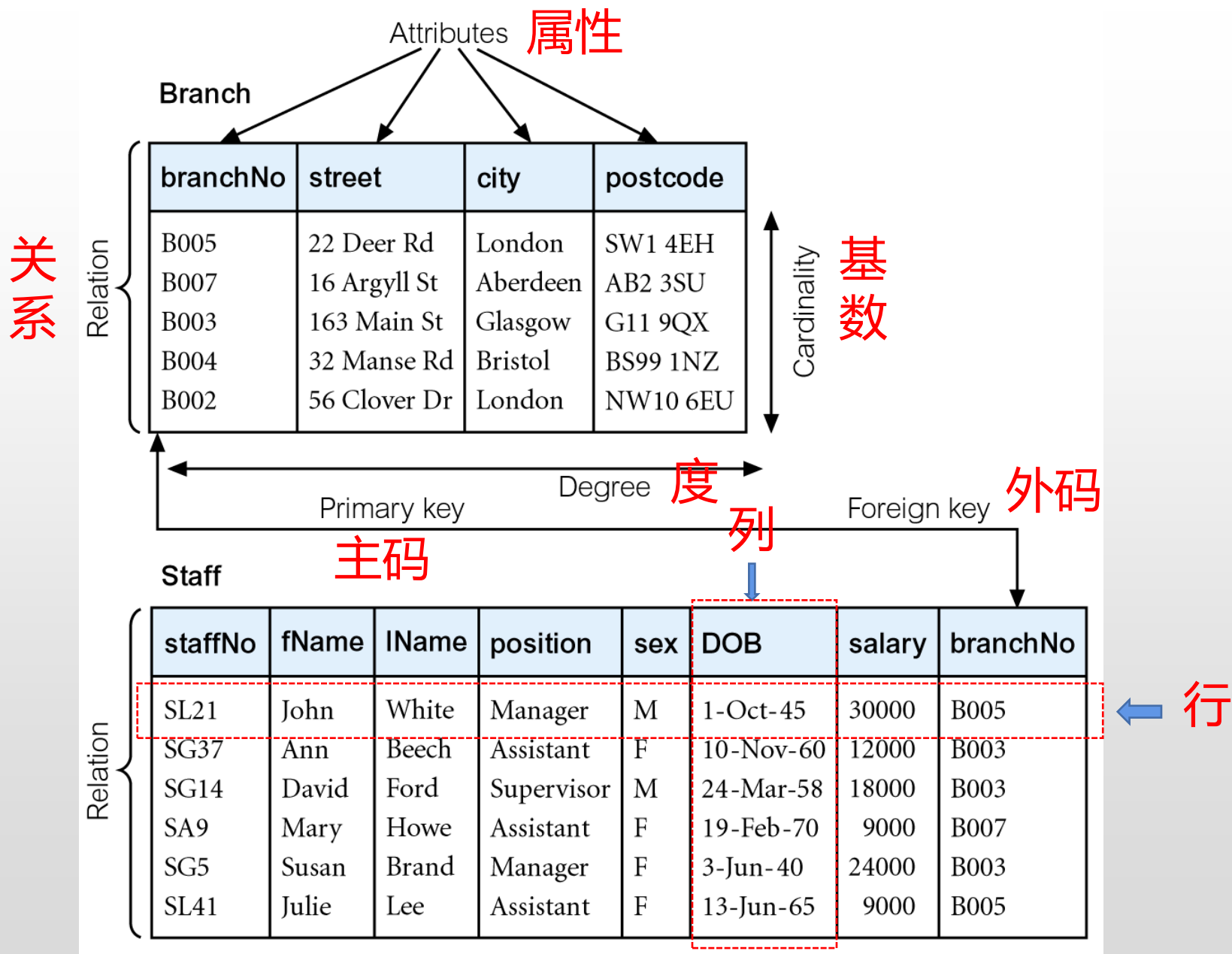


- 属性(Attribute)

- 关系中不同列可以对应相同的域
- 为了加以区分，必须对每列起一个名字，称为属性
- n 目关系必有 n 个属性

- 码(key)

- 候选码(Candidate key): 若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为该关系的一个候选码
- 主码(Primary Key, PK): 若一个关系有多个候选码，则选定其中一个为主码
- 全码(All key): 关系模式的所有属性组是这个关系模式的候选码，称为全码
- 主属性(primary attribute): 候选码的所有属性
- 非主属性(非码属性): 不包含在任何候选码的属性



- 选择示例：

- 查询信息系(IS系)全体学生

$\sigma_{Sdept='IS'} (Student)$

Sno	Sname	Ssex	Sage	Sdept
201215125	张立	男	19	IS


- 查询年龄小于20岁的学生

$\sigma_{Sage < 20} (Student)$

Sno	Sname	Ssex	Sage	Sdept
201215122	刘晨	女	19	IS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS


- 投影示例：

- 查询学生的姓名和所在系

$\pi_{\text{Sname, Sdept}}(\text{Student})$ 

Sname	Sdept
李勇	CS
刘晨	CS
王敏	MA
张立	IS

- 查询Student表中都有哪些系

$\pi_{\text{Sdept}}(\text{Student})$ 

Sdept
CS
MA
IS

● 连接示例：

R		
A	B	C
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

S	
B	E
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

一般连接 $R \bowtie S$ 的结果如下：
 $C < E$

$R \bowtie S$ $C < E$				
A	$R.B$	C	$S.B$	E
a_1	b_1	5	b_2	7
a_1	b_1	5	b_3	10
a_1	b_2	6	b_2	7
a_1	b_2	6	b_3	10
a_2	b_3	8	b_3	10

等值连接 $R \bowtie S$ 的结果如下：
 $R.B = S.B$

A	$R.B$	C	$S.B$	E
a_1	b_1	5	b_1	3
a_1	b_2	6	b_2	7
a_2	b_3	8	b_3	10
a_2	b_3	8	b_3	2

自然连接 $R \bowtie S$ 的结果如下：

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2

- 悬浮元组 (Dangling tuple)
 - 两个关系R 和S在做自然连接时，关系R中某些元组有可能在S中不存在公共属性上值相等的元组，从而造成R中这些元组在操作时被舍弃了，这些被舍弃的元组称为**悬浮元组**
- 外连接(\bowtie , Outer join, Full outer join全外连接)
 - 如果把悬浮元组也保存在结果关系中，而在其他属性上填空值(Null Value)，就叫做外连接，是左外连接与右外连接的并集
- 左外连接(\bowtie , Left outer join, Left join)
 - 只保留左边关系R 中的悬浮元组就叫做左外连接
- 右外连接(\bowtie , Right outer join, Right join)
 - 只保留右边关系R 中的悬浮元组就叫做左外连接

• 外连接示例：

<i>A</i>	<i>B</i>	<i>C</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	5	3
<i>a</i> ₁	<i>b</i> ₂	6	7
<i>a</i> ₂	<i>b</i> ₃	8	10
<i>a</i> ₂	<i>b</i> ₃	8	2
<i>a</i> ₂	<i>b</i> ₄	12	NULL

(b) 左外连接

$R \bowtie S$

<i>A</i>	<i>B</i>	<i>C</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	5	3
<i>a</i> ₁	<i>b</i> ₂	6	7
<i>a</i> ₂	<i>b</i> ₃	8	10
<i>a</i> ₂	<i>b</i> ₃	8	2
NULL	<i>b</i> ₅	NULL	2

(c) 右外连接

$R \bowtie S$

<i>A</i>	<i>B</i>	<i>C</i>	<i>E</i>
<i>a</i> ₁	<i>b</i> ₁	5	3
<i>a</i> ₁	<i>b</i> ₂	6	7
<i>a</i> ₂	<i>b</i> ₃	8	10
<i>a</i> ₂	<i>b</i> ₃	8	2
<i>a</i> ₂	<i>b</i> ₄	12	NULL
NULL	<i>b</i> ₅	NULL	2

(a) 外连接

$R \bowtie S$

• 除示例：

R		
A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

(a)

S		
B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(b)

$R \div S$	
A	
a_1	

(c)

a_1 的象集为 $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$
 a_2 的象集为 $\{(b_3, c_7), (b_2, c_3)\}$
 a_3 的象集为 $\{(b_4, c_6)\}$
 a_4 的象集为 $\{(b_6, c_6)\}$



S 在 (B, C) 上的投影为：
 $\{(b_1, c_2), (b_2, c_1), (b_2, c_3)\}$

- 除的实际应用：

- 设有一个现实意义的集合，希望在另一个集合中找出“包含”该集合的元组集，可用“除”实现

例1：找出选修了所有课程的学生

- “所有课程”
- “学生”
- “学生” \div “所有课程”

例2：找出选修了所有张三所选课的学生

- “张三所选课”
- “学生”
- “学生” \div “张三所选课”

模式定义(创建)

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>;

- 创建模式必须具有DBA权限，或获得了DBA授予的CREATE SCHEMA权限
- 若模式名缺失，则模式名默认为用户名
- CREATE SCHEMA可以接受CREATE TABLE，CREATE VIEW和GRANT子句

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>
[<表定义子句>|<视图定义子句>|<授权定义子句>]

示例：

[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] CREATE SCHEMA AUTHORIZATION WANG;

[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));
```

模式删除

DROP SCHEMA <模式名> <CASCADE|RESTRICT>;

- **CASCADE(级联)**

- 删除模式的同时把该模式中所有的数据库对象全部删除。

- **RESTRICT(限制)**

- 如果该模式中定义了下属的数据库对象(如表、视图等), 则拒绝该删除语句的执行。
- 仅当该模式中没有任何下属的对象时才能执行。

- **[例3.4]: DROP SCHEMA TEST CASCADE; --删除模式TEST及该模式中定义的表TAB1**

[例3.6] 建立一张 “课程Course表” 。

```
CREATE TABLE Course
    (Cno      CHAR(4) PRIMARY KEY,
     Cname    CHAR(40),
     Cpno     CHAR(4),
     Ccredit  SMALLINT,
     FOREIGN KEY (Cpno) REFERENCES Course (Cno)
    );
```

数据类型	含义
CHAR(<i>n</i>), CHARACTER(<i>n</i>)	长度为 <i>n</i> 的定长字符串
VARCHAR(<i>n</i>), CHARACTERVARYING(<i>n</i>)	最大长度为 <i>n</i> 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(<i>p</i> , <i>d</i>)	定点数，由 <i>p</i> 位数字（不包括符号、小数点）组成，小数后面有 <i>d</i> 位数字
DECIMAL(<i>p</i> , <i>d</i>), DEC(<i>p</i> , <i>d</i>)	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(<i>n</i>)	可选精度的浮点数，精度至少为 <i>n</i> 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

示例：

[例3.13] 为学生-课程数据库中的Student，Course，SC三个表建立索引。Student表按学号升序建唯一索引，Course表按课程号升序建唯一索引，SC表按学号升序和课程号降序建唯一索引。

```
CREATE UNIQUE INDEX Stusno ON Student(Sno);  
CREATE UNIQUE INDEX Coucno ON Course(Cno);  
CREATE UNIQUE INDEX SCno ON SC(Sno ASC,Cno DESC);
```

```
CREATE UNIQUE INDEX pg_job_id_index ON pg_job USING btree (job_id) TABLESPACE pg_global  
CREATE INDEX gs_asp_sampletime_index ON gs_asp USING btree (sample_time) TABLESPACE pg_default
```

openGauss索引示例

索引修改

ALTER INDEX <旧索引名> **RENAME TO** <新索引名>;

- 上述命令将索引改名。

[例3.14] 将SC表的SCno索引名改为SCSno。

ALTER INDEX SCno RENAME TO SCSno;

数据查询的一般形式

- 查询语句(SELECT)格式

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>]...| (SELECT 语句) [AS]<别名> --
派生表

[WHERE <条件表达式>]

[GROUP BY <列名1>[HAVING <条件表达式>]]

[ORDER BY <列名2>[ASC|DESC]]

[limit n]; --该选项是openGauss支持的语法结构，表明只显示前面n条记录

- 使用列别名改变查询结果的列标题:

```
SELECT Sname NAME, 'Year of Birth:' BIRTH, 2019-Sage BIRTHDAY, LOWER(Sdept)  
DEPARTMENT  
FROM Student;
```

NAME	BIRTH	BIRTHDAY	DEPARTMENT
李勇	Year of Birth:	1994	cs
刘晨	Year of Birth:	1995	cs
王敏	Year of Birth:	1996	ma
张立	Year of Birth:	1995	is

- 消除取值重复的行

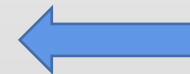
- 如果没有指定DISTINCT关键词，则缺省为ALL
- 指定DISTINCT关键词，去掉表中重复的行

[例3.21] 查询选修了课程的学生学号。

SELECT Sno FROM SC; 等价于: SELECT ALL Sno FROM SC;

SELECT DISTINCT Sno FROM SC;

Sno
20121512
1
20121512
2



Sno
201215121
201215121
201215121
201215122
201215122

- 注意: DISTINCT短语的作用范围是**所有**目标列

```
SELECT DISTINCT Cno, DISTINCT Grade  
FROM SC;
```



```
SELECT DISTINCT Cno, Grade  
FROM SC;
```



- 查询满足条件的元组
 - 通过WHERE子句实现

表3.6 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（逻辑运算）	AND, OR, NOT

[例3.32] 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%';
```

[例3.33] 查询所有不姓刘的学生姓名、学号和性别。

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

- 使用换码字符将通配符转义为普通字符

[例3.34] 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3.35] 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 '\';
```

注：ESCAPE '\ ' 表示 '\ ' 为换码字符

- 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
- 对于空值，排序时显示的次序由具体系统实现来决定

[例3.39] 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3.47] 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;
```

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

2. 自身连接

- 什么是自身连接？
 - 一个表与其自己进行的连接
- 自身连接的使用方法
 - 需要给表起**别名**以示区别
 - 由于所有属性名都是同名属性，因此**必须使用别名前缀**

[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;
```

FIRST表 (Course表)

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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

SECOND表 (Course表)

课程号 Cno	课程名 Cname	先行课 Cpno	学分 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



Cno	Cpno
1	7
3	5
5	6

3. 外连接

- 外连接类型

- (全)外连接 左外连接和右外连接

连接类型	特点
左外连接	<ul style="list-style-type: none">Left out join列出左边关系中所有的元组
右外连接	<ul style="list-style-type: none">Right out join列出右边关系中所有的元组
外连接	<ul style="list-style-type: none">Outer join, full outer join以指定表为连接主体，将主体表中不满足连接条件的元组一并输出左外连接与右外连接的并集
(内)连接	<ul style="list-style-type: none">Inner join, join只输出满足连接条件的元组

嵌套查询

- 查询块(query block): 构造嵌套查询的基本单元
 - SELECT...FROM...WHERE...(S-F-W)称为一个查询块
- 什么是嵌套查询(Embedded query)
 - 也称子查询(subquery), 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询

```
SELECT Sname    /*外层查询/父查询*/  
FROM Student  
WHERE Sno IN  
          (SELECT Sno /*内层查询/子查询*/  
           FROM SC  
           WHERE Cno='2');
```


[例3.55] 查询与“刘晨”在同一个系学习的学生。

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

```
SELECT S1.Sno, S1.Sname, S1.Sdept
FROM Student S1, Student S2
WHERE S2.Sname='刘晨' AND S1.Sdept=S2.Sdept;
```


[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
FROM Student
WHERE Sno IN
      (SELECT Sno
       FROM SC
       WHERE Cno IN
            (SELECT Cno
             FROM Course
             WHERE Cname= '信息系统'));
```

```
SELECT Sno, Sname
FROM Student, SC, Course
WHERE Course.Cname='信息系统' AND Student.Sno=SC.Sno AND SC.Cno=Course.Cno;
```

[例3.58] 查询非计算机科学系中比计算机科学系**任意**学生年龄小的学生姓名和年龄。

```
SELECT Sname, Sage
FROM Student
WHERE Sage < ANY(SELECT Sage
                  FROM Student
                  WHERE Sdept='CS') AND Sdept <> 'CS';
/*父查询块中的条件 */
```



Sname	Sage
王敏	18
张立	19

```
SELECT Sname,Sage
FROM Student
WHERE Sage < (SELECT MAX(Sage)
              FROM Student
              WHERE Sdept = 'CS') AND Sdept <> 'CS';
```



用聚集函数实现

[例3.59] 查询非计算机科学系中比计算机科学系**所有**学生年龄小的学生姓名和年龄。

```
SELECT Sname, Sage
FROM Student
WHERE Sage < ALL(SELECT Sage
                  FROM Student
                  WHERE Sdept='CS') AND Sdept <> 'CS';
/*父查询块中的条件 */
```



Sname	Sage
王敏	18

```
SELECT Sname,Sage
FROM Student
WHERE Sage < (SELECT MIN(Sage)
              FROM Student
              WHERE Sdept = 'CS') AND Sdept <> 'CS';
```



用聚集函数实现

[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 Cno='1' AND Sno=Student.Sno);
```

是否为相关子查询?

[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 Cno='1' AND Sno=Student.Sno);
```

- 用NOT EXISTS谓词表示:

SELECT DISTINCT Sno

FROM SC SCX

WHERE NOT EXISTS (SELECT *

--不存在

FROM SC SCY

--有一门课程

WHERE SCY.Sno ='201215122' AND

--该同学没有学过

NOT EXISTS (SELECT *

FROM SC SCZ

WHERE

SCZ.Sno=SCX.Sno

AND

SCZ.Cno=SCY.Cno));

[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;
```

- **UNION**：将多个查询结果合并起来时，系统**自动去掉重复元组**
- **UNION ALL**：将多个查询结果合并起来时，**保留重复元组**

[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'
```



```
SELECT *  
FROM Student  
WHERE Sdept= 'CS' AND Sage<=19;
```

INTERSECT

```
SELECT *  
FROM Student  
WHERE Sage<=19;
```


[例 3.68] 查询计算机科学系的学生与年龄不大于19岁的学生的差集。

```
SELECT *  
FROM Student  
WHERE Sdept='CS'
```



```
SELECT *  
FROM Student  
WHERE Sdept='CS' AND Sage>19;
```

EXCEPT

```
SELECT *
```

```
FROM Student
```

- openGauss、oracle用到的差集关键词为MINUS，而不是EXCEPT

```
WHERE Sage<=19;
```

插入数据

- 语法格式：

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

INTO子句	VALUES子句
<ul style="list-style-type: none">• 指定要插入数据的表名及属性列• 属性列的顺序可与表定义中的顺序不一致• 没有指定属性列：表示要插入的是一条完整的元组，且属性列属性与表定义中的顺序一致• 指定部分属性列：插入的元组在其余属性列上取空值	<ul style="list-style-type: none">• 提供的值必须与INTO子句匹配<ul style="list-style-type: none">• 值的个数• 值的类型

- 两种插入数据的两种方式

- 插入元组：用于插入新元组
- 插入子查询结果：利用已有数据导出的结果

[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Student表中。

```
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 ('201215128','1')

```
INSERT INTO SC(Sno, Cno)
VALUES ('201215128','1');
```

- 关系数据库管理系统将在新插入记录的Grade列上自动地赋空值或者

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

- 插入子查询结果：

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

- 查询中的SELECT子句的目标列必须与INTO子句后面的属性列匹配
 - 值的个数、值的类型

[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。

第一步：建表

```
CREATE TABLE Dept_age
(Sdept CHAR(15), --系名
Avg_age SMALLINT /*学生平均年龄 */
);
```

第二步：插入数据

```
INSERT
INTO Dept_age (Sdept, Avg_age)
SELECT Sdept, AVG(Sage)
FROM Student
GROUP BY Sdept;
```

修改数据

- 语句格式:

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

- 功能:

- 修改指定表中满足WHERE子句条件的元组
- SET子句给出<表达式>的值用于取代相应的属性列
- 如果省略WHERE子句, 表示要修改表中的所有元组

- 数据修改的三种方式：

- 修改某一个元组的值

[例3.73] 将学生201215121的年龄改为22岁。

```
UPDATE Student SET Sage=22 WHERE Sno='201215121';
```

- 修改多个元组的值

[例3.74] 将所有学生的年龄都增加1岁。

```
UPDATE Student SET Sage=Sage +1;
```

- 带子查询的修改语句

[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 (SELECT Sno
                                      FROM Student
                                      WHERE Sdept='CS');
```

删除数据

- 语句格式:

DELETE FROM <表名> [WHERE <条件>];

- 功能:

- 删除指定表中满足WHERE子句条件的元组。

- WHERE子句

- 指定要删除的元组。
- 缺省表示要删除表中的全部元组，表的定义仍在字典中。

- 数据删除的三种方式：

- 删除某一个元组的值

[例3.76] 删除学号为201215128的学生记录

```
DELETE FROM Student WHERE Sno='201215128';
```

- 删除多个元组的值

[例3.77] 删除所有的学生选课记录

```
DELETE FROM SC;
```

- 带子查询的删除语句

[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 (SELECT Sno
                                FROM Student
                                WHERE Sdept='CS');
```


openGauss之TRUNCATE命令

- 语法格式：

TRUNCATE TABLE table_name;

- 功能：清理表数据，DELETE、DROP和TRUNCATE三者比较

DELETE	DROP	TRUNCATE
<ul style="list-style-type: none">• 删除表中满足条件的所有行• 逐行删除，每删除一行将在事务日志登记删除记录• 删除内容，不删除定义，不释放空间	<ul style="list-style-type: none">• 删除内容和定义，释放空间	<ul style="list-style-type: none">• 效果同DELETE FROM table_name;• 不扫描表，按数据页删除，因而删除速度快，使用系统资源和事务日志少• 删除内容，不删除定义，释放空间

[例3.77]可用TRUNCATE改写：TRUNCATE TABLE SC;

数据更新小结

- 数据更新包括数据的插入、修改和删除
 - 三种操作都**只能在单表上**操作
 - 语法格式上易与多表查询混淆
- 当**修改或删除**操作涉及多张表时，只能通过**子查询完成**
- 一些RDBMS产品，如openGauss、MySQL支持**一条insert语句实现插入多条记录**，而不需要分别执行多条insert语句
- openGauss的truncate删除命令效率比delete更高，因其需要的事务日志资源更少

1. GRANT命令

- GRANT命令格式:

```
GRANT <权限> [, <权限>] ...  
ON <对象类型> <对象名> [, <对象类型> <对象名>] ...  
TO <用户> [, <用户>] ...  
[WITH GRANT OPTION];
```

- 语义: 将对指定操作对象的指定操作权限授予指定的用户

谁能够发出GRANT语句?

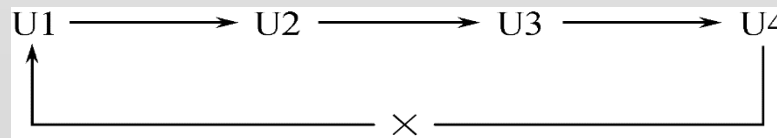
- DBA; 数据库对象创建者(即属主owner);
- 拥有该权限的用户

接收权限的用户:

- 一个或多个具体用户;
- PUBLIC(即全体用户)

WITH GRANT OPTION子句

- 指定了该子句, 则获得权限的用户可以把这种权限再授予其他用户
- 没有指定, 则不能传播已获得的权限
- SQL标准规定不允许循环授权



[例4.1] 把查询Student表权限授给用户U1。

```
GRANT SELECT ON TABLE Student TO U1;
```

[例4.2] 把对Student表和Course表的全部权限授予用户U2和U3。

```
GRANT ALL PRIVILEGES ON TABLE Student, Course  
TO U2, U3;
```

[例4.3] 把对表SC的查询权限授予所有用户。

```
GRANT SELECT ON TABLE SC TO  
PUBLIC;
```

[例4.4] 把查询Student表和修改学生学号的权限授给用户U4。

```
GRANT UPDATE(SNO), SELECT ON TABLE  
Student TO U4;
```

对属性列的授权时必须明确指出相应属性列名

[例4.5] 把对表SC的INSERT权限授予U5用户，并允许他再将此权限授予其他用户。

```
GRANT INSERT ON TABLE SC TO U5 WITH  
GRANT OPTION;
```

- 执行例4.5后，U5不仅拥有了对表SC的INSERT权限，还可以传播此权限

[例4.6] U5用户发布以下命令，将得到的权限转授给U6。

```
GRANT INSERT ON TABLE SC TO U6 WITH  
GRANT OPTION;
```

- U6还可以转授该权限给U7

```
GRANT INSERT ON TABLE SC TO U7;
```

[例4.7] U6转授该权限给U7，但U7不能再传播该权限给其他用户。

2.REVOKE命令

- REVOKE命令格式:

```
REVOKE <权限>[,<权限>]...  
ON <对象类型> <对象名>[,<对象类型> <对象名>]...  
FROM <用户>[,<用户>]...  
[CASCADE | RESTRICT];
```

- 语义：由数据库管理员或其他授权者收回已授予的权限

[例4.8] 把用户U4修改学生学号的权限收回。

```
REVOKE UPDATE(SnO) ON TABLE Student  
FROM U4;
```

[例4.9] 收回所有用户对表SC的查询权限。

```
REVOKE SELECT ON TABLE SC FROM PUBLIC;
```

[例4.10] 把用户U5对SC表的INSERT权限收回。
REVOKE INSERT ON TABLE SC FROM U5
CASCADE;

- 将用户U5的INSERT权限收回的时候应该使用CASCADE，否则拒绝执行该语句
- 如果U6或U7还从其他用户处获得对SC表的INSERT权限，则他们仍具有此权限，系统只收回直接或间接从U5处获得的权限

• 执行例4.8~4.10语句后学生-课程数据库中的用户权限定义表

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系Student	SELECT	不能
DBA	U2	关系Student	ALL	不能
DBA	U2	关系Course	ALL	不能
DBA	U3	关系Student	ALL	不能
DBA	U3	关系Course	ALL	不能
DBA	U4	关系Student	SELECT	不能

[例4.11] 通过角色来实现将一组权限授予一个用户

- 实现步骤：

- `CREATE ROLE R1;`
- `GRANT SELECT, UPDATE, INSERT ON TABLE Student TO R1;`
- `GRANT R1 TO 王平,张明,赵玲;`
- `REVOKE R1 FROM 王平;`

[例4.12] 角色权限的修改：增加权限

`GRANT DELETE ON TABLE STUDENT TO R1;`

[例4.13] 角色权限的修改：收回权限

`REVOKE SELECT ON TABLE STUDENT FROM R1;`