Лекция №1

> 💡 На первой лекции будем рассматривать структуру базу данных. Вспомним реляционные базы данных, разберем понятие “СУБД”, напишем первые запросы с помощью MySQL.

Что мы узнаем:

- Историю языка SQL
- Модель данных
- Узнаем про императивное и декларативное программирование
- Понятие внешнего и первичного ключа
- Основные операторы SQL
- Понятие СУБД
- Работа с UI MySQL
- Оператор SELECT

Здравствуйте, уважаемые студенты! На нашем курсе мы будем работать на языке SQL. Перед началом работы хотелось бы немного углубиться в историю.

Истоки SQL возвращают нас в 1970-е годы, когда в лабораториях IBM было создано новое программное обеспечение баз данных System R, а для управления язык - SEQUEL. Сейчас это название применяется в качестве альтернативы произношения SQL.

В 1979 году в компании Oracle (на тот момент компания называлась Relational Software), распознали коммерческий потенциал языка SQL и выпустила собственную модифицированную версию.

- **Аббревиатура SEQUEL расшифровывалась как Structured English QUEry Language — «структурированный английский язык запросов». Позже по юридическим соображениям («„sequel“ был торговой маркой британской авиастроительной группы компаний Hawker Siddeley») язык SEQUEL был переименован в SQL (Structured Query Language — «язык структурированных запросов»).**

Сегодня SQL стал стандартом языка запросов к базам данных. Он удовлетворяет как отраслевые, так и академические потребности и используется как на индивидуальных компьютерах, так и на корпоративных серверах. С развитием технологий баз данных приложения на основе SQL становятся все более доступными для обычного пользователя. В современной корпоративной культуре, знание SQL это одно из основных требований к аналитику. Давайте разбираться.

Поговорим о реляционных базах данных(БД)

Данные - информация, окружающая нас повсюду. Для визуализации данных чаще всего используют **таблицы.** Таблицы состоят из строк и столбцов. Они являются объектами, которые содержат все данные в базах данных. **Базы данных (БД) —** это структурная совокупность взаимосвязанных данных определённой предметной области: реальных объектов, процессов, явлений.
БД решает следующие задачи:

- Хранение данных

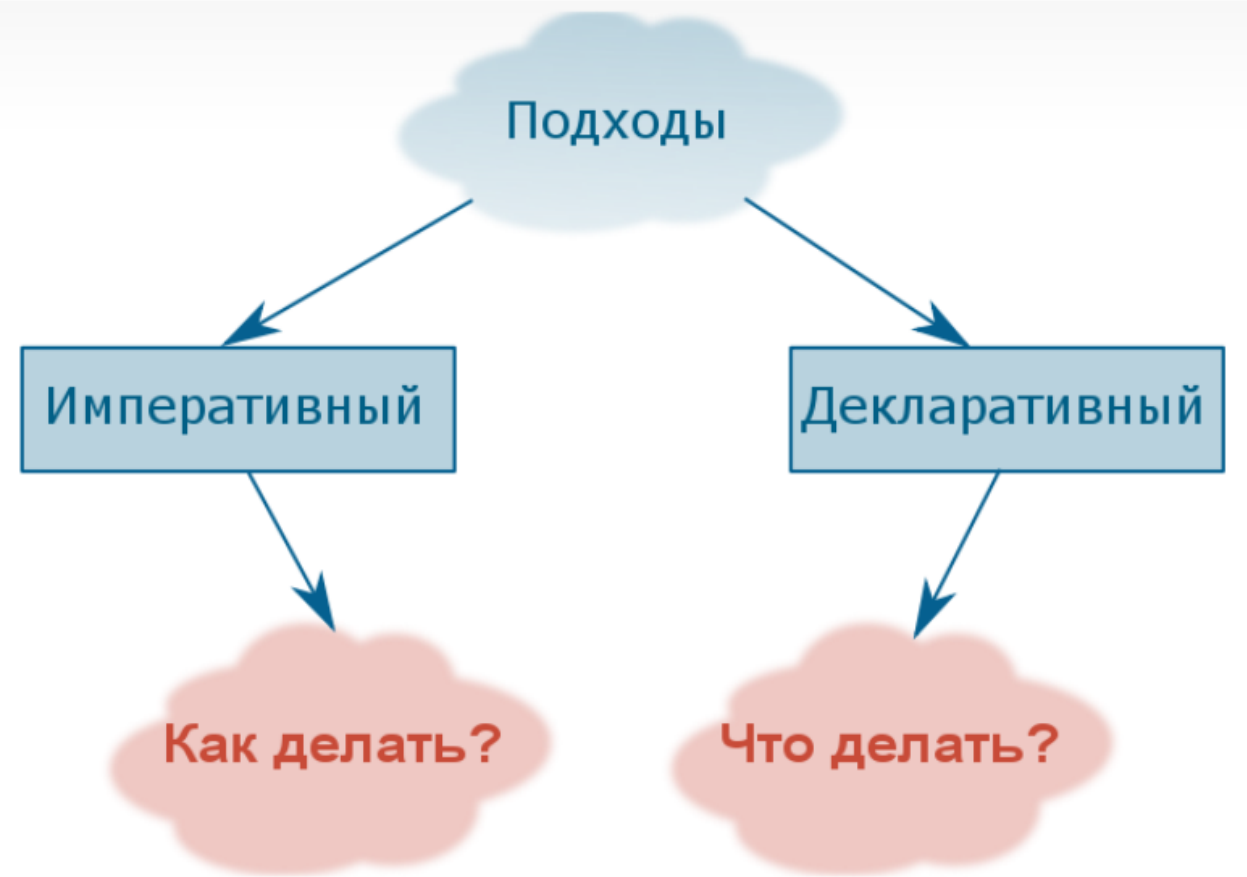
- Получение данных
- Обработка данных

Технологии БД были отнюдь не всегда. Изначально данные хранились в формате плоских файлов (flat files). Эти данные имели простую структуру: данные представляли из себя записи, разделенные на поля фиксированной длины. В реальной жизни происходит необходимость организовывать сложные записи, которые нужно перенести в БД. Основным понятием в теории баз данных является **модель данных**. Она характеризует способ организации данных и методы доступа к ним. Изначально были предложены иерархическая и сетевая модель данных. В ходе эволюций теории была разработана реляционная модель данных. Их особенность заключается в хранении данных в формате таблицы.

Id	Name	Group
1	Михаил	IT - Специалист | 7777 | выходные - утро |
2	Антон	IT - Специалист | 8888 | выходные - утро |
3	Тимофей	IT - Специалист | 9999| будни - вечер|

Таблицы, которые связаны между собой, называются **реляционными**. **Реляционная база данных** — база, где данные хранятся в формате таблиц, они строго структурированы и связаны друг с другом. Данный термин введен в 1969 году ученым из IMB, Эдгаром Коддом. Девять лет спустя технологию стали использовать в коммерческих целях. Преимуществом реляционных баз данных является способность осуществлять связи между элементами, тем самым, облегчая жизнь программисту. Ранее многие задачи решались с помощью процедурных языков программирования реализовывать операции каскадного удаления и обновления данных. Приведем пример: пусть информация о пользователях и заказах, которые они сделали, хранятся в двух файлах: “Пользователь”, “Заказы”. Для удаления заказа конкретного пользователя нужно организовать проход по каждому из файлов и удаление данных по конкретному условию. Работа с БД во многом организована иначе: SQl является декларативным языком программирования. Для результата мы указываем, **что** хотим получить без указания способа получения. До этого мы работали с

императивными языками, где для получения решения мы указывали, как получился результат.



Чтобы получить сумму 2 чисел, мы использовали 3 переменные: в первых 2 находились числа, в 3 - результат:

```
int firstNumber = 1;  // первое число
int secondNumber = 2; // второе число
int result = firstNumber + secondNumber;
```

Для решения описанной выше задачи можно и применять БД. Информация хранится в двух измерениях: **строчки** и **столбцы**. Строчки в таблице часто именуют как “записи”, также их называют “**кортежами**”. Столбцы называют “**полями**” или же “**атрибутами**”. Для удобного запоминания составим небольшую табличку:

Каждая запись разбита на несколько полей, представляющие из себя конкретный элемент. В нашем примере база данных, скорее всего, из нескольких таблиц (преподаватели, кураторы и т.д.). Понимание взаимосвязи таблиц - ключ к понимаю основной архитектуры баз данных.

Чтобы получить представление о работе реляционных БД, необходимо понимать, как организовывается связь между ними. Реляционная БД включает в себя множество таблиц, соединенных друг с другом ключевым полем. Каждая таблица содержит в себе **первичный ключ**. Первичный ключ (Primary key) – поле(или набор полей) позволяющее однозначно идентифицировать запись в БД. Если ключ состоит из нескольких полей его называют составным. Связь происходит по **внешнему ключу**. Внешний ключ (foreign key) - поле в таблице, значение которого соответствует первичному ключу в другой таблице. Обратите внимание, что внешний ключ может ссылаться на существующий первичный.

Не всегда в таблице по существующим данным можно однозначно идентифицировать запись, даже используя несколько полей, также значения этих полей могут изменится (изменение первичного ключа может повлечь изменение во многих связанных таблицах). В таких случаях часто используются автоматически генерируемые атрибуты добавляемые к таблице, например числовые последовательности увеличивающиеся при каждой вставке в таблицу.
Суррогатный ключ - автоматически сгенерированное уникальное поле, никак не связанное с информационным содержанием записи.
Естественный ключ — ключ, состоящий из информационных полей таблицы.
У таблицы может быть несколько ключей или не быть вообще, если ключей несколько то самый короткий из ключей выбирают в качестве первичного (это может быть как естественный так и суррогатный ключ).
Давайте рассмотрим эти понятия на примере задачи по созданию базы обучающихся, пример списка сведений о студентах, подумайте какие ключи тут можно выделить подумайте 1-2 минуты и разберем варианты:

Студенты

Фамилия	Имя	Год рождения	Паспорт серия	Паспорт номер
Иванов	Петр	1992	0111	121245
Пупкин	Федор	1995	1102	457879
Смирнов	Иван	1986	0013	787952
Смирнов	Степан	1997	0013	784593
Петрова	Ирина	1996	1802	596485

Серия и номер паспорта — хорошие кандидаты для первичного ключа (потенциальный ключ), набор полей “Фамилия”, “Имя” и “год рождения” — скорее, не будет уникальным с увеличением данных. Вернемся к набору из полей “Серия” и “Номер паспорта” посмотрим на недостатки этого естественного первичного ключа:

- может изменятся
- может отсутствовать
- возможны технические ошибки при вводе, опечатки приводящие к дубликатам
- достаточно широкий (два текстовых поля) — почему это важная характеристика? Если в базе много таблиц будут ссылаться на эту таблицу, то они должны будут использовать ключ таблицы, т. е. хранить эти поля. Пусть наша таблица “Студенты” связана с таблицей “Телефоны”.Таблица телефонов в данном случае будет выглядеть примерно так:

Телефоны_студентов

Студент_Паспорт_серия	Студент_Паспорт_номер	Телефон	Использование
0111	121245	89211285696	основной
1102	457879	89651238956	основной
0013	787952	65428967	основной
0013	787952	49626548596	дополнительный

Задача выделения первичного ключа не такая простая как могло показаться в начале, в большинстве случаев в реляционных базах используются суррогатные ключи, особенно если таблицы связаны между собой. К таблице добавляется одно числовое поле - id.

Студенты

Студент_id	Фамилия	Имя	Год рождения	Паспорт серия	Паспорт номер
1	Иванов	Петр	1992	0111	121245
2	Пупкин	Федор	1995	1102	457879
3	Смирнов	Иван	1986	0013	787952
4	Смирнов	Степан	1997	0013	784593
5	Петрова	Ирина	1996	1802	596485

Но где же можно вести БД?

Базу данных можно вести и на листе бумаги, бумажные ежедневники и блокноты также являются базами данных, на некоторых кстати нанесены по краям цветовые или буквенные метки, что по сути является индексом для упрощения поиска данных. Т.е., под базой данных понимаются непосредственно сами данные. Когда мы говорим про данные на жестком диске компьютера, например, то нам требуются специализированные программы для работы с ними, в русскоязычной терминологии класс таких программ получил название — система управления базами данных, сокращенно СУБД. В англоязычной терминологии DBMS - Database Management System. На сегодняшний день существует огромное множество различных СУБД от коммерческих до открытых разрабатываемых open-source сообществом, использующих разные модели хранения данных, различные технологии поиска и хранения данных. На сайте db-engines.com ежемесячно составляется рейтинг СУБД, сейчас в нем более 350 СУБД (https://db-engines.com/en/ranking). Давайте посмотрим на первые 20 из них:

Rank			DBMS	Database Model	Score		
Jun 2022	May 2022	Jun 2021			Jun 2022	May 2022	Jun 2021
1.	1.	1.	Oracle	Relational, Multi-model	1287.74	+24.92	+16.80
2.	2.	2.	MySQL	Relational, Multi-model	1189.21	-12.89	-38.65
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	933.83	-7.37	-57.25
4.	4.	4.	PostgreSQL	Relational, Multi-model	620.84	+5.55	+52.32
5.	5.	5.	MongoDB	Document, Multi-model	480.73	+2.49	-7.49
6.	6.	↑ 7.	Redis	Key-value, Multi-model	175.31	-3.71	+10.06
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	159.19	-1.14	-7.85
8.	8.	8.	Elasticsearch	Search engine, Multi-model	156.00	-1.70	+1.29
9.	9.	↑ 10.	Microsoft Access	Relational	141.82	-1.62	+26.88
10.	10.	↓ 9.	SQLite	Relational	135.44	+0.70	+4.90
11.	11.	11.	Cassandra	Wide column	115.45	-2.56	+1.34
12.	12.	12.	MariaDB	Relational, Multi-model	111.58	+0.45	+14.79
13.	↑ 14.	↑ 26.	Snowflake	Relational	96.42	+2.91	+61.67
14.	↓ 13.	↓ 13.	Splunk	Search engine	95.56	-0.79	+5.30
15.	15.	15.	Microsoft Azure SQL Database	Relational, Multi-model	86.01	+0.68	+11.22
16.	16.	16.	Amazon DynamoDB	Multi-model	83.88	-0.58	+10.12
17.	17.	↓ 14.	Hive	Relational	81.58	-0.03	+1.89
18.	18.	↓ 17.	Teradata	Relational, Multi-model	70.41	+2.02	+1.07
19.	19.	↓ 18.	Neo4j	Graph	59.53	-0.61	+3.78
20.	20.	20.	Solr	Search engine, Multi-model	56.61	-0.64	+4.52

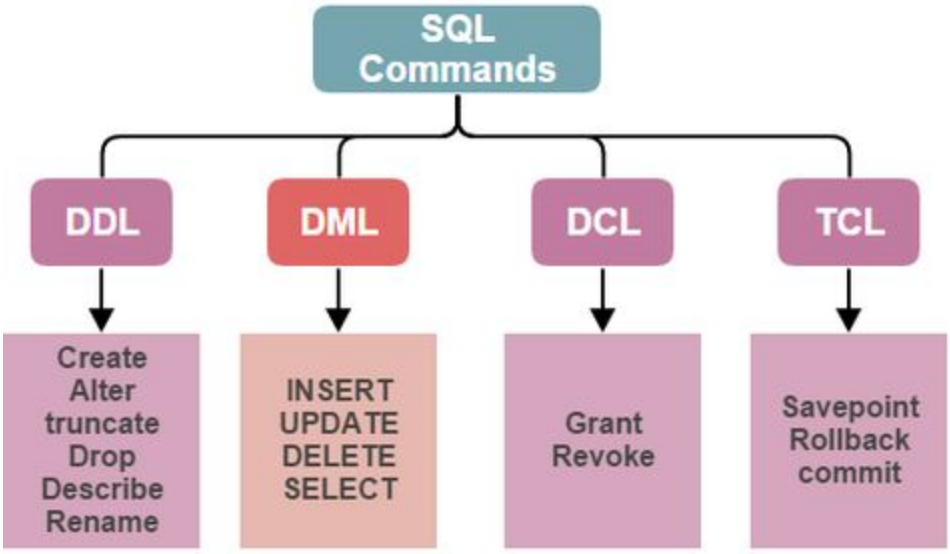
Итак, мы выяснили, что существует множество реляционных СУБД, есть стандартизированный язык SQL, но вместе с тем практически в каждой СУБД реализован свой язык запросов где-то расширяющий стандарт, где-то изменяющий, какие-то конструкции стандарта могут быть не реализованы, такие вариации получили названия ***диалектов*** SQL у некоторых наиболее популярных даже есть названия:

СУБД	**Диалект**
Oracle	PL/SQL (Procedural Language SQL)
MSSQL	T/SQL (Transact SQL)
Postgresql	PL/pgSQL (Procedural Language PostGres SQL)

Основные операторы SQL

Оператор (statement) *— это наименьшая автономная часть языка программирования, команда или набор команд. (примеры из языков программирования: оператор присваивания :=, оператор IF, While, For и другие). Программа представляет собой последовательность операторов.*

В языке SQL можно выделить несколько групп операторов по типу выполняемых ими задач:



DDL (Data Definition Language)

Эта группа операторов для определения данных. Они необходимы, когда нужно произвести манипуляции с таблицами. Эти операторы SQL используются в тех случаях, когда нужно создать в базе новую таблицу или, напротив, удалить старую. Они включают в себя следующие командные слова:

- **CREATE** — создание нового объекта в существующей базе.
- **ALTER** — изменение существующего объекта.
- **DROP** — удаление объекта из базы.

DML (Data Manipulation Language)

Эти операторы языка SQL предназначены для манипуляции данными. С помощью операторов мы можем изменять содержимое таблиц. Они помогают добавлять и удалять информацию, изменять значение строк, столбцов и прочих атрибутов. Эти операторы помогают удалить заказ, который отменили клиенты популярного маркетплейса. Это операторы:

- **SELECT** — позволяет выбрать данные в соответствии с необходимым условием.
- **INSERT** — осуществляют добавление новых данных.

- **UPDATE** — производит замену существующих данных.
- **DELETE** — удаление информации.

DCL (Data Control Language)

Операторы, позволяющие предоставлять право доступа к файлам. Делают файлы либо приватными, либо открытыми. Операторы необходимы, чтобы ограничить кого-либо из сотрудников в доступе к информации или, наоборот, позволить работать с базой новому специалисту.

- **GRANT**— предоставляет доступ к объекту.
- **REVOKE**— аннулирует выданное ранее разрешение на доступ.
- **DENY**— запрет, который прекращает действие разрешения.

TCL (Transaction Control Language)

Данный блок операторов предназначен для управления транзакциями. Слово “транзакция” знакомо нам из банковской сферы. На самом деле, транзакция - набор инструкции, которые выполняются как единое целое. У транзакции всего два исхода: проведена успешно, если все необходимые команды выполнены или откат, если в какой-либо инструкции произошёл сбой, то вся операция, включая предыдущие команды, отменяется. Пример - банковские платежи.

Представьте, что вы хотите заказать доставку пиццы, оплатив заказ онлайн. После ввода суммы зачастую банк высылает код подтверждения. При вводе кода мы наслаждаемся вкусной “Пепперони” , но если код не был введен, платеж отменяется автоматически

Детально о транзакциях и о требованиях к ним мы поговорим в самом конце нашего курса, на 6 лекции.

- **BEGIN TRANSACTION** — начало транзакции.
- **COMMIT TRANSACTION** — изменение команд транзакции.
- **ROLLBACK TRANSACTION** — отказ в транзакции.

- **SAVE TRANSACTION** — формирование промежуточной точки сохранения внутри операции.

Основные компоненты СУБД

Попробуем построить обобщенную архитектуру СУБД, несмотря на большое количество различных вендоров и СУБД можно выделить общие функциональные блоки:

а) подсистема постоянного хранения данных (Storage Engine). В большинстве случаев СУБД использует файлы и каталоги операционной системы, некоторые, например Oracle могут работать напрямую с дисками минуя слой операционной системы. Может использоваться сжатие данных.

б) парсер и транслятор запросов (Query parser). СУБД получает от пользователя SQL запрос (это текст) его необходимо проверить на синтаксис, перевести (транслировать) во внутренний формат, определить какие объекты, таблицы например используются

в) оптимизатор запросов (Query optimizer) Запрос не определяет четкого алгоритма действий над объектами (в каком порядке соединять таблице, какие индексы использовать и многие другие технические детали), поэтому СУБД пытается построить наиболее оптимальный план выполнения — алгоритм выполнения запроса. Дальше в СУБД будет выполняться выбранный план запроса.

г) подсистема выполнения (Query executor). Получает готовый план и шаг за шагом выполняет инструкции.

д) системы кэширования данных. Обращение в постоянное хранилище, к жесткому диску довольно дорогостоящий и медленный процесс, поэтому часто используемые данные СУБД пытаются кэшировать — хранить например в оперативной памяти или ssd дисках.

Большинство СУБД используют клиент-серверную архитектуру, на выделенном сервере или кластере устанавливается СУБД, с клиентских компьютеров выполняется подключение и передача запросов, вся вычислительная нагрузка выполнятся сервером. Для подключения к СУБД используются компоненты доступа — подпрограммы, иногда их называют драйверами, обычно они

предоставляются разработчиком СУБД, для использования их в языках программирования разработаны модули и фреймворки которые скрывают рутинную работу от программиста.

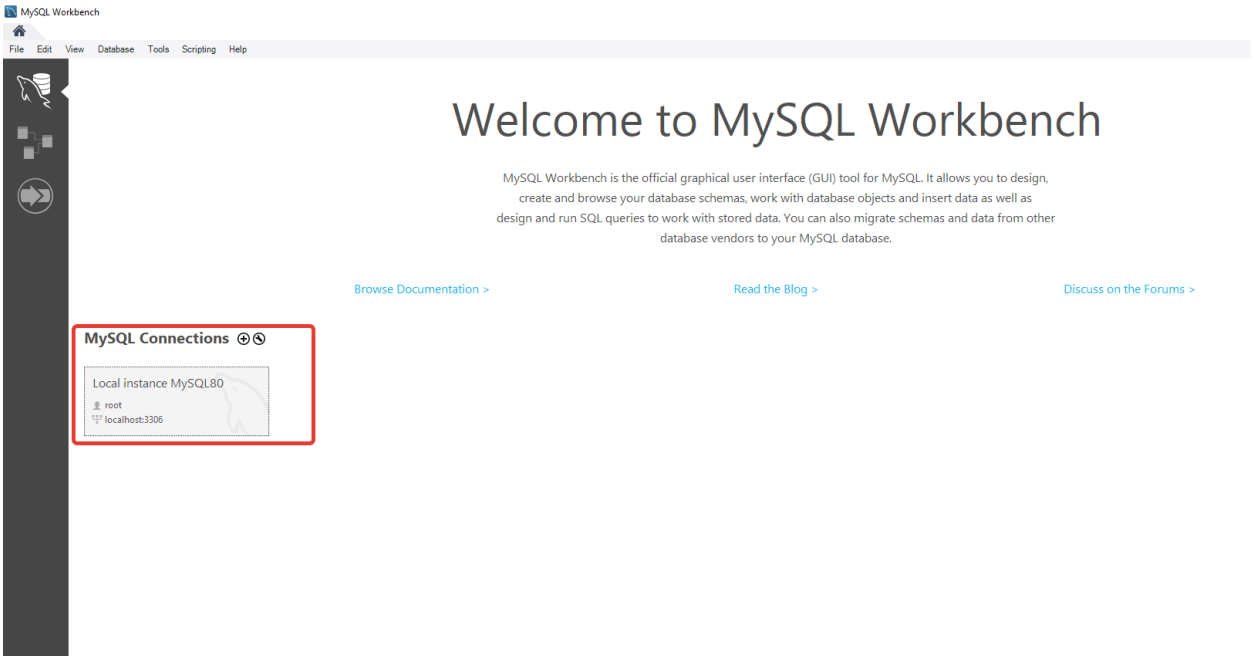
В нашей программе мы будем использовать MySQL.

MySQL появилась в 1995 году. Она изначально была легка, доступна и интуитивно понятна. В итоге ее стали использовать компании со всего мира. В настоящее время система MySQL является негласным стандартом для баз данных.

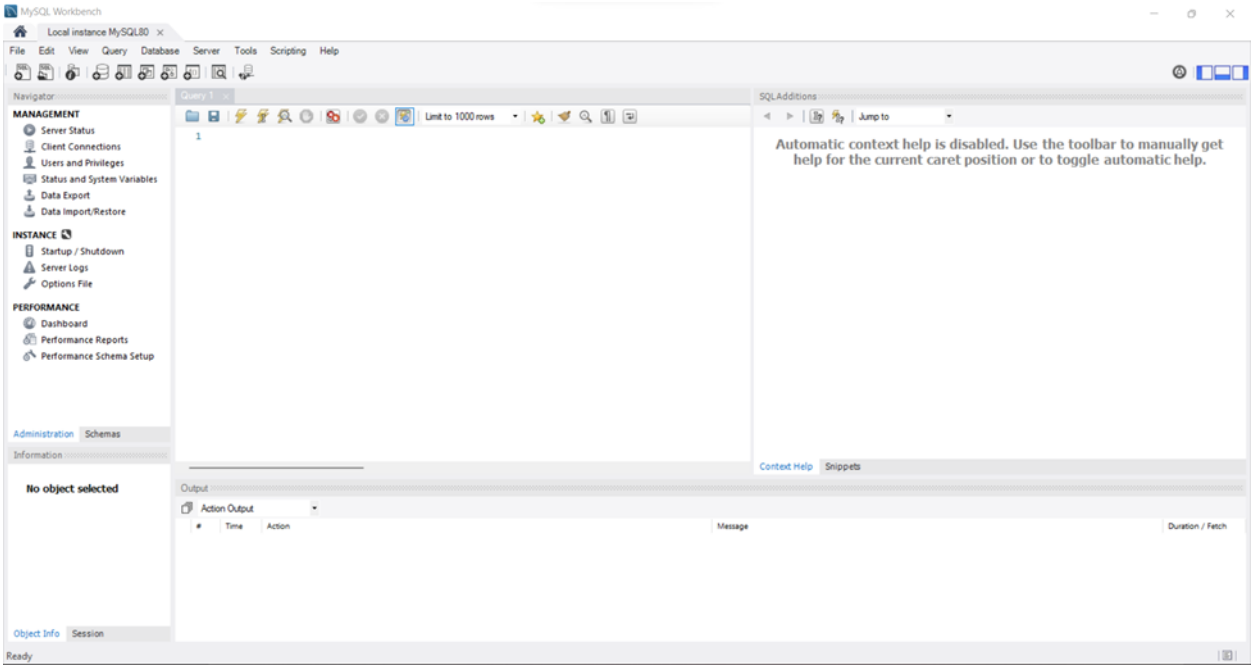
Программа гибкая и простая в использовании. Она даже позволяет пользователям поменять исходный код, чтобы настроить сервер баз данных MySQL конкретно под себя. Доплачивать за это не придется даже в расширенных коммерческих версиях. В установке этой СУБД также нет ничего сложного — процесс займет не больше получаса.

Работа с БД ,используя графический интерфейс, создание и просмотр объектов

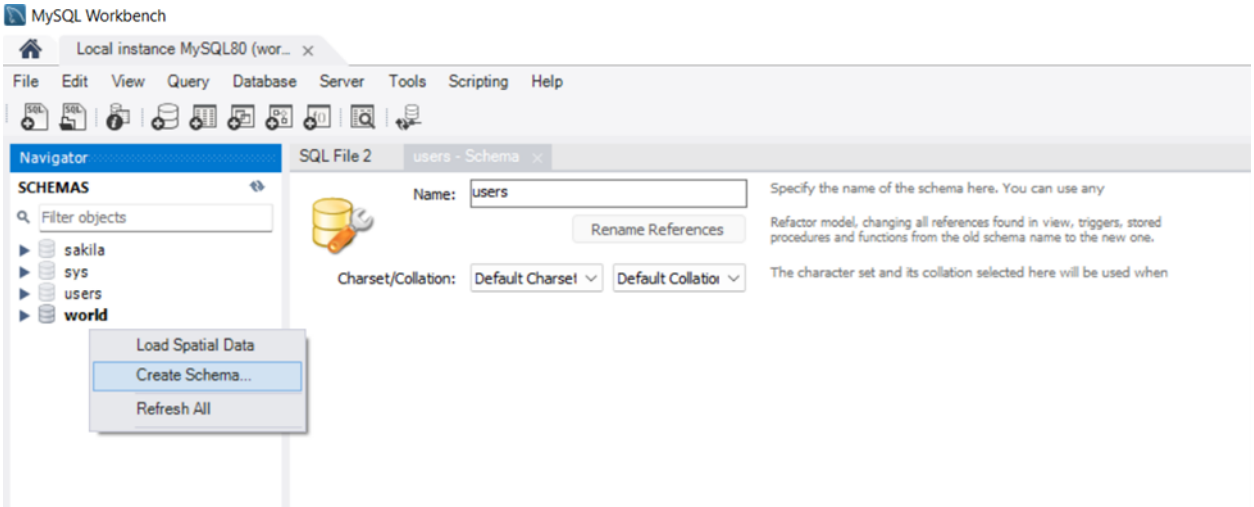
Рассмотрим интерфейс MySQl.



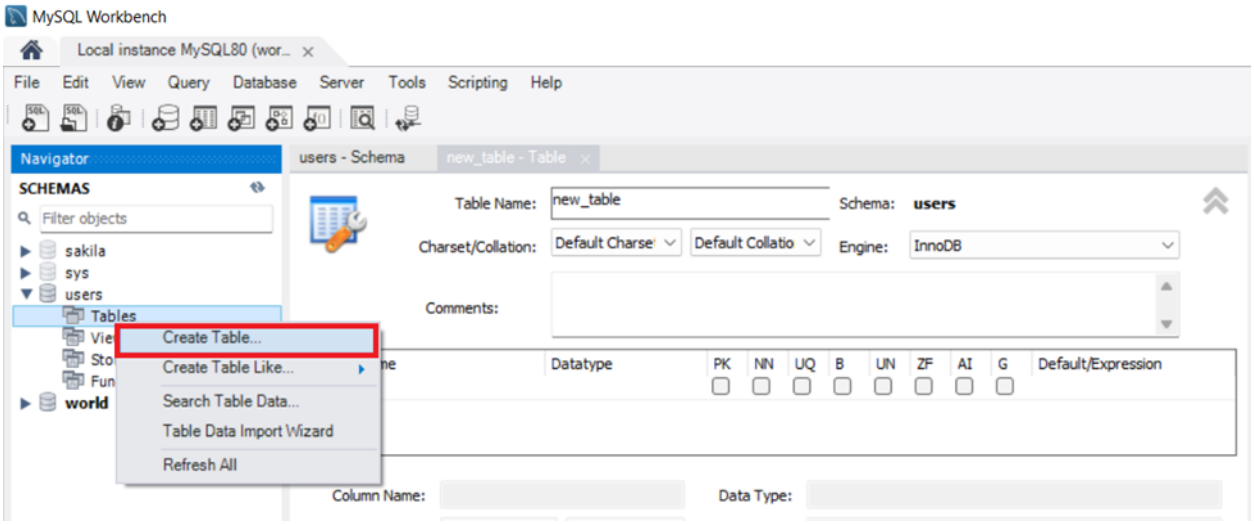
Необходимо войти в нашу БД: для этого нажимаем на имя нашего подключения и вводим данные для входа (их вы прописывали при установке MySQL). Если все выполнится корректно, нас ожидает вот такое окно:



Попробуем создать нашу первую схему: переходим во вкладку “Shemas” в правой нижней части экрана и нажимаем на клавишу создания схемы: UI мы будем использовать только на сегодняшнем уроке. Команды в терминале будем изучать на следующем уроке.



В результате получим следующее окно:



Имя нашей базы данных можно изменить со стандартного на “**users**”. Обратите внимание, что имя должно стоять из 1-2 слов, без нижних подчеркиваний. Кодировка utf-8 поддерживает и русские символы.

Вернемся к базе данных. Первую табличку назовем **students**. Она будет хранить данные о пользователях информационной системы, в поле “table Name” впишем имя таблицы, в разделе формы “Columns” создадим поля таблицы:

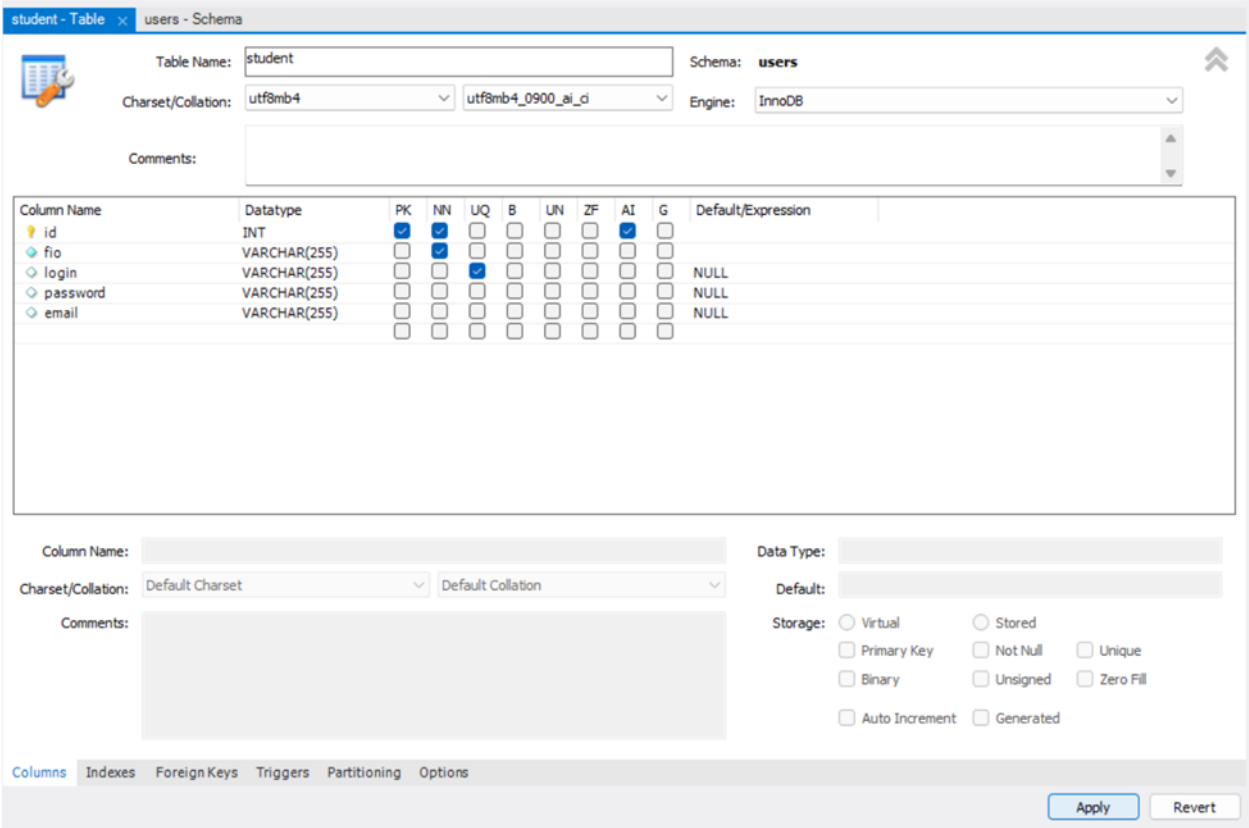
— **Первое поле id** будет содержать уникальный номер пользователя, зададим ему свойства: Auto Increment (AI), Not Null (NN), Primary key(PK) и Unique (UQ), в разделе Data type выберем целочисленный тип *integer*.

— **Второе поле fio**, где будет хранится Ф.И.О. пользователя, установим полю свойства: Not Null, в разделе Data type выберем строковый тип VARCHAR и зададим количество символов в 255.

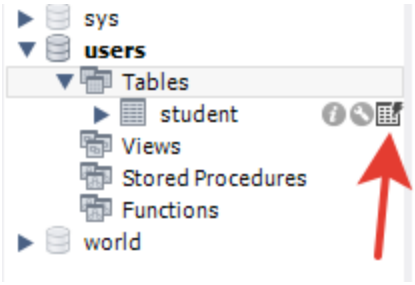
— **Третье поле login**, будет содержать логин пользователя, оно должно быть уникальным, как и поле id, поэтому установим ему свойство Unique и зададим количество символов в 255.

— Следующие поля: **password** содержащее пароль, e_mail содержащее адрес электронной почты и поле **type**, содержащее тип пользователя будут без особых свойств, со строковым типом VARCHAR длинной в 255 символов.

После проделанных манипуляций форма с именем таблицы users будет выглядеть так:



Для сохранения нашей таблицы используем кнопку “Apply”.
Добавим данные в нашу табличку, заполнив первыми значениями:



Запрос выборки данных с простыми условиями

Давайте разберем синтаксис простого SQL запроса выборки данных и научимся читать его определение. Многие операторы вам уже знакомы

1. Вывод всех данных из таблицы

SELECT * **FROM** student;

Этим запросом будут выведены все строки из таблицы student и все столбцы (* - означает все доступные стобцы)

2. Вывод ограниченного числа столбцов — нужно явно перечислить столбцы

SELECT fio, login **FROM** student;

Этим запросом будут выведены все строки из таблицы student, но только столбцы fio и login

3. Применение фильтров, отбор данных по условиям

SELECT * **FROM** student **WHERE** login=’test2’;

4. Применение оператора “LIKE”:

Оператор имеет ряд символов подстановки:

“%:” - любая подстрока, которая имеет любое количество символов, либо является пустой.

Пример: "WHERE Car LIKE 'Audi%' соответствует таким значениям как: "Audi A3" / "Audi Q3”,”Audi TT”

- "Audi A3"
- "Audi Q3”
- ”Audi TT”
- “Audi RS4”

“_:” - любой одиночный символ

Пример: "WHERE Car LIKE 'Audi A_' соответствует таким значениям как:

- "Audi A3"
- "Audi A4"
- "Audi A6"
- "Audi A8"

После буквы “A” идет 1 одиночный символ.

Полезные ссылки и рекомендации:

- https://habr.com/ru/company/oleg-bunin/blog/348172/ - естественные и суррогатные ключи
- Руководство по стилю написания SQL: https://www.sqlstyle.guide/ru/
- Старайтесь не использовать русские буквы при работе с СУБД

Книги:

- “Изучаем SQL”, книга Бейли Л.
- Алан Бьюли "Изучаем SQL" (2007)
- Энтони Молинаро "SQL. Сборник рецептов" (2009)