

连接



目录

CONTENTS

1

连接的理解

2

外键表

3

同维主子表

4

交叉连接

5

SQL子查询转换成JOIN

01

连接的理解

1. 笛卡尔积

Employee

| ID | NAME | DEPT |
|----|--------|------|
| 1 | David | 1 |
| 2 | Daniel | 2 |
| 3 | Andrew | 1 |



Department

| ID | NAME |
|----|-------|
| 1 | Sales |
| 2 | R&D |



| ID | NAME | DEPT | ID | NAME |
|----|--------|------|----|-------|
| 1 | David | 1 | 1 | Sales |
| 1 | David | 1 | 2 | R&D |
| 2 | Daniel | 2 | 1 | Sales |
| 2 | Daniel | 2 | 2 | R&D |
| 3 | Andrew | 1 | 1 | Sales |
| 3 | Andrew | 1 | 2 | R&D |

2. 条件过滤

| ID | NAME | DEPT | ID | NAME |
|----|--------|------|----|-------|
| 1 | David | 1 | 1 | Sales |
| 4 | David | 4 | 2 | R&D |
| 2 | Daniel | 2 | 4 | Sales |
| 2 | Daniel | 2 | 2 | R&D |
| 3 | Andrew | 1 | 1 | Sales |
| 3 | Andrew | 4 | 2 | R&D |

Employee.DEPT =
Department.ID



| ID | NAME | DEPT | ID | NAME |
|----|--------|------|----|-------|
| 1 | David | 1 | 1 | Sales |
| 2 | Daniel | 2 | 2 | R&D |
| 3 | Andrew | 1 | 1 | Sales |

Employee

| ID | NAME | DEPT |
|----|--------|------|
| 1 | David | 1 |
| 2 | Daniel | 2 |
| 3 | Andrew | 1 |

JOIN



Department

| ID | NAME |
|----|-------|
| 1 | Sales |
| 2 | R&D |



| Employee | Department |
|----------------|------------|
| [1, David, 1] | [1, Sales] |
| [2, Daniel, 2] | [2, R&D] |
| [3, Andrew, 1] | [1, Sales] |

SPL将两个或多个集合连接后，以集合成员构成的二元组为成员，而不是简单的展开所有集合的数据结构拼在一起。SPL的做法不仅更符合JOIN的概念和原意，表间关系更加清晰可见，语法也比SQL更加简洁。

等值JOIN的常见类型

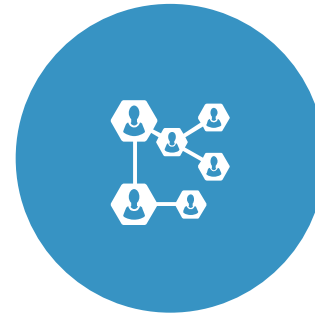
等值JOIN的常见类型



外键表



同维表



主子表

现实中绝大多数JOIN都是等值JOIN，上面三种JOIN已经涵盖了绝大多数等值JOIN的情况。SPL充分利用了这些特征，创造了更简单的书写形式并获得更高效率的运算性能。

外键表

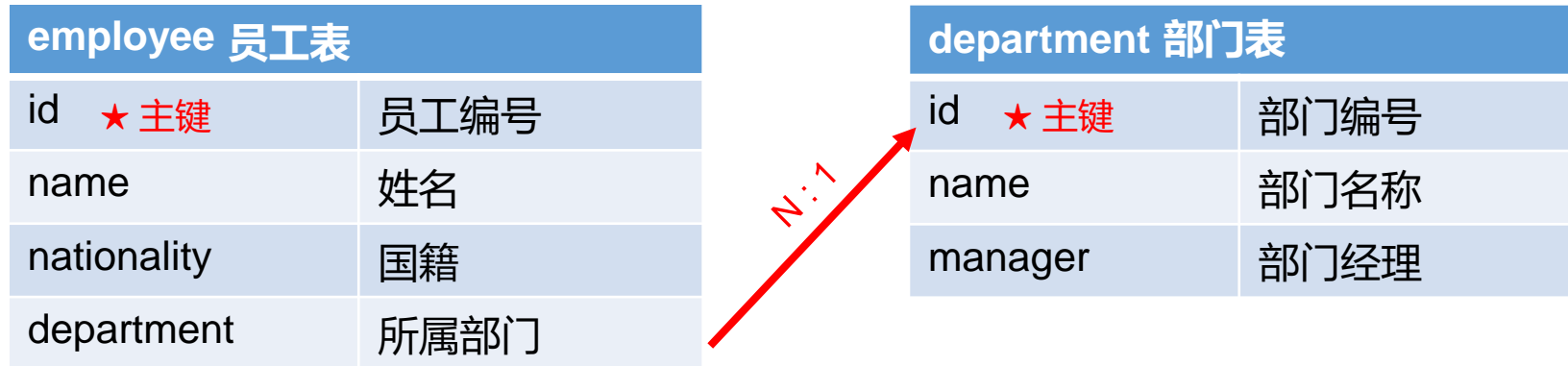


表 A 的某些字段与表 B 的**主键**关联。A 表中与 B 表主键关联的字段称为 A 指向 B 的外键，B 也称为 A 的外键表。外键表是多对一的关系。

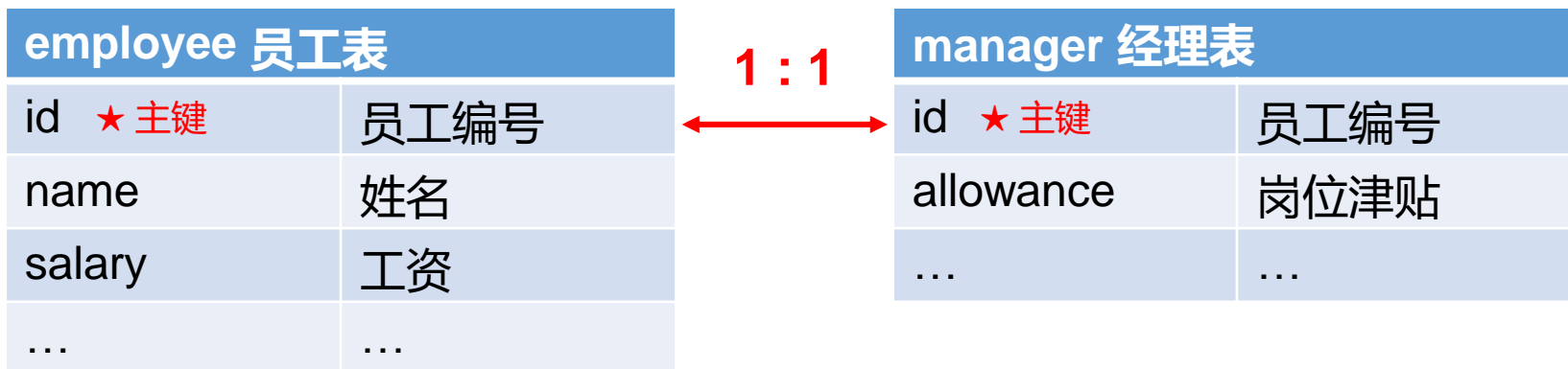


表 A 的**主键**与表 B 的**主键**关联，A 和 B 互称为同维表。同维表是一一对一的关系，同维表之间的关系是对等的。

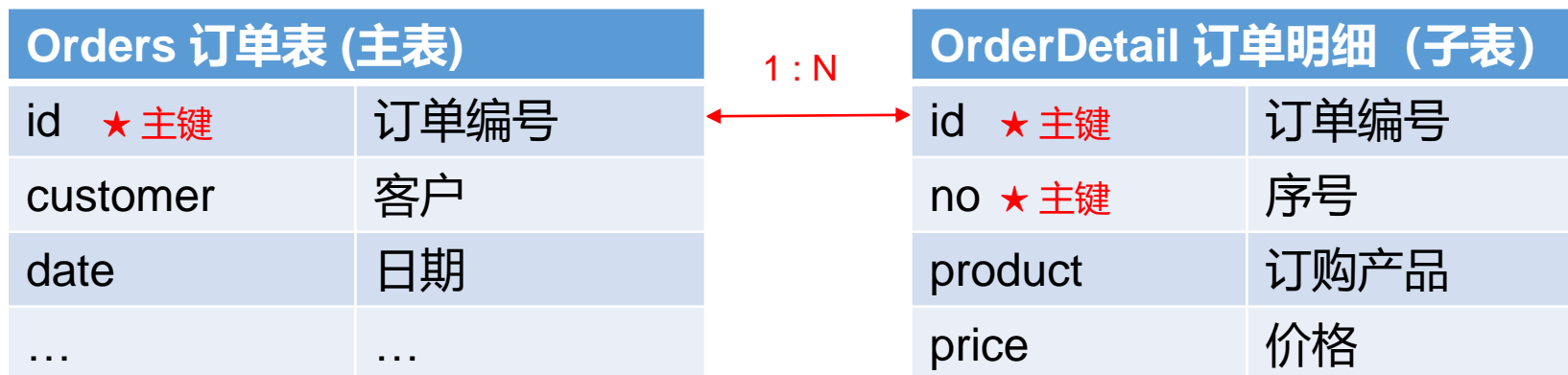


表 A 的**主键**与表 B 的**部分主键**关联，A 称为主表，B 称为子表。主子表是一对多的关系。

使用连接的流程



首先判断连接的类型，常见的类型有：外键表，同维表和主子表。

根据连接类型使用相应的函数进行连接。在后面的章节会详细介绍。

连接以后，可以用“字段.属性”的方式访问其他表的成员。

我们改变对 JOIN 运算的看法，摒弃笛卡尔积的思路，把多表关联运算看成是稍复杂些的单表运算。这样，我们相当于从最常见的等值 JOIN 运算中基本消除了关联，书写和理解都要简单很多。

02

外键表

外键表 - 外键对象化

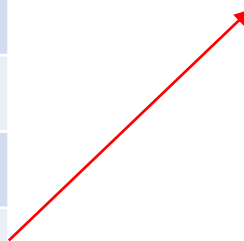
通过**外键对象化**，把外键字段转换成外键表中对应的引用，这样就可以当单表处理。

(1) 单个外键

请选出产品部的所有员工。

| employee 员工表 | |
|--------------|------|
| id ★ 主键 | 员工编号 |
| name | 姓名 |
| nationality | 国籍 |
| department | 所属部门 |

| department 部门表 | |
|----------------|------|
| id ★ 主键 | 部门编号 |
| name | 部门名称 |
| manager | 部门经理 |



外键表 - 外键对象化

员工表

| ID | NAME | NATIONALITY | DEPARTMENT |
|-----|------|-------------|------------|
| 103 | Rudy | American | Product |



| ID | NAME | NATIONALITY | DEPARTMENT |
|-----|------|-------------|-----------------------|
| 103 | Rudy | American | [11, Product, Robert] |

| ID | NAME | MANAGER |
|----|---------|---------|
| 11 | Product | Robert |

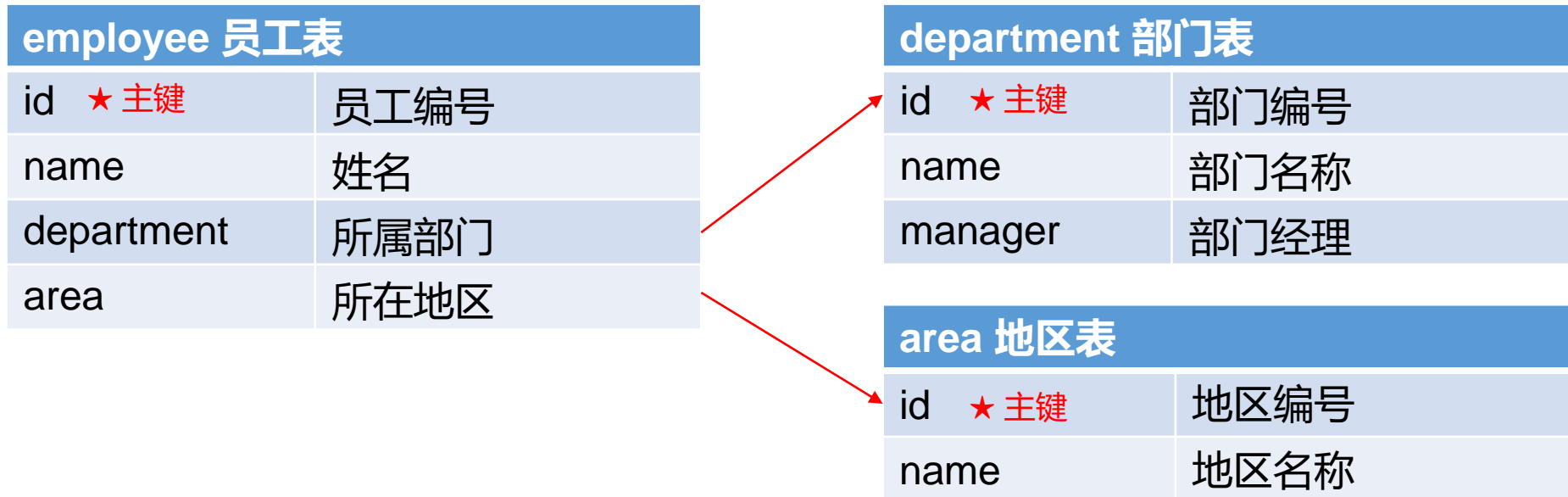
经过外键对象化，员工表中的department.name 即是“所属部门的名称”。完整SPL如下：

| | A | B |
|---|-------------------------------------|--|
| 1 | =db.query("select * from employee") | =db.query("select * from department") |
| 2 | =A1.switch(department, B1:id) | =A2.select(department.name=="Product") |

外键表 - 外键对象化

(2) 单层多个外键

员工表的 area 字段又是指向地区表的外键。请选出在北京的产品部员工。



外键表 - 外键对象化

EMPLOYEE

| ID | NAME | DEPARTMENT | AREA |
|-----|------|------------|------|
| 103 | Rudy | 11 | 101 |



外键对象化



| ID | NAME | DEPARTMENT | AREA |
|-----|------|---------------------|----------------|
| 103 | Rudy | [11,Product,Robert] | [101, Beijing] |

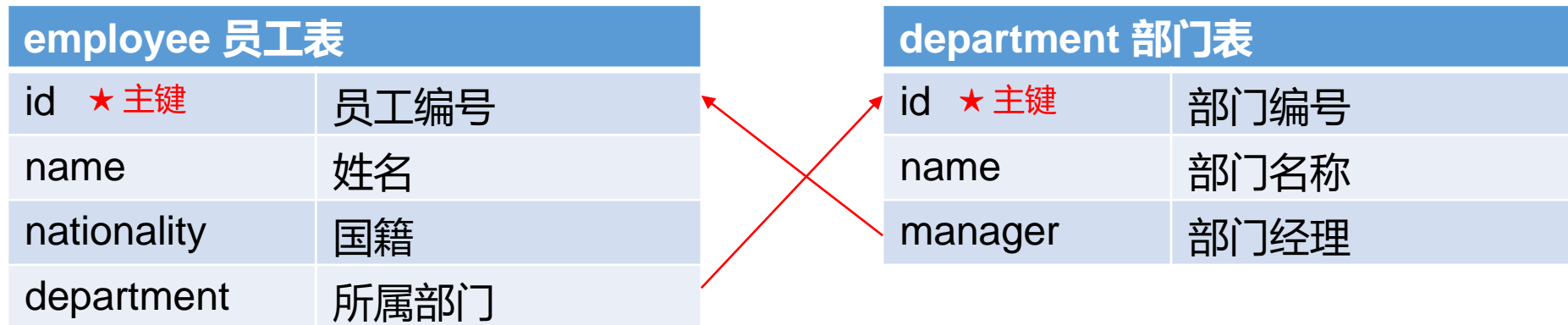
完整SPL如下:

| | A | B |
|---|--|---|
| 1 | =db.query("select * from employee") | =db.query("select * from department") |
| 2 | =db.query("select * from area") | |
| 3 | =A1.switch(department, B1:id; area,A2:id) | =A3.select(department.name=="Product" && area.name=="Beijing") |

外键表 - 外键对象化

(3) 多层外键

哪些美国籍员工有一个中国籍经理？



外键表 - 外键对象化

对于多层外键连接，只需要逐层进行外键对象化。

1

department

| ID | NAME | MANAGER |
|----|---------|---------|
| 11 | Product | Robert |

外键对象化



| ID | NAME | MANAGER |
|----|---------|---------------------------------|
| 11 | Product | [101, Robert, Chinese, Product] |

2

employee

| ID | NAME | NATIONALITY | DEPARTMENT |
|-----|------|-------------|------------|
| 103 | Rudy | American | Product |

外键对象化



| ID | NAME | NATIONALITY | DEPARTMENT |
|-----|------|-------------|---|
| 103 | Rudy | American | [11,Product,[101,Robert,Chinese,Product]] |

外键表 - 外键对象化

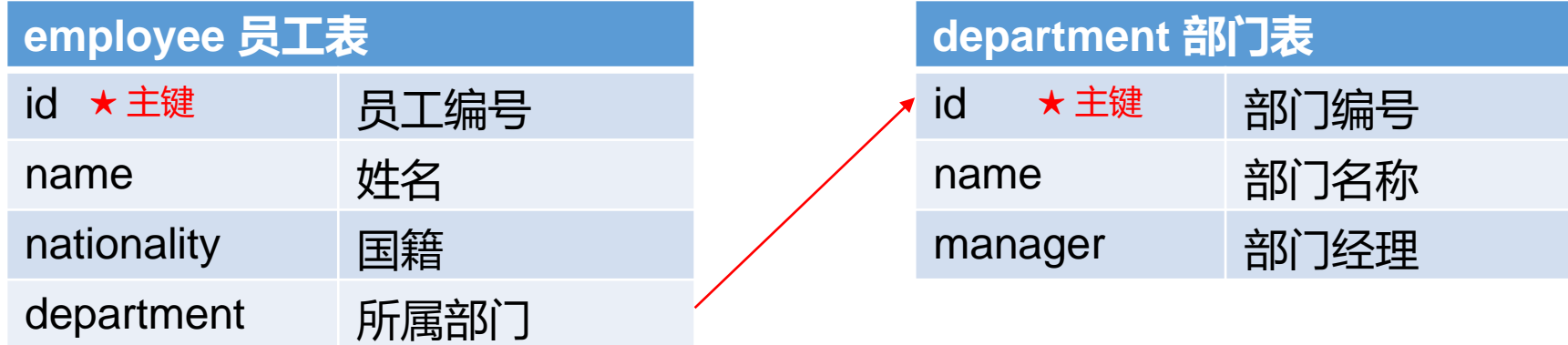
完整SPL如下：

| | A | B |
|---|---|--|
| 1 | <code>=db.query("select * from employee")</code> | <code>=db.query("select * from department")</code> |
| 2 | <code>=B1.switch(manager, A1:id)</code> | <code>=A1.switch(department, A2:id)</code> |
| 3 | <code>=B2.select(nationality=="American" && department.manager.nationality=="Chinese")</code> | |

外键表 - 外键对象化不支持的情况

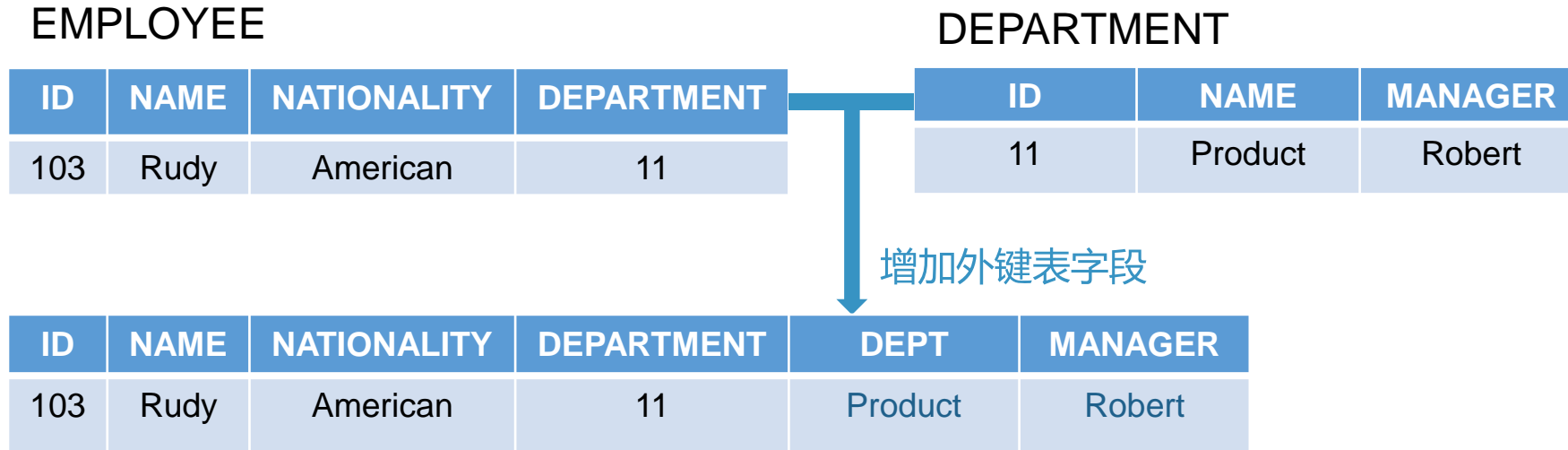
(1) 外键指向空记录

当所属部门的指向为空时，对象化会使该字段的代码值丢失。



外键表 - 外键对象化不支持的情况

解决方法一：不使用外键对象化，增加需要的外键表字段。

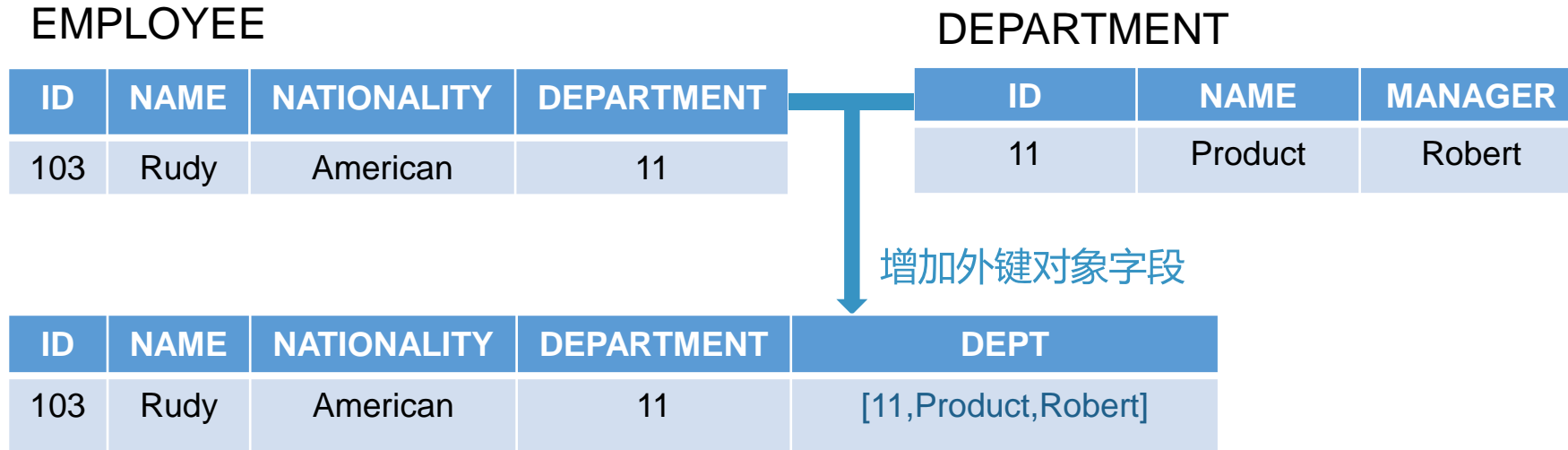


完整SPL如下：

| | A | B |
|---|--|---|
| 1 | <code>=db.query("select * from employee")</code> | <code>=db.query("select * from department")</code> |
| 2 | <code>=A1.join(department, B1:id, name:dept, manager)</code> | <code>=A2.select(department.name=="Product")</code> |

外键表 - 外键对象化不支持的情况

解决方法二：仍然外键对象化，但是外键对象产生在新增字段上。



完整SPL如下：

| | A | B |
|---|-------------------------------------|---------------------------------------|
| 1 | =db.query("select * from employee") | =db.query("select * from department") |
| 2 | =A1.join(department, B1:id, ~:dept) | =A2.select(department.name=="产品部") |

外键表 - 外键对象化不支持的情况

(2) 多字段外键

请选出2018年回款的价格超过500的订购产品名称。



外键表 - 外键对象化不支持的情况

1

OrderDetail

| ID | ORDER | PRODUCT | PRICE |
|-------|-------|---------|-------|
| 10248 | 1 | 17 | 238 |
| 10248 | 2 | 18 | 475 |

外键对象化

OrderDetail

| ID | ORDER | PRODUCT | PRICE |
|-------|-------|-------------|-------|
| 10248 | 1 | [17, cake] | 238 |
| 10248 | 2 | [18, apple] | 475 |

2

OrderPayment

| ID | ORDER | ORDERNO | DATE |
|-----|-------|---------|------------|
| 101 | 10248 | 1 | 2012-07-26 |
| 103 | 10248 | 2 | 2012-08-15 |

增加外键对象

OrderPayment

| ID | ORDER | ORDERNO | DATE | DETAIL |
|-----|-------|---------|------------|------------------------------|
| 101 | 10248 | 1 | 2012-07-26 | [10248, 1, [17, cake], 238] |
| 103 | 10248 | 2 | 2012-08-15 | [10248, 2, [18, apple], 475] |

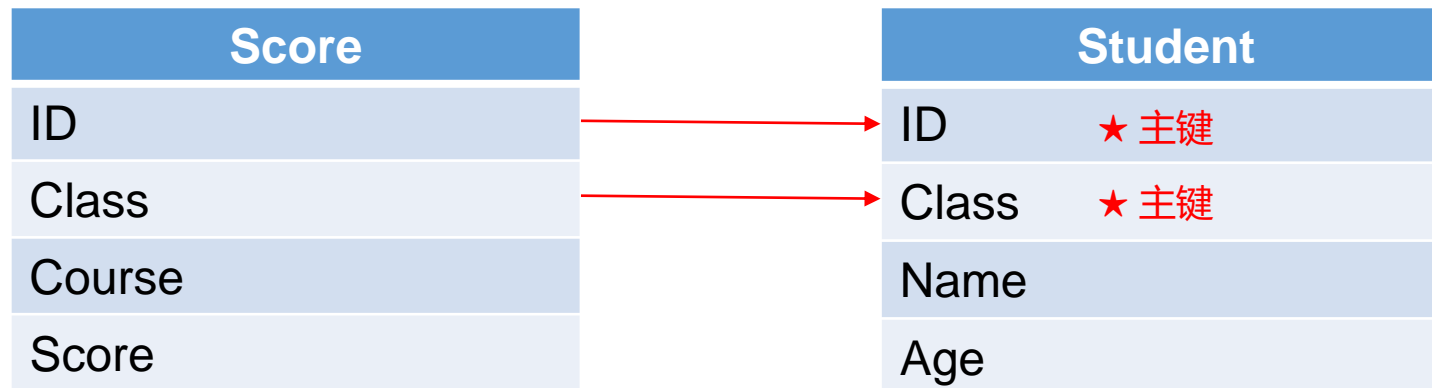
外键表 - 外键对象化不支持的情况

完整SPL如下：

| | A | B |
|---|--|---|
| 1 | <code>=db.query("select * from OrderPayment")</code> | <code>=db.query("select * from OrderDetail")</code> |
| 2 | <code>=db.query("select * from Product")</code> | <code>=B1.switch(product, A2:id)</code> |
| 3 | <code>=A1.join(order:orderno,B2:id:no,~:detail)</code> | <code>=A3.select(year(date)==2018)</code> |
| 4 | <code>=B3.select(detail.price>500)</code> | <code>=A4.new(id,date,detail.product.name:name,detail.price:price)</code> |

外键表 - 外键表有条件的情况

查询一班的学生成绩。



外键表 - 外键表有条件的情况

第一种方法：先对外键表过滤，再进行连接过滤。

1

Student

| ID | Class | Name | Age |
|--------------|--------------------|-------------------|---------------|
| 1 | Class 1 | David | 16 |
| 2 | Class 2 | Daniel | 17 |

过滤

Student

| ID | Class | Name | Age |
|----|---------|-------|-----|
| 1 | Class 1 | David | 16 |

2

Score

| ID | Class | Course | Score |
|--------------|--------------------|--------------------|---------------|
| 1 | Class 1 | Math | 89 |
| 2 | Class 2 | English | 95 |

连接过滤

Score

| ID | Class | Course | Score |
|----|---------|--------|-------|
| 1 | Class 1 | Math | 89 |

外键表 - 外键表有条件的情况

完整SPL如下:

| | A |
|---|------------------------------------|
| 1 | =db.query("select * from Score") |
| 2 | =db.query("select * from Student") |
| 3 | =A2.select(Class=="Class 1") |
| 4 | =A1.join@i(ID, A3:ID) |

A3: 对班级进行过滤以后, 多主键连接 (班级+学生ID) 变成单主键连接 (学生ID) 。

A4: 使用了join@i连接选项, 找不到对应值时删除该记录, 用于连接过滤。

外键表 - 外键表有条件的情况

第二种方法：如果条件是针对主键的，还可以使用多字段连接过滤。

| Score | | | | Student | | | |
|--------------|--------------------|--------------------|---------------|--------------|--------------------|-------------------|---------------|
| ID | Class | Course | Score | ID | Class | Name | Age |
| 1 | Class 1 | Math | 89 | 1 | Class 1 | David | 16 |
| 1 | Class 1 | English | 94 | 2 | Class-2 | Daniel | 17 |
| 2 | Class-2 | Math | 92 | ... | ... | ... | ... |
| 2 | Class-2 | English | 95 | | | | |
| ... | ... | ... | ... | | | | |

连接过滤

连接字段是学生ID+班级两个字段，可以在连接时把班级指定为常数条件（一班），从而实现连接过滤。

外键表 - 外键表有条件的情况

完整SPL如下：

| | A |
|---|---------------------------------------|
| 1 | =db.query("select * from Score") |
| 2 | =db.query("select * from Student") |
| 3 | =A1.join@i(ID:"Class 1", A2:ID:Class) |

直接多主键连接，可以保持学生表的索引，从而在连接时保证速度。

03

同维主子表



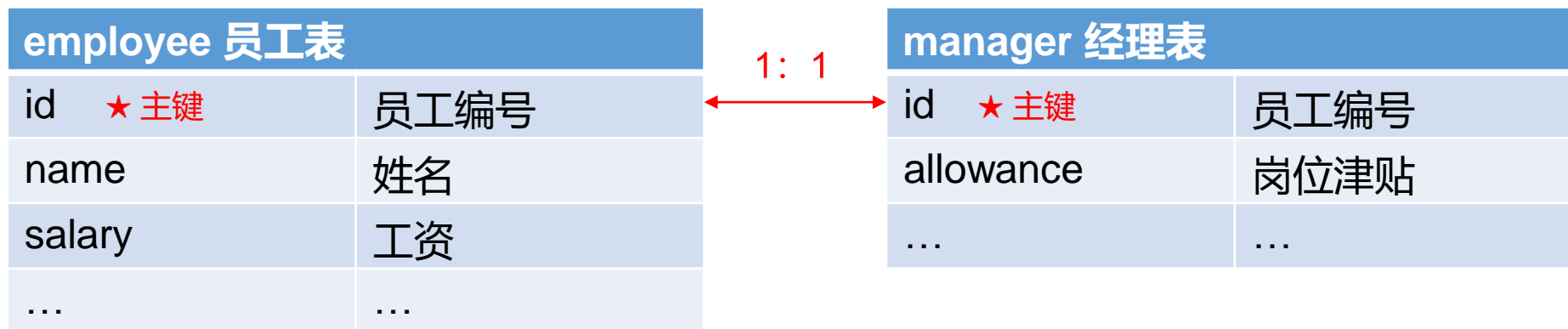
同维表和主子表

同维表和主子表有很多相似之处，所以把它们放在同一章节讲解。

同维表之间的关系是对等的，主子表只要转化为同维表以后，就可以当作单表来运算了。

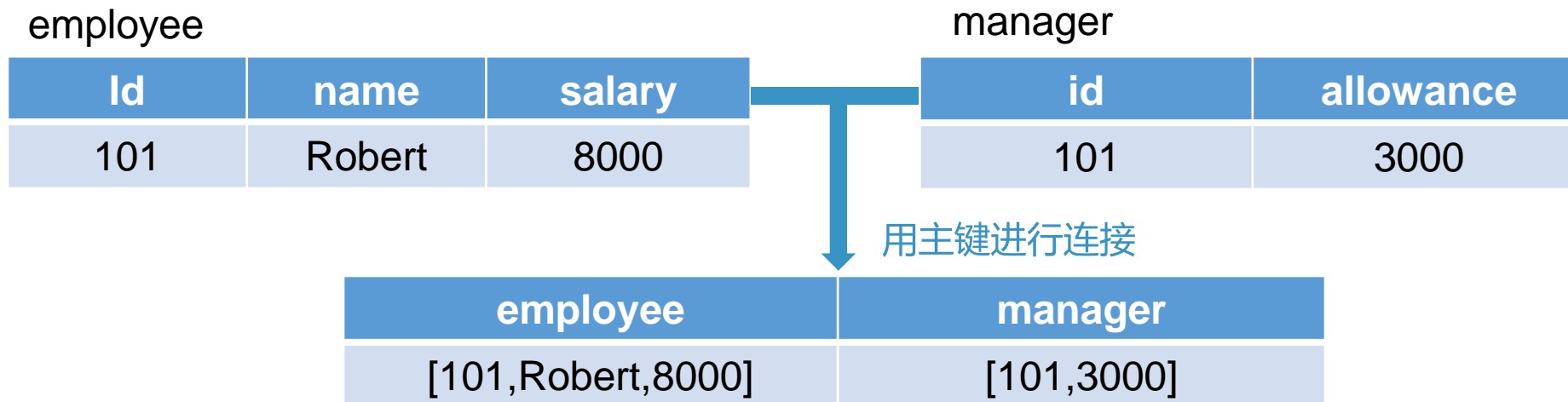
3.1 同维表

要统计所有员工（包括经理）的总收入（加上津贴）。



员工表用来存储员工信息。经理也是员工，经理会比普通员工多一些属性，另用一个经理表来保存。两表共用同样的员工编号。

遇到同维表类型，直接按照主键进行连接。

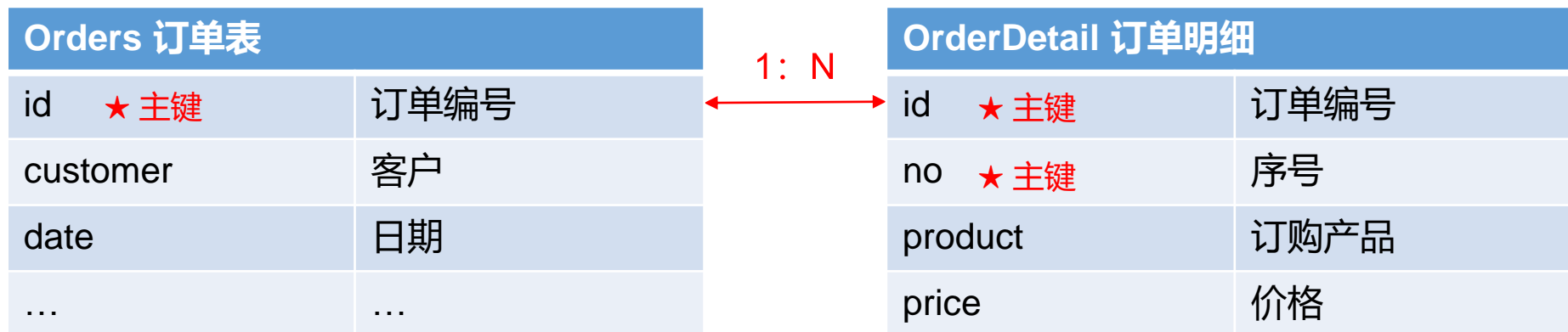


完整SPL如下：

| | A | B |
|---|-------------------------------------|--|
| 1 | =db.query("select * from employee") | =db.query("select * from manager") |
| 2 | join(A1:employee,id;B1:manager,id) | =A2.new(employee.id,employee.name,employee.salary+manager.allowance) |

3.2 单个主子表

例：订单表的主键是 id，订单明细表中的主键是 id 和 no，前者的主键是后者的一部分。
现在要计算每张订单的总金额。



主子表关系是不对等的，从子表引用主表则和外键表类似。
这里我们主要介绍从主表引用子表的情况。



完整SPL如下:

| | A | B |
|---|-----------------------------------|---|
| 1 | =db.query("select * from Orders") | =db.query("select * from OrderDetail") |
| 2 | =join(A1:Order,id;B1:Detail,id) | =A2.groups(Order.id:order,Order.customer:customer; sum(Detail.price):price) |

如果订单和订单明细针对id都是有序的, 我们可以使用join@m进行有序归并, 从而提高join的速度与性能。

| | A |
|---|-----------------------------------|
| 2 | =join@m(A1:Order,id;B1:Detail,id) |

3.3 多个主子表

假如订单表还有一个子表用于记录回款情况。我们现在想知道哪些订单还没有回款，也就是累计回款金额小于订单总金额的订单。



简单地把这三个表 JOIN 起来是不对的，订单明细和订单回款表会发生多对多的关系。

Id 在订单明细表和订单回款表中都不是唯一主键，如何把它们转变成具有唯一性的事实主键？

1

OrderDetail

| id | no | product | price |
|-------|----|---------|-------|
| 10248 | 1 | 17 | 238 |
| 10248 | 2 | 18 | 475 |

子表按id分组

| member |
|-------------------------------------|
| [[10248,1,17,238],[10248,2,18,475]] |

id变成主键

OrderPayment

| id | date | amount |
|-------|------------|--------|
| 10248 | 2012-07-26 | 238 |
| 10248 | 2012-08-15 | 475 |

子表按id分组

| member |
|---|
| [[10248,2012-07-26,238],[10248,2012-08-15,475]] |

id变成主键

2

Orders

| id | customer | date |
|-------|----------|------------|
| 10248 | VINET | 2012-07-04 |

用主表主键 id 连接

| order | detail | payment |
|---------------------------|------------------------|------------------------------|
| [10248,VINET, 2012-07-04] | [[10248,1,17,238],...] | [[10248,2012-07-26,238],...] |

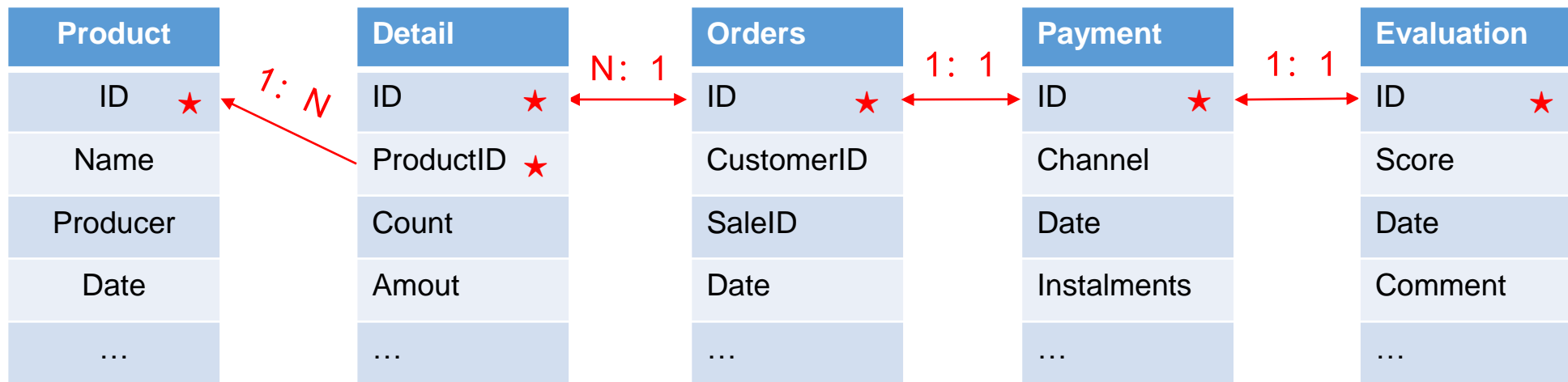
完整SPL如下:

| | A | B |
|---|---|---|
| 1 | =db.query("select * from Orders") | =db.query("select * from OrderDetail") |
| 2 | =db.query("select * from OrderPayment") | |
| 3 | =B1.group(id) | =A2.group(id) |
| 4 | =join(A1:Orders,id; A3:Detail, id; B3:Payment, id) | =A4.new(Orders.id:id,Orders.product:p roduct,Detail.sum(price):price,Paymen t.sum(amount):amount) |
| 5 | =B4.select(amount<price) | |

子表对着主表主键 id 分组后, id 就变成了事实上的主键。之后我们就可以把同维主子表当作同维表来处理了。

3.4 外键表、同维表、主子表混合

查询商品名称中包含“water”，在2019年1月份下单，订单总金额大于200元，没有使用分期，得到5星好评的订单信息。



这个例子中既包含同维表和主子表，也有外键表。需要把问题拆开来解决：先把外键对象化，子表再按主表主键分组，最后都是同维表了连接起来。

1

| ID | ProductID | Count | Amout |
|-------|-----------|-------|-------|
| 10248 | 1 | 17 | 238 |

外键对象化

| ID | ProductID | Count | Amout |
|-------|----------------|-------|-------|
| 10248 | [1, cake,...] | 17 | 238 |

2

| ID | ProductID | Count | Amout |
|-------|----------------|-------|-------|
| 10248 | [1,cake,...] | 17 | 238 |
| 10248 | [2,apple,...] | 18 | 475 |

子表按id分组

| member | | | |
|---|--|--|--|
| [[10248,[1,...],17,238],[10248,[2,...],18,475]] | | | |

3



用主表主键连接

| Orders | Detail | Payment | Evaluation |
|---------------------------|---|------------------------|-----------------------------|
| [10248,VINET, 2012-07-04] | [[10248,[1,...],17,238],[10248,[2,...],18,475]] | [10248,3,2012-07-04,0] | [10248,5,2012-07-16,"Good"] |

完整SPL如下:

| | A | B |
|---|--|--|
| 1 | =db.query("select * from Orders") | =A1.select(year(Date)==2019&&month(Date)==1) |
| 2 | =db.query("select * from Product") | =A2.select(like(Name, "*water*")) |
| 3 | =db.query("select * from Detail") | =A3.switch @i(ProductID,B2:ProductID) |
| 4 | =B3.group(ID) | =A4.select(sum(Amount)>=200) |
| 5 | =db.query("select * from Payment") | =A3.select(Instalments==0) |
| 6 | =db.query("select * from Evaluation") | =A4.select(Score==5) |
| 7 | =join(B1:Orders,ID;B4:Detail,ID;B5:Payment, ID;B6:Evaluation,ID) | |

04

交叉连接

4.1 非等值连接

请列出社区居民所处的年龄段。

Community

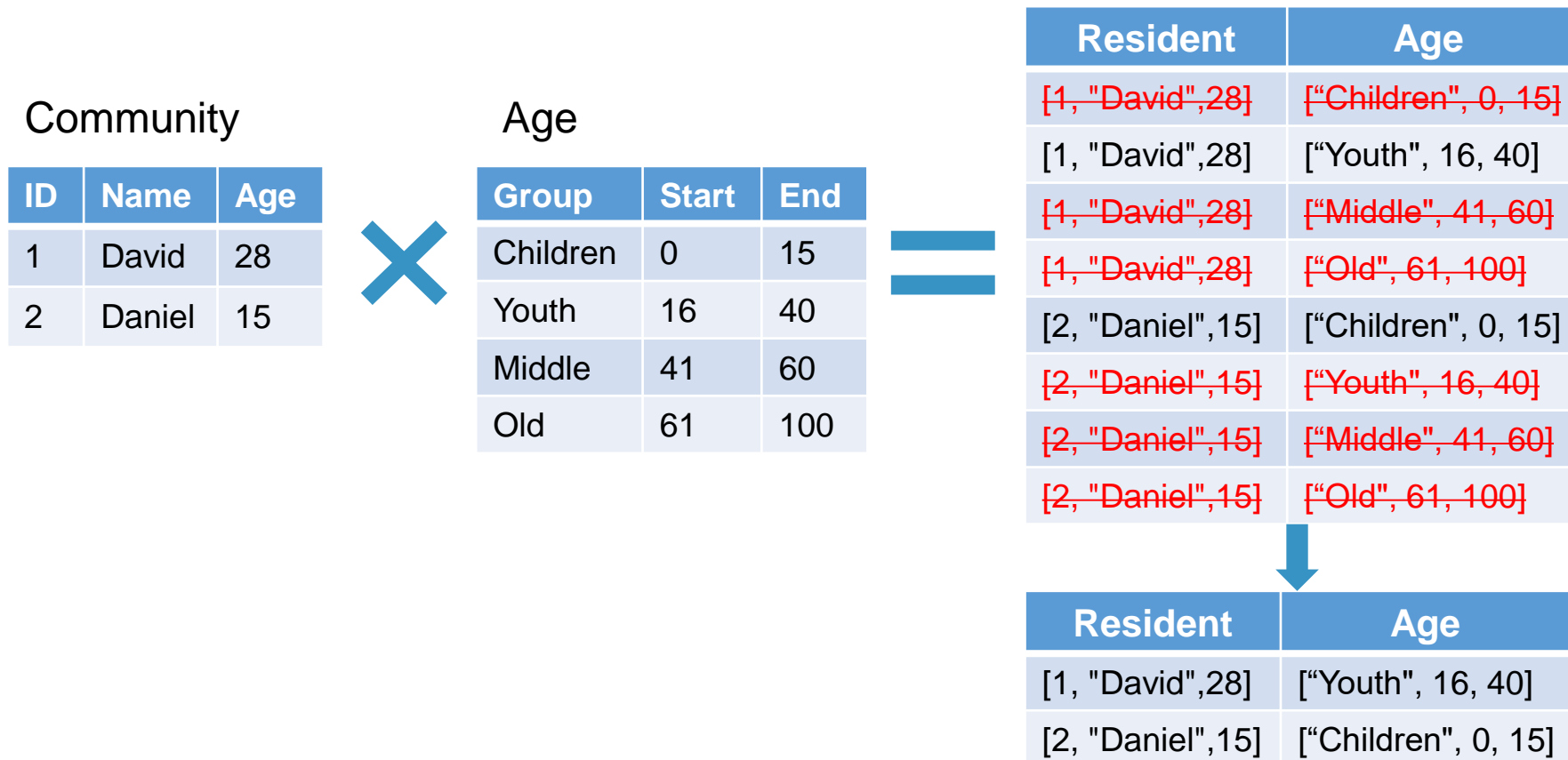
| ID | Name | Age |
|----|--------|-----|
| 1 | David | 28 |
| 2 | Daniel | 15 |
| 3 | Andrew | 65 |

Age

| Group | Start | End |
|----------|-------|-----|
| Children | 0 | 15 |
| Youth | 16 | 40 |
| Middle | 41 | 60 |
| Old | 61 | 100 |

两表之间并没有直接关联，我们需要把两表叉乘后，再按照起始结束年龄比较出所处年龄段。

两个表叉乘同时，按照年龄区间过滤（起始= \leq 年龄 \leq 结束）。



完整的SPL如下:

| | A | B |
|---|--|---|
| 1 | <code>=db.query("select * from Community")</code> | <code>=db.query("select * from Age")</code> |
| 2 | <code>=xjoin(A1:Resident; B1:Age, B1.Start<=A1.Age && B1.End>=A1.Age)</code> | <code>=A2.new(Resident.ID:ID, Resident.Name:Name,Resident.Age:Age,A ge.Grouop:Group)</code> |

xjoin函数支持过滤条件，可以在连接时过滤。

4.2 等值连接

矩阵表的数据结构如下，求两个矩阵的乘积。

| Matrix | |
|--------|----|
| row | 行号 |
| col | 列号 |
| value | 值 |

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

这里我们假设矩阵A*B中，矩阵A的列数等于矩阵B的行数。

1. 两个矩阵表一边叉乘，一边过滤（选出A的列号等于B的行号的记录）。

MatrixA

| row | col | value |
|-----|-----|-------|
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 1 | 3 | 3 |
| 2 | 1 | 4 |
| 2 | 2 | 5 |
| 2 | 3 | 6 |



MatrixB

| row | col | value |
|-----|-----|-------|
| 1 | 1 | 1 |
| 1 | 2 | 4 |
| 2 | 1 | 2 |
| 2 | 2 | 5 |
| 3 | 1 | 3 |
| 3 | 2 | 6 |



| A | B |
|----------------------|----------------------|
| [1, 1, 1] | [1, 1, 1] |
| [1, 1, 1] | [1, 2, 4] |
| [1, 1, 1] | [2, 1, 2] |
| [1, 1, 1] | [2, 2, 5] |
| [1, 1, 1] | [3, 1, 3] |
| [1, 1, 1] | [3, 2, 6] |
| ... | ... |

2. 值的乘积求和。

| row | col | sum(A.value*B.value) |
|-----|-----|----------------------|
| 1 | 1 | 14 |
| 1 | 2 | 32 |
| 2 | 1 | 32 |
| 2 | 2 | 77 |

完整的SPL如下:

| | A | B |
|---|---|---|
| 1 | <code>=db.query("select * from MatrixA")</code> | <code>=db.query("select * from MatrixB")</code> |
| 2 | <code>=xjoin(A1:A; B1:B,A1.col==B1.row)</code> | <code>=A2.groups(A.row,B.col;sum(A.value * B.value))</code> |

本例的数学公式如下:

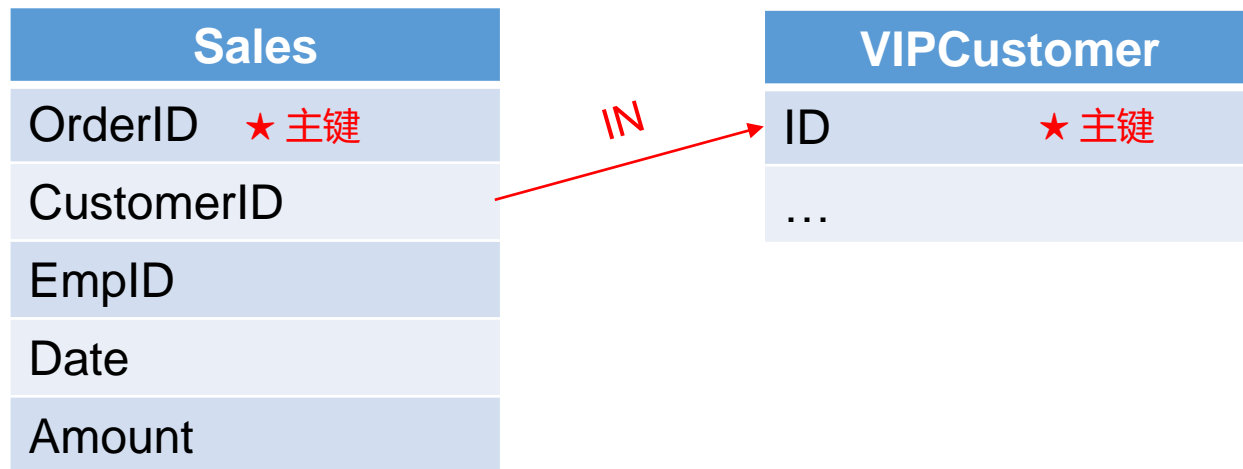
$$C = AB = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} = \begin{pmatrix} 1 \times 1 + 2 \times 2 + 3 \times 3 & 1 \times 4 + 2 \times 5 + 3 \times 6 \\ 4 \times 1 + 5 \times 2 + 6 \times 3 & 4 \times 4 + 5 \times 5 + 6 \times 6 \end{pmatrix} = \begin{pmatrix} 14 & 32 \\ 32 & 77 \end{pmatrix}$$

05

SQL子查询转换成JOIN

5.1 IN 子查询

查询VIP客户的销售信息。



SQL语句如下：

```
select * from Sales where CustomerID in ( select ID from VIPCustomer )
```

Sales

| OrderID | CustomerID | EmpID | Date | Amout | | VIPCustomer |
|--------------|------------------|--------------|-----------------------|-----------------|-----------|-------------|
| 1 | VINET | 2 | 2018-07-04 | 2440 | 连接过滤 → | CENTC |
| 2 | CENTC | 1 | 2018-07-05 | 3730 | | KNEOE |
| 3 | OTTIK | 5 | 2018-07-08 | 1863 | | ... |
| ... | ... | ... | ... | ... | | ... |

在VIP客户表中，客户ID是唯一的，两表可以通过客户ID建立连接。两表连接时，找不到对应的客户（非VIP客户）需要删除。

完整的SPL如下:

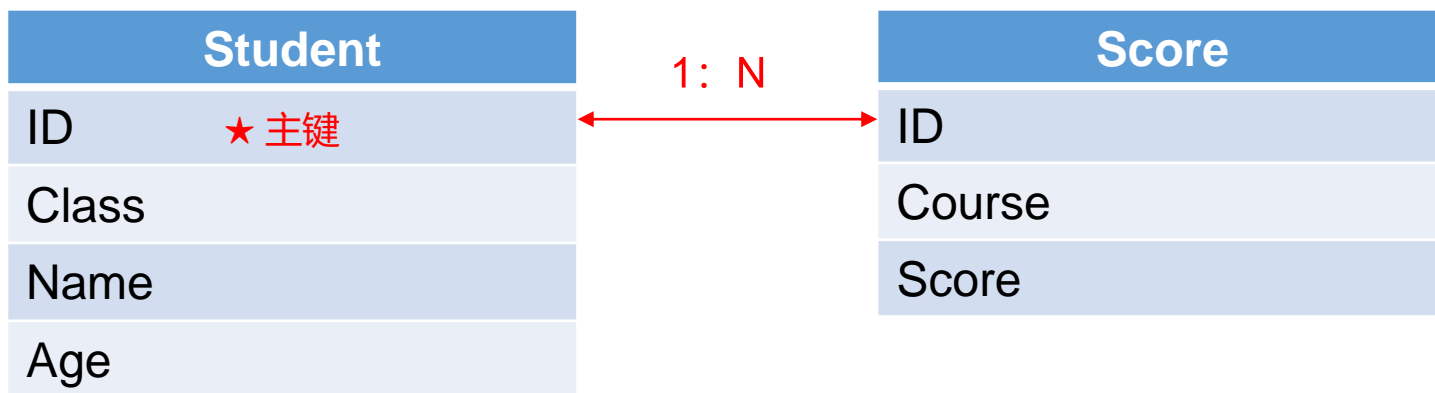
| | A | B |
|---|---|---|
| 1 | <code>=db.query("select * from Sales")</code> | <code>=db.query("select * from VIPCustomer")</code> |
| 2 | <code>=A1.join@i(CustomerID, B1:ID)</code> | |

A2: 使用了join@i连接选项, 找不到对应值时删除该记录, 用于连接过滤。

如果在连接后还需要使用外键表对象, 可以使用switch@i进行外键对象化并过滤。

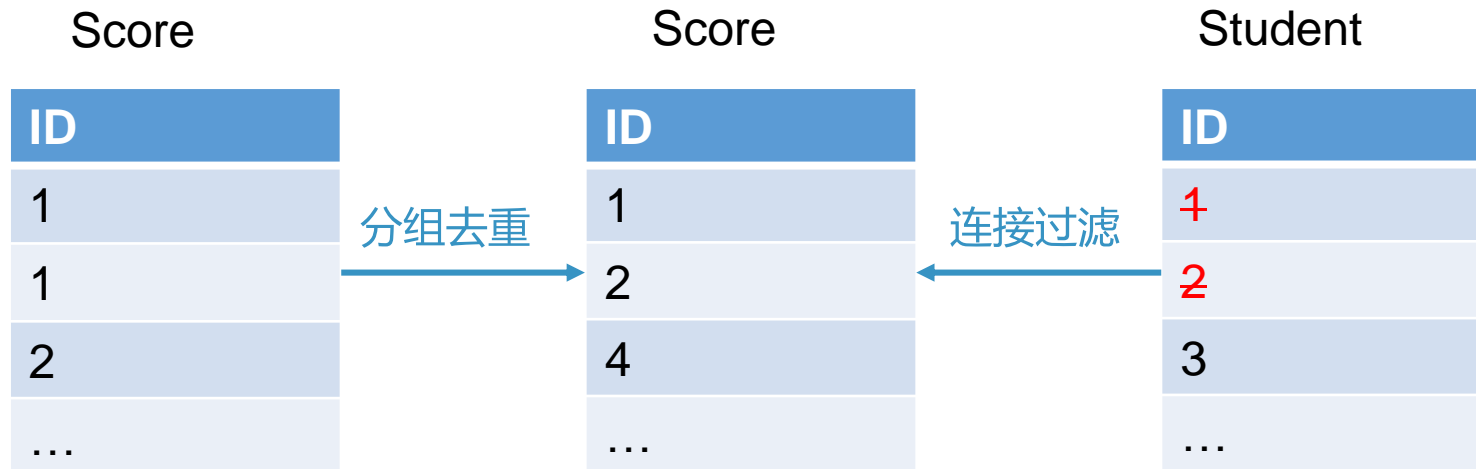
5.2 NOT IN 子查询

查找没有考试成绩的学生。



SQL语句如下:

```
select * from Student t1 where ID not in ( select ID from Score t2 where t1.ID = t2.ID )
```



在学生成绩表中学生ID不是唯一的，不能直接连接。需要先对学生ID进行分组去重，之后学生ID就相当于主键了，可以建立连接。

完整SPL如下:

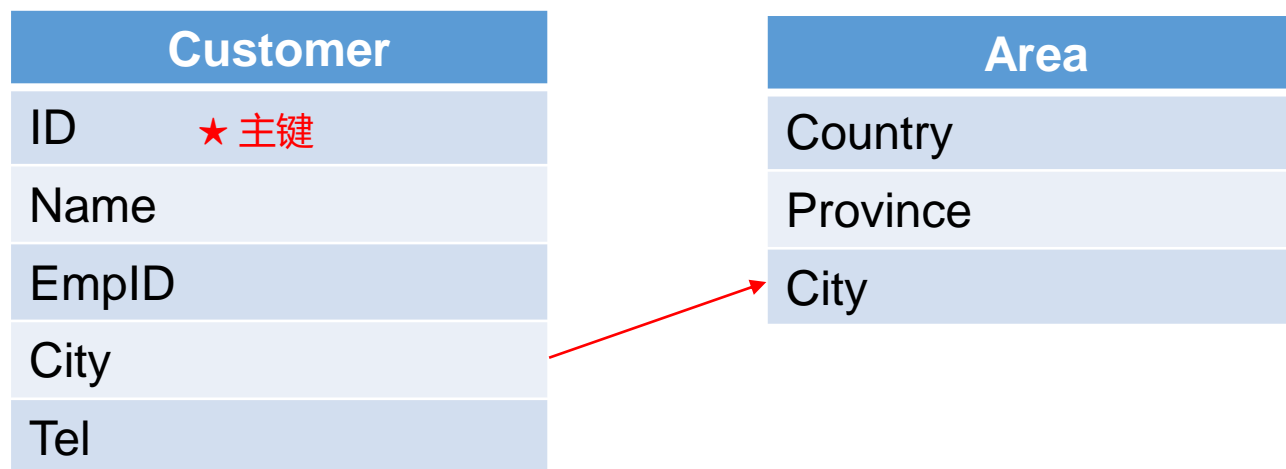
| | A | B |
|---|---|---|
| 1 | <code>=db.query("select * from Student")</code> | <code>=db.query("select * from Score")</code> |
| 2 | <code>=B1.group@1(ID)</code> | <code>=A1.join@d(ID, A2:ID)</code> |

A2: 使用了join@d连接选项, 找不到对应值时保留该记录 (与i选项相反), 用于连接过滤。

在IN和EXISTS语句中, 连接前需要判断连接字段在子查询中是否事实上的主键 (不是主键字段的, 经过条件过滤后也可能变成事实上的主键)。如果不是主键, 需要针对该字段分组后再连接。

5.3 EXISTS 子查询

例1：查询河北省的客户信息。



SQL语句如下：

```
select * from Customer t1 where exists ( select City from Area t2 where Province = 'HeBei' and t1.City=t2.City)
```

| Customer | | | | | Area | |
|------------------|----------------------|--------------|---------------------|------|----------|--------------|
| ID | Name | EmpID | City | | Province | City |
| ANTON | SanChuan | 1 | ShiJiaZhuang | | HeBei | ShiJiaZhuang |
| BOTTM | GuangTong | 2 | ShangHai | 连接过滤 | HeBei | TangShan |
| BSBEV | GuangHao | 3 | TangShan | | HeBei | QinHuangDao |
| ... | ... | ... | ... | | ... | ... |

观察子查询where条件中，连接字段除了城市字段以外，还有常数条件：省份=河北。对于有常数条件的 EXISTS 子查询，我们可以看作是多字段连接（省份+城市）。

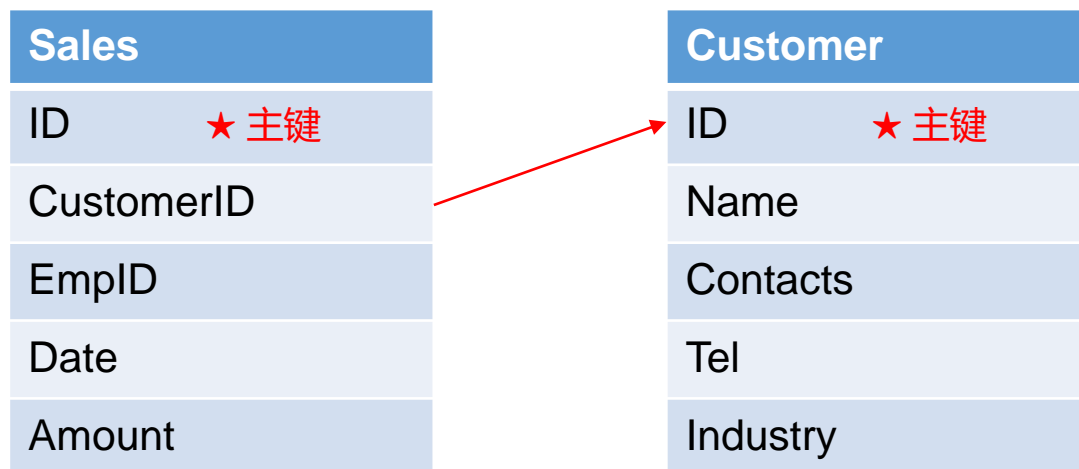
完整的SPL如下:

| | A | B |
|---|---|---------------------------------|
| 1 | =db.query("select * from Customer") | =db.query("select * from Area") |
| 2 | =A1.join@i("HeBei":City, A2:Province:City) | |

如果没有（省份=河北）这样的常数条件时，直接使用join@i()连接过滤即可（可能需要先分组去重）。另外，IN子查询除了空值和常量外，是可以转换为EXISTS子查询的，SPL对于两者的处理也是一样的。

5.4 NOT EXISTS 子查询

查询新增的互联网行业客户（客户表中没有的客户）。



SQL语句如下：

```
select * from Sales t1 where not exists ( select * from Customer t2 where t1.CustomerID=t2.ID and Industry = 'Internet' )
```

连接过滤，在客户表中找不到的客户会被删除。

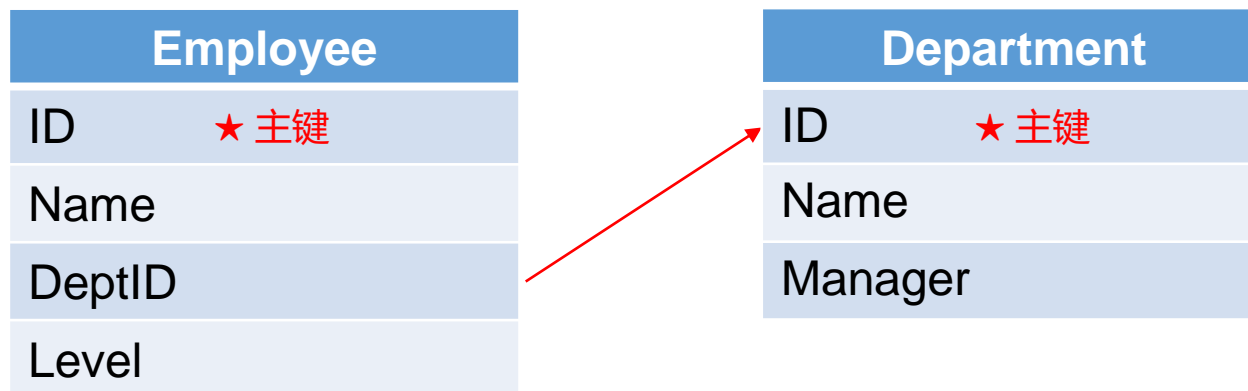
| Sales | | | Customer | |
|-------|------------|------|----------|-------------|
| ID | CustomerID | | ID | Industry |
| 4 | 2 | 连接过滤 | 1 | Internet |
| 2 | 1 | | 2 | Manufacture |

完整的SPL如下：

| | A | B |
|---|----------------------------------|-------------------------------------|
| 1 | =db.query("select * from Sales") | =db.query("select * from Customer") |
| 2 | =B1.select(Industry="Internet") | =A1.join@i(CustomerID, A2:ID) |

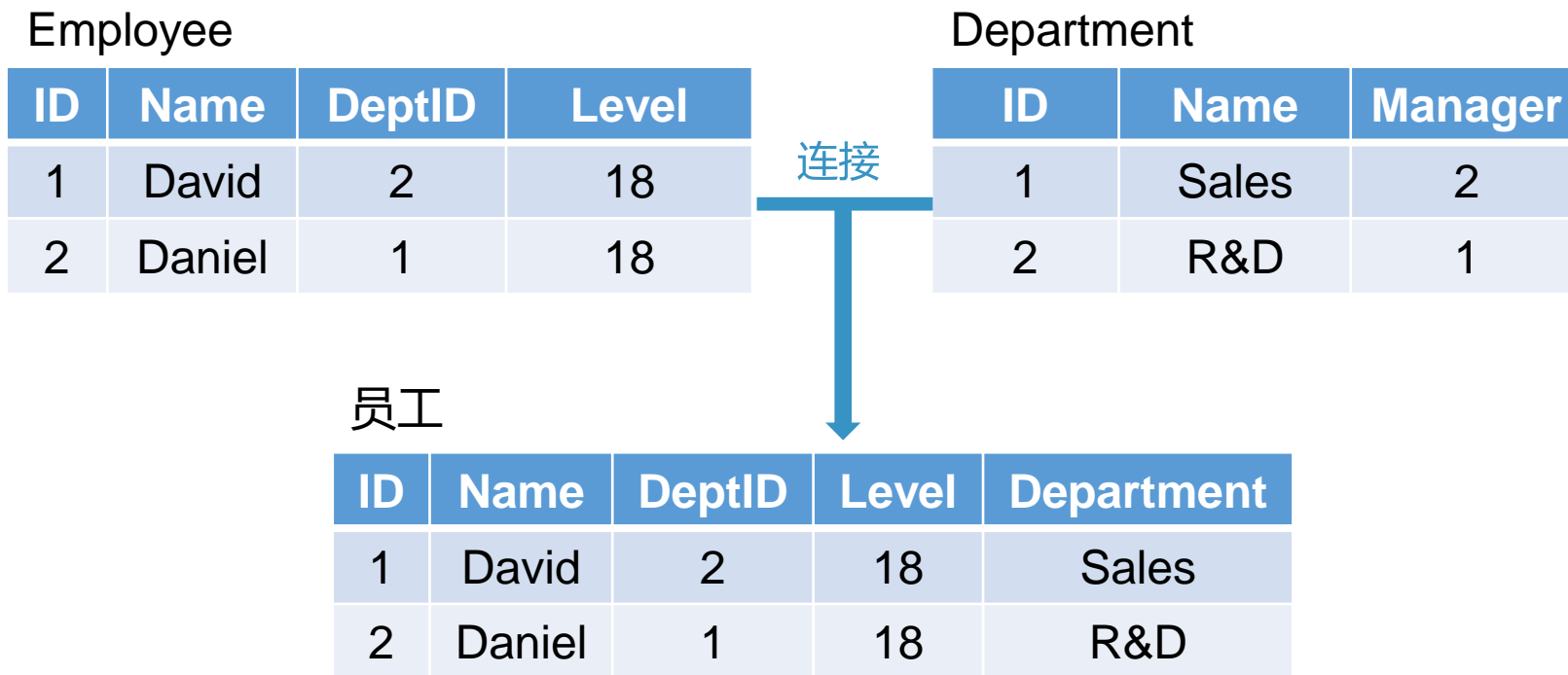
5.5 SELECT 子查询

查询员工的部门名称。



SQL语句如下：

```
select ID, Name, ( select Name from Department where Employee.DeptID=ID ) Department  
from Employee
```



观察 select 子查询中的 where 条件，用到了部门ID作为连接条件，相当于外键关联，直接连接即可。

完整的SPL如下:

| | A | B |
|---|--|--|
| 1 | <code>=db.query("select * from Employee")</code> | <code>=db.query("select * from Department")</code> |
| 2 | <code>=A1.join(DeptID, B1:ID, Department)</code> | |

5.6 WHERE 子查询

查找岗位工资高于8000的员工。



SQL语句如下：

```
select * from 员工 t1 where ( select 岗位工资 from 岗位工资表 t2 where t1.岗位级别=t2.岗位级别 ) > 8000
```

| Employee | | | | PostSalary | | |
|--------------|-------------------|--------------|---------------|---------------|-----------------|-----------------|
| ID | Name | DeptID | Level | Level | Salary | Allowance |
| 1 | David | 2 | 18 | 16 | 5000 | 1000 |
| 2 | Daniel | 1 | 18 | 17 | 7000 | 2000 |
| 3 | Andrew | 4 | 16 | 18 | 9000 | 3000 |

连接过滤

观察子查询where条件中，只有岗位级别作为连接字段，可以直接进行连接过滤。

完整的SPL如下:

| | A |
|---|---------------------------------------|
| 1 | =db.query("select * from Employee") |
| 2 | =db.query("select * from PostSalary") |
| 3 | =A2.select(Salary>8000) |
| 4 | =A1.join@i(Level, A3:Level) |

SELECT 和 WHERE 子查询是类似的，总是要用到主表的条件，相当于做外键关联。多个条件时相当于多字段外键连接。

THANKS

感谢聆听 批评指导