MySQL 数据库

课前默写

1、写出SQL语言的分类
2、写出DDL语句
3、写出DML语句
4、写出DQL语句

课程回顾

1、数据库概述
2、SQL语言分类
3、常用的SQL语句

今日内容

1、数据库简介
2、SQL语言
3、DQL数据查询

教学目标

1、掌握SQL语言数据的完整性
2、掌握内连接
3、掌握外连接
4、掌握子查询
5、熟练多表查询操作
6、了解SQL优化

第四章 数据的完整性

作用：保证用户输入的数据保存到数据库中是正确的。

确保数据的完整性 = 在创建表时给表中添加约束

完整性的分类：

- 实体完整性:
- 域完整性:
- 引用完整性:

4.1 实体完整性

实体：即表中的一行(一条记录)代表一个实体（entity）

实体完整性的作用：标识每一行数据不重复。

约束类型：

主键约束（primary key）

唯一约束(unique)

自动增长列(auto_increment)

4.1.1 主键约束（primary key）

注：每个表中要有一个主键。

特点：数据唯一，且不能为null

示例：

第一种添加方式：

```
CREATE TABLE student(

id int primary key,

name varchar(50)

);
```

第二种添加方式：此种方式优势在于，可以创建联合主键

```
CREATE TABLE student(

id int,

name varchar(50),

primary key(id)

);
```

```
CREATE TABLE student(

classid int,

stuid int,

name varchar(50),

primary key(classid, stuid)

);
```

第三种添加方式：

```
CREATE TABLE student(

id int,

name varchar(50)

);

ALTER TABLE student  ADD  PRIMARY KEY (id);
```

4.1.2 唯一约束(unique)

特点：数据不能重复。

```
CREATE TABLE student(

Id int primary key,

Name varchar(50) unique

);
```

4.1.3 自动增长列(auto_increment)

sqlserver数据库 (identity) oracle数据库(sequence)

给主键添加自动增长的数值，列只能是整数类型

```
CREATE TABLE student(

Id int primary key auto_increment,

Name varchar(50)

);
INSERT INTO student(name) values('tom');
```

4.2 域完整性

域完整性的作用：限制此单元格的数据正确，不对照此列的其它单元格比较

域代表当前单元格

域完整性约束：数据类型 非空约束（not null） 默认值约束(default)

check约束（mysql不支持）check(sex='男'or sex='女')

4.2.1 数据类型:（数值类型、日期类型、字符串类型）

4.2.2 非空约束：not null

```
CREATE TABLE student(

Id int pirmary key,

Name varchar(50) not null,

Sex varchar(10)

);

INSERT INTO student values(1,’tom’,null);
```

4.2.3 默认值约束 default

```
CREATE TABLE student(

Id int pirmary key,

Name varchar(50) not null,

Sex varchar(10) default ‘男’

);
insert intostudent1 values(1,'tom','女');

insert intostudent1 values(2,'jerry',default);
```

4.3 引用完整性（参照完整性）

```
外键约束：FOREIGN KEY
```

示例：

```
CREATE TABLE student(

sid int pirmary key,

name varchar(50) not null,

sex varchar(10) default ‘男’

);
```

```
create table score(
        id int,
        score int,
        sid int , -- 外键列的数据类型一定要与主键的类型一致
        CONSTRAINT fk_score_sid foreign key(sid) references student(id)
);
```

第二种添加外键方式。

```
ALTER TABLEscore1 ADD CONSTRAINT fk_stu_score FOREIGN KEY(sid) REFERENCES stu(id);
```

第五章 多表查询

多个表之间是有关系的，那么关系靠谁来维护?

多表约束：外键约束。

5.1 多表的关系

5.1.1 一对多关系

客户和订单，分类和商品，部门和员工.

一对多建表原则：在多的一方创建一个字段，字段作为外键指向一的一方的主键.

5.1.2 多对多关系

学生和课程:

多对多关系建表原则：需要创建第三张表,中间表中至少两个字段，这两个字段分别作为外键指向各自一方的主键.

5.1.3 一对一关系

在实际的开发中应用不多.因为一对一可以创建成一张表.

两种建表原则:

唯一外键对应：假设一对一是一个一对多的关系，在多的一方创建一个外键指向一的一方的主键，将外键设置为unique.

主键对应：让一对一的双方的主键进行建立关系.

5.2 多表查询

多表查询有如下几种：

1 合并结果集；UNION 、 UNION ALL

2 连接查询

2.1内连接 [INNER] JOIN ON

2.2外连接 OUTER JOIN ON

- 左外连接 LEFT [OUTER] JOIN
- 右外连接 RIGHT [OUTER] JOIN
- 全外连接（MySQL不支持）FULL JOIN

2.3自然连接 NATURAL JOIN

3 子查询

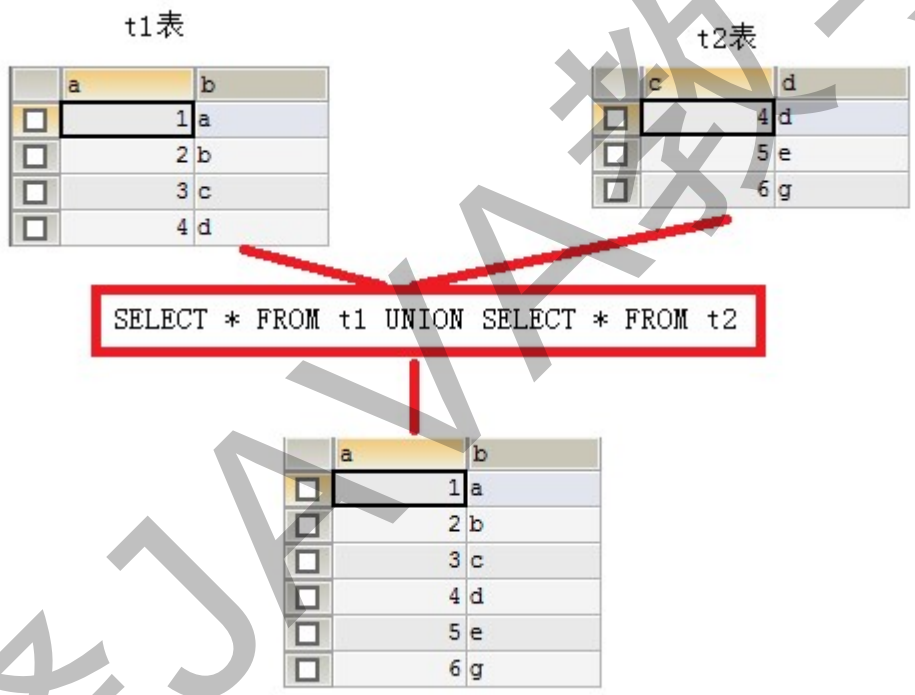
5.2.1 合并结果集

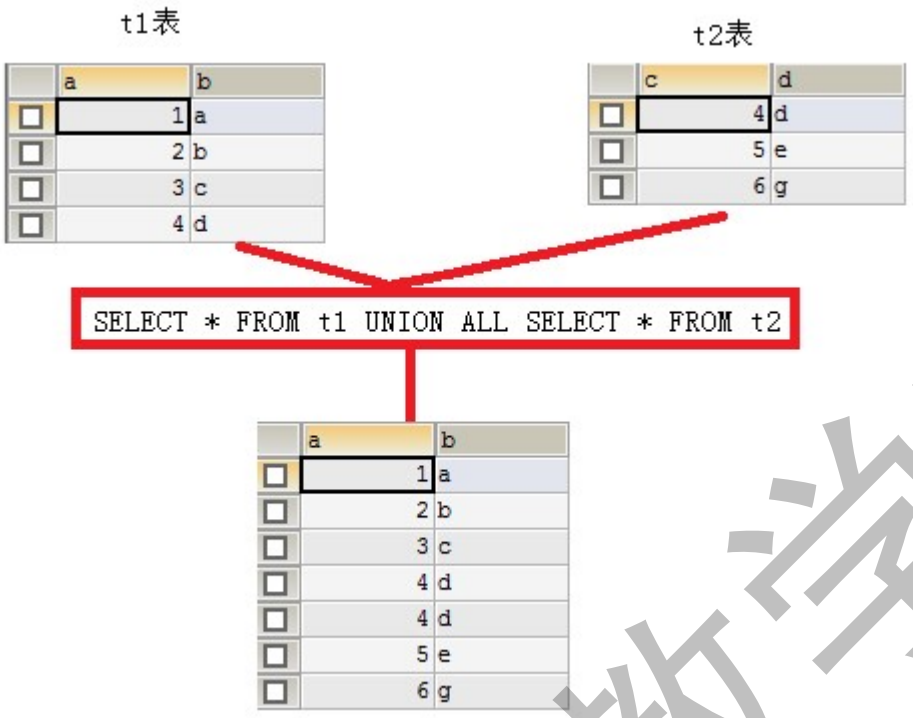
作用：合并结果集就是把两个select语句的查询结果合并到一起!

合并结果集有两种方式:

l UNION：去除重复记录，例如：SELECT* FROM t1 UNION SELECT * FROM t2;

l UNION ALL：不去除重复记录，例如：SELECT * FROM t1 UNION ALL SELECT * FROM t2。

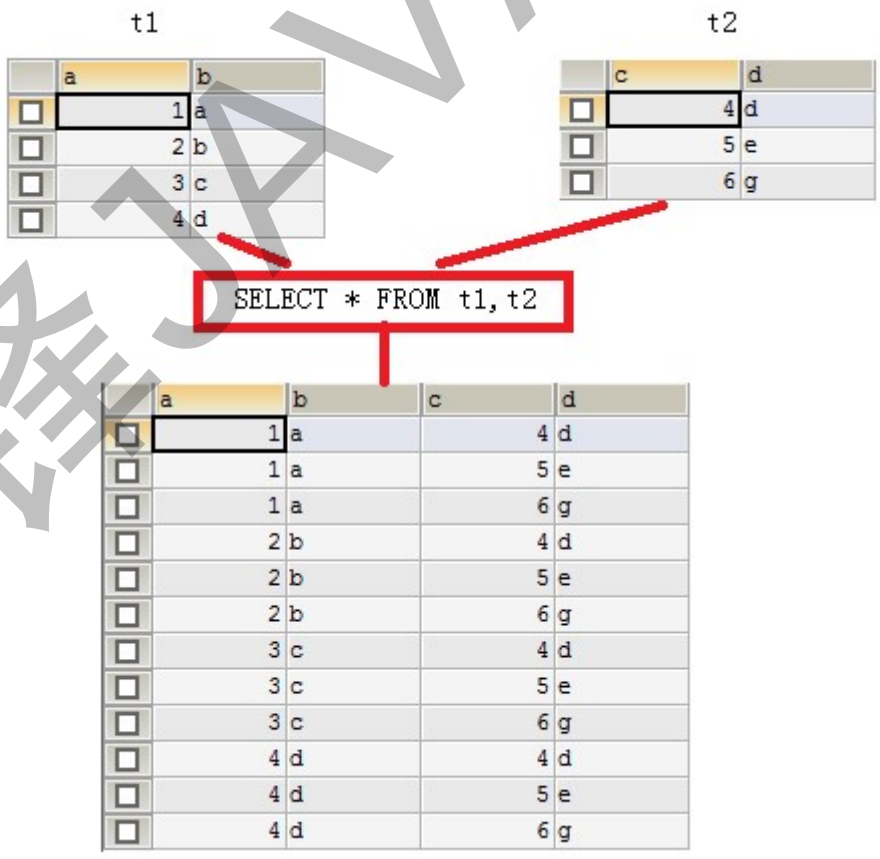




注意：被合并的两个结果：列数、列类型必须相同。

5.2.2 连接查询

连接查询就是求出多个表的乘积，例如t1连接t2，那么查询出的结果就是t1*t2。



连接查询会产生笛卡尔积，假设集合A={a,b}，集合B={0,1,2}，则两个集合的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。可以扩展到多个集合的情况。

那么多表查询产生这样的结果并不是我们想要的，那么怎么去除重复的，不想要的记录呢，当然是通过条件过滤。通常要查询的多个表之间都存在关联关系，那么就通过关联关系去除笛卡尔积。

示例 1：现有两张表

emp表

```
CREATE TABLE emp(
    empno       INT,
    ename       VARCHAR(50),
    job     VARCHAR(50),
    mgr     INT,
    hiredate    DATE,
    sal     DECIMAL(7,2),
    comm        decimal(7,2),
    deptno      INT
) ;
#添加数据SQL语句省略
```

dept表

```
CREATE TABLE dept(
    deptno      INT,
    dname       varchar(14),
    loc     varchar(13)
);
#添加数据SQL语句省略
```

执行如下SQL语句

```
select * from emp,dept;
```

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州

使用主外键关系做为条件来去除无用信息

```
SELECT * FROM emp,dept WHERE emp.deptno=dept.deptno;
```

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州

上面查询结果会把两张表的所有列都查询出来，也许你不需要那么多列，这时就可以指定要查询的列了。

```
SELECT emp.ename,emp.sal,emp.comm,dept.dname
FROM emp,dept
WHERE emp.deptno=dept.deptno;
```

ename	sal	comm	dname
张飞	24500.00	(NULL)	教研部
曾阿牛	50000.00	(NULL)	教研部
黄盖	13000.00	(NULL)	教研部
甘宁	8000.00	(NULL)	学工部
刘备	29750.00	(NULL)	学工部
诸葛亮	30000.00	(NULL)	学工部
周泰	11000.00	(NULL)	学工部
庞统	30000.00	(NULL)	学工部
黛绮丝	16000.00	3000.00	销售部
殷天正	12500.00	5000.00	销售部
谢逊	12500.00	14000.00	销售部
关羽	28500.00	(NULL)	销售部
韦一笑	15000.00	0.00	销售部
程普	9500.00	(NULL)	销售部

一：内连接

上面的连接语句就是内连接，但它不是SQL标准中的查询方式，可以理解为方言！

SQL标准的内连接为：

```
SELECT *
FROM emp e
INNER JOIN dept d
ON e.deptno=d.deptno;
```

内连接的特点：查询结果必须满足条件。

二：外连接

包括左外连接和右外连接，外连接的特点：查询出的结果存在不满足条件的可能。

a.左外连接

```
SELECT * FROM emp e
LEFT OUTER JOIN dept d
ON e.deptno=d.deptno;
```

左连接是先查询出左表（即以左表为主），然后查询右表，右表中满足条件的显示出来，不满足条件的显示NULL。

我们还是用上面的例子来说明。其中emp表中“张三”这条记录中，部门编号为50，而dept表中不存在部门编号为50的记录，所以“张三”这条记录，不能满足e.deptno=d.deptno这条件。但在左连接中，因为emp表是左表，所以左表中的记录都会查询出来，即“张三”这条记录也会查出，但相应的右表部分显示NULL。

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1015	张三	保洁员	1009	1999-12-31	80000.00	20000.00	50	(NULL)	(NULL)	(NULL)

b.右外连接

右连接就是先把右表中所有记录都查询出来，然后左表满足条件的显示，不满足显示NULL。例如在dept表中的40部门并不存在员工，但在右连接中，如果dept表为右表，那么还是会查出40部门，但相应的员工信息为NULL。

```
SELECT * FROM emp e
RIGHT OUTER JOIN dept d
ON e.deptno=d.deptno;
```

empno	ename	job	mgr	hiredate	sal	COMM	deptno	deptno	dname	loc
1007	张飞	经理	1009	2001-09-01	24500.00	(NULL)	10	10	教研部	北京
1009	曾阿牛	董事长	(NULL)	2001-11-17	50000.00	(NULL)	10	10	教研部	北京
1014	黄盖	文员	1007	2002-01-23	13000.00	(NULL)	10	10	教研部	北京
1001	甘宁	文员	1013	2000-12-17	8000.00	(NULL)	20	20	学工部	上海
1004	刘备	经理	1009	2001-04-02	29750.00	(NULL)	20	20	学工部	上海
1008	诸葛亮	分析师	1004	2007-04-19	30000.00	(NULL)	20	20	学工部	上海
1011	周泰	文员	1008	2007-05-23	11000.00	(NULL)	20	20	学工部	上海
1013	庞统	分析师	1004	2001-12-03	30000.00	(NULL)	20	20	学工部	上海
1002	黛绮丝	销售员	1006	2001-02-20	16000.00	3000.00	30	30	销售部	广州
1003	殷天正	销售员	1006	2001-02-22	12500.00	5000.00	30	30	销售部	广州
1005	谢逊	销售员	1006	2001-09-28	12500.00	14000.00	30	30	销售部	广州
1006	关羽	经理	1009	2001-05-01	28500.00	(NULL)	30	30	销售部	广州
1010	韦一笑	销售员	1006	2001-09-08	15000.00	0.00	30	30	销售部	广州
1012	程普	文员	1006	2001-12-03	9500.00	(NULL)	30	30	销售部	广州
(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	40	财务部	武汉

连接查询心得:

连接不限与两张表，连接查询也可以是三张、四张，甚至N张表的连接查询。通常连接查询不可能需要整个笛卡尔积，而只是需要其中一部分，那么这时就需要使用条件来去除不需要的记录。这个条件大多数情况下都是使用主外键关系去除。

两张表的连接查询一定有一个主外键关系，三张表的连接查询就一定有两个主外键关系，所以在大家不是很熟悉连接查询时，首先要学会去除无用笛卡尔积，那么就是用主外键关系作为条件来处理。如果两张表的查询，那么至少有一个主外键条件，三张表连接至少有两个主外键条件。

5.2.3 子查询

一个select语句中包含另一个完整的select语句。

子查询就是嵌套查询，即SELECT中包含SELECT，如果一条语句中存在两个，或两个以上SELECT，那么就是子查询语句了。

l 子查询出现的位置：

a. where后，作为条为被查询的一条件的一部分；

b. from后，作表；

l 当子查询出现在where后作为条件时，还可以使用如下关键字：

a. any

b. all

l 子查询结果集的形式：

a. 单行单列（用于条件）

b. 单行多列（用于条件）

c. 多行单列（用于条件）

d. 多行多列（用于表）

示例：

1. 工资高于JONES的员工。

分析：

查询条件：工资>JONES工资，其中JONES工资需要一条子查询。

第一步：查询JONES的工资

```
SELECT sal FROM emp WHERE ename='JONES';
```

第二步：查询高于甘宁工资的员工

```
SELECT * FROM emp WHERE sal > (第一步结果);
```

结果：

```
SELECT * FROM emp WHERE sal > (SELECT sal FROM emp WHERE ename='JONES');
```

2. 查询与SCOTT同一个部门的员工。

l 子查询作为条件

l 子查询形式为单行单列

分析：

查询条件：部门=SCOTT的部门编号，其中SCOTT 的部门编号需要一条子查询。

第一步：查询SCOTT的部门编号

```
SELECT deptno FROM emp WHERE ename='SCOTT';
```

第二步：查询部门编号等于SCOTT的部门编号的员工

```
SELECT * FROM emp WHERE deptno = (SELECT deptno FROM emp WHERE ename='SCOTT');
```

3. 工资高于30号部门所有人的员工信息

分析：

```
SELECT * FROMemp WHERE sal>(SELECT MAX(sal)FROM emp WHERE deptno=30);
```

查询条件：工资高于30部门所有人工资，其中30部门所有人工资是子查询。高于所有需要使用all关键字。

第一步：查询30部门所有人工资

```
SELECT sal FROM emp WHERE deptno=30;
```

第二步：查询高于30部门所有人工资的员工信息

```
SELECT * FROM emp WHERE sal > ALL (第一步)
```

结果：

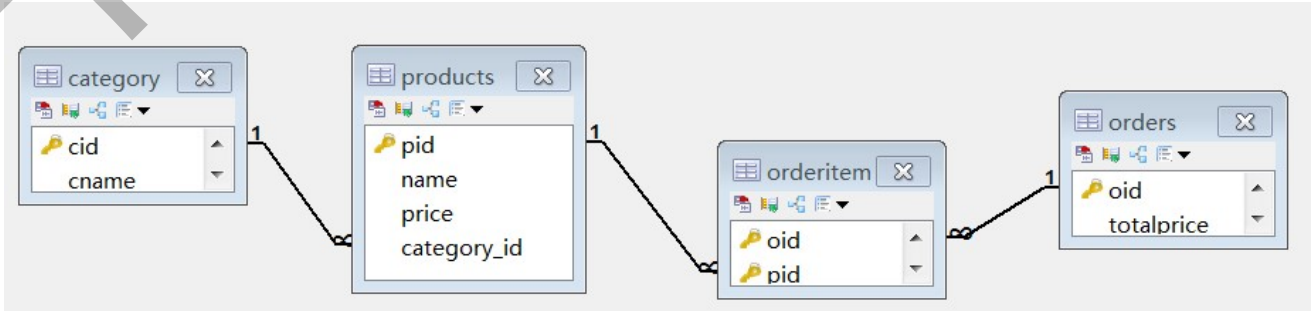
```
SELECT * FROM emp WHERE sal > ALL (SELECT sal FROM emp WHERE deptno=30)
```

l 子查询作为条件

l 子查询形式为多行单列（当子查询结果集形式为多行单列时可以使用ALL或ANY关键字）

第六章 综合练习

某网上商城数据库如下图所示



```
#一对多的实现
#创建分类表
create table category(
  cid varchar(32) PRIMARY KEY ,
  cname varchar(100)         #分类名称
);

# 商品表
CREATE TABLE `products` (
  `pid` varchar(32) PRIMARY KEY  ,
  `name` VARCHAR(40) ,
  `price` DOUBLE
);

#添加外键字段
alter table products add column category_id varchar(32);

#添加约束
alter table products add constraint product_fk foreign key (category_id) references category
(cid);

#多对多的实现
#订单表
create table `orders`(
  `oid` varchar(32) PRIMARY KEY ,
  `totalprice` double   #总计
);

# 订单项表
create table orderitem(
  oid varchar(50),-- 订单id
  pid varchar(50)-- 商品id
);
#联合主键（可省略）
alter table `orderitem` add primary key (oid,pid);

# 订单表和订单项表的主外键关系
alter table `orderitem` add constraint orderitem_orders_fk foreign key (oid) references
orders(oid);

# 商品表和订单项表的主外键关系
alter table `orderitem` add constraint orderitem_product_fk foreign key (pid) references
products(pid);


#初始化数据
#给商品表初始化数据
insert into products(pid,name,price,category_id) values('p001','联想',5000,'c001');
insert into products(pid,name,price,category_id) values('p002','海尔',3000,'c001');
insert into products(pid,name,price,category_id) values('p003','雷神',5000,'c001');
insert into products(pid,name,price,category_id) values('p004','JACK JONES',800,'c002');
insert into products(pid,name,price,category_id) values('p005','真维斯',200,'c002');

insert into products(pid,name,price,category_id) values('p006','花花公子',440,'c002');
```

```
insert into products(pid,name,price,category_id) values('p007','劲霸',2000,'c002');
insert into products(pid,name,price,category_id) values('p008','香奈儿',800,'c003');
insert into products(pid,name,price,category_id) values('p009','相宜本草',200,'c003');
insert into products(pid,name,price,category_id) values('p010','梅明子',200,null);


#给分类表初始化数据
insert into category values('c001','电器');
insert into category values('c002','服饰');
insert into category values('c003','化妆品');
insert into category values('c004','书籍');
```

6.1 综合练习-【多表查询】

1>查询用户的订单,没有订单的用户不显示

2>查询所有用户的订单详情

3>查询所有订单的用户详情

6.2 综合练习2-【子查询】

1>查看用户为张三的订单详情

2>查询出订单的价格大于300的所有用户信息。

3>查询订单价格大于300的订单信息及相关用户的信息。

6.2 综合练习3-【分页查询】

1>查询所有订单信息，每页显示5条数据

第七章 SQL优化

1.应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

2.对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

3.应尽量避免在 where 子句中使用!=或<>操作符，否则将引擎放弃使用索引而进行全表扫描。

4.应尽量避免在 where 子句中使用 or 来连接条件，否则将导致引擎放弃使用索引而进行全表扫描

5.in 和 not in 也要慎用，否则会导致全表扫描

6.应尽量避免在 where 子句中对字段进行表达式操作

7.应尽量避免在where子句中对字段进行函数操作

8.很多时候用 exists 代替 in 是一个好的选择

9.尽量使用数字型字段

10.尽可能的使用 varchar/nvarchar 代替 char/nchar ，因为首先变长字段存储空间小，可以节省存储空间，其次对于查询来说，在一个相对较小的字段内搜索效率显然要高些

11.任何地方都不要使用 select * from t ，用具体的字段列表代替“*”，不要返回用不到的任何字段。

12.尽量使用表变量来代替临时表。如果表变量包含大量数据，请注意索引非常有限（只有主键索引）。

13.避免频繁创建和删除临时表，以减少系统表资源的消耗。

14.在新建临时表时，如果一次性插入数据量很大，那么可以使用 select into 代替 create table，避免造成大量 log ，以提高速度；如果数据量不大，为了缓和系统表的资源，应先create table，然后insert。

作业题

```
数据库结构
创建四张表 分别存储 学生信息 课程信息 分数 讲师信息表 存储相应数据
学生信息表 Student
字段名   字段类型 字段约束 / 含义
Sno     Varchar(3)      Not null  / 学员编号
Sname   Varchar(4)      Not null  / 学员姓名
Ssex    Varchar(2)      Not null  / 性别
Sbirthday       Datetime        生日
Classnum        Varchar(5)      班级号

CREATE TABLE STUDENT
(
SNO VARCHAR(3) NOT NULL,
SNAME VARCHAR(4) NOT NULL,
SSEX VARCHAR(2) NOT NULL,
SBIRTHDAY DATETIME,
CLASS VARCHAR(5)
)
课程信息表 course
字段名   字段类型 字段约束 / 含义
Cno     Varchar(5)      Not null / 课程编号
Cname   Varchar(10)     Not null / 课程名称
Tno     Varchar(10)     Not null / 授课老师编号


CREATE TABLE COURSE
(CNO VARCHAR(5) NOT NULL,
CNAME VARCHAR(10) NOT NULL,
TNO VARCHAR(10) NOT NULL)
成绩表score
字段名   字段类型 字段约束 / 含义
Sno     Varchar(3)      Not null  / 学员编号
Cno     Varchar(5)      Not null  / 课程编号
Degree  Double(3,1)     Not null  / 分数

CREATE TABLE SCORE
(SNO VARCHAR(3) NOT NULL,
CNO VARCHAR(5) NOT NULL,
DEGREE NUMERIC(10, 1) NOT NULL)
讲师表teacher
字段名   字段类型 字段约束 / 含义
Tno     Varchar(3)      Not null  / 讲师编号
Tname   Varchar(4)      Not null  / 讲师姓名
Tsex    Varchar(2)      Not null  / 讲师性别
Tbirthday       Datetime        Not null  / 出生日期
Prof    Varchar(6)      等级
```

```
Depart  Varchar(10)     所属院系

CREATE TABLE TEACHER
(TNO VARCHAR(3) NOT NULL,
TNAME VARCHAR(4) NOT NULL, TSEX VARCHAR(2) NOT NULL,
TBIRTHDAY DATETIME NOT NULL, PROF VARCHAR(6),
DEPART VARCHAR(10) NOT NULL)
```

向表中存储数据

```
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (108 ,'曾华' ,'男' ,1977-09-01,95033);
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (105 ,'匡明' ,'男' ,1975-10-02,95031);
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (107 ,'王丽' ,'女' ,1976-01-23,95033);
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (101 ,'李军' ,'男' ,1976-02-20,95033);
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (109 ,'王芳' ,'女' ,1975-02-10,95031);
INSERT INTO STUDENT (SNO,SNAME,SSEX,SBIRTHDAY,CLASS) VALUES (103 ,'陆君' ,'男' ,1974-06-03,95031);
GO
INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('3-105' ,'计算机导论',825)
INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('3-245' ,'操作系统' ,804);
INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('6-166' ,'数据电路' ,856);
INSERT INTO COURSE(CNO,CNAME,TNO)VALUES ('9-888' ,'高等数学' ,100);
GO
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (103,'3-245',86);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (105,'3-245',75);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (109,'3-245',68);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (103,'3-105',92);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (105,'3-105',88);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (109,'3-105',76);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (101,'3-105',64);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (107,'3-105',91);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (108,'3-105',78);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (101,'6-166',85);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (107,'6-106',79);
INSERT INTO SCORE(SNO,CNO,DEGREE)VALUES (108,'6-166',81);
GO
INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)
VALUES (804,'李诚','男','1958-12-02','副教授','计算机系');
INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)
VALUES (856,'张旭','男','1969-03-12','讲师','电子工程系');
INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)
VALUES (825,'王萍','女','1972-05-05','助教','计算机系');
INSERT INTO TEACHER(TNO,TNAME,TSEX,TBIRTHDAY,PROF,DEPART)
VALUES (831,'刘冰','女','1977-08-14','助教','电子工程系');
```

1、 查询Student表中的所有记录的Sname、Ssex和Class列。
2、 查询教师所有的单位即不重复的Depart列。
3、 查询Student表的所有记录。
4、 查询Score表中成绩在60到80之间的所有记录。
5、 查询Score表中成绩为85，86或88的记录。
6、 查询Student表中“95031”班或性别为“女”的同学记录。
7、 以Class降序查询Student表的所有记录。
8、 以Cno升序、Degree降序查询Score表的所有记录。
9、 查询“95031”班的学生人数。

10、查询Score表中的最高分的学生学号和课程号。
11、查询‘3-105’号课程的平均分。
12、查询Score表中至少有5名学生选修的并以3开头的课程的平均分数。
13、查询最低分大于70，最高分小于90的Sno列。
14、查询所有学生的Sname、Cno和Degree列。
15、查询所有学生的Sno、Cname和Degree列。
16、查询所有学生的Sname、Cname和Degree列。
17、查询“95033”班所选课程的平均分。
18、假设使用如下命令建立了一个grade表：

```
create table grade(low number(3,0),upp number(3),rank char(1));
insert into grade values(90,100,’A’);
insert into grade values(80,89,’B’);
insert into grade values(70,79,’C’);
insert into grade values(60,69,’D’);
insert into grade values(0,59,’E’);
commit;
```

现查询所有同学的Sno、Cno和rank列。
19、查询选修“3-105”课程的成绩高于“109”号同学成绩的所有同学的记录。
20、查询score中选学一门以上课程的同学中分数为非最高分成绩的记录。
21、查询成绩高于学号为“109”、课程号为“3-105”的成绩的所有记录。
22、查询和学号为108的同学同年出生的所有学生的Sno、Sname和Sbirthday列。
23、查询“张旭“教师任课的学生成绩。
24、查询选修某课程的同学人数多于5人的教师姓名。
25、查询95033班和95031班全体学生的记录。
26、查询存在有85分以上成绩的课程Cno.
27、查询出“计算机系“教师所教课程的成绩表。
28、查询“计算机系”与“电子工程系“不同职称的教师的Tname和Prof。
29、查询选修编号为“3-105“课程且成绩至少高于选修编号为“3-245”的同学的Cno、Sno和Degree,并按Degree从高到低次序排序。
30、查询选修编号为“3-105”且成绩高于选修编号为“3-245”课程的同学的Cno、Sno和Degree.
31、查询所有教师和同学的name、sex和birthday.
32、查询所有“女”教师和“女”同学的name、sex和birthday.
33、查询成绩比该课程平均成绩低的同学的成绩表。
34、查询所有任课教师的Tname和Depart.
35 查询所有未讲课的教师的Tname和Depart.
36、查询至少有2名男生的班号。
37、查询Student表中不姓“王”的同学记录。
38、查询Student表中每个学生的姓名和年龄。
39、查询Student表中最大和最小的Sbirthday日期值。
40、以班号和年龄从大到小的顺序查询Student表中的全部记录。
41、查询“男”教师及其所上的课程。
42、查询最高分同学的Sno、Cno和Degree列。
43、查询和“李军”同性别的所有同学的Sname.
44、查询和“李军”同性别并同班的同学Sname.
45、查询所有选修“计算机导论”课程的“男”同学的成绩表

面试题

有三张表,学生表S,课程C,学生课程表SC,学生可以选修多门课程,一门课程可以被多个学生选修,通过SC 表关联。【基础】

1）写出建表语句;

2）写出SQL 语句,查询选修了所有选修课程的学生;

3）写出SQL 语句,查询选修了至少5 门以上的课程的学生。