第三章：关系数据库标准语言SQL

数据库查询

SELECT语句语法格式

结果集中可以包含重复行

结果集中只能包含唯一行

从基本结果集中返回额外的行

```
SELECT [ ALL | DISTINCT ]
        [ TOP ( expression ) [ PERCENT ] [ WITH TIES ] ]
  <select_list>                                    /*指定要选择的列及其限定*/
  [ INTO new_table ]                               /*INTO 子句，指定结果存入新表*/
  [ FROM table_source ]                            /*FROM 子句，指定表或视图*/
  [ WHERE search_condition ]                       /*WHERE 子句，指定查询条件*/
  [ GROUP BY group_by_expression]                  /*GROUP BY 子句，指定分组表达式*/
  [ HAVING search_condition]                       /*HAVING 子句，指定分组统计条件*/
  [ ORDER BY order_expression [ ASC | DESC ] ]     /*ORDER BY 子句，指定排序表达式和顺序*/
```

升序

降序

SELECT语句的处理顺序

1	FROM
2	ON
3	JOIN
4	WHERE
5	GROUP BY
6	WITH CUBE 或 WITH ROLLUP
7	HAVING
8	SELECT
9	DISTINCT
10	ORDER BY
11	TOP

所有被使用的子句必须按语法说明中显示的顺序严格的排序。

SELECT_list语句语法格式

```
<select_list> ::=
    {
            *                                                    /*选择当前表或视图的所有列*/
      | { table_name | view_name | table_alias }.*               /*选择指定的表或视图的所有列*/
      | { { column_name | $IDENTITY | $ROWGUID }                /*选择指定的列*/
          | udt_column_name [ { . | :: } { { property_name | field_name }| method_name ( argument [ ,...n] ) } ]
                                                                 /*选择用户定义数据类型的属性、方法和字段
*/
          | expression
          [ [ AS ] column_alias ]                                /*AS 子句，定义列别名*/
        }
      | column_alias = expression                                /*选择指定列并更改列标题*/
    } [ ,...n ]
```

返回标识列

返回行 GUID 列

用户定义类型列的名称

用户定义类型的方法、属性或字段

替换列名的可选名

1、选择所有列

EG：查询PXSCJ数据库中XSB表的所有数据。

```
SELECT * FROM XSB;
```

2、选择表中指定的列

EG：查询PXSCJ数据库的XSB表中各个同学的姓名、专业和总学分。

```
SELECT XSB.Stu_ID,XSB.Sname,XSB.Tcredit
FROM XSB;
```

3、消除结果集中的重复行

EG：查询XSB表中的专业，消除结果集中重复行。

```
SELECT DISTINCT XSB.Major  FROM XSB;
```

比较：查询XSB表中的专业和性别，消除结果集中重复行。

```
SELECT DISTINCT XSB.Major,XSB.Ssex  FROM XSB;
```

4、定义列别名

EG：查询XSB表中学生的学号、姓名和总学分，结果中各列的标题分别指定为“学号”、“姓名”和“总学分”。

```
SELECT XSB.Stu_ID AS 学号,XSB.Sname AS 姓名,XSB.Tcredit AS 总学分
FROM XSB;
```

5、限制结果集返回行数

EG：查询XSB表中学生的姓名、专业和总学分，返回结果集的前6行。

```
SELECT TOP(6) XSB.Sname,XSB.Major,XSB.Tcredit
FROM XSB;
```

比较：查询XSB表中学生的姓名、专业和总学分，返回结果集的前6%行。

```
SELECT TOP(6)PERCENT XSB.Sname,XSB.Major,XSB.Tcredit
FROM XSB;
```

6、替换查询结果中的数据

EG：查询XSB表中学生的学号、姓名和总学分，对总学分按以下规则替换：若总学分为空值，则替换为“尚未选课”；若总学分小于50，则替换为“不合格”；若总学分在50与52之间，则替换为“合格”；若总学分大于52，则替换为“优秀”。列标题更改为“等级”。

```
SELECT XSB.Stu_ID,XSB.Sname,等级=
CASE
WHEN XSB.Tcredit IS NULL THEN '尚未选课'
WHEN XSB.Tcredit <50 THEN '不合格'
WHEN XSB.Tcredit >=50 and XSB.Tcredit <=52 THEN '合格'
ELSE '优秀'
END
FROM XSB;
```

7、计算列值

EG：按150分计算成绩并查询学生的成绩情况。

```
SELECT CJB.Stu_ID,CJB.C_ID,成绩=CJB.Grade*1.5
FROM CJB;
```

比较：

```
SELECT CJB.Stu_ID,CJB.C_ID,CJB.Grade=CJB.Grade*1.5
FROM CJB;
```

8、聚合函数

（1）SUM 和 AVG

SUM/AVG ([ALL | DISTINCT] expression)

SUM/AVG在计算时，忽略NULL值。

EG：求学号为081101的学生所学课程的平均成绩和总成绩。

```
SELECT AVG(CJB.Grade) AS 平均成绩,SUM(CJB.Grade) AS 总成绩
FROM CJB
WHERE CJB.Stu_ID='081101';
```

（2）MAX 和 MIN

MAX/MIN ([ALL | DISTINCT] expression)

MAX/MIN在计算时，忽略NULL值。

EG：求选修101课程的学生的最高分和最低分。

```
SELECT MAX(CJB.Grade) AS 最高分,MIN(CJB.Grade) AS 最低分
FROM CJB
WHERE CJB.C_ID='101';
```

（3）COUNT

COUNT{ ([ALL | DISTINCT] expression) | *}

COUNT在计算时，忽略NULL值；COUNT（*）返回总数目，包含空值。

EG：求学生的总数。

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

EG：统计备注不为空的学生数。

```
SELECT COUNT(XSB.Remark)  FROM XSB;
```

EG：求选修了课程的学生总数。

```
SELECT COUNT(DISTINCT CJB.Stu_ID)  FROM CJB;
```

WHERE子句语法格式

[WHERE <search_condition>] /*结果集中返回的行的条件*/

< search_condition > ::=

判定运算

{ [NOT] <predicate> | (<search_condition>) }

[{ AND | OR } [NOT] { <predicate> | (<search_condition>) }]

[,...n]

判定运算包括比较运算、模式匹配、范围比较、空值比较、CONTAIN谓词、FREETEXT谓词和子查询。

WHERE子句语法格式

<predicate> ::=

{ expression { = | < > | ! = | > | > = | ! > | < | < = | ! < } expression /*比较运算*/

| string_expression [NOT] LIKE string_expression [ESCAPE 'escape_character'] /*字符串模式匹配*/

| expression [NOT] BETWEEN expression AND expression /*指定范围*/

| expression IS [NOT] NULL /*是否空值判断*/

| CONTAINS ({ column | * } , '< contains_search_condition >') /*包含式查询*/

| FREETEXT ({ column | * } , 'freetext_string') /*自由式查询*/

搜索指定的单词、短语等

搜索指定的单词、短语等

| expression [NOT] IN (subquery | expression [,...n]) /*IN 子句*/

| expression { = | < > | ! = | > | > = | ! > | < | < = | ! < } { ALL | SOME | ANY} (subquery)

/*比较子查询*/

| EXISTS (subquery) } /*EXIST 子查询*/

1、表达式比较

{ expression { = | < > | ! = | > | > = | ! > | < | < = | ! < } expression　　/*比较运算*/

EG：查询PXSCJ数据库XSB表中学号为081101的同学的情况。

```
SELECT XSB.* FROM XSB WHERE XSB.Stu_ID='081101';
```

EG：查询XSB表中总学分大于50的同学的情况。

```
SELECT XSB.* FROM XSB WHERE XSB.Tcredit>50;
```

EG：查询XSB表中通信工程专业总学分大于等于52的同学的情况。

```
SELECT XSB.*
FROM XSB
WHERE XSB.Major='通信工程' AND XSB.Tcredit>=52;
```

2、模式匹配

string_expression [NOT] LIKE string_expression [ESCAPE 'escape_character']　/*字符串模式匹配*/

字符串和通配符

允许在字符串中搜索通配符

通配符列表		
通配符	说明	示例
%	包含零个或多个字符的任意字符串。	WHERE title LIKE '%computer%' 将查找在书名中任意位置包含单词 "computer" 的所有书名。
_（下划线）	任何单个字符。	WHERE au_fname LIKE '_ean' 将查找以 ean 结尾的所有 4 个字母的名字（Dean、Sean 等）。
[ ]	指定范围 ([a-f]) 或集合 ([abcdef]) 中的任何单个字符。	WHERE au_lname LIKE '[C-P]arsen' 将查找以 arsen 结尾并且以介于 C 与 P 之间的任何单个字符开始的作者姓氏，例如 Carsen、Larsen、Karsen 等。在范围搜索中，范围包含的字符可能因排序规则的排序规则而异。
[^]	不属于指定范围 ([a-f]) 或集合 ([abcdef]) 的任何单个字符。	WHERE au_lname LIKE 'de[^l]%' 将查找以 de 开始并且其后的字母不为 l 的所有作者的姓氏。

EG：查询PXSCJ数据库XSB表中姓“王”且单名的学生的情况。

```
SELECT XSB.* FROM XSB WHERE XSB.Sname LIKE '王_';
```

EG：查询XSB表中学号倒数第3个数字为1，且倒数第1个数在1~5之间的学生的学号、姓名和专业。

```
SELECT XSB.Stu_ID,XSB.Sname,XSB.Major
FROM XSB
WHERE XSB.Stu_ID LIKE '%1_[12345]';
```

EG：查询XSB表中名字包含%的学生学号和姓名。

```
SELECT XSB.Stu_ID,XSB.Sname
FROM XSB
WHERE XSB.Sname LIKE '%#%%' ESCAPE'#';
```

定义转义字符

比较：

EG：查询PXSCJ数据库XSB表中姓“王”且全名为3个汉字的学生的情况。

```
SELECT XSB.*
FROM XSB
WHERE XSB.Sname LIKE '王_ _';
```

结果 消息

	sno	sname	ssex	sage	sdept
1	20051507	王芳	女	19	IS
2	20051508	王民生	男	22	IS
3	20051509	王小明	男	19	IS

EG：查询PXSCJ数据库XSB表中姓“王”且全名为3个汉字的学生的情况。

```
SELECT XSB.*
FROM XSB
WHERE XSB.Sname LIKE '王
%' and len(Sname)=3;;
```

结果 消息

	sno	sname	ssex	sage	sdept
1	20051508	王民生	男	22	IS
2	20051509	王小明	男	19	IS

3、范围比较

| expression [NOT] BETWEEN expression AND expression　/*指定范围*/

expression [NOT] IN (subquery | expression [,...n])　/*IN 子句*/

EG：查询PXSCJ数据库XSB表中不在1989年出生的学生的情况。

```
SELECT XSB.* FROM XSB
WHERE XSB.Sdate NOT BETWEEN '1989-1-1' AND '1989-12-31';
```

EG：查询XSB表中专业为“计算机”、“通信工程”或“无线电”的学生情况。

```
SELECT XSB.* FROM XSB
WHERE XSB.Major IN('计算机','通信工程','无线电');
```

4、空值比较

| expression IS [NOT] NULL　　/*是否空值判断*/

不使用NOT时，若表达式expression的值为空值，则返回TRUE，否则返回FALSE；使用NOT时，结果相反。

EG：查询PXSCJ数据库XSB表中总学分尚不定的学生情况。

```
SELECT XSB.* FROM XSB
WHERE XSB.Tcredit IS NULL;
```

5、子查询

子查询除了可以用在SELECT语句中，也可以用在INSERT、UPDATE和DELETE语句中。通常与IN、EXIST谓词及比较运算符结合使用。

（1）IN子查询

| expression [NOT] IN (subquery) /*IN子句*/

用于进行一个给定值是否在子查询结果集中的判断。当表达式expression与子查询subquery的结果表中的某个值相等时，IN谓词返回TRUE，否则返回FALSE；使用了NOT，则相反。

EG：查询选修了课程号为206的课程的学生基本情况。

```
SELECT XSB.* FROM XSB
WHERE XSB.Stu_ID IN
(SELECT CJB.Stu_ID FROM CJB
WHERE CJB.C_ID='206');
```

系统先执行子查询，产生一个结果表，再执行查询。

EG：查询未选修离散数学的学生情况。

```
SELECT XSB.* FROM XSB
WHERE XSB.Stu_ID NOT IN
  (
    SELECT CJB.Stu_ID
    FROM CJB
    WHERE CJB.C_ID IN
      (
        SELECT KCB.C_ID
        FROM KCB
        WHERE KCB.Cname='离散数学'
      )
  );
```

（2）比较子查询

expression { = | < > | ! = | > | > = | ! > | < | < = | ! < } { ALL | SOME | ANY} (subquery)

/*比较子查询

*/

ALL：指定表达式要与子查询结果集中的每个值都进行比较，满足时返回TRUE，否则返回FALSE；

SOME或ANY：表达式只要与子查询结果集中的某个值满足比较的关系，就返回TRUE，否则返回FALSE。

EG：查询选修了离散数学的学生学号。

```
SELECT CJB.Stu_ID FROM CJB  WHERE CJB.C_ID IN
( SELECT KCB.C_ID   FROM KCB
WHERE KCB.Cname='离散数学' );
```

可否将IN改为“=” ？

EG：查找比所有计算机系的学生年龄都大的学生。

```
SELECT XSB.* FROM XSB
WHERE XSB.Sdate <ALL
(SELECT XSB.Sdate FROM XSB
WHERE XSB.Major='计算机');
```

比较：

```
SELECT XSB.* FROM XSB
 WHERE XSB.Sdate <
 (SELECT XSB.Sdate FROM XSB
WHERE XSB.Major='计算机');
```

EG：查找206号课程成绩不低于101号课程最低成绩的学生学号。

```
SELECT CJB.Stu_ID FROM CJB
WHERE CJB.C_ID='206' AND CJB.Grade !<ANY
(SELECT CJB.Grade FROM CJB
WHERE CJB.C_ID='101');
```

（3）EXISTS 子查询

[NOT] EXISTS (subquery)　　/*EXIST子查询*/

EXISTS谓词用于测试子查询的结果是否为空表，若结果集不为空，则返回TRUE，否则返回FALSE；与NOT结合使用，其返回值相反。

由 EXISTS 引入的子查询的选择列表通常几乎都是由星号 (*) 组成。由于只是测试是否存在符合子查询中指定条件的行，因此不必列出列名。

EG：查询选修了206号课程的学生姓名。

```
SELECT XSB.Sname FROM XSB WHERE EXISTS
(SELECT CJB.* FROM CJB
WHERE CJB.Stu_ID=XSB.Stu_ID AND CJB.C_ID='206');
```

处理过程：

① 查找外层查询中XSB表的第一行，根据学号值处理内层查询；

② 若结果不为空，则取出该行姓名值作为结果集的一行

③ 依次查找外层查询的2、3、4……行，直至XSB中所有行查找完为止。

EG：查询没有选修206号课程的学生姓名。

```
SELECT XSB.Sname FROM XSB WHERE NOT EXISTS
(SELECT CJB.* FROM CJB
WHERE CJB.Stu_ID=XSB.Stu_ID AND CJB.C_ID='206');
```

比较：

```
SELECT XSB.Sname FROM XSB WHERE XSB.Stu_ID NOT IN
(SELECT CJB.Stu_ID FROM CJB
 WHERE CJB.C_ID='206');
```

```
SELECT XSB.Sname FROM XSB WHERE XSB.Stu_ID <>
(SELECT CJB.Stu_ID FROM CJB
WHERE CJB.C_ID='206');
```

EG：查询选修了全部课程的学生姓名。(没有一门课程是他不选修的。)

```
SELECT XSB.Sname FROM XSB
WHERE NOT EXISTS
(
    SELECT * FROM KCB
    WHERE NOT EXISTS
     (
        SELECT * FROM CJB
        WHERE CJB.Stu_ID=XSB.Stu_ID AND CJB.C_ID=KCB.C_ID
     )
);
```

SELECT关键字后面也可以定义子查询。

EG：从XSB表中查找所有女生的姓名、学号及其与081101号学生的年龄差距。

```
SELECT XSB.Sname,XSB.Stu_ID ,YEAR(XSB.Sdate)-YEAR
(
 (SELECT XSB.Sdate
  FROM XSB
  WHERE XSB.Stu_ID='081101')
 ) AS 年龄差距
FROM XSB
WHERE XSB.Ssex=0;
```

ANY（或SOME）、ALL谓词与聚集函数、IN谓词的转换

	=	<>或!=	<	<=	>	>=
ANY	IN	----	< MAX	<= MAX	> MIN	>= MIN
ALL	----	NOT IN	< MIN	<= MIN	> MAX	>= MAX

聚集函数实现子查询通常比直接用ANY或者ALL查询效率高。

FROM子句语法格式

```
[ FROM { <table_source> } [ ,...n ] ]
<table_source> ::=
{
  table_or_view_name [ [ AS ] table_alias ]                    /*查询表或视图，可指定别名*/
        [ WITH ( < table_hint > [ [ , ]...n ] ) ]              /*指定查询优化器对此表和此语句使用优化或锁定策略*/
  | rowset_function [ [ AS ] table_alias ]                     /*指定其中一个行集函数（如 OPENROWSET）*/
        [ ( bulk_column_alias [ ,...n ] ) ]                    /*代替结果集内列名的可选别名*/
  | user_defined_function [ [ AS ] table_alias ] ]             /*指定表值函数*/
  | OPENXML <openxml_clause>                                   /*通过 XML 文档提供行集视图*/
  | derived_table [ AS ] table_alias [ ( column_alias [ ,...n ] ) ]   /*子查询*/
  | <joined_table>                                             /*连接表*/
  | <pivoted_table>                                            /*将行转换为列*/
  | <unpivoted_table>                                          /*将列转换为行*/
}
```

返回用户自定义函数

必须使用AS为子查询产生的中间表定义一个别名

执行子查询返回的表

1、table_or_view_name [[AS] table_alias]　　/*查询表或视图，可指定别名*/

指定SELECT语句要查询的表或视图，表和视图可以是一个或多个。

EG：从KCB表中查找101号课程的开课学期。

```
SELECT KCB.Term FROM KCB
WHERE KCB.C_ID='101';
```

EG：查找081101号学生计算机基础课的成绩。

```
SELECT CJB.Grade FROM CJB,KCB
WHERE CJB.Stu_ID='081101' AND KCB.Cname='计算机基础' AND
KCB.C_ID=CJB.C_ID;
```

EG：查询选修了学号为081102的同学所选修的全部课程的学生的学号。(081102学生选修的课程没有一门是他不选修的。）

```
SELECT DISTINCT CJ1.Stu_ID FROM CJB AS CJ1
WHERE NOT EXISTS
(
    SELECT * FROM CJB AS CJ2
    WHERE CJ2.Stu_ID='081102' AND NOT EXISTS
        (
            SELECT * FROM CJB AS CJ3
            WHERE CJ3.Stu_ID=CJ1.Stu_ID AND CJ3.C_ID=CJ2.C_ID
        )
);
```

2、derived_table [AS] table_alias [(column_alias [,...n])]　　/*子查询*/

子查询可以在FROM子句中使用，derived_table表示执行子查询返回的表，必须使用AS关键字为子查询产生的表定义一个别名。

EG：从XSB表中查找总学分大于50的男生的姓名和学号。

```
SELECT STUDENT.Stu_ID,STUDENT.Sname FROM
(SELECT XSB.* FROM XSB WHERE XSB.Tcredit>50)AS STUDENT
WHERE STUDENT.Ssex=1;
```

```
SELECT XSB.Stu_ID,XSB.Sname
FROM XSB
WHERE XSB.Tcredit>50 AND XSB.Ssex=1;
```

```
SELECT XSB.Stu_ID, XSB T.Sname FROM
(SELECT XSB.* FROM XSB WHERE XSB.Tcredit>50)AS STUDENT
WHERE XSB.Ssex=1;
```

EG：在XSB表中查找1990年1月1日以前出生的学生的姓名和专业，分别使用name和speciality表示。

```
SELECT m.name,m.speciality
FROM(SELECT XSB.* FROM XSB WHERE XSB.Sdate<'19900101')
AS m(num,name,sex,birthday,speciality,score,mem);
```

```
SELECT m.Sname name,m.Major speciality
FROM(SELECT XSB.* FROM XSB WHERE XSB.Sdate<'19900101')
AS m;
```

```
SELECT m.Sname AS name,m.Major AS speciality
FROM(SELECT XSB.* FROM XSB WHERE XSB.Sdate<'19900101')
AS m;
```

```
SELECT m.name,m.speciality
FROM(SELECT XSB.* FROM XSB WHERE XSB.Sdate<'19900101')
AS m(name, speciality);
```

连接查询（涉及多个表的查询）

在T-SQL中，连接查询有两大表示形式：

- 符合SQL标准的连接谓词表示形式；
- T-SQL扩展的，使用JOIN关键字的表示形式。

1、连接谓词

在SELECT语句的WHERE子句中使用比较运算符给出连接条件对表进行连接。

EG：查找PXSCJ数据库每个学生的基本情况以及选修的课程情况。

```
SELECT XSB.*,CJB.* FROM XSB,CJB
WHERE XSB.Stu_ID=CJB.Stu_ID;
```

比较：

```
SELECT XSB.*,CJB.C_ID,CJB.Grade FROM XSB,CJB
WHERE XSB.Stu_ID=CJB.Stu_ID;
```

EG：查找选修了206号课程且成绩在80分以上的学生姓名及成绩。

```
SELECT XSB.Sname,CJB.Grade FROM XSB,CJB
WHERE XSB.Stu_ID=CJB.Stu_ID AND CJB.C_ID='206' AND CJB.Grade>=80;
```

EG：查找选修了“计算机基础”课程且成绩在80分以上的学生学号、姓名、课程名及成绩。

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.Cname,CJB.Grade
FROM XSB,KCB,CJB
WHERE XSB.Stu_ID=CJB.Stu_ID AND KCB.C_ID=CJB.C_ID AND
KCB.Cname='计算机基础' AND CJB.Grade>=80;
```

连接查询和子查询的区别：

连接查询可以合并两个或多个表中的数据；

带子查询的SELECT语句的结果只能来自一个表。

2、以JOIN关键字指定的连接

<joined_table> ::=

{ 要连接的表 连接类型 连接条件

<table_source> <join_type> <table_source> ON <search_condition>

| <table_source> CROSS JOIN <table_source> 交叉连接

}

<join_type> ::=

[{ INNER | { { LEFT | RIGHT | FULL } [OUTER] } } [<join_hint>]]

JOIN

以JOIN关键字指定的连接有三种类型：内连接、外连接、交叉连接。

（1）内连接（INNER JOIN）

EG：查找PXSCJ数据库每个学生的基本情况以及选修的课程情况。

```
SELECT *
FROM XSB INNER JOIN CJB ON XSB.Stu_ID=CJB.Stu_ID;
```

```
SELECT *
FROM XSB JOIN CJB ON XSB.Stu_ID=CJB.Stu_ID;
```

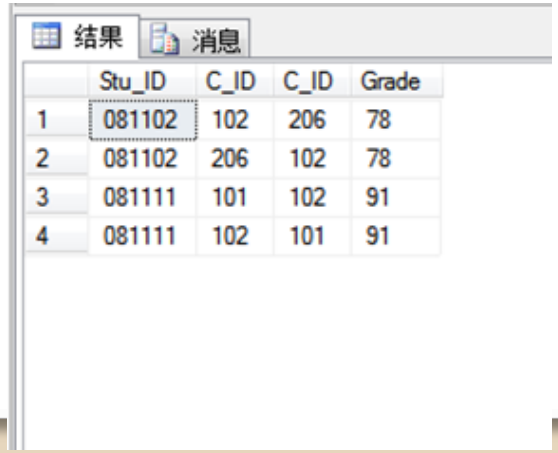
执行结果将包含XSB表和CJB表的所有字段（不去除重复字段）。

EG：查询选修了206号课程且成绩在80分以上的学生姓名和成绩。

```
SELECT XSB.Sname,CJB.Grade
FROM XSB JOIN CJB ON XSB.Stu_ID=CJB.Stu_ID
WHERE CJB.C_ID='206' AND CJB.Grade>=80;
```

EG：查找不同课程成绩相同的学生的学号、课程号和成绩。（自连接）

```
SELECT a.Stu_ID,a.C_ID,b.C_ID,a.Grade
FROM CJB a JOIN CJB b ON a.Grade=b.Grade AND a.Stu_ID=b.Stu_ID AND
a.C_ID!=b.C_ID;
```



	Stu_ID	C_ID	C_ID	Grade
1	081102	102	206	78
2	081102	206	102	78
3	081111	101	102	91
4	081111	102	101	91

EG：查找选修了“计算机基础”课程且成绩在80分以上的学生学号、姓名、课程名和成绩。

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.Cname,CJB.Grade
FROM XSB JOIN CJB JOIN KCB ON
CJB.C_ID=KCB.C_ID ON XSB.Stu_ID=CJB.Stu_ID
WHERE KCB.Cname='计算机基础' AND CJB.Grade>=80;
```

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.Cname,CJB.Grade
FROM XSB JOIN CJB ON XSB.Stu_ID=CJB.Stu_ID JOIN KCB ON
CJB.C_ID=KCB.C_ID
WHERE KCB.Cname='计算机基础' AND CJB.Grade>=80;
```

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.Cname,CJB.Grade
FROM XSB JOIN CJB JOIN KCB ON
 XSB.Stu_ID=CJB.Stu_ID ON CJB.C_ID=KCB.C_ID
WHERE KCB.Cname='计算机基础' AND CJB.Grade>=80;
```

（2）外连接（OUTER JOIN）

外连接的结果表不但包含满足连接条件的行，还包括相应表中的所有行。

左外连接（LEFT OUTER JOIN）：结果表中除了包括满足连接条件的行外，还包括左表的所有行；

右外连接（RIGHT OUTER JOIN）：结果表中除了包括满足连接条件的行外，还包括右表的所有行；

完全外连接（FULL OUTER JOIN）：结果表中除了包括满足连接条件的行外，还包括两个表的所有行；

其中OUTER关键字可以省略。

EG：查找所有学生情况，以及他们选修的课程号，若学生未选修任何课，也要包括其情况。

```
SELECT XSB.*,CJB.C_ID
FROM XSB LEFT JOIN CJB ON XSB.Stu_ID=CJB.Stu_ID;
```

```
SELECT XSB.*,CJB.C_ID
FROM  CJB RIGHT JOIN XSB ON XSB.Stu_ID=CJB.Stu_ID;
```

EG：查找被选修了的课程的选修情况和所有开设的课程名。

```
SELECT CJB.*,KCB.Cname
FROM CJB RIGHT JOIN KCB ON CJB.C_ID=KCB.C_ID;
```

```
SELECT CJB.*,KCB.Cname
FROM KCB LEFT JOIN CJB ON CJB.C_ID=KCB.C_ID;
```

（3）交叉连接（CROSS JOIN）

将两个表进行笛卡尔积运算，结果表由第一个表的每一行与第二个表的每一行拼接后形成的表。

EG：列出学生所有可能的选课情况。

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.C_ID,KCB.Cname
FROM XSB CROSS JOIN KCB;
```

EG：列出软件工程学生所有可能的选课情况。

```
SELECT XSB.Stu_ID,XSB.Sname,KCB.C_ID,KCB.Cname
FROM XSB CROSS JOIN KCB
WHERE XSB.Major='软件工程';
```

GROUP BY子句语法格式

[GROUP BY group_by_expression] /*GROUP BY 子句，指定分组表达式*/

GROUP BY 子句具有符合 ISO 的语法和不符合 ISO 的语法。在一条 SELECT 语句中只能使用一种语法样式。对于所有的新工作，请使用符合 ISO 的语法。提供不符合 ISO 的语法的目的是为了实现向后兼容。

1、非ISO标准的GOURP BY子句

[GROUP BY [ALL] group_by_expression [,...n]

[WITH { CUBE | ROLLUP }]]

分组表达式

指定CUBE、ROLLUP操作

指定结果集内不仅包含由 GROUP BY 提供的行，同时还包含汇总行。

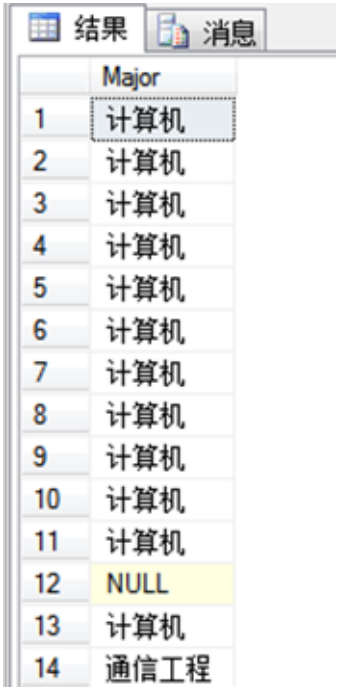
EG：输出PXSCJ数据库中各专业名。

```
SELECT XSB.Major FROM XSB
GROUP BY XSB.Major;
```

比较：

```
SELECT XSB.Major FROM XSB;
```

```
SELECT DISTINC XSB.Major FROM XSB;
```



<select> 列表中任何非聚合表达式中的每个表列或视图列都必须包括在 GROUP BY 列表中；

GROUP BY 子句中的表达式可以包含 FROM 子句中表、派生表或视图的列。这些列不必显示在 SELECT 子句 <select> 列表中。

EG：求各专业的学生数。

```
SELECT XSB.Major,COUNT(*) AS 学生人数
FROM XSB
GROUP BY XSB.Major;
```

EG：求被选修的各门课程的平均成绩和选修该课程的人数。

```
SELECT CJB.C_ID,AVG(CJB.Grade) AS 平均成绩,COUNT(CJB.Stu_ID) AS 选修人数
FROM CJB
GROUP BY CJB.C_ID;
```

EG：求每个专业的总人数、男生、女生人数及所有专业学生总人数。

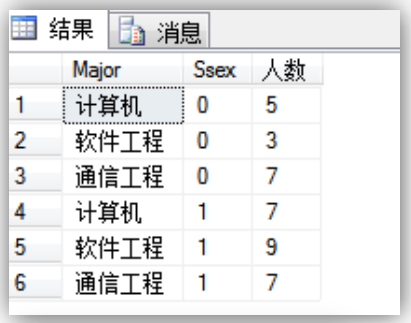
```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY XSB.Major,XSB.Ssex WITH ROLLUP;
```


结果 消息

	Major	Ssex	人数
1	计算机	0	5
2	计算机	1	7
3	计算机	NULL	12
4	软件工程	0	3
5	软件工程	1	9
6	软件工程	NULL	12
7	通信工程	0	7
8	通信工程	1	7
9	通信工程	NULL	14
10	NULL	NULL	38

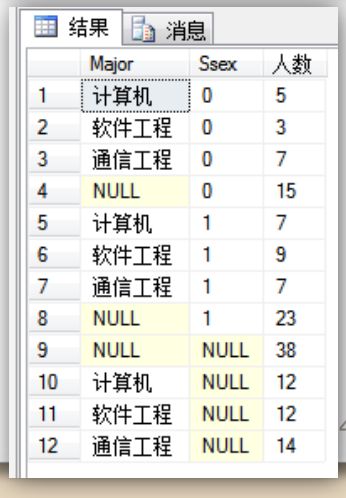
比较：

```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY XSB.Major,XSB.Ssex;
```


结果 消息

	Major	Ssex	人数
1	计算机	0	5
2	软件工程	0	3
3	通信工程	0	7
4	计算机	1	7
5	软件工程	1	9
6	通信工程	1	7

```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY XSB.Major,XSB.Ssex WITH CUBE;
```


结果 消息

	Major	Ssex	人数
1	计算机	0	5
2	软件工程	0	3
3	通信工程	0	7
4	NULL	0	15
5	计算机	1	7
6	软件工程	1	9
7	通信工程	1	7
8	NULL	1	23
9	NULL	NULL	38
10	计算机	NULL	12
11	软件工程	NULL	12
12	通信工程	NULL	14

2、ISO标准的GOURP BY子句

```
GROUP BY
{
      <column_expression>
      | ROLLUP ( < composite element list > )
      | CUBE ( < composite element list > )
      | GROUPING SETS ( < grouping set item list >)
}
```

分组的字段名表达式

生成简单的 GROUP BY 聚合行以及小计行或超聚合行，还生成一个总计行。

生成简单的 GROUP BY 聚合行、ROLLUP 超聚合行和交叉表格行。

在一个查询中指定数据的多个分组。

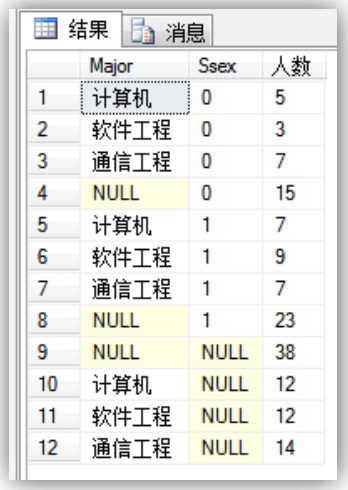
EG：求每个专业的总人数、男生、女生人数及所有专业学生总人数。

```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY  ROLLUP(XSB.Major,XSB.Ssex);
```

	Major	Ssex	人数
1	计算机	0	5
2	计算机	1	7
3	计算机	NULL	12
4	软件工程	0	3
5	软件工程	1	9
6	软件工程	NULL	12
7	通信工程	0	7
8	通信工程	1	7
9	通信工程	NULL	14
10	NULL	NULL	38

比较：

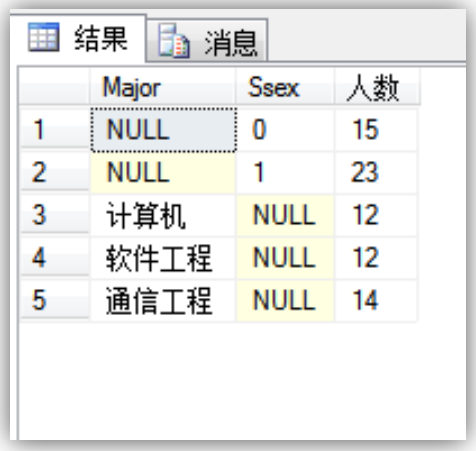
```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY  CUBE(XSB.Major,XSB.Ssex);
```



	Major	Sex	人数
1	计算机	0	5
2	软件工程	0	3
3	通信工程	0	7
4	NULL	0	15
5	计算机	1	7
6	软件工程	1	9
7	通信工程	1	7
8	NULL	1	23
9	NULL	NULL	38
10	计算机	NULL	12
11	软件工程	NULL	12
12	通信工程	NULL	14

EG：根据专业和性别对人数进行聚合。

```
SELECT XSB.Major,XSB.Ssex,COUNT(*) AS 人数
FROM XSB
GROUP BY GROUPING SETS(XSB.Major,XSB.Ssex);
```



	Major	Ssex	人数
1	NULL	0	15
2	NULL	1	23
3	计算机	NULL	12
4	软件工程	NULL	12
5	通信工程	NULL	14

HAVING子句语法格式

[HAVING <search condition>]

使用HAVING子句来指定组或聚合的搜索条件。

HAVING 只能与 SELECT 语句一起使用。

HAVING 通常在 GROUP BY 子句中使用。如果不使用 GROUP BY 子句，则 HAVING 的行为与 WHERE 子句一样。

search condition 与WHERE子句的查询条件类似，但HAVING子句中可以使用聚合函数，而WHERE子句中不可以。

EG：查找平均成绩在85分以上的学生的学号和平均成绩。

```
SELECT CJB.Stu_ID,AVG(CJB.Grade) AS 平均成绩
FROM CJB
GROUP BY CJB.Stu_ID
HAVING AVG(CJB.Grade)>=85;
```

EG：查找选修课程超过2门且成绩都在80分以上的学生的学号。

```
SELECT CJB.Stu_ID FROM CJB
WHERE CJB.Grade>=80
GROUP BY CJB.Stu_ID
HAVING COUNT(*)>2;
```

执行顺序：

WHERE用于筛选由FROM子句指定的数据对象，GROUP BY对WHERE的结果进行分组，HAVING对分组数据进行过滤。

EG：查找通信工程专业平均成绩在85分以上的学生的学号和平均成绩。

```
SELECT CJB.Stu_ID,AVG(CJB.Grade) AS 平均成绩
FROM CJB,XSB
WHERE XSB.Major='通信工程' AND XSB.Stu_ID=CJB.Stu_ID
GROUP BY CJB.Stu_ID
HAVING AVG(CJB.Grade)>=85;
```

```
SELECT CJB.Stu_ID,AVG(CJB.Grade) AS 平均成绩
FROM CJB,XSB
GROUP BY CJB.Stu_ID
HAVING XSB.Major='通信工程' AND
XSB.Stu_ID=CJB.Stu_ID AND AVG(CJB.Grade)>=85;
```

如果使用了GROUP BY子句，则HAVING语句中的列要包含在GROUP BY子句或聚集函数中。

ORDER BY子句语法格式

```
[ ORDER BY
      {
      order_by_expression              /*指定要排序的列*/
   [ COLLATE collation_name ]
                                       /*指定 collation_name  中的排序规则*/
   [ ASC | DESC ]
      } [ ,...n ]
]
```

EG：将通信工程专业的学生按出生时间先后顺序排序。

```
SELECT * FROM XSB
WHERE XSB.Major='通信工程'
ORDER BY XSB.Sdate;
```

EG：将计算机专业学生的“计算机基础”课程成绩按降序排列。

```
SELECT XSB.Sname,KCB.Cname,CJB.Grade FROM XSB,CJB,KCB
WHERE XSB.Major='计算机' AND KCB.Cname='计算机基础' AND
XSB.Stu_ID=CJB.Stu_ID AND KCB.C_ID=CJB.C_ID
ORDER BY CJB.Grade;
```

EG：查找通信工程专业学生的学号、姓名、出生时间，并产生一个学生总人数行。

```
SELECT XSB.Stu_ID,XSB.Sname,XSB.Sdate
FROM XSB
WHERE XSB.Major='通信工程'
COMPUTE COUNT(XSB.Stu_ID);
```

	Stu_ID	Sname	Sdate
1	081201	王敏	1990-12-31
2	081202	王林	1990-01-13
3	081203	王玉民	1989-07-01
4	081204	马琳琳	1989-10-02
5	081206	李继	1990-06-06
6	081208	李洪庆	1990-08-24
7	081210	孙祥新	1991-03-30
8	081211	孙岩	1989-02-25
9	081216	吴薇华	1989-03-30
10	081218	刘艳敏	1989-12-05

	cnt
1	14

ORDER BY 子句可以与COMPUTE BY 子句一起使用，在对结果排序的同时还产生附加的汇总行；

如果使用 COMPUTE BY，则还必须使用 ORDER BY 子句。

表达式必须与在 ORDER BY 后列出的子句相同或是其子集，并且顺序必须相同。

```
[ COMPUTE
    { { AVG | COUNT | MAX | MIN | STDEV | STDEVP | VAR | VARP
| SUM }
    ( expression ) } [ ,...n ]
    [ BY expression [ ,...n ] ]
]
```

expression 是关联 ORDER BY 子句中 order_by_expression 的相同副本。

EG：将学生按专业排序，并汇总各专业人数和平均学分。

```
SELECT XSB.* FROM XSB
ORDER BY XSB.Major
COMPUTE COUNT(XSB.STU_ID),AVG(XSB.TCREDIT) BY XSB.MAJOR;
```

结果 消息

	Stu_ID	Sname	Ssex	Sdate	Major	Tcredit	Remark
1	081101	王林	1	1990-02-10	计算机	60	NULL
2	081102	程明	1	1991-02-01	计算机	62	提前修完《汇编语言》，并获得学分
3	081103	王燕	0	1989-10-06	计算机	60	NULL
4	081104	魏艳萍	0	1990-08-26	计算机	60	NULL
5	081106	李芳芳	0	1990-11-22	计算机	60	NULL
6	081107	李明	1	1989-05-24	计算机	58	有一门课程不及格
7	081108	林一凡	1	1991-04-21	计算机	60	NULL
8	081109	张强民	1	1990-09-01	计算机	60	NULL

	cnt	avg
1	12	54

	Stu_ID	Sname	Ssex	Sdate	Major	Tcredit	Remark
1	081261	罗琳琳	0	1989-09-05	软件工程	50	提前修完学分
2	091101	王海	1	1991-05-10	软件工程	50	NULL
3	091102	李娜	0	1991-04-12	软件工程	52	NULL
4	091103	王海	1	1991-05-10	软件工程	50	NULL
5	091104	王海	1	1991-05-10	软件工程	50	NULL
6	091105	王海	1	1991-05-10	软件工程	50	NULL
7	091106	王海	1	1991-05-10	软件工程	50	NULL
8	091107	王海	1	1991-05-10	软件工程	50	NULL

	cnt	avg
1	12	50

SELECT语句的其他语法

1、INTO

[INTO new_table]

包含INTO 子句的SELECT语句执行后所创建的表的结构由SELECT所选择的列决定，新创建的表中的记录由SELECT的查询结果决定。

若SELECT查询结果为空，则创建一个只有结构没有记录的表。

EG：由XSB表创建“计算机系学生”表，包括学号和姓名。

```
SELECT XSB.Stu_ID,XSB.Sname
INTO 计算机系学生
FROM XSB
WHERE XSB.Major='计算机';
```

2、UNION（联合查询）

SELECT查询语句

SELECT查询语句

```
{ <query_specification> | ( <query_expression> ) }
    UNION [ ALL ] <query_specification | ( <query_expression> )
[ UNION [ ALL ] <query_specification> | ( <query_expression> ) [ ...n ] ]
```

结果中包括所有行，
不去除重复行。

使用UNION子句可以将两个或多个SELECT查询的结果合并成一个结果集。

（1）所有查询中的列数和列的顺序必须相同；

（2）数据类型必须兼容；

（3）若不指定INTO子句，结果将合并到第一个表中。

EG：查找学号为081101和学号为081210的两位同学的信息。

```
SELECT XSB.* FROM XSB WHERE XSB.Stu_ID='081101'
UNION ALL
SELECT XSB.* FROM XSB WHERE XSB.Stu_ID='081210';
```

结果 消息

	Stu_ID	Sname	Ssex	Sdate	Major	Tcredit	Remark
1	081101	王林	1	1990-02-10	计算机	60	NULL
2	081210	孙祥新	1	1991-03-30	通信工程	54	提前休完一门课程

3、EXCEPT 和 INTERSECT

{ <query_specification> | (<query_expression>) }

{ EXCEPT | INTERSECT }

{ <query_specification> | (<query_expression>) }

EXCEPT 和 INTERSECT用于比较两个查询的结果，返回非重复值。

（1）所有查询中的列数和列的顺序必须相同；

（2）数据类型必须兼容；

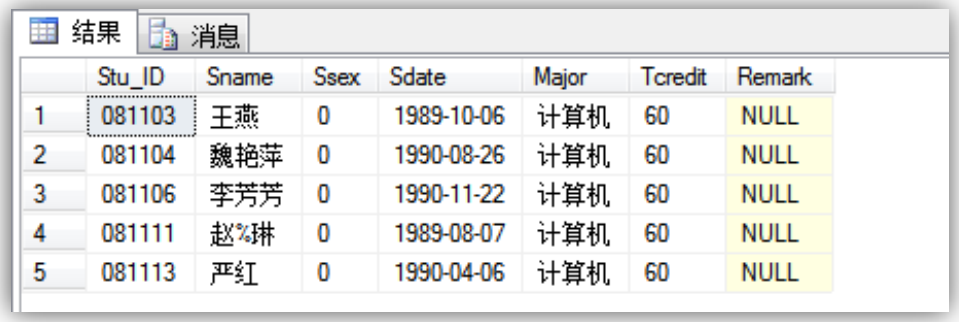
EXCEPT：从关键字左边的查询中返回右边查询没有找到的所有非重复值。

INTERSECT：关键字左右两边的两个查询都返回的所有非重复值。

EXCEPT 和 INTERSECT返回的结果集列名与关键字左侧的查询返回的列名相同。

EG：查找专业为计算机但性别不为男的学生信息。

```
SELECT XSB.* FROM XSB WHERE XSB.Major='计算机'
EXCEPT
SELECT XSB.* FROM XSB WHERE XSB.Ssex=1;
```



结果 消息

	Stu_ID	Sname	Ssex	Sdate	Major	Tcredit	Remark
1	081103	王燕	0	1989-10-06	计算机	60	NULL
2	081104	魏艳萍	0	1990-08-26	计算机	60	NULL
3	081106	李芳芳	0	1990-11-22	计算机	60	NULL
4	081111	赵%琳	0	1989-08-07	计算机	60	NULL
5	081113	严红	0	1990-04-06	计算机	60	NULL

EG：查找总学分大于42且性别为男的学生信息。

```
SELECT XSB.* FROM XSB WHERE XSB.Tcredit>42
INTERSECT
SELECT XSB.* FROM XSB WHERE XSB.Ssex=1;
```