第四章　第四次作业

1. 关于视图的操作，下列哪条语句是正确的?

   A. 可以使用 UPDATE 语句更新视图中的任意数据

   B. 如果视图包含聚合函数，则该视图是不可（插入）更新的

   C. 删除基表不会影响视图

   D. 视图是只读的，不能对其执行 DELETE 操作

2. 如果在 SQL 查询中使用右外连接（RIGHT OUTER JOIN），并且右表中某些记录没有与左表中的记录匹配，那么：

   A. 右表中没有匹配的记录将被过滤掉

   B. 右表中的记录将显示 NULL

   C. 右表中的记录将完整保留，左表没有匹配的记录会显示 NULL

   D. 查询将无法执行

3. 给定以下 SQL 语句：

   ```
   CREATE VIEW employee_view AS
   SELECT id, name, department
   FROM employees
   WHERE department = 'Sales';
   ```

   该语句的作用是?

   A. 创建一个名为 employee_view 的物理表

   B. 创建一个包含所有员工信息的物理表

   C. 创建一个只包含销售部门员工的虚拟表

   D. 修改现有的员工表以只包含销售部门员工

4. 以下哪个 SQL 语句可以删除视图?

   A. DELETE VIEW view_name;

   B. DROP VIEW view_name;

   C. REMOVE VIEW view_name;

   D. ERASE VIEW view_name;

5. 假设有两张表，销售记录表 sales 和员工表 employees，要求查询出所有员工的姓名以及他们的销售记录，如果某个员工没有销售记录，销售记录应显示为 NULL。以下哪个 SQL 语句是正确的?

A.

```
SELECT employees.employee_name, sales.sale_amount FROM employees
↪  LEFT OUTER JOIN sales ON employees.employee_id =
↪  sales.employee_id;
```

B.

```
SELECT employees.employee_name, sales.sale_amount FROM employees
↪  RIGHT OUTER JOIN sales ON employees.employee_id =
↪  sales.employee_id;
```

C.

```
SELECT employees.employee_name, sales.sale_amount FROM employees
↪  FULL OUTER JOIN sales ON employees.employee_id =
↪  sales.employee_id;
```

D.

```
SELECT employees.employee_name, sales.sale_amount FROM employees
↪  INNER JOIN sales ON employees.employee_id = sales.employee_id;
```

6. 上机题 1：查询个人客户 (individual) 中 “吕东” 所有办理的业务信息。(输出：身份证号、姓名全名、客户编号、账户编号、开户行名称、可用余额、产品名称、产品类型名称)；

注意：客户全名可用 CONCAT 关键字，CONCAT(str1,str2,···)。

```
select ID_NUMBER                        身份证号,
       concat(LAST_NAME, FIRST_NAME)    姓名全名,
       CUST_ID                          客户编号,
       ACCOUNT_ID                       账户编号,
       branch.NAME                      开户行名称,
       account.AVAIL_BALANCE            可用余额,
       product.NAME                     产品名称,
       product_type.NAME                产品类型名称
from individual
         left join account using (CUST_ID)
         left join branch on account.OPEN_BRANCH_ID = branch.BRANCH_ID
         left join product using (PRODUCT_CD)
         left join product_type using (PRODUCT_TYPE_CD)
where concat(LAST_NAME, FIRST_NAME) = " 吕东";
```

	身份证号	姓名全名	客户编号	账户编号	开户行名称	可用余额	产品名称	产品类型名称
1	450881196612220768	吕东	4	10	上海市总行	59934.1200	个人活期储蓄账户	存款
2	450881196612220768	吕东	4	11	上海市总行	650600.0000	个人定期存款账户	存款
3	450881196612220768	吕东	4	12	上海市总行	67800.0000	个人通知存款账户	存款

7. 上机题 2：创建视图，查询客户（customer）表，列出所有客户的姓名全名、及其在单位的职位名称（title）(提示：对私客户职位为 NULL)。(输出：全名，职位名称)。

   注意：可使用 CASE 关键字处理对公对私客户，CASE [column_name] WHEN [value1] THEN [result1]... ELSE [default] END。

```
create view 视图1 as
select (case CUST_TYPE_CD
            when 'B' then concat(officer.LAST_NAME, officer.FIRST_NAME)
            else concat(individual.LAST_NAME, individual.FIRST_NAME) end)
            ↪  as 姓名全名,
       (case CUST_TYPE_CD when 'B' then TITLE else NULL end)
       ↪  as 职位名称
from customer
         left join individual using (CUST_ID)
         left join officer using (CUST_ID);
```

	姓名全名	职位名称
1	尤青	
2	许文强	
3	何婕	
4	吕东	
5	施珊珊	
6	张晓	
7	孔庆东	
8	曹方	
9	严匡	
10	华哨	校长
11	金歌	董事长
12	魏俊杰	董事长
13	陶海桥	董事长

8. 上机题 3：查询各个银行中余额最多的账户的客户信息。(输出：银行名称、姓名全名、账户余额)

   在 account 表中插入一条测试元组：

   insert into account values (30, 2000000, null, null, '2022-10-12', ' 正常', 5, 2, 15, 'CS');

   提示：需要处理银行中余额最多的不止一个客户的情况。

```
insert into account
values (30, 2000000, null, null, '2022-10-12', '正常', 5, 2, 15, 'CS');
select branch.NAME
↪  银行名称,
        (case CUST_TYPE_CD
            when 'B' then concat(officer.LAST_NAME, officer.FIRST_NAME)
            else concat(individual.LAST_NAME, individual.FIRST_NAME) end)
            ↪  姓名全名,
        account.AVAIL_BALANCE
        ↪  账户余额
from (select account.ACCOUNT_ID, OPEN_BRANCH_ID, AVAIL_BALANCE,
↪  max_avail_balance
        from account
                join
            (select OPEN_BRANCH_ID, max(AVAIL_BALANCE) as
            ↪  max_avail_balance
            from account
            group by OPEN_BRANCH_ID) as b using (OPEN_BRANCH_ID)
        where AVAIL_BALANCE = max_avail_balance) as abAI
            left join branch on BRANCH_ID = OPEN_BRANCH_ID
            left join account using (ACCOUNT_ID)
            left join individual using (CUST_ID)
            left join officer using (CUST_ID)
            left join customer using (CUST_ID);
```

	银行名称	姓名全名	账户余额
1	上海市总行	张晓	3330000.0000
2	建国支行	许文强	2000000.0000
3	建国支行	施珊珊	2000000.0000
4	南京分行	何婕	650000.0000
5	杭州分行	施珊珊	340023.0000

9. 上机题 4：查询所有同时办理了产品编号为 “CS” 和 “RS” 的个人客户（individual 表）的身份证号（ID_NUMBER）和姓名全名。（输出：ID_NUMBER，全名）。

   要求：使用 NOT EXISTS 进行包含关系查询。

```
select individual.ID_NUMBER 身份证号, concat(individual.LAST_NAME,
↪  individual.FIRST_NAME) 姓名全名
from individual
```

```
where not exists(select *
                 from account
                 where PRODUCT_CD != 'CS' and PRODUCT_CD != 'RS' and
                 ↪  account.CUST_ID = individual.CUST_ID);
```

	身份证号	姓名全名
1	41032619780815564X	许文强
2	370900196802069281	何婕
3	320623197108259227	施珊珊
4	321111197703197600	孔庆东

10. 上机题 5：查询所有不在 2013 年（例如在 2011、2019 年等）有交易记录的账户编号（ACCOUNT_ID）。(输出：账户编号)

    要求：需要用外连接来写该查询语句；

    注意：某些账户可能在 2011、2013 年都有交易记录，但只要在 2013 年有一条交易记录的账户都会被排除在结果集之外。

这题表述不清啊，是指“在 2011、2019 年等有交易记录”呢？还是“在 2013 年没有交易记录”呢？还是两个条件都要？

那么暂且认为是“在 2013 年没有交易记录”吧，也就是这样：

	2013 年有交易记录	2013 年无交易记录
其他年有交易记录	不要	要
其他年无交易记录	不要	要

```
select ACCOUNT_ID
from account
            left join (select *
                       from acc_transaction
                       where year(TXN_DATE) != 2013) as not2013 using
                       ↪  (ACCOUNT_ID)
where TXN_DATE is not null;
```

	ACCOUNT_ID
1	4
2	5
3	18
4	19
5	21
6	28
7	10
8	17
9	13
10	29
11	27
12	3
13	23
14	12
15	22
16	15
17	28

11. 教材 3.11：使用大学 University 数据库模式，请用 SQL 写出如下查询。

a. 找出至少选修了一门 Comp. Sci. 课程的每名学生的 ID 和姓名，保证结果中没有重复的姓名。

b. 找出没有选修 2017 年之前开设的任何课程的每名学生的 ID 和姓名。

c. 找出每个系的教师的最高工资值。可以假设每个系至少有一位教师。

d. 从前述查询所计算出的每个系的最高工资中选出所有系中的最低值。

a.

```
select distinct ID, name
from course
         left join takes using (course_id)
         left join student using (ID)
         left join department on course.dept_name =
         ↪  department.dept_name
where course.dept_name = "Comp. Sci.";
```

b.

```
select ID, name
from student
where not exists(select *
                 from takes
                 where takes.ID = student.ID and year < 2017);
```

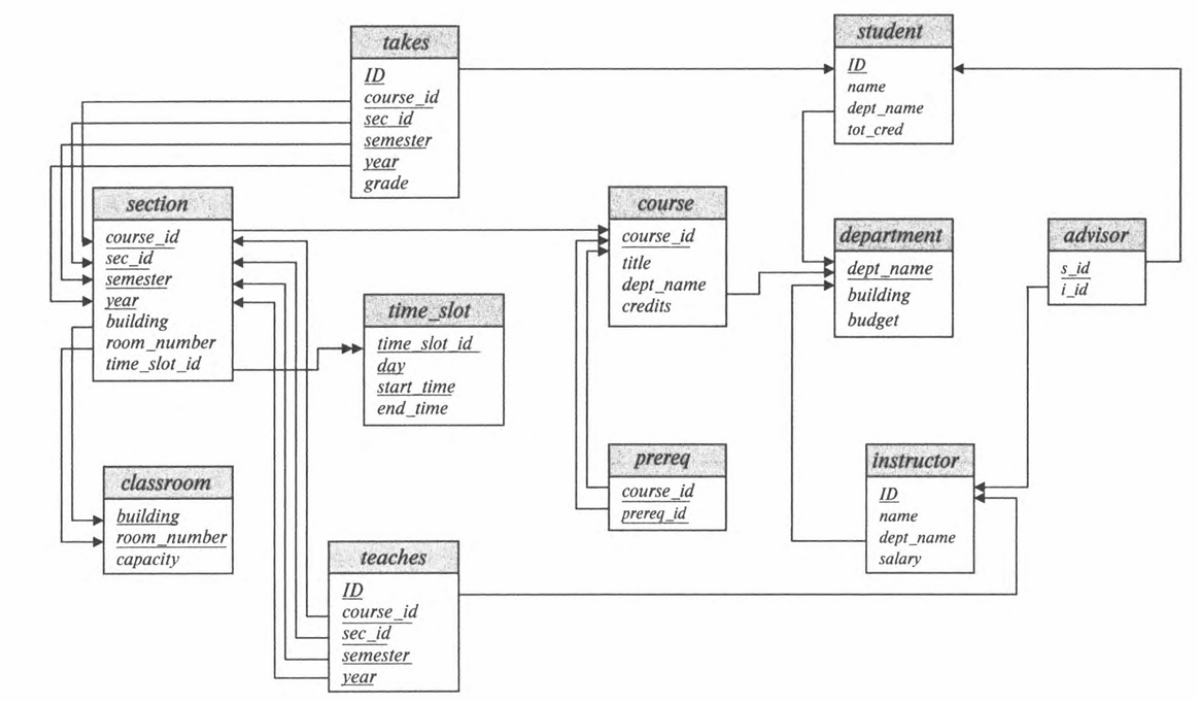

图 4.1: 大学数据库的模式图

c.

```
select dept_name, max(salary)
from instructor
group by dept_name;
```

d.

```
with a as (select dept_name, max(salary) max_salary
           from instructor
           group by dept_name)
select *
from a
where max_salary = (select min(max_salary)
                    from a);
```

验证一下

教材的前言中介绍了官网：`https://db-book.com/`，官网上可以下载示例数据：`https://db-book.com/university-lab-dir/sample_tables-dir/index.html`，于是可以验证了。

```sql
select distinct ID, name
from course
         left join takes using (course_id)
         left join student using (ID)
         left join department on course.dept_name =
         ↪  department.dept_name
where course.dept_name = "Comp. Sci.";

select ID, name
from student
where not exists(select *
                 from takes
                 where takes.ID = student.ID
                   and year < 2017);

select dept_name, max(salary)
from instructor
group by dept_name;

with a as (select dept_name, max(salary) max_salary
           from instructor
           group by dept_name)
select *
from a
where max_salary = (select min(max_salary)
                    from a);
```

	ID	name
1	00128	Zhang
2	12345	Shankar
3	45678	Levy
4	54321	Williams
5	76543	Brown
6	98765	Bourikas

	ID	name
1	00128	Zhang
2	12345	Shankar
3	19991	Brandt
4	23121	Chavez
5	44553	Peltier
6	45678	Levy
7	54321	Williams
8	55739	Sanchez
9	70557	Snow
10	76543	Brown
11	76653	Aoi
12	98765	Bourikas
13	98988	Tanaka

	dept_name	max(salary)
1	Biology	72000.00
2	Comp. Sci.	92000.00
3	Elec. Eng.	80000.00
4	Finance	90000.00
5	History	62000.00
6	Music	40000.00
7	Physics	95000.00

	dept_name	max_salary
1	Music	40000.00

12. 教材 3.16: 考虑图 4.2 中的雇员数据库，其中主码被加了下划线。请给出下面每个查询的 SQL 表达式

    a. 找出每位这样的雇员的 ID 和姓名: 该雇员所居住的城市与其工作的公司所在城市一样。

    b. 找出所居住的城市和街道与其经理相同的每位雇员的 ID 和姓名。

    c. 找出工资高于其所在公司所有雇员平均工资的每位雇员的 ID 和姓名。

    d. 找出工资总和最小的公司。

a.
```
select ID, person_name
from employee
        join works using (ID)
        join company using (company_name)
where employee.city = company.city;
```

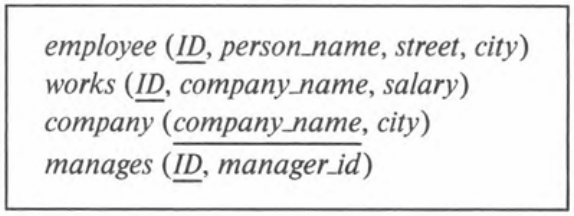

图 4.2: 雇员数据库

b.
```sql
select subordinate.ID, subordinate.person_name
from manages
         join employee as subordinate using (ID)
         join employee as superior on manages.manager_id = superior.ID
where superior.street = subordinate.street
  and superior.city = subordinate.city;
```

c.
```sql
-- 这里假设 employee.ID <--> works.ID 是一一对应的关系
select ID, person_name
from works
         join (select company_name, avg(salary) as avg_salary
               from works
               group by company_name) as a using (company_name)
         join employee using (ID)
where works.salary > a.avg_salary;
```

d.
```sql
with a as (select company_name, sum(salary) as sum_salary
           from company
                    join works using (company_name)
           group by company_name)
select company_name
from a
where sum_salary = (select min(sum_salary) from a);
```

13. 教材 3.17：考虑图 4.2 中的雇员数据库。请给出下面每个查询的 SQL 表达式。
    a. 为 “First Bank Corporation” 的所有雇员增长 10% 的工资。
    b. 为 “First Bank Corporation” 的所有经理增长 10% 的工资。
    c. 删除 “Small Bank Corporation” 的雇员在 works 关系中的所有元组。

a.
```
update works
set salary = salary * 1.1
where company_name = "First Bank Corporation";
```

b.
```
update works
join manages on works.ID = manages.manager_id
set salary = salary * 1.1
where company_name = "First Bank Corporation";
```

c.
```
delete from works
where company_name = "Small Bank Corporation";
```

14. 教材 3.18：请给出图 4.2 的雇员数据库的 SQL 模式定义。为每个属性选择合适的域，并为每个关系模式选择合适的主码。引人任何合理的外码约束。

```
create table employee
(
    ID          varchar(10) primary key,
    person_name varchar(100),
    street      varchar(100),
    city        varchar(100)
);

create table works
(
    ID           varchar(10) primary key,
    company_name varchar(100),
    salary       decimal(8, 2)
);

create table company
(
    company_name varchar(100) primary key,
    city         varchar(100)
);
```

```sql
create table manages
(
    ID         varchar(10) primary key,
    manager_id varchar(10)
);


alter table works
    add (foreign key (company_name) references company (company_name),
         foreign key (ID) references employee (ID));

alter table manages
    add (foreign key (manager_id) references employee (ID),
         foreign key (ID) references employee (ID));
```