国家“十一五”规划教材

数据库原理与应用教程

机械工业出版社
China Machine Press

第4章 数据操作

- 4.1 数据查询功能
- 4.2 数据更改功能
- 4.3 查询语句扩展

HZ BOOKS

4.1 数据查询功能

- 4.1.1 查询语句的基本结构
- 4.1.2 简单查询
- 4.1.3 多表连接查询
- 4.1.4 子查询

查询语句基本格式

SELECT <目标列名序列>　　--需要哪些列

FROM <数据源>　　--来自于哪些表

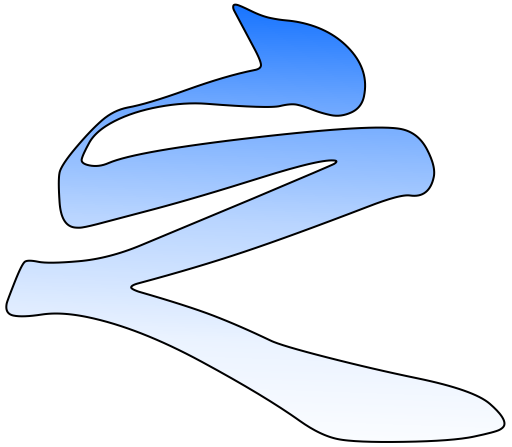
[WHERE <检索条件>]　　--根据什么条件

[GROUP BY <分组依据列>]

[HAVING <组提取条件>]

[ORDER BY <排序依据列>]

4.1.2 简单查询



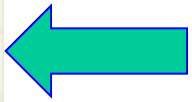
1. 选择表中若干列

1. 查询指定的列

- 查询表中用户感兴趣的部分属性列。
- 例1：查询全体学生的学号与姓名。

```
SELECT Sno, Sname FROM Student
```

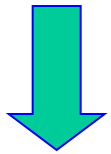
	Sno	Sname
1	9512101	李勇
2	9512102	刘晨
3	9512103	王敏
4	9521101	张立
5	9521102	吴宾
6	9521103	张海
7	9531101	钱小平
8	9531102	王大力



	Sno	Sname	Ssex	Sage	Sdept
1	9512101	李勇	男	19	计算机系
2	9512102	刘晨	男	20	计算机系
3	9512103	王敏	女	20	计算机系
4	9521101	张立	男	22	信息系
5	9521102	吴宾	女	21	信息系
6	9521103	张海	男	20	信息系
7	9531101	钱小平	女	18	数学系
8	9531102	王大力	男	19	数学系

例2. 查询全体学生的姓名、学号、所在系

```
SELECT Sname, Sno, Sdept
  FROM Student
```



	Sname	Sno	Sdept
1	李勇	9512101	计算机系
2	刘晨	9512102	计算机系
3	王敏	9512103	计算机系
4	张立	9521101	信息系
5	吴宾	9521102	信息系
6	张海	9521103	信息系
7	钱小平	9531101	数学系
8	王大力	9531102	数学系

	Sno	Sname	Ssex	Sage	Sdept
1	9512101	李勇	男	19	计算机系
2	9512102	刘晨	男	20	计算机系
3	9512103	王敏	女	20	计算机系
4	9521101	张立	男	22	信息系
5	9521102	吴宾	女	21	信息系
6	9521103	张海	男	20	信息系
7	9531101	钱小平	女	18	数学系
8	9531102	王大力	男	19	数学系

2. 查询全部列

- 例3. 查询全体学生的记录

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

- 等价于：

```
SELECT * FROM Student
```

3. 查询经过计算的列

- 例4. 查询全体学生的姓名及其出生年份。

```
SELECT Sname,2010 - Sage
       FROM Student
```

	Sname	(无列名)
1	李勇	1991
2	刘晨	1990
3	王敏	1990
4	张立	1988
5	吴宾	1989
6	张海	1990
7	钱小平	1992
8	王大力	1991

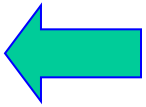
	Sno	Sname	Ssex	Sage	Sdept
1	9512101	李勇	男	19	计算机系
2	9512102	刘晨	男	20	计算机系
3	9512103	王敏	女	20	计算机系
4	9521101	张立	男	22	信息系
5	9521102	吴宾	女	21	信息系
6	9521103	张海	男	20	信息系
7	9531101	钱小平	女	18	数学系
8	9531102	王大力	男	19	数学系

常量列

- 例5．查询全体学生的姓名和出生年份所在系，并在出生年份列前加入一个列，此列的每行数据均为“出生年份”常量值。

```
SELECT Sname,'出生年份:', 2010-Sage
    FROM Student
```

	Sname	(无列名)	(无列名)
1	李勇	出生年份：	1991
2	刘晨	出生年份：	1990
3	王敏	出生年份：	1990
4	张立	出生年份：	1988
5	吴宾	出生年份：	1989
6	张海	出生年份：	1990
7	钱小平	出生年份：	1992
8	王大力	出生年份：	1991



	Sno	Sname	Ssex	Sage	Sdept
1	9512101	李勇	男	19	计算机系
2	9512102	刘晨	男	20	计算机系
3	9512103	王敏	女	20	计算机系
4	9521101	张立	男	22	信息系
5	9521102	吴宾	女	21	信息系
6	9521103	张海	男	20	信息系
7	9531101	钱小平	女	18	数学系
8	9531102	王大力	男	19	数学系

改变列标题

- 语法：

  列名 | 表达式 [AS] 列标题

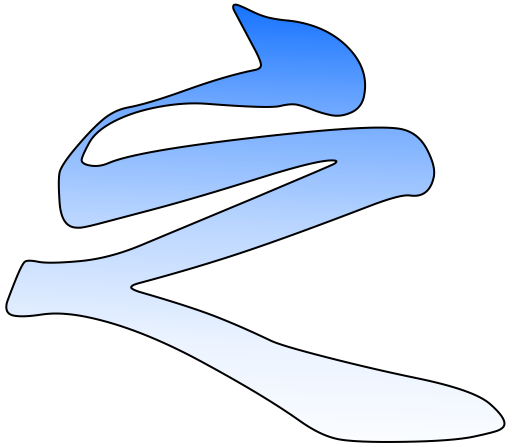
- 或：

  列标题 =列名 | 表达式

- 例：

```
SELECT Sname 姓名, 'Year of Birth' 出生年份,
       2010 - Sage 年份,
   FROM Student
```

4.1.2 简单查询



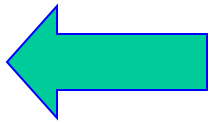
2. 选择表中若干元组

1.消除取值相同的行

例6．查询选修了课程的学生的学号

SELECT Sno FROM SC

	sno
1	9512101
2	9512101
3	9512101
4	9512102
5	9512102
6	9521102
7	9521102
8	9521102
9	9521102
10	9521103
11	9521103
12	9531101
13	9531101
14	9531102

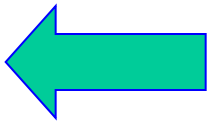


	Sno	Cno	Grade	xklb
1	9512101	c01	90	必修
2	9512101	c02	86	选修
3	9512101	c06	NULL	必修
4	9512102	c02	78	选修
5	9512102	c04	66	必修
6	9521102	c01	82	选修
7	9521102	c02	75	选修
8	9521102	c04	92	必修
9	9521102	c05	50	必修
10	9521103	c02	68	选修
11	9521103	c06	NULL	必修
12	9531101	c01	80	选修
13	9531101	c05	95	必修
14	9531102	c05	85	必修

要去掉结果表中的重复行，可用DISTINCT

SELECT DISTINCT Sno FROM SC

	sno
1	9512101
2	9512102
3	9521102
4	9521103
5	9531101
6	9531102



	Sno	Cno	Grade	xklb
1	9512101	c01	90	必修
2	9512101	c02	86	选修
3	9512101	c06	NULL	必修
4	9512102	c02	78	选修
5	9512102	c04	66	必修
6	9521102	c01	82	选修
7	9521102	c02	75	选修
8	9521102	c04	92	必修
9	9521102	c05	50	必修
10	9521103	c02	68	选修
11	9521103	c06	NULL	必修
12	9531101	c01	80	选修
13	9531101	c05	95	必修
14	9531102	c05	85	必修

2. 查询满足条件的元组

查询条件	谓　词
比较运算符	=, >, >=, <, <=, <>（或!=） NOT+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
逻辑谓词）	AND, OR

比较大小

- 例7. 查询计算机系全体学生的姓名。

```
SELECT Sname FROM Student
    WHERE Sdept = '计算机系'
```

- 例8. 查询年龄在20岁以下的学生的姓名及年龄。

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

- 例9. 查询考试成绩有不及格的学生的学号

```
 SELECT DISTINCT Sno  FROM SC
    WHERE Grade < 60
```

确定范围

- 用BETWEEN...AND和NOT BETWEEN...AND
- 是逻辑运算符，可以用来查找属性值在或不在指定范围内的元组，其中BETWEEN后边指定范围的下限，AND后边指定范围的上限。
- BETWEEN...AND...的格式为：

  列名 | 表达式 [NOT] BETWEEN 下限值 AND 上限值
- 如果列或表达式的值在[不在]下限值和上限值范围内，则结果为True，表明此记录符合查询条件。

示例

- 例10. 查询年龄在20～23岁之间的学生的姓名、所在系和年龄。

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

- 例11. 查询年龄不在20～23之间的学生姓名、所在系和年龄。

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

确定集合

- 使用IN运算符。
- 用来查找属性值属于指定集合的元组。
- 格式为：

  列名［ NOT ］IN（常量1，常量2，… 常量n）

- 当列中的值与IN中的某个常量值相等时，则结果为True，表明此记录为符合查询条件的记录；
- NOT IN：当列中的值与某个常量值相同时，则结果为False，表明此记录为不符合查询条件的记录

示例

- 例12. 查询信息系、数学系和计算机系学生的姓名和性别。

```
SELECT Sname, Ssex  FROM Student
    WHERE Sdept IN ('信息系', '数学系', '计算机系')
```

- 例13. 查询既不是信息系、数学系，也不是计算机系学生的姓名和性别。

```
SELECT Sname, Ssex  FROM Student
 WHERE Sdept NOT IN ('信息系', '数学系', '计算机系')
```

字符匹配

- 使用LIKE运算符
- 一般形式为：

  列名 [NOT] LIKE <匹配串>
- 匹配串中可包含如下四种通配符：
  - _：匹配任意一个字符；
  - %：匹配0个或多个字符；
  - [ ]：匹配[ ]中的任意一个字符；对于连续字母的匹配，例如匹配[abcd]，可简写为[a-d]
  - [^]：不匹配[ ]中的任意一个字符

示例

- 例14. 查询姓‘张’的学生的详细信息。

```
SELECT * FROM Student
  WHERE Sname LIKE '张%'
```

- 例15. 查询学生表中姓‘张’、‘李’和‘刘’的学生的情况。

```
SELECT * FROM Student
  WHERE Sname LIKE '[张李刘]%'
```

- 例16. 查询名字中第2个字为‘小’或‘大’的学生的姓名和学号。

```
SELECT Sname, Sno FROM Student
WHERE Sname LIKE '_[小大]%'
```

示例（续）

- 例17. 查询所有不姓“王”也不姓“张”的学生姓名

```
SELECT Sname FROM Student
   WHERE Sname NOT LIKE '[王张]%'
```

- 或者：

```
SELECT Sname FROM Student
        WHERE Sname LIKE '[^王张]%'
```

- 或者：

```
SELECT Sname FROM Student
   WHERE Sname NOT LIKE '王%'
   AND Sname NOT LIKE '张%'
```

示例（续）

- 例18. 查询姓“王”且名字是2个字的学生姓名。

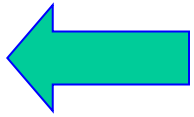
```
SELECT Sname FROM Student
    WHERE Sname LIKE '王_'
```

示例（续）

- 例19. 查询姓王且名字是3个字的学生姓名

```
SELECT Sname FROM Student
    WHERE Sname LIKE '王__'
```

	Sname
1	王敏
2	王大力



	Sno	Sname	Ssex	Sage	Sdept
1	9512101	李勇	男	19	计算机系
2	9512102	刘晨	男	20	计算机系
3	9512103	王敏	女	20	计算机系
4	9521101	张立	男	22	信息系
5	9521102	吴宾	女	21	信息系
6	9521103	张海	男	20	信息系
7	9531101	钱小平	女	18	数学系
8	9531102	王大力	男	19	数学系

注意：尾随空格的处理。

```
SELECT Sname FROM Student
    WHERE rtrim(Sname) LIKE '王__'
```

涉及空值的查询

- 空值（NULL）在数据库中表示不确定的值。
- 例如，学生选修课程后还没有考试时，这些学生有选课记录，但没有考试成绩，因此考试成绩为空值。
- 判断某个值是否为NULL值，不能使用普通的比较运算符。
- 判断取值为空的语句格式为：

  列名 IS NULL
- 判断取值不为空的语句格式为：

  列名 IS NOT NULL

示例

- 例20. 查询没有考试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21. 查询所有有考试成绩的学生的学号和课程号。

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

多重条件查询

- 在WHERE子句中可以使用逻辑运算符AND和OR来组成多条件查询。
  - 用AND连接的条件表示必须全部满足所有的条件的结果才为True；
  - 用OR连接的条件表示只要满足其中一个条件结果即为True。
- 例21. 查询计算机系年龄在20岁以下的学生姓名。

```
SELECT Sname FROM Student
    WHERE Sdept = '计算机系' AND Sage < 20
```

示例（续）

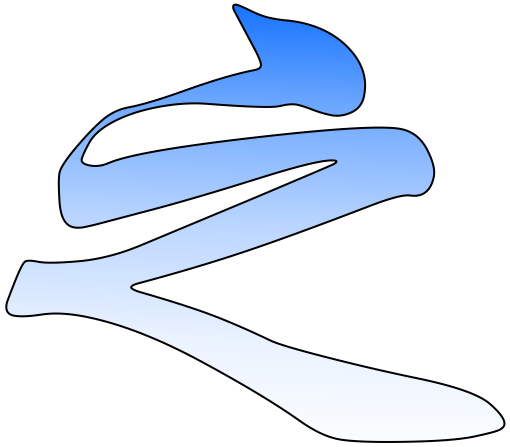
- 例23. 查询计算机系和信息系年龄大于等于20岁的学生姓名、所在系和年龄。

```
SELECT Sname,Sdept, Sage FROM Student
   WHERE (Sdept = '计算机系'
          OR Sdept = '信息系')
    AND Sage >= 20
```

或：

```
SELECT Sname,Sdept, Sage FROM Student
   WHERE Sdept IN ('计算机系', '信息系')
   AND Sage >= 20
```

4.1.2 简单查询##



3. 对查询结果进行排序

对查询结果进行排序

- 可对查询结果进行排序。
- 排序子句为：

  ORDER BY <列名> [ASC | DESC] [,<列名> ...]

- 说明：按<列名>进行升序（ASC）或降序（DESC）排序。

示例

- 例22. 将学生按年龄的升序排序。

```
SELECT * FROM Student ORDER BY Sage
```

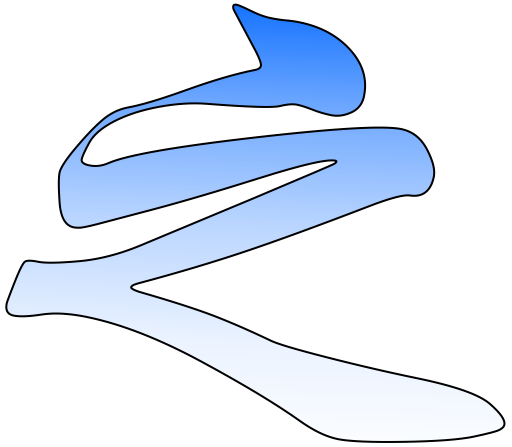
- 例23. 查询选修了‘c02’号课程的学生的学号及其成绩，查询结果按成绩降序排列。

```
SELECT Sno, Grade FROM SC
  WHERE Cno='c02' ORDER BY Grade DESC
```

- 例24. 查询全体学生的信息，查询结果按所在系的系名升序排列，同一系的学生按年龄降序排列。

```
SELECT * FROM Student
  ORDER BY Sdept, Sage DESC
```

4.1.2 简单查询



4. 使用计算函数汇总数据

使用计算函数汇总数据

- SQL提供的计算函数有：
  - COUNT（ * ）：统计表中元组个数；
  - COUNT（[DISTINCT]〈列名〉）：统计本列列值个数；
  - SUM（[DISTINCT]〈列名〉）：计算列值总和；
  - AVG（[DISTINCT]〈列名〉）：计算列值平均值；
  - MAX（[DISTINCT]〈列名〉）：求列值最大值；
  - MIN（[DISTINCT]〈列名〉）：求列值最小值。
- 上述函数中除COUNT（*）外，其他函数在计算过程中均忽略NULL值。

示例

- 例25. 统计学生总人数。

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

- 例26. 统计选修了课程的学生的人数。

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

- 例27 . 计算9512101号学生的考试总成绩之和。

```
SELECT SUM(Grade) FROM SC
 WHERE Sno = '9512101'
```

示例（续）

- 例28．计算’C01’号课程学生的考试平均成绩。

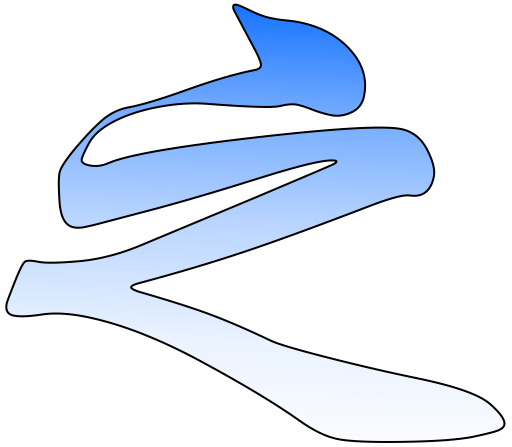
```
SELECT AVG(Grade) FROM SC
    WHERE Cno='C01'
```

- 例29．查询选修了’C01’号课程的学生的最高分和最低分。

```
SELECT MAX(Grade) , MIN(Grade)
        FROM SC WHERE Cno='C01'
```

- 注意：计算函数不能出现在WHERE子句中

4.1.2 简单查询



5. 对查询结果进行分组计算

对查询结果进行分组计算

- 作用：可以控制计算的级别：对全表还是对一组。
- 目的：细化计算函数的作用对象。
- 分组语句的一般形式：

  [GROUP BY <分组条件>]

  [HAVING <组过滤条件>]

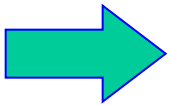
1. 使用GROUP BY

- 例30. 统计每门课程的选课人数，列出课程号和人数。

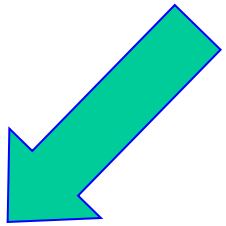
```
SELECT Cno as 课程号,
                COUNT(Sno) as 选课人数
FROM SC GROUP BY Cno
```

- 对查询结果按Cno的值分组，所有具有相同Cno值的元组为一组，然后再对每一组使用COUNT计算，求得每组的学生人数。

Sno	Cno	Grade
951201	C01	80
951201	C02	78
951202	C01	90
952103	C02	88
952103	C01	85
952103	C03	91
952103	C04	74



Sno	Cno	Grade
951201	C01	80
951202	C01	90
952103	C01	85
951201	C02	78
952103	C02	88
952103	C03	91
952103	C04	74



Cno	Count(Sno)
C01	3
C02	2
C03	1
C04	1

- 例31. 查询每名学生的选课门数和平均成绩。

```
SELECT Sno as 学号,
                COUNT(*) as 选课门数,
                AVG(Grade) as 平均成绩
    FROM SC GROUP BY Sno
```

2. 使用HAVING

- HAVING用于对分组自身进行限制，它有点象WHERE子句，但它用于组而不是对单个记录。
- 例32. 查询修了3门以上课程的学生的学号

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

示例

- 例33．查询修课门数等于或大于4门的学生的平均成绩和选课门数。

```
SELECT Sno, AVG(Grade) 平均成绩,
                    COUNT(*) 修课门数
   FROM SC
     GROUP BY Sno
     HAVING COUNT(*) >= 4
```

4.1.3 多表连接查询

- 若一个查询同时涉及两个或两个以上的表，则称之为连接查询。
- 连接查询是关系数据库中最主要的查询
- 连接查询包括内连接、外连接和交叉连接等。

连接基础知识

- 连接查询中用于连接两个表的条件称为连接条件或连接谓词。
- 一般格式为：

[<表名1.>][<列名1>] <比较运算符> [<表名2.>][<列名2>]

必须是可比的

内连接

- SQL-92 内连接语法如下：

```
SELECT ...
  FROM 表名  [INNER] JOIN
     被连接表
     ON  连接条件
```

- 执行连接操作的过程:
- 首先取表1中的第1个元组，然后从头开始扫描表2，逐一查找满足连接条件的元组，
- 找到后就将表1中的第1个元组与该元组拼接起来，形成结果表中的一个元组。
- 表2全部查找完毕后，再取表1中的第2个元组，然后再从头开始扫描表2，　…
- 重复这个过程，直到表1中的全部元组都处理完毕为止。

- 例36.查询计算机系学生的修课情况，要求列出学生的名字、所修课的课程号和成绩。

```
SELECT Sname, Cno, Grade
    FROM Student JOIN SC
    ON Student.Sno = SC.Sno
    WHERE Sdept = '计算机系'
```

- 例37. 查询信息系修了VB课程的学生的修课成绩，要求列出学生姓名、课程名和成绩。

```
SELECT Sname, Cname, Grade
    FROM  Student  s  JOIN  SC
    ON s.Sno = SC. Sno
    JOIN  Course c ON c.Cno = SC.Cno
    WHERE Sdept = '信息系'
          AND Cname = 'VB'
```

- 例38. 查询所有修了VB课程的学生的修课情况，要求列出学生姓名和所在的系。

```
SELECT Sname, Sdept
    FROM Student S JOIN SC
    ON S.Sno = SC. Sno
    JOIN Course C ON C.Cno = SC.cno
    WHERE Cname = 'VB'
```

自连接

- 为特殊的内连接
- 相互连接的表物理上为同一张表。
- 必须为两个表取别名，使之在逻辑上成为两个表。

- 例39. 查询与刘晨在同一个系学习的学生的姓名和所在的系。

```
SELECT S2.Sname, S2.Sdept
   FROM Student S1 JOIN Student S2
   ON S1.Sdept = S2.Sdept
   WHERE S1.Sname = '刘晨'
   AND S2.Sname != '刘晨'
```

外连接

- 只限制一张表中的数据必须满足连接条件，而另一张表中数据可以不满足连接条件。
- ANSI方式的外连接的语法格式为：

  FROM 表1 LEFT | RIGHT [OUTER]

  JOIN 表2 ON 〈连接条件〉
- theta方式的外连接的语法格式为：
  - 左外连接：

    FROM 表1, 表2 WHERE [表1.]列名(+) = [表2.]列名
  - 右外连接：

    FROM 表1, 表2 WHERE [表1.]列名= [表2.]列名(+)

- 例40. 查询学生的修课情况，包括修了课程的学生和没有修课的学生。

```
SELECT Student.Sno, Sname, Cno, Grade
    FROM Student LEFT OUTER JOIN SC
    ON Student.Sno = SC.Sno
```

4.1.4 子查询

- 在SQL语言中，一个SELECT－FROM－WHERE语句称为一个查询块。
- 子查询是一个 SELECT 查询，它嵌套在 SELECT、INSERT、UPDATE、DELETE 语句的 WHERE 或 HAVING 子句内，或其它子查询中
- 子查询的 SELECT 查询总是使用圆括号括起来。

使用子查询进行基于集合的测试

- 使用子查询进行基于集合的测试的语句的一般格式为:

  列名 [NOT] IN (子查询)

示例

- 例41. 查询与刘晨在同一个系的学生。

②
```
SELECT Sno, Sname, Sdept
    FROM Student
   WHERE Sdept IN
```
①
```
      ( SELECT Sdept FROM Student
          WHERE Sname = '刘晨' )
     AND Sname != '刘晨'
```

示例

- 例42. 查询成绩为大于90分的学生的学号、姓名。

```
SELECT Sno, Sname FROM Student
   WHERE Sno IN
①  ( SELECT Sno FROM SC
         WHERE Grade > 90 )
```

②

- 例43. 查询选修了“数据库基础”课程的学生的学号、姓名。

```
SELECT Sno, Sname FROM Student
  WHERE Sno IN
    ( SELECT Sno FROM SC
        WHERE Cno IN
          (SELECT Cno FROM Course
            WHERE Cname = '数据库基础') )
```

使用子查询进行比较测试

- 带比较运算符的子查询指父查询与子查询之间用比较运算符连接，
- 当用户能确切知道内层查询返回的是单值时，可用>、<、=、>=、<=、<>运算符。

- 例44. 查询修了‘c02’课程且成绩高于此课程的平均成绩的学生的学号和成绩。

```
SELECT Sno , Grade FROM SC
    WHERE Cno = 'c02'
     and Grade > (
        SELECT AVG(Grade) from SC
            WHERE Cno = 'c02')
```

使用子查询进行存在性测试

- 一般使用EXISTS谓词。
- 带EXISTS谓词的子查询不返回查询的数据，只产生逻辑真值（有数据）和假值（没有数据）。

- 例45.查询选修了‘c01’号课程的学生姓名。

```
SELECT Sname FROM Student
  WHERE EXISTS
     (SELECT * FROM SC
        WHERE Sno = Student.Sno
                    AND Cno = 'c01')
```

注意

- 注1：处理过程为：先外后内；由外层的值决定内层的结果；内层执行次数由外层结果数决定。
- 注2：由于EXISTS的子查询只能返回真或假值，因此在这里给出列名无意义。所以在有EXISTS的子查询中，其目标列表达式通常都用*。

上句的处理过程

1. 找外层表Student表的第一行，根据其Sno值处理内层查询
- 2. 由外层的值与内层的结果比较，由此决定外层条件的真、假
- 3. 顺序处理外层表Student表中的第2、3、...行。

- 例46.查询没有选修‘c01’号课程的学生姓名和所在系。

```
SELECT Sname, Sdept FROM Student
    WHERE NOT EXISTS
        (SELECT * FROM SC
        WHERE Sno = Student.Sno
              AND Cno = 'c01')
```

4.2 数据更改功能

- 4.2.1 插入数据
- 4.2.2 更新数据
- 4.2.3 删除数据

4.2.1 插入数据

- 插入单行记录的INSERT语句的格式为：

  INSERT INTO <表名> [(<列名表>)]

  VALUES （值表）

- 功能：新增一个符合表结构的数据行，将值表数据按表中列定义顺序[或列名表顺序]赋给对应列名。

注意

- 值列表中的值与列名表中的列按位置顺序对应，它们的数据类型必须一致。
- 如果<表名>后边没有指明列名，则新插入记录的值的顺序必须与表中列的定义顺序一致，且每一个列均有值（可以为空）。

示例

- 例1．将新生记录（95020，陈冬，男，信息系，18岁）插入到Student表中。

```
INSERT INTO Student
    VALUES ('9521105', '陈冬', '男', 18, '信息系')
```

- 例2．在SC表中插入一新记录，成绩暂缺。

```
INSERT INTO SC(Sno, Cno, XKLB)
    VALUES('9521105', 'c01', '必修')
```

实际插入的值为：

('9521105', 'c01', NULL, '必修')

4.2.2 更新数据

- 用UPDATE语句实现。
- 格式：

```
UPDATE 〈表名〉
        SET 〈列名＝表达式〉[,... n]
[WHERE 〈更新条件〉]
```

无条件更新

- 例1. 将所有学生的年龄加1。

```
UPDATE Student
    SET Sage = Sage + 1
```

有条件更新

- 1. 基于本表条件的更新
- 例2. 将‘9512101’学生的年龄改为21岁

```
UPDATE Student SET Sage = 21
    WHERE Sno = ’9512101’
```

2. 基于其他表条件的更新

- 例3：将计算机系全体学生的成绩加5分。
- （1）用子查询实现

```
UPDATE SC SET Grade = Grade + 5
    WHERE Sno IN
      (SELECT Sno FROM Student
        WHERE Sdept = '计算机系')
```

- （2）用多表连接实现

```
UPDATE SC SET Grade = Grade + 5
    FROM SC JOIN Student ON SC.Sno = Student.Sno
    WHERE Sdept = '计算机系'
```

4.2.3 删除数据

- 用DELETE语句实现
- 格式：

DELETE [FROM] <表名>

[WHERE <删除条件>

无条件删除

- 例1. 删除所有学生的选课记录。

```
DELETE FROM SC
```

有条件删除

- （1）基于本表条件的删除。
- 例2. 删除所有不及格学生的修课记录。

```
DELETE FROM SC
    WHERE Grade < 60
```

基于其他表条件的删除

- 例3．删除计算机系不及格学生的修课记录。
- （1）用子查询实现

```
DELETE FROM SC
    WHERE Grade < 60 AND Sno IN (
        SELECT Sno FROM Student
            WHERE Sdept = '计算机系' )
```

- （2）用多表连接实现

```
DELETE FROM SC
  FROM SC JOIN Student ON SC.Sno = Student.S
no
```

4.3 查询语句扩展

- 4.3.1 将查询结果保存到新表中
- 4.3.2　CASE表达式
- 4.3.3 查询结果的并、交、差运算

4.3.1 将查询结果保存到新表中

SELECT 查询列表序列 INTO〈新表名〉

FROM 数据源

…

〈新表名〉：存放查询结果的表名。

- 这个语句包含两个功能：
- 根据查询语句产生的列名和类型创建一个新表；
- 执行查询语句并将查询的结果保存到该新表中。

示例

- 例72. 查询计算机系学生的姓名、选修的课程名和成绩，并将查询结果保存到永久表S_C_G中

```
SELECT Sname, Cname, Grade
INTO S_C_G
FROM Student s JOIN SC ON s.Sno = SC.Sno
JOIN Course c ON c.Cno = SC.Cno
WHERE Sdept = '计算机系'
```

示例

- 例73. 统计每个系的学生人数，并将结果保存到永久表dept_cnt中

```
SELECT Sdept, COUNT(*) AS 人数 INTO dept_cnt
FROM Student
GROUP BY Sdept
```

- 注意，这个语句必须为COUNT (*)起列别名，否则无法创建新表。

4.3.2 CASE表达式

- 是一种多分支表达式，它可以根据条件列表的值返回多个可能的结果表达式中的一个。
- CASE表达式可用在任何允许使用表达式的地方，但它不是一个完整的T-SQL语句，因此不能单独执行，只能作为一个可以单独执行的语句的一部分来使用。
  - 简单CASE表达式
  - 搜索CASE表达式

简单CASE表达式

```
CASE input_expression
    WHEN when_expression THEN result_expression
    [ ... n ]
    [ ELSE else_result_expression ]
END
```

示例

- 例74. 查询选了Java课程的学生的学号、姓名、所在系和成绩，并对所在系进行如下处理：
  - 当所在系为“计算机系”时，在查询结果中显示“CS”；
  - 当所在系为“信息系”时，在查询结果中显示“IM”；
  - 当所在系为“数学系”时，在查询结果中显示“COM”。

```
SELECT s.Sno 学号,Sname 姓名,
  CASE Sdept
      WHEN '计算机系' THEN 'CS'
      WHEN '信息系' THEN 'IM'
      WHEN '数学系' THEN 'MA'
  END AS 所在系,Grade 成绩
  FROM Student s join SC ON s.Sno = SC.Sno
  JOIN Course c ON c.Cno = SC.Cno
WHERE Cname = 'Java'
```

搜索CASE表达式

```
CASE
    WHEN Boolean_expr THEN result_expr
    [ ...n ]
    [ ELSE else_result_expr ]
END
```

示例

- 例74用搜索CASE实现：

```
SELECT s.Sno 学号,Sname 姓名,
  CASE
    WHEN Sdept = '计算机系' THEN 'CS'
    WHEN Sdept = '信息系' THEN 'IM'
    WHEN Sdept = '数学系' THEN 'COM'
  END AS 所在系, Grade 成绩
  FROM Student s join SC ON s.Sno = SC.Sno
  JOIN Course c ON c.Cno = SC.Cno
  WHERE Cname = 'Java'
```

示例

- 例75．查询“C001”课程的考试情况，列出学号、成绩以及成绩等级，对成绩等级的处理如下：
  - 如果成绩大于等于90，则等级为“优”；
  - 如果成绩在80到89分之间，则等级为“良”；
  - 如果成绩在70到79分之间，则等级为“中”；
  - 如果成绩在60到69分之间，则等级为“及格”；
  - 如果成绩小于60分，则等级为“不及格”。

例75的实现代码

```
SELECT Sno,Grade, CASE
    WHEN Grade >= 90 THEN '优'
    WHEN Grade between 80 and 89 THEN '良'
    WHEN Grade between 70 and 79 THEN '中'
    WHEN Grade between 60 and 69 THEN '及格'
    WHEN Grade < 60 THEN '不及格'
  END AS 等级
  FROM SC  WHERE Cno = 'C001'
```

示例

- 例76. 统计每个学生的考试平均成绩，列出学号、考试平均成绩和考试情况，其中考试情况的处理为：
  - 如果平均成绩大于等于90，则考试情况为“好”；
  - 如果平均成绩在80～89，则考试情况为“比较好”；
  - 如果平均成绩在70～79，则考试情况为“一般”；
  - 如果平均成绩在60～69，则考试情况为“不太好”；
  - 如果平均成绩低于60，则考试情况为“比较差”。

例76的实现代码

```
SELECT Sno 学号, AVG(Grade) 平均成绩,
  CASE
    WHEN AVG(Grade) >= 90 THEN '好'
    WHEN AVG(Grade) BETWEEN 80 AND 89 THEN '比较好'
    WHEN AVG(Grade) BETWEEN 70 AND 79 THEN '一般'
    WHEN AVG(Grade) BETWEEN 60 AND 69 THEN '不太好'
    WHEN AVG(Grade) < 60 THEN '比较差'
  END AS 考试情况
 FROM SC
 GROUP BY Sno
```

示例

- 例77. 统计计算机系每个学生的选课门数，包括没有选课的学生。列出学号、选课门数和选课情况，其中对选课情况的处理为：
  - 如果选课门数超过4，则选课情况为“多”；
  - 如果选课门数在2~4，则选课情况为“一般”；
  - 如果选课门数少于2，则选课情况为“少”；
  - 如果学生没有选课，则选课情况为“未选”。

例77的实现代码

```
SELECT S.Sno, COUNT(SC.Cno) 选课门数,CASE
   WHEN COUNT(SC.Cno) > 4 THEN '多'
   WHEN COUNT(SC.Cno) BETWEEN 2 AND 4 THEN '一般'
   WHEN COUNT(SC.Cno) BETWEEN 1 AND 2 THEN '少'
   WHEN COUNT(SC.Cno) = 0 THEN '未选'
END AS 选课情况
FROM Student S LEFT JOIN SC ON S.Sno = SC.Sno
WHERE Dept = '计算机系'
GROUP BY S.Sno
ORDER BY COUNT(SC.Cno) DESC
```

示例

- 例78. 修改全体学生的Java考试成绩，修改规则：
  - 对数学系学生，成绩加10分；
  - 对信息系学生，成绩加5分；
  - 对其他系学生，成绩不变。

```
UPDATE SC SET Grade = Grade +
  CASE Dept
    WHEN '数学系' THEN 10
    WHEN '信息系' THEN 5
    ELSE 0
  END
  FROM Student S JOIN SC ON S.Sno = SC.Sno
  JOIN Course C ON C.Cno = SC.Cno
  WHERE Cname = 'Java'
```

4.3.3 查询结果的并、交、差运算

- SELECT语句的查询结果是元组的集合，所以多个SELECT语句的结果可进行集合操作。
- 集合操作主要包括：
  - UNION（并）、
  - INSTERSECT（交）
  - EXCEPT（差）

1. 并运算

- 并运算可将两个或多个查询语句的结果集合并为一个结果集，这个运算可以使用 UNION 运算符实现。
- UNION是一个特殊的运算符，通过它可以实现让两个或更多的查询产生单一的结果集。

UNION语法格式

```
SELECT语句1
 UNION [ ALL ]
SELECT语句2
 UNION [ ALL ]
… …
SELECT语句n
```

- ALL表示在结果集中包含所有查询语句产生的全部记录，包括重复的记录。如果没有指定ALL，则系统默认是删除合并后结果集中的重复记录。

一些说明

- 所有的SELECT语句列表中列的个数必须相同，而且对应列的语义应该相同。
- 各SELECT语句中每个列的数据类型必须兼容。
- 合并后的结果采用第一个SELECT语句的列标题。
- 如果要对查询的结果进行排序，则ORDER BY子句写在最后一个查询语句之后。

示例

- 例79. 查询李勇和刘晨所选的全部课程，列出课程名和开课学期。

```
SELECT Cname, Semester FROM Course C
JOIN SC ON C.Cno = SC.Cno
JOIN Student S ON S.Sno = SC.sno
WHERE Sname = '李勇'
UNION
SELECT Cname, Semester FROM Course C
JOIN SC ON C.Cno = SC.Cno
JOIN Student S ON S.Sno = SC.sno
WHERE Sname = '刘晨'
```

2. 交运算

- 返回同时在两个集合中出现的记录。
- 其语法格式为：

SELECT语句1
INTERSECT
SELECT语句2
INTERSECT
… …
SELECT语句n

示例

- 例80. 查询李勇和刘晨所选的相同的课程（即查询同时被李勇和刘晨选的课程）,列出课程名和学分。

```
SELECT Cname,Credit
  FROM Student S JOIN SC ON S.Sno = SC.Sno
  JOIN Course C ON C.Cno = SC.Cno
  WHERE Sname = '李勇'
INTERSECT
SELECT Cname,Credit
  FROM Student S JOIN SC ON S.Sno = SC.Sno
  JOIN Course C ON C.Cno = SC.Cno
  WHERE Sname = '刘晨'
```

3. 差运算

- 差运算是返回在一个集合中有，但在另一个集合中没有的记录。
- 实现差运算的SQL运算符为EXCEPT，其语法格式为：

SELECT语句1

EXCEPT

SELECT语句2

EXCEPT

…… ……

SELECT语句n

示例

- 例81. 查询李勇选了但刘晨没有选的课程的课程名和开课学期。

```
SELECT C.Cno, Cname, Semester FROM Course C
   JOIN SC ON C.Cno = SC.Cno
   JOIN Student S ON S.Sno = SC.Sno
   WHERE Sname = '李勇'
EXCEPT
SELECT C.Cno, Cname, Semester FROM Course C
   JOIN SC ON C.Cno = SC.Cno
   JOIN Student S ON S.Sno = SC.Sno
   WHERE Sname = '刘晨'
```