**Центр онлайн-обучения «Нетология-групп»**

**Дипломный проект по профессии**

**“Инженер данных”**

Выполнил: Руди Д.Н

февраль, 2024

**Цель работ:**

Сформируйте ETL-процессы на основе датасета.

**Задание:**

1. Необходимо разработать и задокументировать ETL-процессы заливки данных в хранилище, состоящее из слоёв:
   * NDS - нормализованное хранилище
   * DDS - схема звезда;
2. На основании DDS построить дашборд

**Этапы выполнения работ:**

1. Выбрать исходную реляционную БД
2. Скопировать датасет из исходной реляционной СУБД и загрузить его в облачное хранилище объектов
3. Прочитать датасет из хранилища объектов, провести предварительную обратку данных, сохранить результат во временные таблицы
4. Залить данные из временных таблиц в нормализованную схему данных (NDS)
5. Сформируйте состав таблиц фактов и измерений (DDS)
6. Сформируйте набор метрик и дашбордов на основе DDS
7. Оформить результаты
8. **Краткое описание модели данных**

В качестве СУБД в дипломной работе используется PostgreSQL.

В качестве предметной области выбрана реляционная БД авиаперевозки по России (**bookings**). БД состоит из восьми таблиц и двух представлений.

Краткое описание БД (таблицы и представления):

Таблицы:

* Aircrafts – код воздушного судна (ВС), модель ВС, максимальная дальность полёта (км)
* Airports – код аэропорта, название аэропорта, город, координаты (долгота/широта), временная зона аэропорта
* boarding\_passes – номер билета, id рейса, номер посадочного, номер места
* bookings – номер бронирования, дата бронирования, полная сумма бронирования
* flights – id рейса, номер рейса, время вылета и прилета по расписанию, аэропорты отправления и прибытия, статус рейса, код ВС, фактическое время вылета и прилета
* seats – код ВС, номер места, класс обслуживания
* ticket\_flights – номер билета, id рейса, класс обслуживания, стоимость перелета
* tickets – номер билета, номер бронирования, id пассажира, ФИ пассажира, контактные данные пассажира
  1. Aircrafts:
* Каждая модель воздушного судна идентифицируется своим трехзначным кодом (aircraft\_code).
* Указывается также название модели (model) и
* максимальная дальность полета в километрах (range).
  1. Airports:
* Аэропорт идентифицируется трехбуквенным кодом (airport\_code) и имеет свое имя (airport\_name).
* Название города (city) указывается и может служить для того, чтобы определить аэропорты одного города.
* Также указывается широта (longitude), долгота (latitude) и часовой пояс (timezone).
  1. boarding\_passes:
* При регистрации на рейс, которая возможна за сутки до плановой даты отправления, пассажиру выдается посадочный талон. Он идентифицируется также, как и перелет — номером билета и номером рейса.
* Посадочным талонам присваиваются последовательные номера (boarding\_no) в порядке регистрации пассажиров на рейс (этот номер будет уникальным только в пределах данного рейса)
* В посадочном талоне указывается номер места (seat\_no).
  1. bookings:
* Пассажир заранее (book\_date, максимум за месяц до рейса) бронирует билет себе и, возможно, нескольким другим пассажирам.
* Бронирование идентифицируется номером (book\_ref, шестизначная комбинация букв и цифр).
* Поле total\_amount хранит стоимость перелетов
  1. flights:
* Естественный ключ таблицы рейсов состоит из двух полей — номера рейса (flight\_no) и даты отправления (scheduled\_departure).
* Чтобы сделать внешние ключи на эту таблицу компактнее, в качестве первичного используется суррогатный ключ (flight\_id).
* Рейс всегда соединяет две точки — аэропорты вылета (departure\_airport) и прибытия (arrival\_airport). Такое понятие, как «рейс с пересадками» отсутствует: если из одного аэропорта до другого нет прямого рейса, в билет просто включаются несколько необходимых рейсов.
* У каждого рейса есть запланированные дата и время вылета (scheduled\_departure) и прибытия (scheduled\_arrival). Реальные время вылета (actual\_departure) и прибытия (actual\_arrival) могут отличаться: обычно не сильно, но иногда и на несколько часов, если рейс задержан
  1. seats:
* Места определяют схему салона каждой модели. Каждое место определяется своим номером (seat\_no) и имеет закрепленный за ним класс обслуживания (fare\_conditions) — Economy, Comfort или Business
  1. ticket\_flights:
* Перелет соединяет билет с рейсом и идентифицируется их номерами. Для каждого перелета указываются его стоимость (amount) и класс обслуживания (fare\_conditions)
  1. tickets:
* Билет имеет уникальный номер (ticket\_no), состоящий из 13 цифр. Билет содержит идентификатор пассажира (passenger\_id) — номер документа, удостоверяющего личность, — его фамилию и имя (passenger\_name) и контактную информацию (contact\_date).

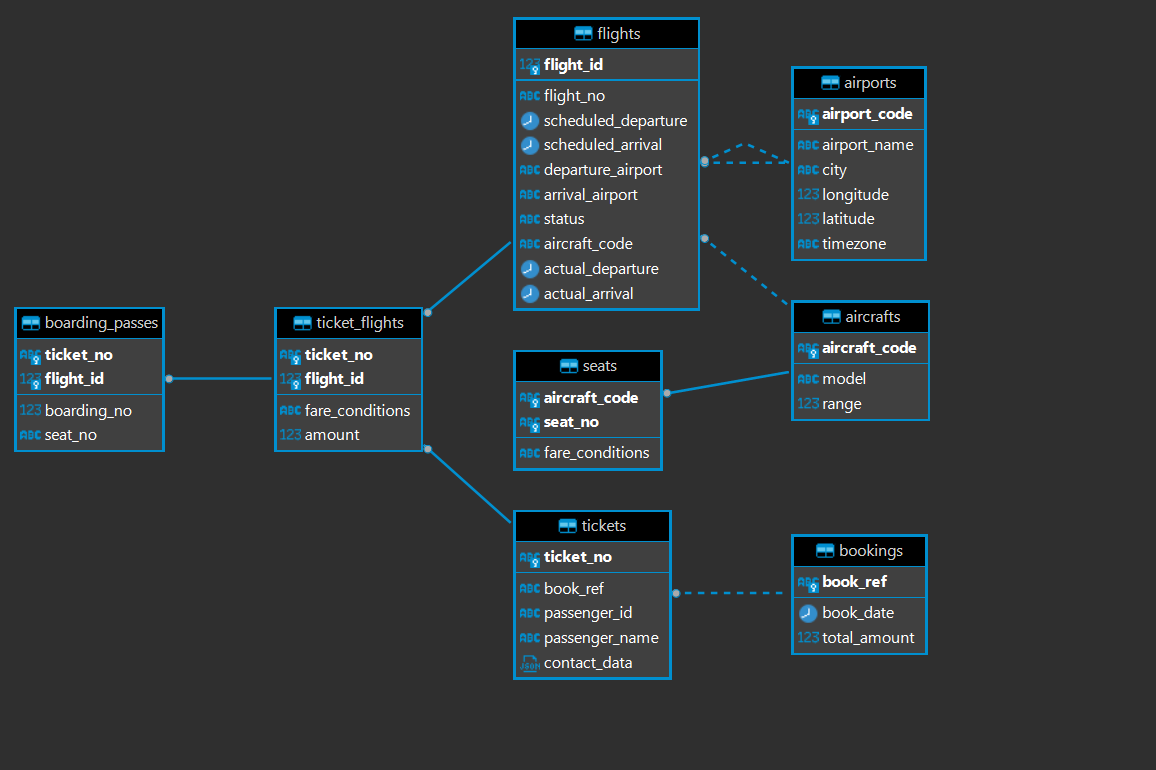


Рис. 1. Схема БД bookings.

1. **Загрузка датасета в облачное хранилище**

На этом шаге выполняется выборка данных из исходной БД bookings, сохранение выборки в промежуточном формате \*.CSV и загрузка файла в объектное хранилище S3.

В качестве хранилища объектов используется облачное S3 хранилище объектов Yandex Cloud, в качестве оркестровщика ETL-процессов – Apache Airflow.

Датасет представляет собой левое соединение таблицы flights со всеми остальными таблицами БД, при этом для экономии ресурсов в набор не добавляются поля, не имущие бизнес-ценности для итогового DWH, например ширина и долгота аэропортов, номера пассажирских мест.

В место поля с номерами мест, при формировании датасета рассчитывается количество пассажирских мест в самолете, это позволяет уменьшить количество записей в датасете и добавляет полезной информации для бизнес – целей итогового DWH. Таким образом формирование \*.CSV является не просто копированием данных из одной БД в другую, а полезной трансформацией уже на первом шаге ETL – процесса.

Сформированный CSV-файл сохраняется в S3 хранилище объектов в облаке Yandex Cloud.

За автоматизацию задач выборки данных, формирования CSV-файла, сохранения файла в хранилище отвечает Airflow DAG – «Еxport\_dag».

Описание DAGа:

«Еxport\_dag» запускается ежедневно, по расписанию. За формирование датасета отвечает пользовательский оператор "Еxport\_csv". "Еxport\_csv" подключается к БД bookings используя рython библиотеку psycopg2. Оператор создает подключение к серверу PostgreSQL и отправляет параметризованный запрос с помощью метода «copy\_expert». В качестве параметра в метод передастся строка с текстом запроса на формирование датасета, строка читается в DAG из переменной окружения с помощью метода Variable.get(). Сама строка так же содержит параметры для фильтрации выборки по дате, чтобы не выбирать данные повторно. Дата выборки вычисляется с помощью функции today() библиотеки datetime.

Результат метода «copy\_expert» перадается в файловый вывод, создается файловый поток с именем в формате текущей даты «02-24-2024.csv», имя вычисляется с помощью функции today.strftime("%m-%d-%Y").

Полученный объект передается оператору «upload\_csv». Оператор «upload\_csv» - пользовательский оператор, который получает на входе объект CSV-файл и сохраняет его в облачное хранилище. Для подключения к хранилищу используется библиотека boto3, параметры соединения передаются из переменной окружения «Variable.get('ARGS\_CONFIG', deserialize\_json=True)»

На рисунке 2 представлен граф DAGа.

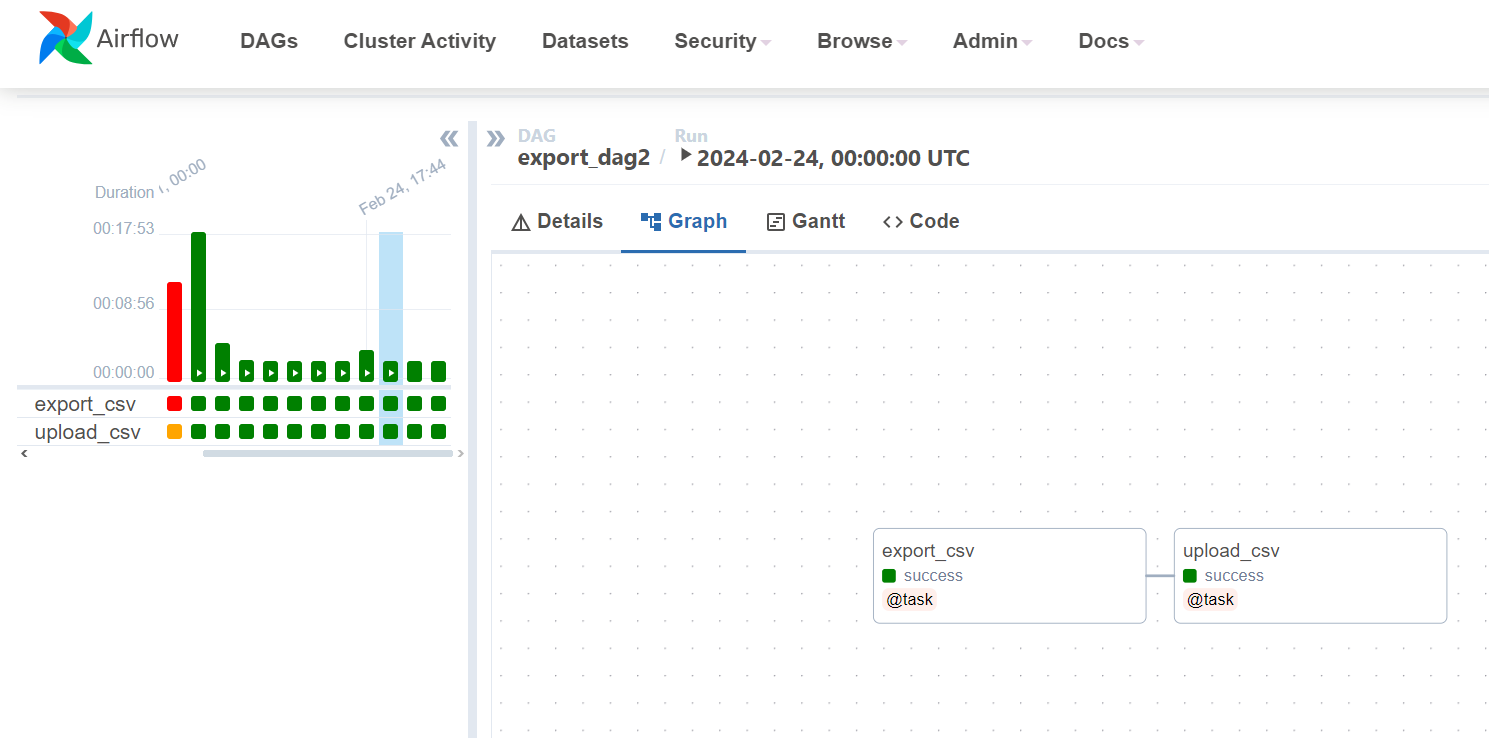


Рис. 3 «Еxport\_dag».

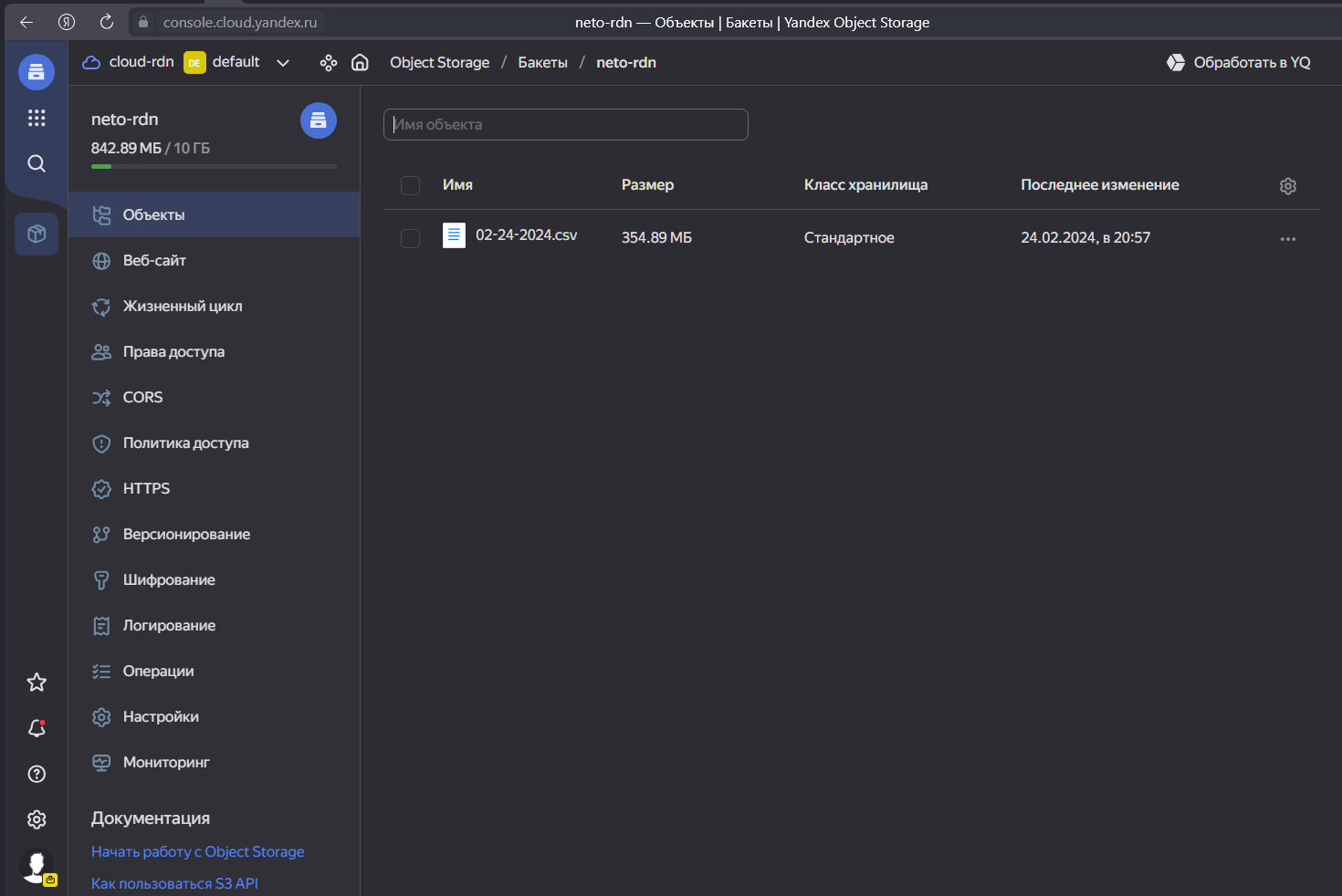


Рис. 4 объектное хранилище.

1. **Чтение объекта из хранилища**

На этом шаге выполняется чтение объекта из хранилища, и сохранение его во временную таблицу в облачном кластере PostgreSQL, за выполнение этого шага отвечает DAG "read\_bucket".

DAG запускается по расписанию ежедневно. За преобразование объекта отвечает пользовательский оператор «read\_csv». Оператор получает в качестве параметра строку в формате csv, вызывает с помощью метода cur.execute() хранимую на сервере функцию «insert\_csv\_data» которой передается строка csv.

Функция «insert\_csv\_data» выполняет в цикле считывание csv-стоки, преобразовывает ее в массив, разбивая строку на элементы по разделителю и выполняет вставку элементов массива в советующие поля таблицы Stage.

После заливки данных в таблицу вызывается пользовательский оператор "del\_csv", который удаляет объект из хранилища.

После завершения "del\_csv" управление передается Airflow оператору «TriggerDagRunOperator», который выполняет вызов DAG «ins\_temp», после чего DAG "read\_bucket" завершается.

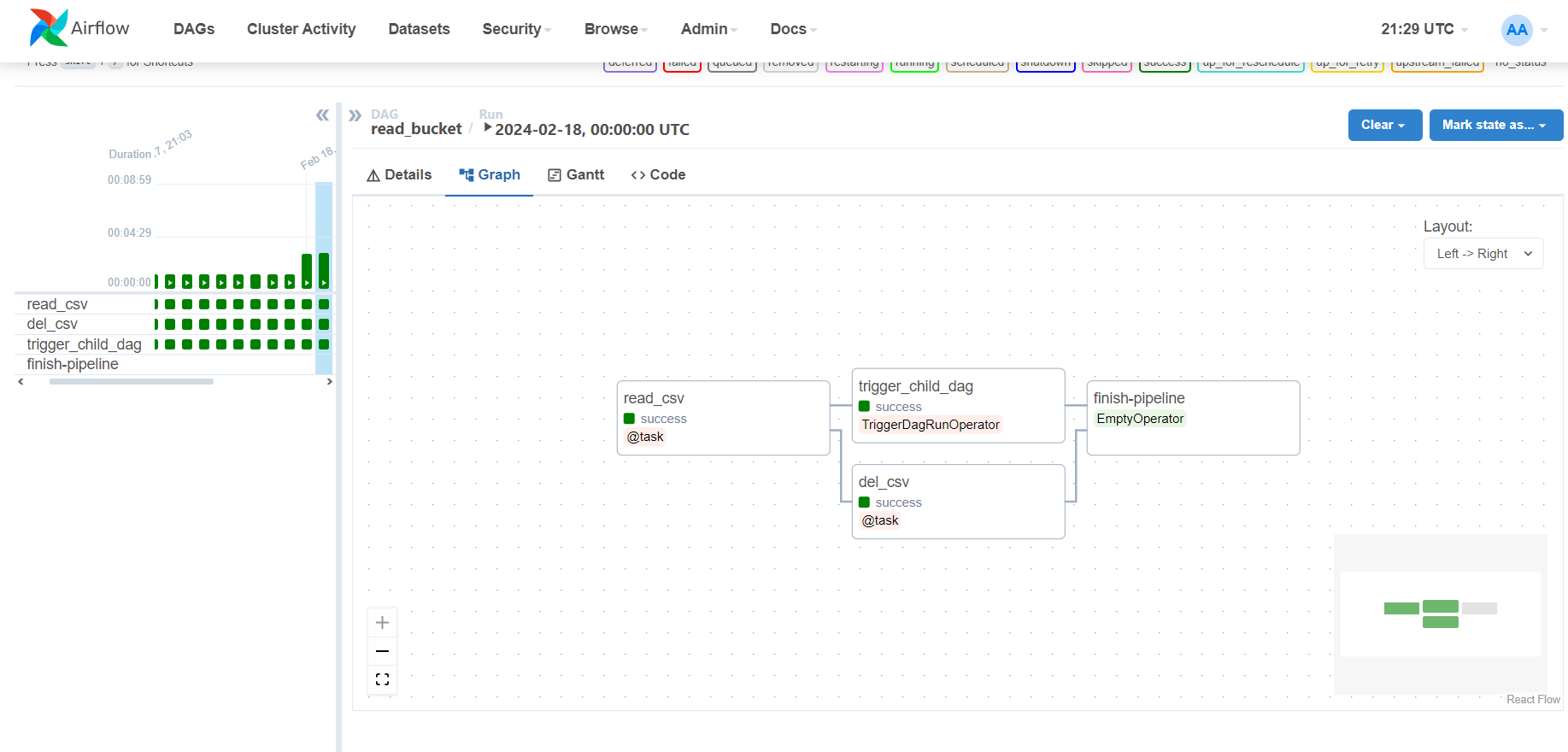


Рис. 5 DAG "read\_bucket".

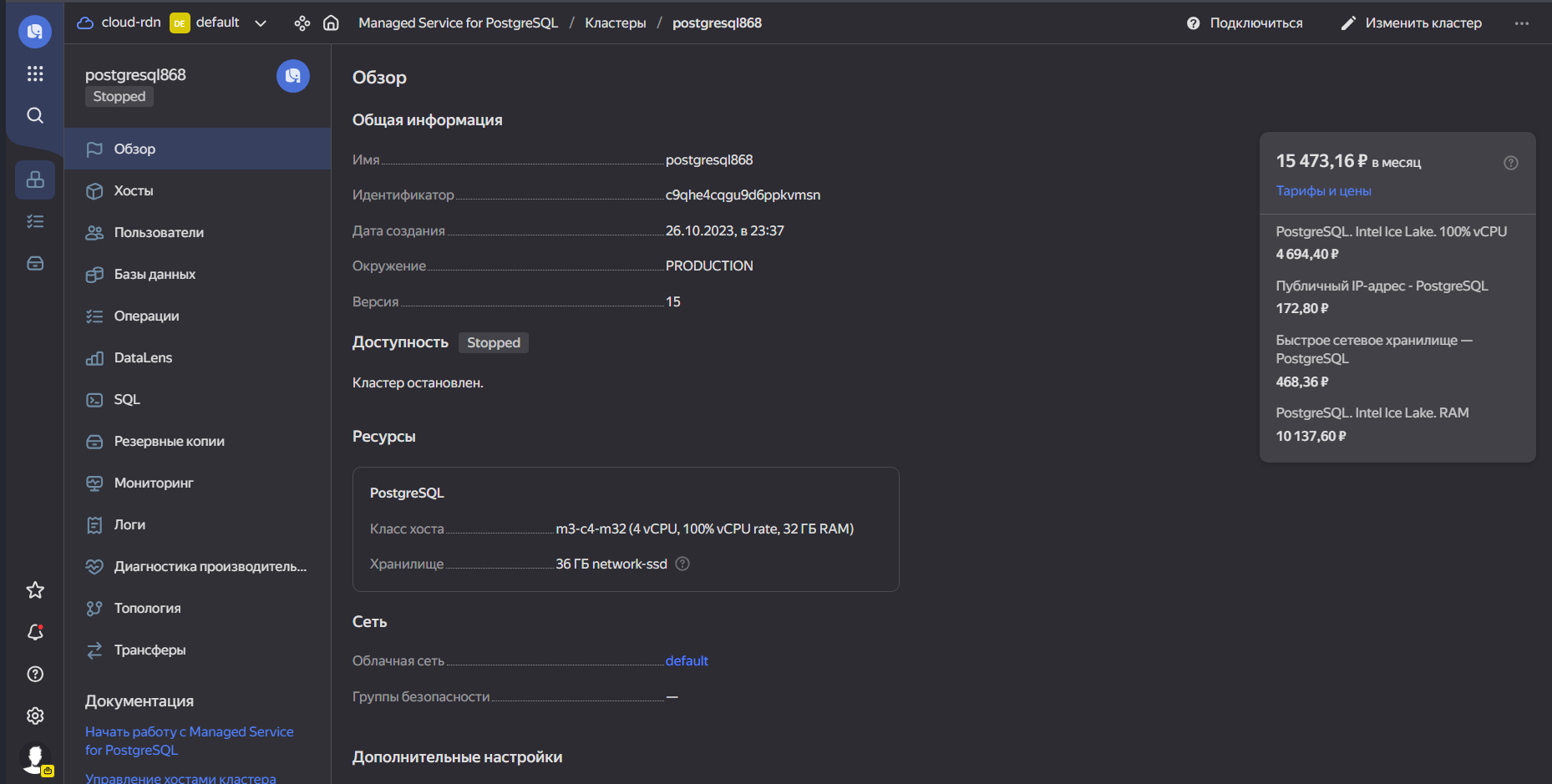


Рис. 6 облачный кластер PostgreSQL.

1. **Предварительная обработка данных**

На этом шаге выполняется выборка данных из таблицы Stage для их предварительной обработки и сохранения во временные таблицы.

Автоматизацию данного шага реализует DAG "ins\_temp". DAG запускается автоматически по завершению работы DAGа "read\_bucket" на предыдущем шаге.

За преобразование данных отвечает пользовательский оператор "ins\_temр", он получает в качестве параметров название таблицы в которую будет сохранять «очищенные» данные, перечень полей таблицы, и массив заголовков для pandas – датафрейма. Оператор подкачается к кластеру PostgreSQL, выполняет выборку столбцов из таблицы Stage с помощью функции cur.execute(), создает pandas – датафрейм, вызывает пользовательскую функцию filter\_df() и передает ей в качестве параметра созданный фрейм данных. Функция filter\_df() выполняет собственно очистку данных от дубликатов и пропущенных значений и снова преобразует датафремй в csv – строку, строка преобразуется в файловый поток без сохранения в файл.

Полученная строка передается в метод copy\_expert() в качестве параметра для сохранения во временную таблицу в БД.

После заполнения временных таблиц, таблица Stage очищается оператором «truncate\_stage» и вызов DAGа «ins\_nda» с помощью оператора «TriggerDagRunOperator».

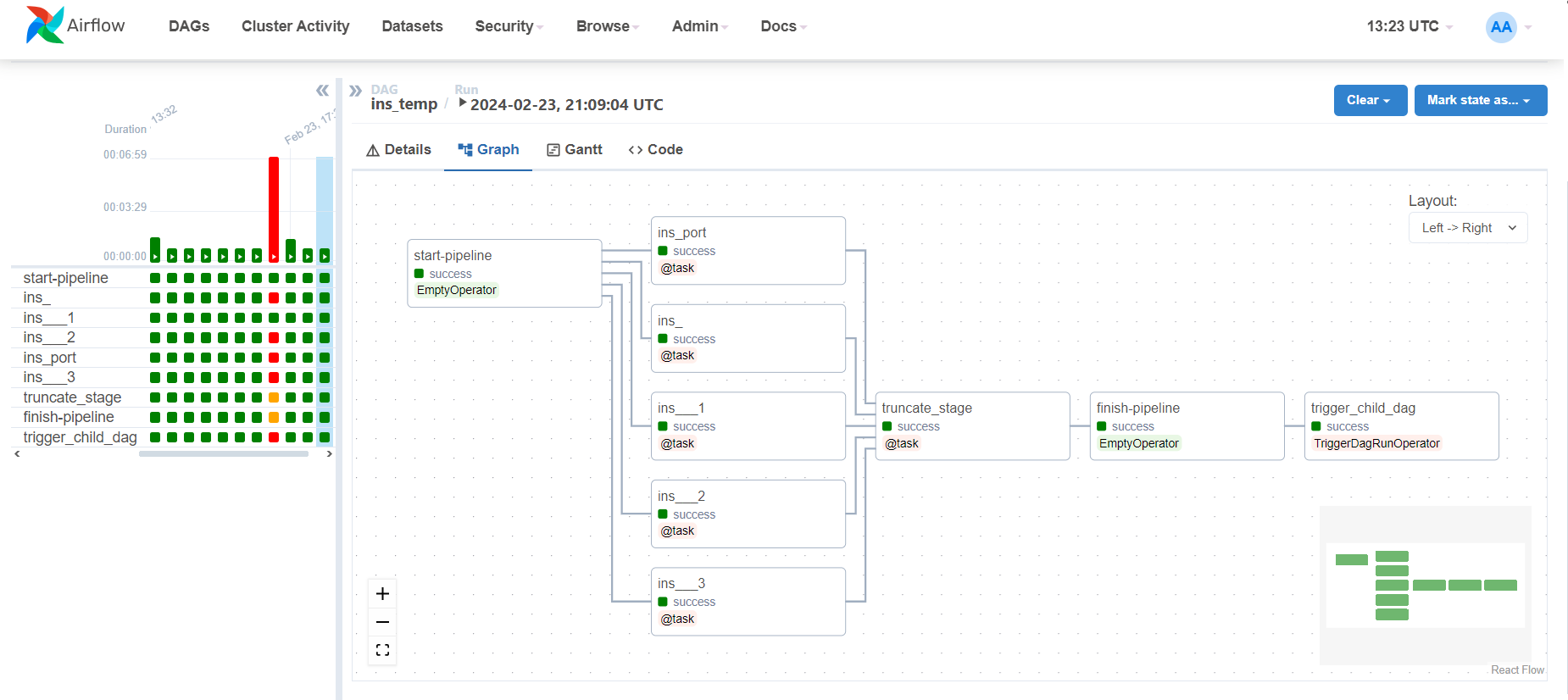


Рис. 7 DAG "ins\_temp".

1. **Заливка данных из временных таблиц в нормализованное хранилище**

После очистки данных и сохранения их во временные таблицы, выполняется заливка обработанных данных из временных таблиц в нормализованное хранилище. Нормализованное хранилище имеет структуру, оптимизированную для построения DWH хранилищ, из него удалены избыточные сущности.

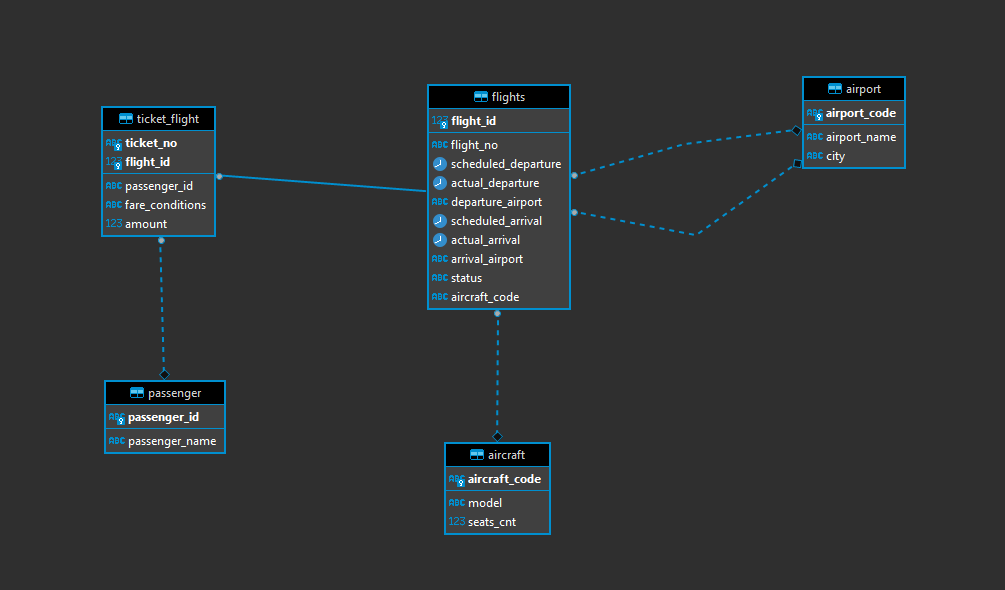


Рис. 8. Схема нормализованного хранилища.

За выполнение этого шага отвечает DAG "ins\_nda". DAG запускается автоматически по завершению работы DAGа "ins\_temp" на предыдущем шаге. Заливку выполняет пользовательский оператор "ins\_nda", который считывает данные из временной таблицы и заливает их в нормализованную БД в таблицу с таким же именем. При заливке выполняется проверка на уникальность ключей и полноту данных в связанных таблицах, дублирующиеся записи игнорируются, таким образом предотвращается повторная заливка сданных, носящих справочный характер.

После выполнения DAGа временные таблицы очищаются оператором "truncate\_temp ()", вызывается DAG "ins\_dds".

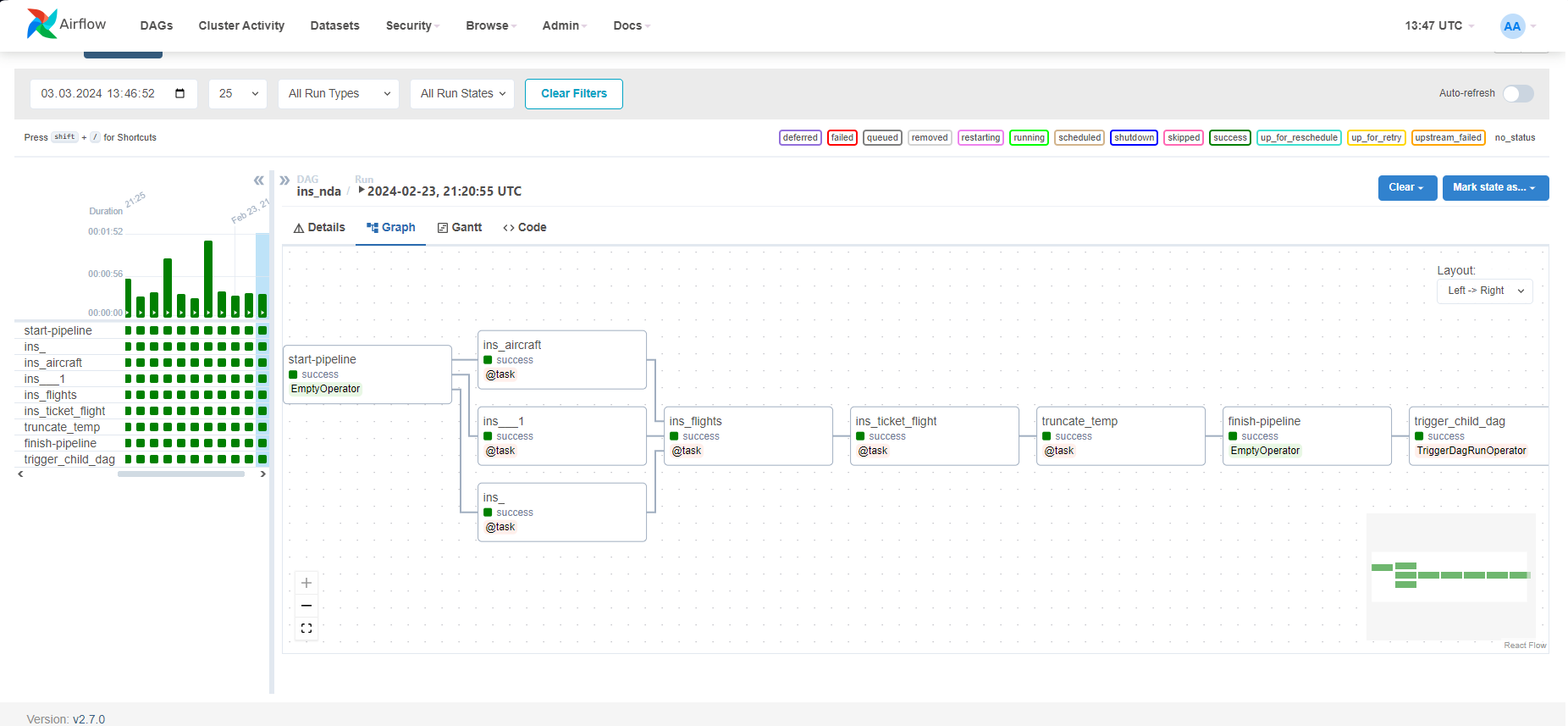


Рис.9 DAG "ins\_nda".

1. **Заливка данных DDS-хранилище**

Данный шаг является заключительным в трансформации, на этом шаге выпорется выборка и трансформация данных из нормализованного хранилища для их заливки в DDS-хранилище (схема звезда).

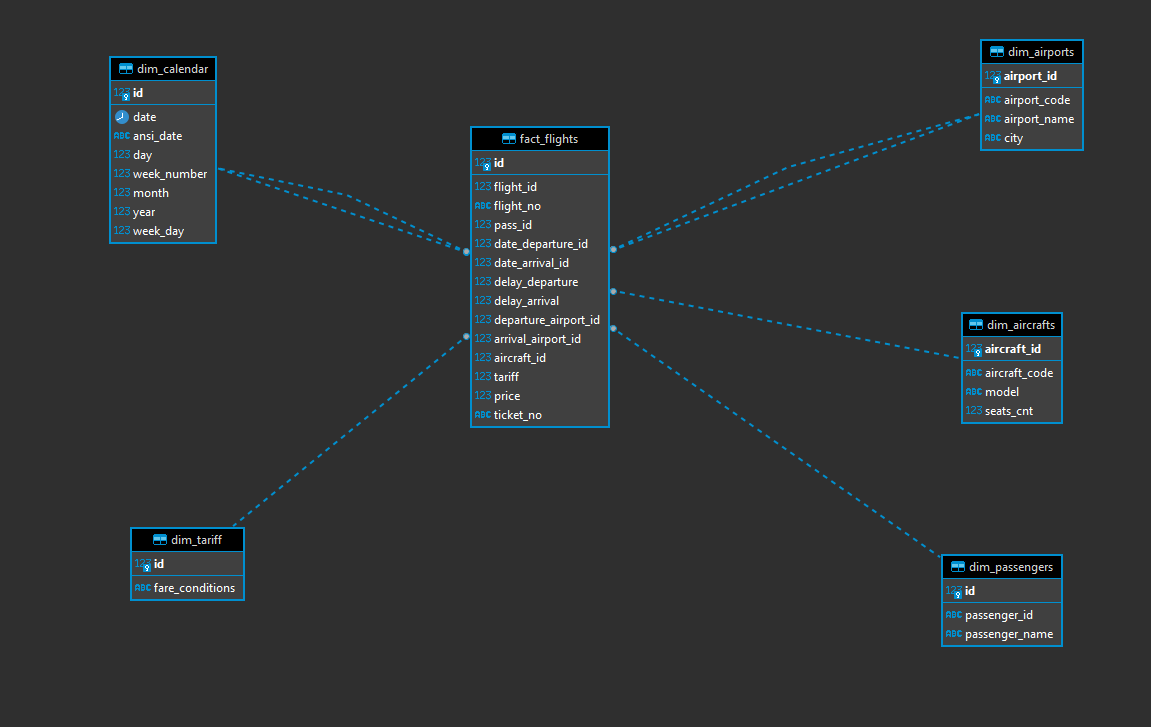


Рис. 10 Схема DDS- хранилища.

Данный шаг реализован в DAG “ins\_dds”. DAG запускается автоматически по завершению работы DAGа "ins\_nda" на предыдущем шаге. DAG выполняет пользовательские операторы, которые выполняют выборку данных в nda хранилище и заливают их в dds хранилище. Сначала выполняется заливка справочников, при этом контролируется уникальность данных, если в выборке появляются повторяющиеся значения то они игнорируются для загрузки, затем выполняется загрузка данных в таблицу фактов. Схема DAGа приведена на рисунке 9.

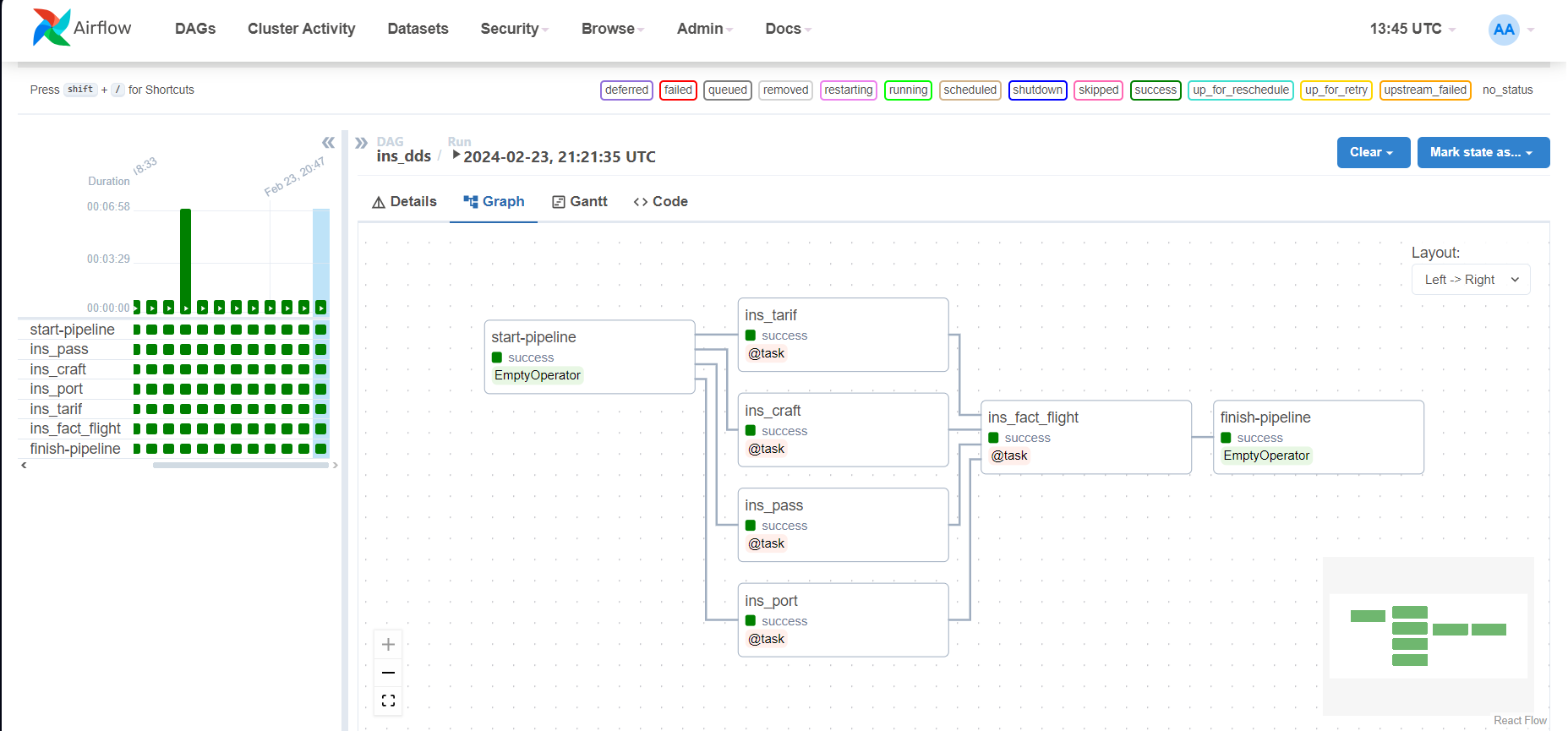


Рис. 11 DAG "ins\_dds".

1. **Визуализация**

Для построения и визуализации аналитики используется облачная BI система от Яндекс – DataLens. Источником для построения аналитики служит хранилище DDS, для подключения к источнику используется стандартный сервис.

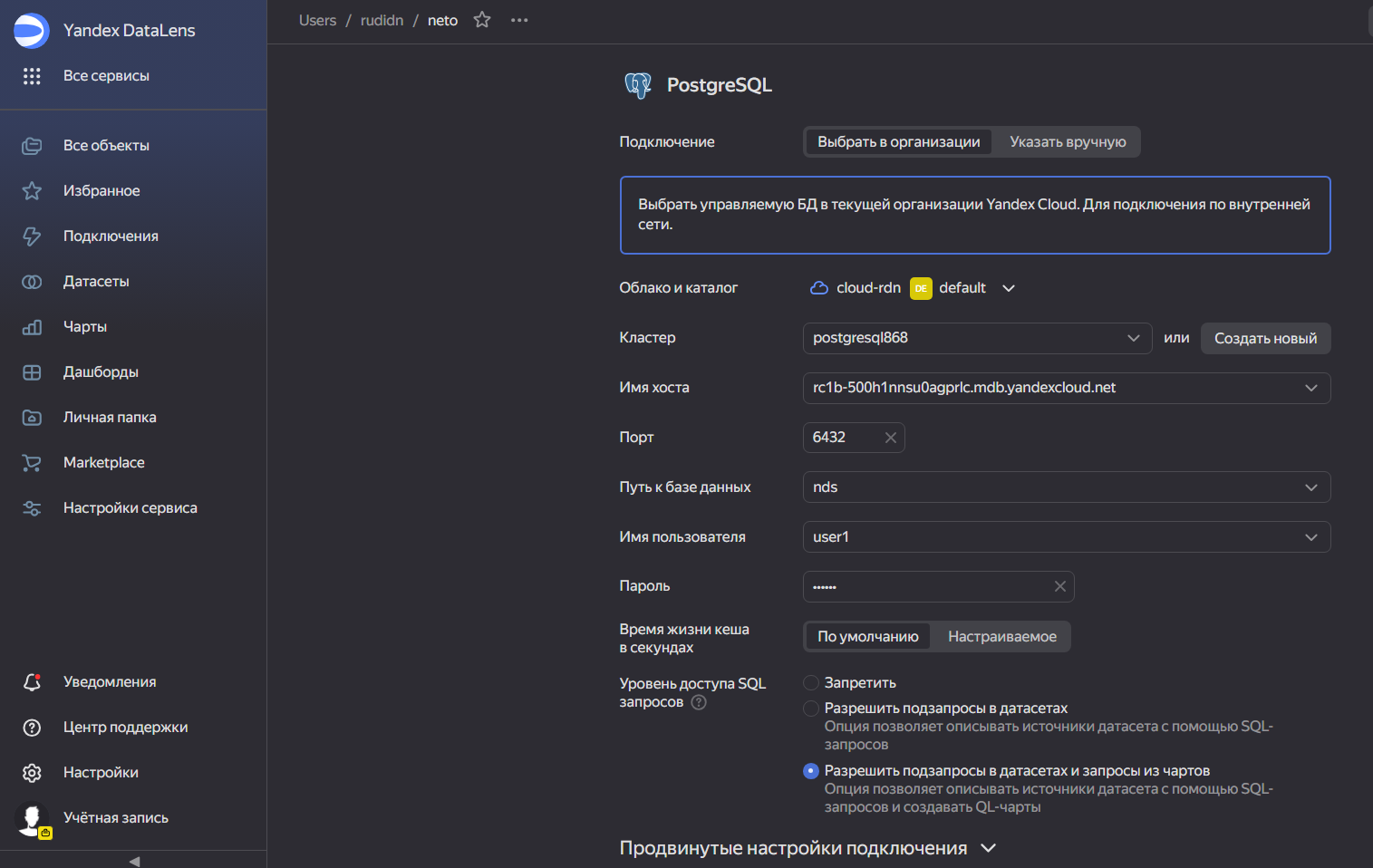


Рис. 12 соединение с БД.

Для построения метрик создается датасет.

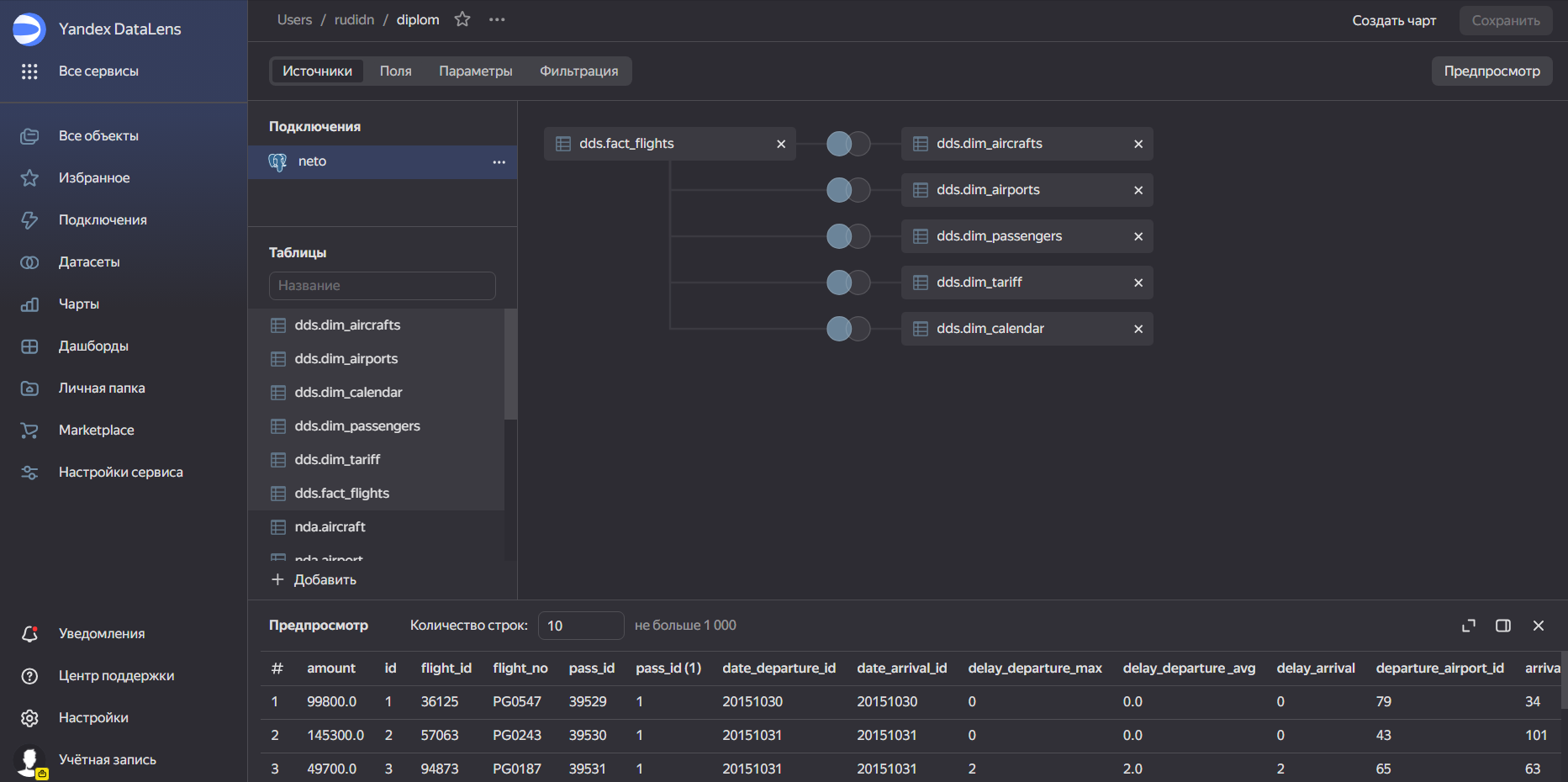


Рис. 13 Дататсет.

После создание датасета настраиваются поля – указывается измерения и меры, добавляются вычисляемые поля.

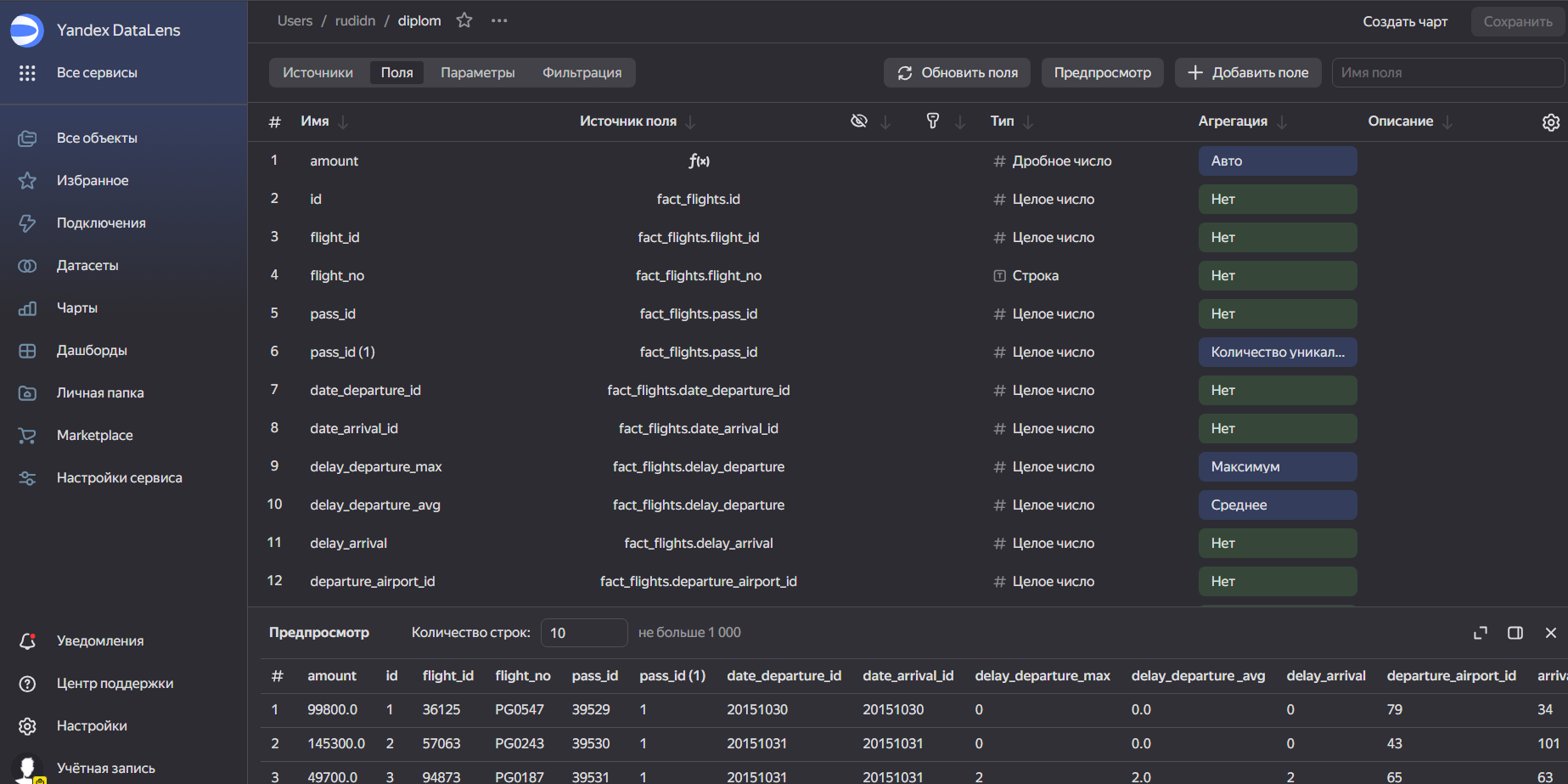
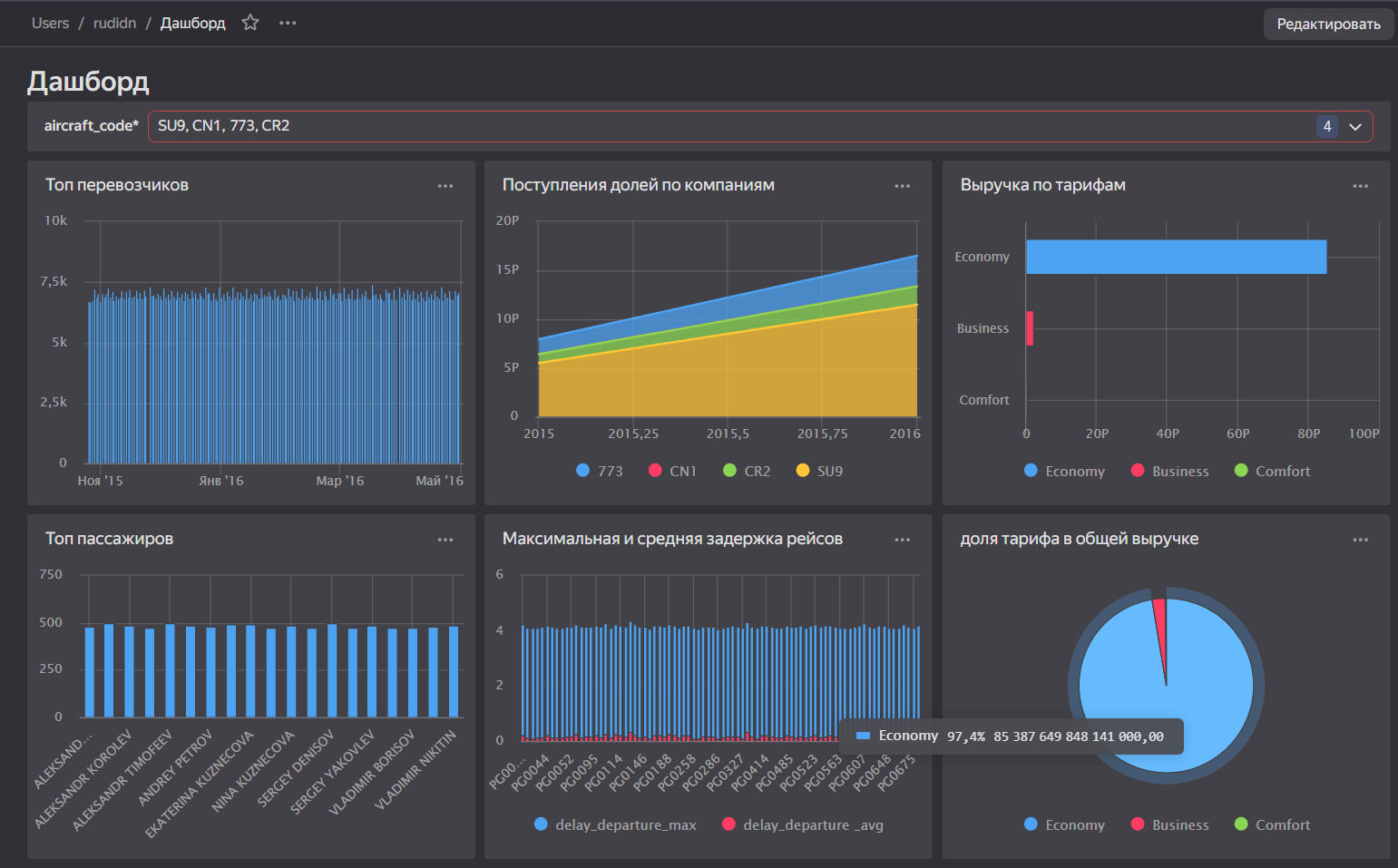


Рис. 14 Настройка полей.

На настроенном датасете создаются графики для визуализации метрик. В качестве метрик выбраны:

1. Топ перевозчиков – лидеры перевозок по месяцам
2. Топ пассажиров – наиболее част летающие пассажиры
3. Величина максимальная и средней задержки рейсов
4. Доля выручки по тарифам от общей выручки



**Приложение №1 к дипломной работе.**

Таблица №1 Перечень прилагаемых файлов:

|  |  |
| --- | --- |
| **Файл** | **Описание** |
| CSV.sql | Скрипт для создания CSV файла |
| insert\_csv\_data.sql | Хранимая функция для заполнения таблицы “Stage” |
| Creat\_temp.sql | Скрипт создания временных таблиц |
| Creat\_nda.sql | Скрипт создания хранилища nda |
| Creat\_dds.sql | Скрипт создания хранилища dds |
| ins\_fact.sql | Запрос для заполнения таблицы фактов |
| export\_dag3.py | Скрипт Airflow DAG отвечающий за загрузку CSV выборки в облачное хранилище |
| ins\_stage\_dag.py | Скрипт Airflow DAG отвечающий за чтение файла из хранилища и заливки его во временную таблицу |
| ins\_temporary\_dag.py | Скрипт Airflow DAG выполняет предварительную обработку (очистку) данных и заливку их во временные таблицы |
| ins\_nda\_dag.py | Скрипт Airflow DAG залвающий данные из временных таблиц в хранилище nda |
| ins\_dds\_dag.py | Скрипт Airflow DAG залвающий данные из хранилищa nda в dds хранилище |
| https://github.com/ruddn784/DOps.git | Ссылка на хранилище содержащее проект |