Примеры выбора ORDER BY из практики

Пример 1.

Создаем таблицу, используя столбцы из event.new_app, без сортировки.
У event.new_app есть партицирование, но для простоты, в этом примере его не будет.
Это тестовая таблица, данные реплицировать на другие ноды не нужно, поэтому движок **MergeTree**.

```
CREATE TABLE sandbox.dwh5691__event__new_app__test
(
    `dwh_dt` Date,
    `platform` Enum8('android' = 1, 'ios' = 2),
    `event` LowCardinality(String),
    `anonymous_id` String,
    `mytracker_id` Nullable(UUID),
    `backend_is_authorised` Nullable(UInt8),
    `backend_user_uuid` String,
    `ts` DateTime('UTC'),
    `session_id` Nullable(UInt64),
    `delivery_method` LowCardinality(String),
    `order_id` String,
    `retailer_id` Nullable(Int16),
)
ENGINE = MergeTree
ORDER BY tuple()
```

Загружаем данные за вчерашний день по ios пользователям.
* Если не перемешать строки с использованием `ORDER BY rand32()`, в несортированную таблицу попадут упорядоченные данные из оригинальной таблицы.

```
INSERT INTO sandbox.dwh5691__event__new_app__test SELECT
    dwh_dt,
    platform,
    event,
    anonymous_id,
    mytracker_id,
    backend_is_authorised,
    backend_user_uuid,
    ts,
    session_id,
    delivery_method,
    order_id,
    retailer_id
FROM event.new_app
WHERE (dwh_dt = yesterday()) AND (platform = 'ios')
ORDER BY rand32()
```

Делаем принудительный **OPTIMIZE** всех партов, чтобы ClickHouse собрал их в один.
Это может уменьшить занимаемое место, но делать так регулярно затратно по ресурсам.
Обычно ClickHouse сам соединяет парты в фоне в неизвестный момент времени.

```
OPTIMIZE TABLE sandbox.dwh5691__event__new_app__test FINAL
```

Пробуем сделать **SELECT** с агрегацией и фильтром.

```
SELECT uniq(anonymous_id)
FROM sandbox.dwh5691__event__new_app__test
WHERE event = 'App Opened'
SETTINGS min_bytes_to_use_direct_io = 1 -- <--   ,     .
```

uniq(anonymous_id)
21119

Рассмотрим производительность этого запроса.

```
SELECT
    query_duration_ms / 1000 AS duration_sec,
    formatReadableSize(memory_usage) AS memory_usage_readable,
    read_rows
FROM system.query_log
WHERE (type = 'QueryFinish') AND (query_id = '736f47cf-dae9-4390-9f57-280d8551b676')
```

duration_sec	memory_usage_readable	read_rows
23.576	141.68 MiB	263,313,107

Как получить статистику SQL-запроса

Рассмотрим сколько весит каждый столбец.

```
SELECT
    name AS colmun,
    any(type) AS type,
    formatReadableSize(sum(data_compressed_bytes) AS size) AS compressed,
    formatReadableSize(sum(data_uncompressed_bytes) AS usize) AS uncompressed,
    round(usize / size, 2) AS compr_ratio
FROM system.columns
WHERE table = 'sandbox.dwh5691__event__new_app__test'
GROUP BY
    table,
    name
ORDER BY size DESC
```

colmun	type	compressed	uncompressed	compr_ratio
anonymous_id	String	9.03 GiB	9.07 GiB	1.0
backend_user_uuid	String	8.40 GiB	8.44 GiB	1.01
mytracker_id	Nullable(UUID)	3.92 GiB	4.17 GiB	1.06
order_id	String	2.11 GiB	2.54 GiB	1.21
session_id	Nullable(UInt64)	1.97 GiB	2.21 GiB	1.12
random_number	UInt32	1006.96 MiB	1004.46 MiB	1.0
ts	DateTime('UTC')	999.15 MiB	1004.46 MiB	1.01
retailer_id	Nullable(Int16)	379.01 MiB	753.34 MiB	1.99
event	LowCardinality(String)	272.09 MiB	502.70 MiB	1.85
delivery_method	LowCardinality(String)	78.60 MiB	251.67 MiB	3.2
backend_is_authorised	Nullable(UInt8)	59.22 MiB	502.23 MiB	8.48
dwh_dt	Date	2.24 MiB	502.23 MiB	223.79
platform	Enum8('android' = 1, 'ios' = 2)	1.12 MiB	251.11 MiB	224.49

Таблица весит **28.17 GiB**

Зная природу данных, можно заметить, что в поле anonymous_id много одинаковых значений, проверим это.

```
SELECT
    formatReadableQuantity(uniqHLL12(anonymous_id)) AS uniq_anon_id,
    formatReadableQuantity(count()) AS total_cnt
FROM sandbox.dwh5691__event__new_app__test
;
```

uniq_anon_id	total_cnt
325.27 thousand	263.31 million

> *uniqHLL12() вычисляет приблизительное число различных значений аргументов.*
> *В данном случае этого достаточно, а функция uniqHLL12() дешевле, чем uniq(). Подробнее в документации*

Для каждого anonymous_id строки часто имеют схожие атрибуты, например, использование одной и той же платформы приложения, одинаковые mytracker_id, backend_user_uuid и т.д.
Добавив сортировку по времени события вторым ключом, мы делаем строки внутри anonymous_id ещё более похожими друг на друга.
Например, несколько последовательных действий могут быть совершены для одного retailer_id, order_id и так далее.
Сделаем сортировку по (anonymous_id, ts).

```
CREATE TABLE sandbox.dwh5691__event__new_app__test__1
(
    `dwh_dt` Date,
    `platform` Enum8('android' = 1, 'ios' = 2),
    `event` LowCardinality(String),
    `anonymous_id` String,
    `mytracker_id` Nullable(UUID),
    `backend_is_authorised` Nullable(UInt8),
    `backend_user_uuid` String,
    `ts` DateTime('UTC'),
    `session_id` Nullable(UInt64),
    `delivery_method` LowCardinality(String),
    `order_id` String,
    `retailer_id` Nullable(Int16)
)
ENGINE = MergeTree
ORDER BY tuple(anonymous_id, ts)

--    INSERT  OPTIMIZE .  ,     .
```

Сравним вес полей у первого и второго варианта таблицы.

Column	Type	ORDER BY tuple()	ORDER BY tuple(anonymous_id, ts)	Степень сжатия
anonymous_id	String	9.03 GiB	173.11 MiB	52.16
backend_user_uuid	String	8.40 GiB	199.66 MiB	42.06
mytracker_id	Nullable(UUID)	3.92 GiB	111.49 MiB	35.18
order_id	String	2.11 GiB	100.51 MiB	20.98
session_id	Nullable(UInt64)	1.97 GiB	104.10 MiB	18.95
ts	DateTime('UTC')	999.15 MiB	728.59 MiB	1.37
retailer_id	Nullable(Int16)	379.01 MiB	75.44 MiB	5.02
event	LowCardinality(String)	272.09 MiB	225.46 MiB	1.21
delivery_method	LowCardinality(String)	78.60 MiB	30.94 MiB	2.54
backend_is_authorised	Nullable(UInt8)	59.22 MiB	18.86 MiB	3.14
dwh_dt	Date	2.24 MiB	2.25 MiB	1.0
platform	Enum8('android' = 1, 'ios' = 2)	1.12 MiB	1.12 MiB	1.0

Таблица уменьшила размер с **28.17 GiB до 1.74 GiB.**

Проверим производительность запроса на новой таблице.

```
SELECT uniq(anonymous_id)
FROM sandbox.dwh5691__event__new_app__test
WHERE event = 'App Opened'
SETTINGS min_bytes_to_use_direct_io = 1 -- <--   ,     .
```

Производительность запроса

duration_sec	memory_usage_readable	read_rows
0.496	36.44 MiB	263,313,107

Производительность выросла на порядки.
Время выполнения **23.576 0.496**, память **141.68 MiB 36.44 MiB**, но количество прочитанных строк (**read_rows**) не изменилось.

Зная, что в основной массе **SELECT**-запросов к этой таблице есть фильтр на поле event, добавим этот ключ сортировки на первое место.
В этом случае, при фильтрации по полю event, ClickHouse не будет перебирать все поля в каждой строке, а прочитает только такие куски данных, в которых нужное значение event.

```
CREATE TABLE sandbox.dwh5691__event__new_app__test__2
(
    `dwh_dt` Date,
    `platform` Enum8('android' = 1, 'ios' = 2),
    `event` LowCardinality(String),
    `anonymous_id` String,
    `mytracker_id` Nullable(UUID),
    `backend_is_authorised` Nullable(UInt8),
    `backend_user_uuid` String,
    `ts` DateTime('UTC'),
    `session_id` Nullable(UInt64),
    `delivery_method` LowCardinality(String),
    `order_id` String,
    `retailer_id` Nullable(Int16)
)
ENGINE = MergeTree
ORDER BY tuple(event, anonymous_id, ts)

--    INSERT  OPTIMIZE .  ,     .
```

Column	Type	ORDER BY tuple (anonymous_id, ts)	ORDER BY tuple(event, anonymous_id, ts)	Степень сжатия
ts	DateTime('UTC')	728.59 MiB	522.21 MiB	1.39
event	LowCardinality(String)	225.46 MiB	1.95 MiB	115.63
backend_user_uuid	String	199.66 MiB	526.74 MiB	0.38
anonymous_id	String	173.11 MiB	540.14 MiB	0.32
mytracker_id	Nullable(UUID)	111.49 MiB	278.02 MiB	0.40
session_id	Nullable(UInt64)	104.10 MiB	238.52 MiB	0.44
order_id	String	100.51 MiB	184.66 MiB	0.54
retailer_id	Nullable(Int16)	75.44 MiB	81.56 MiB	0.92
delivery_method	LowCardinality(String)	30.94 MiB	18.24 MiB	1.69
backend_is_authorised	Nullable(UInt8)	18.86 MiB	12.52 MiB	1.51
dwh_dt	Date	2.25 MiB	2.25 MiB	1.0
platform	Enum8('android' = 1, 'ios' = 2)	1.12 MiB	1.12 MiB	1.0

Таблица выросла с **1.74 GiB до 2.37 GiB**.
Такая прибавка к месту допустима, ведь в замен, большинство **SELECT**-запросов стало работать быстрее.

Убедимся в этом.

```
SELECT uniq(anonymous_id)
FROM sandbox.dwh5691__event__new_app__test
WHERE event = 'App Opened'
SETTINGS min_bytes_to_use_direct_io = 1 -- <--   ,     .
```

Производительность запроса.

duration_sec	memory_usage_readable	read_rows
0.05	4.09 MiB	204,800

Сократилось время выполнения запроса **с 0.496 до 0.05**, память **с 36.44 MiB до 4.09 MiB** и *количество прочитанных строк* ***с 263 млн. до 204 тыс.***
Если сравнивать с такой же таблицей без сортировки, то время выполнения **с 23.576 до 0.05**, память **с 141.68 MiB до 4.09 MiB.**

Сортировка по часто используемому полю в фильтре может ускорить **SELECT**, поскольку ClickHouse не будет перебирать всю таблицу для поиска нужных строк.
Эффективное сжатие тяжелого столбца достигается, если похожие значения расположены физически рядом.
Путем объединения двух подходов к сортировке и учета природы данных, размеров колонок и частоты **SELECT**-запросов к таблице,
можно одновременно сократить использование места на диске и уменьшить время выполнения запросов.