esProc

Innovated data computing engine

Performance optimization - Join

Issued by Raqsoft

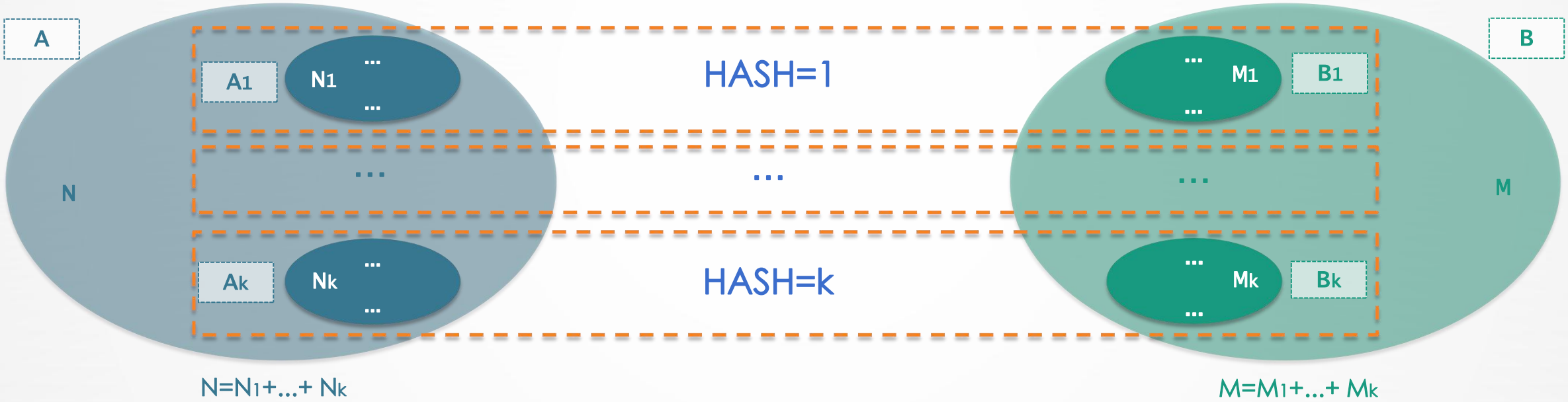
Contents

Join and traditional calculation method

The definition of JOIN in SQL is very simple, that is, two sets (tables) are filtered according to some condition after Cartesian product.

Relational databases usually use HASH method to realize Join, that is, to calculate the HASH value of the related field separately, to put together the records with the same Hash value, and then to make a small-scale traversal comparison!

HASH JOIN Principle in Memory :

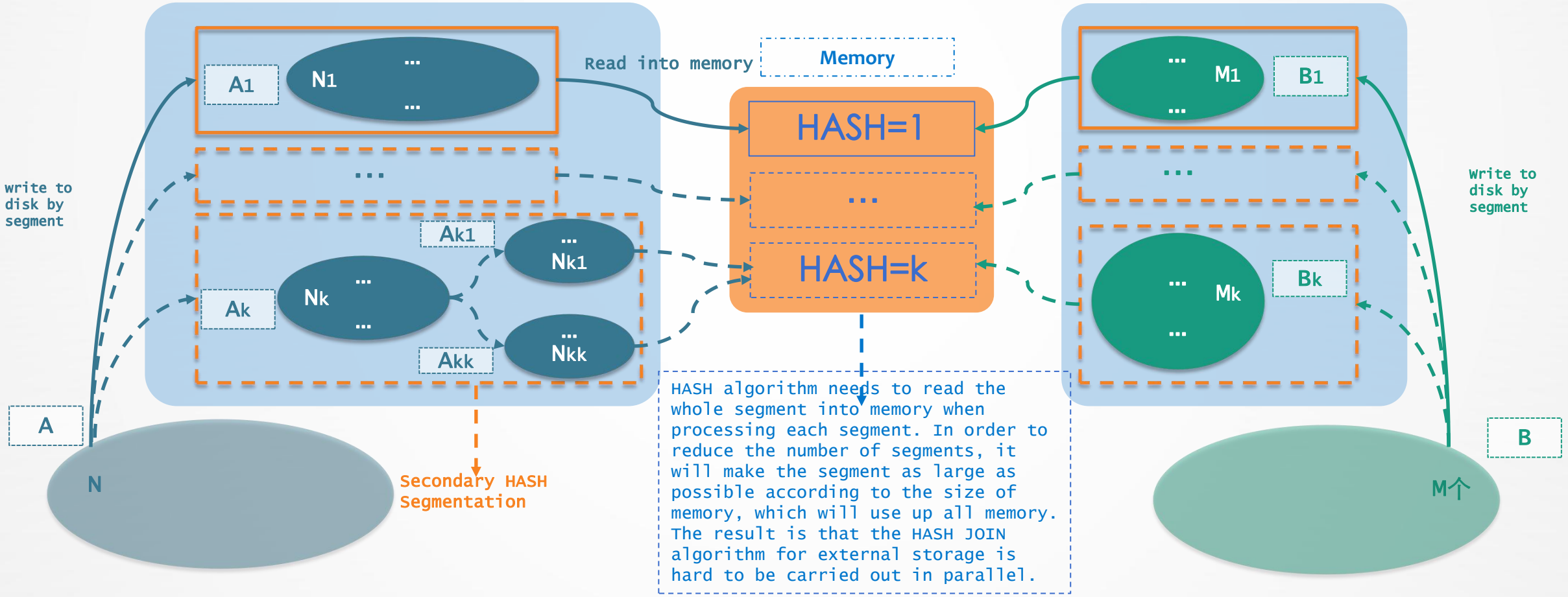


Number of comparisons conventionally : N*M= (N1+...+ Nk) * (M1+...+ Mk) ; Number of comparisons after hash: N1*M1+N2*M2 +...+Nk*Mk
Obviously, the number of the former is generally much greater than that of the latter. (K is the range of HASH values)

Join and traditional calculation method

When the two tables to JOIN are too large to fit in memory, the relational database still adopts HASH segmentation technology. According to the HASH value of the Join field, the data is divided into several piles, each pile is small enough to be loaded in memory and then use in-memory HASH algorithm.

HASH JOIN Principle of external storage:



Analysis of Join Operation

Common Types of Equivalent JOIN

Foreign key table

Main sub table

Same dimension table

In reality, most JOINs are equivalent JOINs. The above three JOINs have covered most of the equivalent JOINs.
By making full use of these features, simpler writing formats and more efficient computing performance can be obtained.

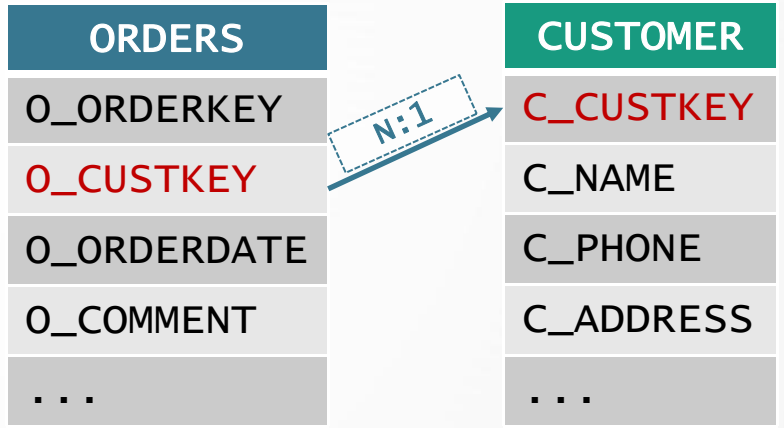
Analysis of Join Operation – Foreign key table

1. Some fields of Table A are associated with the primary key of Table B. Table A is called fact table and table B is called dimension table.

2. 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.

3. Foreign key table is a many-to-one relationship, mainly JOIN and LEFT JOIN, and FULL JOIN is not usually used.

4. Typical example: order table and customer information table

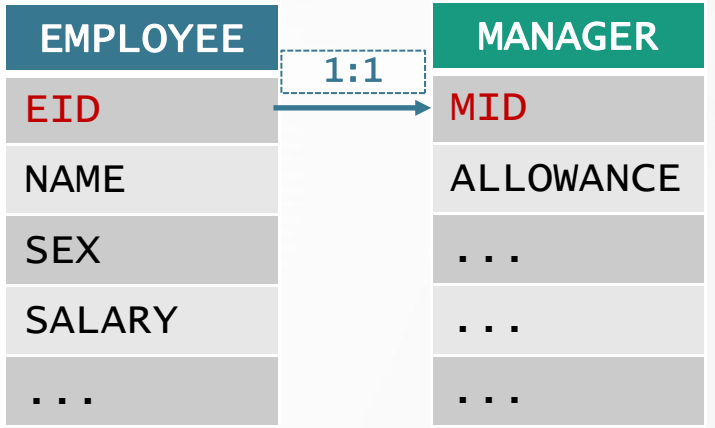


Analysis of Join Operation – Same dimension table

1. The primary key of Table A is associated with the primary key of Table B. A and B are called the same dimension tables.

2. The same dimension table is a one-to-one relationship. JOIN, LEFT JOIN and FULL JOIN may happen.

3. Typical example: employee table and manager table

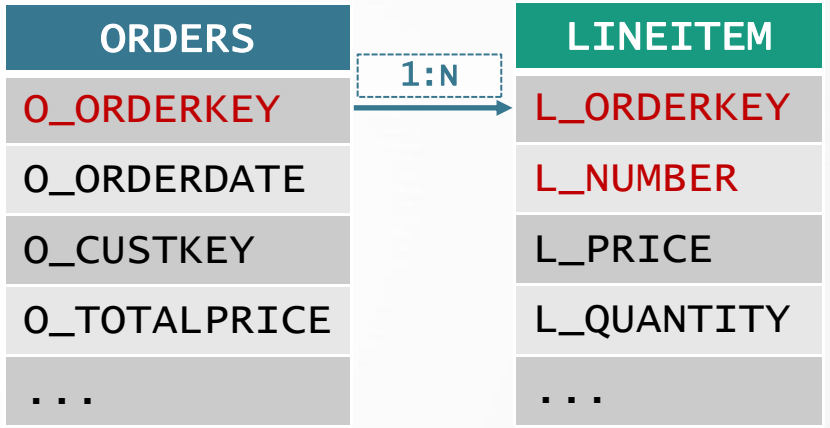


Analysis of Join Operation – Main sub table

1. The primary key of table A is associated with part of the primary key of table B. A is called the main table and B is called the sub-table.

2. The main table and sub table are one-to-many relationship, only JOIN and LEFT JOIN, there will be no FULL JOIN

3. Typical example: order table and order detail table



Contents

In-memory foreign key pre-association – Foreign key attribution

Foreign key attribution can be realized when data can fit into memory. Converting the foreign key customer ID in the order table into an attribute pointing to the customer table record, that is, the value of the customer ID is already a record in a customer table, and the field of the record can be directly referenced for calculation.

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10262	RATTC	1996-07-22	14487.0
2	10263	ERNSH	1996-07-23	43818.0
3	10264	FOLKO	2007-12-18	1101.0
4	10265	BLONP	1996-07-25	5528.0
5	10266	WARTH	1996-07-26	7719.0
6	10267	FRANK	1996-07-29	20858.0
7	10268	GROSR	1996-07-30	19887.0
8	10269	WHITC	1996-07-31	456.0

Foreign key
N:1

CUSTOMER

Index	C_CUSTKEY	C_NAME	C_CITY
1	BLONP	The hao	Dalian
2	CACTU	Wayair freight co. LTD	Dalian
3	CENTC	Three jie industrial	Dalian
4	HUNGC	Hardware mechanical	Dalian
5	MEREP	huake	Dalian
6	ALFKI	Sanchuan industrial ...	Tianjin

Changes before and after join between order table and customer table

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10262	[RATTC,Learn the ...	1996-07-22	14487.0
2	10263	[ERNSH,Resources ...	1996-07-23	43818.0
3	10264	[FOLKO,Wuzhou tr...	2007-12-18	1101.0
4	10265	[BLONP,The hao,D...	1996-07-25	5528.0
5	10266	[WARTH,Upgrade t...	1996-07-26	7719.0
6	10267	[FRANK,Trust frien...	1996-07-29	20858.0
7	10268	[GROSR,Light far tr...	1996-07-30	19887.0
8	10269	[WHITC,Chair day ...	1996-07-31	456.0

C_CUSTKEY	C_NAME	C_CITY
ERNSH	Resources are people	Shenzhen

C_CUSTKEY	C_NAME	C_CITY
WARTH	Upgrade the enterprise	Shijiazhuang

Code example

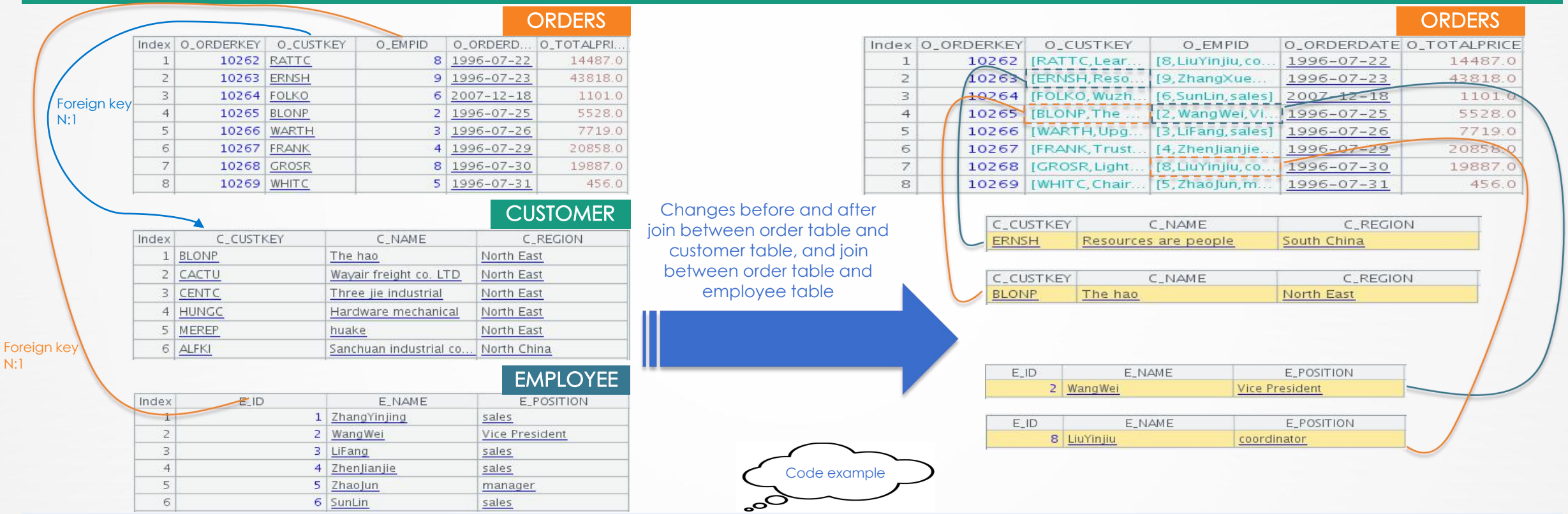
	A	B
1	=ORDERS.switch(O_CUSTKEY,CUSTOMER:C_CUSTKEY)	/Establish foreign key pre-association to convert customer ID into attributes
2	=A1.new(O_CUSTKEY.C_NAME:C_NAME,O_CUSTKEY,O_ORDERDATE,O_TOTALPRICE)	/Query customer's order details
3	=A1.groups(O_CUSTKEY.C_CITY:C_CITY;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize order sales by customer area name

The pre-association can be reused after establishment, that is, the first step only needs to be done once, and the HASH values and comparisons need not be calculated again when the two fields are joined in the future, which can greatly improve the performance. The JOIN operation of SQL does not assume the uniqueness of foreign keys pointing to records, and can not use foreign key attribution method. HASH values should be calculated and compared for every join!

In-memory foreign key pre-association – One-time parsing of multiple foreign keys

HASH JOIN algorithm can only parse one join at a time. N actions need to be performed if there are N JOINs. After each join, the intermediate results need to be kept for the next round. The calculation process is much more complex, and the data will be traversed many times.

If JOIN is parsed by attributing foreign keys, all foreign keys can be pre-associated by traversing the fact table once!



	A	B
1	=ORDERS.switch(O_CUSTKEY,CUSTOMER:C_CUSTKEY;O_EMPID,EMPLOYEE:E_ID)	/Parse Join to convert customer ID and employee ID into attributes
2	=A1.groups(O_CUSTKEY.C_REGION:C_REGION,O_EMPID.E_NAME:E_NAME;sum(O_TOTALPRICE):AMOUNT)	/Summarize sales by customer region and salesperson

In-memory foreign key pre-association – Copy foreign key attributes

Foreign key attribution is simple and efficient, but it can not solve the situation of LEFT JOIN. When it does not match with the record of dimension table, it will lead to the loss of foreign key of fact table.

外键N:1

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10262	RATTC	1996-07-22	14487.0
2	10263	ERNSH	1996-07-23	43818.0
3	10264	FOLKO	2007-12-18	1101.0
4	10265	BLONP	1996-07-25	5528.0
5	10266	WARTH	1996-07-26	7719.0
6	10267	FRANK	1996-07-29	20858.0
7	10268	GROSR	1996-07-30	19887.0
8	10269	WHITC	1996-07-31	456.0

CUSTOMER

Index	C_CUSTKEY	C_NAME	C_REGION
1	CACTU	Wayair freight co. LTD	North East
2	CENTC	Three jie industrial	North East
3	HUNGC	Hardware mechanical	North East
4	MEREP	huake	North East
5	ALFKI	Sanchuan industrial co. L...	North China
6	ANATR	The southeast industrial	North China

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10262	[RATTC,Learn the ker...	1996-07-22	14487.0
2	10263	(null)	1996-07-23	43818.0
3	10264	[FOLKO,Wuzhou trust,...	2007-12-18	1101.0
4	10265	(null)	1996-07-25	5528.0
5	10266	[WARTH,Upgrade the ...	1996-07-26	7719.0
6	10267	[FRANK,Trust friend Jo...	1996-07-29	20858.0
7	10268	[GROSR,Light far trade...	1996-07-30	19887.0
8	10269	[WHITC,Chair day cult...	1996-07-31	456.0

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE	CUSTOMER_fk
1	10262	RATTC	1996-07-22	14487.0	RATTC
2	10263	ERNSH	1996-07-23	43818.0	(null)
3	10264	FOLKO	2007-12-18	1101.0	FOLKO
4	10265	BLONP	1996-07-25	5528.0	(null)
5	10266	WARTH	1996-07-26	7719.0	WARTH
6	10267	FRANK	1996-07-29	20858.0	FRANK
7	10268	GROSR	1996-07-30	19887.0	GROSR
8	10269	WHITC	1996-07-31	456.0	WHITC

The record that does not match is null

C_CUSTKEY	C_NAME	C_REGION
RATTC	Learn the kernel trade	East China

C_CUSTKEY	C_NAME	C_REGION
FOLKO	Wuzhou trust	East China

Not only to convert foreign keys into attributes, but also to solve the LEFT JOIN situation, how to achieve pre-association?

Code example

	A	B
1	=CUSTOMER.keys(C_CUSTKEY)	/Setting primary Key as Customer ID
2	=ORDERS.join(O_CUSTKEY,A1,~:CUSTOMER_fk)	/Establish pre-association
3	=A2.select(CUSTOMER_fk.C_REGION=="North China").sum(O_TOTALPRICE)	/Summarize sales of region “North China”

In-memory foreign key pre-association – Multiple copied foreign key attributes

On the basis of the previous algorithm, the example of multi-field foreign key is as follows:

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_EMPID	O_ORDERD...	O_TOTALPR...
1	10262	RATTC	8	1996-07-22	14487.0
2	10263	ERNSH	9	1996-07-23	43818.0
3	10264	FOLKO	6	2007-12-18	1101.0
4	10265	BLONP	2	1996-07-25	5528.0
5	10266	WARTH	3	1996-07-26	7719.0
6	10267	FRANK	4	1996-07-29	20858.0
7	10268	GROSR	8	1996-07-30	19887.0
8	10269	WHITC	5	1996-07-31	456.0

Foreign key N:1

CUSTOMER

Index	C_CUSTKEY	C_NAME	C_REGION
1	CACTU	Wayair freight co. LTD	North East
2	CENTC	Three jie industrial	North East
3	HUNGC	Hardware mechanical	North East
4	MEREP	huake	North East
5	ALFKI	Sanchuan industrial co...	North China
6	ANATR	The southeast industrial	North China

Foreign key N:1

EMPLOYEE

Index	E_ID	E_NAME	E_POSITION
1	1	ZhangYinjing	sales
2	2	WangWei	Vice President
3	3	LiFang	sales
4	4	ZhenJianjie	sales
5	6	SunLin	sales
6	7	JingShipeng	sales

Index	O_ORDERKEY	O_CUSTKEY	O_EMPID	O_ORDERDA...	O_TOTALPRI...
1	10262	[RATTC,Lea...	[8,LiuYinjiu,...	1996-07-22	14487.0
2	10263	(null)	[9,ZhangXu...	1996-07-23	43818.0
3	10264	[FOLKO,Wuz...	[6,SunLin,sal...	2007-12-18	1101.0
4	10265	(null)	[2,WangWei,...	1996-07-25	5528.0
5	10266	[WARTH,Up...	[3,LiFang,sal...	1996-07-26	7719.0
6	10267	[FRANK,Tru...	[4,ZhenJianji...	1996-07-29	20858.0
7	10268	[GROSR,Ligh...	[8,LiuYinjiu,...	1996-07-30	19887.0
8	10269	[WHITC,Chai...	(null)	1996-07-31	456.0

Index	O_ORDER...	O_CUSTKEY	O_EMPID	O_ORDER...	O_TOTAL...	CUSTOME...	EMPLOYEE...
1	10262	RATTC	8	1996-07...	14487.0	RATTC	8
2	10263	ERNSH	9	1996-07...	43818.0	(null)	9
3	10264	FOLKO	6	2007-12...	1101.0	FOLKO	6
4	10265	BLONP	2	1996-07...	5528.0	(null)	2
5	10266	WARTH	3	1996-07...	7719.0	WARTH	3
6	10267	FRANK	4	1996-07...	20858.0	FRANK	4
7	10268	GROSR	8	1996-07...	19887.0	GROSR	8
8	10269	WHITC	5	1996-07...	456.0	WHITC	(null)

The record that does not match is null

C_CUSTKEY	C_NAME	C_REGION
RATTC	Learn the kernel trade	East China

C_CUSTKEY	C_NAME	C_REGION
FOLKO	Wuzhou trust	East China

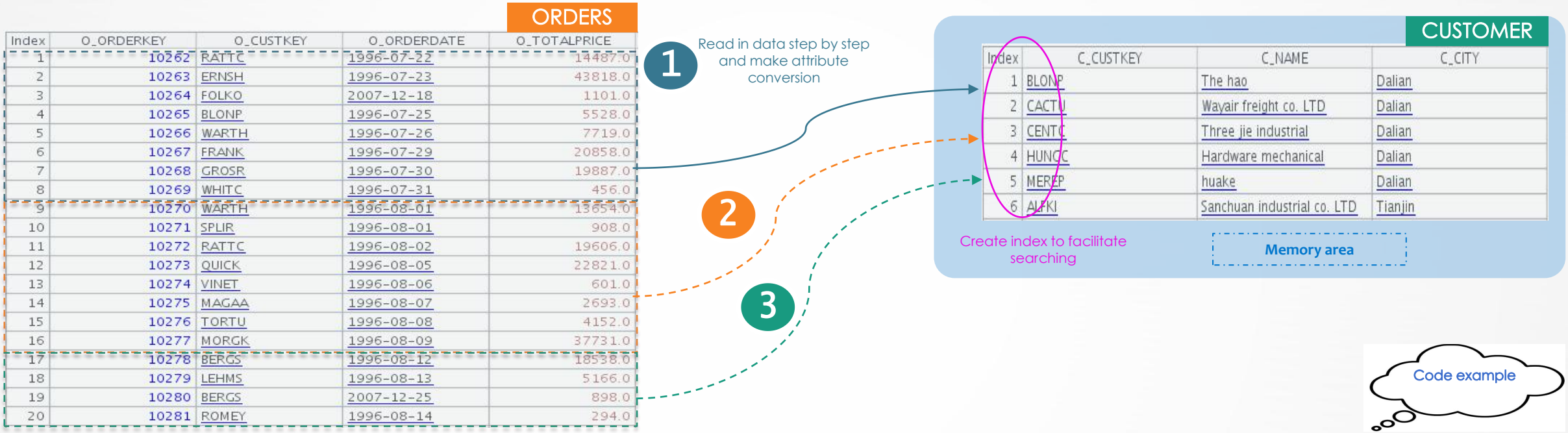
E_ID	E_NAME	E_POSITION
9	ZhangXuemei	sales

Code example

	A	B
1	=CUSTOMER.keys(C_CUSTKEY),EMPLOYEE.keys(E_ID)	/Setting primary Keys as Customer ID, employee ID
2	=ORDERS.join(O_CUSTKEY,CUSTOMER,~:CUSTOMER_fk;O_EMPID,EMPLOYEE,~:EMPLOYEE_fk)	/Establish pre-association
3	=A2.select(CUSTOMER_fk.C_REGION=="North China" && EMPLOYEE_fk.E_POSITION=="sales").sum(O_TOTALPRICE)	/Summarize sales of region "North China" of a salesperson

Partial Memorized Foreign Key - Temporary Pointing Method

Fact table is too large to fit in memory, while dimension table is small (can be all put in memory). Temporary pointing can be used to process foreign keys, that is, attribute conversion of foreign keys while reading in data.

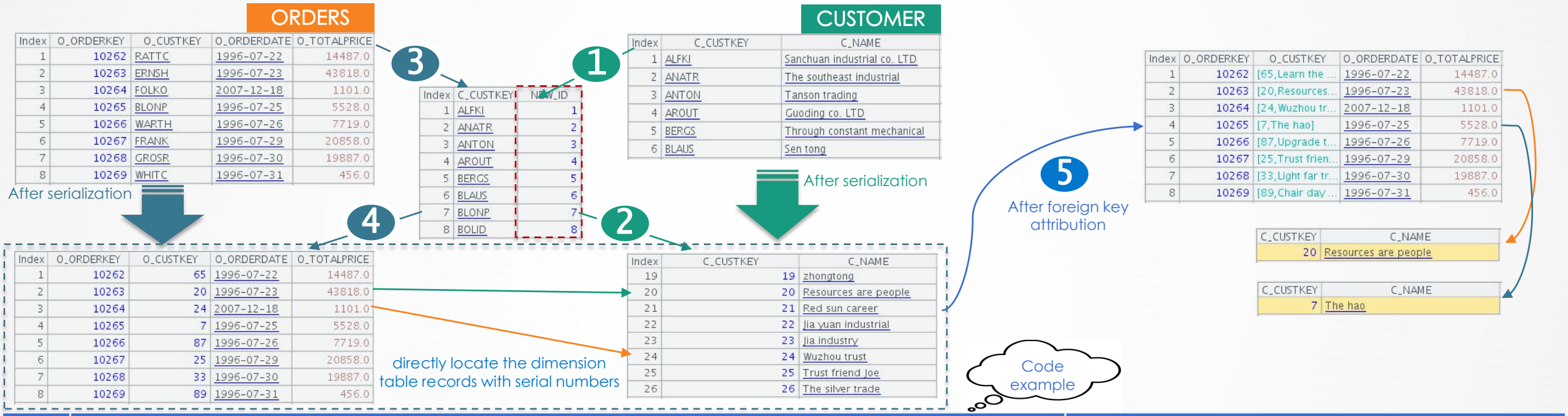


	A	B
1	=file("ORDERS.btx").cursor@b()	/Create cursor for order table records, read in data step by step
2	=A1.switch(O_CUSTKEY,CUSTOMER:C_CUSTKEY)	/Converting the customer ID field in the order table into a record of the customer table based on the primary key of the customer table when the data flows in
3	=A1.groups(O_CUSTKEY.C_CITY:CITY;sum(O_TOTALPRICE):AMOUNT)	/Summarize the order sales according to the city

HASH value is calculated and comparison is made every time join is done, but dimension table index can be reused after establishment, and it also has the characteristics of parsing all foreign keys at one time and easy to parallel. In actual scenarios, it still has advantages over HASH algorithm.

Partial Memorized Foreign Key – Serialization

Variant of the previous algorithm. That is, if we can convert the primary keys of the dimension table into natural numbers starting from 1, then we can directly locate the dimension table records with serial numbers, without calculating and comparing HASH values.



	A	B
1	=file("ORDERS.btx").cursor@b()	/Create cursor for order table records, read in data step by step
2	=A1.switch(O_CUSTKEY,CUSTOMER:#)	/Establish foreign key join using serial numbers
3	=A1.groups(O_CUSTKEY.C_NAME:C_NAME;sum(O_TOTALPRICE):AMOUNT)	/Grouping and aggregation

Foreign key serialization is essentially equivalent to attribution in external storage, and it also has the same reuse mechanism as in memory; SQL uses the concept of disordered set, even if the foreign key is serialized beforehand, it is difficult for database to take advantage of this feature, and it still calculates HASH values and comparisons.

Partial Memorized Foreign Key – Numbering key

Numbering is a byte-based integer used to represent key values. It locates quickly and is often used to optimize memory indexing and foreign key joins.

ID number	Name
310105197306098l6	Dai Li
…	…

Change to numbering key

ID number	Name
2234072400696791302	Dai Li
…	…

Another method of dealing with discontinuous serial numbers to avoid hash calculation and conflict.

Direct numbering requires at least 10 ^ 17 long-type spaces.

Numbering keys can numberize data layer by layer, and many sub-nodes are empty to reduce memory usage.

Divide 17-bit ID number into 8 layers : 31 | 01 | 05 | 1973 | 06 | 09 | 81 | 6

Bits 1-2: 1-99

1	…	10	11	…	15	16	…	31	…	90	…	99	…

Bits 3-4: 1-99

1	…	10	11	…	15	16	…	21	…	90	…	99	…

Bits 5-6: 1-99

1	…	5	…	…	15	16	…	21	…	90	…	99	…

Bits 7,8,9,10: represent birthday year, 1970 is the benchmark here, start from 1:

1	…	3	…	…	15	16	…	21	…	90	…	99	…

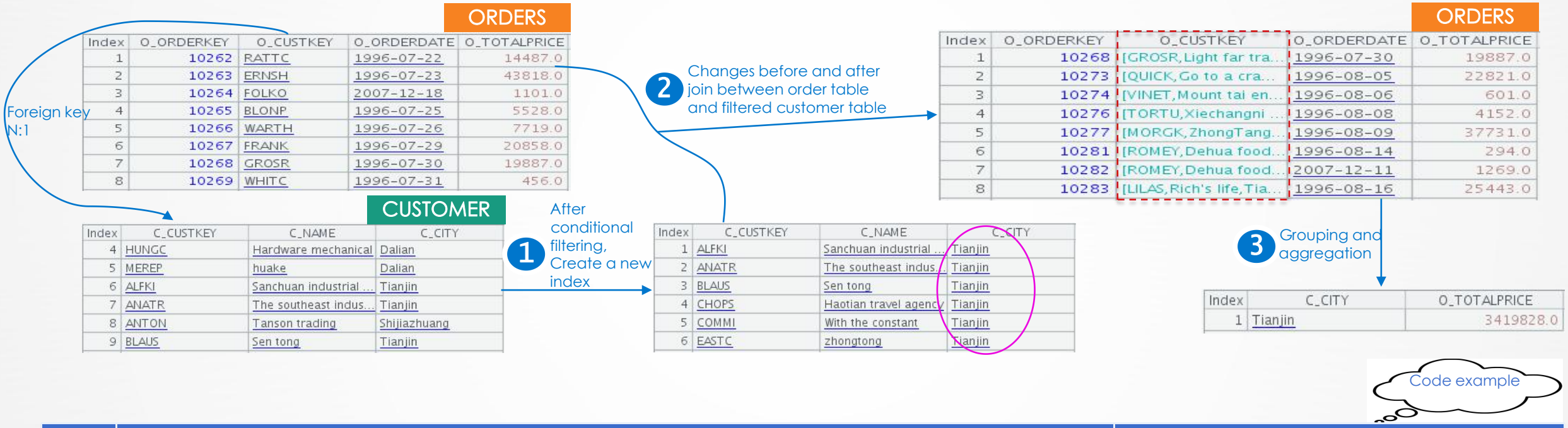
Layering of the remaining 7 bits, just like the above mentioned…

Code example

	A
1	=file("TAX_RETURN.btx").cursor@b()
2	=file("ID_CARDS.btx").import@b().keys@i(cardNo)
3	=A1.switch(cardNo,A2:cardNo)

Dimension table filtering – Utilizing existing index

Dimension table is loaded into memory and indexed. Sometimes it is necessary to join the filtered dimension table. It is necessary to rebuild dimension table index. It is also time-consuming to build dimension table index when dimension table is large. The index of filtered dimension table can be built by using existing dimension table index without recalculating hash values.



Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10262	RATTC	1996-07-22	14487.0
2	10263	ERNSH	1996-07-23	43818.0
3	10264	FOLKO	2007-12-18	1101.0
4	10265	BLONP	1996-07-25	5528.0
5	10266	WARTH	1996-07-26	7719.0
6	10267	FRANK	1996-07-29	20858.0
7	10268	GROSR	1996-07-30	19887.0
8	10269	WHITC	1996-07-31	456.0

Index	C_CUSTKEY	C_NAME	C_CITY
4	HUNGC	Hardware mechanical	Dalian
5	MEREP	huake	Dalian
6	ALFKI	Sanchuan industrial ...	Tianjin
7	ANATR	The southeast indus...	Tianjin
8	ANTON	Tanson trading	Shijiazhuang
9	BLAUS	Sen tong	Tianjin

Index	C_CUSTKEY	C_NAME	C_CITY
1	ALFKI	Sanchuan industrial ...	Tianjin
2	ANATR	The southeast indus...	Tianjin
3	BLAUS	Sen tong	Tianjin
4	CHOPS	Haotian travel agency	Tianjin
5	COMMI	With the constant	Tianjin
6	EASTC	zhongtong	Tianjin

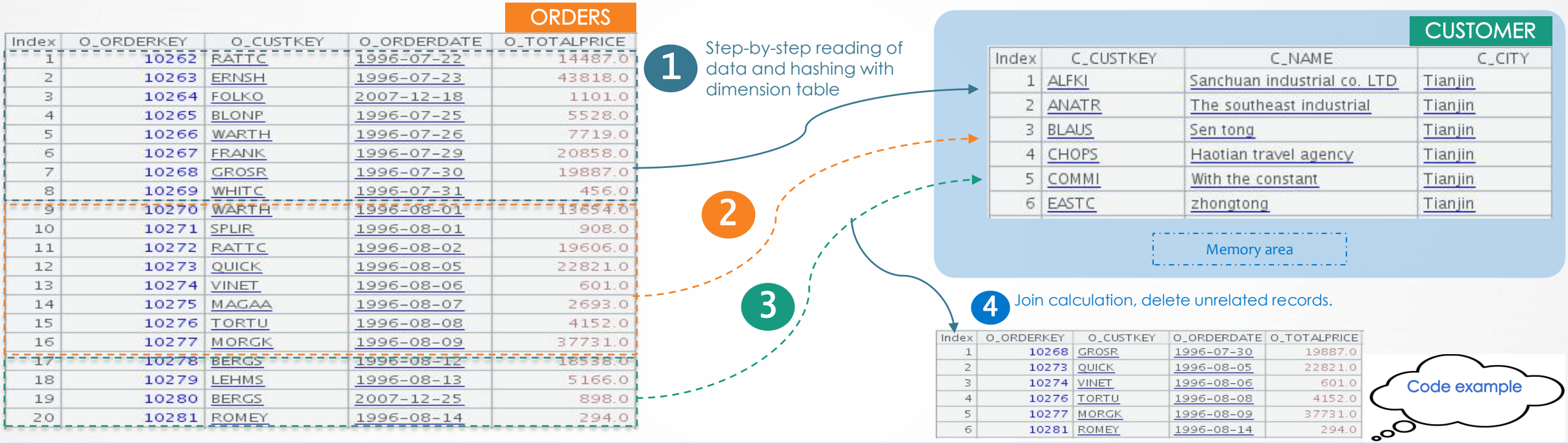
Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10268	[GROSR, Light far tra...	1996-07-30	19887.0
2	10273	[QUICK, Go to a cra...	1996-08-05	22821.0
3	10274	[VINET, Mount tai en...	1996-08-06	601.0
4	10276	[TORTU, Xiechangni ...	1996-08-08	4152.0
5	10277	[MORGK, ZhongTang...	1996-08-09	37731.0
6	10281	[ROMEY, Dehua food...	1996-08-14	294.0
7	10282	[ROMEY, Dehua food...	2007-12-11	1269.0
8	10283	[LILAS, Rich's life, Tia...	1996-08-16	25443.0

Index	C_CITY	O_TOTALPRICE
1	Tianjin	3419828.0

	A	B
1	=CUSTOMER.select@i(C_CITY=="Tianjin")	/Filter the customer table and use the original index to build the index of the filtered customer table
2	=file("ORDERS.ctx").create().cursor().switch@i(O_CUSTKEY,A1:C_CUSTKEY)	/Attribute foreign keys and delete unrelated records
3	=A1.groups(O_ORDERDATE;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize the sales of the order according to the order date

Inner Join – Dimension table field is only used for filtering

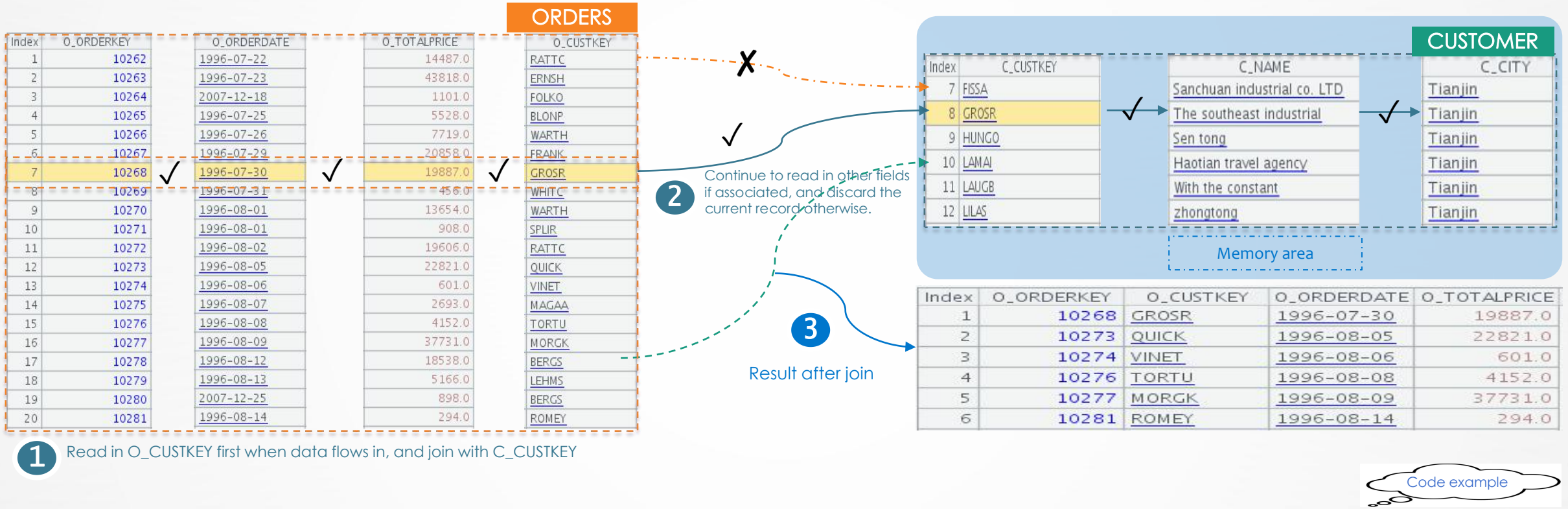
Fact table and dimension table are joined internally, dimension table is only used for filtering. They can read data from fact table and join with filtered dimension table at the same time, discarding records that are not related.



	A	B
1	=file("ORDERS.ctx").create().cursor()	/Create cursor for order table records, read in data step by step.
2	=A1.join@i(O_CUSTKEY,CUSTOMER:C_CUSTKEY)	/When data flows in, associate the customer ID field in the order table with the filtered customer table, and discard records that are not associated.
3	=A2.groups(O_ORDERDATE;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize the sales of the order according to the order date.

Inner Join – Join and filtering when cursor is read out

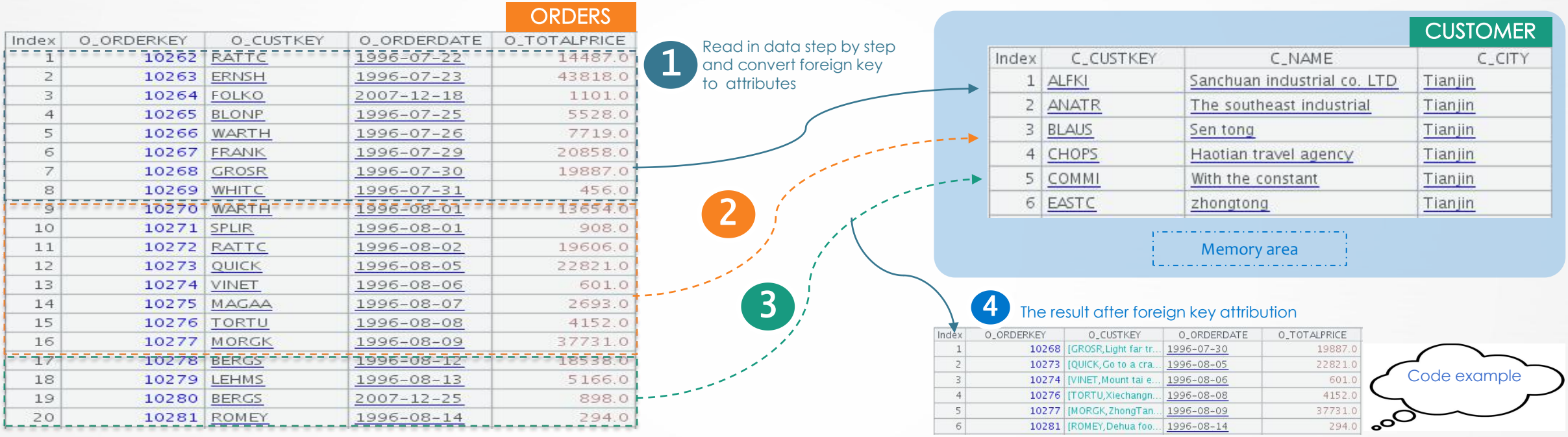
When the cursor reads out, Join and filter. If the record is not associated, it no longer reads out other fields of the record. When more records are filtered out, it can significantly reduce IO operations and improve performance.



	A	B
1	=file("ORDERS.ctx").create().cursor(;CUSTOMER.find(C_CUSTKEY))	/Read in O_CUSTKEY first when data flows in, and join with C_CUSTKEY. Continue to read in other fields if associated, and discard the current record otherwise.
2	=A1.groups(O_ORDERDATE;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize the sales of the order according to the order date.

Inner Join – Attribution at the same time of join filtering

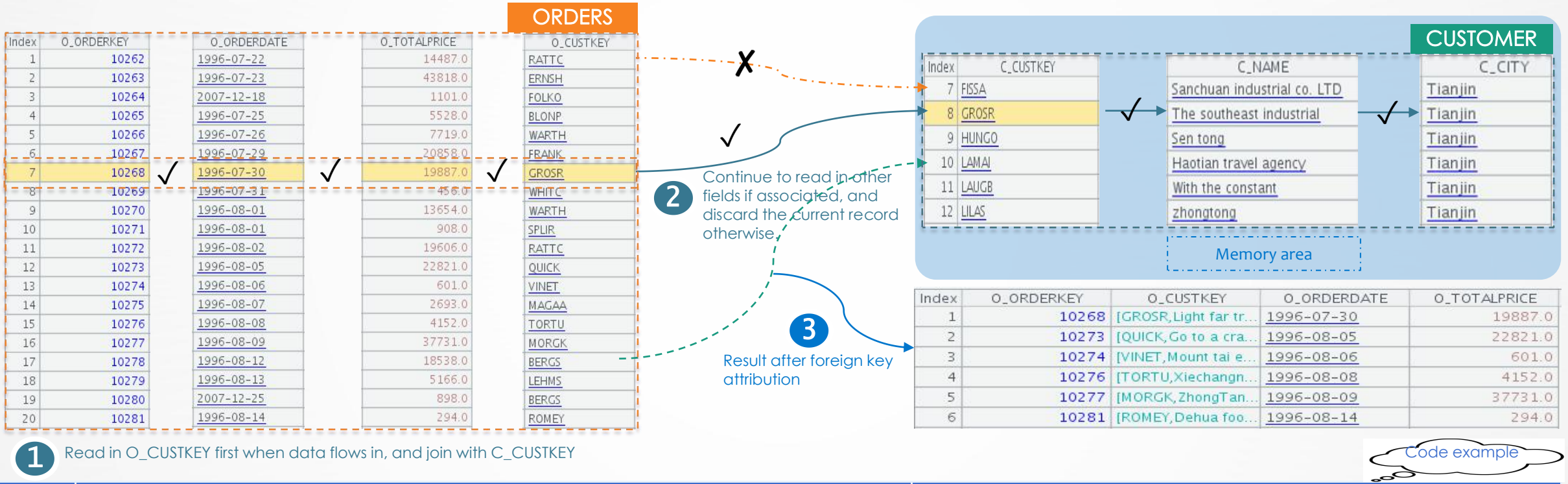
Fact table and dimension table are joined internally. The field of dimension table is used for filtering conditions. We can filter dimension tables first, then read in the data of fact table and join with the filtered dimension table at the same time, discarding records that are not related.



	A	B
1	=file("ORDERS.ctx").create().cursor()	/Create cursor for order table records, read in data step by step.
2	=A1.switch@i(O_CUSTKEY,CUSTOMER:C_CUSTKEY)	/When the data flows in, the customer ID field in the order table and the filtered customer table are attributed to foreign key, and the unrelated records are deleted.
3	=A2.groups(O_CUSTKEY.C_NAME:C_NAME;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize the sales of the order according to the name of the customer company

Inner Join – Join filtering and Attribution when cursor is read out

When the cursor reads out, Join and filter, and then attribute. If the record is not associated, it no longer reads out other fields of the record. When more records are filtered out, it can significantly reduce IO operations and improve performance.



Code example

	A	B
1	=file("ORDERS.ctx").create().cursor(;O_CUSTKEY:CUSTOMER)	/Read in O_CUSTKEY first when data flows in, and attribute foreign key to customer table. Continue to read in other fields if associated, and discard the current record otherwise.
2	=A1.groups(O_CUSTKEY.C_NAME:C_NAME;sum(O_TOTALPRICE):O_TOTALPRICE)	/Summarize the sales of the order according to the name of the customer company

Large dimension table

When the fact table is small (can fit in memory), and the dimension table is too large to fit in memory, JOIN can be transformed into a batch lookup problem, that is, to join with the relevant records after the fact table joining field is found in the original dimension table.

Dimension table(Large)

PRODUCT

Index	P_ID	P_NAME	P_TYPENAME	P_PRICE
1	1	Apple juice	drinks	18.5
2	2	milk	drinks	21.0
3	3	Tomato sauce	condiments	12.0
4	4	salt	condiments	21.0
5	5	Sesame oil	condiments	22.35
6	6	soy sauce	condiments	15.0
7	7	Seafood pow...	Specialty pro...	27.0
8	8	pepper	condiments	40.0
9	9	chicken	Meat/poultry	97.0
10	10	crab	seafood	31.0
11	11	The mass of ...	Daily necessit...	21.0
12	12	German cheese	Daily necessit...	38.0
13	13	lobster	seafood	6.0
14	14	satay	Specialty pro...	23.25
15	15	aginomoto	condiments	15.5
16	16	biscuits	Dim sum	17.45
17	17	pork	Meat/poultry	39.0
18	18	cuttlefish	seafood	62.5
19	19	candy	Dim sum	9.2
20	20	Osmanthus c...	Dim sum	81.0

Fact table(small)

RETURN

Index	L_ORDERKEY	L_PID	L_QUANTITY
1	10858	7	5.0
2	10866	2	21.0
3	10871	16	12.0

1

Fact table(small)

Index	L_ORDERKEY	L_PID	L_QUANTITY
1	10858	7	5.0
2	10866	2	21.0
3	10871	16	12.0

Dimension table records after batch search by fact table(small)

Index	P_ID	P_NAME	P_TYPENAME	P_PRICE
1	2	milk	drinks	21.0
2	7	Seafood powder	Specialty products	27.0
3	16	biscuits	Dim sum	17.45

2

JOIN result

Index	L_ORDER...	L_PID	L_QUANTITY	P_NAME	P_TYPENA...	P_PRICE
1	10858	7	5.0	Seafood p...	Specialty ...	27.0
2	10866	2	21.0	milk	drinks	21.0
3	10871	16	12.0	biscuits	Dim sum	17.45

Code example

	A
1	=file("RETURN.btx").import@b()
2	=file("PRODUCT.btx")
3	=A1.joinx@q(L_PID,A2:P_ID,P_NAME,P_TYPENAME,P_PRICE)
4	=A3.fetch()

A1: Load the return table into memory

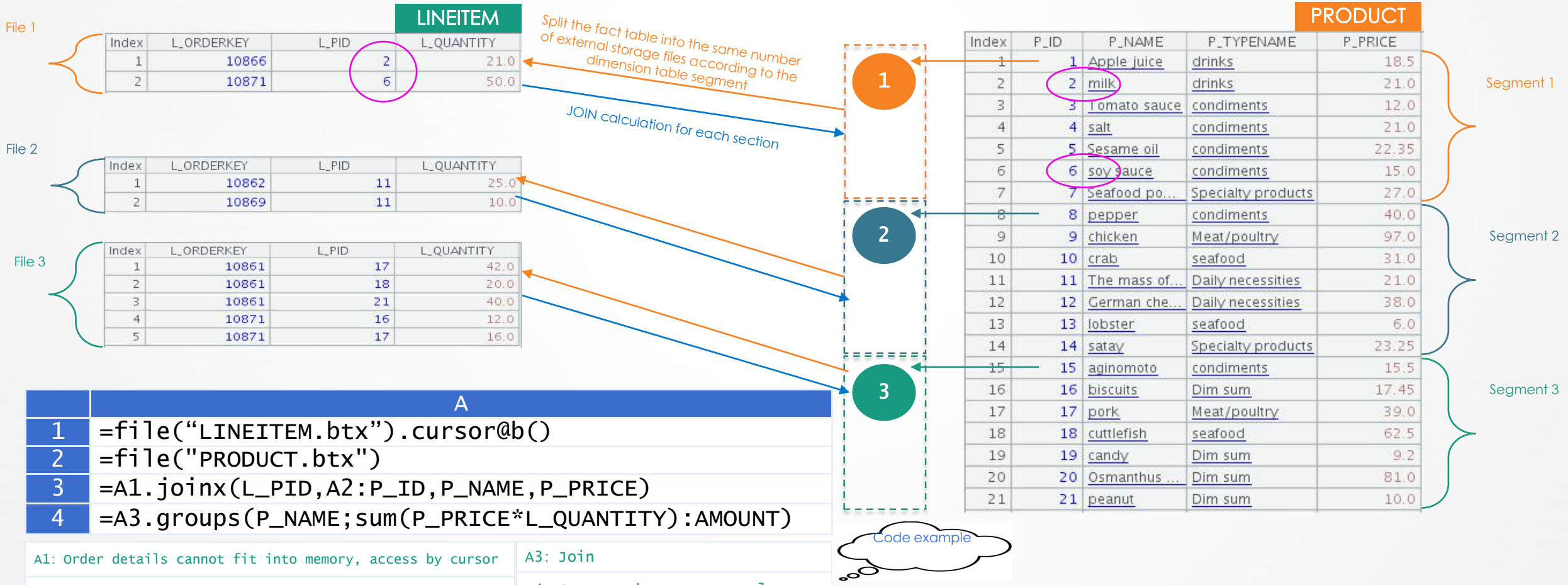
A2: Give the file object of the product table

A3: The return table is joined with the result of Batch Search in the product table according to the return table.

A4: Return table is joined with the reduced product table and the result is calculated.

Unilateral HASH Method

When the fact table and dimension table are too large to fit into memory, the dimension table can be read by the average segment after sorting by the primary key (HASH segment is difficult to guarantee average). When joining, the fact table is divided into the same number of temporary files according to the value of the dimension table segment key. In each file the dimension value corresponds to one segment of the dimension table, so we only need to read in external storage files in turn to join with the segments of dimension table. Compared with the traditional external storage HASH JOIN, it saves the HASH partition of dimension table, and it is impossible to have secondary HASH!



File 1

Index	L_ORDERKEY	L_PID	L_QUANTITY
1	10866	2	21.0
2	10871	6	50.0

File 2

Index	L_ORDERKEY	L_PID	L_QUANTITY
1	10862	11	25.0
2	10869	11	10.0

File 3

Index	L_ORDERKEY	L_PID	L_QUANTITY
1	10861	17	42.0
2	10861	18	20.0
3	10861	21	40.0
4	10871	16	12.0
5	10871	17	16.0

PRODUCT

Index	P_ID	P_NAME	P_TYPENAME	P_PRICE
1	1	Apple juice	drinks	18.5
2	2	milk	drinks	21.0
3	3	Tomato sauce	condiments	12.0
4	4	salt	condiments	21.0
5	5	Sesame oil	condiments	22.35
6	6	soy sauce	condiments	15.0
7	7	Seafood po...	Specialty products	27.0
8	8	pepper	condiments	40.0
9	9	chicken	Meat/poultry	97.0
10	10	crab	seafood	31.0
11	11	The mass of...	Daily necessities	21.0
12	12	German che...	Daily necessities	38.0
13	13	lobster	seafood	6.0
14	14	satay	Specialty products	23.25
15	15	aginomoto	condiments	15.5
16	16	biscuits	Dim sum	17.45
17	17	pork	Meat/poultry	39.0
18	18	cuttlefish	seafood	62.5
19	19	candy	Dim sum	9.2
20	20	Osmanthus ...	Dim sum	81.0
21	21	peanut	Dim sum	10.0

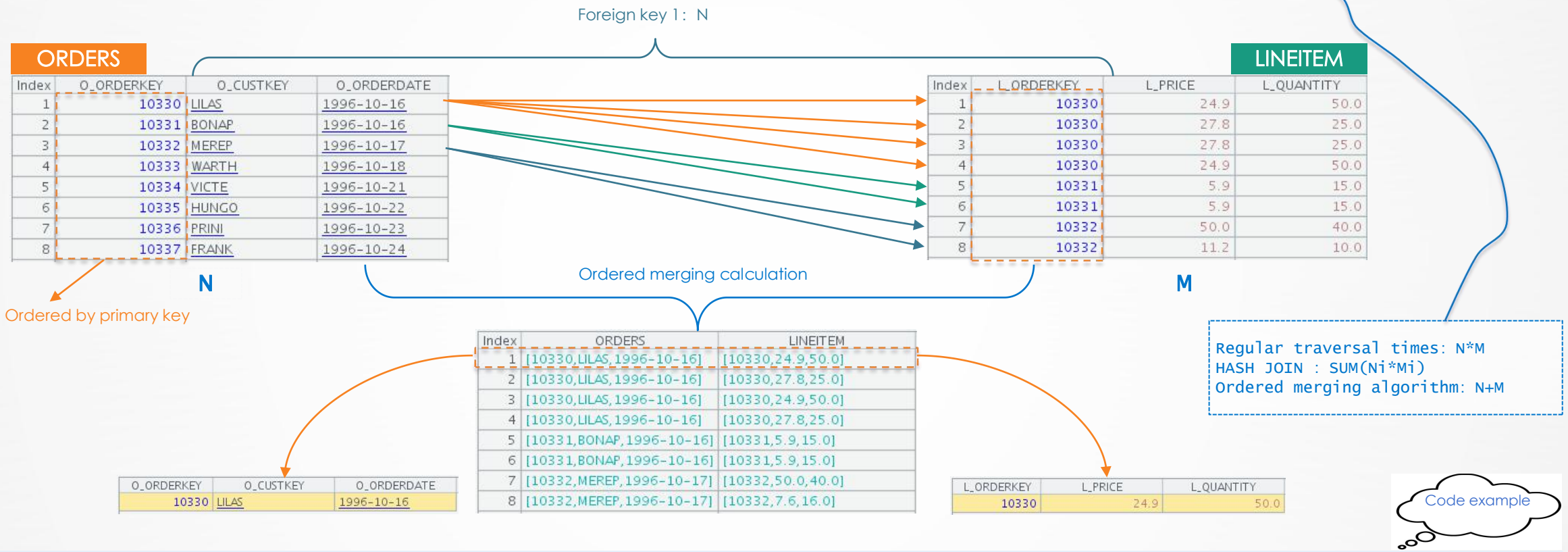
	A
1	=file("LINEITEM.btx").cursor@b()
2	=file("PRODUCT.btx")
3	=A1.joinx(L_PID,A2:P_ID,P_NAME,P_PRICE)
4	=A3.groups(P_NAME;sum(P_PRICE*L_QUANTITY):AMOUNT)

A1: Order details cannot fit into memory, access by cursor	A3: Join
A2: Give the file object of the product table	A4: Group and aggregate sales

Contents

Merge of same dimension table and main sub-table

When the same dimension table or the main sub-table are stored synchronized and orderly, JOIN can be implemented by merging algorithm with one traversal. The complexity is much lower than that of the external storage segmented HASH JOIN.



Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
1	10330	LILAS	1996-10-16
2	10331	BONAP	1996-10-16
3	10332	MEREP	1996-10-17
4	10333	WARTH	1996-10-18
5	10334	VICTE	1996-10-21
6	10335	HUNGO	1996-10-22
7	10336	PRINI	1996-10-23
8	10337	FRANK	1996-10-24

Index	L_ORDERKEY	L_PRICE	L_QUANTITY
1	10330	24.9	50.0
2	10330	27.8	25.0
3	10330	27.8	25.0
4	10330	24.9	50.0
5	10331	5.9	15.0
6	10331	5.9	15.0
7	10332	50.0	40.0
8	10332	11.2	10.0

Index	ORDERS	LINEITEM
1	[10330,LILAS,1996-10-16]	[10330,24.9,50.0]
2	[10330,LILAS,1996-10-16]	[10330,27.8,25.0]
3	[10330,LILAS,1996-10-16]	[10330,24.9,50.0]
4	[10330,LILAS,1996-10-16]	[10330,27.8,25.0]
5	[10331,BONAP,1996-10-16]	[10331,5.9,15.0]
6	[10331,BONAP,1996-10-16]	[10331,5.9,15.0]
7	[10332,MEREP,1996-10-17]	[10332,50.0,40.0]
8	[10332,MEREP,1996-10-17]	[10332,7.6,16.0]

O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
10330	LILAS	1996-10-16

L_ORDERKEY	L_PRICE	L_QUANTITY
10330	24.9	50.0

	A	B
	=ORT=file("ORDERS.btx").cursor@b(),LIT=file("LINEITEM.btx").cursor@b()	/Define variables
1	=joinx(ORT:ORDERS,O_ORDERKEY;LIT:LINEITEM,L_ORDERKEY)	/Ordered merging join
2	=A1.groups(ORDERS.O_CUSTKEY:CUST;sum(LINEITEM.L_PRICE*LINEITEM.L_QUANTITY):AMOUNT)	/Group and aggregation

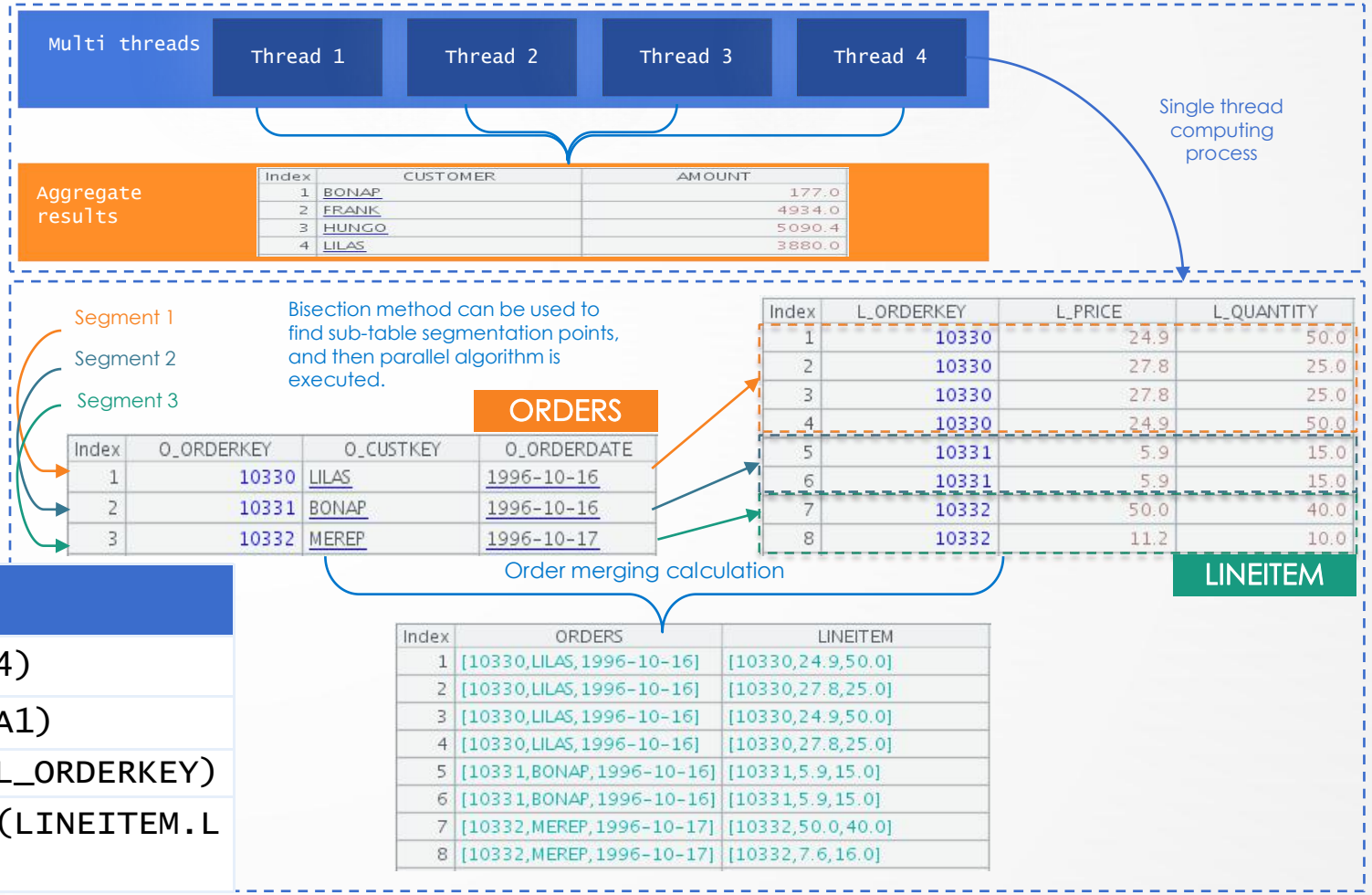
Parallel Merge – Parallel Computing

Parallel computing can significantly improve performance, but traditional HASH JOIN is difficult to achieve parallelism. Parallel HASH segments need to write data to a certain segment at the same time, resulting in shared resource conflicts; while computing a segment will consume almost all memory, other parallel tasks can not be carried out.

Ordered storage is easy to achieve segmented parallelism. Keys are ordered, so the record keys of each segment of the main table belong to a continuous interval, and the sub-table also has this feature, which enables efficient binary search in the sub-table to locate the segment points; that is, data ordering ensures the rationality and efficiency of the segment.

Code example：

	A
1	=file("ORDERS.ctx").create().cursor@m(;;4)
2	=file("LINEITEM.ctx").create().cursor(;;A1)
3	=joinx(A1:ORDERS,O_ORDERKEY;A2:LINEITEM,L_ORDERKEY)
4	=A3.groups(ORDERS.O_CUSTKEY:CUSTOMER;sum(LINEITEM.L_PRICE*LINEITEM.L_QUANTITY):AMOUNT)



Parallel Merge – Generating synchronous data

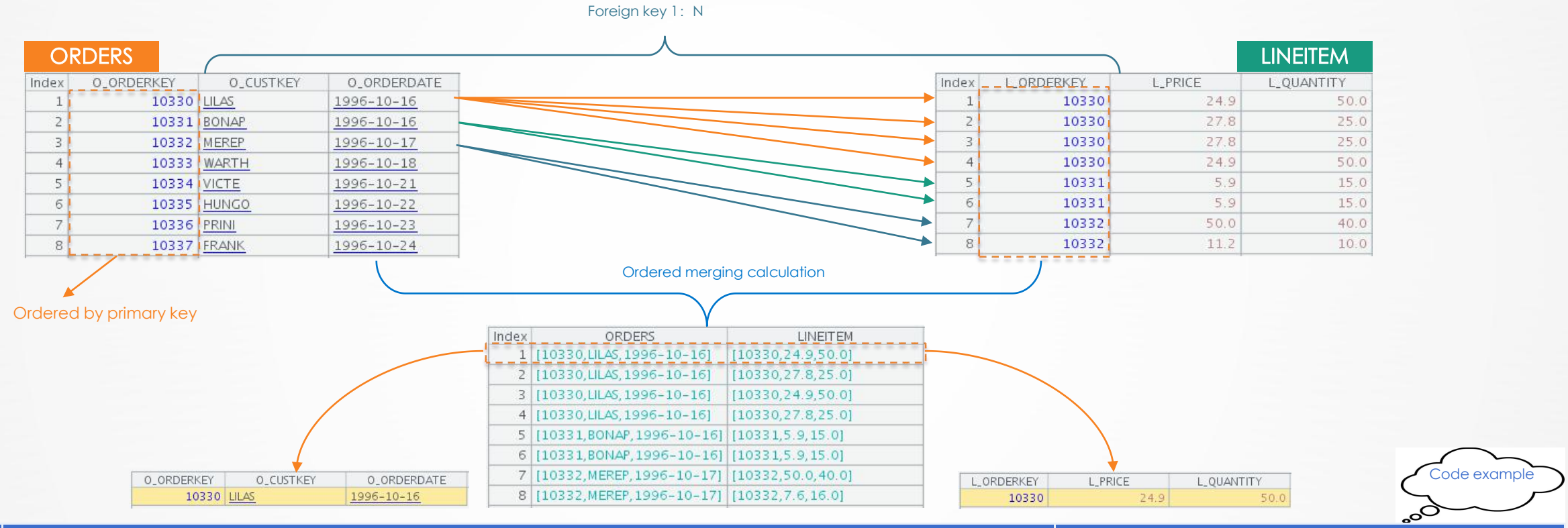
When generating data, it aligns according to a certain benchmark table, which ensures the synchronization of multiple tables during segmentation, and there will be no record misalignment when merging and calculating.

Code for generating group table data to ensure main sub alignment:

	A	B
1	=file("ORDERS.txt").cursor@t(O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)	/Read in orders.txt
2	=A1.sortx(O_ORDERKEY)	/Sort by order id
3	=file("ORDERS.ctx").create(#O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)	/Create and open group table
4	=A3.append(A2)	/Add records from cursor to group table
5	=file("LINEITEM.txt").cursor@t(L_ORDERKEY,L_PRICE,L_QUANTITY)	/Read in lineitem.txt
6	=A5.sortx(L_ORDERKEY)	/Sort by order id
7	=file("LINEITEM.ctx").create(#L_ORDERKEY,L_PRICE,L_QUANTITY;L_ORDERKEY)	/Create and open group table, segment by order id field, records with same order id won't be divided to two segments
8	=A7.append(A6)	/Add records from cursor to group table

Merge of main sub-table – Filter Subtable with Main Table

When the main table is filtered out many records by some condition, the sub-table will still be completely traversed by joining with the method of previous page. This method will make the sub-table traverse according to the key jump of the main table, skip the records that have been filtered out by the main table, reduce the traversal of the sub-table, and speed is faster.

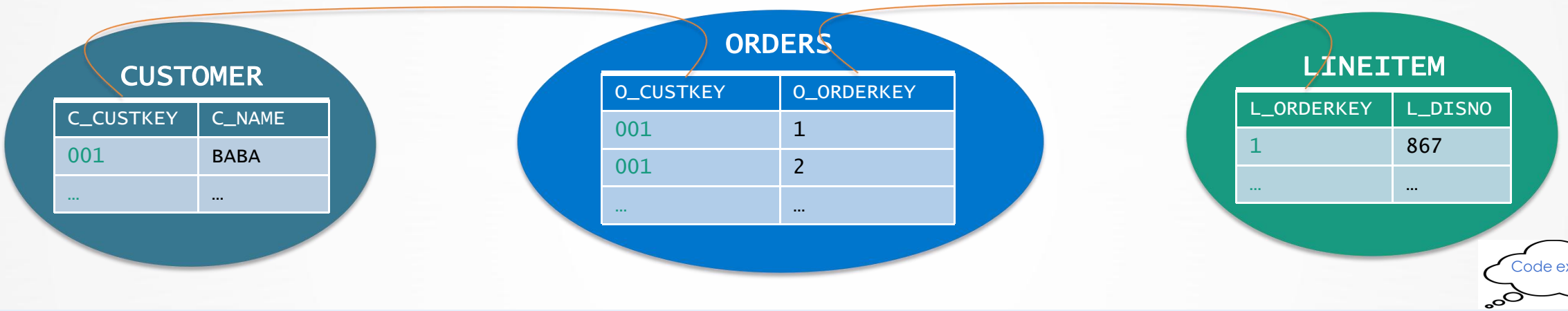


	A	B
1	=file("ORDERS.ctx").create().cursor(O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)	/Create cursor for order table
2	=file("LINEITEM").create().news(A1,L_PRICE,L_QUANTITY,O_CUSTKEY,O_ORDERDATE)	/Create cursor for lineitem table through news, with reference to customer id field of order table
3	=A1.groups(O_CUSTKEY;sum(L_PRICE*L_QUANTITY):AMOUNT)	/Group and aggregate customer's sales amount

Grouping according to the primary key of the main table after join of main table and sub-table

When the main sub-table is joined, the records of the sub-table can be assembled into a subset of the main table (field values are set) to handle the operation of grouping the main table after joining, and the aggregate operation can be written directly in the cursor.

Inter-table relationship diagram:



Code example

	A
1	1995-03-15
2	=file("CUSTOMER.ctx").create().cursor@m(C_CUSTKEY,C_MKTSEGMENT;C_MKTSEGMENT=="BUILDING").fetch().keys@i(C_CUSTKEY)
3	=file("ORDERS.ctx").create().cursor@m(O_ORDERKEY,O_ORDERDATE,O_SHIPPRIORITY;O_ORDERDATE<A1 && A2.find(O_CUSTKEY))
4	=file("LINEITEM.ctx").create().new(A3,O_ORDERKEY,sum(L_EXTENDEDPRICE * (1-L_DISCOUNT)):revenue,O_ORDERDATE,O_SHIPPRIORITY;L_SHIPDATE>A1)
5	=A4.fetch().sort(revenue:-1,O_ORDERDATE)

Grouping according to the primary key of the main table after join of main table and sub-table

The schematic results of the previous page are as follows:

CUSTOMER

Index	C_CUSTKEY	C_MKTSEGMENT
1	1	BUILDING
2	2	AUTOMOBILE
3	3	AUTOMOBILE
4	4	MACHINERY
5	5	HOUSEHOLD
6	6	AUTOMOBILE
7	7	AUTOMOBILE
8	8	BUILDING

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_SHIPPRIORI...
1	1	36901	1996-01-02	0
2	2	78002	1996-12-01	0
3	3	123314	1993-10-14	0
4	4	136777	1995-10-11	0
5	5	44485	1994-07-30	0
6	6	55624	1992-02-21	0
7	7	39136	1996-01-10	0
8	32	130057	1995-07-16	0
9	33	66958	1993-10-27	0
10	34	61001	1998-07-21	0

LINEITEM

Index	L_ORDERKEY	L_EXTENDED...	L_DISCOUNT	L_SHIPDATE
1	1	21168.23	0.04	1996-03-13
2	1	45983.16	0.09	1996-04-12
3	1	13309.6	0.1	1996-01-29
4	1	28955.64	0.09	1996-04-21
5	1	22824.48	0.1	1996-03-30
6	1	49620.16	0.07	1996-01-30
7	2	44694.46	0.0	1997-01-28
8	3	54058.05	0.06	1994-02-02

1 The condition is BUILDING

Index	C_CUSTKEY	C_MKTSEGMENT
1	1	BUILDING
2	8	BUILDING
3	11	BUILDING
4	13	BUILDING
5	18	BUILDING
6	27	BUILDING
7	30	BUILDING
8	32	BUILDING

2 Perform primary key join and conditional filtering in cursor

Index	O_ORDERKEY	O_ORDERDATE	O_SHIPPRIORITY
1	96	1994-04-17	0
2	165	1993-01-30	0
3	256	1993-10-19	0
4	258	1993-12-29	0
5	321	1993-03-21	0
6	356	1994-06-30	0
7	388	1992-12-16	0
8	417	1994-02-06	0

3 Join, conditional filtering, grouping and aggregation

Index	O_ORDERKEY	revenue	O_ORDERD...	O_SHIPPRIO...
1	577	57986.6224	1994-12-19	0
2	1281	82111.5548	1994-12-11	0
3	1637	268170.64...	1995-02-08	0
4	2053	144855.19...	1995-02-07	0
5	2114	29584.063...	1995-01-16	0
6	3430	8263.765	1994-12-12	0
7	3814	153733.96...	1995-02-22	0
8	6791	45774.437...	1995-02-02	0

4 Sort by revenue and order date

Index	O_ORDERKEY	revenue	O_ORDERD...	O_SHIPPRIO...
1	2456423	406181.01...	1995-03-05	0
2	3459808	405838.69...	1995-03-04	0
3	492164	390324.061	1995-02-19	0
4	1188320	384537.93...	1995-03-09	0
5	2435712	378673.05...	1995-02-26	0
6	4878020	378376.79...	1995-03-12	0
7	5521732	375153.92...	1995-03-13	0
8	2628192	373133.30...	1995-02-22	0

Integrated storage of main and sub-table

Integrated storage can further improve the computing performance. Solidify the combination of main and sub-tables in the storage format, and there is no need to join when using, so as to achieve higher performance!

Example for storage:

	A
1	=db.cursor("select * from ORDERS order by O_ID")
2	=db.cursor("select * from LINEITEM order by L_ID")
3	=file("MULTIPLE.ctx").create(#O_ID,C_ID,O_DATE)
4	=A3.append(A1)
5	=A3.attach(LINEITEM,#L_SUBID,PRICE,NUMS)
6	=A5.append(A2)

Code example

ORDERS

O_ID	C_ID	O_DATE
10248	VINET	2018-03-02
10249	TOMSP	2018-03-03
…	…	…
…	…	…

LINEITEM

L_ID	L_SUBID	PRICE	NUMS
10248	1024801	14.00	12
10248	1024802	9.00	10
10249	1024901	18.00	9
…		…	…

Combined storage

Building process

O_ID	L_SUBID	PRICE	NUMS	C_ID	O_DATE
10248				VINET	2018-03-02
	1024801	14.00	12		
	1024802	9.00	10		
10249				TOMSP	2018-03-03
	1024901	18.00	9		
…		…	…	…	…

Query example:

Code example

	A	B
1	=file("MULTIPLE.ctx").create().attach(LINEITEM)	/Open the attached table lineitem
2	=A1.cursor@m(O_ID,C_ID,PRICE,NUMS;;4)	/Create multi-cursors, the number is 4
3	=A2.groups(C_ID:CUSTOMER;sum(PRICE*NUMS):AMOUNT)	/Group and aggregate customer's sales amount

When there are 100 million orders, each record corresponds to about 10 order details, the actual test results of this case are as follows:

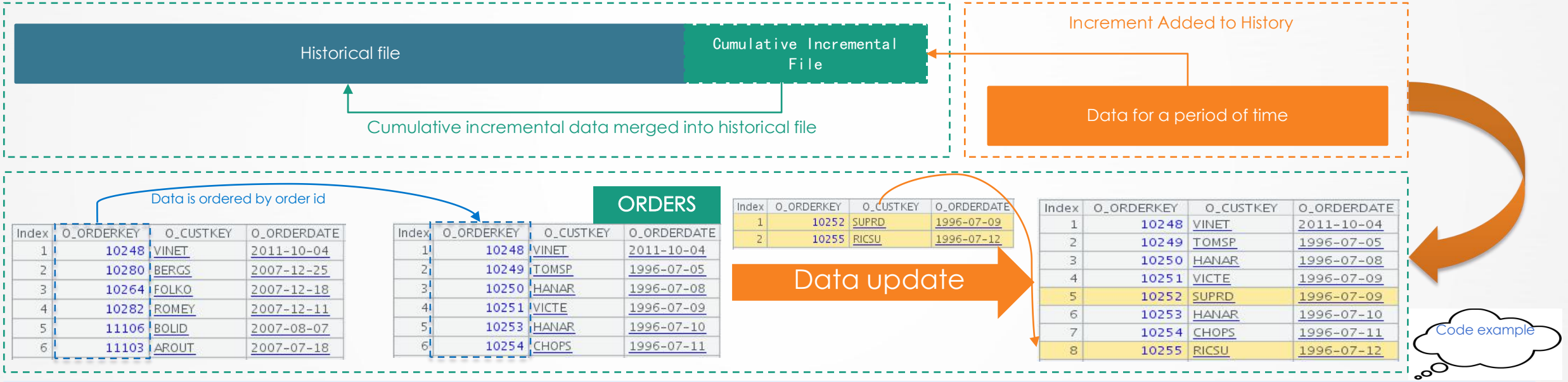
Time consuming (second)

2 joinx of main and sub	Combined storage	Combined storage (4 threads)
781	602	368

Merge of same dimension table and main sub-table – Ordered and Data Updating

The premise of orderly merging is to sort historical data by primary key and store it.

The process of adding data is also orderly merging. It is still a low-cost merging calculation to merge the newly added data after it's sorted separately and the ordered historical data without reordering all the historical data.



Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
1	10248	VINET	2011-10-04
2	10280	BERGS	2007-12-25
3	10264	FOLKO	2007-12-18
4	10282	ROMEY	2007-12-11
5	11106	BOLID	2007-08-07
6	11103	AROUT	2007-07-18

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
1	10248	VINET	2011-10-04
2	10249	TOMSP	1996-07-05
3	10250	HANAR	1996-07-08
4	10251	VICTE	1996-07-09
5	10253	HANAR	1996-07-10
6	10254	CHOPS	1996-07-11

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
1	10252	SUPRD	1996-07-09
2	10255	RICSU	1996-07-12

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE
1	10248	VINET	2011-10-04
2	10249	TOMSP	1996-07-05
3	10250	HANAR	1996-07-08
4	10251	VICTE	1996-07-09
5	10252	SUPRD	1996-07-09
6	10253	HANAR	1996-07-10
7	10254	CHOPS	1996-07-11
8	10255	RICSU	1996-07-12

	A	B
1	=file("ORDERS.ctx").create()	/Open order file
2	=db.query@x("select * from ORDERS where O_ORDERDATE>='1996-07-09' order by O_ORDERKEY")	/Fetch new data from database
3	=A1.append(A2.cursor())	/Append incremental data to original order table

Contents

Converting SQL subqueries into Join – Note

This section explains when sub-queries can be converted to JOIN.
Refer to the previous sections for JOIN optimization methods.

Assume that the tables involved in all examples are stored orderly by primary key

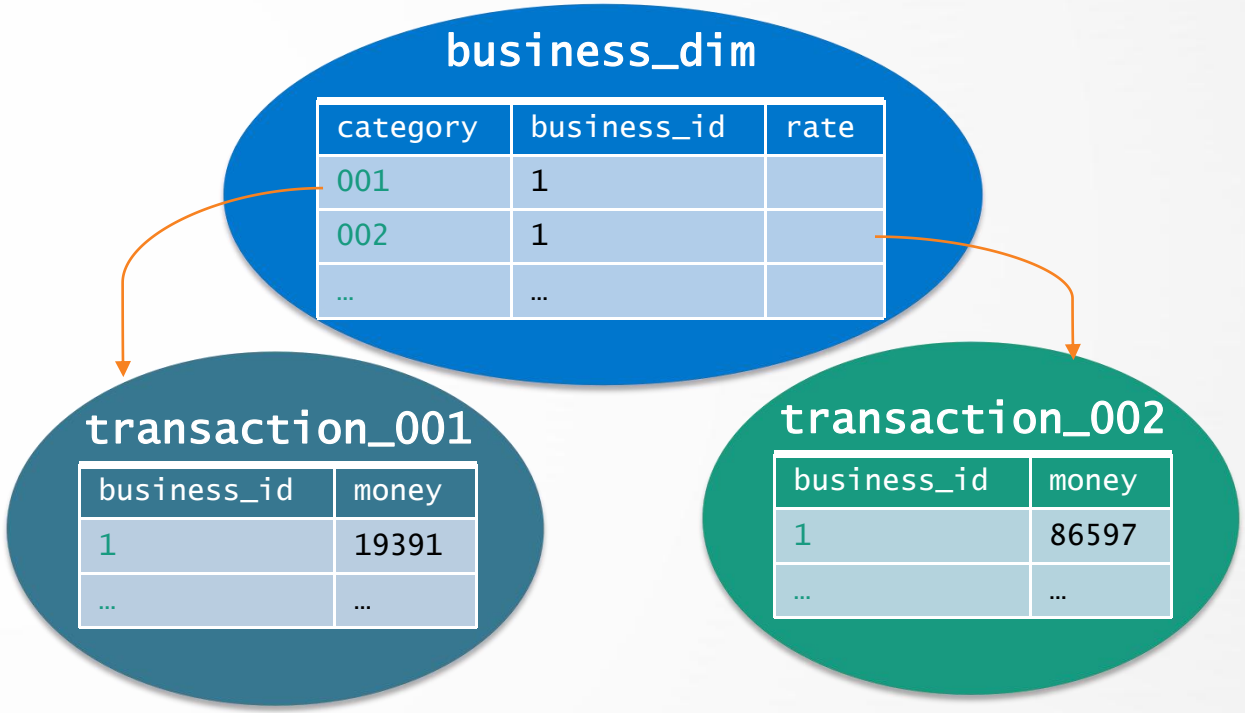
Join key is only for some field of the primary keys of the dimension table

For example, banking fact tables are saved as several tables (insurance main table, financial main table, etc.) according to business. The corresponding dimension tables of different business have the same structure, so a classification field is added to synthesize these dimension tables into a large dimension table.

Dimension table structure（category，business_id as primary keys）：category，business_id，rate，…
The fact table contains only business_id field but no category field. The category value of the fact main table is determined beforehand. For example, the category corresponding to the insurance main table is 001, as shown in the right figure:

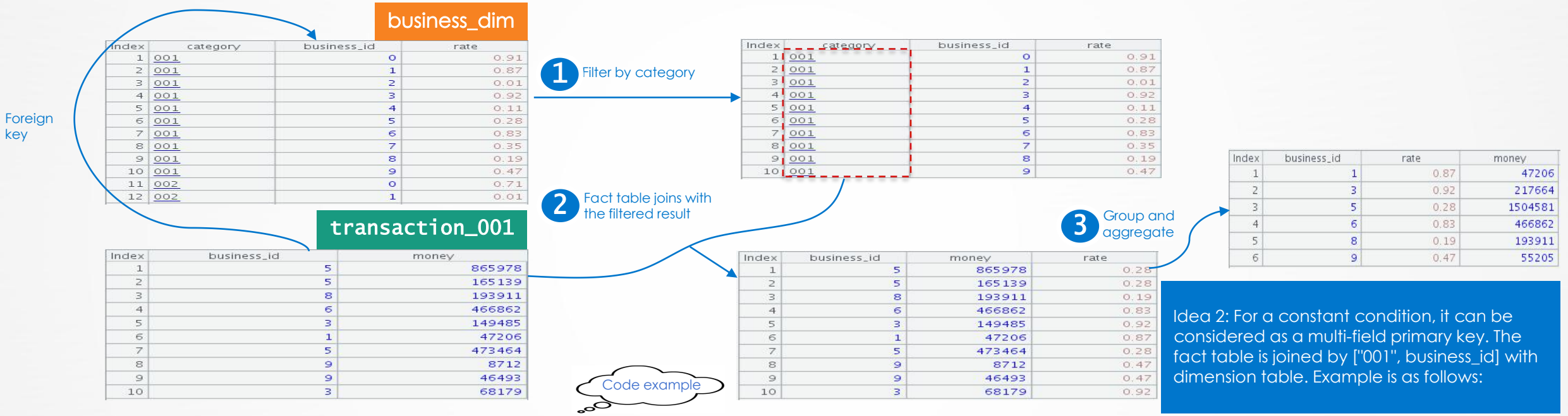
```
select
        d.business_id, d.rate, sum(t.money) as money
from
        transaction_001 as t, business_dim as d
where
        d. category = ‘001’ and t. business_id = d.
business_id
group by
        d.business_id, d.rate
```

SQL



Fact main table contains only part of the fields of dimension table primary key

Optimize Idea 1: Filter the dimension table to get a new dimension table with business_id as the primary key, so that it can be joined with the fact table with business_id field. The dimension table needs to be indexed according to business_id field when it is joined.



Idea 2: For a constant condition, it can be considered as a multi-field primary key. The fact table is joined by ["001", business_id] with dimension table. Example is as follows:

	A
1	=file("business_dim.btx").import@b()
2	=A1.keys@i(category, business_id)
3	=A2.select(category=="001")
4	=file("transaction_001.btx").cursor@b(business_id,money)
5	=A4.join(business_id, A3:business_id, rate)
6	=A5.groups(business_id, rate; sum(money):money)

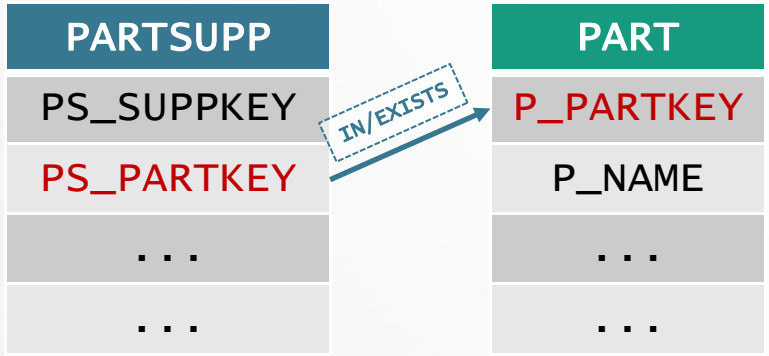
	A
1	=file("business_dim.btx").import@b()
2	=A1.keys@i(category, business_id)
3	=file("transaction_001.btx").cursor@b(business_id,money)
4	=A3.join("001": business_id, A1, rate)
5	=A4.groups(business_id, rate; sum(money):money)

In and Exists associated query by foreign key and dimension table

1 In associated query by foreign key and dimension table

```sql
SELECT
	PS_SUPPKEY, COUNT(1) AS S_COUNT
FROM
	PARTSUPP
WHERE
	PS_PARTKEY IN (
		SELECT
			P_PARTKEY
		FROM
			PART
		WHERE
			P_NAME LIKE 'bisque%%'
	)
GROUP BY PS_SUPPKEY
```

SQL



2 Exists associated query by foreign key and dimension table

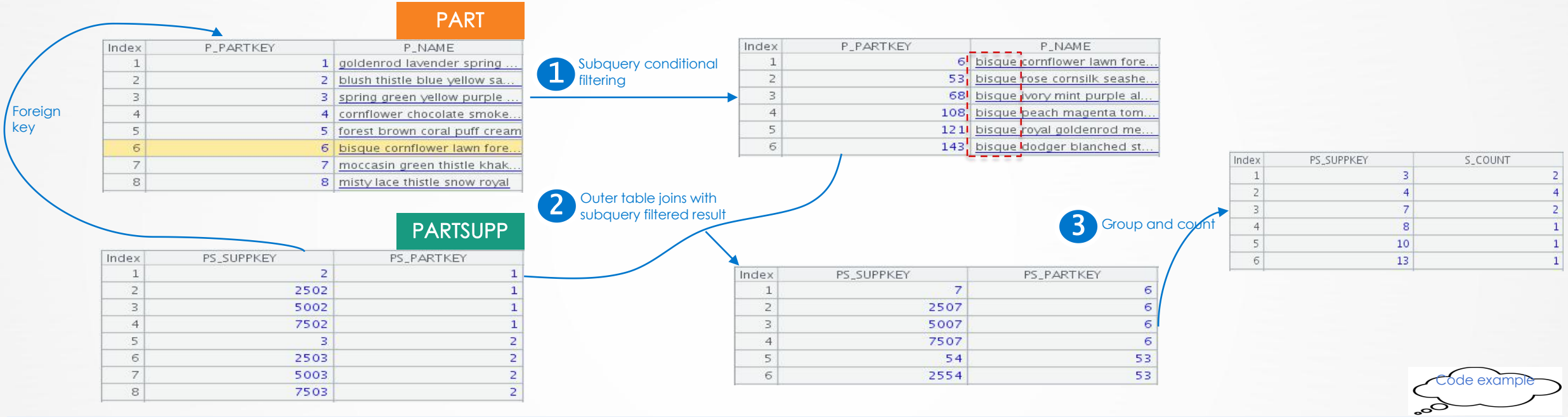
```sql
SELECT
	PS_SUPPKEY, COUNT(1) AS S_COUNT
FROM
	PARTSUPP
WHERE
	EXISTS (
		SELECT *
FROM
PART
		WHERE
			P_PARTKEY = PS_PARTKEY
			AND P_NAME LIKE 'bisque%%'
	)
GROUP BY PS_SUPPKEY
```

SQL

The above examples can be converted into foreign key JOIN!

In and Exists associated query by foreign key and dimension table

Optimizing idea: Subquery is filtered, read into memory and indexed. Outer table is joined with subquery and filtered when cursor is read out. If they are not related, other fields are no longer read out. When more records are filtered out, IO operations can be significantly reduced to improve performance.



	A	B
1	=file("PART.ctx").create().cursor(P_PARTKEY,P_NAME; like(P_NAME, "bisque*"))	/Filter PART when data flows in
2	=A1.fetch().index()	/Create index
3	=file("PARTSUPP.ctx").create().cursor(PS_SUPPKEY, PS_PARTKEY; A2.find(P_PARTKEY))	/When data flows in, read in the PS_PARTKEY field first, and then join with the P_PARTKEY field. If it can be related, continue to read other fields, otherwise discard the current record.
4	=A3.groups(PS_SUPPKEY; count(1):S_COUNT)	/Group and count

In and Exists associated query by main table and sub-table

1 In associated query by main table and sub-table

```sql
SELECT
      O_ORDERPRIORITY, COUNT(*) AS O_COUNT
FROM
      ORDERS
WHERE
      O_ORDERDATE >= DATE '1995-10-01'
      AND O_ORDERDATE < DATE '1995-10-01' +
INTERVAL '3' MONTH
      AND O_ORDERKEY IN (
            SELECT
                  L_ORDERKEY
            FROM
                  LINEITEM
            WHERE
                  L_COMMITDATE< L_RECEIPTDATE
      )

GROUP BY
      O_ORDERPRIORITY
```

SQL

2 EXISTS associated query by main table and sub-table

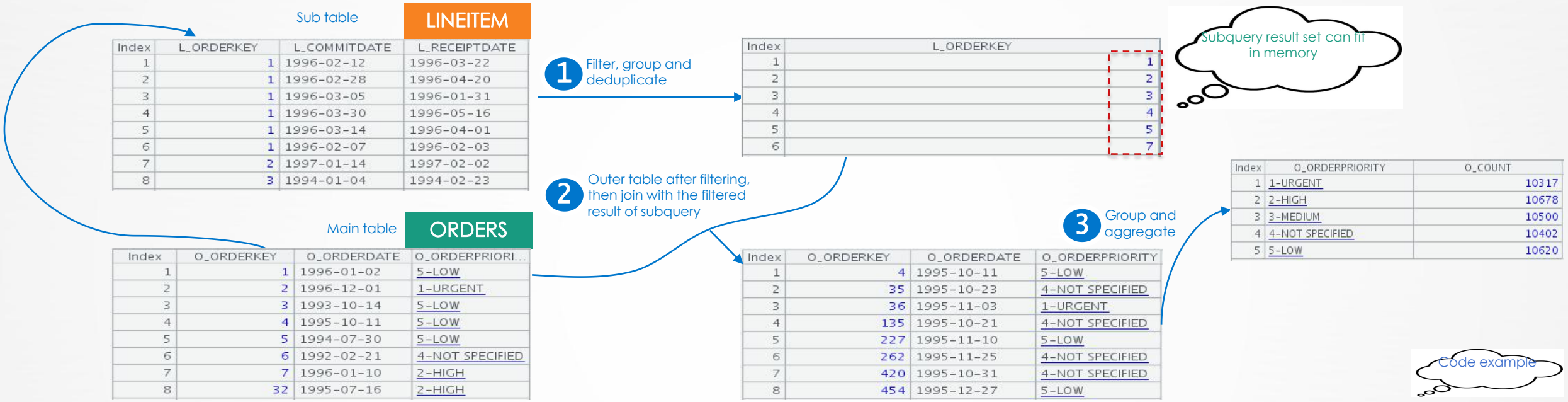
```sql
SELECT
      O_ORDERPRIORITY, COUNT(*) AS O_COUNT
FROM
      ORDERS
WHERE
      O_ORDERDATE >= DATE '1995-10-01'
      AND O_ORDERDATE < DATE '1995-10-01' +
INTERVAL '3' MONTH
      AND EXISTS (
            SELECT
                  *
            FROM
                  LINEITEM
            WHERE
                  L_ORDERKEY = O_ORDERKEY
                  AND L_COMMITDATE < L_RECEIPTDATE
            )
GROUP BY
      O_ORDERPRIORITY
```

SQL

ORDERS、LINEITEM are main and sub-table. The primary key of ORDERS table is O_ORDERKEY，and the primary key of LINEITEM table are L_ORDERKEY, L_LINENUMBER； Selected field is not logical primary key after filtering!

In and Exists associated query by main table and sub-table

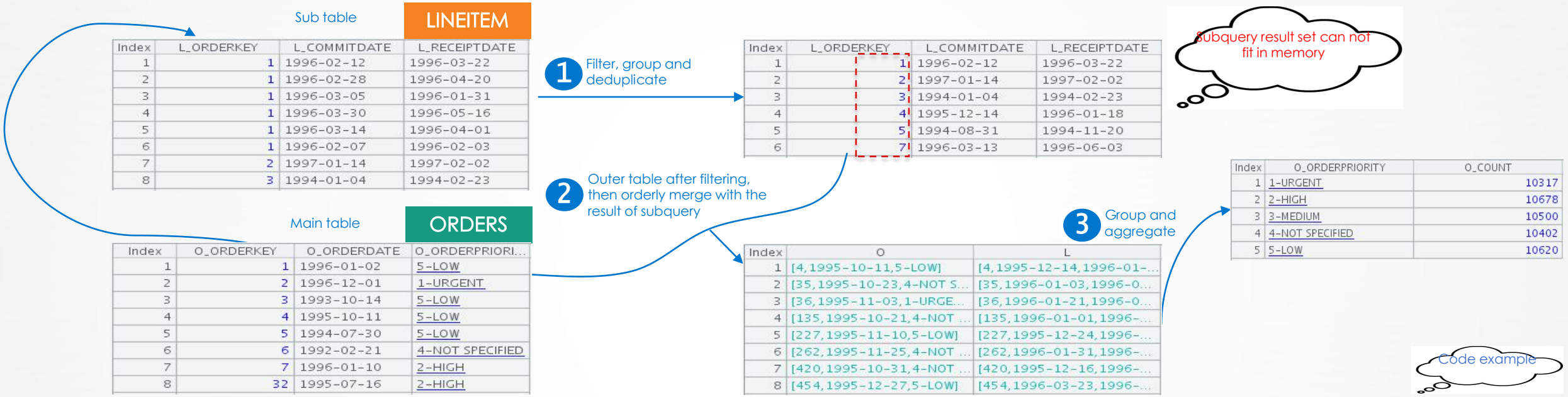
Optimizing idea: The joined field is not a logical primary key. it needs to group and deduplicate for sub-query and then join, which becomes the case similar to logical primary key!



	A	B
1	1995-10-01	=after@m(A1,3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY,L_COMMITDATE,L_RECEIPTDATE)	/Define cursor on btx file corresponding to LINEITEM table
3	=A2.select(L_COMMITDATE < L_RECEIPTDATE)	/Add filtering operation to cursor
4	=A3.groups(L_ORDERKEY)	/Deduplicate L_ORDERKEY
5	=file("ORDERS.btx").cursor@b(O_ORDERKEY,O_ORDERDATE,O_ORDERPRIORITY)	/Define cursor on btx file corresponding to order table
6	=A5.select(O_ORDERDATE>=A1 && O_ORDERDATE < B1)	/Add filtering operation to cursor
7	=A6.join@i(O_ORDERKEY, A4:L_ORDERKEY)	/Join and filter for ORDERS cursor,@i indicates inner join
8	=A7.groups(O_ORDERPRIORITY:count(1):O_COUNT)	/Calculate group for cursor to get final result

In and Exists associated query by main table and sub-table

Optimizing idea: On the basis of the previous page, the outer and inner tables are ordered according to the join field, which can be optimized by merging join of ordered cursor.



Index	L_ORDERKEY	L_COMMITDATE	L_RECEIPTDATE
1	1	1996-02-12	1996-03-22
2	1	1996-02-28	1996-04-20
3	1	1996-03-05	1996-01-31
4	1	1996-03-30	1996-05-16
5	1	1996-03-14	1996-04-01
6	1	1996-02-07	1996-02-03
7	2	1997-01-14	1997-02-02
8	3	1994-01-04	1994-02-23

Index	L_ORDERKEY	L_COMMITDATE	L_RECEIPTDATE
1	1	1996-02-12	1996-03-22
2	2	1997-01-14	1997-02-02
3	3	1994-01-04	1994-02-23
4	4	1995-12-14	1996-01-18
5	5	1994-08-31	1994-11-20
6	7	1996-03-13	1996-06-03

Index	O_ORDERKEY	O_ORDERDATE	O_ORDERPRIORI...
1	1	1996-01-02	5-LOW
2	2	1996-12-01	1-URGENT
3	3	1993-10-14	5-LOW
4	4	1995-10-11	5-LOW
5	5	1994-07-30	5-LOW
6	6	1992-02-21	4-NOT SPECIFIED
7	7	1996-01-10	2-HIGH
8	32	1995-07-16	2-HIGH

Index	O	L
1	[4,1995-10-11,5-LOW]	[4,1995-12-14,1996-01-...
2	[35,1995-10-23,4-NOT S...	[35,1996-01-03,1996-0...
3	[36,1995-11-03,1-URGE...	[36,1996-01-21,1996-0...
4	[135,1995-10-21,4-NOT ...	[135,1996-01-01,1996-...
5	[227,1995-11-10,5-LOW]	[227,1995-12-24,1996-...
6	[262,1995-11-25,4-NOT ...	[262,1996-01-31,1996-...
7	[420,1995-10-31,4-NOT ...	[420,1995-12-16,1996-...
8	[454,1995-12-27,5-LOW]	[454,1996-03-23,1996-...

Index	O_ORDERPRIORITY	O_COUNT
1	1-URGENT	10317
2	2-HIGH	10678
3	3-MEDIUM	10500
4	4-NOT SPECIFIED	10402
5	5-LOW	10620

	A	B
1	1995-10-01	=after@m(A1,3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY,L_COMMITDATE,L_RECEIPTDATE)	/Define cursor on btx file corresponding to LINEITEM table
3	=A2.select(L_COMMITDATE < L_RECEIPTDATE)	/Add filtering operation to cursor
4	=A3.group@1(L_ORDERKEY)	/Deduplicate L_ORDERKEY
5	=file("ORDERS.btx").cursor@b(O_ORDERKEY,O_ORDERDATE,O_ORDERPRIORITY)	/Define cursor on btx file corresponding to order table
6	=A5.select(O_ORDERDATE>=A1 && O_ORDERDATE < B1)	/Add filtering operation to cursor
7	=joinx(A6:O,O_ORDERKEY; A4:L,L_ORDERKEY)	/Ordered inner join
8	=A7.groups(O.O_ORDERPRIORITY;count(1):O_COUNT)	/Calculate group for cursor to get final result

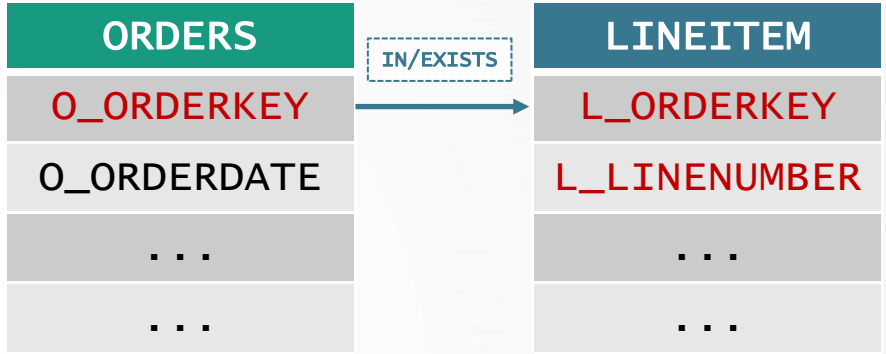
Main and sub-table, Joined field filtered into logical primary key

In associated query by main table and sub-table

```SQL
SELECT
	O_ORDERPRIORITY, COUNT(*) AS O_COUNT
FROM
	ORDERS
WHERE
	O_ORDERDATE >= DATE '1995-10-01'
	AND O_ORDERDATE < DATE '1995-10-01' +
INTERVAL '3' MONTH
	AND O_ORDERKEY IN (
		SELECT
			L_ORDERKEY
		FROM
			LINEITEM
		WHERE L_LINENUMBER = 1
			L_COMMITDATE< L_RECEIPTDATE
	)

GROUP BY
	O_ORDERPRIORITY
```

SQL

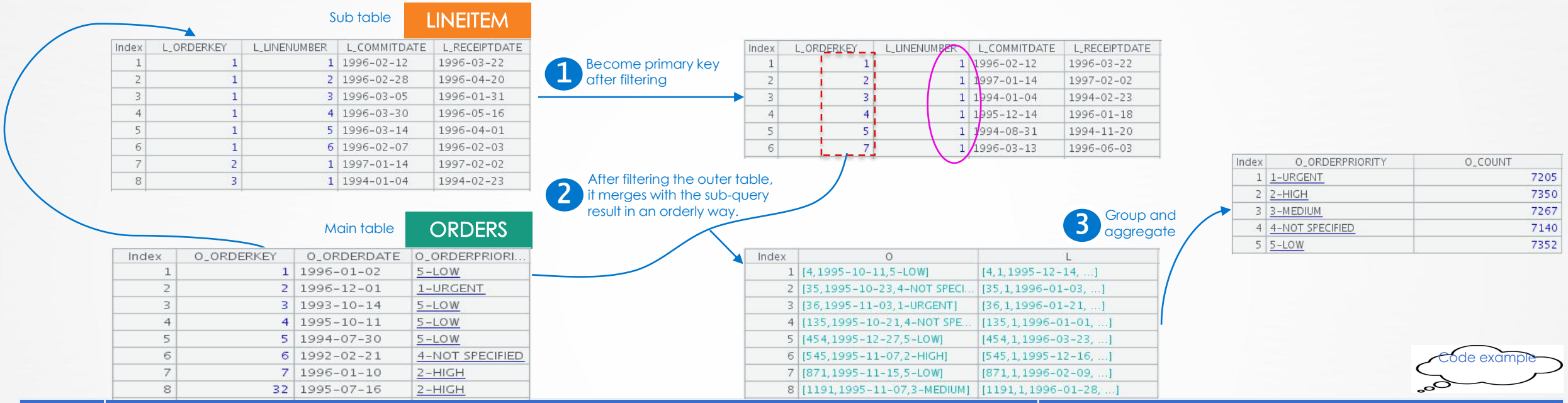


ORDERS and LINEITEM are the main sub-table, the primary key of ORDERS is O_ORDERKEY, and the primary keys of LINEITEM are L_ORDERKEY and L_LINENUMBER, and the field selected by sub-query is not logical primary key.

After limiting L_LINENUMBER = 1, the selected L_ORDERKEY becomes the logical primary key!

Main and sub-table, Joined field filtered into logical primary key

Optimizing idea: The outer table and inner table (filtered into primary key) are ordered by the joined field, and can be optimized by merging join of ordered cursor!

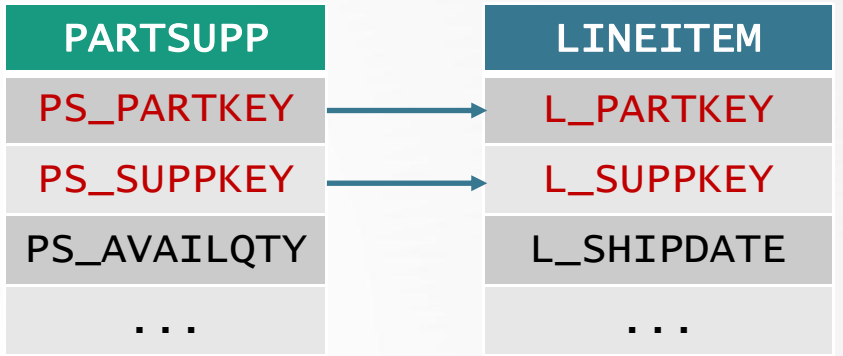


	A	B
1	1995-10-01	=after@m(A1,3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY,L_LINENUMBER,L_COMMITDATE,L_RECEIPTDATE)	/Define cursor on btx file corresponding to LINEITEM table
3	=A2.select(L_LINENUMBER == 1 && L_COMMITDATE < L_RECEIPTDATE)	/Add filtering operation to cursor
4	=file("ORDERS.btx").cursor@b(O_ORDERKEY,O_ORDERDATE,O_ORDERPRIORITY)	/Define cursor on btx file corresponding to order table
5	=A4.select(O_ORDERDATE>=A1 && O_ORDERDATE < B1)	/Add filtering operation to cursor
6	=joinx(A5:O,O_ORDERKEY; A3:L,L_ORDERKEY)	/ordered inner Join
7	=A6.groups(O.O_ORDERPRIORITY;count(1):O_COUNT)	/Calculate group for cursor to get final result

Converting SQL subqueries into Join – WHERE subquery

WHERE subquery example:

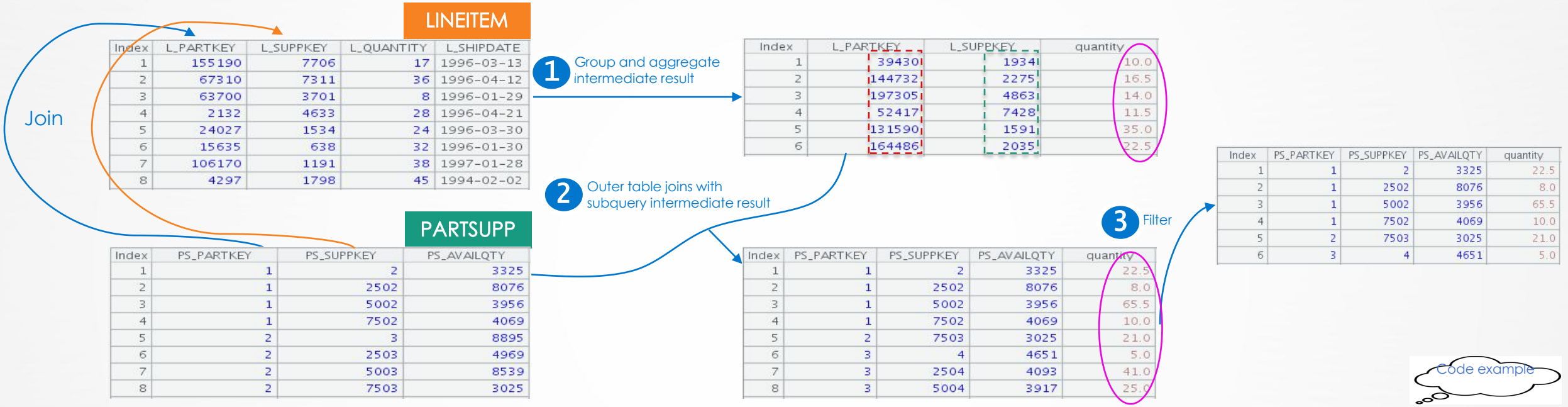
```SQL
SELECT
	PS_SUPPKEY
FROM
	PARTSUPP
WHERE
	PS_AVAILQTY > (
	SELECT
		0.5 * SUM(L_QUANTITY)
	FROM
		LINEITEM
	WHERE
		L_PARTKEY = PS_PARTKEY
		AND L_SUPPKEY = PS_SUPPKEY
		AND L_SHIPDATE >= DATE '1995-04-01'
		AND L_SHIPDATE < DATE '1995-04-01' + INTERVAL '1'
YEAR
	)
```



The LINEITEM in the sub-query is joined with the outer main table PARTSUPP according to PARTKEY and SUPPKEY. We can filter LINEITEM first, then aggregate a temporary table according to PARTKEY and SUPPKEY, and then join the outer PARTSUPP with the aggregated table.

Converting SQL subqueries into Join – WHERE subquery

Optimizing idea: Subquery is grouped according to the joined field involved, and a temporary dimension table is calculated, then JOIN with the outer table!



Index	L_PARTKEY	L_SUPPKEY	L_QUANTITY	L_SHIPDATE
1	155190	7706	17	1996-03-13
2	67310	7311	36	1996-04-12
3	63700	3701	8	1996-01-29
4	2132	4633	28	1996-04-21
5	24027	1534	24	1996-03-30
6	15635	638	32	1996-01-30
7	106170	1191	38	1997-01-28
8	4297	1798	45	1994-02-02

Index	L_PARTKEY	L_SUPPKEY	quantity
1	39430	1934	10.0
2	144732	2275	16.5
3	197305	4863	14.0
4	52417	7428	11.5
5	131590	1591	35.0
6	164486	2035	22.5

Index	PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY
1	1	2	3325
2	1	2502	8076
3	1	5002	3956
4	1	7502	4069
5	2	3	8895
6	2	2503	4969
7	2	5003	8539
8	2	7503	3025

Index	PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	quantity
1	1	2	3325	22.5
2	1	2502	8076	8.0
3	1	5002	3956	65.5
4	1	7502	4069	10.0
5	2	7503	3025	21.0
6	3	4	4651	5.0
7	3	2504	4093	41.0
8	3	5004	3917	25.0

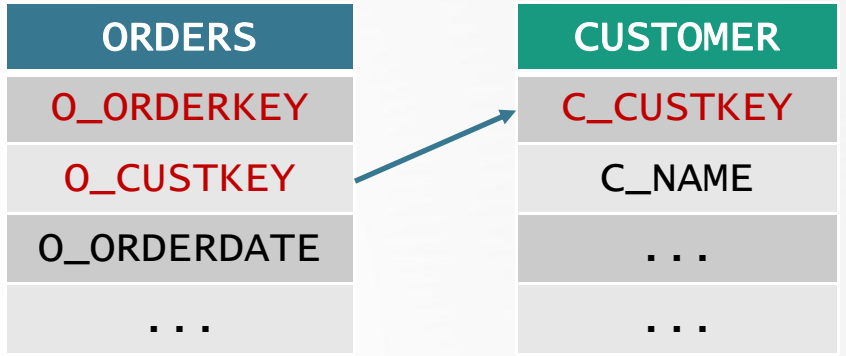
Index	PS_PARTKEY	PS_SUPPKEY	PS_AVAILQTY	quantity
1	1	2	3325	22.5
2	1	2502	8076	8.0
3	1	5002	3956	65.5
4	1	7502	4069	10.0
5	2	7503	3025	21.0
6	3	4	4651	5.0

	A	B
1	=after@y(date,1)	/Get the next year of parameter date
2	=file("LINEITEM.btx").cursor@b(L_PARTKEY,L_SUPPKEY,L_QUANTITY,L_SHIPDATE)	/Define cursor on btx file corresponding to LINEITEM table
3	=A2.select(L_SHIPDATE >= date && L_SHIPDATE < A1)	/Add filtering operation to cursor
4	=A3.groups@u(L_PARTKEY,L_SUPPKEY;sum(L_QUANTITY) * 0.5:quantity)	/Group and aggregate, @u indicates the result set is not ordered by the grouping filed
5	=file("PARTSUPP.btx").cursor@b(PS_PARTKEY,PS_SUPPKEY,PS_AVAILQTY)	/Add filtering operation to cursor
6	=A5.join@i(PS_PARTKEY:PS_SUPPKEY,A4:L_PARTKEY:L_SUPPKEY,quantity)	/Join and filter by PARTSUPP,@i indicates inner join
7	=A6.select(PS_AVAILQTY>quantity).fetch()	/Filter the cursor to get the result

Set Operations – Difference Set Operation

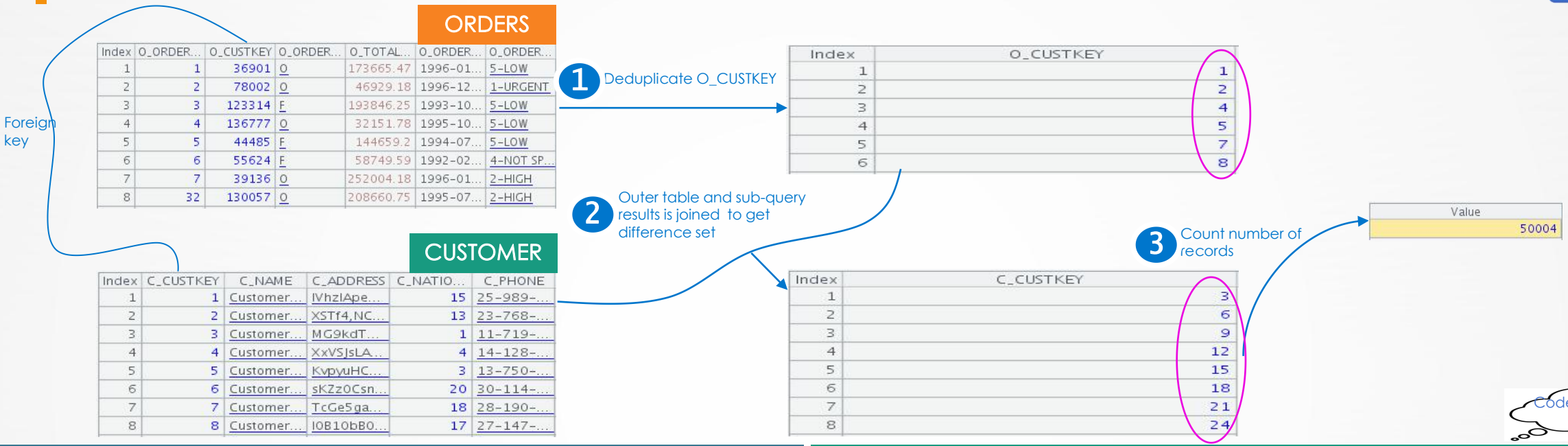
Find records that exist in one table but not in another:

```SQL
SELECT
	COUNT(1)
FROM
	CUSTOMER
WHERE
	NOT EXISTS (
		SELECT *
		FROM
			ORDERS
		WHERE
			O_CUSTKEY = C_CUSTKEY
	)
```



This problem can be transformed into the difference set operation of two sets!

Set Operations – Difference Set Operation



Index	O_ORDER...	O_CUSTKEY	O_ORDER...	O_TOTAL...	O_ORDER...	O_ORDER...
1	1	36901	O	173665.47	1996-01...	5-LOW
2	2	78002	O	46929.18	1996-12...	1-URGENT
3	3	123314	F	193846.25	1993-10...	5-LOW
4	4	136777	O	32151.78	1995-10...	5-LOW
5	5	44485	F	144659.2	1994-07...	5-LOW
6	6	55624	F	58749.59	1992-02...	4-NOT SP...
7	7	39136	O	252004.18	1996-01...	2-HIGH
8	32	130057	O	208660.75	1995-07...	2-HIGH

Index	O_CUSTKEY
1	1
2	2
3	4
4	5
5	7
6	8

Index	C_CUSTKEY	C_NAME	C_ADDRESS	C_NATIO...	C_PHONE
1	1	Customer...	IVhzIApe...	15	25-989-...
2	2	Customer...	XSTf4,NC...	13	23-768-...
3	3	Customer...	MG9kdT...	1	11-719-...
4	4	Customer...	XxVSJsLA...	4	14-128-...
5	5	Customer...	KvpyuHC...	3	13-750-...
6	6	Customer...	sKZz0Csn...	20	30-114-...
7	7	Customer...	TcGe5ga...	18	28-190-...
8	8	Customer...	l0B10bB0...	17	27-147-...

Index	C_CUSTKEY
1	3
2	6
3	9
4	12
5	15
6	18
7	21
8	24

Code example

1 Subquery result can fit in memory

	A	B
1	=file("ORDERS.btx").cursor@b(O_CUSTKEY)	/Define cursor
2	=A1.groups(O_CUSTKEY)	/Deduplicate O_CUSTKEY
3	=file("CUSTOMER.btx").cursor@b(C_CUSTKEY)	/Define cursor
4	=A3.join@d(C_CUSTKEY,A2:O_CUSTKEY)	/Join, @d indicates to get the difference set
5	=A4.total(count(1))	/Aggregate number of records for cursor

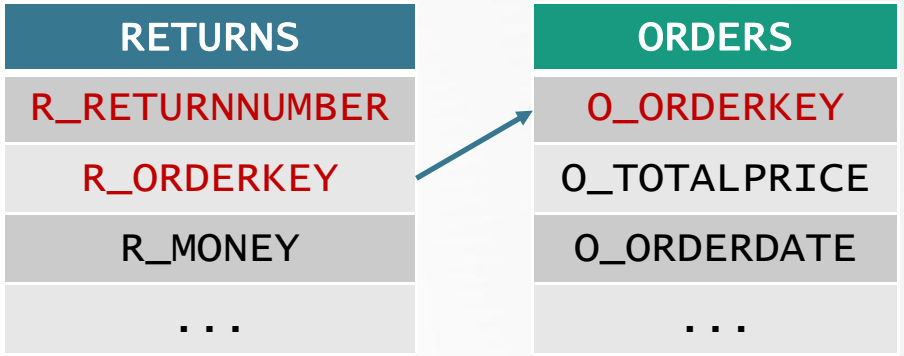
2 Subquery has a large amount of data and can not fit in memory, so it can be merged.

	A	B
1	=file("ORDERS.btx").cursor@b(O_CUSTKEY)	/Define cursor
2	=A1.groupx(O_CUSTKEY:C_CUSTKEY)	/Group and deduplicate
3	=file("CUSTOMER.btx").cursor@b(C_CUSTKEY)	/Define cursor
4	=[A3,A2].mergex@d(C_CUSTKEY)	/Merge, @d indicates to get the difference set
5	=A4.total(count(1))	/Aggregate number of records for cursor

Set Operations – Intersect Set Operation

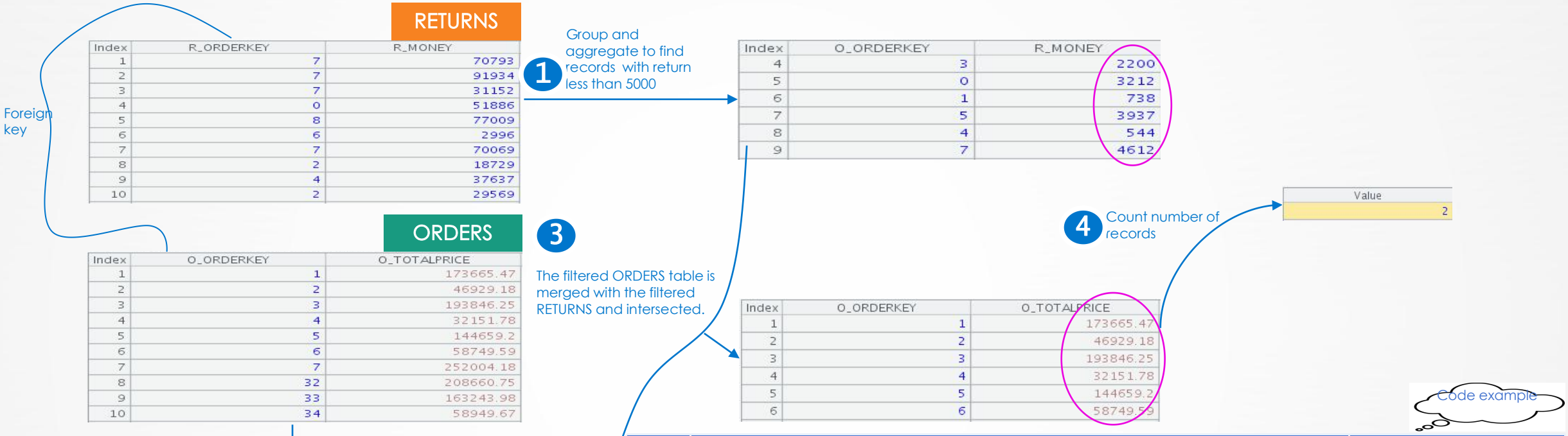
Find orders with an amount of more than 1000 but a return of less than 5000

```SQL
SELECT
	O_ORDERKEY
FROM
	ORDERS
WHERE
	O_TOTALPRICE > 10000
INTERSECT
SELECT
	R_ORDERKEY
FROM
	RETURNS
GROUP BY
	R_ORDERKEY HAVING SUM(R_MONEY) < 5000
```



This problem can be transformed into the intersection operation of two sets!

Set Operations – Intersect Set Operation



	A	B
1	=file("RETURNS.btx").cursor@b(R_ORDERKEY,R_MONEY)	/Define cursor
2	=A1.group(R_ORDERKEY:O_ORDERKEY;sum(R_MONEY):R_MONEY).select(R_MONEY < 5000)	/Group and deduplicate
3	=file("ORDERS.btx").cursor@b(O_ORDERKEY,O_TOTALPRICE).select(O_TOTALPRICE>10000)	/Define cursor
4	=[A3,A2].mergex@i(O_ORDERKEY)	/Merge, @i means intersect
5	=A4.total(count(1))	/Aggregate number of records

Same table join, EXISTS Non-Equivalent Condition

```sql
SELECT
	L_SUPPKEY, COUNT(*) AS NUMWAIT
FROM
	LINEITEM L1,
WHERE
	L1.L_RECEIPTDATE > L1.L_COMMITDATE
	AND EXISTS (
		SELECT
			*
		FROM
			LINEITEM L2
		WHERE
			L2.L_ORDERKEY = L1.L_ORDERKEY
			AND L2.L_SUPPKEY <> L1.L_SUPPKEY
		)
	AND NOT EXISTS (
		SELECT
			*
		FROM
			LINEITEM L3
		WHERE
			L3.L_ORDERKEY = L1.L_ORDERKEY
			AND L3.L_SUPPKEY <> L1.L_SUPPKEY
			AND L3.L_RECEIPTDATE > L3.L_COMMITDATE
		)
GROUP BY
	L_SUPPKEY
```

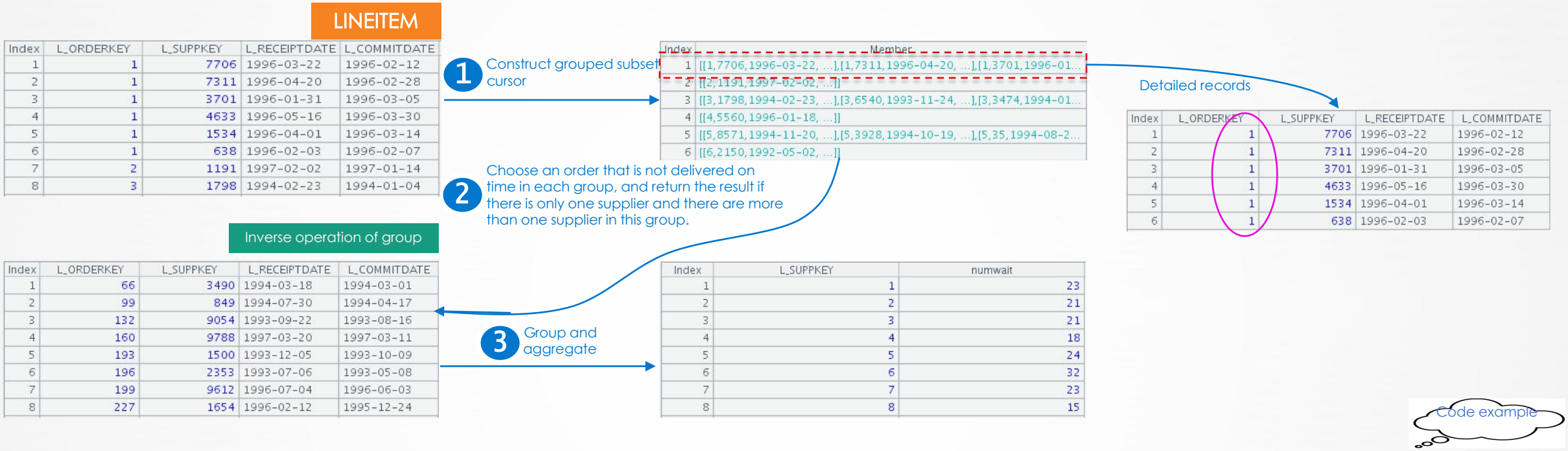
SQL

LINEITEM
L_ORDERKEY
L_LINENUMBER
L_SUPPKEY
L_COMMITDATE
...

An order corresponds to multiple LINEITEM records, which have the same L_ORDERKEY and are stored continuously.

Optimizing idea: Find out orders with multiple suppliers and only one supplier not delivering on time. Because the data is stored in order of orders, orderly grouping can be made according to orders, and each group of order can looped to determine whether there are order items that are not delivered on time, whether there are multiple suppliers, and whether there are only one supplier that did not deliver on time!

Same table join, EXISTS Non-Equivalent Condition

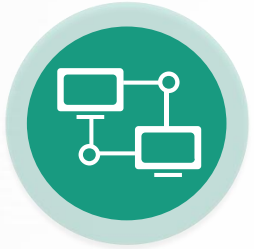


	A	B
1	=file("LINEITEM.btx").cursor@b(L_ORDERKEY,L_SUPPKEY,L_RECEIPTDATE,L_COMMITDATE)	/Define cursor on btx file corresponding to LINEITEM table
2	=A1.group(L_ORDERKEY)	/Add grouping to ordered cursor
3	=A2.conj((t=~.select(L_RECEIPTDATE>L_COMMITDATE),if(t.len()>0&&t.select@1(t(1).L_SUPPKEY!=L_SUPPKEY)==null&&~.select@1(t(1).L_SUPPKEY!=L_SUPPKEY)!= null,t,null)))	/Choose the order that is not delivered on time in each group to the temporary variable t. If the length of t is greater than 0 and there is only one supplier in t and there are more than one supplier in this group, then return t. Otherwise return null, conj is equivalent to the inverse operation of group.
4	=A3.groups@u(L_SUPPKEY;count(1):numwait)	/Calculate group for cursor to get final result

Converting SQL subqueries into Join – Summary

INs described by sub-queries can be changed to EXISTS. Equivalent EXISTS is essentially a join. For SQL like select * from A where exists (select * from B where …) , the following characteristics should be clarified:

Is the join field the primary key or the logical primary key of each table?



The size of tables A and B, can they be loaded into memory after performing other filtering conditions?

If both tables cannot be loaded into memory, decide whether the two tables are ordered by the joined field.

Optimizing idea :

If there is a table that fit into memory, the in-memory join method can be used. The relative SPL functions are cs.switch()、cs.join(), option @i, @d corresponds to exists and not exists respectively.

Subquery requires that the values of the joined field be unique. If they are not logical primary keys, they must be de-duplicated first. A. groups () can be used to de-duplicate them.

If both tables are too large to fit in memory, it is necessary to check whether the two tables are ordered by the joined field. If not ordered, cs. sortx () can be used to sort; the ordered two tables can be joined by joinx ().

Innovation makes progress!