2026年春季学期

数据库系统概论

An Introduction to Database Systems

第三章 关系数据库标准语言SQL

中国科学技术大学
人工智能与数据科学学院

黄振亚，huangzhy@ustc.edu.cn

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

嵌套查询(续)

- 嵌套查询概述
  - 一个SELECT-FROM-WHERE语句称为一个查询块
  - 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

嵌套查询(续)

查询选修2号课程的学生姓名。

```
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 '）；
```

嵌套查询(续)

- 层层嵌套方式反映了 SQL语言的结构化
  - 上层的查询块称为外层查询或父查询
  - 下层查询块称为内层查询或子查询
- SQL语言允许多层嵌套查询
  - 一个子查询中可以嵌套其他子查询
    - 子查询限制：不能使用ORDER BY子句
- 有些嵌套查询可以用连接运算替代

嵌套查询求解方法

- 不相关子查询
- 相关子查询

3.4.3 嵌套查询

一、带有IN谓词的子查询

二、 带有比较运算符的子查询

三、 带有ANY（SOME）或ALL谓词的子查询

四、 带有EXISTS谓词的子查询

一、带有IN谓词的子查询

[例3.57] 查询与“刘晨”在同一个系学习的学生。

此查询要求可以分步来完成

① 确定“刘晨”所在系名

```
SELECT  Sdept
FROM    Student
WHERE  Sname= ' 刘晨 ';
```

结果为： CS

Sno	Sname	Ssex	Sage	Sdept
20180001	LY	M	20	CS
20180002	LC	W	19	CS
20180003	WM	W	18	MA
20180004	ZL	M	19	IS

带有IN谓词的子查询（续）

② 查找所有在CS系学习的学生。

```
SELECT  Sno，Sname，Sdept
FROM    Student
WHERE  Sdept = ' CS ';
```

结果为：

Sno	Sname	Sdept
20180001	李勇	CS
20180002	刘晨	CS

带有IN谓词的子查询（续）

一条sql：将第一步查询嵌入到第二步查询的条件中

【解3.57】

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

此查询为不相关子查询（可以改写成连接查询）。

返回

带有IN谓词的子查询（续）

[例3.57] 查询与“刘晨”在同一个系学习的学生。

一条sql：用自身连接完成[例3.57]查询要求

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

嵌套查询求解方法

- 不相关子查询：

  子查询的查询条件不依赖于父查询

  - 由里向外逐层处理。即每个子查询在上一级查询处理之前求解，子查询的结果用于建立其父查询的查找条件。
  - 可以转为连接查询

带有IN谓词的子查询（续）

[例3.58]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’
           )
      );
```

③ 最后在Student关系中取出Sno和Sname

② 然后在SC关系中找出选修了3号课程的学生学号

① 首先在Course关系中找出“信息系统”的课程号，为3号

此查询为不相关子查询（可以改写成连接查询）

使用嵌套还是使用连接？

带有IN谓词的子查询（续）

[例3.58]查询选修了课程名为“Information System”的学生学号和姓名(Mysql 例子)

```
SELECT Sno, Sname
FROM Student
WHERE Sno IN(
    SELECT Sno
    FROM SC
    WHERE Cno IN(
        SELECT Cno
        FROM Course
        WHERE Cname='Information System'
    )
);
```

	Sno	Sname
▶	20180001	LY
	20180002	LC

带有IN谓词的子查询（续）

用连接查询实现[例3. 58]

[例3. 58]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，SC，Course
WHERE Student.Sno = SC.Sno  AND
      SC.Cno = Course.Cno AND
      Course.Cname='信息系统';
```

3.4.3 嵌套查询

一、带有IN谓词的子查询

二、 带有比较运算符的子查询

三、 带有ANY（SOME）或ALL谓词的子查询

四、 带有EXISTS谓词的子查询

二、带有比较运算符的子查询

- 当能确切知道内层查询返回单值时，可用比较运算符（>，<，=，>=，<=，!=或<>）。
- 与ANY或ALL谓词配合使用

带有比较运算符的子查询（续）

例：假设一个学生只可能在一个系学习，并且必须属于一个系，则在[例3.57]可以用 = 代替IN：

[例3.57] 查询与“刘晨”在同一个系学习的学生。

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

带有比较运算符的子查询（续）

注意： 子查询一定要跟在比较符之后

错误的例子：

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

带有比较运算符的子查询（续）

［例3.59］找出每个学生超过他选修课程平均成绩的课程号。

```
SELECT Sno，  Cno
FROM  SC  x
WHERE Grade >= (SELECT AVG(Grade)
                FROM  SC y
                WHERE y.Sno = x.Sno);
```

相关子查询

带有比较运算符的子查询（续）

- 可能的执行过程：

1. 从外层查询中取出SC的一个元组x，将元组x的Sno值（20180001）传送给内层查询。

```
SELECT AVG(Grade)
FROM SC y
WHERE y.Sno='20180001';
```

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

2. 执行内层查询，得到值88.3（近似值），用该值代替内层查询，得到外层查询：

```
SELECT Sno， Cno
FROM  SC x
WHERE Grade >=88.3;
```

Sno	AVG(Grade)
20180001	88.3333
20180002	85.0000

带有比较运算符的子查询（续）

3. 执行这个查询，得到

（20180001，1）

4.外层查询取出下一个元组重复做上述1至3步骤，直到外层的SC元组全部处理完毕。结果为:

（20180001，1）

（20180002，2）

	Sno	Cno
▶	20180001	1
	20180002	2

这是一个相关子查询：无法将子查询一次性求解出来，然后求接父查询。因此，内外查询相关，必须反复求值

嵌套查询求解方法（续）

- 相关子查询：子查询的查询条件依赖于父查询
  - 首先取外层查询中表的第一个元组，根据它与内层查询相关的属性值处理内层查询，若WHERE子句返回值为真，则取此元组放入结果表
  - 然后再取外层表的下一个元组
  - 重复这一过程，直至外层表全部检查完为止
  - 不可转为连接查询

3.4.3 嵌套查询

一、带有IN谓词的子查询

二、 带有比较运算符的子查询

三、 带有ANY（SOME）或ALL谓词的子查询

四、 带有EXISTS谓词的子查询

带有ANY（SOME）或ALL谓词的子查询

需要配合使用比较运算符

> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值
= ANY	等于子查询结果中的某个值
=ALL	等于子查询结果中的所有值（通常没有实际意义）
!=（或<>）ANY	不等于子查询结果中的某个值
!=（或<>）ALL	不等于子查询结果中的任何一个值

谓词语义

- ANY：任意一个值
- ALL：所有值

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.60] 查询非计算机系中比计算机系任意一个学生年龄小的学生姓名，年龄和系名

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

带有ANY（SOME）或ALL谓词的子查询 （续）

结果：

Sname	Sage	Sdept
王敏	18	MA
张立	19	IS

执行过程：

1.RDBMS执行此查询时，首先处理子查询，找出CS系中所有学生的年龄，构成一个集合(20，19)

2. 处理父查询，找所有不是CS系且年龄小于20 或 19的学生

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.60] 查询非计算机系中比计算机系某一学生年龄小的学生姓名，年龄和系名

用聚集函数实现[例3.60]

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

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.61] 查询其他系中比计算机科学系所有学生年龄都小的学生姓名及年龄。

方法一：用ALL谓词

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

带有ANY（SOME）或ALL谓词的子查询 （续）

方法二：用聚集函数

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

带有ANY（SOME）或ALL谓词的子查询 （续）

[例3.61] 查询其他系中比计算机科学系所有学生年龄都小的学生姓名及年龄(Mysql 例子)

方法一：用ALL谓词

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

方法二：用聚集函数

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

	Sname	Sage
▶	WM	18

带有ANY（SOME）或ALL谓词的子查询 （续）

表3.7 ANY（或SOME），ALL谓词与聚集函数、IN谓词的等价转换关系

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

3.4.3 嵌套查询

一、带有IN谓词的子查询

二、 带有比较运算符的子查询

三、 带有ANY（SOME）或ALL谓词的子查询

四、 带有EXISTS谓词的子查询（难点）

带有EXISTS谓词的子查询(续）

- **EXISTS谓词**
  - 存在量词 ∃
  - 带有EXISTS谓词的子查询不返回任何数据，只产生逻辑真值“true”或逻辑假值“false”。
    - 若内层查询结果非空，则外层的WHERE子句返回真值
    - 若内层查询结果为空，则外层的WHERE子句返回假值
  - 由EXISTS引出的子查询，其目标列表达式通常都用 *
    - 因为带EXISTS的子查询只返回真值或假值，给出列名无实际意义
- **NOT EXISTS谓词**
  - 若内层查询结果非空，则外层的WHERE子句返回假值
  - 若内层查询结果为空，则外层的WHERE子句返回真值

带有EXISTS谓词的子查询(续）

[例3.62]查询所有选修了1号课程的学生姓名。

思路分析：

- 本查询涉及Student和SC关系
- 在Student中依次取每个元组的Sno值，用此值去检查SC关系
- 若SC中存在这样的元组，其Sno值等于此Student.Sno值，并且其Cno= '1'，则取此Student.Sname送入结果关系

带有EXISTS谓词的子查询(续）

- [例3.62]查询所有选修了1号课程的学生姓名。

用嵌套查询

```
SELECT Sname
FROM  Student
WHERE EXISTS
        (SELECT *
         FROM SC
         WHERE Sno = Student.Sno AND Cno= ' 1 ');
```

带有EXISTS谓词的子查询(续）

- **[例3.62] 用连接运算**

```
SELECT Sname
FROM Student, SC
WHERE Student.Sno=SC.Sno AND SC.Cno= '1';
```

带有EXISTS谓词的子查询(续）

［例3.63］ 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 Sno = Student.Sno AND Cno='1');
```

若内层查询结果为空，则外层的WHERE子句返回真值，Sname放入结果表

带有EXISTS谓词的子查询(续）

- 不同形式的查询间的替换
  - 一些带EXISTS或NOT EXISTS谓词的子查询不能被其他形式的子查询等价替换
  - 所有带IN谓词、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS谓词的子查询等价替换

带有EXISTS谓词的子查询(续）

例：[例3.57]查询与“刘晨”在同一个系学习的学生。

可以用带EXISTS谓词的子查询替换 (第四种解法)：

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

带有EXISTS谓词的子查询(续）

用EXISTS/NOT EXISTS实现全称量词

SQL语言中没有全称量词∀ （For all）

可以把带有全称量词的谓词转换为等价的带有存在量词的谓词：

$$(\forall x)P \equiv \neg (\exists x(\neg P))$$

带有EXISTS谓词的子查询(续）

［例3. 64］查询选修了全部课程的学生姓名。

```
SELECT Sname
FROM Student
WHERE NOT EXISTS
    （SELECT *
      FROM Course
      WHERE（1） NOT EXISTS
            (SELECT *
             FROM SC
             WHERE（2） Sno= Student.Sno
                 AND Cno= Course.Cno
            )
    ）；
```

不存在没有被选修的课程

where	
子查询1	
Where1	
子查询2	
Where 2	

- NOT EXISTS谓词
  - 若内层查询结果非空，则外层的WHERE子句返回假值
  - 若内层查询结果为空，则外层的WHERE子句返回真值

带有EXISTS谓词的子查询(续）

［例3.64］查询选修了全部课程的学生姓名。

思路分析：查询这样的学生，没有一门课是他不选的

where	
子查询1	
Where1	
子查询2	
Where 2	

- **NOT EXISTS谓词**
  - 若内层查询结果非空，则外层的**WHERE**子句返回假值
  - 若内层查询结果为空，则外层的**WHERE**子句返回真值

Sno	Sname	Ssex	Sage	Sdept
20180001	LY	M	20	CS
20180002	LC	W	19	CS
20180003	WM	W	18	MA
20180004	ZL	M	19	IS
20180005	CXQ	M	19	DS

Cno	Cname
1	DB_Design
2	MATH
3	Information System

Sno	Cno	Grade	Semester
20180001	1	92	20192
20180001	2	85	20201
20180001	3	88	20202
20180002	2	90	20192
20180002	3	80	20201

带有EXISTS谓词的子查询(续）

[例3.64] 查询选修了全部课程的学生姓名(Mysql 例子)

```
SELECT Sname
FROM Student
WHERE NOT EXISTS(
    SELECT *
    FROM Course
    WHERE NOT EXISTS(
        SELECT *
        FROM SC
        WHERE Sno= Student.Sno
        AND Cno= Course.Cno
    )
);
```

	Sname

结果为空，因为没有人选修了全部课程！

带有EXISTS谓词的子查询(续）

用EXISTS/NOT EXISTS实现逻辑蕴函(难点)

- SQL语言中没有蕴函(Implication)逻辑运算
- 可以利用谓词演算将逻辑蕴函谓词等价转换为：

$$p \rightarrow q \equiv \neg p \vee q$$

带有EXISTS谓词的子查询(续）

[例3.65]查询至少选修了学生20180002选修的全部课程的学生号码。

思路分析：

- 用逻辑蕴函表达：查询学号为x的学生，对所有的课程y，只要20180002学生选修了课程y，则x也选修了y。
- 形式化表示：

  用P表示谓词 “学生20180002选修了课程y”

  用q表示谓词 “学生x选修了课程y”

  则上述查询为: $(\forall y)\ p \rightarrow q$

带有EXISTS谓词的子查询(续）

- 等价变换：

$$(\forall y)p \rightarrow q \equiv \neg (\exists y (\neg(p \rightarrow q)))$$
$$\equiv \neg (\exists y (\neg(\neg p \vee q)))$$
$$\equiv \neg \exists y(p \wedge \neg q)$$

- 变换后语义：不存在这样的课程y，学生20180002选修了y，而学生x没有选。

带有EXISTS谓词的子查询(续）

- 用NOT EXISTS谓词表示：

```
SELECT DISTINCT Sno
FROM SC SCX
WHERE NOT EXISTS                                          //不存在
        (SELECT *
         FROM SC SCY                                      //0002学生学过的课程y
         WHERE SCY.Sno = ' 20180002 '  AND
               NOT EXISTS                                 //x同学没学过
               (SELECT *
                FROM SC SCZ
                WHERE SCZ.Sno=SCX.Sno AND
                      SCZ.Cno=SCY.Cno));
```

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

3.4.4 集合查询

- 集合操作的种类
  - 并操作UNION
  - 交操作INTERSECT
  - 差操作EXCEPT
- 注：参加集合操作的各查询结果的列数必须相同；对应项的数据类型也必须相同
  - 同目，同属性

集合查询（续）

[例3.66] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法一：

```
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

集合查询（续）

[例3.66] 查询计算机科学系的学生及年龄不大于19岁的学生。

方法二：

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

集合查询（续）

[例3.67] 查询选修了课程1或者选修了课程2的学生。

```
SELECT Sno
FROM SC
WHERE Cno=' 1 '
UNION
SELECT Sno
FROM SC
WHERE Cno= ' 2 ';
```

集合查询（续）

[例3.68] 查询计算机科学系的学生与年龄不大于19岁的学生的交集

方法一

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

集合查询（续）

- [例3.68] 实际上就是查询计算机科学系中年龄不大于19岁的学生

  方法二

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

集合查询（续）

[例3.69] 查询选修课程1的学生集合与选修课程2的学生集合的交集

```
SELECT Sno
FROM SC
WHERE Cno='1'
INTERSECT
SELECT Sno
FROM SC
WHERE Cno='2';
```

可以用AND吗？

集合查询（续）

[例3.69]实际上是: 查询既选修了课程1又选修了课程2的学生

```
SELECT Sno
FROM SC
WHERE Cno=' 1 ' AND Sno IN
                    (SELECT Sno
                     FROM SC
                     WHERE Cno=' 2 ');
```

集合查询（续）

[例3.70] 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'
EXCEPT
SELECT  *
FROM Student
WHERE Sage <=19;
```

集合查询（续）

[例3.70]实际上是查询计算机科学系中年龄大于19岁的学生

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

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

3.4.5 基于派生表的查询

- 子查询不仅可以出现在WHERE子句中，还可以出现在FROM子句中，这时子查询生成的临时派生表（Derived Table）成为主查询的查询对象

```
SELECT
FROM [table], (select    from    where)
WHERE
```

3.4.5 基于派生表的查询

[例3.59]找出每个学生超过他自己选修课程平均成绩的课程号

方法一：
```
SELECT Sno， Cno
FROM  SC  x
WHERE Grade >= (SELECT AVG(Grade)
                FROM  SC y
                WHERE y.Sno = x.Sno);
```

方法二：
```
SELECT Sno, Cno
FROM SC, (SELECT Sno, Avg(Grade)
          FROM SC
          GROUP BY Sno)
           AS   Avg_sc(avg_sno,avg_grade)
WHERE SC.Sno = Avg_sc.avg_sno
      and SC.Grade >=Avg_sc.avg_grade
```

3.4.5 基于派生表的查询

- 如果子查询中没有聚集函数，派生表可以不指定属性列，子查询SELECT子句后面的列名为其缺省属性

[例3.60]查询所有选修了1号课程的学生姓名，可以用查询完成

```
SELECT Sname
 FROM    Student,
         (SELECT Sno FROM SC WHERE Cno='1') AS SC1
 WHERE  Student.Sno=SC1.Sno;
```

3.4 数据查询

- 3.4.1 单表查询
- 3.4.2 连接查询
- 3.4.3 嵌套查询
- 3.4.4 集合查询
- 3.4.5 基于派生表的查询
- 3.4.6 Select语句的一般形式

3.4.6 SELECT语句的一般格式

SELECT [ALL|DISTINCT]

<目标列表达式> [别名] [，<目标列表达式> [别名]] …

FROM <表名或视图名> [别名]

[，<表名或视图名> [别名]] …

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]

1.目标列表达式格式

- ✓ 目标列表达式格式

  （1） *

  （2） <表名>.*

  （3） COUNT([DISTINCT|ALL]*)

  （4） [<表名>.]<属性列名表达式>[,<表名>.]<属性列名表达式>]…

其中<属性列名表达式>可以是由属性列、作用于属性列的聚集函数和常量的任意算术运算（+，-，*，/）组成的运算公式

2. 聚集函数的一般格式

{ COUNT
SUM
AVG
MAX
MIN } （[DISTINCT|ALL] <列名>）

3. WHERE子句的条件表达式的可选格式

（1）

<属性列名> θ { <属性列名> / <常量> / [ANY|ALL] （SELECT语句） }

（2）

<属性列名> [NOT] BETWEEN { <属性列名> / <常量> / （SELECT语句） } AND { <属性列名> / <常量> / （SELECT语句） }

3. WHERE子句的条件表达式的可选格式

（3）

<属性列名> [NOT] IN { （<值1>[，<值2>] …） | （SELECT语句） }

（4） <属性列名> [NOT] LIKE <匹配串>

（5） <属性列名> IS [NOT] NULL

（6） [NOT] EXISTS （SELECT语句）

3. WHERE子句的条件表达式的可选格式

（7）