



第三章 SQL语言



SQL语言概述

- 1970年，美国IBM研究中心的E.F.Codd连续发表多篇论文，提出关系模型。
- 1972年，IBM公司开始研制实验型关系数据库管理系统SYSTEM R，配制的查询语言称为SQUARE (Specifying Queries As Relational Expression)语言，在语言中使用了较多的数学符号。
- 1974年，Boyce和Chamberlin把SQUARE修改为SEQUEL (Structured English Query Language)语言。后来SEQUEL简称为**SQL (Structured Query Language)**，即“结构化查询语言”。



SQL语言概述

■ 有关标准

- 随着SQL语言应用的日益广泛，ANSI和ISO先后制定了多个SQL标准：
 - ✓ SQL-86：较为简单，主要包括数据定义语言、数据操纵语言、嵌入式语法等几个部分。
 - ✓ SQL-89：增加了对完整性约束的支持。
 - ✓ SQL-92：也称SQL2，是SQL-89的超集，增加了许多新特性，如新的数据类型，更丰富的数据操作，更强的完整性、安全性支持等。
 - ✓ SQL-99：增加对面向对象模型的支持等许多新特征。



SQL语言概述

■ SQL的特点

- 综合统一
- 高度非过程化
- 面向集合的操作方式
- 以同一种语法结构提供两种使用方法
- 语言简洁，易学易用



SQL语言概述

■ SQL的特点：综合统一

- SQL是一种一体化的语言，它包括了数据定义、数据查询、数据操纵和数据控制等方面的功能，它可以完成数据库活动中的全部工作。而以前的非关系模型的数据语言一般包括存储模式描述语言、概念模式描述语言、外部模式描述语言和数据操纵语言等等，这种模型的数据语言，一是内容多，二是掌握和使用起来都不象SQL那样简单、实用。



SQL语言概述

■ SQL的特点：高度非过程化

- SQL语言是一种高度非过程化的语言，它没有必要一步步地告诉计算机“如何”去做，而只需要描述清楚用户要“做什么”，SQL语言就可以将要求交给系统，自动完成全部工作。



SQL语言概述

■ SQL的特点：面向集合的操作

- 非关系模型（层次、网状）采用的是面向记录的操作方式，操作对象是一条记录。例如查询所有平均成绩在80分以上的学生姓名，用户必须一条一条地把满足条件的学生记录找出来（通常要说明具体处理过程，即按照哪条路径，如何循环等）。而SQL语言采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操作的对象也可以是元组的集合。



SQL语言概述

■ SQL的特点：两种使用方式

- SQL语言可以直接以**命令方式**交互使用，也可以嵌入到程序设计语言中以**程序方式**使用。现在很多数据库应用开发工具，都将SQL语言直接溶入到自身的语言之中，使用起来更方便。这些使用方式为用户提供了灵活的选择余地。此外，尽管SQL的使用方式不同，但SQL语言的语法基本是一致的。



SQL语言概述

■ SQL的特点：易学易用

- SQL语言非常简洁，虽然SQL语言功能很强，但它只有为数不多的几条命令，下表给出了分类的命令动词，另外SQL的语法也非常简单，它很接近自然语言（英语），因此容易学习、掌握。

SQL 功能	命令动词
数据查询	SELECT
数据定义	CREATE、 DROP、 ALTER
数据操纵	INSERT、 UPDATE、 DELETE
数据控制	GRANT、 REVOKE



SQL语言概述

■ SQL的组成

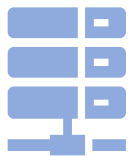
- 核心SQL主要有四个部分:

(1) 数据定义语言, 即SQL DDL(Data Definition Language), 用于定义基本表、视图、索引等结构。

(2) 数据查询语言 (Query Language), 从数据库获取指定的数据, 是数据库的核心操作。

(3) 数据操纵语言, 即SQL DML(Data Manipulation Language), 分成插入、删除和修改三种操作。

(4) 数据控制语言, 即SQL DCL(Data Control Language), 这一部分包括对基本表和视图的授权、完整性规则的描述、事务控制等内容。



第三章 SQL语言

1

查询语言

2

DDL语言

3

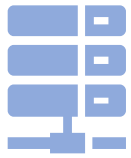
DML语言

4

DCL语言

4

视图



查询语言

■ 查询语句基本结构:

SELECT desired attributes

FROM one or more tables

WHERE condition about tuples of
 the tables

GROUP BY one or more attributes

ORDER BY one or more attributes;



查询语言

■ 查询语句基本结构:

- 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集函数。
- HAVING短语: 筛选出只有满足指定条件的组
- ORDER BY子句: 对查询结果表按指定列值的升序或降序排序



查询语言

■ 用到的示例关系模型

学生-课程数据库

□ 学生表: Student(Sno, Sname, Ssex, Sage, Sdept)

□ 课程表: Course(Cno, Cname, Cpno, Ccredit)

□ 学生选课表: SC(Sno, Cno, Grade)



查询语言

- 主要内容：

- 单表查询

- 聚集和分组

- 多表查询

- 子查询

- 集合查询



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、对查询结果进行分组



单表查询

[例1] 查询全体学生的学号与姓名。

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

[例2] 查询全体学生的姓名、学号、所在系。

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



单表查询

[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;
```



单表查询

■ 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



单表查询

■ 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, 2022-Sage
```

```
FROM Student;
```

输出结果：

Sname	2022-Sage
-------	-----------

李勇	2000
----	------

刘晨	2001
----	------

王名	2001
----	------

张立	2002
----	------



单表查询

■查询经过计算的值

[例5] 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 , 2020-Sage AS 'Year of Birth: ' ,  
       LOWER(Sdept)  
FROM Student;
```



单表查询

■ 查询经过计算的值

使用列别名改变查询结果的列标题

```
SELECT Sname AS NAME, 'Year of Birth: ' AS BIRTH,  
       2022-Sage AS BIRTHDAY, LOWER(Sdept) AS DEPARTMENT  
FROM Student;
```

输出结果:

NAME	BIRTH	BIRTHDAY	DEPARTMENT
-----	-----	-----	-----
李勇	Year of Birth:	2001	cs
刘晨	Year of Birth:	2002	is
王名	Year of Birth:	2000	ma
张立	Year of Birth:	2001	is



单表查询

■ 选择表中的若干元组

● 保留重复行

用ALL关键字

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

ALL为默认关键字，可以省略，等价于：

```
SELECT Sno FROM SC;
```

● 消除取值重复的行

在SELECT子句中使用DISTINCT短语消除重复行

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



单表查询

- 选择表中的若干元组
 - 消除取值重复的行

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;
```




单表查询

■ 选择表中的若干元组

● 查询满足条件的元组

用WHERE子句表达查询条件

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



单表查询

■ 选择表中的若干元组

● 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6] 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;
```



单表查询

■ 选择表中的若干元组

● 确定范围

✓ 使用谓词 BETWEEN ... AND ...

NOT BETWEEN ... AND ...

[例7] 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;
```



单表查询

■ 选择表中的若干元组

● 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8] 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' );
```



单表查询

■ 选择表中的若干元组

● 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9] 查询既不是信息系、数学系，也不是计算机科学系的学生姓名和性别。

```
SELECT Sname, Ssex
```

```
FROM Student
```

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



单表查询

■ 选择表中的若干元组

● 字符串匹配

✓ WHERE 子句中可以对字符串进行模版匹配，形如：

WHERE <Attribute> LIKE <pattern>

or <Attribute> NOT LIKE <pattern>

其中<pattern>是匹配模板，即：固定字符串或含通配符的字符串
通配符：

% (百分号) 代表任意长度（长度可以为0）的字符串

_ (下横线) 代表任意单个字符



单表查询

■ 选择表中的若干元组

● 字符串匹配

[例10] 查询学号为95001的学生的详细情况。

```
SELECT * FROM Student  
WHERE Sno LIKE '95001';
```

等价于:

```
SELECT * FROM Student  
WHERE Sno = '95001';
```



单表查询

- 选择表中的若干元组
 - 字符串匹配

[例11] 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%' ;
```




单表查询

- 选择表中的若干元组
 - 字符串匹配

[例12] 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_';
```



单表查询

- 选择表中的若干元组
 - 字符串匹配

[例13] 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%';
```



单表查询

- 选择表中的若干元组
 - 字符串匹配

[例14] 查询所有不姓刘的学生姓名。

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



单表查询

■ 选择表中的若干元组

● 字符串匹配

- ✓ 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15] 查询DB_Design课程的课程号和学分。

```
SELECT Cno, Ccredit
```

```
FROM Course
```

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

```
ESCAPE '\'
```

! Mysql不能用\作为转义符



单表查询

■ 选择表中的若干元组

● 字符串匹配

- ✓ 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16] 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 '*';
```



单表查询

■ 选择表中的若干元组

● 字符串匹配

- ✓ 当进行固定字符串匹配时可以用 '=' 代替 'like'
- ✓ 对于包含单引号的字符串，在条件表达式中用双引号代替单引号

[例16] 查询课程名为 '数据库' 的课程的具体情况。

```
SELECT *
```

```
FROM Course
```

```
WHERE Cname = ' ' ' 数据库' ' ' ;
```



单表查询

■ 选择表中的若干元组

● 多重条件查询

- ✓ 用逻辑运算符AND, OR和 NOT来联结多个查询条件

[例17] 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;
```



单表查询

■ 选择表中的若干元组

● 多重条件查询

[例18] 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 ';
```




单表查询

■ 选择表中的若干元组

● 多重条件查询

[例19]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;
```



单表查询

- 选择表中的若干元组
 - 涉及空值的查询

Sno	Sname	Age
1001	张三	NULL

对上表执行如下查询：

```
SELECT Sno
```

```
FROM Stu
```

```
WHERE Age < 20 OR Age >= 20;
```

查询结果是什么？



单表查询

■ 选择表中的若干元组

● 涉及空值的查询

规则：

- ✓ Where 语句中的条件表达式有三种可能的计算结果：
True, False, 或者 UnKnown
- ✓ 任何一个值与NULL进行比较, 返回的结果是UnKnown
- ✓ Where子句对被查询表中每一条记录进行条件表达式的计算, 只有在计算结果为True时当前记录才会被选中



单表查询

■ 选择表中的若干元组

● 涉及空值的查询

三值逻辑计算(Three-Valued Logic):

TRUE = 1, FALSE = 0, and UNKNOWN = $\frac{1}{2}$

AND = MIN; OR = MAX, NOT(x) = $1-x$

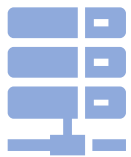
例:

TRUE AND (FALSE OR NOT(UNKNOWN))

= MIN(1, MAX(0, (1 - $\frac{1}{2}$)))

= MIN(1, MAX(0, $\frac{1}{2}$))

= MIN(1, $\frac{1}{2}$) = $\frac{1}{2}$.



单表查询

- 选择表中的若干元组
 - 涉及空值的查询

```
SELECT Sno
FROM Stu
WHERE Age < 20 OR Age >= 20;
```

<-----> <----->
UNKNOWN UNKNOWN
<----->
UNKNOWN

所以没有任何一条age为NULL记录会被选中



单表查询

■ 选择表中的若干元组

● 涉及空值的查询

- ✓ 测试是否空值需用谓词 IS NULL 或 IS NOT NULL
- ✓ “IS NULL” 不能用 “= NULL” 代替

[例20] 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;
```



单表查询

- 选择表中的若干元组
 - 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] 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;
```



单表查询

■ 对查询结果进行排序

● 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

Note: Order子句只能出现在select语句的最后部分



单表查询

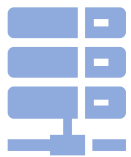
■ 对查询结果进行排序

[例22] 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23] 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;
```



查询语言

- 主要内容:

- 单表查询

- 聚集和分组

- 多表查询

- 子查询

- 集合查询



单表查询

■ 使用集函数 (aggregation function)

- 在Select子句中可以使用集函数，对指定的列进行聚合计算

- 计数 COUNT ([DISTINCT|ALL] <列名> | *)
- 计算总和 SUM ([DISTINCT|ALL] <列名>)
- 计算平均值 AVG ([DISTINCT|ALL] <列名>)
- 求最大值 MAX ([DISTINCT|ALL] <列名>)
- 求最小值 MIN ([DISTINCT|ALL] <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



单表查询

■ 使用集函数

[例24] 查询学生总人数。

```
SELECT COUNT(stuid, stuname)
FROM Student;
```

[例25] 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



单表查询

■ 使用集函数

[例26] 计算1号课程的学生平均成绩。

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

[例27] 查询选修1号课程的学生最高分数。

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



单表查询

■ 使用集函数 (aggregation function)

● 集函数中的空值NULL

- 空值不加入SUM、AVG和COUNT的计算, 也不会成为列中的MIN、MAX值
- 但如果列中没有非空值, 则集函数结果会返回空值

Sno	Cno	Grade
100	1001	2
101	1001	NULL

```
SELECT count(*)  
FROM SC  
WHERE Cno = '1001';
```

VS.

```
SELECT count(Grade)  
FROM SC  
WHERE Cno = '1001';
```



单表查询

■ 对查询结果进行分组

- 使用GROUP BY子句分组
 - ✓ 细化集函数的作用对象
 - ✓ 未对查询结果分组时，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
- 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



单表查询

■ 对查询结果进行分组

● 使用GROUP BY子句分组

Select * From - Where 查询结果:

A	B	C
1	a	o
1	b	p
2	a	o
2	c	p
3	a	o

对它进行分组, 如: Group by A



单表查询

■ 对查询结果进行分组

● 使用GROUP BY子句分组

对它进行分组，如：Group by A

根据A的取值不同，分成了三组

A	B	C
1	a	o
1	b	p
2	a	o
2	c	p
3	a	o



单表查询

■ 对查询结果进行分组

● 使用GROUP BY子句分组

可以对每一组进行集函数操作，如count (*)

每一个分组会对应最终查询结果中的一条记录

最终输出结果可以是分组字段 (A) 和/或集函数的值

Select A, count(*) From TB Where ... Group by A

A	B	C		A	Count(*)
1	a	o	}	1	2
1	b	p			
2	a	o	}	2	2
2	c	p			
3	a	o	→	3	1



单表查询

■ 对查询结果进行分组

● 使用GROUP BY子句分组

[例28] 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



单表查询

■ 对查询结果进行分组

● 使用GROUP BY子句分组

[例29] 求各个课程号，相应的选课人数，和选课学生的年级。

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

错误！ Grade不是被分组的字段，也没有被包含在集函数中

为什么？



单表查询

■ 对查询结果进行分组

● 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。
- ✓ HAVING子句中出现的列只能是在group by子句或集函数中出现的列



单表查询

- 对查询结果进行分组
 - 使用Having短语筛选最终输出结果

[例30] 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;
```



单表查询

- 对查询结果进行分组
 - 使用Having短语筛选最终输出结果

[例31] 查询有3门以上课程是90分以上的学生的学号及（90分以上的）课程数

```
SELECT Sno, COUNT(*)  
FROM SC  
WHERE Grade >= 90  
GROUP BY Sno  
HAVING COUNT(*) >= 3;
```



单表查询

- 对查询结果进行分组
 - 使用Having短语筛选最终输出结果

Having VS. Where:

Where子句不能使用集函数

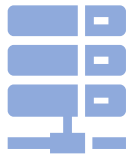
求选了3门课以上的学生学号

```
select sno from SC  
where count(*)>=3 group by sno
```



```
select sno from SC  
group by sno having count(*)>=3
```





单表查询

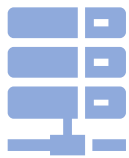
■ Grouping and Aggregation

```
SELECT  S  
FROM    R1,...,Rn  
WHERE   C1  
GROUP BY a1,...,ak  
HAVING  C2
```

S = may contain attributes a_1, \dots, a_k and/or any aggregates but NO OTHER ATTRIBUTES

C1 = is any condition on the attributes in R_1, \dots, R_n

C2 = is any condition on aggregate expressions



单表查询

■ Grouping and Aggregation

```
SELECT  S  
FROM    R1,...,Rn  
WHERE   C1  
GROUP BY a1,...,ak  
HAVING  C2
```

Evaluation steps:

1. Compute the FROM-WHERE part, obtain a table with all attributes in R_1, \dots, R_n
2. Group by the attributes a_1, \dots, a_k
3. Compute the aggregates in C2 and keep only groups satisfying C2
4. Compute aggregates in S and return the result

作业

- ❑ 课本第130页第5题1~7小题
- ❑ SQL练习.doc
- ❑ 作业在下下次上课（4月11日）前提交