2-21关联查询和数据分组

1. 课堂案例

课堂讲解

一、上节回顾

采集数据

二、学习目标

关联查询

数据分组

三、教学过程描述

1、关联查询

1-1、内联结

将两个表中存在联结关系的字段符合联结关系的那些记录形成记录集的联结。

格式

select A.字段,B.字段 from A inner join B on A.id = B.id

或者

Select A.字段,B.字段 from A, B where A.id = B.id

1、自连接

自连接(self join)是SQL语句中经常要用的连接方式，使用自连接可以将自身表的一个镜像当作另一个表来对待，即将一张表看成多张表来做连接，从而能够得到一些特殊的数据。

例如：在emp中的每一个员工都有自己的mgr(经理)，并且每一个经理自身也是公司的员工，自身也有自己的经理。下面我们需要将每一个员工自己的名字和经理的名字都找出来。这时候我们该怎么做呢？

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos

WORKER表中的MANAGER_ID等于
MANAGER表中的EMPLOYEE_ID

```
SELECT worker.last_name || ' works for '
       || manager.last_name
FROM   employees worker, employees manager
WHERE  worker.manager_id = manager.employee_id ;
```

WORKER.LAST_NAME||'WORKSFOR'||MANAGER.LAST_NAME
Kochhar works for King
De Haan works for King
Mourgos works for King
Zlotkey works for King
Hartstein works for King
Whalen works for Kochhar
Higgins works for Kochhar
Hunold works for De Haan
Ernst works for Hunold

19 rows selected.

-- 自连接

```
create table t_emp(
    emp_id int primary key,
    emp_name varchar(20),
    manager_id int references t_emp(emp_id)
);
```

```
insert into t_emp values(1, '飞刀客', null);
insert into t_emp values(2, '中神通', 1);
insert into t_emp values(3, '踏雪', 1);
insert into t_emp values(4, '挨刀妹', 1);

select * from t_emp;
-- 查询飞刀客手下员工
select worker.emp_name 员工 ,mgr.emp_name 领导
from t_emp worker, t_emp mgr
where worker.manager_id = mgr.emp_id;
```

自然连接：natural join

在连接条件中使用等于=运算符比较被连接列的列值，但会删除连接表中重复列。

```
SELECT * FROM e_order o NATURAL JOIN e_user us ;
```

2、等值连接：=

使用等于=比较连接列的列值，在查询结果中列出接连表中的所有列，包括其中的重复列。

```
-- inner join 内连接 只连接匹配的行
select u.*, o.order_code from e_user u inner join e_order o on u.id = o.u_id;

-- 上面的等效下面的语句
select u.*, o.order_code from  e_user u join e_order o where u.id = o.u_id;

select u.*, o.order_code from  e_user u, e_order o where u.id = o.u_id;
```

3、非等值连接：!=、<>

在连接条件中，可以使用其他比较运算符，比较被连接列的列值，如：<、>、!= <>等。

```
select * from e_user u inner join e_order o where u.id <> o.u_id;
```

1-2、外联

1、外左联

联结A、B表的意思就是将表A中的全部记录和表B中联结的字段与表A的联结字段符合联结条件的那些记录形成的记录集的联结

Select A.字段,B字段 from A Left JOIN B ON A.id = B.id

结果

先查找左表的内容，然后通过关联条件，显示右表相关的内容，

若右表不存在对应内容，输出null

2、外右联

和左联差不多

Select A.字段 from A Right JOIN B ON A.id = B.id

先查找右表的内容，然后通过关联条件，显示左表相关的内容，

若左表不存在对应内容，输出null

1-3、交叉连接：cross join

笛卡尔连接

用户表4条记录

id	t_name
1	lis
2	tom
3	admin
4	mary

订单表6条记录

id	u_id	order_code	create_date
1	1	电视剧	2009-09-22
2	1	空调	2009-09-22
3	1	笔记本	2009-09-22
4	2	手机	2009-09-22
5	2	电池	2009-09-22
6	3	薯片	2009-09-22

id	t_name	id1	u_id	order_code	create_date
1	lis	1	1	电视剧	2009 09 22
2	tom	1	1	电视剧	2009-09-22
3	admin	1	1	电视剧	2009-09-22
4	mary	1	1	电视剧	2009-09-22
1	lis	2	1	空调	2009-09-22
2	tom	2	1	空调	2009-09-22
3	admin	2	1	空调	2009 09 22
4	mary	2	1	空调	2009-09-22
1	lis	3	1	笔记本	2009-09-22
2	tom	3	1	笔记本	2009-09-22
3	admin	3	1	笔记本	2009-09-22
4	mary	3	1	笔记本	2009-09-22
1	lis	4	2	手机	2009 09 22
2	tom	4	2	手机	2009-09-22
3	admin	4	2	手机	2009-09-22
4	mary	4	2	手机	2009-09-22
1	lis	5	2	电池	2009-09-22
2	tom	5	2	电池	2009-09-22
3	admin	5	2	电池	2009 09 22
4	mary	5	2	电池	2009 09 22
1	lis	6	3	薯片	2009-09-22
2	tom	6	3	薯片	2009-09-22
3	admin	6	3	薯片	2009-09-22
4	mary	6	3	薯片	2009-09-22

4*6 = 24

select * from e_user,e_order;

笛卡尔积又叫笛卡尔乘积,是一个叫笛卡尔的人提出来的。简单的说就是两个集合相乘的结果。

```
SELECT COUNT(*) FROM e_user;
SELECT COUNT(*) FROM e_order,e_user;
```

1.笛卡尔积在下列情况下产生:

1）忽略了一个连接条件；

2）一个连接条件失效；

3）第一张表的所有行和第二张表的所有行连接；

2.为了避免笛卡尔积的产生，通常包含一个有效连接条件的WHERE子句。

2、限定使用二义性列名

2-1、表前缀在多张表中限定列名 ；

2-2、用表前缀提高系统性能；

2-3、名区分不同表中同名的列；

3、数据分组

3-1、SQL深入介绍

1、DELETE、TRUNCATE、DROP的区别

	删除表数据	删除表结构	可以回滚	速度
Delete	是	否	可以	很慢
Truncate	是	否	不可以	相对很快
Drop	是	是	不可以	最快

TRUNCATE和DELETE都是删除表数据;

但是TRUNCATE相当于初始化,比如如果有自增长ID,用TRUNCATE删除后,自增长ID会从1开始,而DELETE则不会;

DROP用于删除表,数据库等;

使用truncate删除表后，自增长id从1开始

例子:truncate TABLE c_address;

2、定义列名的别名

-- 别名

可以使用‘AS’关键字:

```
SELECT NAME AS 名称,sex AS 性别,age AS 年龄 FROM tb_emp;
```

也可以省略‘AS’:

```
SELECT NAME  名称,sex  性别,age  年龄 FROM tb_emp;
```

如果列名之间有多个空格，需要使用单引号括起来。

```
SELECT NAME '名    称',sex  '性    别',age  '年龄' FROM tb_emp;
```

注意:

可以省略 as 关键字

如果别名中使用特殊字符，或者是强制大小写敏感，或有空格时，都可以通过为别名添加引号实现

。

3、算术表达式

-- 算数运算符

1.对数值型数据列、变量、常量，可以使用算数操作符创建表达式(+ -* /)。

2.对日期型数据列、变量、常量，可以使用部分算数操作符创建表达式(+ -)。

例如:

```
SELECT NOW()+10;
```

任何值与Null相操作(比如加减乘除)，结果都是Null

Select Null + 5 As Result;

3.运算符不仅可以在列和常量之间进行运算，也可以在多列之间进行运算。

例如:

```
SELECT 12+16 AS 加法;
```

```
SELECT (1+3)*2;
```

select id as '用户 编号', t_name as 用户名称, t_salary as 月薪, t_salary*12 as 年薪 from e_user;

-- date_format() 对时间类型进行格式化的函数

```
SELECT DATE_FORMAT(birthday,'%Y/%m/%d') 生日 FROM e_user;
```

-- 表示 日加 10

select date_format(t_birthday+10, '%Y-%m-%d') from e_user;

4、合并函数

完整语法:

GROUP_CONCAT(expr)

GROUP_CONCAT()是MySQL数据库提供的一个函数，通常跟GROUP BY一起用

SELECT id,GROUP_CONCAT(t_name) FROM e_user GROUP BY t_password;

5、排序函数

升序:

```
SELECT * FROM tb_emp ORDER BY age ASC;
```

降序:

select * from tb_emp order by age desc;

语法

以下是 SQL SELECT 语句使用 ORDER BY 子句将查询数据排序后再返回数据:

SELECT field1, field2,...fieldN table_name1, table_name2...

ORDER BY field1, [field2...] [ASC [DESC]]

你可以使用任何字段来作为排序的条件，从而返回排序后的查询结果。

你可以设定多个字段来排序。

你可以使用 ASC 或 DESC 关键字来设置查询结果是按升序或降序排列。

默认情况下，它是按升序排列。

你可以添加 WHERE...LIKE 子句来设置条件。

3-2、分组过滤

使用having子句过滤分组结果：

行已经被分组,使用了分组函数满足having子句中条件的分组将被显示。

3-2-1、使用GROUP BY 子句

```
/**
Group by分组语句
获取不同性别的用户数量
**/
SELECT sex,COUNT(sex) 数量 FROM tb emp GROUP BY sex;
```

3-2-2、使用HAVING子句

having一般跟在group by之后，执行记录组选择的一部分来工作的。

where则是执行所有数据来工作的。

再者having可以用聚合函数，如having sum(qty)>1000

3-2-3、数据去重复

分组函数可以对行集进行操作，并且为每组给出一个结果。

使用group by

column1，column2,..按columm1,column2进行分组，即column1,column2组合相同的值为一个组。

分组函数语法：

SELECT [column,] group_function(column), ... FROM table

[WHERE condition]

[GROUP BY column]

[ORDER BY column];

分组函数使用准则：

DISTINCT 使函数只考虑非重复值，

ALL则考虑包括重复值在内的所有值。默认为ALL. □□

带有expr参数的函数的数据类型可以为CHAR,VARCHAR2,NUMBER,DATE。

所有分组函数都忽略空值。

3-2-3-1、DISTINCT: 去除重复数据

```
SELECT DISTINCT(t_password) FROM e_user;
```

注意:

多个字段同时生效才能去重复

select distinct name, id from table

select id, distinct name from table

很遗憾，除了错误信息你什么也得不到，distinct必须放在开头。难到不能把distinct放到where条件里？能，照样报错。。。。。。

防止表中出现重复数据

在MySQL数据表中设置指定的字段为 PRIMARY KEY（主键） 或者 UNIQUE（唯一）索引来保证数据的唯一性。

统计重复数据

将统计表中 first_name 和 last_name的重复记录数:

```
mysql> SELECT COUNT(*) as repetitions, last_name, first_name
    -> FROM person_tbl
    -> GROUP BY last_name, first_name
    -> HAVING repetitions > 1;
```

以上查询语句将返回 person_tbl 表中重复的记录数。

一般情况下，查询重复的值，请执行以下操作:

确定哪一列包含的值可能会重复。

在列选择列表使用COUNT(*)列出的那些列。

在GROUP BY子句中列出的列。

HAVING子句设置重复数大于1。

过滤重复数据

过滤重复数据

如果你需要读取不重复的数据可以在 SELECT 语句中使用 DISTINCT 关键字来过滤重复数据。

```
mysql> SELECT DISTINCT last_name, first_name
    -> FROM person_tbl;
```

你也可以使用 GROUP BY 来读取数据表中不重复的数据:

```
mysql> SELECT last_name, first_name
    -> FROM person_tbl
```

```
    -> GROUP BY (last_name, first_name);
```

删除重复数据

删除重复数据

如果你想删除数据表中的重复数据，你可以使用以下的SQL语句：

```
mysql> CREATE TABLE tmp SELECT last_name, first_name, sex
    ->              FROM person_tbl;
    ->              GROUP BY (last_name, first_name, sex);
mysql> DROP TABLE person_tbl;
mysql> ALTER TABLE tmp RENAME TO person_tbl;
```

3-2-3-2、GROUP BY

```
SELECT t_name,t_password FROM e_user GROUP BY t_password;
```

3-3、GROUP BY &HAVING & WHERE区别?

相同点:having子句与where子句都是设定条件语句。

不同点:having条件表达式为聚合语句

having子句执行优先级低于聚合语句

HAVING子句可以让我们筛选成组后的各组数据.

例如:统计分组数据(group by)时要用到聚合函数，对分组数据再次判断需要使用到having。

使用where则会报错。如果上述关系均不需要，则直接使用where即可。

4、MySQL表连接运算顺序

INNER JOIN > CROSS JOIN > LEFT JOIN > RIGHT JOIN

5、子查询的使用

5-1、测试数据

```
CREATE TABLE e_user(
  id int NOT NULL PRIMARY KEY AUTO_INCREMENT,
  t_name varchar(10) DEFAULT NULL,
  t_job varchar(50) DEFAULT NULL,
  t_salary decimal(10, 2),
  t_birthday date,
  t_sex char(3) default '男'
```

)

insert into e_user values(null,'lis','Java软件工程师', 12000.0,'1980-09-22','男');
insert into e_user values(null,'admin','Android工程师', 10000.0,'1982-10-01','女');
insert into e_user values(null,'tom','DBA工程师', 8000.0,'1985-03-08','男');
insert into e_user values(null,' dave ','前端工程师', 9000.0,'1985-10-01','女');
insert into e_user values(null,'Amy','测试工程师', 7500.0,'1990-11-13','女');
insert into e_user values(null,'Kim','产品经理', 8500.0,'1991-12-05','男');
insert into e_user values(null,'Mary','项目经理', 9000.0,null,'女');

5-2、单行子查询

1）只返回一行结果；
2）使用单行比较运算符；

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to

-- 查询薪资大于平均的员工信息

select * from e_user where t_salary >(select avg(t_salary) from e_user);

id	t_name	t_job	t_salary	t_birthday	t_sex
1	lis	Java软件工程师	12000	1980-09-22	男
2	admin	Android工程师	10000	1982-10-01	女

5-3、多行子查询

由于列子查询返回的结果集是 N 行一列，因此不能直接使用 = > < >= <= <> 这些比较标量结果的操作符。

在列子查询中可以使用 IN、ANY、SOME 和 ALL 操作符：

1、IN：在指定项内，同IN(v1,v2,...)

2、ANY:与比较操作符联合使用，表示与子查询返回的任何值比较是否为true,则返回true

id	t_name	t_job	t_salary	t_birthday	t_sex
1	lis	Java软件工程师	12000	1980-09-22	男
2	admin	Android工程师	10000	1982-10-01	女
3	tom	DBA工程师	8000	1985-03-08	男
4	dave	前端工程师	9000	1985-10-01	女
5	Amy	测试工程师	7500	1990-11-13	女
6	Kim	产品经理	8500	1991-12-05	男
7	Mary	项目经理	9000	(Null)	女

-- 查询薪资小于 java软件工程师的 员工名和职位以及薪资

```
select u.id,u.t_name,u.t_job,u.t_salary from e_user u
where u.t_salary < ANY
                    (select t_salary from e_user where t_job ='Java软件工程师');
```

id	t_name	t_job	t_salary
2	admin	Android工程师	10000
3	tom	DBA工程师	8000
4	dave	前端工程师	9000
5	Amy	测试工程师	7500
6	Kim	产品经理	8500
7	Mary	项目经理	9000

3、ALL:与比较操作符联合使用，表示与子查询返回的所有值比较都为true，则返回true

id	t_name	t_job	t_salary	t_birthday	t_sex
1	lis	Java软件工程师	12000	1980-09-22	男
2	admin	Android工程师	10000	1982-10-01	女
3	tom	DBA工程师	8000	1985-03-08	男
4	dave	前端工程师	9000	1985-10-01	女
5	Amy	测试工程师	7500	1990-11-13	女
6	Kim	产品经理	8500	1991-12-05	男
7	Mary	项目经理	9000	(Null)	女

-- 查询薪资小于等于 前端工程师的 员工名和职位以及薪资,并排除前端工程师职位

```
select u.id,u.t_name,u.t_job,u.t_salary from e_user u
```

where u.t_salary <= ALL (select t_salary from e_user where t_job ='前端工程师')
AND t_job <> '前端工程师';

id	t_name	t_job	t_salary
3	tom	DBA工程师	8000
5	Amy	测试工程师	7500
6	Kim	产品经理	8500
7	Mary	项目经理	9000

四、小结

1、如果表设计的不好少用联表查询，否者影响性能

2、一般我们都是使用左联

2. 课堂练习

1、实现左右联表的操作

2.、完成一个学生考试的数据分组

3. 课后练习

完成项目的新闻表和分类表的关联查询

4. 资料扩展

1、UNION

查询两张表中的文章 id 号及标题，并去掉重复记录：

SELECT aid,title FROM table1 UNION SELECT bid,title FROM table2

UNION 查询结果说明

重复记录是指查询中各个字段完全重复的记录，如上例，若 title 一样但 id 号不一样算作不同记录。

第一个 SELECT 语句中被使用的字段名称也被用于结果的字段名称，如上例的 aid。

各 SELECT 语句字段名称可以不同，但字段属性必须一致。

2、UNION ALL

查询两张表中的文章 id 号及标题，并返回所有记录：

SELECT aid,title FROM table1 UNION ALL SELECT bid,title FROM table2