,AVG(order_items.quantity*order_items.unit_price) 订单平均金额
FROM customers,orders,order_items
WHERE orders.order_id = order_items.order_id AND orders.`status`='Shipped' AND orders.customer_id = customers.customer_id
GROUP BY customers.customer_id
```

customers (36r × 5c)

| 客户ID | 姓名 | 订单数量 | 订单总金额 | 订单平均金额 |
|---|---|---|---|---|
| 4 | AbbVie | 11 | 689,433.7200000001 | 62,675.79272727273 |
| 5 | Centene | 10 | 489,908.11 | 48,990.811 |
| 8 | International Paper | 17 | 967,863.6499999999 | 56,933.155882352934 |
| 6 | Community Health Systems | 15 | 1,244,472.15 | 82,964.81 |
| 7 | Alcoa | 10 | 528,794.22 | 52,879.422 |
| 9 | Emerson Electric | 17 | 952,330.0900000001 | 56,019.417058823536 |
| 45 | CenturyLink | 18 | 1,125,903.79 | 62,550.21055555556 |

8、 查询所有客户的每个年度的商品采购总金额，按照以下格式显示和排序：

客户 ID、客户名称、年度、采购金额（降序）。

```
SELECT customers.customer_id 客户 ID,customers.name 客户名称,customer_price.year_time 年度,customer_price.total_price 采购金额
FROM (
    SELECT SUM(order_items.quantity*order_items.unit_price) total_price,YEAR(orders.order_date) year_time,orders.customer_id id
    FROM orders,order_items
    WHERE orders.order_id = order_items.order_id AND orders.`status`<>"Canceled"
    GROUP BY YEAR(orders.order_date),orders.customer_id
) AS customer_price,customers
WHERE customer_price.id = customers.customer_id
GROUP BY customers.customer_id,customer_price.year_time
ORDER BY 采购金额 DESC;
```

customers (64r × 4c)

| 客户ID | 客户名称 | 年度 | 采购金额 |
|---|---|---|---|
| 1 | Raytheon | 2,017 | 2,406,081.53 |
| 9 | Emerson Electric | 2,016 | 1,941,234.88 |
| 8 | International Paper | 2,015 | 1,468,200.87 |
| 49 | NextEra Energy | 2,017 | 1,449,154.87 |
| 48 | Southern | 2,017 | 1,304,990.28 |
| 58 | Health Net | 2,017 | 1,269,323.7699999998 |
| 46 | Supervalu | 2,016 | 1,213,595.97 |
| 2 | Plains GP Holdings | 2,015 | 1,198,331.5899999999 |
| 4 | AbbVie | 2,017 | 1,143,716.87 |
| 47 | General Mills | 2,016 | 1,133,519.45 |

9、 查询没有库存的商品 id 和商品名称。

```
SELECT DISTINCT products.product_id 商品 id,products.product_name 商品名称
FROM products
WHERE products.product_id NOT IN(
    SELECT inventories.product_id
    FROM inventories
    );
```

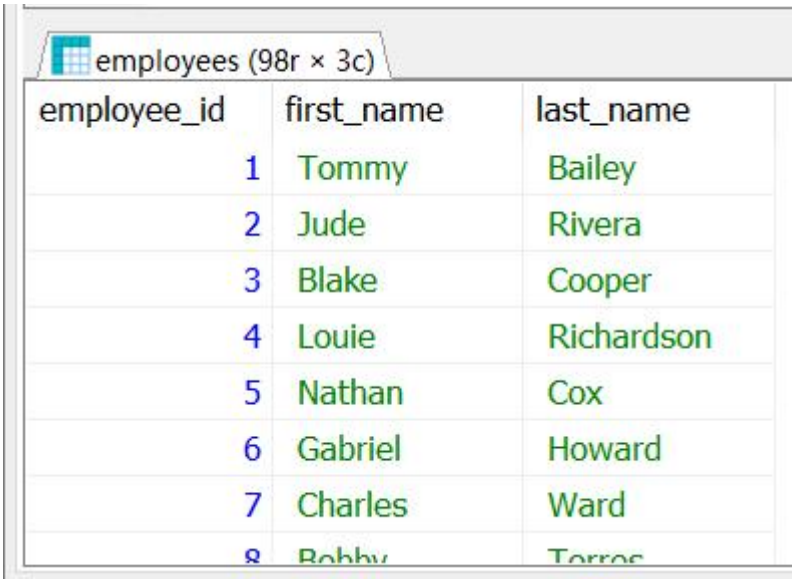
products (80r × 2c)

| 商品id | 商品名称 |
|---|---|
| 1 | G.Skill Ripjaws V Series |
| 10 | Crucial |
| 16 | Intel Core i7-6900K |
| 28 | Supermicro X9SRH-7TF |
| 45 | Intel Xeon E5-2685 V3 (OEM/Tray) |
| 48 | AMD FirePro S7000 |
| 49 | Samsung MZ-75E4T0B |
| 51 | Intel Xeon E5-2695 V4 |

10、 查询没有任何订单的销售人员名单(id, first name, last name)，要求分别使用 IN 、EXISTS 谓词 和外连接。

```
SELECT DISTINCT
```

```
employees.employee_id,employees.first_name,employees.last_name
FROM employees
WHERE employees.employee_id NOT IN
    (
    SELECT DISTINCT employees.employee_id
    FROM employees,orders
    WHERE employees.employee_id = orders.salesman_id
    );
SELECT DISTINCT
employees0.employee_id,employees0.first_name,employees0.last_name
FROM employees employees0
WHERE NOT EXISTS(
    SELECT *
    FROM orders
    WHERE orders.salesman_id = employees0.employee_id
    );
SELECT DISTINCT
employees.employee_id,employees.first_name,employees.last_name
FROM employees LEFT JOIN orders ON
employees.employee_id=orders.salesman_id
WHERE orders.order_id IS NULL;
```



employees (98r × 3c)

| employee_id | first_name | last_name |
|---|---|---|
| 1 | Tommy | Bailey |
| 2 | Jude | Rivera |
| 3 | Blake | Cooper |
| 4 | Louie | Richardson |
| 5 | Nathan | Cox |
| 6 | Gabriel | Howard |
| 7 | Charles | Ward |
| 8 | Bobby | Torres |

11、 统计每个仓库按产品类型（Category）的库存商品数量，显示仓库名称、产品类型和库存数量，并按照仓库名称和产品类型对结果进行排序。

```
SELECT warehouses.warehouse_name 仓库名
称,product_categories.category_name 产品类型,SUM(inventories.quantity) 库
```

```
存数量
FROM warehouses,product_categories,inventories,products
WHERE warehouses.warehouse_id=inventories.warehouse_id AND
inventories.product_id=products.product_id AND
products.category_id=product_categories.category_id
GROUP BY 仓库名称,产品类型
ORDER BY 仓库名称,产品类型;
```

结果 #1 (32r × 3c)

| 仓库名称 | 产品类型 | 库存数量 |
|---|---|---|
| Beijing | CPU | 2,472 |
| Beijing | Mother Board | 1,779 |
| Beijing | Storage | 6,543 |
| Beijing | Video Card | 2,688 |
| Bombay | CPU | 1,925 |
| Bombay | Mother Board | 1,178 |
| Bombay | Storage | 2,025 |
| Bombay | Video Card | 2,220 |

12、 查询采购了至少与 100 号订单的产品相同的客户信息(ID 与客户名称)。若理解为某位客户的所有订单中包含 100 号订单的话，如下

```
SELECT DISTINCT customers0.customer_id ID,customers0.name 客户名称
FROM (
    SELECT order_items.product_id id
    FROM order_items,orders
    WHERE order_items.order_id = 100
    ) AS pid,customers customers0
WHERE NOT EXISTS(
    SELECT *
--  FROM order_items
    WHERE pid.id NOT IN(
        SELECT order_items.product_id
        FROM orders,order_items
        WHERE customers0.customer_id = orders.customer_id AND
order_items.product_id
        )
    );
```

customers (47r × 2c)

| ID | 客户名称 |
|---|---|
| 1 | Raytheon |
| 2 | Plains GP Holdings |
| 3 | US Foods Holding |
| 4 | AbbVie |
| 5 | Centene |
| 6 | Community Health Systems |
| 7 | Alcoa |
| 8 | International Paper |

若理解为某位客户的某次订单中包含 100 号订单的话，如下

```
SELECT DISTINCT customers0.customer_id ID,customers0.name 客户名称
FROM customers customers0,orders orders0
WHERE customers0.customer_id = orders0.customer_id AND
orders0.`status`='Shipped' AND NOT EXISTS
    (
    SELECT *
    FROM order_items order_items100
    WHERE order_items100.order_id = 100 AND order_items100.product_id NOT
IN
        (
        SELECT order_items0.product_id
        FROM order_items order_items0
        WHERE orders0.order_id = order_items0.order_id
        )
    );
```

customers (2r × 2c)

| ID | 客户名称 |
|---|---|
| 8 | International Paper |
| 6 | Community Health Systems |

13、 查询管理员工数量最多的经理信息。

```
SELECT m.id 经理 ID,CONCAT(employees.first_name,' ',employees.last_name)
```

```
经理姓名,employees.email 经理邮件,employees.phone 经理电
话,DATE_FORMAT(employees.hire_date,'%Y-%M-%D') 雇佣日
期,employees.job_title 职位
FROM (
    SELECT  manager.employee_id id,COUNT(*) amount
    FROM employees employee,employees manager
    WHERE employee.manager_id = manager.employee_id
    GROUP BY manager.employee_id
    ) AS m,employees
WHERE m.id = employees.employee_id AND m.amount >= ALL
    (
    SELECT  COUNT(*) amount
    FROM employees employee,employees manager
    WHERE employee.manager_id = manager.employee_id
    GROUP BY manager.employee_id
    );
```

employees (1r × 6c)

| 经理ID | 经理姓名 | 经理邮件 | 经理电话 | 雇佣日期 | 职位 |
|---|---|---|---|---|---|
| 1 | Tommy Bailey | tommy.bailey@example.com | 515.123.4567 | 2016-June-17th | President |

14、 使用 SQL 将 COUNTRIES 的数据复制到 COUNTRY_A，然后将 region=4 的国家从 COUNTRY_A 中删除；使用 SQL 将 COUNTRIES 的数据复制到 COUNTRY_B，然后将 region=2 的国家从 COUNTRY_B 中删除。

根据下图，基于上述修改后的 COUNTRY 数据，假设 A=COUNTRY_A，B=COUNTRY_B，给出对应的 SQL 查询结果，并解释结果意义。

```
CREATE TABLE country_a SELECT * FROM countries;
CREATE TABLE country_b SELECT * FROM countries;

DELETE FROM country_a WHERE country_a.region_id = 2;
DELETE FROM country_b WHERE country_b.region_id = 4;
```

LEFT JOIN 如下

```
SELECT *
FROM country_a LEFT JOIN country_b ON country_a.country_id =
country_b.country_id;
```

解释：选择所有包含在 A 中的数据。

结果 #1 (20r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| KW | Kuwait | 4 | (NULL) | (NULL) | (NULL) |
| ML | Malaysia | 3 | ML | Malaysia | 3 |
| NG | Nigeria | 4 | (NULL) | (NULL) | (NULL) |
| NL | Netherlands | 1 | NL | Netherlands | 1 |
| SG | Singapore | 3 | SG | Singapore | 3 |
| UK | United Kingdom | 1 | UK | United Kingdom | 1 |
| ZM | Zambia | 4 | (NULL) | (NULL) | (NULL) |
| ZW | Zimbabwe | 4 | (NULL) | (NULL) | (NULL) |

LEFT JOIN EXCLUDING INNER JOIN 如下

```
SELECT *
FROM country_a LEFT JOIN country_b ON country_a.country_id =
country_b.country_id
WHERE country_b.country_id IS NULL;
```

解释：选择所有包含在 A 中但不包含在 B 中的数据。

结果 #1 (6r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| EG | Egypt | 4 | (NULL) | (NULL) | (NULL) |
| IL | Israel | 4 | (NULL) | (NULL) | (NULL) |
| KW | Kuwait | 4 | (NULL) | (NULL) | (NULL) |
| NG | Nigeria | 4 | (NULL) | (NULL) | (NULL) |
| ZM | Zambia | 4 | (NULL) | (NULL) | (NULL) |
| ZW | Zimbabwe | 4 | (NULL) | (NULL) | (NULL) |

INNER JOIN 如下

```
SELECT *
FROM country_a INNER JOIN country_b ON country_a.country_id =
country_b.country_id;
```

解释：选择所有既包含在 A 中也包含在 B 中的数据。

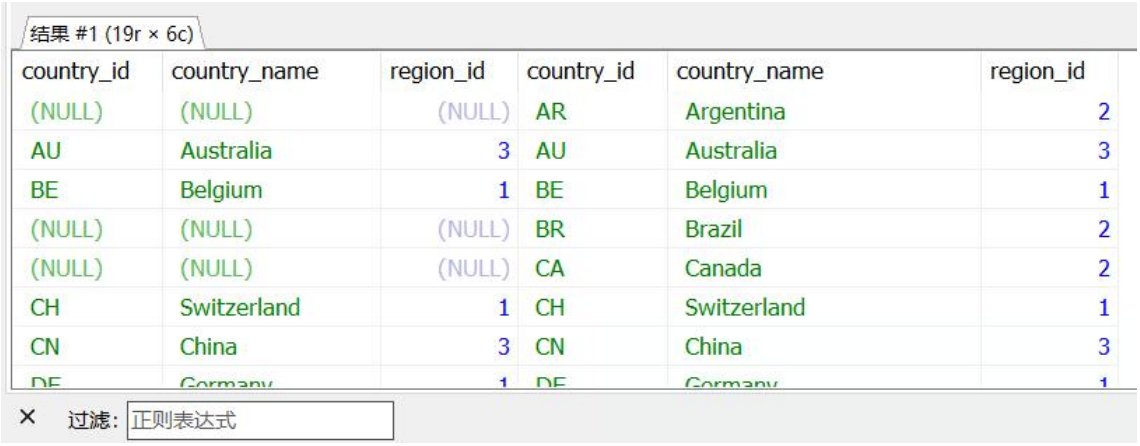
结果 #1 (14r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| AU | Australia | 3 | AU | Australia | 3 |
| BE | Belgium | 1 | BE | Belgium | 1 |
| CH | Switzerland | 1 | CH | Switzerland | 1 |
| CN | China | 3 | CN | China | 3 |
| DE | Germany | 1 | DE | Germany | 1 |
| DK | Denmark | 1 | DK | Denmark | 1 |
| FR | France | 1 | FR | France | 1 |
| IN | India | 3 | IN | India | 3 |

RIGHT JOIN 如下

```
SELECT *
FROM country_a RIGHT JOIN country_b ON country_a.country_id =
country_b.country_id;
```

解释：选择所有包含在 B 中的数据。



结果 #1 (19r × 6c)

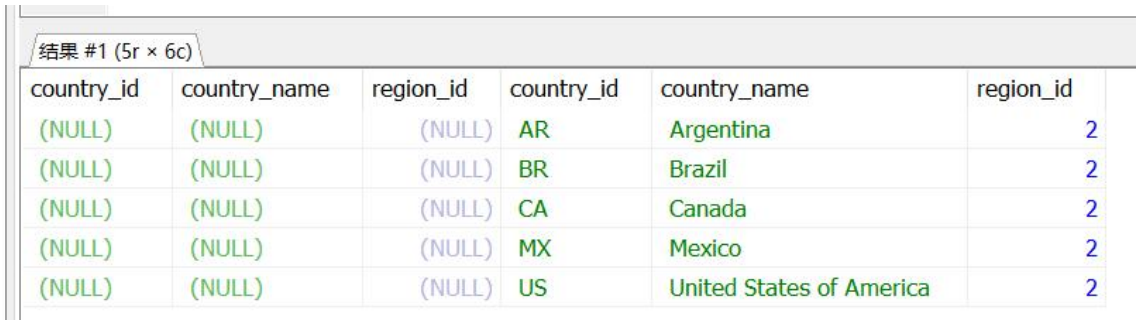
| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| (NULL) | (NULL) | (NULL) | AR | Argentina | 2 |
| AU | Australia | 3 | AU | Australia | 3 |
| BE | Belgium | 1 | BE | Belgium | 1 |
| (NULL) | (NULL) | (NULL) | BR | Brazil | 2 |
| (NULL) | (NULL) | (NULL) | CA | Canada | 2 |
| CH | Switzerland | 1 | CH | Switzerland | 1 |
| CN | China | 3 | CN | China | 3 |
| DE | Germany | 1 | DE | Germany | 1 |

过滤: 正则表达式

RIGHT JOIN EXCLUDING INNER JOIN 如下

```
SELECT *
FROM country_a RIGHT JOIN country_b ON country_a.country_id =
country_b.country_id
WHERE country_a.country_id IS NULL;
```

解释：选择所有包含在 B 中但不包含在 A 中的数据。



结果 #1 (5r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| (NULL) | (NULL) | (NULL) | AR | Argentina | 2 |
| (NULL) | (NULL) | (NULL) | BR | Brazil | 2 |
| (NULL) | (NULL) | (NULL) | CA | Canada | 2 |
| (NULL) | (NULL) | (NULL) | MX | Mexico | 2 |
| (NULL) | (NULL) | (NULL) | US | United States of America | 2 |

FULL OUTER JOIN EXCLUDING INNER JOIN 如下

```
SELECT *
FROM country_a LEFT JOIN country_b ON country_a.country_id =
country_b.country_id
UNION
SELECT *
FROM country_a RIGHT JOIN country_b ON country_a.country_id =
country_b.country_id;
```

解释：选择所有要么包含在 A 中，要么包含在 B 中的数据。

country_a (25r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| AU | Australia | 3 | AU | Australia | 3 |
| BE | Belgium | 1 | BE | Belgium | 1 |
| CH | Switzerland | 1 | CH | Switzerland | 1 |
| CN | China | 3 | CN | China | 3 |
| DE | Germany | 1 | DE | Germany | 1 |
| DK | Denmark | 1 | DK | Denmark | 1 |
| EG | Egypt | 4 | (NULL) | (NULL) | (NULL) |
| FR | France | 1 | FR | France | 1 |

FULL OUTER JOIN 如下

```
SELECT *
FROM country_a LEFT JOIN country_b ON country_a.country_id =
country_b.country_id
WHERE country_b.country_id IS NULL
UNION
SELECT *
FROM country_a RIGHT JOIN country_b ON country_a.country_id =
country_b.country_id
WHERE country_a.country_id IS NULL;
```

解释：选择所有包含在 A 中或包含在 B 中的数据。

country_a (11r × 6c)

| country_id | country_name | region_id | country_id | country_name | region_id |
|---|---|---|---|---|---|
| EG | Egypt | 4 | (NULL) | (NULL) | (NULL) |
| IL | Israel | 4 | (NULL) | (NULL) | (NULL) |
| KW | Kuwait | 4 | (NULL) | (NULL) | (NULL) |
| NG | Nigeria | 4 | (NULL) | (NULL) | (NULL) |
| ZM | Zambia | 4 | (NULL) | (NULL) | (NULL) |
| ZW | Zimbabwe | 4 | (NULL) | (NULL) | (NULL) |
| (NULL) | (NULL) | (NULL) | AR | Argentina | 2 |
| (NULL) | (NULL) | (NULL) | BR | Brazil | 2 |

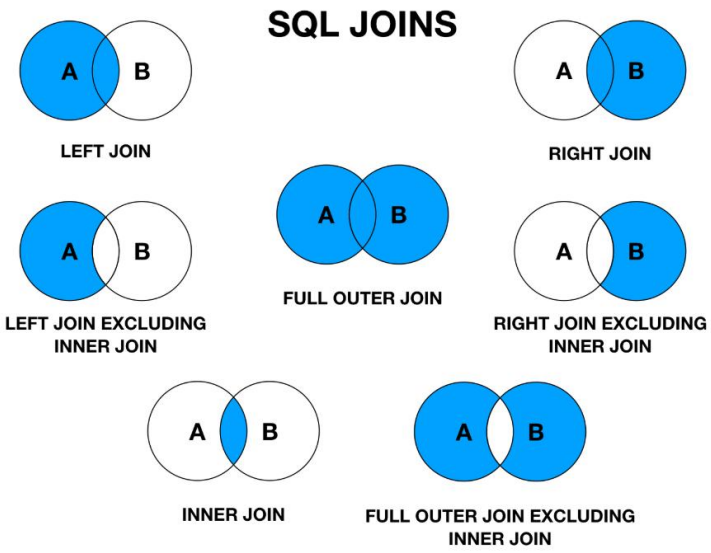
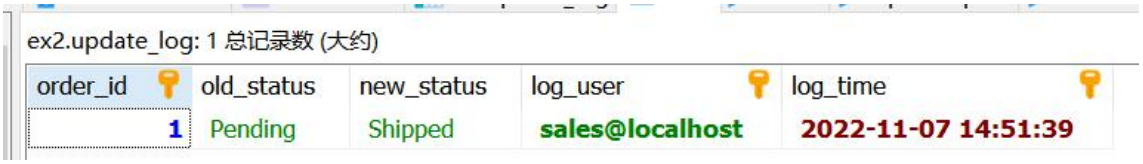



图 2 SQL 连接类型

15、 将 orders 表复制一份结构和数据到 order_trg，在新建的 order_trg 表上创建一个 after update 触发器，当把 STATUS 为“Pending”值的记录修改为“Shipped”时，要求在触发器中将每一条订单记录的修改结果记录到一个日志表 update_log，该表需要保存订单 ID、修改前的 STATUS 值、修改后的 STATUS 值、当前的用户名和修改时间（精确到秒）。

```
CREATE TABLE orders_trg SELECT * FROM orders;
CREATE TABLE update_log(
    order_id BIGINT,
    old_status VARCHAR(20),
    new_status VARCHAR(20),
    log_user VARCHAR(50),
    log_time DATETIME,
    PRIMARY KEY(order_id,log_user,log_time)
);
delimiter $
CREATE TRIGGER or_t
AFTER UPDATE ON orders_trg FOR EACH ROW
BEGIN
    if(OLD.`status`='Pending' AND NEW.`status`='Shipped') then
        INSERT INTO update_log
value(OLD.order_id,OLD.`status`,NEW.`status`,USER(),NOW());
    END if;
END;
```

```
UPDATE orders_trg SET orders_trg.`status` = 'Shipped' WHERE
orders_trg.order_id=1;
```



ex2.update_log: 1 总记录数 (大约)

| order_id | old_status | new_status | log_user | log_time |
|---|---|---|---|---|
| 1 | Pending | Shipped | sales@localhost | 2022-11-07 14:51:39 |

## 四、 实验总结与建议

答：本次实验中出现多次与题目意思理解误差的地方，如“查询有销售人员参与成交的订单客户信息”中，订单被“Canceled”后应当是未成交，却不用排除“Canceled”的订单，在自己的结果与参考输出中纠结了许久，以找到与参考相同或相近的解释以及 sql 语句。本次实验也加深了我对 sql 语句的使用，帮助我学习使用创建、查询、插入，更新表格以及创建使用触发器。