ЕЛЕНА ЛИТВАК

SQL С НУЛЯ И БЫСТРО

Елена Литвак

SQL с нуля и быстро

«Издательские решения»

Литвак Е.

SQL с нуля и быстро / Е. Литвак — «Издательские решения»,

ISBN 978-5-00-594959-2

Книга подойдет всем, кто хочет изучить язык запросов SQL, но не знает с чего начать. В ней максимально подробно, доступно с большим количеством скриншотов разобраны практические примеры на основные виды SQL-запросов. Информация разбита на семь разделов. Если неспеша изучать по одному разделу в неделю, то через два месяца вы будете человеком, который разбирается в SQL и умеет делать запросы достаточно высокой сложности. Но можно двигаться и гораздо быстрее.

ISBN 978-5-00-594959-2

Содержание

SQL с нуля и быстро

Елена Литвак

ISBN 978-5-0059-4959-2
Создано в интеллектуальной издательской системе Ridero

ВВЕДЕНИЕ

Дорогие друзья, эта книга предназначена для тех, кто хочет изучить SQL, но не знает с чего начать. Вокруг море информации, есть платные курсы, есть множество книг и толстых справочников по SQL. А, возможно, сухая и непонятная методичка к курсовой работе по базам данных ждет внимания уже почти целый семестр… Неясно за что же из этого нужно хвататься в первую очередь, чтобы быстро начать применять нужные знания в конкретных практических задачах. Особенно грустно становится, если открываешь справочник и видишь там общий синтаксис инструкции SQL с кучей квадратных скобочек и лишних опций. Что из этого нужно обязательно, а что можно пропустить? Как делать в моем конкретном случае?

Дело в том, что практические задачи, требующие знания SQL, бывают разными. С программистами и тестировщиками все понятно, им без баз данных никуда. Но есть еще аналитики данных, маркетологи, экономисты, которые не пишут код, ничего не проектируют и не разрабатывают, но постоянно анализируют большие массивы данных. Для них тоже SQL является базовым инструментом. Хотя этим специалистам нужен не весь SQL целиком, а только основное ядро языка. Но нигде не написано, что входит в это ядро…

Эта книга подойдет для всех. И для студентов направлений обучения, связанных с информационными технологиями, которые только начали изучать SQL, и для аналитиков, и для маркетологов.

Информация в книге разбита на семь разделов. Если неспеша изучать по одному разделу в неделю, то через два месяца вы будете человеком, который разбирается в SQL и умеет делать запросы достаточно высокой сложности. Но можно двигаться и гораздо быстрее.

Как устроена книга? В ней последовательно от простого к сложному на практических примерах изложены основы языка SQL. Как раз то самое ядро, которое нужно и маркетологам, и программистам. При этом ничего лишнего.

Для изучения материала книги желательно иметь самое общее представление о том, как связываются таблицы в реляционной базе данных, интуитивно понимать, что такое первичный и внешний ключ. Сложных предметных областей с большим количеством таблиц и связей здесь не будет. Для понимания синтаксиса SQL достаточно примитивной базы данных из трех таблиц. Это упрощение снизит когнитивную нагрузку и позволит сосредоточиться на синтаксисе языка, а проектированием сложных предметных областей можно будет заняться позже, когда SQL не будет вызывать затруднений.

В книге предлагаются для скачивания две базы данных shop и project_management. На примере базы данных shop изложен весь учебный материал, а для выполнения самостоятельных работ понадобится база project_management. Ссылки для скачивания будут даны далее в тексте. Рекомендую обязательно скачать обе базы данных и проделывать все действия сразу параллельно с чтением. Для самопроверки самостоятельных работ все задания снабжены скриншотами результатов, с которыми нужно сверять свои результаты. SQL – это прежде всего практика. Чтобы научиться делать запросы нужно их делать.

Создание баз данных

Начнем с минимального набора рабочих инструментов, которые нам понадобятся. Их всего два: система управления базами данных (СУБД) MariaDB и менеджер баз данных с удобным интерфейсом HeidiSQL.

Что такое менеджер баз данных?

Изначально предполагалось, что работа с СУБД будет происходить из командной строки и никакой отдельный интерфейс здесь не нужен. Да, вот этот самый черный экран и заклинания на магическом языке SQL. Но потом умные люди додумались сделать удобный интерфейс, чтобы часть заклинаний можно было выполнять интерактивно при помощи кнопок и мыши. Таких интерфейсов к одной и той же СУБД существует много. Они называются *менеджерами баз данных* и устанавливаются отдельно. Мы будем пользоваться менеджером HeidiSQL.

СУБД MariaDB можно скачать по ссылке https://mariadb.org/download/. Менеджер HeidiSQL входит в инсталляционный пакет MariaDB, поэтому качать его отдельно не нужно.

Процесс установки стандартный. Список основных инструментов менять мы не будем (рис.1.1).

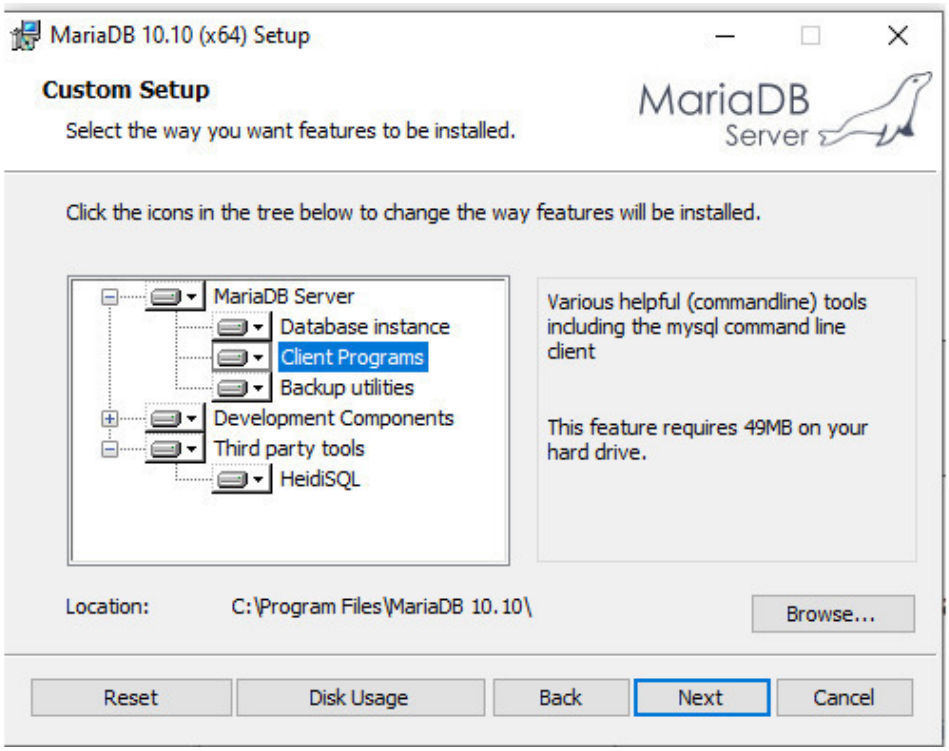

Рис.1.1 – Установка MariaDB

Можно отменить пароль, сняв соответствующий флажок (рис.1.2).

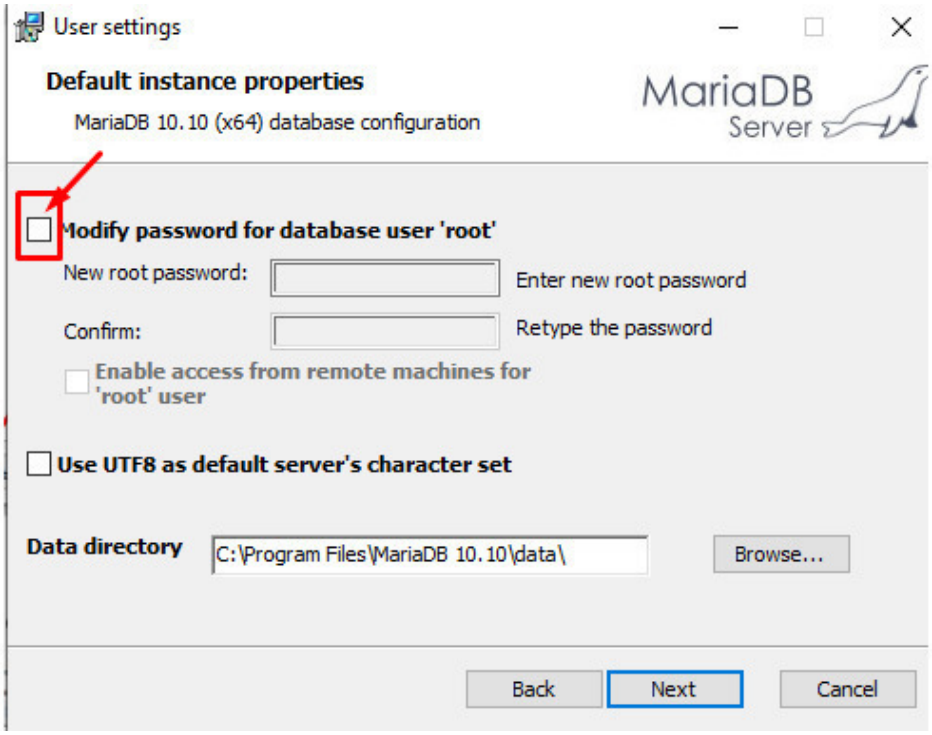

Рис.1.2 – Отмена использования пароля

Далее нажимаем Install и ждем окончания установки.

Если все действия были выполнены корректно, то MariaDB появится в списке служб.

Службы – это программы, которые не имеют интерфейса и работают в фоновом режиме.

Многие службы запускаются автоматически при загрузке системы. Список служб можно посмотреть через следующую последовательность команд:

Панель управления-> Администрирование-> Службы.

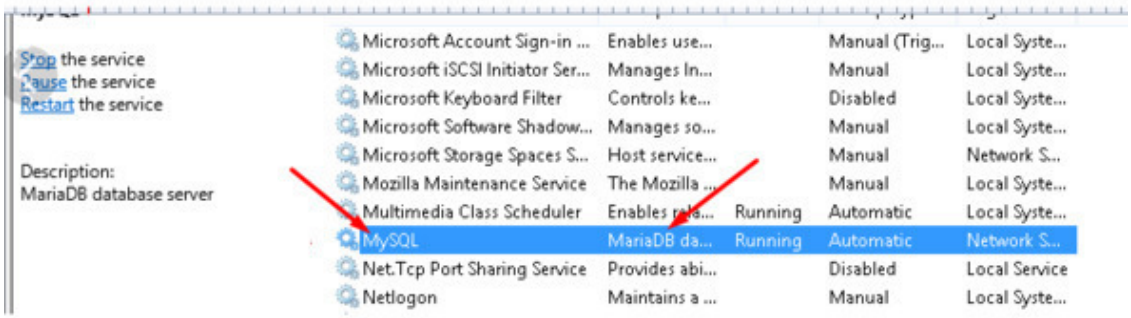

Рис.1.3 – MariaDB в списке служб

Маленькая неожиданность. В описании службы вы найдете MariaDB, но в названии может стоять MySQL. Это нормально. MySQL и MariaDB были некогда одной и той же СУБД, которая впоследствии разделилась на два форка. Историю об этом можно почитать в Википедии, а мы продолжим погружение в практику.

Запустим менеджер HeidiSQL (рис.1.4).

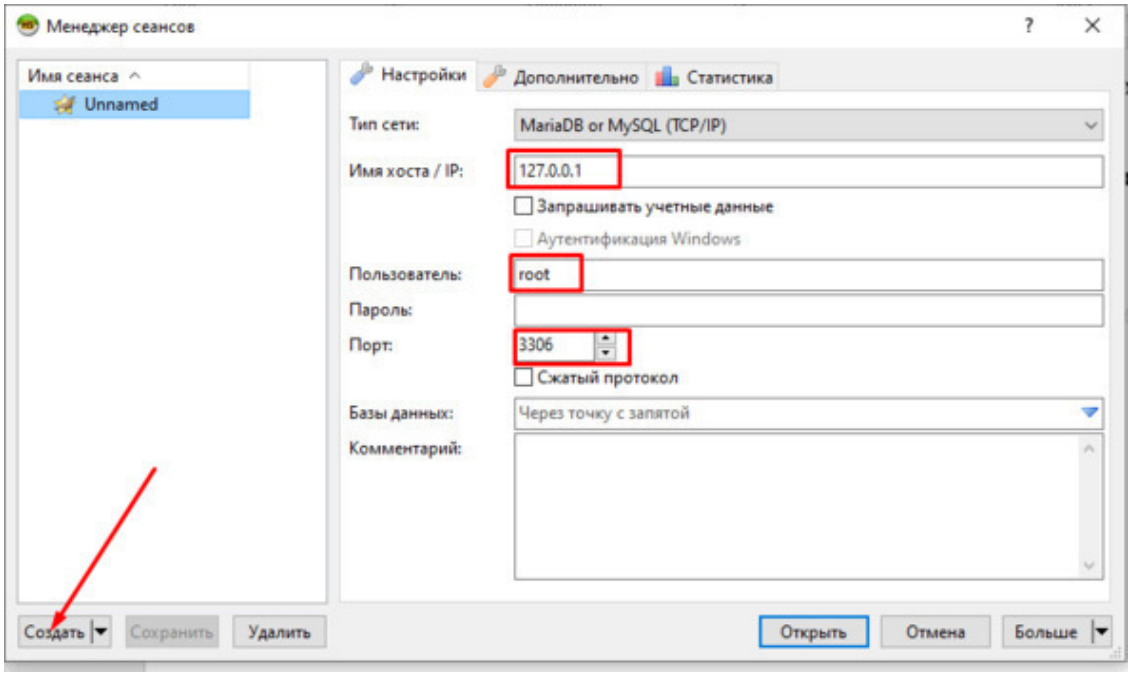

Рис.1.4 – Запуск HeidiSQL

Если MariaDB уже работает службой, то при запуске HeidiSQL достаточно проконтролировать, что поле «Имя хоста» содержит адрес 127.0.0.1. Это зарезервированный IP, который указывает на то, что сервером баз данных является тот компьютер, за которым вы сидите. Этот IP нужно запомнить наизусть. Еще к своему компьютеру можно обратиться через слово «localhost», написав его вместо 127.0.0.1. Но если бы MariaDB была установлена на другом компьютере, то в это поле пришлось бы вводить IP этого компьютера.

В поле «Пользователь» находится слово «root» – это стандартное название для администратора MariaDB с самым широким набором прав.

В поле «Пароль» должен быть пароль или ничего, если пароль не задан.

MariaDB стандартно работает через порт 3306, и тут ничего не нужно менять.

Далее нужно создать сеанс нажатием на кнопку «Создать». Сеанс получит по умолчанию название Unnamed. Его можно сразу же переименовать через контекстное меню, но можно оставить безымянным.

В левой части интерфейса расположено дерево сеанса (рис.1.5).

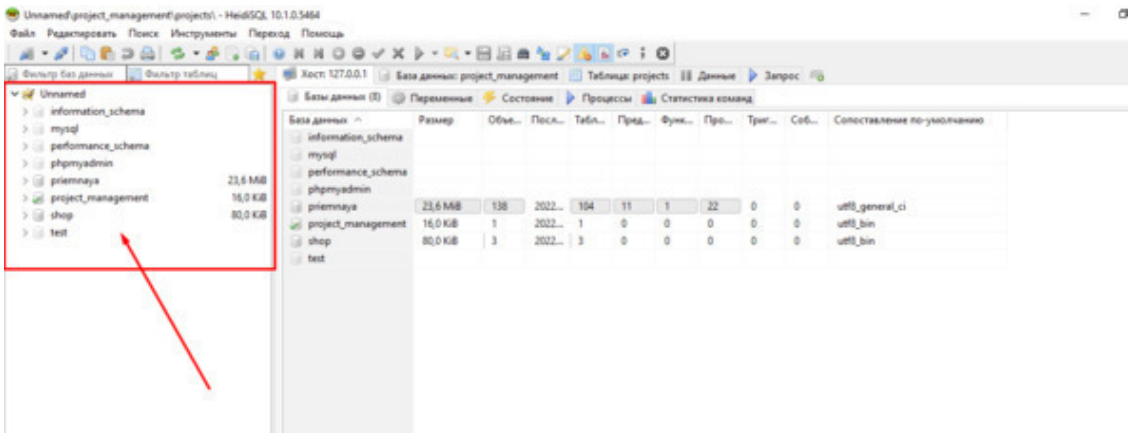

Рис.1.5 – Список баз данных

Дерево показывает сразу весь список баз данных, которые работают под управлением MariaDB в настоящий момент. Даже, если СУБД только что установлена, дерево не будет пустым. В нем находятся базы данных, которые используются для работы самой системы.

Узел базы данных раскрывается щелчком мыши, после чего становятся видны таблицы, из которых база данных состоит (рис.1.6).

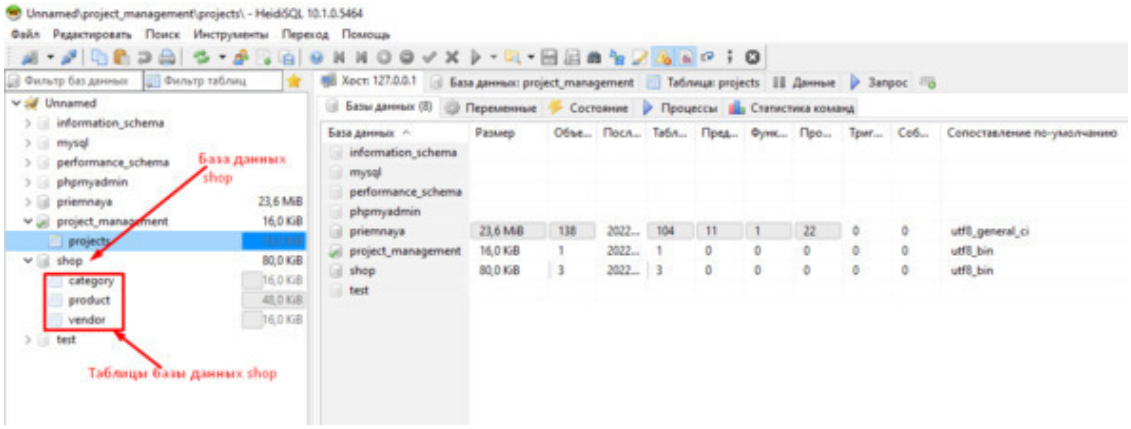

Рис.1.6 – Таблицы базы данных

Как создать базу данных?

Создадим базу данных нажатием правой кнопки мыши на названии сеанса «Unnamed» (рис.1.7).

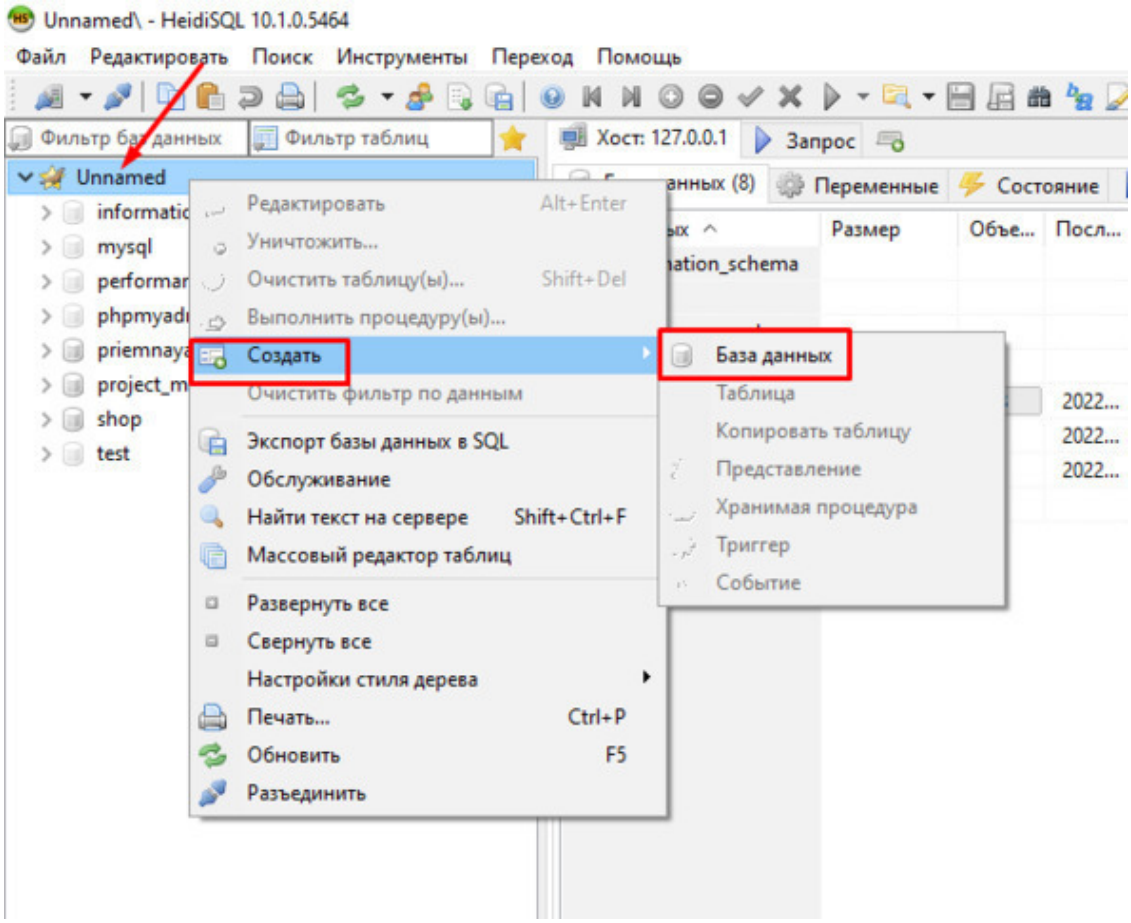

Рис.1.7 – Создание базы данных

Что такое кодировка и сопоставление

При создании базы данных необходимо задать ее имя. Пусть это будет «project_management». Также нужно проверить поле «Сопоставление» (рис.1.8).

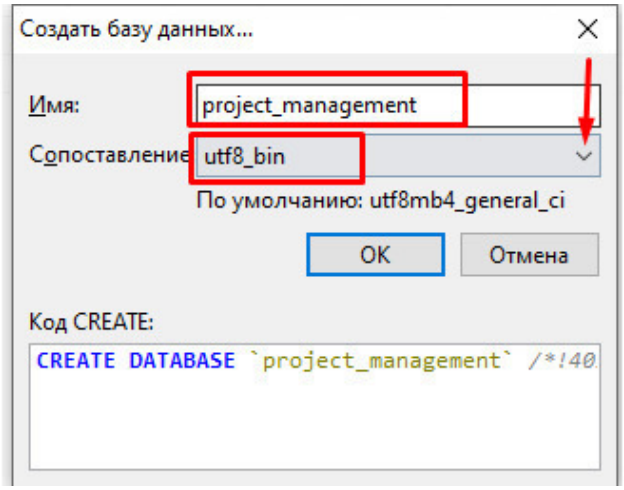

Рис.1.8 -Имя и сопоставление

Для того чтобы можно было работать с данными на русском языке, нужно использовать кодировку UTF8. Кодировка сама по себе отвечает за набор символов. А вот правила сравнения этих символов задаются сопоставлением.

Что такое правила сравнения? Ну, например, при одном и том же наборе символов можно различать большие и маленькие буквы, а можно не различать. Это уже будут разные правила сопоставления. Есть и другие более тонкие отличия в этих правилах.

Нам нужно указать сопоставление utf8_bin. Есть еще ряд сопоставлений, которые можно использовать, но остановимся на utf8_bin.

Обратите внимание на нижнюю часть окна. Мы вписали имя и сопоставление, а в нижней части сформировалась команда

CREATE DATABASE `project_manager`

Это и есть обещанный интерактив. Если бы мы не использовали менеджер HeidiSQL, то команду пришлось бы набирать руками в командной строке.

Жмем OK и обязательно обновляем сеанс. Для этого нужно выделить левой кнопкой мыши сеанс и нажать либо F5, либо кнопку на панели инструментов (рис.1.9).

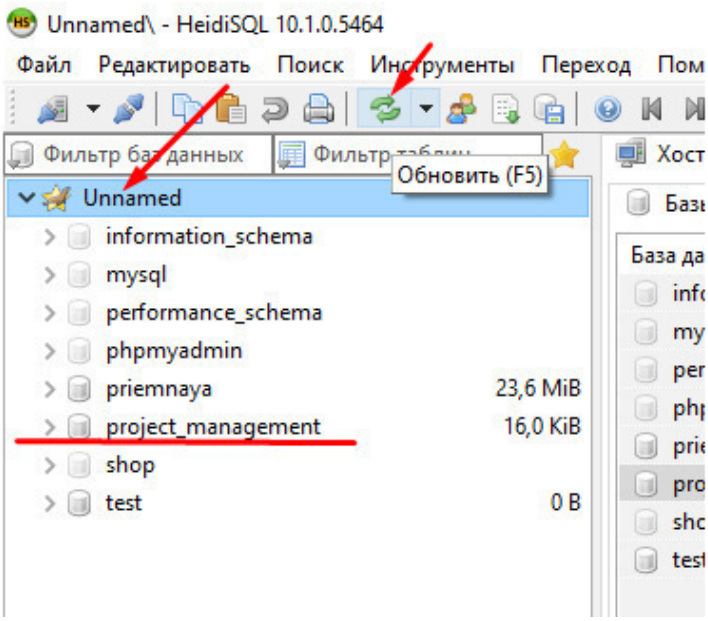

Рис.1.9 – База данных в дереве сеанса

После этого база данных project_manager отобразится в дереве сеанса.

Кстати, проверим сопоставление. Перейдем в правой части интерфейса на вкладку «Хост» и увидим, что напротив нашей базы данных установлено сопоставление utf8_bin (рис.1.10). Если в какой-то базе данных некорректно отображаются русские буквы, то именно здесь нужно проверять сопоставление. Например, сопоставление latin1_swedish_ci, которое мы видим у базы данных «test», не отображает кириллицу вообще.

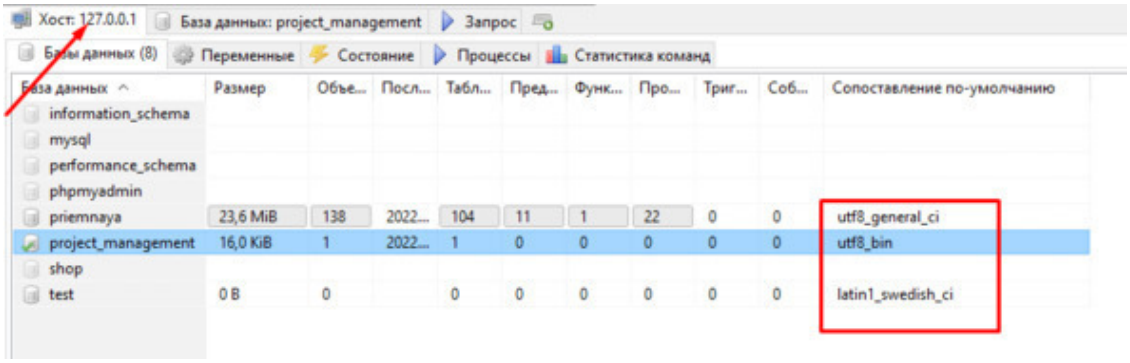

Рис.1.10 – Базы данных и их соспоставление

Изменить сопоставление можно через команду «Редактировать» в контекстном меню нужной базы данных (рис.1.11).

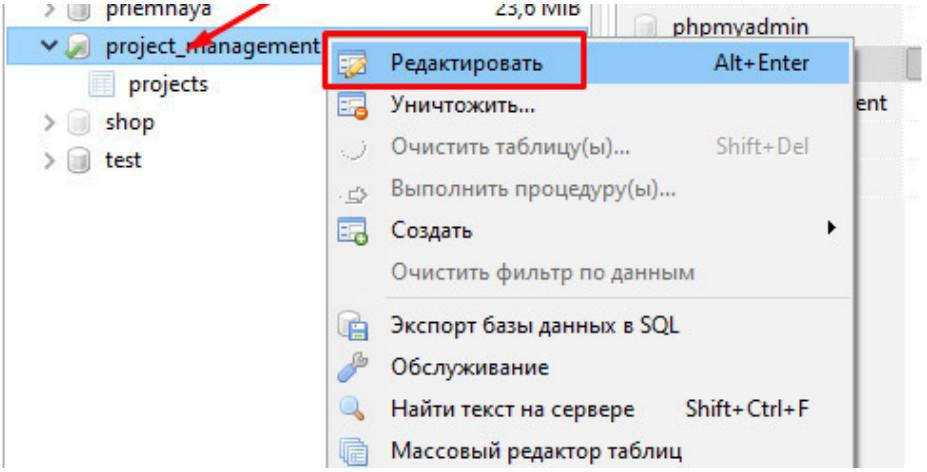

Рис.1.11 – Редактирование базы

Как создать таблицу?

Для создания таблицы выделяем нужную базу данных и в контекстном меню правой кнопки мыши выбираем команды «Создать» и «Таблица» (рис.1.12)

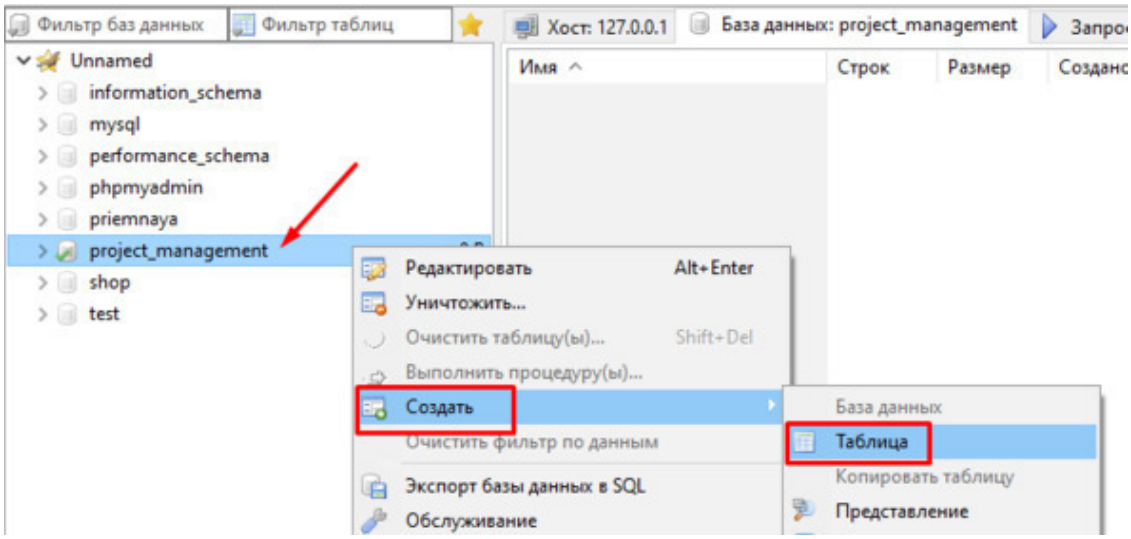

Рис.1.12 – Создание таблицы

Прежде всего таблице нужно задать имя. В нашем случае это будет «projects» (рис.1.13).

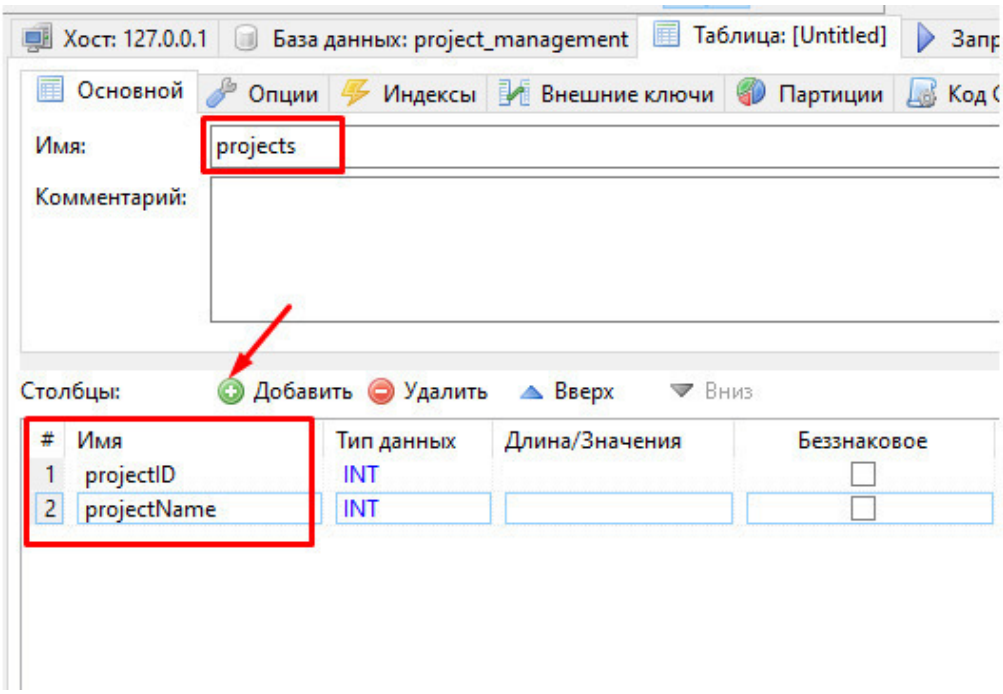

Рис.1.13 – Создание таблицы

Далее нажатием на кнопку «Добавить» добавляем в таблицу поля. У нас их будет два: projectID и projectName (рис.1.13).

Поле projectID будет иметь тип INT. То есть оно может содержать целые числа. С помощью контекстного меню для поля projectID зададим свойство «Primary», как показано на рис.1.14. Это означает, что данное поле будет первичным ключом таблицы.

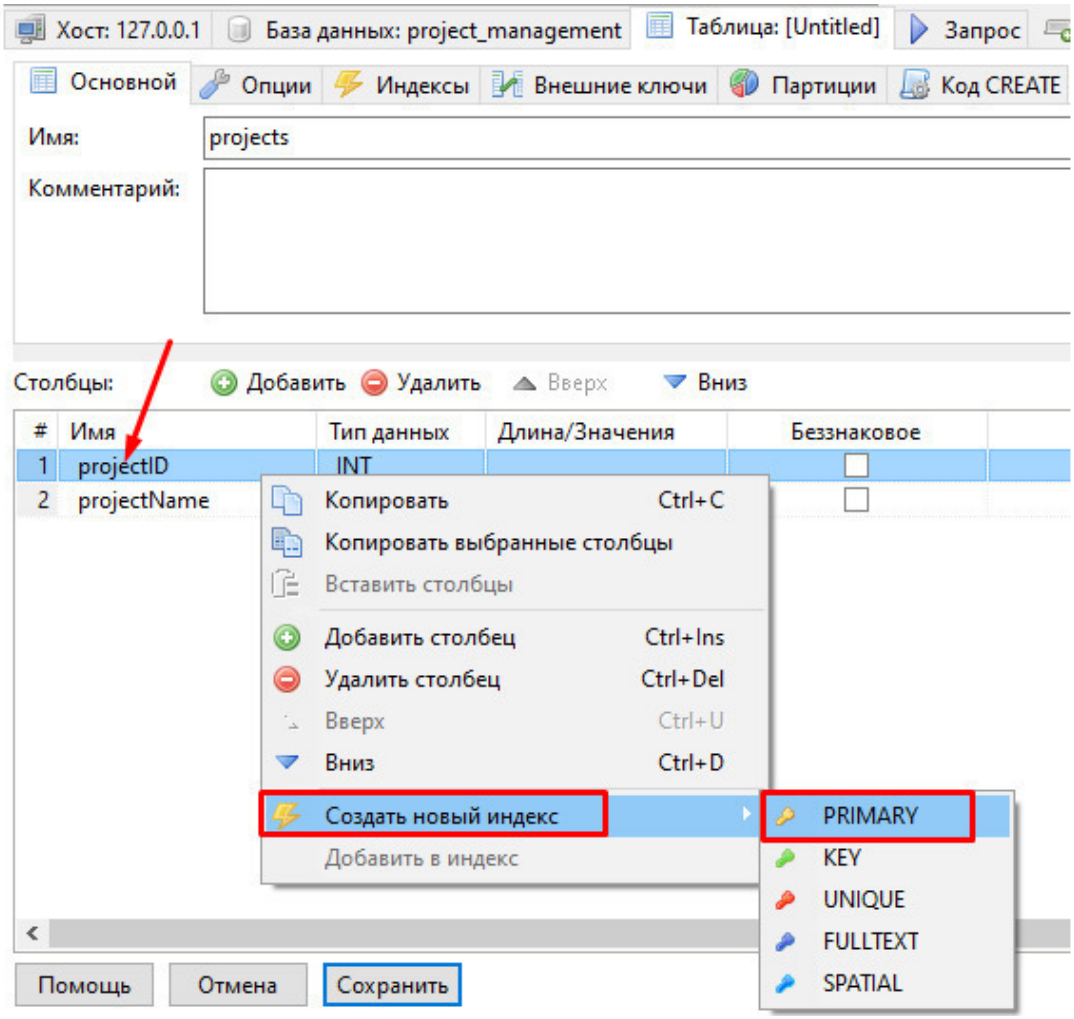

Рис.1.14 – Задание первичного ключа

Зададим этому полю также свойство «Беззнаковое» и свойство AUTO_INCREMENT (рис.1.15).

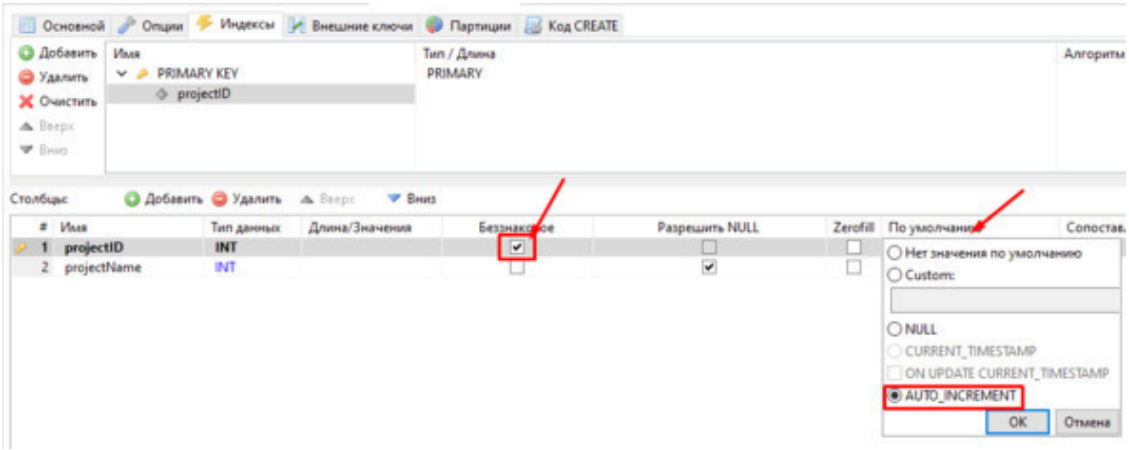

Рис.1.15 – Свойства первичного ключа

Свойство «Беззнаковое» нужно, чтобы в нумерации были только положительные числа, а автоинкремент позволит автоматически увеличивать значение поля на 1 при добавлении новой записи.

Для поля projectName зададим тип данных varchar с длиной значения 50 (рис.1.16). Этот тип данных подходит для текстовых строк. Строка типа varchar (50) будет занимать память в соответствии со своей реальной длиной, но эта длина не может превышать 50 символов.

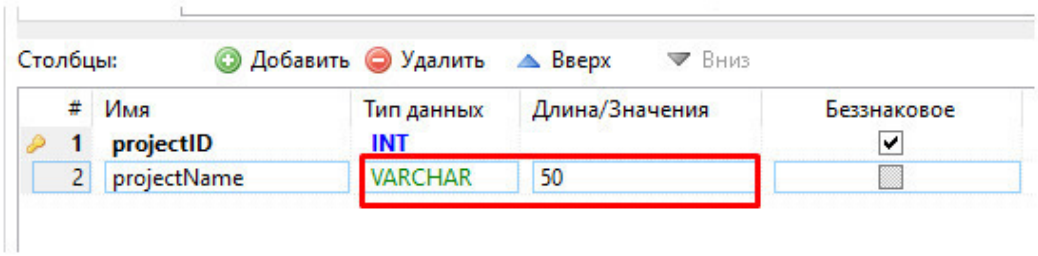

Рис.1.16 – Поле с типом varchar (50)

У MariaDB широкий спектр различных типов данных. Вы их можете видеть в выпадающем списке (рис.1.17). Изучать некоторые из их мы будем по мере использования. Получить информацию о каждом типе можно сразу во всплывающей подсказке.

На рис.1.17 подсказка сообщает нам, что тип SMALLINT может содержать либо числа от -32768 до +32767, либо, если мы объявим поле беззнаковым и тем самым отсечем отрицательные числа, от 0 до 65535.

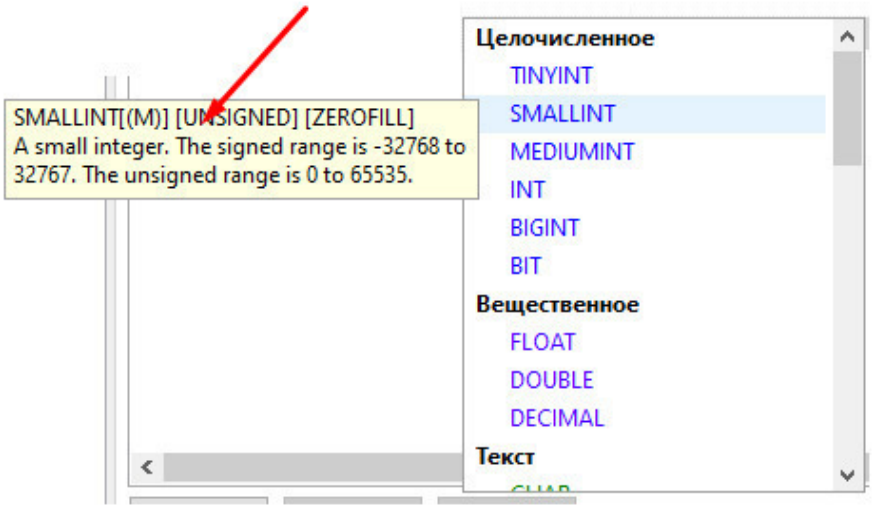

Рис.1.17 – Типы данных

Кстати, обратите внимание, что для строковых типов автоматически указывается сопоставление. И оно именно такое, как мы выбрали при создании базы данных – utf8_bin (рис.1.18).

Иногда бывает нужно задать для отдельного поля сопоставление, отличное от того, которое задано на уровне всей базы. Это можно сделать через свойство «Сопоставление» (рис.1.18) путем выбора из выпадающего списка.

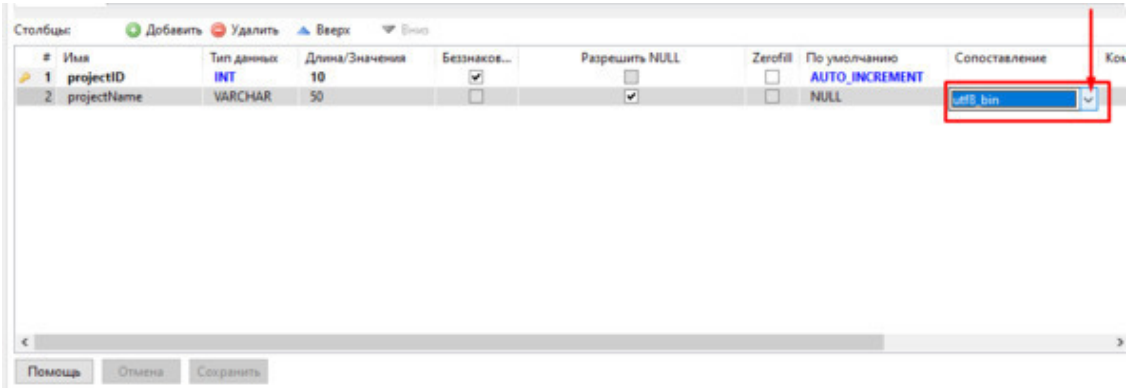

Рис.1.18 -Сопоставление отдельного поля

Жмем «Сохранить». Таблица создана.

Теперь перенесем взгляд в нижнее окно интерфейса (рис.1.19). Там все время что-то происходило, пока мы работали! Потому что там наши действия дублируются SQL-кодом.

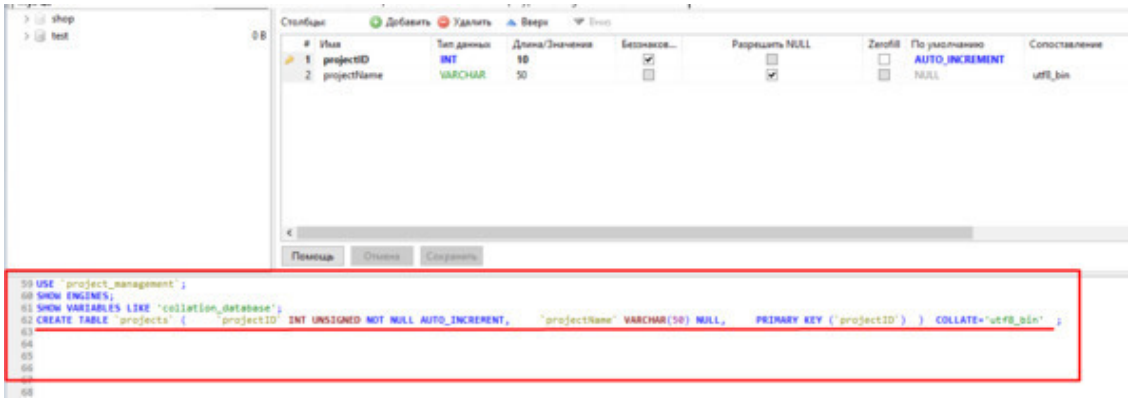

Рис.1.19 – SQL-код

Например, когда мы нажали «Сохранить», то появился код

```
CREATE TABLE `projects` (
`projectID` INT UNSIGNED NOT NULL AUTO_INCREMENT,
`projectName` VARCHAR (50) NULL,
PRIMARY KEY (`projectID`)
)
COLLATE='utf8_bin';
```

Этот же код можно увидеть теперь на вкладке «Код CREATE» (рис.1.20).

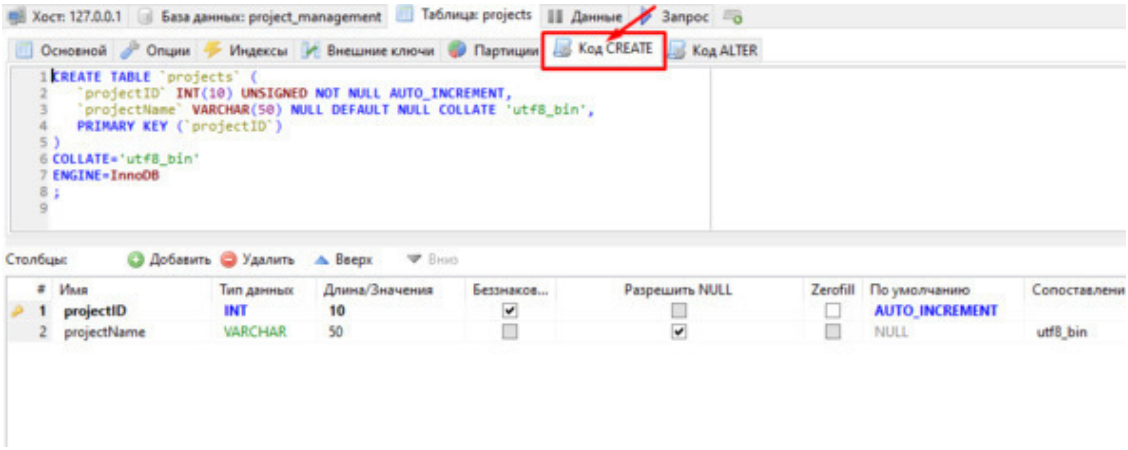

Рис.1.20 – Код CREATE

Это инструкция «CREATE TABLE», которая создает таблицу. В доисторические времена, когда не было менеджеров баз данных, нужно было вписывать весь этот SQL-код руками в командную строку. И, естественно, синтаксис инструкции CREATE требовалось знать наизусть.

Сейчас синтаксис этой инструкции нужен тем, кто глубоко изучает SQL. Для начинающих достаточно знать, что такая инструкция существует.

Добавление данных в таблицу

После того, как таблица сохранилась, переключимся на вкладку «Данные» (рис.1.21).

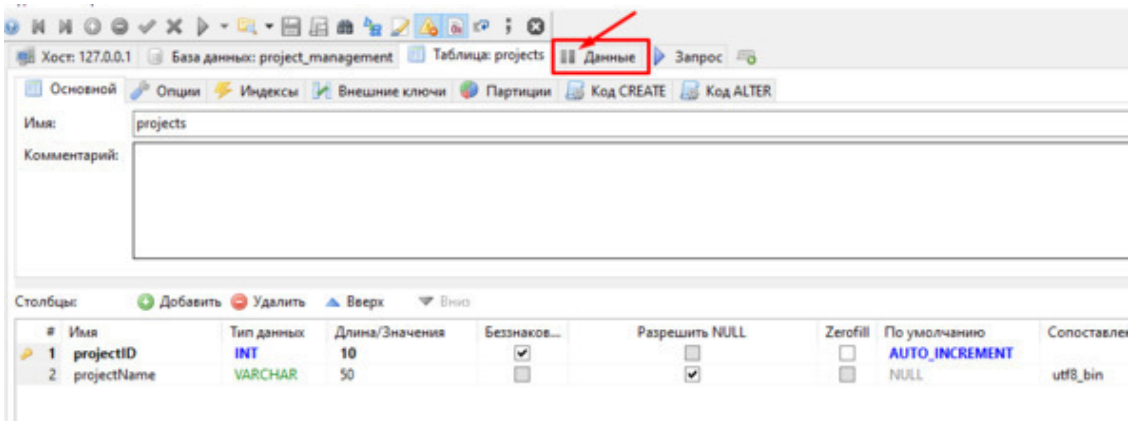

Рис. 1.21 – Вкладка «Данные»

Здесь новую строку в таблицу можно добавить либо кнопкой, либо клавишей Insert (рис.1.22).

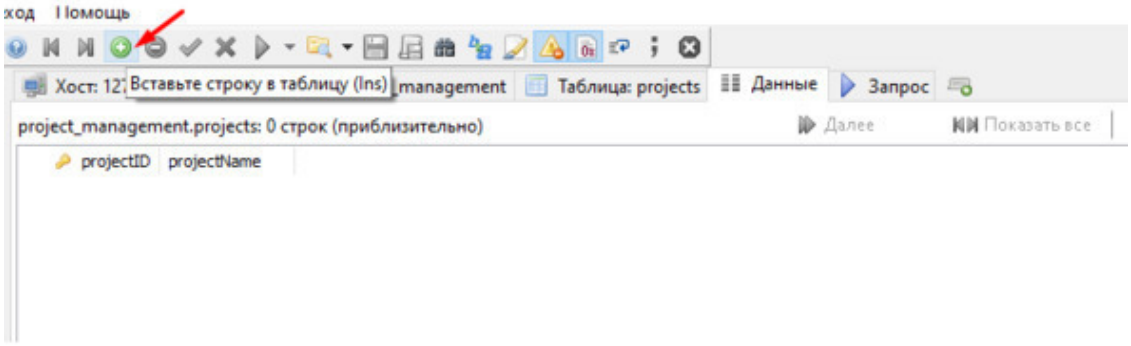

Рис.1.22 – Добавление данных

Пусть в нашей таблице хранится список проектов. Добавим следующие записи (рис.1.23).

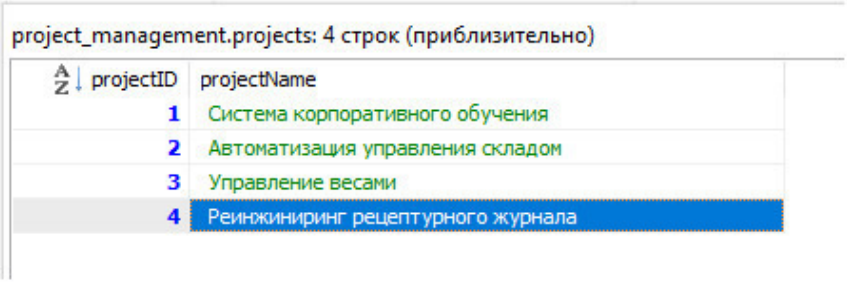

Рис.1.23 – Добавление данных

Удалить выбранную строку можно кнопкой (рис.1.24).

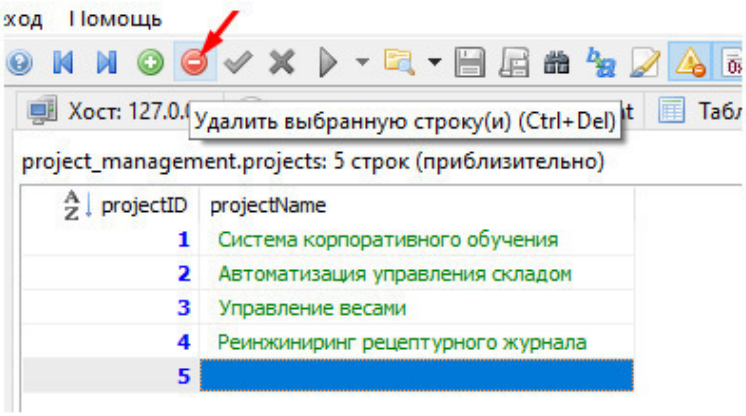

Рис.1.24 – Удаление строки

Самостоятельная работа №1

– Создайте в базе данных project_manager таблицу employee (сотрудники) со следующими характеристиками (рис.1.25):

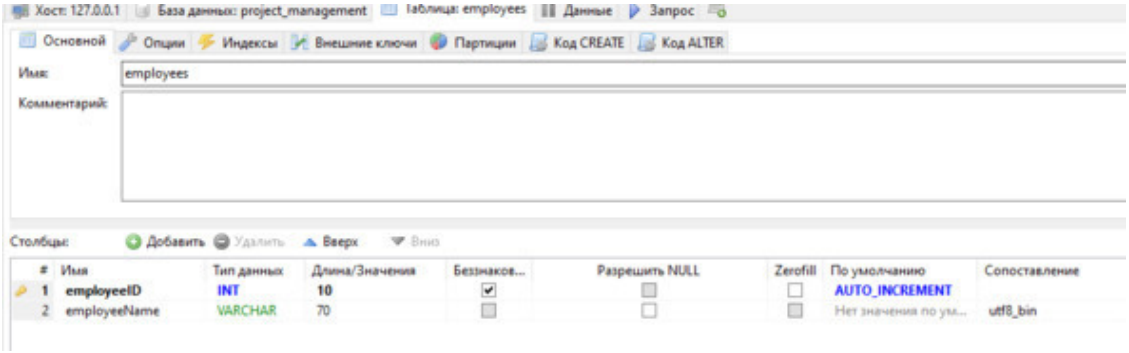

Рис.1.25 – Поля таблицы employees

– Заполните созданную таблицу данными (рис.1.26)

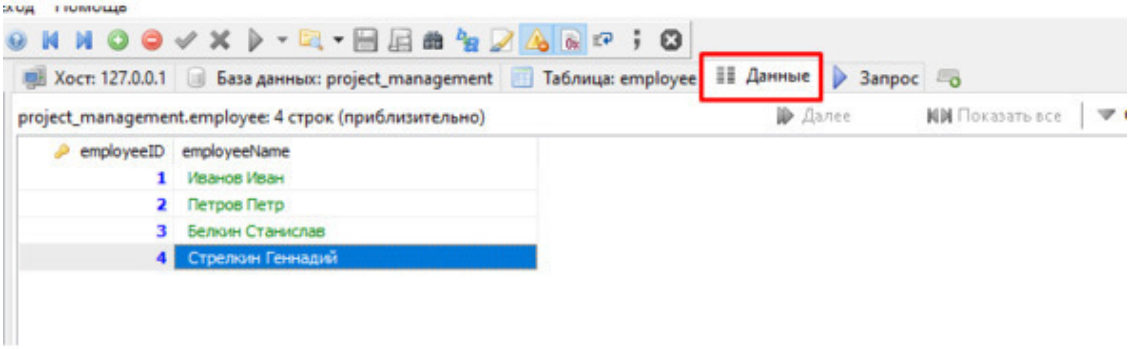

Рис.1.26 – Данные таблицы

Внешние ключи

Теперь у нас есть таблица «projects», которая содержит список проектов, и таблица «employees», которая содержит список сотрудников. Требуется распределить сотрудников по проектам с условием, что один сотрудник может работать в нескольких проектах.

Это означает, что между сущностями «проекты» и «сотрудники» имеется связь типа «много-ко-многим». (Если не очень понятно о чем идет речь, то более подробно об определении вида связи и о построении связей можно прочитать в книге «Как научиться проектировать базы данных и остаться в живых».)

В данном случае требуется создать ассоциирующую таблицу, которая будет содержать первичные ключи двух исходных таблиц в качестве внешних. Создадим таблицу «job» с первичным ключом jobID и полем employeeID (рис.1.27).

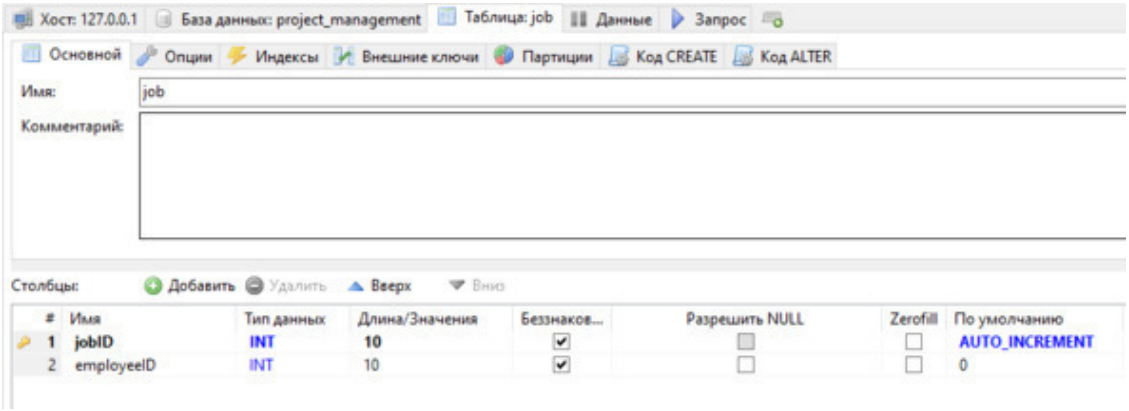

Рис.1.27 – Таблица job

Теперь сделаем из поля employeeID внешний ключ. Для этого перейдем на вкладку «Внешние ключи» и добавим новый внешний ключ (рис.1.28).

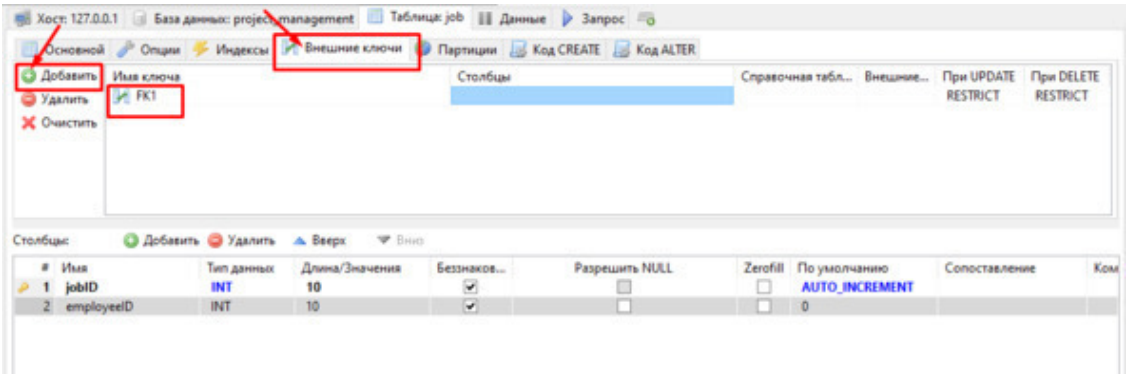

Рис.1.28 – Внешний ключ

Далее нужно показать на какую таблицу внешний ключ ссылается. Для этого нужно заполнить три свойства внешнего ключа (рис.1.29).

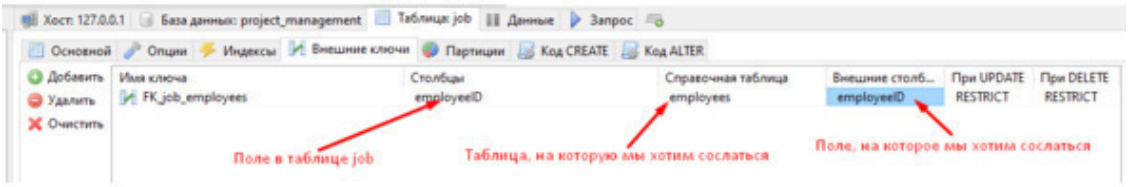

Рис.1.29 – Формирование внешнго ключа

Здесь мы указываем поле таблицы «job», которое будет внешним ключом (employeeID), потом указываем на какую таблицу оно ссылается («employees») и, наконец, на какое конкретно поле таблицы «employees» оно ссылается (employeeID). Внешний ключ сформирован.

Самостоятельная работа №2

1. Добавьте в таблицу «job» еще один внешний ключ – projectID, который ссылается на projectID в таблице «projects».

2. Добавьте еще два поля.

dateBegin – дата начала работы в проекте с типом DATE;

dateEnd – дата окончания работы в проекте с типом DATE.

Для обоих полей допускается значение NULL.

3. Добавьте поле payment – оплата за участие в проекте. Тип данных – MEDIUMINT, NULL не допускается, значение по умолчанию -0 (рис.1.30).

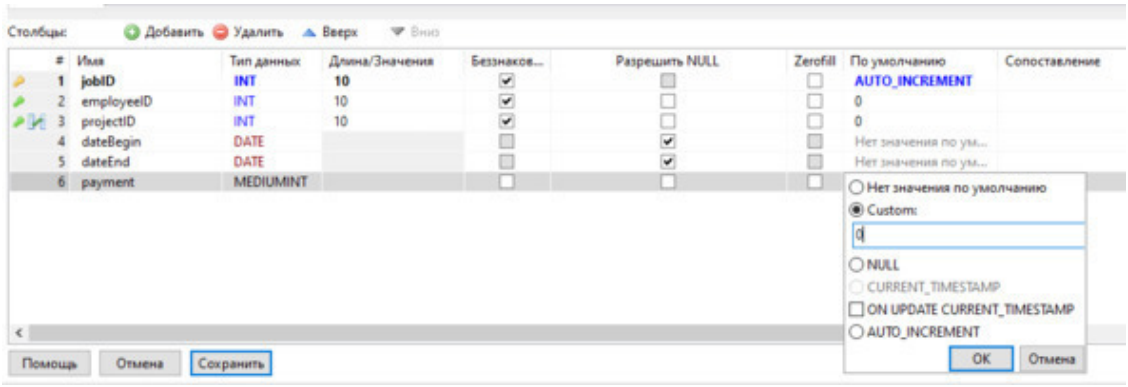

Рис.1.30 – Окончательная структура таблицы «job»

Ограничения целостности

Заполним базу данных в соответствии с рис. 1.31.

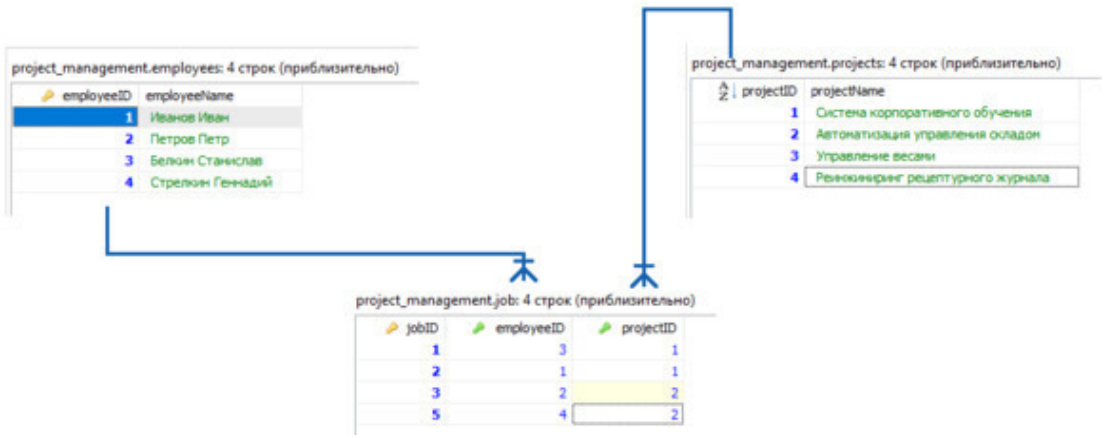

Рис.1.31 – Заполненные таблицы

Говоря простым языком, целостность данных, это, когда проекту с определенным кодом в таблице «job» находится соответствие в таблице «projects», а сотруднику в таблице «job» находится соответствие в таблице «employees».

Если бы в таблице «job» значился сотрудник с кодом 5, которого нет в таблице «employees», то это было бы нарушением целостности данных.

Поскольку базу заполняют живые люди, гарантировать отсутствие ошибок такого рода невозможно. Это то, что называют человеческим фактором. Поэтому желательно максимально защитить себя от человеческого фактора техническими средствами. И такая возможность есть.

При попытке вставить в поле projectID таблицы «job» несуществующий код проекта, СУБД покажет диагностическое сообщение об ошибке (рис.1.32).

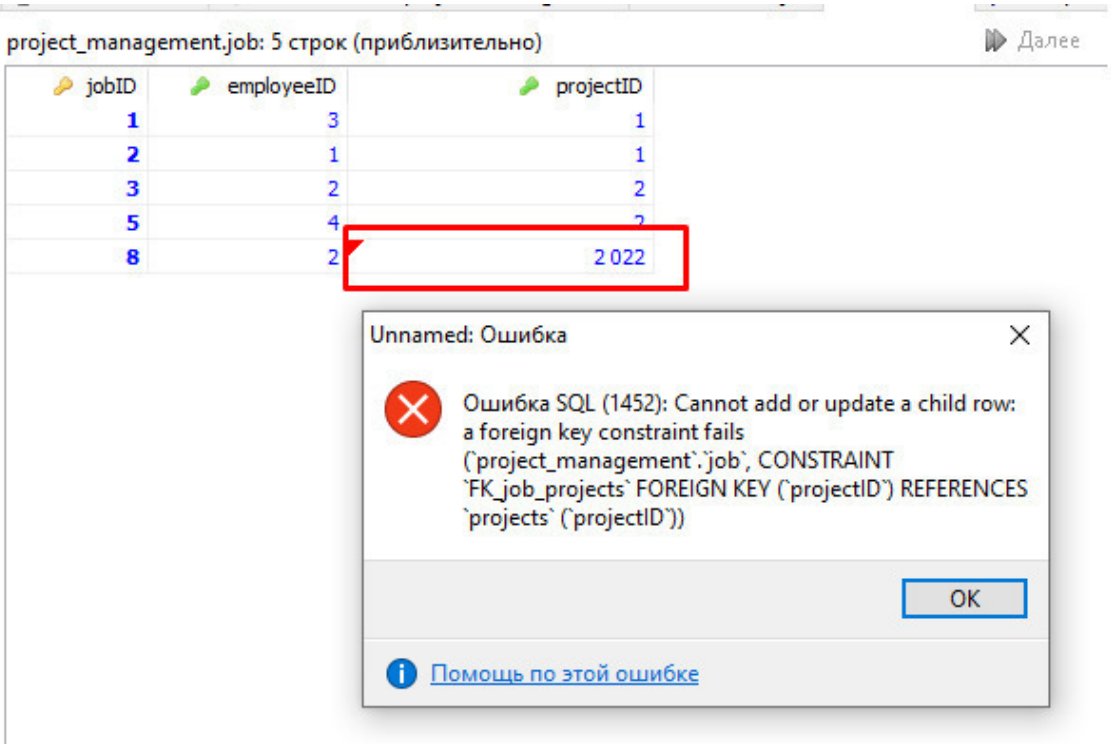

Рис.1.32 – Диагностическое сообщение о нарушении целостности данных

Вообще все диагностические сообщения обязательно нужно читать, переводить и понимать, а не закрывать и пытаться угадать, в чем проблема. Данное сообщение говорит нам, что «запись невозможно добавить или отредактировать, так как нарушается ограничение внешнего ключа». Конечно, именно такую ошибку мы тут допустили умышленно: попытались вставить проект с кодом 2022, хотя существуют только проекты с кодами 1, 2, 3, 4.

Также возможна обратная ситуация. Сотрудника с кодом 1 (Иванова Ивана) решили по каким-то причинам удалить из системы. Но сотрудник уже записан как участник проекта с кодом 1. Как поступить с записью в таблице «job», которая связана с этим сотрудником? Для ответа на этот вопрос есть свойство «ПриDELETE» в настройке внешнего ключа (рис.1.33).

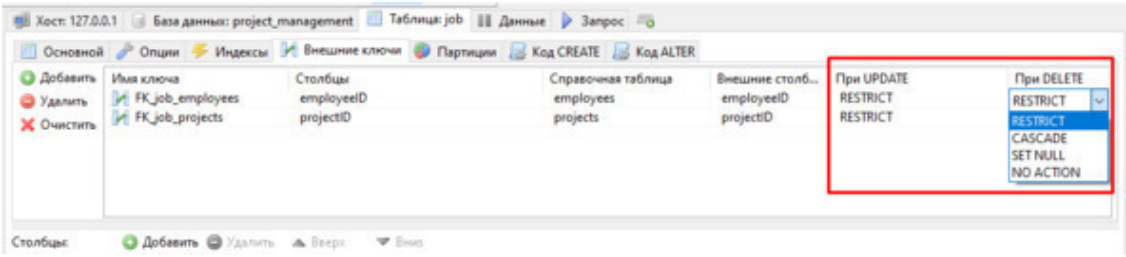

Рис.1.33 – Ограничения при удалении записи

При удалении сотрудника возможны следующие действия со связанной записью:

RESTRICT – ОГРАНИЧИТЬ. В этом случае при попытке удаления сотрудника из таблицы employees появится вот такое диагностическое сообщение:

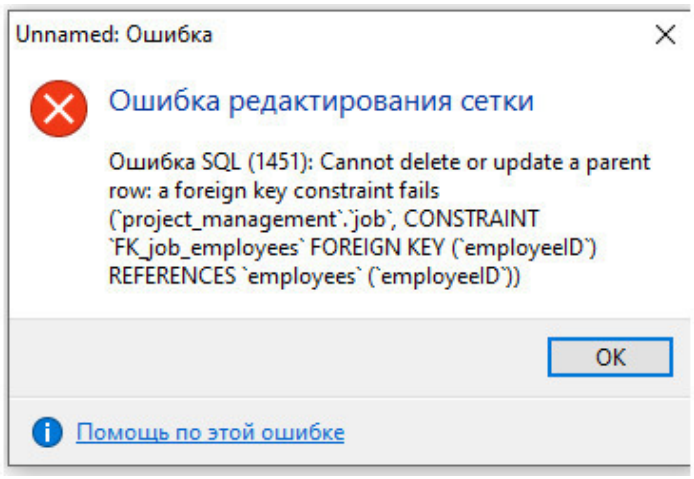

Рис.1.34 – Ограничение RESTRICT

Сообщение говорит нам, что «невозможно удалить или изменить родительскую запись, так как нарушаются ограничения внешнего ключа». Запись с кодом 1 из таблицы «employees» является родительской по отношению к записи в таблице «job». Данное сообщение просто предупреждает нас о том, что удаление выполнить невозможно. Мы можем принять одно из двух решений:

– Сначала удалить связанную дочернюю запись в таблице «job», а уже потом выполнить удаление в таблице «employee». Тогда удаление выполнится.

– После диагностического сообщения понять, что попытка удаления была ошибочной и запись вообще не нужно удалять.

CASCADE – Каскадное удаление. В этом случае при удалении сотрудника с кодом 1 удалятся автоматически все связанные записи из таблицы «job». В некоторых случаях это удобно. Но метод довольно опасный, так как данные «посыплются», как тетрис, и вернуть их вряд ли удастся, если окажется, что удаление было ошибочным.

SET NULL Установить значение NULL. NULL – это «пустое множество», отсутствие какого-либо значения. В этом случае при удалении сотрудника из таблицы «employee», запись в таблице «job» примет следующий вид (рис.1.35):

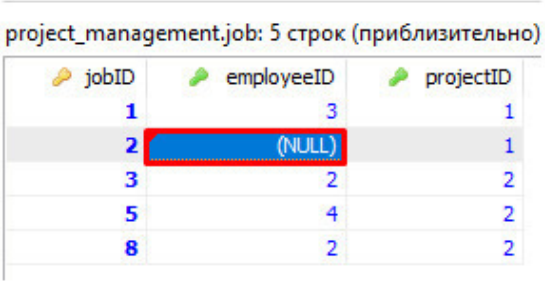

Рис.1.35 – Настройка SET NULL

NO ACTION – Никаких действий. В это случае после удаления записи из таблицы «employees», в таблице «job» ничего не произойдет. Поле employeeID так и будет содержать код 1, который уже не существует в родительской таблице.

В разных задачах могут понадобиться разные настройки, но совершенно очевидно, что наиболее строго целостность данных сохраняет именно настройка RESTRICT. Все остальные настройки провоцируют ситуации хаоса и несоответствия данных. Именно поэтому настройка RESTRICT обычно установлена как выбор по умолчанию.

Такие же настройки могут быть применены при попытке изменить значение первичного ключа родительской записи. Если мы в таблице «employees» захотим изменить значение кода для сотрудника Иванова с 1 на 5, то точно также для связанных записей есть четыре варианта: можно защитить их ограничением RESTRICT, каскадно изменить, установить NULL или ничего не изменять (рис.1.36).

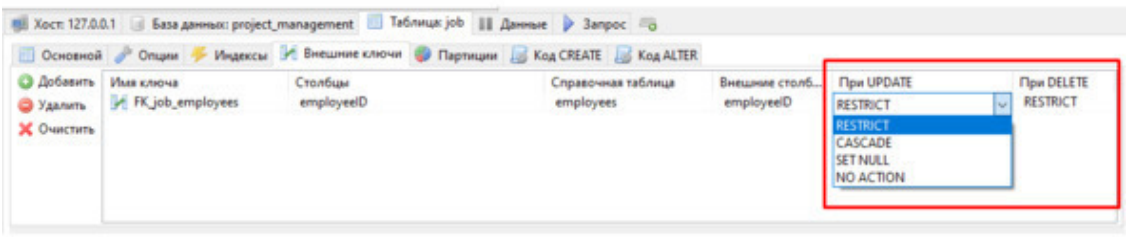

Рис.1.36 – Ограничения целостности данных по внешнему ключу при изменении

Создание резервной копии (дампа базы данных).

Резервная копия (дамп) базы данных создается для повышения надежности хранения данных, а также для переноса базы на другой сервер. Резервная копия представляет собой файл с расширением *.sql, который содержит код создания таблиц и код добавления данных в таблицы. Этот код можно выполнить на другом сервере, тогда по нему будет воссоздана база и данные в ней.

Для создания резервной копии нужно нажать кнопку «Дамп объектов базы данных в файл SQL» (рис.1.37).

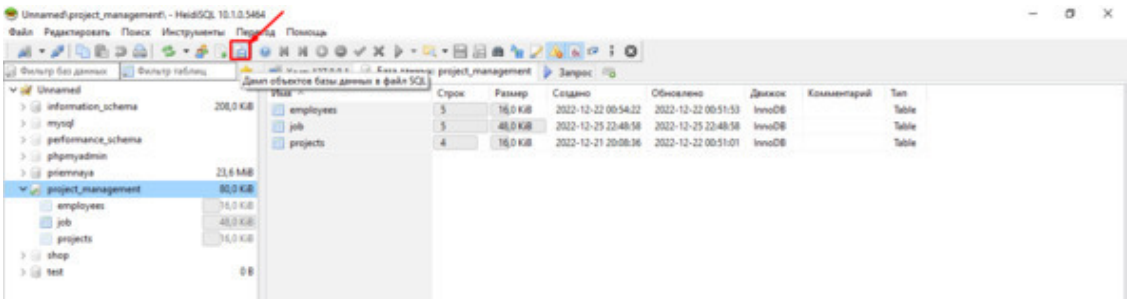

Рис.1.37 – Создание резервной копии базы данных

В открывшемся окне указать нужную базу данных (рис.1.38).

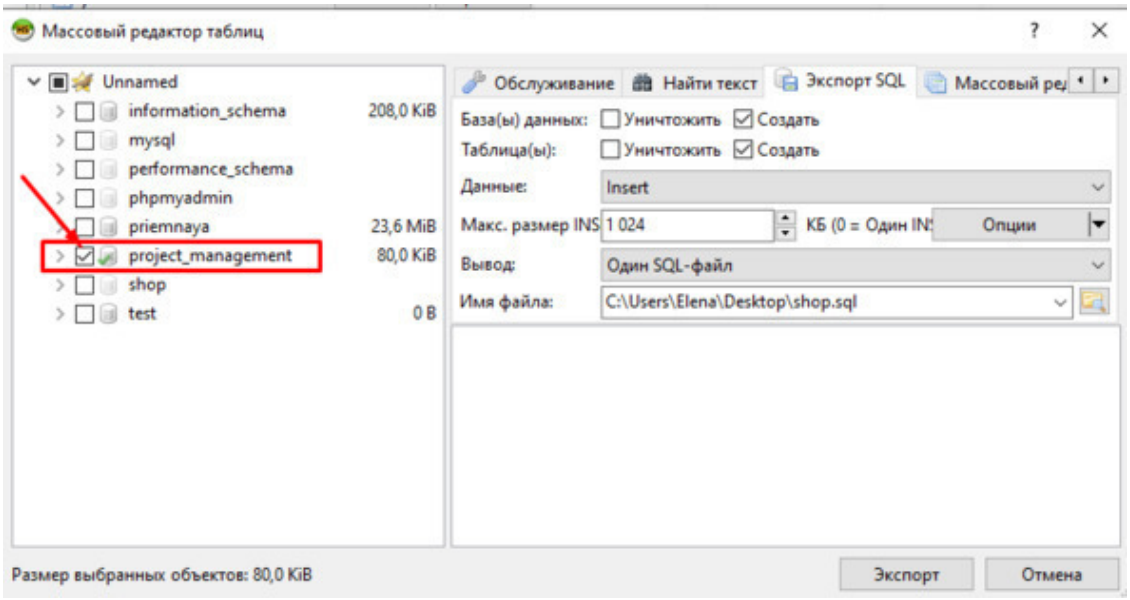

Рис.1.38 – Выбор базы данных

Далее требуется настроить нужные опции. Чаще всего нужно сохранить в резервную копию код создания самой базы, код создания таблиц и код добавления данных. Для этого нужно проконтролировать чтобы были выбраны опции, которые отмечены на рис.1.39.

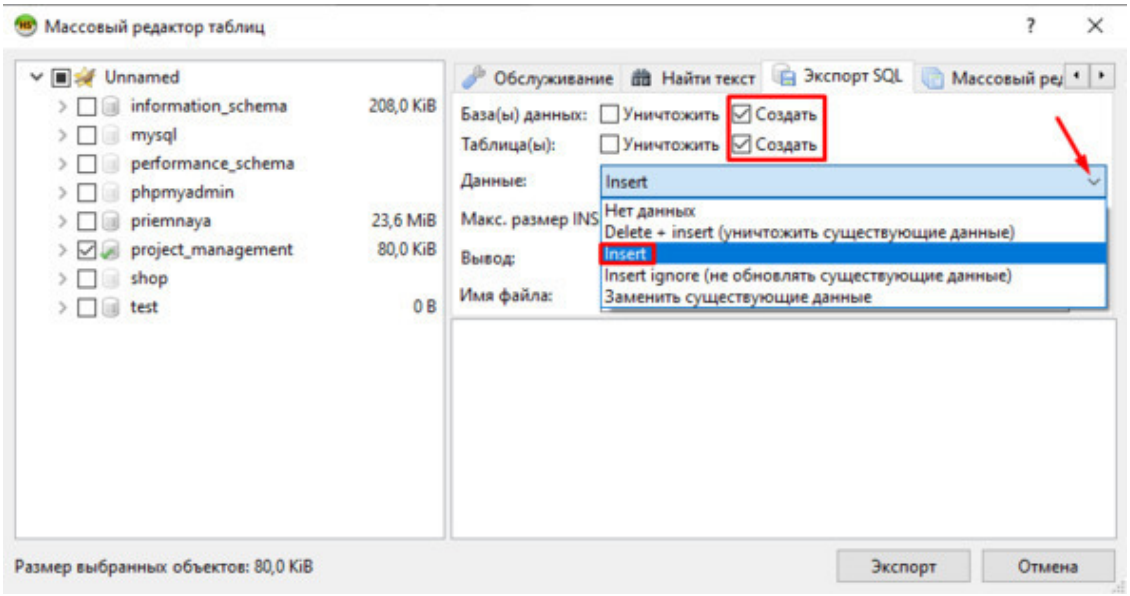

Рис.1.39 – Выбор опций

Бывают ситуации, когда нужна только структура базы с пустыми таблицами. В этом случае настройки будут такими, как на рис.1.40.

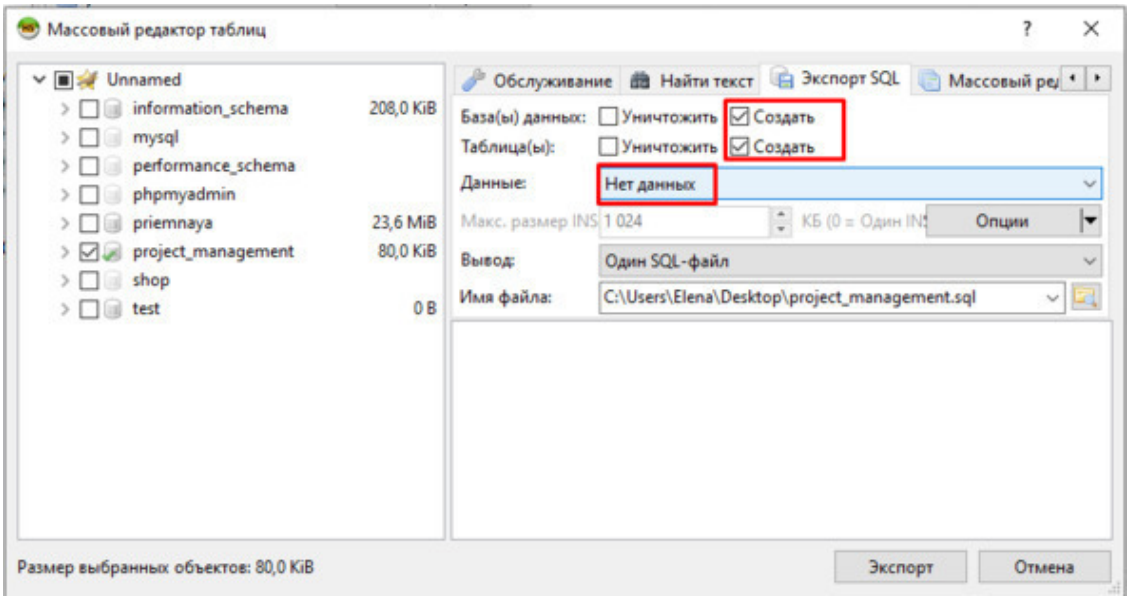

Рис.1.40 – Настройки для пустой структуры базы

Возможны и другие варианты, которые выбираются из выпадающего списка. Например, можно не выгружать инструкции создания базы и таблиц, но выгрузить инструкции удаления старых данных и добавления новых (рис.1.41).

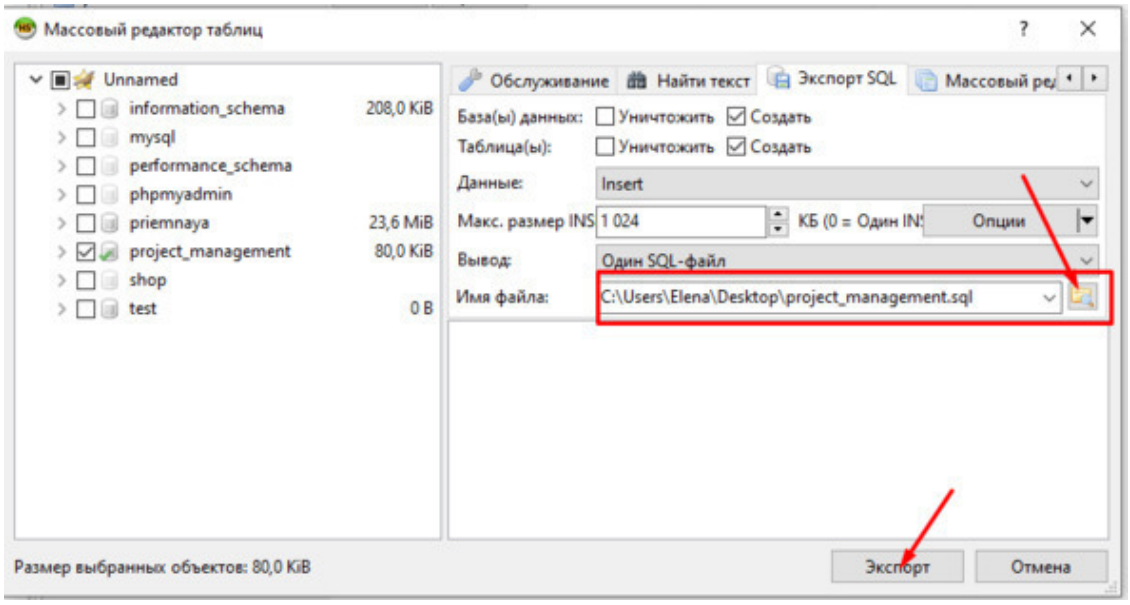

Рис.1.41 – Удаление старых данных и добавление новых через резервную копию.

Мы сохраним вариант, который показан на рис.1.39, укажем имя файла и нажмем «Экспорт» (рис.1.42).

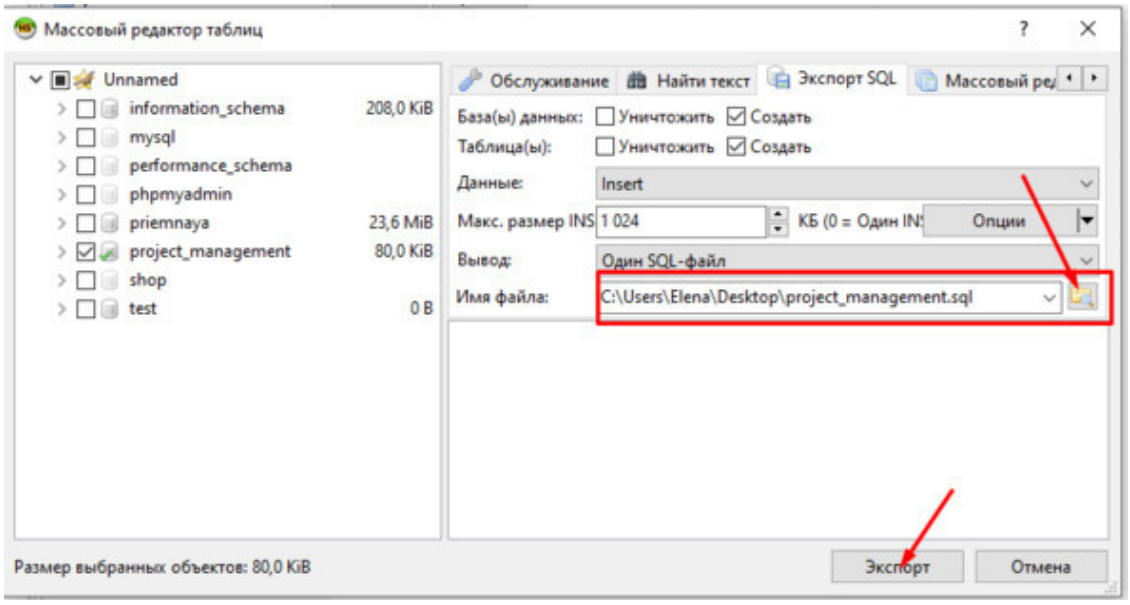

Рис.1.42 – Экспорт

В результате на диске сохранится файл project_management. sql, который можно просматривать в любом редакторе кода (рис.1.43).

Конец ознакомительного фрагмента.