1. 了解表结构

department

student, instructor, course …

2. 自己动手画图

一对多、你有我无…

3. 了解语句执行顺序和结果

from, join, on, where, group by, <aggregate>, having …, select

4. 了解correlated subquery的执行方式

5. 了解inner, left outer, right outer, full outer的结果

6. 了解exists和not exists

exists(有) = Ture | exists(没有) = False;

not exists(没有) = Ture | not exists(有) = False;

# join

**using**

student (**ID**, name, dept\_name, tot\_cred)

instructor (**ID**, name, dept\_name, salary)

takes **(ID, course\_id, sec\_id, semester, year**, grade)

teaches (**ID, course\_id, sec\_id, semester, year**)

**Ass2 3.1 b. Find the IDs of all students who were taught by an instructor named Einstein; make sure there are no duplicates in the result.**

select distinct student.ID

from (student join takes using(ID))

join (instructor join teaches using(ID))

using(course\_id, sec\_id, semester, year)

where instructor.name = ’Einstein’

Suppliers (sid, sname, status, city)

Parts (pid, pname, color, weight, city)

Projects (jid, jname, city)

Shipments (sid, pid, jid, qty): supplier sid supplies some part pid for some project jid with the qty

**2022mid a. get the names of suppliers who supply some red parts for projects located in Macau**

select s.sname

from Shipments sh, Suppliers s, Parts p, Projects j

where sh.sid = s.sid and sh.pid = p.pid and sh.jid = j.jid

and p.color = ‘red’ and j.city = ‘Macau’;

**2022mid c. get the names of parts that are either supplied by a Macau supplier or produced in Hong Kong**

select p.pname

from Parts p, Shipments sh, Suppliers s

where p.pid = sh.pid and sh.sid = s.sid

and (s.city = ‘Macau’ or p.city = ‘Hong Kong’);

account (account\_number, branch\_name, balance)

depositor (ID, account\_number)

customer (ID, customer\_name, customer\_street, customer\_city)

**Ass2 3.8c. Find the name of each branch that has at least one customer who has an account in the bank and who lives in “Harrison”.**

select distinct branch\_name

from account natural join depositor natural join customer

where customer\_city = ’Harrison’

# subquery

employee (ID, person\_name, street, city)

works (ID, company\_name, salary)

**Ass2 3.9b. Find the ID, name, and city of residence of each employee who works for First Bank Corporation” and earns more than $10000.**

select ID, person\_name, city

from employee

where ID in (select ID

from works

where company name = ’First Bank Corporation’ and salary > 10000)

employee (ID, person\_name, street, city)

works (ID, company\_name, salary)

**Ass2 3.9c. Find the ID of each employee who does not work for “First Bank Corporation”.**

select ID

from employee

where ID not in (select ID

from works

where company name = ’First Bank Corporation’)

**3.38 Find the total number of (distinct) students who have taken course sections taught by the instructor with ID 10101**

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);

**2023mid c. Get students names who enrolled not only courses taught by Computer Science department but also courses by History department**

select distinct (ID)

from takes natural join course

where dept\_name = 'Comp. Sci.'

and ID in (select ID

from takes natural join course

where dept\_name = 'History');

**3.26. 3.37 Find courses offered in Fall 2017 and in Spring 2018**

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);

**OR**

(select course\_id from section where sem = 'Fall' and year = 2017)

intersect

(select course\_id from section where sem = 'Spring' and year = 2018)

**3.26. 3.37 Find courses offered in Fall 2017 but not in Spring 2018**

select distinct course\_id

from section where semester = 'Fall' and year= 2017

and course\_id not in (select course\_id

from section

where semester = 'Spring' and year= 2018);

**OR**

(select course\_id from section where sem = 'Fall' and year = 2017)

except

(select course\_id from section where sem = 'Spring' and year = 2018);

(接着上面intersect, except写)

**3.26 Find courses that ran in Fall 2017 or in Spring 2018**

(select course\_id from section where sem = 'Fall' and year = 2017)

union

(select course\_id from section where sem = 'Spring' and year = 2018)

# exists 和correlated subquery

**3.45 Find courses offered in Fall 2017 and in Spring 2018**

select course\_id

from section as S

where S.semester = 'Fall' and S.year = 2017

and exists (select \*

from section as T

where semester = 'Spring' and year= 2018

and S.course\_id = T.course\_id);

**笔记**

select \*

from student S

where exists (select \*

from takes T

where S.ID = T.ID);

# where – group by - having

**3.34 Find the names and average salaries of all departments whose average salary is greater than 42000**

select dept\_name, avg (salary) as avg\_salary

from instructor

group by dept\_name

having avg (salary) > 42000;

**OR**

select dept\_name, avg\_salary

from (select dept\_name, avg (salary) as avg\_salary

from instructor

group by dept\_name) as dept\_avg

where avg\_salary > 42000;

**3.34**

select dept\_name, avg (salary) as avg\_salary

from instructor

where name <> ‘Eric’

group by dept\_name

having avg(salary) > 42000;

**OR**

select dept\_name, avg\_salary

from (select dept\_name, avg (salary) as avg\_salary

from instructor

where name <> ‘Eric’

group by dept\_name) as dept\_avg

where avg\_salary > 42000;

**book 76 找出每个系在2010年春季学期讲授一门课程的教师人数**

select dept\_name, count(distinct ID)

from instructor natural join teaches

when semester = ‘Spring’ and year = 2010

group by dept\_name;

**笔记**

select ID, avg(grade)

from takes

where grade > 2

group by ID

having avg(grade) > 3;

student (**ID**, name, dept\_name, tot\_cred)

takes **(ID, course\_id, sec\_id, semester, year**, grade)

**book 77 对于在2009年讲授的每个课程段，如果该课程段有至少2名学生选课，找出选修该课程段的所有学生的平均学分**

select course\_id, semester, year, sec\_id, avg(tot\_cred)

from takes natural join student

where year = 2009

group by course\_id, semester, year, sec\_id

having count(ID) >= 2

# not exists (…except…)

**if A ⊆ B**

* **if A - B = Ø**
* **if not exists ( A - B )**

**for each i ∈ S: if A ⊆ B\_i: i is true**

* **for each i ∈ S: if A – B\_i = Ø: i is ture**
* **for each i ∈ S: if not exists ( A - B ): i is ture**

company (company\_name, city)

**Ass2 3.9e. Assume that companies may be located in several cities. Find the name of each company that is located in every city in which “Small Bank Corporation” is located.**

**for each i ∈ S, if C[SBC].city ⊆ i.city: i is True**

select S.company\_name

from company S

where not exists ((select city

from company

where company\_name = ’Small Bank Corporation’)

except

(select city

from company T

where T.company\_name = S.company\_name));

student (**ID**, name, dept\_name, tot\_cred)

takes **(ID, course\_id, sec\_id, semester, year**, grade)

course (**course\_id**, title, dept\_name, credits)

**3.46 Find all students who have taken all courses offered in the Biology department.**

**for each i ⊆ S, if C[Bio].course\_id ⊆ i.course\_id: i is True**

**for each i ⊆ S, if C[Bio].course\_id ⊆ i~T.course\_id: i is True**

select distinct S.ID, S.name

from student as S

where not exists ((select course\_id

from course

where dept\_name = 'Biology')

except

(select T.course\_id

from takes as T

where S.ID = T.ID));

Suppliers (sid, sname, status, city)

Shipments (sid, pid, jid, qty): supplier sid supplies some part pid for some project jid with the qty

**2022mid d. Get supplier names who can supply any parts which are supplied by some Macau suppliers.**

**for each i ⊆ S, if S1[Macau].pid ⊆ i.pid: i is True**

**for each i ⊆ S, if S1[Macau]~SH1.pid ⊆ i~SH.pid: i is True**

select s.sname

from Supplier s

where not exists ((select sh1.pid

from Shipments sh1, suppliers s1

where sh1.sid = s1.sid

and s1.city = ‘Macau’)

except

(select sh2.pid

from Shipments sh2

where sh2.sid = s.sid));

**for each i ⊆ S, if i.bid ⊆ (B[red]&S[Tom]).bid: i is True**

**for each i ⊆ S, if i~R.bid ⊆ B[red]&S[Tom]~R.bid: i is True**

Sailors (sid, sname, rating, age)

Boats (bid,bname, color)

Reserves (sid, bid, day)

**2024 mid b. Get sailor names, who did not reserve any other boat except those ‘Red’ boats reserved by 'Tom'**

select S.sid, S.sname

from Sailors S

where not exist (

(select bid

from Reserves

where Reserves.sid = S.sid)

except

(select bid

from Reserves join Boats join Sailors

where Boas.color = ‘Red’ and Sailors.sname = ‘Tom’)

);

# not exisits

**if for ∀ t ∈ T, t condition**

* **if not exists ( t ∈ T, t not condition )**

**for each i ∈ R, if for ∀ t\_i ∈ T, t condition: i is True**

* **for each i ∈ R, if not exists ( t\_i ∈ T, t\_i not condition ): i is True**

查找与这个时间段不冲突的房间

set @new\_check\_in\_time = '2024-04-29 13:00:00';

set @new\_check\_out\_time = '2024-04-29 14:00:00';

select room\_id

from Room R

where not exists(

select \*

from Booking

where Booking.room\_id = R.room\_id

and not(

@new\_check\_in\_time > check\_out\_time

or @new\_check\_out\_time < check\_in\_time));

# left outer join

section (**course\_id, sec\_id, semester, year**, building, room\_number, time\_slot\_id)

teaches (**ID, course\_id, sec\_id, semester, year**)

instructor (**ID**, name, dept\_name, salary)

**Ass3 c. Display the list of all course sections offered in Spring 2018, along with the ID and name of each instructor teaching the section.**

**If a section has more than one instructor, that section should appear as many times in the result as it has instructors.**

**If a section does not have any instructor, it should still appear in the result with the instructor name set to “—”.**

**Section -> Teaches -> Instructor**

select course\_id, sec\_id, ID, coalesce(I.name, '—') as name

from (section natural left outer join teaches) natural left outer join instructor I

where semester = 'Spring' and year = 2018;

# left outer join+ group by和correlated subquery

**Ass3 a. b. Display a list of all instructors, showing each instructor’s ID and the number of sections taught. Make sure to show the number of sections as 0 for instructors who have not taught any section**

select ID, count(course\_id)

from instructor natural left outer join teaches

group by ID;

**OR**

select ID, (select count(course\_id)

from teaches

where teaches.ID=instructor.ID) as num\_of\_course

from instructor;

**Ass2 3.1e. Find the enrollment of each section that was offered in Fall 2017.**

select course\_id, sec\_id, count(ID)

from section natural left outer join takes

where semester = 'Spring' and year = 2017

group by course\_id, sec\_id;

**OR**

select course\_id, sec\_id, (select count(ID)

from takes

where takes.course\_id = section.course\_id

and takes.sec\_id = section.sec\_id

and takes.year = section.year

and takes.semester = section.semester)

from section

where semester = 'Spring' and year = 2017;

department (**dept\_name**, building, budget)

instructor (**ID**, name, dept\_name, salary)

**3.52 List all departments along with the number of instructors in each department**

**OR**

**Ass3 d. Display the list of all departments, with the total number of instructors in each department, without using subqueries. Make sure to show departments that have no instructors, and list those departments with an instructor count of zero.**

select dept\_name, (select count(\*)

from instructor

where instructor.dept\_name = department.dept\_name)

as num\_instructors

from department;

**OR**

select dept\_name, count(ID) as instructor\_num

from department natural left outer join instructor

group by dept\_name;

instructor (**ID**, name, dept\_name, salary)

teaches (**ID, course\_id, sec\_id, semester, year**)

**2023mid e. For each instructor, get his name and the total number of different courses he teaches.**

select ID, name, count(distinct(course\_id))

from instructor natural left outer join teaches

group by ID;

Parts (pid, pname, color, weight, city)

Shipments (sid, pid, jid, qty): supplier sid supplies some part pid for some project jid with the qty

**2022mid f. for each red part, please get the total shipment quantity**

我认为：

select coalesce(sum(sh.quantity), 0)

from parts p left outer join shipments sh

where p.color = ‘red’

group by p.pid;

但答案：

select sum(sh.quantity

from parts p, shipments sh

where p.pid = sh.pid

and p.color = ‘red’

group by p.pid;

**3.61**

select ID, (select sum(credits)

from takes T natural join course

where S.ID = T.ID)

from student S

**OR**

select ID, sum(credits)

from student S left outer join (takes T natural join course)

group by S.ID;

**1. (加限制) left outer join ( )**

**2. ( ) left outer join (加限制)**

select ID, (select sum(credits)

from takes T natural join course

where T.grade <> ‘F’ and T.grade is not null

where S.ID = T.ID)

from student S

**OR**

select ID, sum(credits)

from student S left outer join (select \*

from takes T natural join course

where T.grade <> ‘F’ and T.grade is not null) as my\_table

on S.ID = my\_table.ID

group by S.ID;

# find max

**Ass2 3.1f. Find the maximum enrollment, across all sections, in Fall 2017.**

select max(enrollment)

from (select count(ID) as enrollment

from takes

where semester = ’Fall’ and year = 2017

group by course\_id, sec\_id) as my\_table;

emp (ename, street, city)

work (ename, company\_name, salary)

mgr (ename, mgr\_name)

**2022mid 2b. Find the maximal salary of employees who are working for ‘TFB’ (a company name)**

tf1 ← σwork.company = ‘TFB’ work

tf2 ← σwork.company = ‘TFB’ work

∏tf1.salary (tf1) - ∏tf1.salary (tf1⋈tf1.salary < tf2.salary tf2)

**max: 1 – (some 1 < 2)**

**min: 1 – (some 1 > 2)**

# have max

Ass2 3.1d. Find all instructors earning the highest salary (there may be more than one with the same salary).

select ID, name

from instructor

where salary = (select max(salary)

from instructor)

**Aggregate:**

**1. with my\_table as(aggregate() as my\_count…) … my\_count = max(my\_count) in my\_table**

**2. aggregate()[group by-having] >= all aggregate()[group by]**

works (ID, company\_name, salary)

**Ass2 3.9f. Find the name of the company that has the most employees (or companies, in the case where there is a tie for the most).**

select company\_name

from works

group by company\_name

having count(distinct ID) >= all (select count(distinct ID)

from works

group by company\_name)

takes **(ID, course\_id, sec\_id, semester, year**, grade)

**Ass2 3.1g. Find the sections that had the maximum enrollment in Fall 2017.**

with sec\_enrollment as (select course\_id, sec\_id, count(ID) as enrollment

from takes

where semester = ’Fall’ and year = 2017

group by course\_id, sec\_id)

select course\_id, sec\_id

from sec\_enrollment

where enrollment = (select max(enrollment)

from sec\_enrollment);

Projects (jid, jname, city)

Shipments (sid, pid, jid, qty): supplier sid supplies some part pid for some project jid with the qty

**2022mid e. Get project names which needs the most number of different suppliers to supply parts to**

select j.jname

from projects j, shipments sh

where j.jid = sh.jid

group by j.jid

Having count(distinct sh.sid) >= all (select count(distinct sh1.sid)

from shipments sh1

group by sh1.jid)

student (**ID**, name, dept\_name, tot\_cred)

takes **(ID, course\_id, sec\_id, semester, year**, grade)

**2023mid d. get the student names who passed the most of course (A, A-, B+)**

with my\_table as (select ID, count(distinct(course\_id)) as my\_count

from takes

where grade in ('A', 'A-', 'B+')

group by ID)

select name

from student natural join my\_table

where my\_count = (select max(my\_count) from my\_table);

**OR**

select name

from student natural join takes

where grade in ('A', 'A-', 'B+')

group by ID

Having count(distinct(course\_id)) >= all (select count(distinct(course\_id))

from takes

where grade in ('A', 'A-', 'B+')

group by ID);

# have more than all xx类

**Ass2 3.9d. Find the ID of each employee who earns more than every employee of “Small Bank Corporation”.**

select ID

from works

where salary > all (select salary

from works

where company name = ’Small Bank Corporation’)

**3.42 Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.**

select name

from instructor

where salary > all (select salary

from instructor

where dept\_name = 'Biology');

# aggregate() have more than aggregate(xx类)

**aggregate()<group by-having > have more than aggregate(xx类)<where>**

**Ass2 3.9g. Find the name of each company whose employees earn a higher salary, on average, than the average salary at “First Bank Corporation”.**

select company\_name

from works

group by company\_name

having avg (salary) > (select avg (salary)

from works

where company\_name = ’First Bank Corporation’)

# than some

**3.40 Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.**

select distinct T.name

from instructor as T, instructor as S

where S.dept\_name = 'Biology'

T.salary > S.salary;

**OR**

select name

from instructor

where salary > some (select salary

from instructor

where dept\_name = 'Biology');

# the same as

**Ass2 3.8b. Find the ID of each customer who lives on the same street and in the same city as customer '12345'.**

select F.ID

from customer F join customer S using(customer\_street, customer\_city)

where S.ID = ’12345’;

Parts (pid, pname, color, weight, city)

Projects (jid, jname, city)

Shipments (sid, pid, jid, qty): supplier sid supplies some part pid for some project jid with the qty

**2022mid b. get parts name for parts, which are only supplied for projects located in the same city as producing city of the parts(in other words, only supplied for local projects)**

select distinct(pname)

from Parts p, Projects j, Shipments sh

where p.jid = sh.jid and sh.pid = j.pid

and p.city = j.city;

emp (ename, street, city)

work (ename, company\_name, salary)

mgr (ename, mgr\_name)

**2022mid 2a. find all employee names in the database who live in the same cities and on the same streets as do their managers**

mgr\_info ← mgr ⋈mgr.mgr\_name = emp.ename emp

∏emp\_ename(σmgr\_info.city = emp.city Λ mgr\_info.street = emp.street (mgr\_info ⋈mgr\_info.ename = emp.ename emp)

select e.ename

from (mgr natural join emp m on mgr.mgr\_name = m.ename)

join emp e on mgr.ename = e.name

and m.city = e.city and m.street = e.street;

# self join

emp-super (person, supervisor)

**3.21 Find the supervisor of the supervisor of “Bob”**

Select s.supervisor

from emp-super e, emp-super s

where e.supervisor = s.person

and e.person=‘Bob’

# Trigger

**“BEFORE INSERT”触发器在插入操作之前执行，允许修改或验证即将插入的数据。**

**“AFTER INSERT”触发器在插入操作之后执行，用于后续处理或与其他表进行交互。**

**5.49 convert blank grades to null.**

create trigger setnull\_trigger before update on takes

for each row

begin

if new.grade = ‘ ‘

then set new.grade = null;

end if;

end;

**5.50**

create trigger credits\_earned after update on takes

for each row

begin

if new.grade <> 'F' and new.grade is not null

and (old.grade = 'F' or old.grade is null)

then update student

set tot\_cred= tot\_cred + (select credits

from course

where course.course\_id= new.course\_id)

where student.ID = new.ID;

end if;

end;

**Ass4 5.6**

define a view branch\_cust as follows:

create view branch\_cust as (

select branch\_name, customer\_name

from depositor, account

where depositor.account\_number = account.account\_number);

account (account\_number, branch\_name, balance )

depositor (customer\_name, account\_number)

**new.account\_number, new.customer\_name -> branch\_name, customer\_name**

create trigger insert\_depositor\_trigger after insert on depositor

for each row

begin

insert into branch\_cust (branch\_name, customer\_name)

values ((select branch\_name

from account

where account\_number = new.account\_number), new.customer\_name);

end;

**new.account\_number, new.customer\_name -> branch\_name, customer\_name**

create trigger insert\_account\_trigger after insert on account

for each row

begin

insert into branch\_cust (branch\_name, customer\_name)

select new.branch\_name, depositor.customer\_name

from depositor

where depositor.account\_number = new.account\_number;

end;

# Recursion

**5.55 find which courses are a prerequisite, whether directly or indirectly, for a specific course**

with recursive rec\_prereq(course\_id, prereq\_id) as (

select course\_id, prereq\_id

from prereq

union

select rec\_prereq.course\_id, prereq.prereq\_id,

from rec\_rereq, prereq

where rec\_prereq.prereq\_id = prereq.course\_id

)

select ∗ from rec\_prereq;

c\_p

select course\_id, prereq\_id

from prereq

c\_pp

select c\_p.course\_id, p.prereq\_id

from c\_p, prerep p

where c\_p.prereq\_id = p.course\_id

c\_ppp

select c\_pp.course\_id, p.prereq\_id

from c\_pp, prerep p

where c\_pp.prereq\_id = p.course\_id

Sailors (sid, sname, rating, age)

Boats (bid,bname, color)

Reserves (sid, bid, day)

**2024mid d. Get the sailor names who did not reserve any boat which was reserved by either ‘Tom’ or ‘Eric’.**

select S.sid, S.sname

from Sailors S join Reserves using (sid)

where bid not in (

select bid

from Sailors join Reserves using (sid)

where (sname = ‘Tom’ or sname = ‘Eric’)

);

**OR**

select S.sid, S.sname

from Sailors S join Reserves R using (sid)

where not exists (

select \*

from Sailors join Reserves using (sid)

where (sname = ‘Tom’ or sname = ‘Eric’)

and Reserves.bid = R.bid

);

**2024mid g. ( Using relational algebra ) Find sailor names who only reserved 'Red boat’ but no**

**any other colors.**

select sname

from sailors

join reserves on sailors.sid = reserves.sid

join boats on reserves.bid = boats.bid

where boats.color = 'red'

and sname not in (

select sname

from sailors

join reserves on sailors.sid = reserves.sid

join boats on reserves.bid = boats.bid

where boats.color <> 'red'

);

r

|  |  |  |
| --- | --- | --- |
| ID | name | no |
| 1 | aa | 1111 |
| 2 | bb | 2222 |
| 3 | cc | 3333 |
| 4 | dd | 4444 |

Create view v as select ID, name from r where ID <= 20;

Insert into v values (6, ‘Tom’);

Insert into v values (30, ‘Jenny’);

r

|  |  |  |
| --- | --- | --- |
| ID | name | no |
| 1 | aa | 1111 |
| 2 | bb | 2222 |
| 3 | cc | 3333 |
| 4 | dd | 4444 |
| 6 | Tom | null |
| 30 | Jenny | null |

Select \* from v;

|  |  |  |
| --- | --- | --- |
| ID | name | no |
| 1 | aa | 1111 |
| 2 | bb | 2222 |
| 3 | cc | 3333 |
| 4 | dd | 4444 |
| 6 | Tom | null |

Student (S\_NO, Sname, Ssex, Sage, D\_NO , Sclass)

Course (C\_NO, Cname, Chours, Credit, T\_NO)

SC (S\_NO, C\_NO, Score)

找出张三同学成绩最低的课程号

Select C\_NO

From SC, Student S

Where S.Sname = "张三"

and S.S\_NO = SC.S\_NO

and Score <= all ( Select Score

From SC

Where S\_NO = S.S\_NO );