

Яндекс

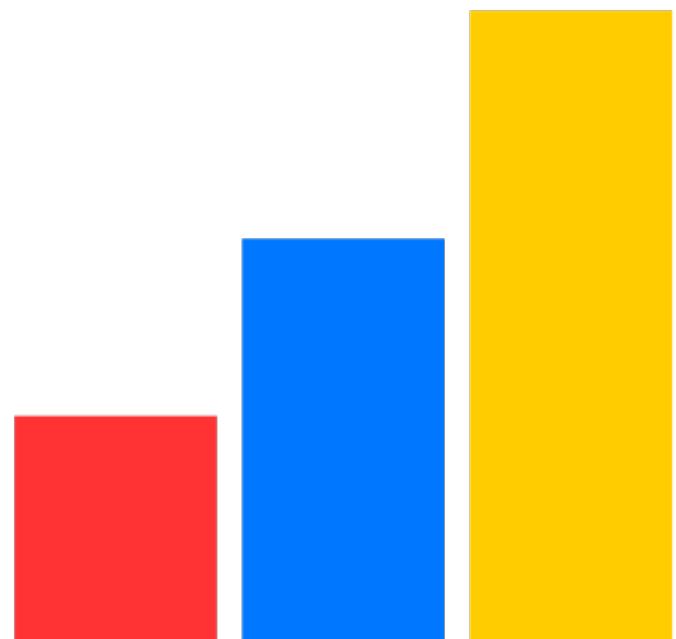
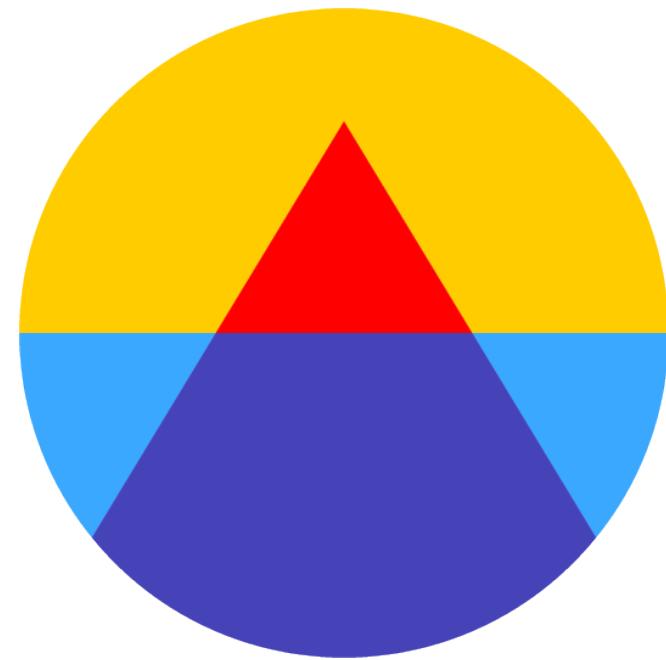


ClickHouse в Метрике и AppMetrica

Мария Мансурова, аналитические продукты Яндекса

Аналитические продукты

- › Яндекс.Метрика
- › AppMetrica
- › Яндекс.Радар
- › ОФД



Яндекс.Метрика

- › Система веб-аналитики №1 в России и № 3 в мире
- › ~ 350К пользователей отчетов в месяц
- › 20 миллиардов событий и 400 миллионов кук в день



AppMetrica

| SDK для мобильной аналитики,
трекинга установок, push
notifications

- › Видим 400 млн устройств
- › 10 тысяч приложений
- › Более 50 миллиардов событий в день



Эволюция Метрики

- › Раньше: ~40 «фиксированных» отчетов
- › Теперь: отчет можно произвольно менять: добавлять измерения, сегментировать, сравнивать и т.д.



Настройте Метрику для вашего сайта.

Сегодня

Вчера

Неделя

Месяц

Квартал

Год

30 июн — 6 июл 2019

Детализация: по дням

Сегмент: 2 условия



Сравнить сегменты



Точность: 100%

Визиты, в которых

Местоположение: Россия



для людей, у которых

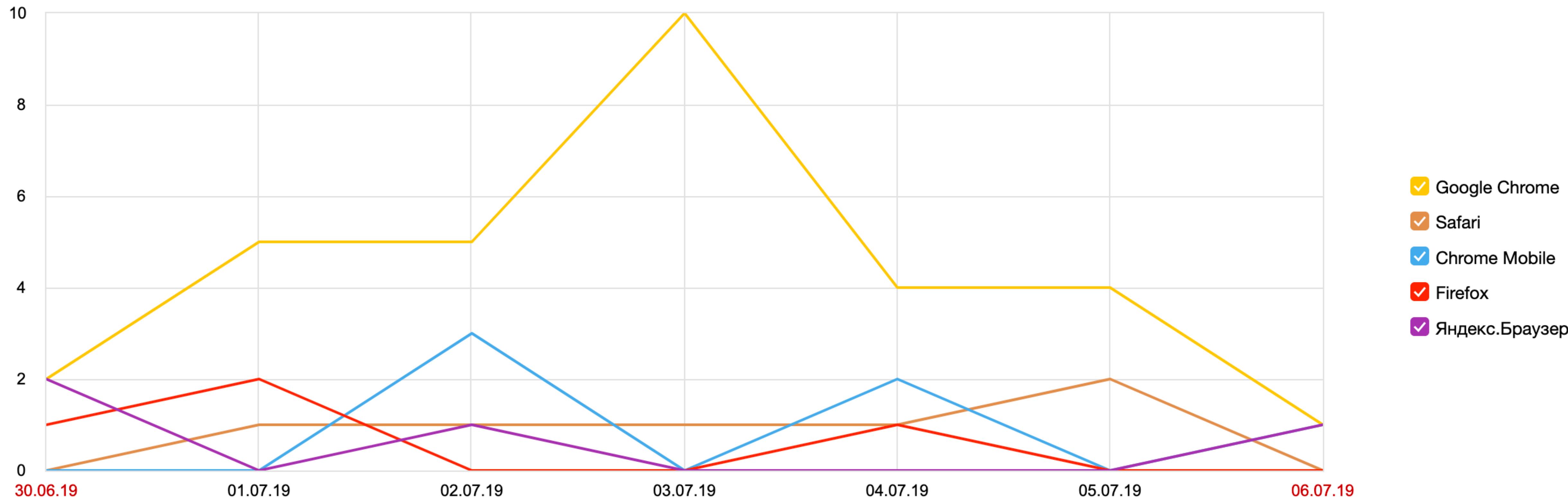
Просмотр URL: https://metrica.yandex.com/about/info/*



Визиты



7/7



Смена парадигмы

Старый подход

- › заранее рассчитанные агрегированные данные (комбинаторный взрыв)
- › ограниченное число срезов
- › ответ всегда готов

Новый подход

- › сырые данные о событиях на сайте
- › произвольные фильтры и сегменты
- › все вычисления в момент запроса



Так и появился
ClickHouse

Наша инсталляция ClickHouse

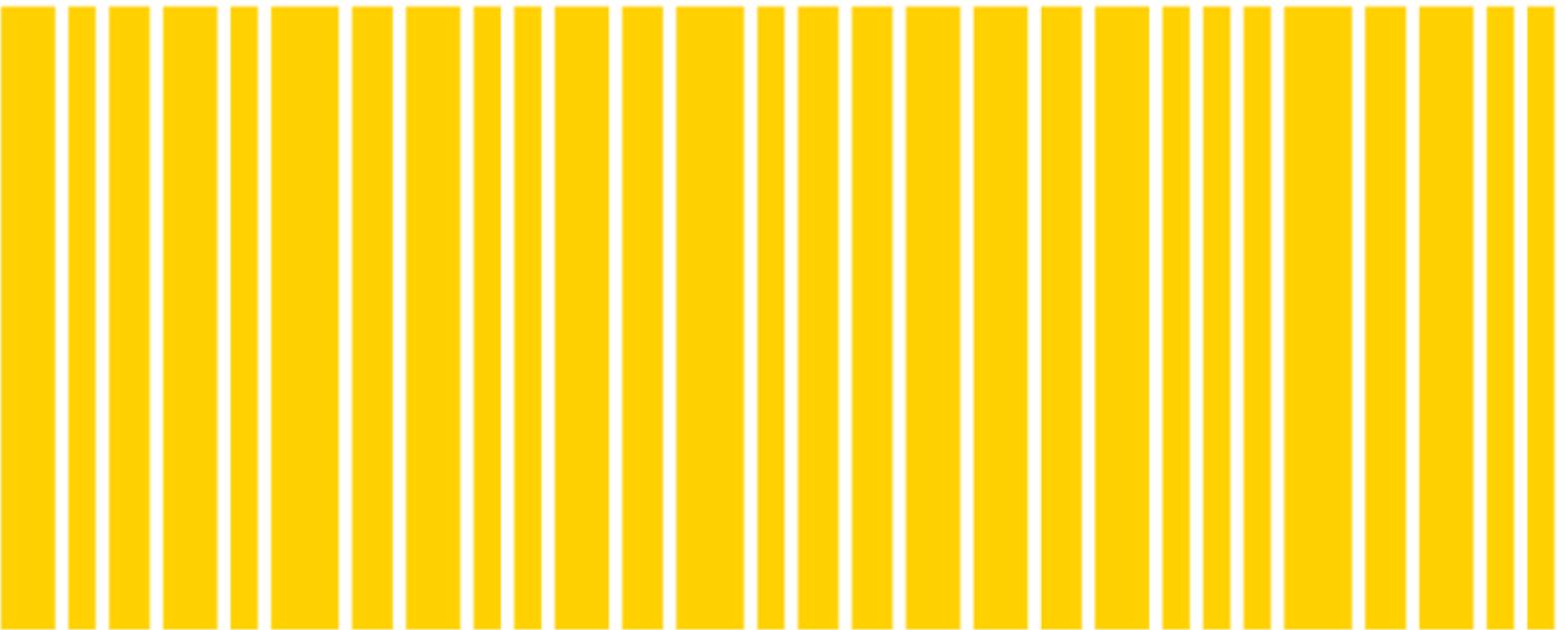
Для Яндекс.Метрики

- › 2 таблицы: хиты и визиты (~ 100 столбцов в каждой)
- › > 5 Пб данных
- › ~ 20 миллиардов событий каждый день



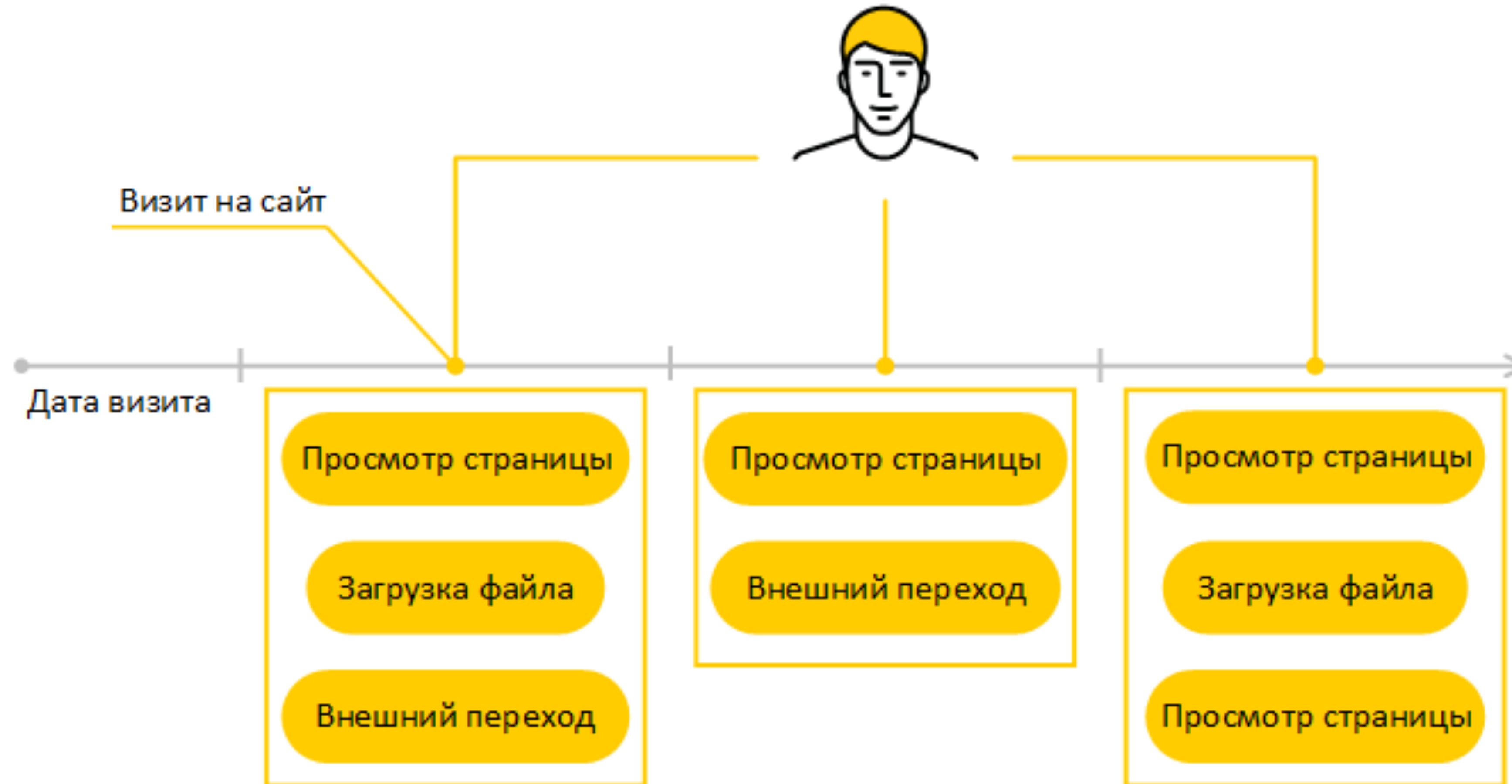
OLAP сценарий работы

- › несколько слабо-связанных таблиц с большим количеством столбцов
- › большое количество строк, но только небольшое подмножество столбцов
- › хранятся небольшие значения: числа, строки
- › простой сценарий обновления данных
- › результат выполнения запроса существенно меньше исходных данных



| Всё выглядит очень
подходяще =)

Визиты



Пользовательские сессии

- › Объединяем действия пользователей на сайте в сессии, если между последовательными событиями прошло меньше N минут.
- › Следовательно - мы постоянно обновляем данные о визитах.





Как обновлять данные?

Инкрементальный лог

VisitID	StartURL	PageViews	Sign
12345678	https://clickhouse.yandex/docs/en/operations/table_engines	1	1
12345678	https://clickhouse.yandex/docs/en/operations/table_engines	1	-1
12345678	https://clickhouse.yandex/docs/en/operations/table_engines	2	1

ClickHouse и аналитики





Чем занимаются
аналитики?

Чем занимаются аналитики?

Основная задача аналитиков - помогать принимать решения на основе данных.

- › Пишут adhoc-код на скриптовых языках (python, реже R)
 - › Работают с большими и зашумленными данными



Примеры задач

- › Какие отчеты смотрят пользователи Метрики и AppMetrica в интерфейсе?
- › Как повлияла рассылка о новой feature на пользователей?
- › Хорошо ли работает Метрика: не теряем ли мы данных?
- › Как часто перевыставляются куки?



| ClickHouse + аналитики = ❤

Иногда ClickHouse не помогает

- › Группировка по большому числу ключей (например, построение сессий по всем пользователям Рунета)
- › Часть данных не лежит в ClickHouse
- › Нет специфических функций, например, полноценный парсинг json



Как мы используем ClickHouse

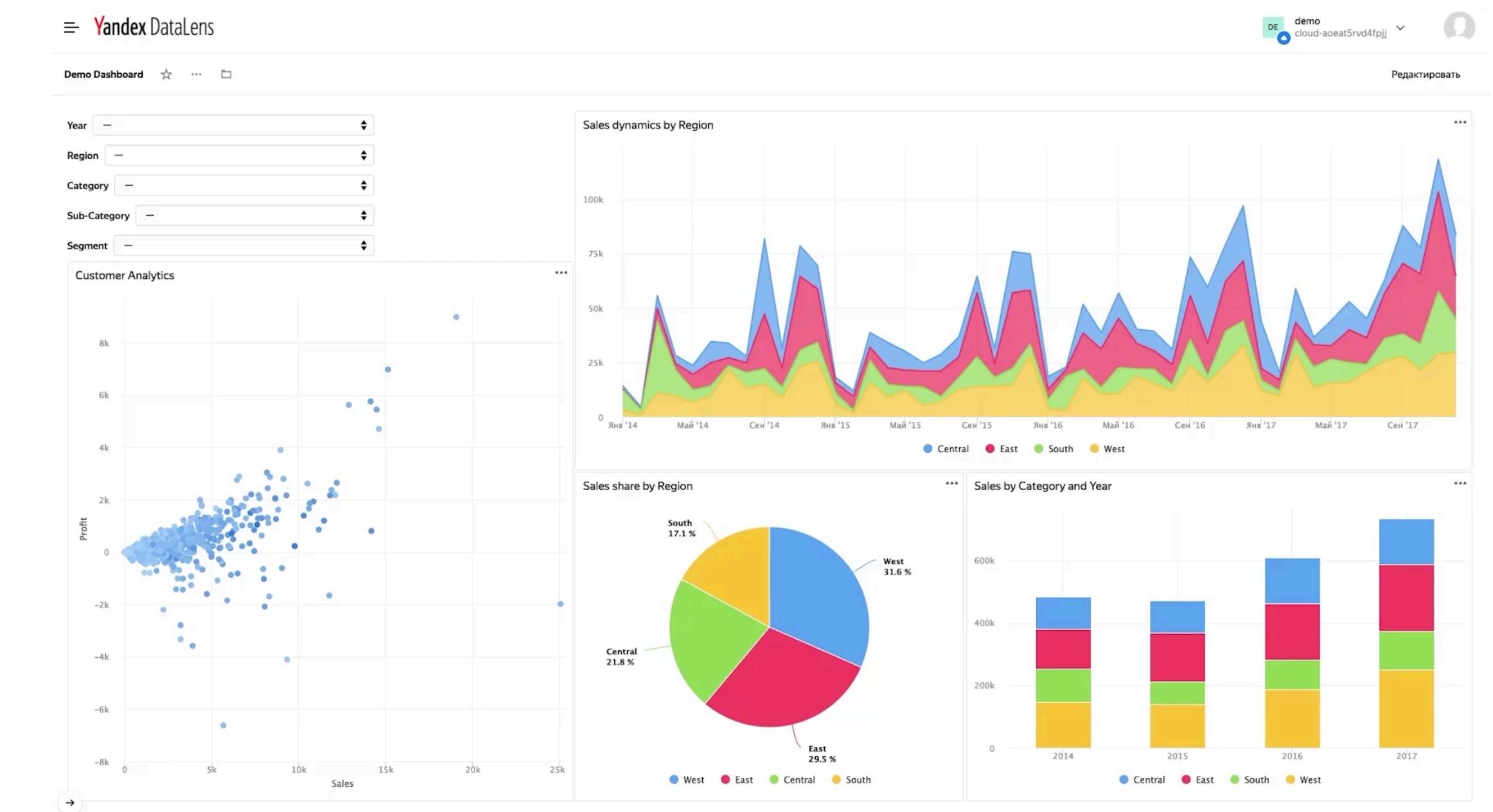
- › CLI
- › Python + HTTP API
- › DataLens



DataLens

Сервис визуализации и анализа данных

- › Работает с данными из ClickHouse
- › Интеграция с API Метрики
- › Дашборды и разные типы визуализации



Case: retention в Яндекс.Метрике



Возвращаемость пользователей

Метрики - retention и rolling retention по когортам

- › Когорты: друзья из детского сада, школы, университета
- › Retention по годам: доля друзей из определенной когорты, с которыми сохранились отношения на N-й год после знакомства



Retention

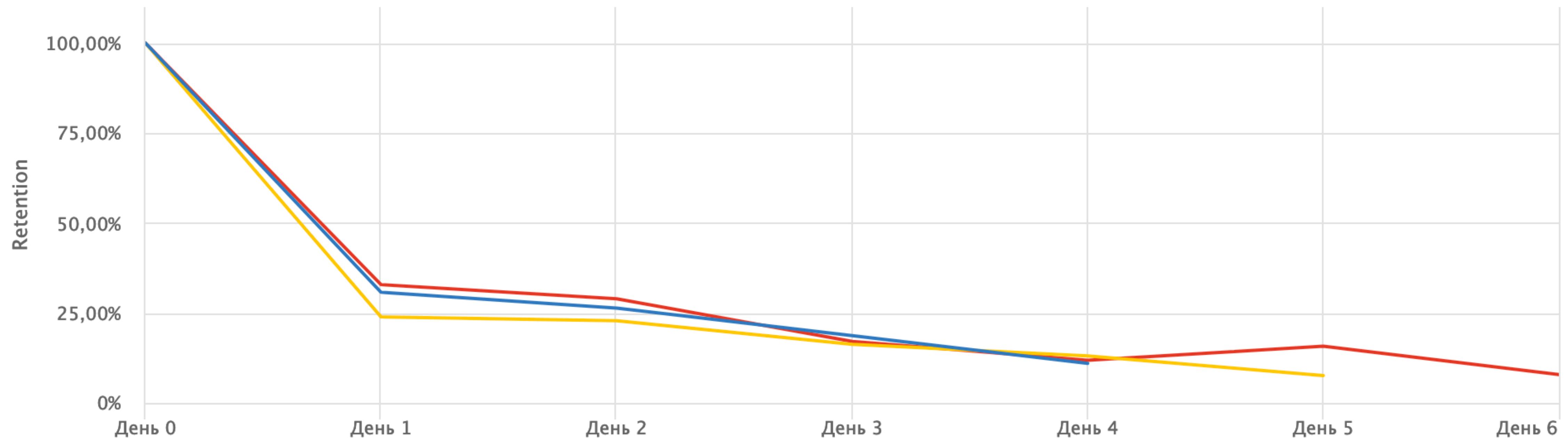


Таблица с когортами пользователей

```
CREATE TABLE retention_users ENGINE = Log AS  
SELECT DISTINCT  
    ClientID as client_id,  
    toMonday(Date) as date  
FROM visits_all  
WHERE (client_id != 0)
```

Считаем retention

```
SELECT uniq(client_id) as clients, min_date, (date - min_date)/7 as week_num
FROM
  (SELECT client_id, min(date) as min_date, max(date) as max_date
   FROM retention_users
   GROUP BY client_id)
ALL INNER JOIN
  (SELECT client_id, date FROM retention_users)
USING client_id
GROUP BY week_num, min_date
```

Массивы в ClickHouse





Как работают массивы в ClickHouse?



В чем отличие между
arrayJoin и ARRAY JOIN?

Массивы

- › `groupArray` - агрегатная функция для создания массивов
- › `arrayJoin` и `ARRAY JOIN` для разбивания массивов по строкам
- › Функции высшего порядка



Пример: groupArray

```
SELECT
    ClientID AS client_id,
    groupArray(LastTrafficSource) AS source,
    groupArray(Date) AS dates
FROM visits_all
GROUP BY client_id
LIMIT 5
```

client_id	source	dates
14715340521072063607	['organic']	['2017-05-05']
1486216021300443344	['referral', 'referral', 'organic', 'referral']	['2017-02-16', '2017-02-16', '2017-04-13', '2017-04-14']
1490794176528267607	['organic']	['2017-04-03']
1489493549710926119	['organic']	['2017-03-21']
1463659610347692840	['organic', 'organic', 'organic']	['2016-11-26', '2016-11-28', '2016-11-25']

Пример: arrayJoin

```
SELECT
    *,
    arrayJoin(Products) AS Product
FROM
(
    SELECT
        ['coffee', 'cookie'] AS Products,
        123 AS ClientID,
        today() AS Date
)
```

Products	ClientID	Date	Product
['coffee', 'cookie']	123	2019-07-05	coffee
['coffee', 'cookie']	123	2019-07-05	cookie

Пример: arrayJoin с 2 массивами

```
SELECT
    *,
    arrayJoin(Products) AS Product,
    arrayJoin(Prices) AS Price
FROM
(
    SELECT
        ['coffee', 'cookie'] AS Products,
        [250, 150] AS Prices,
        123 AS ClientID,
        today() AS Date
)
```

Products	Prices	ClientID	Date	Product	Price
['coffee', 'cookie']	[250, 150]	123	2019-07-05	coffee	250
['coffee', 'cookie']	[250, 150]	123	2019-07-05	coffee	150
['coffee', 'cookie']	[250, 150]	123	2019-07-05	cookie	250
['coffee', 'cookie']	[250, 150]	123	2019-07-05	cookie	150

Пример: ARRAY JOIN

```
SELECT
    *,
    Product,
    Price
FROM
(
    SELECT
        ['coffee', 'cookie'] AS Products,
        [250, 150] AS Prices,
        123 AS ClientID,
        today() AS Date
)
ARRAY JOIN
    Products AS Product,
    Prices AS Price
```

Products	Prices	ClientID	Date	Product	Price
['coffee', 'cookie']	[250, 150]	123	2019-07-05	coffee	250
['coffee', 'cookie']	[250, 150]	123	2019-07-05	cookie	150

ФУНКЦИИ ВЫСШЕГО ПОРЯДКА

Функции

- › arrayMap
- › arrayFilter
- › arrayFirst
- › и многие другие

Lambda функции

- › $x \rightarrow x = 'Order Opened'$
- › $x, y \rightarrow \text{pow}(x, 2) + \text{pow}(y, 2)$



Пример: arrayFilter

```
SELECT
    ClientID AS client_id,
    groupArray(LastTrafficSource) AS sources,
    arrayFilter(x -> (x != 'organic'), sources) AS sources_filtered
FROM visits_all
GROUP BY client_id
LIMIT 5
```

client_id	sources	sources_filtered
14715340521072063607	['organic']	[]
1486216021300443344	['referral', 'referral', 'organic', 'referral']	['referral', 'referral', 'referral']
1490794176528267607	['organic']	[]
1489493549710926119	['organic']	[]
1463659610347692840	['organic', 'organic', 'organic']	[]

Другие полезные функции для массивов

- › `length(arr)`
- › `arrayElement(arr, n)`
- › `indexOf(arr, x)`
- › `countEqual(arr, x)`
- › `arrayEnumerate(arr)`
- › `has(arr, elem)`



Другие полезные функции для массивов

SELECT

```
length(arr) AS len,  
arr[2] AS second_elem,  
indexOf(arr, 4) AS four_index,  
countEqual(arr, 3) AS num_threes,  
arrayEnumerate(arr) AS indexes,  
has(arr, 18) AS has_18,  
has(arr, 4) AS has_4
```

FROM

```
(  
    SELECT [1, 2, 3, 4, 5, 4, 3, 2, 1] AS arr  
)
```

len	second_elem	four_index	num_threes	indexes	has_18	has_4
9	2	4	2	[1,2,3,4,5,6,7,8,9]	0	1

Считаем rolling retention

```
SELECT uniq(client_id) as clients, min_date, week_num
FROM
(SELECT
client_id,
min_date,
arrayJoin(range(toUInt64((max_date - min_date)/7) + 1)) as week_num
FROM
(SELECT client_id, min(date) as min_date, max(date) as max_date
FROM retention_users
GROUP BY client_id))
GROUP BY min_date, week_num
```

Case: Атрибуция в Яндекс.Метрике



Конверсия

Владелец приложения или сайта хочет, чтобы пользователи совершили определенные действия. Это и есть конверсии.

Примеры

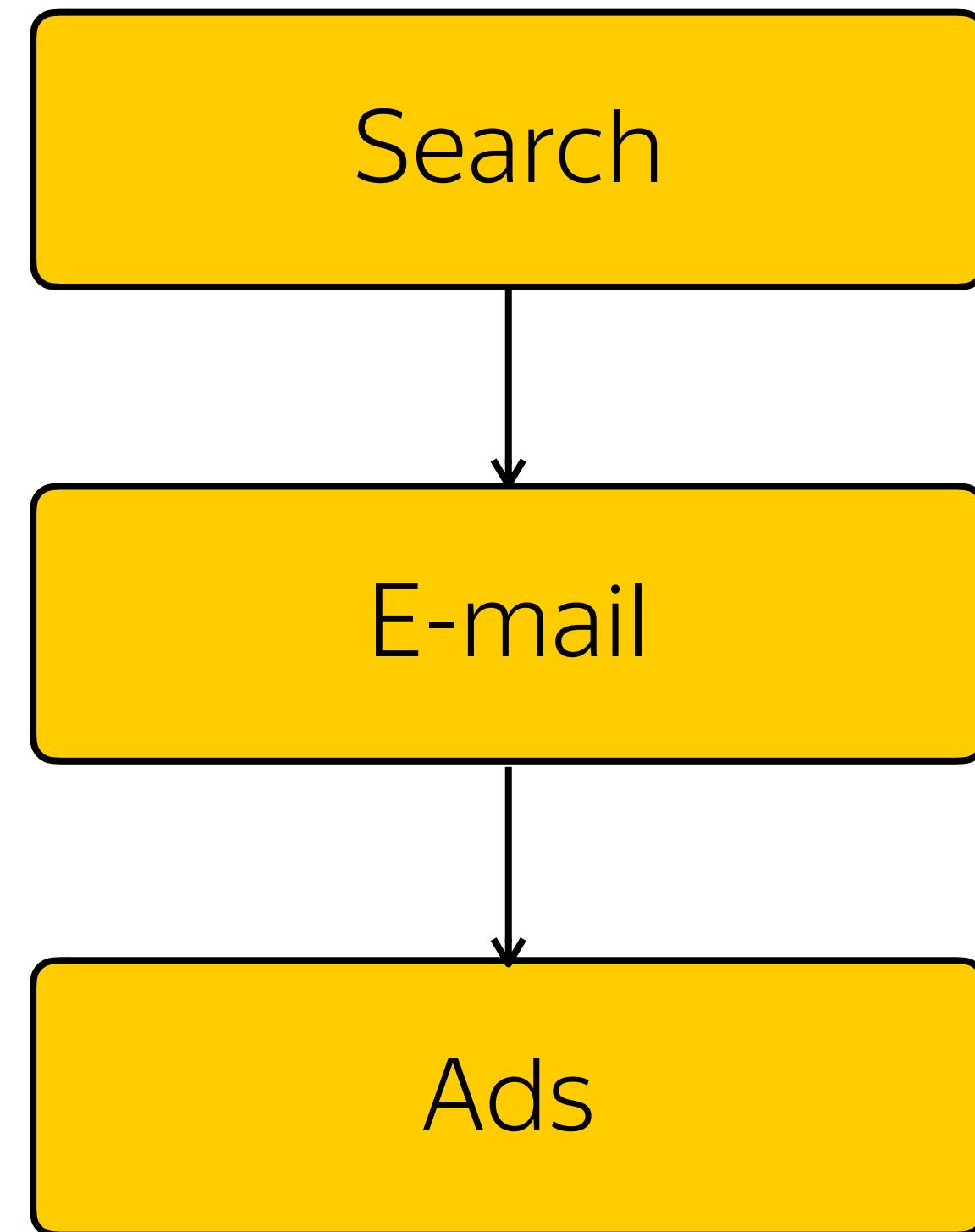
- › Совершить покупку
- › Залогиниться
- › Заполнить форму



Что такое атрибуция?

Модель атрибуции - принцип, по которому для конкретного визита будет назначен источник трафика.

- › Первый переход - откуда пользователь узнал о сайте.
- › Последний переход - понимать, какие источники привели к конверсии.



Кастомные модели атрибуции

- › Учесть, сколько времени прошло между визитами пользователя.
- › Изменять вес в зависимости от поведения пользователя (добавление товара в корзину или неотказ).



Подробнее

- › совместный case
Яндекс.Метрики и 220Вольт
- › кастомные модели атрибуции
- › вся мощь массивов и функций
высшего порядка

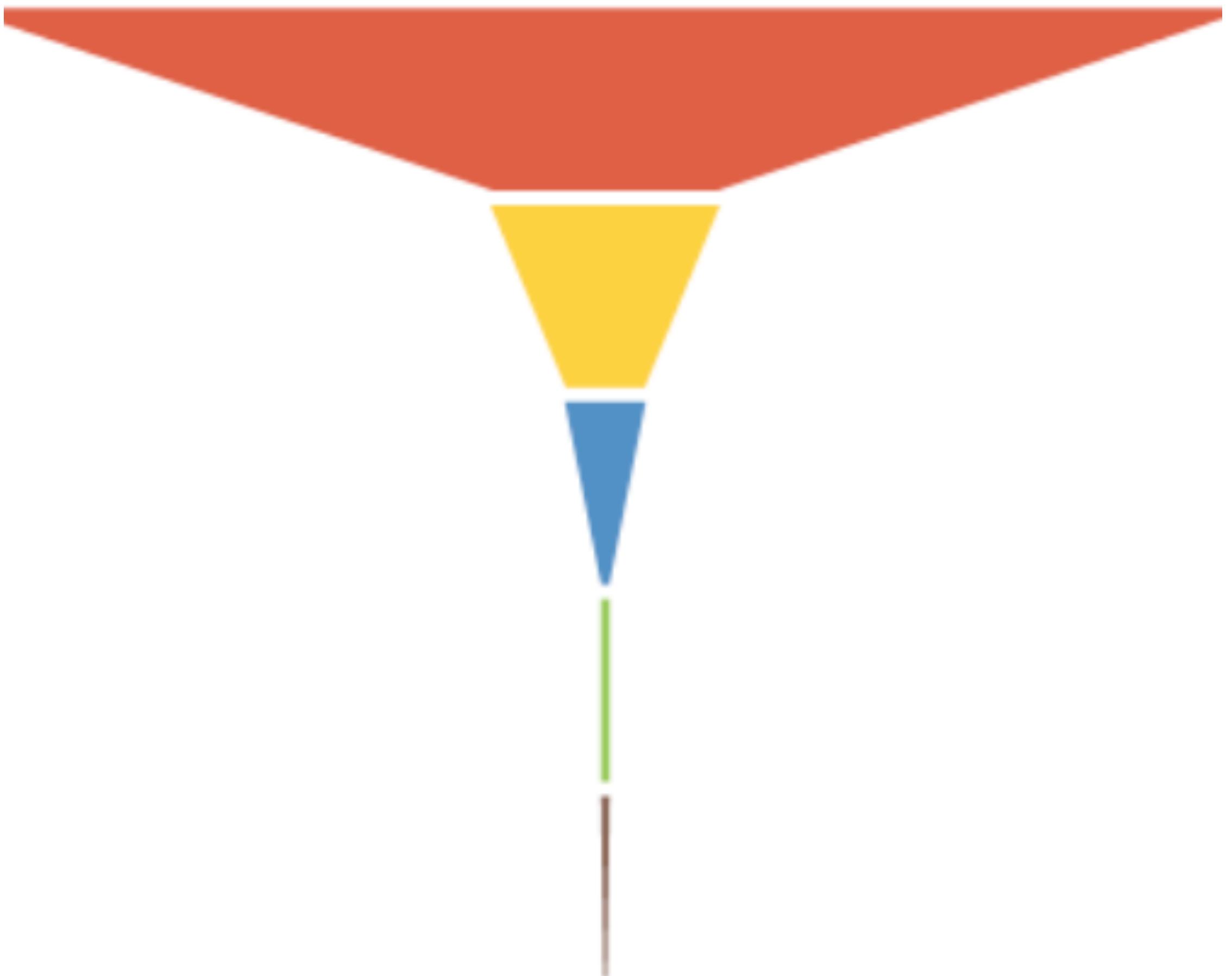


Case: Воронки в AppMetrica

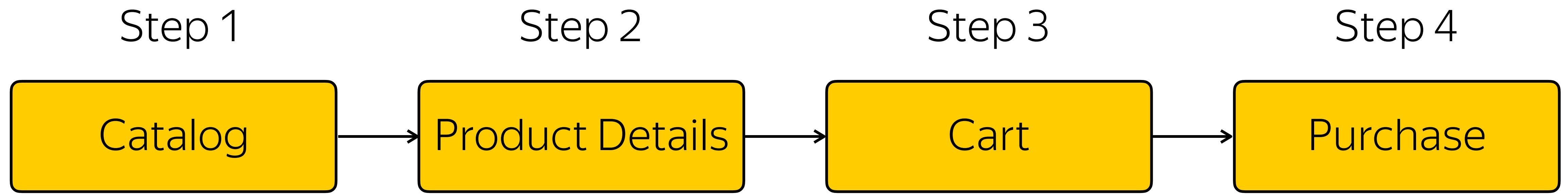


Воронка

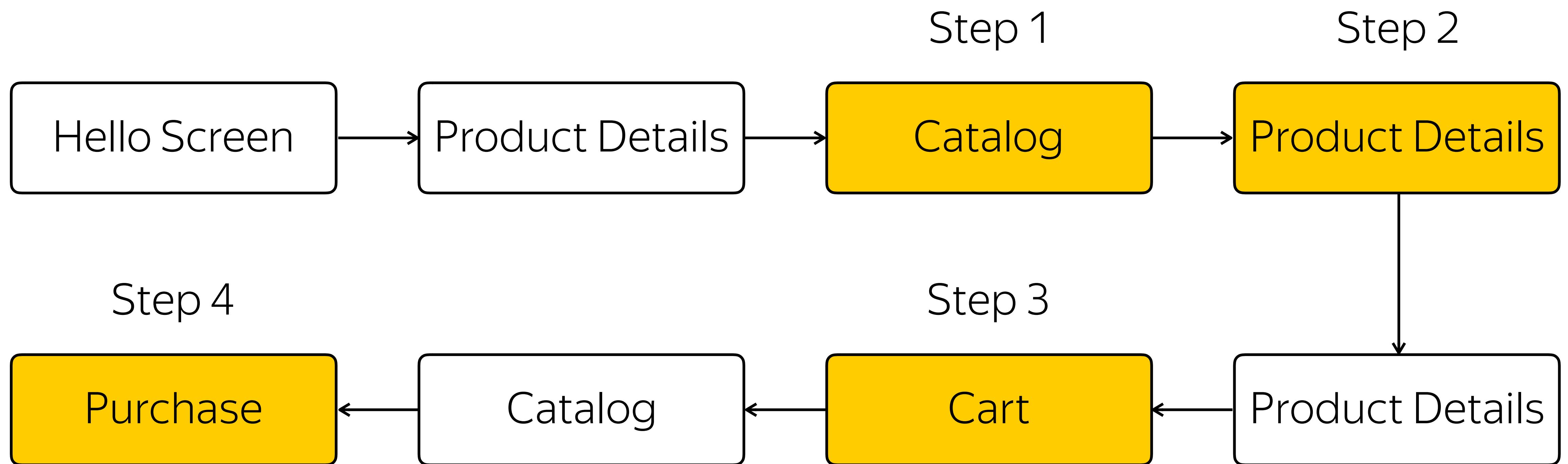
Воронка - это набор шагов, который совершают пользователь на пути к конверсии



Theory



Practice





Conversion rate =
converted users / total
users

Считаем воронку в ClickHouse



Данные

DeviceID	EventName	EventTimestamp
6137721194148732620	Hello Screen	1506718845
6137721194148732620	Product Details	1506718845
6137721194148732620	Catalog	1506724083
6137721194148732620	Product Details	1506724093
6137721194148732620	Product Details	1506752725

Поехали!

Посчитаем воронку из 3х событий

1. Catalog
2. Product
3. Purchase



Подход «в лоб»

sequenceMatch(pattern)
(time, cond1, cond2, ...)

- › Агрегатная функция
- › Сопоставляет последовательность событий с pattern
- › Возвращает 1 или 0



Первый подход

```
SELECT
DeviceID,
max((EventName = 'Catalog')) as step1_condition,
sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),
(EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,
sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),
(EventName = 'Catalog'), (EventName = 'Product'),
(EventName = 'Purchase')) as step3_condition
FROM mobile.events_all
GROUP BY DeviceID
```

Первый подход

```
SELECT  
    DeviceID,  
    max((EventName = 'Catalog')) as step1_condition,  
    sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),  
        (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,  
    sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),  
        (EventName = 'Catalog'), (EventName = 'Product'),  
        (EventName = 'Purchase')) as step3_condition  
FROM mobile.events_all  
GROUP BY DeviceID
```

Первый подход

```
SELECT  
    DeviceID,  
    max((EventName = 'Catalog')) as step1_condition,  
    sequenceMatch('(?1).*(?2)')(toDateTime(EventTimestamp),  
        (EventName = 'Catalog'), (EventName = 'Product')) as step2_condition,  
    sequenceMatch('(?1).*(?2).*(?3)')(toDateTime(EventTimestamp),  
        (EventName = 'Catalog'), (EventName = 'Product'),  
        (EventName = 'Purchase')) as step3_condition  
FROM mobile.events_all  
GROUP BY DeviceID
```

Ответ БД

DeviceID	step1_condition	step2_condition	step3_condition
6137721194148732620	1	1	0
6804205139374406489	1	0	0
5643727272341720158	1	1	1
2272625990964812745	0	0	0
8307742578078641757	1	0	0

Суммируем всё

```
SELECT  
    sum(step1_condition) as step1_achieved,  
    sum(step2_condition) as step2_achieved,  
    sum(step3_condition) as step3_achieved  
FROM (...)
```

step1_achieved	step2_achieved	step3_achieved
22757	9230	2353



Happy end?

Добавим еще один шаг

Будем считать воронку из 4 событий

1. Catalog
2. Product
3. Cart
4. Purchase





Exception: Pattern
application proves too
difficult, exceeding
max iterations

Root cause

Сложность вычислений растет экспоненциально с добавлением шагов из-за .* (последовательность произвольных событий)

Для текущей реализации `sequenceMatch` всегда есть предел на число шагов.



Можем считать по-другому

Условие на достижение второго шага

- › Пользователь посмотрел каталог хотя бы раз
- › Пользователь посмотрел на продукт хотя бы раз
- › Время первого просмотра каталога \leq время последнего просмотра описания продукта



Алгоритм

Шаг 1

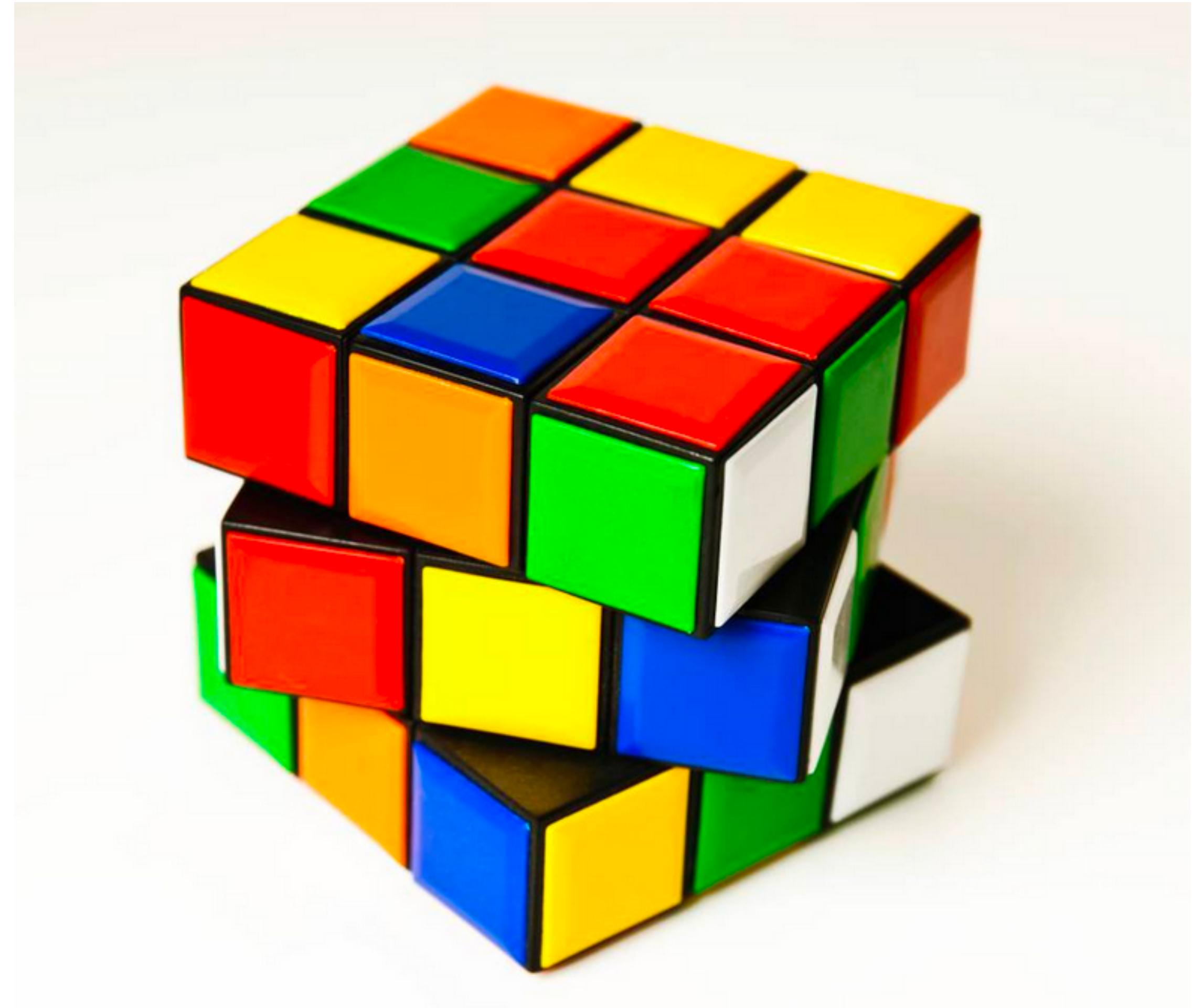
- › Считаем время первого просмотра каталога

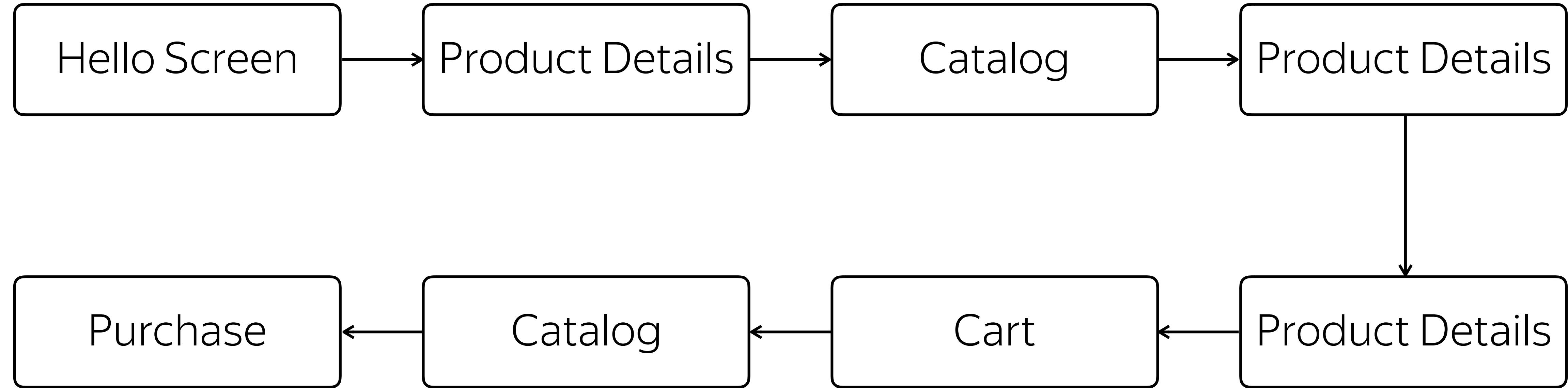
Step 2

- › Считаем время первого просмотра описания продукта после просмотра каталога

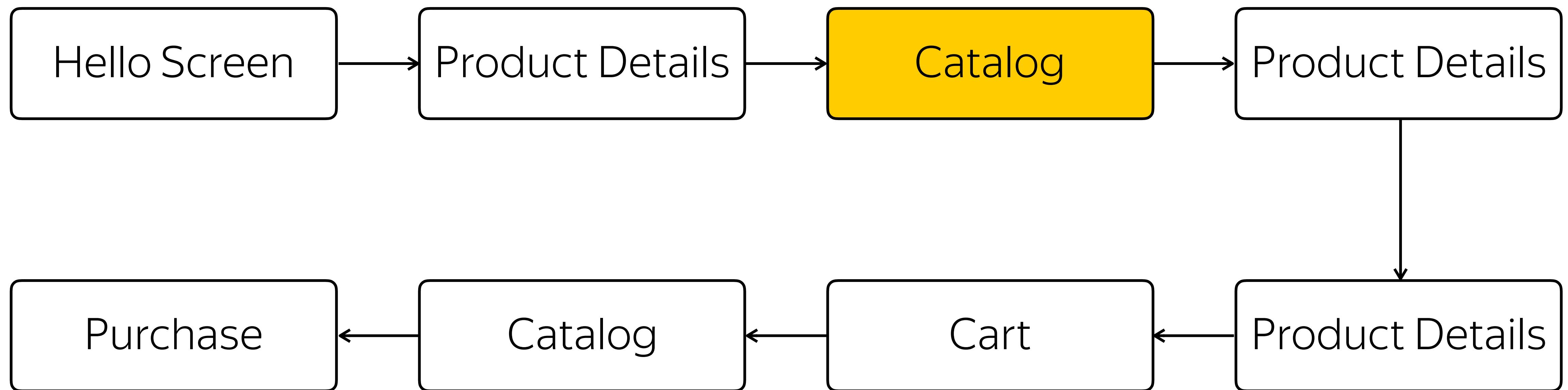
Step 3:

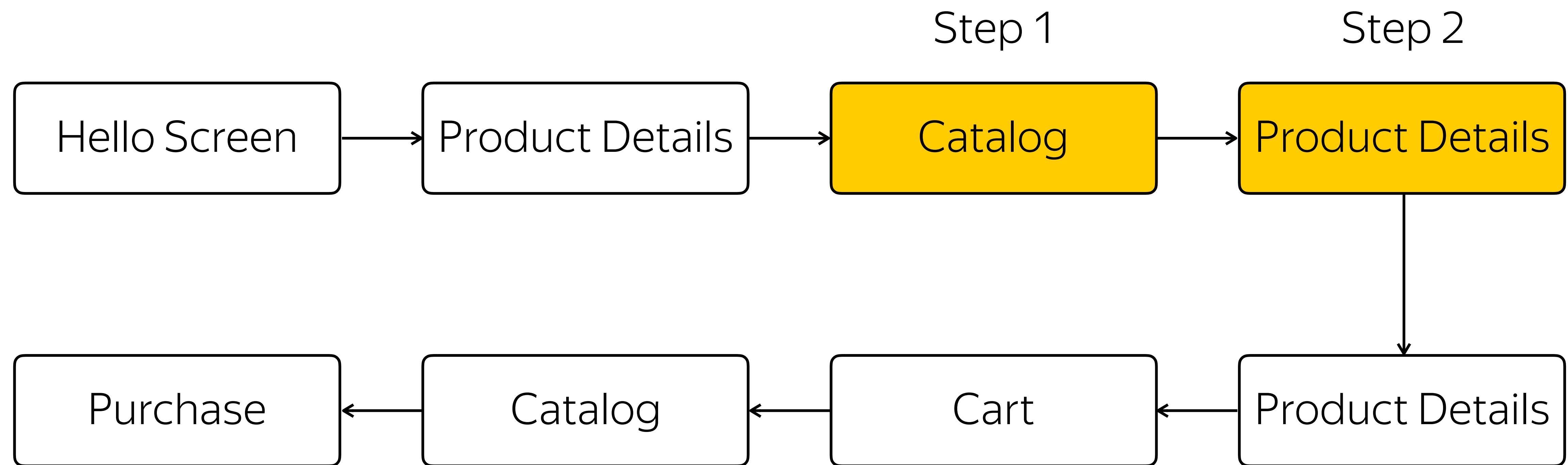
- › И т.д.

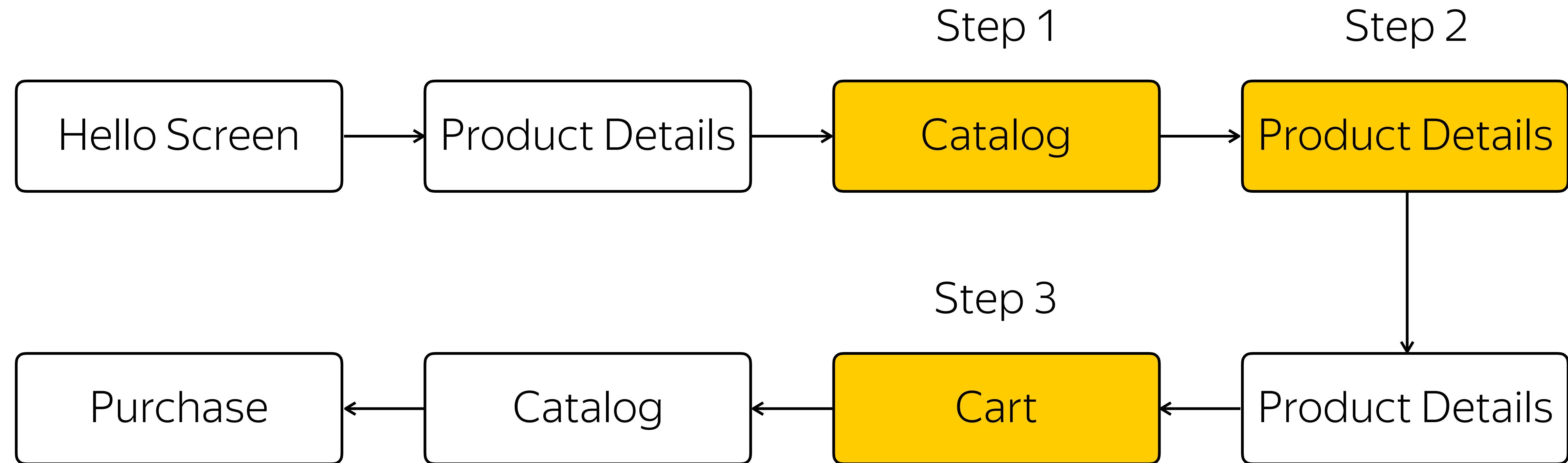


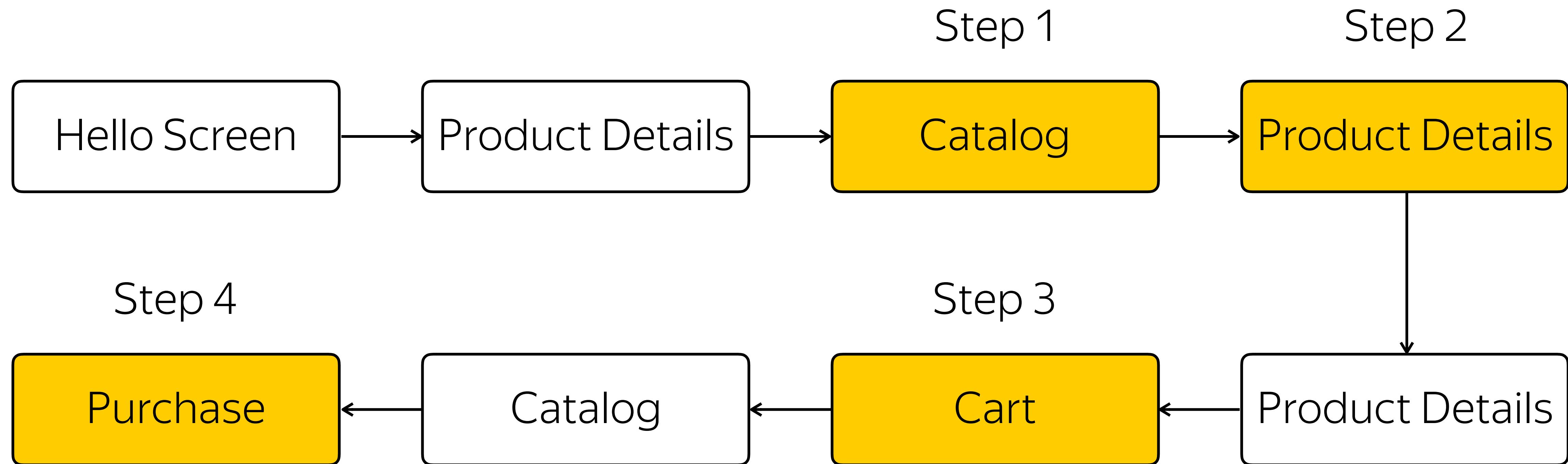


Step 1









Считаем время достижения целей

```
SELECT  
    groupArray(EventName) as events,  
    groupArray(EventTimestamp) as events_times,  
    arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,  
    arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)  
        AND (step1_time != 0), events_times, events)[1] as step2_time,  
    arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)  
        AND (step2_time != 0), events_times, events)[1] as step3_time  
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

Считаем время достижения целей

```
SELECT  
    groupArray(EventName) as events,  
    groupArray(EventTimestamp) as events_times,  
    arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,  
    arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)  
        AND (step1_time != 0), events_times, events)[1] as step2_time,  
    arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)  
        AND (step2_time != 0), events_times, events)[1] as step3_time  
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

Считаем время достижения целей

```
SELECT  
    groupArray(EventName) as events,  
    groupArray(EventTimestamp) as events_times,  
    arrayFilter(time, name -> (name = 'Product'), events_times, events)[1] as step1_time,  
    arrayFilter(time, name -> (name = 'Cart') AND (time >= step1_time)  
        AND (step1_time != 0), events_times, events)[1] as step2_time,  
    arrayFilter(time, name -> (name = 'Purchase') AND (time >= step2_time)  
        AND (step2_time != 0), events_times, events)[1] as step3_time  
FROM (SELECT * FROM mobile.events_all ORDER BY EventTimestamp)
```

Считаем финальные цифры

```
SELECT  
    sum(step1_time != 0) as step1_achieved,  
    sum(step2_time != 0) as step2_achieved,  
    sum(step3_time != 0) as step3_achieved,  
    sum(step4_time != 0) as step4_achieved  
FROM (...)
```

step1_achieved	step2_achieved	step3_achieved
22757	9230	2353



Оно работает :)

Резюмируем

ClickHouse - отличный выбор СУБД для задач аналитики.

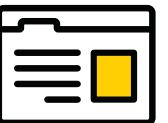
- › Массивы и функции высшего порядка позволяют вычислять сложные метрики.
- › Высокая эффективность позволяет проверять много гипотез.



Спасибо!
Вопросы?



Контакты

-  <https://clickhouse.yandex/>
-  **Google Group** groups.google.com/forum/#!forum/clickhouse
-  clickhouse-feedback@yandex-team.ru
-  [@clickhouse_en](https://t.me/clickhouse_en)
-  github.com/yandex/ClickHouse
-  [@ClickHouseDB](https://twitter.com/ClickHouseDB)