SQL 语句实例

(a) 学生 STUDENTS

学号 SNO	姓名 SNAME	年龄 AGE	性别 SEX	籍贯 BPLACE
990027	胡伟	22	男	湖南
990652	张春明	24	男	河北
990668	王翌	22	女	四川
990674	丁晓春	24	男	黑龙江
990676	贺正生	23	男	湖南
990684	刘文革	24	女	辽宁
991091	程会军	23	男	山西

(b) 课程 COURSES

课程号 CNO	课程名 CNAME	学分 CREDIT
C1	数据库	3
C2	数据结构	3
C3	操作系统	4
C4	软件工程	3

(c) 教师 TEACHERS

教师号 TNO	教师名 TNAME	年龄 AGE	职称 PS
1420	周振华	38	副教授
1481	刘建平	30	讲师
1433	王志伟	28	讲师

(d) 选课 ENROLLS

学号 SNO	课程号 CNO	成绩 GRADE
990027	C1	90
990027	C3	95
990027	C4	88
990652	C1	88
990652	C4	83
990668	C3	84
990674	C2	77
990676	C3	90
990684	C3	85
990684	C1	82
991091	C2	93

(e) 任课 TEACHING

课程号 CNO	班级 CLASS	教师号 TNO	学生数 SNUM
C1	E851	1420	30
C2	E851	1420	22
C3	E651	1481	30
C3	E852	1481	28
C4	E851	1433	24
C1	E852	1420	28

表操作

例 1 对于表的教学管理数据库中的表 **STUDENTS** ，可以定义如下：

```
CREATE TABLE STUDENTS
(SNO NUMERIC (6, 0) NOT NULL
SNAME CHAR (8) NOT NULL
AGE NUMERIC(3,0)
SEX CHAR(2)
BPLACE CHAR(20)
PRIMARY KEY(SNO))
```

例 2 对于表的教学管理数据库中的表 **ENROLLS** ，可以定义如下：

```
CREATE TABLE ENROLLS
(SNO NUMERIC(6,0) NOT NULL
CNO CHAR(4) NOT NULL
GRADE INT
PRIMARY KEY(SNO,CNO)
FOREIGN KEY(SNO) REFERENCES STUDENTS(SNO)
FOREIGN KEY(CNO) REFERENCES COURSES(CNO)
CHECK ((GRADE IS NULL) OR (GRADE BETWEEN 0 AND 100)))
```

例 3 根据表的 **STUDENTS** 表，建立一个只包含学号、姓名、年龄的女学生表。

```
CREATE TABLE GIRL
AS SELECT SNO, SNAME, AGE
FROM STUDENTS
WHERE SEX=' 女 ';
```

例 4 删除教师表 **TEACHER** 。

```
DROP TABLE TEACHER
```

例 5 在教师表中增加住址列。

```
ALTER TABLE TEACHERS
ADD (ADDR CHAR(50))
```

例 6 把 **STUDENTS** 表中的 **BPLACE** 列删除，并且把引用 **BPLACE** 列的所有视图和约束也一起删除。

```
ALTER TABLE STUDENTS
DROP BPLACE CASCADE
```

例 7 补充定义 **ENROLLS** 表的主关键字。

```
ALTER TABLE ENROLLS
ADD PRIMARY KEY (SNO,CNO) ;
```

视图操作（虚表）

例 9 建立一个只包括教师号、姓名和年龄的视图 **FACULTY** 。 **(** 在视图定义中不能包含 **ORDER BY** 子句 **)**

```
CREATE VIEW FACULTY
AS SELECT TNO, TNAME, AGE
FROM TEACHERS
```

例 10 从学生表、课程表和选课表中产生一个视图 **GRADE_TABLE** ， 它包括学生姓名、课程名和成绩。

```
CREATE VIEW GRADE_TABLE
AS SELECT SNAME,CNAME,GRADE
FROM STUDENTS,COURSES,ENROLLS
WHERE STUDENTS.SNO ＝ ENROLLS.SNO AND
COURSES.CNO=ENROLLS.CNO
```

例 11 删除视图 **GRADE_TABLE**

```
DROP VIEW GRADE_TABLE RESTRICT
```

索引操作

例 12 在学生表中按学号建立索引。

```
CREATE UNIQUE INDEX ST
ON STUDENTS (SNO,ASC)
```

例 13 删除按学号所建立的索引。

```
DROP INDEX ST
```

数据库模式操作

例 14 创建一个简易教学数据库的数据库模式 TEACHING_DB ，属主为 ZHANG 。

```
CREATE SCHEMA TEACHING_DB AUTHRIZATION ZHANG
```

例 15 删除简易教学数据库模式 TEACHING_DB 。（（ 1 ）选用 CASCADE ，即当删除数据库模式时，则本数据库模式和其下属的基本表、视图、索引等全部被删除。（ 2 ）选用 RESTRICT ，即本数据库模式下属的基本表、视图、索引等事先已清除，才能删除本数据库模式，否则拒绝删除。）

```
DROP SCHEMA TEACHING_DB CASCADE
```

单表操作

例 16 找出 3 个学分的课程号和课程名。

```
SELECT CNO, CNAME
FROM COURSES
WHERE CREDIT ＝ 3
```

例 17 查询年龄大于 22 岁的学生情况。

```
SELECT *
FROM STUDENTS
WHERE AGE ＞ 22
```

例 18 找出籍贯为河北的男生的姓名和年龄。

```
SELECT SNAME, AGE
FROM STUDENTS
WHERE BPLACE ＝ ' 河北 ' AND SEX ＝ ' 男 '
```

例 19 找出年龄在 20 ～ 23 岁之间的学生的学号、姓名和年龄，并按年龄升序排序。(ASC （升序）或 DESC （降序）声明排序的方式，缺省为升序。)

```
SELECT SNO, SNAME, AGE
FROM STUDENTS
WHERE AGE BETWEEN 20 AND 23
ORDER BY AGE
```

例 20 找出年龄小于 23 岁、籍贯是湖南或湖北的学生的姓名和性别。（条件比较运算符＝、＜ 和逻辑运算符 AND （与），此外还可以使用的运算符有：＞（大于）、＞＝（大于等于）、＜＝（小于等于）、＜＞（不等于）、 NOT （非）、 OR （或）等。

谓词 LIKE 只能与字符串联用，常常是“ ＜列名＞ LIKE pattern”的格式。特殊字符“_”和“%”作为通配符。

谓词 IN 表示指定的属性应与后面的集合（括号中的值集或某个查询子句的结果）中的某个值相匹配，实际上是一系列的 OR （或）的缩写。谓词 NOT IN 表示指定的属性不与后面的集合中的某个值相匹配。

谓词 BETWEEN 是“ 包含于 … 之中 ”的意思。）

```
SELECT SNAME, SEX
FROM STUDENTS
WHERE AGE ＜ 23 AND BPLACE LIKE' 湖％ '
```

或

```
SELECT SNAME, SEX
FROM   STUDENTS
WHERE  AGE ＜ 23  AND  BPLACE  IN  （ ' 湖南 ' ，  ' 湖北 ' ）
```

例 22 找出学生表中籍贯是空值的学生的姓名和性别。（在 **SQL** 中不能使用条件：＜列名＞＝ **NULL** 。在 **SQL** 中只有一个特殊的查询条件允许查询 **NULL** 值：）

```
SELECT SNAME, SEX
FROM   STUDENTS
WHERE  BPLACE IS NULL
```

多表操作

例 23 找出成绩为 **95** 分的学生的姓名。（子查询）

```
SELECT SNAME
FROM     STUDENTS
WHERE     SNO ＝
    (SELECT SNO
     FROM   ENROLLS
     WHERE  GRADE ＝ 95)
```

例 24 找出成绩在 **90** 分以上的学生的姓名。

```
SELECT  SNAME
FROM   STUDENTS
WHERE  SNO IN
     (SELECT SNO
     FROM ENROLLS
     WHERE GRADE ＞ 90)
```

或

```
SELECT  SNAME
FROM   STUDENTS
WHERE  SNO ＝ ANY
     (SELECT SNO
     FROM ENROLLS
     WHERE GRADE ＞ 90)
```

例 25 查询全部学生的学生名和所学课程号及成绩。（连接查询）

```
SELECT  SNAME, CNO, GRADE
FROM   STUDENTS, ENROLLS
WHERE  STUDENTS.SNO ＝ ENROLLS.SNO
```

例 26 找出籍贯为山西或河北，成绩为 **90** 分以上的学生的姓名、籍贯和成绩。（当构造多表连接查询命令时，必须遵循两条规则。第一，连接条件数正好比表数少 **1** （若有三个表，就有两个连接条件 **)** ；第二，若一个表中的主关键字是由多个列组成，则对此主关键字中的每一个列都要有一个连接条件（也有少数例外情况））

```
SELECT  SNAME, BPLACE, GRADE
FROM   STUDENTS, ENROLLS
```

WHERE BPLACE IN (‘山西’，‘河北’) AND GRADE ＞＝ 90 AND STUDENTS.SNO=ENROLLS.SNO

例 28 查出课程成绩在 **80** 分以上的女学生的姓名、课程名和成绩。（ **FROM** 子句中的子查询）

```
SELECT SNAME,CNAME, GRADE
FROM (SELECT SNAME, CNAME , GRADE
      FROM STUDENTS, ENROLLS,COURSES
      WHERE SEX ＝ ' 女 ')
AS TEMP (SNAME, CNAME,GRADE)
WHERE GRADE ＞ 80
```

表达式与函数的使用

例 29 查询各课程的学时数。（算术表达式由算术运算符＋、－、 * 、/ 与列名或数值常量所组成。）

```
SELECT CNAME,COURSE_TIME ＝ CREDIT*16
FROM COURSES
```

例 30 找出教师的最小年龄。(内部函数：**SQL** 标准中只使用 **COUNT** 、**SUM** 、**AVG** 、**MAX** 、 **MIN** 函数，称之为聚集函数（ **Set Function** ）。 **COUNT** 函数的结果是该列统计值的总数目， **SUM** 函数求该列统计值之和， **AVG** 函数求该列统计值之平均值，**MAX** 函数求该列最大值， **MIN** 函数求该列最小值。）

```
SELECT MIN(AGE)
FROM TEACHERS
```

例 31 统计年龄小于等于 **22** 岁的学生人数。（统计）

```
SELECT COUNT(*)
FROM STUDENTS
WHERE AGE < ＝ 22
```

例 32 找出学生的平均成绩和所学课程门数。

```
SELECT SNO, AVG(GRADE), COURSES ＝ COUNT(*)
FROM ENROLLS
GROUP BY SNO
```

例 34 找出年龄超过平均年龄的学生姓名。

```
SELECT SNAME
FROM STUDENTS
WHERE AGE ＞
   (SELECT AVG(AGE)
    FROM STUDENTS)
```

例 35 找出各课程的平均成绩，按课程号分组，且只选择学生超过 3 人的课程的成绩。（ GROUP BY 与 HAVING

GROUP BY 子句把一个表按某一指定列（或一些列）上的值相等的原则分组，然后再对每组数据进行规定的操作。

GROUP BY 子句总是跟在 WHERE 子句后面，当 WHERE 子句缺省时，它跟在 FROM 子句后面。

HAVING 子句常用于在计算出聚集之后对行的查询进行控制。）

```
SELECT CNO, AVG(GRADE), STUDENTS ＝ COUNT(*)
FROM ENROLLS
GROUP BY CNO
HAVING COUNT(*) >= 3
```

相关子查询

例 37 查询没有选任何课程的学生的学号和姓名。（当一个子查询涉及到一个来自外部查询的列时，称为相关子查询（ Correlated Subquery) 。相关子查询要用到存在测试谓词 EXISTS 和 NOT EXISTS ，以及 ALL 、 ANY （ SOME ）等。）

```
SELECT SNO, SNAME
FROM   STUDENTS
WHERE  NOT EXISTS
    (SELECT *
    FROM ENROLLS
    WHERE ENROLLS.SNO=STUDENTS.SNO)
```

例 38 查询哪些课程只有男生选读。

```
SELECT DISTINCT CNAME
FROM   COURSES C
WHERE ' 男 ' ＝ ALL
    (SELECT SEX
     FROM  ENROLLS ， STUDENTS
     WHERE ENROLLS.SNO=STUDENTS.SNO AND
         ENROLLS.CNO=C.CNO)
```

例 39 要求给出一张学生、籍贯列表，该表中的学生的籍贯省份，也是其他一些学生的籍贯省份。

```
SELECT  SNAME, BPLACE
FROM   STUDENTS A
WHERE  EXISTS
    (SELECT *
     FROM STUDENTS B
     WHERE A.BPLACE=B.BPLACE AND
         A.SNO < > B.SNO)
```

例 40 找出选修了全部课程的学生的姓名。

本查询可以改为：查询这样一些学生，没有一门课程是他不选修的。

```
SELECT   SNAME
FROM    STUDENTS
WHERE   NOT EXISTS
     (SELECT *
      FROM COURSES
      WHERE   NOT EXISTS
           (SELECT *
            FROM ENROLLS
            WHERE ENROLLS.SNO  =  STUDENTS.SNO
              AND ENROLLS.CNO  =  COURSES.CNO))
```

关系代数运算

例 41 设有某商场工作人员的两张表：营业员表 SP_SUBORD 和营销经理表 SP_MGR ，其关系数据模式如下：

SP_SUBORD (SALPERS_ID, SALPERS_NAME, MANAGER_ID, OFFICE)

SP_MGR (SALPERS_ID, SALPERS_NAME, MANAGER_ID, OFFICE)

其中，属性 SALPERS_ID 为工作人员的编号 ，SALPERS_NAME 为工作人员的姓名 ，MANAGER_ID 为所在部门经理的编号 ，OFFICE 为工作地点。

若查询全部商场工作人员，可以用下面的 SQL 语句：

```
(SELECT * FROM SP_SUBORD)
UNION
(SELECT * FROM SP_MGR)
```

或等价地用下面的 SQL 语句：

```
SELECT *
FROM (TABLE SP_SUBORD UNION TABLE SP_MGR)
```

（ 2 ） INTERSECT

```
(SELECT * FROM SP_SUBORD)
INTERSECT
(SELECT * FROM SP_MGR)
```

或等价地用下面的 SQL 语句：

```
SELECT *
FROM (TABLE SP_SUBORD INTERSECT TABLE SP_MGR)
```

或用带 ALL 的 SQL 语句：

```
(SELECT * FROM SP_SUBORD)
INTERSECT ALL
(SELECT * FROM SP_MGR)
```

或

```
SELECT *
```

FROM (TABLE SP_SUBORD INTERSECT ALL TABLE SP_MGR)

（ 3 ） EXCEPT

(SELECT * FROM SP_MGR)

EXCEPT

(SELECT * FROM SP_SUBORD)

或等价地用下面的 SQL 语句：

SELECT *

FROM (TABLE SP_MGR EXCEPT TABLE SP_ SUBORD)

或用带 ALL 的 SQL 语句：

(SELECT * FROM SP_MGR)

EXCEPT ALL

(SELECT * FROM SP_SUBORD)

例 42 查询籍贯为四川、课程成绩在 80 分以上的学生信息及其成绩。（自然连接）

(SELECT * FROM STUDENTS

WHERE BPLACE=‘ 四川 ’)

NATURAL JOIN

(SELECT * FROM ENROLLS

WHERE GRADE >=80)

例 3.43 列出全部教师的姓名及其任课的课程号、班级。

（外连接与外部并外连接允许在结果表中保留非匹配元组，空缺部分填以 NULL 。外连接的作用是在做连接操作时避免丢失信息。

外连接有 3 类：

（ 1 ）左外连接（ Left Outer Join ）。连接运算谓词为 LEFT [OUTER] JOIN ，其结果表中保留左关系的所有元组。

（ 2 ）右外连接（ Right Outer Join ）。连接运算谓词为 RIGHT [OUTER] JOIN ，其结果表中保留右关系的所有元组。

（ 3 ）全外连接（ Full Outer Join ）。连接运算谓词为 FULL [OUTER] JOIN ，其结果表中保留左右两关系的所有元组。）

SELECT TNAME, CNO, CLASS

FROM TEACHERS LEFT OUTER JOIN TEACHING USING (TNO)

SQL 的数据操纵

例 44 把教师李映雪的记录加入到教师表 TEACHERS 中。（插入）

INSERT INTO TEACHERS

VALUES(1476 ， ' 李映雪 ' ， 44 ， ' 副教授 ')

例 45 成绩优秀的学生将留下当教师。

INSERT INTO TEACHERS (TNO ， TNAME)

SELECT DISTINCT SNO ， SNAME

FROM STUDENTS ， ENROLLS

WHERE STUDENTS.SNO ＝ ENROLLS.SNO AND GRADE ＞＝ 90

例 47 把所有学生的年龄增加一岁。（修改）

UPDATE STUDENTS

SET AGE ＝ AGE+1

例 48 学生张春明在数据库课考试中作弊，该课成绩应作零分计。

```
UPDATE ENROLLS
SET GRADE ＝ 0
WHERE CNO ＝ 'C1' AND
   ' 张春明 ' ＝
   (SELECT SNAME
   FROM STUDENTS
   WHERE STUDENTS.SNO=ENROLLS.SNO)
```

例 49 从教师表中删除年龄已到 60 岁的退休教师的数据。（删除）

```
DELETE FROM TEACHERS
WHERE  AGE ＞＝ 60
```

SQL 的数据控制

例 50 授予 LILI 有对表 STUDENTS 的查询权。（表 / 视图特权的授予

一个 SQL 特权允许一个被授权者在给定的数据库对象上进行特定的操作。授权操作的数据库对象包括：表 / 视图、列、域等。授权的操作包括： INSERT 、 UPDATE 、 DELETE 、 SELECT 、 REFERENCES 、 TRIGGER 、 UNDER 、 USAGE 、 EXECUTE 等。其中 INSERT 、UPDATE 、DELETE 、SELECT 、REFERENCES 、 TRIGGER 有对表做相应操作的权限，故称为表特权。）

```
GRANT SELECT ON STUDENTS
TO LILI
WITH GRANT OPTION
```

例 51 取消 LILI 的存取 STUDENTS 表的特权。

```
REVOKE ALL
ON STUDENTS
FROM LILI CASCADE
```

不断补充中：

1. 模糊查找:

它判断列值是否与指定的字符串格式相匹配。可用于 char、varchar、text、ntext、datetime 和 smalldatetime 等类型查询。

可使用以下通配字符：

百分号%：可匹配任意类型和长度的字符，如果是中文，请使用两个百分号即%%。

下划线_：匹配单个任意字符，它常用来限制表达式的字符长度。

方括号[]：指定一个字符、字符串或范围，要求所匹配对象为它们中的任一个。[^]：其取值也[] 相同，但它要求所匹配对象为指定字符以外的任一个字符。

例如：

限制以 Publishing 结尾，使用 LIKE '%Publishing'

限制以 A 开头：LIKE '[A]%'

限制以 A 开头外：LIKE '[^A]%'

2.更改表格

ALTER TABLE table_name
ADD COLUMN column_name DATATYPE
说明：增加一个栏位（没有删除某个栏位的语法。)
ALTER TABLE table_name
ADD PRIMARY KEY (column_name)
说明：更改表得的定义把某个栏位设为主键。
ALTER TABLE table_name
DROP PRIMARY KEY (column_name)
说明：把主键的定义删除。

3.group by

在 select 语句中可以使用 group by 子句将行划分成较小的组，然后，使用聚组函数返回每一个组的汇总信息，另外，可以使用 having 子句限制返回的结果集。group by 子句可以将查询结果分组，并返回行的汇总信息 Oracle 按照 group by 子句中指定的表达式的值分组查询结果。

在带有 group by 子句的查询语句中，在 select 列表中指定的列要么是 group by 子句中指定的列，要么包含聚组函数

select max(sal),job emp group by job;

(注意 max(sal),job 的 job 并非一定要出现，但有意义)

查询语句的 select 和 group by ,having 子句是聚组函数唯一出现的地方，在 where 子句中不能使用聚组函数。

select deptno,sum(sal) from emp where sal>1200 group by deptno having sum(sal)>8500 order by deptno;

当在 gropu by 子句中使用 having 子句时，查询结果中只返回满足 having 条件的组。在一个 sql 语句中可以有 where 子句和 having 子句。having 与 where 子句类似，均用于设置限定条件

where 子句的作用是在对查询结果进行分组前，将不符合 where 条件的行去掉，即在分组之前过滤数据，条件中不能包含聚组函数，使用 where 条件显示特定的行。

having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，条件中经常包含聚组函数，使用 having 条件显示特定的组，也可以使用多个分组标准进行分组。

查询每个部门的每种职位的雇员数

select deptno,job,count(*) from emp group by deptno,job;

4.外连接与内连接

有时候，即使在连接的表中没有相应的行，用户可能想从一张表中看数据，Oracle 提供了外连接实现该功能。

内连接是指连接查询只显示完全满足连接条件的记录，即等值连接，外连接的查询结果是内连接查询结果的扩展。外连接不仅返回满足连接条件的所有记录而且也返回了一个表中那些在另一个表中没有匹配行的记录。外连接的操作符是“+”。“+”号放在连接条件中信息不完全的那一边（即没有相应行的那一边）。运算符“+”影响 NULL 行的建立。建一行或多行 NULL 来匹配连接的表中信息完全的行。

外连接运算符“+”只能出现在 where 子句中表达式的一边。
假如在多张表之间有多个连接条件，外连接运算符不能使用 or,in 逻辑运算符与其它条件组合。

假如 emp 表中 deptno=10 的 ename 为空值，dept 表中 deptno=20 的 loc 为空值：

1.

```
select
ename,dept.deptno,loc
from
emp,dept
where
emp.depno(+)=dept.deptno;
```

如果在 dept.deptno 中有的数值在 emp.deptno 中没有值，则在做外连接时，
结果中 ename 会产生一个空值。(emp.deptno=10)

2.

```
select
ename,dept.deptno,loc
from
emp,dept
where
emp.depno=dept.deptno(+);
```

如果在 emp.deptno 中有的数值在 dept.deptno 中没有值，则在做外连接时，
结果中 loc 会产生一个空值。。(dept.deptno=20)

5.自连接

自连接是指同一张表的不同行间的连接。该连接不受其他表的影响。用自连接可以比较同一张表中不同行的某一列的值。因为自连接查询仅涉及到某一张表与其自身的连接。所以在 from 子句中该表名出现两次，分别用两个不同的别名表示，两个别名当作两张不同的表进行处理，与其它的表连接一样，别名之间也使用一个或多个相关的列连接。为了区分同一张表的不同行的列，在名前永别名加以限制。

```
select
worker.ename,
manager.ename manager
from
emp worker,
emp manager
where
work.mgr=manager.empno;
```

6.集合运算

基合运算符可以用于从多张表中选择数据。

①UNION 运算

用于求两个结果集合的并集（两个结果集合的所有记录），并自动去掉重复行。

```
select ename,sal from account where sal>2000
union
select ename,sal from research where sal>2000
union
select ename,sal from sales where sal>2000;
```

注：ename,sal 是必须一致的。

②UNION ALL 运算

用于求两个结果集合的并集（两个结果集中的所有记录），并且不去掉重复行。

```
select ename,sal from account where sal>2000
union
select ename,sal from research where sal>2000
union
select ename,sal from sales where sal>2000;
```

③INTERSECT 运算

intersect 运算返回查询结果中相同的部分。

各部门中有哪些相同的职位？

```
select Job from account
intersect
select Job from research
intersect
select Job from sales;
```

④MINUS 运算

minus 返回两个结果集的差集。（在第一个结果集中存在的，而在第二个结果集中不存在的行。）

有那些职位是财务部中有，而在销售部门中没有？

```
select Job from account
minus
select Job from sales;
```