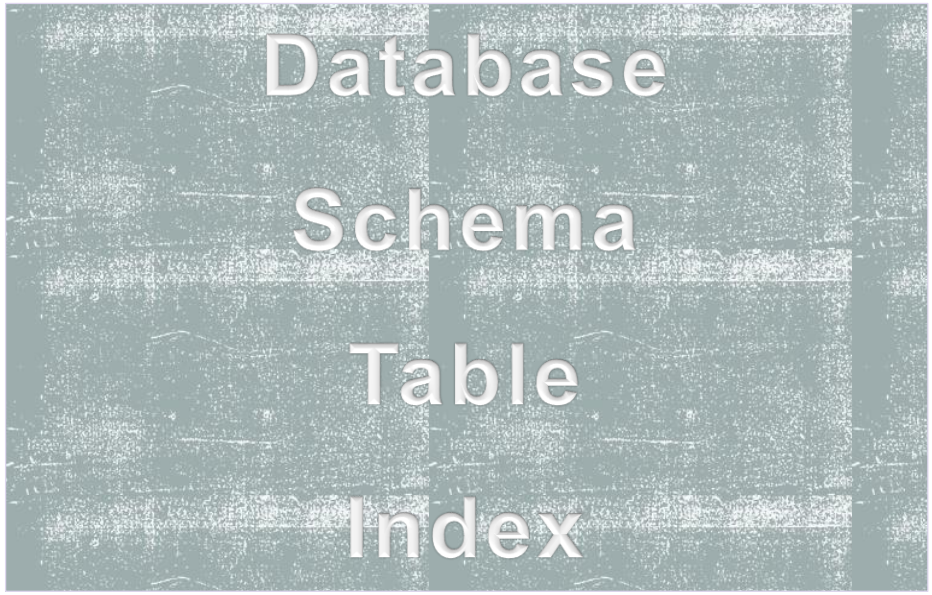
第三章 关系数据库标准语言SQL

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 视图
3.7 小结



3.4 数据查询

- DML语言

  DBMS给上层提供的DML语言主要包括表的增、删、查，其中查询语句是核心DML语句。

- DCL语言

  DBMS给上层程序员提供了DCL语言，主要用于控制程序的执行。主要包括： 事务控制语言、嵌入式SQL语言

[例] 查询选修2号课程且成绩在90分以上的所有学生

```
SELECT  Student.Sno, Sname
FROM    Student, SC
WHERE  Student.Sno = SC.Sno AND
    SC.Cno= ‘2’ AND SC.Grade > 90;
```

3.4 数据查询

1. SELECT 语句的基本句法：

在关系代数中最常用的式子是下列表达式：

$$\pi_{A1,\ldots,An}(\sigma_F(R_1\times\ldots\times R_m))$$

这里 $R_1,\ldots,R_m$为关系，F是公式，$A_1,\ldots,A_n$为属性。

针对上述表达式，SQL语言使用下列句型来表示：

```
SELECT  A1,…,An
FROM  R1,…,Rm
WHERE   F
```

此句型是从关系代数表达式演变过来的，但WHERE子句中的条件表达式F 要比关系代数中公式更灵活。

3.4 数据查询

2. SELECT 语句的完整句法:

```
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> ] …
  [ WHERE <条件表达式> ]
  [ GROUP BY <列名1> [ HAVING <条件表达式> ] ]
  [ ORDER BY <列名2> [ ASC|DESC ] ];
```

注:

- WHERE子句称为**行条件子句**;
- GROUP子句称为**分组子句**;
- HAVING子句称为**组条件子句**;
- ORDER子句称为**排序子句**。

3.4 数据查询

3. SELECT 语句的一般执行过程

- 读取FROM子句中基本表及视图的数据，并执行笛卡尔积操作；
- 选取其中满足WHERE子句中条件表达式的元组；
- 按GROUP BY子句中指定列的值分组，同时提取满足HAVING子句中组条件表达式的那些组；
- 按ORDER BY子句对输出的目标表进行排序，按附加说明ASC升序排列，或按DESC降序排列。

查询语句的语法虽然看上去简单易用，实际上却是SQL语言中最难掌握的语句。主要难点在于查询条件的正确表达。

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

3.4.1 单表查询

单表查询 —— 只涉及到一张表的查询。

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

1. 选择表中的若干列

[例] 查询全体学生的学号与姓名。

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

[例] 查询全部学生的所有信息。

```
SELECT * FROM STUDENT;
```

1. 选择表中的若干列

- 问题：若希望的查询结果表中属性无法直接用SELECT得出，但可以通过运算得出，如何处理这种派生属性?
- [例] 列分别为算术表达式、字符串常量、函数的示例，并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;
```

输出结果:

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

3.4.1 单表查询

- 2. 选择表中的若干元组
- 问题：
  对一个关系SELECT后，结果关系中可能出现重复元组，实现中，为提高效率，SELECT并不消除重复元组。

（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] 查询选修了课程的学生学号。

```
SELECT Sno  FROM SC;
```

等价于：

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

[例] 查询全部被选课程的课程号（去除重复值）。

```
SELECT  DISTINCT  cno  FROM  SC;
```

2. 选择表中的若干元组

（2）查询满足条件的元组

- **如何从表中选择指定元组?**

对应于关系代数运算σ_P，SQL提供where子句解决表元组的选择。对一张表实施where相当于做选择。

- **格式：WHERE <条件表达式>**

<条件表达式>是包含属性名的逻辑表达式P，通过P对元组进行筛选。

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

2. 选择表中的若干元组

（2）查询满足条件的元组

[例] 查询考试成绩有不及格的学生的学号。

```
SELECT DISTINCT Sno
  FROM  SC
  WHERE Grade<60;  //比较大小
```

[例] 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;
//确定范围：闭区间
```

确定集合

- SQL中提供了元素与集合之间的比较运算符
- 谓词：x IN <值表>，x NOT IN <值表>

  其中 <值表> 是一个集合，从关系代数的角度看， 它是一个代数式，从SQL角度看，它是一个SELECT语句

  [例] 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' );
```

  [例] 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' );
```

字符匹配

- **为什么提供字符匹配运算符?**

  关系数据库支持对集合的运算，实际应用中，往往需要从集合中找出类似于某个条件的元组，即模糊查询。SQL的字符匹配运算符为解决这类问题而提出。

- **谓词:**

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

- **通配符:**

  SQL规定符号百分号%及下划线_具有其他含义

  百分号%　　代表任意长度的字符串

  下划线_　　代表任意一个字符

  <匹配串>　为可以含有通配符的字符串

  ESCAPE　　是将百分号% 或下划线_转回其本意

字符匹配示例

[例] 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] 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_ _';
```

[例] 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] 查询以“DB_”开头，且倒数第3个字符为 i的课程的详细情况。

```
SELECT * FROM Course
    WHERE Cname LIKE 'DB\_%i_ _' ESCAPE '\';
   // ESCAPE '\' 表示 “ \ ” 为换码字符
```

2. 选择表中的若干元组

（2）查询满足条件的元组

[例] 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;
```

Notice: “IS” 不能用 “=” 代替

- 多重条件查询：
- 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：IN，BETWEEN…AND

3. ORDER BY子句

ORDER BY 子句：

- 可以按一个或多个属性列排序。如：按(系别，年龄)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：反之。

[例] 列出选修了课程号为‘C6’的所有学生的学号和成绩，并按分数的降序排列。

```
SELECT  sno, grade
FROM    SC
WHERE  cno='C6'
ORDER  BY  grade  DESC;
```

4. 聚集函数

- SQL聚集函数:
  - 计数
    COUNT（[DISTINCT|ALL] *）：统计元组个数
    COUNT（[DISTINCT|ALL] <列名>）：统计一列中值的个数
  - 计算总和
    SUM（[DISTINCT|ALL] <列名>）
  - 计算平均值
    AVG（[DISTINCT|ALL] <列名>）
  - 最大最小值
    MAX（[DISTINCT|ALL] <列名>）
    MIN（[DISTINCT|ALL] <列名>）

集函数只能用于Select子句和Having子句中。

4. 聚集函数

[例] 查询学生总人数。

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

[例] 查询选修了课程的学生人数。

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

[例] 查询选修3号课程学生的最高分、最低分和平均成绩。

```
SELECT MAX(grade), MIN(grade), AVG(grade)
FROM SC
WHERE cno= '3';
```

// 当集函数遇到NULL时，除COUNT(*),都跳过空值而只处理非空值

5. GROUP BY子句

分组是按某（些）列的值对查询的结果进行分类。

- **格式：** GROUP BY A_1，A_2，…,A_n

  其中：A_i为属性名

- 按GROUP子句中指定的列的值进行分组，值相等的为一组。
- HAVING子句可以对分组后的结果作进一步的筛选。
- 当查询语句中有GROUP子句时，集函数作用的对象是一个分组的结果，而不是整个查询的结果。

[例] 求各个课程号及相应的选课人数。

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

5. GROUP BY子句

- 注：分组后，一些详细信息可能损失，不能出现在SELECT结果中。
- 例如，下面的查询

```
SELECT Sno, COUNT(*),Grade
    FROM SC
    WHERE Cno='2'
    GROUP BY Grade;
```

将出错，想一想，为什么?

一般来说，分组查询的SELECT目标列中只允许出现聚集函数和GROUP BY子句中出现过的列。

5. GROUP BY子句

- **问题**： 有时，对于一个分组以后的**结果集合**希望使用限定条件选择部分分组，则可以使用Having 子句。
- **格式**： Having P; P是谓词

[例] 查询选修了三门以上课程的学生学号。

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

注：WHERE子句与HAVING子句的区别: **作用对象不同**。

- WHERE子句作用于基本表或视图，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。

3.4.2 连接查询

- 连接查询：同时涉及多个表的查询
- 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，但名字不必是相同的

3.4.2 连接查询

等值连接：连接运算符为=

1. 等值连接与非等值连接查询

[例] 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;
```

查询结果：

Student.Sno	Sn						
200215121	李						
200215121	李						
200215121	李						
200215122	刘晨	女	19	CS	200215122	2	90
200215122	刘晨	女	19	CS	200215122	3	80

改为自然连接：

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

2. 连接查询

[例] 求“数据结构”课程成绩大于85分的学生姓名和成绩，结果按成绩降序排列。

```
SELECT    STUDENT.sname，grade
FROM      STUDENT，SC，COURSE
WHERE     STUDENT.sno = SC.sno  AND
          SC.cno = COURSE.cno   AND
          COURSE.cname = '数据结构'  AND
          SC.grade > 85
ORDER BY  grade DESC;
```

注：当不同的表中有同名属性时，属性名前要用表名限定。

连接操作的执行过程

- 嵌套循环法(NESTED-LOOP)
  - 采用双循环合并表
  - 效率低
- 步骤：
  - 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

- 排序合并法(SORT-MERGE)　常用于=连接
  - 先排序表
  - 再使用归并算法，让两个表交叉前进
- 步骤：
  - 首先按连接属性对表1和表2排序；
  - 对表1的第一个元组，从头开始扫描表2，顺序查找满足连接条件的元组，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。当遇到表2中第一条大于表1连接字段值的元组时，对表2的查询不再继续。
  - 找到表1的第二条元组，然后从刚才的中断点处继续顺序扫描表2，查找满足连接条件的元组，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。直接遇到表2中大于表1连接字段值的元组时，对表2的查询不再继续。
  - 重复上述操作，直到表1或表2中的全部元组都处理完毕为止。

连接操作的执行过程

- 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。

- 哈希法(HASH-JOIN)
  - 按照链接属性值，开辟多个桶
  - 分别扫描每个表，将元组按照属性值装入对应桶
  - 再将同一个同种元组连接

2.自身连接

一个表与其自己进行连接，称为表的**自身连接**。

[例] 求同时选修了2号课程和3号课程的学生学号。

```
SELECT   a.sno
FROM   CS a, CS b
WHERE   a.sno = b.sno  AND
  a.cno =  '2'   AND b.cno =  '3' ;
```

[例] 找出学分比其先修课的学分高的课程。

```
SELECT C1.name，C2.name
    FROM   Course  C1, Course C2
    WHERE  C1.Cpno=C2.Cno AND
                C1.credit>C2.credit
```

自身连接 （续）

[例] 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;
```

Cno	Cname	Cpno	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

查询结果：

Cno	Pcno
1	7
3	5
5	6

连接

SQL中表的连接有两种表示方法：

- **方法1：在FROM子句中指明进行连接的表名，在WHERE子句中指明连接的列名及其连接条件，如前面的例子。**
- **方法2：利用关键字JOIN进行连接。具体分为以下几种：**
  - **INNER JOIN ： 返回符合连接条件的记录；**
  - **LEFT [OUTER] JOIN：返回符合连接条件的数据行以及左边表中不符合条件的数据行，此时右边数据行以NULL来显示，称为左连接；**
  - **RIGHT [OUTER] JOIN：返回符合连接条件的数据行以及右边表中不符合条件的数据行，此时左边数据行以NULL来显示，称为右连接；**
  - **FULL [OUTER] JOIN：返回符合连接条件的数据行以及左边表和右边表中不符合条件的数据行，此时缺乏数据的数据行会以NULL来显示；**
  - **CROSS JOIN：返回笛卡儿积。**

通过关键词ON与JOIN相对应，指明连接条件。

3. 外连接

上述连接方法2中，LEFT JOIN、RIGHT JOIN与 FULL JOIN统称为外连接。可用来显示不满足连接条件的元组。

某些查询要求只能用外连接来表达。

[例] 查询所有学生的学号、姓名、所选课程的课程号以及这门课的成绩（没有选修任何课程的同学信息也要求被查询出来，相应的选课信息显示为空）。

```
SELECT STUDENT.sno, sname, cno, grade
FROM
STUDENT  LEFT OUTER JOIN  SC
ON STUDENT.SNO=SC.SNO;
```