StarRocks | DataFun.

StarRocks的实时更新

常冰琳

Outline

- Real-time update use cases
- Common approaches
- Updates in StarRocks
- Ongoing & future works

01

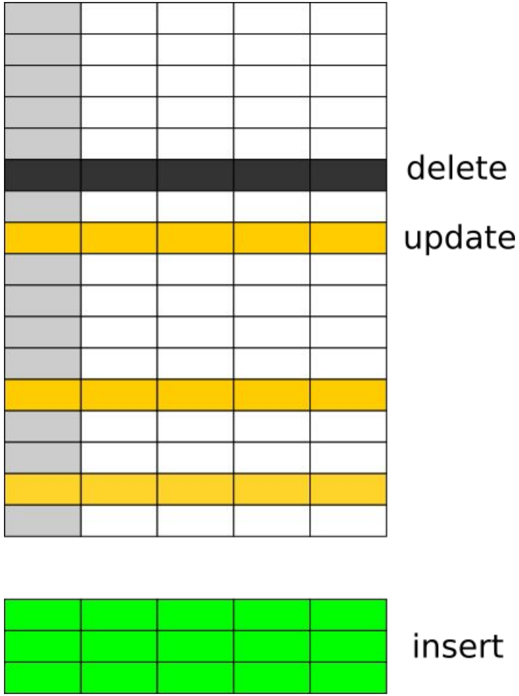
实时更新需求

Why?

- Traditional OLAP
  - T+1 batch ETL, high latency
  - Incremental append only, no update
  - Append update & merge-on-read, poor query performance
- New requirement in real-time analytics
  - realtime data ~ hot data ~ volatile data
  - TP -> AP sync pipeline
- In database ELT

Use Case: full row upsert/delete

- Full row upsert(or delete)
  - most common form
  - MySQL
    - `insert into on duplicate key update`
  - StarRocks
    - unique key load (upsert)
    - primary key load (upsert/delete)
  - TP -> AP CDC sync

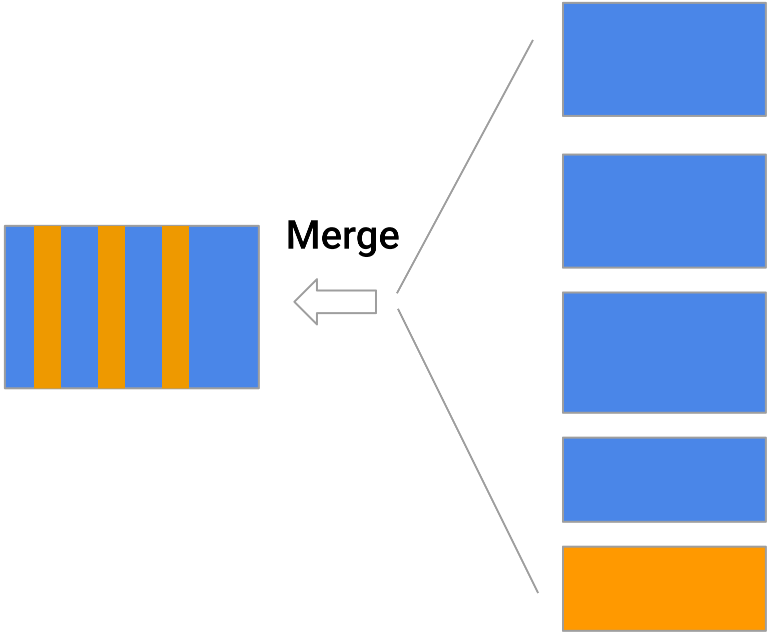


Merge-on-Read

Fast write, slow read

Examples:

- Various LSM Trees
- Hudi Merge-on-Read Table
- StarRocks Unique Key

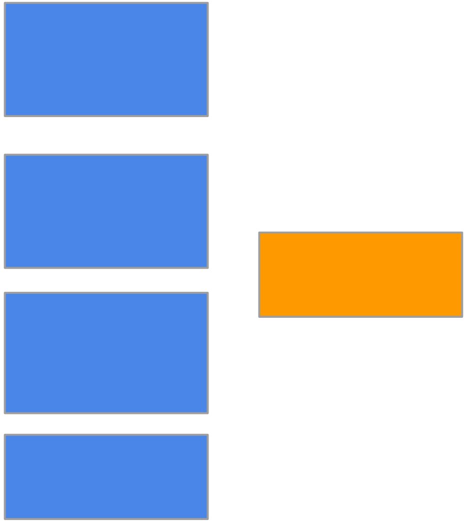


Copy-on-Write

Slow write, Fast read

Example:

- Delta Lake
- Hudi Copy-on-Write
- Iceberg
- Snowflake

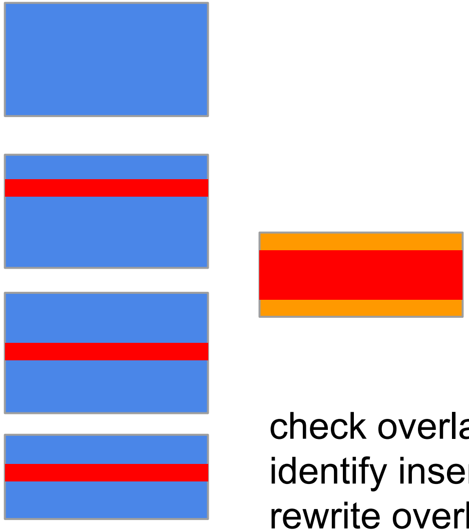


Copy-on-Write

Slow write, Fast read

Example:

- Delta Lake
- Hudi Copy-on-Write
- Iceberg
- Snowflake



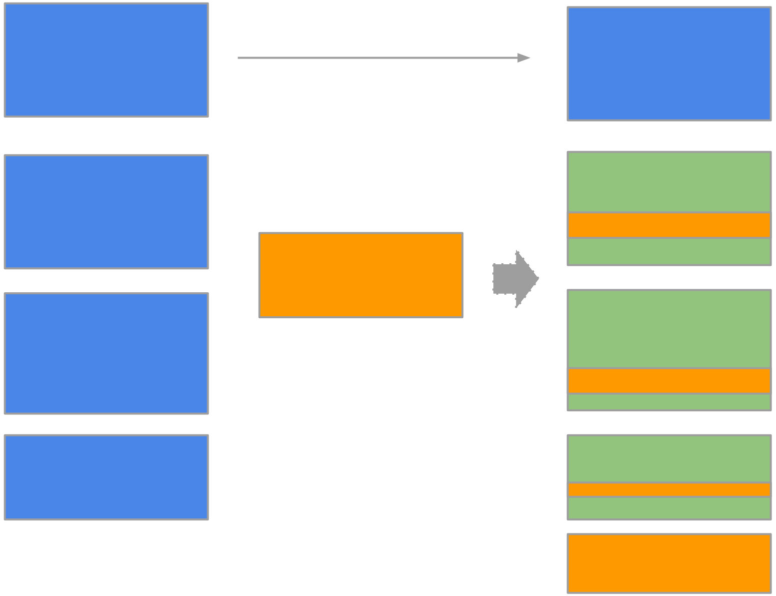
check overlapping files
identify insert/update
rewrite overlapping files

Copy-on-Write

Slow write, Fast read

Example:

- Delta Lake
- Hudi Copy-on-Write
- Iceberg
- Snowflake

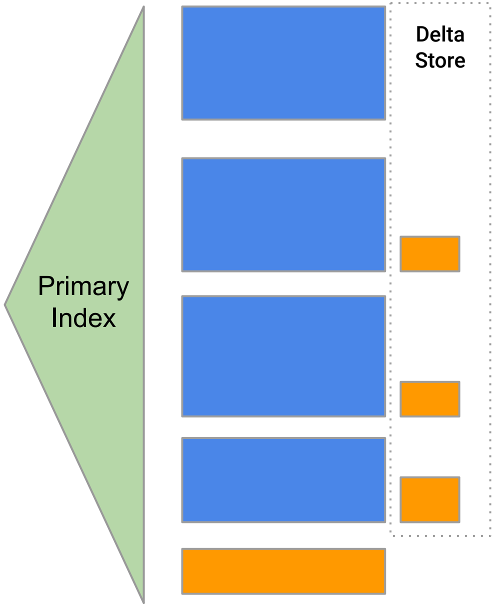


Delta Store

Slow(a bit) write, Fast read

Example:

- Kudu
- Many TP/HTAP Databases

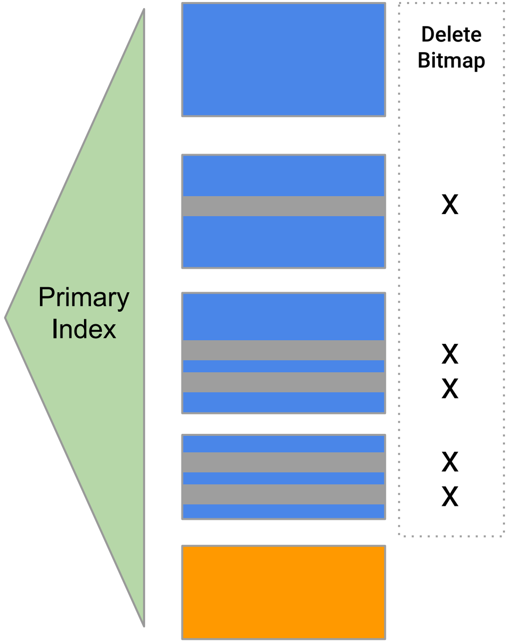


Delete+Insert

Slow(a bit) write, Fast read

Example:

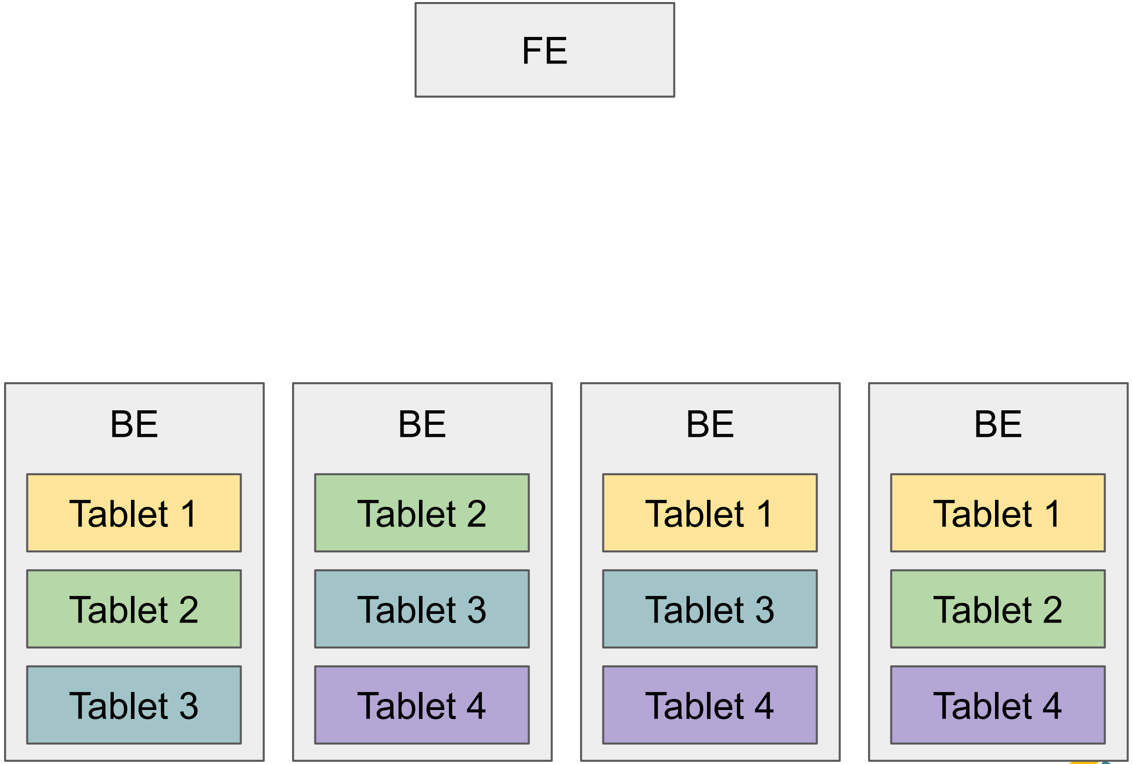
- SQL Server column store
- Alibaba ADB, Hologres
- StarRocks primary key table



02

StarRocks的实时更新

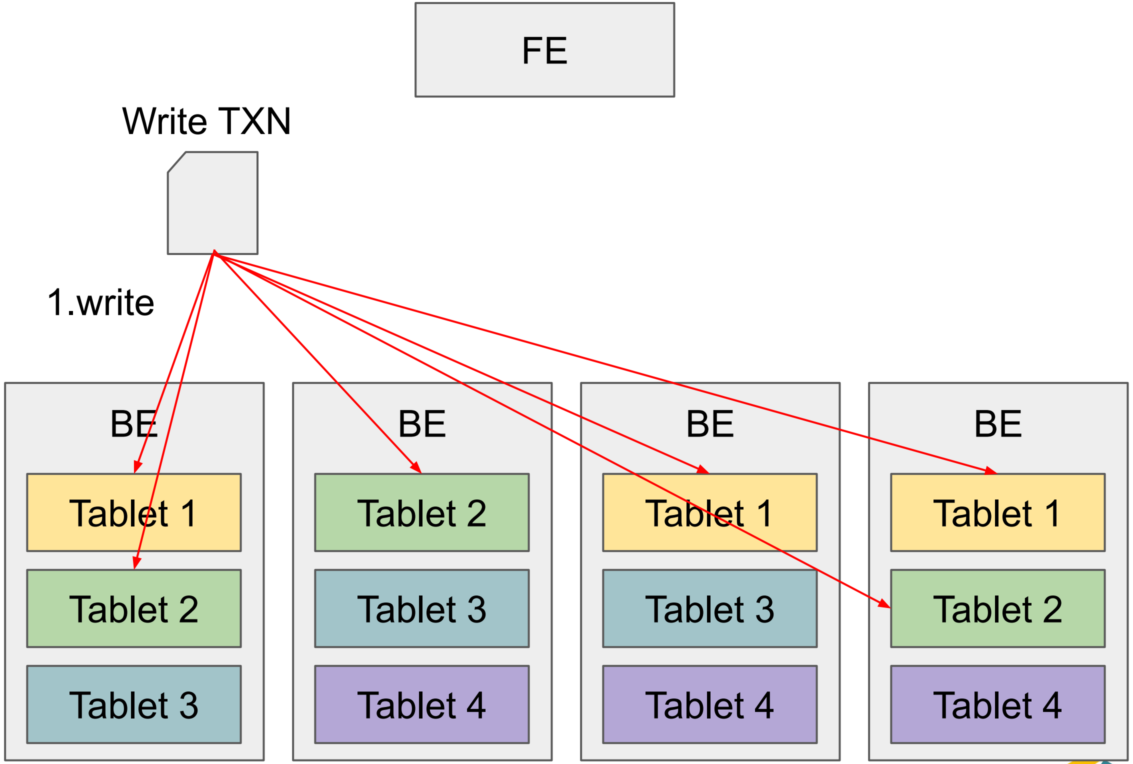
System Overview



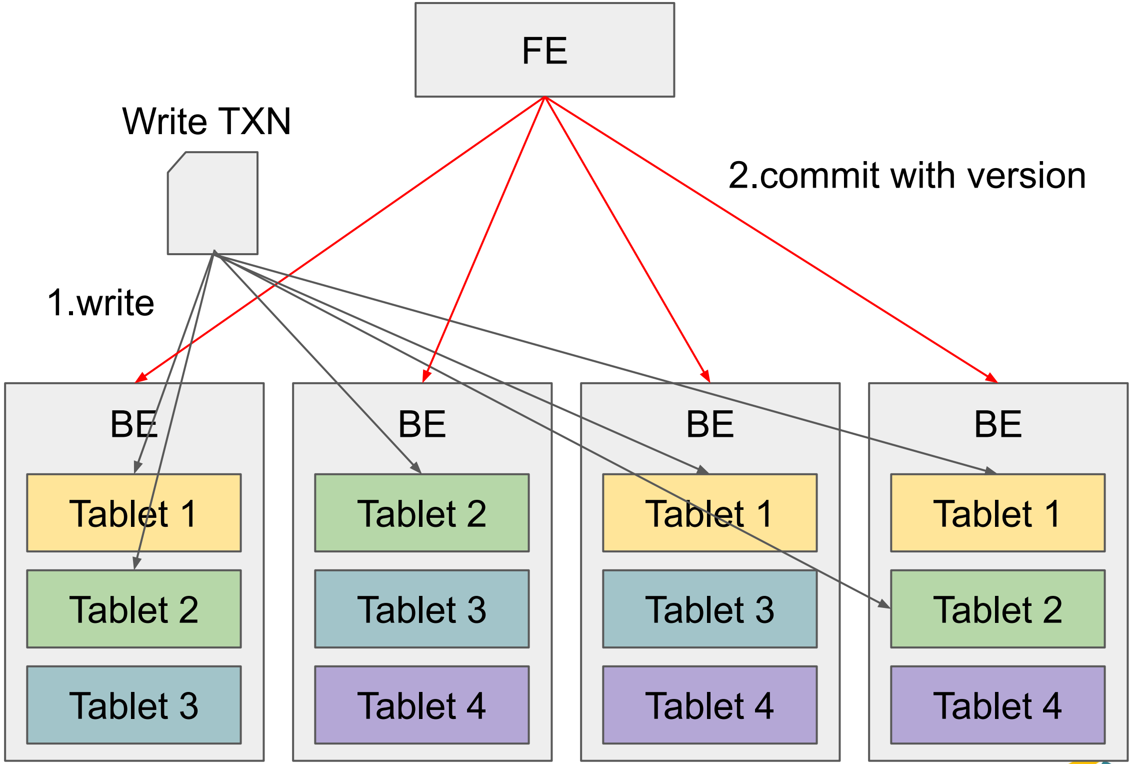
FE

BE	BE	BE	BE
Tablet 1	Tablet 2	Tablet 1	Tablet 1
Tablet 2	Tablet 3	Tablet 3	Tablet 2
Tablet 3	Tablet 4	Tablet 4	Tablet 4

System Overview

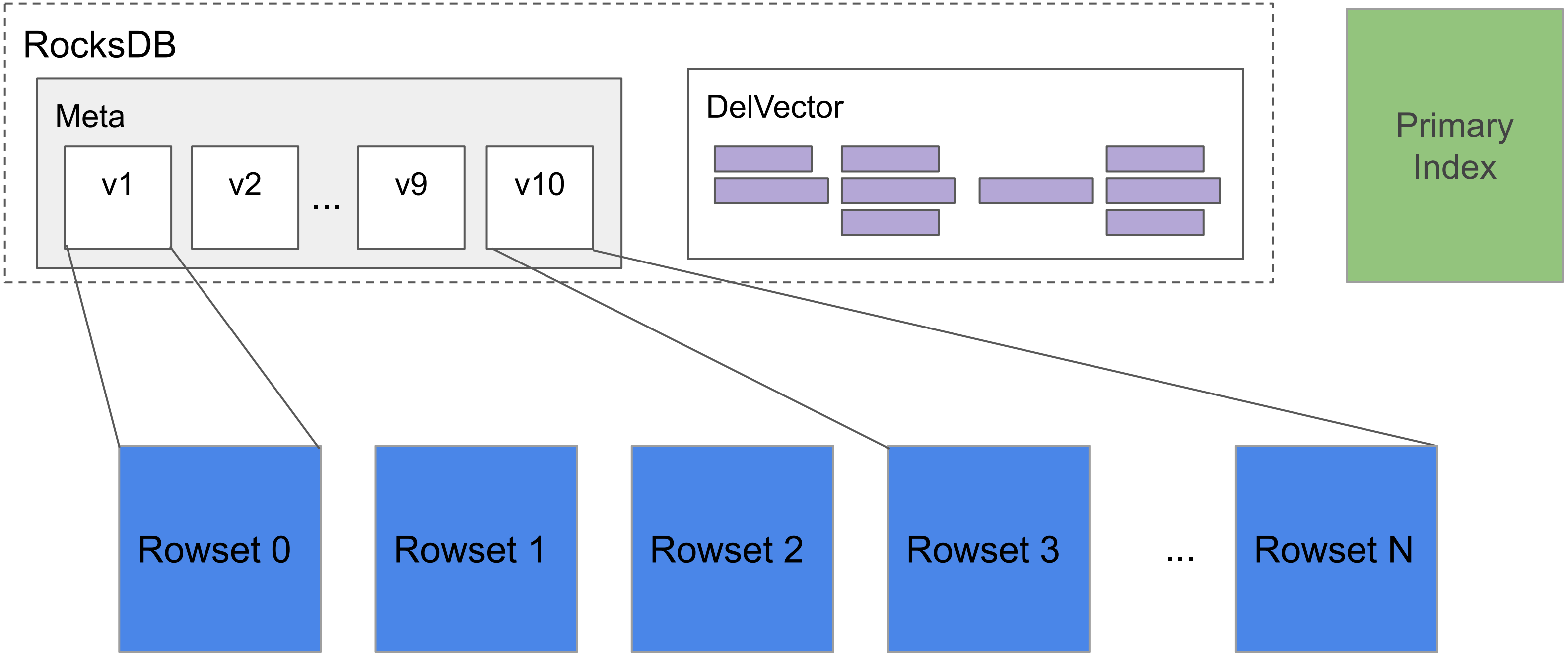


System Overview



StarRocks | DataFun.

Inside a Tablet



Metadata

- PB saved in RocksDB
- Cached in-memory
- Meta
  - versions: list<EditVersionMeta>
    - EditVersion : {major, minor}
    - Rowsets: list<uint32>
    - Delta: list<uint32>
  - rowset_id_next:uint32

```
[
 {
  EditVersion: [4,0],
  Rowsets: [1,2,3]
  Delta: [3]
 },
 {
  EditVersion: [5,0],
  Rowsets: [1,2,3,4]
  Delta: [4]
 },
 {
  EditVersion: [6,0],
  Rowsets: [1,2,3,4,5]
  Delta: [5]
 },
 rowset_id_next: 6
]
```

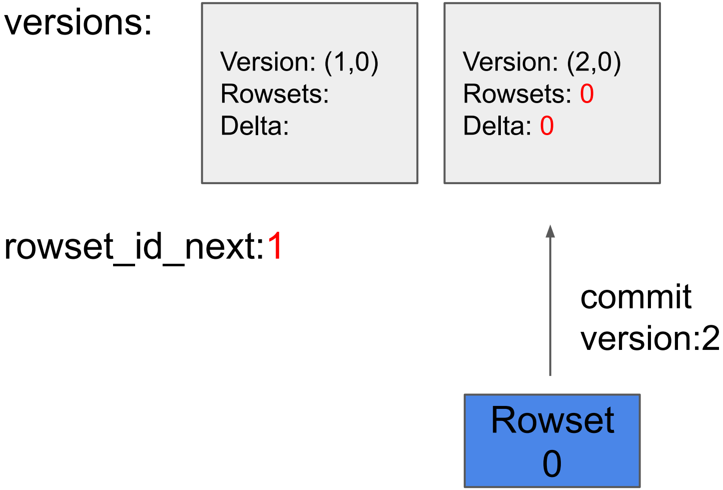
Example

versions:

Version: (1,0)
Rowsets:
Delta:

rowset_id_next:0

Example



versions:

Version: (1,0)
Rowsets:
Delta:

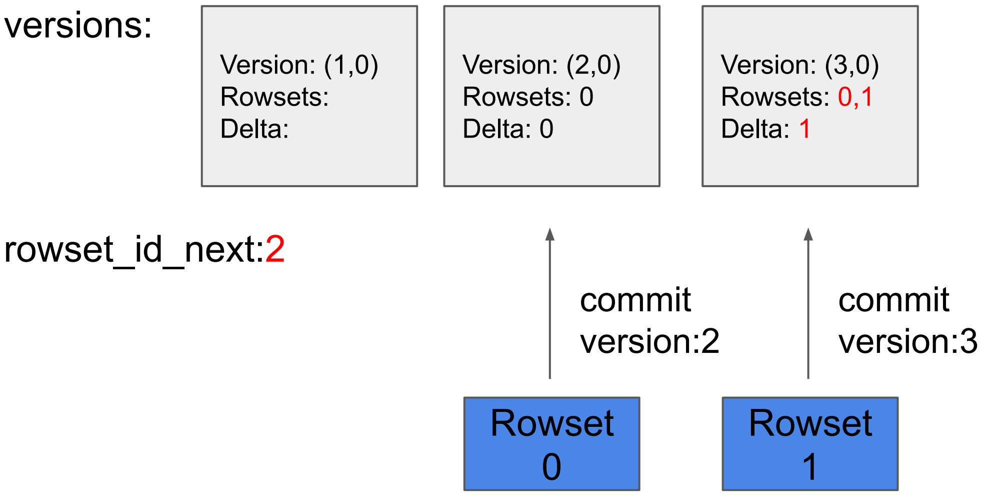
Version: (2,0)
Rowsets: 0
Delta: 0

rowset_id_next:1

commit
version:2

Rowset
0

Example



versions:

Version: (1,0)
Rowsets:
Delta:

Version: (2,0)
Rowsets: 0
Delta: 0

Version: (3,0)
Rowsets: 0,1
Delta: 1

rowset_id_next:2

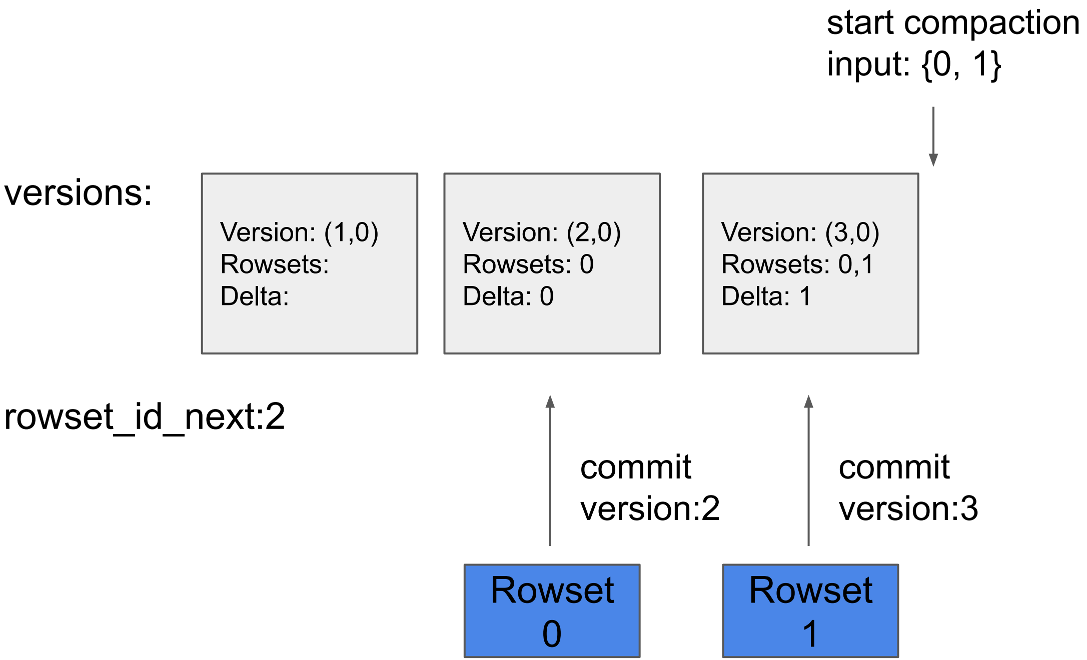
commit
version:2

commit
version:3

Rowset
0

Rowset
1

Example



start compaction
input: {0, 1}

versions:

Version: (1,0)
Rowsets:
Delta:

Version: (2,0)
Rowsets: 0
Delta: 0

Version: (3,0)
Rowsets: 0,1
Delta: 1

rowset_id_next:2

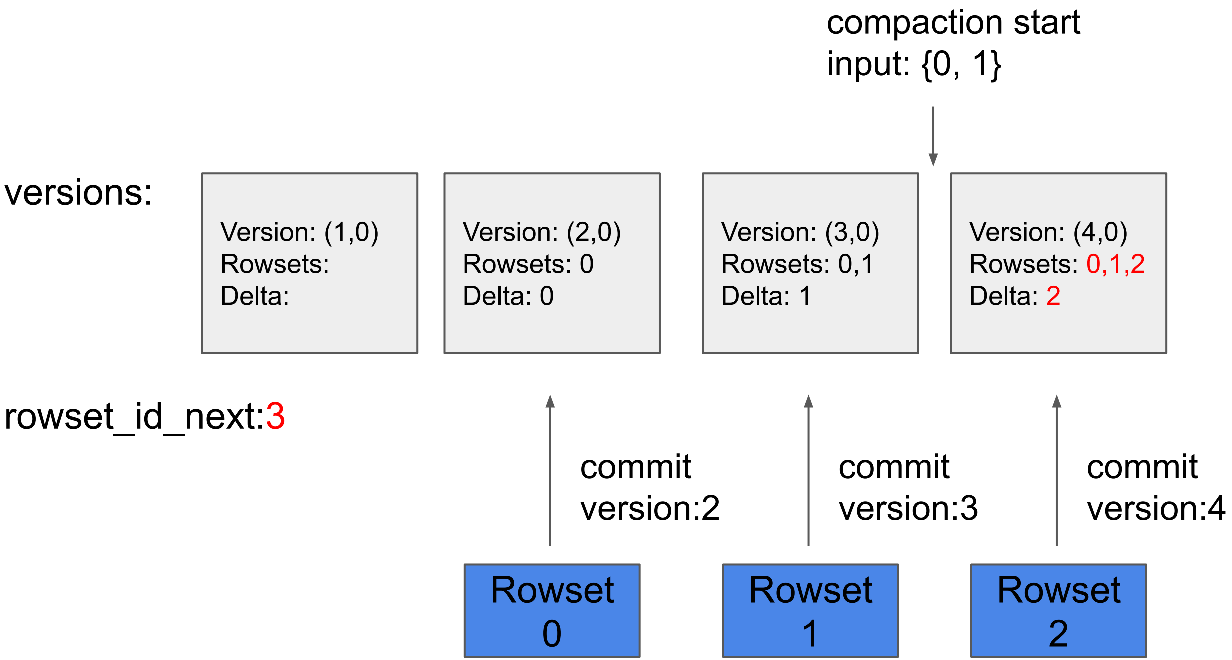
commit
version:2

commit
version:3

Rowset
0

Rowset
1

Example



compaction start
input: {0, 1}

versions:

Version: (1,0)
Rowsets:
Delta:

Version: (2,0)
Rowsets: 0
Delta: 0

Version: (3,0)
Rowsets: 0,1
Delta: 1

Version: (4,0)
Rowsets: 0,1,2
Delta: 2

rowset_id_next:3

commit
version:2

commit
version:3

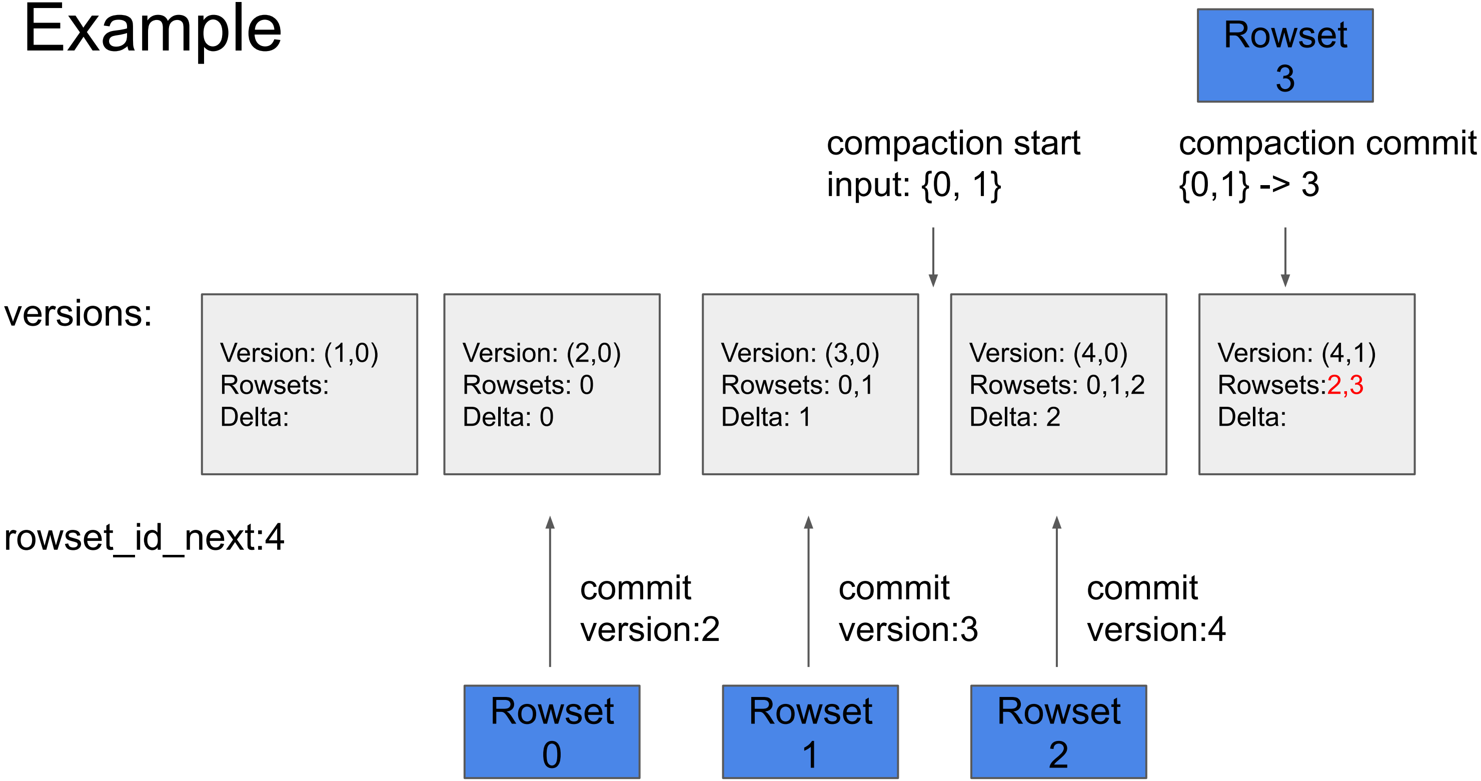
commit
version:4

Rowset
0

Rowset
1

Rowset
2

Example



Rowset 3

compaction start
input: {0, 1}

compaction commit
{0,1} -> 3

versions:

Version: (1,0)
Rowsets:
Delta:

Version: (2,0)
Rowsets: 0
Delta: 0

Version: (3,0)
Rowsets: 0,1
Delta: 1

Version: (4,0)
Rowsets: 0,1,2
Delta: 2

Version: (4,1)
Rowsets:2,3
Delta:

rowset_id_next:4

commit
version:2

commit
version:3

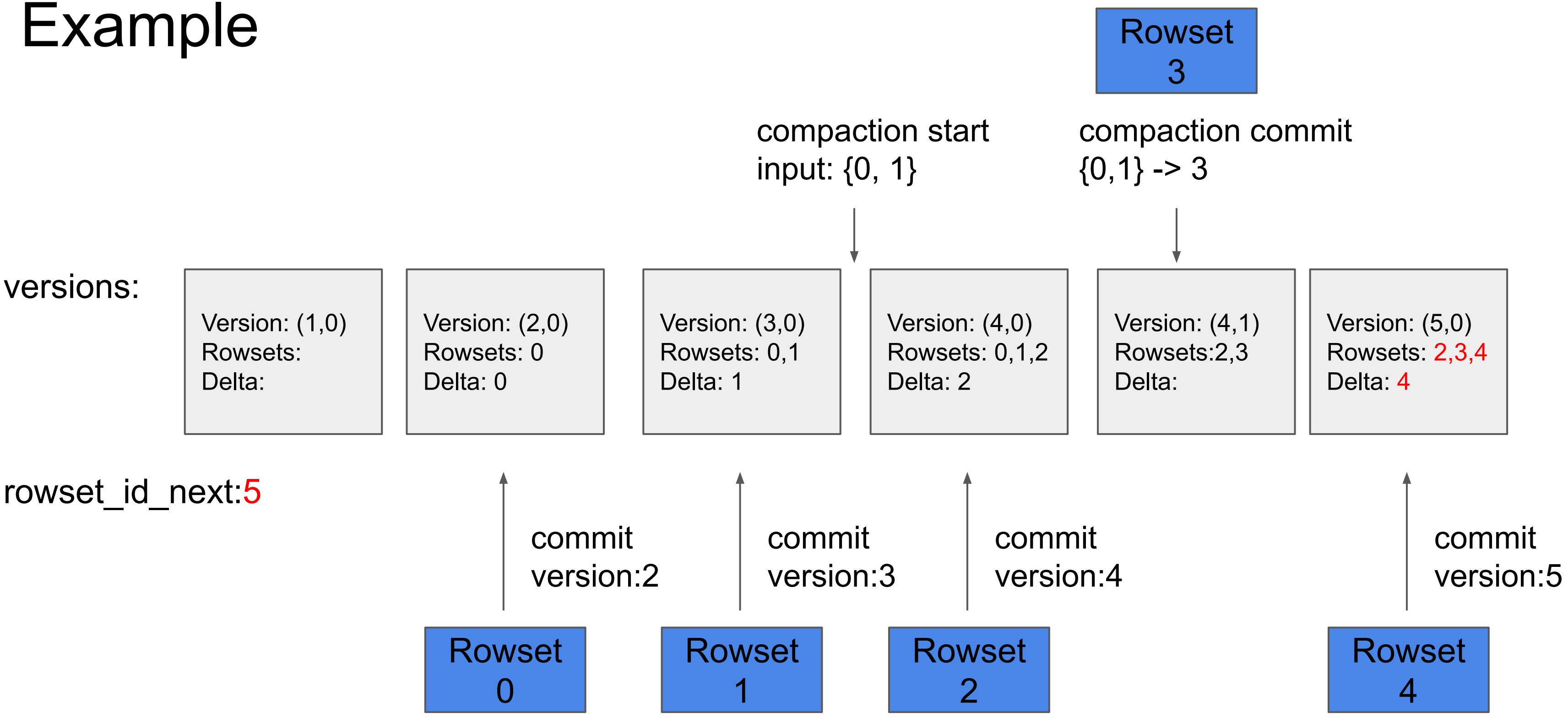
commit
version:4

Rowset 0

Rowset 1

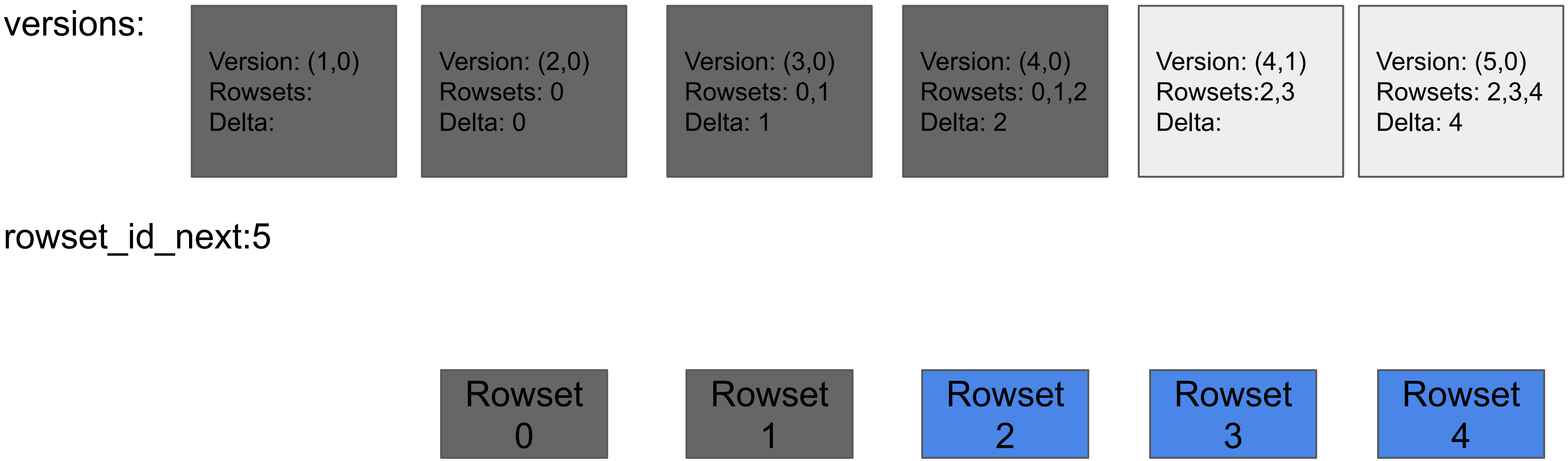
Rowset 2

Example



StarRocks | DataFun.

Example: Version GC



versions:

Version: (1,0)
Rowsets:
Delta:

Version: (2,0)
Rowsets: 0
Delta: 0

Version: (3,0)
Rowsets: 0,1
Delta: 1

Version: (4,0)
Rowsets: 0,1,2
Delta: 2

Version: (4,1)
Rowsets:2,3
Delta:

Version: (5,0)
Rowsets: 2,3,4
Delta: 4

rowset_id_next:5

Rowset 0

Rowset 1

Rowset 2

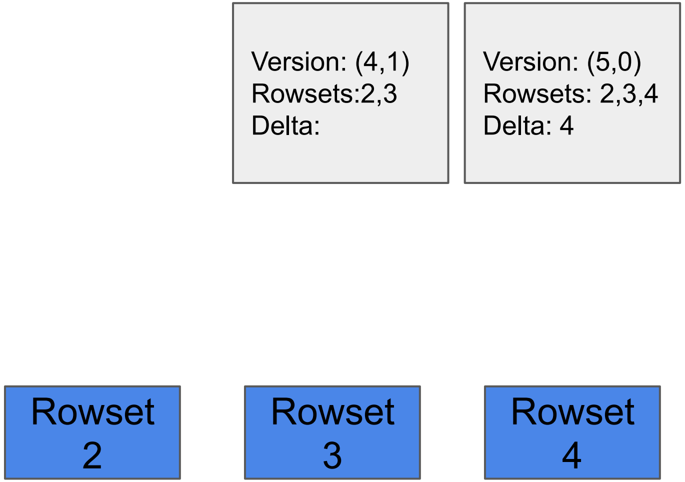
Rowset 3

Rowset 4

Example: Version GC

versions:

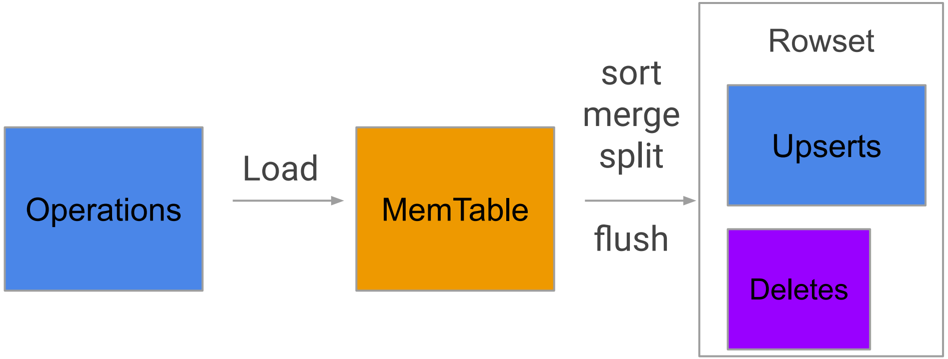
rowset_id_next:5



Version: (4,1)
Rowsets:2,3
Delta:

Version: (5,0)
Rowsets: 2,3,4
Delta: 4

Write Pipeline



id	c0	c1	del
1	a	2	0
3	c	6	0
2	b	4	0
1	a	2	1
2	b	5	0

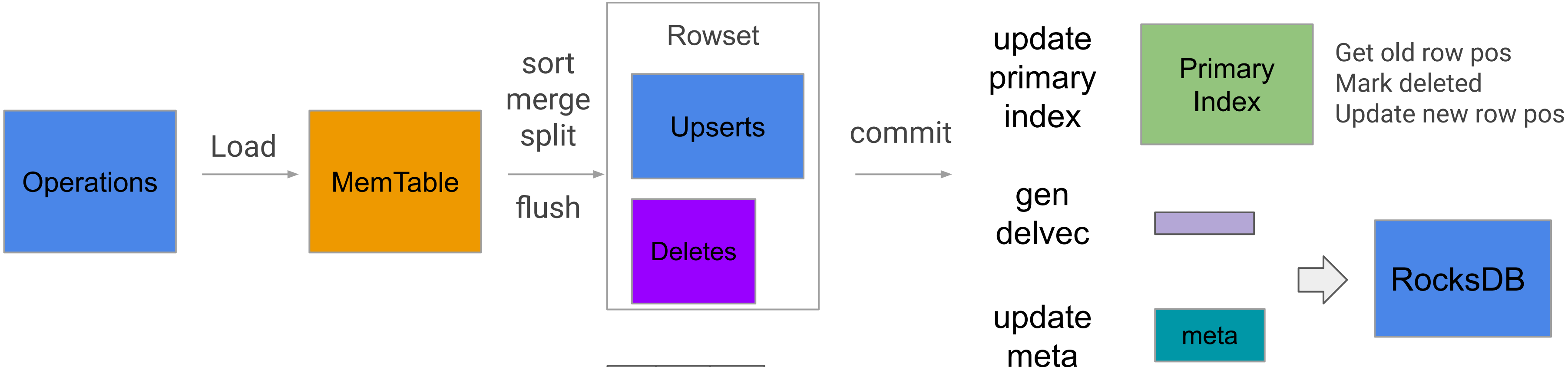
upserts

id	c0	c1
2	b	5
3	c	6

deletes

1

Write Pipeline



Get old row pos
Mark deleted
Update new row pos

upserts

id	c0	c1
2	b	5
3	c	6

deletes

1

Example

Version 1

Rowset 0

id	c0
1	a
2	b
3	c
4	d

Primary Index

id	rs	row
1	0	0
2	0	1
3	0	2
4	0	3

Example

Version 2

Rowset 0

id	c0
1	a
2	b
3	c
4	d

Rowset 1

id	c0
1	aa
3	cc
5	e

Primary Index

id	rs	row
1	0	0
2	0	1
3	0	2
4	0	3

Example

Version 2

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec

1
0
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

Primary Index

id	rs	row
1	1	0
2	0	1
3	1	1
4	0	3
5	1	2

Example

Version 3

Rowset 0

id	c0	delvec
1	a	1
2	b	0
3	c	1
4	d	0

Rowset 1

id	c0
1	aa
3	cc
5	e

Rowset 2

id	c0
6	f

del
3

Primary Index

id	rs	row
1	1	0
2	0	1
3	1	1
4	0	3
5	1	2

Example

Version 3

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec
1
0
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

delvec
0
1
0

Rowset 2

id	c0
6	f

del
3

Primary Index

id	rs	row
1	1	0
2	0	1
4	0	3
5	1	2
6	2	0

Example

Version 4

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec
1
1
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

delvec
0
1
1

Rowset 2

id	c0
6	f

del
3

Rowset 3

id	c0
2	bb
5	ee

Primary Index

id	rs	row
1	1	0
2	0	1
4	0	3
5	1	2
6	2	0

Example

Version 4

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec

1
1
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

delvec

0
1
1

Rowset 2

id	c0
6	f

del
3

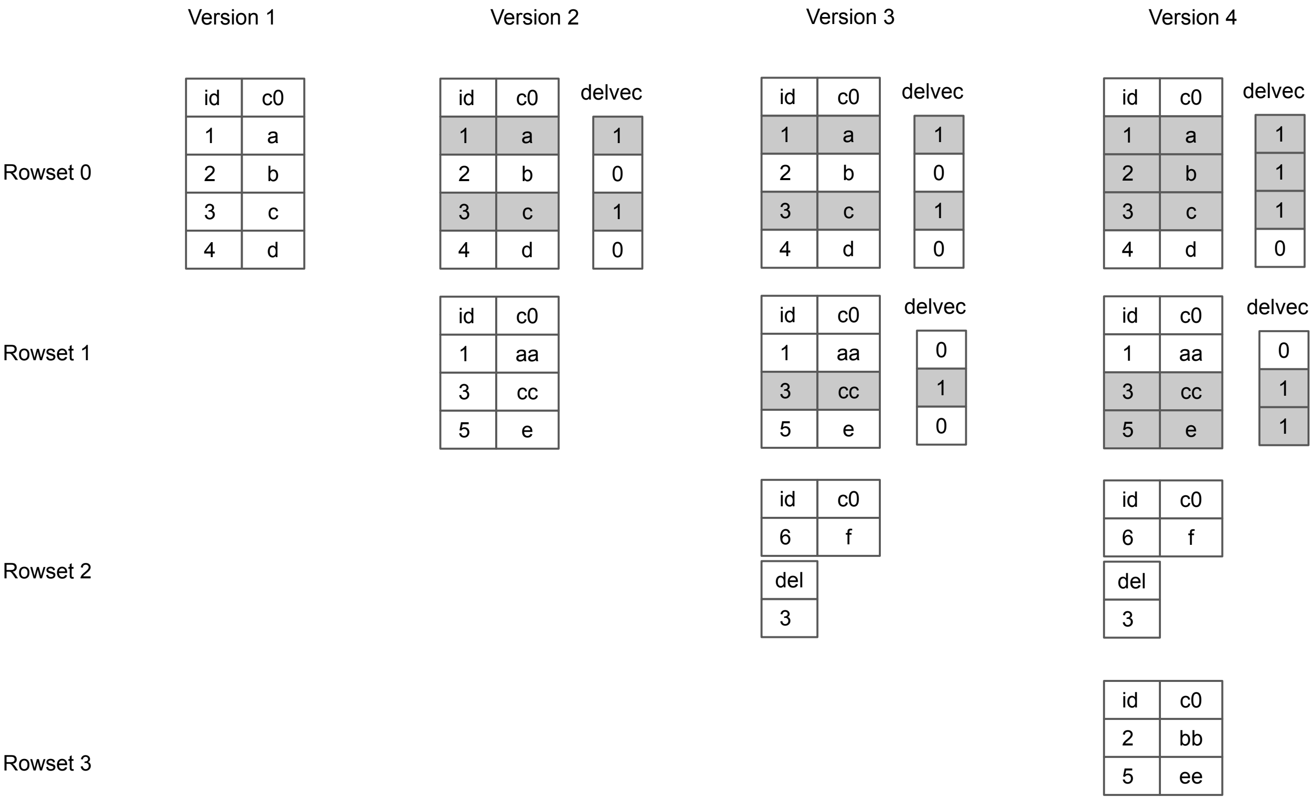
Rowset 3

id	c0
2	bb
5	ee

Primary Index

id	rs	row
1	1	0
2	3	0
4	0	3
5	3	1
6	2	0

MVCC



Version 1

Rowset 0

id	c0
1	a
2	b
3	c
4	d

Version 2

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec
1
0
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

Version 3

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec
1
0
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

delvec
0
1
0

Rowset 2

id	c0
6	f

del
3

Version 4

Rowset 0

id	c0
1	a
2	b
3	c
4	d

delvec
1
1
1
0

Rowset 1

id	c0
1	aa
3	cc
5	e

delvec
0
1
1

Rowset 2

id	c0
6	f

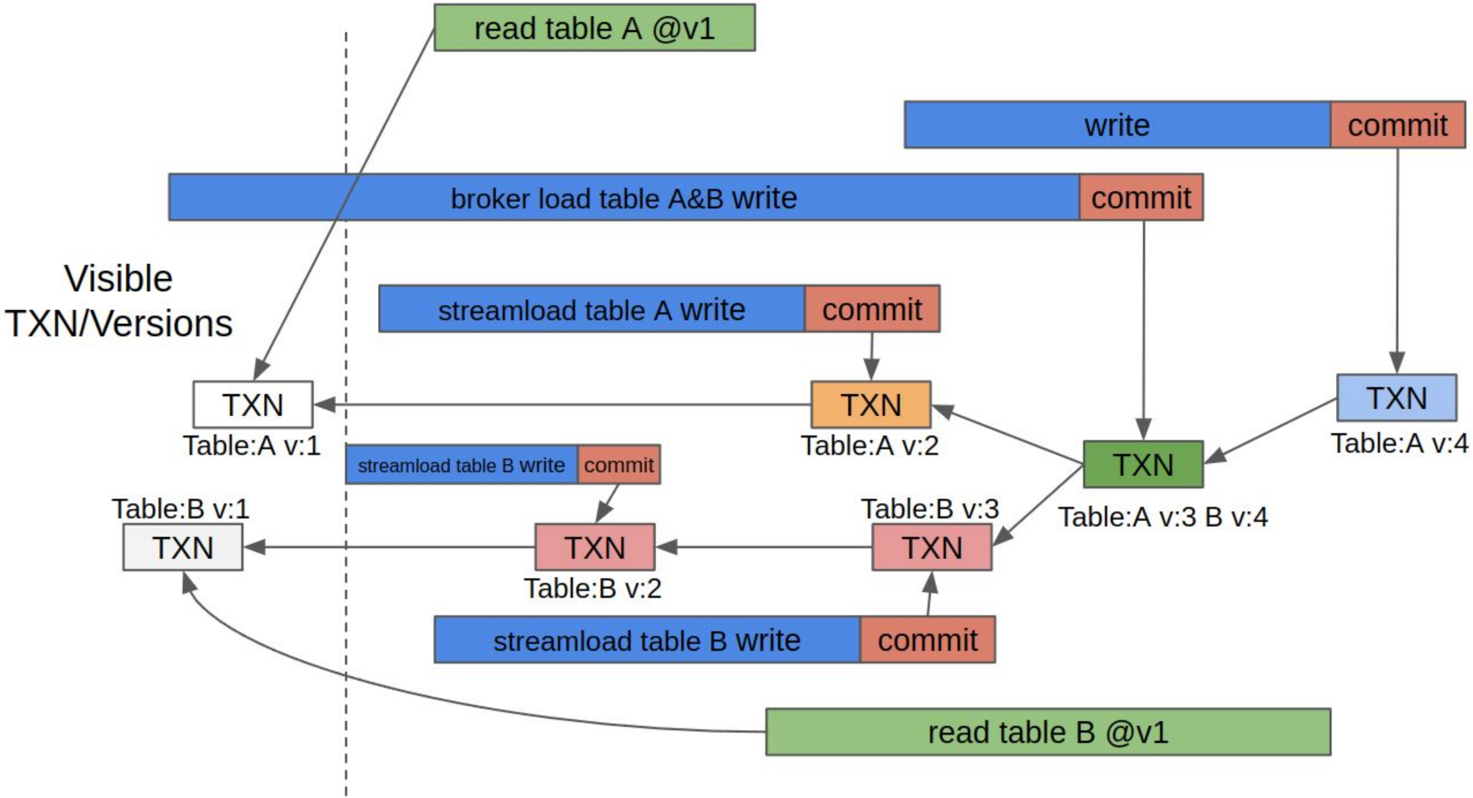
del
3

Rowset 3

id	c0
2	bb
5	ee

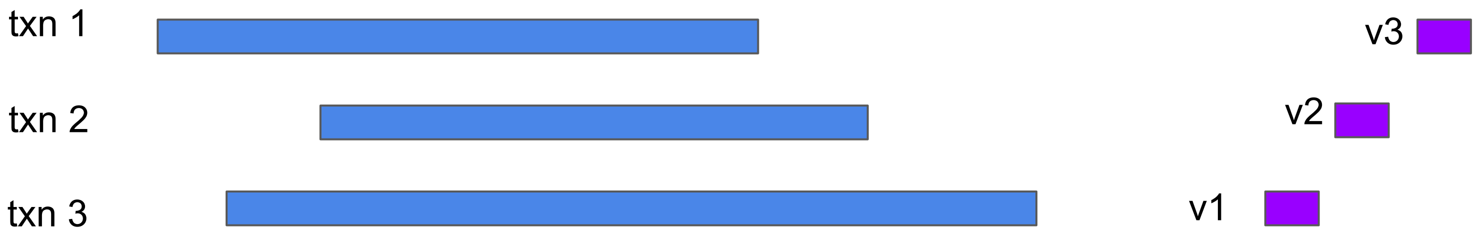
StarRocks | DataFun.

Concurrency Control



Concurrency Control

- Each write-only txn has 2 phases
  - write: run concurrently
  - commit: run serially, should be very fast!
- FE commit with version
  - FE decide txn order



write
receive data/sort,merge,flush memtable
create rowset

commit
update primary index
write delvect&meta

Primary Index

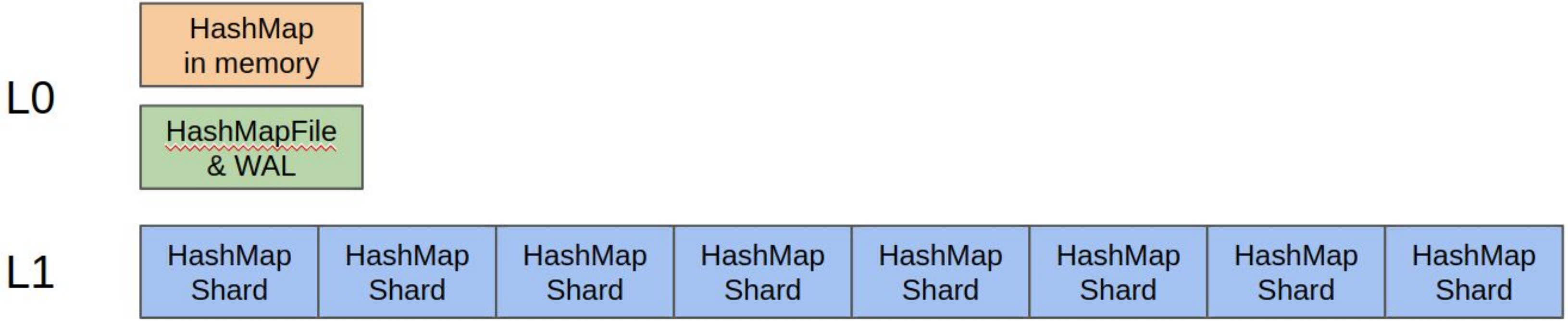
- Primary index update takes >90% time in commit phase
- Efficient in-memory hashmap
  - Key: composite primary key encoded binary slice
  - Value: uint64 row position <32bit rowset_id, 32 bit rowid>
  - phmap
  - fast: 20~200ns/op or 5M~50M op/s per thread
- e.g. 10M row write into 10 bucket in single TXN
  - each bucket update 1M op in hashmap
  - commit duration ~ 0.12s (assuming 10M op/s)

Primary Index Optimization

- Cache miss for large hashmap
  - batch update: prefetch
- Memory usage
  - fix length key: use FixSlice as key (no need to store length)
  - var length key: shard by length (1/2 - 1/3 memory)
  - on-demand loading
    - release if no more load for 6min
- Ongoing work
  - Shard by constant/Low cardinality columns
  - Use 128bit hash as primary key(small probability of conflict)

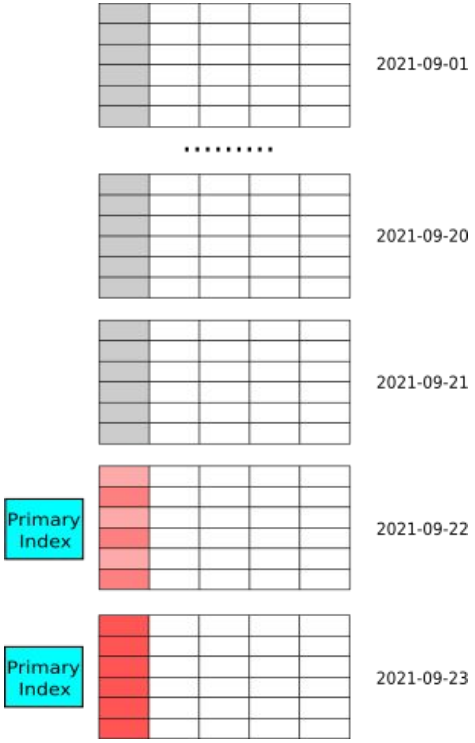
Persistent Primary Index

- https://github.com/StarRocks/starrocks/pull/3044, etc.
- On disk hashmap
- L0 & L1 LSM-like structure



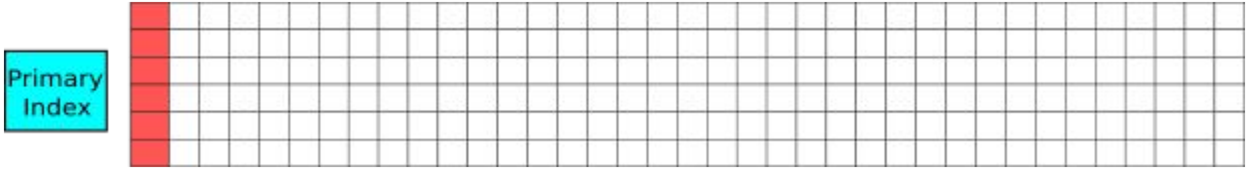
Use Case Cold/Hot Data

- Data partition by date
- Records getting cold gradually
- Only recent partitions have updates
  - so only fraction of partitions load index
- Example:
  - E-commerce orders
  - Taxi/bike rides
  - Client sessions



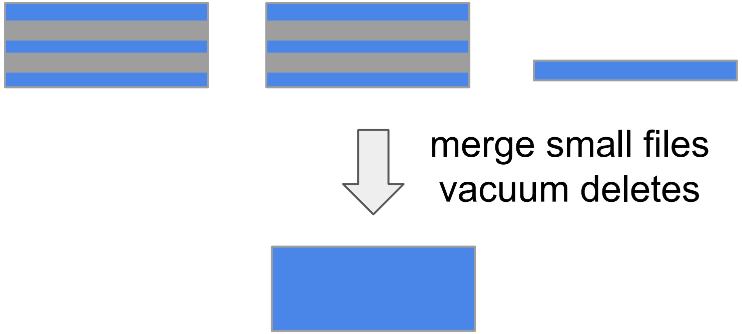
Use Case: Wide Table

- Large columns (>100 columns)
- Primary key only takes fraction of total storage space
- Example:
  - User profile, user_id as primary key

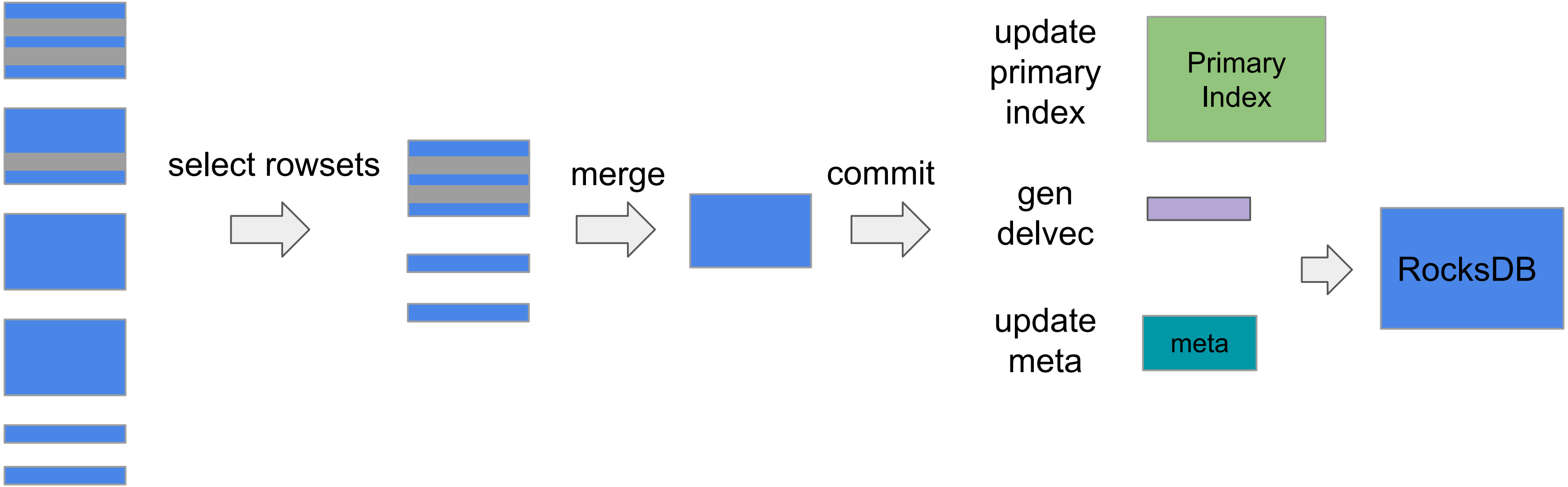


Compaction

- Rowsets with increasing deletes
  - need to read&skip deleted rows, slow down scan
  - delvector copy-on-write, meta overhead
- Small rowset file
- LSM compaction
  - merge sst -> generate new sst
  - atomically replace meta
- Different design
  - No duplicate rows
  - No (range)delete tombstone (vs rocksdb)
  - Need to maintain primary index



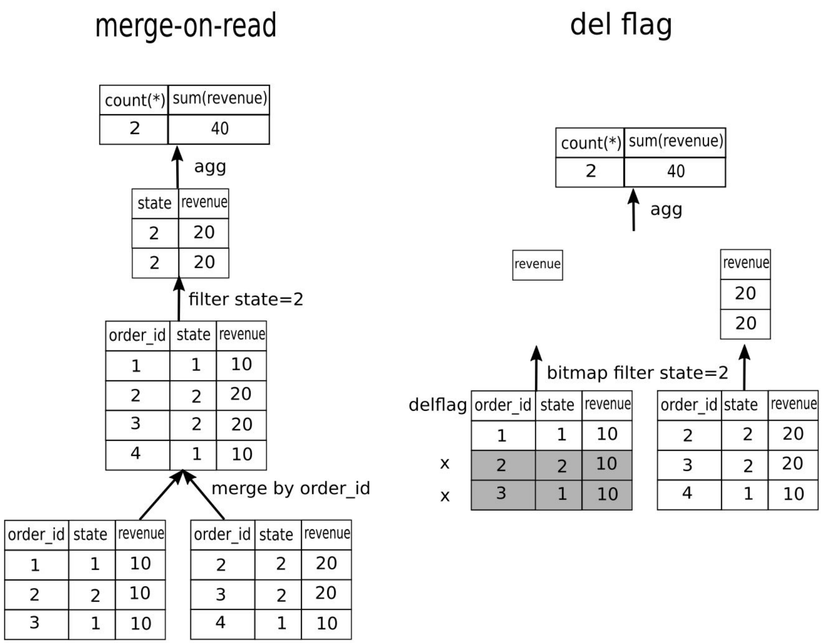
Compaction


select rowsets
merge
commit
update primary index
Primary Index
gen delvec
update meta
meta
RocksDB

Example

- No need to do merge
- No need to read order_id
- Pushdown state = 2
    - can use index
- Scanner only return revenue column
- Scanner can be parallelized

select count(*), sum(revenue) from orders where state=2



merge-on-read

count(*)	sum(revenue)
2	40

agg

state	revenue
2	20
2	20

filter state=2

order_id	state	revenue
1	1	10
2	2	20
3	2	20
4	1	10

merge by order_id

order_id	state	revenue
1	1	10
2	2	10
3	1	10

order_id	state	revenue
2	2	20
3	2	20
4	1	10

del flag

count(*)	sum(revenue)
2	40

agg

revenue

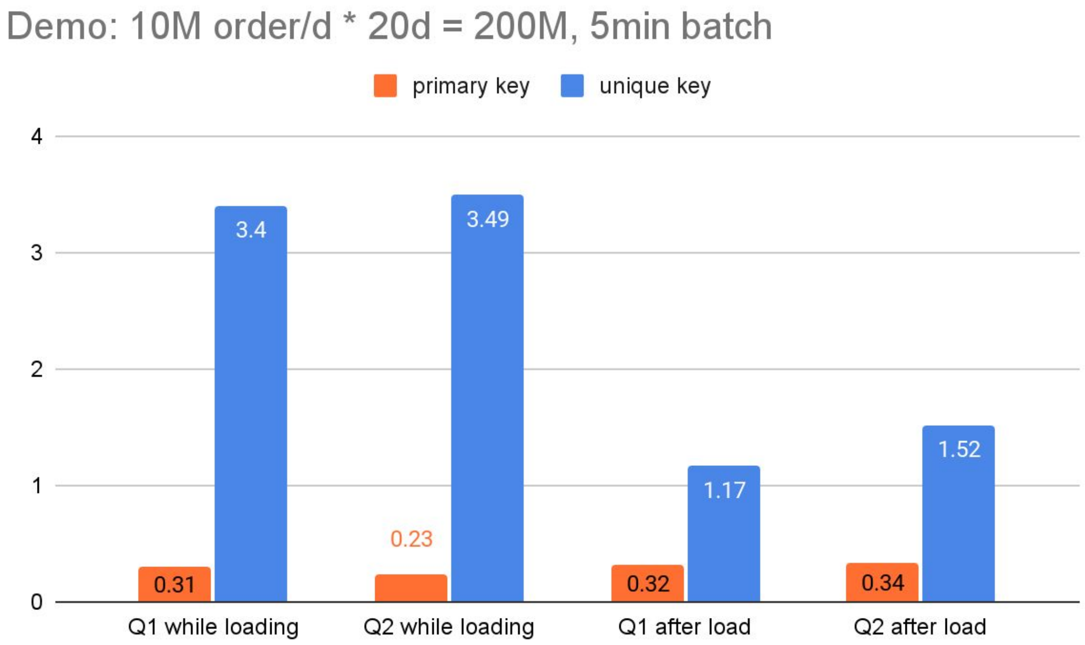
revenue
20
20

bitmap filter state=2

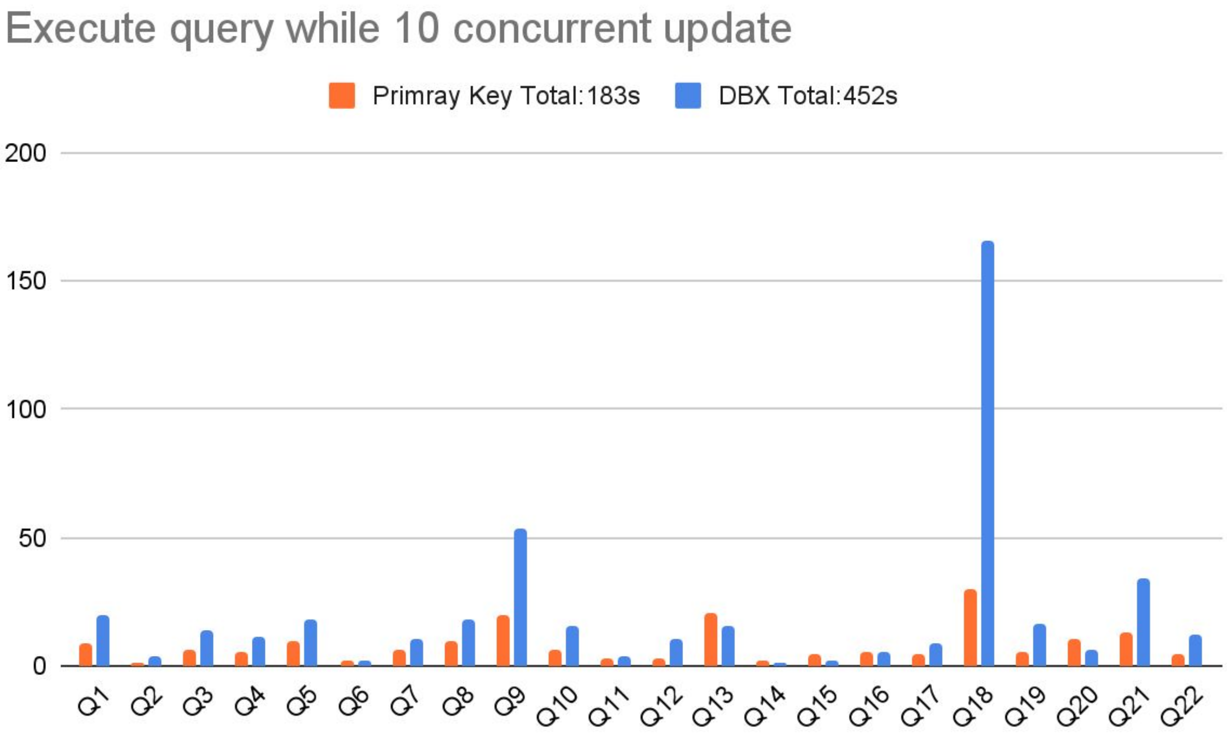
delflag	order_id	state	revenue
	1	1	10
x	2	2	10
x	3	1	10

order_id	state	revenue
2	2	20
3	2	20
4	1	10

Benchmark: Simple Query on Single Table



Benchmark: TPC-H 1T

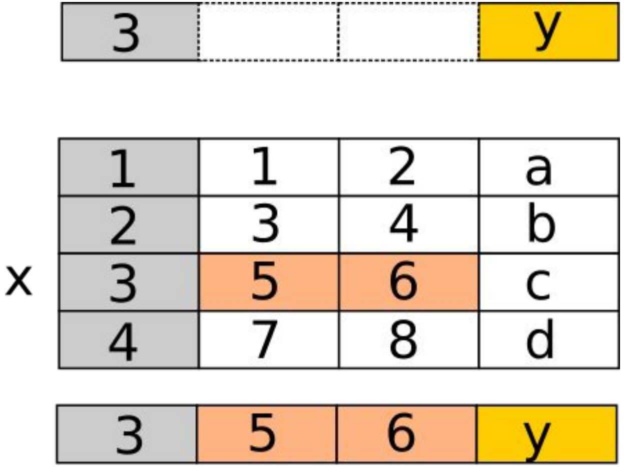


03

当前和未来工作

Read-Write Updates: Partial Update

- Read then write
- Harder than upsert
  - Last writer wins doesn’t work
- Need to abort, retry
  - Or resolve conflict



3			y

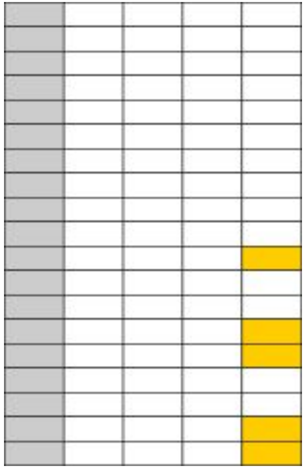
	1	1	2	a
	2	3	4	b
x	3	5	6	c
	4	7	8	d

3	5	6	y

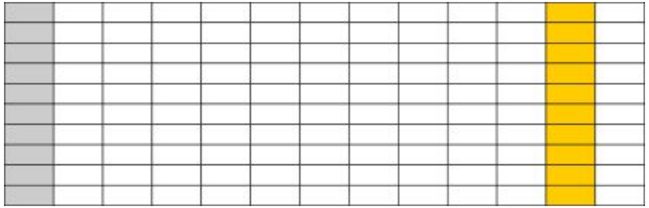
1. find row for 3
2. read column value 5,6
3. mark row deleted
4. append new row

https://github.com/StarRocks/StarRocks/pulls?q=%22partial+update%22

Read-Write Updates: Partial Update



hot column update
e.g. only update order status



whole column batch update
e.g. A ML job batch update an user tag column in user_profile table

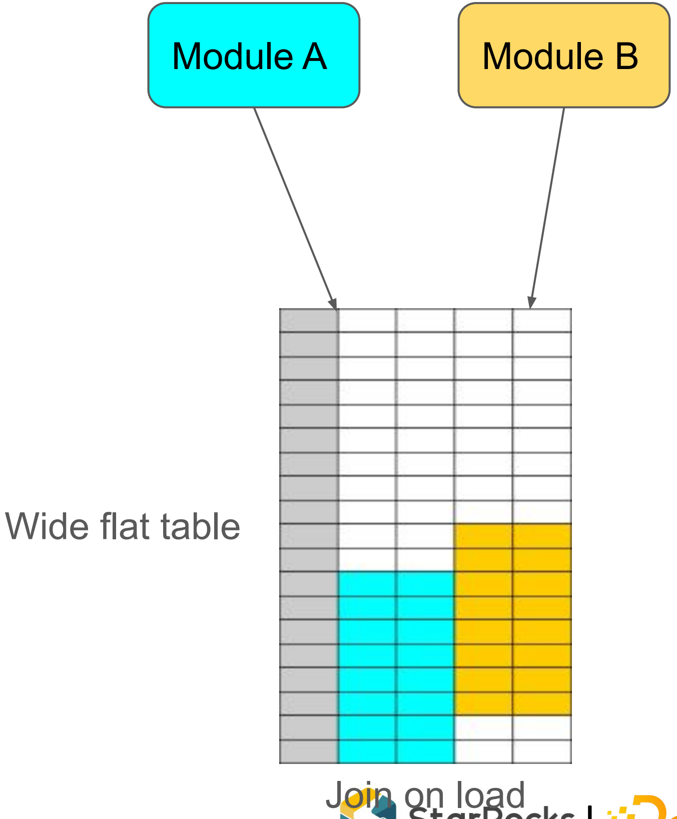
```
merge into dest using src on dest.id = src.id
  when matched then update set dest.v = src.v;
```

Read-Write Updates: Partial Update

- Each module only knows subset of columns
- Example:
  - Ad display, click
  - Order: shop, payment, inventory, logistics, review
- Current solution:
  - Join in stream system(ie.flink), load to AP
  - update in TP, then CDC to AP
  - batch “merge into”

```
merge into order using pay on order.id = pay.id
  when matched then update set
    order.pay_ts = pay.ts
    order.pay_method = pay.method
    ...;
```

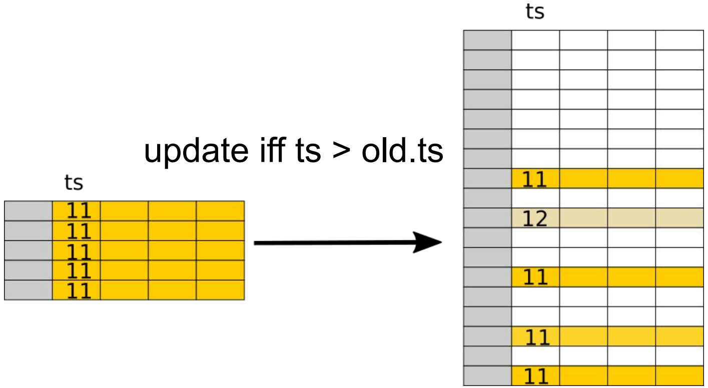
```
merge into order using ship on order.id = ship.id
  when matched then update set
    order.ship_start_ts = ship.start_ts
    order.ship_end_ts = ship.end_ts
    ...;
```



Read-Write Updates

Conditional Update

- out-of-order arrival

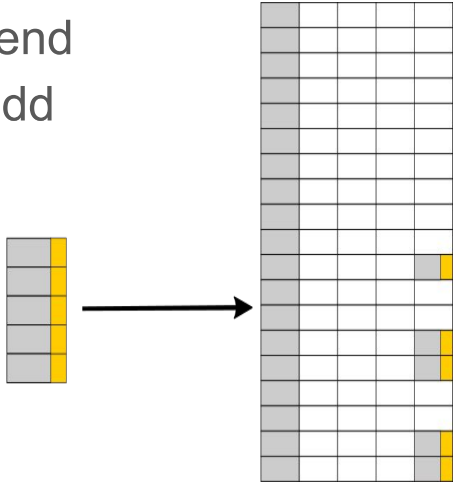


```
merge into dest using src on dest.id = src.id
  when matched and src.ts > dest.ts then
    update set dest.c1 = src.c1, ...
  when not matched then
    insert *;
```

Read-Write Updates

Merge Update

- array append
- map/set add

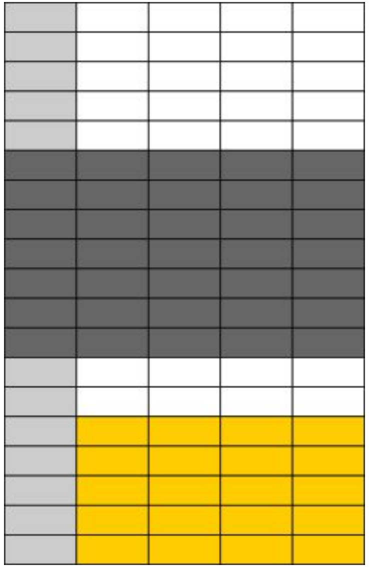


```
merge into dest using src on dest.id = src.id
  when matched then
    update set items=array_append(items, src.item)
```

Read-Write Updates

- General read-write transaction
  - delete where col = value
  - insert into select
  - merge into
  - multi-statement transaction
    - e.g. data fix (batch delete + update)

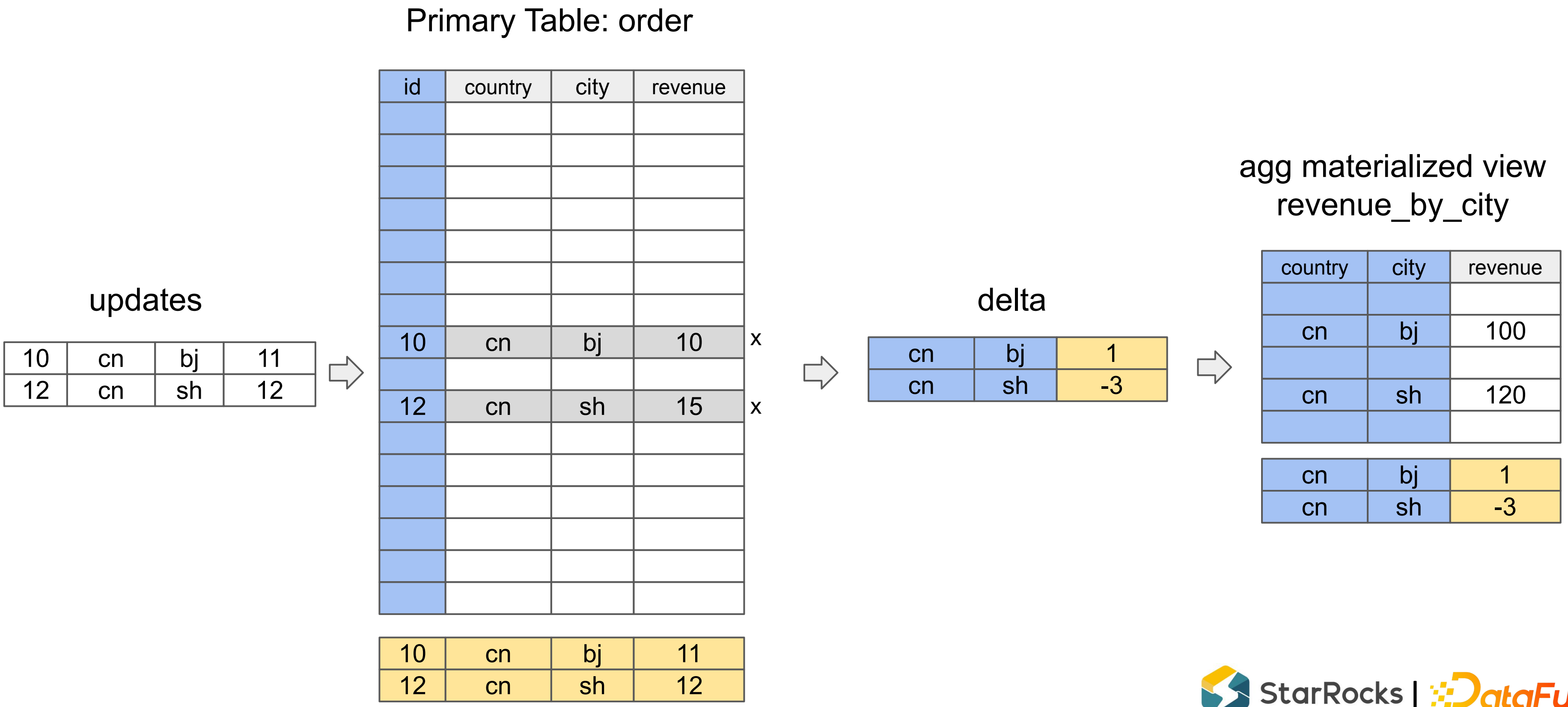
```
begin;
delete from orders where userid = 1001;
insert into orders select * from user1001_fix;
commit;
```



Transaction Difficulty Levels

Type	Traits (* Read-Set: R, Write-Set: W)	Application
Append only	$R:\varnothing \cap W = \varnothing$	append only log analysis
Upsert/Delete	$R:\varnothing$	TP CDC data load with deduplication
Local Update (Partial update Conditional update Merge update)	R=W each written row only depending on self	Many use case Optimization opportunities Ongoing work
General read-write	R!=W #R #W $\gg$ 0	Batch DML ELT Ongoing work
General read-write with rollback	R!=W #R #W $\gg$ 0 non-deterministic	Multi-statement batch DML ELT

Materialized View for Primary Key Table



updates

10	cn	bj	11
12	cn	sh	12

Primary Table: order

id	country	city	revenue	
10	cn	bj	10	x
12	cn	sh	15	x

10	cn	bj	11
12	cn	sh	12

delta

cn	bj	1
cn	sh	-3

agg materialized view
revenue_by_city

country	city	revenue
cn	bj	100
cn	sh	120

cn	bj	1
cn	sh	-3

非常感谢您的观看

StarRocks | DataFun.