Query Sharded Databases

Database Sharding

Database sharding usually follows certain rules in order to distribute data as evenly as possible in different databases.

Example: N MySQL databases, Database No. 1 stores 2013 sales data, Database No. 2 stores 2014 sales data,... Database No. n stores sales data for year 2012+n, as follows:

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
VILJX	9	19700.0	2013-12-31 15:28:05
PJIPE	12	7840.0	2013-12-31 15:28:05

Database No.1 (Year 2013) Sales Data

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

Database No.2 (Year 2014) Sales Data

.....

Database No. n (Year 2012+n) Sales Data

where

Longitudinal connection of the filtered results of different sharded-database tables

Example： For sales data tables in n databases, query all order records with a single sales amount greater than 500.

Database No.1 （Year 2013）
Sales amount greater than 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

Database No.2 （Year 2014）
Sales amount greater than 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

.....

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
VILJX	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

Longitudinal connection of the filtered results of sharded-database by program

where

Longitudinal connection of the filtered results of different sharded-database tables

Example： For sales data tables in n databases, query all order records with a single sales amount greater than 500.

Sequence table

	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()	

When the data volume is large:

Cursor

	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

Reordering the sorted results by a field in each sharded-database table

Example： For sales data tables of n databases, sort in descending order by order amount

Database No.1（Year 2013）

sort in descending order by order amount

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

Database No.2（Year 2014）

sort in descending order by order amount

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

.....

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

Sort the sorted results of sharded-database by order date again

(The result set of the sharded-database is orderly, and the merging algorithm can be used to merge orderly, so as to improve the efficiency.)

order by

Reordering the sorted results by a field in each sharded-database table

Example： For sales data tables of n databases, sort in descending order or ascending order by order amount

Ascending order

	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)	

Descending order

	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

Reordering the sorted results by a field in each sharded-database table

Example： For sales data tables of n databases, sort in ascending order by order amount

Sequence table

	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)	

When the data volume is large:

Cursor

	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

Based on the sorted result, Fetch Y records after skipping X records.

Example： For sales data table of n databases, which is sorted in descending order by order amount, fetch 10 records after skipping the first record.

Database No.1（Year 2013）

sort in descending order by order amount

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

Database No.2（Year 2014）

sort in descending order by order amount

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

.....

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

fetch 10 records after skipping the first record

Based on the result after sharded-database sorting + orderly merging, Fetch Y records after skipping X records.

（We can fetch X+Y records from sharded-database， but can't jump between sharded-databases.）

limit Y offset X

Based on the sorted result, Fetch Y records after skipping X records.

Example： For sales data table of n databases, which is sorted in descending order by order amount, fetch 10 records after skipping the first record.

Sequence table

	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)	

When the data volume is large：

Cursor

	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

The aggregated results of the sharded-database tables need to be re-aggregated, but there are some points for attention.

sum： Sub-database sum，after aggregation is still sum.

count： Sub-database count，sum is needed for aggregation.

avg： Sum and count need to be calculated separately before avg is calculated.

max\min： same as sum

sum/count/avg/max/min

The aggregated results of the sharded-database tables need to be re-aggregated, but there are some points for attention.

Example: For sales data table of n databases, calculate total sales, number of orders, min/max sales value and average sales value.

	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

The results of grouping aggregation of each sharded-database table can not be merged simply when they are grouped and aggregated again.

Example: For sales data table of n sub-databases, group by salesperson ID, calculate sum of sales amount, max and min of sales amount and number of orders.

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

group by

The results of grouping aggregation of each sharded-database table can not be merged simply when they are grouped and aggregated again.

Example： For sales data table of n sharded-databases, group by salesperson ID, calculate sum of sales amount, max and min of sales amount and number of orders.

The result is small for each sharded-database, and the result set of the re-aggregation will only be smaller, use **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)	

group by

The results of grouping aggregation of each sharded-database table can not be merged simply when they are grouped and aggregated again.

Example： For sales data table of n sharded-databases, group by salesperson ID, calculate sum of sales amount, max and min of sales amount and number of orders.

The result set is large for each sharded-database, and the result set of the re-aggregation is not large, use **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)	

group by

The results of grouping aggregation of each sharded-database table can not be merged simply when they are grouped and aggregated again.

Example： For sales data table of n sharded-databases, group by salesperson ID, calculate sum of sales amount, max and min of sales amount and number of orders.

The result set is large for each sharded-database, and the result set of the re-aggregation is also large, use **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

Each sharded-database table only groups and aggregates, and all results are filtered after grouping and aggregation is completed completely.

Example： For sales data tables of n sharded-databases, group by month and salesperson ID, get data with average sales less than 10,000.

Database No.1 (Year 2013)
Group by month, sellerid, calculate total amount of sales and total number of orders

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

Database No.2 (Year 2014)
Group by month, sellerid, calculate total amount of sales and total number of orders

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	6	9428.0

When the amount of data is large, the results are returned orderly according to the grouping dimension in the sharded-database. The merging algorithm can be used to merge the results of different sharded-databases in an orderly way. The results after merging are still orderly, and can continue to be filtered after ordered grouping and aggregation.
(Note that there should be no having in the sharded-database)

group by having

Each sharded-database table only groups and aggregates, and all results are filtered after grouping and aggregation is completed completely.

Example： For sales data tables of n sharded-databases, group by month and salesperson ID, get data with average sales less than 10,000.
Aggregate computation is done only after aggregation, and then filter is implemented using function select.

	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)	

For large amounts of data, cursor needs to be used (select function is also applicable to cursor)

	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 is equivalent to select x group by x，and the implementation method is same as group.

Each sharded-database table only does the de-duplication operation, and all the results do the union (de-duplication), so it can not be merged simply.

Example： For sales data tables of n sharded-databases， get the ID of all salesmen who have transaction records with customer number `YZ'.

Database No.1 (Year 2013)
Get the ID of all salesmen who have transaction records with customer number `YZ' within the year.

sellerid
1
2
3
4
5
6
7

Database No.2 (Year 2014)
Get the ID of all salesmen who have transaction records with customer number `YZ' within the year.

SELLERID
2
4
7
13
16
17
18

.....

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

Obviously, we need to get the union(deduplication) of the results of each sharded-database, rather than merge simply.

distinct

distinct x is equivalent to select x group by x，and the implementation method is same as group.

Each sharded-database table only does the de-duplication operation, and all the results do the union (de-duplication), so it can not be merged simply.

Example： For sales data tables of n sharded-databases， get the ID of all salesmen who have transaction records with customer number `YZ'.

	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)	

For large amounts of data, we can use **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)

Each sharded-database table only does the de-duplication operation, and all the results do the union and de-duplication and then count.

Example: For sales data tables of n sharded-databases, count the number of salesmen with a single sales amount greater than 29,000.

Database No.1 (Year 2013)
Select salesmen with a single sales amount greater than 29,000

sellerid
2
5
6
8
16
18

Database No.2 (Year 2014)
Select salesmen with a single sales amount greater than 29,000

SELLERID
3
4
6
7
14
15

.....

COUNTRESULT
11

When the result set of de-duplication is large in the sharded-database, it can be sorted separately in each sharded-database, then calculate the union by merging algorithm using the ordered characteristic, and finally count.

count(distinct)

Each sharded-database table only does the de-duplication operation, and all the results do the union and de-duplication and then count.

Example： For sales data tables of n sharded-databases，count the number of salesmen with a single sales amount greater than 29,000.

For small amount of data，we can use **query**, **merge@u** and **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()	

For large amount of data with small result set， we can use **cursor**, **mergex@u** and 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 needs to group(distinct) the content of "distinct" and "group by".

Each sharded-database table only performs de-duplication operation, and all results do the union(deduplication) and then count.

Example: For sales data tables of n sharded-databases, group by month, count the number of salesmen with a single sales amount greater than 10,000.

Database No.1 (Year 2013)
Select records with single sales amount greater than 10,000, deduplicate by month and sellerid, and sort

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

Database No.2 (Year 2014)
Select records with single sales amount greater than 10,000, deduplicate by month and sellerid, and sort

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

When the result set of de-duplication is large in the sharded-database, it can be sorted separately in each sharded-database, then calculate the union by merging algorithm using the ordered characteristic, and finally count.

count(distinct) group by

count(distinct) group by needs to group(distinct) the content of "distinct" and "group by".

Each sharded-database table only performs de-duplication operation, and all results do the union(deduplication) and then count.

Example: For sales data tables of n sharded-databases, group by month, count the number of salesmen with a single sales amount greater than 10,000.

	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)	

For large amount of data with small result set, we can use **cursor, mergex@u** and **groups@o**

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

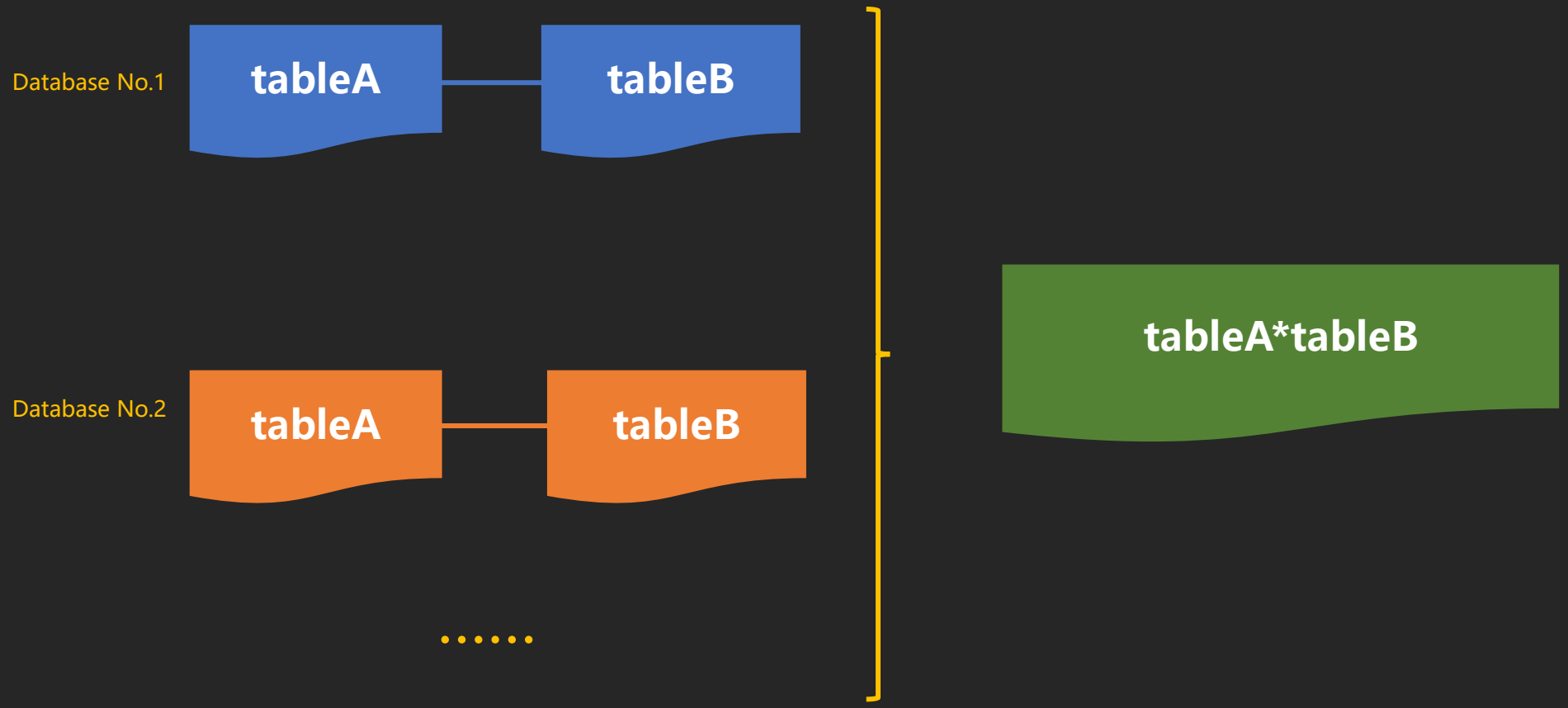
For large amount of data with large result set, we can use **cursor, mergex@u** and **groupx@o**

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

Queries with JOIN

Reasonable data distribution solution can make JOIN only be carried out in sharded-databases, and JOIN is no longer processed in the aggregation stage.

For the solution of synchronous sharded-database, both FULL JOIN and LFET JOIN can be supported, but different types of JOIN need different distribution modes.



Same dimension table and main sub table

product table	
sid ★ Primary key	Product number
producttime	
workshopnum	
…	…

1 : 1

quality table	
sid ★ Primary key	Product number
qlevel	
…	…

Same dimension table: The primary key of table A is associated with the primary key of table B. A and B are called the same dimension table. The same dimension table is a one-to-one relationship, and the relationship between the same dimension table is equal.

orderinfo (Main table)	
orderid ★ Primary key	
customer	
date	
country	
…	…

1 : N

orderdetail (Sub table)	
orderid ★ Primary key	
sid ★ Primary key	
productid	
price	
num	
…	…

Main sub table: The primary key of Table A is associated with part of the primary keys of Table B. A is called the main table and B is called the sub table. The main sub table is a one-to-many relationship.

Foreign key table（Dimension table）

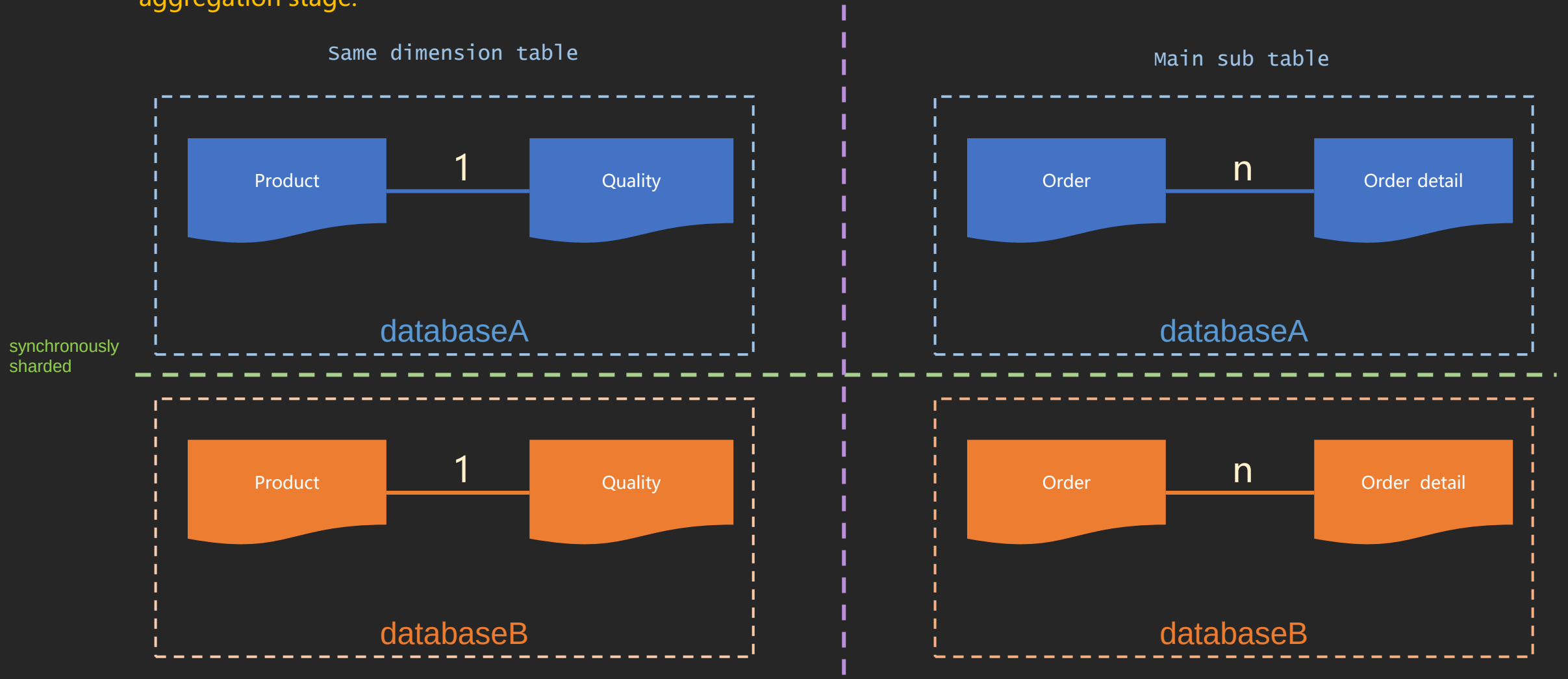
orderdetail	
orderid ★ Primary key	
sid ★ Primary key	
productid	
price	
num	

product	
pid ★ Primary key	
name	
area	
…	…

Some field of Table A are associated with the primary key of Table B. The field associated with the primary key of table B in table A is called the foreign key of A to B, and B is also called the foreign key table of A. The foreign key table is a many-to-one relationship.

Same dimension table and main sub table are synchronously sharded

After data is sharded, join can be processed in each sharded-database, and join need not be considered in the aggregation stage.



Same dimension table

The same dimension tables synchronously sharded can be merged after being joined separately in each sharded-database.

Example： In n sharded-databases, product table and product quality table are synchronously sharded. Group according to workshop number, and count the number of products whose product quality grade is less than 3.

	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)	

For large amount of data， use cursor.

	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)	

Main sub table

The main and sub table synchronously sharded can be merged after being joined separately in each sharded-database.

Example： In n sharded-databases, order table and order detail table are synchronously sharded, group by country and product id, count the total number of products.

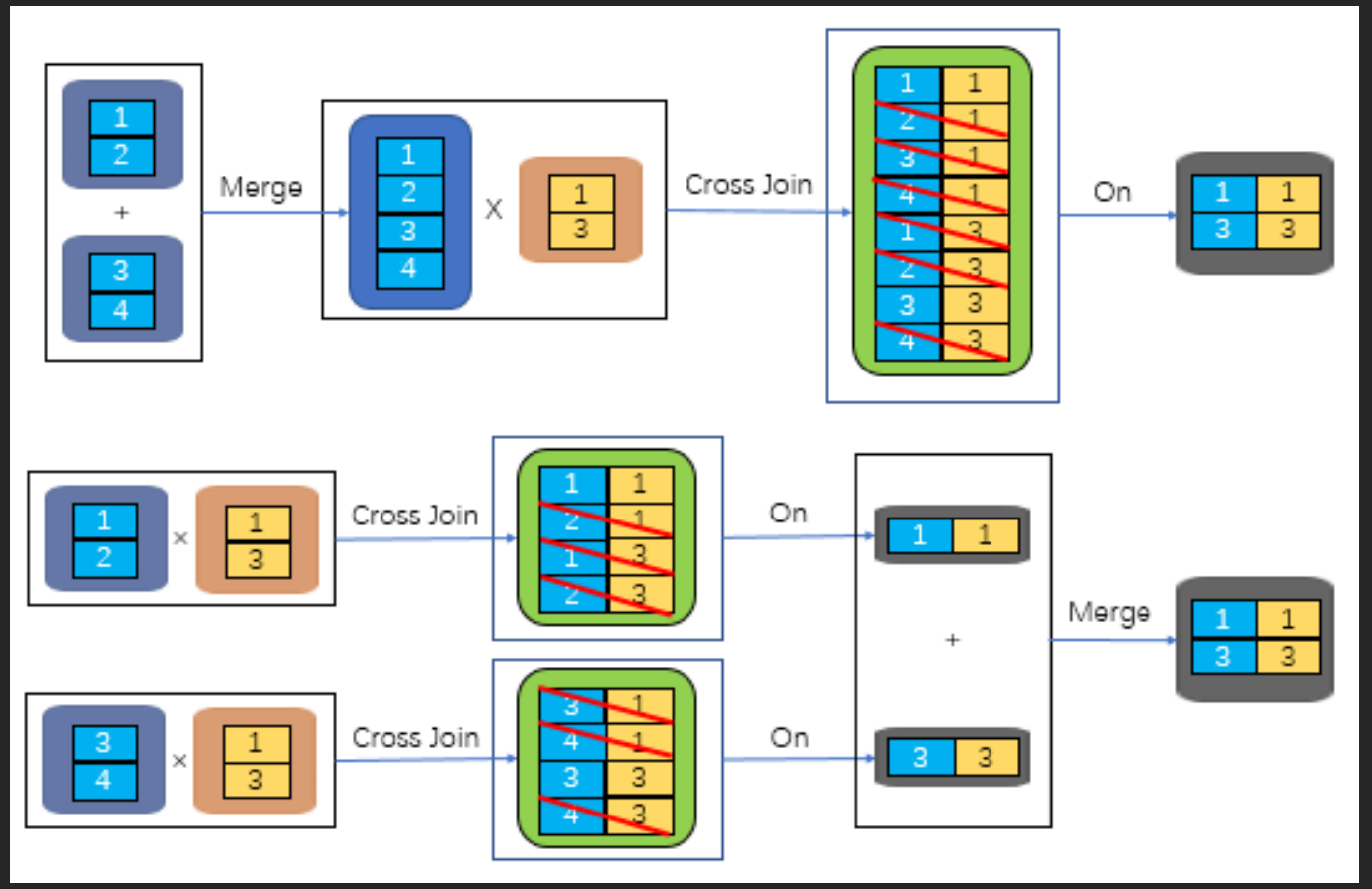
	A	B
1	=n.(connect("mysql"+string(~)))	
2	=SQL="**select** t1.country country,t2.productid productid,**sum**(t2.num) totalnum **from** orderinfo t1 **join** orderdetail t2 **on** t1.orderid**=**t2.orderid **group by** t1.country,t2.productid **order by** t1.country,t2.productid"	
3	fork A1	=A3.**query@x**(SQL)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

For large amount of data， use cursor.

	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)	

Foreign key dimension table is redundantly copied into each sharded-database

If the dimension table (right table) has the same copy in each sharded-database, then the fact table (left table) (each sharded-database has its own part) is joined after aggregation, which is equivalent to the result that the fact table and the dimension table in each sharded-database are joined first and then aggregated. As shown in the picture:



Foreign key table（Dimension table）

Each sharded-database has the same complete foreign key table. Join can be done separately in each sharded-database and then aggregate the result.

Example： In n sharded-databases, the order detail table is evenly distributed in each sharded-database, and the product table is redundant in each sharded-database. Filter out all orders with product area east.

	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()**	

For large amount of data，use cursor.

	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()**	

Sharded-databases with different structure(Heterogeneous Database)

The functions of different databases are not the same, and the sharded-database of heterogeneous databases needs to execute corresponding SQL.

Example: For the same example of grouping aggregation and filtering, because MySQL and Oracle have different "month fetching" function, they need to execute different SQL separately.

Standard function	Meaning	Oracle	MySQL
YEAR(d)	Fetch year	EXTRACT(YEAR FROM d)	YEAR(d)
MONTH(d)	Fetch month	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

Sharded-databases with different structure(Heterogeneous Database)

The functions of different databases are not the same, and the sharded-database of heterogeneous databases needs to execute corresponding SQL.

Example： For the same example of grouping aggregation and filtering, because MySQL and Oracle have different "month fetching" function, they need to execute different SQL separately.

	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()	

To be continued in the next chapter

- Providing SQL Interface to Outside
- Optimizing with multiple distinct
- Foreign key dimension table and fact table are divided into two databases
- …

Coming soon