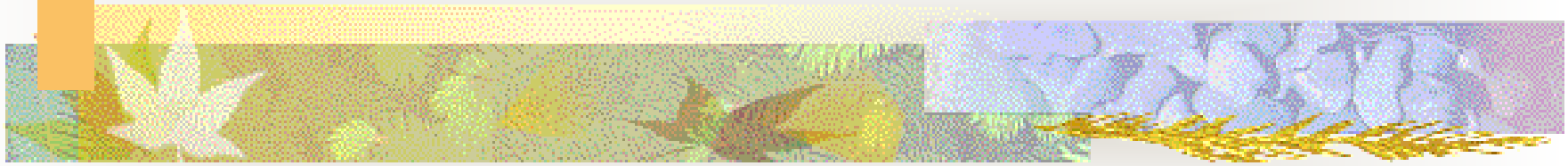
SQL Server数据库

SQL结构化查询语言



是本课程的重点，要在熟悉语句的语法框架的前提下，灵活地写出实现实际需求的**SQL**语句。本章的每个例子，都要在附录**Student**数据库上加以上机练习与变换。

1~2 Select语句

- SQL语句不区分大小写
- 本章的SQL语句可在查询分析器（推荐）、企业管理器SQL窗口里编辑、执行。
- 从数据库中检索行，并允许从一个或多个表中选择一个或多个行或列。语句格式
  - SELECT [all| distinct]*select_list* [INTO *new_table*]
    FROM *table_source*
    [WHERE *search_condition*]
    [GROUP BY *group_by_expression* [HAVING *search_condition*]]
    [ORDER BY *order_expression* [ASC | DESC]]
    - ✓ 整个SELECT语句的含义是：根据WHERE子句的筛选条件表达式，从FROM子句指定的表中找出满足条件记录，再按SELECT语句中指定的字段次序，筛选出记录中的字段值构造一个显示结果表。
      - 若有GROUP子句，则将结果按<分组表达式>的值进行分组，该值相等的记录为一个组。若GROUP子句带HAVING短语，则只有满足指定条件的组才会显示输出。
      - 若有ORDER BY子句，则最终的结果按照子句中指定的列（asc升序、desc降序）排好序后显示出来。
      - 提示：SELECT语句操作的是记录（数据）集合（一个表或多个表），而不是单独的一条记录。语句返回的也是记录集合（满足Where条件的），即结果表或结果集（ResultSet）。
    - ✓ 只有order by子句可以使用列序，别的（where、group by、having、compute by 子句）都不可以。

3 基于单表的查询

- 查询表中**指定的**字段
  - 在结果表中，字段是按照SELECT子句后的各个字段的顺序显示。
    - ✓ SELECT子句后的各个字段的先后顺序可与原表中的顺序不一致。如无特别需要，最好“照搬”表定义时的字段顺序，以减少运算代价。另，详细列出要查的个别字段，而不是“所有”字段，本质上也是为了提高结果的针对性、减少运算代价。
    - ✓ 例5-1：查看学生姓名、性别、住址：select student_name,student_sex,address from student_info
- 通配符 *
  - 表示要显示表的所有字段。
    - ✓ 例5-2：查看student_info表的所有信息： select * FROM student_info
- 使用单引号在结果集中加入字符串
  - 在选择列表中，在一字段的前面加一个单引号串，在结果集中每行都会出现该串，可起一个说明作用。（这与起别名不同，显然会人为地让结果集增大，在应用系统中极少这样做，只是在利用DBMS界面里写SQL时偶尔用用。）
    - ✓ 例5-3：select student_name, '家庭住址',address from student_info
- 使用别名
  - 在结果表的标题行所在列上显示更易读的文字。方法：
    - ✓ **写Select子句的选择列表时，字段名称 [as] 别名 或 别名=字段名称**
      - 别名也可用引号括起来。**但别名以数字开头时，必须加单引号。**
    - ✓ 例5-3：select student_name, address 家庭住址 from student_info

- 显示表达式的值
  - select语句的选择列表也可是算术表达式、函数等。
    - ✓ select getdate() 返回当前服务器时间（返回类型是：datetime类型）。
    - ✓ select cast(year(getdate()) as char(4))+'年'+cast(month(getdate()) as char(2))+'月'+cast(day(getdate()) as char(2))+'日'
    - ✓ 例5-5: select student_name,year(getdate())-year(born_date) 年龄 from student_info
- 使用**distinct** 消除重复的记录
  - 例5-6查询所有学生所属班级的班号：select distinct class_no from student_info
- 使用**where**子句查询特定条件的记录
  - 条件表达式可是关系(大、小、等)、逻辑(not、and、or)、特殊表达式及其混合。
    - ✓ 条件表达式：<、<=、=、>=、>、!=（不等于也可写成<> ）
      - 数值、字符串（日期）都可以比较。例5-7、5-8：1980年以后：born_date>'1980-12-31'
    - ✓ 逻辑表达式：not 非、and 且、or 或。例5-9~11。
    - ✓ 特殊表达式：
      - 通配符：%：任意多个字符；-：任意单个字符（有的DBS规定一个汉字是两个字符）。在SQL Server环境下，在通配符、字符串函数中，一个汉字是一个字符：select **substring('数据库原理及应用',1,3)** 结果是：数据库。姓刘且双字的：student_name like '刘_' 。
      - 单个字符匹配符：
        - [ ]（通配符 — 需匹配的字符）匹配指定范围内或者属于方括号所指定的集合中的任意单个字符。
        - [^]（通配符 — 无需匹配的字符）匹配不处于指定范围内或者不属于方括号内指定集合中的任意单个字符。

使用where子句查询特定条件的记录（特殊表达式）

SQL语句	含义示例
LIKE '5[%]'	5%
LIKE '[_]n'	_n
LIKE '[0-4hg]'	0、1、2、 3、4、 h 或 g
LIKE '[-acdf]'	-、a、c、d 或 f
LIKE '[[ ]'	[
LIKE ']'	]
LIKE 'abc[_]d%'	abc_d 和 abc_de
LIKE 'abc[def]'	abcd、abce 或abcf都可以

- [not] between ... and ... ： 闭区间（包括区间的端点）。例5-12 。
- is [not] null ： 是否为空。例5-13。
- [not] like ： 模式匹配，常和%、_、[]、[^]合用。例5-16~18 。
  - 姓刘、王的学生：student_name like ‘[刘王]%’ 。（不能加逗号，否则会把逗号也当成一种可能）
- [not] in：是否在一个集合里面（一般是子查询、明确限定的集合） 。例5-14~15 。
- [not] exists：指定一个子查询，检测行是否存在 。（子查询的选择列习惯上写成*）
  - 例子见下页。

使用where子句查询特定条件的记录（特殊表达式）

- 比较使用 EXISTS 和 IN 的查询（找出出版书籍的类型是“business” 的出版社名称）
  - SELECT distinct pub_name
    FROM publishers
    WHERE EXISTS (
      SELECT *
      FROM titles
      WHERE pub_id = publishers.pub_id AND type = 'business'
)
  - SELECT distinct pub_name
    FROM publishers
    WHERE pub_id IN (
      SELECT pub_id
      FROM titles
      WHERE type = 'business'
)

- P83习题：5-2 (2)

```
select  student_name
from student_info
where year(born_date)!=1980 and student_sex='女'
```

  - 对要求中的“不在”，不要被限制在“not in”。
  - 在上句中，虽**1980**也可写成串’1980’，但因year(日期)返回的是整型值，所以最好还是选择整型。

使用Order by 对查询结果排序

- 将查询结果按指定的列的值排好序（asc升序或desc降序，默认是asc）然后显示出来。可指定多个列，具体在排序时严格按照紧跟order by的顺序依次发挥作用。
  - select 选择列名表
    from 表
    [where 条件]
    order by 列表达式1 [asc|desc],表达式2 [asc|desc],...
  - 其中的列表达式可以是列名（或其别名）、表达式、非零的整数值（即该列在选择列名表中的次序（从1开始））
    - ✓ 例5-19可写成：select student_id, student_name,born_date from student_info order by 3
    - ✓ 例5-20可改写成：

```
select student_id 学号,student_name 姓名 , ru_date  加入日期
from student_info
where ru_date<‘2000-1-1’
order by 加入日期 desc,student_id
```

  - 空值在排序时，被视为比最小非空值(min)还小。

SQL聚合函数

- 利用聚合函数完成统计功能，通常和group by 一起使用，对每组进行计算，返回单个结果。具见表5-4。除count函数之外，别的聚合函数都忽略空值。
  - count ： 满足条件的行数。
    - select count(*) 19岁以上男生人数
      from student_info
      where year(getdate())-year(born_date)>19 and student_sex= '男'
  - max 、min：分别求最大值、最小值（忽略空值）
    - select min(born_date) 年龄最大, max(born_date) 年龄最小 from student_info
  - avg ： 对数值类型字段求平均值。 （忽略空值）
    - avg([all|distinct] 列名) ：用distinct 则在计算平均值时，不考虑值相同的；用all则全部考虑。默认是all。
    - select avg(course_time) 平均课时 , avg(course_score) 平均学分 from course_info
      where course_start=1
  - sum：对数值类型字段求和。 （忽略空值）
    - sum([all|distinct] 列名) ：用distinct 则在计算总和时，不考虑值相同的；用all则全部考虑。默认是all。
    - select count(course_no) 已有成绩门数 ,sum(result) 总分
      from result_info
      where student_id='20000101' and result is not null

使用Group by子句对查询结果进行分组

- 将查询结果按照指定的字段进行分组（即该字段的值相同的记录被分为一组，有几种值就被分为几组），可对分好的每个组运用聚合函数，得到统计结果。（group by 一般都和聚合函数一起用）
  - select 选择列表 from 表 group by 分组列表
    - ✓ 选择列表中出现的列要么是分组列表中出现的列，要么是运用了统计函数的列，别的不能出现。
    - ✓ 例5-26:select class_no,count(student_id) '学生人数' from student_info group by class_no
      - 分析：在student_info表内符合筛选条件的记录中，class_no有X个不同值就将分成X个组（每组的class_no都相同，这就意味着一组就是一个班所有的学生信息），对每组都运行： count(student_id)，有Y个学号，就代表本组内有Y个学生。（学号是肯定不重复的）
    - ✓ select class_no,student_sex,count(student_id) '学生人数'

      from student_info

      group by class_no, student_sex /*要能分析出：本句可查出每个班男生、女生各有多少名*/
      - 思考：若将上句中所有的class_no, student_sex ， 换成student_sex , class_no，本语句实现何功能？

使用Having子句对分好组后的查询结果进行筛选

- **先分组**，然后将满足条件的组输出。其与Where子句的区别：Where子句作用的对象是表，即从**表中选择出满足筛选条件的记录**。而Having 子句的作用对象是组，即从分好的组中选择满足筛选条件的组。
  - 例5-28：查询总人数大于2的班级**编号及其**学生总数。(班级编号要自动加入选择列)
    - ✓
```
select class_no 班级编号,count(student_id) 总人数
from student_info
group by class_no
   having count(student_id)>2
```
  - 扩展：查询总人数大于2的班级编号、名称、学生总数。
    - ✓
```
select student_info.class_no 班级编号,class_info.class_name 班级名称,count(student_id) 总人数
 from student_info,class_info
 where student_info.class_no=class_info.class_no
 group by student_info.class_no,class_info.class_name
    having count(student_id)>2
```

4 基于多表的连接查询

- 将一个查询同时涉及到两个或两个以上的表，称连接查询。连接有很多种，较重要的是：广义笛卡尔积、等值连接、自然连接。
- 广义笛卡尔积：（表示了关系连接后有多少种组合，无实际的应用意义）。R、S分别有m、n个元素，则R与S的广义笛卡尔积R×S共有(m*n)个元素。

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

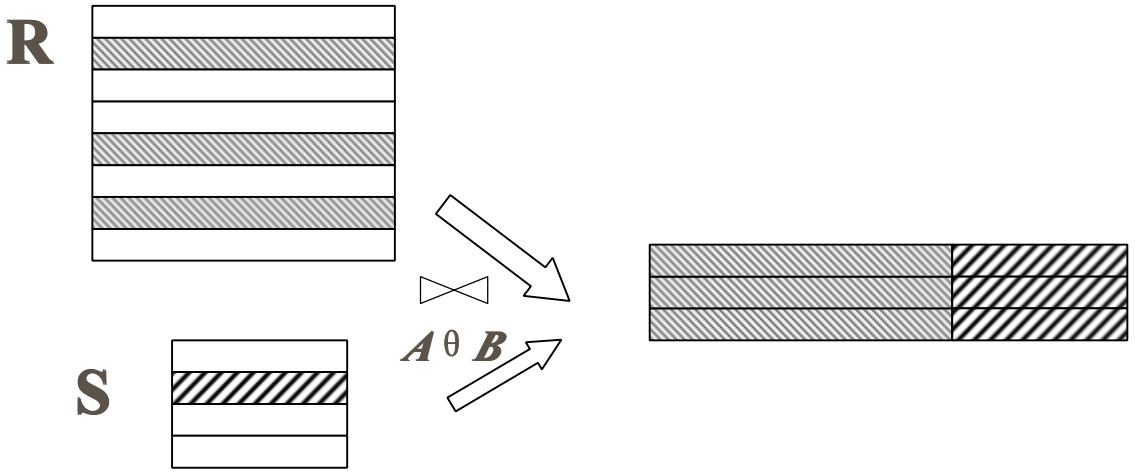
S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c3

R × S

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c3
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c3
a2	b2	c1	a1	b2	c2
a2	b2	c1	a1	b3	c2
a2	b2	c1	a2	b2	c3

- 一般的连接操作是从行的角度进行运算。



自然连接还需要取消重复列，所以是同时从行和列的角度进行运算。

- 连接举例

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

R

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

S

等值连接

- 等值连接 $R \underset{R.B=S.B}{\bowtie} S$。从关系$R$与$S$的广义笛卡尔积中选取指定的列值相等的那些元组。

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

- 自然连接 $R \bowtie S$。（在等值连接基础上去掉重复的列）
  - 它是一种特殊的等值连接
    - ✓ 两个关系中进行比较的分量必须是相同的属性组（包括字段名称也要相同，否则找不到连接的前提条件，两个关系表中，只有存在名称相同的若干列，才能自动去匹配，从而形成一个更大的关系表）
    - ✓ 在结果表中把重复的属性列去掉

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

- 例5-29等值连接

```
select student_info.*,class_info.*
from student_info,class_info
where student_info.class_no=class_info.class_no
```

  - 连接时，显式使用 表名.列名 的形式，就不会发生指代不清的错误。
  - 出现两个完全相同的class_no列，属于冗余。一般要主动去掉，形成5-30的最终结果。

- 例5-30自然连接（在例5-29基础上去掉重复的列）

```
select student_id, student_name, student_sex,born_date, student_info.class_no,
       tele_number,ru_date,address,comment, class_name,director,profession
from student_info,class_info
where student_info.class_no=class_info.class_no
```

- 例5-31表自身的自然连接(查询不同课程但成绩相同的学生学号、课程号、成绩) ★

```
select a.student_id,a.course_no,b.course_no,a.result
from result_info a,result_info b
where a.student_id=b.student_id and a.result=b.result and a.course_no!=b.course_no
```

  - 表自身的连接相当于将表另作了副本，从而形成两个表，便可正常连接。

- 例5-32查询选修了“汇编语言”且其成绩在70分以上的学生的学号、姓名、课程名和成绩（其实课程名已固定为“汇编语言”，列在结果集里，可能是为了方便阅读结果。）

```
select s.student_id,s.student_name,c.course_name,r.result
from student_info s,course_info c,result_info r
where s.student_id=r.student_id and c.course_no=r.course_no and c.course_name='汇编语言'  and r.result>70
```

  - 表连接时，建议：先写连接条件，然后再一一列出查询条件，连接条件和查询条件之间用and。

- P83习题5-2 （7）查询所有女生的马克思主义课程的成绩（除成绩之外，显然还至少要列出学号、姓名）

```
select s.student_id,student_name,result
from student_info s,course_info c,result_info r
where s.student_id=r.student_id and c.course_no=r.course_no and  student_sex='女' and course_name='马克思主义'
```

- 对P83习题5-2 （7）扩充：列出学号、姓名、班级名称、成绩

```
select s.student_id,student_name,class_name
from student_info s,course_info c,result_info r,class_info class
where s.student_id=r.student_id and c.course_no=r.course_no and s.class_no=class.class_no and
      student_sex='女' and course_name='马克思主义'
```

  - 是四个表的连接。更多表的连接时，也按上面的建议，写where子句的条件表达式。

以Join关键字指定的连接（T-SQL中引入的）

- From关键字后面的连接格式：
  - <第一个表> <连接类型> <第二个表> on <连接条件> 或
  - <第一个表> cross join <第二个表>
    - ✓ 不影响where子句中查询条件的编写。
    - ✓ 连接类型：
      - inner join 内连接。是默认的连接类型，inner可省。
      - {**left**|**right**|**full**} outer join：左、右、全 外连接。
- 内连接：按照on指定的连接条件合并两个表。
  - 例5-33:查找每个学生的基本情况以及班级情况

```
select student_info.*,class_name,director,profession
from student_info inner join class_info on student_info.class_no=class_info.class_no
```

  - 例5-34:查询选修了’2’号课程且成绩在60分以上的学生姓名和成绩

```
select student_name,result
from student_info join result_info on student_info.student_id=result_info.student_id
where course_no='2' and result>60
```

    - ✓ 相当于：

```
select student_name,result
from student_info,result_info
where student_info.student_id=result_info.student_id and course_no='2' and result>60
```

- 例5-35 查找选修“汇编语言”课程且成绩在70以上的学号、姓名、课程名、成绩

```
select student_info.student_id,student_name,course_name,result
from student_info join result_info on student_info.student_id=result_info.student_id join
     course_info on result_info.course_no=course_info.course_no
where  course_name='汇编语言' and result>70
```

  - 也可写成：

```
select student_info.student_id,student_name,course_name,result
from  result_info join student_info on student_info.student_id=result_info.student_id join
     course_info on result_info.course_no=course_info.course_no
where  course_name='汇编语言' and result>70
```

    - ✓ 表的顺序可随意。

外连接

- 保留哪一侧表的所有记录，就称那一侧的外连接。对于另一侧，则用null凑足。要根据语义要求，准确领会、选择哪一侧或全外连接。
  - 例5-37：查询200001班**所有学生**的学号、姓名以及他们选修的课程号。分析：显然要将学生表中的200001班的所有学生信息保留，该班学生如有未选课的（即result_info表中的student_id字段取值没有这些学生的学号），应显示出未选（即course_no字段为空）。

```
select student_info.student_id,student_name,result_info.course_no
from student_info left outer join result_info on student_info.student_id=result_info.student_id
where class_no='200001'
```

  - 也可写成：

```
select student_info.student_id,student_name,result_info.course_no
from student_info,result_info
where  student_info.student_id*=result_info.student_id and class_no='200001'
```

    - ✓ 思考：如将上句中From子句的student_info和result_info表位置互换，上述两条语句，应如何调整。
  - 例5-38即是将5-37的where子句去掉。

交叉连接 cross join

- 就是广义笛卡尔积（表示多少种组合可能性）
  - 见例5-39。

```
select student_id,student_name,course_no,course_name
from student_info cross join course_info
```

  - ✓ 也可写成：

```
select student_id,student_name,course_no,course_name
from student_info,course_info
```

5子查询

- 是一个嵌套在外层SQL语句（select、insert、update、delete）中的select语句，也称内部查询，包含子查询的SQL语句称为外部查询。执行过程：先执行内部查询，它查询出来的数据并不会被显示出来（注意和连接的区别），而是传递给外层语句的查询条件。使用子查询，可将复杂的查询分解为若干个简单的查询，因此，有利于提高可读性。
  - 例5-40:查询学生“张小强”所在的班级名称，使用子查询、表的连接。

```
select class_name
from class_info
where class_no=(
   select class_no
   from student_info
   where student_name='张小强'
)
```

    - ✓ 因该学生对应的班号只能有一个（子查询返回单个记录），此时就可用等号，否则只能用 in。（见例5-41、5-42：选修课程号为”13”的学生可有多个学生，也就会有多个学号。）用等号还是in的确定就是根据子查询返回的结果是否是只有一个值。
  - 使用表的连接：

```
select class_name
from class_info,student_info
where class_info.class_no=student_info.class_no and student_name='张小强'
```

使用[not] in的子查询

- 表达式 [not] in (子查询)
  - 例5-43查询选修了“数据库原理”课程的学生学号和姓名。

```
select student_id, student_name
from student_info
where student_id in (
   select student_id
   from result_info
   where course_no in (
      select course_no
      from course_info
      where course_name='数据库原理'
   )
)
```

  - 思考：
    - ✓ 将该语句转化成表的连接。
    - ✓ 例5-44中的没选修，not 应加在哪个子查询中？

更新语句中的子查询

- 插入子查询结果
  - 要添加的表和子查询返回的结果集二者之间，对应的列名可不相同，但对应列的顺序、类型、个数要相同，这样才能相容。（具见例5-45。）
- 带子查询的修改与删除
  - 例5-46将“200001”班的学生成绩增加10分。

```
update result_info
set result=result+10
where student_id in (
   select student_id
   from student_info
   where class_no='200001'
)
```

  - 删除未选任一门课程的学生信息

```
delete from student_info
where student_id not in (
   select distinct student_id from result_info
)
```

使用比较运算符的子查询

- 格式：表达式 比较运算符 all|any (子查询) 。
  - all ：表示对所有情况都满足时，才返回True。
  - any：表示只要存在一个满足就返回真。
  - 例5-47：**比所有200002班学生年龄都大**(暗含查询结果中的学生都不是该班的。)：born_date **< all** (select born_date from...)
    - ✓ < all 相当于 小于 min最小值:born_date < (select min(born_date) from ...)
    - ✓ > all 相当于 大于 max最大值:born_date > (select max(born_date) from ...)
  - 例5-48：**比200002班某个学生年龄大**：born_date < any (select born_date from...)
    - ✓ < any 相当于 小于 max最大值
    - ✓ > any 相当于 大于 min最小值
  - 例5-49：不低于14号课程的最低成绩：
    - ✓ result **!< any** (select result from ...)
    - ✓ result **>=** (select **min**(result) from ...)

使用 [not] exists的子查询

- 测试子查询的结果是否为空——空，返回假；非空，返回真。格式：[not] exists (子查询)
  - 例5-50查选修“13”号课程的学号、姓名

```
select student_id,student_name
from student_info
where exists (
  select  *
  from result_info
  where student_info.student_id=result_info.student_id and course_no='13'
)
```

    - ✓ 思考：如将上句中的exists 换成not exists ，具有什么功能。**没选“13”号课程的？**
  - 例5-51查选修了全部课程的学号、姓名。★

```
select student_id,student_name
from student_info
where not exists(
  select *
  from course_info
  where not exists (
    select *
    from result_info
    where student_info.student_id=result_info.student_id and course_info.course_no= result_info.course_no
  )
)
```

  - 分析：不存在一门课程是他所不选的，这类学生即为所求。

相关子查询

- 子查询的where子句引用外部查询。具体执行过程：外部查询将相关的列值传给内部查询（外部查询的表有多少行，子查询就执行多少次）；若子查询的任何行与其匹配，则当前的外部查询所在的行就作为结果集的一条记录，否则，将考虑外部查询的下一行，如此反复，直至处理完外部表的所有行。
  - 例5-52查找“计算机应用001班”的学生学号、姓名。

```
select student_id,student_name
from student_info
where '计算机应用001班' in (
  select class_name
  from class_info
  where class_no=student_info.class_no
)
```

    - ✓ where子句也可写成：class_no in (select class_no from class_info where class_name= '计算机应用001班')
      - 上述两种写法，依据自己的习惯选择即可。

使用Union运算符合并多个结果

- 使用Union组合的结果集必须具有相同的结构（列数要相同，列的类型要兼容），组合后，会自动去掉重复的记录。
  - 例5-53显示“200001班”、“200002班”的学号、姓名。

```
select student_id,student_name
from student_info
where class_no='200001'
union
select student_id,student_name
from student_info
where class_no='200002'
```

    - ✓ 也可写成一个查询：
      - where class_no in ('200001','200002')
      - where class_no='200001' or class_no='200002'

在查询的基础上创建新表

- 用select 选择列表 into 新表的表名 from ...，创建新表并将结果集作为该新表的初始内容。新表可是基表（也称永久表）或临时表。
  - 例5-54:将查询得到的学号、姓名和班级名插入到新建的表student_class中。
  - 例5-55:创建200001班的学生表studentsIn200001
  - 例5-56:创建学生信息表的一个空副本（即只有表的结构而无记录数据）。
    - ✓ select * into student_info_backup from student_info where 0>1
      - 因 **0>1** 恒假， 所以查询得到的结果为空，但保留了表的结构定义，以供新表的结构定义使用。