



集算器

创新大数据计算引擎

性能优化-连接运算

润乾软件出品



目录

Contents

1

JOIN运算理解

2

外键表

3

主子与同维表

4

子查询转换

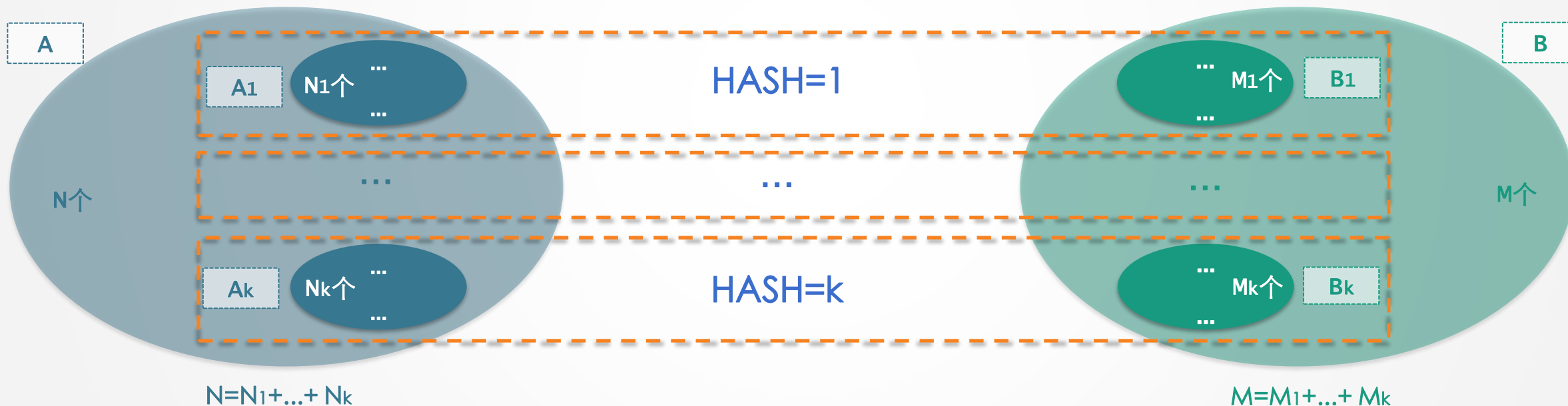


连接运算及传统计算方法

SQL对JOIN的定义非常简单，就是两个集合（表）做笛卡尔积后再按某种条件过滤！

关系数据库一般采用HASH方法实现连接，即分别计算关联字段的HASH值，将HASH同值记录拼到一起再做小范围遍历比较！

全内存HASH JOIN原理：



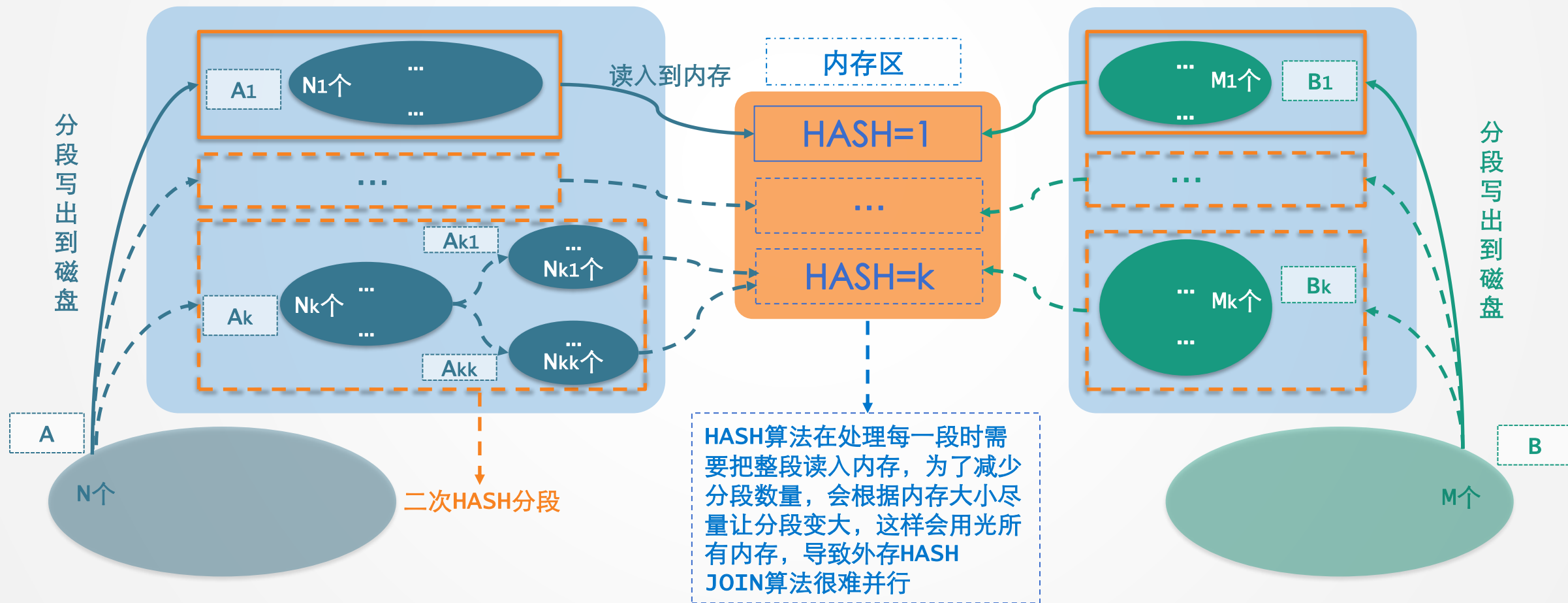
常规比较次数： $N * M = (N_1 + \dots + N_k) * (M_1 + \dots + M_k)$ ；HASH后比较次数： $N_1 * M_1 + N_2 * M_2 + \dots + N_k * M_k$
显然，前者次数一般远大于后者！（k是HASH取值范围）



连接运算及传统计算方法

当要 JOIN 的两个表大到内存无法放下时，关系数据库仍采用 HASH 分段技术。根据关联字段的 HASH 值，将数据分成若干堆，每堆都足够小到能装入内存再用内存 HASH 算法。

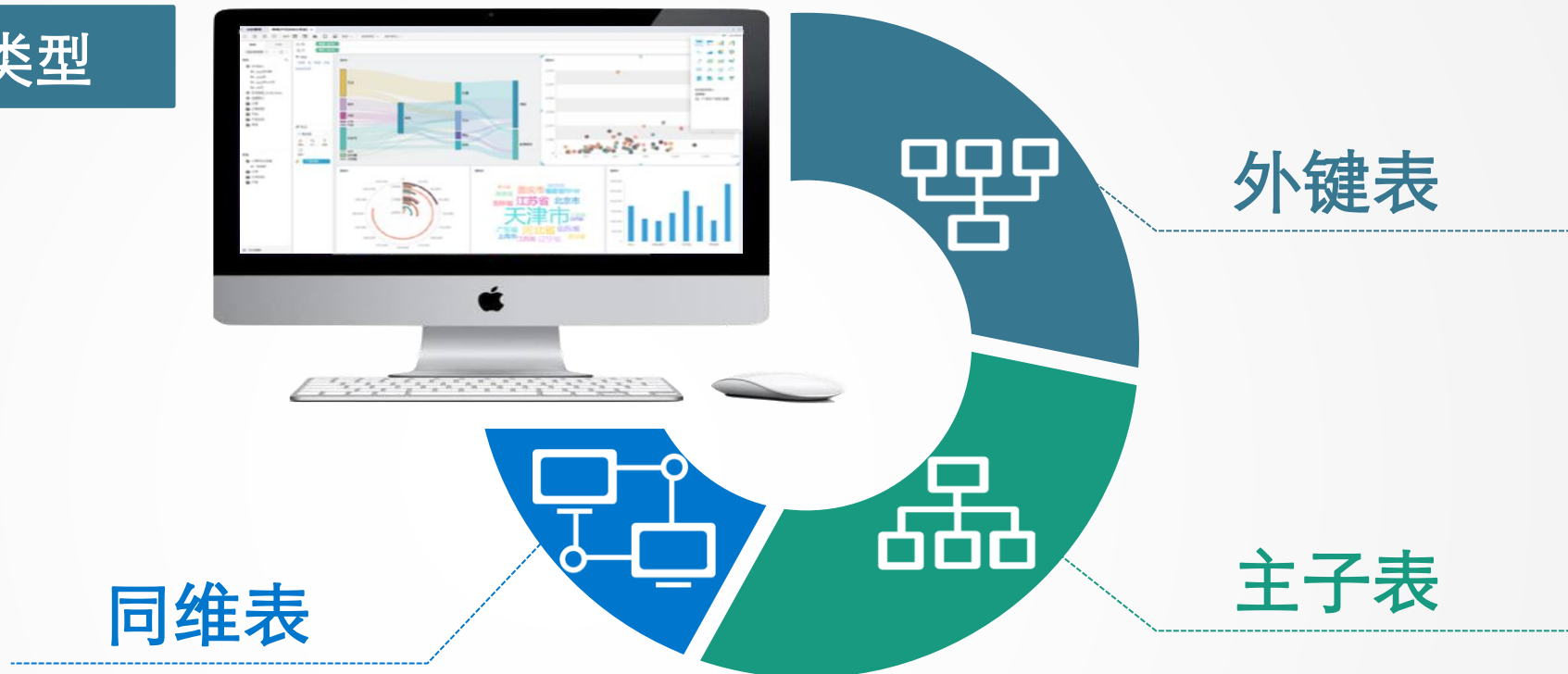
外存HASH JOIN原理：





连接运算剖析

等值JOIN的常见类型



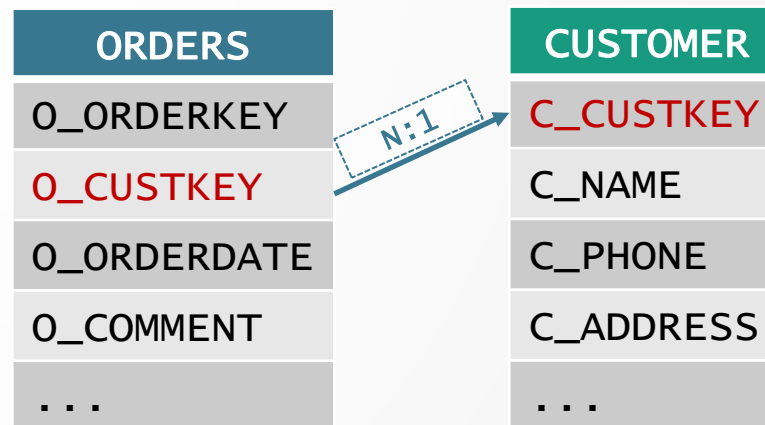
现实中绝大多数JOIN都是等值JOIN，上面三种JOIN已经涵盖了绝大多数等值JOIN的情况

充分利用了这些特征，可获得更简单的书写形式和更高效的运算性能



连接运算剖析 – 外键表

- ① 表 A 的某些字段与表 B 的主键关联，A 表称为事实表，B 表称为维表
- ② 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
- ③ 外键表是多对一的关系，主要是JOIN和LEFT JOIN,一般不会用到 FULL JOIN
- ④ 典型例子：订单表和客户信息表



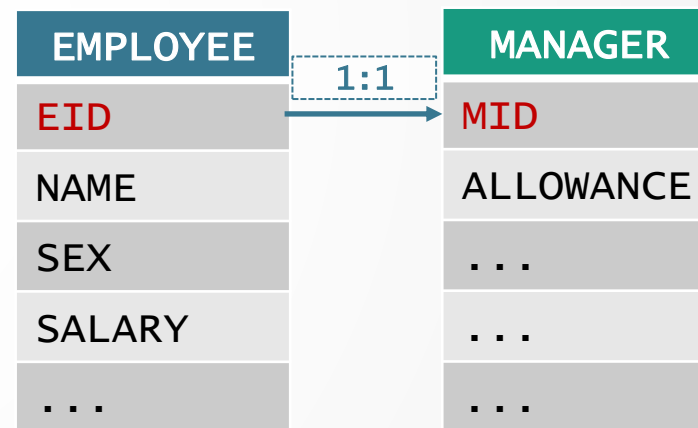


连接运算剖析 – 同维表

① 表 A 的主键与表 B 的主键关联，A 和 B 互称为同维表

② 同维表是一对一的关系，JOIN、LEFT JOIN 和 FULL JOIN 的情况都会有

③ 典型例子：员工表和经理表



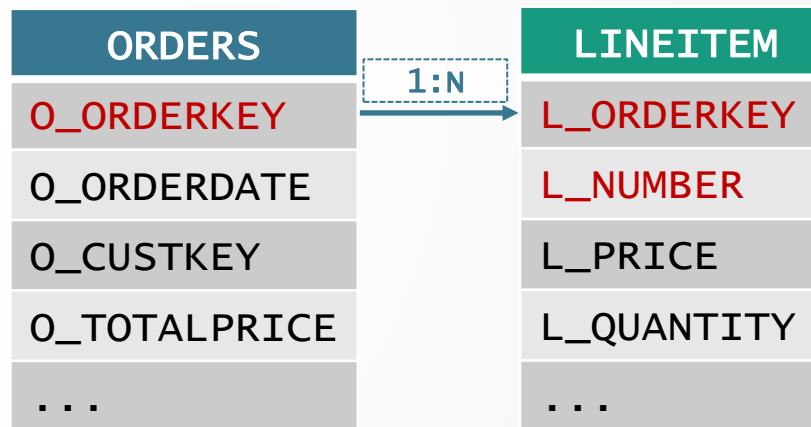


连接运算剖析 – 主子表

① 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

② 主子表是一对多的关系，只有 JOIN 和 LEFT JOIN，不会有 FULL JOIN

③ 典型例子：订单表和订单明细表





目录

Contents

1

JOIN运算理解

2

外键表

3

主子与同维表

4

子查询转换



全内存外键预关联 - 外键属性化

数据都能够装入内存时可实现外键属性化。将订单表中的外键客户ID转换成指向客户表记录的属性，即客户ID的取值就已经是某个客户表中的记录，就可以直接引用记录的字段进行计算。

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

外键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

订单表与客户表建立连接前后的变化



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

示例代码

	A	B
1	=ORDERS.switch(O_CUSTKEY, CUSTOMER:C_CUSTKEY)	/建立外键预关联，把客户ID转换成属性
2	=A1.new(O_CUSTKEY.C_NAME:C_NAME, O_CUSTKEY, O_ORDERDATE, O_TOTALPRICE)	/查询客户的订单详情
3	=A1.groups(O_CUSTKEY.C_CITY:C_CITY; sum(O_TOTALPRICE):O_TOTALPRICE)	/按照客户区域名称汇总订单的销售额

预关联建立后可复用，即第1步只要做一次，以后再做这两个字段的关联时就不必再计算 HASH 值和比对了，可大幅提高性能。而SQL的 JOIN 运算不假定外键指向记录的唯一性，不能使用外键属性化方法，每次关联时都要计算 HASH 值并比对！



全内存外键预关联 - 多外键一次解析

HASH JOIN算法每次只能解析掉一个关联，有N个JOIN要执行N遍动作，每次关联后需要保持中间结果供下一轮使用，计算过程复杂得多，数据也会被遍历多次。

外键属性化的办法解析JOIN，对事实表遍历一次即可完成所有外键的预关联！

外键N:1

ORDERS					
Index	O_ORDERKEY	O_CUSTKEY	O_EMPID	O_ORDERDATE	O_TOTALPRICE
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

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

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	5	Zhaojun	manager
6	6	SunLin	sales

外键N:1

ORDERS					
Index	O_ORDERKEY	O_CUSTKEY	O_EMPID	O_ORDERDATE	O_TOTALPRICE
1	10262	[RATTC, Lear...	[8, LiuYinjiu, co...	1996-07-22	14487.0
2	10263	[ERNSH, Reso...	[9, ZhangXue...	1996-07-23	43818.0
3	10264	[FOLKO, Wuzh...	[6, SunLin, sales]	2007-12-18	1101.0
4	10265	[BLONP, The ...]	[2, WangWei, Vi...	1996-07-25	5528.0
5	10266	[WARTH, Upg...	[3, LiFang, sales]	1996-07-26	7719.0
6	10267	[FRANK, Trust...	[4, Zhenjianjie...	1996-07-29	20858.0
7	10268	[GROSR, Light...	[8, LiuYinjiu, co...	1996-07-30	19887.0
8	10269	[WHITC, Chair...	[5, Zhaojun, m...	1996-07-31	456.0

C_CUSTKEY	C_NAME	C_REGION
ERNSH	Resources are people	South China

C_CUSTKEY	C_NAME	C_REGION
BLONP	The hao	North East

E_ID	E_NAME	E_POSITION
2	WangWei	Vice President

E_ID	E_NAME	E_POSITION
8	LiuYinjiu	coordinator

订单表与客户表、雇员表建立连接前后的变化



示例代码

	A	B
1	=ORDERS.switch(O_CUSTKEY, CUSTOMER:C_CUSTKEY;O_EMPID, EMPLOYEE:E_ID)	/建立关联，把客户ID、雇员ID转换成属性
2	=A1.groups(O_CUSTKEY.C_REGION:C_REGION,O_EMPID.E_NAME:E_NAME;sum(O_TOTALPRICE):AMOUNT)	/按客户所在区域和销售人员汇总销售额



全内存外键预关联 - 复制外键属性

外键属性化简单高效，但解决不了LEFT JOIN的情况，当与维表记录匹配不上时，会导致事实表外键丢失。

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

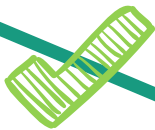
ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
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

外键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. L...	North China
6	ANATR	The southeast industrial	North China



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

匹配不上的记录为 null

不把外键转换成属性，还要解决LEFT JOIN的情况，如何做到预关联呢？

示例代码

	A	B
1	=CUSTOMER.keys(C_CUSTKEY)	/设置主键为客户ID
2	=ORDERS.join(O_CUSTKEY,A1,~:CUSTOMER_fk)	/建立预关联
3	=A2.select(CUSTOMER_fk.C_REGION=="North china").sum(O_TOTALPRICE)	/汇总地区等于北的订单金额



全内存外键预关联 - 多个复制外键

在前面算法的基础上，多字段外键的例子，用法如下：

ORDERS

Index	O_ORDERKEY	O_CUSTKEY	O_EMPID	O_ORDERDA...	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

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

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

外键N:1

外键N:1



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)

匹配不上的记录为 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



A

B

1 =CUSTOMER.keys(C_CUSTKEY), EMPLOYEE.keys(E_ID)

/设置主键客户ID、雇员ID

2 =ORDERS.join(O_CUSTKEY, CUSTOMER, ~:CUSTOMER_fk; O_EMPID, EMPLOYEE, ~:EMPLOYEE_fk)

/建立预关联

3 =A2.select(CUSTOMER_fk.C_REGION=="North China" && EMPLOYEE_fk.E_POSITION=="sales").sum(O_TOTALPRICE)

/汇总华北地区销售代表的订单金额



部分内存化外键 - 临时指向手段

事实表大到无法放入内存，而维表较小(可以全部放入内存)，可以采用临时指向的方法来处理外键，即：边读入数据边进行外键属性化的转换。

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
9	10270	WARTH	1996-08-01	13654.0
10	10271	SPLIR	1996-08-01	908.0
11	10272	RATTC	1996-08-02	19606.0
12	10273	QUICK	1996-08-05	22821.0
13	10274	VINET	1996-08-06	601.0
14	10275	MAGAA	1996-08-07	2693.0
15	10276	TORTU	1996-08-08	4152.0
16	10277	MORGK	1996-08-09	37731.0
17	10278	BERGS	1996-08-12	18538.0
18	10279	LEHMS	1996-08-13	5166.0
19	10280	BERGS	2007-12-25	898.0
20	10281	ROMEY	1996-08-14	294.0

1 逐步读入数据并进行属性化转换

2

3

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	HUNCC	Hardware mechanical	Dalian
5	MEREP	huake	Dalian
6	ALFKI	Sanchuan industrial co. LTD	Tianjin

建立索引便于查找

内存区

示例代码

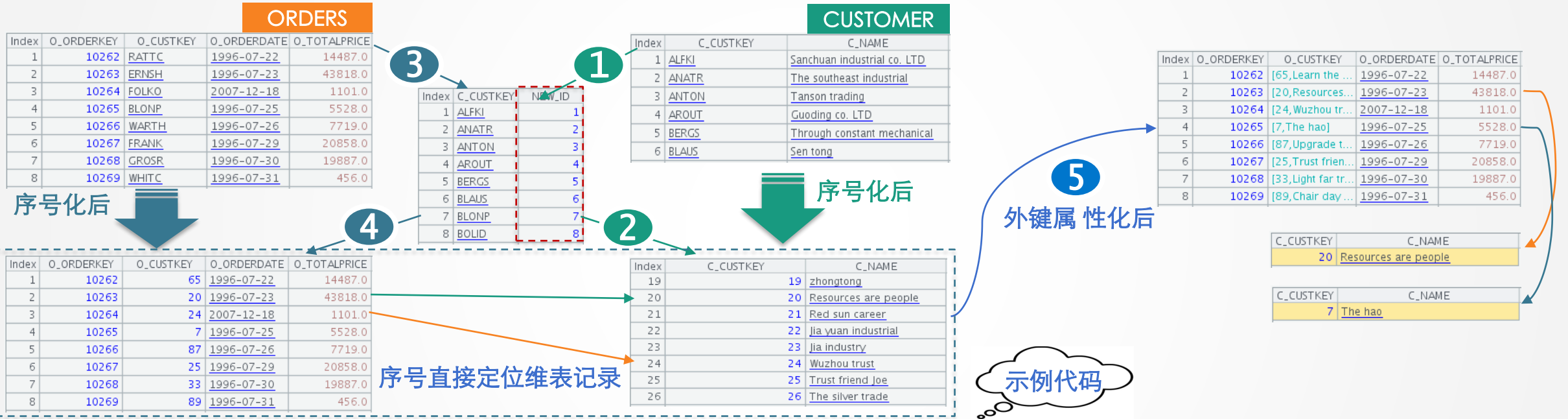
	A	B
1	=file("ORDERS.btx").cursor@b()	/建立订单表记录游标，逐步读入数据
2	=A1.switch(O_CUSTKEY,CUSTOMER:C_CUSTKEY)	/在流入数据时将订单表中的客户ID字段根据客户表的主键转换成客户表的记录
3	=A1.groups(O_CUSTKEY.C_CITY:CITY;sum(O_TOTALPRICE):AMOUNT)	/按照客户所在城市汇总订单的销售额

每次做关联时还要再计算 HASH 和比对，但维表索引建立后可复用，且同样具有一次解析全部外键以及易于并行的特点，实际场景下比 HASH 算法仍有较大优势



部分内存化外键 - 序号化

前面算法上的变种。即：如果我们能把维表的主键都转换成从 1 开始的自然数，然后用序号直接定位维表记录，不需要计算和比对 HASH 值。



	A	B
1	=file("ORDERS.btx").cursor@b()	/建立订单表记录游标，逐步读入数据
2	=A1.switch(O_CUSTKEY, CUSTOMER:#)	/利用序号建立外键关联
3	=A1.groups(O_CUSTKEY.C_NAME:C_NAME; sum(O_TOTALPRICE):AMOUNT)	/分组汇总

外键序号化本质上相当于在外存实现属性化，它也具有像在内存一样的复用机制；SQL使用了无序集合的概念，即使事先把外键序号化了，数据库也难以利用这个特点，仍然会去计算 HASH 值和比对



部分内存化外键 - 排号键

排号是由字节构成的整数用于表示键值，它定位速度快，常用于优化内存索引和外键关联

身份证号	姓名
31010519730609816	戴丽
...	...

转为排号键

身份证号	姓名
2234072400696791302	戴丽
...	...

将17位身份证号分8层: 31 | 01 | 05 | 1973 | 06 | 09 | 81 | 6

第1-2位: 1-99

1	...	10	11	...	15	16	...	31	...	90	...	99	...
---	-----	----	----	-----	----	----	-----	----	-----	----	-----	----	-----

第3-4位: 1-99

1	...	10	11	...	15	16	...	21	...	90	...	99	...
---	-----	----	----	-----	----	----	-----	----	-----	----	-----	----	-----

第5-6位: 1-99

1	...	5	15	16	...	21	...	90	...	99	...
---	-----	---	-----	-----	----	----	-----	----	-----	----	-----	----	-----

第7-10位: 表示生日年份, 这里以1970年作为基准, 从1开始:

1	...	3	15	16	...	21	...	90	...	99	...
---	-----	---	-----	-----	----	----	-----	----	-----	----	-----	----	-----

剩余7位的分层, 依次类推...

示例代码

```

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

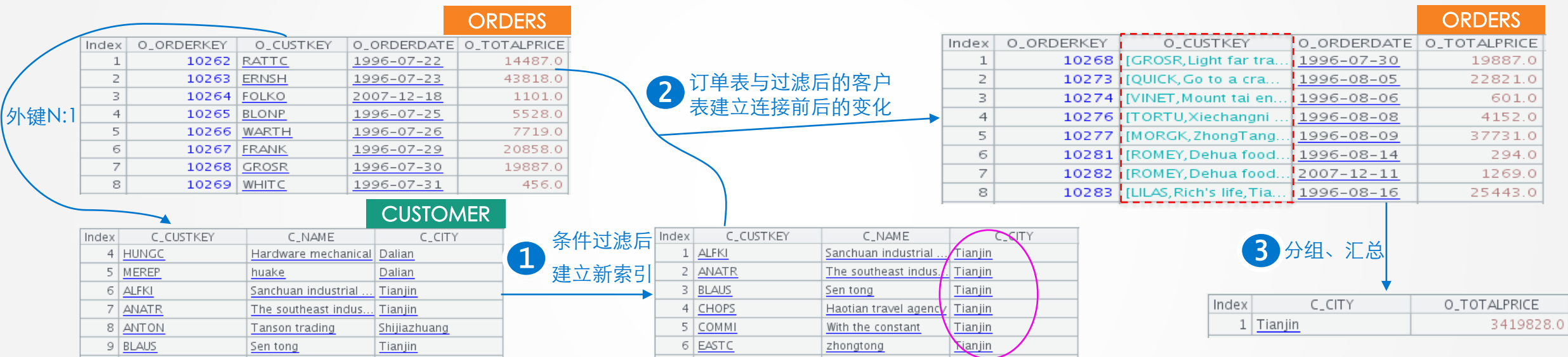
```

- 另一种处理不连续序号的方法, 避免hash计算和冲突
- 直接序号需要至少 10^{17} 个long型的空间, 会导致内存不够
- 排号键可以将数据分层序号化, 很多子节点为空, 减少内存占用



维表过滤- 利用已建索引

维表装入内存并建好索引，有时需要针对过滤后的维表做关联，这样需要重建维表索引，维表较大时建也很耗时，可利用维表已有索引建立过滤后维表的索引，不用重新计算哈希值。



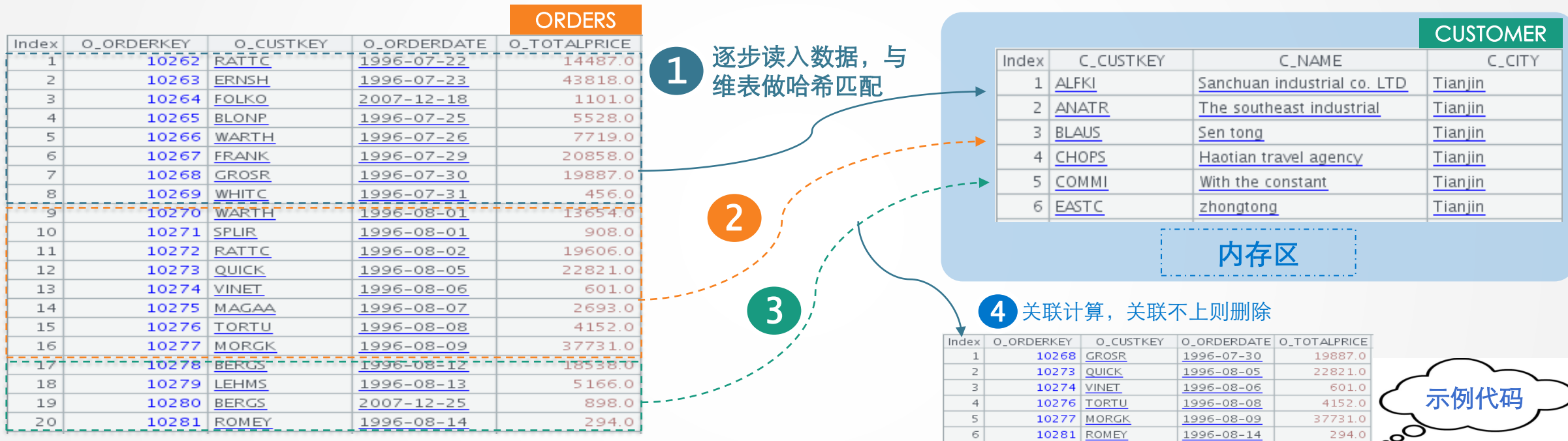
示例代码

	A	B
1	=CUSTOMER.select@i(C_CITY=="Tianjin")	/过滤客户表，并利用原有索引建立过滤后的客户表的索引
2	=file("ORDERS.ctx").create().cursor().switch@i(O_CUSTKEY,A1:C_CUSTKEY)	/外键属性化并删除关联不上的记录
3	=A1.groups(O_ORDERDATE;sum(O_TOTALPRICE):O_TOTALPRICE)	/按照订购日期汇总订单的销售额



内连接 – 维表字段仅用于过滤

事实表和维表做内连接，维表仅用于过滤中，可以读入事实表数据同时和过滤后维表关联，丢弃关联不上的记录。

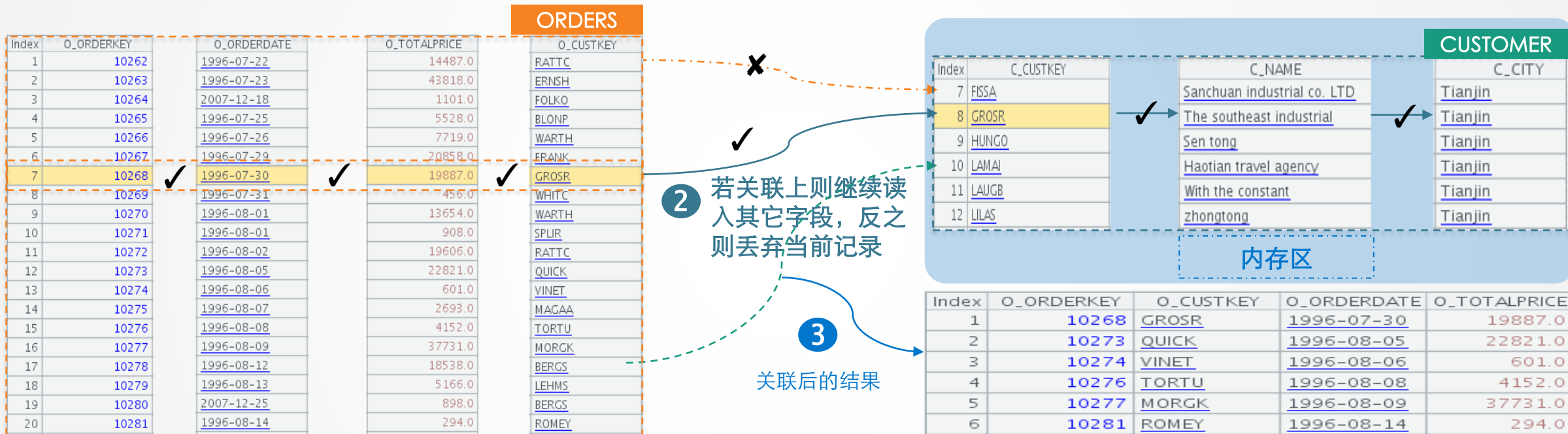


	A	B
1	<code>=file("ORDERS.ctx").create().cursor()</code>	/建立订单表记录游标，逐步读入数据
2	<code>=A1.join@i(O_CUSTKEY,CUSTOMER:C_CUSTKEY)</code>	/在流入数据时将订单表中的客户ID字段和过滤后的客户表做关联，丢弃关联不上的记录
3	<code>=A2.groups(O_ORDERDATE;sum(O_TOTALPRICE):O_TOTALPRICE)</code>	/按照订购日期汇总订单的销售额



内连接 — 游标读出时关联并过滤

在游标读出时关联并过滤，关联不上则不再读出其它字段，过滤掉较多记录时可以明显减少IO操作从而提升性能。



① 流入数据时先读入O_CUSTKEY，和C_CUSTKEY关联

示例代码

	A	B
1	<code>=file("ORDERS.ctx").create().cursor(;CUSTOMER.find(O_CUSTKEY))</code>	/在流入数据时先读入客户ID字段，然后和客户表做关联，能关联上则继续读入其它字段，反之则丢弃当前记录
2	<code>=A1.groups(O_ORDERDATE; sum(O_TOTALPRICE):O_TOTALPRICE)</code>	/按照订购日期汇总订单的销售额



内连接 - 关联过滤的同时属性化

事实表和维表做内连接，维表的字段用于过滤条件中，我们可以先对维表进行过滤，再边读入事实表数据边和过滤后维表做关联，丢弃关联不上的记录。

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
9	10270	WARTH	1996-08-01	13654.0
10	10271	SPLIR	1996-08-01	908.0
11	10272	RATTC	1996-08-02	19606.0
12	10273	QUICK	1996-08-05	22821.0
13	10274	VINET	1996-08-06	601.0
14	10275	MAGAA	1996-08-07	2693.0
15	10276	TORTU	1996-08-08	4152.0
16	10277	MORGK	1996-08-09	37731.0
17	10278	BERGS	1996-08-12	18538.0
18	10279	LEHMS	1996-08-13	5166.0
19	10280	BERGS	2007-12-25	898.0
20	10281	ROMEY	1996-08-14	294.0

1 逐步读入数据，并进行外键属性化转换

2

3

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

内存区

4 外键属性化后的结果

Index	O_ORDERKEY	O_CUSTKEY	O_ORDERDATE	O_TOTALPRICE
1	10268	[GROSR,Light far tr...	1996-07-30	19887.0
2	10273	[QUICK,Go to a cra...	1996-08-05	22821.0
3	10274	[VINET,Mount tai e...	1996-08-06	601.0
4	10276	[TORTU,Xiechangn...	1996-08-08	4152.0
5	10277	[MORGK,ZhongTan...	1996-08-09	37731.0
6	10281	[ROMEY,Dehua foo...	1996-08-14	294.0

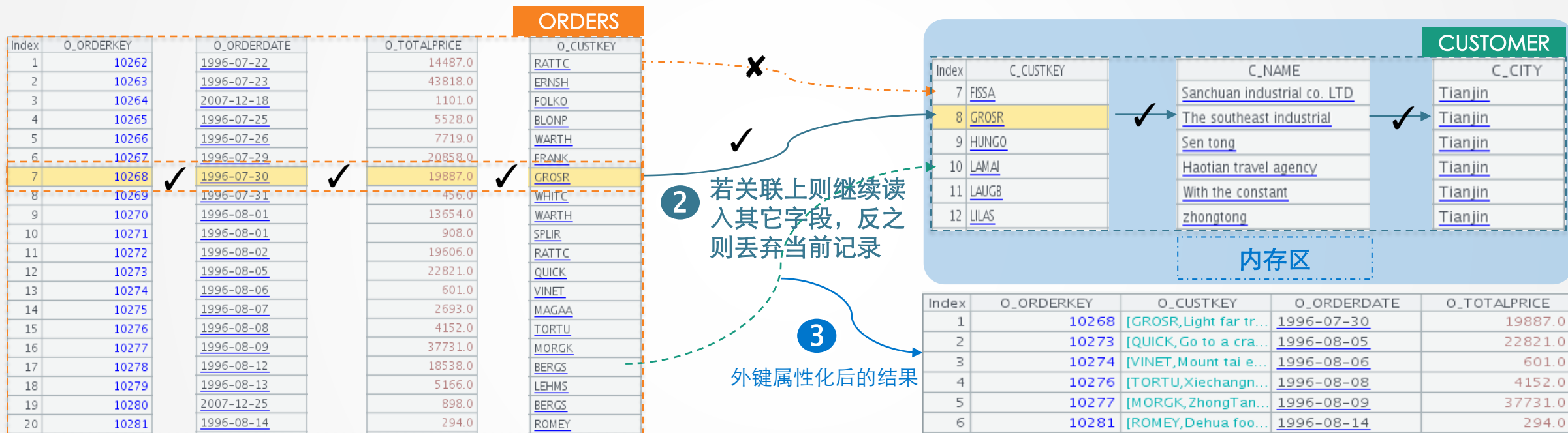
示例代码

	A	B
1	<code>=file("ORDERS.ctx").create().cursor()</code>	/建立订单表记录游标，逐步读入数据
2	<code>=A1.switch@i(O_CUSTKEY,CUSTOMER:C_CUSTKEY)</code>	/在流入数据时将订单表中的客户ID字段和过滤后的客户表做外键属性化，并删除关联不上的记录
3	<code>=A2.groups(O_CUSTKEY.C_NAME:C_NAME;sum(O_TOTALPRICE):O_TOTALPRICE)</code>	/按照客户公司名称汇总订单的销售额



内连接 — 游标读取时关联过滤并属性化

在游标读出时可以同时关联并过滤，然后再属性化，关联不上则不再读出其它字段，过滤掉较多记录时可以明显减少IO操作从而提升性能。



示例代码

	A	B
1	<code>=file("ORDERS.ctx").create().cursor(;O_CUSTKEY:CUSTOMER)</code>	/在流入数据时先读入客户ID字段，然后和客户表做外键属性化，能关联上则继续读入其它字段，反之则丢弃当前记录
2	<code>=A1.groups(O_CUSTKEY.C_NAME:C_NAME;sum(O_TOTALPRICE):O_TOTALPRICE)</code>	/按照客户公司名称汇总订单的销售额

大维表



当事实表小(内存可放入), 维表大到无法放入内存, 可以把JOIN转化为批量查找问题, 即: 用事实表关联字段在原维表中查找后相应记录做关联

维表(大)

PRODUCT

Index	P_ID	P_NAME	P_TYPERNAME	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

事实表(小)

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

按事实表批量查找后的维表记录小

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

JOIN关联结果

Index	L_ORDER...	L_PID	L_QUANTITY	P_NAME	P_TYPERNA...	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

示例代码

事实表(小)

RETURN

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

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

A1: 把退货表装入内存

A2: 给出产品表的文件对象

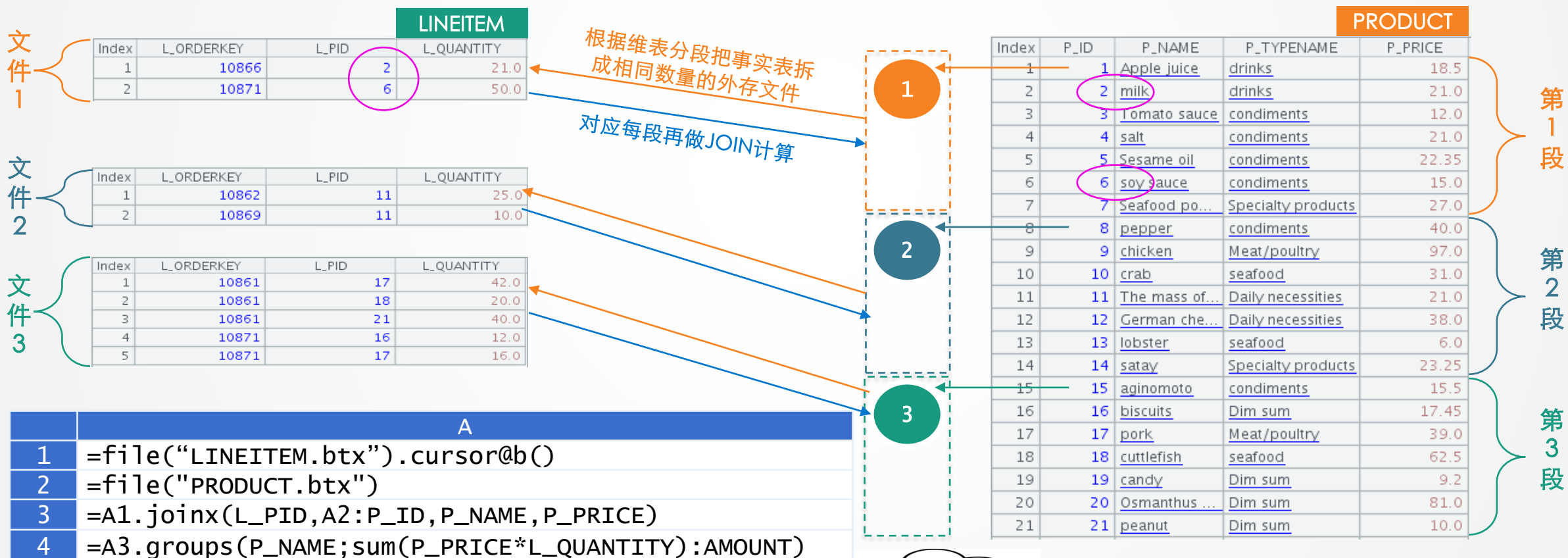
A3: 按退货表在产品表中批量查找后的结果, 退货表再与之关联

A4: 退货表与缩小范围后的产品表做关联, 计算出结果



单边HASH手段

事实表和维表都大到无法装入内存时，维表先按主键排序后即可平均分段读取(HASH分段难以保证平均)，关联时事实表根据维表分段键值划分成相同数量的临时文件，每个文件包含的维值都只对应维表的一个分段，这样只需依次读入外存文件和维表的段做连接。与传统的外存HASH JOIN相比，节省了对维表的HASH划分，也不可能出现二次HASH!



```

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: 订单明细无法装入内存，使用游标访问

A3: 关联

A2: 给出产品表的文件对象

A4: 分组汇总销售额

示例代码



目录

Contents

1

JOIN运算理解

2

外键表

3

主子与同维表

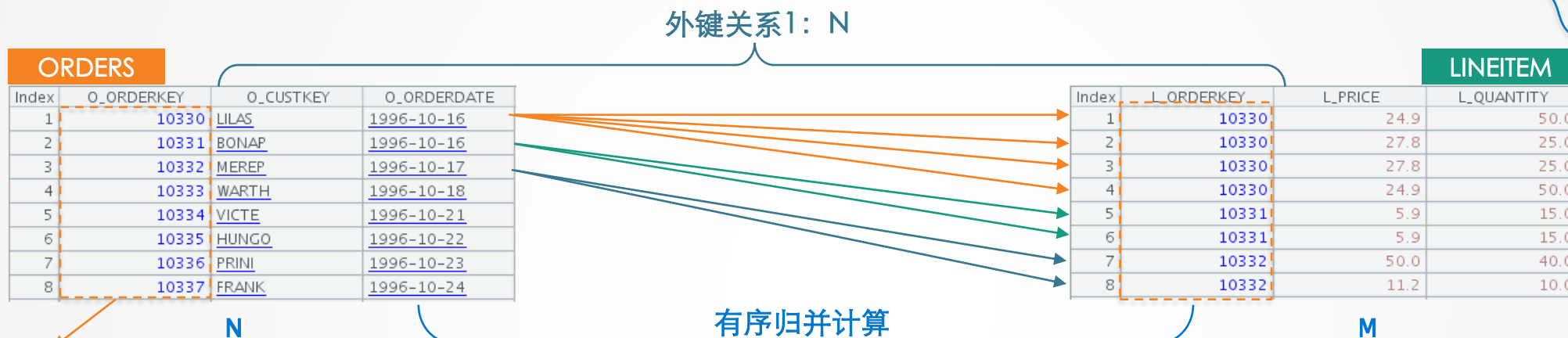
4

子查询转换

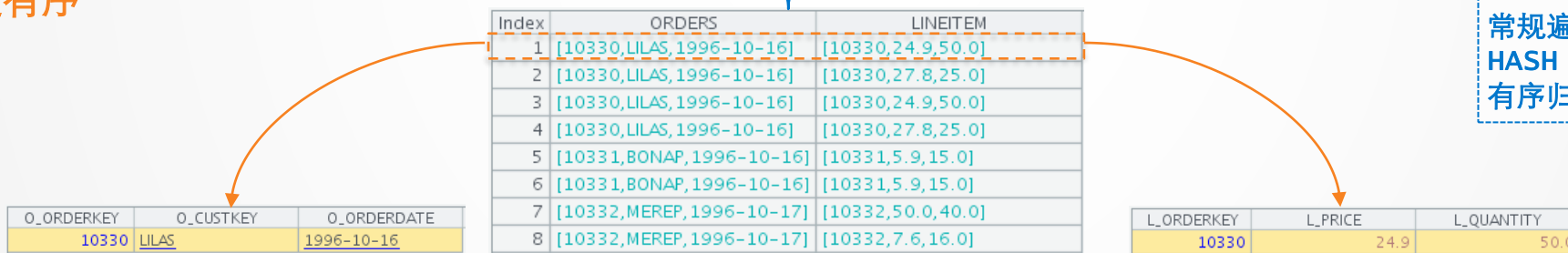


同维和主子表归并

同维表或主子表同步有序存储时，可以用归并算法一次遍历实现JOIN，复杂度要比外存分段HASH JOIN低很多！



按主键有序



常规遍历次数: $N * M$
 HASH JOIN: $\sum(N_i * M_i)$
 有序归并算法: $N + M$

示例代码

	A	B
1	<code>=ORT=file("ORDERS.btx").cursor@b(),LIT=file("LINEITEM.btx").cursor@b()</code>	/定义变量
2	<code>=joinx(ORT:ORDERS,O_ORDERKEY;LIT:LINEITEM,L_ORDERKEY)</code>	/有序归并连接
3	<code>=A2.groups(ORDERS.O_CUSTKEY:CUST;sum(LINEITEM.L_PRICE*LINEITEM.L_QUANTITY):AMOUNT)</code>	/分组汇总



并行归并 - 并行计算

并行计算能够显著提高性能，但传统HASH JOIN很难实现并行，并行HASH分段时需要同时向某个分段写出数据，造成共享资源冲突；而计算某一段又会几乎耗光所有内存，其它并行任务就无法进行。

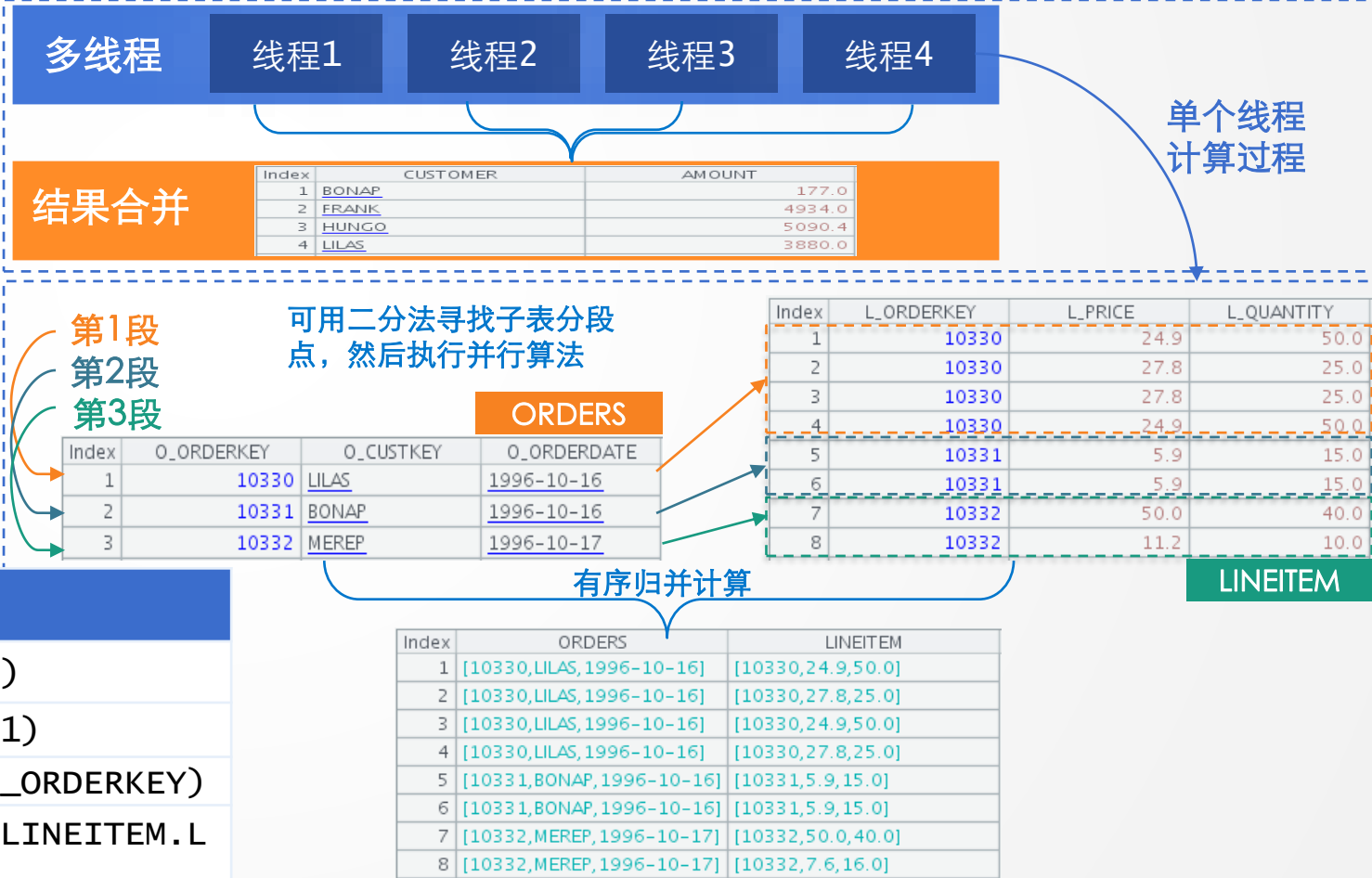
有序存储易于实现分段并行。键值有序，所以主表每段的记录键值都属于某个连续区间，子表也有这个特性，这就可以在子表中执行高效的二分查找迅速定位出分段点；即数据有序保证了分段的合理性及高效性。

示例代码：

```

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)

```



并行归并 – 生成同步数据



在生成数据时按某个基准表对齐，分段时即可保证多个表同步，归并计算时就不会发生记录错位！

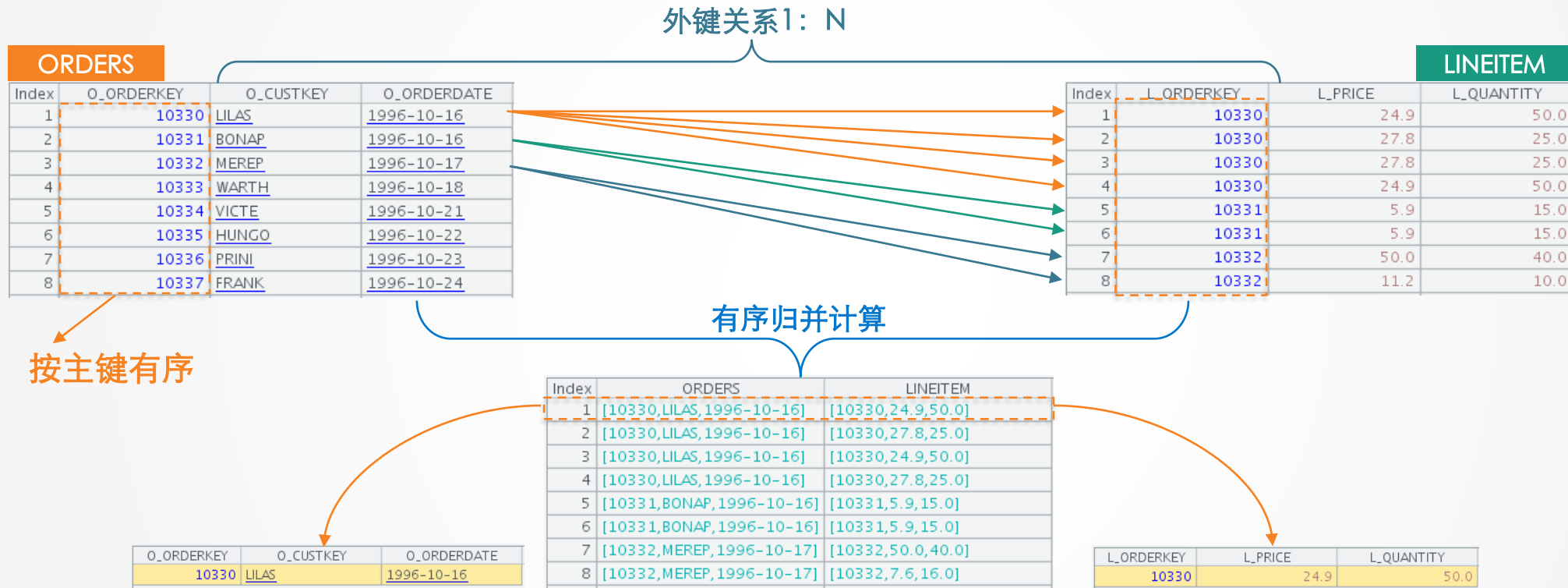
生成组表数据保证主子对齐代码：

	A	B
1	<code>=file("ORDERS.txt").cursor@t(O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)</code>	/流式读入订单表.txt
2	<code>=A1.sortx(O_ORDERKEY)</code>	/按订单ID排序
3	<code>=file("ORDERS.ctx").create(#O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)</code>	/创建、打开组表
4	<code>=A3.append(A2)</code>	/将游标中的记录追加写入到组表中
5	<code>=file("LINEITEM.txt").cursor@t(L_ORDERKEY,L_PRICE,L_QUANTITY)</code>	/流式读入订单明细.txt
6	<code>=A5.sortx(L_ORDERKEY)</code>	/按订单ID排序
7	<code>=file("LINEITEM.ctx").create(#L_ORDERKEY,L_PRICE,L_QUANTITY;L_ORDERKEY)</code>	/创建、打开组表，按订单ID字段分段，不会把订单ID同值的记录分到两段中
8	<code>=A7.append(A6)</code>	/将游标中的记录追加写入到组表中



主子表归并 - 用主表过滤子表

主表被某种条件过滤掉很多记录时，用前一页的办法关联，子表仍会被完全遍历，而这个办法会让子表遍历时根据主表的键跳，略过主表已经过滤掉的记录，减少子表的遍历，速度更快



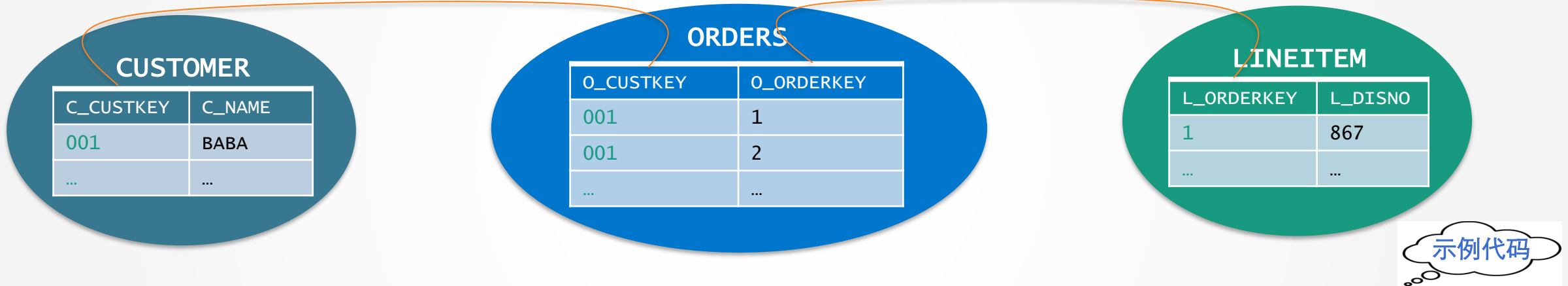
	A	B
1	<code>=file("ORDERS.ctx").create().cursor(O_ORDERKEY,O_CUSTKEY,O_ORDERDATE)</code>	/创建订单表游标
2	<code>=file("LINEITEM.ctx").create().news(A1,L_PRICE,L_QUANTITY,O_CUSTKEY,O_ORDERDATE)</code>	/通过news创建订单明细表游标，引用订单表客户ID字段
3	<code>=A2.groups(O_CUSTKEY;sum(L_PRICE*L_QUANTITY):AMOUNT)</code>	/分组汇总客户的消费额



主子表关联后按照主表主键做分组运算

主子表关联时可以把子表记录拼成主表的子集（字段取值为集合）以处理连接后又对主表分组的运算，可直接在游标里写汇总运算。

表间关系示意图：



```

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)

```



主子表关联后按照主表主键做分组运算

上一页运行结果示意图：

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

1 条件等于 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 在游标中执行主键关联和条件过滤

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

3 关联, 条件过滤、分组汇总运算

Index	O_ORDERKEY	O_ORDERDATE	O_SHIPRIORITY
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

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

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

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

4 按收入和日期排序



主子表一体化存储

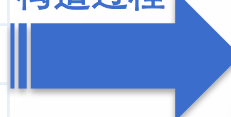
采用层次存储还可以进一步提升计算性能，将主子表组合固化在存储格式中，使用时无需再次关联从而获得更高性能！

存储举例：

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

示例代码

构造过程



示例代码

查询举例：

	A	B
1	<code>=file("MULTIPLE.ctx").create().attach(LINEITEM)</code>	/打开附表订单明细
2	<code>=A1.cursor@m(O_ID,C_ID,PRICE,NUMS;;4)</code>	/建立附表多路游标,路数4
3	<code>=A2.groups(C_ID:CUSTOMER;sum(PRICE*NUMS):AMOUNT)</code>	/按客户分组汇总销售额

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

组合存储

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		
...	

当订单1亿条，每条对应大约10条订单明细时，本案例实际测试结果：

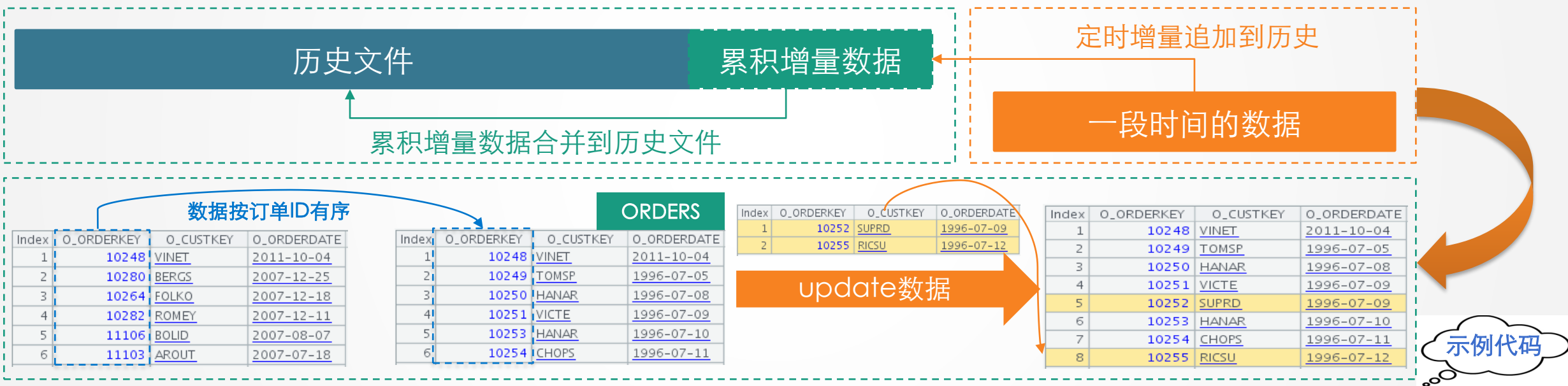
耗时单位（秒）		
2个主子joinx	主子合一	主子合一(4线程)
781	602	368



同维和主子表归并 - 有序与数据更新

有序归并的前提是将历史数据按主键排序后存储

追加数据的过程也是有序归并，把新增数据单独排序后和已有序的历史数据归并，而不必把所有历史数据重新排序，仍然是低成本的归并计算



	A	B
1	<code>=file("ORDERS.ctx").create()</code>	/打开订单表
2	<code>=db.query@x("select * from ORDERS where O_ORDERDATE>='1996-07-09' order by O_ORDERKEY")</code>	/从数据库取出新数据并关闭
3	<code>=A1.append(A2.cursor())</code>	/累计增量数据到原订单表



目录

Contents

1

JOIN运算理解

2

外键表

3

主子与同维表

4

子查询转换

子查询转换成连接 — 说明



此部分内容讲解什么情况下的子查询可以转为JOIN，JOIN的优化方法请参照前面的内容



假定所有例子中涉及到的表都按主键有序存储



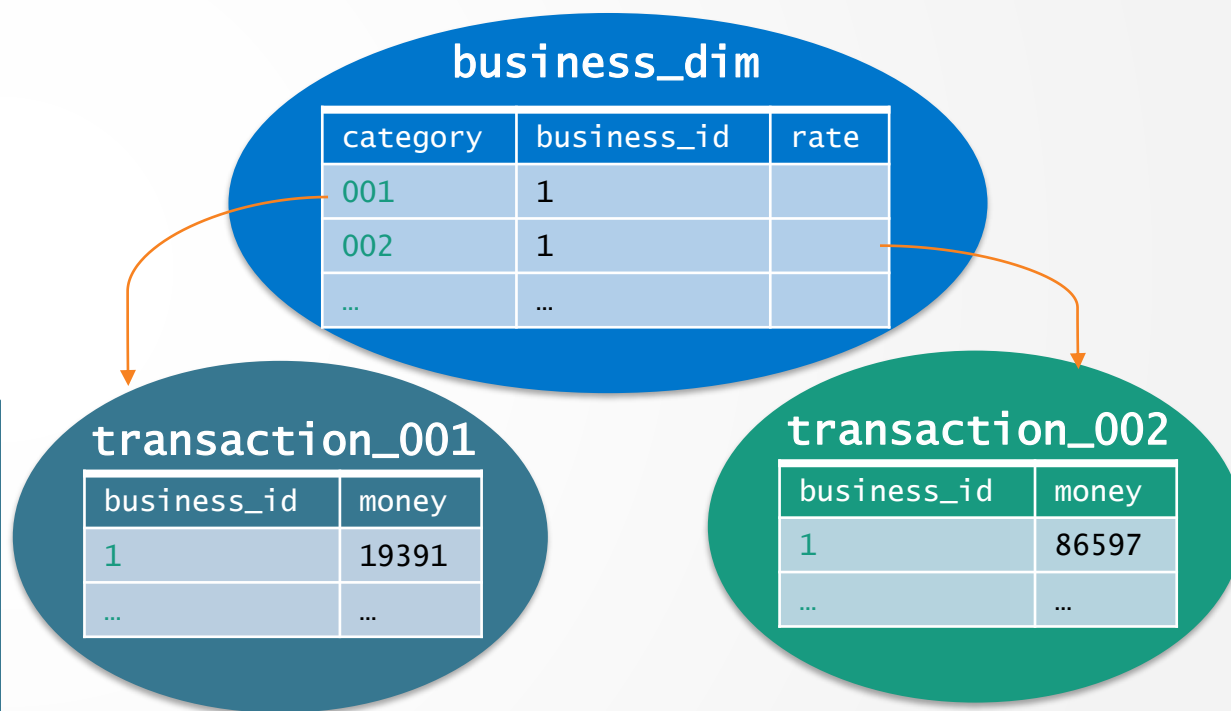
关联键只针对维表主键的部分字段

比如：银行业事实表按业务分成多个表（保险主表、理财主表等），不同业务对应的维表结构相同，因此增加了一个分类字段把这些维表合成了一个大维表。

维表结构（category, business_id为主键）：
category, business_id, rate, ...
事实表只包含business_id字段没有category字段，事实主表的category值是事先定好的，比如保险主表对应的category是001，如右图所示：

```
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





事实主表只包含维表主键的部分字段

优化思路1：对维表进行过滤得到以business_id为主键的新维表，这样就可以跟事实表用business_id字段关联，这样关连的时候维表需要按business_id字段建索引



```

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)

```

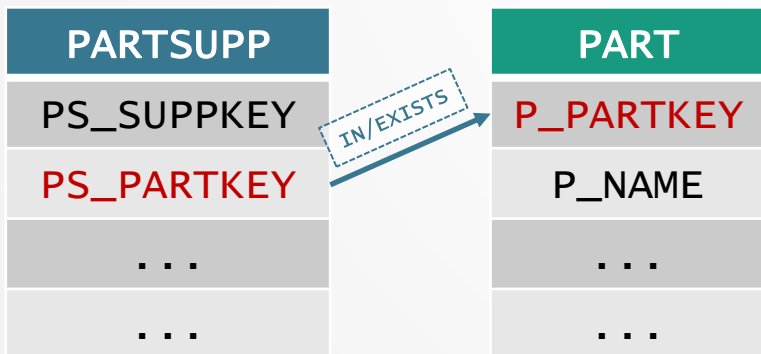


外键和维表做IN、EXISTS关联查询

① 外键和维表做IN关联查询

```
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 外键和维表做EXISTS关联查询

```
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

以上例子可转化为外键式JOIN!



外键和维表做IN、 EXISTS关联查询

优化思路：子查询过滤后读入内存并建立索引，外层表在游标读出时与子查询关联并过滤，关联不上则不再读出其它字段，过滤掉较多记录时可以明显减少IO操作从而提升性能。



示例代码

	A	B
1	<code>=file("PART.ctx").create().cursor(P_PARTKEY,P_NAME; like(P_NAME, "bisque*"))</code>	/在流入数据时对PART过滤
2	<code>=A1.fetch().index()</code>	/建索引
3	<code>=file("PARTSUPP.ctx").create().cursor(PS_SUPPKEY, PS_PARTKEY; A2.find(P_PARTKEY))</code>	/在流入数据时先读入PS_PARTKEY字段，然后和P_PARTKEY做关联，能关联上则继续读入其它字段，反之则丢弃当前记录
4	<code>=A3.groups(PS_SUPPKEY; count(1):S_COUNT)</code>	/分组、计数

主表和子表做IN、EXISTS关联查询



① 主表和子表做IN关联查询

```
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 主表和子表做EXISTS关联查询

```
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为主子表，ORDERS表的主键是O_ORDERKEY，LINEITEM表的主键为L_ORDERKEY、L_LINENUMBER；选出字段经过过滤后不是逻辑主键！



主表和子表做IN、 EXISTS关联查询

优化思路：关联字段不是逻辑主键，需对子查询分组去重后再做关联，变成类似于逻辑主键的情况！

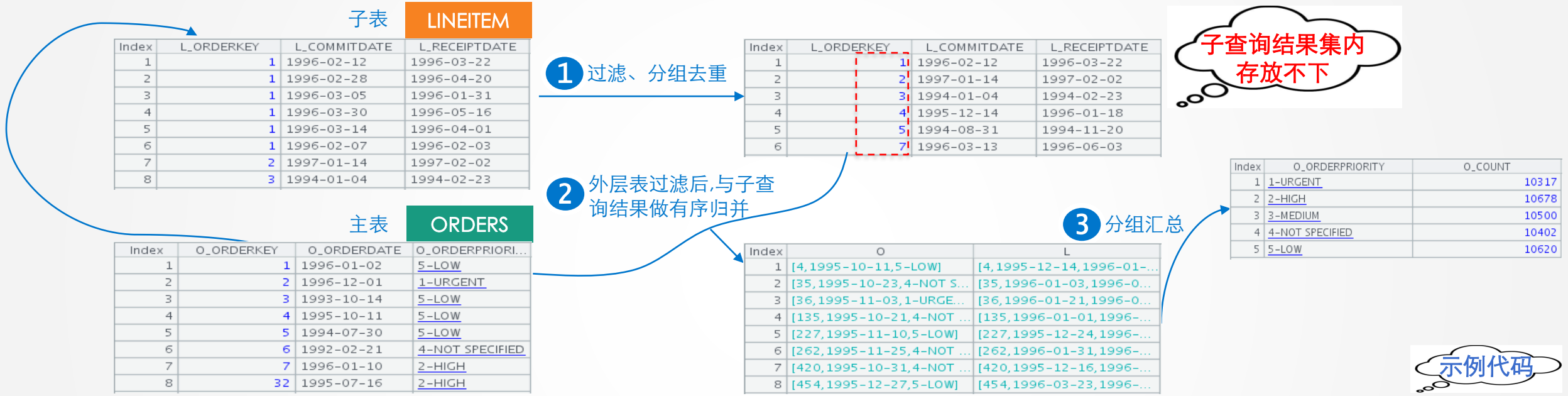


	A	B
1	1995-10-01	=after@m(A1, 3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY, L_COMMITDATE, L_RECEIPTDATE)	/在LINEITEM表所对应的集文件上定义游标
3	=A2.select(L_COMMITDATE < L_RECEIPTDATE)	/对游标附加过滤操作
4	=A3.groups(L_ORDERKEY)	/对L_ORDERKEY去重
5	=file("ORDERS.btx").cursor@b(O_ORDERKEY, O_ORDERDATE, O_ORDERPRIORITY)	/在ORDERS表所对应的集文件上定义游标
6	=A5.select(O_ORDERDATE>=A1 && O_ORDERDATE < B1)	/对游标附加过滤操作
7	=A6.join@i(O_ORDERKEY, A4:L_ORDERKEY)	/对ORDERS游标连接过滤, @i选项表示内连接
8	=A7.groups(O_ORDERPRIORITY; count(1):O_COUNT)	/对游标计算分组得到最终结果



主表和子表做IN、 EXISTS关联查询

优化思路：在上页基础上，外层表和内层表按关联字段有序，可以利用有序游标的归并连接来做优化！



	A	B
1	1995-10-01	=after@m(A1, 3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY, L_COMMITDATE, L_RECEIPTDATE)	/在LINEITEM表所对应的集文件上定义游标
3	=A2.select(L_COMMITDATE < L_RECEIPTDATE)	/对游标附加过滤操作
4	=A3.group@1(L_ORDERKEY)	/对L_ORDERKEY去重
5	=file("ORDERS.btx").cursor@b(O_ORDERKEY, O_ORDERDATE, O_ORDERPRIORITY)	/在ORDER表所对应的集文件上定义游标
6	=A5.select(O_ORDERDATE>=A1 && O_ORDERDATE < B1)	/对游标附加过滤操作
7	=joinx(A6:O, O_ORDERKEY; A4:L, L_ORDERKEY)	/做有序内连接
8	=A7.groups(O.O_ORDERPRIORITY; count(1):O_COUNT)	/对游标计算分组得到最终结果

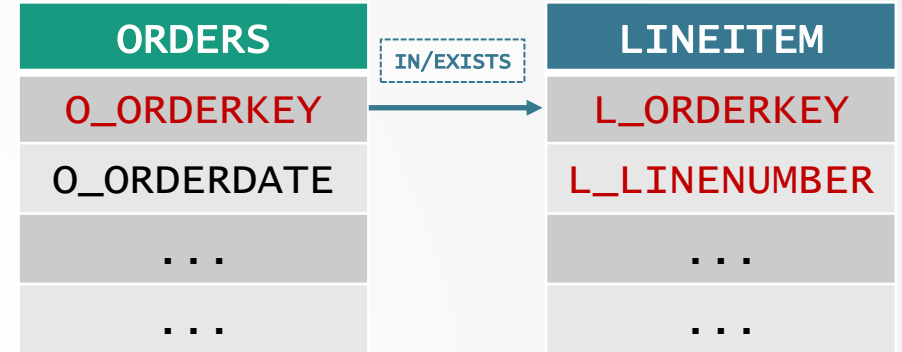


主表和子表，关联字段过滤成为逻辑主键

! 主表和子表做IN(EXIST)关联查询

```
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



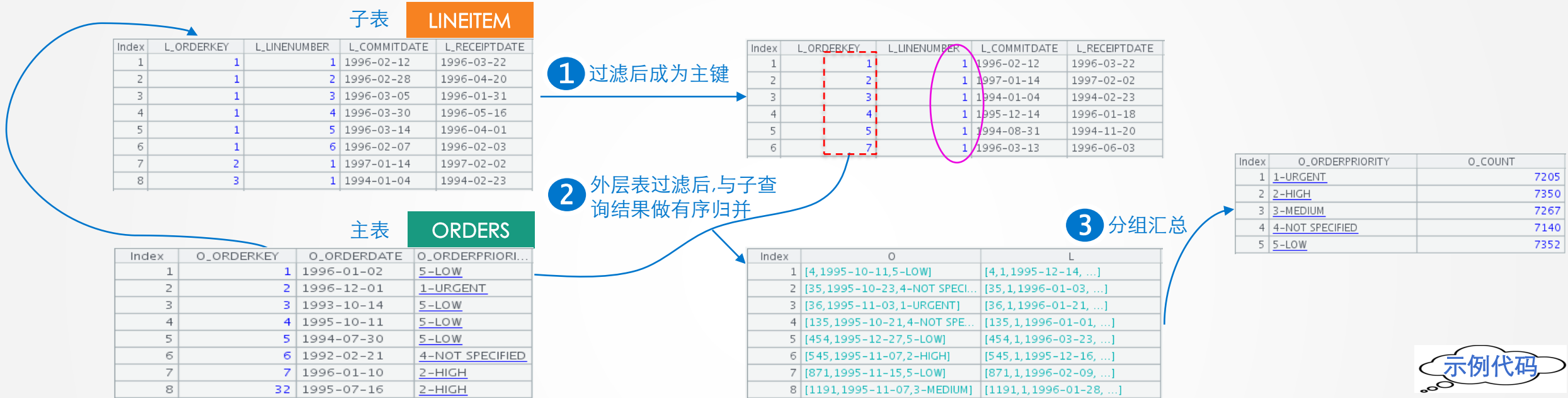
ORDERS、LINEITEM为主子表，ORDERS表的主键是O_ORDERKEY，LINEITEM表的主键为L_ORDERKEY、L_LINENUMBER；且子查询选出字段不是逻辑主键！

限定了L_LINENUMBER = 1后，选出的L_ORDERKEY成为了逻辑主键！



主表和子表, 关联字段过滤成为逻辑主键

优化思路: 外层表和内层表(过滤成为主键)按关联字段有序, 可以利用有序游标的归并连接来做优化!



示例代码

	A	B
1	1995-10-01	=after@m(A1, 3)
2	=file("LINEITEM.btx").cursor@b(L_ORDERKEY, L_LINENUMBER, L_COMMITDATE, L_RECEIPTDATE)	/在LINEITEM表所对应的集文件上定义游标
3	=A2.select(L_LINENUMBER == 1 && L_COMMITDATE < L_RECEIPTDATE)	/对游标附加过滤操作
4	=file("ORDERS.btx").cursor@b(O_ORDERKEY, O_ORDERDATE, O_ORDERPRIORITY)	/在ORDER表所对应的集文件上定义游标
5	=A4.select(O_ORDERDATE >= A1 && O_ORDERDATE < B1)	/对游标附加过滤操作
6	=joinx(A5:O, O_ORDERKEY; A3:L, L_ORDERKEY)	/做有序内连接
7	=A6.groups(O.O_ORDERPRIORITY; count(1):O_COUNT)	/对游标计算分组得到最终结果

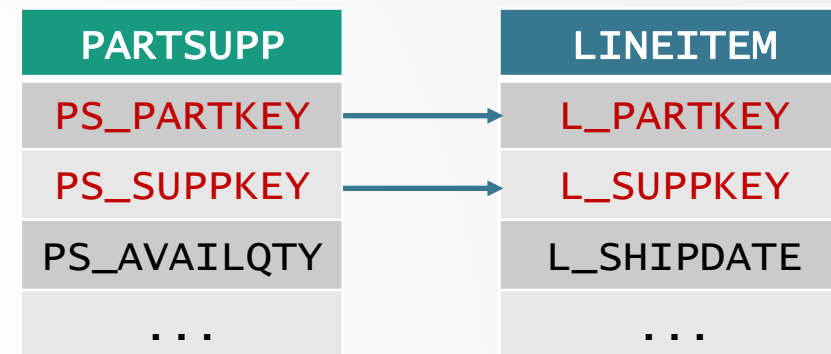
子查询转换成连接 — WHERE子查询



! WHERE子查询举例:

```
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
  )
```

SQL

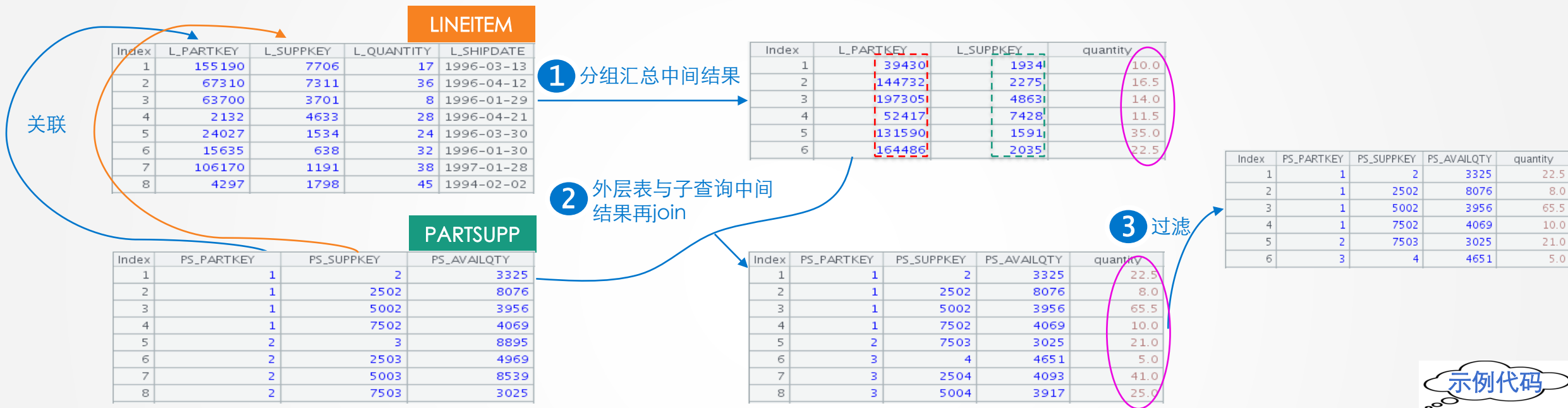


子查询里的LINEITEM 与外层主表 PARTSUPP按PARTKEY、SUPPKEY做关联，可先把LINEITEM过滤后按PARTKEY、SUPPKEY汇总出一个临时表，然后外层PARTSUPP再与汇总表做关联！



子查询转换成连接 - WHERE子查询

优化思路：子查询按照涉及到的关联字段分组，计算出一个临时维表再与外层的表做JOIN！



	A	B
1	=after@y(date,1)	/获取参数日期的下一年
2	=file("LINEITEM.btx").cursor@b(L_PARTKEY,L_SUPPKEY,L_QUANTITY,L_SHIPDATE)	/在LINEITEM表所对应的集文件上定义游标
3	=A2.select(L_SHIPDATE >= date && L_SHIPDATE < A1)	/对游标附加过滤操作
4	=A3.groups@u(L_PARTKEY,L_SUPPKEY;sum(L_QUANTITY) * 0.5:quantity)	/分组汇总,@u表示结果集不按分组字段排序
5	=file("PARTSUPP.btx").cursor@b(PS_PARTKEY,PS_SUPPKEY,PS_AVAILQTY)	/对游标附加过滤操作
6	=A5.join@i(PS_PARTKEY:PS_SUPPKEY,A4:L_PARTKEY:L_SUPPKEY,quantity)	/对PARTSUPP连接过滤,@i选项表示内连接
7	=A6.select(PS_AVAILQTY>quantity).fetch()	/对游标过滤得到最终结果

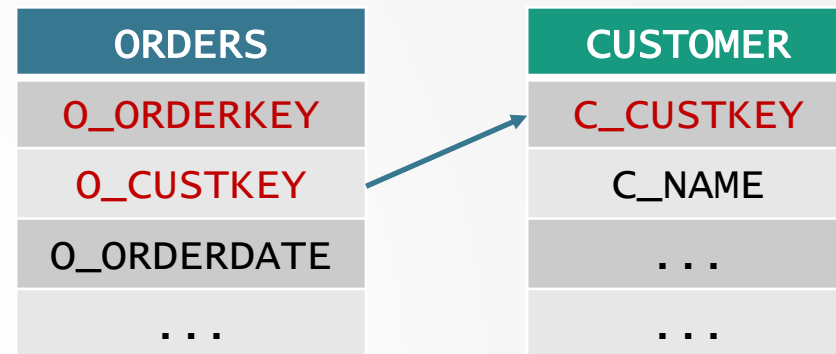
集合运算 — 差集运算



! 找出在一个表中存在而在另一个表中不存在的记录:

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

SQL



此问题可转化为两个集合的差集运算!



集合运算 — 差集运算



① 子查询结果内存可放下

② 子查询数据量大, 内存放不下, 可做归并运算

	A	B
1	=file("ORDERS.btx").cursor@b(O_CUSTKEY)	/定义游标
2	=A1.groups(O_CUSTKEY)	/O_CUSTKEY去重
3	=file("CUSTOMER.btx").cursor@b(C_CUSTKEY)	/定义游标
4	=A3.join@d(C_CUSTKEY,A2:O_CUSTKEY)	/关联, @d为取差集
5	=A4.total(count(1))	/对游标汇总记录数

	A	B
1	=file("ORDERS.btx").cursor@b(O_CUSTKEY)	/定义游标
2	=A1.groupx(O_CUSTKEY:C_CUSTKEY)	/分组去重
3	=file("CUSTOMER.btx").cursor@b(C_CUSTKEY)	/定义游标
4	=[A3,A2].mergex@d(C_CUSTKEY)	/归并, @d为取差集
5	=A4.total(count(1))	/对游标汇总记录数

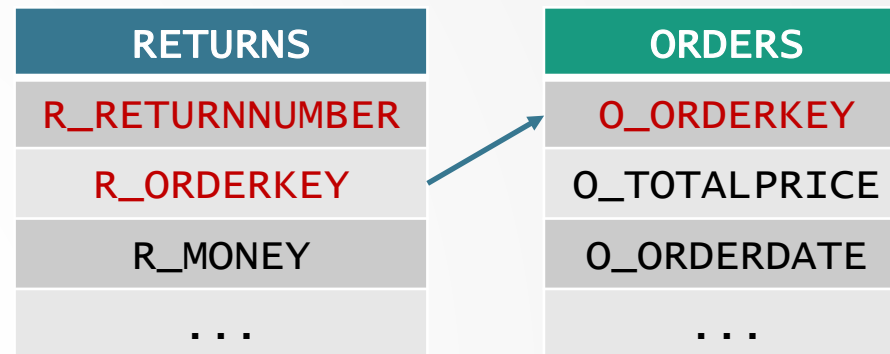
集合运算 — 交集运算



求出订单金额超过10000，但回款小于5000的订单

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

SQL



此问题可转化为两个集合的交集运算!



集合运算 - 交集运算

外键

RETURNS

Index	R_ORDERKEY	R_MONEY
1	7	70793
2	7	91934
3	7	31152
4	0	51886
5	8	77009
6	6	2996
7	7	70069
8	2	18729
9	4	37637
10	2	29569

ORDERS

Index	O_ORDERKEY	O_TOTALPRICE
1	1	173665.47
2	2	46929.18
3	3	193846.25
4	4	32151.78
5	5	144659.2
6	6	58749.59
7	7	252004.18
8	32	208660.75
9	33	163243.98
10	34	58949.67

2 订单金额超过10000

Index	O_ORDERKEY	O_TOTALPRICE
1	1	173665.47
2	2	46929.18
3	3	193846.25
4	4	32151.78
5	5	144659.2
6	6	58749.59

1 分组、汇总，找出回款小于5000

Index	O_ORDERKEY	R_MONEY
4	3	2200
5	0	3212
6	1	738
7	5	3937
8	4	544
9	7	4612

3 过滤后的ORDERS表与过滤后的RETURNS合并，取交集

Index	O_ORDERKEY	O_TOTALPRICE
1	1	173665.47
2	2	46929.18
3	3	193846.25
4	4	32151.78
5	5	144659.2
6	6	58749.59

4 统计记录数

Value
2

示例代码

	A	B
1	<code>=file("RETURNS.btx").cursor@b(R_ORDERKEY,R_MONEY)</code>	/定义游标
2	<code>=A1.group(R_ORDERKEY:O_ORDERKEY;sum(R_MONEY):R_MONEY).select(R_MONEY < 5000)</code>	/分组去重
3	<code>=file("ORDERS.btx").cursor@b(O_ORDERKEY,O_TOTALPRICE).select(O_TOTALPRICE>10000)</code>	/定义游标
4	<code>=[A3,A2].mergex@i(O_ORDERKEY)</code>	/归并, @i取交集
5	<code>=A4.total(count(1))</code>	/汇总记录数



同表关联, EXISTS 非等值条件

```
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
```

LINEITEM
L_ORDERKEY
L_LINENUMBER
L_SUPPKEY
L_COMMITDATE
...

一个订单对应 多条LINEITEM 的记录，这些记录的 L_ORDERKEY 相同且连续存储

优化思路：为找出有多个供应商供货并且有且仅有一个供应商没有按时交货的订单，因为数据是按订单顺序存放的，这样可以按订单有序分组，循环每组订单判断是否有未按时交货的订单项，是否有多个供货商，并且是否只有一个供应商没有按时交货！



同表关联, EXISTS 非等值条件

LINEITEM				
Index	L_ORDERKEY	L_SUPPKEY	L_RECEIPTDATE	L_COMMITDATE
1	1	7706	1996-03-22	1996-02-12
2	1	7311	1996-04-20	1996-02-28
3	1	3701	1996-01-31	1996-03-05
4	1	4633	1996-05-16	1996-03-30
5	1	1534	1996-04-01	1996-03-14
6	1	638	1996-02-03	1996-02-07
7	2	1191	1997-02-02	1997-01-14
8	3	1798	1994-02-23	1994-01-04

① 构成分组子集游标

Index	Member
1	[[1,7706,1996-03-22, ...],[1,7311,1996-04-20, ...],[1,3701,1996-01-31, ...]]
2	[[2,1191,1997-02-02, ...]]
3	[[3,1798,1994-02-23, ...],[3,6540,1993-11-24, ...],[3,3474,1994-01-04, ...]]
4	[[4,5560,1996-01-18, ...]]
5	[[5,8571,1994-11-20, ...],[5,3928,1994-10-19, ...],[5,35,1994-08-20, ...]]
6	[[6,2150,1992-05-02, ...]]

明细记录

Index	L_ORDERKEY	L_SUPPKEY	L_RECEIPTDATE	L_COMMITDATE
1	1	7706	1996-03-22	1996-02-12
2	1	7311	1996-04-20	1996-02-28
3	1	3701	1996-01-31	1996-03-05
4	1	4633	1996-05-16	1996-03-30
5	1	1534	1996-04-01	1996-03-14
6	1	638	1996-02-03	1996-02-07

② 选出每一组中没有按时发货的订单, 如果供应商只有一个并且此组中供应商有多个则返回结果

组的逆操作

Index	L_ORDERKEY	L_SUPPKEY	L_RECEIPTDATE	L_COMMITDATE
1	66	3490	1994-03-18	1994-03-01
2	99	849	1994-07-30	1994-04-17
3	132	9054	1993-09-22	1993-08-16
4	160	9788	1997-03-20	1997-03-11
5	193	1500	1993-12-05	1993-10-09
6	196	2353	1993-07-06	1993-05-08
7	199	9612	1996-07-04	1996-06-03
8	227	1654	1996-02-12	1995-12-24

③ 分组汇总

Index	L_SUPPKEY	numwait
1	1	23
2	2	21
3	3	21
4	4	18
5	5	24
6	6	32
7	7	23
8	8	15

示例代码

	A	B
1	=file("LINEITEM.btx").cursor@b(L_ORDERKEY, L_SUPPKEY, L_RECEIPTDATE, L_COMMITDATE)	/在LINEITEM表所对应的集文件上定义游标, 参数为选出列
2	=A1.group(L_ORDERKEY)	/对有序游标附加分组, 结果为排列构成的游标
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)))	/选出每一组中未按时发货的订单给临时变量t, 如果t长度大于0并且t中的供应商只有一个并且此组中供应商有多个则返回t, 否则返回null, conj相当于group的逆操作
4	=A3.groups@u(L_SUPPKEY; count(1):numwait)	/对游标计算分组得到最终结果

子查询转换成连接 — 总结



用子查询描述的 IN 都可以改用 EXISTS，等值 EXISTS 本质上是做连接，对于翻译 `select * from A where exists (select * from B where ...)` 样式的 SQL，要弄清楚下列特征：



关联字段是否是各表的主键或者逻辑主键？



A、B表的规模，执行其它过滤条件后是否能载入内存？



如果都不能装入内存则要考察两个表是否按关联字段有序？

优化思路：

如果有一个表能载入内存则可用内存连接方法，相关的SPL函数有 `cs.switch()`、`cs.join()`，选项 `@i`、`@d` 分别对应 `exists` 和 `not exists`

子查询要求按关联字段值唯一，如果不是逻辑主键则要先去重，可用 `A.groups()` 去重

如果两表都大不能载入内存则要考察两个表是否按关联字段有序，如果无序可以用 `cs.sortx()` 排序，有序的两表可以用 `joinx()` 做连接

创新技术 推动应用进步!

