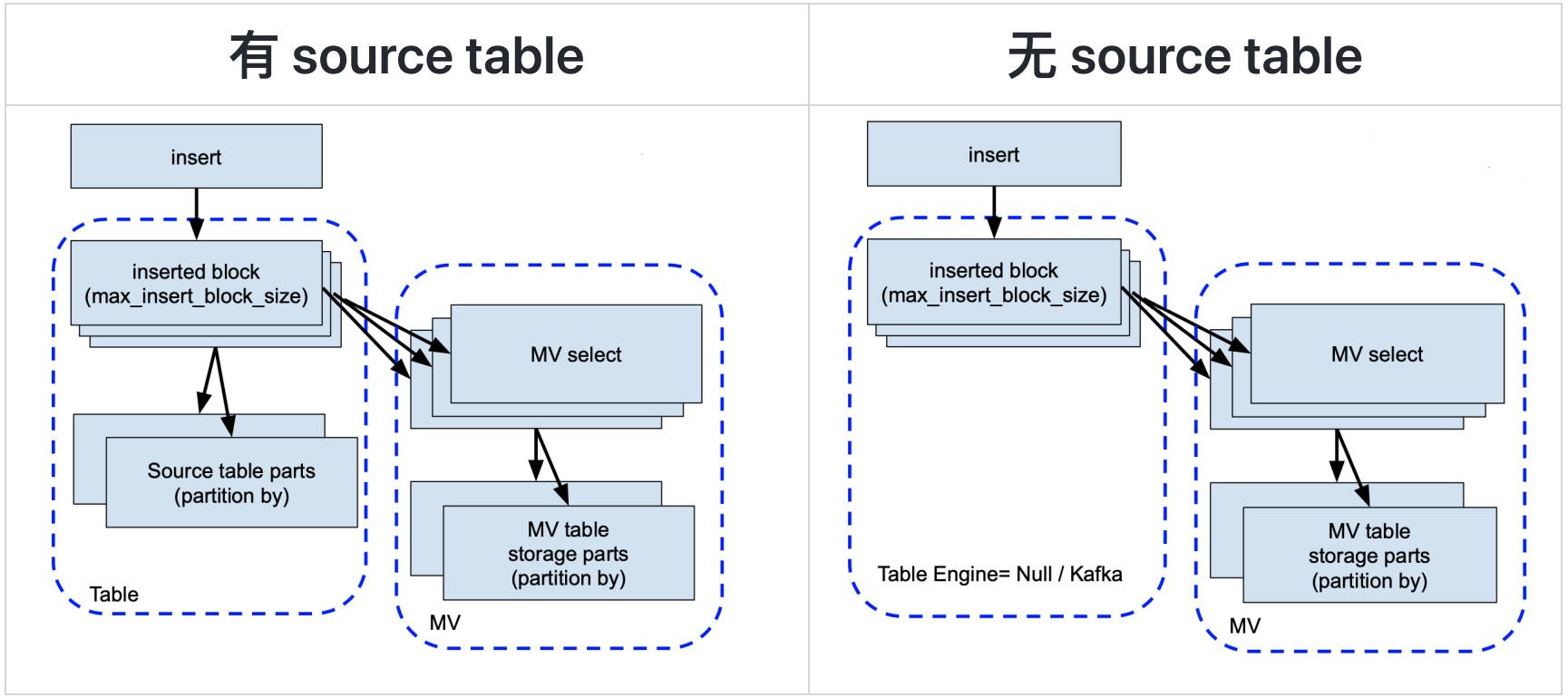
二十四分钟精通ClickHouse Materialized View

MV是一个Trigger

- MV不会读取source table读取
- 调用一次insert时，MV select可能会被trigger多次

数据写入

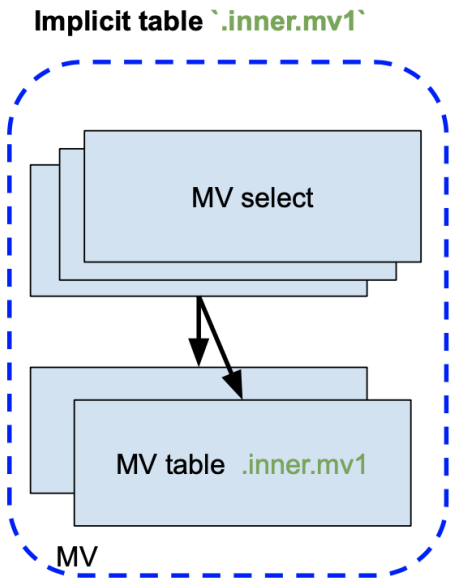

有 source table
insert
inserted block
(max_insert_block_size)
Source table parts
(partition by)
Table
MV select
MV table
storage parts
(partition by)
MV
无 source table
insert
inserted block
(max_insert_block_size)
Table Engine= Null / Kafka
MV select
MV table
storage parts
(partition by)
MV

创建MV

- 直接创建
- 使用 `TO` 创建

直接创建

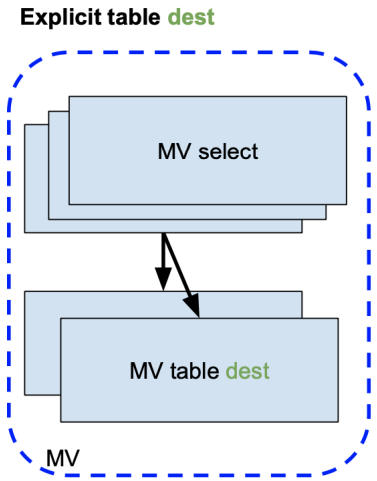
```
CREATE MATERIALIZED VIEW mv1
ENGINE = SummingMergeTree
ORDER BY (id, d)
AS
SELECT id, d, count() AS cnt
FROM source
GROUP BY id, d;
```



使用TO创建

```
CREATE TABLE dest
(id String, d Date, cnt UInt64)
ENGINE = SummingMergeTree
ORDER BY (id, d);

CREATE MATERIALIZED VIEW mv1
TO dest
AS
SELECT id, d, count() AS cnt
FROM source
GROUP BY id, d;
```



区别

Implicit table

- optimize_move_to_prewhere 在查询MV时不可用
- 可以使用populate在创建时插入数据
- drop mv时，会自动drop inner table

Explicit table

- 不能使用populate创建，需要使用insert手动插入（见下文）
- drop mv时，dest table不会被删除

如何使用

使用 `TO` , ALWAYS

- 显示创建table方便运维，因为本身就是一张普通表，并且使其可见
- `polulate` 实际不可用
- 他会针对所有的数据运行，数据越打，持续时间越长，甚至会超时或内存不足。这在7x24小时运行的系统中基本不会采用
- 在执行过程中插入到source table的数据不会被插入到MV中

特别注意

MV中的聚合计算不包含source table所有数据

```
CREATE MATERIALIZED VIEW mv1
ENGINE = AggregatingMergeTree
PARTITION BY toYYYYMM(hour)
ORDER BY hour
POPULATE
AS
SELECT toStartOfHour(time) hour,
        maxState(cnt_by_minute) max_by_hour,
        sumState(cnt_by_minute) sum_by_hour
FROM
(
        SELECT minute, count() AS cnt_by_minute
        FROM source
        GROUP BY minute
)
GROUP BY hour
```

```
-- sql1
insert into source values (now()), (now());
-- max_by_hour = 2

-- sql2
insert into source values (now());
insert into source values (now());
-- max_by_hour = 1
```

MV的计算是针对插入的block，而不是source table所有数据

source table的数据操作不会影响MV中的数据

- source table中数据删改，MV中数据不会变化
- source table和MV可以存储不同时长的数据。
  例如source table中存储最近半年的数据，但是MV中存储10年以内的聚合数据

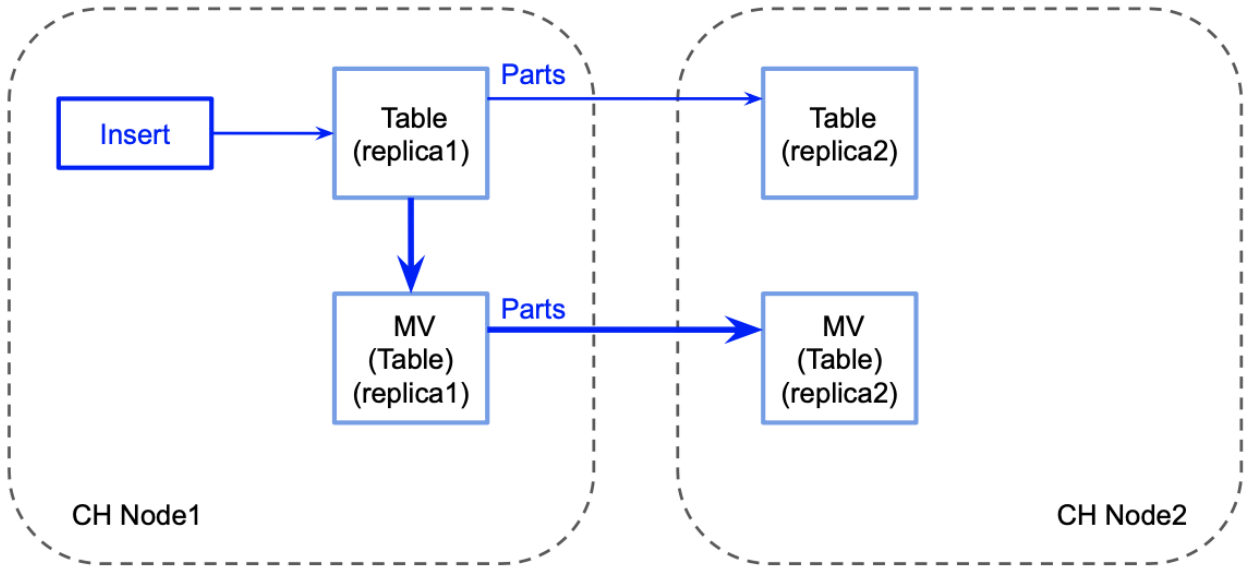
MV with Replicated Engines

MV的storage table就是普通的table，因此也可以像普通table一样使用Replicated Engine。

创建方式

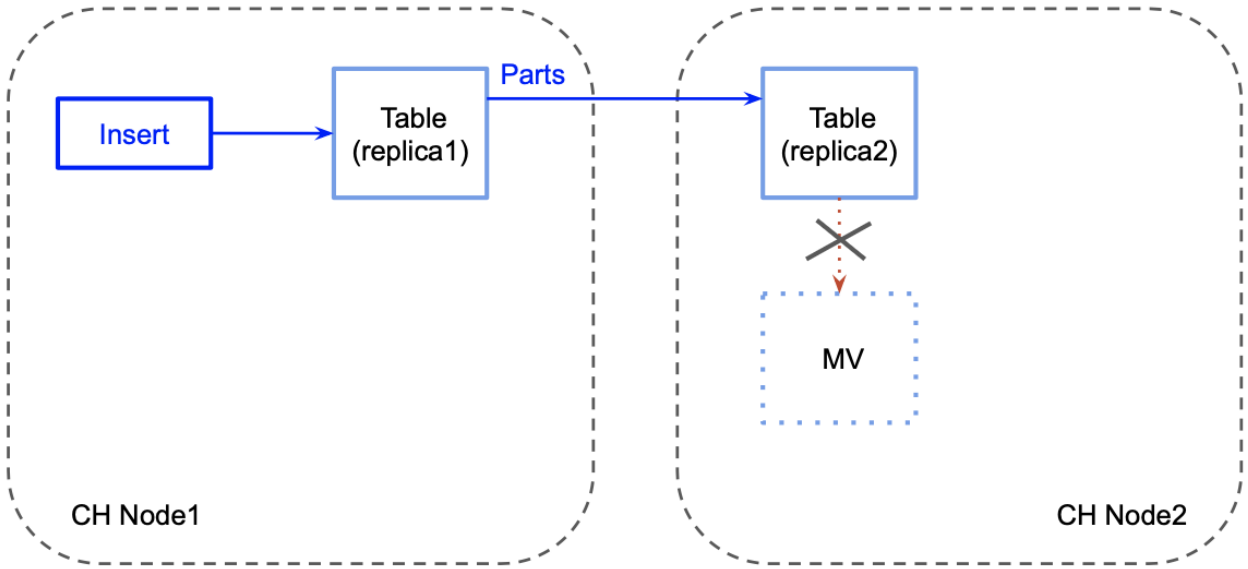
- 不使用 `TO` 创建时，要设置engine，这会创建在inner table
- 使用 `TO` 创建时，engine要设置在dest table中

Replica机制


Insert
Table
(replica1)
Parts
Table
(replica2)
MV
(Table)
(replica1)
Parts
MV
(Table)
(replica2)
CH Node1
CH Node2

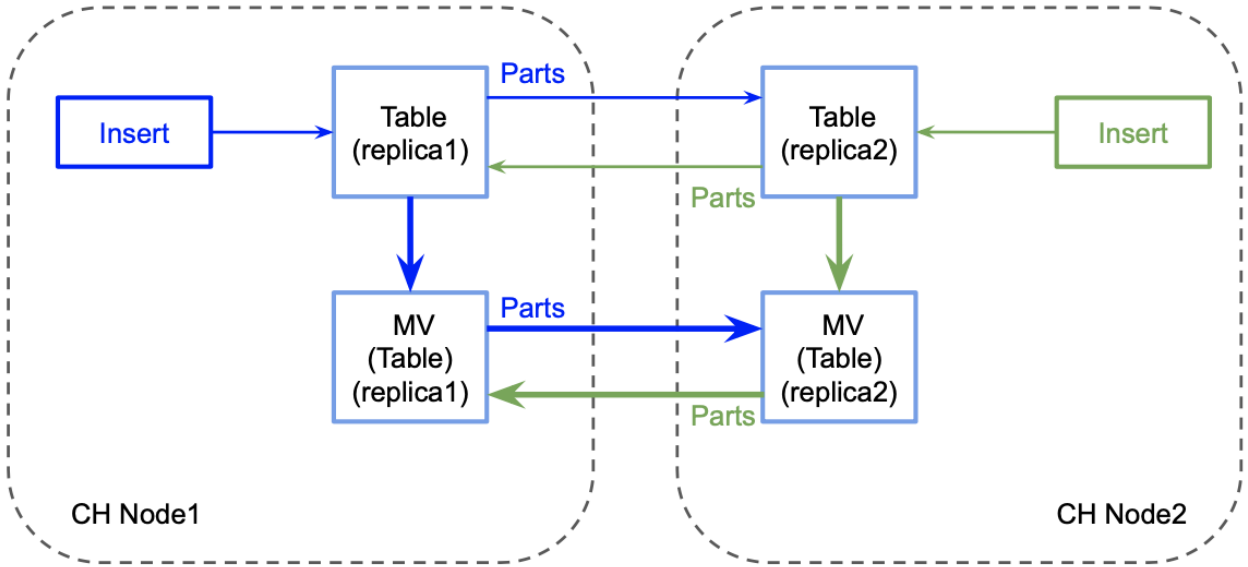
Replica机制

1. 数据写入发生在运行query的node中，写入其中的source table
2. 插入的数据块会发送给其他node中对应的replicated table（例如从replica1发送到replica2）。replica2**不会**从replica1直接读取
3. 在node内，MV从写入source table的数据中获取数据
4. 在创建时，此table使用了replicated engine，因此该table中的插入数据块，会被发送到其他node对应的replicated mv storage table中
5. 每一个数据块是原子的、可去重的（通过checksum）
6. 只有原始数据会进行发送，而不是merge之后的数据，以减少网络使用



Replication与数据的insert没有关系，它使用的数据插入part的 文件，而不是query的log。

完整replicated的MV



更新MV

Implicit table (.inner.mv1)

```
DETACH TABLE mv1

ALTER TABLE `.inner.mv1`
     ADD COLUMN b Int64 AFTER a,
     MODIFY ORDER BY (a, b)

ATTACH MATERIALIZED VIEW mv1
ENGINE = SummingMergeTree
ORDER BY (a, b) AS
SELECT a, b, sum(amount) AS s
FROM source
GROUP BY a, b
```

Explicit table (TO dest)

```
ALTER TABLE dest
     ADD COLUMN b Int64 AFTER a,
     MODIFY ORDER BY (a, b)

DROP TABLE MV1

CREATE MATERIALIZED VIEW mv1
TO dest
SELECT a, b, sum(amount) AS s
FROM source
GROUP BY a, b
```

不停机同步数据到MV

1. 创建MV，在where条件中设置date列大于将来某个日期（一般mv都会包含一个date字段）。
2. 上线并等到到该日期到达后，MV中将开始写入数据
3. 插入该日期之前的数据
4. 在第3步运行完成后， 此MV的数据将完整可用

不停机同步数据到MV

```
CREATE TABLE dest(a Int64, d Date, cnt UInt64)
ENGINE = SummingMergeTree
PARTITION BY toYYYYMM(d) ORDER BY (a, d);

-- create MV c where date >= in_the_future
CREATE MATERIALIZED VIEW mv1 TO dest AS
SELECT a, d, count() AS cnt
FROM source
WHERE d >= '2023-02-14'
GROUP BY a, d;

-- arrives 2023-02-14
INSERT INTO dest -- insert all for before in_the_future
SELECT a, d, count() AS cnt
FROM source
WHERE d < '2023-02-14' -- piece by piece by 1 month (or .. day) GROUP BY a, d;
```

🔥TAKEAWAY

- MV只是一个trigger，将数据存储到一个普通表
- ALWAYS 使用 `TO` 创建MV
- MV不从source 读取数据，也不会因为source table的数据变更而受影响
- MV的select中只处理当次传入的所有数据，而不是source table的所有数据

参考：

- https://den-crane.github.io/Everything_you_should_know_about_materialized_views_commented.pdf
- https://clickhouse.com/docs/zh/sql-reference/statements/create/view/ -->