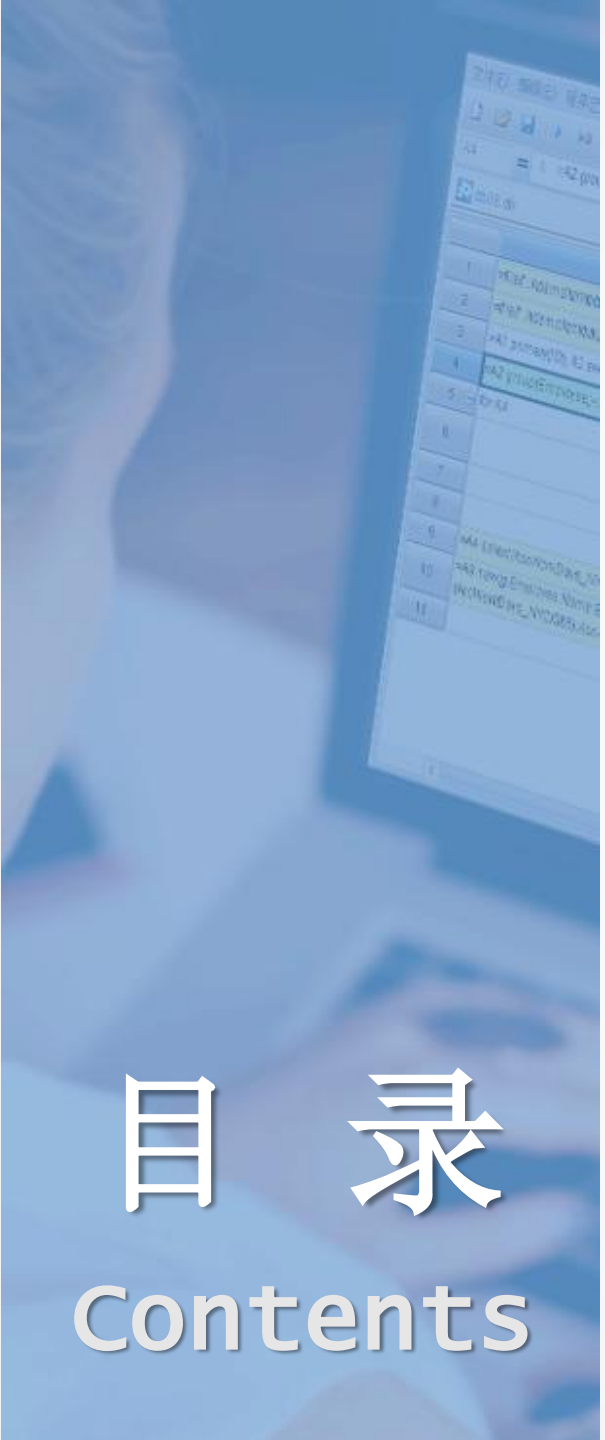
集算器

创新大数据计算引擎

性能优化——遍历

润乾软件出品



目 录

Contents

存储格式

不同数据存储方案下遍历性能差异

相同的6千万数据，不同的存储格式：

类型	时间（秒）
Oracle	328
ctx(二进制)	26
txt	50

存储格式	特点	性能排名
二进制	占用空间最小，解析最快	1
文本	文本的好处是通用，但性能不好	2
数据库	也是二进制，但普遍IO性能差， 库内遍历快，外部取数都很慢	3

	A	B	C	D	E
1	>n=4	/n是并行数	=now()		
2	>m=60000000	/m是orders表的数据记录数			
3	=int(m/n)				
4	fork to(n)				
5		=connect("oracle")	=(A4-1)*A3	=A4*A3	
6		=B5.cursor("select * from orders where rownum>"+string(C5)+" and rownum<="+string(D5))			
7		for B6,10000			
8		=B5.close()			
9	=interval@s(C1,now())				

基于数据库存储的SPL

	A	B
1	=now()	
2	=file("/home/sjr/ctx/ORDERS_orderkey.ctx").create()	
3	fork to(4)	
4		=A2.cursor(;,A3:4)
5		for B4, 10000
6		=B4.close()
7	=interval@s(A1,now())	

基于二进制存储的SPL

	A	B
1	=now()	
2	=file("/home/sjr/tpch_2_17_0/tbls/orders.tbl")	
3	>n=4	
4	fork to(n)	
5		=A2.cursor(;A4:4)
6		for B5, 10000
7		=B5.close()
8	=interval@s(A1,now())	

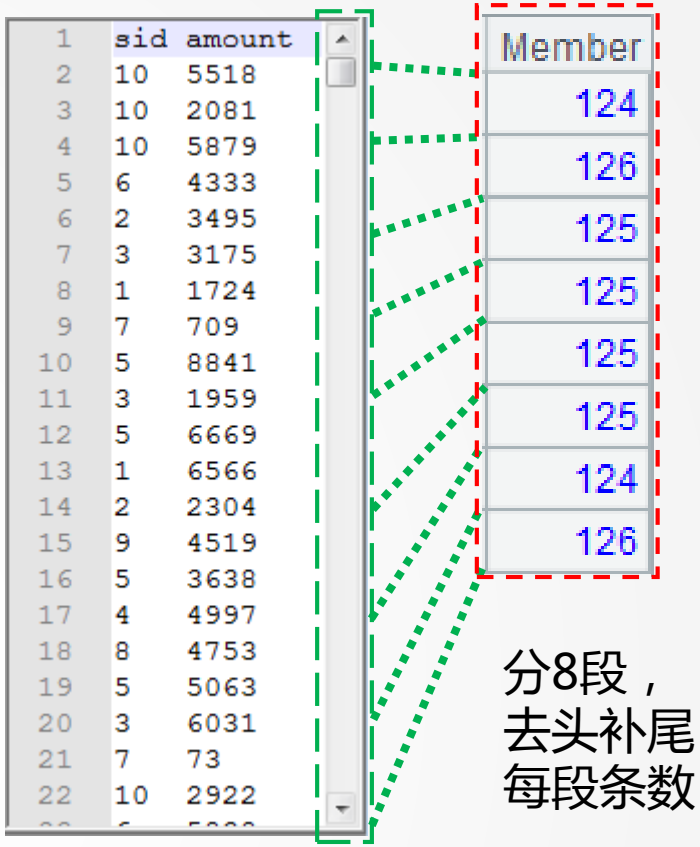
基于文本存储的SPL

分段--文本

分段为了并行，需要满足的四点要求

1. 每段的数据量基本相同
2. 分段数可灵活动态指定
3. 每个分段是连续紧凑存储的
4. 允许数据追加

文本文件可同时保证这 4 个目标。
按文件的总字节大小均分，为保证记录完整，对记录去头补尾。



sid	amount
…	…
124	1024
…	…

按总字节分段点，记录被分割

去头补尾分段点，保证记录完整

代码示例

	A
1	=file("sales.txt").cursor@t(;2:8).fetch()

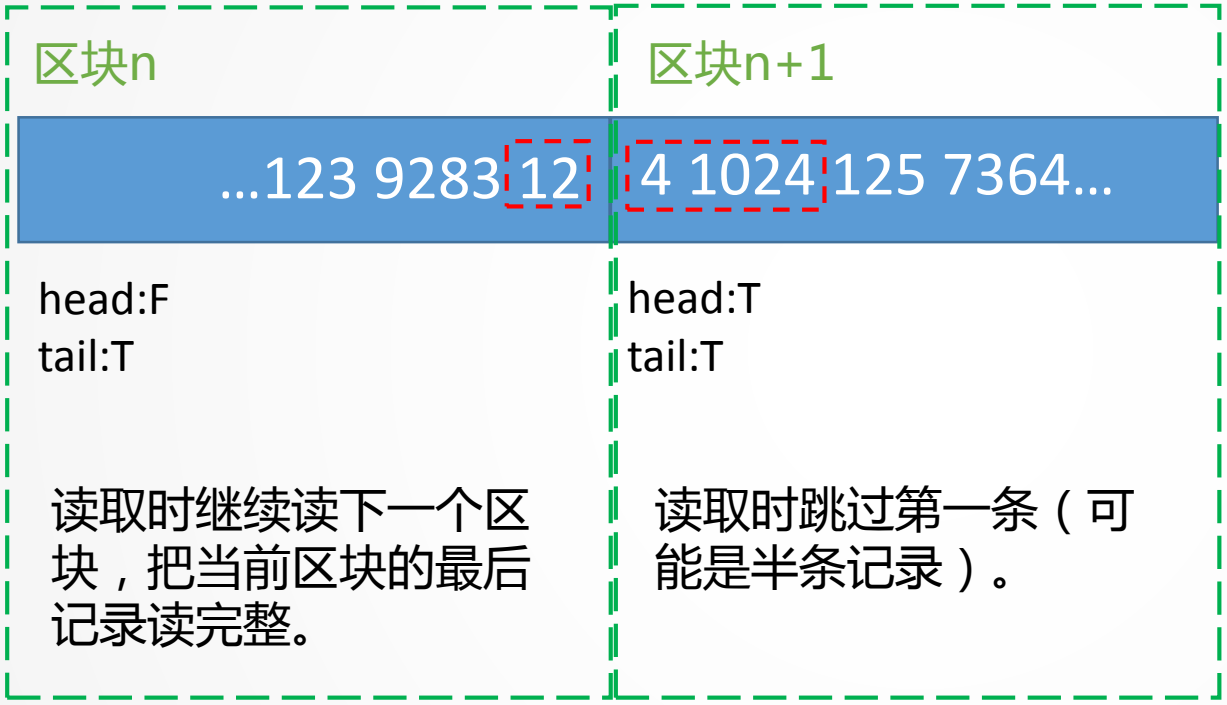
分八段读第二段，对应比较txt文件内容

123	10	5176
124	1	8193
125	8	7797
126	2	9167
127	1	8170
128	10	7769
129	6	1100
130	2	8786
131	3	6389
132	4	318
133	8	1407

sid	amount
2	9167
1	8170
10	7769
6	1100
2	8786
3	6389
4	318
8	1407

分段--二进制

二进制文件无法识别记录何时结束，需采用区块分段方案



head和tail标记表明本区块中第一条记录是否是上一区块记录的延续以及最后一条记录是否完整。T代表不完整，F代表完整。

	sid	amount
1		
2	10	5518
3	10	2081
4	10	5879
5	6	4333
6	2	3495
7	3	3175
8	1	1724
9	7	709
10	5	8841
11	3	1959
12	5	6669
13	1	6566
14	2	2304
15	9	4519
16	5	3638
17	4	4997
18	8	4753
19	5	5063
20	3	6031
21	7	73
22	10	2922

对应的文本结构示例

分段--倍增分段技术

预留一个固定长度的索引区，假设可以保存10个位置的信息

第一次满

1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
1	2	3	4	5	6	7	8	9	10

1 号位保留，2 号位填入原 3 号数的内容，3 号位填入原 5 号位的内容，…，i 号位填入原 2i-1 号位的内容

第二次满

1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
1、2	3、4	5、6	7、8	9、10	11、12	13、14	15、16	17、18	19、20

第三次满

1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
1、2、3、4	5、6、7、8	9、10、11、12	13、14、15、16	17、18、19、20	21、22、23、24	25、26、27、28、	29、30、31、32	33、34、35、36	37、38、39、40

如果继续追加记录时，第n次倍增后，每个索引区号位中填满2^(n-1)条记录才使用下一个号位

列式存储

因为列存的每列数据连续，对于取出列数较少的场景下，遍历性能优于行存

	A
1	=file_btx.cursor@b(productid,quantity,area)
2	=now()
3	for A1,1000000
4	=interval@s(A2,now())
5	=file_ctx.create().cursor(productid,quantity,area)
6	=now()
7	for A5,1000000
8	=interval@s(A6,now())

5000万行存文件取产品ID、数量、地区进行遍历，耗时51秒

5000万列存文件取产品ID、数量、地区进行遍历，耗时22秒

productid	quantity	area
…	…	…
14028	52	1221
15938	64	1112
9364	96	1467
2646	24	1230
…	…	…

另一方面，由于数据是按列连续存放的，需要多列计算会发生硬盘随机访问现象，涉及列较多时就未必会比行存有优势，多线程还会进一步加剧这个问题。在机械硬盘上用列存时要特殊注意这个问题，不一定总能提高性能。

列式存储--倍增分段

倍增分段方案是以记录数为基础的，所以对于列存也是适合的

订单ID	1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
	1、2、3、4	5、6、7、8	9、10、11、12	13、14、15、16	17、18、19、20	21、22、23、24	25、26、27、28、	29、30、31、32	33、34、35、36	37、38、39、40
用户ID	1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
	1、2、3、4	5、6、7、8	9、10、11、12	13、14、15、16	17、18、19、20	21、22、23、24	25、26、27、28、	29、30、31、32	33、34、35、36	37、38、39、40
产品ID	1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
	1、2、3、4	5、6、7、8	9、10、11、12	13、14、15、16	17、18、19、20	21、22、23、24	25、26、27、28、	29、30、31、32	33、34、35、36	37、38、39、40
…	1号位	2号位	3号位	4号位	5号位	6号位	7号位	8号位	9号位	10号位
	1、2、3、4	5、6、7、8	9、10、11、12	13、14、15、16	17、18、19、20	21、22、23、24	25、26、27、28、	29、30、31、32	33、34、35、36	37、38、39、40

各个列都采用倍增分段方式追加数据后，分段点对于各列总是落在同一条记录上，不会发生错位的情况。

有序压缩

列存更容易做合并压缩

北京	男	赵大
北京	男	钱二
北京	女	孙三
上海	男	李四
上海	女	周五

先地区排序

北京（3个），上海（2个）；男（2个），女（1个），男（1个），女（1个）；总字符数为 8

北京	男	赵大
北京	男	钱二
上海	男	李四
北京	女	孙三
上海	女	周五

先性别排序

北京（2个），上海（1个），北京（1个），上海（1个）；男（3个），女（2个）；总字符为 10

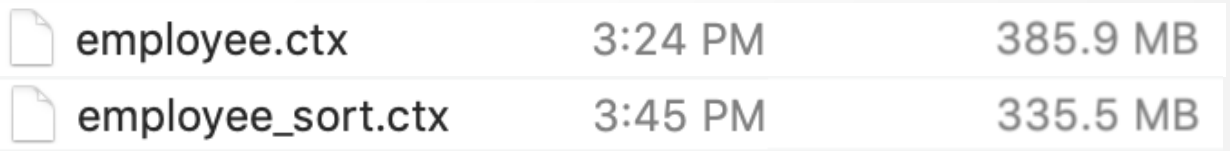
这里只数字符个数，括号中的次数不记

地区的字符数比性别要长，把长的排到前面的存储量会更小。

排序写入的代码示例

	A
1	=file_employee.create().cursor().sortx(level,height,weight,city)
2	=file_employee_sort.create(#id,name,sex,city,birthday,salary,level,height,weight,company).append(A1)

排序后压缩的列存文件容量对比



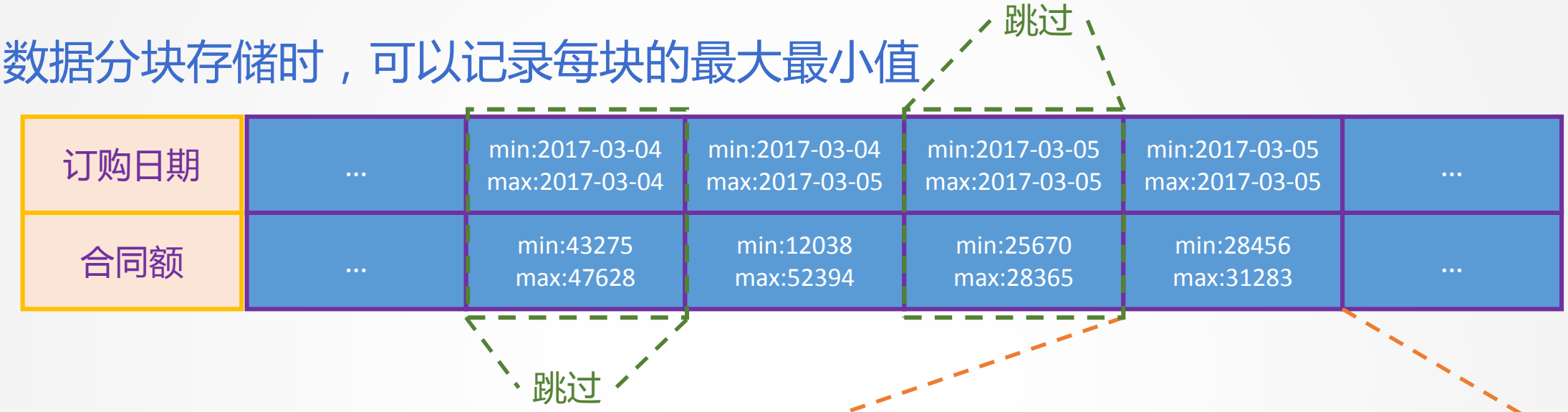
内存压缩

java内存使用敏感，内存不足时严重影响性能

数据从外存读入时不做对象化，原样保存至内存。使用时才对象化，损失性能，但减少内存占用。
列式存储和有序压缩配合也可以减少内存占用。但会加大记录产生的复杂度，所以需要权衡。

minmax索引

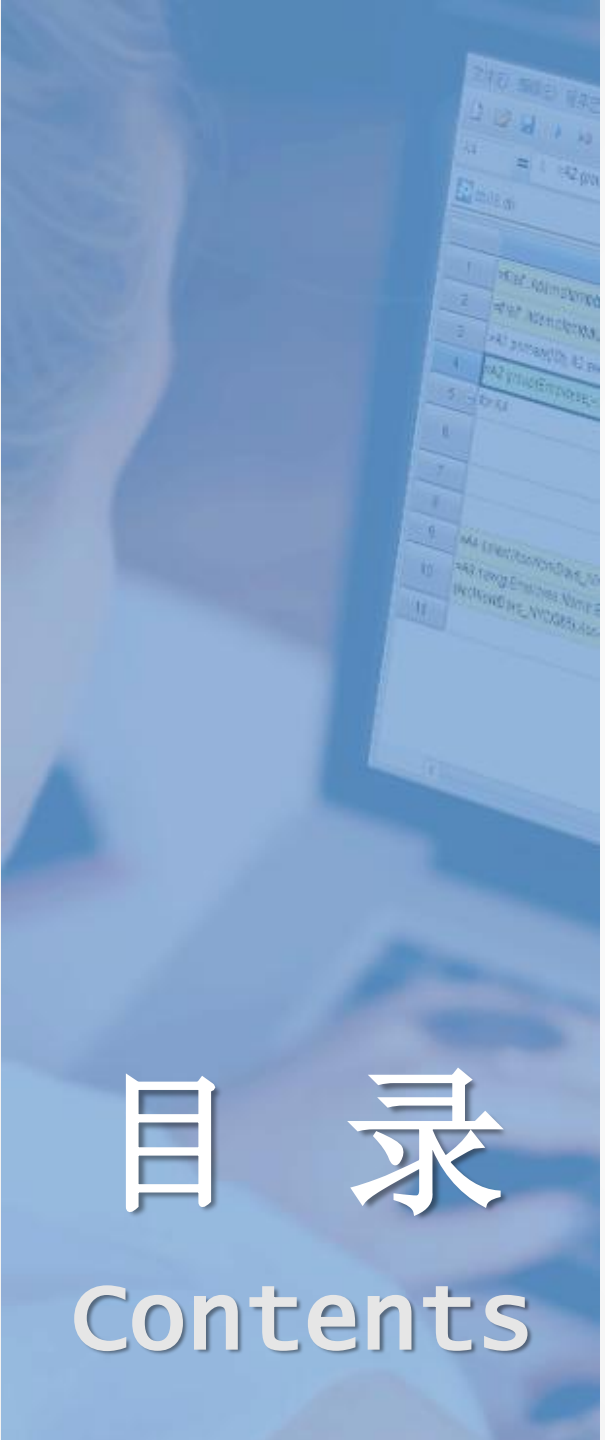
数据分块存储时，可以记录每块的最大最小值



利用数据块上的最大最小值信息，当有过滤条件时，若条件的区间与最大最小值区间没有交集，则直接跳过该块数据。

比如合同额的过滤区间为29000~30000，则绿色虚线的数据块被跳过。

订购日期	合同额	订单号	...
2017-03-05	28456	7364875	...
2017-03-05	29137	7364876	...
...	...	...	...
2017-03-05	30294	7517645	...
...	...	...	...



目 录

Contents

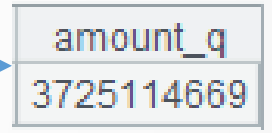
立即计算

使用游标时，一些计算会立即执行

比如：cs.groups这样的聚合函数

求所有订单的价格总和

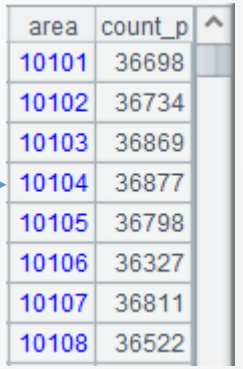
	A
1	=order_cursor.groups(;sum(price):amount_q)



amount_q
3725114669

按照订单的区域进行订单数量的统计

	A
1	=order_cursor.groups(area;count(~):count_p)



area	count_p
10101	36698
10102	36734
10103	36869
10104	36877
10105	36798
10106	36327
10107	36811
10108	36522

延迟计算

还有些运算在游标上定义后不会马上计算

比如：cs.select/cs.new/…

求订单折扣小于0.5的价格总和

	A
1	=order_cursor.select(discount<0.5)
2	=A1.groups(;sum(price):amount_q)

Value
com.raqsoft.dm.cursor.BFileCursor@2233a68f

amount_q
1843718959

延迟计算的好处是写起来和内存运算差不多，容易理解，但又不会真地生成中间结果集（内存计算就会）占用空间（或缓存）

游标前过滤

可减少java对象的产生；利用minmax索引，列存时不访问的列直接跳过

举例：统计折扣率大于0.9的订单，其所在各区域下的价格总和

	A	
1	=now()	游标前过滤 耗时26秒
2	=order_file.create().cursor(;discount>0.9)	
3	=A2.groups(area;sum(price):amount)	
4	=interval@s(A1,now())	
5	=now()	对游标用 select过滤 耗时56秒
6	=order_file.create().cursor()	
7	=A6.select(discount>0.9).groups(area;sum(price):amount)	
8	=interval@s(A5,now())	

过滤条件

找出姓名包含”张”字，且薪资小于10000的员工

	A
1	=employee_ctx.create().cursor(;salary < 10000　&& like(name,“*张*")).fetch()

多个条件 && 时注意书写次序，如果前面的子项为假时，后面就不会再计算了，这样把容易为假的条件项写到前面，会减少后面条件项的计算次数。

薪资小于10000的员工数，相比姓名中包含”张”字的员工数要少，把结果集较小的条件项写到前面，后面条件项在较小的前项结果集再过滤，也会减少计算次数。

条件1	条件2	&&后
false	ture	false
false	false	false
ture	false	false
ture	ture	ture

多路游标（内存）

数据分段后，可以使用多路游标并行遍历

举例：按城市统计用户数（user为序表）

	A
1	=user.cursor@m(;4)
2	=A1.groups(city;count(~):num)

city	num
Aiken	66505
Albany	66213
Albuquerque	65928
Anchorage	65849
Annapolis	66127
Asheville	65790
Atlanta	66182
Atlantic City	66165
Augusta	66071
Austin	66335
Baltimore	65756

多路游标的分组聚合使用与普通游标一样

id	city
1	Atlanta
2	Charleston
3	Tuscaloosa
4	Lawrence
5	OklahomaCity
6	Honolulu
7	Cleveland
8	Pennsylvania
9	Virginia Beach
10	Tulsa
11	Greenville

第1段

id	city
2490369	Detroit
2490370	Jersey City
2490371	Columbus
2490372	Rochester
2490373	Las Vegas
2490374	Birmingham
2490375	Bismark
2490376	Albuquerque
2490377	Dover
2490378	Jersey City
2490379	Charlotte

第2段

4路游标各段数据的部分内容

id	city
4997121	Norman
4997122	Birmingham
4997123	Pennsylvania
4997124	Long Island
4997125	Saint Louis
4997126	Billings
4997127	Washington
4997128	Sacramento
4997129	Bloomington
4997130	Norfolk
4997131	Tampa

第3段

id	city
7503873	Lansing
7503874	Huntington
7503875	Huntsville
7503876	Reno
7503877	Denver
7503878	Springfield
7503879	Charlotte
7503880	Lafayette
7503881	Topeka
7503882	Tucson
7503883	Meridian

第4段

多路游标（外存）

数据分段后，可以使用多路游标并行遍历

举例：统计各城市下身高大于170cm的男性用户（user_ctx为组表文件对象）

	A
1	=user_ctx.create().cursor@m(;gender==“M” && height>170;4)
2	=A1.groups(city;count(~):num)

city	num
Aiken	66505
Albany	66213
Albuquerque	65928
Anchorage	65849
Annapolis	66127
Asheville	65790
Atlanta	66182
Atlantic City	66165
Augusta	66071
Austin	66335
Baltimore	65756

多路游标的分组聚合使用与普通游标一样

id	city
1	Atlanta
2	Charleston
3	Tuscaloosa
4	Lawrence
5	OklahomaCity
6	Honolulu
7	Cleveland
8	Pennsylvania
9	Virginia Beach
10	Tulsa
11	Greenville

第1段

id	city
2490369	Detroit
2490370	Jersey City
2490371	Columbus
2490372	Rochester
2490373	Las Vegas
2490374	Birmingham
2490375	Bismark
2490376	Albuquerque
2490377	Dover
2490378	Jersey City
2490379	Charlotte

第2段

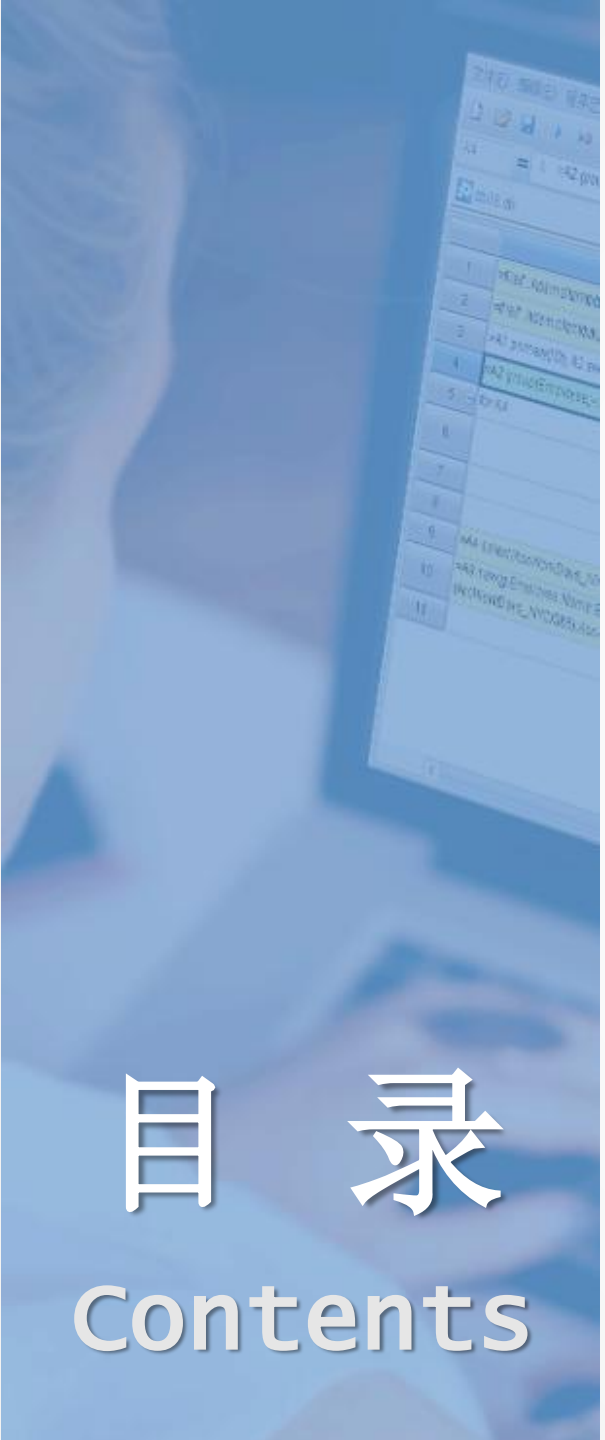
4路游标各段数据的部分内容

id	city
4997121	Norman
4997122	Birmingham
4997123	Pennsylvania
4997124	Long Island
4997125	Saint Louis
4997126	Billings
4997127	Washington
4997128	Sacramento
4997129	Bloomington
4997130	Norfolk
4997131	Tampa

第3段

id	city
7503873	Lansing
7503874	Huntington
7503875	Huntsville
7503876	Reno
7503877	Denver
7503878	Springfield
7503879	Charlotte
7503880	Lafayette
7503881	Topeka
7503882	Tucson
7503883	Meridian

第4段



目 录

Contents

小分组

内存可以装下分组后结果集时

举例：某网上商城用户在千万级，订单数据量上亿

id	gender	type
3679666	F	6071
5538579	F	46
3479704	F	3351
4599465	F	6024
2141998	F	3061
148397	M	3480
709163	M	3222
8409353	M	785
5568863	M	9355
804994	F	7679
3599727	F	3335

订单部分数据

按订单类别分组统计订单数量（一万个type）

	A
1	=order_cursor.groups(type;count(~):count)

type	count
1	10119
2	10108
3	10197
4	10065
5	10086
6	10118
7	10068
8	10118
9	10001
10	9927
11	10059

一万条结果
内存可以装下

大排序

当内存中无法装下需要排序的数据时

举例：亿级订单信息中，对用户ID进行排序

	A
1	=order_ctx.create().cursor().sortx(userid).fetch(10)

①读一部分订单记录写出成临时文件

tmpdata2238386729201306812
tmpdata3908783783193554803
tmpdata5387736568463644291
tmpdata5284908551445175383
tmpdata5207250637950334006
tmpdata7241345240835400243
tmpdata425586817048643446
tmpdata5260285369551512440
tmpdata2235245067521657899

②查看临时文件

	A
1	=file("tmp/tmpdata22352450675216578 99").cursor@b()

③每个临时文件对用户ID有序

Index	orderid	userid
1	1195090	147
2	1177245	281
3	1411649	305
4	531773	537
5	1457604	859
6	821707	1571
7	687850	1717
8	2268545	1739
9	2276485	1992
10	295927	2091
11	1326086	2141
12	1184737	2378
13	2272474	2388
14	2098612	3433
15	781530	3597
16	1875256	3853
17	706940	4635
18	558733	4803
19	2200906	4823

④最后将临时文件做多路归并

大分组

当分组后的结果集大到内存无法装下时

id	gender	type
3679666	F	6071
5538579	F	46
3479704	F	3351
4599465	F	6024
2141998	F	3061
148397	M	3480
709163	M	3222
8409353	M	785
5568863	M	9355
804994	F	7679
3599727	F	3335

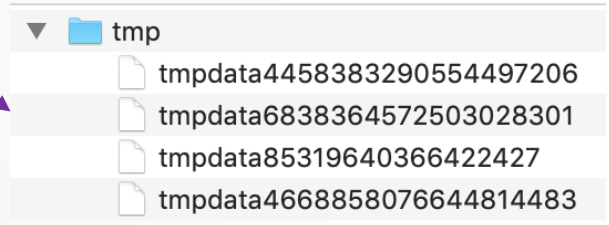
订单部分数据

大分组的情况：按用户id分组统计订单数量（千万个用户id）

	A
1	=order_cursor.groupx(id;count(~):count)
2	=A1.fetch(1000)
3	>A1.close()

id	count
1	12
2	8
3	7
4	8
5	15
6	13
7	4
8	8
9	14
10	7
11	16

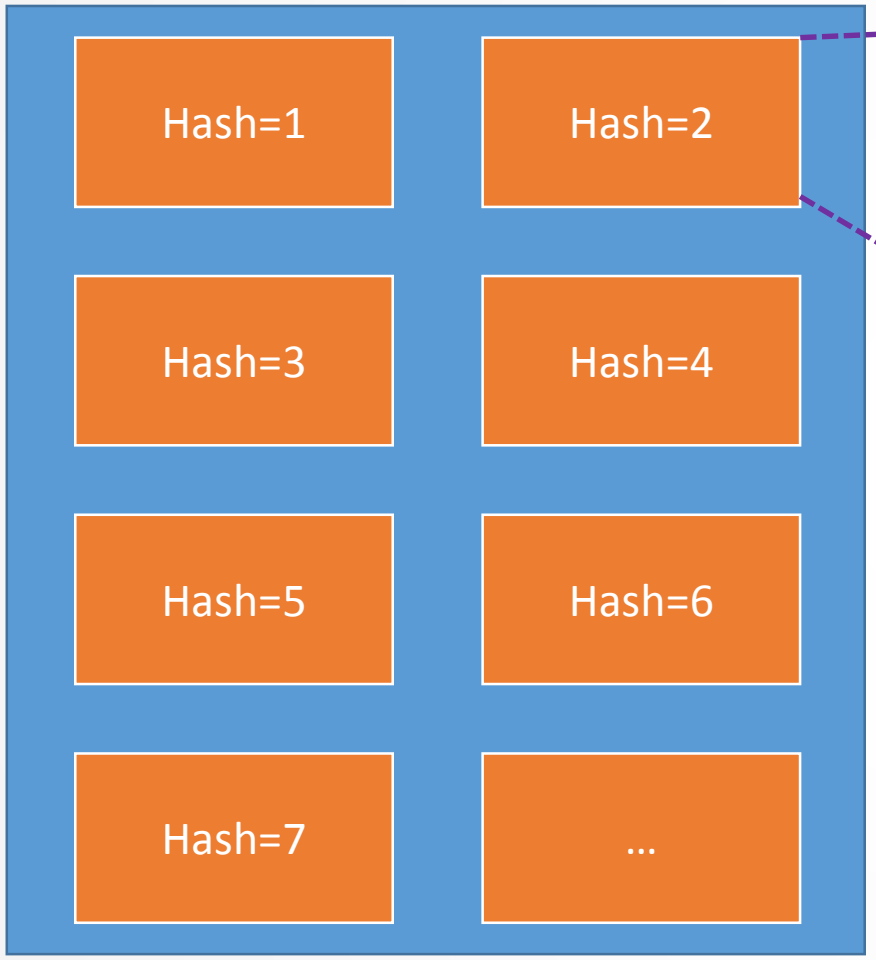
全量结果内存无法装下，这里取前1000个用户id



外存缓存文件

按用户id分组后，结果集大到内存无法一次装下。采用类似大排序的方法，生成临时外存缓存文件，利用外存来完成分组汇总计算。

hash大分组



userid	price	…
102	20	…
202	10	…
202	5	…
102	15	…
…	…	…

不同userid的Hash键相同，再遍历找同值聚合

	A
1	=order_ctx.create().cursor()
2	=A1.groupx@u(userid;sum(price):amount)
3	=A2.fetch()

通过hash计算分组字段得到hash键

由于hash函数对排序字段非递增，分组结果无序

聚合理解

业务上有意义的其它形式聚合运算--将topN理解成聚合

全集上的topN

	A	B
1	=order_ctx_cursor.groups(;top(-3;due))	//所有订单中应缴款最高的三条订单记录

分组后的topN

	A	B
1	=order_ctx_cursor.groups(area;top(-2;due):top2)	//每个地区应缴款最高的两条订单记录

area	top2
10101	[[11967703,553329878,64528, ...],[35545768,5...
10102	[[13194936,458055115,207603, ...],[19183387,...
10103	[[41484229,565824001,107989, ...],[21870156,...
10104	[[48504327,19439204,202589, ...],[26110931,5...
10105	[[27906626,385692639,141642, ...],[37720790,...
10106	[[21837967,297887618,268908, ...],[7448959,5...

	customer	email	contractam...	due	invoiceam...	service
2	Wendy Bach	efk0n@col...	59974	59836	59414	2010-06
9	Charlene ...	1xr7eo@co...	59863	59790	59790	2009-05

有序分组

当分组字段本身有序时，可以采用有序分组

举例：订单数据（日期有序），统计2018年每个月的订单数量

	A	
1	=order_ctx_cursor.group(month(orderdate):month;count(~):count).fetch()	/当订购日期的月份变化，则作为一个新的分组

orderid	userid	orderdate	...
826378	283674	2018-01-01	...
...	...	...	...
19387343	63742	2018-02-01	...
...	...	...	...
83625134	109527	2018-03-01	...
...	..	...	...

1月数据

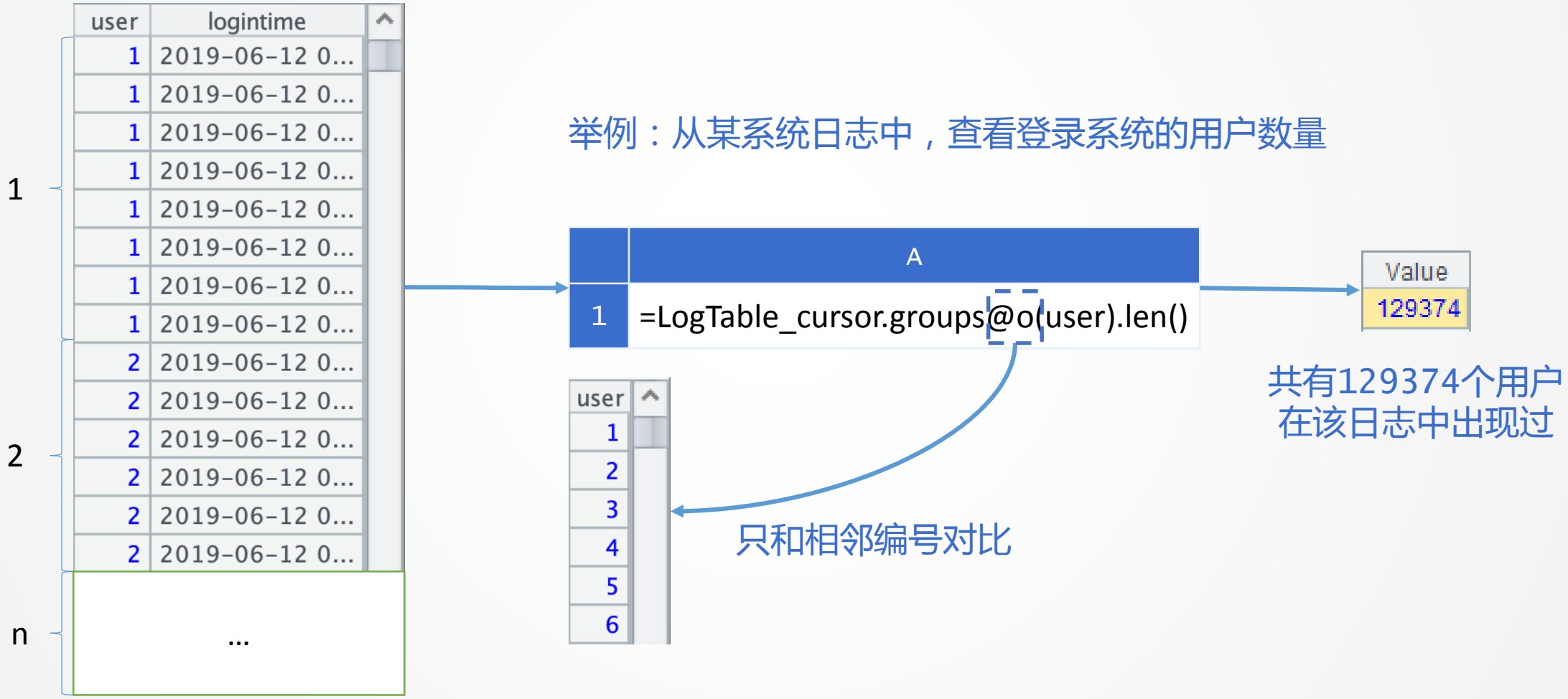
2月数据

3月数据

month	count
1	10567435
2	13522376
3	13583181
4	11247788
5	13318732
6	12294849
7	12451471
8	14102229
9	10039713
10	10531743
11	13792462
12	13201660

有序去重

当数据有序时，使用归并分组快速去重（count distinct）



半序分组

当某字段有序时，需要按后面的字段分组

举例：按日期有序的订单文件，需要对产品字段分组统计销售额

orderdate	productid	due
2018-01-01	16	36250
2018-01-01	39	39719
2018-01-01	51	25696
2018-01-01	65	27279
2018-01-01	24	27834
2018-01-01	15	34660
2018-01-01	42	32415
2018-01-01	87	38065
2018-01-01	22	21338
2018-01-01	34	21130

原数据对日期有序

	A
1	=order_ctx_cursor.group@q(orderdate;productid;sum(due):amount).fetch()

大数据计算时，对某字段有序，仅后面的字段需要分组可以使用@q选项，进行内存分组。

这样的处理方式可以避免大分组的外存临时文件读写，提高效率。

orderdate	productid	amount
2018-01-01	94	924067
2018-01-01	95	811462
2018-01-01	96	1066541
2018-01-01	97	928465
2018-01-01	98	659242
2018-01-01	99	745950
2018-01-01	100	732033
2018-01-02	1	896970
2018-01-02	2	790339
2018-01-02	3	847484
2018-01-02	4	618941

对日期和产品字段分组后统计销售额

半序排序

当某字段有序时，仅后面的字段需要排序

举例：按日期有序的订单文件，需要对产品字段排序

	A
1	=order_ctx_cursor.group@qs(orderdate;productid).fetch(10000)

相比半序分组，这里仅排序不做分组聚合

orderdate	productid	due
2018-01-01	16	36250
2018-01-01	39	39719
2018-01-01	51	25696
2018-01-01	65	27279
2018-01-01	24	27834
2018-01-01	15	34660
2018-01-01	42	32415
2018-01-01	87	38065
2018-01-01	22	21338
2018-01-01	34	21130

原数据对日期有序

半序排序后

orderdate	productid	due
2018-01-03	100	28036
2018-01-03	100	24317
2018-01-03	100	33572
2018-01-03	100	29315
2018-01-03	100	36037
2018-01-03	100	26493
2018-01-04	1	33688
2018-01-04	1	24155
2018-01-04	1	20183
2018-01-04	1	37617

订购日期为2018-01-03
按产品id排序后的结果

订购日期为2018-01-04
按产品id排序后的结果

序号分组与排序

当某些数据字段可以看作为序号时可以利用序号分组与排序

orderdate	productid	due
2018-09-18	52	24273
2018-05-23	95	20293
2018-01-26	11	39598
2018-04-27	3	23821
2018-08-26	52	24253
2018-03-05	12	37760
2018-01-30	61	34050
2018-01-10	76	32106
2018-06-22	12	21878
2018-04-01	37	35438

订单文件部分数据

分组举例：将订购日期按月份分组统计订单数

	A
1	=order_ctx_cursor.groupx@n(month(orderdate):month; count(~):count)

month(orderdate)后可以看做序号1~12，可以利用序号进行分组聚合。

month	count
1	84384
2	76262
3	84738
4	81737
5	85150
6	81787

分组后的部分结果

排序举例：将订购日期按月份排序

	A
1	=order_ctx_cursor.groupx@ns(month(orderdate))

month(orderdate)后可以看做序号1~12，可以利用序号进行排序

orderdate	productid	due
2018-01-01	14	22203
2018-01-11	1	38822
2018-01-30	2	36406
2018-01-06	40	39912
2018-01-06	86	34709
2018-01-05	23	24207
2018-01-27	59	29561
2018-01-12	30	35678

排序后的部分结果

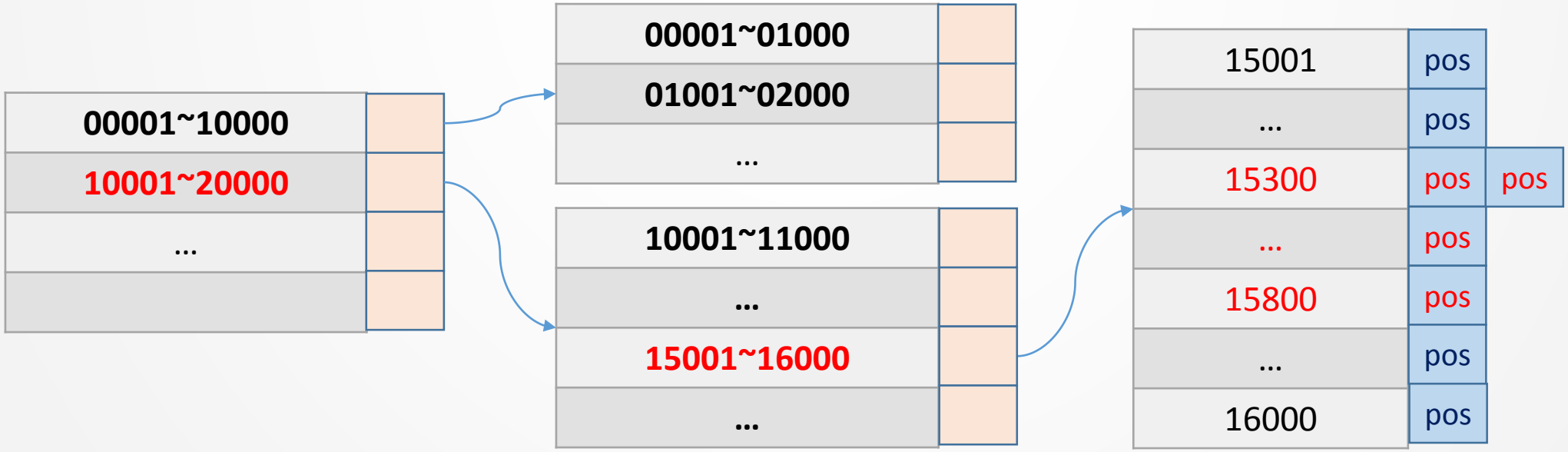
利用索引排序

针对已建有排序索引字段的排序，有时可以利用索引

举例：按消费额排序，统计相应消费额下的订单个数

	A
1	=order_file.create()
2	=A1.icursor(;consume>15300 && consume<15800,consume_idx).groups@o(consume;count(~):count)

本例利用索引排序返回有序游标，利用有序分组快速统计。但如果物理无序，取数量大时性能未必更好



分段排序与分组

当排序字段可以按某规则分组，且每组小到内存可以装下

举例：对订单文件，按用户id进行排序

orderid	userid	orderdate
100000067473	17985431	2018-09-03
100000067474	16063511	2018-09-03
100000067475	16492110	2018-09-03
100000067476	11034981	2018-09-03
100000067477	18328142	2018-09-03
100000067478	17000786	2018-09-03
100000067479	11605948	2018-09-03
100000067480	19937216	2018-09-03
100000067481	15634998	2018-09-03
100000067482	18729157	2018-09-03
100000067483	16680981	2018-09-03
100000067484	12862752	2018-09-03

order_file的部分内容

	A
1	=order_file.cursor@b()
2	=A1.sortx@n(userid;userid\100000)
3	=user_file.export@b(A2)

A2执行后按userid\100000分组，会建立n个临时文件。

```
[2019-06-25 17:28:37]
INFO: Create temporary file: E:\esProc\tmp\tmpdata1165466636928713499

[2019-06-25 17:28:37]
INFO: Create temporary file: E:\esProc\tmp\tmpdata4073295682889221679

[2019-06-25 17:28:37]
INFO: Create temporary file: E:\esProc\tmp\tmpdata1462223521012627634

[2019-06-25 17:28:37]
INFO: Create temporary file: E:\esProc\tmp\tmpdata8416877084849575604
```

执行A2时后台日志

分段排序理解：
userid相同则userid\10000相同，且两者同序。
相比sortx，使用选项@n可以省去最后的归并动作。

类似地，也可以用groupx@n来做大分组。

orderid	userid	orderdate
100373833987	10000001	2018-12-11
101006951386	10000001	2018-12-14
100628626955	10000001	2018-12-18
100146933415	10000001	2018-12-31
100555368239	10000002	2018-01-02
100327959700	10000002	2018-01-02
100082799868	10000002	2018-01-03

排序后的user_file数据

分组维度冗余

舍去分组维度中出现的“冗余项”

举例：根据订单数据和用户数据，按用户编号和用户姓名统计每位用户的订单总额

	A
1	=users.keys(userid)
2	=order_ctx.create().cursor(userid,due).switch(userid,A1)
3	=A2.groups(userid.userid;userid.name,sum(due):amount)

对应SQL为：

```
SELECT u.userid, u.name, SUM(o.due) AS aoumnt
FROM users u
        LEFT JOIN orders o ON u.userid = o.userid
GROUP BY u.userid, u.name
```

从实际情况分析，显然SQL中分组维度u.name多余了，但按SQL的语法要求，即使多余也是无法省略的。

userid	name
1	Brady Brewster
2	Fern Eliot
3	Juan Lowell
4	Jonathan Hoover
5	Judy Alick

A1为用户数据

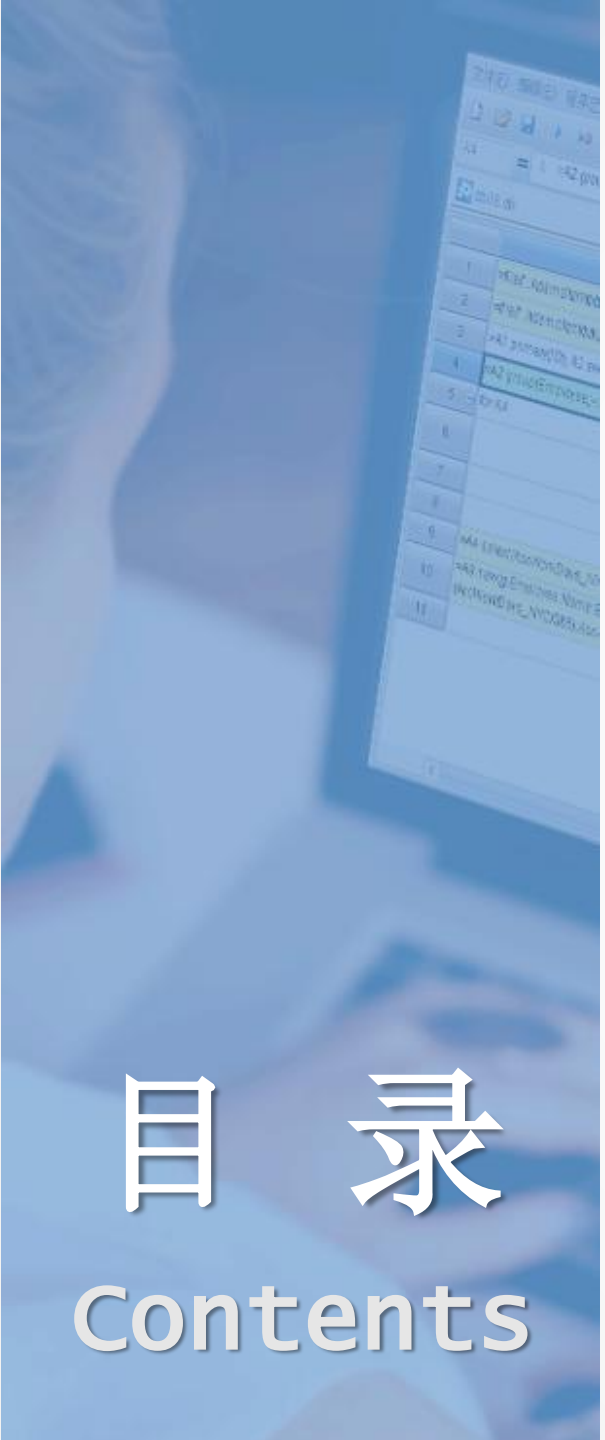
userid	name
79548	Andrew Crofts

userid	due
79548	17458
215588	45352
208430	24904
785633	36733
649860	26436

A2为订单数据
通过userid
与用户数据关联

userid	name	amount
1	Brady Brewster	313130
2	Fern Eliot	478274
3	Juan Lowell	414755
4	Jonathan Hoover	237866
5	Judy Alick	362272

A3按用户id分组
求订单总额
这里姓名跟着id即可

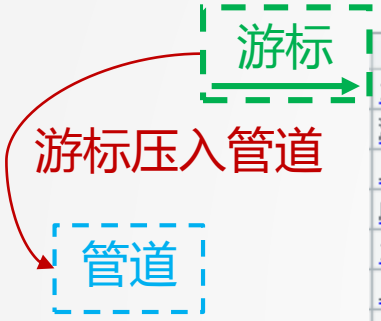


目 录

Contents

遍历复用

计算按城市分组的金额求和与按日期分组的金额求和



city	date	amount
北京	2019-05-15	81
深圳	2019-05-14	18
上海	2019-05-16	2
广州	2019-05-15	37
北京	2019-05-13	49
上海	2019-05-15	55
广州	2019-05-14	32
上海	2019-05-16	50
北京	2019-05-14	55
广州	2019-05-16	60
上海	2019-05-15	55
深圳	2019-05-13	7
深圳	2019-05-16	59
深圳	2019-05-15	19
深圳	2019-05-13	20
深圳	2019-05-15	27
上海	2019-05-16	55
北京	2019-05-14	26

数据文件

同一个游标可以压入多个不同的管道中。
遍历一次游标可计算出多个结果。

	A
1	=file.cursor@b()
2	=channel(A1).groups(city;sum(amount):camount)
3	=A1.groups(date;sum(amount):damount)
4	=A2.result()

date	damount
2019-05-13	1140
2019-05-14	1130
2019-05-15	1261
2019-05-16	1646

city	camount
上海	1292
北京	1458
广州	1187
深圳	1240

游标中定义了按日期分组的金额求和

管道中定义了按城市分组的金额求和

数据分拆

分组后的每组数据，需要独立拆分时，可以使用数据分拆来完成

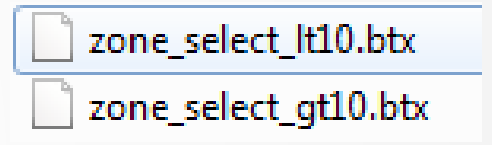
zone	usertag	score
407	grnuup	407
298	daddrg	679
812	upbunb	672
730	dwdany	587
11	bpiedi	680
184	egiwag	455
685	pqpnnq	420
486	byaunr	689
890	npiipi	512
971	npncpd	551

user_file的部分内容

举例：将区域号大于10和小于等于10的数据拆分为两个独立文件

	A
1	=user_file.cursor@b()
2	=file("zone_gt10.btx")
3	=A1.select(zone<=10;A2)
4	=file(“zone_lt10.btx”).export@b(A3)

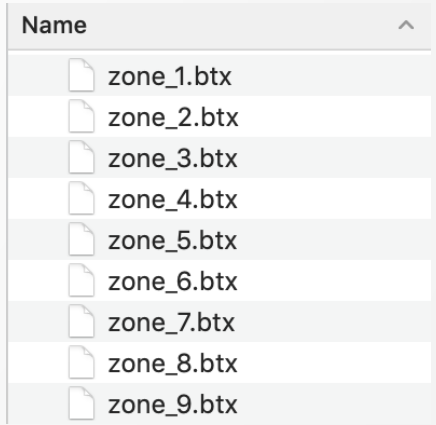
此处导出文件的动作也可以变为向管道推送



按条件拆分后的结果文件

举例：1000个区域分组后的数据独立存储到各文件

	A
1	=user_file.cursor@b()
2	=1000.(file("zone_"+string(~)+".btx"))
3	=A1.groupn(zone;A2)
4	=A1.skip()



导出的部分区域文件

有序游标

用户行为分析，单用户内的运算复杂，但跨用户的运算几乎没有

举例：获取2018年采购咖啡和牛奶数量均大于10的用户订单信息，保存为latte2018.btx

orderid	userid	category	quantity	orderdate
63044682	1	wine	3	2018-02-16
63057527	1	Coffee	6	2018-04-04
63059556	1	Tea	2	2018-04-12
63067460	1	Cola	10	2018-05-11
63075249	1	Cola	5	2018-06-08
63061783	2	Cola	5	2018-04-20
63062280	2	wine	4	2018-04-22
63062678	2	Milk	10	2018-04-23
63064165	2	wine	4	2018-04-29
63064634	2	Coffee	7	2018-04-30
63067961	2	Coffee	6	2018-05-12

某网上超市2018年
历史数据（用户有序）
order2018.btx

	A	B	C
1	=file(“order2018.btx").cursor@b()		
2	for A1;userid	=A2.groups(category;sum(quantity):amount).select(category==" Coffee" || category=="Milk")	
3		=B2.(~.amount>10)	
4		if B3.len()==2 && B3(1) && B3(2)	=file("latte2018.btx").export@ab(A2)

orderid	userid	category	quantity	orderdate
63044682	1	wine	3	2018-02-16
63057527	1	Coffee	6	2018-04-04
63059556	1	Tea	2	2018-04-12
63067460	1	Cola	10	2018-05-11
63075249	1	Cola	5	2018-06-08

A2中每次读取同一用户
购买信息

Index	category	amount
1	Coffee	6

当前用户仅采购了咖啡，
且数量也没大于10
B4为false

有序游标--变化条件

日志分析，以一个特殊字串开头，找到这样的字串就算组的开始

举例：某日志，以---flag---作为事件的起始标识，下一行为用户标签，求单次事件中记录行数最多的用户

Member
---flag---
usertag:rkhmfy
ajqssojpdwhugxmgmg
srtpthodnpemqnjzwcsplnhkkjscfprvsaokps
nbmghigkmewzukqmlmcvqczvjuehyaohlaabgkhotlyher...
bmxtxcovdtijzbigmnzrzvlykekwxpcpxpvosonucebzelpai
ywciadvsjexnxeoj
qmgzrhqvfmtwtzilpnznsmummttbzpd
---flag---
usertag:hommtp
bcgirppghtdvhuib
mulcbpngerzmlxdvlesegkyvvxrwtyi
ldrwcxiveiydguupxwmekkaezuhfdxhfoloh
jeljpnjedsnemycvamjtnmatswxocpsmvwvmfqfyjkfumalf...

日志文件部分数据

	A	B
1	=file("log.txt").cursor@i()	=[,]
2	for A1;~=="---flag---"	=[A2(2),A2.len()-2]
3		>B1=if(B2(2)>B1(2),B2,B1)

Member
---flag---
usertag:rkhmfy
ajqssojpdwhugxmgmg
srtpthodnpemqnjzwcsplnhkkjscfprvsaokps
nbmghigkmewzukqmlmcvqczvjuehyaohlaabgkhotlyher...
bmxtxcovdtijzbigmnzrzvlykekwxpcpxpvosonucebzelpai
ywciadvsjexnxeoj
qmgzrhqvfmtwtzilpnznsmummttbzpd

第一组事件记录

Member
usertag:dufcuf
13

程序执行完毕后，B1为最多行数及其对应的标记。

组内迭代

迭代聚合语法可以一边遍历一边计算，目标集合只需要遍历一次

举例：计算每用户每月截止今日的累计销售额（数据对用户、日期有序）

userid	orderdate	due
10000001	2018-01-02	19
10000001	2018-01-23	10
10000001	2018-01-30	16
10000001	2018-02-10	16
10000001	2018-03-21	14
10000001	2018-03-26	17
10000001	2018-04-03	15
10000001	2018-05-21	12

order_cursor的部分内容

	A
1	=order_cursor.derive(iterate(~~+~.due,0;userid,month(orderdate)):total)
2	=A1.fetch@x(;userid)

其中的iterate(~~+~.due,0;userid,month(orderdate)表达式参数含义：“~~+~.due”为due的累加值，每当userid和月份month(orderdate)变化时，累加变量清零重新进行累加。

userid	orderdate	due	total
10000001	2018-01-02	19	19
10000001	2018-01-23	10	29
10000001	2018-01-30	16	45
10000001	2018-02-10	16	16
10000001	2018-03-21	14	14
10000001	2018-03-26	17	31
10000001	2018-04-03	15	15
10000001	2018-05-21	12	12

A2中的total列为第一个用户id的各月累积值

程序游标

利用程序游标可以避免临时外存文件的读写，从而提升性能

举例：根据2018年采购咖啡和牛奶数量均大于10的用户订单信息，查询该用户当年最后两次购买记录

方法一：使用“有序游标”一节保存的latte2018.btx

	A
1	=latte2018_cursor.groups(userid;top(-2;orderdate):top2)

userid	top2
2	[[63126426,2,wine, ...],[63117504,2,Milk, ...]]
6	[[63131220,6,Coffee, ...],[63127286,6,Milk, ...]]
9	[[63127325,9,Tea, ...],[63124019,9,Tea, ...]]

orderid	userid	category	quantity	orderdate
63126426	2	wine	9	2018-12-12
63117504	2	Milk	5	2018-11-09

方法二：使用程序游标，避免临时中间文件落地

	A	B	C	D
1	func	=file(“order2018.btx").cursor@b()		
2		for B1;userid	=B2.groups(category;sum(quantity):amount).select(category=="Coffee" || category=="Milk")	
3			=C2.(~.amount>10)	
4			if C3.len()==2 && C3(1) && C3(2)	return C2
5	=cursor@c(A1).groups(userid;top(-2;orderdate):top2)			

相当于有序游标一节的计算内容

手工并行

相比使用方便的函数选项，手工并行更为灵活

举例：并行统计各月文件中采购牛奶数量大于3的用户的当月全部订单信息

	A	B	C	D
1	=12.(file("month_"+string(#)+".txt"))			
2	fork to(12)	for A1(A2).cursor@t();userid	=B2.groups(category;sum(quantity):amount).select(category=="Milk")	
3			if C2.len()==1 && C2.amount>3	=file("milk_gt3_"+string(A2)+".txt").export@at(B2)

A2分12个线程，每个线程利用有序游标计算当月采购牛奶数据量大于3的用户，并将该用户当月的全部订单信息导出到对应月份的文件（milk_gt3_月份.txt）中。

	orderid	userid	category	quantity	orderdate
1	63036881	3	Cola	1	2018-01-19
2	63033101	4	Milk	5	2018-01-05
3	63037888	5	Tea 7	2018-01-22	
4	63033796	7	wine	4	2018-01-08
5	63033205	8	Coffee	9	2018-01-06
6	63038443	8	Milk	4	2018-01-24
7	63039666	8	Tea 6	2018-01-29	

month_1.txt

	orderid	userid	category	quantity	orderdate
1	63033101	4	Milk	5	2018-01-05
2	63033205	8	Coffee	9	2018-01-06
3	63038443	8	Milk	4	2018-01-24
4	63039666	8	Tea 6	2018-01-29	
5	63035725	12	Milk	5	2018-01-15
6	63036028	20	Coffee	7	2018-01-16
7	63038625	20	Milk	5	2018-01-25

milk_gt3_1.txt

创新技术

推动应用进步！


各设备故障分析
近7天设备的总故障率
近7天
1.53%
近1天
0.95%
近6小时
2.40%
近7天故障情况
全年故障综合分析
近7天故障次数前5的设备
服务器CPU使用率
今日
昨日
今天硬盘情况