# Chapter 8 SQL1

## ▼ 1. 基本模式定义

creat table 定义一个关系

- [x] ~~尝试用creat table 在mysql里面建立关系~~

insert update delete

## ▼ 2. 查询的基本结构

select from where

```
select name
from instructor
where dept_name = 'Comp.Sci.' and salary > 70,000;
```

2. 多关系查询

找出所有教师的姓名，已经他们所在系的名称和系所在的建筑的名称

```
select name, instructor.dept_name, building
from instructor, department
where instructor.dept_name = department.depat_name
```

**select 选定所需要的属性**

**from 需要访问的关系列表**

**where 在from的列表中的属性需要满足的条件**

```
select A_1, A_2, ...,A_n
from r_1, r_2, ...,r_n
where P;
```

- [x] ~~关系模式的定义：由各个属性列表，及各属性的域组成~~

在思考时应该以from, where, select的顺序思考

from r_1, r_2, ...,r_n 的本质是对每个relationship做一次for循环，也就是笛卡尔积

```
for each t1 in r1:
  for each t2 in r2:
    ...
    for each t_n in r_n:
      t = {t1, t2, ..., t_n}
      R.append(t)
```

- [x] ~~关系代数 $\prod$ 投影运算：$\prod_{id,name,salary}(instructor)$ 对应的是~~**~~select~~**

$\sigma$对应的是where, 而$\prod$对应的是select

# ▼ 3.附加的基本运算

as 更名运算

```
//old_name as new_name
select name as instructor_name, course_id
from instructor, teaches
where instructor.ID = teaches.ID;
```

对于大学中所有讲授课程的教师， 找出他们的姓名以及他们讲授的所有课程的课程ID

```
select T.name, S.course_id
from instructor as T, teaches as S
where T.ID = S.ID;
```

找出那些工资至少比BIOLOGY系某一位教师的工资要高的教师的名字

也可以被理解为（他们的工资比BIOLOGY的教师的最低工资高）

```
select distinct T.name
from instructor as T, instructor as S
where T.salary > S.salary and S.dept_name = "BIOLOGY
```

字符串运算

**like**

```
select dept_name
from department
where building like '%Watson%'
```

select 中的属性说明 *** 表示select 终端所有属性都被选中**

```
select instructor.*
from instructor, teach
where instructor.ID = teaches.ID
```

**order by**排列元组的显示次序

```
select name
from instructor
where dept_name = 'Physics'
order by name;
```

```
select *
from instructor
order by salary desc, name asc;
```

**between** where

找出工资在90000和100000之间

```
select name
from instructor
where salary between 90000 and 100000;
```

## ▼ 4. 集合运算

**union, intersect, except**

1. union

在2017年秋季学期或2018春季学期开设的所有课程的集合：

```
(select course_id
from section
where semester = 'Fall' and year = 2017;)
union
(select course_id
from section
where semester = 'Spring' and year = 2018;)
```

注意：**union** 运算自动去除重复 如果想要保留重复项 **union all**

2. intersect

在2017年秋季学期**和**2018春季学期**都**开设的所有课程的集合：

```
(select course_id
from section
where semester = 'Fall' and year = 2017;)
intersect
(select course_id
from section
where semester = 'Spring' and year = 2018;)
```

3. except

在2017年秋季学期**但不在**2018春季学期开设的所有课程的集合：

```
(select course_id
from section
where semester = 'Fall' and year = 2017;)
except
(select course_id
from section
where semester = 'Spring' and year = 2018;)
```

注意except all操作是单纯的减法(4-2)， 允许重复

## ▼ 5. 空值null value

1. unknow逻辑值

**and**

true **and** unknown  = unknown

false **and** unknown = false

unknown **and** unknown = unknown

**or**

true **or** unknown  = true

false **or** unknown = unknown

unknown **or** unknown = unknown

**not** unknown = unknown

找出instructor 中salary 为空的所有教师

```
select name
from instructor
where salary is null;
```

注意1:

在select distinct 时重复的元组必须被去除

```
{('A', null), ('A', null)//被认为时相同的两份拷贝,会被删除一个
```

注意2:

```
null = null  //return unknown
null is null //true
```

# ▼ 6. 聚集函数 aggregate function

1. **avg, min, max, sum, count**

找出Computer science教师的平均工资**(在计算平均值时不要去重)**

```
select avg(salary) as avg_salary
from instructor
where dept_name = 'Comp.Sci';
```

找出在2018春季学期授课的教师总数(不管一个教师讲授了几个课程,他都应该制备计算一次)

```
select count (distinct ID)
from teaches
where semester = 'Spring' and year = 2018;
```

2. **group by分组聚集**

找出每个系的平均工资

```
select dept_name, avg(salary) as avg_salary
from instructor
group by dept_name;
```

当不使用goup by 的时候我们**from r** 将一整个关系表考虑成一个分组

找出每个系在2018年春季学期授课的教师人数

```
select dept_name, count(distinct instructor.ID) as instr_count
from instructor, teaches
where instructor.ID = teaches.ID and
      semester = 'Spring' and
      year = 2018
group by dept_name;
```

**注意: 任何没有出现在group by中的属性,如果在select中出现, 那么他只能是聚集函数的参数,否则就是错误的查询**

3. **having 对分组的限定条件**

找出教师平均工资超过42000美元的系

```
select dept_name, avg(salary) as avg_salary
from instructor
group by dept_name
having avg(salary)>42000;
```

**注意:于select类似, 任何没有出现在group by中的属性,如果在having中出现, 那么他只能是聚集函数的参数,否则就是错误的查询**

- [x] ~~聚集, group by, having 的运算序列理解~~

1. 选定关系,使用**from** 进行笛卡尔积
2. **where** 子句的条件作用在**from** 关系笛卡尔积的关系上
3. 如果出现了**group by**, 满足了**where**的元组通过**group by** 分组到了一个特定的组中

   如果没有出现**group by**, 满足**where**的整个元组集被当成一个分组
4. 如果出现和**having**子句,将被应用到每个分组上不满足**having**的分组被去掉

5. **select** 利用剩下的分组产生 聚集函数得到的单个元组 i.e.查询结果中的元组

对于在2017年讲授的每一个课程段, 如果该课程段至少由2名学生选课, 找出选修该课程段的所有学生的总学分的平均值

```
select course_id, semester, year, sec_id, avg(tot_cred
from student, takes
where student.ID = take.ID and year = 2017
group by course_id, semester, year, sec_id
having count(student.ID)>=2;
```

4. **对空值和布尔值的聚集**

空值Null在聚集运算中的处理原则:除了`count(*)` 其他函数都忽略`null`

(关系代数和聚集运算符γ)

# ▼ 7. 嵌套字查询

## 集合成员资格

找出在2017年秋季和2018年春季都开课的所有课程

```
//先找到2018春的所有课程
select course_id
from section
where semester = 'Spring' and year = 2018

//嵌套
select distinct course_id
from section
where semester = 'Fall' and year = 2017 and
      course_id in (select course_id
                    from section
                    where semester = 'Spring' and year = 2018);
```

`in` 嵌套相当于是`intersect` 与之对应的`not in` 对应的就是`except`

找到既不叫Mozart也不叫Einstein的教师

```
select  distinct name
from instructor
where name not in ('Mozart','Einstein')
```

找出选修了教师id=10101所教授的课程的(不重复)的学生总数

```
select count(distinct ID)
from takes
where (course_id, sec_id, semester, year)in(select course_id, sec_id, semester,year
                                            from teaches
                                            where teaches.ID = '10101');
```

## 集合比较

找出工资至少比Biology系的某一位工资要高的所有教师姓名

利用`as` 重命名

```
select distinct name
from instructor as T, instructor as S
where T.salary > S.salary and S.dept_name = 'Biology';
```

利用`some`

```
select name
from instructor
where salary > some(select salay
                    from instructor
                    where dept_name = 'Biology');
```

同样`some` 等价于`not in`

找出工资至少比Biology系的**每一位**工资要高的所有教师姓名`> all`

```
select name
from instructor
where salary > all (select salary
                    from instructor
                    where dept_name = 'Biology');
```

找出平均工资最高的系

```
select dept_name
from instructor
group by dept_name
having avg(salary) >= all(select avg(salary)
                          from instructor
                          group by dept_name);
```

## 空关系测试

找出在2017年秋季和2018年春季都开课的所有课程

```
select course_id
from section as S
where semester = 'Fall' and year = 2017
      exists(select *
            from section as T
            where semester = 'Spring' and year = 2018
            and S.course_id = T.course_id
```

找出选修了Biology系开设的所有课程的所有学生

```
select S.ID, S.name
from student as S
where not exists((select course_id
                  from course
                  where dept_name = 'Biology')//找出所有Biology开设的课程集合
                  except
                  (select T.course_id
                  from takes as T
                  where S.ID = T.ID));//找出学生S.ID选秀的所有课程
```

- [ ] 找出选修了教师id=10101所教授的课程的(不重复)的学生总数

```
select count(distinct ID)
from takes
where exists(select course_id, sec_id, semester,year)
            from teaches
            where teaches.ID = '10101'
                  and takes.course_id = teaches.course_id
                  and takes.sec_id = teaches.sec_id
```

```
                    and takes.semester= teaches.semester
                    and takes.year= teaches.year);
```

## 重复元组存在性测试

找出在2017年最多开设一次的课程

```
select T.course_id
from course as T
where unique(select R.course_id
             from section as R
             where T.course_id = R.course_id and
                     R.year = 2017);
```

`unique` 返回的是true 和false,因为同一个课程的不同section 的course_id是相同的, 所以有一个以上的course id `unique` 就会返回false

等价于:

```
select T.course_id
from course as T
where 1 >= (select count(R.course_id)
             from section as R
             where T.course_id = R.course_id and
                     R.year = 2017);
```

找出在2017年**至少开设两次**的课程

```
select T.course_id
from course as T
where not unique(select R.course_id
             from section as R
             where T.course_id = R.course_id and
                     R.year = 2017);
```

## from中的子查询

找出系平均工资超过42000美元的那些系的教师的平均工资

```
select dept_name, avg_salary
from (select dept_name, avg(salary) as avg_salary
```

```
      from instructor
      group by instructor)
where avg_salary > 42000;
```

找出所有系中教师工资总额最大的系

```
select max(tot_salary)
from (select dept_name, sum(salary)
      from instructor
      group by dept_name) as dept_total(dept_name, tot_salary);
```

### with 临时定义关系

找出有最大预算值的那些系

```
with max_budget(value) as
    (select max(budget)
    from department)

select budget
from department, max_budget
where department.budget = max_budget.value;
```

找出工资总额大于**所有系平均工资总额**的所有系

```
with dept_total(dept_name, value) as
      (select dept_name, sum(salary)
      from instructor
      group by dept_name)

with dept_total_avg(value) as
      (select avg(value)
      from dept_total)

select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

## ▼ 8. 删除

删除instructor 中所有位于Wastson 大楼的系里办公的老师

```
delete from instructor
where dept_name in (select dept_name
                    from department
                    where building = 'Watson');
```

删除工资低于大学平均工资的所有教师记录

```
delete from instructor
where salary<(select avg(salary)
              from instructor);
```

## ▼ 9. 插入

computer science系开设名为”Database Systems”的课程CSC-3170有3个学时

```
insert into course
      values('CSC-3170', 'Database Systems', 'Comp Sci.', 4);
```

在查询结果的基础上插入元组

让Music系每个修满144学时的学生成为Music系的教师

```
insert into instructor
      select ID, name, dept_name, 18000
      from student
      where dept_name = 'Music' and tot_cred>144;
```

## ▼ 10. 更新

给工资低于70000所有教师的工资涨5%

```
update instructor
set salary = salary*1.05
where salary < 70000;
```

给工资高于100000所有教师的工资涨3%, 给其余所有教师的工资涨5%

```
update instructor
set salary = salary*1.03
where salary > 100000;

update instructor
set salary = salary*1.05
where salary <= 100000;
```

由于改变数据的时间顺序,这两个操作不能替换顺序,不然有些教师可能涨两次工资

可以使用`case` 避免

```
update instructor
set salary = case
    when salary <=100000 then salary *1.05
    else salary *1.03
end
```