esProc

Innovated data computing engine

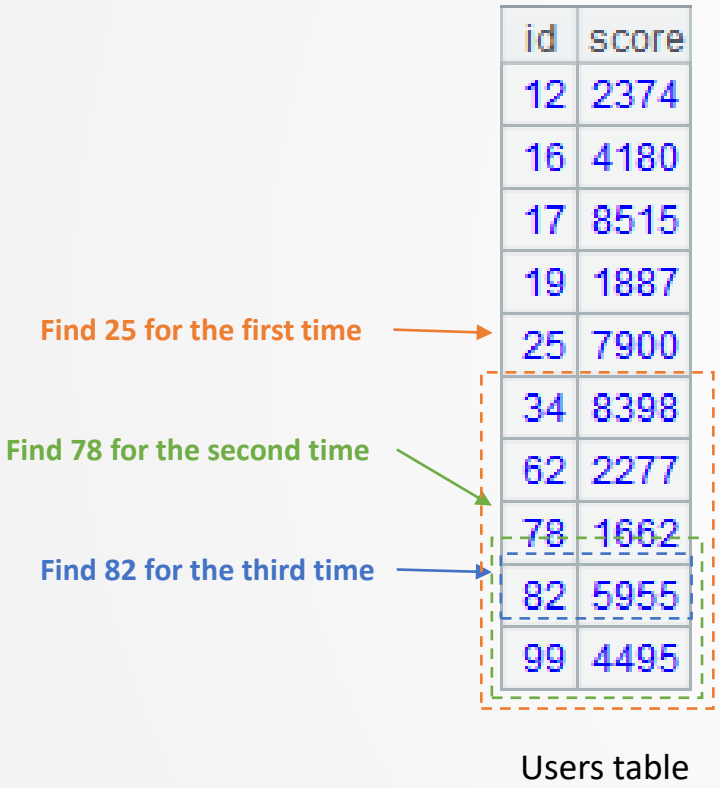
Performance optimization - Search

Issued by Raqsoft

Contents

Search for single-key value--memory--conventional dichotomy

Find user information with id 82



Users table

In this case, sequential lookup (traversal) requires nine comparisons, while dichotomy only uses three comparisons.

The time complexity of sequential lookup is O (n).

The time complexity of dichotomy lookup is O (log_2).

	A	B
1	=users.select@b(82)	/Find a member whose primary key ID equals 82 in the users table

Search for single-key value--memory--Positioning by order number

Find user information with id 9

	A	B
1	=users(9)	/Direct positioning using sequence number

When the key value in the data table itself is the ordinal number, the corresponding record can be found directly by using the ordinal index.

When the key value is close to the ordinal number or easy to convert to the ordinal number, the ordinal number positioning method can also be used.

Index	id	score
1	1	84
2	2	2339
3	3	8512
4	4	3461
5	5	7795
6	6	4423
7	7	6337
8	8	5970
9	9	1498
10	10	10000

Users table with order number

Index	day	price
1	2019-04-25	22
2	2019-04-26	20
3	2019-04-27	24
4	2019-04-28	42
5	2019-04-29	41
6	2019-04-30	3
7	2019-05-01	7
8	2019-05-02	11
9	2019-05-03	22
10	2019-05-04	16

Date day changed to the number of days starting from 2009-04-24

Index	dayid	price
1	1	22
2	2	20
3	3	24
4	4	42
5	5	41
6	6	3
7	7	7
8	8	11
9	9	22
10	10	16

When searching, we can change the date parameter to order number, and then use the method of order number positioning.

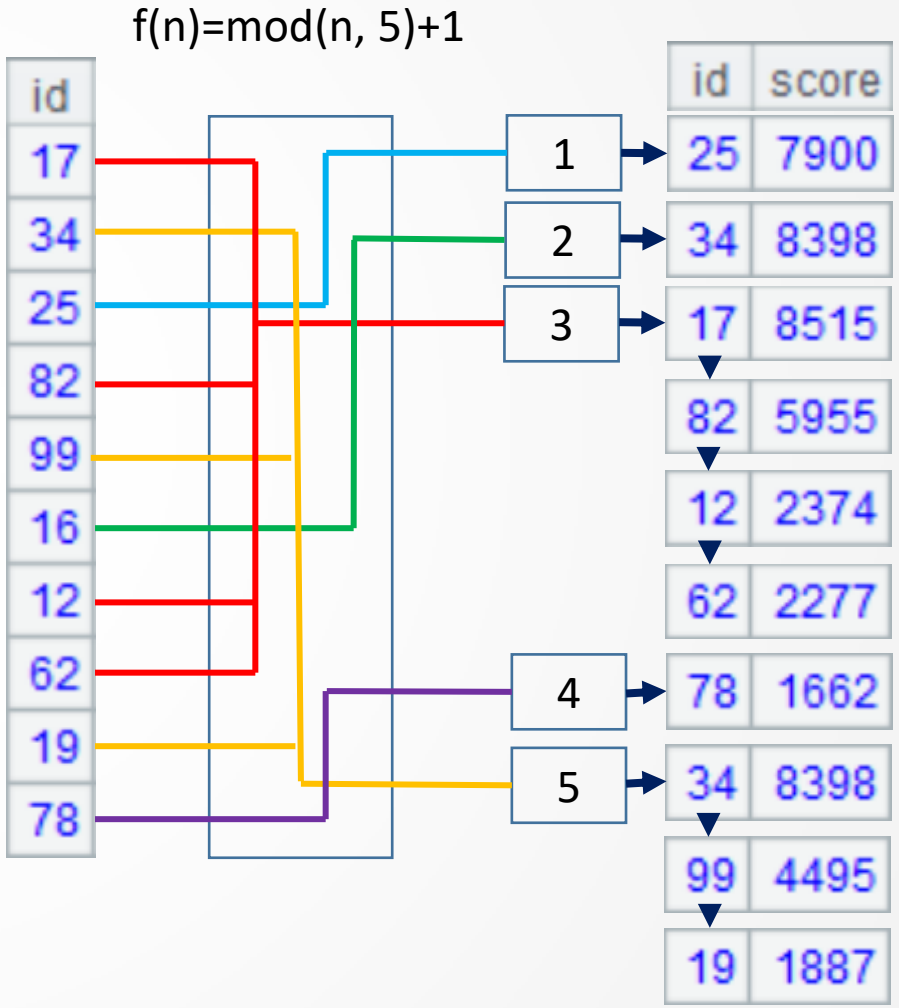
Search for single-key value--memory--Hash index

Find user information with id 82

	A	B
1	=users.keys(id).index()	/Create hash index for ID
2	=users.find(82)	/Computing relative position in users table

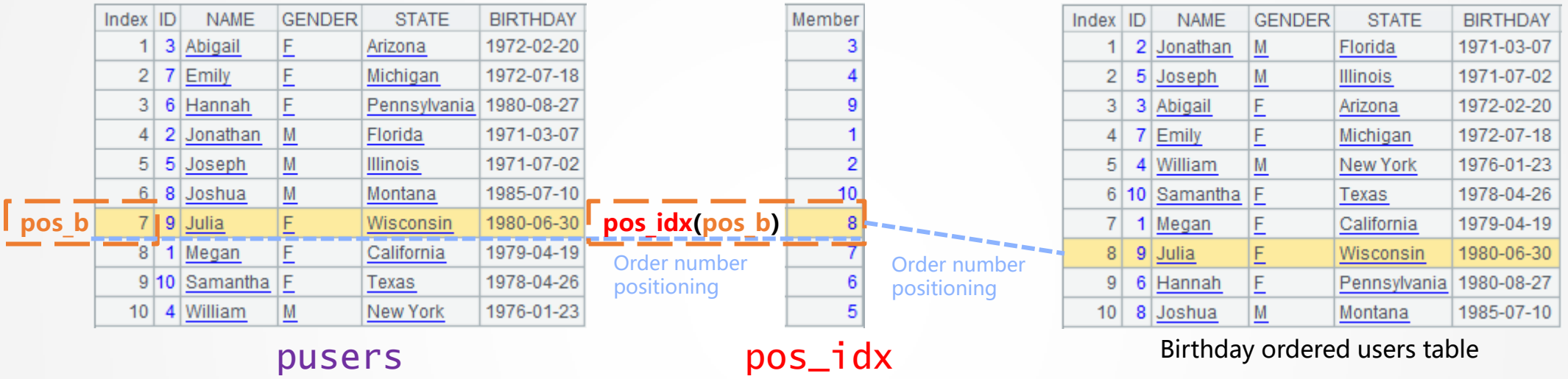
Convert an unordered key ID to a serial number, but it may cause conflicts.

When searching, the hash function can be used to find the corresponding serial number. The conflict part needs to be traversed, but it is faster than the direct traversal.



Search for single-key value--memory-Position index

Find user information with name Julia



pusers

Index	ID	NAME	GENDER	STATE	BIRTHDAY
1	3	Abigail	F	Arizona	1972-02-20
2	7	Emily	F	Michigan	1972-07-18
3	6	Hannah	F	Pennsylvania	1980-08-27
4	2	Jonathan	M	Florida	1971-03-07
5	5	Joseph	M	Illinois	1971-07-02
6	8	Joshua	M	Montana	1985-07-10
7	9	Julia	F	Wisconsin	1980-06-30
8	1	Megan	F	California	1979-04-19
9	10	Samantha	F	Texas	1978-04-26
10	4	William	M	New York	1976-01-23

pos_idx

Member
3
4
9
1
2
10
8
7
6
5

Birthday ordered users table

Index	ID	NAME	GENDER	STATE	BIRTHDAY
1	2	Jonathan	M	Florida	1971-03-07
2	5	Joseph	M	Illinois	1971-07-02
3	3	Abigail	F	Arizona	1972-02-20
4	7	Emily	F	Michigan	1972-07-18
5	4	William	M	New York	1976-01-23
6	10	Samantha	F	Texas	1978-04-26
7	1	Megan	F	California	1979-04-19
8	9	Julia	F	Wisconsin	1980-06-30
9	6	Hannah	F	Pennsylvania	1980-08-27
10	8	Joshua	M	Montana	1985-07-10

		A	B
(One time)	Create index	1	>pos_idx=users.psort(NAME)	/Position index number pos_idx sorted by name
		2	>pusers=users(pos_idx)	/pusers is the index table of users table when names are ordered
(can be reused)	Index lookup	3	>pos_b=pusers.pselect@b(NAME:"Julia")	/Searching Julia's ordinal number in index table by dichotomy
		4	=users(pos_idx(pos_b))	/The pos_b number value of the index number is the ordinal position of Julia in the original table.

Search for single-key value--memory--Multilayer sequence number positioning

Find ID number

id	name	...
10000000000000001		
.....		
110105197306098l6	Zhangsan	
.....		
5421101984081202 3	Lisi	
.....		
81072219630218415	Wangwu	
.....		
90999920601231999		

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

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

Divide 17-bit ID number into 8 layers： 11 | 01 | 05 | 1973 | 06 | 09 | 81 | 6

Bits 1 and 2: Numbers are discontinuous. For those numbers that do not exist, the corresponding lower nodes are empty.

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

Bits 3 and 4 : 1~99

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

Bits 5 and 6 : 1~99

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

Bits 7,8,9,10： represent birthday year, 1970 is the benchmark here.

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

For the rest 7-bit, please think it over.

Search for single-key value--External storage--conventional dichotomy

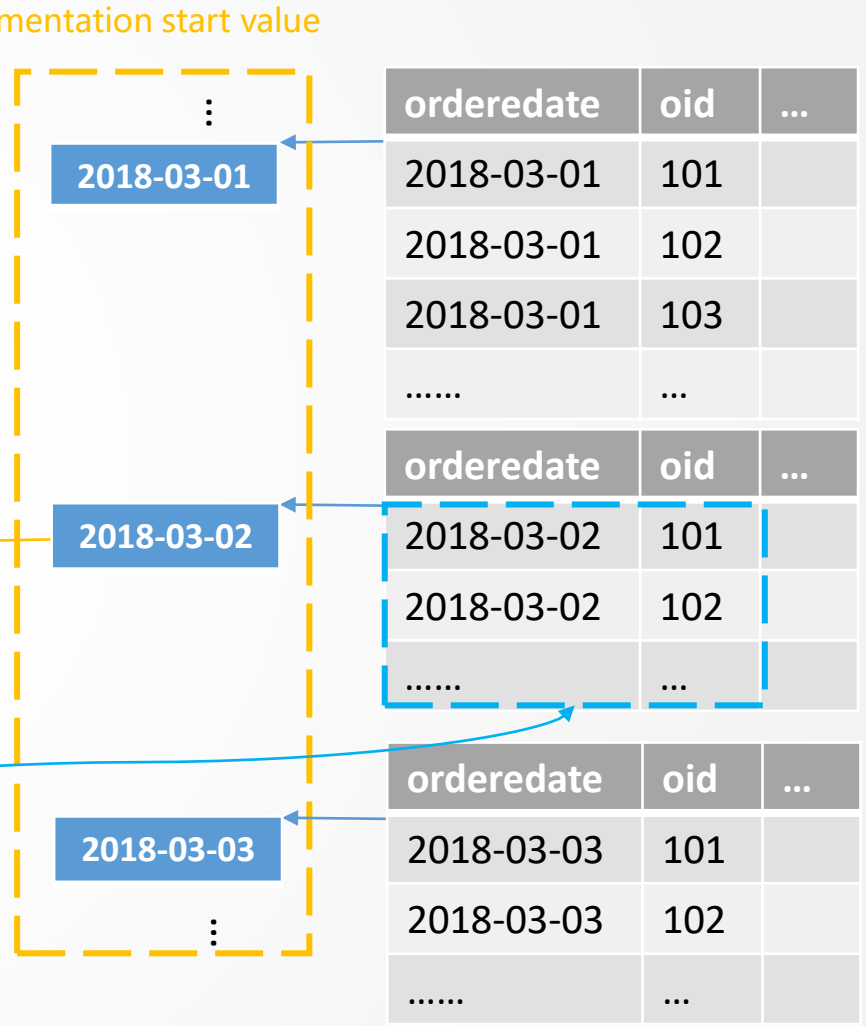
Find order information for 2018-03-02

	A	B
1	=orders_file.iselect@b(date("2018-03-02"),orderdate).fetch()	/order date ordered, use dichotomy

External storage data can not be accurately located to find a particular item. There may be many records in a block, and they are recorded in the starting value of each block.

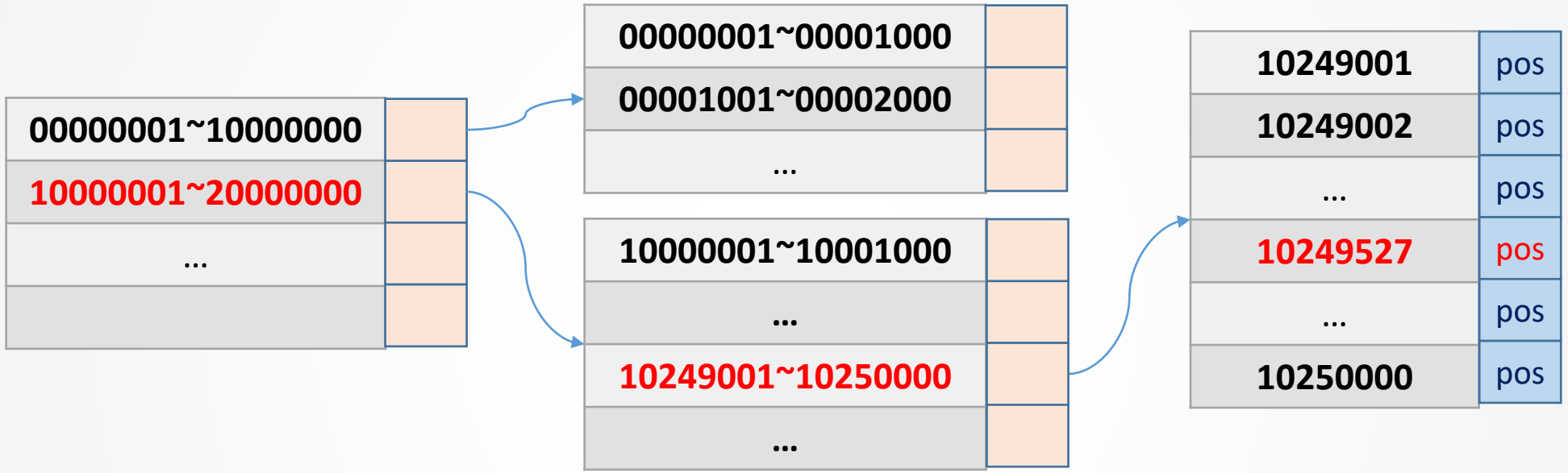
Dichotomy search of segment start value shows whether the key to be searched is in this block or not.

The data in the block is read in and then searched by memory dichotomy.



Search for single-key value--External storage--Sort Index

Find User with id 10249527



Create index

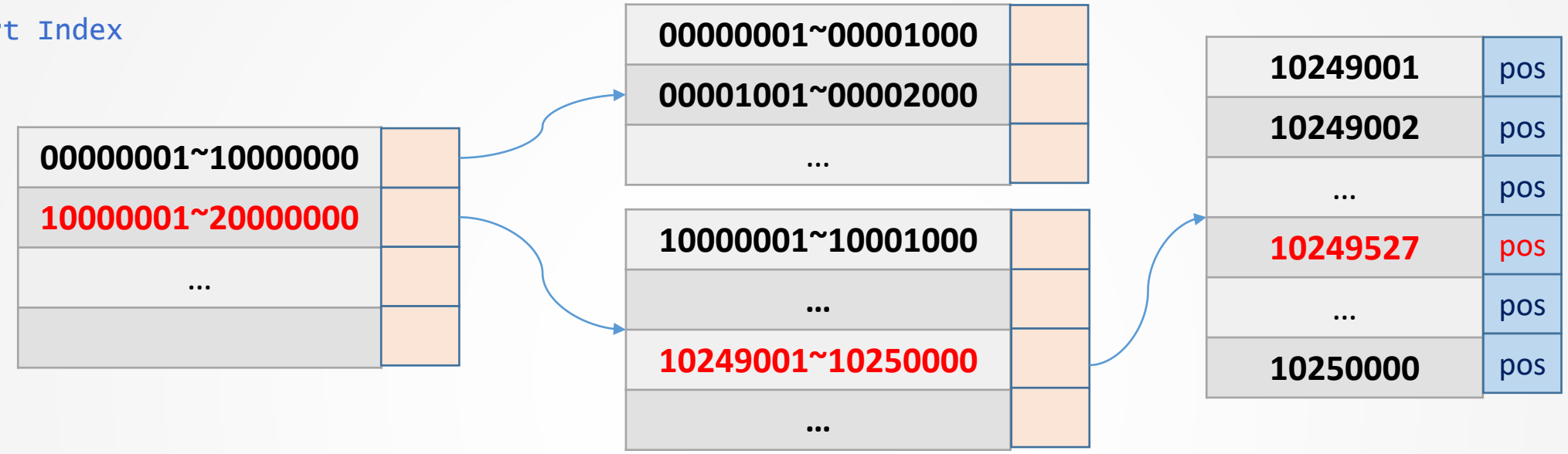
	A	B
1	=user_file.create()	/Open the file
2	=A1.index(id_idx,id)	/Create sort index for ID

Search

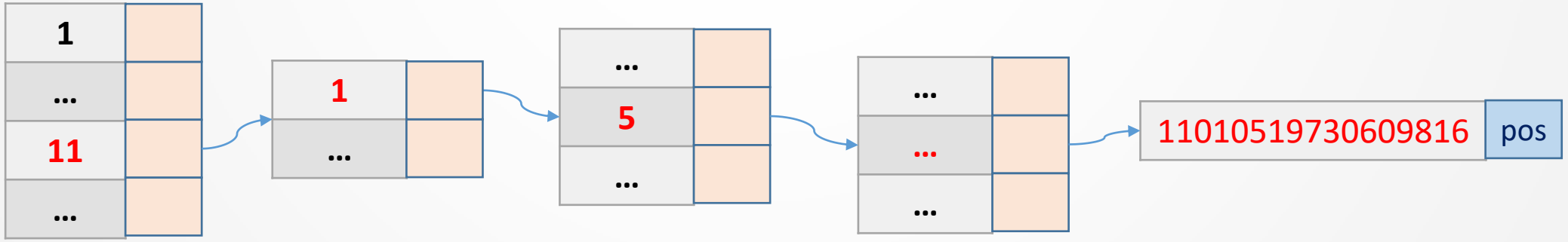
	A	B
1	=user_file.create()	/Open the file
2	=A1.icursor(;id==10249527,id_idx)	/User with id 10249527

Search for single-key value——Analogy of Memory and External Storage Search Technology

Sort Index



Multilayer sequence number positioning



Search for single-key value--External storage-- Hash index

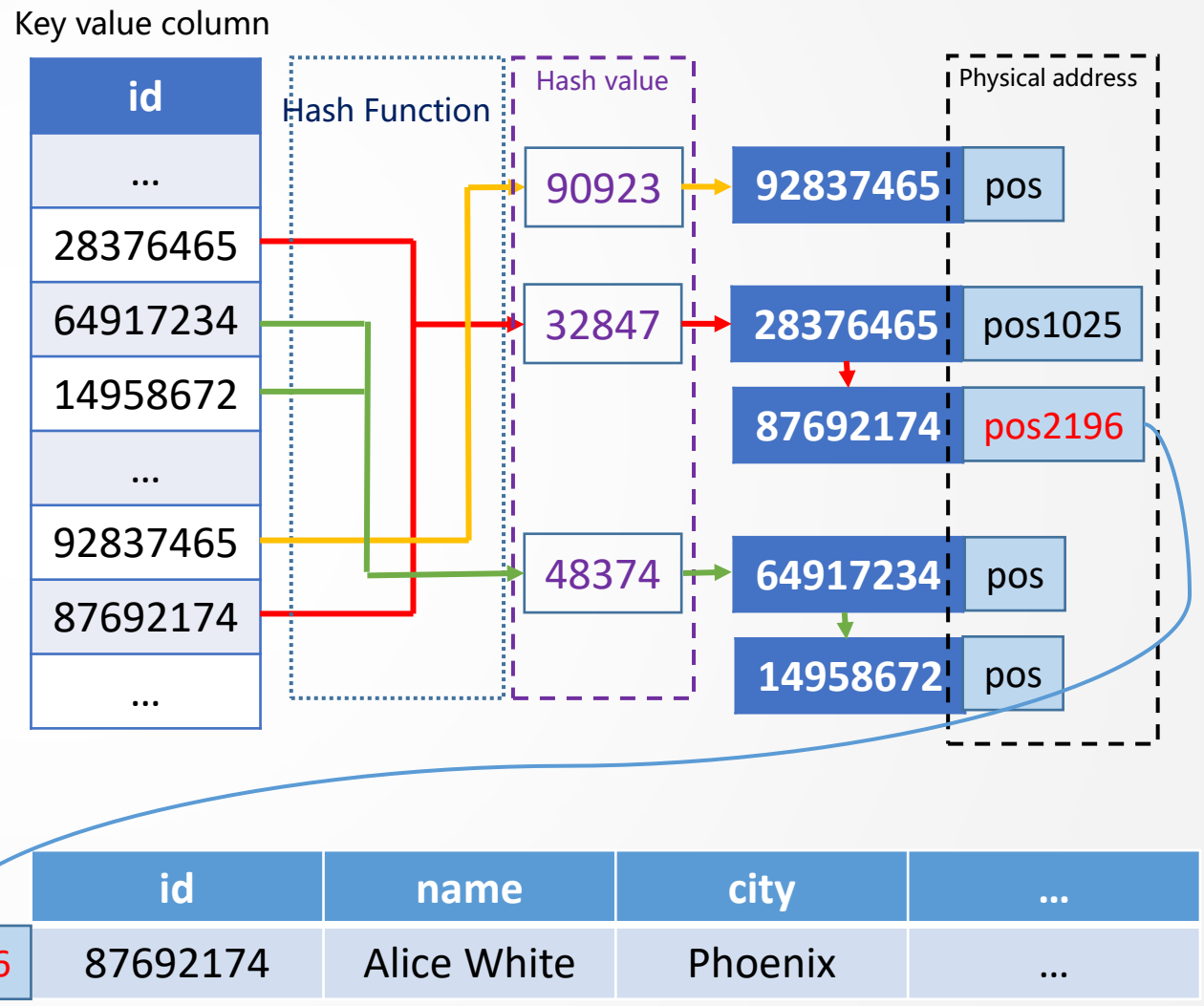
Find user with Id number 10249527

	A	B
1	=user_file.create()	/Open the file
2	=A1.index(id_idx:hash_den,id)	/Create hash index for id

	A	B
1	=user_file.create()	/Open the file
2	=A1.icursor(;id==10249527,id_idx)	/Search for user with id number 10249527

Hash index is suitable for searching single key value, but because hash function is not monotonous, it is not suitable for searching intervals.

In the first two layers, the sort index can quickly determine the range through the maximum and minimum value of the interval, which is suitable for interval search.



Contents

Search for Multi-key values--key value sorting

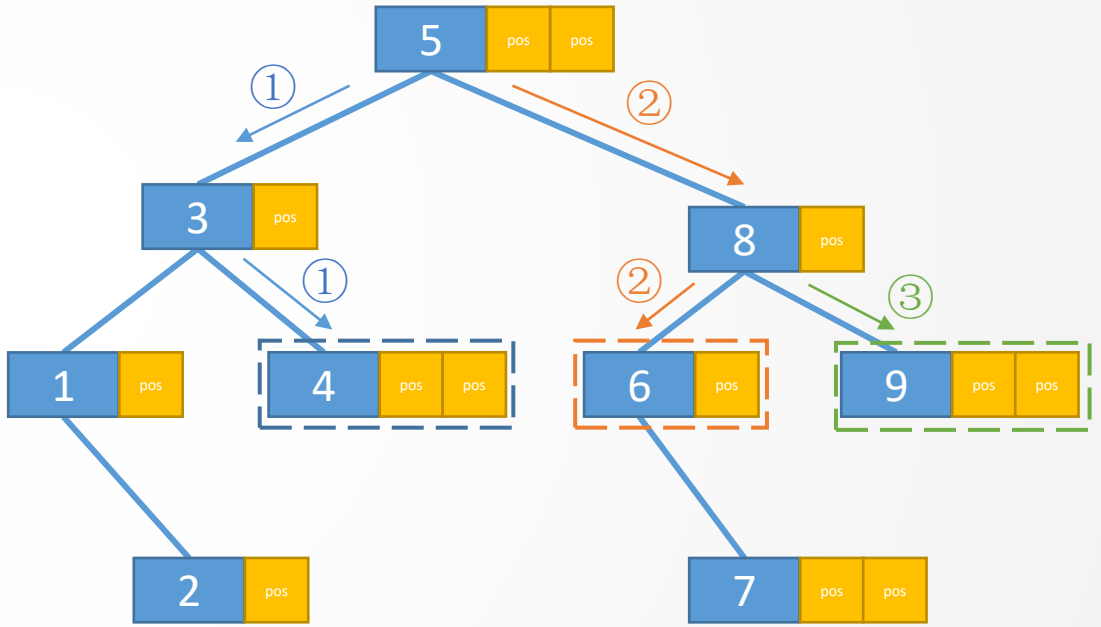
Find all user information with scores of 4, 6, 9

When there is no index, the original table and the set of key values to be searched are ordered

	A
1	=user_file.iselect([4,6,10],score)

When there is a sort index, when the set of key values to be searched is ordered

	A
1	=user_file.create()
2	=A1.icursor(;[4,6,9].contain(score),score_idx)



Example of Binary Trees for dichotomy search

Search for Multi-key values--Row-wise storage

For scenarios where the entire record needs to be taken out,
row storage is more appropriate than column storage

A3

Index	id	data
1	1	vf5mous8qnwc3bp24y6tz79ax0ihd1jrlkge
2	2	cvsofpehx65wqzm3bk02dty4j9r7inl1g8ua
3	3	iexocb0kdwts9fqj1p8h5nmurz43gav72ly6
4	4	wcx96mpiur4sf1vaqe8zodhb5n02ykjtg7l3
5	5	ewma6zngo4chr52uyjp1bfsq3t8lv9i7xdk0
6	6	do56m9bin8xa1c30hgy7qtusrz2w4fjlvekp
7	7	ieqna69bcthoxgd108k3flpw2rjmvzy4u57s
8	8	ocvh2ek0ptfzqx14n57aid68lmyujgr3b9sw
9	9	cgsiub74nje185qv3hrao2kwmylz60px9dft
10	10	xnliom6zbesrg7k8yf39512duqjpt4cvwh0a

Column Storage

A3

Index	id	data
1	1	vf5mous8qnwc3bp24y6tz79ax0ihd1jrlkge
2	2	cvsofpehx65wqzm3bk02dty4j9r7inl1g8ua
3	3	iexocb0kdwts9fqj1p8h5nmurz43gav72ly6
4	4	wcx96mpiur4sf1vaqe8zodhb5n02ykjtg7l3
5	5	ewma6zngo4chr52uyjp1bfsq3t8lv9i7xdk0
6	6	do56m9bin8xa1c30hgy7qtusrz2w4fjlvekp
7	7	ieqna69bcthoxgd108k3flpw2rjmvzy4u57s
8	8	ocvh2ek0ptfzqx14n57aid68lmyujgr3b9sw
9	9	cgsiub74nje185qv3hrao2kwmylz60px9dft
10	10	xnliom6zbesrg7k8yf39512duqjpt4cvwh0a

Row storage

Search for Multi-key values——Valued indexes

Establishing a value-added index can include commonly used data columns

Establishing Valued Index

	A	B
1	=file("id_data.ctx").create()	/Open the file
2	=A2.index(id_idx;id;data)	/Create valued index

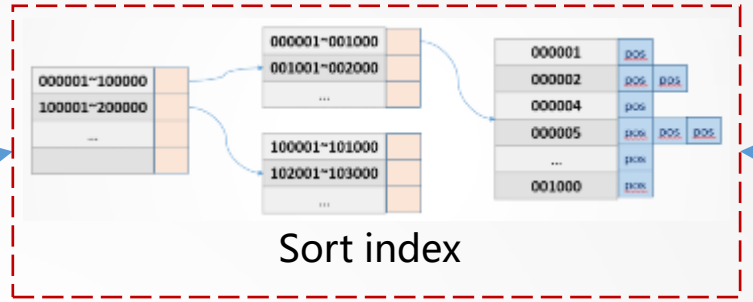
Using Valued Index to search

	A	B
1	=file("id_data.ctx").create()	/Open the file
2	=A1.icursor(id==7,id_idx).fetch()	/Using Valued Index to search

id	data
1	vf5mous8qnwc3bp24y6tz79ax0ihd1jrlkge
2	cvsofpehx65wqzm3bk02dty4j9r7inl1g8ua
3	iexocb0kdwts9fqj1p8h5nmurz43gav72ly6
4	wcx96mpiur4sf1vaqe8zodhb5n02ykjtg7l3
5	ewma6zngo4chr52uyjp1bfsq3t8lv9i7xdk0
6	[illegible]
7	ieqna69bcthoxgd108k3flpw2rjmvzy4u57s
8	[illegible]
9	cgsiub74nje185qv3hrao2kwmylz60px9dft
10	xnliom6zbesrg7k8yf39512duqjpt4cvwh0a

Using Valued Index Files
No more access to the original column storage table

Search for Multi-key values——Index Caches



Sort index

Random key-value queries directly using indexes:

First query， time :80 seconds

Second query， time:78 seconds

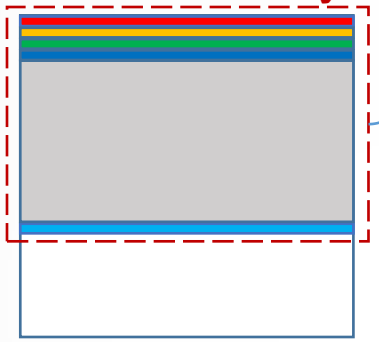
Third query， time:77 seconds

Fourth query， time:76 seconds

.....

Nth query, time:25 seconds

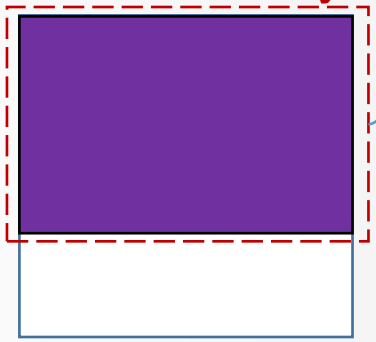
Cache Memory



Preload index cache:

Each random key query takes about 25 seconds

Cache Memory



Each time an index is used for key value queries, the operating system generates a cache.

After N times of using index queries, the efficiency will reach the limit.

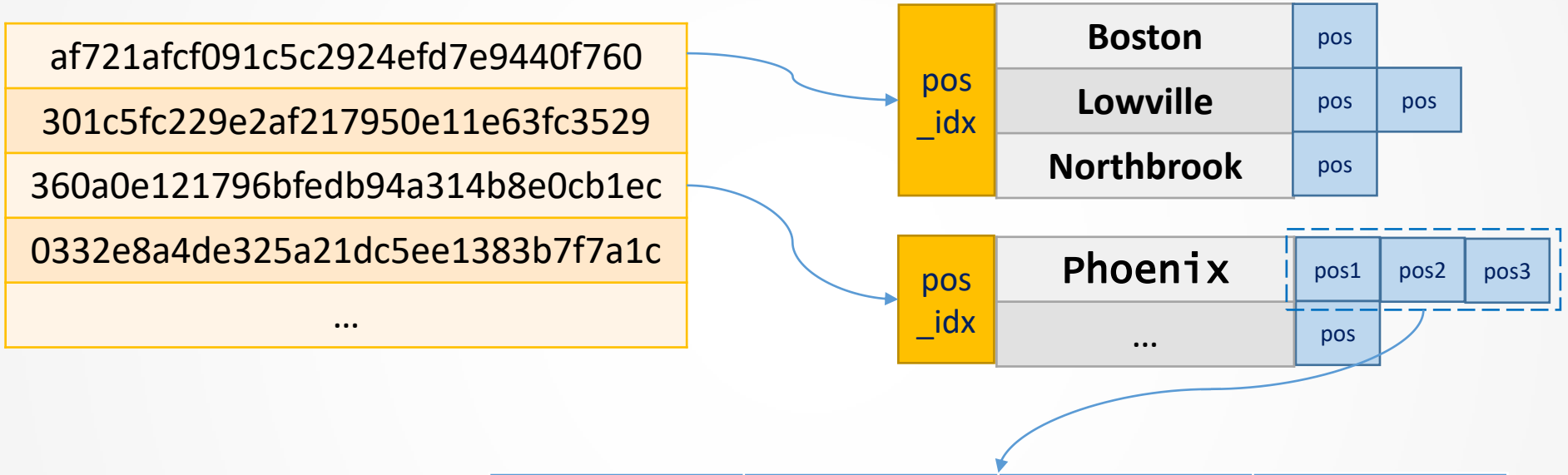
Index caches can be pre-loaded so that each query is in the most efficient state.

	A	B
1	=file(“id_data.ctx”).create().index@3(id_idx)	/Load three-level index cache

Contents

Result Set Search - Conventional Index

Find Phoenix users



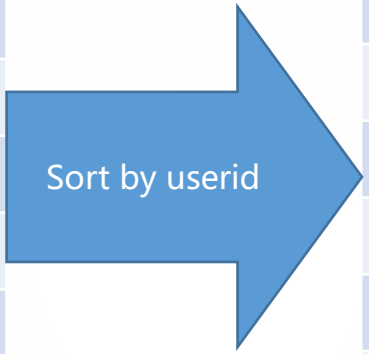
	id	name	city	...
pos1	27	Alice White	Phoenix	...
pos2	1025	John Smith	Phoenix	...
pos3	9527	Sandra Dee	Phoenix	...

Index supports returning multiple records, which are sorted by physical location in the index.

Result Set Search-Physically Ordered

Find orders with userid 1001

sid	userid	orderdate	...
……	……	……	
10000001048	1002	2018-03-07	
...			
10000001237	1001	2018-03-07	
...			
10000800052	1001	2018-05-13	
...			
11000000053	1004	2018-09-20	
...			
80000000054	1001	2019-06-17	
……	……	……	



userid	orderdate	……
……	……	
1001	2018-03-07	
1001	2018-05-13	
1001	2019-06-02	
1002	2018-03-07	
……	……	

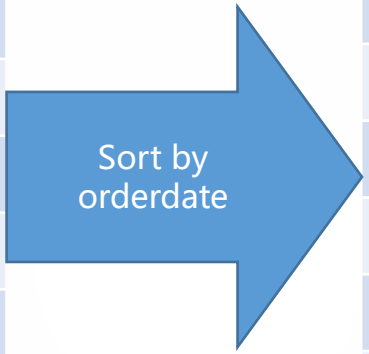
When userid is physically ordered, there is no need to index

	A	B
1	=user_file.cursor(;userid==1001)	/Query

Result Set Search-Physically Ordered

Find orders dated March 7, 2018

sid	userid	orderdate	...
……	……	……	
10000001048	1001	2018-03-07	
...			
10000001237	1002	2018-03-07	
...			
10000800052	1001	2018-05-13	
...			
11000000053	1004	2018-09-20	
...			
80000000054	1001	2019-06-17	
……	……	……	



orderdate	userid	……
……	……	
2018-03-07	1001	
2018-03-07	1002	
……	……	
2018-05-13	1001	
……	……	

When orderdate is physically ordered, there is no need to index

	A	B
1	=user_file.cursor(;orderdate==date("2018-03-07"))	/Query

Result Set Search-Physically Ordered

Find out orders whose order date is 2018-03-07 and the city is Beijing.

...	city	orderdate	...
	.....	.....	
	beijing	2018-03-07	
	...		
	shanghai	2018-03-07	
	...		
	beijing	2018-03-07	
	...		
	beijing	2018-09-20	
	...		
	guangzhou	2019-06-17	
	.....	.....	

Sort by date and city respectively

orderdate	city	...
.....	.....	
2018-03-07	beijing	
2018-03-07	beijing	
2018-03-07	shanghai	
.....	.....	
2018-09-20	beijing	
.....	.....	

Sort by city and date respectively

city	orderdate	...
.....	.....	
beijing	2018-03-07	
.....	2018-03-07	
beijing	2018-09-20	
.....	.....	
shanghai	2018-03-07	
.....	.....	

When searching for a city, you can sort the file again redundantly.

	A
1	=users_DC_file.create().cursor(;orderdate==date("2018-03-07") && city=="beijing")

	A
1	=users_CD_file.create().cursor(;city=="beijing")

Result Set Search——Data Update

Update of modified file data

date	price	...
...		
2019-04-23	50	
2019-04-24	50	
2019-04-25	50	
2019-04-26	50	
2019-04-27	50	

date	price	...
2019-04-24	51	
2019-04-26	49	

Update data(Supplementary Area)

date	price	...
...		
2019-04-23	50	
2019-04-24	51	
2019-04-25	50	
2019-04-26	49	
2019-04-27	50	

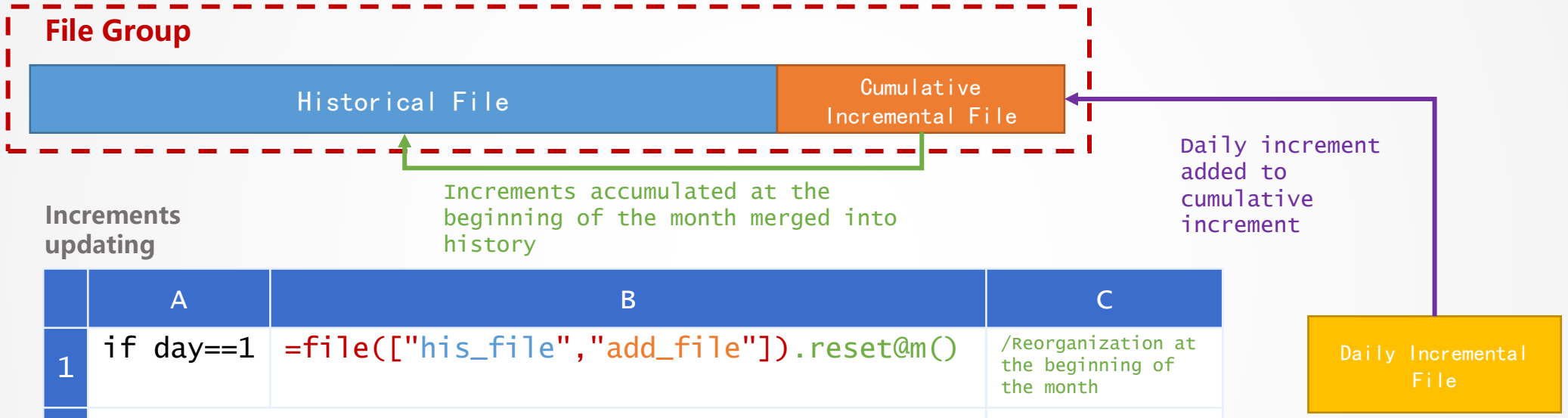
Rapid reorganization means that only the part after the first complement data appears is reorganized. Previous data need not be rewritten.

When recent cumulative incremental data change

	A	B
1	=add_file.create().update(update_data)	/Update file data
2	=add_file.reset@q()	/Rapid Reorganization of Supplementary Area Data

Result Set Search--Data Update

Update of Daily Data Files



	A	B	C
1	if day==1	=file(["his_file","add_file"]).reset@m()	/Reorganization at the beginning of the month
2	=file("add_file").create().append@m(add_data)		/Daily supplement

File group query

	A	B
1	=file(["his_file","add_file"])	/Create file group
2	=A1.create().cursor(;id=="3197608180")	/Query

Contents

Multi-condition Search-Interval Search

Query users whose birth date is before January 1, 1985

File is ordered by birth date, no index is needed.
In case of disorder, sort index can be established and then carry out interval search.

birthday	userid	city	...
.....	.....	.....	.....
1984-12-30	50103784	shanghai	.....
1984-12-31	92876392	beijing	.....
1984-12-31	12495255	guangzhou	.....
1984-12-31	20973177	beijing	.....
1985-01-01	31237438	shanghai	.....
.....	.....	.....	.....
1990-09-20	10928313	beijing	
.....	.....	.....	.....

	A	B
1	=users_file.create().cursor(;birthday<date("1985-01-01"))	/Ordered by birthday. Direct search without indexing.

Multi-Condition Search--Multi-Field Index--Joint Index

Query users whose birth date is before January 1, 1985 and whose city is Beijing

Establish a joint index of birthdays and cities

	A	B
1	=users_file.create().index(B_C_idx;birthday,city)	/Joint index of birthdays and cities

Query with joint index

	A	B
1	=users_file.create().icursor(;birthday<date("1985-01-01") && city=="beijing"))	/For every birthday, the city is orderly and the joint index takes effect.

Use this joint index to find users whose city is Beijing

	A	B
1	=users_file.create().icursor(;city=="beijing")	/Index is not entirely orderly to city, index is invalid

birthday	city
.....	.....
1984-12-30	shanghai
1984-12-31	beijing
1984-12-31	beijing
1984-12-31	shenzhen
1985-01-01	shanghai
.....	.....
1990-09-20	beijing
.....	.....

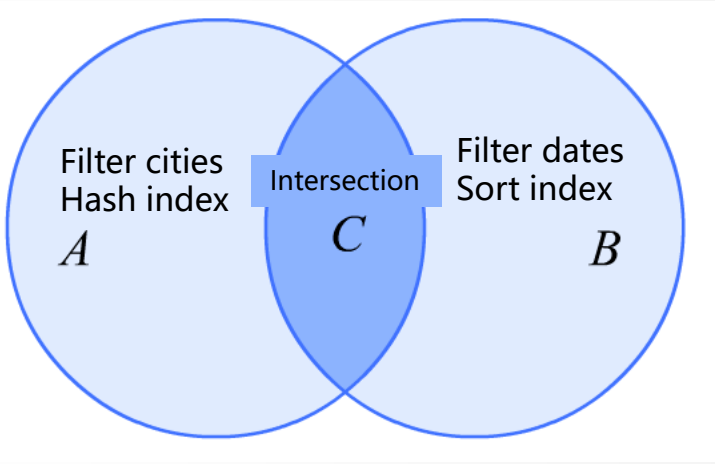
Joint index
Birthday is ordered, city is ordered under the same date

Multi-Condition Search--Multi-Field Index--Individual Index

Query users whose birth date is before January 1, 1985 and whose city is Beijing

Establish index for cities and index for birthdays separately

	A	B
1	=users_file.create()	/Open the file
2	=A1.index(city_idx:1;city)	/Create Hash index for cities
3	=A1.index(bday_idx;birthday)	/Create sort index for birthdays



Intersection of users whose city is Beijing and users whose birthday is before January 1, 1985

	A	B
1	=users_file.create()	/Open the file
2	=A1.icursor(;city=="beijing" && birthday<date("1985-01-01"))	/Select users whose city is Beijing, and users whose birthday is before January 1,1985, do the intersection of the two.

Multi-Condition Search——Order of conditions

Find users whose gender is female and whose birth date is before January 01, 1990

id	……	gender	birthday	……
……		……	……	
100048		F	1984-01-11	
100049		M	1974-03-26	
100050		F	1989-12-15	
100051		M	1997-06-07	
100052		M	1983-09-13	
100053		M	1984-11-21	
100054		F	1988-08-02	
……		……	……	

	A	B
1	=users.select(gender=="F" && birthday<date(1990-01-01))	/Filter gender first, then birthday
2	=users.select(birthday<date(1990-01-01) && gender=="F")	/Filter birthday first, then gender

The conditions of the first row is:
gender== "F" && birthday<date(1990-01-01),
The front sub-item gender returns a smaller result set and takes 683 milliseconds to query.

The conditions of the second row is:
birthday<date(1990-01-01) && gender== "F" ,
The front sub-item birthday returns a larger result set and takes 1796 milliseconds to query.

Female users are fewer, but birthdays are generally before1990.

Multi-Condition Search——Full Text Index

like("X*") type, sort index can be used

like("Zhou Xing*")

like("*X*") type, Full-text index need to be used

	A	B
	=file("users").create().index@w(name_idx;name)	/Full-text indexing of names
1	=users_file.create().icursor(like(name,"*Yu*")).fetch()	/Use full-text index
2	=users_file.cursor().select(like(company,"*Yu*")).fetch()	/Use ordinary cursor to query records with the same condition

Full-text indexing that supports like function is much faster than sequential lookup.

Key word	Name
...	...,...,...
Zhou	Zhou Yu, Zhou Yu Liang, Zhou Zong
Yu	Zhou Yu, Wang Yu Liang, Zhou Yu Liang
Zong	Jiang Zong, Zhou Zong, Gao Zong
Yu Liang	Wang Yu Liang, Zhou Yu Liang
...	...,...

Full text search

Innovation makes progress!