第5章 数据查询

本章要点：

- 投影查询
- 选择查询
- 分组查询和统计计算
- 排序查询
- 连接查询
- 子查询
- SELECT 查询的其他子句

对于数据库应用，最重要的操作是查询，即从数据库的一个或多个表中查询出所要求的数据信息。查询语言用来对已经存在于数据库中的数据按照特定的行、列、条件表达式或者一定次序进行检索。本章介绍投影查询、选择查询、分组查询和统计计算、排序查询、连接查询、子查询、SELECT 查询的其他子句等内容。

T-SQL 对数据库的查询使用 SELECT 语句，SELECT 语句具有灵活的使用方式和强大的功能，其选项很丰富、查询条件和嵌套使用比较复杂。

语法格式：

```
SELECT select_list                                  /* 指定要选择的列 */
 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 */
```

5.1 投影查询

投影查询通过 SELECT 语句的 SELECT 子句来表示，由选择表中的部分或全部列组成结果表。

语法格式：

```
SELECT [ ALL | DISTINCT ] [ TOP n [ PERCENT ] [ WITH TIES ] ] < select_list >
```

select_list 指出了结果的形式，其格式为：

```
{   *                                          /* 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 */
  | { colume_name | expression | $ IDENTITY | $ ROWGUID }
      /* 选择指定的列并更改列标题,为列指定别名,还可用于为表达式结果指定名称 */
      [ [ AS ] column_alias ]
  | column_alias = expression
} [ , ... n ]
```

1. 投影指定的列

使用 SELECT 语句可选择表中的一个列或多个列,如果是多个列,各列名中间要用逗号分开。

语法格式:

```
SELECT column_name [ , column_name...]
 FROM table_name
 WHERE search_condition
```

其中,FROM 子句用于指定表,WHERE 在该表中检索符合 search_condition 条件的列。

【例 5.1】 查询 student 表中所有学生的学号、姓名和专业。

```
USE stsco
SELECT stid, stname, speciality
FROM student
```

查询结果:

```
stid          stname          speciality
------------ -------------- -------------
201001        罗俊杰          通信
201002        韩红丽          通信
201004        冯露            通信
202001        徐桥            计算机
202002        袁志敏          计算机
202005        董莎            计算机
```

2. 投影全部列

在 SELECT 子句指定列的位置上使用 * 号时,则为查询表中所有列。

【例 5.2】 查询 student 表中所有列。

```
USE stsco
SELECT *
FROM student
```

该语句与下面语句等价:

```
USE stsco
SELECT stid, stname, stsex, stbirthday, speciality, tc
FROM student
```

查询结果:

stid	stname	stsex	stbirthday	speciality	tc
201001	罗俊杰	男	2000-06-15	通信	52
201002	韩红丽	女	1999-09-23	通信	49
201004	冯露	女	1999-08-07	通信	50
202001	徐桥	男	2000-02-25	计算机	52
202002	袁志敏	男	1999-12-04	计算机	48
202005	董莎	女	2000-04-19	计算机	50

3. 修改查询结果的列标题

为了改变查询结果中显示的列标题,可以在列名后使用 AS 子句。

```
AS column_alias
```

其中,column_alias 是指定显示的列标题,AS 可省略。

【例 5.3】 查询 student 表中通信专业学生的 stid、stname、tc,并将结果中各列的标题分别修改为学号、姓名、总学分。

```
USE stsco
SELECT stid AS '学号', stname AS '姓名', tc AS '总学分'
FROM student
```

查询结果:

学号	姓名	总学分
201001	罗俊杰	52
201002	韩红丽	49
201004	冯露	50
202001	徐桥	52
202002	袁志敏	48
202005	董莎	50

4. 去掉重复行

去掉结果集中的重复行可使用 DISTINCT 关键字,其语法格式如下:

```
SELECT DISTINCT column_name [ , column_name...]
```

【例 5.4】 查询 student 表中 speciality 列,消除结果中的重复行。

```
USE stsco
SELECT DISTINCT speciality
FROM student
```

查询结果:

speciality
计算机
通信

5.2 选 择 查 询

选择查询通过 WHERE 子句实现，WHERE 子句给出查询条件，该子句必须紧跟 FROM 子句之后。

语法格式：

```
WHERE < search_condition >
```

其中，search_condition 为查询条件，< search_condition >语法格式为：

```
{ [ NOT ] < predicate > | (< search_condition > ) }
    [ { AND | OR } [ NOT ] { < predicate > | (< search_condition >) } ]
} [ ,...n ]
```

其中，predicate 为判定运算，< predicate >语法格式为：

```
{ 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 */
  | expression [ NOT ] IN ( subquery | expression [,...n] )             /* IN 子句 */
  | expression { = | < | <= | > | >= | <> | != | !< | !> } { ALL | SOME | ANY } ( subquery )
                                                                        /* 比较子查询 */
  | EXIST ( subquery )                                                  /* EXIST 子查询 */
}
```

现将 WHERE 子句的常用查询条件列于表 5.1 中，以使读者更清楚地了解查询条件。

表 5.1 查询条件

查询条件	谓　词
比较	<=，<，=，>=，>，!=，<>，!>，!<
指定范围	BETWEEN AND，NOT BETWEEN AND，IN
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

说明：在 SQL 中，返回逻辑值的运算符或关键字都称为谓词。

1. 表达式比较

比较运算符用于比较两个表达式值，比较运算的语法格式如下：

```
expression { = | < | <= | > | >= | <> | != | !< | !> } expression
```

其中，expression 是除 text、ntext 和 image 之外类型的表达式。

【例 5.5】 查询 student 表中专业为计算机或性别为女的学生。

```
USE stsco
SELECT *
FROM student
WHERE speciality = '计算机' OR stsex = '女'
```

查询结果:

stid	stname	stsex	stbirthday	speciality	tc
201002	韩红丽	女	1999-09-23	通信	49
201004	冯露	女	1999-08-07	通信	50
202001	徐桥	男	2000-02-25	计算机	52
202002	袁志敏	男	1999-12-04	计算机	48
202005	董莎	女	2000-04-19	计算机	50

2. 范围比较

BETWEEN、NOT BETWEEN、IN 是用于范围比较的三个关键字,用于查找字段值在(或不在)指定范围的行。

【例 5.6】 查询 score 表成绩为 86、94、95 的记录。

```
USE stsco
SELECT *
FROM score
WHERE grade IN (86,94,95)
```

查询结果:

stid	cid	grade
201001	205	94
201001	801	95
201004	801	86
202001	801	94

3. 模式匹配

字符串模式匹配使用 LIKE 谓词,LIKE 谓词表达式的语法格式如下:

```
string_expression [ NOT ] LIKE string_expression [ ESCAPE 'escape_character']
```

其含义是查找指定列值与匹配串相匹配的行,匹配串(即 string_expression)可以是一个完整的字符串,也可以含有通配符。通配符有以下两种。

%:代表 0 或多个字符。

_:代表一个字符。

LIKE 匹配中使用通配符的查询也称模糊查询。

【例 5.7】 查询 student 表中姓董的学生情况。

```
USE stsco
SELECT *
FROM student
```

```
WHERE stname LIKE '董%'
```

查询结果：

```
stid        stname      stsex       stbirthday        speciality    tc
--------    --------    --------    ------------    ----------    --------
202005      董莎        女          2000-04-19      计算机        50
```

4. 空值使用

空值是未知的值，判定一个表达式的值是否为空值时，使用 IS NULL 关键字，语法格式如下：

```
expression IS [ NOT ] NULL
```

【例 5.8】 查询已选课但未参加考试的学生情况。

```
USE stsco
SELECT *
FROM score
WHERE grade IS NULL
```

查询结果：

```
stid          cid       grade
-----------   -------   -----------
202002        801       NULL
```

5.3 分组查询和统计计算

检索数据常常需要进行分组查询和统计计算，本节介绍使用聚合函数、GROUP BY 子句、HAVING 子句进行分组查询和统计计算的方法。

1. 聚合函数

聚合函数实现数据统计或计算，用于计算表中的数据，返回单个计算结果。除 COUNT 函数外，聚合函数忽略空值。

SQL Server 所提供的常用聚合函数如表 5.2 所示。

表 5.2 聚合函数

函 数 名	功 能
AVG	求组中数值的平均值
COUNT	求组中项数
MAX	求最大值
MIN	求最小值
SUM	返回表达式中数值总和
STDEV	返回给定表达式中所有数值的统计标准偏差
STDEVP	返回给定表达式中所有数值的填充统计标准偏差
VAR	返回给定表达式中所有数值的统计方差
VARP	返回给定表达式中所有数值的填充的统计方差

聚合函数一般参数语法格式如下：

```
( [ ALL | DISTINCT ] expression )
```


其中，ALL 表示对所有值进行聚合函数运算，ALL 为默认值，DISTINCT 表示去除重复值，expression 指定进行聚合函数运算的表达式。

【例 5.9】 查询 102 课程的最高分、最低分、平均成绩。

```
USE stsco
SELECT MAX(grade) AS '最高分',MIN(grade) AS '最低分',AVG(grade) AS '平均成绩'
FROM score
WHERE cid = '102'
```

该语句采用 MAX 求最高分、MIN 求最低分、AVG 求平均成绩。

查询结果：

```
最高分                  最低分                  平均成绩
-------------------- -------------------- -----------
93                   71                   84
```

【例 5.10】 求学生的总人数。

```
USE stsco
SELECT COUNT( * ) AS '总人数'
FROM student
```

该语句采用 COUNT(*)计算总行数，总人数与总行数一致。

查询结果：

```
总人数
-----------
6
```

【例 5.11】 查询计算机专业学生的总人数。

```
USE stsco
SELECT COUNT( * ) AS '总人数'
FROM student
WHERE speciality = '计算机'
```

该语句采用 COUNT(*)计算总人数，并用 WHERE 子句指定的条件限定为计算机专业。

查询结果：

```
总人数
-----------
3
```

2. GROUP BY 子句

GROUP BY 子句用于将查询结果表按某一列或多列值进行分组，其语法格式如下：

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

```
[ WITH { CUBE | ROLLUP } ] ]
```

其中，group_by_expression 为分组表达式，通常包含字段名，ALL 显示所有分组，WITH 指定 CUBE 或 ROLLUP 操作符，在查询结果中增加汇总记录。

注意：聚合函数常与 GROUP BY 子句一起使用。

【例 5.12】 查询各门课程的最高分、最低分、平均成绩。

```
USE stsco
SELECT cid AS '课程号', MAX(grade)AS '最高分',MIN (grade)AS '最低分', AVG(grade)AS '平均成绩'
FROM score
WHERE NOT grade IS NULL
GROUP BY cid
```

该语句采用 MAX、MIN、AVG 等聚合函数，并用 GROUP BY 子句对 cid(课程号)进行分组。

查询结果：

课程号	最高分	最低分	平均成绩
102	93	71	84
203	92	70	82
205	94	68	83
801	95	76	88

提示：如果 SELECT 子句的列名表包含聚合函数，则该列名表只能包含聚合函数指定的列名和 GROUP BY 子句指定的列名。

【例 5.13】 求选修各门课程的平均成绩和选修人数。

```
USE stsco
SELECT cid AS '课程号', AVG(grade) AS '平均成绩', COUNT( * ) AS '选修人数'
FROM score
GROUP BY cid
```

该语句采用 AVG、COUNT 等聚合函数，并用 GROUP BY 子句对 cid(课程号)进行分组。

查询结果：

课程号	平均成绩	选修人数
102	84	3
203	82	3
205	83	3
801	88	6

3. HAVING 子句

HAVING 子句用于对分组按指定条件进一步进行筛选，最后只输出满足指定条件的分组，HAVING 子句的语法格式如下：

```
[ HAVING < search_condition > ]
```

其中,search_condition 为查询条件,可以使用聚合函数。

当 WHERE 子句、GROUP BY 子句、HAVING 子句在一个 SELECT 语句中时,执行顺序如下。

(1) 执行 WHERE 子句,在表中选择行。

(2) 执行 GROUP BY 子句,对选取行进行分组。

(3) 执行聚合函数。

(4) 执行 HAVING 子句,筛选满足条件的分组。

【例 5.14】 查询选修课程 2 门以上且成绩在 80 分以上的学生的学号。

```
USE stsco
SELECT stid AS '学号', COUNT(cid) AS '选修课程数'
FROM score
WHERE grade >= 80
GROUP BY stid
HAVING COUNT( * )>= 2
```

该语句采用 COUNT 聚合函数、WHERE 子句、GROUP BY 子句、HAVING 子句。

查询结果:

```
学号          选修课程数
------------  -----------
201001        3
201004        3
202001        2
202005        2
```

【例 5.15】 查询至少有 4 名学生选修且以 8 开头的课程号和平均分数。

```
USE stsco
SELECT cid AS '课程号', AVG (grade) AS '平均分数'
FROM score
WHERE cid LIKE '8 % '
GROUP BY cid
HAVING COUNT( * )>= 4
```

该语句采用 AVG 聚合函数、WHERE 子句、GROUP BY 子句、HAVING 子句。

查询结果:

```
课程号        平均分数
------------  -----------
801           88
```

5.4 排 序 查 询

SELECT 语句的 ORDER BY 子句用于对查询结果按升序(默认或 ASC)或降序(DESC)排列行,可按照一个或多个字段的值进行排序,ORDER BY 子句的语法格式如下:

```
[ ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n ]
```

其中，order_by_expression 是排序表达式，可以是列名、表达式或一个正整数。

【例 5.16】 将计算机专业的学生按出生时间先后排序。

```
USE stsco
SELECT *
FROM student
WHERE speciality = '计算机'
ORDER BY stbirthday
```

该语句采用 ORDER BY 子句进行排序。

查询结果：

stid	stname	stsex	stbirthday	speciality	tc
202002	袁志敏	男	1999 - 12 - 04	计算机	48
202001	徐桥	男	2000 - 02 - 25	计算机	52
202005	董莎	女	2000 - 04 - 19	计算机	50

【例 5.17】 将通信专业学生按“数字电路”课程成绩降序排序。

```
USE stsco
SELECT a.stname, b.cname, c.grade
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND b.cname = '数字电路' AND a.speciality = '通信'
ORDER BY c.grade DESC
```

该语句采用谓词连接和 ORDER BY 子句进行排序。

查询结果：

stname	cname	grade
罗俊杰	数字电路	93
冯露	数字电路	89
韩红丽	数字电路	71

5.5 连接查询

当一个查询涉及两个或多个表的数据时，需要指定连接列进行连接查询。

连接查询是关系数据库中的重要查询，在 T-SQL 中，连接查询有两大类表示形式：一类是连接谓词表示形式，另一类是使用关键字 JOIN 表示形式。

5.5.1 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，将这种表示形式称为连接谓词表示形式。连接谓词又称为连接条件，其一般语法格式如下：

[<表名 1.>] <列名 1> <比较运算符> [<表名 2.>] <列名 2>

比较运算符有：<、<=、=、>、>=、!=、<>、! <、! >。

连接谓词还有以下形式：

[<表名 1.>] <列名 1> BETWEEN [<表名 2.>] <列名 2> AND[<表名 2.>] <列名 3>

由于连接多个表存在公共列，为了区分是哪个表中的列，引入表名前缀指定连接列。例如，student.stid 表示 student 表的 stid 列，score.stid 表示 score 表的 stid 列。

为了简化输入，SQL 允许在查询中使用表的别名，可在 FROM 子句中为表定义别名，然后在查询中引用。

经常用到的连接如下所述。

- 等值连接：表之间通过比较运算符“=”连接起来，称为等值连接。
- 非等值连接：表之间使用非等号进行连接，称为非等值连接。
- 自然连接：如果在目标列中去除相同的字段名，称为自然连接。
- 自连接：将同一个表进行连接，称为自连接。

【例 5.18】 查询学生的情况和选修课程的情况。

```
USE stsco
SELECT student.*, score.*
FROM student, score
WHERE student.stid = score.stid
```

该语句采用等值连接。

查询结果：

stid	stname	stsex	stbirthday	speciality	tc	stid	cid	grade
201001	罗俊杰	男	2000-06-15	通信	52	201001	102	93
201001	罗俊杰	男	2000-06-15	通信	52	201001	205	94
201001	罗俊杰	男	2000-06-15	通信	52	201001	801	95
201002	韩红丽	女	1999-09-23	通信	49	201002	102	71
201002	韩红丽	女	1999-09-23	通信	49	201002	205	68
201002	韩红丽	女	1999-09-23	通信	49	201002	801	76
201004	冯露	女	1999-08-07	通信	50	201004	102	89
201004	冯露	女	1999-08-07	通信	50	201004	205	87
201004	冯露	女	1999-08-07	通信	50	201004	801	86
202001	徐桥	男	2000-02-25	计算机	52	202001	203	92
202001	徐桥	男	2000-02-25	计算机	52	202001	801	94
202002	袁志敏	男	1999-12-04	计算机	48	202002	203	70
202002	袁志敏	男	1999-12-04	计算机	48	202002	801	NULL
202005	董莎	女	2000-04-19	计算机	50	202005	203	84
202005	董莎	女	2000-04-19	计算机	50	202005	801	89

【例 5.19】 对上例进行自然连接查询。

```
USE stsco
SELECT student.*, score.cid, score.grade
FROM student, score
```

```
WHERE student.stid = score.stid
```

该语句采用自然连接。

查询结果：

stid	stname	stsex	stbirthday	speciality	tc	cid	grade
201001	罗俊杰	男	2000-06-15	通信	52	102	93
201001	罗俊杰	男	2000-06-15	通信	52	205	94
201001	罗俊杰	男	2000-06-15	通信	52	801	95
201002	韩红丽	女	1999-09-23	通信	49	102	71
201002	韩红丽	女	1999-09-23	通信	49	205	68
201002	韩红丽	女	1999-09-23	通信	49	801	76
201004	冯露	女	1999-08-07	通信	50	102	89
201004	冯露	女	1999-08-07	通信	50	205	87
201004	冯露	女	1999-08-07	通信	50	801	86
202001	徐桥	男	2000-02-25	计算机	52	203	92
202001	徐桥	男	2000-02-25	计算机	52	801	94
202002	袁志敏	男	1999-12-04	计算机	48	203	70
202002	袁志敏	男	1999-12-04	计算机	48	801	NULL
202005	董莎	女	2000-04-19	计算机	50	203	84
202005	董莎	女	2000-04-19	计算机	50	801	89

【例 5.20】 查询选修了“微机原理”且成绩在 80 分以上的学生姓名。

```
USE stsco
SELECT a.stid, a.stname, b.cname, c.grade
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND b.cname = '微机原理' AND C.grade > = 80
```

该语句实现了多表连接，并采用别名以缩写表名。

查询结果：

stid	stname	cname	grade
201001	罗俊杰	微机原理	94
201004	冯露	微机原理	87

说明：本例中为 student 指定的别名是 a，为 course 指定的别名是 b，为 score 指定的别名是 c。

【例 5.21】 查询选修了“801”课程且成绩高于学号“201002”的成绩的学生姓名。

```
USE stsco
SELECT a.cid, a.stid, a.grade
FROM score a, score b
WHERE a.cid = '801' AND a.grade > b.grade AND b.stid = '201002' AND b.cid = '801'
ORDER BY a.grade DESC
```

该语句实现了自连接，使用自连接需要为一个表指定两个别名。

查询结果:

```
cid       stid            grade
------    ------------    ------------
801       201001          95
801       202001          94
801       202005          89
801       201004          86
```

5.5.2 以 JOIN 为关键字指定的连接

T-SQL 扩展了以 JOIN 关键字指定连接的表示方式,使表的连接运算能力进一步增强。JOIN 连接在 FROM 子句的< joined_table >中指定。

语法格式:

```
< joined_table > : : =
{
< table_source > < join_type > < table_source > ON < search_condition >
  | < table_source > CROSS JOIN < table_source >
  | < joined_table >
}
```

说明:

其中,< join_type >为连接类型,ON 用于指定连接条件。< join_type >的语法格式如下:

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

其中,INNER 表示内连接,OUTER 表示外连接,CROSS 表示交叉连接,此为 JOIN 关键字指定的 3 种连接类型。

(1) 内连接

内连接按照 ON 所指定的连接条件合并两个表,返回满足条件的行。

内连接是系统默认的,可省略 INNER 关键字。

【例 5.22】 查询学生的情况和选修课程的情况。

```
USE stsco
SELECT *
FROM student INNER JOIN score ON student.stid = score.stid
```

该语句采用内连接,查询结果与例 5.18 相同。

【例 5.23】 查询选修了 102 课程且成绩在 85 分以上的学生情况。

```
USE stsco
SELECT a.stid, a.stname, b.cid, b.grade
FROM student a JOIN score b ON a.stid = b.stid
WHERE b.cid = '102' AND b.grade > = 85
```

该语句采用内连接,省略 INNER 关键字,使用了 WHERE 子句。

查询结果：

stid	stname	cid	grade
201001	罗俊杰	102	93
201004	冯露	102	89

(2) 外连接

在内连接的结果表，只有满足连接条件的行才能作为结果输出。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。外连接有以下 3 种。

- 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。

【例 5.24】 采用左外连接查询教师任课情况。

```
USE stsco
SELECT tname, cid
FROM teacher LEFT JOIN lecture ON (teacher.tid = lecture.tid)
```

该语句采用左外连接。

查询结果：

tname	cid
沈思敏	102
傅洁	NULL
许强	203
郑书雅	205
程建明	801

【例 5.25】 采用右外连接查询教师任课情况。

```
USE stsco
SELECT tid, cname
FROM lecture RIGHT JOIN course ON (course.cid = lecture.cid)
```

该语句采用右外连接。

查询结果：

tid	cname
102104	数字电路
204102	数据库系统
204105	微机原理
NULL	计算机网络
801106	高等数学

注意：外连接只能对两个表进行。

(3) 交叉连接

【例 5.26】 采用交叉连接查询教师和课程所有可能组合。

```
USE stsco
SELECT teacher.tname,course.cname
FROM teacher CROSS JOIN course
```

该语句采用交叉连接。

查询结果：

tname	cname
沈思敏	数字电路
傅洁	数字电路
许强	数字电路
郑书雅	数字电路
程建明	数字电路
沈思敏	数据库系统
傅洁	数据库系统
许强	数据库系统
郑书雅	数据库系统
程建明	数据库系统
沈思敏	微机原理
傅洁	微机原理
许强	微机原理
郑书雅	微机原理
程建明	微机原理
沈思敏	计算机网络
傅洁	计算机网络
许强	计算机网络
郑书雅	计算机网络
程建明	计算机网络
沈思敏	高等数学
傅洁	高等数学
许强	高等数学
郑书雅	高等数学
程建明	高等数学

5.6 子 查 询

在 SQL 中，一个 SELECT-FROM-WHERE 语句称为一个查询块。在 WHERE 子句或 HAVING 子句所指定条件中，可以使用另一个查询块的查询结果作为条件的一部分，这种将一个查询块嵌套在另一个查询块的子句指定条件中的查询称为嵌套查询。例如：

```
USE stsco
SELECT *
FROM student
```

```
WHERE stid IN
   (SELECT stid
    FROM score
    WHERE cid = '203'
   )
```

在本例中，下层查询块 SELECT stid FROM score WHERE cid＝'203'的查询结果，作为上层查询块 SELECT * FROM student WHERE stid IN 的查询条件，上层查询块称为父查询或外层查询，下层查询块称为子查询或内层查询，嵌套查询的处理过程是由内向外，即由子查询到父查询，子查询的结果作为父查询的查询条件。

T-SQL 允许 SELECT 多层嵌套使用，即一个子查询可以嵌套其他子查询，以增强查询能力。

子查询通常与 IN、EXISTS 谓词和比较运算符结合使用。

5.6.1 IN 子查询

IN 子查询用于进行一个给定值是否在子查询结果集中的判断，其语法格式如下：

```
expression [ NOT ] IN ( subquery )
```

当表达式 expression 与子查询 subquery 的结果集中的某个值相等时，IN 谓词返回 TRUE，否则返回 FALSE；若使用了 NOT，则返回的值相反。

【例 5.27】 查询选修了课程号为 203 的课程的学生情况。

```
USE stsco
SELECT *
FROM student
WHERE stid IN
   (SELECT stid
    FROM score
    WHERE cid = '203'
   )
```

该语句采用 IN 子查询。

查询结果：

stid	stname	stsex	stbirthday	speciality	tc
202001	徐桥	男	2000－02－25	计算机	52
202002	袁志敏	男	1999－12－04	计算机	48
202005	董莎	女	2000－04－19	计算机	50

【例 5.28】 查询选修某课程的学生人数多于 4 人的教师姓名。

```
USE stsco
SELECT tname AS '教师姓名'
FROM teacher
WHERE tid IN
   (SELECT tid
     FROM lecture
```

```
    WHERE cid IN
      (SELECT b.cid
       FROM course a, score b
       WHERE a.cid = b.cid
       GROUP BY b.cid
       HAVING COUNT(b.cid)> 4
      )
  )
```

该语句采用 IN 子查询,在子查询中使用了谓词连接、GROUP BY 子句、HAVING 子句。

查询结果:

```
教师姓名
------------------
程建明
```

5.6.2 比较子查询

比较子查询是指父查询与子查询之间用比较运算符进行关联,其语法格式如下:

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

其中,expression 为要进行比较的表达式,subquery 是子查询,ALL、SOME 和 ANY 是对比较运算的限制。

【例 5.29】 查询比所有计算机专业学生年龄都小的学生。

```
USE stsco
SELECT *
FROM student
WHERE stbirthday > ALL
  (SELECT stbirthday
   FROM student
   WHERE speciality = '计算机'
  )
```

该语句采用比较子查询。

查询结果:

```
stid        stname      stsex    stbirthday          speciality      tc
----------  ----------  -------- ------------------- ------------- ----------
201001      罗俊杰      男       2000-06-15          通信            52
```

【例 5.30】 查询课程号 801 的成绩高于课程号 205 的成绩的学生。

```
USE stsco
SELECT stid AS '学号'
FROM score
WHERE cid = '801' AND grade >= ANY
  (SELECT grade
```

```
    FROM score
    WHERE cid = '205'
   )
```

该语句采用比较子查询。

查询结果：

```
学号
---------------
201001
201002
201004
202001
202005
```

5.6.3 EXISTS 子查询

EXISTS 谓词用于测试子查询的结果是否为空表，若子查询的结果集不为空，则 EXISTS 返回 TRUE，否则返回 FALSE，如果为 NOT EXISTS，其返回值与 EXISTS 相反，其语法格式如下：

```
[ NOT ] EXISTS ( subquery )
```

【例 5.31】 查询选修 205 课程的学生姓名。

```
USE stsco
SELECT stname AS '姓名'
FROM student
WHERE EXISTS
   (SELECT *
    FROM score
    WHERE score.stid = student.stid AND cid =  '205'
   )
```

该语句采用 EXISTS 子查询。

查询结果：

```
姓名
-------------
罗俊杰
韩红丽
冯露
```

【例 5.32】 查询所有任课教师姓名和学院。

```
USE stsco
SELECT tname AS '教师姓名', school AS '学院'
FROM teacher
WHERE tid IN
   (SELECT tid
    FROM lecture a
```

```
    WHERE EXISTS
      (SELECT *
       FROM course b
       WHERE a.cid = b.cid
      )
  )
```

该语句采用 EXISTS 子查询。

查询结果:

教师姓名	学院
沈思敏	计算机学院
许强	通信学院
郑书雅	通信学院
程建明	数学学院

提示: 子查询和连接往往都要涉及两个表或多个表,其区别是连接可以合并两个表或多个表的数据,而带子查询的 SELECT 语句的结果只能来自一个表。

5.7 SELECT 查询的其他子句

SELECT 查询的其他子句包括 UNION、EXCEPT、INTERSECT、INTO、CTE、FROM、TOP 谓词等,下面分别介绍。

1. UNION

使用 UNION 可以将两个或多个 SELECT 查询的结果合并成一个结果集。

语法格式:

```
{ < query specification > | (< query expression > ) }
    UNION [ ALL ] < query specification > | (< query expression > )
    [ UNION [ ALL ] < query specification > | (< query expression > ) [...n] ]
```

说明:

< query specification >和(< query expression >都是 SELECT 查询语句。

使用 UNION 合并两个查询的结果集的基本规则如下。

- 所有查询中的列数和列的顺序必须相同。
- 数据类型必须兼容。

【例 5.33】 查询总学分大于 50 及学号小于 201051 的学生。

```
USE stsco
SELECT *
FROM student
WHERE tc > 50
UNION
SELECT *
FROM student
```

```
WHERE stid < 201051
```

该语句采用 UNION 将两个查询的结果合并成一个结果集。

查询结果：

stid	stname	stsex	stbirthday	speciality	tc
201001	罗俊杰	男	2000-06-15	通信	52
201002	韩红丽	女	1999-09-23	通信	49
201004	冯露	女	1999-08-07	通信	50
202001	徐桥	男	2000-02-25	计算机	52
202005	董莎	女	2000-04-19	计算机	50

2. EXCEPT 和 INTERSECT

EXCEPT 和 INTERSECT 用于比较两个查询结果，返回非重复值，EXCEPT 从左查询中返回右查询没有找到的所有非重复值，INTERSECT 返回 INTERSECT 操作数左右两边的两个查询都返回的所有非重复值。

语法格式：

```
{ < query_specification > | ( < query_expression > ) }
{ EXCEPT | INTERSECT }
{ < query_specification > | ( < query_expression > ) }
```

说明：

其中，< query specification >和< query expression >都是 SELECT 查询语句。

使用 EXCEPT 或 INTERSECT 的两个查询的结果集组合起来的基本规则如下。

- 所有查询中的列数和列的顺序必须相同。
- 数据类型必须兼容。

【例 5.34】 查询学过 801 课程但未学过 102 课程的学生。

```
USE stsco
SELECT a.stid AS '学号', a.stname AS '姓名'
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND c.cid = '801'
EXCEPT
SELECT a.stid AS '学号', a.stname AS '姓名'
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND c.cid = '102'
```

该语句从 EXCEPT 操作数左侧的查询返回右侧查询没有找到的所有非重复值。

查询结果：

学号	姓名
202001	徐桥
202002	袁志敏
202005	董莎

【例 5.35】 查询既学过 801 课程又学过 102 课程的学生。

```
USE stsco
SELECT a.stid AS '学号', a.stname AS '姓名'
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND c.cid = '801'
INTERSECT
SELECT a.stid AS '学号', a.stname AS '姓名'
FROM student a, course b, score c
WHERE a.stid = c.stid AND b.cid = c.cid AND c.cid = '102'
```

该语句返回 INTERSECT 操作数左右两边的两个查询都返回的所有非重复值。

查询结果:

学号	姓名
201001	罗俊杰
201002	韩红丽
201004	冯露

3. INTO

INTO 用于创建新表并将查询所得的结果插入新表中。

语法格式:

```
[ INTO new_table ]
```

说明:

new_table 是要创建的新表名,创建的新表的结构由 SELECT 所选择的列决定,新表中的记录由 SELECT 的查询结果决定,若 SELECT 的查询结果为空,则创建一个只有结构而没有记录的空表。

【例 5.36】 由 student 表创建 st 表,包括学号、姓名、性别、专业和学分。

```
USE stsco
SELECT stid, stname, stsex, speciality, tc INTO st
FROM student
```

该语句通过 INTO 子句创建新表 st,新表的结构和记录由 SELECT INTO 语句决定。

4. CTE

CTE 用于指定临时结果集,这些结果集称为公用表表达式(Common Table Expression,CTE)。

语法格式:

```
[ WITH <common_table_expression> [ ,...n ] ]
AS ( CTE_query_definition )
```

其中

```
<common_table_expression>::=
    expression_name [ ( column_name [ ,...n ] ) ]
```

说明：

- expression_name：CTE 的名称。
- column_name：在 CTE 中指定的列名，其个数要和 CTE_query_definition 返回的字段个数相同。
- CTE_query_definition：指定一个其结果集填充 CTE 的 SELECT 语句。CTE 下方的 SELECT 语句可以直接查询 CTE 中的数据。

注意：CTE 源自简单查询，并且在单条 SELECT、INSERT、UPDATE 或 DELETE 语句的执行范围内定义，该子句也可用在 CREATE VIEW 语句中，公用表表达式可以包括对自身的引用，这种表达式称为递归公用表表达式。

【例 5.37】 使用 CTE 从 score 表中查询学号、课程号和成绩，并指定新列名为 c_stid、c_cid、c_grade，再使用 SELECT 语句从 CTE 和 student 表中查询姓名为“徐桥”的学号、课程号和成绩。

```
USE stsco;
WITH cte_st(c_stid, c_cid, c_grade)
AS (SELECT stid, cid, grade FROM score)
SELECT c_stid, c_cid, c_grade
FROM cte_st, student
WHERE student.stname = '徐桥' AND student.stid = cte_st.c_stid
```

该语句通过 CTE 子句查询姓名为“徐桥”的学号、课程号和成绩。

查询结果：

```
c_stid      c_cid       c_grade
----------- ----------- -----------
202001      203         92
202001      801         94
```

【例 5.38】 计算 1～10 的阶乘。

```
WITH Cfact(n, k)
AS (
       SELECT n = 1, k = 1
       UNION ALL
       SELECT n = n + 1, k = k * (n + 1)
       FROM Cfact
       WHERE n < 10
    )
SELECT n, k FROM Cfact
```

该语句通过递归公用表表达式计算 1～10 的阶乘。

查询结果：

```
n                    k
-------------------- -----------
1                    1
2                    2
3                    6
4                    24
5                    120
6                    720
7                    5040
8                    40320
9                    362880
10                   3628800
```

5. FROM

FROM 指定用于 SELECT 的查询对象。

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 * /
     | rowset_function [ [ AS ] table_alias ]                     / * 行集函数 * /
           [ ( bulk_column_alias [ ,…n ] ) ]
     | user_defined_function [ [ AS ] table_alias ]               / * 指定表值函数 * /
     | OPENXML < openxml_clause >                                 / * XML 文档 * /
     | 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 * /
}
```

说明：

- table_or_view_name：指定 SELECT 语句要查询的表或视图。
- rowset_function：行集函数,通常返回一个表或视图。
- derived_table：是由 SELECT 查询语句的执行而返回的表,必须为其指定一个别名,也可以为列指定别名。
- joined_table：joined_table 为连接表。
- pivoted_table：将行转换为列。

< pivoted_table >的语法格式如下：

```
< pivoted_table > : : =
        table_source PIVOT < pivot_clause > [AS] table_alias
< pivot_clause > : : =
        ( aggregate_function ( value_column ) FOR pivot_column IN (< column_list >) )
```

- < unpivoted_table >：将列转换为行。

< unpivoted_table >的语法格式如下：

```
< unpivoted_table > : : =
        table_source UNPIVOT < unpivot_clause > table_alias
< unpivot_clause > : : =
        ( value_column FOR pivot_column IN ( < column_list > ) )
```

【例 5.39】 查找 student 表中 2000 年 4 月 30 日以前出生的学生的姓名和性别，并列出其专业属于通信还是计算机，1 表示是，0 表示否。

```
USE stsco
SELECT stname, stsex,通信,计算机
FROM student
PIVOT
(
COUNT(stid)
FOR speciality
IN (通信,计算机)
)AS pvt
WHERE stbirthday<'2000 - 04 - 30'
```

该语句通过 PIVOT 子句将通信、计算机行转换为列。

查询结果：

stname	stsex	通信	计算机
董莎	女	0	1
冯露	女	1	0
韩红丽	女	1	0
徐桥	男	0	1
袁志敏	男	0	1

6. TOP 谓词

使用 SELECT 语句进行查询时，有时需要列出前几行数据，可以使用 TOP 谓词对结果集进行限定。

语法格式：

```
TOP n [ percent ] [ WITH TIES]
```

说明：

- TOP n：获取查询结果的前 n 行数据。
- TOP n percent：获取查询结果的前 n%行数据。
- WITH TIES：包括最后一行取值并列的结果。

注意：TOP 谓词写在 SELECT 单词后面。使用 TOP 谓词时，应与 ORDER BY 子句一起使用，列出前几行才有意义。如果选用 WITH TIES 选项，则必须使用 ORDER BY 子句。

【例 5.40】 查询总学分前 2 名的学生情况。

```
USE stsco
```

```
SELECT TOP 2 stid,stname,tc
FROM student
ORDER BY tc DESC
```


该语句通过 TOP 谓词与 ORDER BY 子句一起使用,获取前 2 名的学生情况。

查询结果:

```
stid        stname        tc
----------- ------------- -----------
201001      罗俊杰        52
202001      徐桥          52
```

【例 5.41】 查询总学分前 3 名的学生情况(包含专业)。

```
USE stsco
SELECT TOP 3 WITH TIES stid,stname,speciality,tc
FROM student
ORDER BY tc DESC
```

该语句通过 TOP 谓词,选用 WITH TIES 选项并与 ORDER BY 子句一起使用,获取前 3 名的学生情况。

查询结果:

```
stid        stname        speciality      tc
----------- ------------- --------------- -----------
201001      罗俊杰        通信            52
202001      徐桥          计算机          52
201004      冯露          通信            50
202005      董莎          计算机          50
```

5.8 综 合 训 练

1. 训练要求

本章介绍 T-SQL 中数据定义语言(DDL)、数据操纵语言(DML)和数据查询语言(DQL),并重点讨论 SELECT 查询语句对数据库进行各种查询的方法,下面结合 stsco 学生成绩数据库进行数据查询的综合训练。

(1) 查询 student 表中通信专业学生的情况。

(2) 查询 score 表中学号为 202002,课程号为 203 的学生成绩。

(3) 查找学号为 201004,课程名为“高等数学”的学生成绩。

(4) 查找选修了 801 课程且为计算机专业学生的姓名及成绩,查出的成绩按降序排列。

(5) 查找学号为 201001 学生所有课程的平均成绩。

2. T-SQL 语句编写

根据题目要求,进行语句编写。

(1) 编写 T-SQL 语句如下:

```
USE stsco
```

```
SELECT *
FROM student
WHERE speciality = '通信'
```

查询结果：

stid	stname	stsex	stbirthday	speciality	tc
201001	罗俊杰	男	2000-06-15	通信	52
201002	韩红丽	女	1999-09-23	通信	49
201004	冯露	女	1999-08-07	通信	50

(2) 编写 T-SQL 语句如下：

```
USE stsco
SELECT *
FROM score
WHERE stid = '202002' AND cid = '203'
```

查询结果：

stid	cid	grade
202002	203	70

(3) 编写 T-SQL 语句如下：

```
USE stsco
SELECT *
FROM score
WHERE stid = '201004' AND cid IN
   (SELECT cid
    FROM course
    WHERE cname = '高等数学'
   )
```

该语句在子查询中，由课程名查出课程号，在外查询中，由课程号(在子查询中查出)和学号查出成绩。

查询结果：

stid	cid	grade
201004	801	86

(4) 编写 T-SQL 语句如下：

```
USE stsco
SELECT a.stname,c.grade
FROM student a,course b,score c
WHERE b.cid = '801' AND a.stid = c.stid AND b.cid = c.cid
ORDER BY grade DESC
```

该语句采用连接查询和 ORDER BY 子句进行查询。

查询结果：

stname	grade
罗俊杰	95
徐桥	94
董莎	89
冯露	86
韩红丽	76
袁志敏	NULL

(5) 编写 T-SQL 语句如下：

```
USE stsco
SELECT stid,avg(grade) AS 平均成绩
FROM score
WHERE stid = '201001'
GROUP BY stid
```

该语句采用聚合函数和 GROUP BY 子句进行查询。

查询结果：

stid	平均成绩
201001	94

5.9 小　　结

本章主要介绍了以下内容。

(1) T-SQL 中最重要的部分是查询功能，查询语言是 T-SQL 的核心，使用 SELECT 语句，包含 SELECT 子句、FROM 子句、WHERE 子句、GROUP BY 子句、HAVING 子句、ORDER BY 子句等。

(2) 投影查询、选择查询和排序查询。

投影查询通过 SELECT 语句的 SELECT 子句来表示，由选择表中的部分或全部列组成结果表。

选择查询通过 WHERE 子句实现，WHERE 子句给出查询条件，该子句必须紧跟 FROM 子句之后。

排序查询通过 ORDER BY 子句实现，查询结果按升序(默认或 ASC)或降序(DESC)排列行，可按照一个或多个字段的值进行排序。

(3) 连接查询是关系数据库中的重要查询，在 T-SQL 中，连接查询有两大类表示形式：一类是连接谓词表示形式，另一类是使用关键字 JOIN 表示形式。

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，将这种表示形式称为连接谓词表示形式。

在使用 JOIN 关键字指定的连接中，在 FROM 子句中用 JOIN 关键字指定连接的多个

表的表名，用 ON 子句指定连接条件。JOIN 关键字指定的连接类型有 3 种：INNER JOIN 表示内连接，OUTER JOIN 表示外连接，CROSS JOIN 表示交叉连接。

外连接有以下 3 种：左外连接(LEFT OUTER JOIN)、右外连接(RIGHT OUTER JOIN)、完全外连接(FULL OUTER JOIN)。

(4) 将一个查询块嵌套在另一个查询块的子句指定条件中的查询称为嵌套查询，在嵌套查询中，上层查询块称为父查询或外层查询，下层查询块称为子查询(Subquery)或内层查询。子查询通常包括 IN 子查询、比较子查询和 EXISTS 子查询。

(5) SELECT 查询的其他子句包括 UNION、EXCEPT、INTERSECT、INTO、CTE、FROM、TOP 谓词等。

习 题 5

一、选择题

5.1 使用 student 表查询年龄最小的学生的姓名和年龄，下列实现此功能的查询语句中，正确的是_____。

A. SELECT Sname,Min(Sage) FROM student

B. SELECT Sname,Sage FROM student WHERE Sage= Min(Sage)

C. SELECT TOP1 Sname,Sage FROM student

D. SELECT TOP1 Sname,Sage FROM student ORDER BY Sage

5.2 设在某 SELECT 语句的 WHERE 子句中，需要对 Grade 列的空值进行处理。下列关于空值的操作中，错误的是_____。

A. Grade IS notNULL　　B. Grade ISNULL

C. Grade=NULL　　D. Not(Grade ISNULL)

5.3 设在 SQL Server 中，有学生表(学号，姓名，年龄)，其中，姓名为 varchar(10)类型。查询姓“张”且名字是三个字的学生的详细信息，正确的语句是_____。

A. SELECT * FROM 学生表 WHERE 姓名 LIKE '张_'

B. SELECT * FROM 学生表 WHERE 姓名 LIKE '张__'

C. SELECT * FROM 学生表 WHERE 姓名 LIKE '张_' AND LEN(姓名)=3

D. SELECT * FROM 学生表 WHERE 姓名 LIKE '张__' AND LEN(姓名)=3

5.4 设在 SQL Server 中，有学生表(学号，姓名，所在系)和选课表(学号，课程号，成绩)。查询没选课的学生姓名和所在系，下列语句中能够实现该查询要求的是_____。

A. SELECT 姓名,所在系 FROM 学生表 a LEFT JOIN 选课表 b
ON a.学号=b.学号 WHERE a.学号 IS NULL

B. SELECT 姓名,所在系 FROM 学生表 a LEFT JOIN 选课表 b
ON a.学号=b.学号 WHERE b.学号 IS NULL

C. SELECT 姓名,所在系 FROM 学生表 a RIGHT JOIN 选课表 b
ON a.学号=b.学号 WHERE a.学号 IS NULL

D. SELECT 姓名,所在系 FROM 学生表 a RIGHT JOIN 选课表 b
ON a.学号=b.学号 WHERE b.学号 IS NULL

5.5 下述语句的功能是将两个查询结果合并成一个结果,其中正确的是_____。

A.
```
SELECT sno, sname, sage FROM student WHERE sdept = 'cs'
ORDER BY sage
UNION
SELECT sno, sname, sage FROM student WHERE sdept = 'is'
ORDER BY sage
```

B.
```
SELECT sno, sname, sage FROM student WHERE sdept = 'cs'
UNION
SELECT sno, sname, sage FROM student WHERE sdept = 'is'
ORDER BY sage
```

C.
```
SELECT sno, sname, sage FROM student WHERE sdept = 'cs'
UNION
SELECT sno, sname FROM student WHERE sdept = 'is'
ORDER BY sage
```

D.
```
SELECT sno, sname, sage FROM student WHERE sdept = 'cs'
ORDER BY sage
UNION
SELECT sno, sname, sage FROM student WHERE sdept = 'is'
```

二、填空题

5.6 在 EXISTS 子查询中,子查询的执行次数是由_____决定的。

5.7 在 IN 子查询和比较子查询中,先执行_____层查询,再执行_____层查询。

5.8 在 EXISTS 子查询中,先执行_____层查询,再执行_____层查询。

5.9 UNION 操作用于合并多个 SELECT 查询的结果,如果在合并结果时不希望去掉重复数据,应使用_____关键字。

5.10 在 SELECT 语句中同时包含 WHERE 子句和 GROUP BY 子句,则先执行_____子句。

三、问答题

5.11 什么是 SQL? 简述 SQL 的分类。

5.12 SELECT 语句中包括哪些子句? 简述各个子句的功能。

5.13 什么是连接谓词? 简述连接谓词表示形式的语法规则。

5.14 内连接、外连接有什么区别? 左外连接、右外连接和全外连接有什么区别?

5.15 简述常用聚合函数的函数名称和功能。

5.16 在一个 SELECT 语句中,当 WHERE 子句、GROUP BY 子句和 HAVING 子句同时出现在一个查询中时,SQL 的执行顺序如何?

5.17 在 SQL Server 中使用 GROUP BY 子句有什么规则?

5.18 什么是子查询? IN 子查询、比较子查询、EXISTS 子查询有何区别?

四、应用题

5.19 查询 student 表中总学分大于或等于 50 分学生的情况。

5.20 查找徐桥“高等数学”的成绩。

5.21 查找选修了“数字电路”的学生姓名及成绩,并按成绩降序排列。

5.22 查找“数字电路”和“微机原理”的平均成绩。

5.23 查询每个专业最高分的课程名和分数。

5.24　查询通信专业最高分的学生的学号、姓名、课程号和分数。

5.25　查询有两门以上(含两门)课程均超过 80 分的学生姓名及其平均成绩。

5.26　查询选学了至少 3 门课程的学生姓名。

实验 5　数 据 查 询

实验 5.1　数据查询 1

1. 实验目的及要求

(1) 理解 SELECT 语句的语法格式。

(2) 掌握 SELECT 语句的操作和使用方法。

(3) 具备编写和调试 SELECT 语句以进行数据库查询的能力。

2. 验证性实验

对 storepm 数据库 EmplInfo 表进行数据查询,验证和调试查询语句的代码。

(1) 使用两种方式查询 EmplInfo 表的所有记录。

① 使用列名表。

```
USE storepm
SELECT EmplNo, EmplName, Sex, Birthday, Native, Wages, DeptNo
FROM EmplInfo
```

② 使用 * 。

```
USE storepm
SELECT *
FROM EmplInfo
```

(2) 查询 EmplInfo 表有关员工号、姓名和籍贯的记录。

```
USE storepm
SELECT EmplNo, EmplName, Native
FROM EmplInfo
```

(3) 使用两种方式查询籍贯为上海和四川的员工信息。

① 使用 IN 关键字。

```
USE storepm
SELECT *
FROM EmplInfo
WHERE Native IN ('上海', '四川')
```

② 使用 OR 关键字。

```
USE storepm
SELECT *
FROM EmplInfo
WHERE Native = '上海' OR Native = '四川'
```

(4) 通过两种方式查询 EmplInfo 表中工资为 3500～4500 元的员工。

① 通过指定范围关键字。

```
USE storepm
SELECT *
FROM EmplInfo
WHERE Wages BETWEEN 3500 AND 4500
```

② 通过比较运算符。

```
USE storepm
SELECT *
FROM EmplInfo
WHERE Wages >= 3500 AND Wages <= 4500
```

(5) 查询籍贯是北京的员工的姓名、出生日期和部门号。

```
USE storepm
SELECT EmplName, Birthday, DeptNo
FROM EmplInfo
WHERE Native LIKE '北京%'
```

(6) 查询各个部门的员工人数。

```
USE storepm
SELECT DeptNo AS 部门号, COUNT(EmplNo) AS 员工人数
FROM EmplInfo
GROUP BY DeptNo
```

(7) 查询每个部门的总工资和最高工资。

```
USE storepm
SELECT DeptNo AS 部门号, SUM(Wages) AS 总工资, MAX(Wages) AS 最高工资
FROM EmplInfo
GROUP BY DeptNo
```

(8) 查询员工工资,按照工资从高到低的顺序排列。

```
USE storepm
SELECT *
FROM EmplInfo
ORDER BY Wages DESC
```

(9) 从高到低排列员工工资,查询前 3 名员工的信息。

```
USE storepm
SELECT TOP 3 EmplName, Wages
FROM EmplInfo
ORDER BY Wages DESC
```

3. 设计性试验

对 storepm 数据库 GoodsInfo 表进行数据查询,设计、编写和调试查询语句的代码,完成以下操作。

(1) 使用两种方式查询 GoodsInfo 表的所有记录。

① 使用列名表。

② 使用 * 。

(2) 查询 GoodsInfo 表有关商品号、商品名称和库存量的记录。

(3) 使用两种方式查询商品类型为笔记本电脑和服务器的商品信息。

① 使用 IN 关键字。

② 使用 OR 关键字。

(4) 通过两种方式查询 GoodsInfo 表中单价为 1000～8000 元的商品。

① 通过指定范围关键字。

② 通过比较运算符。

(5) 查询商品类型为“平板”的商品信息。

(6) 查询各类商品的库存量。

(7) 查询各类商品品种个数和最高单价。

(8) 查询各个商品的单价,按照从高到低的顺序排列。

(9) 从高到低排列商品的单价,查询前 3 个商品的信息。

4. 观察与思考

(1) LIKE 的通配符“%”和“_”有何不同?

(2) IS 能用“=”来代替吗?

(3) “=”与 IN 在什么情况下作用相同?

(4) 空值的使用可分为哪几种情况?

(5) 聚集函数能否直接使用在 SELECT 子句、WHERE 子句、GROUP BY 子句、HAVING 子句中?

(6) WHERE 子句与 HAVING 子句有何不同?

(7) COUNT(*)、COUNT(列名)、COUNT(DISTINCT 列名)三者的区别是什么?

实验 5.2 数据查询 2

1. 实验目的及要求

(1) 理解连接查询、子查询以及联合查询的语法格式。

(2) 掌握连接查询、子查询以及联合查询的操作和使用方法。

(3) 具备编写和调试连接查询、子查询以及联合查询语句以进行数据库查询的能力。

2. 验证性实验

对 storepm 数据库进行数据查询,验证和调试数据查询的代码。

(1) 对员工表 EmplInfo 和部门表 DeptInfo 进行交叉连接,观察所有的可能组合。

```
USE storepm
SELECT *
FROM EmplInfo CROSS JOIN DeptInfo
```

或

```
USE storepm
SELECT *
FROM EmplInfo, DeptInfo
```

(2) 查询每个员工及其所在部门的情况。

① 使用 JOIN 关键字的表示方式。

```
USE storepm
SELECT *
FROM EmplInfo INNER JOIN DeptInfo ON EmplInfo.DeptNo = DeptInfo.DeptNo
```

② 使用连接谓词的表示方式。

```
USE storepm
SELECT *
FROM EmplInfo, DeptInfo
WHERE EmplInfo.DeptNo = DeptInfo.DeptNo
```

(3) 采用自然连接查询员工及其所属部门的情况。

```
USE storepm
SELECT EmplInfo.*, DeptName
FROM EmplInfo JOIN DeptInfo ON EmplInfo.DeptNo = DeptInfo.DeptNo
```

该语句进行自然连接,去掉了结果集中的重复列。

(4) 查询部门号 D001 的员工工资高于员工号为 E003 的工资的职工情况。

① 使用 JOIN 关键字的表示方式。

```
USE storepm
SELECT a.EmplNo, a.EmplName, a.Wages, a.DeptNo
FROM EmplInfo a JOIN EmplInfo b ON a.Wages > b.Wages
WHERE a.DeptNo = 'D001' AND b.EmplNo = 'E003'
ORDER BY a.Wages DESC
```

② 使用连接谓词的表示方式。

```
USE storepm
SELECT a.EmplNo, a.EmplName, a.Wages, a.DeptNo
FROM EmplInfo a, EmplInfo b
WHERE a.Wages > b.Wages AND a.DeptNo = 'D001' AND b.EmplNo = 'E003'
ORDER BY a.Wages DESC
```

(5) 分别采用左外连接、右外连接、全外连接查询员工所属的部门。

① 左外连接。

```
USE storepm
SELECT EmplName, DeptName
FROM EmplInfo LEFT JOIN DeptInfo ON EmplInfo.DeptNo = DeptInfo.DeptNo
```

该语句采用关键字 LEFT JOIN 进行左外连接,当左表有记录而在右表中没有匹配记录时,右表对应列被设置为空值 NULL。

② 右外连接。

```
USE storepm
SELECT EmplName, DeptName
FROM EmplInfo RIGHT JOIN DeptInfo ON EmplInfo.DeptNo = DeptInfo.DeptNo
```

该语句采用关键字 RIGHT JOIN 进行右外连接，当右表有记录而在左表中没有匹配记录时，左表对应列被设置为空值 NULL。

③ 全外连接。

```
USE storepm
SELECT EmplName, DeptName
FROM EmplInfo FULL JOIN DeptInfo ON EmplInfo.DeptNo = DeptInfo.DeptNo
```

该语句采用关键字 FULL JOIN 进行全外连接。

(6) 查询销售部和财务部员工名单。

```
USE storepm
SELECT EmplNo, EmplName, DeptName
FROM EmplInfo a, DeptInfo b
WHERE a.DeptNo = b.DeptNo AND DeptName = '销售部'
UNION
SELECT EmplNo, EmplName, DeptName
FROM EmplInfo a, DeptInfo b
WHERE a.DeptNo = b.DeptNo AND DeptName = '财务部'
```

该语句采用集合操作符 UNION 进行并运算以实现集合查询。

(7) 分别采用 IN 子查询和比较子查询查询财务部和经理办的员工信息。

① IN 子查询。

```
USE storepm
SELECT *
FROM EmplInfo
WHERE DeptNo IN
   (SELECT DeptNo
    FROM DeptInfo
    WHERE DeptName = '财务部' OR DeptName = '经理办'
   )
```

该语句采用 IN 子查询。

② 比较子查询。

```
USE storepm
    SELECT *
    FROM EmplInfo
    WHERE DeptNo = ANY
       (SELECT DeptNo
        FROM DeptInfo
        WHERE DeptName IN ('财务部', '经理办')
       )
```

该语句采用比较子查询，其中，关键字 ANY 用于对比较运算符“=”进行限制。

(8) 列出所有比 D001 部门员工年龄都小的员工和出生日期。

```
USE storepm
SELECT EmplNo AS 员工号, EmplName AS 姓名, Birthday AS 出生日期
FROM EmplInfo
```

```
WHERE Birthday > ALL
   (SELECT Birthday
    FROM EmplInfo
    WHERE DeptNo = 'D001'
   )
```

该语句采用比较子查询,其中,关键字 ALL 用于对比较运算符“>”进行限制。

(9) 查询销售部的员工姓名。

```
USE storepm
SELECT EmplName AS 姓名
FROM EmplInfo
WHERE EXISTS
   (SELECT *
    FROM DeptInfo
    WHERE EmplInfo.DeptNo = DeptInfo.DeptNo AND DeptNo = 'D001'
   )
```

该语句采用 EXISTS 子查询。

3. 设计性试验

在数据库 storepm 中,设计、编写和调试查询语句的代码,完成以下操作。

(1) 对商品表 GoodsInfo 和订单明细表 DetailInfo 进行交叉连接,观察所有的可能组合。

(2) 查询商品销售情况。

① 使用 JOIN 关键字的表示方式。

② 使用连接谓词的表示方式。

(3) 采用自然连接查询商品销售情况。

(4) 查询员工销售情况。

① 使用 JOIN 关键字的表示方式。

② 使用连接谓词的表示方式。

(5) 对员工表 EmplInfo 和订单表 OrderInfo 分别进行左外连接、右外连接、全外连接。

① 左外连接。

② 右外连接。

③ 全外连接。

(6) 查询销售部的员工姓名、销售日期及销售总金额,并按销售总金额降序排列。

(7) 查询刘建新的销售总金额。

(8) 查询销售部和财务部的员工号。

4. 观察与思考

(1) 使用 JOIN 关键字的表示方式和使用连接谓词的表示方式有什么不同?

(2) 内连接与外连接有何区别?

(3) 举例说明 IN 子查询、比较子查询和 EXISTS 子查询的用法。

(4) 关键字 ALL、SOME 和 ANY 对比较运算有何限制?