Лекция №1

Инструменты для работы с базами данных

Об авторе

Меркушов Михаил Сергеевич - более 3 лет занимаюсь разработкой баз данных и созданием программ. Оканчиваю Волгоградский технический универститет по специальность “Информатика и вычислительная техника”, профиль “Автоматизированное проектирование киберфизических систем”. В данный момент занимаюсь разработкой приложения для людей с ограниченными возможностями для Бельгийского университета (компания mail инвестиционная технологическая корпорация)

Введение

Мы начинаем курс посвященный работе с реляционными базами данных, наиболее распространенной моделью хранения данных. Данный курс будет полезен как разработчикам и инженерам по тестированию, поскольку практически каждое приложение работает с данными и обращается к различным СУБД, так и аналитикам для извлечения, фильтрации и трансформации данных.

На первом уроке мы рассмотрим какие инструменты существуют, вспомним основные понятия реляционных баз данных. Обратимся к истории возникновения СУБД и языка SQL поговорим про его особенности, рассмотрим процесс установки СУБД MySQL Workbench и напишем запросы выборки данных.

На следующих уроках курса будем знакомится более детально с операторами языка SQL и учится правильно и эффективно применять их.

Термины лекции

Система управления базами данных, сокр. СУБД (англ. Database Management System, сокр. DBMS) *— совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.*

Реляционные базы данных *(от англ. Relation – связь) — базы данных, в которых данные распределены по отдельным, но связанным между собой таблицам.*

Кортеж (tuple) *— это множество пар {имя аттрибута, значение}.*

Отношение (relation) *— это множество кортежей, соответствующих одной схеме.*

Первичный ключ (Primary key) *– поле(или набор полей) позволяющее однозначно идентифицировать запись в БД. Если ключ состоит из нескольких полей его называют* ***составным.***

Суррогатный ключ *— автоматически сгенерированное уникальное поле, никак не связанное с информационным содержанием записи.*

Естественный ключ *— ключ состоящий из информационных полей таблицы.*

SQL *— Структурированный язык запросов*

Декларативный язык программирования *— язык программирования описывающий результат действия, не описывающий шаги для получения результата.*

Оператор (statement) *— это наименьшая автономная часть языка программирования, команда или набор команд.*

ODBC *(Open Database Connectivity) — программный интерфейс (API) доступа к базам данных.*

Оглавление

Понятия базы данных, СУБД

Вспомним какие основные задачи решают базы данных:

1.хранение, данные должны быть надежно сохранены

2.получение данных, должны быть средства чтения данных

обработка данных

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:

398 systems in ranking, June 2022

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

О том, как составляется рейтинг можно прочитать на сайте, для нас важно отметить разнообразие СУБД и большой отрыв первый пятерки. Ещё одна важная деталь — это различные используемые модели данных, мы видим тут много реляционных (используют структурирование данных по таблицам), меньше документно-ориентированных (данные хранятся в виде отдельных документов), графовых (данные выстраиваются в виде связанных графов) и других СУБД, некоторые СУБД поддерживают работу с различными моделями данных.

В этом курсе мы рассмотрим работу с реляционными базами данных. Данная модель получила наиболее широкое распространение она основана на математическом аппарате и покрывает большую часть прикладных задач, новые - постреляционные модели часто строятся на основе реляционной модели.

Сначала познакомимся с наиболее употребляемыми понятиями:

Реляционные базы данных *(от англ. Relation – связь) — базы данных, в которых данные распределены по отдельным, но связанным между собой таблицам.*

Иногда вам могут встретится термины из реляционной алгебры, поэтому давайте их приведем и соотнесем с применяемыми на практике:

Кортеж (tuple) *— это множество пар {имя аттрибута, значение}.*

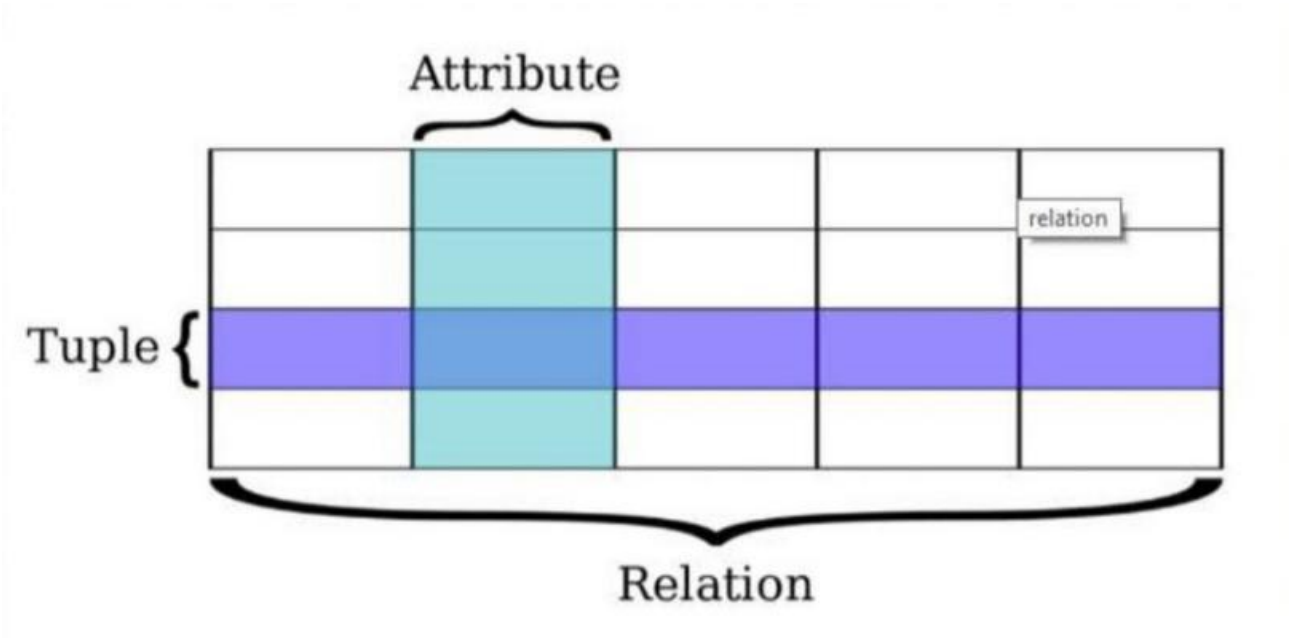
(например {Фамилия, Петров}, {Телефон, +7921-123-56-69}) — фактически это строка таблицы, где имена атрибутов — это столбцы таблицы.

Отношение (relation) *- это множество кортежей, соответствующих одной схеме.*

например:

{
{ {Фамилия, Петров}, {Телефон, +7921-123-56-69}},
{ {Фамилия, Смирнов}, {Телефон, +7956-987-78-21}},
{ {Фамилия, Чехов}, {Телефон, +7955-968-24-36}}
}

Такое множество может быть визуально представлено в виде таблицы, где описание схемы будет определено столбцами таблицы:



Небольшой пример на кортежи:

В таблице “Сотрудники” имеется кортеж данных:

{ [**Идентиф_код**], **'2931123455'** }

{ [**Фамилия_и_инициалы**], '**Петренко П.П.**' }

Идентиф_код	**Фамилия_и_инициалы**
2931123455	**Петренко П.П.**

Первичный ключ (Primary key) *– поле(или набор полей) позволяющее однозначно идентифицировать запись в БД. Если ключ состоит из нескольких полей его называют* ***составным.***

Например, в таблице номеров телефонов — номер телефона может быть ***первичным ключом****.*

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. (в некоторых СУБД их называют

автоинкрементом).

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

Думаю что тут всё достаточно очевидно,

Серия и номер паспорта — хороший кандидат 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:

1.может изменятся

2.может отсутствовать

3.возможны технические ошибки при вводе, опечатки приводящие к дубликатам

4. 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. 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.

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

Перечислим свойства первичного ключа:

1.однозначно идентифицирует строку таблицы, задан для каждой строки

2.неизменяемый или редко изменяемый

3.достаточно узкий

Основные термины из теории реляционных баз данных, которые нам понадобятся в курсе мы вспомнили, можно двигаться дальше.

Язык SQL

В конце 1970-х компанией IBM был разработан прототип СУБД System R, в ней был реализован язык запросов (Structured Query Language) **SQL** — Структурированный Язык Запросов. В дальнейшем он стал применяться во многих СУБД и в силу своего широкого распространения постепенно стал стандартом для языков запросов в реляционных СУБД. В дальнейшем он изменялся и расширялся, но были приняты стандарты, которые поддерживаются многими СУБД полностью или частично.

Первый стандарт был утвержден в 1986 году, получил название ANSI/SQL и многие СУБД поддерживают его полностью.

1986 — первый международный стандарт, SQL-86 ещё называют ANSI/ISO

1992 — стандарт SQL-92 (SQL2)

1999 — стандарт SQL3:1999 (SQL3)

2003 — стандарт SQL 2003

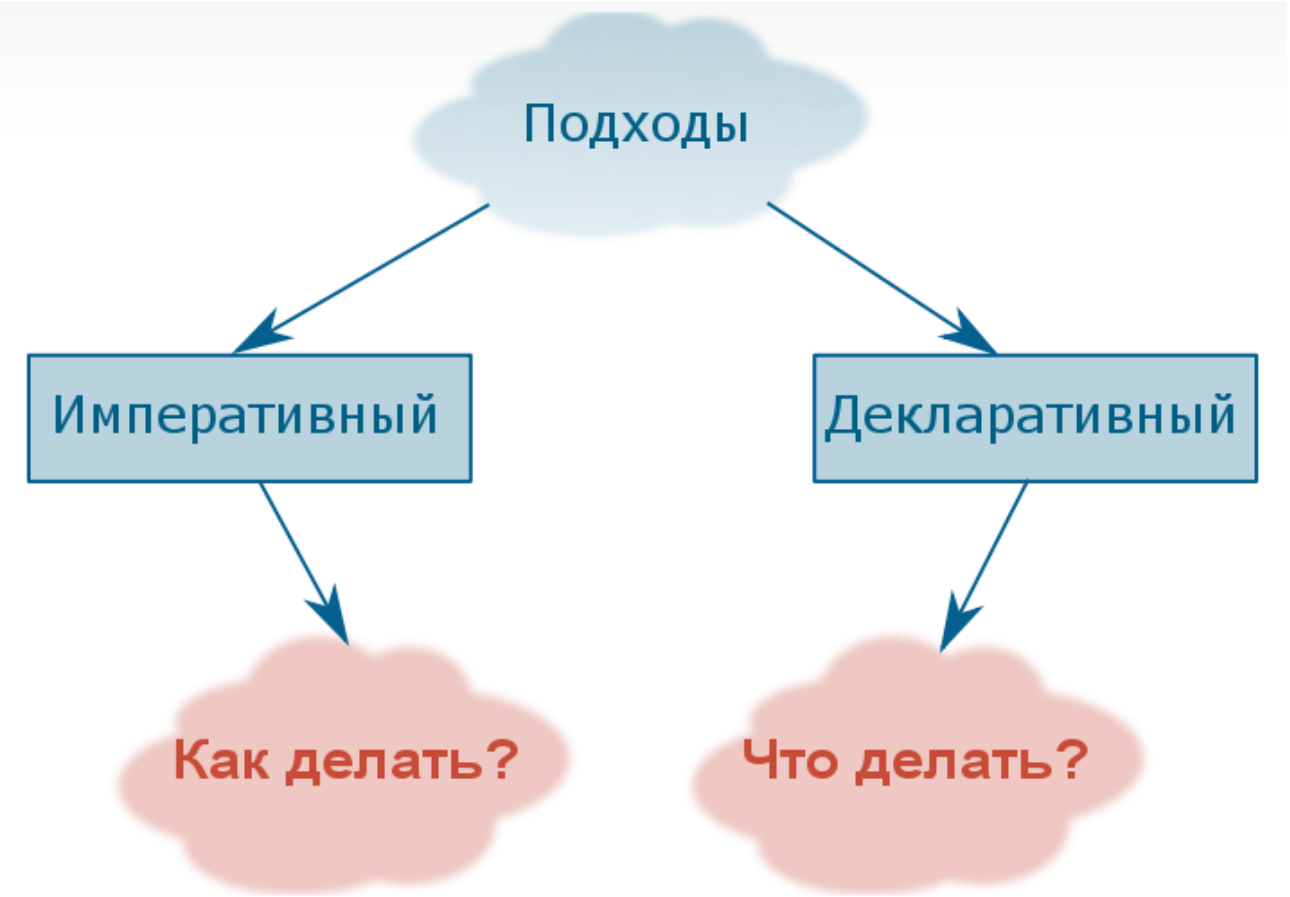
2006 — стандарт SQL 2003

2008 — стандарт SQL 2008

2011 — стандарт SQL 2011

2016 — стандарт SQL 2016.

Наверное, многие уже знакомы с языками программирования, в отношении SQL нужно отметить, что язык описывает результат действия. Если, например на python программист пишет, как получить требуемый результат, описывает все алгоритмы и шаги, то в SQL описывается результат, что требуется получить, не указывая какие алгоритмы и шаги выполнить для получения. Такие языки называются ***декларативными***.



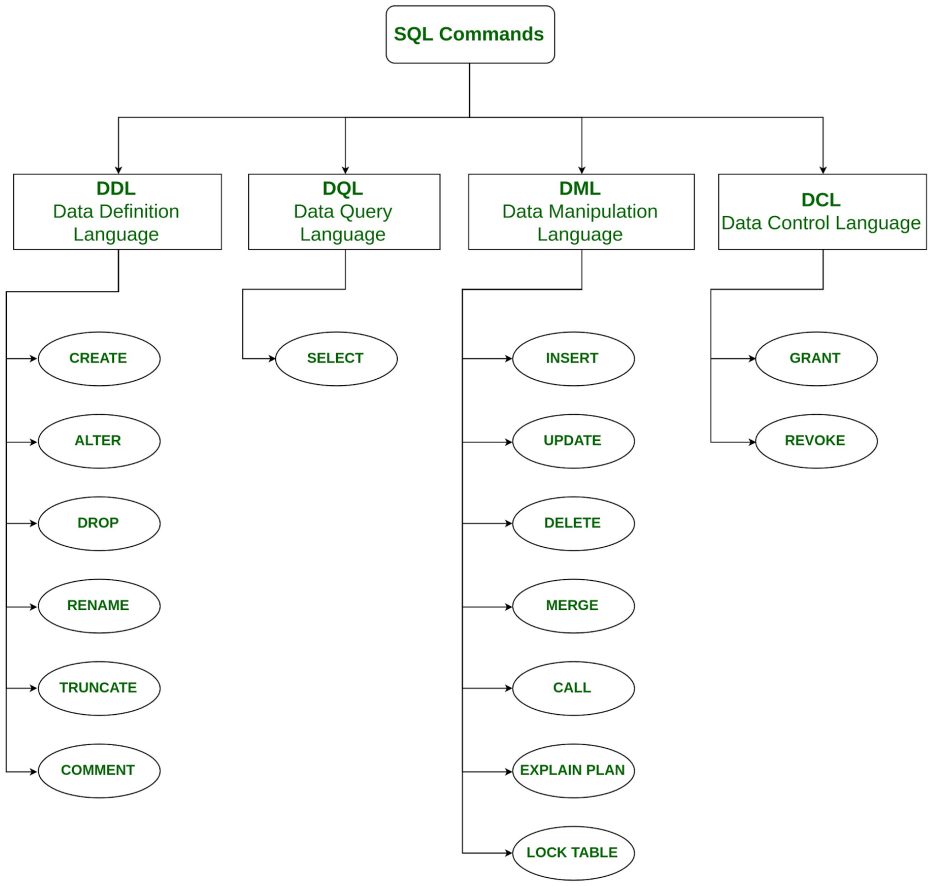
Итак, мы выяснили, что существует множество реляционных СУБД, есть стандартизированный язык SQL, но вместе с тем практически в каждой СУБД реализован свой язык запросов где-то расширяющий стандарт, где-то изменяющий, какие-то конструкции стандарта могут быть не реализованы, такие вариации получили названия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



Data Definition Language (DDL) – это группа операторов определения данных. С помощью этих операторов мы определяем объекты базы, это операторы:

create - создание новых объектов базы данных

alter – изменение существующих объектов

drop – удаление объектов

Data Manipulation Language (DML) – это группа операторов для манипуляции с данными. К ним относятся:

insert – добавление новых данных

update – изменение данных

delete – удаление данных

select – выборка данных

Data Control Language (DCL) – группа операторов определения доступа к данным, с помощью них мы можем управлять правами доступа к объектам базы.

Grant – предоставить права на объект базы

revoke – отозвать у пользователя права на объект

Transaction Control Language (TCL) – группа операторов для управления транзакциями. ***Транзакция*** - это набор команд или инструкций которые выполняются как единый блок, в котором либо все команды успешно выполнены и все изменения сохранены, либо при возникновении ошибки, ни одна команда из блока не будет выполнена и изменения не будут внесены в базу, база останется в том же состоянии в котором была до выполнения транзакции. Про транзакции в базах данных мы поговорим детальнее в следующих уроках, пока же перечислим основные команды для работы с ними:

begin transacton – определяет начало транзакции

commit transaction — применяет транзакцию, конечная точка

rollback transaction — откатывает все изменения транзакции

save transaction — устанавливает промежуточную точку сохранения внутри транзакции

В принципе это и все операторы sql, мы посмотрели обобщенно на этот инструмент и дальше будем детальнее разбираться с каждым оператором.

Сейчас посмотрим что СУБД делает, получая от пользователя инструкции на языке SQL.

Основные компоненты СУБД, установка

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***). Как мы уже говорили 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. Как правило модули для работы с базой данных предоставляют api для подключения к бд, выполнения запроса и получения результирующего набора данных. Для упрощения взаимодействия с СУБД были разработаны универсальные стандарты, широкое распространение получил ODBC

ODBC (Open Database Connectivity) — *это программный интерфейс (API) доступа к базам данных.*

Этот API не зависит ни от одной СУБД или операционной системы, поддерживается большинством современных СУБД.

В среде java широко распространен стандарт JDBC (Java DataBase Connectivity).

В этом курсе мы будем использовать СУБД MySQl, сейчас рассмотрим основные моменты установки. Презентации с

инструкцией приложены к материалам урока и вы сможете выполнить установку и минимальную настройку, если возникнут трудности то разберем их на семинаре.

Клиентские приложения для работы с СУБД

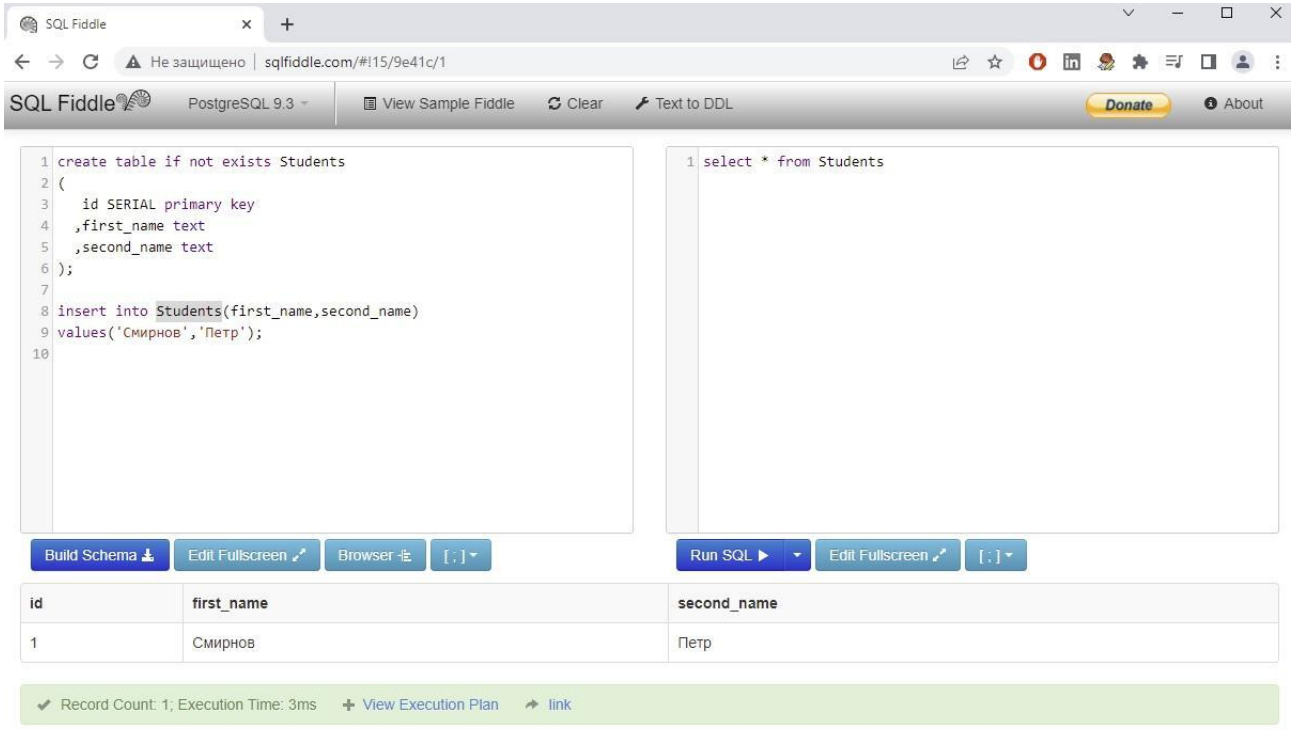
Для работы с СУБД и написания SQL запросов существует отдельный класс приложений. Часть из них предоставляются разработчиками СУБД например SQL Server Management Studio, так и разрабатываются сторонними компаниями например DataGrip от компании JetBrains.

Название	**Разработчик**	**Требуется лицензия**	**Поддерживаемые СУБД**
Dbeaver	open-source	нет	Более 20ти
Azure Data Studio	Microsoft	нет	Более 5ти, плагины
dbForge	Devart	да	Более 3ех
DataGrip	JetBrain	да	Более 10ти
MSMS	Microsoft	нет	MSSQL
PGAdmin	open-source	нет	Postgresql

Иногда для отладки или чтобы обменяться кодом используются online sql песочницы, они позволяют выполнить запрос прямо на сайте:

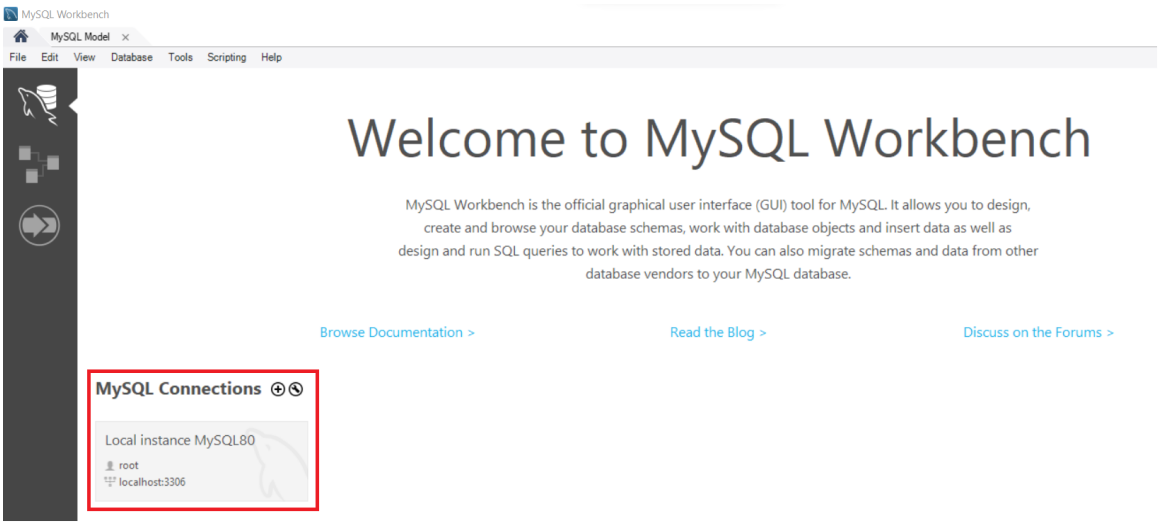
1.sqlfiddle.com
2.www.db-fiddle.com

Под песочницей (sandbox) – понимается изолированная среда разработки. Обычно такая среда используется для тестирования программ, безопасного выполнения полученных из не проверенных источников программ. Применительно к базам данных такие среды также позволяют быстро проверить запрос на разных СУБД.

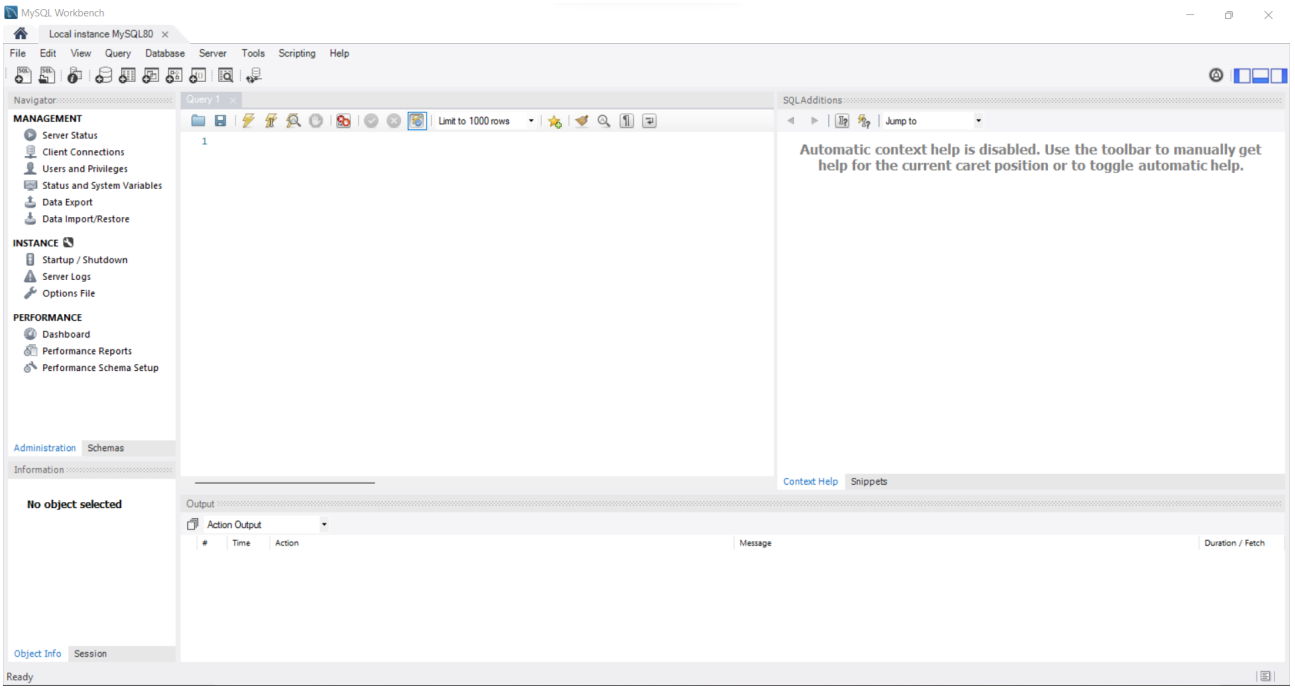


Работа с БД ,используя графический интерфейс, создание и просмотр объектов

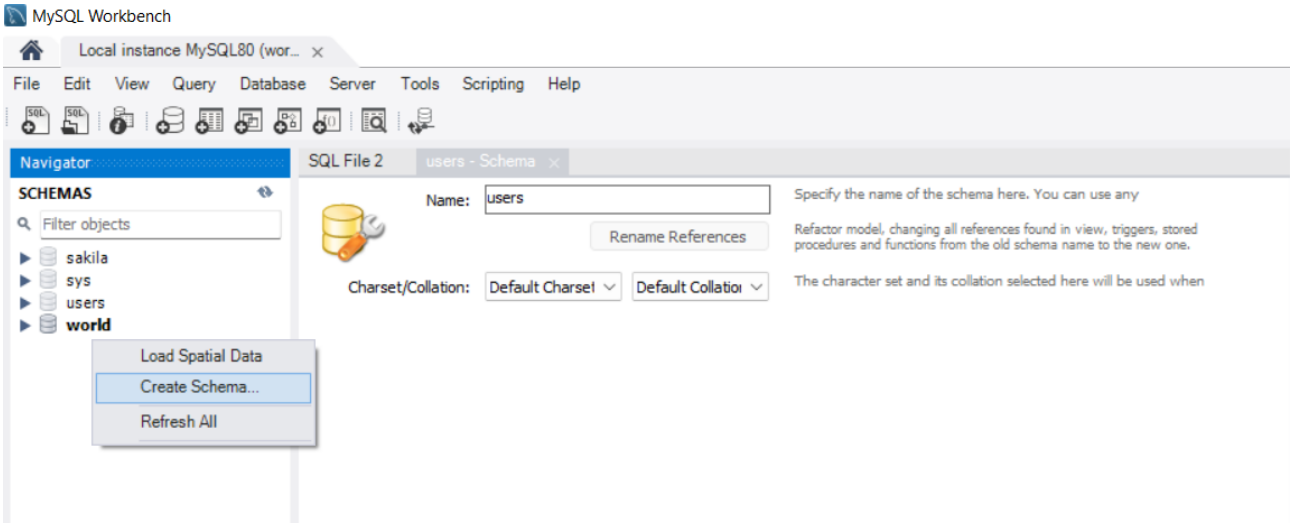
Рассмотрим интерфейс MySQl.



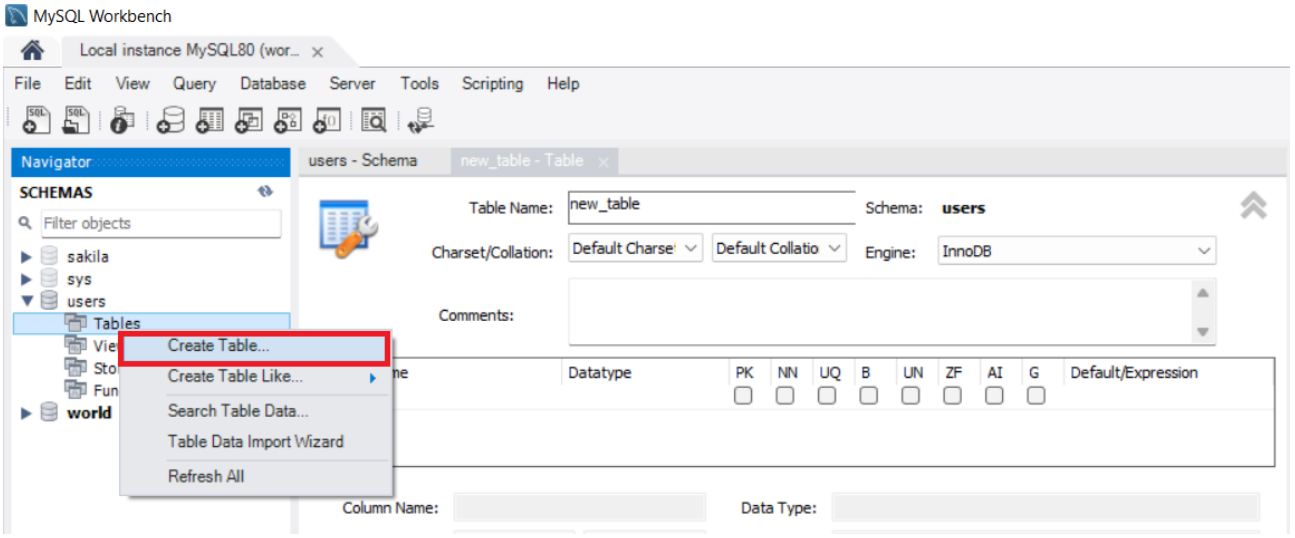
Необходимо войти в нашу БД: для этого нажимаем на имя нашего подключения и вводим данные для входа (их вы прописывали при установке MySQL). Если все выполнится корректно, нас ожидает вот такое окно:



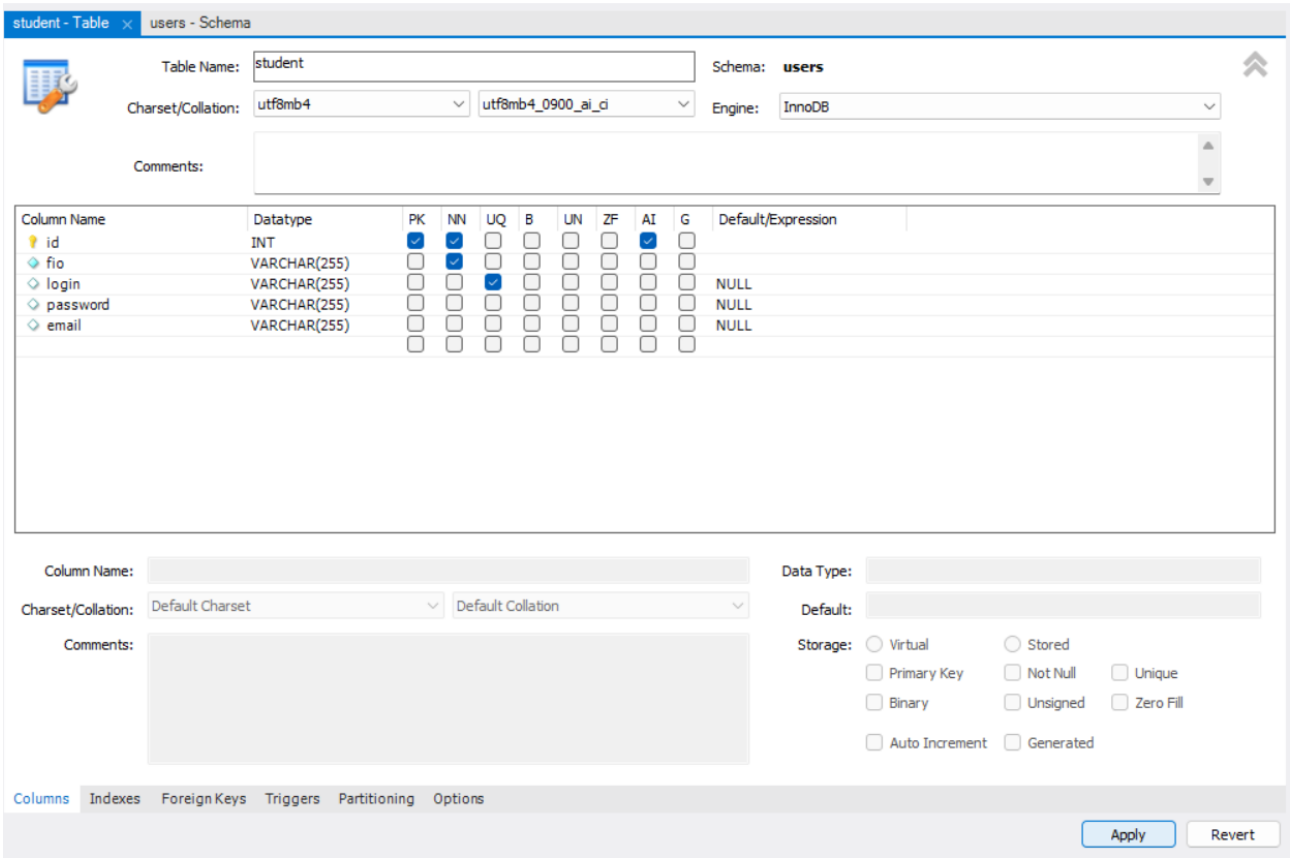
Попробуем создать нашу первую схему: переходим во вкладку “Shemas” в правой нижней части экрана и нажимаем на клавишу создания схемы:
UI мы будем использовать только на сегодняшнем уроке.
Команды в терминале будем изучать на следующем уроке.



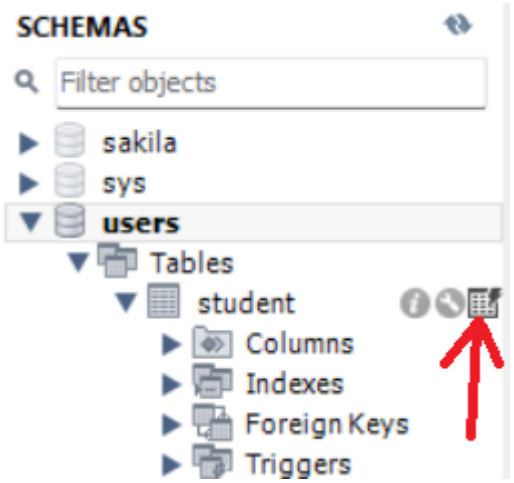
В результате получим следующее окно:



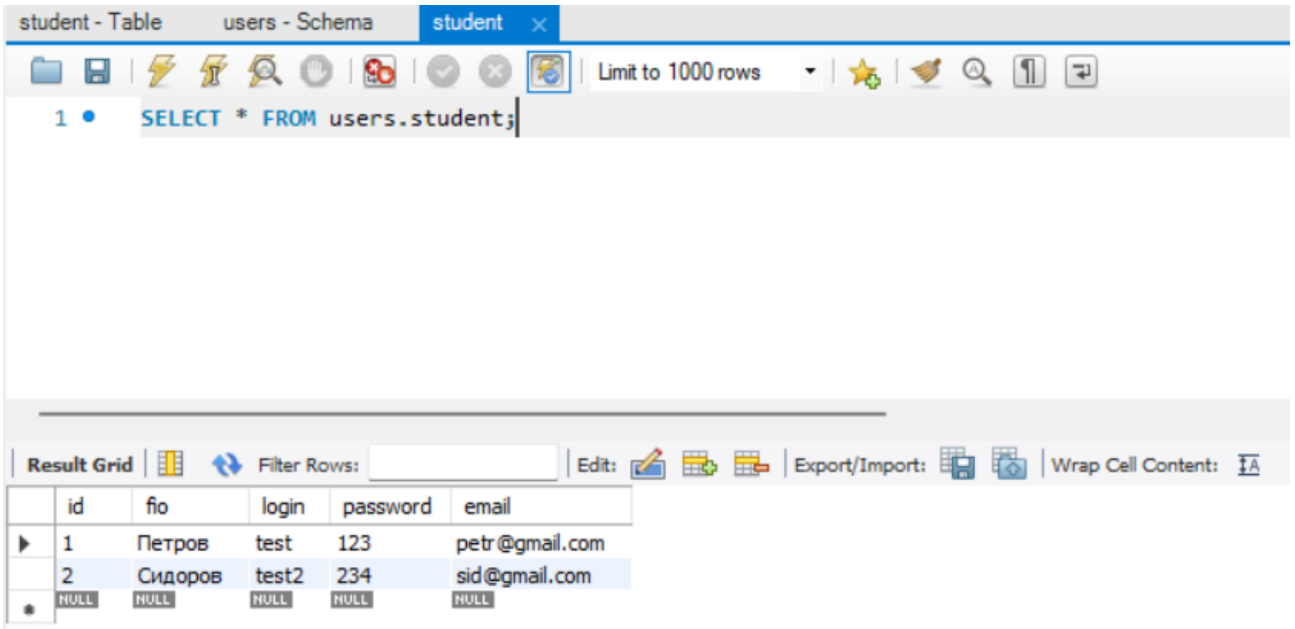
Имя нашей базы данных можно изменить со стандартного на “**users**”. Обратите внимание, что имя должно стоять из 1-2 слов, без нижних подчеркиваний. Кодировка utf-8 поддерживает и русские символы.
Вернемся к базе данных. Первую табличку назовем **students**. Она будет хранить данные о пользователях информационной 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:



Поле ID заполняется автоматически при помощи свойства “автоинкремент”:



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';
```

Итоги

Сегодня мы повторили основные понятия реляционных баз данных, узнали какие программные средства применяются для работы с базами данных, из каких компонентов состоит СУБД, как подключиться к СУБД и начать работать с базами данных, начали изучать язык SQL и написали запросы выборки данных.

На следующей лекции мы продолжим изучение языка sql, научимся писать запросы создания таблиц, вставки и

редактирования данных. Рассмотрим какие типы данных используются в базах, какие ограничения они накладывают и научимся правильно выбирать тип колонки.

Домашнее задание

Попробуйте установить СУБД MySQl. Спроектируйте структуру базы данных для хранения информации картинной галереи. Требуется хранить сведения о картинах.

Галерее для работы требуется получать сведения о картине:

1.дату реставрации,

2. дату поступления в галерею,

3.название картины,

4.ФИО художника,