

实验五 高级数据查询

1. 实验目的

- (1) 掌握连接查询的方法
- (2) 掌握子查询的方法
- (3) 熟悉如何为表和字段取别名
- (4) 掌握如何使用正则表达式查询

2. 实验内容

- (1) 使用连接运算符实现多表查询，包括内连接、外连接和复合条件连接查询；
- (2) 在SELECT语句中嵌套子查询，常使用的操作符有：ANY(SOME)，ALL，IN，EXIST；
- (3) 为字段和表创建别名并在查询中使用
- (4) 在检索或替换某个复合要求的文本内容的语句中使用正则表达式；

3. 实验作业

- (1) 创建数据库EmployDB，在库中创建表employee和表dept。

表employee结构

字段名	字段说明	数据类型	说明
e_no	员工编号	INT	主键，非空
e_name	员工姓名	VARCHAR(20)	非空
e_gender	员工性别	CHAR(20)	非空
dept_no	部门编号	INT	非空
e_job	职位	VARCHAR(20)	非空
e_salary	薪水	SMALLINT	非空
hiredate	入职日期	DATE	允许空值

表dept结构

字段名	字段说明	数据类型	说明
d_no	部门编号	INT	主键，外键，非空
d_name	部门名称	VARCHAR(20)	非空
d_location	部门地址	VARCHAR(20)	非空

(2) 向两个表中插入如下数据。

表employee中记录

e_no	e_name	e_gender	dept_no	e_job	e_salary	hiredate
1001	SMITH	m	20	CLERK	800	2005-11-12
1002	ALLEN	f	30	SALESMAN	1600	2003-05-12
1003	WARD	f	30	SALESMAN	1250	2003-05-12
1004	JONES	m	20	MANAGER	2975	1998-05-18
1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
1006	BLAKE	f	30	MANAGER	2850	1997-02-15
1007	CLARK	m	10	MANAGER	2450	2002-09-12
1008	SCOTT	m	20	ANALYST	3000	2003-05-12
1009	KING	f	10	PRESIDENT	5000	1995-01-01
1010	TURNER	f	30	SALESMAN	1500	1997-10-12
1011	ADAMS	m	20	CLERK	1100	1999-10-05
1012	JAMES	m	30	CLERK	950	2008-06-15

表dept中的记录

d_no	d_name	d_location
10	ACCOUNTING	Shanghai
20	RESEARCH	Beijing
30	SALES	Shenzhen
40	OPERATIONS	Fujian

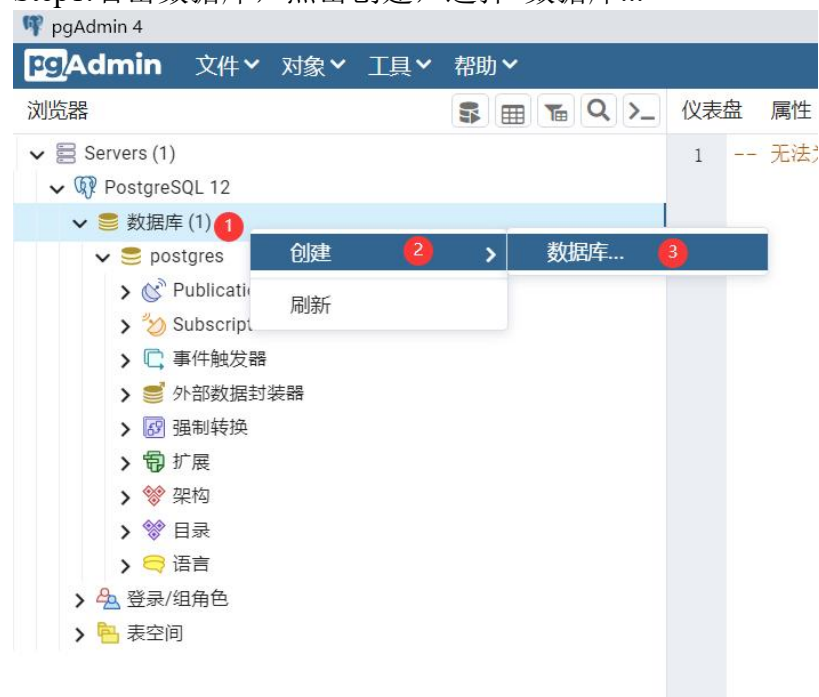
- (3) 查询employee表中所有女员工（F）的薪水。
- (4) 查询employee表中销售人员（SALESMAN）的最低工资。
- (5) 查询employee表中名字以字母N或S结尾的记录。
- (6) 查询员工BLAKE所在部门和部门所在地。
- (7) 查询所有员工的部门和部门信息。
- (8) 查询所有2001到2005年入职的员工信息，查询部门编号为20和30的员工信息并使用UNION合并两个查询结果。
- (9) 使用LIKE查询员工姓名中包含字母a的记录。
- (10) 使用~查询员工姓名中包含T、C或者M三个字母中任意一个的记录。
- (11) 查询employee表中查询员工姓名以字母A或S开头的员工信息。

4. 思考与体会

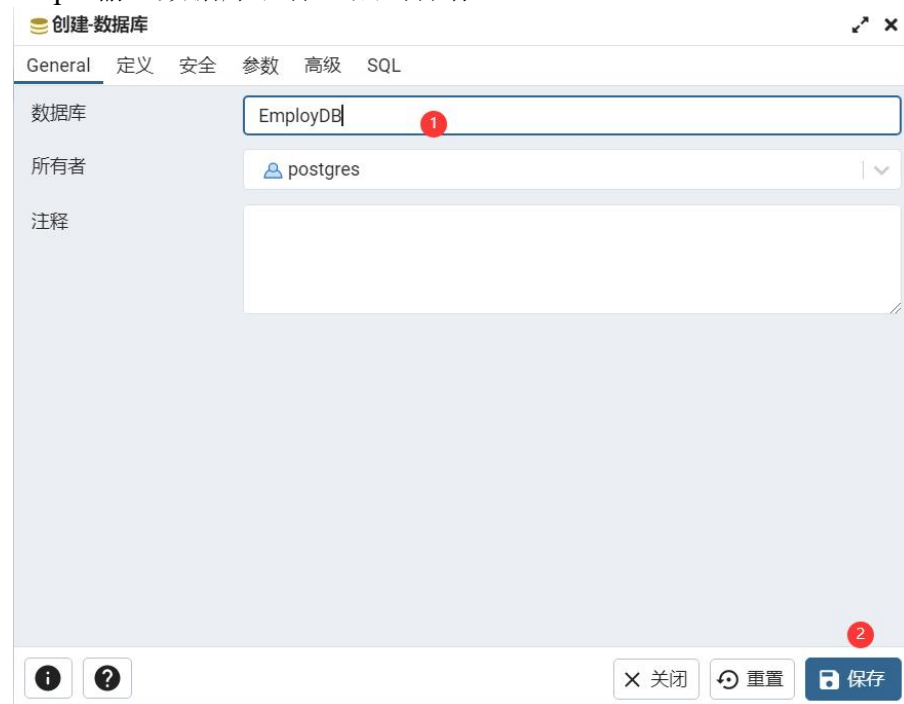
- (1) 在相等联接查询中，在查询某些列时是否可以忽略列前的表名呢？
- (2) ORDER BY可以和LIMIT混合使用吗？如果可以，顺序是怎样的？

实验报告

(1) 创建数据库EmployDB，在库中创建表employee和表dept。
Step1.右击数据库，点击创建，选择“数据库...”



Step2.输入数据库名称，点击保存



Step3.在EmployDB下选择架构，在public右击，选择创建表；



Step4.输入表名称，然后添加属性，点击保存，建立employee表；

创建表

General 列 高级 约束 分区 参数 安全 SQL

名称 ² employee ¹

所有者 postgres

架构 public

表空间 选择一项...

分区表? ☐

注释

创建表

General 列 高级 约束 分区 参数 安全 SQL

继承自表 ¹ 选择要从其继承...

列 ²

名称	数据类型	长度/精度	规模	不为 NUL...	主键?	默认值
<input type="text"/>	选择一项...			<input type="checkbox"/>	<input type="checkbox"/>	<input type="text"/>

³ 填入相应内容

'名称 in 列' cannot be empty.

employee

General 列 高级 约束 参数 安全 SQL

继承自表 选择要从其继承...

列	名称	数据类型	长度/精度	规模	不为 NUL...	主键?	默认值
✎	e_no	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
✎	e_name	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	e_gender	character	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	dept_no	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	e_job	character varying	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	e_salary	smallint			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	hiredate	date			<input type="checkbox"/>	<input type="checkbox"/>	

同样的步骤，建立表dept

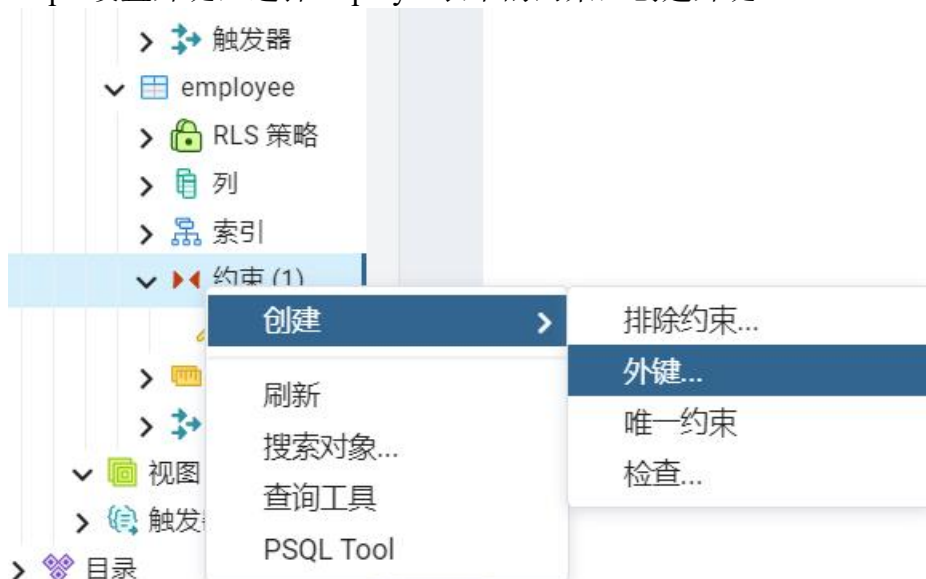
创建表

General 列 高级 约束 分区 参数 安全 SQL

继承自表 选择要从其继承...

列	名称	数据类型	长度/精度	规模	不为 NUL...	主键?	默认值
✎	d_no	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
✎	d_name	character varyi...	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	
✎	d_location	character varyi...	20		<input checked="" type="checkbox"/>	<input type="checkbox"/>	

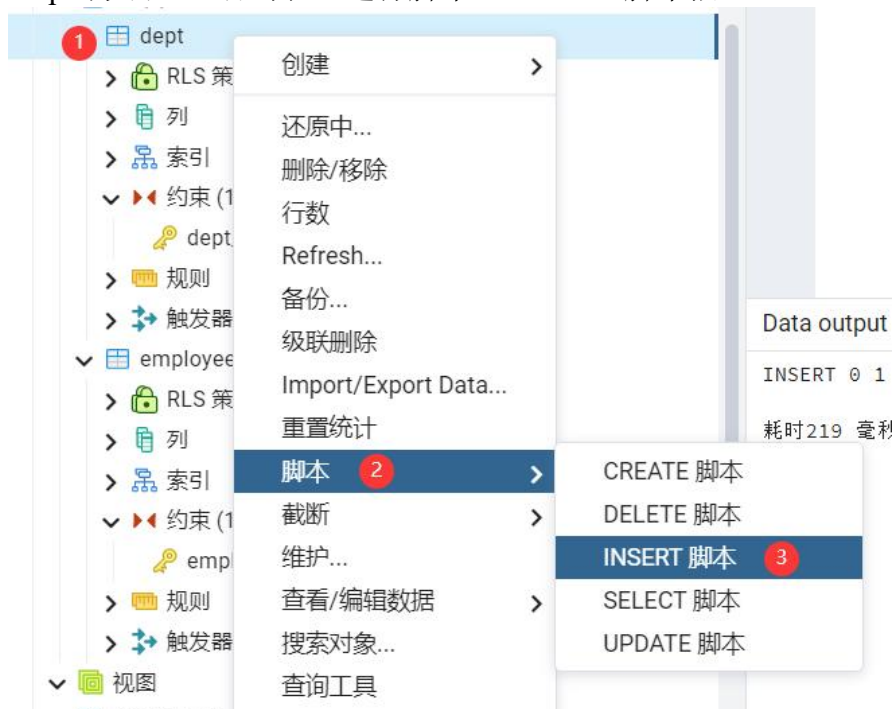
Step5.设置外键，选择employee表下的约束，创建外键





(2) 向表中插入数据。

Step1.方法一，右击表，选择脚本-->INSERT脚本插入



输入信息后点击运行即可

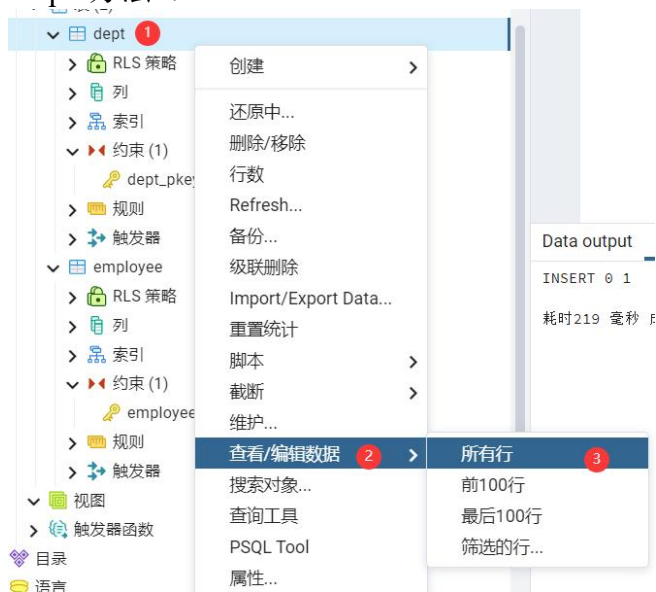
```
1 INSERT INTO public.dept(  
2     d_no, d_name, d_location)  
3     VALUES (10, 'ACCOUNTING', 'Shanghai');
```

显示这个说明插入成功：

INSERT 0 1

耗时219 毫秒 成功返回查询。

Step2.方法2:



点击新建，输入数据，保存即可；

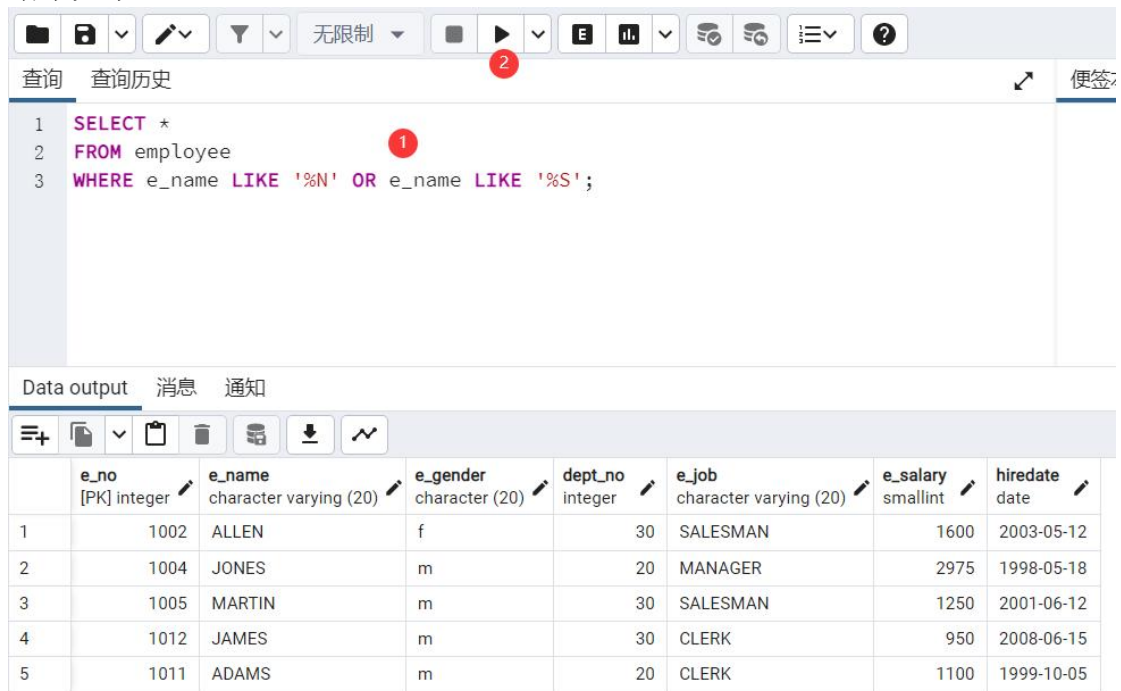
Data output			
消息 通知			
1+ d_no [PK] integer d_name character varying (20) d_location character varying (20)			
1	[null]	[null]	[null]
2	10	ACCOUNTING	Shanghai

Data output			
消息 通知			
1+ d_no [PK] integer d_name character varying (20) d_location character varying (20)			
1	40	OPERATIONS	Fujian
2	30	SALES	Shenzhen
3	20	RESEARCH	Beijing
4	10	ACCOUNTING	Shanghai

✓ 数据保存成功.

Total rows: 1 of 1 Query complete 00:00:01.261 Ln 1, Col 1

(5) 查询employee表中名字以字母N或S结尾的记录；
 代码如下：使用LIKE语句，并使用通配符%（替代0个或多个字符）写出所示格式，“%N”意思就是以N结尾的串；
 结果如下：



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

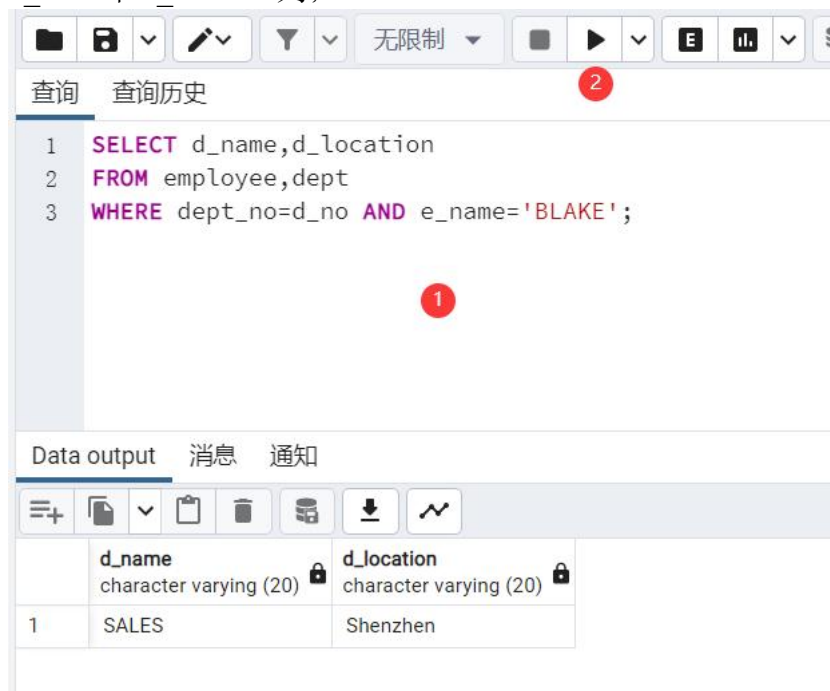
```

1 SELECT *
2 FROM employee
3 WHERE e_name LIKE '%N' OR e_name LIKE '%S';
  
```

Below the query editor, the "Data output" tab is active, displaying a table with 8 columns: e_no, e_name, e_gender, dept_no, e_job, e_salary, and hiredate. The table contains 5 rows of data.

e_no	e_name	e_gender	dept_no	e_job	e_salary	hiredate
1002	ALLEN	f	30	SALESMAN	1600	2003-05-12
1004	JONES	m	20	MANAGER	2975	1998-05-18
1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
1012	JAMES	m	30	CLERK	950	2008-06-15
1011	ADAMS	m	20	CLERK	1100	1999-10-05

(6) 查询员工BLAKE所在部门和部门所在地；
 需要的查询的结果列为d_name和d_location，但是姓名BLAKE在employee表中，而部门信息在dept表中，因此选择对两个表进行笛卡尔积，并选择出部门编号相等的行来（去除不对的匹配），然后选择名字为BLAKE的员工，输出其d_name和d_location列；



The screenshot shows a SQL query editor with a toolbar at the top. The query is as follows:

```

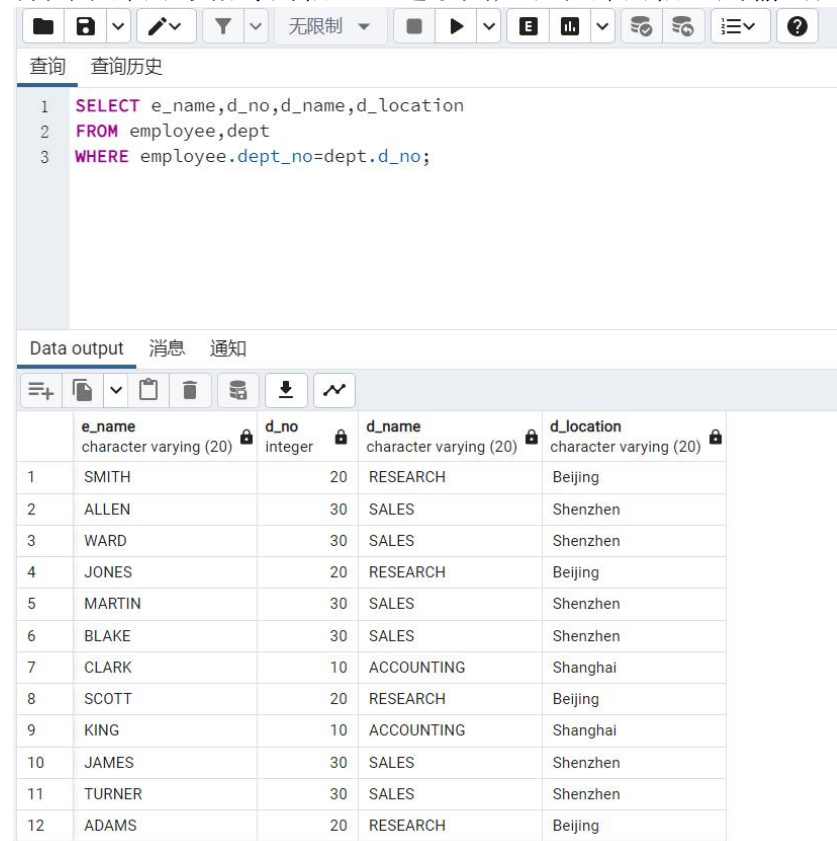
1 SELECT d_name,d_location
2 FROM employee,dept
3 WHERE dept_no=d_no AND e_name='BLAKE';
  
```

Below the query editor, the "Data output" tab is active, displaying a table with 2 columns: d_name and d_location. The table contains 1 row of data.

d_name	d_location
SALES	Shenzhen

(7) 查询所有员工的部门和部门信息；
 部门信息在dept表中，因此需要先进行笛卡尔积运算（等值连接也可），再选

择其中部门号相等的信息，选取其姓名和部门信息列输出；



查询 查询历史

```

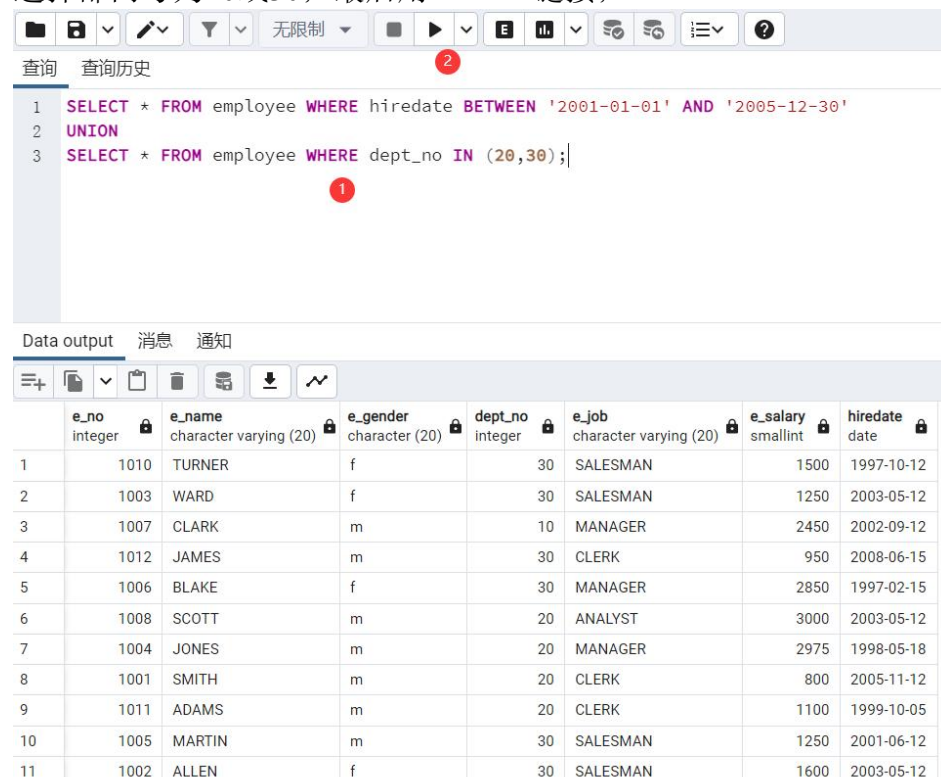
1 SELECT e_name,d_no,d_name,d_location
2 FROM employee,dept
3 WHERE employee.dept_no=dept.d_no;

```

Data output 消息 通知

	e_name character varying (20)	d_no integer	d_name character varying (20)	d_location character varying (20)
1	SMITH	20	RESEARCH	Beijing
2	ALLEN	30	SALES	Shenzhen
3	WARD	30	SALES	Shenzhen
4	JONES	20	RESEARCH	Beijing
5	MARTIN	30	SALES	Shenzhen
6	BLAKE	30	SALES	Shenzhen
7	CLARK	10	ACCOUNTING	Shanghai
8	SCOTT	20	RESEARCH	Beijing
9	KING	10	ACCOUNTING	Shanghai
10	JAMES	30	SALES	Shenzhen
11	TURNER	30	SALES	Shenzhen
12	ADAMS	20	RESEARCH	Beijing

(8) 查询所有2001到2005年入职的员工信息，查询部门编号为20和30的员工信息并用UNION合并两个查询结果；
2001到2005年即雇用日期在2001-01-01到2005-12-30之间，dept_no IN (20,30)即选择部门号为20或30；最后用UNION链接；



查询 查询历史

```

1 SELECT * FROM employee WHERE hiredate BETWEEN '2001-01-01' AND '2005-12-30'
2 UNION
3 SELECT * FROM employee WHERE dept_no IN (20,30);

```

Data output 消息 通知

	e_no integer	e_name character varying (20)	e_gender character (20)	dept_no integer	e_job character varying (20)	e_salary smallint	hiredate date
1	1010	TURNER	f	30	SALESMAN	1500	1997-10-12
2	1003	WARD	f	30	SALESMAN	1250	2003-05-12
3	1007	CLARK	m	10	MANAGER	2450	2002-09-12
4	1012	JAMES	m	30	CLERK	950	2008-06-15
5	1006	BLAKE	f	30	MANAGER	2850	1997-02-15
6	1008	SCOTT	m	20	ANALYST	3000	2003-05-12
7	1004	JONES	m	20	MANAGER	2975	1998-05-18
8	1001	SMITH	m	20	CLERK	800	2005-11-12
9	1011	ADAMS	m	20	CLERK	1100	1999-10-05
10	1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
11	1002	ALLEN	f	30	SALESMAN	1600	2003-05-12

(9) 使用LIKE查询员工姓名中包含字母a的记录；
 由于表中员工姓名均为大写，用
SELECT * FROM employee WHERE e_name LIKE 'a%';
 查询后没有任何结果，因此选择将a改成A进行查询；包含字母A，则需要使用通配符%，包含A，即 ‘%A%’，查询即可；

查询 查询历史

```

1 SELECT *
2 FROM employee
3 WHERE e_name LIKE '%A%';
    
```

Data output 消息 通知

	e_no [PK] integer	e_name character varying (20)	e_gender character (20)	dept_no integer	e_job character varying (20)	e_salary smallint	hiredate date
1	1002	ALLEN	f	30	SALESMAN	1600	2003-05-12
2	1003	WARD	f	30	SALESMAN	1250	2003-05-12
3	1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
4	1006	BLAKE	f	30	MANAGER	2850	1997-02-15
5	1007	CLARK	m	10	MANAGER	2450	2002-09-12
6	1012	JAMES	m	30	CLERK	950	2008-06-15
7	1011	ADAMS	m	20	CLERK	1100	1999-10-05

(10) 使用~查询员工姓名中包含T、C或者M三个字母中任意一个的记录；
 postgresql中使用正则表达式时需要使用 ‘~’，再在后面添加正则表达式，即通配符‘[content]’，意思是在[]里出现的字母都可以识别到字符列的任何单一字符；代码和结果如下图：

查询 查询历史

```

1 SELECT *
2 FROM employee
3 WHERE e_name ~ '[TCM]';
    
```

Data output 消息 通知

	e_no [PK] integer	e_name character varying (20)	e_gender character (20)	dept_no integer	e_job character varying (20)	e_salary smallint	hiredate date
1	1001	SMITH	m	20	CLERK	800	2005-11-12
2	1005	MARTIN	m	30	SALESMAN	1250	2001-06-12
3	1007	CLARK	m	10	MANAGER	2450	2002-09-12
4	1008	SCOTT	m	20	ANALYST	3000	2003-05-12
5	1012	JAMES	m	30	CLERK	950	2008-06-15
6	1010	TURNER	f	30	SALESMAN	1500	1997-10-12
7	1011	ADAMS	m	20	CLERK	1100	1999-10-05

	dept_no [PK] integer	d_some integer	e_no integer
1	10	123	101
2	20	134	102
3	30	234	103
4	40	456	104

查询时写全表名，可以执行：

```

1 SELECT employee.dept_no
2 FROM employee JOIN temp2
3 ON employee.dept_no = temp2.dept_no;

```

选择的表列没有前缀时，会出现同名列冲突的情况：

```

ERROR: column reference "dept_no" is ambiguous
LINE 1: SELECT dept_no
                ^

```

SQL 状态：42702

字符：8

当条件不包含表名时，也会出现冲突

```

1 SELECT employee.dept_no
2 FROM employee JOIN temp2
3 ON dept_no = dept_no;

```

总结：当两个表之间不存在同名列时，此时每列的位置与所属数据表可知，因此可以忽略列前的表名；但查询语句中出现同名列时，必须加上表名加以区分；

(2) ORDER BY可以和LIMIT混合使用吗？如果可以，顺序是怎样的？

可以，但不建议；

当表中的行存在有相同列值时，执行ORDER BY查询和用LIMIT返回的查询结果可能会不一样；

例如，在temp2表中插入有列值相等的行，使用ORDER BY排序后，发现顺序为

查询	查询历史
1	SELECT *
2	FROM public.temp2
3	ORDER BY d_some;

	dept_no [PK] integer	d_some integer	e_no integer
1	60	123	89
2	50	123	80
3	10	123	101
4	20	134	102
5	30	234	103
6	70	234	99
7	80	456	125
8	40	456	104

但是使用LIMIT语句后，可以成功运行，但是输出结果为：

```
1 SELECT *
2 FROM public.temp2
3 ORDER BY d_some LIMIT 2;
```

Data output 消息 通知			
	dept_no [PK] integer	d_some integer	e_no integer
1	10	123	101
2	60	123	89

与排序结果不同（按刚才的结果应该会输出dept_no为60和50）
由于前三个的d_some是一样的，这里再看LIMIT=3的情况：

```
1 SELECT *
2 FROM public.temp2
3 ORDER BY d_some LIMIT 3;
```

Data output 消息 通知			
	dept_no [PK] integer	d_some integer	e_no integer
1	10	123	101
2	60	123	89
3	50	123	80

发现虽然输出的是刚才排序过的前三的，但是顺序发生了变化；
因此，如果ORDER BY时列有相同的值，那么会自由地以任何顺序返回这些行，
如果想让顺序相对确定下来，可以在ORDER BY时添加附加列以消除随机性。

因此两个语句一起使用不会报错，使用顺序是先ORDER BY，再使用LIMIT；
顺序反了会报错：

```
1 SELECT *
2 FROM public.temp2
3 LIMIT 3 ORDER BY d_some ;
```

Data output 消息 通知	
ERROR: syntax error at or near "ORDER" LINE 3: LIMIT 3 ORDER BY d_some ; ^	