《Python统计计算》

（秋季学期）

翟祥
北京林业大学

第9章 SQL

主要内容

- **数据整合**
  - SQL语句介绍
  - 数据纵向合并
  - 数据横向合并

数据整合

数据整合

分析师希望得到信用卡持卡人的人口学信息，如下所示：

card_id	disp_id	issued	type	sex	birth_date
1	9	1998-10-16	金卡	男	1935-10-16
2	19	1998-03-13	普通卡	男	1942-12-28
3	41	1995-09-03	金卡	男	1968-08-27
4	42	1998-11-26	普通卡	男	1935-08-17
5	51	1995-04-24	青年卡	女	1979-12-02
7	56	1998-06-11	普通卡	男	1960-03-31
8	60	1998-05-20	青年卡	男	1980-02-19
9	76	1997-10-25	普通卡	女	1967-10-01
10	77	1996-12-07	普通卡	女	1956-02-18
11	79	1997-10-25	金卡	女	1969-03-10
12	83	1996-09-11	青年卡	女	1978-12-25
13	87	1994-06-29	普通卡	女	1946-11-17

信用卡（CARD）

客户信息（CLIENTS）

SQL语句介绍

SQL语言的基本知识

SQL语言的主要特点

1. Structured Query Language 结构化查询语言。
2. SQL语言类似于英语的自然语言，简洁易用。
3. SQL语言是一种非过程语言，即用户只要提出“干什么”即可，不必管具体操作过程，也不必了解数据的存取路径，只要指明所需的数据即可。
4. SQL语言是一种面向集合的语言，每个命令的操作对象是一个或多个关系，结果也是一个关系。
5. SQL语言既是自含式语言，又是嵌入式语言。可独立使用，也可嵌入到宿主语言中。

SQL语言的基本知识

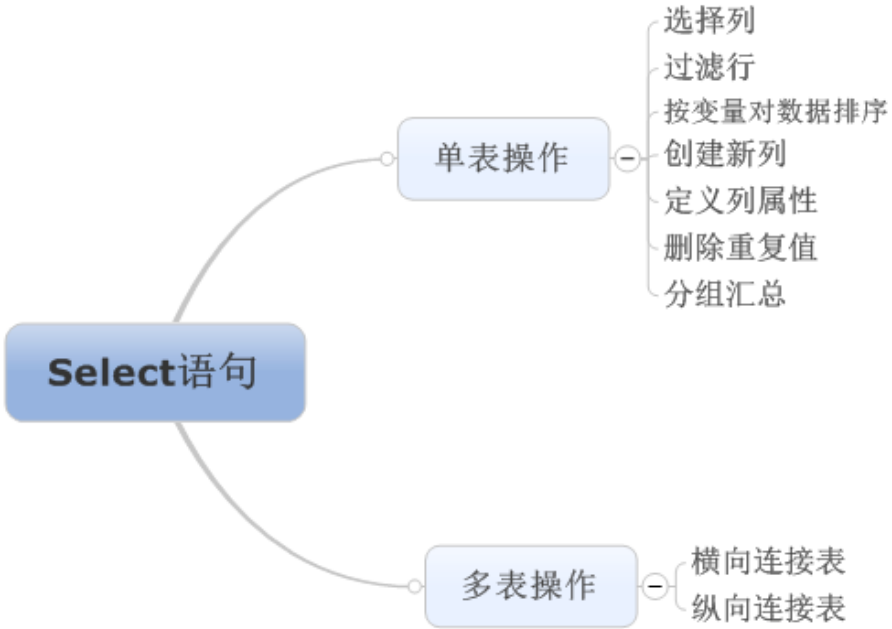
- SQL语句的动词只有九条。

数据定义DDL	CREATE，DROP，ALTER
数据查询DQL	SELECT
数据操纵DML	INSERT，UPDATE，DELETE
数据控制DCL	GRANT，REVOTE

SQL语言的基本知识

Select数据查询语句

- SELECT数据查询是最核心和常用的操作。



SELECT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>[， <表名或视图名>]…

[WHERE <条件表达式>]--

[GROUP BY <列名1>

[HAVING <条件表达式>]]

[ORDER BY <列名2 >[ASC | DESC]]；

选择表中指定列

- 在SELECT语句后指定需要输出的列。

例子：展示销售数据中的年份、市场、销售和利润。

```
select year,market,sale,profit
       from sale;
```

选择表中所有的变量使用“*”作为简写

```
select *
       from sale;
```

选择满足条件的行

- 使用WHERE语句查询结果中满足条件的行。

例子：展示2012年的销售数据。

```
select *
       from sale
       where year=2012;
```

WHERE语句

- 用*WHERE 语句*选择满足特定条件的观测。

- WHERE 语句的一般形式：

WHERE *where-表达式*;

- *where-表达式*是由一系列运算符和操作数组成的用来选择观测的条件表达式。
  - 运算数包含常量和变量。
  - 运算符包含算术运算符，比较算符和逻辑算符。

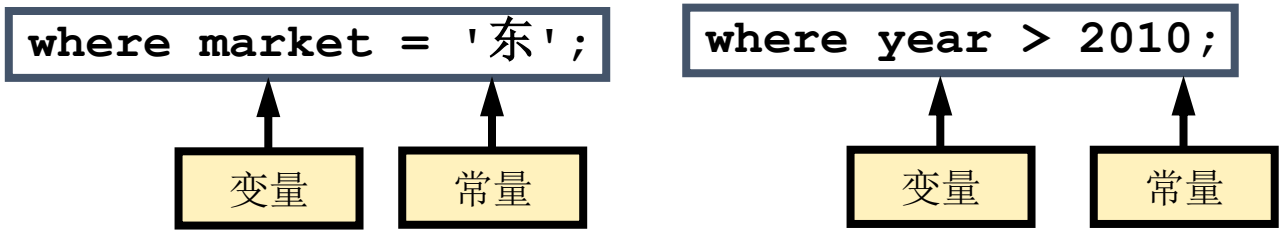
运算数

常量运算数是固定值。

- 字符值必须包含在引号中并且区分大小写。
- 数值不用引号。

一个*变量运算数*必须是来自输入数据集的一个变量。

例如：



比较运算符

- *比较运算符*比较一个变量和一个值，或一个变量和另一个变量。

符号	说明
=	等于
! =	不等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
in	在列表中

比较运算符

- 例如：

```
Where market = '东';
```

```
where year in (2012,2010);
```

算数运算符

- *算数运算符*表示要进行算数计算。

符号	说明
*	乘法
/	除法
+	加法
-	减法

- 例如：

```
where (profit+0.0)/sale < 0.08;
```

逻辑运算符

- *逻辑运算符*合并或修改表达式。

	算符	说明
	AND	与
	OR	或
	NOT	非

- 例如：

```
where year in (2012,2010) and
market='东';
```

对行进行排序

- 使用ORDER BY子句指定行。

例子：按年份排序。

```
select year,market,sale,profit
       from sale
       order by year;
```

- 也可以使用变量在SELECT语句中的序号。

```
select year,market,sale,profit
       from sale
       order by 1;
```

创建新列

•根据上面介绍的表达式创建新变量。

例子：计算利润率。

```
select (profit+0.0)/sale as rate
       from sale;
```

注：加0.0是为了进行简单的整型和浮点型转换

•如果希望保留原始的变量，可以使用以下语句。

```
select a.*,(profit+0.0)/sale as rate
       from sale as a;
```

注：as后面为表的别名

创建新列

- 根据新创建的变量进行筛选。

R中的SQLDF包的写法为：

```
Select a.*,(profit+0.0)/sale as rate from
              sale as a where rate<0.08;
```

注：其他SQL语言要写成如下的形式：

```
Select a.*,(profit+0.0)/sale as rate from
              sale as a where calculated
              rate<0.08;
```

删除重复的行

- 使用DISTINCT关键字删除查询结果中重复的行。

例子：展示销售数据中的不同年份。

```
select DISTINCT year
       from sale;
```

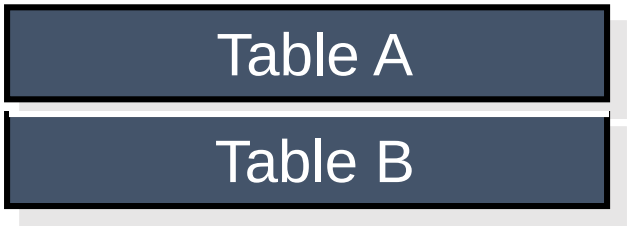
纵向连接表

学习目标

- 纵向连接多张表。
- 比较 SQL 连接和R语言本身的连接方式。

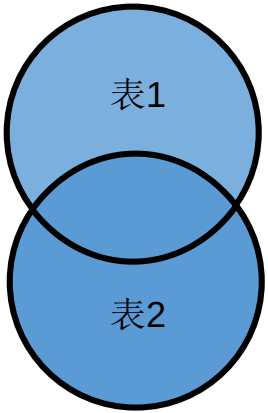
多表的纵向合并

- 在SQL语句中使用集合运算完成。

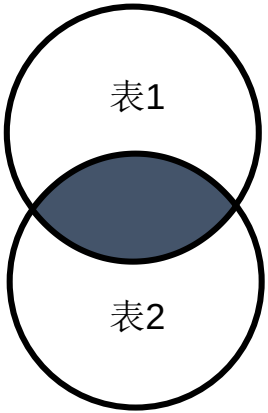


集合查询—并交差

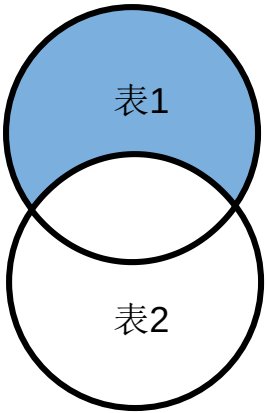
- 将两张表中的记录看作一个集合，则
  - 并集是两张表中重复的记录只保留一份，不重复都保留；
  - 交集是只保留一份重复的记录；
  - 差集是只保留表1中不重复的记录。



并UNION



交INTERSECT

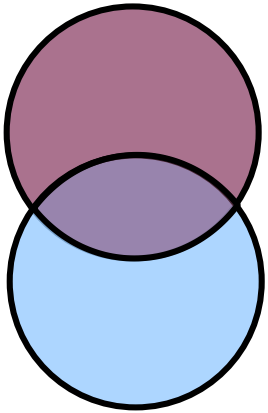


差MINUS

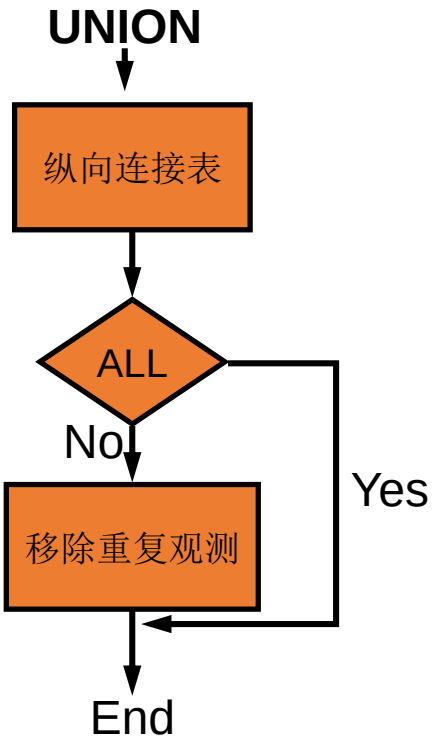
集合操作的类型

- 并集(UNION)
- 并集是将两个表中的同名列合并。

```
select *
   from one
union
select *
   from two;
```



并集操作的流程图



并集操作示例

- SQL 操作会生成一个中间表（intermediate result set）

Table ONE

X	A
1	a
1	a
1	b
2	c
3	v
4	e
6	g

Table TWO

X	B
1	x
2	y
3	z
3	v
5	w

```
select *
   from one
union
select *
   from two;
```

Intermediate Results

❶	❷
1	a
1	a
1	b
1	x
2	c
2	y
3	v
3	v
3	z
4	e
5	w
6	g

并集操作示例

•注意，union后面没有跟随all选项，因此剔除重复值。

Table ONE

X	A
1	a
1	a
1	b
2	c
3	v
4	e
6	g

Table TWO

X	B
1	x
2	y
3	z
3	v
5	w

```
select *
   from one
union
select *
   from two;
```

Intermediate Results

❶	❷
1	a
1	a
1	b
1	x
2	c
2	y
3	v
3	v
3	z
4	e
5	w
6	g

并集操作示例

- 最后结果

Table ONE

X	A
1	a
1	a
1	b
2	c
3	v
4	e
6	g

Table TWO

X	B
1	x
2	y
3	z
3	v
5	w

```
select *
   from one
union
select *
   from two;
```

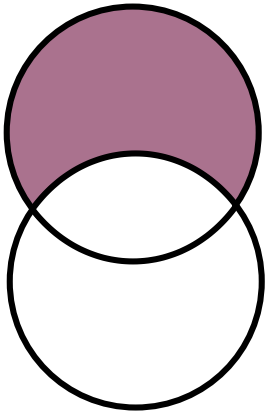
Final Results

X	A
1	a
1	b
1	x
2	c
2	y
3	v
3	z
4	e
5	w
6	g

集合操作的类型

- 差集(EXCEPT)
- 只保留第一张表有，而且第二张表没有的观测。

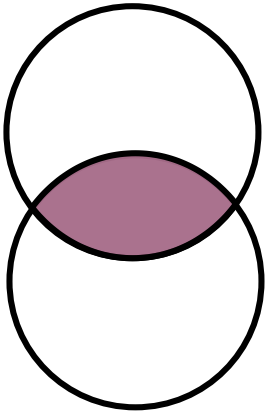
```
select *
   from one
except
select *
   from two;
```



集合操作的类型

- 交集(INTERSECT)
- 只保留两张表都有的观测。

```
select *
   from one
intersect
select *
   from two;
```



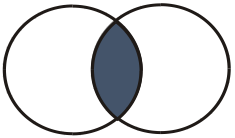
横向连接表

学习目标

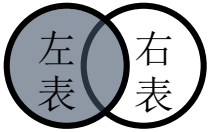
- 根据共同列连接多张表
- 只包括匹配行

横向连接查询基础

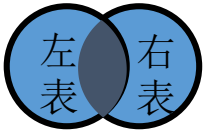
- 交叉连接(cross join，笛卡尔乘积)：查询结果包括两张表观测的所有组合情况，这是SQL实现两表合并的基础，但是极少单独做这种操作；
- 内连接(inner join)：查询结果只包括两张表向匹配的观测，用法简单，但是在数据分析中谨慎使用，因为会造成样本的缺失；



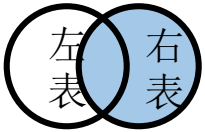
- 外连接(outer join)包括左连接、右连接，全连接。



显示左表所有观测



同时显示两张表的所有观测



显示右表所有观测

笛卡尔积

- 笛卡尔积是横向连接的基础，它将返回两张表内所有观测的组合，令Table1的观测数为n(n=3)， Table2的观测数为m(m=2)，则结果表的观测为n*m(3*2=6)

Table1

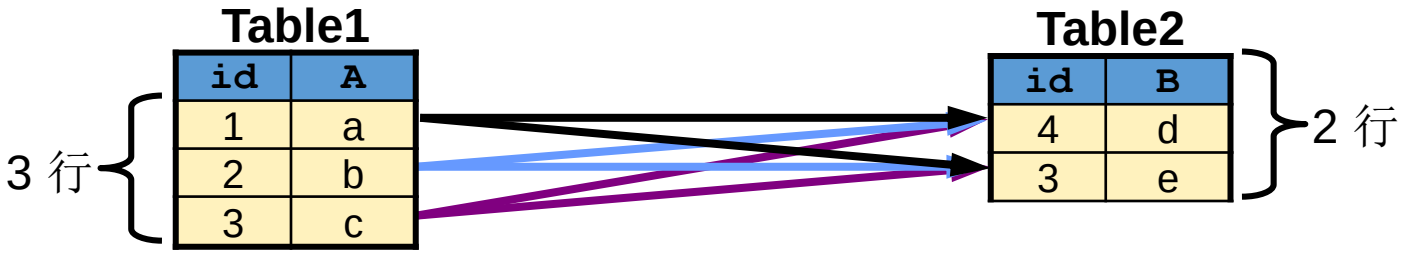
id	A
1	a
2	b
3	c

Table2

id	B
4	d
3	e

```
select *
   from table1, table2;
```

笛卡尔积



结果集

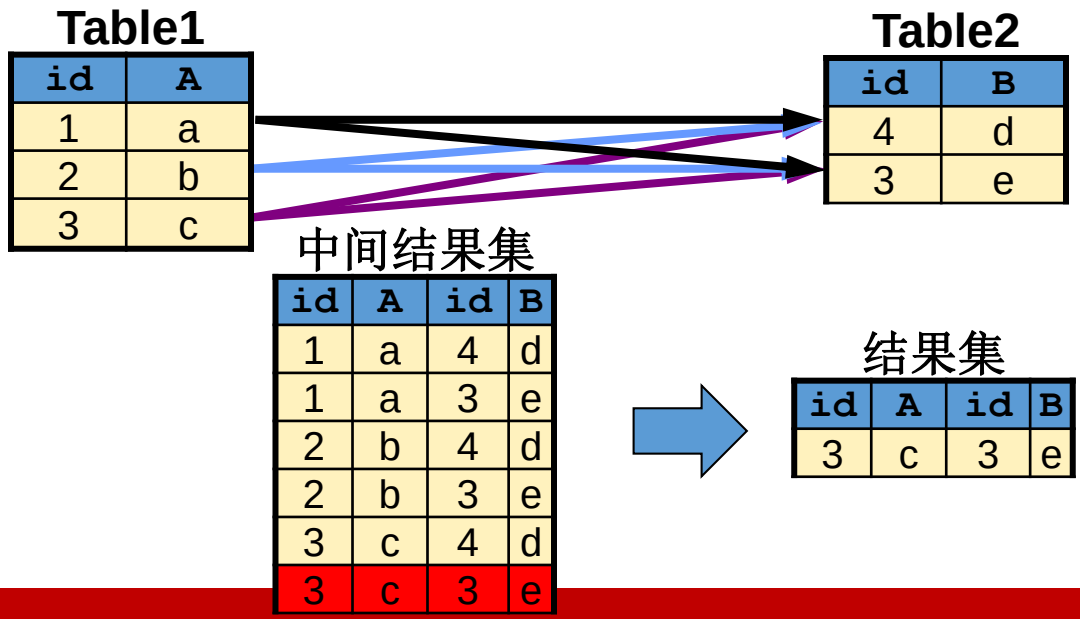
id	A	X	B
1	a	4	d
1	a	3	e
2	b	4	d
2	b	3	e
3	c	4	d
3	c	3	e

6 行

内连接

- 内连接在笛卡尔积基础之上加入了连接条件，其实就是限制条件，这类似单表操作中对观测进行筛选。

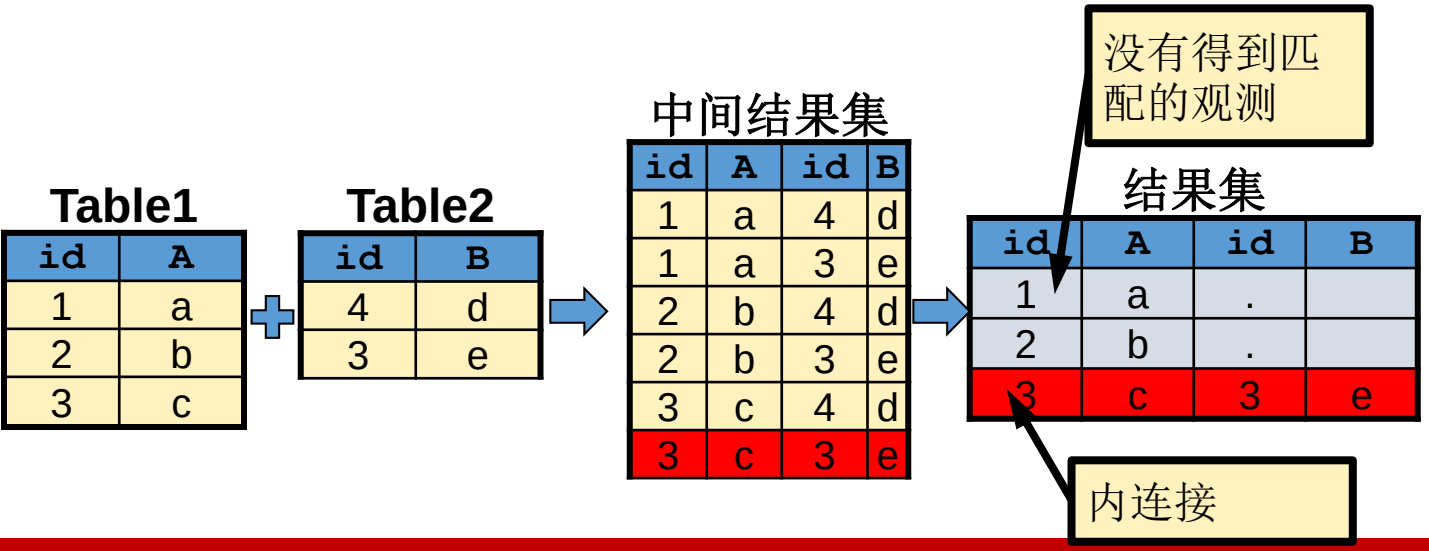
```
select * from table1 as a inner join table2 as b on a.id=b.id
select * from table1 as a,table2 as b where a.id=b.id
```



左连接

- 左连接等价于两部分的叠加：内连接+左表中没有匹配的观测。

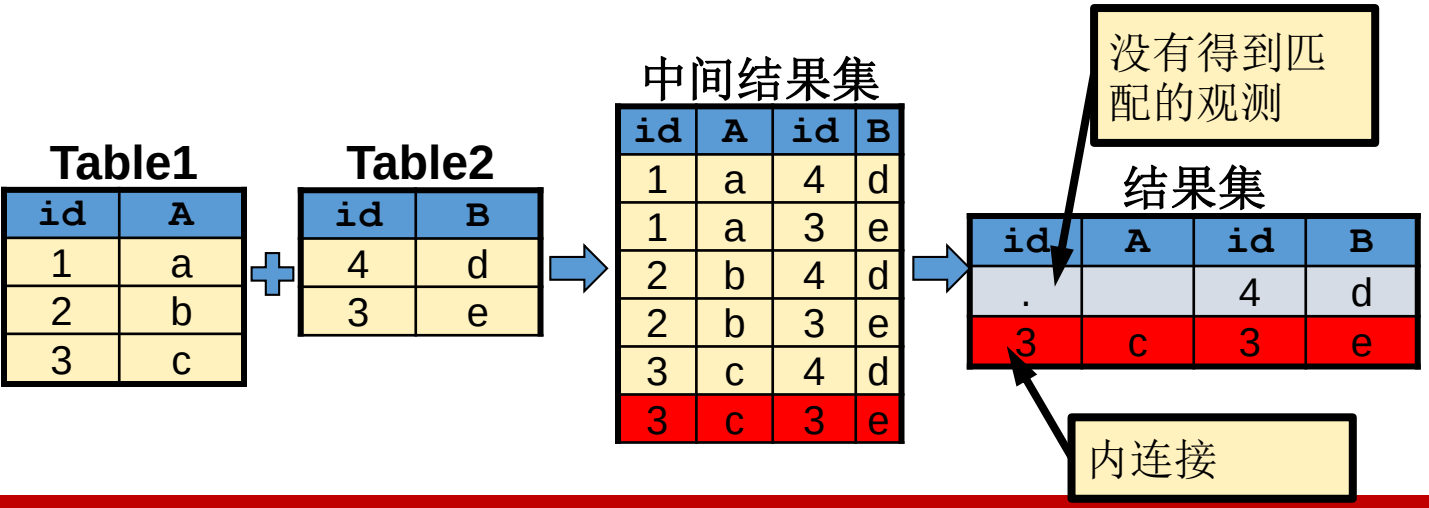
```
select * from table1 as a left join table2 as b on
a.id=b.id
```



右连接

- 右连接等价于两部分的叠加：内连接+右表中没有匹配的观测。

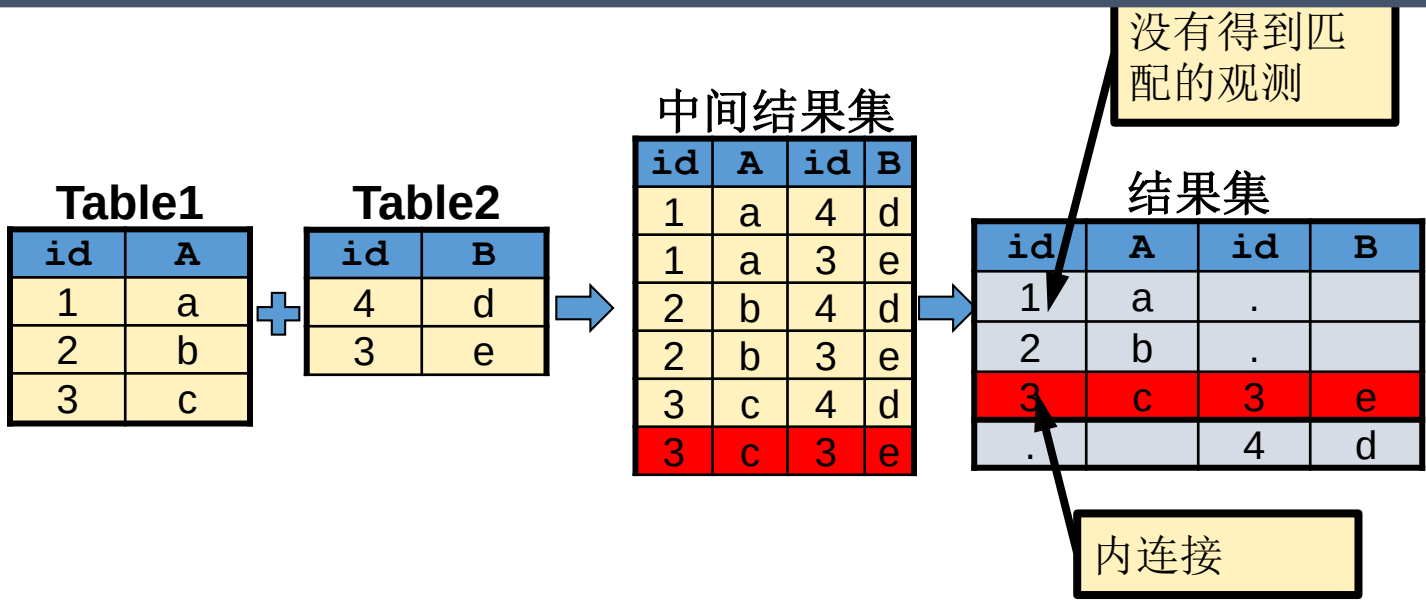
```
select * from table1 as a right join table2 as b on
a.id=b.id
```



全连接

- 全连接等价于三部分的叠加：内连接+左表中没有匹配的观测+右表中没有匹配的观测。

```
select * from table1 as a full join table2 as b on
a.id=b.id
```



谢谢Q/A