数据库系统概论-03-关系数据库标准语言 SQL

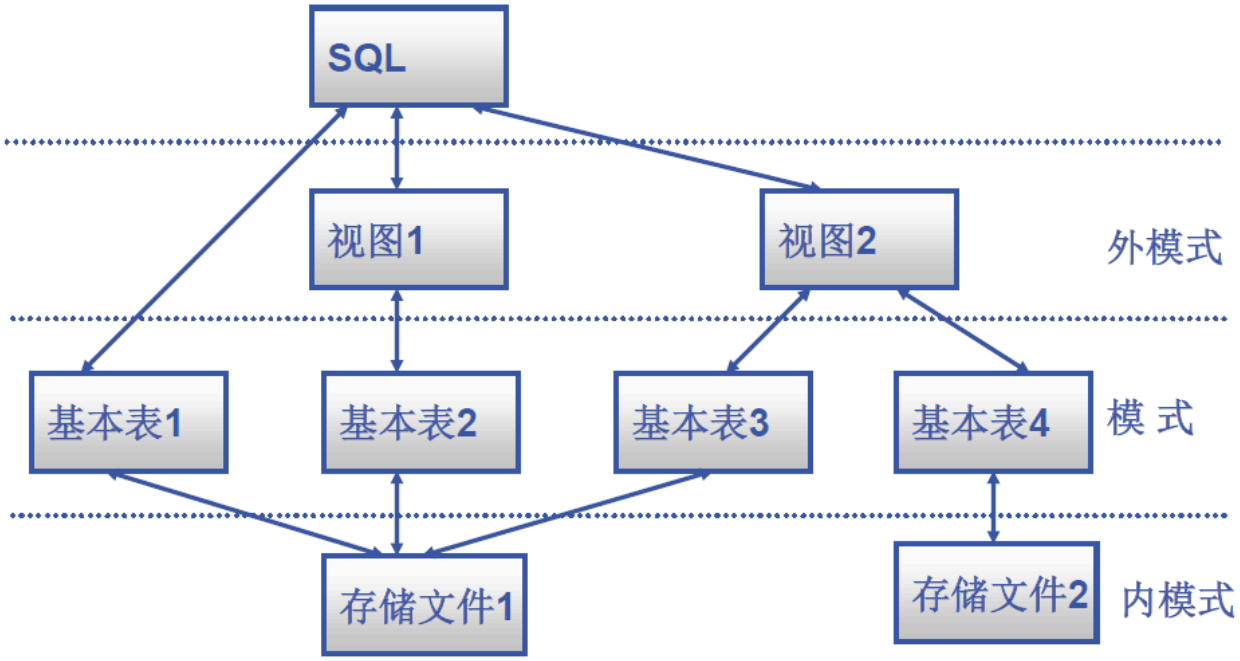
2022-06-16 | 2025-11-26 | 南京大学软件学院本科课程， 2022Spring-数据库系统概论 | 178

16k | 15 分钟

SQL 结构化查询语言，是关系数据库的标准语言。集数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体。

仅有 9 个动词： `CREATE`, `DROP`, `ALTER`, `SELECT`, `INSERT`, `UPDATE`, `DELETE`, `GRANT`, `REVOKE`

1. SQL 与关系数据库



基本表是本身独立存在的表。一个关系对应一个基本表，一个或多个基本表对应一个存储文件，一个表可以带若干索引。

视图时从一个或几个基本表导出的表。数据库中只存放视图的定义而不存放视对应的数据。用户可以在视图上再定义视图。

2. 层次化的数据库对象命名机制

1. 一个关系数据库管理系统的实例（Instance）中可以建立多个数据库（database）
2. 一个数据库中可以建立多个模式（schema）
3. 一个模式下通常包括多个表（table）、视图（view）和索引（index）等数据库对象

图 3.3 数据库对象命名机制的层次结构

3. 数据定义

操作对象	操作方式		
	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	ALTER INDEX

3.1 定义模式

```
CREATE SCHEMA <schemaName> AUTHORIZATION <username> [<table_defination_clause> | <view_definat

# [例 3.1] 为用户 WANG 定义一个学生-课程模式 S-T
CREATE SCHEMA "S_T" AUTHORIZATION WANG;

# [例 3.2] 该语句没有指定<模式名>，<模式名>隐含为<用户名>
CREATE SCHEMA AUTHORIZATION WANG; # 未指定模式名，默认为用户名

# [例 3.3]为用户 ZHANG 创建了一个模式 TEST，并且在其中定义一个表 TAB1
CREATE SCHEMA TEST AUTHORIZATION ZHANG;
CREATE TABLE TAB1 (
    COL1 SMALLINT,
```

```
    COL2 INT,
    COL3 CHAR(20),
    COL4 NUMERIC(10,3),
    COL5 DECIMAL(5,2)
);
```

3.2 删除模式

```
DROP SCHEMA <schemaName> <CASCADE | RESTRICT>

DROP SCHEMA WANG RESTRICT; # 只能删除空表，否则拒绝执行

# [例 3.4] 删除模式 ZHANG 同时该模式中定义的表 TAB1 也被删除
DROP SCHEMA ZHANG CASCADE;
```

```
CREATE TABLE <tableName> (
    <colName> <datatype> [<col_integrity_constraints>]
    [,<colName> <datatype> [<col_integrity_constraints>]]
    ...
    [,<table_integrity_constraints>]
);

# [例 3.5]  建立“学生”表 Student。学号是主码，姓名取值唯一。
CREATE TABLE Student (
    Sno CHAR(9) PRIMARY KEY, # 列级完整性约束条件，Sno 是主码，Sname 取唯一值
    Sname CHAR(20) UNIQUE,
    Ssex CHAR(2),
    Sage SMALLINT,
    Sdept CHAR(20)
);

#  [例 3.6] 建立一个“课程”表 Course
CREATE TABLE Course (
    Cno CHAR(4) PRIMARY KEY,
    Cname CHAR(40) NOT NULL,
    Cpno CHAR(4),
    Ccredit SMALLINT,
    FOREIGN KEY (Cpno) REFERENCES Course(Cno)
);

```

```
# [例 3.7]  建立一个学生选课表 SC
CREATE TABLE SC (
    Sno CHAR(9),
    Cno CHAR(4),
    Grade SMALLINT,
    PRIMARY KEY (Sno,Cno), # 主码由两个属性构成，必须作为表级完整性进行定义
    FOREIGN KEY (Sno) REFERENCES Student(Sno), # 表级完整性约束条件，Sno 是外码，被参照表是 Studen
    FOREIGN KEY (Cno)REFERENCES Course(Cno) # 表级完整性约束条件， Cno 是外码，被参照表是 Course
);
```

数据类型	含义
CHAR(*n*),CHARACTER(*n*)	长度为n的定长字符串
VARCHAR(*n*), CHARACTERVARYING(*n*)	最大长度为n的变长字符串
CLOB	字符串大对象
BLOB	二进制大对象
INT，INTEGER	长整数（4字节）
SMALLINT	短整数（2字节）
BIGINT	大整数（8字节）
NUMERIC(*p*，*d*)	定点数，由*p*位数字（不包括符号、小数点）组成，小数后面有*d*位数字
DECIMAL(*p*, *d*), DEC(*p*, *d*)	同NUMERIC
REAL	取决于机器精度的单精度浮点数
DOUBLE PRECISION	取决于机器精度的双精度浮点数
FLOAT(*n*)	可选精度的浮点数，精度至少为*n*位数字
BOOLEAN	逻辑布尔量
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS
TIMESTAMP	时间戳类型
INTERVAL	时间间隔类型

5. 修改基本表

```
ALTER TABLE <tableName>
    [ADD [COLUMN] <colName> <datatype> [<col_integrity_constraints>]] # 添加空值列
    [DROP [COLUMN] <colName> [CASCADE | RESTRICT]] # 删除列
    [ADD <table_integrity_constraints>] # 添加约束
    [DROP CONSTRAINT <integrity_constraints> [RESTRICT | CASCADE]] # 删除约束
    [ALTER COLUMN <colName> <datatype>]; # 修改列名或数据类型

# [例 3.8] 向 Student 表增加“入学时间”列，其数据类型为日期型
ALTER TABLE Student ADD S_entrance DATE;

# [例 3.9] 将年龄的数据类型由字符型（假设原来的数据类型是字符型）改为整数。
ALTER TABLE Student ALTER COLUMN Sage INT;
```

```
13
14  # [例 3.10] 增加课程名称必须取唯一值的约束条件。
15  ALTER TABLE Course ADD UNIQUE(Cname);
```

6. 删除基本表

```
1  DROP TABLE <tableName> [RESTRICT | CASCADE];
2
3  #  [例 3.11]  删除 Student 表
4  DROP TABLE Student RESTRICT; # 有其他对象依赖该表，不能删除
5
6  #  [例 3.12] 若表上建有视图，选择 RESTRICT 时表不能删除;选择 CASCADE 时可以删除表，视图也自动删除。
7  DROP TABLE Student CASCADE; # 索引、视图、触发器等一并删除
```

7. 索引

不在考试范围。

8. 数据字典

数据字典是关系数据库管理系统内部的一组系统表，它记录了数据库中所有定义信息：关系模式定义 ，视图定义，索引定义，完整性约束定义，各类用户对数据库的操作权限，统计信息等。

9. 数据查询单表查询

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。

```
1  SELECT [ALL | DISTINCT] <目标列表达式> [,<目标列表达式>] ...
2       FROM <tableName> [,<tableName>] ... | (<select_caluse>)
3       [AS] <alias>
4       [WHERE <conditional_expression>]
5       [GROUP BY <colName> [HAVING <conditional_expression>]]
6       [ORDER BY <colName>] [ASC | DESC]];
7
8  # [例 3.16]  查询全体学生的学号与姓名。
9  SELECT Sno, Sname FROM Student;
10
```

```
# [例 3.17]  查询全体学生的姓名、学号、所在系。
SELECT Sname, Sno, Sdept FROM Student;

# [例 3.18]  查询全体学生的详细记录
SELECT Sno, Sname, Ssex, Sage, Sdept FROM Student;
SELECT * FROM Student; # 与上一行等价

# [例 3.19]  查全体学生的姓名及其出生年份。
SELECT Sname, 2014-Sage FROM Student;

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

# [例 3.21]  查询选修了课程的学生学号。
SELECT Sno FROM SC;
SELECT ALL Sno FROM SC; # 与上一行等价
SELECT DISTINCT Sno FROM SC;

# [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 Sn FROM SC
WHERE Grade < 60;

# [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;

# [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');
```

```
57
58  # [例 3.29] 查询学号为 201215121 的学生的详细情况。
59  SELECT * FROM Student
60  WHERE Sno LIKE '201215121';
61  SELECT * FROM Student
62  WHERE Sno = '201215121'; # 与上一行等价
63
64  # [例 3.30] 查询所有姓刘学生的姓名、学号和性别。
65  SELECT Sname, Sno, Ssex FROM Student
66  WHERE Sname LIKE '刘 %'; # % 匹配任意长度字符串
67
68  # [例 3.31] 查询姓“欧阳”且全名为三个汉字的学生的姓名。
69  SELECT Sname FROM Student
70  WHERE Sname LIKE '欧阳_'; # _ 匹配任意单个字符
71
72  # [例 3.32]  查询名字中第 2 个字为“阳”字的学生的姓名和学号。
73  SELECT Sname，Sno FROM Student
74  WHERE Sname LIKE '__阳 %';
75
76  # [例 3.33]  查询所有不姓刘的学生姓名、学号和性别。
77  SELECT Sname, Sno, Ssex FROM Student
78  WHERE Sname NOT LIKE '刘 %';
79
80  #  [例 3.34]  查询 DB_Design 课程的课程号和学分。
81  SELECT Cno，Ccredit FROM Course
82  WHERE Cname LIKE 'DB\_Design' ESCAPE '\ '; # ESCAPE '\' 表示 '\' 为换码字符
83
84  # [例 3.35]  查询以"DB_"开头，且倒数第 3 个字符为 i 的课程的详细情况。
85  SELECT * FROM Course
86  WHERE Cname LIKE 'DB\_%i_ _' ESCAPE '\ ';
87
88  # [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
89  SELECT Sno，Cno FROM SC
90  WHERE Grade IS NULL; # IS 不可换成 =
91
92  # [例 3.38]  查询计算机系年龄在 20 岁以下的学生姓名。
93  SELECT Sname FROM Student
94  WHERE Sdept = 'CS' AND Sage < 20;
95
96  # [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。
97  SELECT Sname, Ssex FROM Student
98  WHERE Sdept IN ('CS', 'MA', 'IS');
99  SELECT Sname, Ssex FROM Student
100 WHERE Sdept = ' CS' OR Sdept = 'MA' OR Sdept= 'IS'; # 与上一行等价
101
102 # [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。
```

```
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;

# [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';

# [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;

# [例 3.46]  求各个课程号及相应的选课人数。
/* 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。*/
SELECT Cno, COUNT(Sno) FROM SC
GROUP BY Cno;
SELECT Cno, COUNT(*) FROM SC
GROUP BY Cno;

# [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;

# [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
SELECT Sno, AVG(Grade) FROM SC GROUP BY Sno
HAVING AVG(Grade) >= 90;
```

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

9.1 链接查询

```
# [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;

# [例 3.50] 对上例使用自然连接
SELECT Student.Sno,Sname,Ssex,Sage,Sd ept,Cno,Grade FROM Student,SC
WHERE Student.Sno = SC.Sno;

# [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;

# [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;

# [例 3.53] 改写 [例 3.49]
# 左外连接：列出左表中所有的元组
SELECT Student.Sno, Sname, Ssex, Sage, Sdept, Cno, Grade
FROM Student LEFT OUT JOIN SC ON (Student.Sno = SC.Sno);

# [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;
```

9.2 嵌套查询

上层的查询块称为外层查询或父查询，下层查询块称为内层查询或子查询。

SQL 语言允许多层嵌套查询，即一个子查询中还可以嵌套其他子查询。如果子查询的查询条件不依赖于父查询，称为**不相关子查询**；否则称为**相关子查询**。

子查询的限制： **不能使用 `ORDER BY` 子句** 。有些嵌套查询可以用连接运算替代，**谨慎使用嵌套查询。**

```
# [例 3.55] 查询与“刘晨”在同一个系学习的学生。
# 不相关子查询
SELECT Sno, Sname, Sdept
FROM Student
WHERE Sdept IN (
    SELECT Sdept
    FROM Student
    WHERE Sname = '刘晨'
);

# 自身链接
SELECT S1.Sno, S1.Sname, S1.Sdept
FROM Student S1,
     Student S2
WHERE S1.Sdept = S2.Sdept
  AND S2.Sname = '刘晨';

# [例 3.56]查询选修了课程名为“信息系统”的学生学号和姓名
SELECT Sno, Sname
WHERE Sno IN (
    SELECT Sno
    FROM SC
    WHERE Cno IN (
        SELECT Cno
        FROM Course
        WHERE Cname = '信息系统'
    )
);

# 用连接查询实现[例 3.56]
SELECT Sno, Sname
FROM Student, SC, Course
WHERE Student.Sno = SC.Sno
  AND SC.Cno = Course.Cno
  AND Course.Cname = '信息系统';

# 在[例 3.55]中，由于一个学生只可能在一个系学习，则可以用 = 代替 IN:
SELECT Sno, Sname, Sdept
FROM Student
```

```
WHERE Sdept = (
    SELECT Sdept
    FROM Student
    WHERE Sname = '刘晨'
);

# [例 3.57] 找出每个学生超过他选修课程均成绩的课程号。
SELECT Sno, Cno
FROM SC x
WHERE Grade >= (
    SELECT AVG(Grade)
    FROM SC y
    WHERE y.Sno = x.Sno
);

# [例 3.58] 查询非计算机科学系中比计算机科学系任意一个学生年龄小的学生姓名和年龄
# 嵌套查询
SELECT Sname, Sage
FROM Student
WHERE Sdept <> 'CS'
AND Sage < ANY (
    SELECT Sage
    FROM Student
    WHERE Sdept = 'CS'
);

# 用聚集函数实现[例 3.58]
SELECT Sname, Sage
FROM Student
WHERE Sage < (SELECT MAX(Sage) FROM Student WHERE Sdept = 'CS ')
  AND Sdept <> 'CS';

# [例 3.59] 查询非计算机科学系中比计算机科学系所有学生年龄都小的学生姓名及年龄。
# 方法一：用 ALL 谓词
SELECT Sname, Sage
FROM Student
WHERE Sage < ALL (SELECT Sage FROM Student WHERE Sdept = 'CS')
  AND Sdept <> 'CS';

# 方法二：用聚集函数
SELECT Sname, Sage
FROM Student
WHERE Sage < (SELECT MIN(Sage) FROM Student WHERE Sdept = 'CS')
  AND Sdept <> 'CS';

# [例 3.60]查询所有选修了 1 号课程的学生姓名。
```

```
SELECT Sname
FROM Student
WHERE EXISTS(
              SELECT *
              FROM SC
              WHERE Sno = Student.Sno
                AND Cno = '1'
           );

#  [例 3.61] 查询没有选修 1 号课程的学生姓名。
SELECT Sname
FROM Student
WHERE NOT EXISTS(
         SELECT *
         FROM SC
         WHERE Sno = Student.Sno
           AND Cno = '1'
     );

# [例 3.55]查询与“刘晨”在同一个系学习的学生。
SELECT Sno, Sname, Sdept
FROM Student S1
WHERE EXISTS(
              SELECT *
              FROM Student S2
              WHERE S2.Sdept = S1.Sdept
                AND S2.Sname = '刘晨'
           );

# [例 3.62] 查询选修了全部课程的学生姓名。
# 即，不存在没有修过的课程
SELECT Sname
FROM Student
WHERE NOT EXISTS(
         SELECT *
         FROM Course
         WHERE NOT EXISTS(
                 SELECT * FROM SC WHERE Sno = Student.Sno AND Cno = Course.Cno
             )
     );

# [例 3.62] 改
SELECT Sname
FROM student
WHERE Sno IN (
     SELECT Sno
```

```
    FROM SC
    Group by Sno
    HAVING count(*) = (SELECT count(*) FROM course)
);
SELECT Sname
FROM Student
WHERE (SELECT COUNT(*) FROM course) = (
    SELECT COUNT(*) FROM SC
    GROUP BY Sno
    HAVING SC.Sno = Student.Sno
);

# [例 3.63]查询至少选修了学生 201215122 选修的全部课程的学生号码。
# 不存在这样的课程 y，学生 201215122 选修了 y，而学生 x 没有选。
SELECT DISTINCT Sno
FROM SC SCX
WHERE NOT EXISTS(  # 学生 201215122 选修了而学生 x 没有选的课程
        SELECT *
        FROM SC SCY
        WHERE SCY.Sno = '201215122' AND NOT EXISTS(
                SELECT *  # 学生 x 选修的课程 y
                FROM SC SCZ
                WHERE SCZ.Sno = SCX.Sno
                  AND SCZ.Cno = SCY.Cno
            )
    );
```

9.3 集合查询

```
# [例 3.64] 查询计算机科学系的学生及年龄不大于 19 岁的学生。
SELECT *
FROM Student
WHERE Sdept = 'CS'
UNION
SELECT *
FROM Student
WHERE Sage <= 19;

# [例 3.65] 查询选修了课程 1 或者选修了课程 2 的学生。
SELECT Sno
FROM SC
WHERE Cno = '1'
UNION
SELECT Sno
FROM SC
```

```
WHERE Cno = '2';

# [例 3.66] 查询计算机科学系的学生与年龄不大于 19 岁的学生的交集。
SELECT *
FROM Student
WHERE Sdept = 'CS' INTERSECT
SELECT *
FROM Student
WHERE Sage <= 19;

# [例 3.66] 实际上就是查询计算机科学系中年龄不大于 19 岁的学生。
SELECT *
FROM Student
WHERE Sdept = 'CS'
  AND Sage <= 19;

# [例 3.67]查询既选修了课程 1 又选修了课程 2 的学生。
SELECT Sno
FROM SC
WHERE Cno = '1' INTERSECT
SELECT Sno
FROM SC
WHERE Cno = '2';

# [例 3.67]也可以表示为:
SELECT Sno
FROM SC
WHERE Cno = '1'
  AND Sno IN (
    SELECT Sno
    FROM SC
    WHERE Cno = '2'
);

# [例 3.68]实际上是查询计算机科学系中年龄大于 19 岁的学生
SELECT *
FROM Student
WHERE Sdept = 'CS'
  AND Sage > 19;
```

10. 数据更新

10.1 数据插入

```
INSERT INTO <表名> [(<属性列 1>[, <属性列 2 >...)]
VALUES (<常量 1> [, <常量 2>]... );

# [例 3.69] 将一个新学生元组（学号： 201215128;姓名：陈冬;性别：男; 所在系：IS;年龄：18 岁）插入到 Stu
INSERT INTO Student (Sno, Sname, Ssex, Sdept, Sage)
VALUES ('201215128', '陈冬', '男', 'IS', 18);

# [例 3.70] 将学生张成民的信息插入到 Student 表中。
INSERT INTO Student
VALUES ('201215126', '张成民', '男', 18, 'CS');

# [例 3.71] 插入一条选课记录（'200215128','1'）。
INSERT INTO SC(Sno, Cno)
VALUES ('201215128', '1');

# 或者
INSERT INTO SC
VALUES ('201215128', '1', NULL);

# [例 3.72] 对每一个系，求学生的平均年龄，并把结果存入数据库
CREATE TABLE Dept_age
(
    Sdept CHAR(15) Avg_age SMALLINT
);
INSERT INTO Dept_age(Sdept, Avg_age)
SELECT Sdept.AVG(Sage)
FROM Student
GROUP BY Sdept;
```

10.2 数据修改

```
UPDATE <表名>
SET <列名>=<表达式>[, <列名>=<表达式>].
..[WHERE <条件>];

# [例 3.73] 将学生 201215121 的年龄改为 22 岁
UPDATE Student
SET Sage = 22
WHERE Sno = ' 201215121 ';

# [例 3.74] 将所有学生的年龄增加 1 岁。
UPDATE Student
SET Sage = Sage + 1;
```

```

# [例 3.75] 将计算机科学系全体学生的成绩置零。
UPDATE SC
SET Grade = 0
WHERE Sno IN (SELETE Sno FROM Student WHERE Sdept = 'CS');
```

10.3 数据删除

```
DELETE
FROM <表名> [WHERE <条件>];

# [例 3.76] 删除学号为 201215128 的学生记录。
DELETE
FROM Student
WHERE Sno = '201215128';

# [例 3.77] 删除所有的学生选课记录。
DELETE
FROM SC;

# [例 3.78] 删除计算机科学系所有学生的选课记录。
DELETE
FROM SC
WHERE Sno IN (SELETE Sno FROM Student WHERE Sdept= 'CS');
```

11. SQL 中的空值

判断空值用 `IS NULL` 或 `IS NOT NULL`。

有 `NOT NULL` 限制的 `UNIQUE` 属性不能为空值，码不能为空值。

空值与其他值的算术运算结果为空值，空值与其他值的比较运算结果为 `UNKNOWN`。

x	y	x AND y	x OR y	NOT x
T	T	T	T	F
T	U	U	T	F
T	F	F	T	F
U	T	U	T	U
U	U	U	U	U
U	F	F	U	U
F	T	F	T	T
F	U	F	U	T
F	F	F	F	T

T表示TRUE，F表示FALSE，U表示UNKNOWN

```
# [例 3.83] 选出选修 1 号课程的不及格的学生以及缺考的学生。
SELECT Sno
FROM SC
WHERE Grade < 60
  AND Cno = '1'
UNION
SELECT Sno
FROM SC
WHERE Grade IS NULL
  AND Cno = '1';

# 或者
SELECT Sno
FROM SC
WHERE Cno = '1'
  AND (Grade < 60 OR Grade IS NULL);
```

12. 视图

数据只存放视图的定义，不存放视图对应的数据。

视图的作用：

1. 视图能够简化用户的操作

2. 使用户能够以多种角度看待统一数据
3. 视图对重构数据库提供了一定程度的逻辑独立性
4. 试图能够对机密数据提供安全保护
5. 适当的利用视图可以更清晰地表达查询

12.1 定义视图

从单个基本表导出的、并且只是去掉了基本表的某些行和某些列，但保留了主码，这样的视图称为**行列子集视图。**

```
CREATE VIEW <视图名> [(<列名> [,<列名>]...)]
AS <子查询> [WITH CHECK OPTION];
# WITH CKECK OPTION 表示对视图进行更新操作时要保证更新后的行满足谓词条件（即子查询中的条件）

# [例 3.85]建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有信息系的学生。
CREATE VIEW IS_Student AS
SELECT Sno, Sname, Sage
FROM Student
WHERE Sdept = 'IS'
WITH CHECK OPTION;

# [例 3.86] 建立信息系选修了 1 号课程的学生的视图（包括学号、 姓名、成绩）。
CREATE VIEW IS_S1(Sno, Sname, Grade) AS
SELECT Student.Sno, Sname, Grade
FROM Student,
     SC
WHERE Sdept = 'IS'
  AND Student.Sno = SC.Sno
  AND SC.Cno = '1';

# [例 3.87] 建立信息系选修了 1 号课程且成绩在 90 分以上的学生的视图。
CREATE VIEW IS_S2 AS
SELECT Sno, Sname, Grade
FROM IS_S1
WHERE Grade >= 90;

# [例 3.88] 定义一个反映学生出生年份的视图。
CREATE VIEW BT_S(Sno, Sname, Sbirth) AS
SELECT Sno, Sname, 2014 - Sage
FROM Student;

# [例 3.89] 将学生的学号及平均成绩定义为一个视图
CREATE VIEW S_G(Sno, Gavg) AS
SELECT Sno, AVG(Grade)
```

```
FROM SC
GROUP BY Sno;

# [例 3.90]将 Student 表中所有女生记录定义为一个视图
# 修改基表 Student 的结构后，Student 表与 F_Student 视图的映象关系被破坏，导致该视图不能正确工作。
CREATE VIEW F_Student(F_Sno, name, sex, age, dept) AS
SELECT * #不指定属性列
FROM Student
WHERE Ssex = '女';
```

12.2 删除视图

```
DROP VIEW <视图名>[CASCADE];

# [例 3.91 ] 删除视图 BT_S 和 IS_S1
DROP VIEW BT_S; #成功执行
DROP VIEW IS_S1; #拒绝执行
DROP VIEW IS_S1 CASCADE;
# 成功执行
```

12.3 查询视图

视图消解法：进行有效性检查，从数据字典中取出视图的定义，把定义中的子查询和用户的查询结合起来，转换成等价的对基本表的查询，执行修正后的查询。

```
# [例 3.92] 在信息系学生的视图中找出年龄小于 20 岁的学生。
SELECT Sno, Sage
FROM IS_Student
WHERE Sage < 20;
# 视图消解转换后的查询语句为：
SELECT Sno, Sage
FROM Student
WHERE Sdept = 'IS'
  AND Sage < 20;

# [例 3.93] 查询选修了 1 号课程的信息系学生
SELECT IS_Student.Sno, Sname
FROM IS_Student,
     SC
WHERE IS_Student.Sno = SC.Sno
  AND SC.Cno = '1';

# [例 3.94]在 S_G 视图中查询平均成绩在 90 分以上的学生学号和平均成绩
SELECT *
```

```
FROM S_G
WHERE Gavg >= 90;

# S_G 视图的子查询定义：
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;

# 错误：
SELECT Sno，AVG(Grade)
FROM SC
WHERE AVG(Grade) >= 90
GROUP BY Sno;

# 正确：
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno
HAVING AVG(Grade) >= 90;

# 或者使用派生表
SELECT *
FROM (SELECT Sno, AVG(Grade) FROM SC GROUP BY Sno) AS S_G(Sno, Gavg)
WHERE Gavg >= 90;
```

12.4 更新视图

允许对行列子集视图进行更新，对其他类型视图的更新不同系统有不同限制。

一般地，行列子集视图时可更新的。一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新。

一个不允许更新的视图上定义的视图也不允许更新。

```
# [例 3.95] 将信息系学生视图 IS_Student 中学号“201215122”的学生姓名改为“刘辰”。
UPDATE IS_Student
SET Sname= '刘辰'
WHERE Sno = '201215122';
# 转换后的语句：
UPDATE Student
SET Sname= '刘辰'
WHERE Sno = '201215122'
  AND Sdept = 'IS';

# [例 3.96] 向信息系学生视图 IS_S 中插入一个新的学生记录，其中学号为“201215129”，姓名为“赵新”，年龄为
INSERT INTO IS_Student
```

```
VALUES ('201215129', '赵新', 20);
# 转换为对基本表的更新：
INSERT INTO Student(Sno, Sname, Sage, Sdept)
VALUES ('200215129', '赵新', 20, 'IS');

# [例 3.97]删除信息系学生视图 IS_Student 中学号为“201215129”的记录
DELETE
FROM IS_Student
WHERE Sno = '201215129';
# 转换为对基本表的更新：
DELETE
FROM Student
WHERE Sno = '201215129'
  AND Sdept = 'IS';

# 例 3.89 定义的视图 S_G 为不可更新视图。
# 这个对视图的更新无法转换成对基本表 SC 的更新
UPDATE S_G
SET Gavg=90
WHERE Sno = '201215121';

# 将 SC 中成绩在平均成绩之上的元组定义成一个视图
CREATE VIEW GOOD_SC AS
SELECT Sno, Cno, Grade
FROM SC
WHERE Grade > (SELECT AVG(Grade) FROM SC);
```


原创

数据库系统概论-02-关系数据库 数据库系统概论-04-数据库安全性