



3.4 查 询

- **3.4.1 单表查询**
 - **3.4.2 连接查询**
 - **3.4.3 嵌套查询**
 - **3.4.4 集合查询**
 - **3.4.5 SELECT 语句的一般格式**
-



3.4.1 单表查询

查询仅涉及一个表，是一种最简单的查询操作

- 一、选择表中的若干列
 - 二、选择表中的若干元组
 - 三、对查询结果排序
 - 四、使用集函数
 - 五、对查询结果分组
-



1. 查询指定列

[例 1] 查询全体学生的学号与姓名。

```
SELECT Sno , Sname  
FROM S ;
```

[例 2] 查询全体学生的姓名、学号、所在系。

```
SELECT Sname , Sno , Sdept  
FROM S ;
```

注意： 目标表达式中的各个列的先后顺序可以与表中的顺序不一致。



2. 查询全部列

[例 3] 查询全体学生的详细记录。

```
SELECT Sno , Sname , Ssex , Sage , Sdept  
FROM Student ;
```

或

```
SELECT *  
FROM Student ;
```



3. 查询经过计算的值

SELECT 子句的 < 目标列表达式 > 为表达式

- 算术表达式
 - 字符串常量
 - 函数
 - 列别名
 - 等
-



3. 查询经过计算的值

[例 4] 查全体学生的姓名及其出生年份。

```
SELECT  Sname , 2004-Sage  
FROM  Student ;
```

输出结果:

Sname 无列名

| | |
|----|------|
| 李勇 | 1976 |
| 刘晨 | 1977 |
| 王名 | 1978 |
| 张立 | 1978 |



3. 查询经过计算的值

查询全体学生的姓名、出生年份和所有系，
要求用小写字母表示所有系名。

```
SELECT  Sname,'Year of Birth:',2004-Sage,  
        LOWER(Sdept)
```

```
FROM    Student
```

输出结果：

| Sname | 无列名 | 无列名 | 无列名 |
|-------|----------------|-------|-------|
| ----- | ----- | ----- | ----- |
| 李勇 | Year of Birth: | 1984 | cs |
| 刘晨 | Year of Birth: | 1985 | is |
| 王名 | Year of Birth: | 1986 | ma |
| 张立 | Year of Birth: | 1985 | is |



4. 使用列别名改变查询结果的列标题

```
SELECT      Sname  NAME, 'Year of Birth: ' BIRTH,  
            2000-Sage      BIRTHDAY,  
            LOWER(Sdept) DEPARTMENT
```

FROM Student

或者:

- ```
SELECT Sname as NAME,
 'Year of Birth: ' as BIRTH,
 2000-Sage as BIRTHDAY,
 LOWER(Sdept) as DEPARTMENT
```
- ```
FROM Student
```

输出结果

| NAME | BIRTH | BIRTHDAY | DEPARTMENT |
|-------|----------------|----------|------------|
| ----- | ----- | ----- | ----- |
| 李勇 | Year of Birth: | 1984 | cs |
| 刘晨 | Year of Birth: | 1985 | is |
| 王名 | Year of Birth: | 1986 | ma |
| 张立 | Year of Birth: | 1984 | is |



4. 使用列别名改变查询结果的列标题

- 例 1 : **select 姓名 , 工资 , 工资 *0.8 as ' 奖金 ' from 职工**
 - 例 2 : **select 姓名 , 工资 ,(cast(工资 *0.8 as decimal(18,1))) as ' 奖金 ' from 职工**
 - 例 3 : **select 姓名 , 工资 *5.0/1000 as 代扣税 , 工资 *0.08 as 奖金 , 工资 -(工资 *5.0/1000)+(工资 *0.08) as 应发工资 from 职工**
-



5. 替换查询结果中的数据

- **select** 姓名, 工资, 工资 =
 - **case**
 - **when** 工资 > 2000 **then** '优秀'
 - **when** 工资 >= 1800 **and** 工资 <= 2000 **then** '优良'
 - **when** 工资 >= 1500 **and** 工资 < 1800 **then** '一般'
 - **when** 工资 < 1500 **then** '差'
 - **end**
 - **from** 职工
-



6. 连接列值

- **select** 姓名 + ' 的工资是: ' + 工资
as 工资说明
 - **from** 职工 (**错误!!**)
 - **select** 姓名 + ' 的工资是: ' + **cast**(工资
as varchar(10)) as 工资说明
 - **from** 职工 (**正确!!**)
 - **注意: 要转换成字符型 / 字符必须用单引号**
-



举例

- **TableName = " 产品基本信息 "**
 - **Set OBJConn =
Server.CreateObject("ADODB.Connection")**
 - **OBJConn.open "Trade"**
 - **SQLstr = "Select * from" & TableName**
 - **Set Ra = OBJConn.Execute(SQLstr)**
 - **.....**
-



举例

- **username=request.querystring("username")**
 - **password=request.querystring("password")**
 - **set rs=server.createobject("adodb.recordset")**
 - **sql="select * from table1 where
_username='&username&'
and password='&password&'"**
 - **rs.open sql,conn,1,3**
-



7. 查询字符串长度、大写、小写函数

- ✓ **select** **len**('You are a girl'),**len**(' 你是女孩 '),**len**(' 你是 girl')
 - ✓ **select** ' 计算机系 ' + **space**(5) + ' 网络专业 '
 - ✓ **select**
lower('ABCDEfg'),**lower**('WonDERful')
 - ✓ **select** **upper**('wonderful'),
upper('ABcdefg')
-



8. 查询字符串截取函数

- ✓ **select ltrim(' 计算机网络专业 '),**
rtrim(' 计算机网络专业 ')
 - ✓ **select left(' 计算机系网络专业 ',4), right(' 计算机系网络专业 ',4)**
 - ✓ **select reverse(12345)**
 - ✓ **select reverse(' 计算中心 ')**
-



9. 日期时间函数

- **getdate():** 现在日期时间
 - **year, month, day :** 年, 月, 日
 - **datepart(格式串, 日期型表达式) :** 日期部分
 - ✓ **select getdate() as ' 今日现在 '**
 - ✓ **select year('2011-4-2')**
 - ✓ **select month('2011-4-2')**
 - ✓ **select day('2011-4-2')**
 - ✓ **select datepart(yyyy,getdate()),
datepart(mm,getdate())**
 - ✓ **select datepart(dd,getdate()), datepart(hh,getdate())**
 - ✓ **select datepart(n, getdate()), datepart(s, getdate())**
-



9. 日期时间函数

- **dateadd(格式串, 数值, 日期)** : 日期加
 - **datediff(格式串, 日期 1 , 日期 2)** : 日期差
 - **select dateadd(year,2,'2011-4-2')**
 - **select dateadd(month,-1,'2011-4-2')**
 - **select dateadd(day,3,getdate())**
 - **select datediff(day,'2011-4-1',getdate())**
 - **select datediff(month,'2011-3-2',getdate())**
 - **select datediff(year,'2001-3-1',getdate())**
-



9. 日期时间函数

- 例：查询年龄小于 12 岁的女同学

Select

姓名,

年龄 =datediff(year, 出生年月 ,getdate())

from student

where

(datediff(year, 出生年月 ,getdate())<=12)

and

(性别 =‘ 女’)



二、选择表中的若干元组

- 消除取值重复的行
 - 查询满足条件的元组
-



1. 消除取值重复的行

– 在 **SELECT** 子句中使用 **DISTINCT** 短语

假设 **SC** 表中有下列数据

| Sno | Cno | Grade |
|--------------|------------|--------------|
| ----- | ----- | ----- |
| 95001 | 1 | 92 |
| 95001 | 2 | 85 |
| 95001 | 3 | 88 |
| 95002 | 2 | 90 |
| 95002 | 3 | 80 |



ALL 与 DISTINCT

[例 6] 查询选修了课程的学生学号。

**(1) SELECT Sno
FROM SC;**

或 (默认 ALL)

**SELECT ALL Sno
FROM SC;**

结果: Sno

200215121

200215121

200215121

200215122

200215122



**(2) SELECT DISTINCT Sno
FROM SC;**

结果:

Sno

200215121

200215122

指定 DISTINCT ， 表示去掉重复行。缺省为 ALL



例题（续）

- **注意** **DISTINCT** 短语的作用范围是所有目标列

例：查询选修课程的各种成绩

错误的写法

```
SELECT DISTINCT Cno ,  DISTINCT Grade  
FROM SC;
```

正确的写法

```
SELECT DISTINCT Cno ,  Grade  
FROM SC;
```



2. 查询满足条件的元组

WHERE 子句常用的查询条件

表 3.3 常用的查询条件

| 查 询 条 件 | 谓 词 |
|---------|---|
| 比 较 | =, >, <, >=, <=, !=, <>, !>, !<; NOT + 上述比较运算符 |
| 确定范围 | BETWEEN AND, NOT BETWEEN AND |
| 确定集合 | IN, NOT IN |
| 字符匹配 | LIKE, NOT LIKE |
| 空 值 | IS NULL, IS NOT NULL |
| 多重条件 | AND, OR |



(1) 比较大小

在 **WHERE** 子句的 **< 比较条件 >** 中使用比较运算符

– **= , > , < , >= , <= , != 或 <> , !> , !< ,**

– 逻辑运算符 **NOT** + 比较运算符

– [例 7] 查询计算机科学系全体学生的名单

– **select sname**

– **from s**

– **where sdept= 'CS'**



— [例 8] 查询所有年龄在 20 岁以下的学生姓名及其年龄。

```
SELECT Sname , Sage  
FROM Student  
WHERE Sage < 20 ;
```

或

```
SELECT Sname , Sage  
FROM Student  
WHERE NOT Sage >= 20 ;
```



[例 9] 查询考试成绩有不及格的学生们的学号

- **Select distinct sno**

- **From sc**

- **Where grade<60**



(2) 确定范围

- 使用谓词 **BETWEEN ... AND ...**
NOT BETWEEN ... AND ...

[例 10] 查询年龄在 **20~23 岁** (包括 20 岁和 23 岁) 之间的学生的姓名、系别和年龄。

```
SELECT Sname , Sdept , Sage  
FROM S  
WHERE Sage BETWEEN 20 AND 23 ;
```



[例 11] 查询年龄不在 20~23 岁之间的学生姓名、系别和年龄。

```
SELECT Sname , Sdept , Sage  
FROM S  
WHERE Sage NOT BETWEEN 20 AND 23 ;
```



(3) 确定集合

使用谓词 **IN < 值表 >, NOT IN < 值表 >**

< 值表 > : 用逗号分隔的一组取值

[例 12] 查询信息系（**IS**）、数学系（**MA**）和计算机科学系（**CS**）学生的姓名和性别。

SELECT Sname , Ssex

FROM S

WHERE Sdept IN ('IS' , 'MA' , 'CS');



[例 13] 查询既不是信息系、数学系，也不是
计算机科学系的学生的姓名和性别。

SELECT Sname , Ssex

FROM Student

WHERE Sdept NOT IN ('IS' , 'MA' , 'CS');



IN 举例

例：显示姓名为“张怡宁”的订购单信息

Use business

```
Select * from 订购单 where 职工号 in  
( select 职工号 from 职工 where 姓名 = '张怡宁' )
```

例：显示工资大于 2000 的职工的订购单信息

Use business

```
Select * from 订购单 where 职工号 in  
(select 职工号 from 职工 where 工资 > 2000)
```



(4) 字符串匹配

- **[NOT] LIKE ‘< 匹配串 >’ [ESCAPE ‘< 换码字符 >’]**

< 匹配串 > : 指定匹配模板

匹配模板：固定字符串或含通配符的字符串

当匹配模板为固定字符串时，

可以用 = 运算符取代 LIKE 谓词

用 != 或 <> 运算符取代 NOT LIKE 谓词



通配符

1) % (百分号) 代表任意长度 (长度可以为 0) 的字符串

– 例: **a%b** 表示以 **a** 开头, 以 **b** 结尾的任意长度的字符串。如 **acb**, **addgb**, **ab** 等都满足该匹配串

2) _ (下横线) 代表任意单个字符

– 例: **a_b** 表示以 **a** 开头, 以 **b** 结尾的长度为 3 的任意字符串。如 **acb**, **afb** 等都满足该匹配串



通配符

3) [] 表示匹配 [] 中列出的任一个字符。

例: $x[mnk]y$ 表示以 x 开头, 以 y 结尾, 第二个字符为 m, n, k 中的任意一个字符的由三个字符组成的字符串;

4) [^] 表示不在方括号里列出的任一个字符。



ESCAPE 短语:

- 当用户要查询的字符串本身就含有 % 或 _ 时, 要使用 **ESCAPE** '< 换码字符 >' 短语对通配符进行转义。



例题

1) 匹配模板为固定字符串

[例 14] 查询学号为 200215121 的学生的详细情况。

```
SELECT *  
FROM Student  
WHERE Sno LIKE '200215121';
```

等价于：

```
SELECT *  
FROM Student  
WHERE Sno = '200215121';
```



2) 匹配模板为含通配符的字符串

[例 15] 查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname , Sno , Ssex  
FROM Student  
WHERE Sname LIKE ‘刘 %’ ;
```



匹配模板为含通配符的字符串（续）

[例 16] 查询姓 " 欧阳 " 且全名为三个汉字的学生
的姓名。

```
SELECT Sname  
FROM Student  
WHERE Sname LIKE ' 欧阳 _' ;
```



匹配模板为含通配符的字符串（续）

[例 17] 查询名字中第 2 个字为 " 阳 " 字的学生的姓名和学号。

```
SELECT Sname , Sno  
FROM Student  
WHERE Sname LIKE '_ 阳 %' ;
```




匹配模板为含通配符的字符串（续）

[例 18] 查询所有不姓刘的学生姓名。

```
SELECT Sname , Sno , Ssex  
FROM Student  
WHERE Sname NOT LIKE '刘 %' ;
```



3) 使用换码字符将通配符转义为普通字符

[例 19] 查询 DB_Design 课程的课程号和学分。

```
SELECT Cno , Ccredit
```

```
FROM Course
```

```
WHERE Cname LIKE 'DB\_Design' ESCAPE '\'
```



使用换码字符将通配符转义为普通字符 (续)

[例 20] 查询以 "DB_" 开头，且倒数第 3 个字符为 i 的课程的具体情况。

```
SELECT *
```

```
FROM Course
```

```
WHERE Cname LIKE 'DB\__%i\__' ESCAPE '\';
```

ESCAPE '\' 表示 “\” 为换码字符，紧跟在 “\” 后面的字符 “_” 不再具有通配符的含义，转义为普通的 “_” 字符



(5) 涉及空值的查询

- 使用谓词 **IS NULL** 或 **IS NOT NULL**
- “**IS NULL**” 不能用 “**= NULL**” 代替

[例 21] 某些学生选修课程后没有参加考试，所以有选课记录，但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号。

```
SELECT Sno , Cno  
  
FROM SC  
  
WHERE Grade IS NULL ;
```



[例 22] 查所有有成绩的学生学号和课程号。

```
SELECT Sno , Cno  
FROM SC  
WHERE Grade IS NOT NULL ;
```



(6) 多重条件查询

用逻辑运算符 **AND** 和 **OR** 来联结多个查询条件

- **AND** 的优先级高于 **OR**
- 可以用括号改变优先级

可用来实现多种其他谓词

- **[NOT] IN**
 - **[NOT] BETWEEN ... AND ...**
-



[例 23] 查询计算机系年龄在 20 岁以下的学生姓名。

```
SELECT Sname  
FROM Student  
WHERE Sdept= 'CS' AND Sage<20 ;
```



改写 [例 12]

[例 12] 查询信息系（ **IS** ）、数学系（ **MA** ）和计算机科学系（ **CS** ）学生的姓名和性别。

```
SELECT Sname , Ssex
```

```
FROM Student
```

```
WHERE Sdept IN ( 'IS' , 'MA' , 'CS' )
```

可改写为：

```
SELECT Sname , Ssex
```

```
FROM Student
```

```
WHERE Sdept= ' IS ' OR Sdept= ' MA' OR Sdept= ' CS ' ;
```



改写 [例 10]

[例 10] 查询年龄在 **20~23** 岁（包括 **20** 岁和 **23** 岁）之间的学生的姓名、系别和年龄。

SELECT Sname , Sdept , Sage

FROM Student

WHERE Sage BETWEEN 20 AND 23 ;

可改写为:

SELECT Sname , Sdept , Sage

FROM Student

WHERE Sage>=20 AND Sage<=23 ;



三、对查询结果排序

使用 **ORDER BY** 子句

- 可以按一个或多个属性列排序
- 升序： **ASC** ； 降序： **DESC** ； 缺省值为升序

当排序列含空值时

- **ASC** ： 排序列为空值的元组最后显示
 - **DESC** ： 排序列为空值的元组最先显示
-



[例 24] 查询选修了 3 号课程的学生们的学号及其成绩，查询结果按分数降序排列。

```
SELECT Sno , Grade  
FROM SC  
WHERE Cno= ' 3 '  
ORDER BY Grade DESC ;
```



[例 25] 查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

SELECT *

FROM Student

ORDER BY Sdept , Sage DESC ;

注意：排序的属性可以不止一列



例：使用姓名笔画升序排序职工信息。

use test

select * from course

order by Cname collate

chinese_prc_stroke_cs_as_ks_ws asc

例：使用姓名音序升序排序职工信息

use test

select * from course

order by Cname collate chinese_prc_cs_as
asc



显示部分记录的排序

- 利用 **top** 关键字可以显示排序后的部分记录信息。可以直接用 “**Top 数字**”，显示指定条数记录。也可以使用 “**Top 数字 Percent**” 显示所有满足条件记录的前百分之几条记录。
- 例：显示工资最高的前 3 条职工信息

Select top 3 * from 职工 order by 工资 desc

例：显示工资最低的前 20% 条职工记录

**Select top 20 percent * from 职工 order by
工资**



四、使用集函数

5类主要集函数

— 计数

COUNT (*) 计算元组个数

COUNT ([DISTINCT|ALL] < 列名 >) 计算一列值个数

— 计算总和

SUM ([DISTINCT|ALL] < 列名 >)

值的总和，此列必须是数值型

— 计算平均值

AVG ([DISTINCT|ALL] < 列名 >)

值的平均值，此列必是数值型



求最大值

– MAX (< 列名 >)

求最小值

– MIN (< 列名 >)

- **DISTINCT** 短语：在计算时要取消指定列中的重复值
- **ALL** 短语：不取消重复值
- **ALL** 为缺省值



[例 26] 查询学生总人数。

```
SELECT COUNT(*)  
FROM Student ;
```

[例 27] 查询选修了课程的学生人数。

```
SELECT COUNT(DISTINCT Sno)  
FROM SC ;
```

注：用 **DISTINCT** 以避免重复计算学生人数



[例 28] 计算 1 号课程的学生平均成绩。

```
SELECT AVG(Grade)  
FROM SC  
WHERE Cno= ' 1 ' ;
```

[例 29] 查询选修 1 号课程的学生最高分数。

```
SELECT MAX(Grade)  
FROM SC  
WHER Cno= ' 1 ' ;
```



-
- **[例 30] 查询学生 200215121 选修课程的总学分**
 - **Select sum (ccredit) 、该生总学分'**
 - **From sc,c**
 - **Where sc.cno=c.cno and sno= '200215121'**
-



举例

- 显示 wh2 仓库工资大于 1800 的职工的平均工资

**Select sum(工资)/count(*) as ‘平均工资’ from
职工 where 仓库号 =‘wh2’ and 工资 >1800**

- 显示仓库面积最大的仓库信息

**Select * from 仓库 where 面积 =(select max(面
积) from 仓库)**



五、对查询结果分组

使用 **GROUP BY** 子句分组

将查询的结果按某一系列或多列的值分组，值相等的为一组

细化集函数的作用对象

- 未对查询结果分组，集函数将作用于整个查询结果
 - 对查询结果分组后，集函数将分别作用于每个组，即每一组都有一个函数值
-



使用 **GROUP BY** 子句分组

[例 31] 求各个课程号及相应的选课人数。

```
SELECT Cno , COUNT(Sno)  
FROM SC  
GROUP BY Cno ;
```

结果

| | Cno | COUNT(Sno) |
|----------|------------|-------------------|
| 1 | 22 | |
| | 2 | 34 |
| | 3 | 44 |
| 4 | 33 | |
| | 5 | 48 |



对查询结果分组 几点说明

- **GROUP BY** 子句的作用对象是查询的中间结果表
 - 分组方法：按指定的一列或多列值分组，值相等的为一组
 - 使用 **GROUP BY** 子句后， **SELECT** 子句的列名列表中只能出现分组属性和集函数
-



使用 **HAVING** 短语筛选最终输出结果

[例 32] 查询选修了 3 门以上课程的学生学号。

```
SELECT Sno  
FROM SC  
GROUP BY Sno  
HAVING COUNT(*) >=3 ;
```



例题

[例] 查询有 3 门以上课程是 20 分以上的
学生的学号及（20 分以上的）课程数

```
SELECT Sno, COUNT(*) 选课门数  
FROM SC  
WHERE Grade>=20  
GROUP BY Sno  
HAVING COUNT(*)>=3;
```



使用 **HAVING** 短语筛选最终输出结果

- 只有满足 **HAVING** 短语指定条件的组才输出
 - **HAVING** 短语与 **WHERE** 子句的区别：作用对象不同
 - **WHERE** 子句作用于基表或视图，从中选择满足条件的元组。
 - **HAVING** 短语作用于组，从中选择满足条件的组。
-