4. Критерии тяжелого запроса и тестирование

Критерии тяжелого запроса

Дать чёткого определения тяжелого запроса невозможно, поэтому выделяются запросы, которые потребляют много ресурсов. Запрос считается тяжелым, если выполнено хотя бы одно из условий:

Метрика	Что означает	Условие
duration_min	Время выполнения запроса в минутах.	>= 20
cpu_time_min	Процессорное время выполнения запроса в минутах. По сути это то вычислительное время, которое занимал ваш запрос на процессорах. Оно обычно больше duration_min, т.к. выполнение запроса происходит многопоточно на множестве процессоров.	>= 40
cpu_score	Некоторая оценка сложности запроса. Некоторые наблюдения: если cpu_score больше 30, то запрос требует пристального внимания. **Если же cpu_score больше 60, то запрос следует срочно оптимизировать (на 99% его можно сильно улучшить).**	>= 60
memory_gb	Потребление запросом оперативной памяти в ГБ.	>= 30

Например, пересчет данных analytics.line_items за 3 месяца в среднем имеет следующие метрики производительности: **duration_min** = 33.6, **cpu_time_min** = 80.2, **cpu_score** = 41.5, **memory_gb** = 167

Что делать, если запрос тяжелый

Самое простое – уменьшить глубину данных, например каждый день пересчитывать данные не за год/месяцы, а за последние 10 дней.

Если же уменьшить глубину данных невозможно, то оптимизировать запрос согласно предыдущим страницам из этого раздела. Если выполнены все пункты оптимизаций из этого раздела, но запрос все еще остался тяжелым – обращайтесь в ~dwh-support

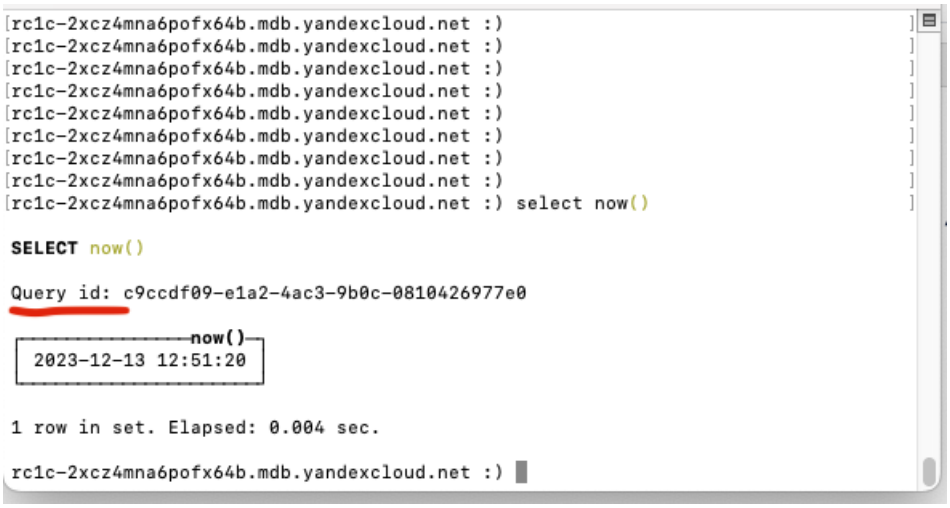
Как получить метрики запроса

Как протестировать запрос

Вариант 1: через query_id

Для этого нужно использовать clickhouse-client. Как установить.

После выполнения запроса он показывает дополнительную статистику, включая query_id. Зная query_id, можно легко найти статистику своего запроса в таблице system.query_log.



Получаем статистику с помощью query_log

```
WITH ['c9ccdf09-e1a2-4ac3-9b0c-0810426977e0', '61f0c404-5cb3-11e7-907b-a6006ad3dba0'] AS required_query_ids  --
<--   query_id,   .
SELECT
    query,
    query_start_time AS start_time,
    query_start_time + (query_duration_ms / 1000) AS end_time,
    round(query_duration_ms / 60000, 2) AS duration_min,
    round(((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 60000000, 2)
AS cpu_time_min,
    round(((((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 1000) /
query_duration_ms) * 17, 2) AS cpu_score,
    round(memory_usage / 1073741824, 2) AS memory_gb,
    round(read_bytes / 1073741824, 2) AS read_gb,
    round(result_bytes / 1073741824, 2) AS result_gb
FROM system.query_log
WHERE type IN ('QueryFinish', 'ExceptionWhileProcessing') AND query_id in required_query_ids
ORDER BY query_start_time
```

Вариант 2: добавить в начало запроса комментарий

Запрос, для которого нужно собрать показатели производительности

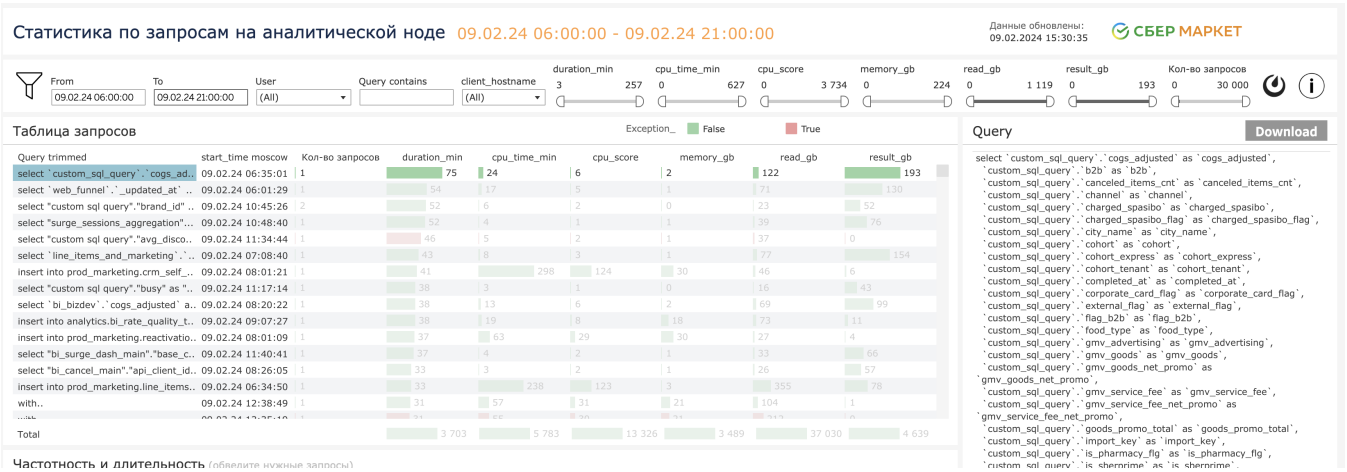
```
/* dnelyubov_test_query_01 */ -- <--    ,       .
SELECT * FROM dm.sber_api__sber_crm_b2c LIMIT 10
```

Получаем статистику запроса из query_log с помощью фильтра по query

```
WITH
    '%/* dnelyubov_test_query_%' AS required_query_comment,  -- <---       .
    today() AS required_event_date -- <---  ,    .
SELECT
    query,
    query_start_time AS start_time,
    query_start_time + (query_duration_ms / 1000) AS end_time,
    round(query_duration_ms / 60000, 2) AS duration_min,
    round(((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 60000000, 2)
AS cpu_time_min,
    round(((((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 1000) /
query_duration_ms) * 17, 2) AS cpu_score,
    round(memory_usage / 1073741824, 2) AS memory_gb,
    round(read_bytes / 1073741824, 2) AS read_gb,
    round(result_bytes / 1073741824, 2) AS result_gb
FROM system.query_log
WHERE event_date = required_event_date
    AND type IN ('QueryFinish', 'ExceptionWhileProcessing')
    AND query LIKE required_query_comment
ORDER BY query_start_time
```

Системная таблица system.query_log хранит данные за последние несколько дней.

Вариант 3: найти свой запрос в дашборде для поиска тяжелых запросов



С помощью фильтров можно найти свой запрос. Однако, дашборд обновляется только один раз 30 минут, что может быть не всегда удобно.

Дашборд.

Страница с описанием дашборда.

Тестирование запросов

Для того, чтобы корректно сравнить производительность запросов нужно прогнать запрос с конструкцией CREATE TABLE ... AS SELECT ...

- Поскольку это тестовая таблица движок должен быть нереплецированным, например MergeTree или ReplacingMergeTree.
- Указать пустой ключ сортировки, т.е. ORDER BY tuple()
- В конце запроса добавить настройку, которая отключает использование кеша. SETTINGS min_bytes_to_use_direct_io = 1

Создавать таблицу на лету обязательно без модификатора **ON CLUSTER**.
Подробнее про `ON CLUSTER`.

Например: **CREATE TABLE ... ~~ON CLUSTER~~ ... AS SELECT ...**

Пример

После оптимизации запроса нужно сравнить его производительность.

1. Запрос до оптимизации

```
WITH
    cte_shipments AS
    (
        SELECT
            shipped_at,
            delivery_window_id
        FROM analytics.int_spree_shipments
        WHERE state = 'shipped'
    ),
    cte_delivery_windows AS
    (
        SELECT
            id,
            kind
        FROM analytics.int_delivery_windows
    )
SELECT
    toYYYYMM(shipped_at) AS shipped_at_month,
    kind,
    count()
FROM cte_shipments AS s
INNER JOIN cte_delivery_windows AS dw ON dw.id = s.
delivery_window_id
GROUP BY
    shipped_at_month,
    kind
```

2. Запрос после оптимизации

```
WITH
    cte_shipments AS
    (
        SELECT
            shipped_at,
            delivery_window_id
        FROM analytics.int_spree_shipments
        WHERE state = 'shipped'
    ),
    cte_delivery_windows AS
    (
        SELECT
            id,
            kind
        FROM analytics.int_delivery_windows
        WHERE id IN ( --     CTE  .
            SELECT delivery_window_id
            FROM cte_shipments
        )
    )
SELECT
    toYYYYMM(shipped_at) AS shipped_at_month,
    kind,
    count()
FROM cte_shipments AS s
INNER JOIN cte_delivery_windows AS dw ON dw.id = s.
delivery_window_id
GROUP BY
    shipped_at_month,
    kind
```

Нужно запустить 2 версии запроса, воспользуемся вариантом поиска запросов по специально оставленному комментарию.

Прогон первого неоптимизированного запроса

```
/* dnelyubov_test_query_01 */ -- ,         system.query_log.
CREATE TABLE sandbox.data_1234__dnelyubov__query_test --       ,      .
ENGINE = MergeTree() --  , ..    .
ORDER BY tuple() AS --   ,
WITH
    cte_shipments AS
    (
        SELECT
            shipped_at,
            delivery_window_id
        FROM analytics.int_spree_shipments
        WHERE state = 'shipped'
    ),
    cte_delivery_windows AS
    (
        SELECT
            id,
            kind
        FROM analytics.int_delivery_windows
    )
SELECT
    toYYYYMM(shipped_at) AS shipped_at_month,
    kind,
    count()
FROM cte_shipments AS s
INNER JOIN cte_delivery_windows AS dw ON dw.id = s.delivery_window_id
GROUP BY
    shipped_at_month,
    kind
SETTINGS min_bytes_to_use_direct_io = 1 --     ,    .
;

DROP TABLE sandbox.data_1234__dnelyubov__query_test --   , ..   .
```

Прогон второго оптимизированного запроса

```
/* dnelyubov_test_query_02 */ -- ,         system.query_log.
CREATE TABLE sandbox.data_1234__dnelyubov__query_test --       ,      .
ENGINE = MergeTree() --  , ..    .
ORDER BY tuple() AS --   ,
WITH
    cte_shipments AS
    (
        SELECT
            shipped_at,
            delivery_window_id
        FROM analytics.int_spree_shipments
        WHERE state = 'shipped'
    ),
    cdelivery_windows AS
    (
        SELECT
            id,
            kind
        FROM analytics.int_delivery_windows
        WHERE id IN ( --     CTE  .
            SELECT delivery_window_id
            FROM cte_shipments
        )
    )
SELECT
    toYYYYMM(shipped_at) AS shipped_at_month,
    kind,
    count()
FROM cte_shipments AS s
INNER JOIN cte_delivery_windows AS dw ON dw.id = s.delivery_window_id
GROUP BY
    shipped_at_month,
    kind
SETTINGS min_bytes_to_use_direct_io = 1 --     ,    .
;

DROP TABLE sandbox.data_1234__dnelyubov__query_test --   , ..   .
```

Получаем метрики двух запросов

```
WITH
    '%/* dnelyubov_test_query_%' AS required_query_comment,  -- <---       .
    today() AS required_event_date -- <---  ,    .
SELECT
    query,
    query_start_time AS start_time,
    query_start_time + (query_duration_ms / 1000) AS end_time,
    round(query_duration_ms / 60000, 2) AS duration_min,
    round(((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 60000000, 2)
AS cpu_time_min,
    round(((((ProfileEvents['UserTimeMicroseconds']) + (ProfileEvents['SystemTimeMicroseconds'])) / 1000) /
query_duration_ms) * 17, 2) AS cpu_score,
    round(memory_usage / 1073741824, 2) AS memory_gb,
    round(read_bytes / 1073741824, 2) AS read_gb,
    round(result_bytes / 1073741824, 2) AS result_gb
FROM system.query_log
WHERE event_date = required_event_date
    AND type IN ('QueryFinish', 'ExceptionWhileProcessing')
    AND query LIKE required_query_comment
ORDER BY query_start_time
```

query	start_time	end_time	**duration_min**	**cpu_time_min**	**cpu_score**	**memory_gb**	**read_gb**	result_gb
/* dnelyubov_test_query_01 */ -- K	31.01.2024 11:59	31.01.2024 12:01	1.43	26.6	317.25	1.76	11.62	0

/* dnelyubov_test_query_02 */ -- K	31.01.2024 12:07	31.01.2024 12:08	0.71	9.21	220.88	3.71	12.4	0

Как видно, оптимизированная версия запроса работает в 2 раза быстрее, потребляет меньше cpu_time_min, но использует в два раза больше оперативной памяти. Запрос считается тяжелым из-за высокого cpu_score.