Урок 1

Базы данных (БД) и СУБД

С использованием данных мы сталкиваемся каждый день: при заказе еды, в библиотеке, при использовании банковского приложения. В сфере IT с базами данных прежде всего приходится работать дата-инженерам, аналитикам, дата-сайентистам, бэкэнд-разработчикам, а иногда даже менеджерам.

В курсе мы рассматриваем работу с реляционными базами данных. Слово «*реляционный*» происходит от английского "*relation*" и означает отношение, логическую связь между таблицами, которые друг с другом связаны.

В основе любых реляционных баз данных лежат таблицы. Каждая таблица состоит из полей (столбцов) и записей (строк), на пересечении которых находятся значения (значение поля для записи). Применительно к полям существует два понятия:

1. **Первичный ключ (primary key).** Он нужен для того, чтобы уникально идентифицировать какую-либо строку в таблице. Для всех записей в одной таблице первичным ключом является одно или несколько полей данной таблицы.

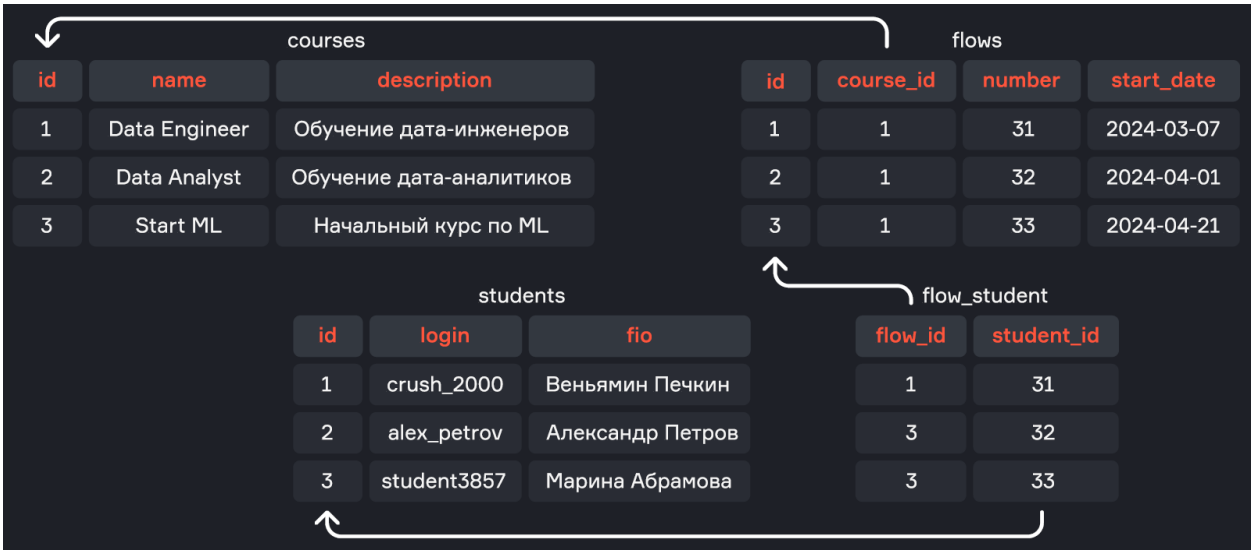
   Первичным ключом могут быть:

   - реальные уникальные идентификаторы пользователя в рамках одной таблицы. Например, номер паспорта, номер заказа, email, СНИЛС;

- синтетические значения идентификаторов. Такие значения не имеют смысла в физическом мире и используются только для различения строк в таблице.

2. **Внешний ключ (foreign key).** Это ссылка из одной таблицы на первичный ключ другой таблицы. Например, у нас есть ID заказа в одной таблице, а в другой — набор заказанных товаров. У каждого из заказанных товаров будет стоять ID заказа, к которому он относится.

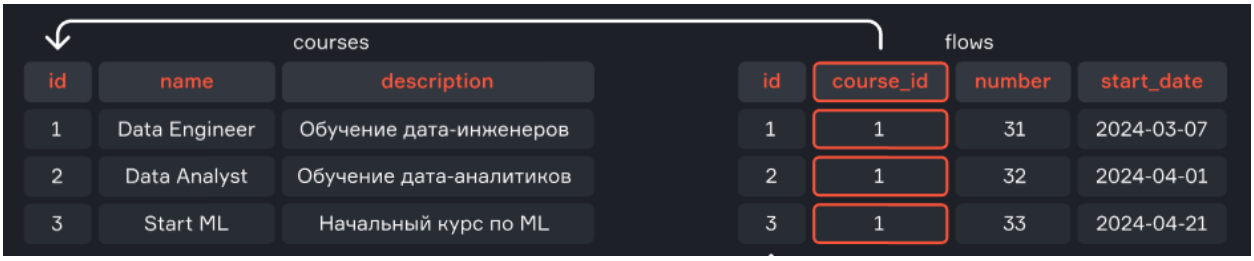
ПРИМЕР



Рассмотрим набор таблиц, содержащих информацию о студентах и курсах, на которых они обучаются:

- таблица `courses` имеет первичный ключ — уникальный синтетический `id`;
- таблица `flows` (потоки обучения) имеет первичный ключ — свой `id`, не зависящий от идентификаторов других таблиц. А также внешний ключ `course_id`, который ссылается на таблицу `courses`. В примере на схеме все потоки имеют `course_id` = 1 и относятся к курсу Data Engineer из таблицы `courses`;
- таблица `students` имеет первичный ключ — `id`;
- таблица `flow_student` имеет два внешних ключа — `flow_id` и `student_id`.

Выше мы рассматривали связь **один ко многим**. Одному первичному ключу (например ID) может соответствовать несколько внешних ключей в соседней таблице. Один ID-шник — много ссылок на него, поэтому связь называется «один ко многим».



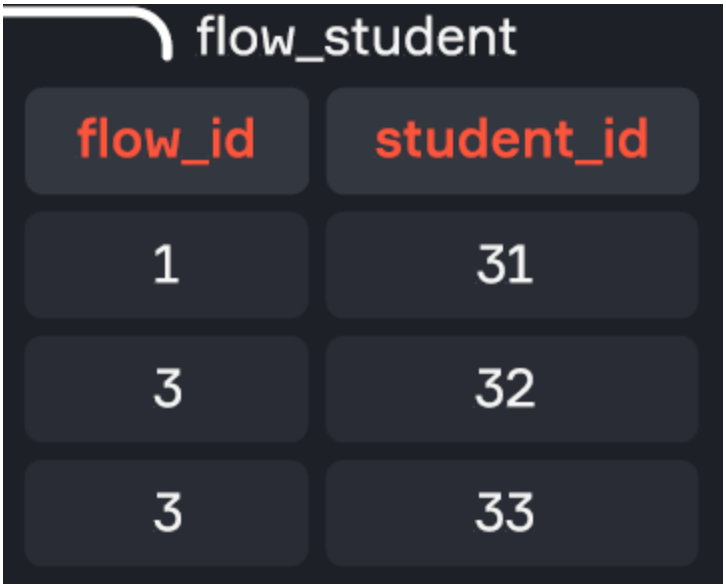
courses

id	name	description
1	Data Engineer	Обучение дата-инженеров
2	Data Analyst	Обучение дата-аналитиков
3	Start ML	Начальный курс по ML

flows

id	course_id	number	start_date
1	1	31	2024-03-07
2	1	32	2024-04-01
3	1	33	2024-04-21

Также у нас в базе студентов есть и связь, которая называется **многие ко многим**. Это поля с потоками и студентами, ведь каждый из этих студентов может учиться на нескольких потоках, а в каждый поток может зачисляться несколько студентов. Такая связь реализуется с помощью дополнительной вспомогательной таблицы `flow_student`. Это таблица, в которой один внешний ключ — это ссылка на таблицу Flows, а второй внешний ключ — это ссылка на таблицу `students`.



flow_student

flow_id	student_id
1	31
3	32
3	33

Работой базы данных управляет система управления базами данных (СУБД). Это программное обеспечение, которое берет на себя ответственность за консистентность, оптимизацию, репликацию, блокировки, доступ и прочие

функции. Систем управления базами данных очень много, и их работа немного отличается. Наиболее распространенными СУБД являются Postgres, MySQL, Greenplum, Vertica, Oracle, MS SQL, Clickhouse, Redis, MongoDB. С некоторыми из них мы познакомимся в следующих модулях курса.

Язык управления запросами SQL

Мы можем получать данные из базы данных с помощью запросов к СУБД, написанных на языке управления запросами SQL (Structured Query Language). Это *декларативный унифицированный* язык. Определение «декларативный» значит, что в запросе мы описываем результат выполнения запроса, а не логику извлечения данных системой. А благодаря унифицированности языка, его основные конструкции можно использовать для работы с разными базами данных. Тем не менее, стоит помнить, что для разных СУБД язык может немного отличаться синтаксически или за счет введения дополнительных конструкций.

Итак, мы отправляем в СУБД SQL-запрос, система управления оптимизирует этот запрос, смотрит, насколько он правильный, есть ли у нас доступ к этим таблицам, есть ли вообще такие таблицы и поля, а затем обращается к базе данных за нужными строками, которые мы хотим получить. Это строки с нужной нам информацией возвращаются, например, в интерфейсе приложения, или в командной строке, или в дашборде.

SQL-запрос представляет собой конструкцию из операторов, которые также называют инструкциями или ключевыми словами. Благодаря им СУБД

«понимает», какие действия с данными ей необходимо произвести.

Рассмотрим простой запрос `SELECT * FROM courses;`

- SELECT ... FROM ... — основная конструкция в SQL, которая даёт команду базе данных ВЫБЕРИ <колонки> ИЗ <таблица>
- * используется после SELECT и сообщает, что нужно выбрать все столбцы из указанной после FROM таблицы

Таким образом, в результате запроса мы получим все строки из таблицы `courses`.

id	name	description
1	Data Engineer	Обучение дата-инженеров
2	Data Analyst	Обучение дата-аналитиков
3	Start ML	Начальный курс по ML

Запрос `SELECT COUNT(id) FROM students;` считает количество строк в таблице `students`.

Рассмотренные выше запросы относятся к категории DQL (Data Query Language), которая только забирает данные из таблицы. Сам SQL — это такое общее название для семейства подъязыков, отличающихся операторами и назначением запросов. Полный перечень подъязыков, на которые делят SQL следующий:

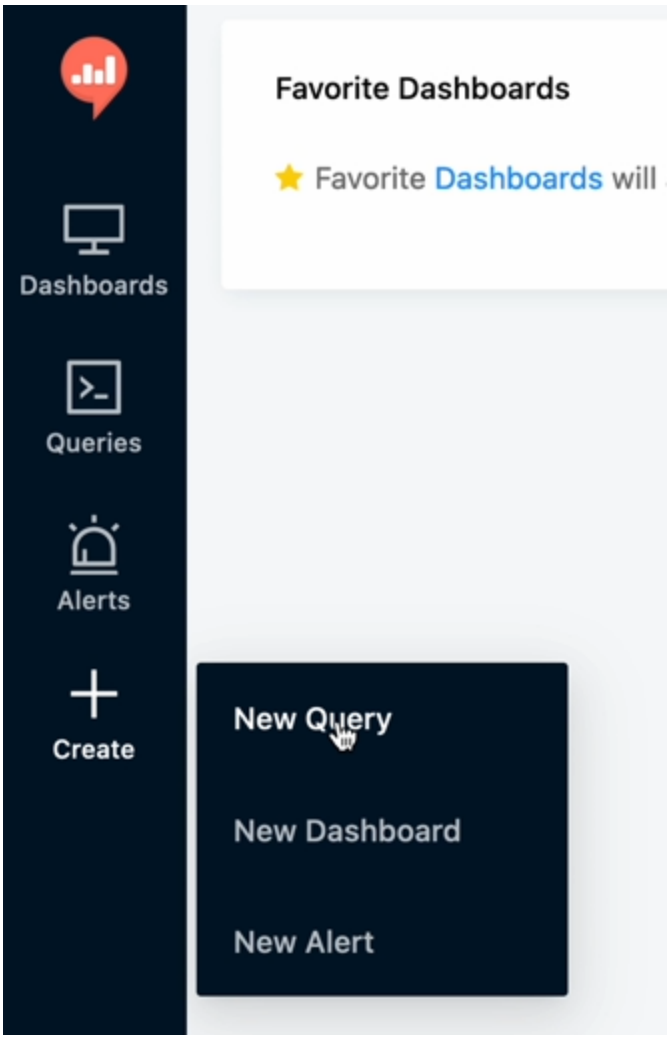
- **DQL (Data Query Language)** — язык, который используется для выполнения запросов к данным, поиска и извлечения данных из базы;
- **DML (Data Manipulation Language)** — язык, с помощью которого добавляются, изменяются или удаляются строки;
- **DDL (Data Definition Language)** — язык, с помощью которого создаются или изменяются какие-то сущности, например, таблицы;

- **DCL (Data Control Language)** — язык, который позволяет управлять доступами к данным;
- **TCL (Transaction Control Language)** — язык управления транзакциями, который используется для контроля обработки транзакций в базе данных.

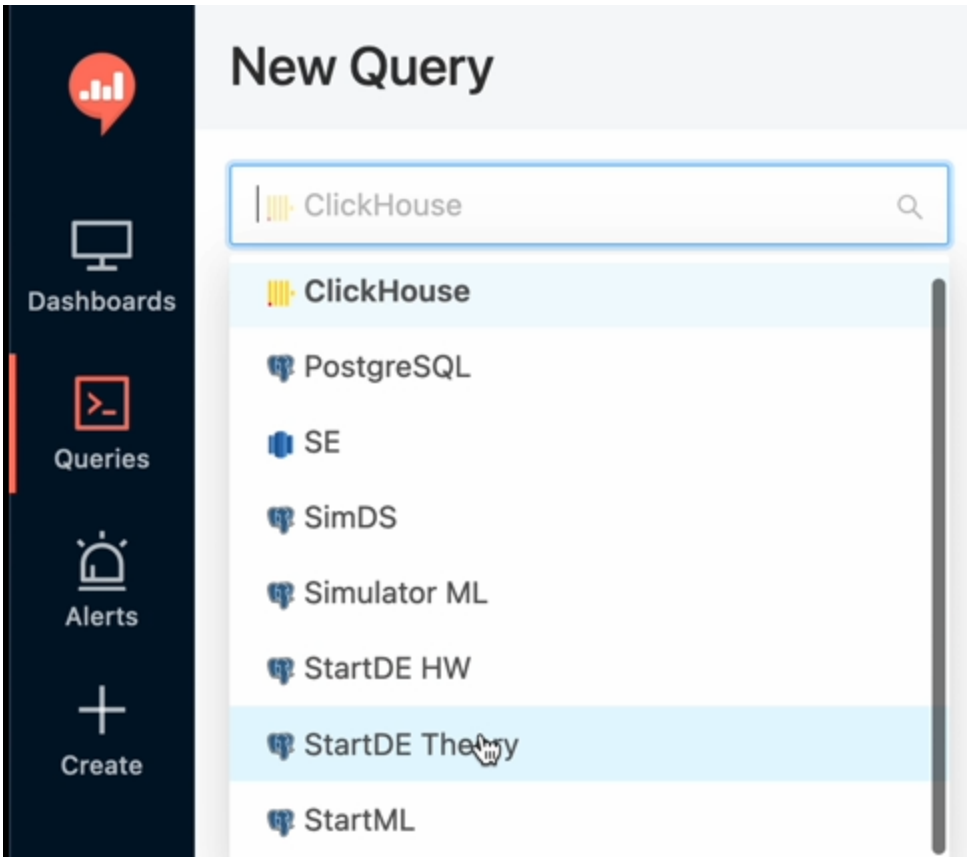
Работа в Redash

Ниже представлена краткая инструкция работы с инструментом Redash, который является сервисом для работы с базами данных и предоставляет возможность не только написать запросы, но и визуализировать результат в дашбордах.

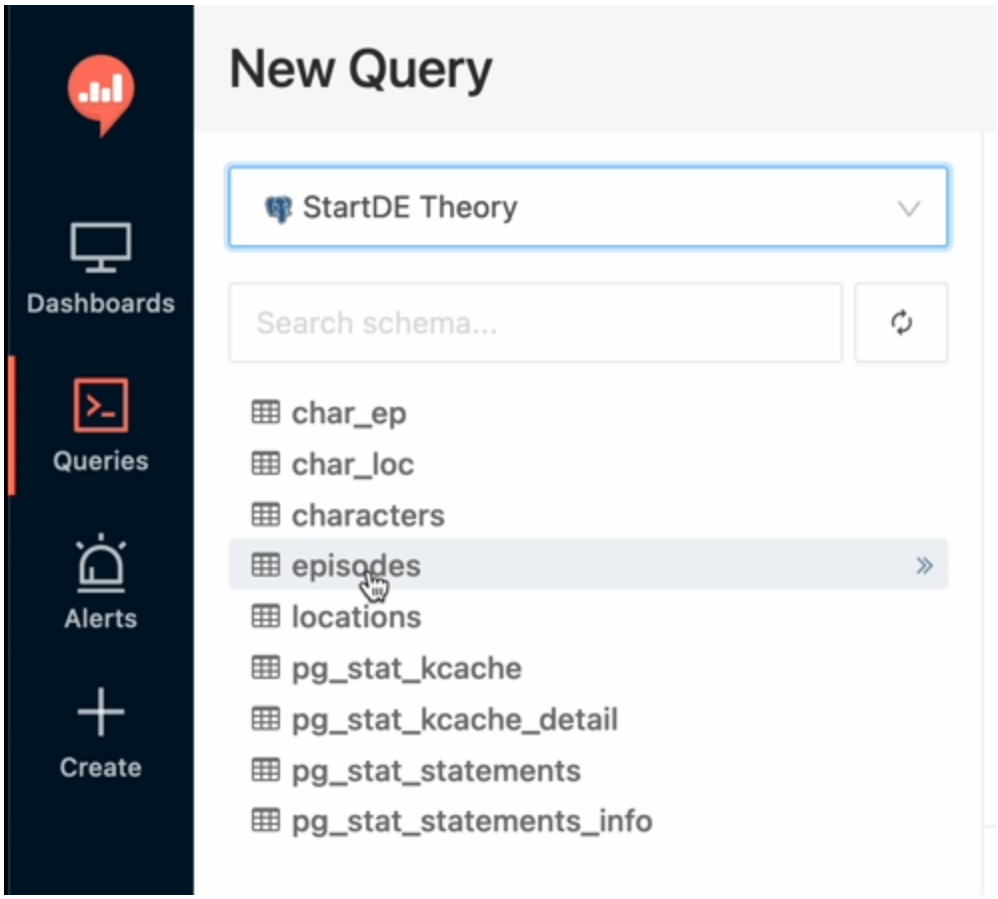
1. Для того чтобы начать писать запрос, необходимо нажать на кнопку `Create` и выбрать пункт New Query.



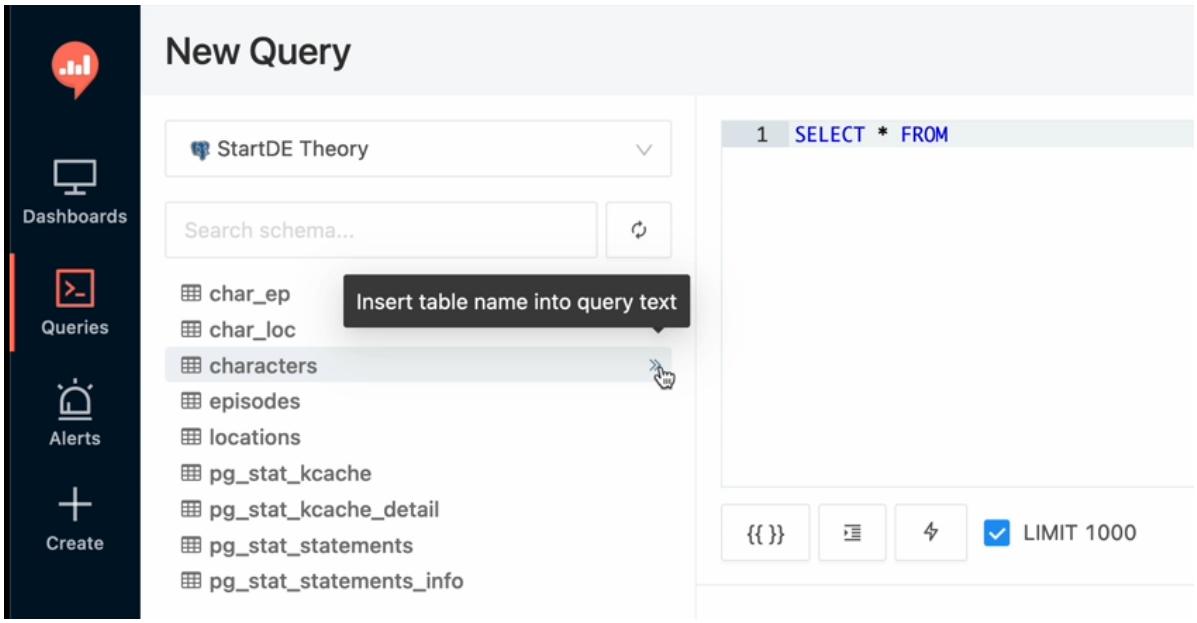
2. Появится набор баз данных, к которым у нас есть доступ. Нас интересуют база данных `StartDE Theory`.



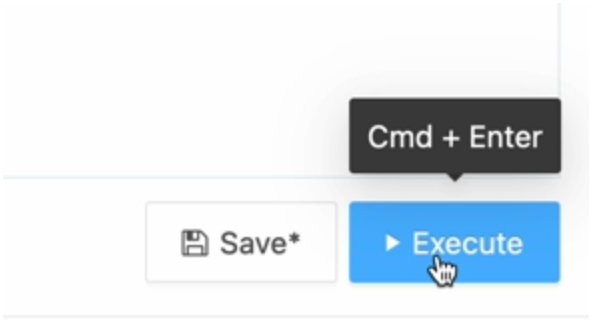
3. Мы видим несколько таблиц, в том числе системных.



4. Если нажать на правую стрелочку около таблицы, мы сможем получить ее название сразу в нашем `SELECT`.

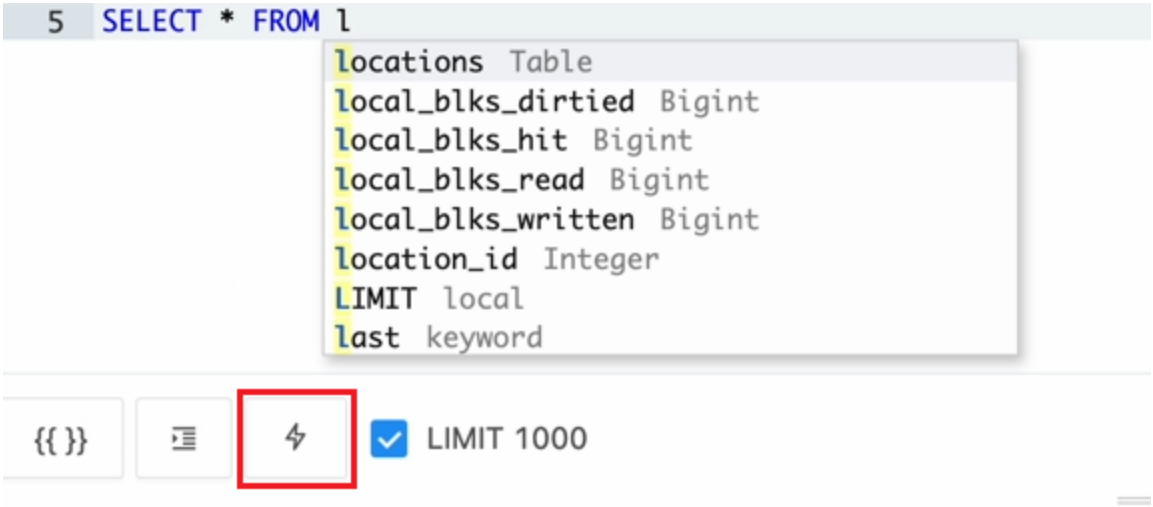


5. Внизу справа у нас есть кнопка `Execute`. С ее помощью запрос отправляется в СУБД.

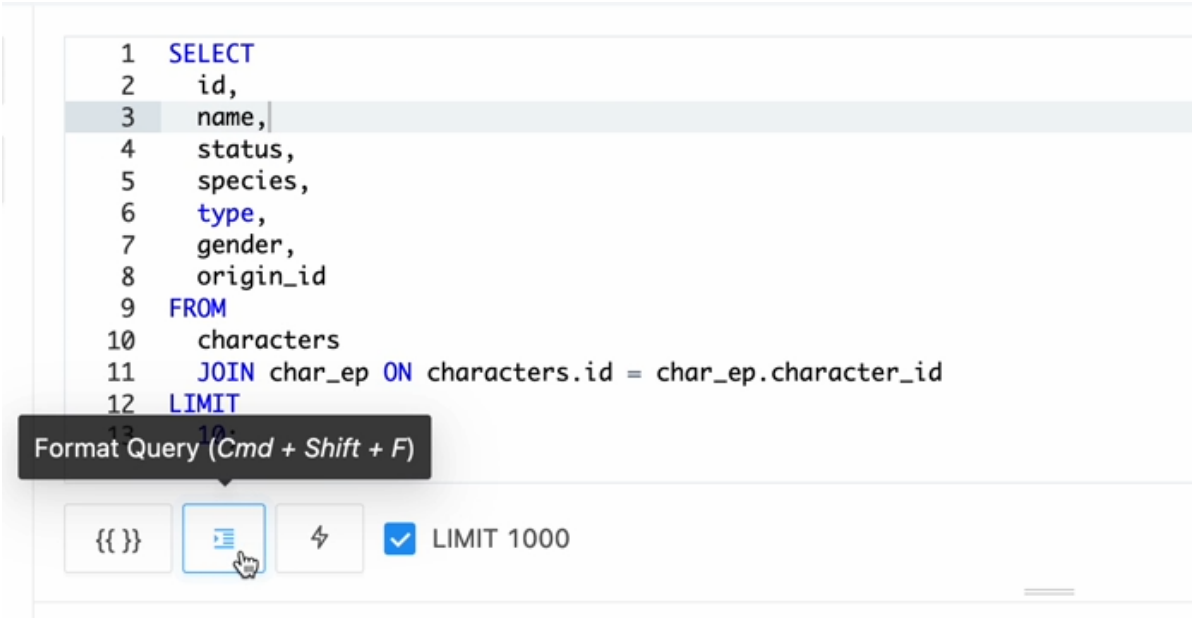


Некоторые фичи в Redash:

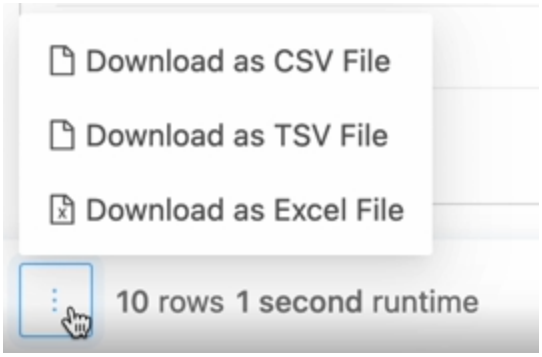
- **Автокомплит** (автозаполнение). Если мы пишем SELECT звездочка FROM l, видим, что у нас вылезает подсказка, из которой можно выбрать нужное нам значение. Автокомплит можно отключить.



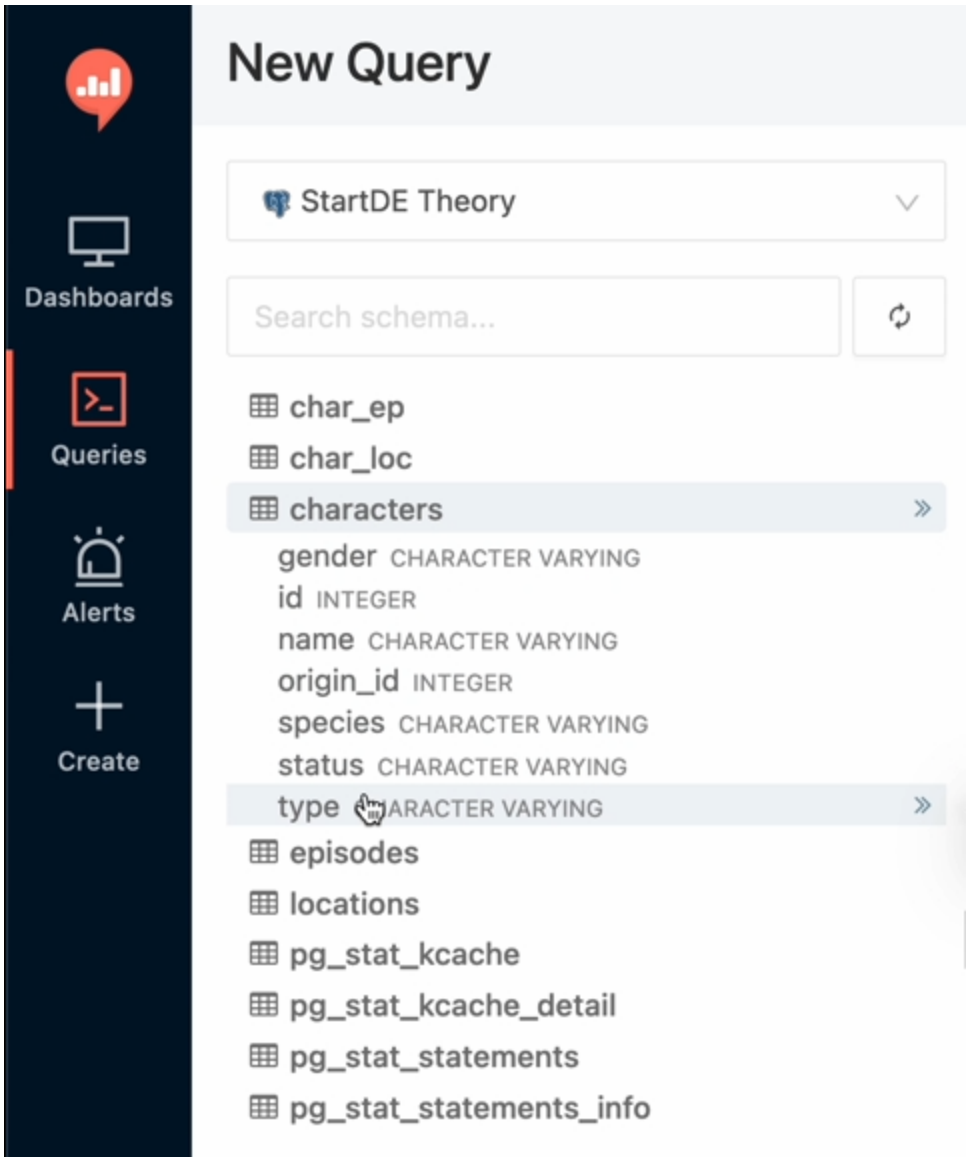
- **Форматирование.** Актуально для больших запросов на много строк. Когда необходимо привести все в порядок форматирование, мы нажимаем на соответствующую кнопку и получаем более аккуратный код, которым уже не стыдно поделиться со своими коллегами.



- Возможность **сохранить файл** в нужном формате для дальнейшей обработки с помощью других инструментов.

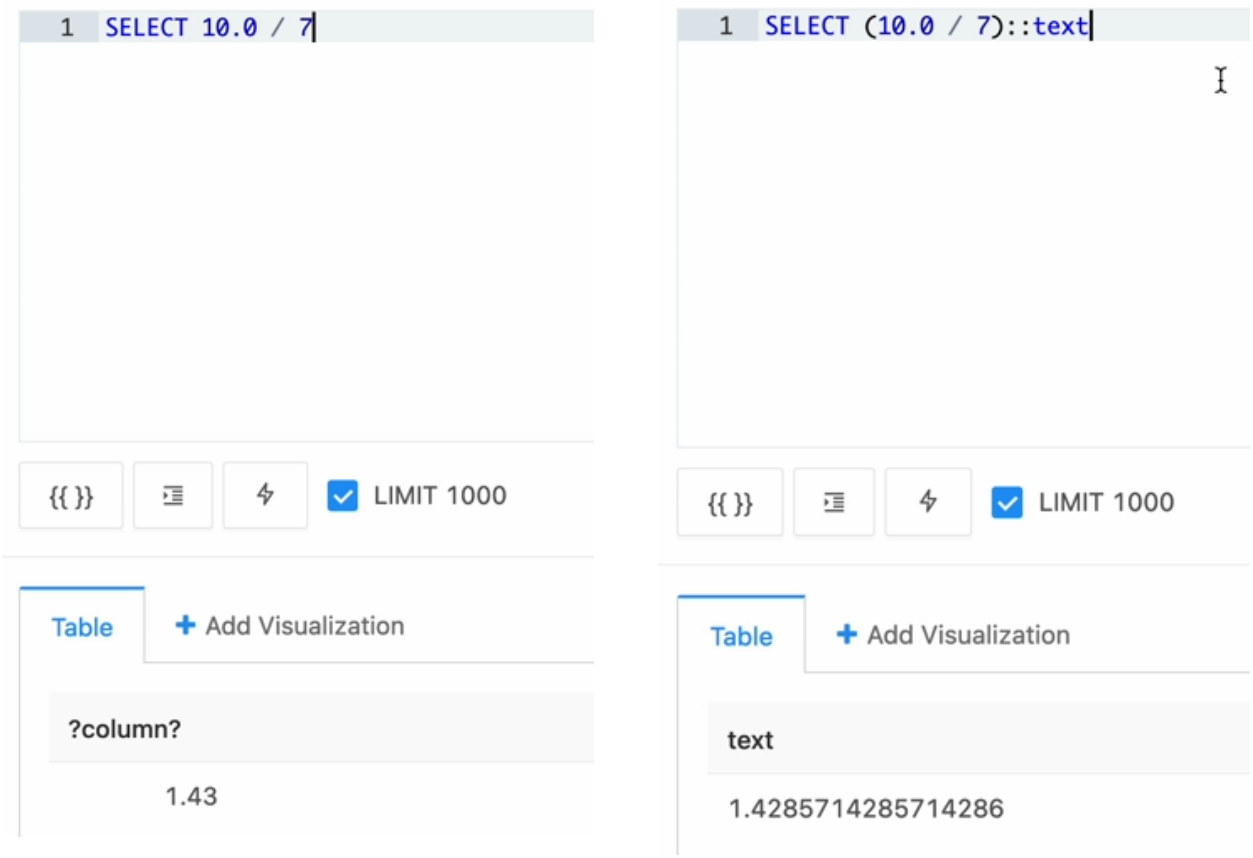


- **Просмотр** таблиц и наборов их полей.



Нюансы отображения в Redash:

- Redash «обрубает» до двух чисел после запятой при отображении результата вычисления. Можно убедиться в этом, преобразовав результат в текстовый тип.



- Redash видоизменяет дату, которая хранится в PostgreSQL. На самом деле в PostgreSQL, как и в большинстве баз данных, даты хранятся в следующем формате: сначала год, потом месяц, потом дата.

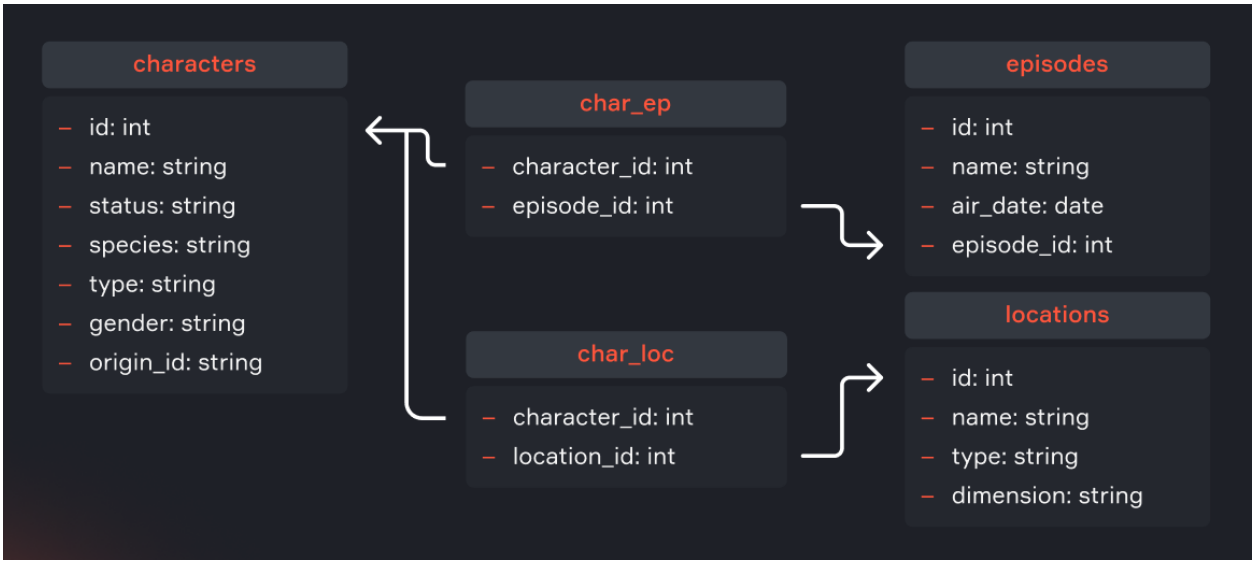
```sql
SELECT *
  FROM episodes
 WHERE air_date = '2013-12-02';
 -- '02/12/13';
```

{{ }} LIMIT 1000

Table + Add Visualization

id	name	air_date	episode_id
1	Pilot	02/12/13	S01E01

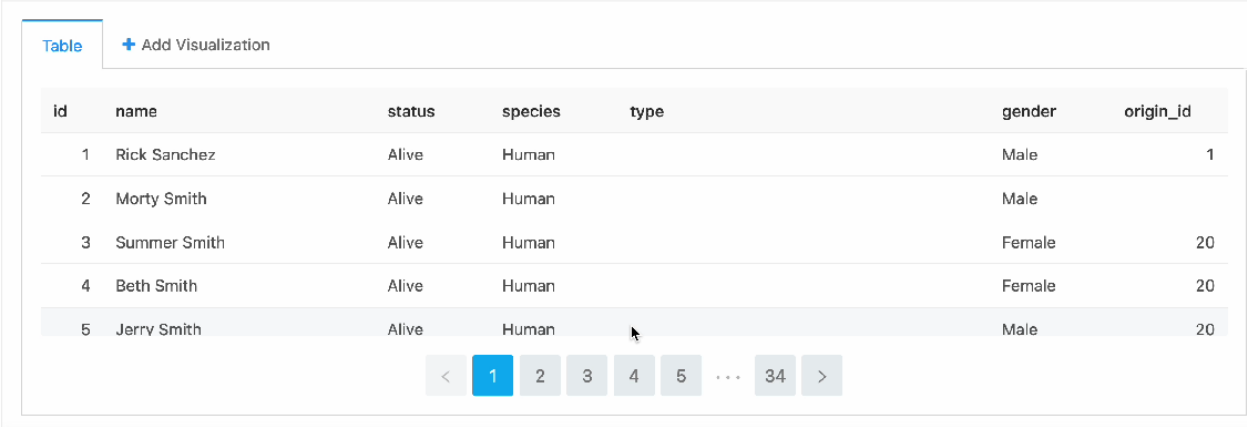
ПРИМЕР



Это схема по мультсериалу «Рик и Морти». Она включает в себя следующие таблицы:

- персонажи (таблица `characters`);
- эпизоды (таблица `episodes`);
- локации, в которых происходит действие (таблица `locations`);
- таблица, показывающая, в каких эпизодах и какие были персонажи (связывающая таблица `char_ep`);
- таблица, показывающая, в каких локациях и какие были персонажи (связывающая таблица `char_loc`).

Выполним запрос `SELECT * FROM characters;` и проанализируем результат.



Мы написали максимально неоптимальный запрос для любой СУБД по следующим причинам:

1. Вывели все поля (столбцы)

   Мы выбрали звездочку, то есть все поля, которые есть у нас в таблице именно в том виде, в котором они хранятся в базе данных. Однако очень часто необходимо выгрузить эти данные с помощью запроса и отнести в какое-то большое хранилище данных. И здесь уже очень важно, какой именно набор полей мы получаем, в каком именно порядке они располагаются. Разработчики базы данных могут добавить новое поле в таблицу, удалить существующее, поменять их местами и просто забыть нам об этом сказать. В таком случае наш пайплайн обрушится. Поэтому лучше всего всегда использовать список полей, которые нам нужны, и в

нужном порядке. Это `id`, `name`, `status`, `species`, `type`, `gender` и `origin_id`. Все эти поля перечисляются через запятую.

```
SELECT id, name, status, species, type, gender, origin_id
    FROM characters
```

2. Вывели все записи (строки)

   Как мы видим, нам вернулось 34 страницы по 25 записей в каждой. Вряд ли мы сейчас будем работать со всеми. Поэтому, если нам не нужны все строки, необходимо в конце любого SELECT писать LIMIT, например, 10. Это ограничивает набор возвращающихся записей, и мы можем работать с тем количеством, которое указали. Таким образом мы не нагружаем сеть, не забираем всю таблицу (которая может занимать десятки гигабайт).

```
SELECT id, name, status, species, type, gender, origin_id
    FROM characters
 LIMIT 10
```

> 💡 Необходимо ограничивать количество выбранных строк только в том в случае, если они действительно нужны не все.

Ниже приведены операторы DQL (Data Query Language), которые использовались в данном уроке:

SELECT	Основной оператор для запроса данных из одной или нескольких таблиц. Позволяет выбрать конкретные столбцы и строки на основе заданных условий
FROM	Указывает на таблицу в базе данных, к которой обращается запрос
LIMIT	Ограничивает количество возвращаемых строк

Резюме

- Таблица в реляционной БД состоит из полей (столбцов) и записей (строк), на пересечении которых находятся значения (значение поля для записи)
- Применительно к полям существует два понятия:
  - **Первичный ключ.** Он нужен для того, чтобы уникально идентифицировать какую-либо строку в таблице. Для всех записей в одной таблице первичным ключом является одно или несколько полей данной таблицы
  - **Внешний ключ.** Это ссылка из одной таблицы на первичный ключ другой таблицы
- Применительно к таблицами в реляционной БД существуют следующие связи:
  - **Один ко многим.** Одному первичному ключу может соответствовать несколько внешних ключей в соседней таблице
  - **Многие ко многим.** Каждая запись в одной таблице может соответствовать множеству записей в другой таблице и наоборот
- **SQL (Structured Query Language)** — это декларативный унифицированный язык. В разных СУБД он может незначительно отличаться. Перечень подъязыков SQL:
  - **DQL (Data Query Language)**
  - **DML (Data Manipulation Language)**
  - **DDL (Data Definition Language)**
  - **DCL (Data Control Language)**
  - **TCL (Transaction Control Language)**
- При работе в Redash можно использовать:
  - **Автокомплит** (автозаполнение)
  - **Форматирование**

    - **Сохранение файла** в нужном формате
    - **Просмотр** таблиц и наборов их полей
- При работе в Redash важно помнить:
    - Redash «обрубает» до двух чисел после запятой при отображении результата вычисления
    - Redash видоизменяет дату, которая хранится в Postgres
- При написании SQL-запроса к базе данных необходимо придерживаться следующих правил:
    - Всегда использовать список полей в том порядке, которые нам нужны (приписывать их в явном виде)
    - Ограничивать количество выбранных строк в том случае, если они действительно нужны не все