

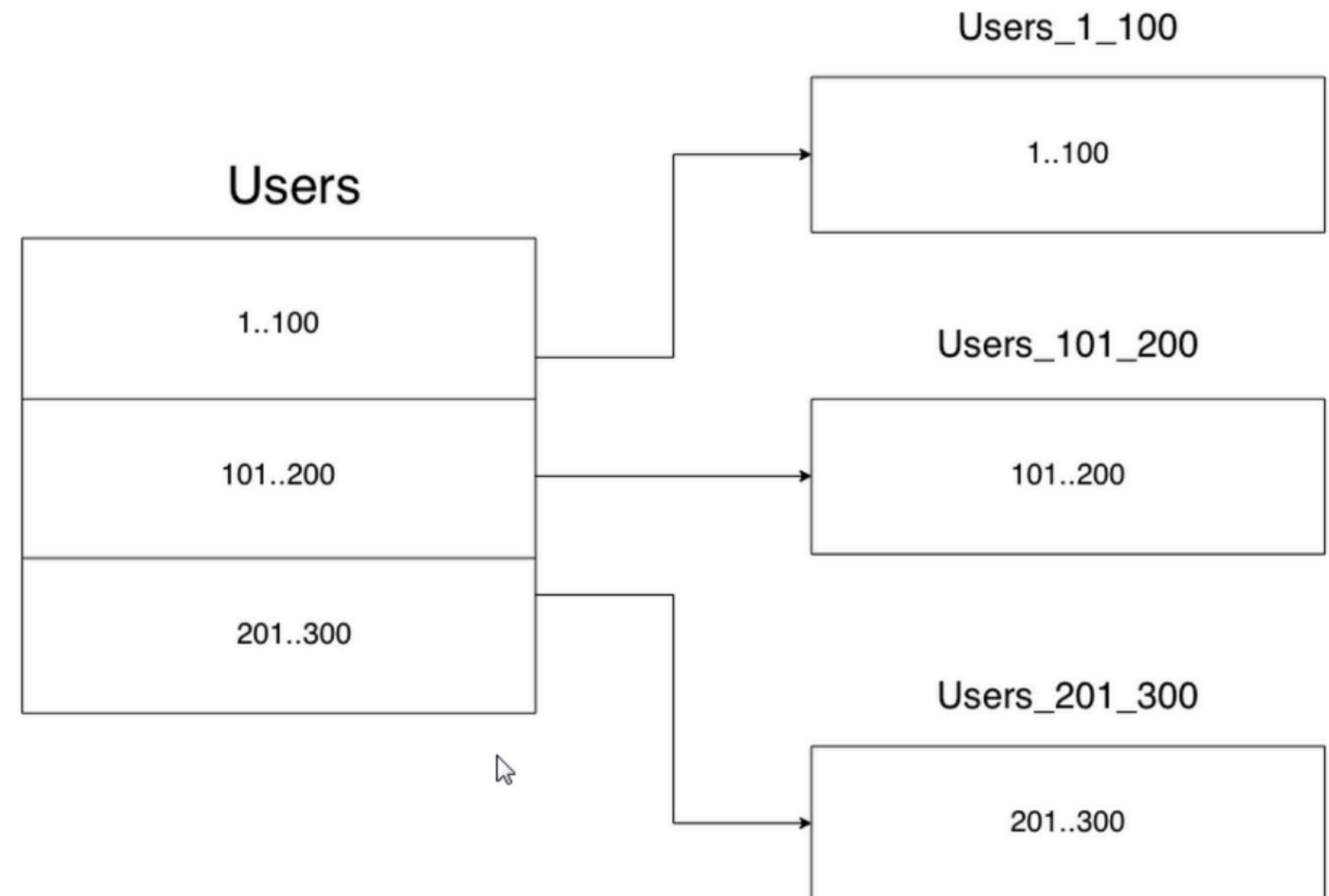


Партицирование. Реплицирование. OLAP. Введение в МРР

Попов Илья,
Москва, 2023

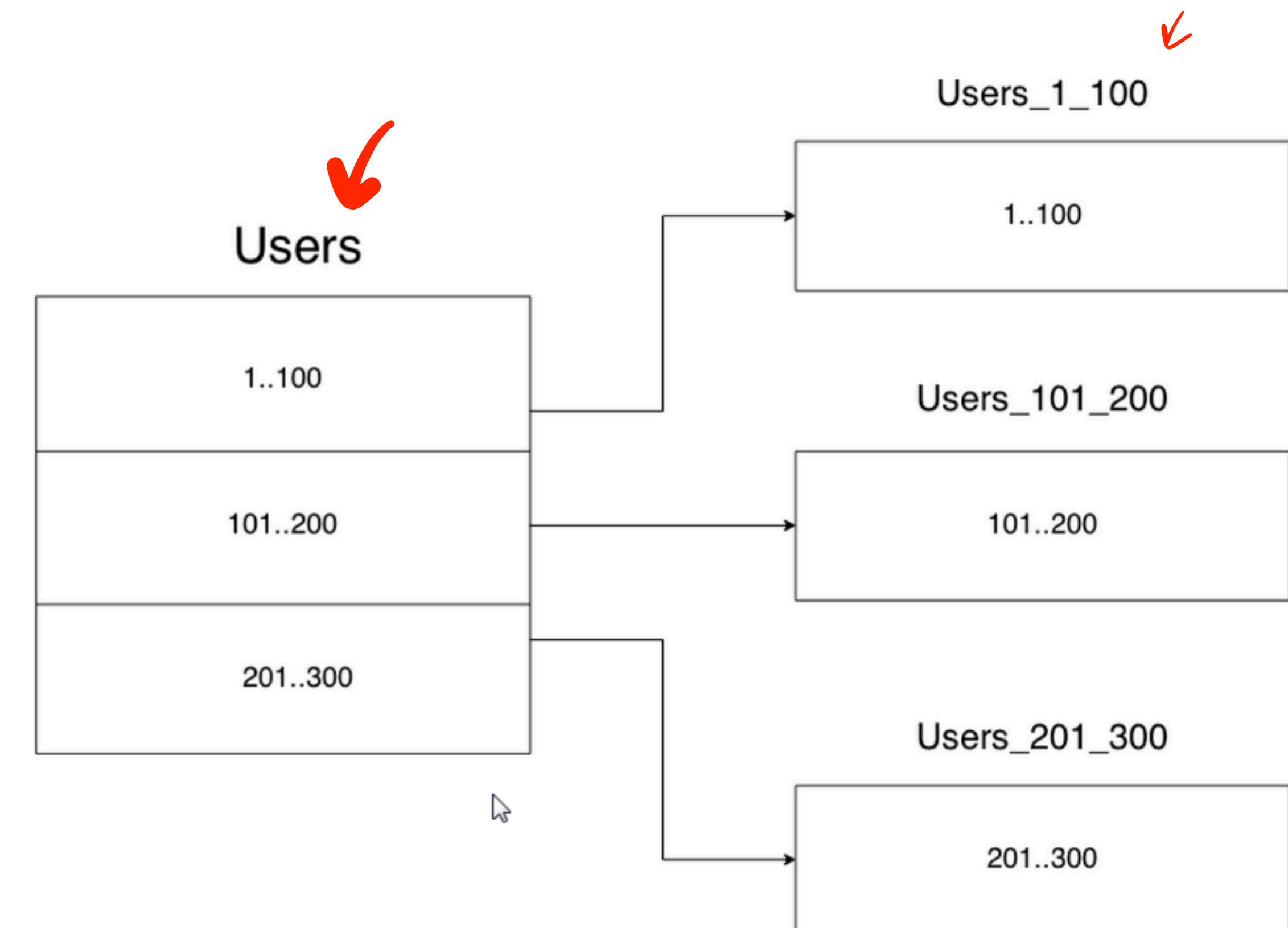
Партицирование.

› Таблица становится слишком большой и громоздкой для обслуживания



Партицирование.

- › Таблица становится слишком большой и громоздкой для обслуживания
- › Записи используется неравномерно
- › Востребованность данных зависит от ключа
- › Скорость чтения и записи падает, возникают блокировки



Партицирование.

1. Декларативное (при создании таблицы)
 - › Диапазонное

```
CREATE TABLE people_partitioned (
    person_id      SERIAL      PRIMARY KEY,
    first_name     VARCHAR(128) NOT NULL,
    last_name      VARCHAR(128) NOT NULL,
    birthday       DATE        NOT NULL,
    ...
) PARTITION BY RANGE (birthday);

CREATE TABLE people_partitioned_birthdays_1800_to_1850
PARTITION OF people_partitioned
FOR VALUES FROM ('1800-01-01') TO ('1849-12-31');

CREATE TABLE people_partitioned_birthdays_1850_to_1900
PARTITION OF people_partitioned
FOR VALUES FROM ('1850-01-01') TO ('1899-12-31');

CREATE TABLE people_partitioned_birthdays_1900_to_1950
PARTITION OF people_partitioned
FOR VALUES FROM ('1900-01-01') TO ('1949-12-31');

CREATE TABLE people_partitioned_birthdays_1950_to_2000
PARTITION OF people_partitioned
FOR VALUES FROM ('1950-01-01') TO ('1999-12-31');
```

Партицирование.

1. Декларативное (при создании таблицы)

- › Диапазонное
- › По списку

```
CREATE TABLE trafficViolations_p_list(  
    seq_id      TEXT,  
    violation_type TEXT, ,  
    ...  
)  
PARTITION BY LIST (violation_type);
```

```
CREATE TABLE trafficViolations_p_list_warning  
    PARTITION OF trafficViolations_p_list  
    FOR VALUES IN ('Warning');
```

```
CREATE TABLE trafficViolations_p_list_sero  
    PARTITION OF trafficViolations_p_list  
    FOR VALUES IN ('SERO');
```

```
CREATE TABLE trafficViolations_p_list_Citation  
    PARTITION OF trafficViolations_p_list  
    FOR VALUES IN ('Citation');
```

```
CREATE TABLE trafficViolations_p_list_ESERO  
    PARTITION OF trafficViolations_p_list  
    FOR VALUES IN ('ESERO');
```

```
CREATE TABLE trafficViolations_p_list_default  
    PARTITION OF trafficViolations_p_list DEFAULT;
```

Партицирование.

1. Декларативное (при создании таблицы)

- › Диапазонное
- › По списку
- › По хэшу

```
CREATE TABLE trafficViolations_p_hash(  
    seqid      TEXT,  
    councils   SMALLINT,  
    ...  
)  
PARTITION BY HASH(councils);
```

- CREATE TABLE trafficViolations_p_hash_p1
 PARTITION OF trafficViolations_p_hash
 FOR VALUES WITH (MODULUS 5, REMAINDER 0);

```
CREATE TABLE trafficViolations_p_hash_p2  
    PARTITION OF trafficViolations_p_hash  
    FOR VALUES WITH (MODULUS 5, REMAINDER 1);
```

```
CREATE TABLE trafficViolations_p_hash_p3  
    PARTITION OF trafficViolations_p_hash  
    FOR VALUES WITH (MODULUS 5, REMAINDER 2);
```

```
CREATE TABLE trafficViolations_p_hash_p4  
    PARTITION OF trafficViolations_p_hash  
    FOR VALUES WITH (MODULUS 5, REMAINDER 3);
```

```
CREATE TABLE trafficViolations_p_hash_p5  
    PARTITION OF trafficViolations_p_hash  
    FOR VALUES WITH (MODULUS 5, REMAINDER 4);
```

Партицирование.

1. Декларативное (при создании таблицы)

- › Диапазонное
- › По списку
- › По хэшу

2. Через наследование

- › Можем комбинировать разные виды
- › Допускается создание дополнительных столбцов в партициях
- › Поддерживает пользовательские методы партицирования

```
CREATE TABLE measurement(  
    city_id    INT NOT NULL,  
    logdate    DATE NOT NULL,  
    peaktemp   INT,  
    unitsales  INT  
); ——————
```

```
CREATE TABLE measurement_y2006m02(  
    CHECK(logdate >= DATE '2006-02-01' AND logdate < DATE '2006-03-01')  
) INHERITS (measurement);
```

```
CREATE TABLE measurement_y2006m03(  
    CHECK(logdate >= DATE '2006-03-01' AND logdate < DATE '2006-04-01')  
) INHERITS (measurement);
```

...

```
CREATE TABLE measurement_y2007m11(  
    CHECK(logdate >= DATE '2007-11-01' AND logdate < DATE '2007-12-01')  
) INHERITS (measurement);
```

```
CREATE TABLE measurement_y2007m12(  
    CHECK(logdate >= DATE '2007-12-01' AND logdate < DATE '2008-01-01')  
) INHERITS (measurement);
```

```
CREATE TABLE measurement_y2008m01(  
    CHECK(logdate >= DATE '2008-01-01' AND logdate < DATE '2008-02-01')  
) INHERITS (measurement);
```

...

```
CREATE TABLE measurement_city_id1(  
    CHECK(city_id = 1)  
) INHERITS (measurement);
```

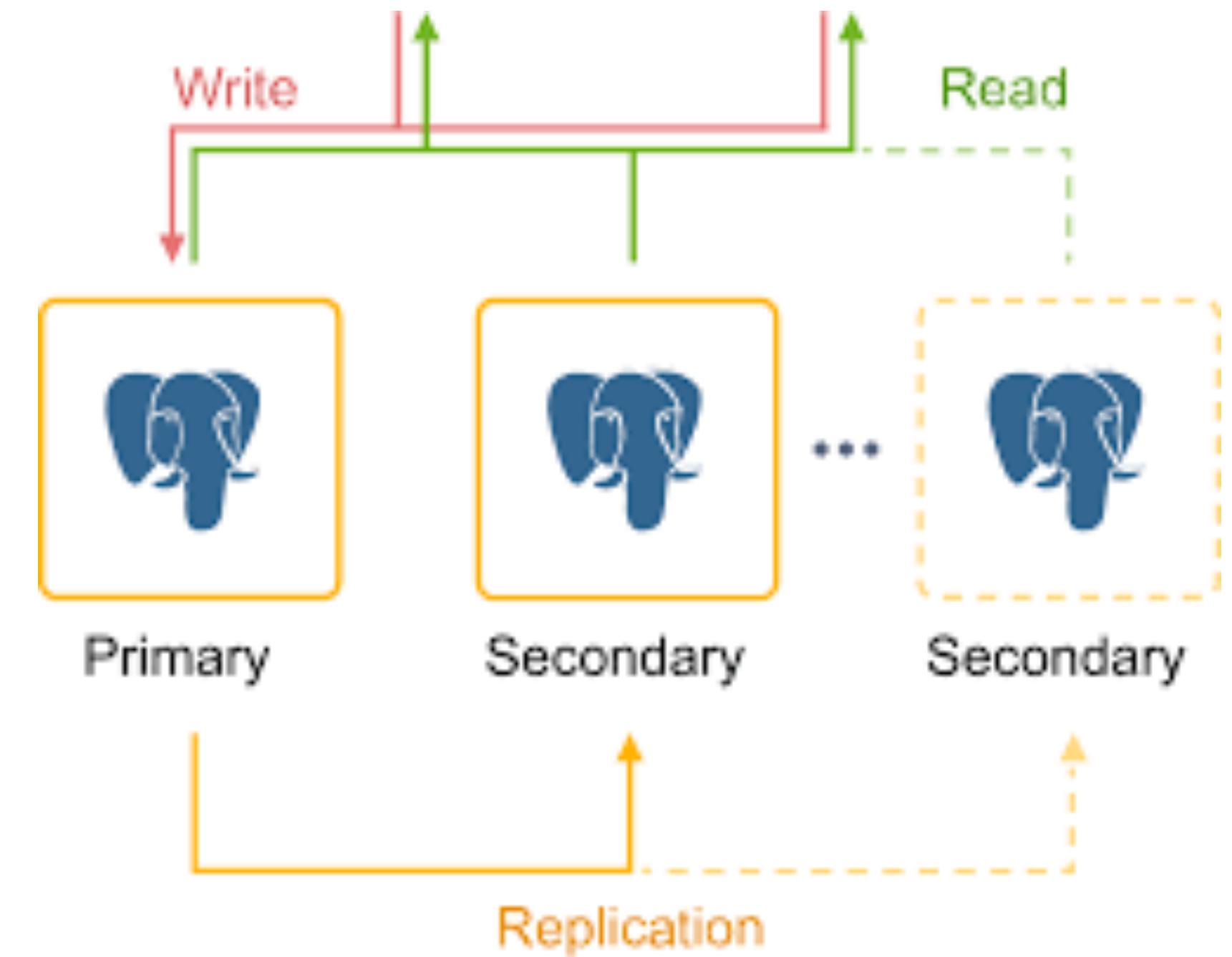
Демо

Репликация

Репликация - копирование данных с одного хоста на другой.

Для чего нужна репликация:

- › Балансировка и отказоустойчивость
- › Оптимизация нагрузки. Например, выделение отдельных реплик под аналитические запросы;



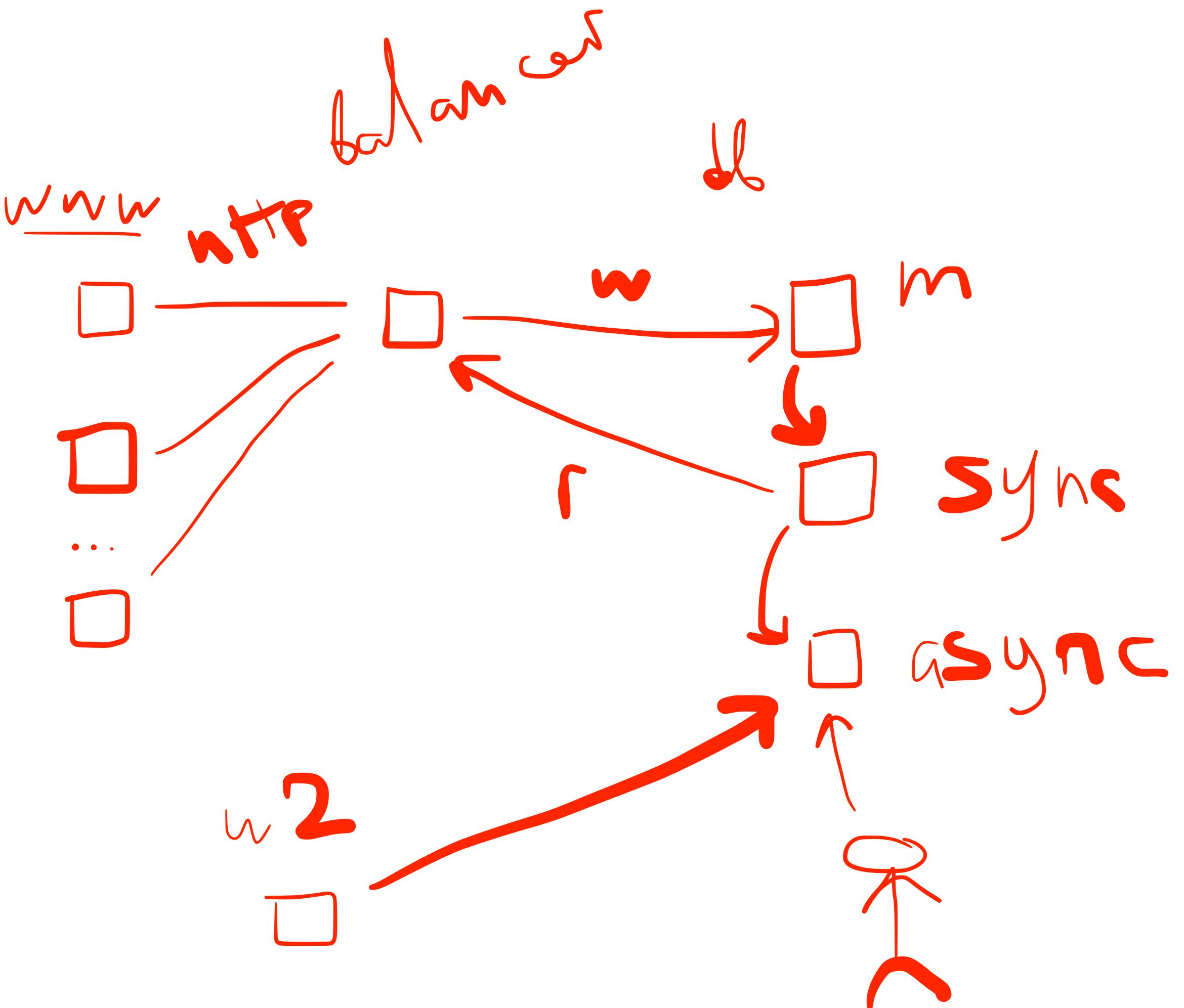
Репликация

Какие бывают ноды:

- › Master
- › Slave

Какие бывают реплики:

- › Синхронные
- › Асинхронные



Демо

CDC

| CDC - Change Data Capture

Подход, при котором записываются и переносятся только изменения, внесенные транзакциями.

CDC позволяет осуществлять репликацию не только между СУБД одного типа, но и между различными СУБД.

Один из самых популярных тулов для CDC - **debezium**.
Хороший гайд по [CDC](#).



debezium

OLTP (OnLine Transactional Processing)

OLTP (OnLine Transactional Processing)

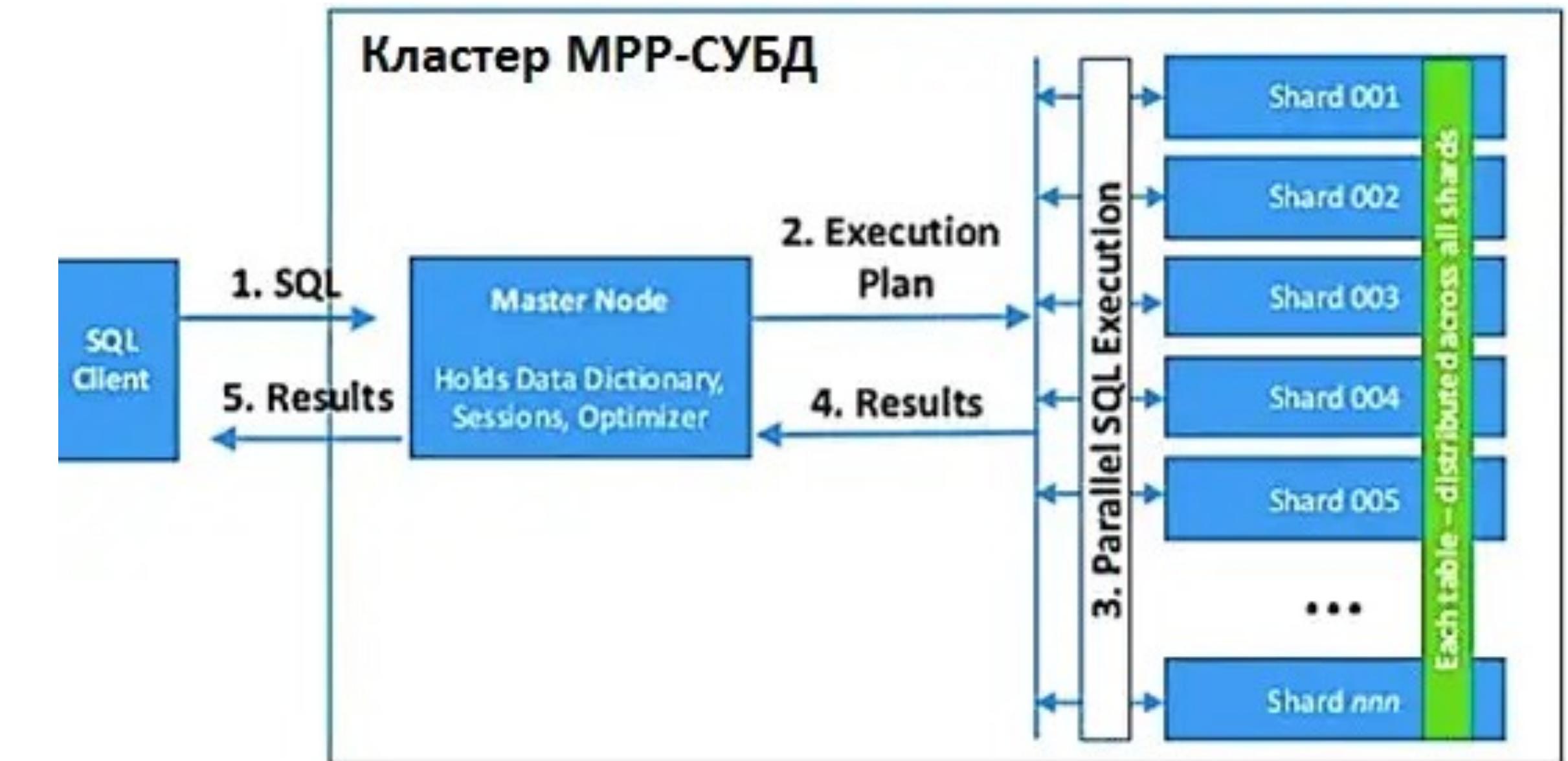
OLAP (OnLine Analytical Processing)

MPP

Мы хотим создать распределенную СУБД для эффективного хранения и обработки данных.

Какие тут могут быть подходы:

- › SE - Shared Everything
- › SD - Shared Disks
- › SN - Shared Nothing
- › CD - Clustered Disk
- › CE - Clustered Everything

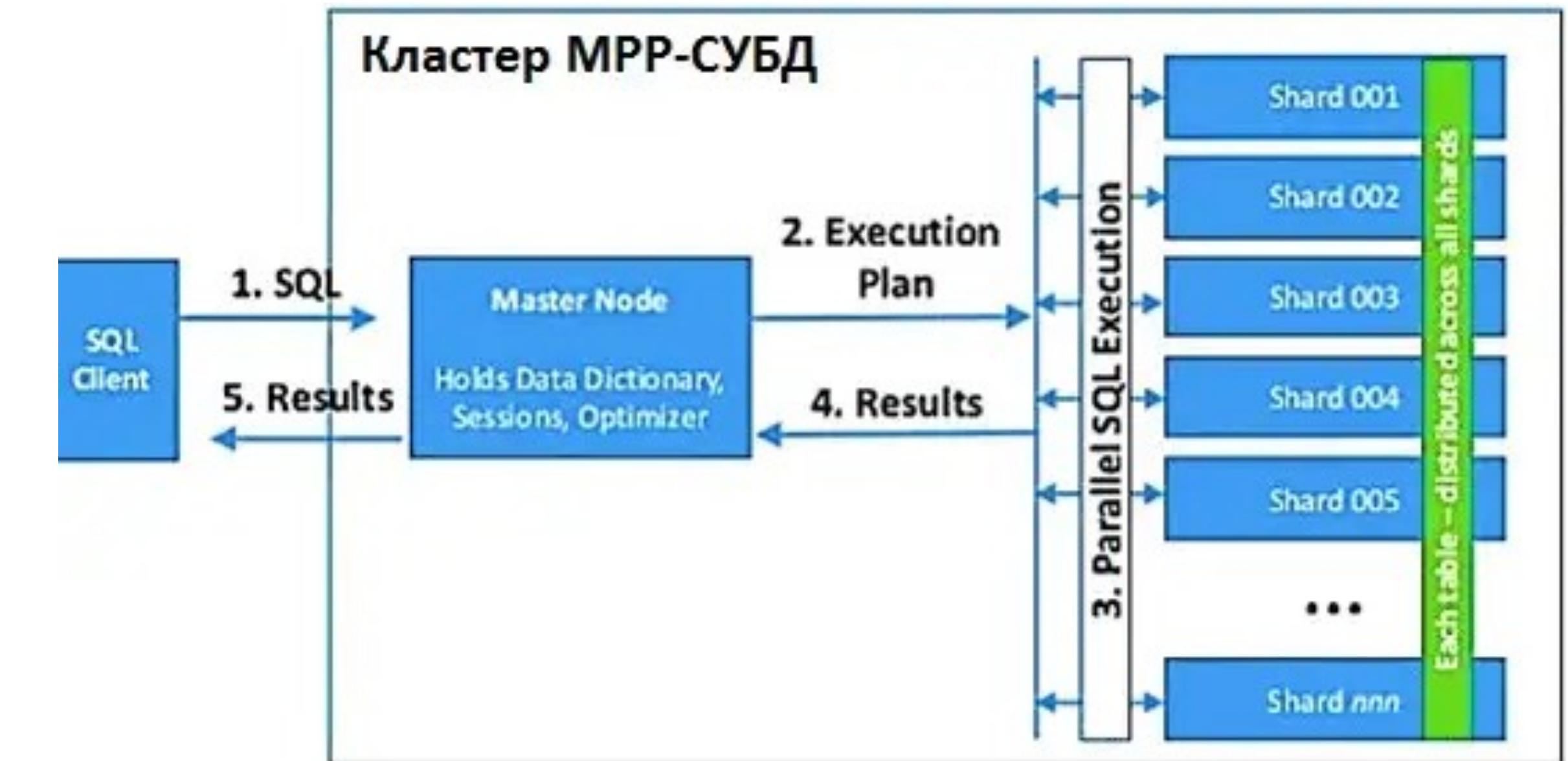


MPP massive parallel processing) –
массивно-параллельная архитектура.

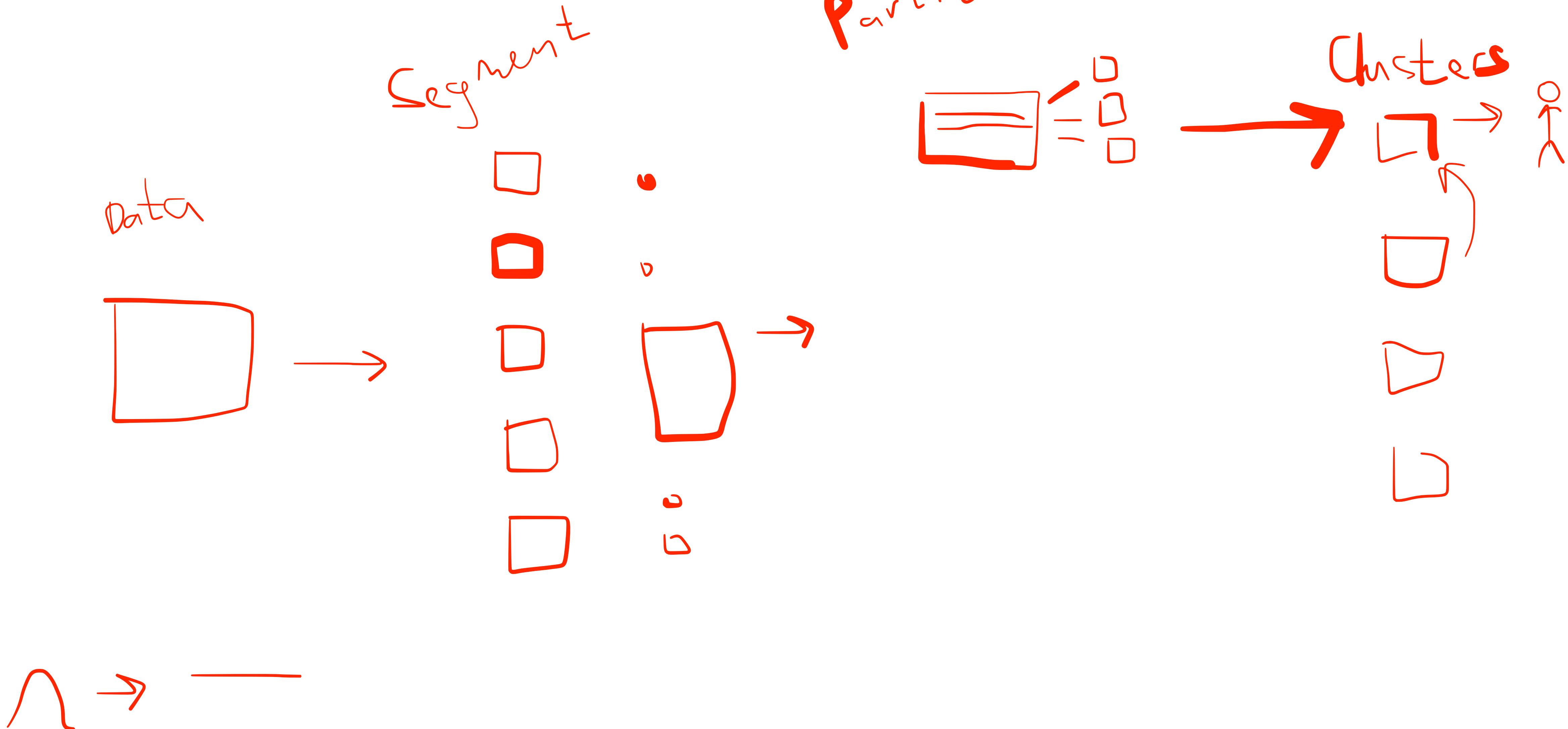
MPP

Главная особенность такой архитектуры состоит в том, что память физически разделена.

Данные бьются по кусочкам, хранятся и обрабатываются на разных машинах, при необходимости обмениваются между собой данными.

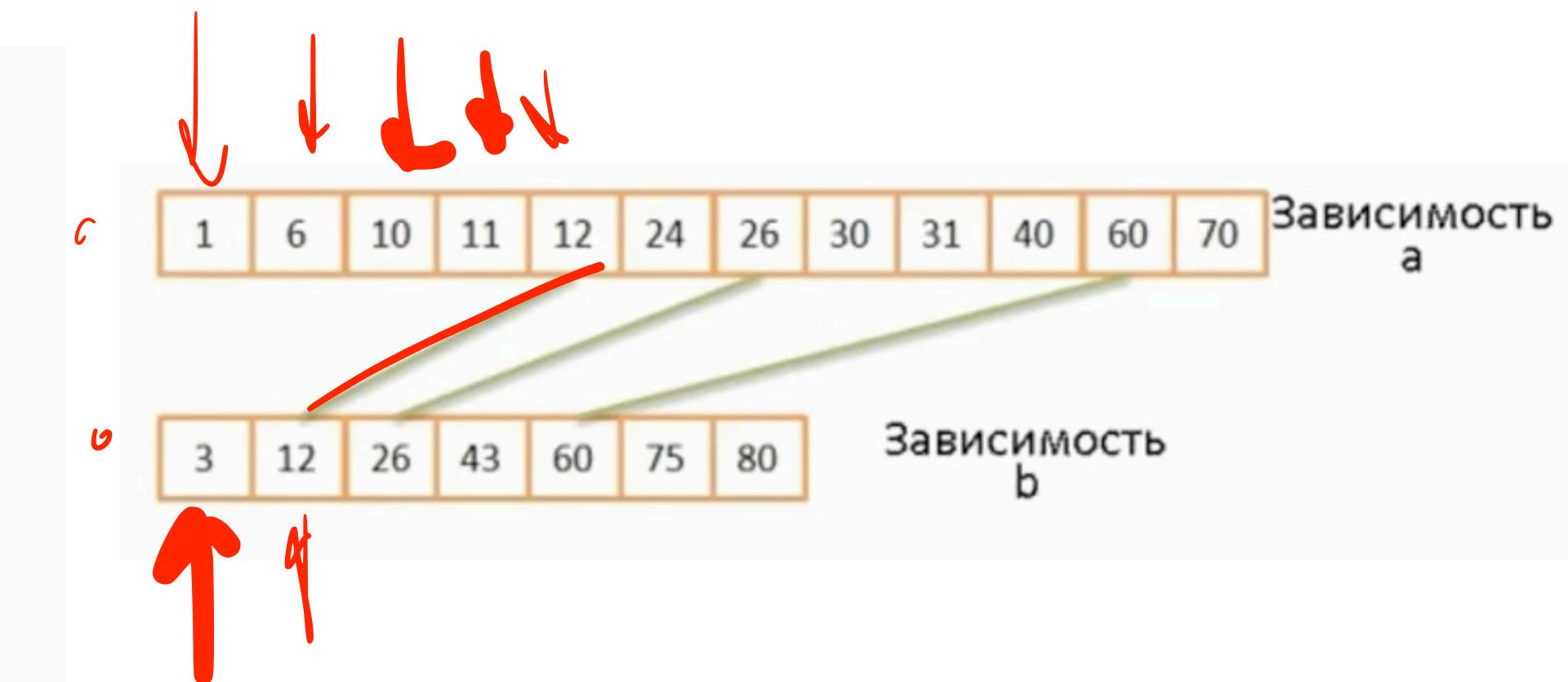
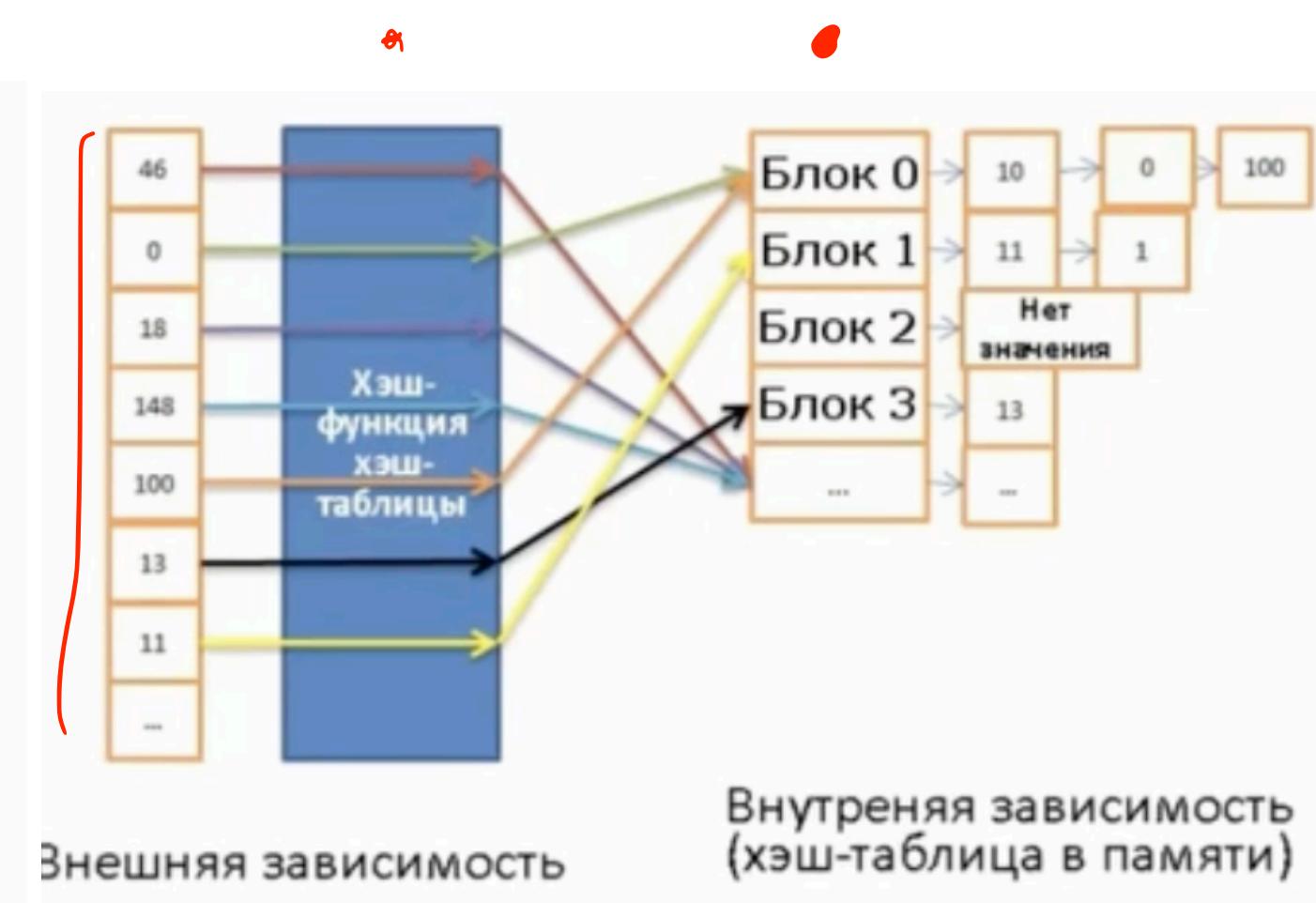
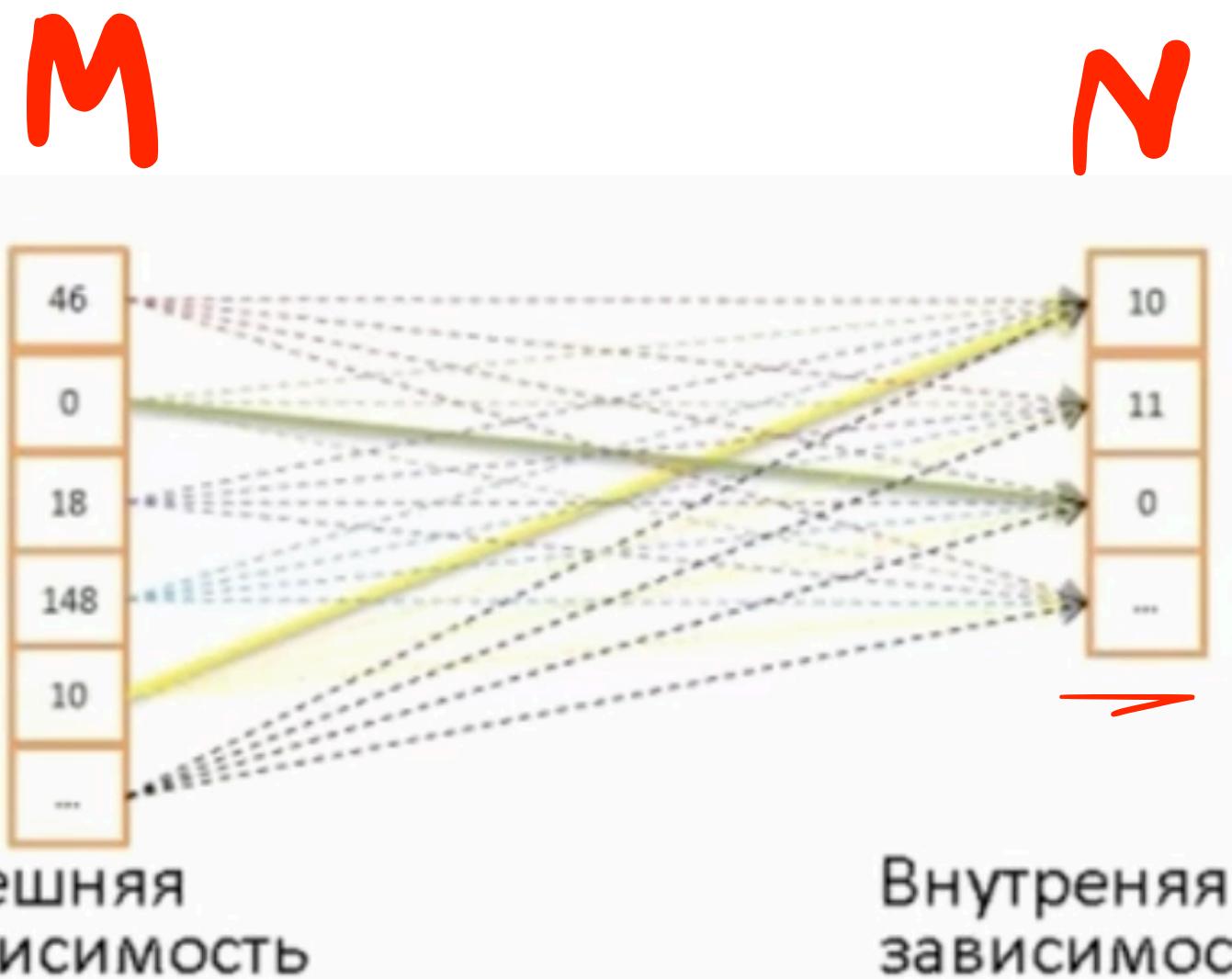


MPP (порисуем)



JOIN в МРР

M \bowtie N



Nested Loop

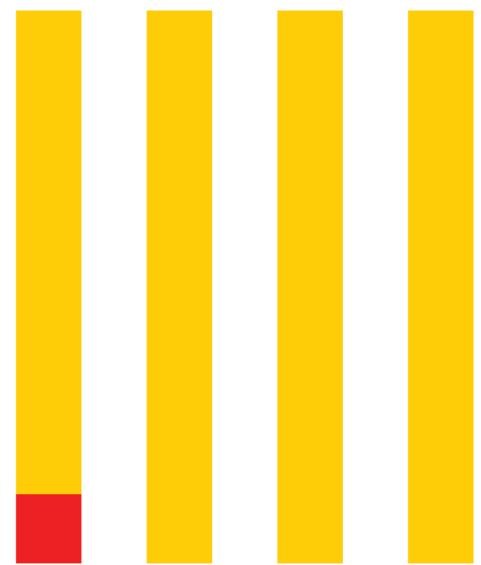
Hash join

Merge join

MPP



GREENPLUM
DATABASE®



ClickHouse

Yandex



OZON
Avito

VERTICA

X
teradata.
BTS

MPP

Во всех таких системах есть способ задать распределение данных по узлам:

- › Teradata - primary index
- › Greenplum - distribution index
- › Vertica - segmentation key

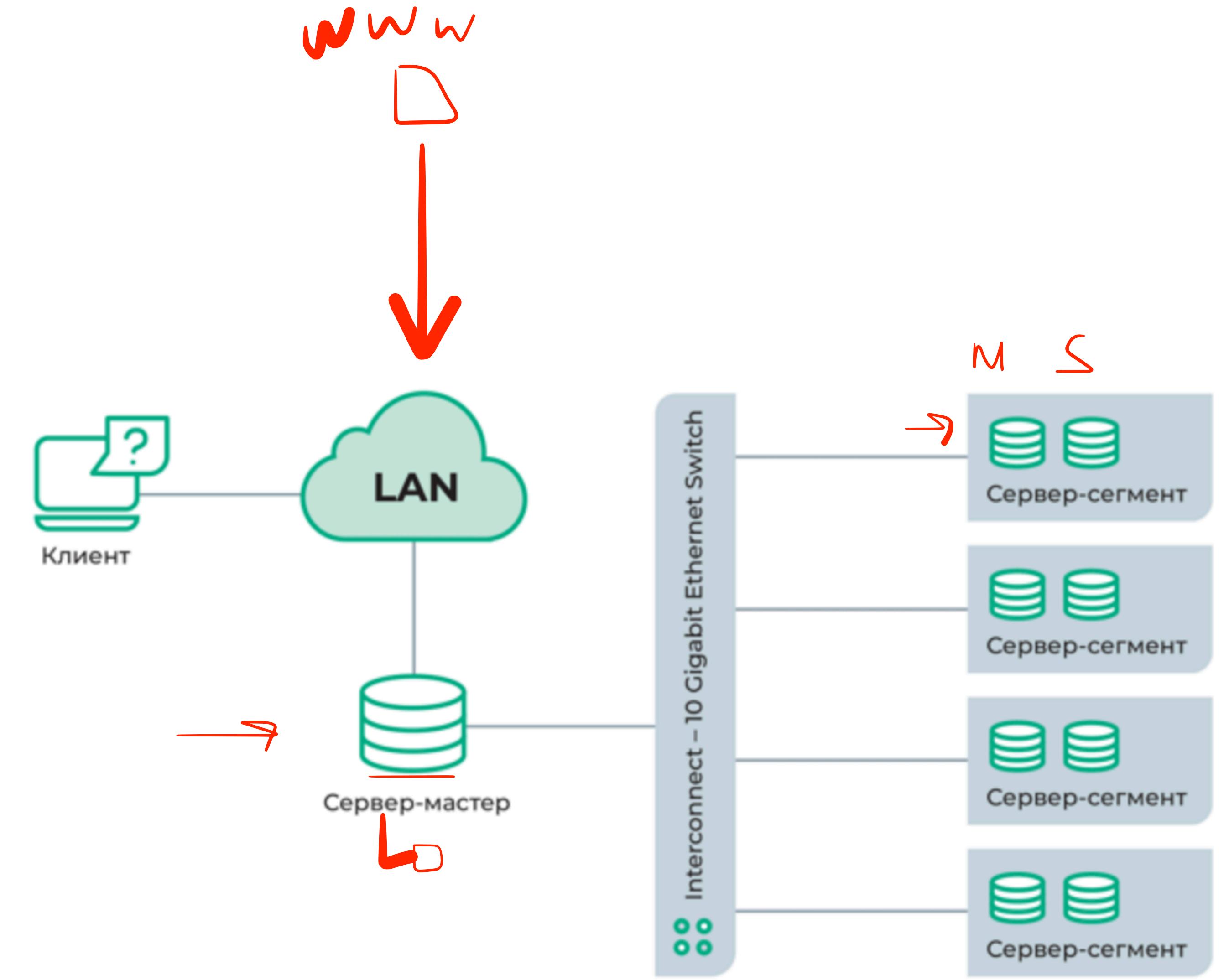
Существует также несколько способов доступа к данным:

- ›  Использование первичного индекса - все MPP
- › Использование вторичного индекса - Teradata
- › Использование проекций - Vertica
- ›  Полное сканирование таблицы - все MPP

Лучший курс по MPP - ШАД Data Warehousing 

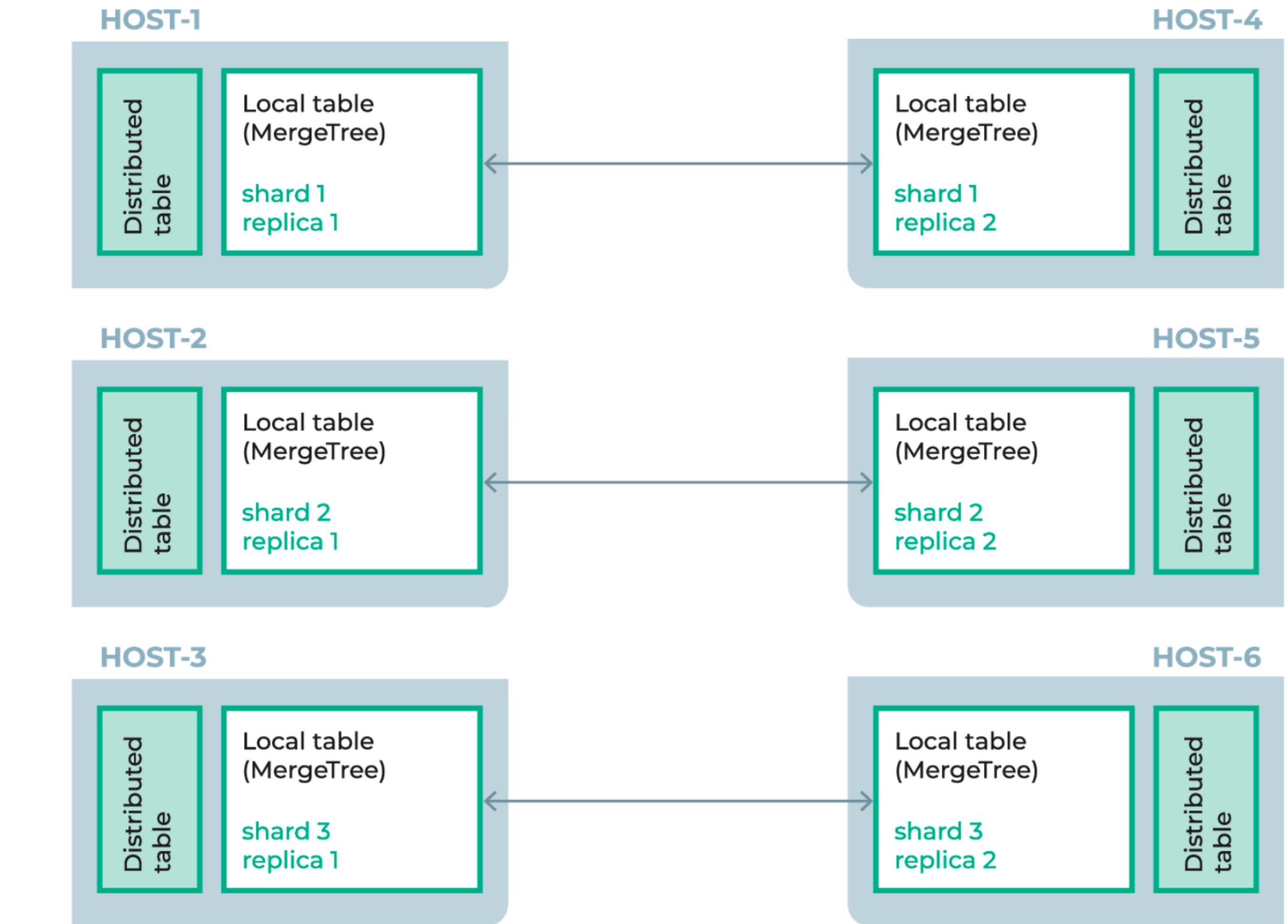
GreenPlum

- › Open Source ✓
- › Под капотом - Postgres ✓
- › Можно делать локальный и распределенный JOIN
- › Полностью соответствует ACID
- › Можно настраивать CDC с Postgres с минимальной обработкой
- › Масштабирование “на горячую” ↴
- › Хорошая статья про



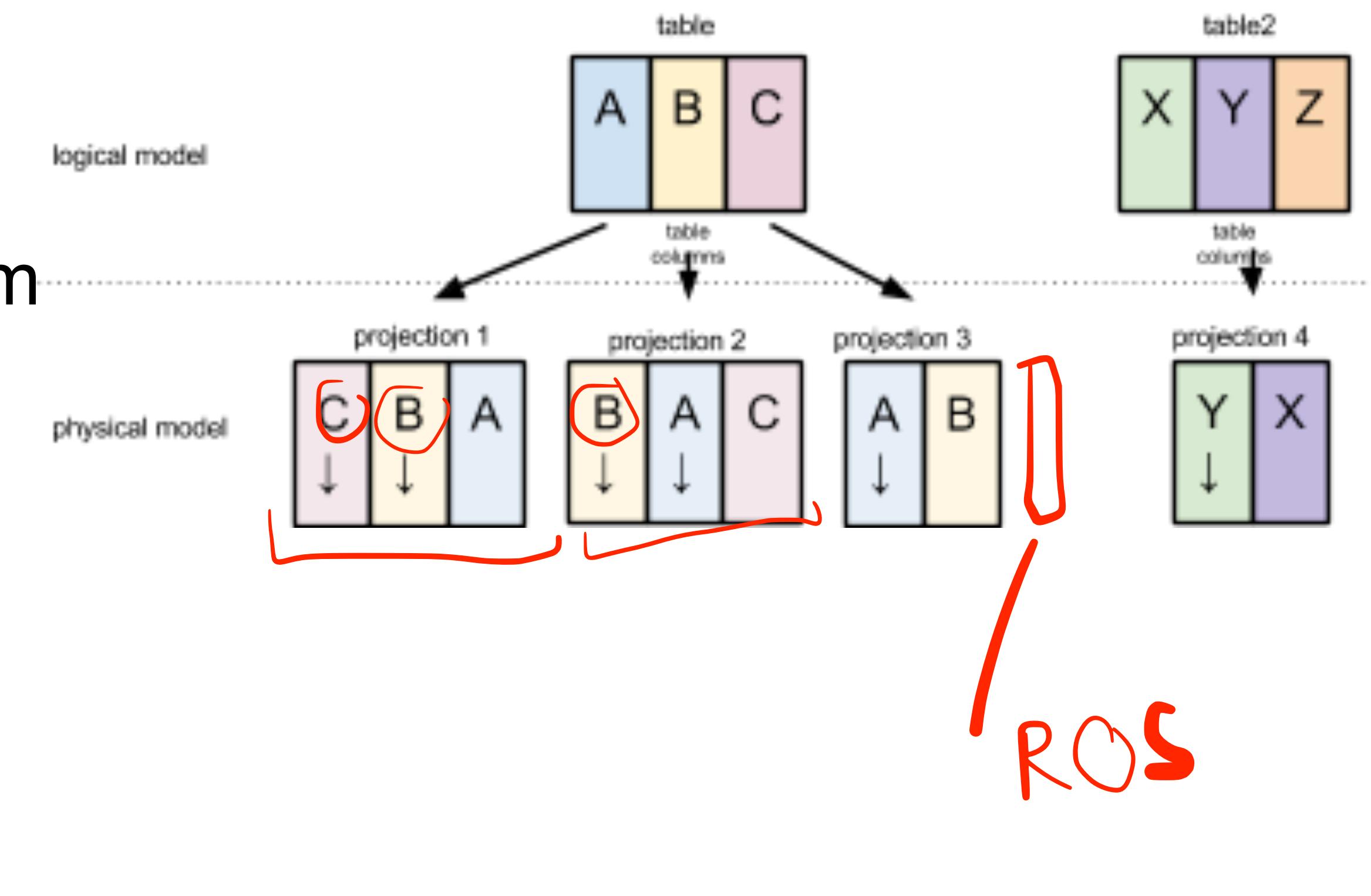
Clickhouse

- › OpenSource ✓
- › Написан в Яндексе на C++ ✓
- › ОФИГЕТЬ какой быстрый
- › Линейная масштабируемость и кросс-ЦОДовая репликация
- › Асинхронная репликация данных✓
- › Крутые алгоритмы сжатия✓
- › Векторные, картежные и массивные операции
- › Читает все, что только возможно прочитать ✓
- › Лучший курс по ClickHouse



Vertica

- › Распространяется вендорами ✗
- › Очень гибкое администрирование
- › МежклUSTERная репликация ✓
- › Шикарный инструмент проекций ✓
- › Интеграции с Kafka, HDFS, Debezium
- › Низкий порог входа ✓
- › INSERT/UPDATE через WOS/ROS память
- › Моя персональная любовь ❤
- › Второй по крутизне курс по Vertica (VPN Only)



Спасибо!

