От сырого кликстрима к чистым датасетам или история развития Feature Storage в Lamoda

Дана Злочевская

Team leader of
Ranking & Recommendations

Михаил Нестеров

Senior Data Engineer

latech

О себе

- Data Scientist
- В LaTech занимаюсь ранжированием и рекомендациями

Дана Злочевская

Team leader of
Ranking & Recommendations

- Инженер данных.
- Участвовал в разработке хранилищ как с использованием реляционных СУБД, так и стека технологий Hadoop.

Михаил Нестеров

Senior Data Engineer

latech

Lamoda Tech в цифрах

100+ профессионалов

Data Scientist'ы, Data Engineer'ы, продуктовые аналитики, бизнес-аналитики, системные аналитики, project и product менеджеры.

1,5 Пбайт

Размер нашего DWH и Data Lake Hadoop. 245 000 процессов обеспечивают загрузку более 200 ГБ данных ежедневно.

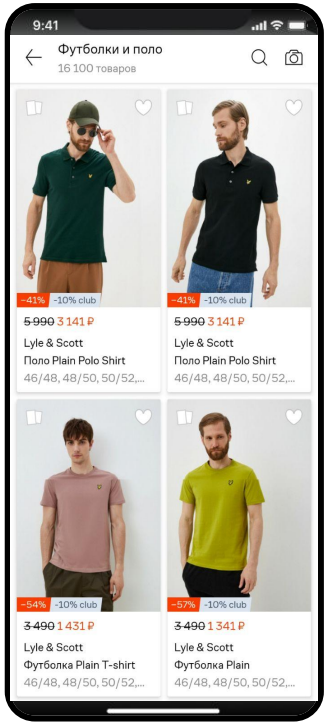
50+ ML пайплайнов и 100+ DAG-ов для обработки данных

Регулярно обучаются и обновляют модели для эффективной работы: ранжирование, рекомендации, сегментирование, предсказание оттока, прайсинг и т.д.

Зоопарк технологий D&A

latech

Основные источники данных - clickstream



kafka

Spark Streaming

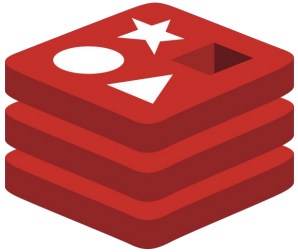
hadoop

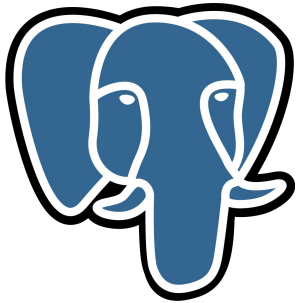
Hive-таблица с сырыми событиями

latech

Основные источники данных - бэк базы данных

1. Данные по текущему стоку и его доступности
2. Информация по заказам, возвратам
3. Информация по пользователям - их скидки, устройства, гео





ORACLE®

Регулярно раз в день / в час

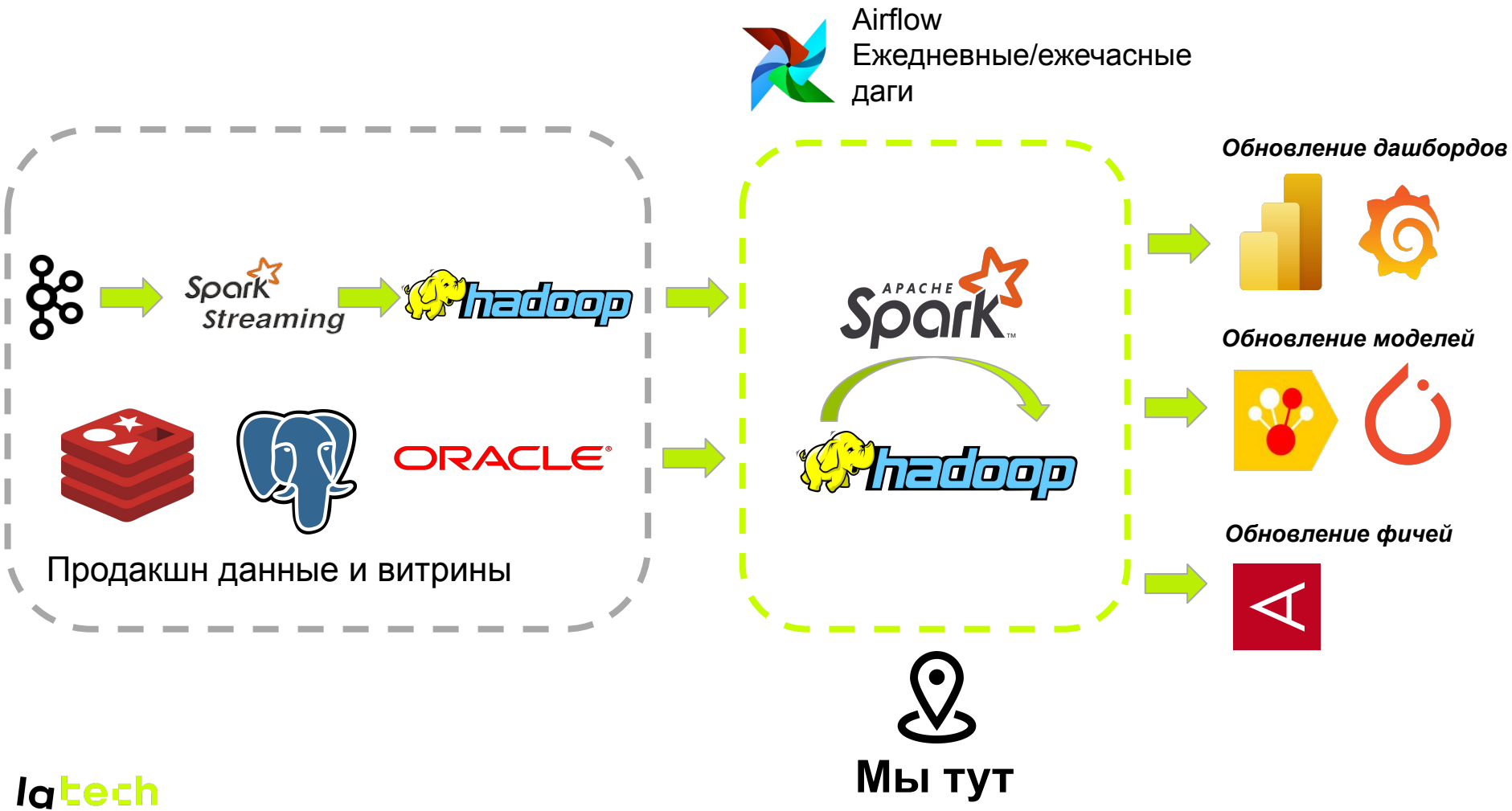
1. Рассчитывать метрики для дашбордов
2. Собирать датасеты для моделей машинного обучения
3. Обновлять фичей для работы моделей

Обновление дашбордов

Обновление моделей

Обновление фичей

latech


Airflow
Ежедневные/ежечасные даги
Spark Streaming
hadoop
ORACLE
Продакшн данные и витрины
APACHE Spark
hadoop
Обновление дашбордов
Обновление моделей
Обновление фичей
Мы тут

latech

Эволюция подхода к работе с данными для батч-моделей

Нет единого подхода к хранению и формированию датасетов

< 2021

Внедрили единый подход к хранению датасетов - Feature Storage

2021

Внедрили новый слой данных для работы с данными кликстрима - Action Storage

2023

latech

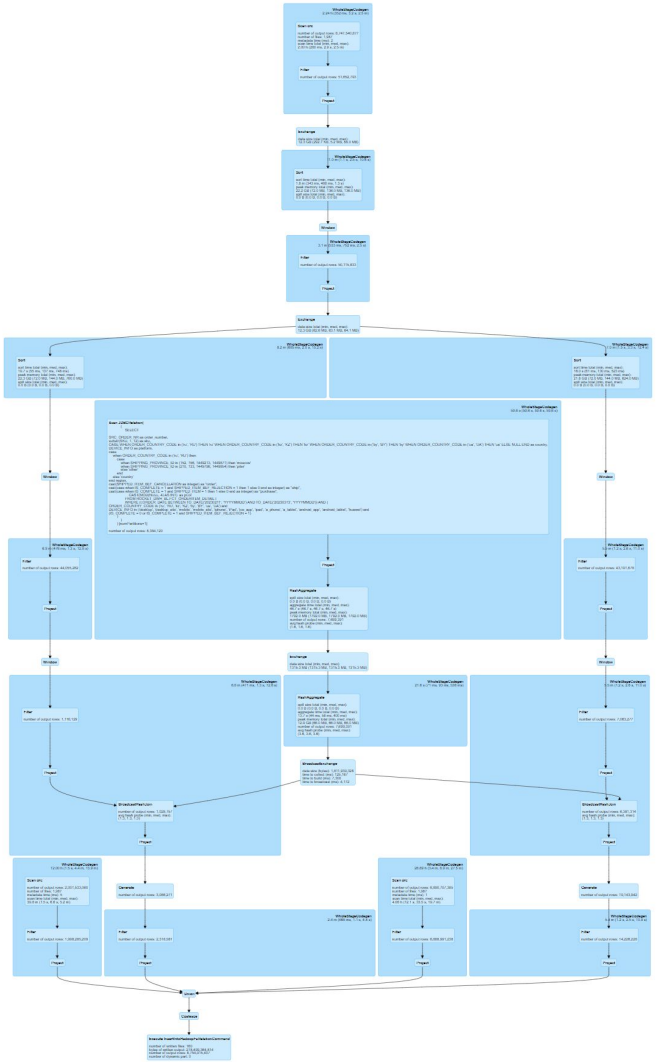
Не было единого подхода к хранению промежуточных данных, датасеты сохранялись в hive/hdfs в произвольном формате

Одни и те же операции с данными дублировались в разных DAG-ах, в том числе тяжелые запросы

Вся логика работы с данными находилась в одном DAG-е

В чем боль?

- Даги стало сложно поддерживать
- Дублирование логики и данных
- Невозможность переиспользования данных
- Нет единого формата хранения датасетов
- Отсутствие прозрачной логики версионирования



Что хотели получить?

Прийти к новой архитектуре, которая будет обладать:

1

Прозрачностью в плане логики и пайплайнов

- Каждый этап обработки данных должен выполняться отдельно
- Исполнение моделей должно выполняться изолированно от вычисления фичей

2

Конфигурируемостью и в то же время гибкостью

- Понятно как добавлять новые фичи
- Есть единый способ хранения, подходящий под все типы данных

3

Воспроизводимостью работы

- Можем получить значения фичей на любой момент времени в прошлом

Feature Storage

latech

Feature storage - наш фреймворк для работы с данными

- Зафиксировали единый способ хранения обработанных данных **- Feature Set**
- Реализовали пакет на PySpark, унифицирующий сценарии сохранения/чтения FeatureSet-ов - **Feature Storage**
- **Внедрили новую логику построения пайплайнов данных**, разделяющая этапы обработки данных:

1. **Каждый Feature Set вычисляется в своем даге**
   Теперь каждый этап обработки данных выполняется изолированно

2. **Реализован механизм версионирования**
   Теперь можно вычислять и хранить несколько версий одного FeatureSet-а

3. **Внедрили общий принцип партиционирования:** по версии, ключу и времени расчёта
   Для всех данных прозрачен способ работы с ними

latech

Пример FeatureSet

Feature Set: Контентные фичи по товарам

Ключ: ['ts_nodash', 'location', 'sku']
Версия: v2
Партиции: Версия + ключ (названия колонок)

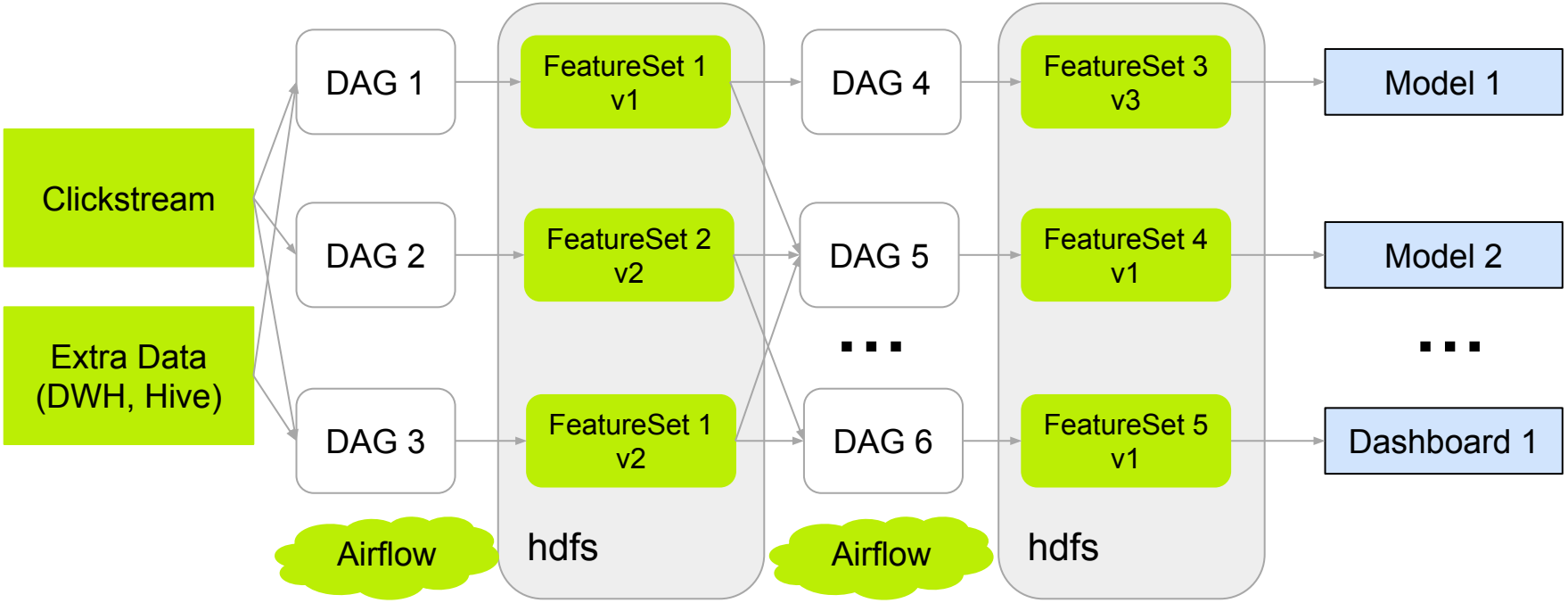
sku	location	sku_all_categories	sku_gender	sku_site_section	sku_brand_id	sku_brand_name	sku_is_new	sku_is_discount	ts_nodash
MP002XW0REKY	ru	[1, 369, 2, 355, ...	women	REGULAR	25196	Vittoria Vicci	0	1	20211119T020000
ST040AWBAQL2	by	[15, 1, 2, 39, 4153]	women	WHITE	24500	Style Shoes	0	1	20211119T020000
MP002XM24ZIR	ru	[3816, 491, 1, 47...	men	REGULAR	25021	Bask	0	0	20211119T020000

```
[dana.zlochevskaya@jupyterhub-rnd-1 monorep]$ hdfs dfs -ls /products/feature_storage/features/sku_content_features
Found 2 items
drwxr-xr-x   - airflow hdfs          0 2021-03-19 21:03 /products/feature_storage/features/sku_content_features/default
drwxr-xr-x   - airflow hdfs          0 2021-11-24 12:48 /products/feature_storage/features/sku_content_features/v2
```

```python
from feature_storage import FeatureStorage

spark = ...
spark_df = spark.table(...).filter(...).withColumn(...).groupBy(...).agg(...)
fs = FeatureStorage(
    feature_storage_hdfs_path="/products_dev/feature_storage/features",
    spark=spark,
)

fs.write_feature_set(
    data=spark_df,
    name="my_feature_set_name",
    key=["ts_nodash", "country", ... , "SKU"], # your featureset must contain "ts_nodash" key (column)
    version="default",
    num_partitions=256, # Optional
    sort_columns=["country", ...], # Optional
    full_sort=False, # False is default
    compress_codec="zlib", # Optional
)
```


Clickstream
Extra Data
(DWH, Hive)
DAG 1
DAG 2
DAG 3
Airflow
FeatureSet 1
v1
FeatureSet 2
v2
FeatureSet 1
v2
hdfs
DAG 4
DAG 5
...
DAG 6
Airflow
FeatureSet 3
v3
FeatureSet 4
v1
FeatureSet 5
v1
hdfs
Model 1
Model 2
...
Dashboard 1

Примеры популярных фича-сетов

- Основные события кликстрима
- Атрибуция целевых событий
- Признаки взаимодействия пользователя и товаров
- Фичи для товарам, фичи для пользователей
- Агрегаты данных разных уровней для построения дашбордов и регулярной аналитики

latech

Пример внедрения сжатия файлов

Наши FeatureSet стали занимать слишком много места, нужно было его оптимизировать

Решение: Благодаря единому способу работы с датасетами через Feature Storage, достаточно было внедрить механизм компрессии в FeatureStorage и начать его использовать в каждом FeatureSet

```
FS_ZLOG_ACTIONS_NAME=zlog_actions_for_ranking
FS_ZLOG_ACTIONS_VERSION=v2
FS_ZLOG_ACTIONS_KEY=ts_nodash,platform,location,uid,session_id,action_name,event_ts,sku
```

Пример внедрения сжатия файлов

Наши FS стали занимать слишком много места, нужно было его оптимизировать

Решение: Благодаря единому способу работы с датасетами через Feature Storage, достаточно было внедрить механизм компрессии в FeatureStorage и начать его использовать в каждом FeatureSet

```
FS_ZLOG_ACTIONS_NAME=zlog_actions_for_ranking
FS_ZLOG_ACTIONS_VERSION=v2
FS_ZLOG_ACTIONS_KEY=ts_nodash,platform,location,uid,session_id,action_name,event_ts,sku
```

Новая версия пакета Feature Storage

```
FS_ZLOG_ACTIONS_NAME=zlog_actions_for_ranking
FS_ZLOG_ACTIONS_VERSION=v2
FS_ZLOG_ACTIONS_KEY=ts_nodash,platform,location,uid,session_id,action_name,event_ts,sku

FS_ZLOG_ACTIONS_SORT_COLUMNS=location,platform,action_name,session_id
FS_ZLOG_ACTIONS_FULL_SORT=false
FS_ZLOG_ACTIONS_COMPRESS_CODEC=zlib
```

Пришло время двигаться дальше

latech

Сложности работы с событиями кликстрима

Основной источник данных для ML событий - **кликстрим (50 ТБ +)**

1. События кликстрима используются в большинстве моделей, нужен доступ к событиям изолированно
2. Каждое событие имеет свой набор атрибутов и требует отдельный обработчик
3. События кликстрима меняются, необходимо бесшовно переходить на новую версию при сохранении историчности
4. Нужно сразу реагировать на изменение потока получаемых данных и проблемы в них

Сложности работы с событиями кликстрима

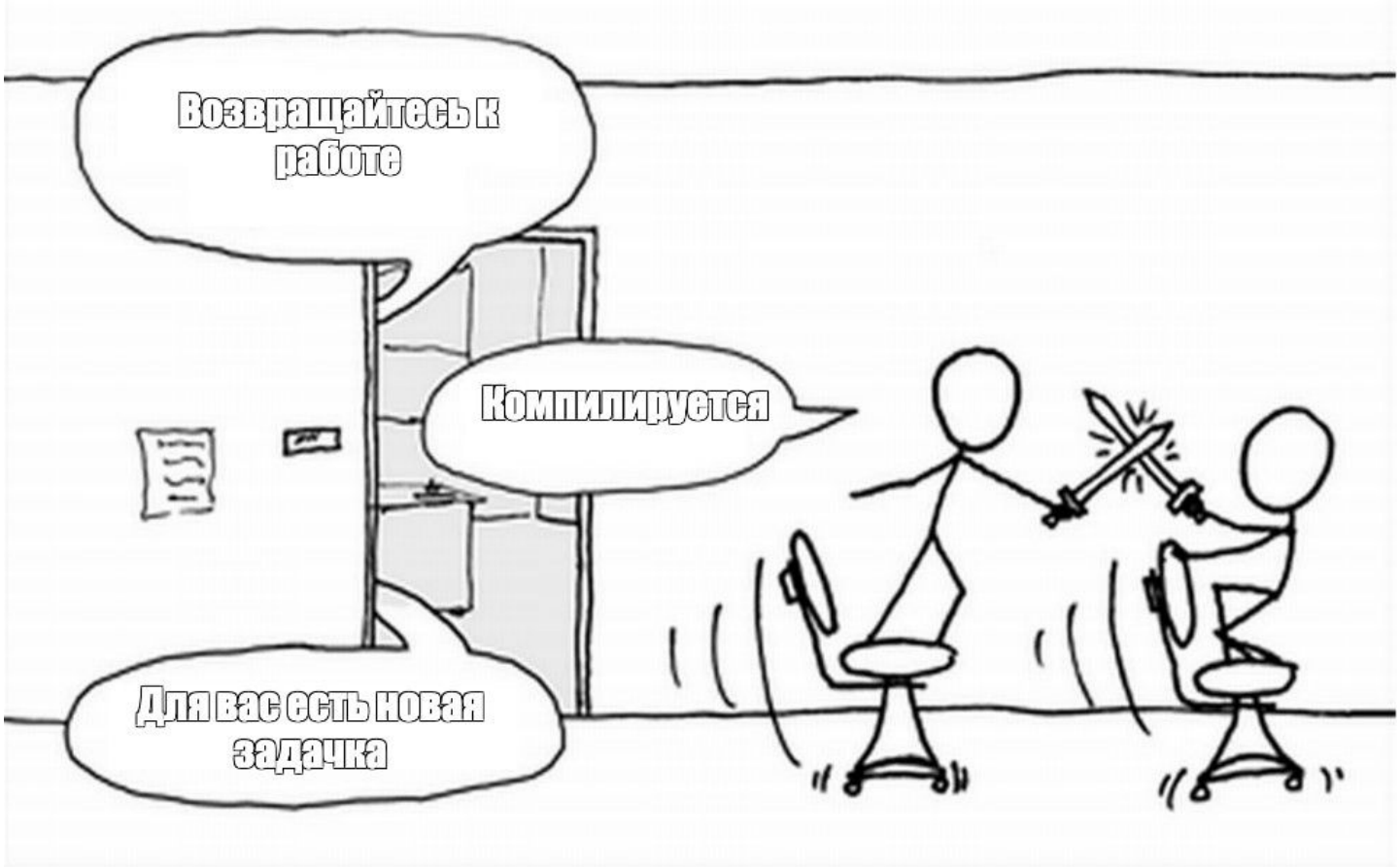
Основной источник данных для ML событий - **кликстрим (50 ТБ +)**

1. События кликстрима используются в большинстве моделей, нужен доступ к событиям изолированно
2. Каждое событие имеет свой набор атрибутов и требует отдельный обработчик
3. События кликстрима меняются, необходимо бесшовно переходить на новую версию при сохранении историчности
4. Нужно сразу реагировать на изменение потока получаемых данных и проблемы в них

Решение: *Хранить важные события кликстрима в одном месте единым способом = Action Storage*

Action Storage

latech


Возвращайтесь к работе
Компилируется
Для вас есть новая задачка

latech

Требования к инструменту

Дать возможность изолированно создавать/изменять событие при необходимости, создавая его новую версию

Требования к инструменту

Дать возможность изолированно создавать/изменять событие при необходимости, создавая его новую версию

Добавить валидацию данных для своевременного реагирования на проблемы

Требования к инструменту

Дать возможность изолированно создавать/изменять событие при необходимости, создавая его новую версию

Добавить валидацию данных для своевременного реагирования на проблемы

Обеспечить единым способом работы с событиями из clickstream

Обладать простой кодовой базой для чтения/работы с событиями

Требования к инструменту

Дать возможность изолированно создавать/изменять событие при необходимости, создавая его новую версию

Добавить валидацию данных для своевременного реагирования на проблемы

Обеспечить единым способом работы с событиями из clickstream

Обладать простой кодовой базой для чтения/работы с событиями

DVC

PyDeequ

great expectations

Требования к инструменту

Дать возможность изолированно создавать/изменять событие при необходимости, создавая его новую версию

Добавить валидацию данных для своевременного реагирования на проблемы

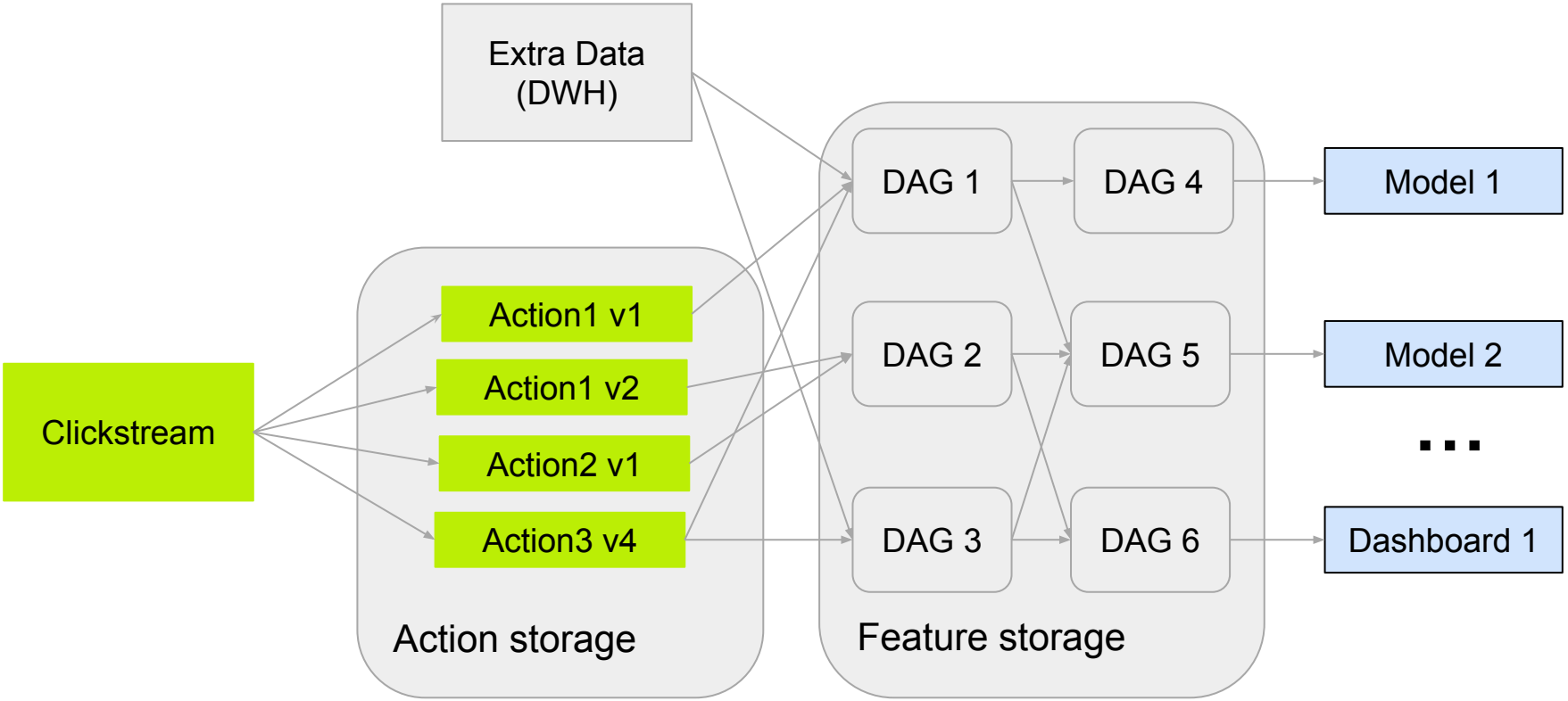
Обеспечить единым способом работы с событиями из clickstream

Обладать простой кодовой базой для чтения/работы с событиями

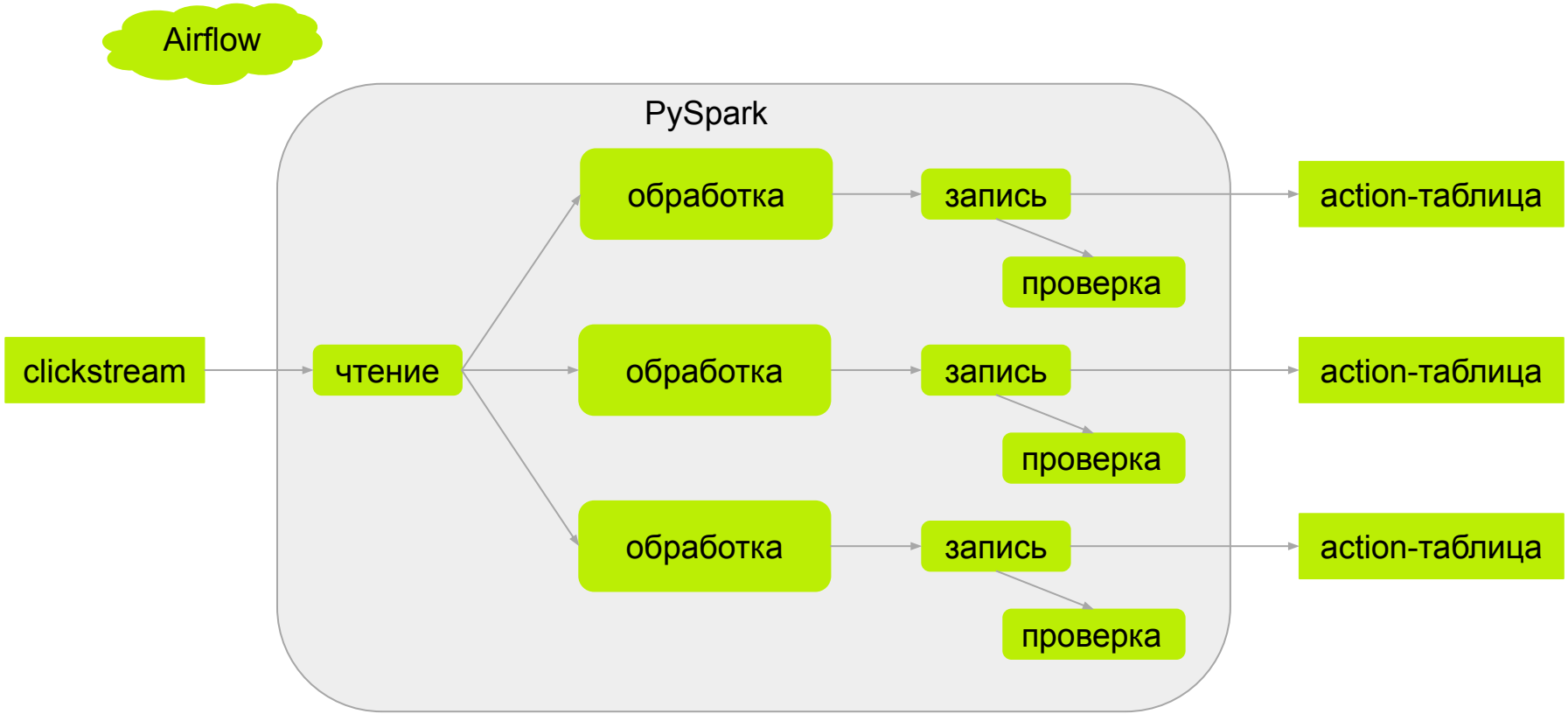
DVC

PyDeequ

great expectations

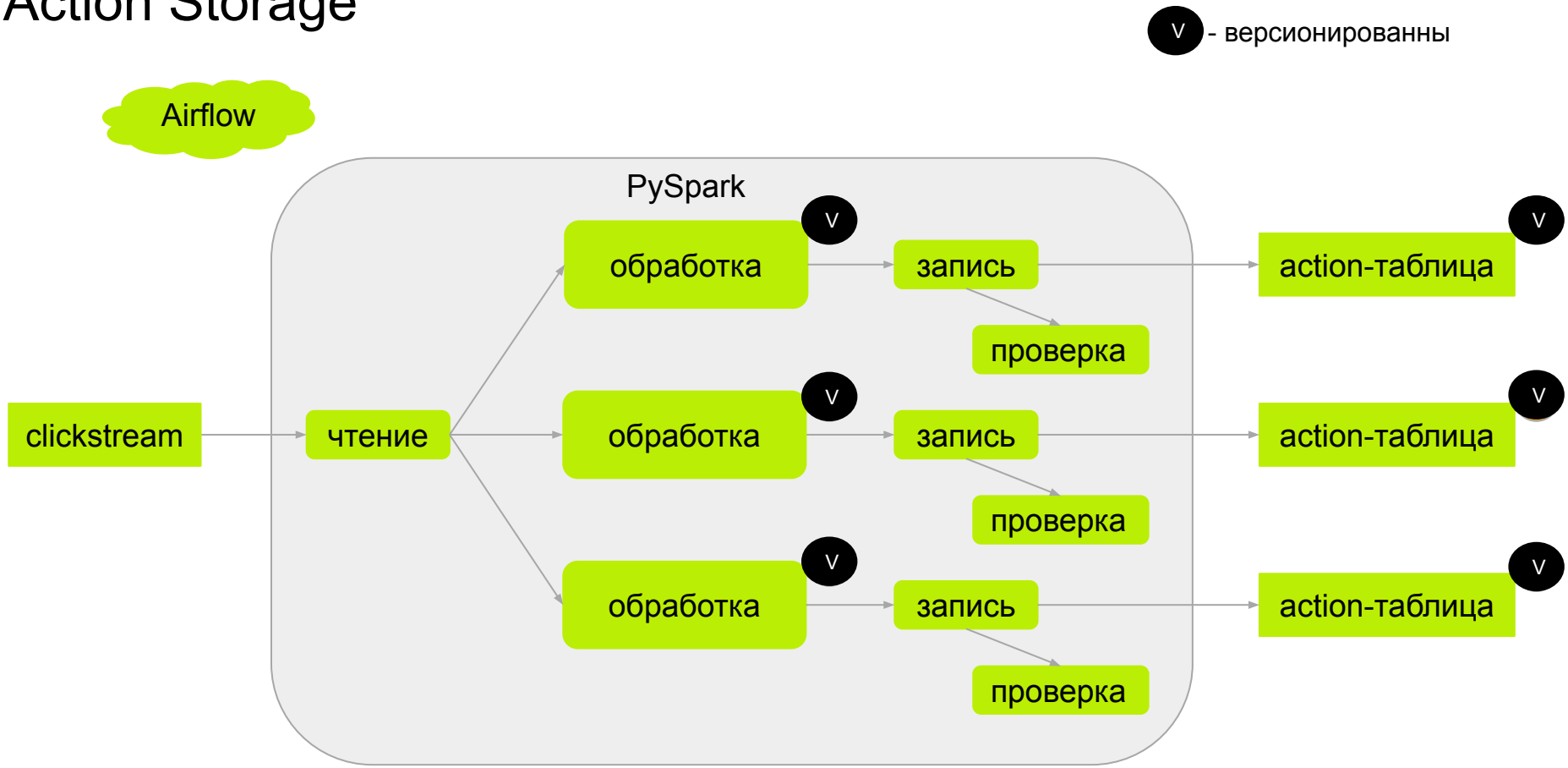

Extra Data
(DWH)
Clickstream
Action1 v1
Action1 v2
Action2 v1
Action3 v4
Action storage
DAG 1
DAG 2
DAG 3
DAG 4
DAG 5
DAG 6
Feature storage
Model 1
Model 2
...
Dashboard 1

Action Storage



latech

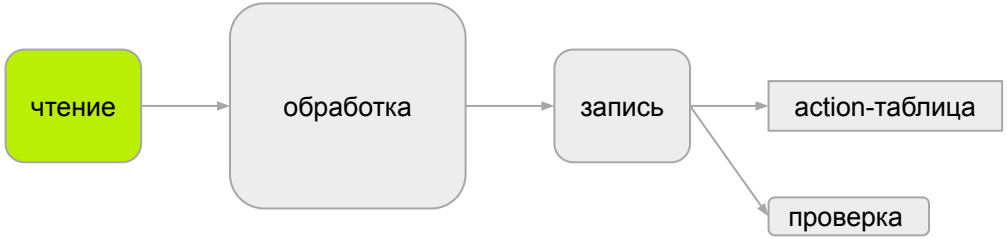
Action Storage



latech

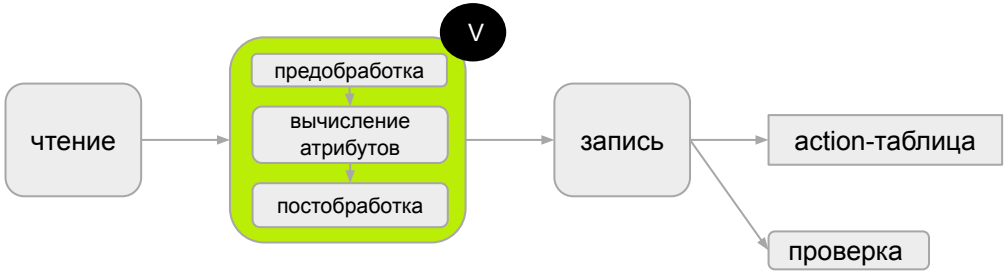
Чтение

1. Читаем порцию (1 день)
2. Cache



Обработка

1. Версионирование
2. Основные этапы:
   a. Предобработка
   b. Вычисление атрибутов
   c. Постобработка

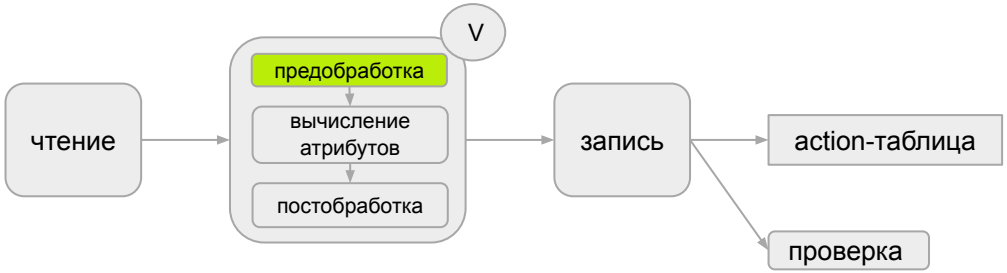



```python
class ClickCatalogV1:

    @classmethod
    def parse(cls, input_df: DataFrame) -> DataFrame:
        pre_transformed = cls.pre_transform(input_df)
        parsed = cls.column_parse(pre_transformed)
        post_transformed = cls.post_transform(parsed)
        return post_transformed
```

Предобработка

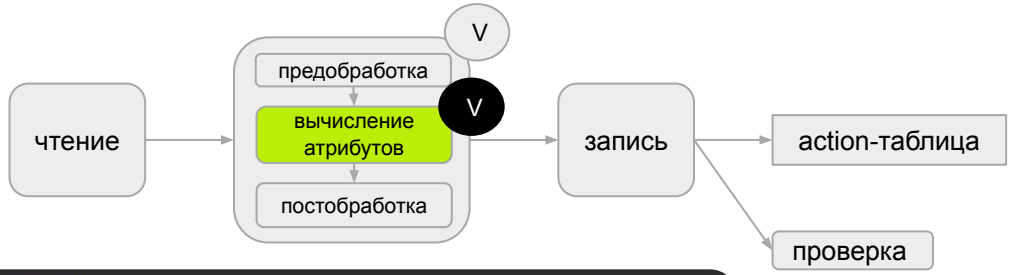
1. Фильтрация
2. Плоская структура
3. Простые преобразования




```
@staticmethod
def pre_transform(df):
    return (
        df.filter(F.col("action") == "catalog_click")
        .withColumn('sku', F.explode(F.split(F.col('skus'), pattern: '[;,]')))
    )
```

Вычисление атрибутов

1. Атрибут - обработчик
   a. Версионирование
   b. Без фильтров
   c. Без join-ов



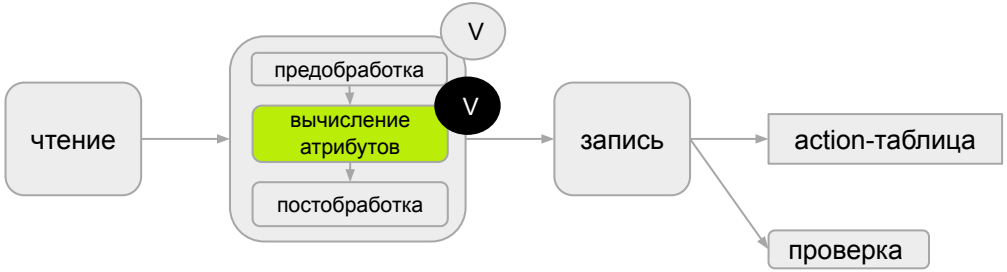

```
column_parsers = OrderedDict(
    platform=PlatformParserV1,
    location=None,
    sku=ExplodedSkuParserV1,
    dt=None,
)

@staticmethod
def column_parse(df):
    for col_name, col_parser in cls.column_parsers.items():
        if col_parser is not None:
            pre_transformed = col_parser.parse(pre_transformed, col_name)

    return pre_transformed.select(*cls.column_parsers.keys())
```

Вычисление атрибутов

1. Атрибут - алгоритм
   a. Версионирование
   b. Без join-ов
   c. Без фильтров

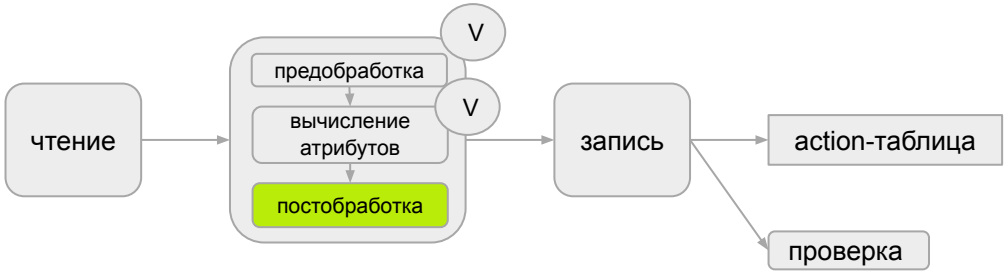



```python
class ExplodedSkuParserV1:

    @staticmethod
    def parse(df: DataFrame, col_name: str):
        return df.withColumn(
            col_name,
            F.col(col_name)
             .substr(1, 12)
        )
```

Постобработка

1. Фильтрация
2. Простые преобразования
3. Дедупликация




```python
@staticmethod
def post_transform(df):
    distinct_cols = get_cols_to_deduplicate(df)
    return (
        df.transform(default_sku_filter)
        .dropDuplicates(distinct_cols)
    )
```

Пример обработки события

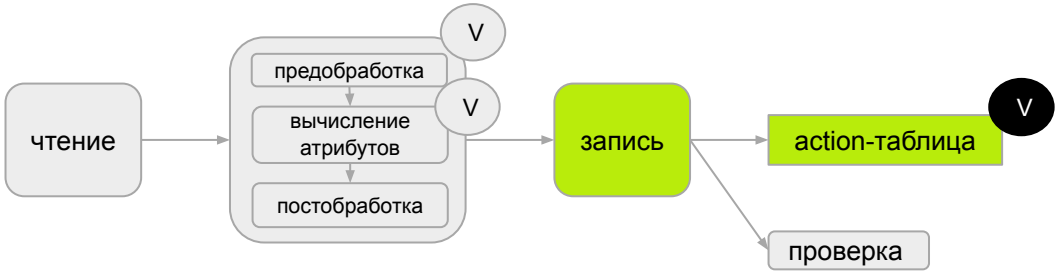
```python
class ClickCatalogV1(BaseActionDataMaker):
    column_parsers = OrderedDict(
        platform=PlatformParserV1,
        location=None,
        sku=ExplodedSkuParserV1,
        dt=None,
    )
```

```python
@staticmethod
def pre_transform(df):
    return (
        df.filter(F.col("action") == "catalog_click")
        .withColumn('sku', F.explode(F.split(F.col('skus'), pattern: '[;,]')))
    )

@staticmethod
def post_transform(df):
    distinct_cols = get_cols_to_deduplicate(df)
    return (
        df.transform(default_sku_filter)
        .dropDuplicates(distinct_cols)
    )
```

Запись

1. Таблица версонирована
2. Регулирование числа партиций
3. Сортировка и сжатие




```
> show tables from action_storage_db;
+-------------------------+
|          tab_name       |
+-------------------------+
| cart_add__v1            |
| catalog_item_viewed__v1 |
| click_catalog__v1       |
| click_recsys__v1        |
| fav_add__v1             |
| impression_recsys__v1   |
| order__v1               |
| product_page__v1        |
+-------------------------+
```

Проверка качества данных

```
location {
  allowedValues {
    value = ["ru", "kz", "by", "ua"]
    severity = "warning"
  }
}
platform {
  allowedValues = [
      "fake_app"
    ]
}
sku {
  rlike = "^[A-Z0-9]{12}$"
}
```

Настройка в Airflow

PyDeequ:
https://github.com/awslabs/python-deequ

```
location {
  allowedValues {
    value = ["ru", "kz", "by", "ua"]
    severity = "warning"
  }
}
platform {
  allowedValues = [
      "fake_app"
    ]
}
sku {
  rlike = "^[A-Z0-9]{12}$"
}
```

+

platform	location	sku	dt
android_app	by	MP002XB024KI	2023-08-05
android_app	by	MP002XW0LV1U	2023-08-05
android_app	by	RTLABI733601	2023-08-05
android_app	by	RTLACU964801	2023-08-05
android_app	by	RTLABM655701	2023-08-05

=

```
location {
  allowedValues {
    value = ["ru", "kz", "by", "ua"]
    severity = "warning"
  }
}
platform {
  allowedValues = [
      "fake_app"
    ]
}
sku {
  rlike = "^[A-Z0-9]{12}$"
}
```

+

platform	location	sku	dt
android_app	by	MP002XB024KI	2023-08-05
android_app	by	MP002XW0LV1U	2023-08-05
android_app	by	RTLABI733601	2023-08-05
android_app	by	RTLACU964801	2023-08-05
android_app	by	RTLABM655701	2023-08-05

=

constraint	check_level	constraint_status
Compliance(location contained in ru,kz,by,ua,`location` IS NULL OR …	Warning	Success
Compliance(sku regex,sku rlike '(^[A-Z0-9]{12}$)',None)	Error	Success
Compliance(platform contained in fake_app,`platform` IS NULL OR `pl…	Error	Failure

latech

Планы развития

- “Сложные” события
- Развитие проверки качества данных
- Перевести оставшиеся загрузки на action-таблицы
- Каталог данных

latech

Выводы

- Feature Storage и Action Storage позволяют нам:
  - Сократить TTM разработки ML продуктов
  - Иметь прозрачный процесс построения data пайплайнов без дублирования кода и данных
  - Удобно работать с данными на базе единого подхода

- Не бойтесь менять и оптимизировать подход к данным, это итеративный процесс

- Делать кастомные инструменты, которые подходят именно вам, может быть эффективнее чем внедрять сложные open source решения

latech

Контакты

@danazlo

dana.zlochevskaya@lamoda.ru

Дана Злочевская

Team leader of
Ranking & Recommendations

@Nesteroid

mihail.nesterov@lamoda.ru

Михаил Нестеров

Senior Data Engineer

latech

//code the lif
de the lifesty
e latech://co
lifestyle late
//code the lif