数据库原理与应用

西南交通大学电气工程学院

第三章 关系数据库标准语言SQL

单选题 1分

设置

SQL语言具有两种使用方式，分别为交互式SQL和（　　）。

A 提示式SQL

B 多用户SQL

C 嵌入式SQL

D 解释式SQL

提交

3.3 数据定义

例：建立3.2节中的表Course，它由课程号Cno，课程名称Cname，先行课Cpno，学分Ccredit组成，其中Cno为主码。

```
Create Table Course (
Cno char(7) Primary Key ,
Cname char(40),
Cpno char(7),
Ccredit smallint ,
Foreign Key (Cpno) References Course(Cno))
```

主码和外码数据类型及长度必须一致

列级主码约束

Cpno是外码

外码约束

例：建立3.2节中的表SC,由学号Sno,课号Cno和成绩Grade属性组成，其中学号和课号为联合主码。

```
Create Table SC (
Sno char(8) ,
Cno char(7) ,
Grade smallint ,
Primary Key (Sno,Cno),
/*主码由两个属性构成，必须用表级完整性定义
Foreign Key (Sno) References Student(Sno),
Foreign Key (Cno) References Course(Cno))
```

表级主码约束

与Student的Sno及Course的Cno建立外码约束

2．修改基表

```
Alter  Table  <表名>
        [Add  <新列名><数据类型>[完整性约束]]
        [Drop <完整性约束名>]
        [Alter Column <列名><数据类型>];
```

(1) 增加列

例：向Student表中增加phone列。

```
Alter Table Student Add Phone char(8)
```

(2) 修改列

例：将Student表中的Phone列改为整形。

```
Alter Table Student Alter Column Phone int;
```

3. 删除基表

将整个表结构彻底删除。表中的数据也将被删除。

Drop Table table_name [Restrict | Cascade]

- **Restrict：删除表是有限制的**
  - **欲删除的基本表不能被其他表的约束所引用**
  - **如果存在依赖该表的对象，则此表不能被删除。**
- **Cascade：删除该表没有限制**
  - **在删除基本表的同时，相关的依赖对象一起删除**

例如： Drop Table Student Cascade

注意：MS SQL Server 不支持Drop Table使用Restrict和 Cascade

3.3.3 索引的建立与删除

- **索引**是为**加快查询速度**而创建的一种结构。
  - 针对基表建立，表中一般存放**关键字和指向记录的指针**。

索引提供在该属性(组)上快速查找具有某个特定值的元组的方法

1、创建索引

```
CREATE [UNIQUE ][CLUSTER]INDEX <索引名>
On <表名>(<列名>[<次序>] [,<列名>[<次序>]]…);
```

- **CLUSTER**： 聚簇索引
- **UNIQUE**： 唯一性索引，每个索引值只对应唯一的记录
- **次序**：索引中记录排列的顺序
  - ASC(升序)和DESC(降序)。**默认为ASC(升序)。**

3.3.3 索引的建立与删除

□ 聚簇索引

- 指索引中邻近的记录在主文件中也是临近存储的。
  - 每一个表只能有**一个**聚簇索引。

□ 非聚簇索引

- 指索引中邻近的记录在主文件中不一定是邻近存储的。
  - 一个表可以创建**多个**非聚簇索引。

例1：在学生表格的学号字段上建立聚簇索引。

```
Create Cluster Index Sno_Index On Student(Sno);
```

注意： SQL Sever中使用的聚簇关键词为**Clustered**

3.3.3 索引的建立与删除

注意：SQL Sever在基表上设置主码时，系统自动为主码列建立聚簇索引。

例2：在SC表上按学号升序和课号降序建唯一索引。

```
Create UNIQUE Index SCno_Index On SC(Sno ,Cno DESC);
```

2. 删除索引

```
DROP INDEX <索引名>;
```

例3：删除例1建立的索引Sno_Index:

```
DROP INDEX Sno_Index;
```

数据查询是数据库应用的核心功能。

查询语句与关系代数:

SELECT A1,A2,A3 FROM TABLE1,TABLE2 WHERE F

$$\Pi_{A1,A2,A3}(\sigma_F(TABLE1 \times TABLE2))$$

- 对 From 子句中的各关系，作笛卡儿积（×）
- 对 Where 子句中的逻辑表达式F进行选择（σ）运算
- 根据 Select 子句的属性列表，对结果作投影（π）操作
- 查询操作的对象和结果都是关系

3.4 数据查询

- 数据查询
  - 单表查询
    - 查询若干列
    - 查询若干记录
    - 对查询结果进行排序
    - 使用集函数
    - 对查询结果分组
  - 连接查询
  - 嵌套查询
  - 集合查询（自学）

3.4.1 单表查询

Student(Sno，Sname，Ssex，Sage，Sdept)

1.选择表中的若干列 (简单查询)

1) 查询指定列

例1 查询全体学生的学号与姓名。

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

2) 查询全部列

例2 查询全体学生的所有信息。

- 列出全部列名，顺序可根据需要指定。

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

- 用符号 *，各列顺序为表中原来的顺序。

```
SELECT  *  FROM Student;
```

Student(Sno，Sname，Ssex，Sage，Sdept)

3) 查询经过计算的值

例3 查询全体学生的姓名及其出生年份 。

```
SELECT Sname, year(getdate())-Sage  FROM Student;
```

- 可以指定别名来改变查询结果中的列标题

```
SELECT Sname  姓名, year(getdate())-Sage 出生年份 FROM Student
```

2.选择表中的若干元组（带有条件）

1) 消除取值重复的行

例4 查询选修了课程的学生的学号。

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

2) 查询满足条件的元组（where子句）

where子句常用的查询条件

查询条件	谓词
比较	=，>，<，>=，<=，!=，<>，!>，!<，NOT+上述运算符
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

3.4.1 单表查询

① 比较大小

例5 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

例6 查询年龄在20至23岁之间的学生的姓名。

不同DBMS对BETWEEN...AND的处理方式不同

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

③确定集合

例7 查询信息系（IS）和数学系（MA）的学生的姓名。

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

④字符匹配

[NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']

- <匹配串>可以是固定字符串；也可以含有通配符
  - %(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。

例8 查询学号为201215121 的学生的详细情况。

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

当匹配串为固定字符串时，可以用 = 运算符取代 LIKE 谓词

例9 查询所有姓刘的学生的情况。

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

例10 查询姓“欧阳”且全名为三个汉字的学生情况 。

```
SELECT * FROM Student  WHERE Sname LIKE '欧阳_'
```

不同DBMS中一个汉字可能需要两个_或一个_

3.4.1 单表查询

查询字符串本身含有%或_时，使用ESCAPE ‘<换码字符>’将通配符转义为普通字符。

例11 查询DB_Design课程的情况。

```
SELECT *  FROM Course
WHERE Cname LIKE 'DB\_Design'  ESCAPE  '\' ;
```

表示 \ 为换码字符

不再具有通配符的含义，转义为普通的'_'字符。

例12 查询以”DB_”开头，且倒数第三个字符为 *i* 的课程的情况

```
SELECT *  FROM Course
WHERE Cname LIKE 'DB\_%i_ _'  ESCAPE  '\' ;
```

普通字符

通配符

课堂练习

关系R(书号,书名)，如果要检索书名中至少包含4个字母，且第3个字母为M的图书，则SQL查询语句中WHERE条件表达式应写成WHERE 书名 LIKE（　　）。

A.'%_ _M%'

B.'_ _M%'

C.' _ _M_%'

D.'_%M_ _'

⑤ 涉及空值的查询（IS NULL 和 IS NOT NULL）

例13 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;
```

注意： 'IS NULL'不能用'= NULL' 代替。

例14 查询所有有成绩记录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（AND和OR ）

例15 查询成绩为99分或100分的学生的学号和课程号 。

```
SELECT Sno, Cno  FROM SC
WHERE Grade =99 OR Grade=100;
```

3. 对查询结果排序

ORDER BY，升序:ASC(默认)，降序:DESC

例16 查询全体学生情况，查询结果按所在系升序排列，对同一系中的学生按年龄降序排列 。

SELECT * FROM Student ORDER BY Sdept, Sage DESC;

4. 使用集函数

COUNT([DISTINCT|ALL] [*|<列名>]) 统计元组或列值的个数

SUM([DISTINCT|ALL] <列名>) 计算一列值的总和

AVG([DISTINCT|ALL] <列名>) 计算一列值的平均值

MAX([DISTINCT|ALL] <列名>) 求一列值中的最大值

MIN([DISTINCT|ALL] <列名>) 求一列值中的最小值

例17 查询总共开设了多少门课?

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

例18 统计计算信息系学生的平均年龄 。

```
SELECT AVG(Sage) FROM Student WHERE Sdept='IS'
```

例19 查询课程的总学分数 。

```
SELECT SUM(Ccredit) FROM Course
```

例20 统计选修课程的学生总人数 。

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

如何统计每门课程的选课人数

5.对查询结果进行分组（GROUP BY）

分组方法：按指定的一列或多列值分组，值相等的为一组

- **未对查询结果分组，集函数作用于整个查询结果，即整个查询结果只有一个函数值。**
- **对查询结果分组后，集函数分别作用于每一个组，即每一组都有一个函数值。**

例21 统计每门课程的选课人数。

```
SELECT Cno, COUNT(*) FROM SC  GROUP BY Cno
```

- 使用分组子句，SELECT后的列名列表中只能出现分组属性和集函数
- 如果分组后还要求按一定的条件对这些组进行筛选，最终只输出满足指定条件的组，则可以使用HAVING短语指定筛选条件。

例22 统计选课人数小于20人的课程号及选课人数 。

```
SELECT Cno, COUNT(*) FROM SC GROUP BY Cno
HAVING COUNT(*)<20
```