

第三章 关系数据库标准语言SQL

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

3.6 空值的处理

3.7 视图

3.8 小结

数据查询

● 语句格式

SELECT [ALL|DISTINCT] <目标列表达式>[,<目标列表达式>] ...

FROM <表名或视图名>[,<表名或视图名>]...(SELECT 语句)

[AS]<别名>

[**WHERE** <条件表达式>]

[**GROUP BY** <列名1> [**HAVING** <条件表达式>]]

[**ORDER BY** <列名2> [ASC|DESC]];

数据查询

- **SELECT子句**：指定要显示的属性列
- **FROM子句**：指定查询对象（基本表或视图）
- **WHERE子句**：指定查询条件
- **GROUP BY子句**：对查询结果按指定列的值分组，该属性列值相等的元组为一个组。通常会在每组中作用聚集函数。
- **HAVING短语**：只有满足指定条件的组才予以输出
- **ORDER BY子句**：对查询结果表按指定列值的升序或降序排序

3.4 数据查询

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 基于派生表的查询

3.4.6 Select语句的一般形式

3. 4. 1 单表查询

- 查询仅涉及一个表

- 1.选择表中的若干列

- 2.选择表中的若干元组

- 3.ORDER BY子句

- 4.聚集函数

- 5.GROUP BY子句

1. 选择表中的若干列

● 查询指定列

[例3.16] 查询全体学生的学号与姓名。

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

[例3.17] 查询全体学生的姓名、学号、所在系。

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

选择表中的若干列（续）

● 查询全部列

➤ 选出所有属性列：

- ✓ 在SELECT关键字后面列出所有列名
- ✓ 将<目标列表表达式>指定为 *

[例3.18] 查询全体学生的详细记录

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

```
FROM Student;
```

或

```
SELECT *
```

```
FROM Student;
```

查询经过计算的值（续）

● 查询经过计算的值

- SELECT子句的<目标列表达式>不仅可以为表中的属性列，也可以是表达式

[例3.19] 查全体学生的姓名及其出生年份。

```
SELECT Sname,2014-Sage          /*假设当时为2014年*/
```

```
FROM Student;
```

输出结果：

Sname	（无列名）
李勇	1994
刘晨	1995
王敏	1996
张立	1995

查询经过计算的值（续）

[例3.20] 查询全体学生的姓名、出生年份和所在的院系，要求用小写字母表示系名。

```
SELECT Sname,'Year of Birth: ',2014-Sage,LOWER(Sdept)
FROM Student;
```

输出结果：

Sname	（无列名）	（无列名）	（无列名）
-------	-------	-------	-------

李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

查询经过计算的值（续）

- 使用列别名改变查询结果的列标题:

```
SELECT Sname NAME, 'Year of Birth:' BIRTH,  
       2014-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

3.4.1 单表查询

- 查询仅涉及一个表:

- 1.选择表中的若干列

- 2.选择表中的若干元组

- 3.ORDER BY子句

- 4.聚集函数

- 5.GROUP BY子句

2. 选择表中的若干元组

● 消除取值重复的行

如果没有指定DISTINCT关键词，则缺省为ALL

[例3.21] 查询选修了课程的学生学号。

```
SELECT Sno FROM SC;
```

等价于：

```
SELECT ALL Sno FROM SC;
```

执行上面的SELECT语句后，结果为：

Sno
201215121
201215121
201215121
201215122
201215122

消除取值重复的行（续）

- 指定DISTINCT关键词，去掉表中重复的行

```
SELECT DISTINCT Sno  
FROM SC;
```

执行结果：

Sno
201215121
201215122

(2) 查询满足条件的元组

表3.6 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（逻辑运算）	AND, OR, NOT

① 比较大小

[例3.22] 查询计算机科学系全体学生的名单。

```
SELECT Sname  
FROM Student  
WHERE Sdept='CS';
```

[例3.23]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;
```

[例3.24]查询考试成绩有不及格的学生的学号。

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

② 确定范围

- 谓词: BETWEEN ... AND ...
NOT BETWEEN ... AND ...

[例3.25] 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;
```

[例3.26] 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 Student
WHERE Sage NOT BETWEEN 20 AND 23;
```


③ 确定集合

- 谓词：IN <值表>, NOT IN <值表>

[例3.27]查询计算机科学系（CS）、数学系（MA）和信息系（IS）学生的姓名和性别。

```
SELECT Sname, Ssex  
FROM Student  
WHERE Sdept IN ('CS','MA','IS');
```

[例3.28]查询既不是计算机科学系、数学系，也不是信息系的学生
的姓名和性别。

```
SELECT Sname, Ssex  
FROM Student  
WHERE Sdept NOT IN ('IS','MA','CS');
```

④ 字符匹配

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

<匹配串>可以是一个完整的字符串，也可以含有通配符%和_

- %（百分号） 代表任意长度（长度可以为0）的字符串
 - ✓ 例如a%b表示以a开头，以b结尾的任意长度的字符串
- _（下横线） 代表任意单个字符。
 - ✓ 例如a_b表示以a开头，以b结尾的长度为3的任意字符串

字符匹配（续）

➤ 匹配串为固定字符串

[例3.29] 查询学号为201215121的学生的详细情况。

```
SELECT *  
FROM Student  
WHERE Sno LIKE '201215121';
```

等价于：

```
SELECT *  
FROM Student  
WHERE Sno = '201215121';
```

如果LIKE后面的匹配串中不含通配符，则可以用=运算符取代LIKE谓词，用<>运算符取代NOT LIKE谓词。

字符匹配（续）

■ 匹配串为含通配符的字符串

[例3.30] 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%';
```

[例3.31] 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__';
```

字符匹配（续）

[例3.32] 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%';
```

[例3.33] 查询所有不姓刘的学生姓名、学号和性别。

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

字符匹配（续）

- 使用换码字符将通配符转义为普通字符

[例3.34] 查询DB_Design课程的课程号和学分。

```
SELECT Cno, Ccredit
FROM   Course
WHERE  Cname LIKE 'DB\_Design' ESCAPE '\';
```

[例3.35] 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 “_” 转义为 “_” 普通字符

思考：除了 ‘\’，其他字符可以变为换码字符么？如果可以，怎么转变？

⑤ 涉及空值的查询

❖ 谓词: IS NULL 或 IS NOT NULL

■ “IS” 不能用 “=” 代替(若代替, 会报错)

[例3.36] 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3.37] 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

- 逻辑运算符：AND和 OR来连接多个查询条件

- AND的优先级高于OR

- 可以用括号改变优先级

```
SELECT * from Student  
WHERE Sno='1' or Ssex='男' and Sage=18;      ?
```

↔ Sno='1' or (Ssex='男' and Sage=18)

[例3.38] 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;
```


多重条件查询（续）

● 改写[例3.27]

[例3.27] 查询计算机科学系（CS）、数学系（MA）和信息系（IS）学生的姓名和性别。

```
SELECT Sname, Ssex  
FROM Student  
WHERE Sdept IN ('CS ','MA ','IS')
```

可改写为：

```
SELECT Sname, Ssex  
FROM Student  
WHERE Sdept= ' CS' OR Sdept= ' MA' OR Sdept= 'IS ';
```

3.4.1 单表查询

- 查询仅涉及一个表:

1. 选择表中的若干列
2. 选择表中的若干元组

3. ORDER BY子句

4. 聚集函数

5. GROUP BY子句

3. ORDER BY子句

- 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

- 对于空值，排序时显示的次序由具体系统实现来决定

ORDER BY子句（续）

[例3.39]查询选修了3号课程的学生学号及其成绩，查询结果按分数降序排列。

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

[例3.40]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;
```

3.4.1 单表查询

- 查询仅涉及一个表:

1. 选择表中的若干列

2. 选择表中的若干元组

3. ORDER BY子句

4. 聚集函数

5. GROUP BY子句

4. 聚集函数

● 聚集函数：

➤ 统计元组个数

COUNT(*)

➤ 统计一列中值的个数

COUNT([DISTINCT|ALL] <列名>)

➤ 计算一列值的总和（此列必须为数值型）

SUM([DISTINCT|ALL] <列名>)

➤ 计算一列值的平均值（此列必须为数值型）

AVG([DISTINCT|ALL] <列名>)

➤ 求一列中的最大值和最小值

MAX([DISTINCT|ALL] <列名>)

MIN([DISTINCT|ALL] <列名>)

聚集函数（续）

[例3.41] 查询学生总人数。

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

[例3.42] 查询选修了课程的学生人数。

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

[例3.43] 计算1号课程的学生平均成绩。

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

聚集函数 （续）

[例3.44] 查询选修1号课程的学生最高分数。

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

[例3.45] 查询学生201215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno='201215012' AND SC.Cno=Course.Cno;
```

聚集函数只能用于SELECT 子句和GROUP BY 中的 HAVING 子句

3.4.1 单表查询

- 查询仅涉及一个表:

- 1.选择表中的若干列

- 2.选择表中的若干元组

- 3.ORDER BY子句

- 4.聚集函数

- 5.GROUP BY子句

5. GROUP BY子句

●GROUP BY子句分组：

细化聚集函数的作用对象

- 如果未对查询结果分组，聚集函数将作用于整个查询结果
- 对查询结果分组后，聚集函数将分别作用于每个组
- 功能：按指定的一系列或多列值分组，值相等的为一组

GROUP BY子句（续）

[例3.46] 求各个课程号及相应的选课人数。

```
SELECT Cno, COUNT(Sno)
```

```
FROM SC
```

```
GROUP BY Cno;
```

查询结果可能为：



COUNT(Sno)

Cno	(无列名)
1	22
2	34
3	44
4	33
5	48

GROUP BY子句（续）

[例3.47] 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;
```



选择组的条件:元组个数>3的组才被选出

GROUP BY子句（续）

[例3.48] 查询平均成绩大于等于87分的学生学号和平均成绩
下面的语句是**不对**的：

```
SELECT Sno, AVG(Grade)
FROM SC
WHERE AVG(Grade)>=87
GROUP BY Sno;
```

错误！

因为**WHERE**子句中是不能用聚集函数作为条件表达式
正确的查询语句应该是：

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade)>=87;
```

	Sno	(无列名)
1	201215121	88

GROUP BY子句（续）

- **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。

思考题

[例] 查询有至少3门课程成绩是85分及85分以上的学生学号，及该同学成绩获得85分以上课程的门数。

```
SELECT Sno, COUNT(*) cx_num  
FROM SC  
WHERE Grade>=85  
GROUP BY Sno  
HAVING COUNT(*)>=3;
```

	Sno	cx_num
1	201215121	3

首先选出**Grade>=85**的元组，形成中间结果；然后按照**Sno**分组，统计每组元组个数；最后筛选出输出结果。

第三章 关系数据库标准语言SQL

3.1 SQL概述

3.2 学生-课程数据库

3.3 数据定义

3.4 数据查询

3.5 数据更新

3.6 空值的处理

3.7 视图

3.8 小结

3.4 数据查询

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 基于派生表的查询

3.4.5 Select语句的一般形式

3.4.2 连接查询

- 连接查询：同时涉及两个以上的表的查询
- 连接条件或连接谓词：用来连接两个表的条件
一般格式：
 - [<表名1>.<列名1> <比较运算符> [<表名2>.<列名2>
 - [<表名1>.<列名1> **BETWEEN** [<表名2>.<列名2> **AND** [<表名2>.<列名3>
- 连接字段：连接谓词中的列名称
 - 连接条件中的各连接字段类型必须是可比的，但名字不必相同

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

1. 等值与非等值连接查询

- 等值连接：连接运算符为=

[例 3.49] 查询每个学生及其选修课程的情况

```
SELECT Student.*, SC.*  
  
FROM Student, SC  
  
WHERE Student.Sno = SC.Sno;
```

等值与非等值连接查询（续）

查询结果：

Student.Sno	Sname	Ssex	Sage	Sdept	SC.Sno	Cno	Grade
201215121	李勇	男	20	CS	201215121	1	92
201215121	李勇	男	20	CS	201215121	2	85
201215121	李勇	男	20	CS	201215121	3	88
201215122	刘晨	女	19	CS	201215122	2	90
201215122	刘晨	女	19	CS	201215122	3	80

连接操作的执行过程

1) 嵌套循环法 (NESTED-LOOP)

- 首先在表1中找到第一个元组，然后从头开始扫描表2，逐一查找满足连接件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组。
- 表2全部查找完后，再找表1中第二个元组，然后再从头开始扫描表2，逐一查找满足连接条件的元组，找到后就将表1中的第二个元组与该元组拼接起来，形成结果表中一个元组。
- 重复上述操作，直到表1中的全部元组都处理完毕

连接操作的执行过程（续）

2) 索引连接（INDEX-JOIN）

- 对表2按连接字段建立索引
- 对表1中的每个元组，依次根据其连接字段值查询表2的索引，从中找到满足条件的元组，找到后就将表1中的第一个元组与该元组拼接起来，形成结果表中一个元组

等值与非等值连接查询（续）

- 自然连接

[例 3.50] 对[例 3.49]用自然连接完成。

```
SELECT  Student.Sno, Sname,Ssex,Sage,Sdept,Cno,Grade  
  
FROM    Student,SC  
  
WHERE   Student.Sno = SC.Sno;
```


等值与非等值连接查询（续）

- 一条SQL语句可以同时完成选择和连接查询，这时WHERE子句是由连接谓词和选择谓词组成的复合条件。

[例 3.51] 查询选修2号课程且成绩在90分以上的所有学生的学号和姓名。

```
SELECT Student.Sno, Sname  
FROM   Student, SC  
WHERE  Student.Sno=SC.Sno AND  
       (SC.Cno=' 2 ' AND SC.Grade>90);
```

➤ 执行过程:

- ✓ 先从SC中挑选出Cno='2'并且Grade>90的元组形成一个中间关系
- ✓ 再和Student中满足连接条件的元组进行连接得到最终的结果关系

等值与非等值连接查询（续）

连接运算符 不是 = 的连接操作

[<表名1>.]<列名1><比较运算符>[<表名2>.]<列名2>

比较运算符： >、<、>=、<=、!=

[<表名1>.]<列名1> BETWEEN [<表名2>.]<列名2> AND [<表名2>.]<列名3>

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

2. 自身连接

- 自身连接：一个表与其自己进行连接, 需要给表起别名以示区别
- 由于所有属性名都是同名属性, 因此必须使用别名前缀

[例 3.52] 查询每一门课的间接先修课 (即先修课的先修课)

```
SELECT FIRST.Cno, SECOND.Cpno
```

```
FROM Course FIRST, Course SECOND
```

```
WHERE FIRST.Cpno = SECOND.Cno;
```

--> 起别名

自身连接（续）



FIRST表（Course表）

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL 语言	6	4

SECOND表（Course表）

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL 语言	6	4

自身连接（续）

查询结果：

Cno	Cpno
1	7
3	5
5	6

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

3. 外连接

- 外连接与普通连接的区别

- 普通连接操作只输出满足连接条件的元组
- 外连接操作以指定表为连接主体，将主体表中不满足连接条件的元组一并输出
- 左外连接
 - ✓ 列出左边关系中所有的元组
- 右外连接
 - ✓ 列出右边关系中所有的元组

外连接（续）

[例 3. 53] 改写[例 3.49]

```
SELECT Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade  
FROM Student LEFT JOIN SC ON  
      (Student.Sno=SC.Sno);
```

外连接（续）

执行结果：

Student.Sno	Sname	Ssex	Sage	Sdept	Cno	Grade
201215121	李勇	男	20	CS	1	92
201215121	李勇	男	20	CS	2	85
201215121	李勇	男	20	CS	3	88
201215122	刘晨	女	19	CS	2	90
201215122	刘晨	女	19	CS	3	80
201215123	王敏	女	18	MA	NULL	NULL
201215125	张立	男	19	IS	NULL	NULL

连接查询（续）

1.等值与非等值连接查询

2.自身连接

3.外连接

4.多表连接

4. 多表连接

- 多表连接：两个以上的表进行连接

[例3.54]查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT Student.Sno, Sname, Cname, Grade  
FROM   Student, SC, Course  /*多表连接*/  
WHERE  Student.Sno = SC.Sno  
        AND SC.Cno = Course.Cno;
```