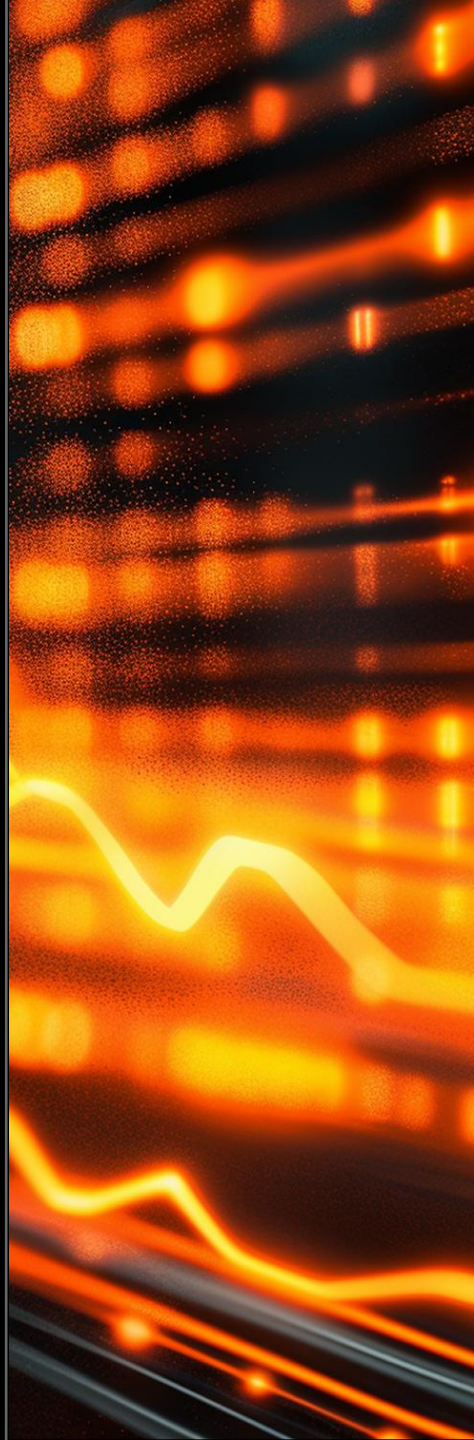
PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Новая жизнь открытого расширения PipelineDB: работа с временными рядами на потоке

Вадим Яценко, «Тантор Лабс»

Алексей Копытов, архитектор проекта



История
и назначение
PipelineDB

Архитектура
PipelineDB
и развитие
в «Тантор Лабс»

Планы
на будущее

История развития PipelineDB

PipelineDB Core – OpenSource

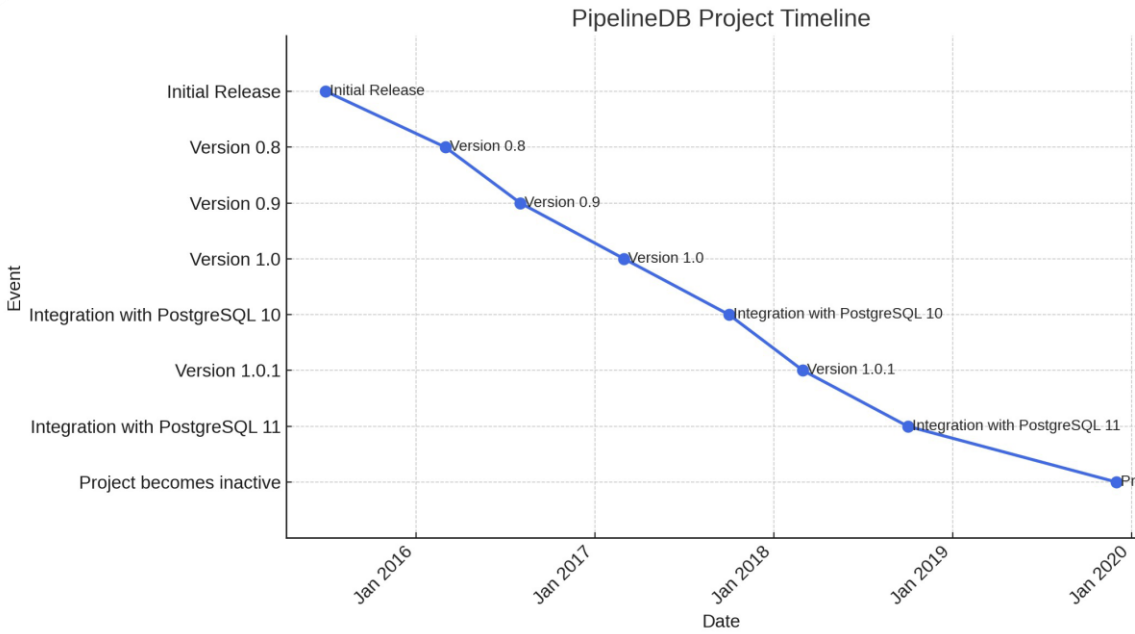
PipelineDB Core в OpenSource
как форк PostgreSQL до 2018 года.

С 2018 года трансформировался в расширение Postgres, это упростило использование.

PipelineDB Cluster для Enterprise

Коммерческое решение
с возможностями кластеризации, HA,
Realtime Alerting, Vacuumless storage
формат, Load Balancing и т. д. Код закрыт.

В 2019 году PipelineDB приобретена Confluent,
с тех пор проект закрыт.



Tantor PipelineDB – новая жизнь старого расширения

PipelineDB – не самое простое расширение

PipelineDB чувствительно к деталям внутреннего устройства PostgreSQL. Изначально было реализовано в виде форка.

Портирование требует значительных усилий

Изменения в методах доступа к таблицам, планировщке и исполнителе запросов PostgreSQL от версии к версии требуют значительных изменений в коде PipelineDB.

	PipelineDB 1.0.0	Tantor PipelineDB 1.3.4
PostgreSQL 11	+	–
PostgreSQL 15	–	+
PostgreSQL 16	–	+
PostgreSQL 17	–	+

Расширение PipelineDB. Зачем?

Нет application-кода

PipelineDB позволяет обрабатывать данные в реальном времени, используя только SQL.

С помощью механизма continuous query планировщик запросов реализует вычисления непрерывно на потоке

ETL внутри БД

PipelineDB — это стандартное расширение PostgreSQL.

PipelineDB предоставляет возможности ETL.

Возможно передавать данные непосредственно в базу данных и непрерывно трансформировать их с помощью SQL-запросов

Сокращение объема данных

PipelineDB хранит только выходные данные непрерывных запросов, которые постепенно обновляются по мере приема данных.

Размер базы данных не зависит от объема данных, обрабатываемых с течением времени

Возможности PipelineDB

Непрерывная агрегация

Возможность использования непрерывные агрегаты, фильтровать и преобразовывать потоковые данные в сводные данные в реальном времени с помощью непрерывных запросов SQL (continuous queries) и сохраняйте результаты в таблицы PipelineDB.

Возможности PipelineDB

Непрерывная агрегация

Возможность использования непрерывные агрегаты, фильтровать и преобразовывать потоковые данные в сводные данные в реальном времени с помощью непрерывных запросов SQL (continuous queries) и сохраняйте результаты в таблицы PipelineDB.

Вероятностные структуры данных

PipelineDB поддерживает структуры данных и алгоритмы, такие как фильтры Bloom filters, count-min sketch, Filtered-Space-Saving top-k, HyperLogLog, and t-digest для очень точных аппроксимаций потоков большого объема.

Возможности PipelineDB

Непрерывная агрегация

Возможность использования непрерывные агрегаты, фильтровать и преобразовывать потоковые данные в сводные данные в реальном времени с помощью непрерывных запросов SQL (continuous queries) и сохраняйте результаты в таблицы PipelineDB.

Вероятностные структуры данных

PipelineDB поддерживает структуры данных и алгоритмы, такие как фильтры Bloom filters, count-min sketch, Filtered-Space-Saving top-k, HyperLogLog, and t-digest для очень точных аппроксимаций потоков большого объема.

Объединение потоков с таблицами

Потоковая передача аналитических данных часто требует контекста.

PipelineDB позволяет объединять потоковые данные с историческими данными для сравнения и анализа в реальном времени.

Возможности PipelineDB

Непрерывная агрегация

Возможность использования непрерывные агрегаты, фильтровать и преобразовывать потоковые данные в сводные данные в реальном времени с помощью непрерывных запросов SQL (continuous queries) и сохраняйте результаты в таблицы PipelineDB.

Объединение потоков с таблицами

Потоковая передача аналитических данных часто требует контекста.

PipelineDB позволяет объединять потоковые данные с историческими данными для сравнения и анализа в реальном времени.

Вероятностные структуры данных

PipelineDB поддерживает структуры данных и алгоритмы, такие как фильтры Bloom filters, count-min sketch, Filtered-Space-Saving top-k, HyperLogLog, and t-digest для очень точных аппроксимаций потоков большого объема.

Запросы с заданными интервалами

Непрерывные запросы на временных скользящих окнах (1 секунда, 1 минута, 1 день, 30 дней и т. д.) с возможностью сохранять результаты запросов, либо удалять «сырые» данные по истечении времени окна.

Сценарии использования PipelineDB

Мониторинг транспорта

Потоковая обработка данных, сообщаемых датчиками, например, дорожными датчиками и светофорами. Отображение непрерывно изменяющихся условий дорожного движения в реальном времени (потоки, колебания трафика). Использование для динамического реагирования на события (ДТП).

Гидрологический мониторинг

Мониторинг потока данных датчиков. отображения изменений качества воды (например, когда вода загрязняется).

Реагирование на события может запускаться динамически.

Интернет транспортных средств (IoV)

PipelineDB может работать с PostGIS для отслеживания местоположений транспортных средств в реальном времени, объединения траекторий и динамического рисования панели.

(разделение времени и региональное распределение транспортных средств).

Мониторинг сетей

Анализа трафика на уровне сетевого протокола в офисных сетях, шлюзах операторов и некоторых серверах. Все, что требует быстрого реагирования, статистики в реальном времени, отслеживания конверсий и событий, может быть объединено с уведомлениями, моделями подписки или кроссплатформенного обмена данными.

Сценарии использования PipelineDB

Мониторинг транспорта

Потоковая обработка данных, сообщаемых датчиками, например, дорожными датчиками и светофорами. Отображение непрерывно изменяющихся условий дорожного движения в реальном времени (потоки, колебания трафика). Использование для динамического реагирования на события (ДТП).

Гидрологический мониторинг

Мониторинг потока данных датчиков. отображения изменений качества воды (например, когда вода загрязняется).

Реагирование на события может запускаться динамически.

Интернет транспортных средств (IoV)

PipelineDB может работать с PostGIS для отслеживания местоположений транспортных средств в реальном времени, объединения траекторий и динамического рисования панели.

(разделение времени и региональное распределение транспортных средств).

Мониторинг сетей

Анализа трафика на уровне сетевого протокола в офисных сетях, шлюзах операторов и некоторых серверах. Все, что требует быстрого реагирования, статистики в реальном времени, отслеживания конверсий и событий, может быть объединено с уведомлениями, моделями подписки или кроссплатформенного обмена данными.

Сценарии использования PipelineDB

Мониторинг транспорта

Потоковая обработка данных, сообщаемых датчиками, например, дорожными датчиками и светофорами. Отображение непрерывно изменяющихся условий дорожного движения в реальном времени (потоки, колебания трафика). Использование для динамического реагирования на события (ДТП).

Гидрологический мониторинг

Мониторинг потока данных датчиков. отображения изменений качества воды (например, когда вода загрязняется).

Реагирование на события может запускаться динамически.

Интернет транспортных средств (IoV)

PipelineDB может работать с PostGIS для отслеживания местоположений транспортных средств в реальном времени, объединения траекторий и динамического рисования панели.

(разделение времени и региональное распределение транспортных средств).

Мониторинг сетей

Анализа трафика на уровне сетевого протокола в офисных сетях, шлюзах операторов и некоторых серверах. Все, что требует быстрого реагирования, статистики в реальном времени, отслеживания конверсий и событий, может быть объединено с уведомлениями, моделями подписки или кроссплатформенного обмена данными.

Сценарии использования PipelineDB

Мониторинг транспорта

Потоковая обработка данных, сообщаемых датчиками, например, дорожными датчиками и светофорами. Отображение непрерывно изменяющихся условий дорожного движения в реальном времени (потоки, колебания трафика). Использование для динамического реагирования на события (ДТП).

Гидрологический мониторинг

Мониторинг потока данных датчиков. отображения изменений качества воды (например, когда вода загрязняется).

Реагирование на события может запускаться динамически.

Интернет транспортных средств (IoV)

PipelineDB может работать с PostGIS для отслеживания местоположений транспортных средств в реальном времени, объединения траекторий и динамического рисования панели.

(разделение времени и региональное распределение транспортных средств).

Мониторинг сетей

Анализ трафика на уровне сетевого протокола в офисных сетях, шлюзах операторов и некоторых серверах. Всё, что требует быстрого реагирования, real-time статистики, отслеживания конверсий и событий, что может быть объединено с уведомлениями, моделями подписки или кроссплатформенного обмена данными.

Кейс использования PipelineDB: платформа Tantor

- Графический инструмент для администрирования и мониторинга кластеров PostgreSQL, более 4 лет на рынке в Production

1 000+
установок
на рынке РФ

5 000+
наблюдаемых инстансов
на один тенант

100+
metric preset-ов, собираемых
по каждому инстансу БД

- Использует PipelineDB для работы с временными рядами на потоке. Оптимизирует объем хранения данных и производительность БД.
- PipelineDB собирает и хранит как метрики хоста и СУБД, так и SQL-запросы с планами выполнения, логи и другую полезную информацию для администратора или разработчика.

Кейс использования PipelineDB: платформа Tantor

- **CONTINUOUS VIEWS + JOINS**
  Для обработки входного потока метрик, соединения с метаданными об объектах и сохранения результата агрегатов.

- **CONTINUOUS TRANSFORMS**
  Для обработки входного потока метрик, соединения с метаданными об объектах и передачи результатов в другой поток/таблицу.

История
и назначение
PipelineDB

Архитектура
PipelineDB
и развитие
в «Тантор Лабс»

Планы
на будущее

Терминология PipelineDB

Streams

Абстракция PipelineDB, которая позволяет клиентам записывать данные временных рядов в **Continuous Views.**

Интерфейс записи данных в потоки идентичен интерфейсу записи в таблицы. Потоки представлены в PipelineDB как внешние таблицы (FDW).

Continuous Views

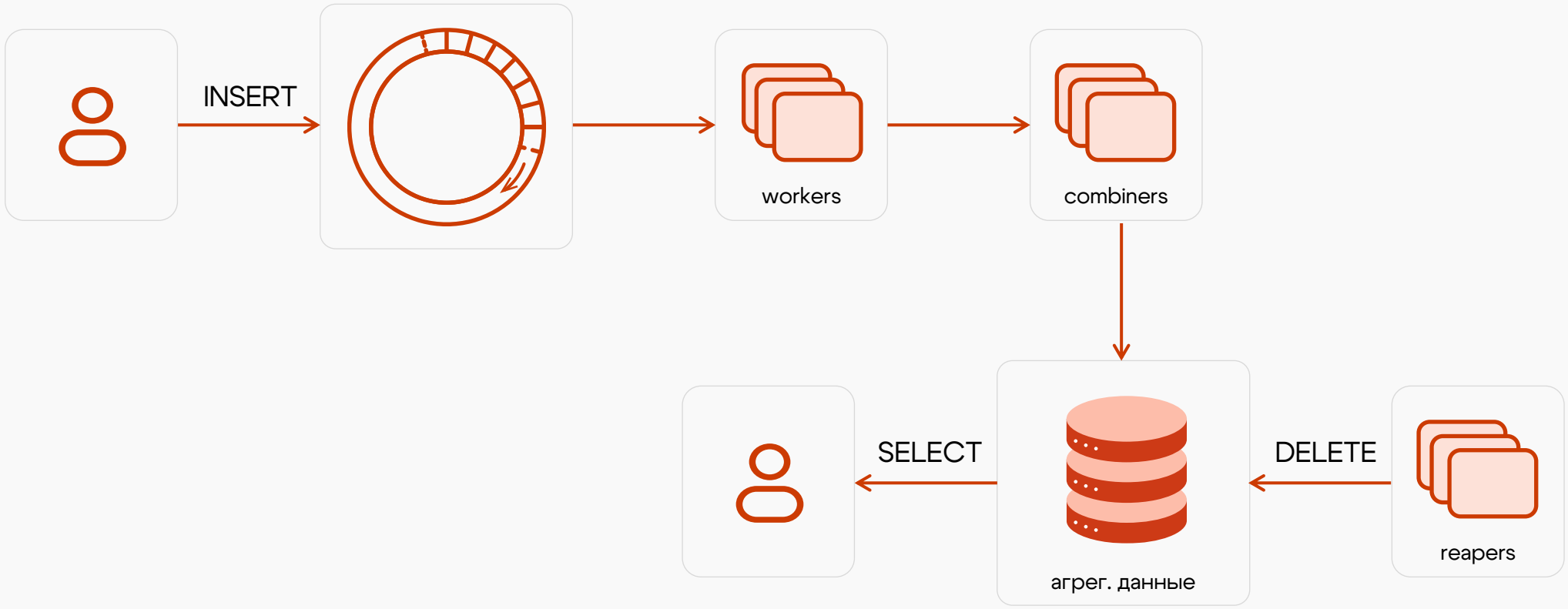
Фундаментальная абстракция PipelineDB, во многом похожая на обычное представление, за исключением того, что оно работает с использованием потоков (streams) и таблиц и постепенно обновляется в реальном времени по мере записи новых данных.

Continuous Transforms

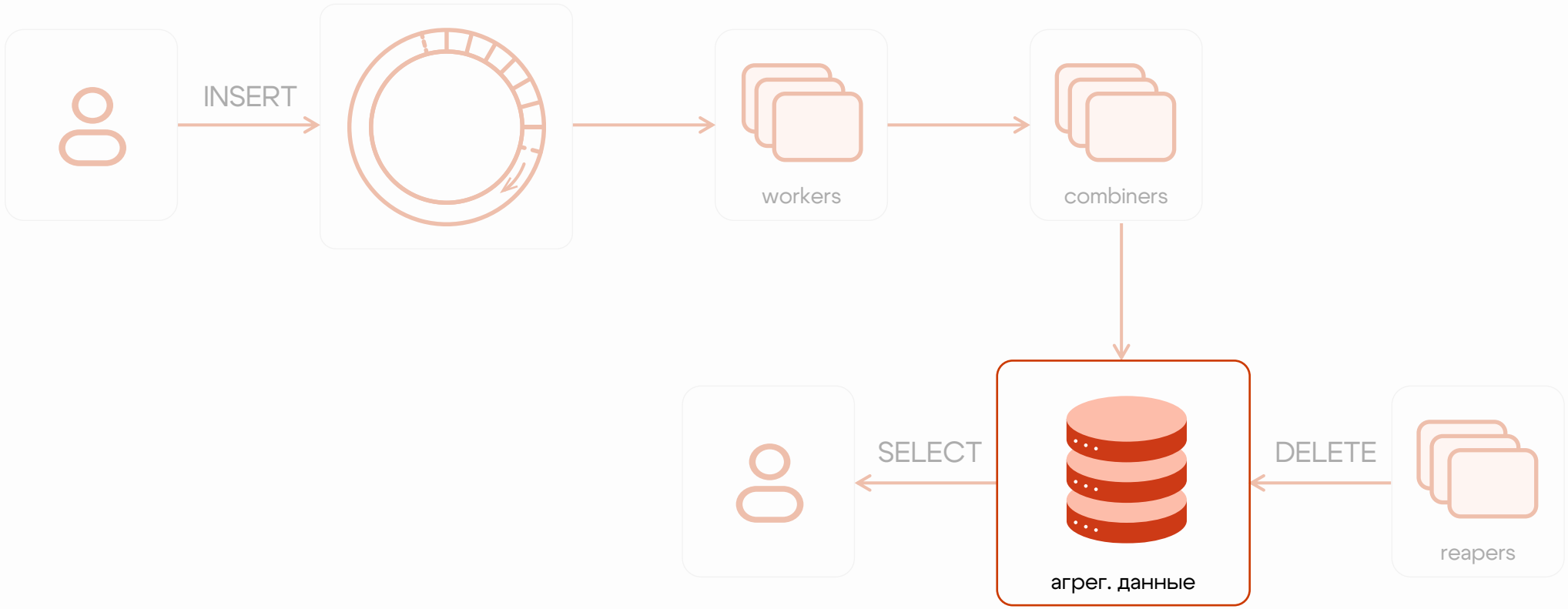
Абстракция PipelineDB для непрерывного преобразования входящих данных временных рядов без их сохранения.

Результат преобразования можно передать в другой поток или записать во внешнее хранилище данных.

Архитектура: общая схема



Архитектура: общая схема

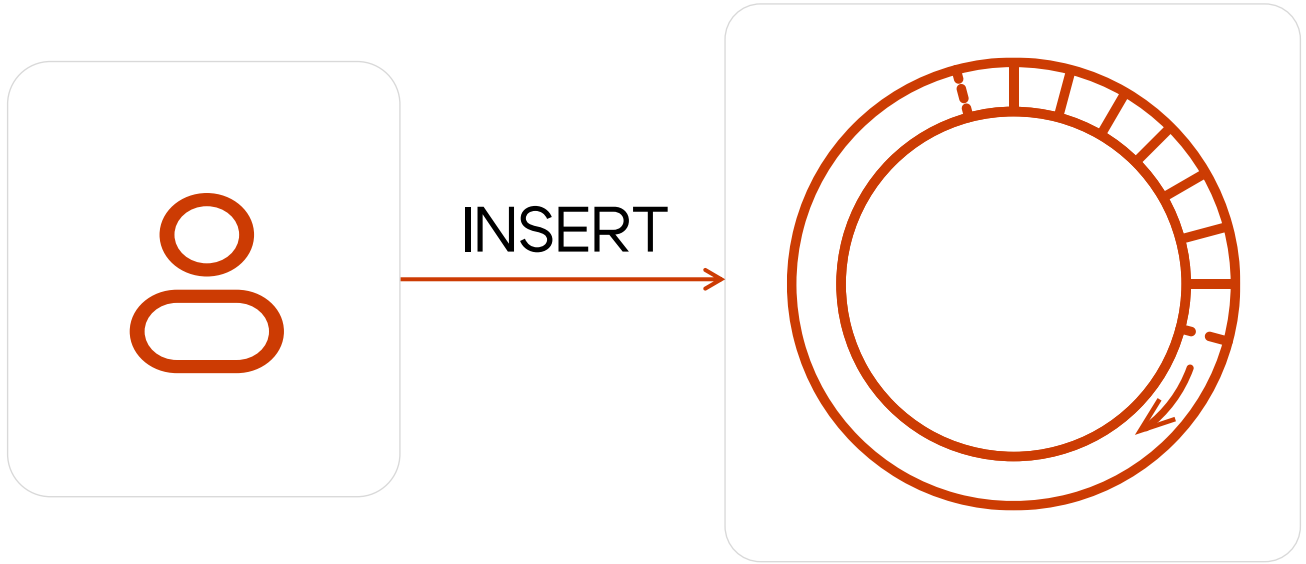


Архитектура: запись в поток

- данные группируются в блоки
- и записываются в кольцевой буфер

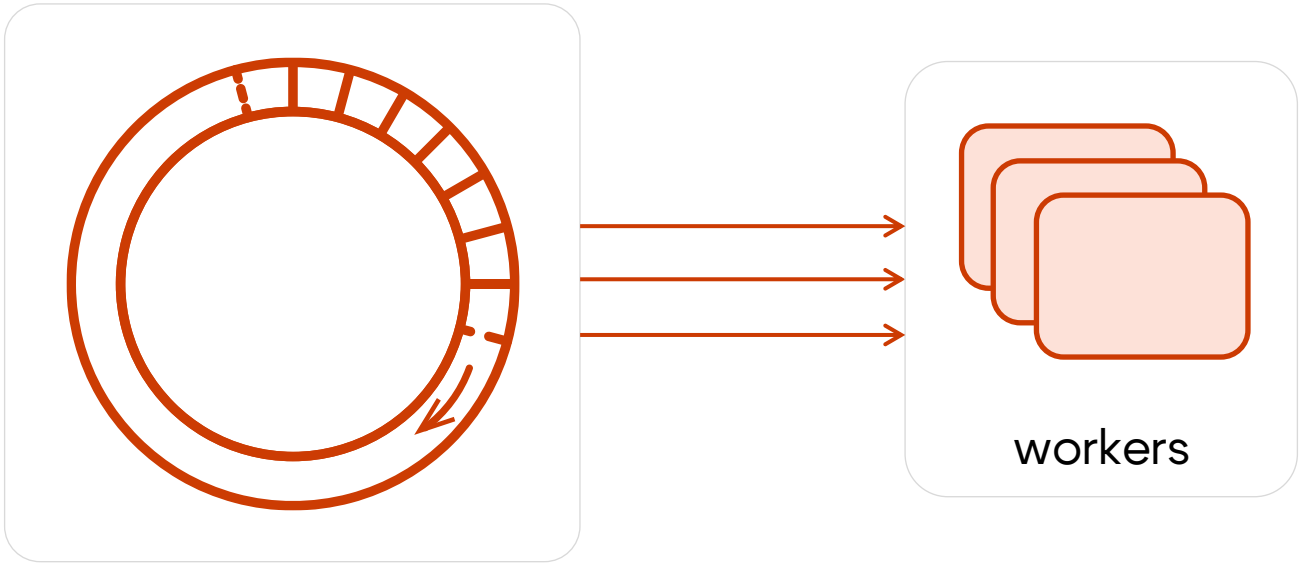
```
CREATE FOREIGN TABLE stream (x INT)
SERVER pipelinedb;
CREATE VIEW v AS
SELECT x, COUNT(*)
FROM stream GROUP BY x;

INSERT INTO stream
VALUES (1), (2), (3);
```



Архитектура: worker-процессы

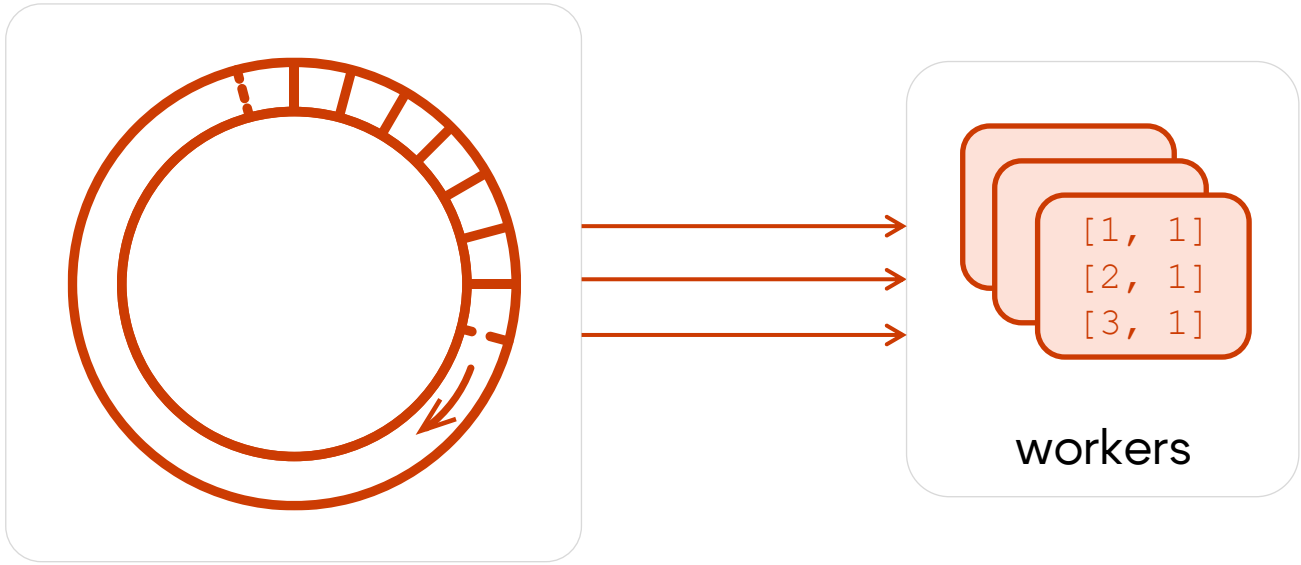
- worker-процессы читают данные из кольцевого буфера
- их задача – подсчитать промежуточные агрегаты для входящих данных



Архитектура: worker-процессы

- результаты отправляются в combiner-процессы

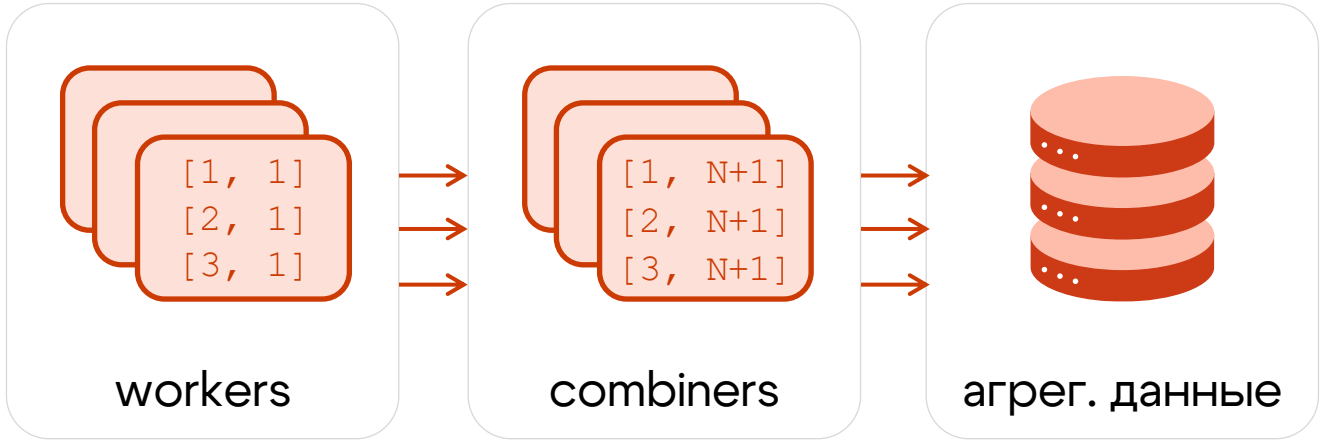
```
SELECT x, COUNT(*) FROM
microbatch GROUP BY X;
```



Архитектура: combiner-процессы

- combiner-процессы комбинируют данные из worker-процессов с текущими данными на диске
- материализованные представления обновляются в соответствии с результатами

```
SELECT x, SUM(count) FROM
  (SELECT * FROM worker
   UNION
   SELECT * FROM existing) s
GROUP BY x;
```



Архитектура: агрегатные функции

```
CREATE VIEW v AS
SELECT avg(y), COUNT(*)
FROM stream GROUP BY x;

INSERT INTO stream(x,y)
VALUES
  (0, 4),
  (1, 1),
  (1, 1);
```

Архитектура: агрегатные функции

```
CREATE VIEW v AS
SELECT avg(y), COUNT(*)
FROM stream GROUP BY x;

INSERT INTO stream(x,y)
VALUES
  (0, 4),
  (1, 1),
  (1, 1);
```

```
SELECT * FROM v_mrel;

 x |  avg  | count
---+-------+-------
 0 | {1,4} |     1
 1 | {2,2} |     2
```

Архитектура: агрегатные функции

```
CREATE VIEW v AS
SELECT avg(y), COUNT(*)
FROM stream GROUP BY x;

INSERT INTO stream(x,y)
VALUES
  (0, 4),
  (1, 1),
  (1, 1);
```

```
SELECT * FROM v_mrel;

  x |  avg   | count
----+--------+-------
  0 | {1,4}  |     1
  1 | {2,2}  |     2
```

transition state

```
\d+ v
                                View "public.v"
 Column |  Type   | Collation | Nullable | Default | Storage | Description
--------+---------+-----------+----------+---------+---------+-------------
 avg    | numeric |           |          |         | main    |
 count  | bigint  |           |          |         | plain   |
View definition:
 SELECT int8_avg(v_mrel.avg) AS avg,
    v_mrel.count
   FROM ONLY v_mrel;
```

finalize function

Архитектура: агрегатные функции

- Агрегатные функции — ключевая функциональность PipelineDB
- Поддерживается большинство агрегатных функций PostgreSQL, но некоторые прозрачно заменяются на реализации, оптимизированные для бесконечных потоков. Например, **`COUNT(DISTINCT)`**
- PipelineDB предоставляет статистические агрегаты

```
bloom_agg ( expression )
freq_agg ( expression )
topk_agg ( expression , k )
hll_agg ( expression )
dist_agg ( expression )
bucket_agg ( expression , bucket_id )
exact_count_distinct ( expression )
first_values ( n ) WITHIN GROUP
keyed_max ( key, value )
keyed_min ( key, value )
set_agg ( expression )
array_agg ( expression )
avg ( expression ) bit_and
( expression ) bit_or (
expression ) bool_and (
expression ) bool_or (
expression ) count ( * )
count ( DISTINCT expression)
count ( expression )
every ( expression )
json_agg ( expression )
json_object_agg ( key, value )
jsonb_agg ( expression )
jsonb_object_agg ( key, value )
max ( expression )
min ( expression
string_agg ( expression, delimiter )
sum ( expression )
```

```
corr ( y, x )
covar_pop ( y, x )
covar_samp ( y, x )
regr_avgx ( y, x )
regr_avgy ( y, x )
regr_count ( y, x )
regr_intercept ( y, x )
regr_r2 ( y, x )
regr_slope ( y, x )
regr_sxx ( y, x )
regr_sxy ( y, x )
regr_syy ( y, x )
stddev ( expression )
stddev_pop ( expression )
variance ( expression )
var_pop ( expression )
percentile_cont ( fraction )
percentile_cont ( array of fractions
rank ( arguments )
dense_rank ( arguments )
percent_rank ( arguments )
```

Архитектура: пользовательские агрегаты

- PipelineDB позволяет использовать пользовательские агрегаты в CV
- нужно определить **combinefunc,** чтобы сделать их комбинируемыми

```
CREATE AGGREGATE
combinable_agg(x)
  sfunc=sfunc, funalfunc=finalfunc,
  combinefunc=my_combinef
  unc
);

CREATE VIEW AS
SELECT x, combinable_agg(y) FROM stream GROUP
BY x;
```

Архитектура: пользовательские агрегаты

- PipelineDB позволяет использовать пользовательские агрегаты в CV
- нужно определить **combinefunc,** чтобы сделать их комбинируемыми

```
CREATE AGGREGATE
combinable_agg(x)
  sfunc=sfunc, funalfunc=finalfunc,
  combinefunc=my_combinef
  unc
);

CREATE VIEW AS
SELECT x, combinable_agg(y) FROM stream GROUP
BY x;
```

- При обновлении материализованного представления:

```
transition_state = my_combinefunc(microbatch_tstate,
                                  existing_tstate)
```

Архитектура: пользовательские агрегаты

- PipelineDB позволяет использовать пользовательские агрегаты в CV
- нужно определить **combinefunc,** чтобы сделать их комбинируемыми

```
CREATE AGGREGATE
combinable_agg(x)
  sfunc=sfunc, funalfunc=finalfunc,
  combinefunc=my_combinef
  unc
);

CREATE VIEW AS
SELECT x, combinable_agg(y) FROM stream GROUP
BY x;
```

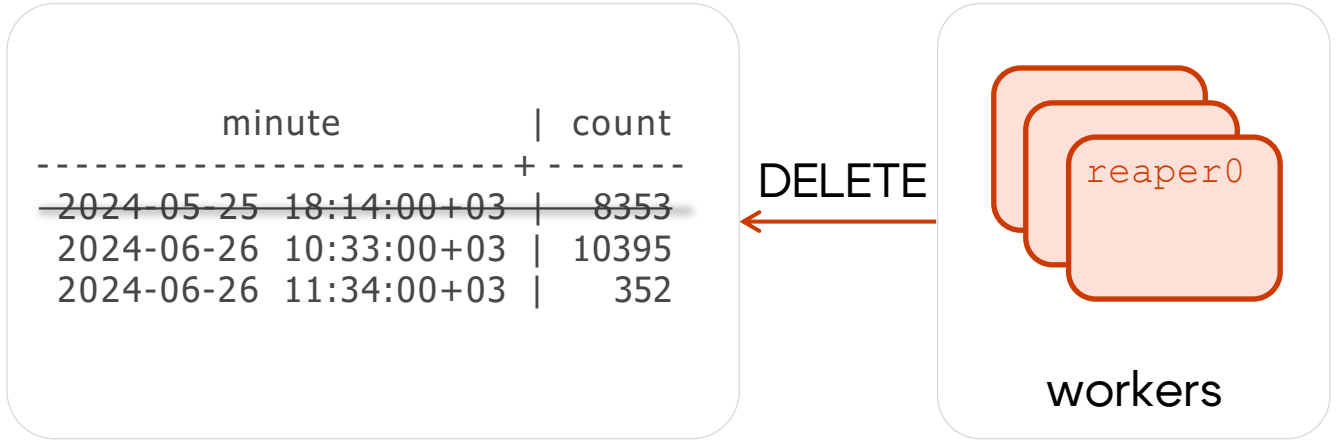
- При обновлении материализованного представления:

```
transition_state = my_combinefunc(microbatch_tstate,
                                  existing_tstate)
```

- При выполнении **SELECT * FROM v**:

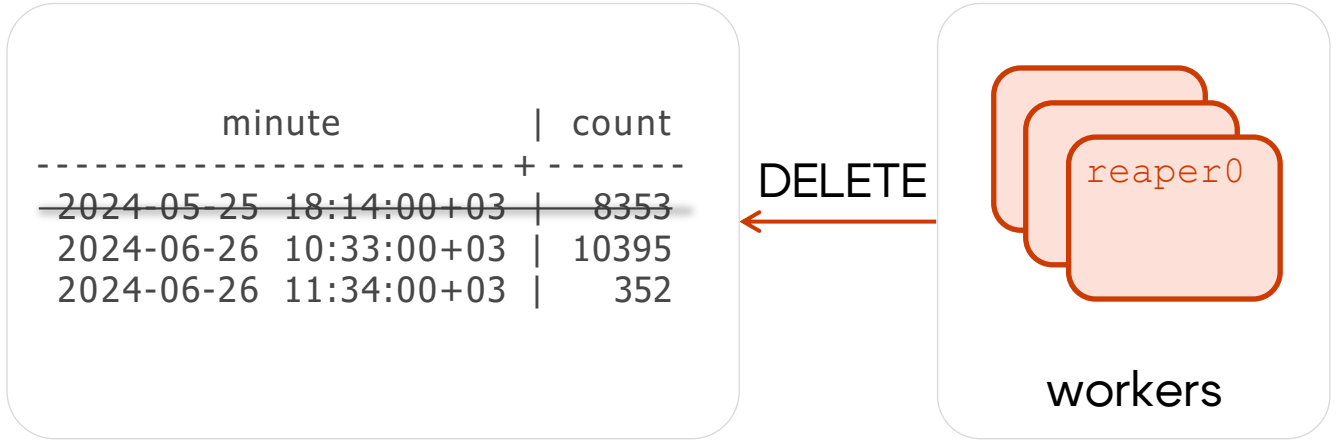
```
agg_value = finalfunc(existing_tstate)
```

Архитектура: reaper-процессы



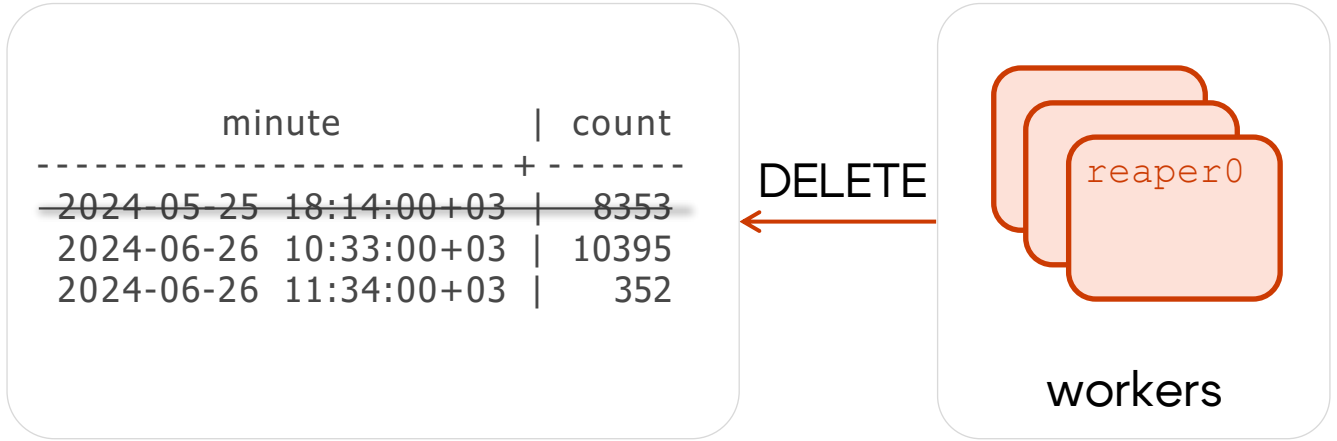
Архитектура: reaper-процессы

- удаляют данные с истёкшим TTL (time-to-live)



Архитектура: reaper-процессы

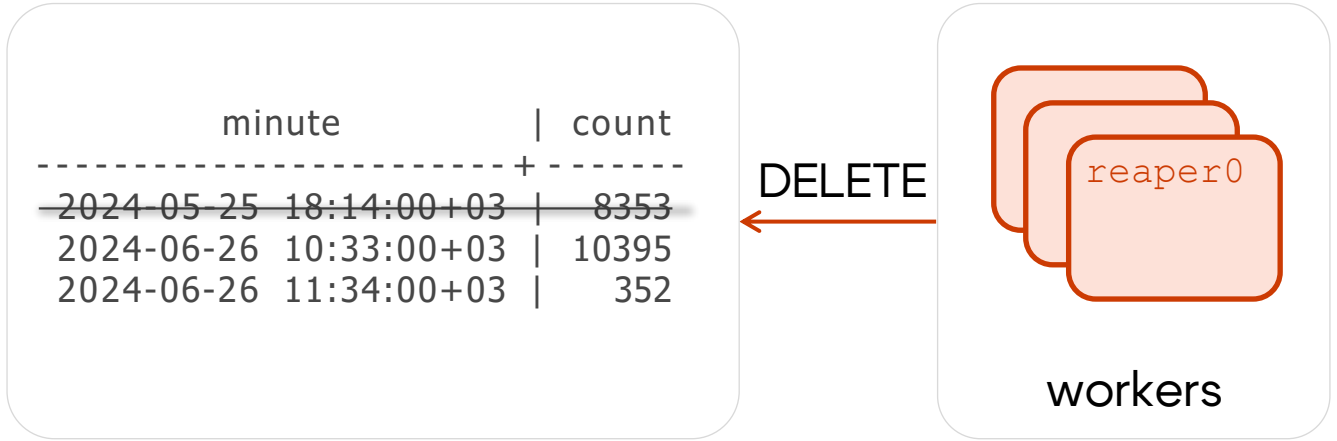
- удаляют данные с истёкшим TTL (time-to-live)
- интенсивность процесса управляется конфигурацией



Архитектура: reaper-процессы

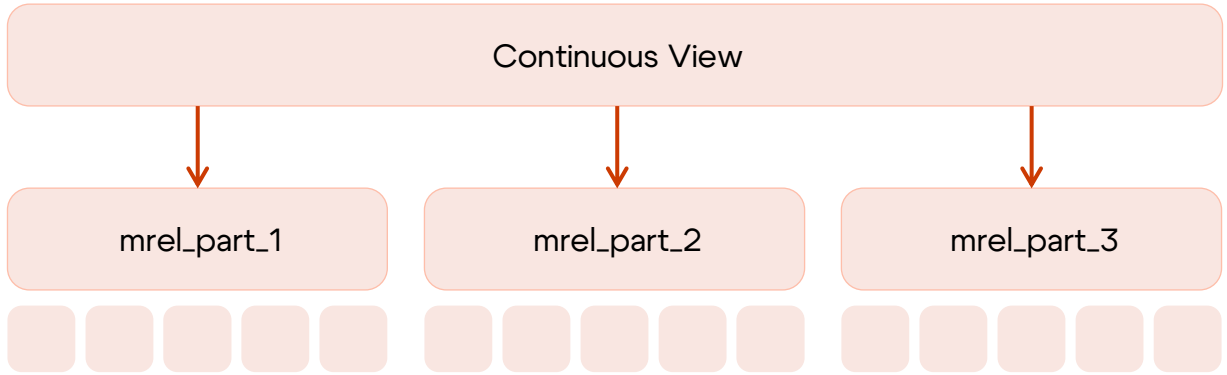
- удаляют данные с истёкшим TTL (time-to-live)
- интенсивность процесса управляется конфигурацией

```
CREATE VIEW v_ttl WITH
 (ttl = '1 month',
  ttl_column = 'minute')
 AS
 SELECT
  minute(arrival_timestamp),
  COUNT(*)
 FROM stream GROUP
 BY minute;
```



Новая функциональность: партицирование

- Востребовано в высоконагруженных системах
- Не присутствует в официальных релизах PipelineDB
- Наша работа основана на экспериментальном патче от авторов PipelineDB и включает ряд улучшений



ЧТО ЭТО?

Разбиение одной большой таблицы на несколько физических файлов/таблиц так, что логически для пользователя все партиции выглядят как единое целое.

В случае PipelineDB – это разбиение материализованных представлений на диске

ДЛЯ ЧЕГО?

Для минимизации накладных расходов на обновление индексов, выборки данных при чтении, а также удаления или архивирования старых данных.

КОГДА СТОИТ ИСПОЛЬЗОВАТЬ?

При необходимости разделения больших объёмов данных на подмножества, например по временным диапазонам (актуальные и исторические данные) с разными подходами к обработке и хранению.

Новая функциональность: партицирование

- поддерживается партицирование с заданным интервалом по колонкам типа TIMESTAMP или TIMESTAMPTZ
- партиции создаются автоматически

```
CREATE FOREIGN TABLE
stream (
  x INT,
  ts TIMESTAMP
) SERVER pipelinedb;

CREATE VIEW v
WITH (
  partition_by=ts,
  partition_duration='1 hour'
)
AS SELECT ts, x FROM
stream;
```

Новая функциональность: партицирование

- поддерживается партицирование с заданным интервалом по колонкам типа TIMESTAMP или TIMESTAMPTZ
- партиции создаются автоматически

```
CREATE FOREIGN TABLE
stream (
  x INT,
  ts TIMESTAMP
) SERVER pipelinedb;

CREATE VIEW v
WITH (
  partition_by=ts,
  partition_duration='1 hour'
)
AS SELECT ts, x FROM
stream;
```

```
\dt
              List of relations
 Schema |  Name  |       Type
--------+--------+-------------------
 public | v_mrel | partitioned table
(1 row)
```

Новая функциональность: партицирование

- поддерживается партицирование с заданным интервалом по колонкам типа TIMESTAMP или TIMESTAMPTZ
- партиции создаются автоматически

```
CREATE FOREIGN TABLE
stream (
  x INT,
  ts TIMESTAMP
) SERVER pipelinedb;

CREATE VIEW v
WITH (
  partition_by=ts,
  partition_duration='1 hour'
)
AS SELECT ts, x FROM
stream;
```

```
\dt
                List of relations
 Schema |  Name  |       Type
--------+--------+-------------------
 public | v_mrel | partitioned table
(1 row)
```

```
INSERT INTO stream VALUES
  (0, '2024-06-26 13:01:00'),
  (1, '2024-06-26 13:02:00'),
  (2, '2024-06-26 14:01:00'),
  (3, '2024-06-26 14:02:00'),
  (4, '2024-06-26 15:01:00'),
  (5, '2024-06-26 16:02:00');
```

Новая функциональность: партицирование

- поддерживается партицирование с заданным интервалом по колонкам типа TIMESTAMP или TIMESTAMPTZ
- партиции создаются автоматически

```
CREATE FOREIGN TABLE
stream (
  x INT,
  ts TIMESTAMP
) SERVER pipelinedb;

CREATE VIEW v
WITH (
  partition_by=ts,
  partition_duration='1 hour'
)
AS SELECT ts, x FROM
stream;
```

```
\dt
              List of relations
 Schema |  Name  |       Type
--------+--------+-------------------
 public | v_mrel | partitioned table
(1 row)
```

```
INSERT INTO stream VALUES
  (0, '2024-06-26 13:01:00'),
  (1, '2024-06-26 13:02:00'),
  (2, '2024-06-26 14:01:00'),
  (3, '2024-06-26 14:02:00'),
  (4, '2024-06-26 15:01:00'),
  (5, '2024-06-26 16:02:00');
```

```
\dt
                                  List of relations
 Schema |                      Name                       |       Type
--------+-------------------------------------------------+-------------------
 public | v_mrel                                          | partitioned table
 public | mrel_165244_20240626130000_20240626140000       | table
 public | mrel_165244_20240626140000_20240626150000       | table
 public | mrel_165244_20240626150000_20240626160000       | table
 public | mrel_165244_20240626160000_20240626170000       | table
(5 rows)
```

Новая функциональность: партицирование

- Ручное указание границ диапазонов не поддерживается
- «Умное» определение границ диапазонов для следующих популярных интервалов:
  - '1 year'
  - '1 month'
  - '1 week'
  - '1 day'
  - '1 hour'
  - '1 minute'
  - '1 second'
- Для «некруглых» интервалов используется формула:

```
interval_ts =
  '2000-01-01 00:00:00' + partition_duration
lower_bound =
  timestamp - (timestamp % inverval_ts)
```

Новая функциональность: партицирование

- Ручное указание границ диапазонов не поддерживается
- «Умное» определение границ диапазонов для следующих популярных интервалов:
  - ‘1 year’
  - ‘1 month’
  - ‘1 week’
  - ‘1 day’
  - ‘1 hour’
  - ‘1 minute’
  - ‘1 second’
- Для «некруглых» интервалов используется формула:

```
interval_ts =
  '2000-01-01 00:00:00' + partition_duration
lower_bound =
  timestamp - (timestamp % inverval_ts)
```

```
CREATE VIEW v_1w WITH
(
  partition_by=ts,
  partition_duration='1 week'
)
AS SELECT ts, x FROM stream;

CREATE VIEW v_2w WITH
(
  partition_by=ts,
  partition_duration='2 weeks'
)
AS SELECT ts, x FROM stream;

INSERT INTO stream VALUES
  (0, '2024-06-26 13:01:00');
```

Новая функциональность: партицирование

- Ручное указание границ диапазонов не поддерживается
- «Умное» определение границ диапазонов для следующих популярных интервалов:
  - '1 year'
  - '1 month'
  - '1 week'
  - '1 day'
  - '1 hour'
  - '1 minute'
  - '1 second'
- Для «некруглых» интервалов используется формула:

```
interval_ts =
  '2000-01-01 00:00:00' + partition_duration
lower_bound =
  timestamp - (timestamp % inverval_ts)
```

```
CREATE VIEW v_1w WITH
(
  partition_by=ts,
  partition_duration='1 week'
)
AS SELECT ts, x FROM stream;

CREATE VIEW v_2w WITH
(
  partition_by=ts,
  partition_duration='2 weeks'
)
AS SELECT ts, x FROM stream;

INSERT INTO stream VALUES
  (0, '2024-06-26 13:01:00');
```

Понедельник

```
\dt
                        List of relations
 Schema |                    Name                    |       Type
--------+--------------------------------------------+------------------
 public | mrel_165454_20240624000000_20240701000000  | table
 public | mrel_165471_20240615000000_20240629000000  | table
 public | v_1w_mrel                                  | partitioned table
 public | v_2w_mrel                                  | partitioned table
```

Суббота
'2000-01-01' +
N * '2 weeks'

История
и назначение
PipelineDB

Архитектура
PipelineDB
и развитие
в «Тантор Лабс»

Планы
на будущее

Планы на будущее

Производительность

PipelineDB уже проверена на высоких нагрузках.

На данный момент фокусируемся на функциональности, но планируем дальнейшую оптимизацию производительности в будущем.

Архивация

Умное управление устаревшими данными позволяет быстро удалять ненужные данные или преобразовывать нужные в колоночный формат со сжатием.

Сообщество

Востребованность проекта как на российском, так и на международном рынке требует возрождения сообщества пользователей и разработчиков вокруг проекта.

Планы на будущее: архивация

Планы на будущее: архивация

reaper-процессы
должны знать про партиции

Сейчас удаляют данные «пачками» для минимизации накладных расходов. Партицирование позволяет эффективно удалять ненужные данные партициями

Планы на будущее: архивация

reaper-процессы должны знать про партиции

Сейчас удаляют данные «пачками» для минимизации накладных расходов. Партицирование позволяет эффективно удалять ненужные данные партициями

Tantor SE

Колоночный формат

Автоматическое преобразование архивных данных в колоночный формат со сжатием (например, Hydra Columnar или DuckDB)

Планы на будущее: архивация

reaper-процессы должны знать про партиции

Сейчас удаляют данные «пачками» для минимизации накладных расходов. Партицирование позволяет эффективно удалять ненужные данные партициями

Tantor SE

Колоночный формат

Автоматическое преобразование архивных данных в колоночный формат со сжатием (например, Hydra Columnar или DuckDB)

Выгрузка во внешнее хранилище

Автоматическая выгрузка в S3 / облако а-ля **pg_tier** с сохранением доступа из PostgreSQL через FDW

Планы на будущее: архивация

reaper-процессы должны знать про партиции

Сейчас удаляют данные «пачками» для минимизации накладных расходов. Партицирование позволяет эффективно удалять ненужные данные партициями

Tantor SE

Колоночный формат

Автоматическое преобразование архивных данных в колоночный формат со сжатием (например, Hydra Columnar или DuckDB)

Выгрузка во внешнее хранилище

Автоматическая выгрузка в S3 / облако а-ля **pg_tier** с сохранением доступа из PostgreSQL через FDW

Tantor SE

Минимизация накладных расходов

Фоновая архивация / удаление данных могут негативно влиять на основную нагрузку.

Как не перегрузить систему блокировками и операциями с диском?

Tantor PipelineDB в OpenSource

Присоединяйтесь к сообществу Tantor PipelineDB!

Tantor PipelineDB на GitHub

- Проект возрожден, как форк проекта PipelineDB одноименной компании (PipelineDB LLC)
- Исправлено множество ошибок и сделаны оптимизации производительности
- Добавлена новая функциональность

Документация Tantor PipelineDB

- Документация обновлена и адаптирована, с учетом новых версий PostgreSQL
- Добавлено описание новой функциональности

tantor

PG BootCamp Russia 2025 Ekaterinburg

PGBootCamp.ru

Благодарим за внимание!

www.tantorlabs.ru