



2024年春季学期

数据库系统概论

An Introduction to Database Systems

第三章 关系数据库标准语言SQL

中国科学技术大学 大数据学院

黄振亚, huangzhy@ustc.edu.cn



第三章 关系数据库标准语言SQL

129

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

130

□ 语句格式

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

131

- **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

132

- 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语句的一般形式



3.4.1 单表查询

133

- 查询仅涉及一个表：
 - 一、 选择表中的若干列
 - 二、 选择表中的若干元组
 - 三、 **ORDER BY**子句
 - 四、 聚集函数
 - 五、 **GROUP BY**子句



一、选择表中的若干列

134

□ 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;
```

- <目标列表达式>属性顺序可以不一致
- 查询结果是一个关系



一、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。

```
183 ● SELECT Sno,Sname FROM Student;
184
```

100% 1:181

Result Grid Filter Rows: Search Edit:

Sno	Sname
200215122	LC
200215121	LY
200215123	WM
200515125	ZL

[例

-
-

类似于关系代数中的哪一个操作?



2. 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;
```

```
5 • SELECT *  
6 FROM student;  
7
```

Sno	Sname	Ssex	Sage	Sdept
200215121	LY	M	20	CS
200215122	LC	W	19	CS
200215123	WM	W	18	MA
200215125	ZL	M	19	IS



3. 查询经过计算的值

137

- **SELECT**子句的<目标列表达式>可以为:
 - 算术表达式
 - 字符串常量
 - 函数
 - 列别名



查询经过计算的值（续）

138

[例3.19] 查全体学生的姓名及其出生年份。

```
SELECT Sname, 2014-Sage
```

/算术表达式，*假定当年的年份为2014年*/

```
FROM Student;
```

输出结果：

```
Sname 2014-Sage
```

李勇	1994
----	------

刘晨	1995
----	------

王敏	1996
----	------

张立	1995
----	------



查询经过计算的值（续）

139

[例3.20] 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:', 2023-Sage,  
        LOWER(Sdept)  
FROM Student;
```

输出结果：

```
Sname 'Year of Birth:' 2023-Sage LOWER(Sdept)
```

李勇	Year of Birth:	2003	cs
刘晨	Year of Birth:	2004	is
王敏	Year of Birth:	2005	ma
张立	Year of Birth:	2004	is



查询经过计算的值（续）

140

- 使用列别名改变查询结果的列标题:

```
SELECT Sname NAME, 'Year of Birth: ' BIRTH,  
       2023-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	2003	cs
刘晨	Year of Birth:	2004	is
王敏	Year of Birth:	2005	ma
张立	Year of Birth:	2004	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(Mysql 例子)

```
SELECT Sname NAME,  
'Year of Birth:' BIRTH,  
2023-Sage BIRTHDAY,  
lower(Sdept) DEPARTMENT  
FROM Student;
```

	NAME	BIRTH	BIRTHDAY	DEPARTMENT
▶	LY	Year of Birth:	2003	cs
	LC	Year of Birth:	2004	cs
	WM	Year of Birth:	2005	ma
	ZL	Year of Birth:	2004	is



3.4.1 单表查询

142

- 查询仅涉及一个表：
 - 一、 选择表中的若干列
 - 二、 选择表中的若干元组
 - 三、 **ORDER BY**子句
 - 四、 聚集函数
 - 五、 **GROUP BY**子句



二、选择表中的若干元组

□ 1. 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
200215121
200215121
200215121
200215122
200215122

与关系代数的区别？



消除取值重复的行（续）

144

- 指定DISTINCT关键词，去掉表中重复的行

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

执行结果：

<u>Sno</u>
200215121
200215122



2. 查询满足条件的元组

表3.4 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

```

14 • SELECT Sname
15 FROM Student
16 WHERE Sdept='CS';
17

```

[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'; # WHERE Sdept="CS";

```

Result Grid | Filter Rows:

Sname
LY
LC

[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;

```

执行过程?



(2) 确定范围

147

- 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

148

- 谓词: **IN** <值表>, **NOT IN** <值表> <值表>是一个集合

[例3.27] 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'); #("IS", "MA", "CS");
```

其他语句? 逻辑表达式

[例3.28] 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');
```



(4) 字符匹配

149

- 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



(4) 字符匹配

150

1) 匹配串为固定字符串

[例3.29] 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';
```



字符匹配（续）

2) 匹配串为含通配符的字符串

%: 任意多个字符; _: 单个字符（汉字为2个字符）

【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%';
```

```
18 • SELECT Sname,Sno,Ssex  
19 FROM Student  
20 WHERE Sname LIKE 'L%';
```

Result Grid | Filter Rows: | Edit:

Sname	Sno	Ssex
LC	200215122	W
LY	200215121	M
HULL	HULL	HULL

【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__'; （两个下划线）
```




字符匹配（续）

152

[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。

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



字符匹配（续）

153

3) 使用换码字符 **ESCAPE '\'** 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



实例-练习

154

```
SELECT Cno FROM Course WHERE Cname LIKE "DB\_Design";
```

```
SELECT Cno FROM Course WHERE Cname LIKE "DB_Design";
```

```
SELECT Cno FROM Course WHERE Cname LIKE "DB/_Design" escape "/";
```

结果分别是什么？

	Cno	Cname	Cpno	Ccredit
▶	1	DB_Design	5	4
	2	MATH	NULL	2
	3	Information System	1	4
	4	Operating System	6	3
	5	Data Structure	7	4
	6	Data Processing	NULL	2
	7	PASCAL	6	4
	NULL	NULL	NULL	NULL



实例-练习

```
187 * INSERT INTO Course (Cno, Cname, Cpno, Ccredit)
188 187 * INSERT INTO Course (Cno, Cname, Cpno, Ccredit)
189 188 VALUES (8, "DBEDesign", 5, 4);
190 189 187 * INSERT INTO Course (Cno, Cname, Cpno, Ccredit)
191 190 188 VALUES (8, "DBEDesign", 5, 4);
192 191 189 * SELECT * FROM Course WHERE Cname LIKE "DB\_Design" ;
193 192 190 * SELECT * FROM Course WHERE Cname LIKE "DB_Design";
100% 193 191 * SELECT * FROM Course WHERE Cname LIKE "DB/_Design" escape "/";
Result 100% 192 * Delete From Course where Cno = 8;
193
```

Result Grid

Cno	Cname	Cpno	Ccredit
1	DB_Design	5	4
NULL	NULL	NULL	NULL

#right 能找到DB_Design

#right,这里_表示通配符, 它能找出来DB_Design 和 DBEDesign

#right,这里_不表示通配符, 它能找出来DB_Design



(5) 涉及空值的查询

156

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

[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

157

- 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 ...

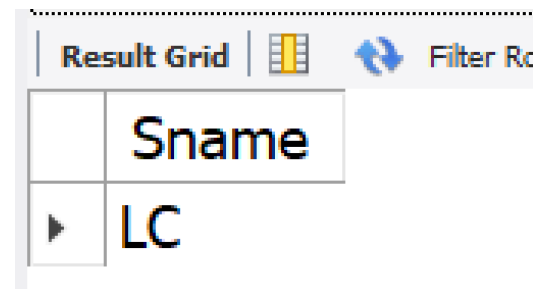


多重条件查询（续）

[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;
```

Sno	Sname	Ssex	Sage	Sdept
200215121	LY	M	20	CS
200215122	LC	W	19	CS
200215123	WM	W	18	MA
200215125	ZL	M	19	IS
NULL	NULL	NULL	NULL	NULL





多重条件查询（续）

159

□ 改写[例3.27]

[例3.27] 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 ';
```




3.4.1 单表查询

162

- 查询仅涉及一个表：
 - 一、 选择表中的若干列
 - 二、 选择表中的若干元组

 - 三、 **ORDER BY**子句
 - 四、 聚集函数
 - 五、 **GROUP BY**子句



三、ORDER BY子句

163

- ORDER BY子句（对结果进行排序）
 - 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



ORDER BY子句 (续)

164

[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;
```



ORDER BY子句 (续)

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

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

Sno	Grade
200215121	88
200215122	80

[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(Mysql 例子)

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

Sno	Sname	Ssex	Sage	Sdept
200215121	LY	M	20	CS
200215122	LC	W	19	CS
200515125	ZL	M	19	IS
200215123	WM	W	18	MA



三、ORDER BY子句

当排序列含空值时，排序时显示的次序由具体系统实现决定

- 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

MySQL的实现方式

```

44 • SELECT *
45 FROM Course
46 ORDER BY Cjno ASC;

```

Cno	Cname	Cjno	Ccredit
2	MATH	NULL	2
6	Data Processing	NULL	2
3	Information System	1	4
1	DB Design	5	4

```

44 • SELECT *
45 FROM Course
46 ORDER BY Cjno DESC;

```

Cno	Cname	Cjno	Ccredit
4	Operating System	6	3
7	PASCAL	6	4
1	DB_Design	5	4
3	Information System	1	4
2	MATH	NULL	2
6	Data Processing	NULL	2



3.4.1 单表查询

167

- 查询仅涉及一个表：
 - 一、 选择表中的若干列
 - 二、 选择表中的若干元组

 - 三、 **ORDER BY**子句
 - 四、 **聚集函数**
 - 五、 **GROUP BY**子句



四、聚集函数

168

❖ 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）

169

[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）

170

[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200215012选修课程的总学分数。

```
SELECT SUM(Ccredit)
FROM SC, Course （连接查询，后面介绍）
WHERE Sno='200215012' AND SC.Cno=Course.Cno;
```



聚集函数 (续)

[例3.44] 查询选修1号课程的学生最高分数(Mysql 例子)

```
select MAX(Grade)
from SC
where Cno='1';
```

MAX(Grade)
92

[例3.45] 查询学生200215121选修课程的总学分数(Mysql 例子)

```
select SUM(Ccredit)
from SC, Course
where Sno='200215121' AND SC.Cno=Course.Cno;
```

SUM(Ccredit)
10



聚集函数 (续)

1 Result Grid Filter Rows: Edit:

Cno	Cname	Cpno	Ccredit
1	DB_Design	5	4
2	MATH	NULL	2
3	Information System	1	4
4	Operating System	6	3
5	Data Structure	7	4
6	Data Processing	NULL	2
7	PASCAL	6	4

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

[例3.45] 查询学生**200215121**选修课程的总学分数(Mysql 例子)

```
select SUM(Ccredit)
from SC, Course
where Sno='200215121' AND SC.Cno=Course.Cno;
```

Result Grid Filter Rows

SUM(Ccredit)
10



聚集函数 (续)

173

- 聚集函数遇到空值NULL?
 - Count (*)
 - Count (Sage)
 - MAX (grade)
 - SUM (Ccredit)

[例3.43] 计算1号课程的学生平均成绩。

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

如果有学生没有成绩怎么办?

课后尝试



3.4.1 单表查询

174

- 查询仅涉及一个表：
 - 一、 选择表中的若干列
 - 二、 选择表中的若干元组

 - 三、 **ORDER BY**子句
 - 四、 聚集函数
 - 五、 **GROUP BY**子句



五、GROUP BY子句

175

□ GROUP BY子句分组：

细化聚集函数的作用对象

- 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**
- 作用对象是**查询的中间结果表**
- 按指定的一列或多列值分组，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:

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



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;
```

如果删除group by?

```
57 • SELECT Cno,COUNT(Sno)
58 FROM SC
59 GROUP BY Cno;
```

```
61 • SELECT Cno,COUNT(Sno)
62 FROM SC;
--
```

result Grid | Filter Rows:

Cno	COUNT(Sno)
1	1
2	2
3	2

```
24 19:25:53 SELECT Cno,COUNT(Sno) FROM SC LIMIT 0, 1000
```

Error Code: 1140. In aggregated query without GROUP BY, expressi...



GROUP BY子句 (续)

178

[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;
```



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;
```

执行顺序

Sno	Cno	Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80
NULL	NULL	NULL

- 1. GROUP BY Sno
- 2. Count(*)

sno	Count(*)
200215121	3
200215122	2

3. Having

sno	Count(*)
200215121	3



GROUP BY子句（续）

180

[例3.48] 查询平均成绩大于等于90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)>=90
GROUP BY Sno;
```

因为WHERE子句中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)>=90;
```



GROUP BY子句 (续)

[例3.48] 查询平均成绩大于等于80分的学生学号和平均成绩(Mysql 例子)

```
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade) >= 80;
```

	Sno	Cno	Grade
▶	200215121	1	92
	200215121	2	85
	200215121	3	88
	200215122	2	90
	200215122	3	80

Result Grid Filter Rows:

	Sno	AVG(Grade)
▶	200215121	88.3333
	200215122	85.0000



GROUP BY子句（续）

182

- **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。

注意：**where**子句不能用聚集函数作为条件表达式，聚集函数只用于**select**子句和**group by**子句中的**having**语句