目录

[Mysql day05 1](#_Toc501036270)

[1 select查询结构 1](#_Toc501036271)

[2 子查询 2](#_Toc501036272)

[2.1 条件子查询 2](#_Toc501036273)

[2.2 from子查询(行内视图) 4](#_Toc501036274)

[2.3 字段列表中的子查询 5](#_Toc501036275)

[3 多表查询 6](#_Toc501036276)

[3.1 标准连接语法 8](#_Toc501036277)

[4 select 查询结构 11](#_Toc501036278)

[5 事务 11](#_Toc501036279)

[5.1 事务特性:ACID 11](#_Toc501036280)

[5.2 mysql事务操作 12](#_Toc501036281)

[5.2.1 开启事务 12](#_Toc501036282)

[5.2.2 提交事务 12](#_Toc501036283)

[5.2.3 回滚事务 12](#_Toc501036284)

[5.3 事务隔离 13](#_Toc501036285)

[5.3.1 数据访问冲突: 14](#_Toc501036286)

[5.3.2 事务隔离级别 14](#_Toc501036287)

[6 作业 17](#_Toc501036288)

Mysql day05

# select查询结构

select

distinct

from

where

group by

having

order by

# 子查询

* 条件子查询
* from子查询
* 查询字段列表中的子查询

## 条件子查询

查询结果,作为另一个查询的过滤条件

* 单值子查询  
  =   
  <>  
  >  
  <
* 多值子查询  
  in  
  > all 大于最大值  
  > any 大于最小值
* 多列子查询  
  (a,b) in 子查询

条件子查询

set names gbk;

use hr;

-- 1. 拿最低工资的员工

-- 第一步查最低工资值,第二步用工资过滤员工

select min(salary)

from employees;

select employee\_id,first\_name,salary

from employees

where salary=(

select min(salary)

from employees

);

-- 2. 工资低于平均工资

-- 第一步平均工资,第二步用平均工资过滤员工

select avg(salary)

from employees;

select employee\_id,first\_name,salary

from employees

where salary<(

select avg(salary)

from employees

);

-- 3. 只有一个人的部门中的员工信息

-- 第一步分组查询部门人数过滤只有一人的部门

-- 第二部用部门过滤查询员工

select department\_id dept

from employees

where department\_id is not null

group by dept

having count(\*)=1;

select employee\_id,first\_name,salary,

department\_id

from employees

where department\_id in(

select department\_id dept

from employees

where department\_id is not null

group by dept

having count(\*)=1

);

-- 4. 每个部门拿最高工资的员工信息

-- 第一步求每个部门的最高工资

-- 第二步用部门和最高工资值过滤员工

select department\_id max(salary)

from employees

where department\_id is not null

group by department\_id;

select employee\_id,first\_name,salary,

department\_id

from employees

where (department\_id,salary) in(

select department\_id,max(salary)

from employees

where department\_id is not null

group by department\_id

);

## from子查询(行内视图)

* select ... from (select...) t
* mysql中,必须起别名

行内视图

-- 5. 手下人数最多的人，查询其个人信息

-- 第一步分组查手下人数

-- 第二步得到人数最大值

-- 第三步分组查手下人数,用最大人数过滤

-- 第四步用员工id过滤

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id;

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t;

select manager\_id

from employees

where manager\_id is not null

group by manager\_id

having count(\*)=(

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t

);

select employee\_id,first\_name,salary,

department\_id

from employees

where employee\_id in(

select manager\_id

from employees

where manager\_id is not null

group by manager\_id

having count(\*)=(

select max(c) from (

select manager\_id,count(\*) c

from employees

where manager\_id is not null

group by manager\_id

) t -- mysql行内视图必须有别名

)

);

## 字段列表中的子查询

select a,b,c,(select ...) from ...

字段列表中的子查询

-- 6. 员工信息后面添加平均工资值

select employee\_id,first\_name,salary,

(

select avg(salary)

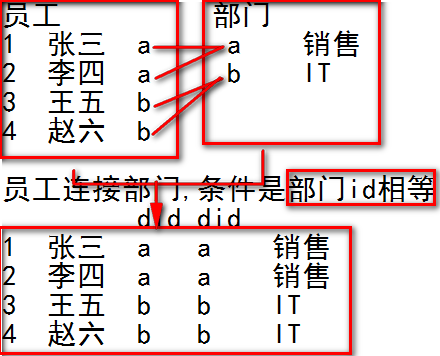
from employees

) avg -- 字段别名

from employees;

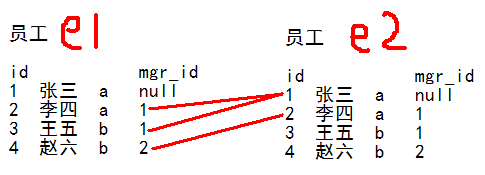
# 多表查询

以指定条件,将两张表连接成一张表,再从这张表查询



* select t1.c1,t1.c2,t1.c3,  
   t2.c1,t2.c4  
  from tb1 t1, tb2 t2 -- 多张表  
  where t1.did=t2.did -- 连接条件
* 自连接

将一张表看做是两张表来连接



多表连接查询

-- 7. 部门表

select \* from departments;

-- 8. 员工,显示所在的部门名

select

e.employee\_id,

e.first\_name,

e.salary,

d.department\_id,

d.department\_name

from employees e,departments d

where

e.department\_id=d.department\_id;

-- 9. 部门,显示部门经理first\_name

select

d.department\_id,

d.department\_name,

e.first\_name

from departments d, employees e

where

d.manager\_id=e.employee\_id;

-- 10. 地区表

select \* from locations;

-- 11. 部门, 部门所在城市

select

d.department\_id,

d.department\_name,

l.city

from departments d, locations l

where d.location\_id=l.location\_id;

-- 12. 员工,显示他的主管first\_name

select

e1.employee\_id,

e1.first\_name,

e1.salary,

e2.first\_name mgr\_name

from employees e1,employees e2

where

e1.manager\_id=e2.employee\_id;

## 标准连接语法

前面的表连接语法,是非标准语法,但多数数据库都支持这种语法,在做外连接查询时,不同数据库都有各自的扩展语法

标准语法,解决了外连接语法不兼容问题

* select ...  
  from a inner join b on 条件  
   inner join c on 条件  
   inner join d on 条件
* inner 可以省略
* 左外连接、右外连接  
  左外连接: 左侧表条件以外的数据也查询出来  
  右外连接: 右侧表条件以外的数据也查询出来  
  select ...  
  from a left outer join b on 条件
* outer 可以省略
* 全外连接: full outer join

标准表连接语法

-- 13. 107个员工, 显示部门名,没部门是null

select

e.employee\_id,

e.first\_name,

e.salary,

d.department\_id,

d.department\_name

from

employees e

left join departments d

on e.department\_id=d.department\_id;

-- 14. 员工,部门名,城市,部门经理,员工的主管

select

e1.employee\_id,e1.first\_name,

d.department\_name,

l.city,

e2.first\_name dept\_mgr,

e3.first\_name mgr\_name

from

employees e1

join departments d

on e1.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

join employees e2

on d.manager\_id=e2.employee\_id

join employees e3

on e1.manager\_id=e3.employee\_id;

表连接练习

-- 15. 按城市分组，计算每个城市的员工数量

select

l.city, count(\*)

from

employees e

join departments d

on e.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

group by l.city;

-- 16. Seattle 市所有的员工信息

select

e.employee\_id,e.first\_name,e.salary,

l.city

from

employees e

join departments d

on e.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

where l.city='Seattle';

-- 17. 在 Seattle 拿最低工资的人

select

min(salary)

from

employees e

join departments d

on e.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

where l.city='Seattle';

select

e.employee\_id,e.first\_name,e.salary,

l.city

from

employees e

join departments d

on e.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

where l.city='Seattle' and salary=(

select

min(salary)

from

employees e

join departments d

on e.department\_id=d.department\_id

join locations l

on d.location\_id=l.location\_id

where l.city='Seattle'

);

# select 查询结构

select

distinct

from

join on

left join on

right join on

where

group by

having

order by

# 事务

* 事务由多个数据操作(增删改查)组合
* 事务是数据操作的最小单元
* 事务中所有操作全部成功,事务整体成功
* 当事务中一项操作失败,事务整体失败

例如,游戏中 A 账户向 B 账户转账

1. A 更新,减掉金额
2. B 更新,加金额失败(断电),转账事务要整体失败,回退到以前的状态

## 事务特性:ACID

A - 原子性 Atomic

C - 一致性 Consistency

转账前 a+b = 100

转帐后 a+b = 100

I - 隔离性 Isolation

一个事物进行中时，

另一事物不能操作数据

D - 持久性 Durancy

提交事务之后，

数据持久生效

## mysql事务操作

### 开启事务

* begin
* start transaction
* set autocommit=0  
  设置关闭自动提交

### 提交事务

commit

### 回滚事务

rollback

事务测试

set names gbk;

use test;

desc kecheng;

-- day02 - 多对多关系 - kecheng表

|  |  |
| --- | --- |
| 会话1 | 会话2 |
| begin; | begin; |
| insert into  kecheng(name)  values('a'),('b');  update kecheng  set name='x'  where id=1;  select \* from  kecheng; |  |
|  | select \* from  kecheng; |
| commit; |  |
|  | select \* from  kecheng; |
|  | commit; |
|  | select \* from  kecheng; |
| 会话一 | 会话二 |
| bigin; | bigin; |
| insert into  kecheng(name)  values('c'),('d');  update kecheng  set name='y'  where id=1;-- 锁定一行数据  select \* from  kecheng; |  |
|  | -- 不能修改加锁数据,除非其它事务结束  update kecheng  set name='k'  where id=1; |
| rollback; |  |
| select \*  from kecheng; |  |
|  | update kecheng  set name='k'  where id=1; |
|  | commit; |

## 事务隔离

事务完全隔离,一个事务完成,另一个事务才能执行,效率低

事务不隔离,会有数据访问冲突,数据不安全

数据库当中,可以设置事务的隔离性,来兼顾安全和性能

### 数据访问冲突:

* 脏读(一般不允许发生)

一个会话未提交的数据,被另一个会话读取

* 不可重复读

同一行数据,再次执行相同查询,与第一次查询结果不一致

* 幻读
  + 其它会话新插入并提交的数据,查询不到
  + 其它会话删除并提交的数据,仍然能查询到

### 事务隔离级别

set tx\_isolation='READ-UNCOMMITTED';

set tx\_isolation='read-committed';

set tx\_isolation='repeatable-read';

set tx\_isolation='serializable';

READ-UNCOMMITTED: 脏读,不可重复读,幻读

read-committed: 不可重复读,幻读

repeatable-read: 幻读

serializable: x

隔离级别测试

|  |  |
| --- | --- |
| 会话1 | 会话2 |
| rollback; | rollback; |
| set tx\_isolation='READ-UNCOMMITTED' | |
| begin; | begin; |
|  | select \* from  kecheng; |
| insert into  kecheng(name)  values('e'),('f');  update kecheng  set name='h'  where id=1;  select \* from  kecheng; |  |
|  | select \* from  kecheng; |
| rollback; |  |
|  | select \* from  kecheng; |
|  | rollback; |
| 会话1 | 会话2 |
| set tx\_isolation='read-committed'; | |
| begin; | begin; |
|  | select \* from  kecheng; |
| insert into  kecheng(name)  values('g'),('h');  update kecheng  set name='w'  where id=1;  select \* from  kecheng; |  |
|  | select \* from  kecheng; |
| commit; |  |
|  | select \* from  kecheng;-- 不一致,不可重复读 |
|  | commit; |
| 会话1 | 会话2 |
| set tx\_isolation='repeatable-read'; | |
| begin; | begin; |
|  | select \* from  kecheng; |
| insert into  kecheng(name)  values('i'),('j');  update kecheng  set name='v'  where id=1;  select \* from  kecheng; |  |
|  | select \* from  kecheng; |
| commit; |  |
|  | select \* from  kecheng;-- 事务日志查询 |
|  | commit; |
|  | select \* from  kecheng; |
| 会话1 | 会话2 |
| rollback;  begin; | rollback;  begin; |
|  | select \* from  kecheng; |
| insert into  kecheng(name)  values('k'),('l'); |  |
| commit; |  |
|  | select \* from  kecheng; |
|  | selct count(\*) from  kecheng; |
|  | update kecheng  set name=  concat('\*',name); |
|  | select \* from  kecheng; |
|  | commit; |
| 会话1 | 会话2 |
| rollback;  begin; | rollback;  begin; |
|  | select \* from  kecheng; |
| delete from kecheng  where name  in('\*k','\*l','\*i'); |  |
| commit; |  |
|  | select \* from  kecheng; |
|  | update kecheng set  name=  concat(name,'\*'); |
|  | select \* from  kecheng; |
|  | commit; |
|  | select \* from  kecheng; |

# 作业

只有一个下属的主管信息

select employee\_id,first\_name,salary

from employees

where employee\_id in(

select manager\_id

from employees

where manager\_id is not null

group by manager\_id

having count(\*)=1

);

平均工资最高的部门编号

select department\_id,round(avg(salary),2) c

from employees

where department\_id is not null

group by department\_id

having c=(

select max(c) from(

select department\_id,round(avg(salary),2) c

from employees

where department\_id is not null

group by department\_id

) T

);

平均工资最低的工种，查询做这些工作的人

select employee\_id,first\_name,salary,job\_id

from employees

where job\_id=(

select job\_id

from employees

group by job\_id

having round(avg(salary))=(

select min(s) from(

select job\_id,round(avg(salary)) s

from employees

group by job\_id

) t

)

);

4.2.14 用户地址表 (tb\_address)

文档中少一个 user\_id

4.2.13 订单物流表(tb\_order\_shipping)

order\_id既是主键也是外键