

友乾营·第十四期

分库后的统计查询

分库

分库通常会遵循一定规则，目的是将数据尽可能地均匀分布在不同库中

举例：n个MySQL数据库，1号库存放2013年销售数据，2号库存放2014年销售数据，...,n号库存放2012+n年的销售数据，如下所示：

CLIENT	SELLERID	AMOUNT	ORDERDATE
FHYBR	17	12000.0	2013-12-28 15:28:05
HANAR	16	4312.0	2013-12-28 15:28:05
YZ	14	19100.0	2013-12-28 15:28:05
MIP	10	5390.0	2013-12-29 15:28:05
AVU	2	27000.0	2013-12-30 15:28:05
VILX	9	19700.0	2013-12-31 15:28:05
PJIPE	12	7840.0	2013-12-31 15:28:05

1号库 (2013年)
销售数据

CLIENT	SELLERID	AMOUNT	ORDERDATE
JOPO	9	392.0	2014-01-01 15:28:05
AYWYN	13	25800.0	2014-01-04 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
DNEDL	10	24200.0	2014-01-07 15:28:05
HANAR	18	7252.0	2014-01-08 15:28:05
SAVEA	8	11600.0	2014-01-09 15:28:05
PJIPE	10	7742.0	2014-01-12 15:28:05
SAVEA	15	19900.0	2014-01-12 15:28:05

2号库 (2014年)
销售数据

.....

n号库 (2012+n年)
销售数据

where

各分库表过滤后的结果纵向连接

举例：n个库的销售数据表，查询单笔销售额大于500的所有订单记录

1号库 (2013年)
销售额大于500

CLIENT	SELLERID	AMOUNT	ORDERDATE
FHYBR	5	29700.0	2013-04-05 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05
YZ	8	29600.0	2013-01-06 15:28:05
FHYBR	2	29400.0	2013-08-19 15:28:05
SJCH	18	29100.0	2013-05-29 15:28:05
BTMMU	16	29100.0	2013-11-13 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05
SAVEA	9	28800.0	2013-02-12 15:28:05

2号库 (2014年)
销售额大于500

CLIENT	SELLERID	AMOUNT	ORDERDATE
AYWYN	13	25800.0	2014-01-04 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
DNEDL	10	24200.0	2014-01-07 15:28:05
HANAR	18	7252.0	2014-01-08 15:28:05
SAVEA	8	11600.0	2014-01-09 15:28:05
PJIPE	10	7742.0	2014-01-12 15:28:05
SAVEA	15	19900.0	2014-01-12 15:28:05
FRNSH	18	6272.0	2014-01-13 15:28:05

CLIENT	SELLERID	AMOUNT	ORDERDATE
SJCH	5	25500.0	2013-12-27 15:28:05
FHYBR	17	12000.0	2013-12-28 15:28:05
HANAR	16	4312.0	2013-12-28 15:28:05
YZ	14	19100.0	2013-12-28 15:28:05
MIP	10	5390.0	2013-12-29 15:28:05
AVU	2	27000.0	2013-12-30 15:28:05
PJIPE	12	7840.0	2013-12-31 15:28:05
VILX	9	19700.0	2013-12-31 15:28:05
AYWYN	13	25800.0	2014-01-04 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
DNEDL	10	24200.0	2014-01-07 15:28:05
HANAR	18	7252.0	2014-01-08 15:28:05
SAVEA	8	11600.0	2014-01-09 15:28:05
PJIPE	10	7742.0	2014-01-12 15:28:05
SAVEA	15	19900.0	2014-01-12 15:28:05
BTMMU	8	18700.0	2014-01-13 15:28:05

由程序将分库过滤结果纵向连接

where

各分库表过滤后的结果纵向连接

举例：n个库的销售数据表，查询单笔销售额大于500的所有订单记录

序
表

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales where amount > 500"	
3	fork A1	=A3.query@x(SQL)
4	=A3.conj()	

数据量较大时：

游
标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales where amount > 500"	
3	fork A1	=A3.cursor(SQL)
4	=A3.conjx()	

order by

各分库表按某字段排序后的结果再排序

举例：n个库的销售数据表，按订单金额逆序排序

1号库（2013年）
按销售额逆序排序

CLIENT	SELLERID	AMOUNT	ORDERDATE
FHYBR	5	29700.0	2013-04-05 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05
YZ	8	29600.0	2013-01-06 15:28:05
FHYBR	2	29400.0	2013-08-19 15:28:05
SJCH	18	29100.0	2013-05-29 15:28:05
BTMMU	16	29100.0	2013-11-13 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05

2号库（2014年）
按销售额逆序排序

CLIENT	SELLERID	AMOUNT	ORDERDATE
HP	6	29800.0	2014-10-08 15:28:05
JOPO	3	29700.0	2014-12-15 15:28:05
DILRT	14	29700.0	2014-05-24 15:28:05
BTMMU	7	29700.0	2014-04-30 15:28:05
HANAR	15	29600.0	2014-07-12 15:28:05
ERNSH	4	29600.0	2014-10-07 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05

.....

CLIENT	SELLERID	AMOUNT	ORDERDATE
HP	6	29800.0	2014-10-08 15:28:05
DILRT	14	29700.0	2014-05-24 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05
JOPO	3	29700.0	2014-12-15 15:28:05
BTMMU	7	29700.0	2014-04-30 15:28:05
FHYBR	5	29700.0	2013-04-05 15:28:05
ERNSH	4	29600.0	2014-10-07 15:28:05
YZ	8	29600.0	2013-01-06 15:28:05
HANAR	15	29600.0	2014-07-12 15:28:05
FHYBR	2	29400.0	2013-08-19 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
BTMMU	16	29100.0	2013-11-13 15:28:05
SJCH	18	29100.0	2013-05-29 15:28:05
AYWYN	5	28900.0	2014-09-01 15:28:05
QHHW	16	28900.0	2014-10-30 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05
AVLI	11	28800.0	2013-05-17 15:28:05

将分库排序结果再次按订购日期排序
（分库结果集有序，可使用归并算法
进行有序归并，提升效率）

order by

各分库表按某字段排序后的结果再排序

举例：n个库的销售数据表，按订单金额降序或升序排序

升
序

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(score)	

降
序

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount desc"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(-score)	

order by

各分库表按某字段排序后的结果再排序

举例：n个库的销售数据表，按订单金额升序排序

序
表

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(score)	

数据量较大时：

游
标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount"	
3	fork [A1:B1]	=A3. cursor (SQL)
4	=A3. mergex (score)	

limit Y offset X

基于排序结果，取第几条后的多少条记录

举例：n个库的销售数据表，按订单金额逆序排序，随后跳过第1条的记录，取10条记录

1号库 (2013年)
按销售额逆序排序

CLIENT	SELLERID	AMOUNT	ORDERDATE
FHYBR	5	29700.0	2013-04-05 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05
YZ	8	29600.0	2013-01-06 15:28:05
FHYBR	2	29400.0	2013-08-19 15:28:05
SJCH	18	29100.0	2013-05-29 15:28:05
BTMMU	16	29100.0	2013-11-13 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05

2号库 (2014年)
按销售额逆序排序

CLIENT	SELLERID	AMOUNT	ORDERDATE
HP	6	29800.0	2014-10-08 15:28:05
JOPO	3	29700.0	2014-12-15 15:28:05
DILRT	14	29700.0	2014-05-24 15:28:05
BTMMU	7	29700.0	2014-04-30 15:28:05
HANAR	15	29600.0	2014-07-12 15:28:05
ERNSH	4	29600.0	2014-10-07 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
BTMMU	16	29100.0	2014-10-30 15:28:05

.....

CLIENT	SELLERID	AMOUNT	ORDERDATE
HP	6	29800.0	2014-10-08 15:28:05
DILRT	14	29700.0	2014-05-24 15:28:05
QUICK	6	29700.0	2013-11-04 15:28:05
JOPO	3	29700.0	2014-12-15 15:28:05
BTMMU	7	29700.0	2014-04-30 15:28:05
FHYBR	5	29700.0	2013-04-05 15:28:05
ERNSH	4	29600.0	2014-10-07 15:28:05
YZ	8	29600.0	2013-01-06 15:28:05
HANAR	15	29600.0	2014-07-12 15:28:05
FHYBR	2	29400.0	2013-08-19 15:28:05
PJIPE	14	29400.0	2014-01-05 15:28:05
BTMMU	16	29100.0	2013-11-13 15:28:05
SJCH	18	29100.0	2013-05-29 15:28:05
AYWYN	5	28900.0	2014-09-01 15:28:05
QHHW	16	28900.0	2014-10-30 15:28:05
SAVEA	7	28900.0	2013-03-23 15:28:05
AVLI	11	28800.0	2013-05-17 15:28:05

跳过
第1
条取
10条

基于分库排序+有序归并后的结果，跳过X条记录，取Y条记录
(分库可以取X+Y条，但不能跳)

limit Y offset X

基于排序结果，取第几条后的多少条记录

举例：从n个库的销售数据表中，按订单金额逆序排序，随后跳过第1条的记录，取10条记录

序
表

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount limit 11 "	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(-amount).to(2,11)	

数据量较大时：

游
标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from sales order by amount desc limit 11 "	
3	fork A1	=A3. cursor (SQL)
4	=A3. mergex (-amount)	=A4.skip(1)
5	=A4.fetch(10)	

| sum/count/avg/max/min

各分库表聚合后的结果需要再聚合，但有些注意事项

sum：分库sum，汇总还是sum

count：分库count，汇总是要用sum

avg：要变成sum和count分别计算后再计算avg

max\min：和sum一样

sum/count/avg/max/min

各分库表聚合后的结果需要再聚合，但有些注意事项

举例：n个库的销售数据表，求销售总额、订单数、销售额最值和销售额均值

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select sum(amount) totalamount, count(amount) countamount, max(amount) maxamount, min(amount) minamount from sales"	
3	fork A1	=A3.cursor(SQL)
4	=A3.conjx().total(sum(totalamount), sum(countamount), max(maxamount), min(minamount))	
5	=create(totalamount, countamount, avgamount, maxamount, minamount).insert(0, A4(1), A4(2), if(A4(2)!=0, round(A4(1)/A4(2)), null), A4(3), A4(4))	

group by

各分库表的分组聚合结果再分组聚合，不能简单合并

举例：n个库的销售数据表，按销售人员ID分组，对销售金额求和、最大最小值与订单数

sellerid	totalamount	countamount	maxamount	minamount
1	313228.0	20	26400.0	4998.0
2	298006.0	23	29400.0	1666.0
3	142054.0	14	23100.0	1568.0
4	237614.0	14	28400.0	1764.0
5	252294.0	20	29700.0	490.0
6	246922.0	18	29700.0	1372.0

SELLERID	TOTALAMOUNT	COUNTAMOUNT	MAXAMOUNT	MINAMOUNT
1	265934	20	27400	98
2	301502	26	27800	392
3	279986	20	29700	686
4	228070	20	29600	1862
5	175478	16	28900	392
6	198150	17	29800	294

SELLERID	TOTALAMOUNT	COUNTAMOUNT	MAXAMOUNT	MINAMOUNT
1	687850.0	51	29000.0	98.0
2	734276.0	64	29400.0	392.0
3	554724.0	44	29700.0	686.0
4	565594.0	48	29600.0	98.0
5	564516.0	50	29700.0	196.0
6	672064.0	50	29800.0	294.0

I group by

各分库表的分组聚合结果再分组聚合，不能简单合并

举例：n个库的销售数据表，按销售人员ID分组，对销售金额求和、最大最小值与订单数

对于各库结果较小，再聚合运算的结果集只会更小，使用**query+groups**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select sellerid, sum(amount) totalamount, count(amount) countamount,max(amount) maxamount,min(amount) minamount from sales group by sellerid"	
3	fork A1	=A3. query @x(SQL)
4	=A3.conj(). groups (sellerid;sum(totalamount):totalamount,sum(countamount):countamount,max(maxamount):maxamount,min(minamount):minamount)	
5	=A4.new(sellerid,totalamount,countamount,if(countamount==0,null,round(totalamount/countamount)):avgamount,maxamount,minamount)	

I group by

各分库表的分组聚合结果再分组聚合，不能简单合并

举例：n个库的销售数据表，按销售人员ID分组，对销售金额求和、最大最小值与订单数

对于各库结果较大，再分组聚合的结果不大，使用**cursor+groups**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select sellerid, sum(amount) totalamount, count(amount) countamount,max(amount) maxamount,min(amount) minamount from sales group by sellerid"	
3	fork A1	=A3. cursor (SQL)
4	=A3. conjx().groups (sellerid;sum(totalamount):totalamount,sum(countamount):countamount,max(maxamount):maxamount,min(minamount):minamount)	
5	=A4.new(sellerid,totalamount,countamount,if(countamount==0,null,round(totalamount/countamount)):avgamount,maxamount,minamount)	

I group by

各分库表的分组聚合结果再分组聚合，不能简单合并

举例：n个库的销售数据表，按销售人员ID分组，对销售金额求和、最大最小值与订单数

对于各库结果较大，再分组聚合的结果也很大，使用**cursor+groupx**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select sellerid, sum(amount) totalamount, count(amount) countamount,max(amount) maxamount,min(amount) minamount from sales group by sellerid order by sellerid"	
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex().groupx@o(sellerid;sum(totalamount):totalamount,sum(countamount):countamount,max(maxamount):maxamount,min(minamount):minamount)	
5	=A4.new(sellerid,totalamount,countamount,if(countamount==0,null,round(totalamount/countamount)):avgamount,maxamount,minamount)	

group by having

各分库表仅分组聚合，所有结果彻底完成分组聚合后再过滤

举例：n个库的销售数据表，按月份和销售员ID分组，统计平均销售额小于10000的数据

1号库 (2013年)
按月、销售员id
分组，统计销售
总额和订单总数

ORDERMONTH	SELLERID	TOTALAMOUNT	TOTALCOUNT
1	1	24800.0	2
1	2	10300.0	1
1	3	37766.0	3
1	4	26700.0	1
1	5	36800.0	2
1	6	67858.0	4
1	8	34206.0	3
1	9	26200.0	1

2号库 (2014年)
按月、销售员id
分组，统计销售
总额和订单总数

ORDERMONTH	SELLERID	TOTALAMOUNT	TOTALCOUNT
1	3	9408.0	2
1	4	5782.0	1
1	6	5292.0	1
1	7	20000.0	1
1	8	68220.0	6
1	9	2646.0	2
1	10	38704.0	3
1	11	3528.0	1

.....

ORDERMONTH	SELLERID	AVGAMOUNT
1	2	8932.0
1	16	6444.0
1	17	8859.0
1	18	5945.0
1	19	9993.0
2	3	8918.0
2	4	8368.0
2	5	9428.0

数据量较大时，分库中分别按分组维度有序地返回结果。程序在处理时可采用归并算法，有序归并各库结果。归并后的结果仍是有序的，可继续采用有序分组聚合后再过滤。（要注意的是：分库中不应having）

group by having

各分库表仅分组聚合，所有结果彻底完成分组聚合后再过滤

举例：n个库的销售数据表，按月份和销售员ID分组，统计平均销售额小于10000的数据
聚合计算是在汇总之后才能最终完成，随后再使用函数select实现过滤

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select month(orderdate) ordermonth,sellerid,sum(amount) totalamount,count(amount) totalcount from sales group by ordermonth,sellerid"	
3	fork A1	=A3.query@x(SQL)
4	=A3.conj().groups(ordermonth,sellerid;sum(totalamount):totalamount,sum(totalcount):totalcount).new(ordermonth,sellerid,(totalamount/totalcount):avgamount).select(avgamount<10000)	

对于大数据量，需要使用游标 (select函数同样适用于游标)

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select month(orderdate) ordermonth,sellerid,sum(amount) totalamount,count(amount) totalcount from sales group by ordermonth,sellerid order by ordermonth,sellerid"	
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex(ordermonth,sellerid).groupx@o(ordermonth,sellerid;sum(totalamount):totalamount,sum(totalcount):totalcount).new(ordermonth,sellerid,(totalamount/totalcount):avgamount).select(avgamount<10000)	

distinct

distinct x 相当于 select x group by x，实现方法和group相同

各分库表仅做去重运算，所有结果做并集（去重），不能简单合并

举例：n个库的销售数据表，统计所有对客户编号为 'YZ' 有过成交记录的销售人员ID

1号库（2013年）
统计当年对客户编号为 'YZ'
有成交记录的销售人员ID

sellerid
1
2
3
4
5
6
7
8

2号库（2014年）
统计当年对客户编号为 'YZ'
有成交记录的销售人员ID

SELLERID
2
4
7
13
16
17
18
19

.....

SELLERID
1
8
9
11
12
13
14
15
16
17
18
19

显然需要将各分库的结果做并集（去重），而非简单的合并。

distinct

distinct x 相当于 select x group by x，实现方法和group相同

各分库表仅做去重运算，所有结果做并集（去重），不能简单合并

举例：n个库的销售数据表，统计所有对客户编号为 'YZ' 有过成交记录的销售人员ID

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select distinct sellerid from sales where client='YZ' order by sellerid"	
3	fork A1	=A3. query@x (SQL)
4	=A3. merge@u (sellerid)	

对于大数据量，可以使用**cursor**、**mergex@u**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select distinct sellerid from sales where client='YZ' order by sellerid"	
3	fork A1	=A3. cursor (SQL)
4	=A3. mergex@u (sellerid)	

count(distinct)

各分库表仅做去重运算，所有结果做并集去重后再计数

举例：n个库的销售数据表，统计单笔销售额大于29000的销售员个数

1号库 (2013年)
统计单笔销售额大于
29000的销售员

sellerid
2
5
6
8
16
18

2号库 (2014年)
统计单笔销售额大于
29000的销售员

SELLERID
3
4
6
7
14
15

.....

COUNTRESULT
11

对于分库中去重结果集较大时，
可以在各库分别排序，后续代码
可利用有序的特点用归并算法求
并集后再计数

count(distinct)

各分库表仅做去重运算，所有结果做并集（去重）后再计数

举例：n个库的销售数据表，统计单笔销售额大于29000的销售员个数

对于小数据量，可以使用**query**、**merge@u**和**len**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select distinct sellerid from sales where amount>29000 order by sellerid"	
3	fork A1	=A3. query@x (SQL)
4	=A3. merge@u (sellerid).len()	

对于大数据量小结果集，可以使用**cursor**、**mergex@u**和**total**

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select distinct sellerid from sales where amount>29000 order by sellerid"	
3	fork A1	=A3. cursor (SQL)
4	=A3. mergex@u (sellerid).total(count(~))	

count(distinct) group by

count(distinct) group by要把distinct的内容和group by内容一起group(distinct)

各分库表仅做去重运算，所有结果做并集（去重）后再计数

举例：n个库的销售数据表，按月份分组，统计单笔销售额大于10000的销售员个数

1号库（2013年）
统计单笔销售额大于
10000，且按月、销售
员id去重并排序

ORDERMONTH	SELLERID
1	1
1	2
1	3
1	4
1	5
1	6
1	8
1	9
1	10

2号库（2014年）
统计单笔销售额大于
10000，且按月、销售
员id去重并排序的结果

ORDERMONTH	SELLERID
1	7
1	8
1	10
1	13
1	14
1	15
1	20
2	1
2	4

.....

ORDERMONTH	COUNTID
1	18
2	17
3	19
4	20
5	19
6	18
7	17
8	15
9	16
10	14
11	20
12	19

对于分库中去重结果集较大时，
可以在各库分别排序，后续代码
可利用有序的特点用归并算法求
并集后再计数

count(distinct) group by

count(distinct) group by要把distinct的内容和group by内容一起group(distinct)

各分库表仅做去重运算，所有结果做并集（去重）后再计数

举例：n个库的销售数据表，按月份分组，统计单笔销售额大于10000的销售员个数

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select distinct month(orderdate) ordermonth,sellerid from sales where amount > 10000 order by ordermonth,sellerid"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge@u(ordermonth,sellerid).groups@o(ordermonth;count(sellerid):countseller)	

对于大数据量小结果集，可以使用cursor、mergex@u和groups@o

	A	B
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex@u(ordermonth,sellerid).groups@o(ordermonth;count(sellerid):countseller)	

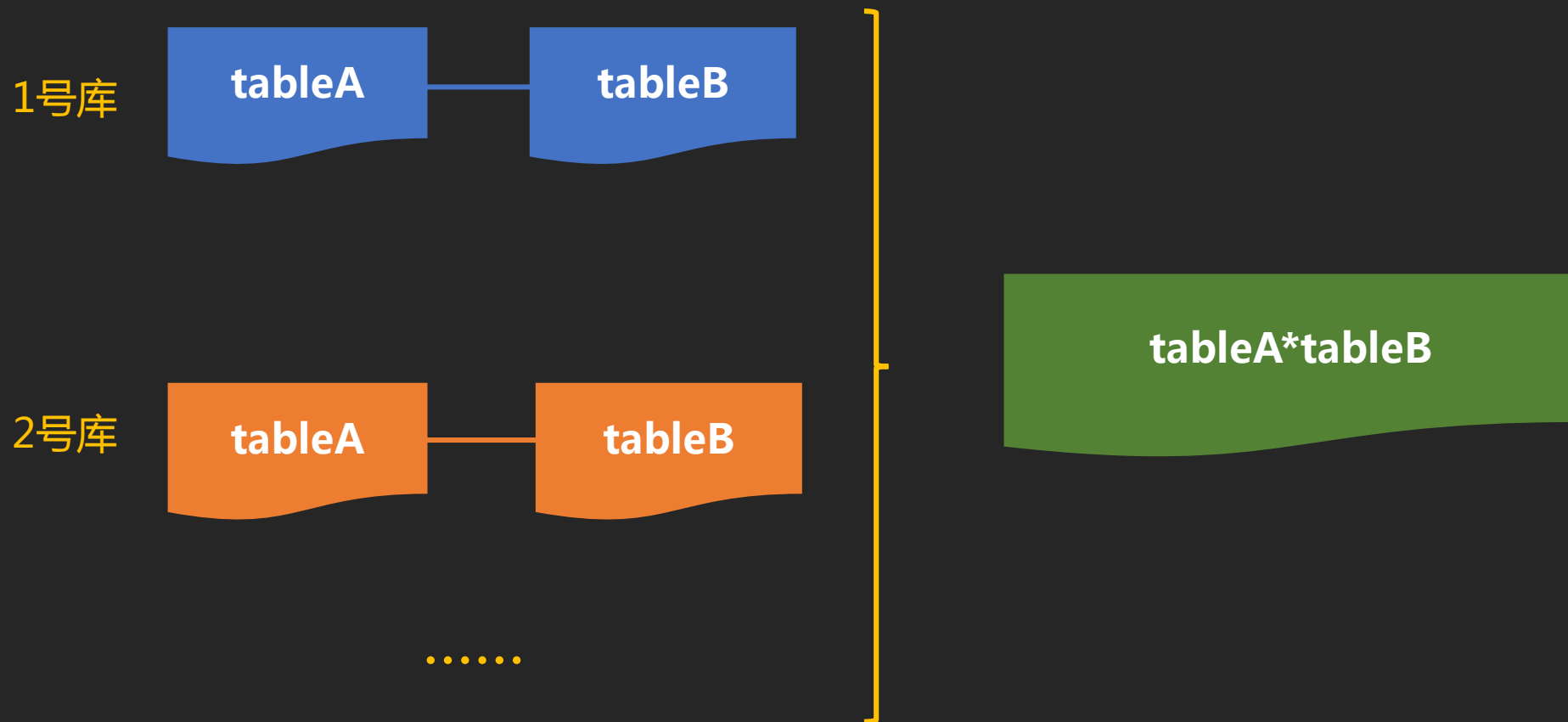
对于大数据量大结果集，可以使用cursor、mergex@u和groupx@o

	A	B
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex@u(ordermonth,sellerid).groupx@o(ordermonth;count(sellerid):countseller)	

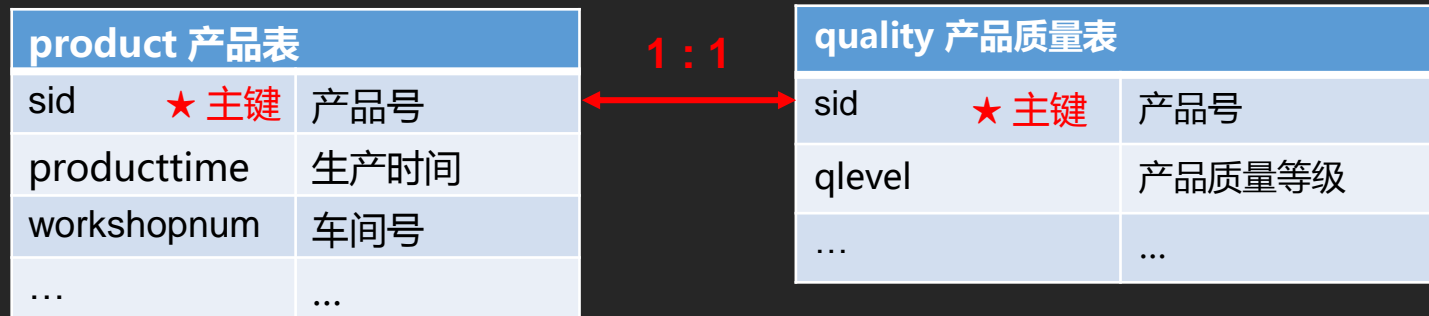
I 带JOIN的查询

合理的数据分布方案，可以使JOIN只在分库进行，汇总阶段不再处理JOIN

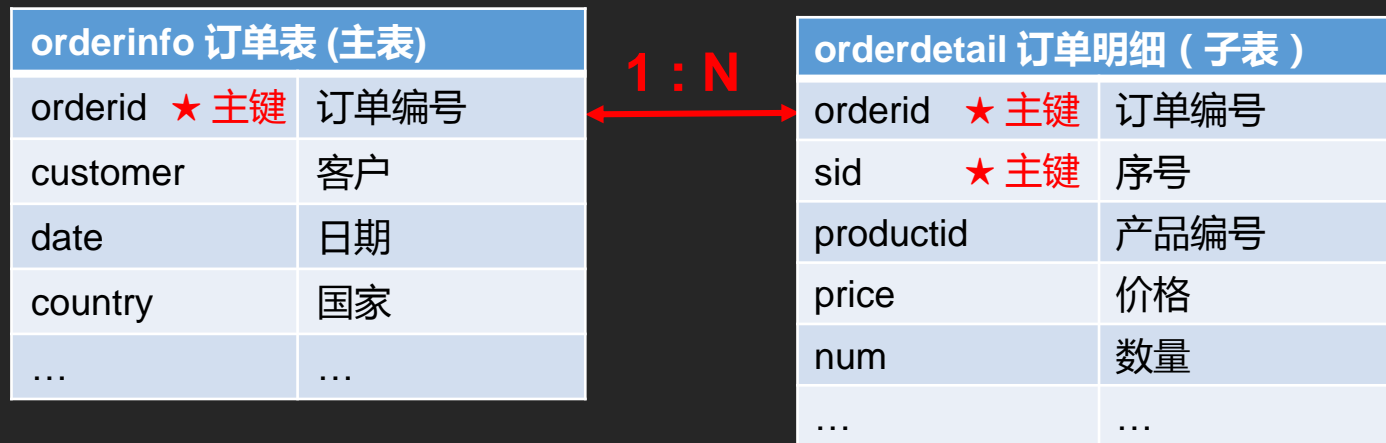
在同步分库的方案下，FULL JOIN，LEFT JOIN都可以支持，但不同类型的JOIN关系需要不同的分布方式



I 同维与主子表



同维表：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。



主子表：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。

外键表（维表）

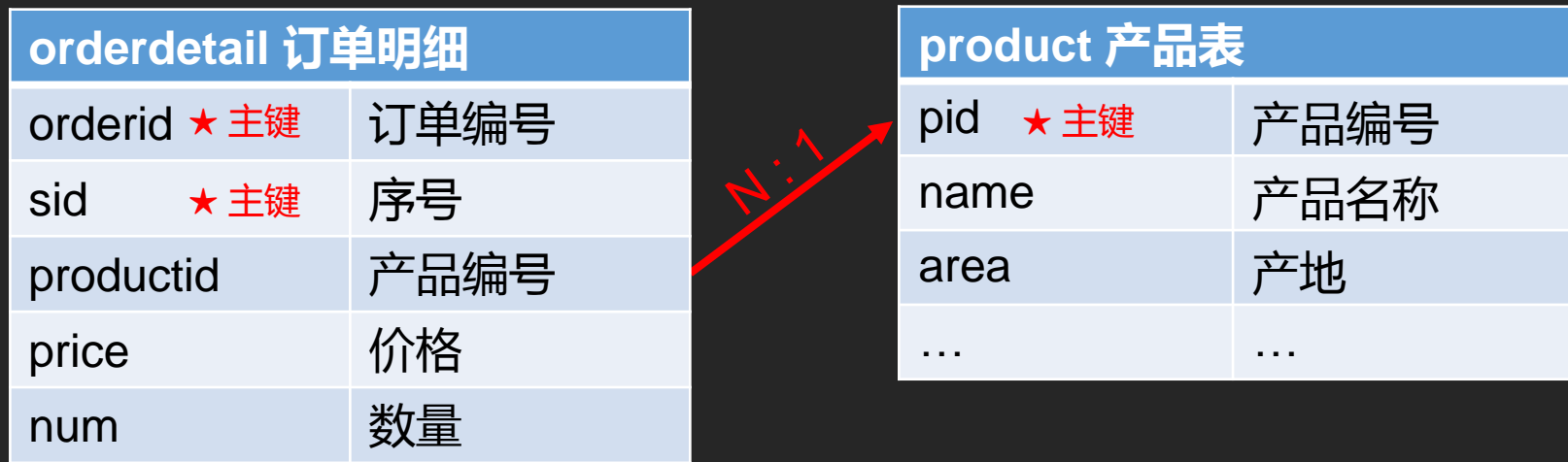


表 A 的某些字段与表 B 的**主键**关联。A 表中与 B 表主键关联的字段称为 A 指向 B 的外键，B 也称为 A 的外键表。外键表是多对一的关系。

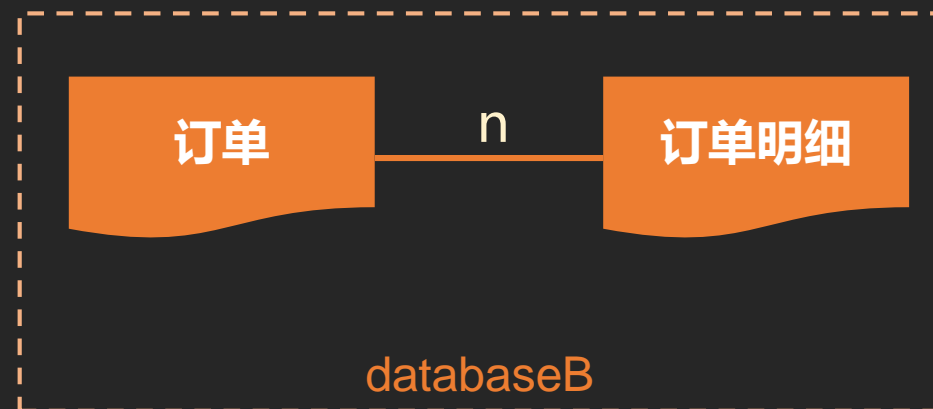
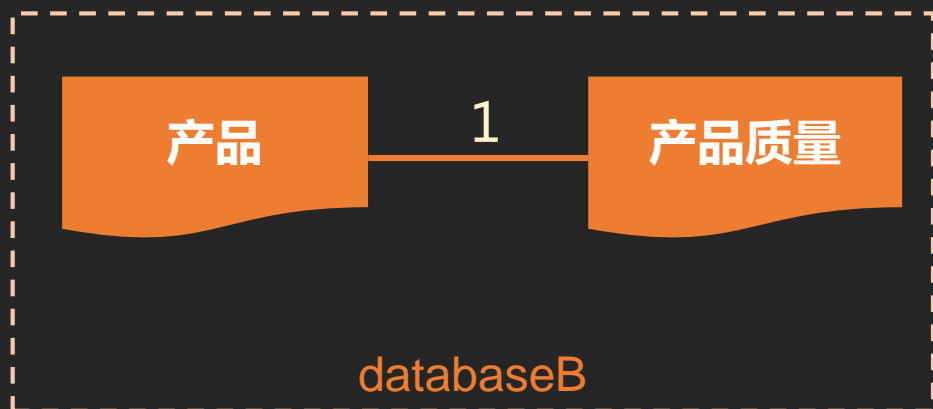
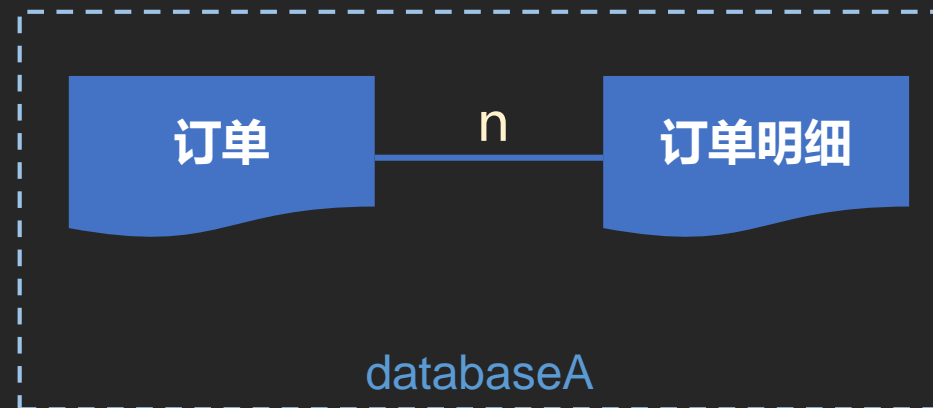
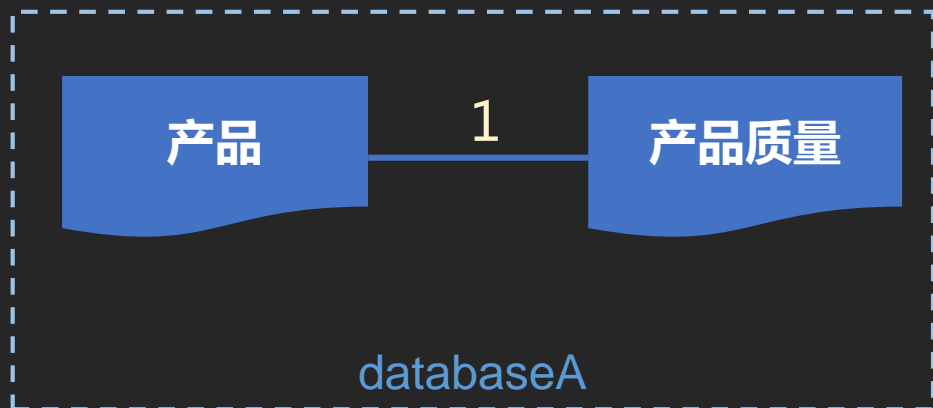
同维主子表同步分库

分库后，在各分库中把join处理完即可，汇总阶段不用再考虑join

同维表

主子表

同步
分库



I 同维表

同步分库的同维表，可在各分库中分别关联后合并

举例：n个库中，产品表与产品质量表属同步分库，按车间号分组，统计产品质量等级小于3的产品个数

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select t1.workshopnum workshopnum,count(t2.qlevel) lt3count from product t1 join quality t2 on t1.sid=t2.sid where t2.qlevel<3 group by t1.workshopnum order by t1.workshopnum"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(workshopnum).groups@o(workshopnum;sum(lt3count):lt3count)	

对于大数据量，使用游标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select t1.workshopnum workshopnum,count(t2.qlevel) lt3count from product t1 join quality t2 on t1.sid=t2.sid where t2.qlevel<3 group by t1.workshopnum order by t1.workshopnum"	
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex(workshopnum).groupx@o(workshopnum;sum(lt3count):lt3count)	

I 主子表

同步分库的主子表，可在各分库中分别关联后合并

举例：n个库中，订单与订单明细属同步分库，按国家、产品编号分组，统计产品总数

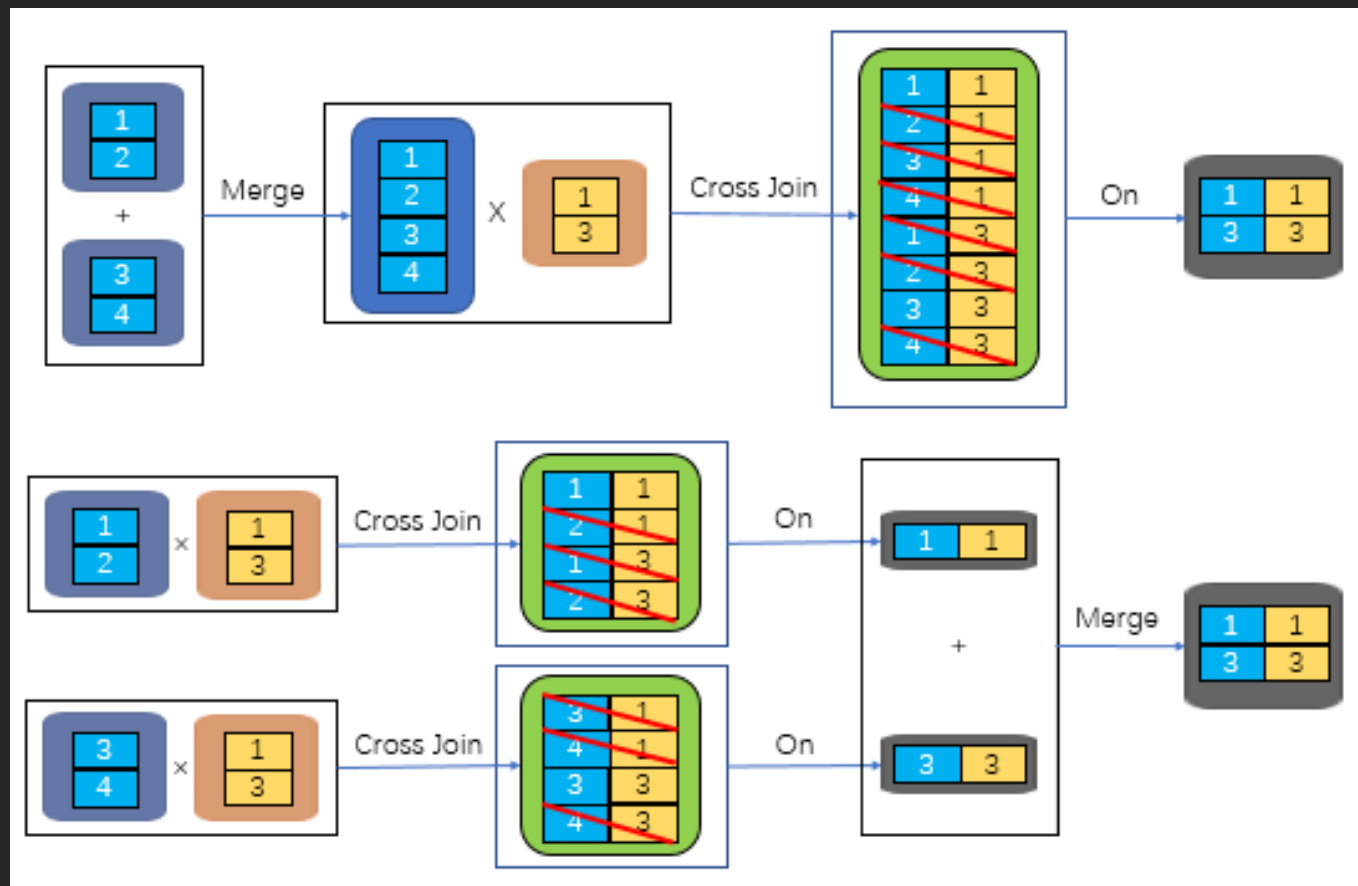
	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select t1.country country,t2.productid productid,sum(t2.num) totalnum from orderinfo t1 join orderdetail t2 on t1.orderid=t2.orderid group by t1.country,t2.productid order by t1.country,t2.productid"	
3	fork A1	=A3.query@x(SQL)
4	=A3.merge(country,productid).groups@o(country,productid;sum(totalnum):totalnum)	

对于大数据量，使用游标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select t1.country country,t2.productid productid,sum(t2.num) totalnum from orderinfo t1 join orderdetail t2 on t1.orderid=t2.orderid group by t1.country,t2.productid order by t1.country,t2.productid"	
3	fork A1	=A3.cursor(SQL)
4	=A3.mergex(country,productid).groupx@o(country,productid;sum(totalnum):totalnum)	

外键维表复制冗余

维表（右表）在每个库中都有同一份，则事实表（左表）（每个数据库中各有其一部分）在汇总后再连接，等同于各库中的事实表与维表先连接后，再汇总到一起的结果。如图：



I 外键表（维表）

各分库均有一份相同的完整外键表，可在各分库中分别关联后合并

举例：n个库中，订单明细在各分库均匀分布，产品表在各分库冗余，过滤出所有产地为东部的订单

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from orderdetail t1 join product t2 on t1.productid=t2.pid where t2.area='east'"	
3	fork A1	=A3.query@x(SQL)
4	=A3.conj()	

对于大数据量，使用游标

	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="select * from orderdetail t1 join product t2 on t1.productid=t2.pid where t2.area='east'"	
3	fork A1	=A3.cursor(SQL)
4	=A3.conjx()	

I 异构数据库分库

各数据库函数不尽相同，异构库分库需要执行相应的SQL

举例：依旧是分组聚合过滤的例子，由于MySQL和Oracle的“取月”函数不同，需要分别执行不同SQL

标准函数	含义	Oracle	MySQL
YEAR(d)	取年	EXTRACT(YEAR FROM d)	YEAR(d)
MONTH(d)	取月	EXTRACT(MONTH FROM d)	MONTH(d)

MySQL: **select month(orderdate)** ordermonth,sellerid,**sum**(amount) totalamount,**count**(amount) totalcount **from** sales **group by** **month(orderdate)**,sellerid **order by** ordermonth,sellerid

Oracle: **select extract (month from orderdate)** ordermonth,sellerid,**sum**(amount) totalamount,**count**(amount) totalcount **from** sales **group by** **extract (month from orderdate)**,sellerid **order by** ordermonth,sellerid

I 异构数据库分库

各数据库函数不尽相同，异构库分库需要执行相应的SQL

举例：依旧是分组聚合过滤的例子，由于MySQL和Oracle的“取月”函数不同，需要分别执行不同SQL

	A	B
1	=[[connect@l("mysql"),"MYSQL"],[connect@l("oracle"),"ORACLE"]]	
2	=SQL="select month(orderdate) ordermonth,sellerid,sum(amount) totalamount,count(amount) totalcount from sales group by month(orderdate),sellerid"	
3	fork A1	=SQL.sqltranslate(A3(2))
4		=A3(1).query(B3)
5	=A3.conj()	

I 下回分解

- 对外提供SQL接口
- 有多个distinct的优化
- 外键维表和事实表被分入两个库
- ...

敬请期待

好多乾

润乾线上直销系统



折扣

70%

60%

80%

佣金

玩转好多乾

<http://www.raqsoft.com.cn/wx/hdq-strategy.html>

