

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ**
**Федеральное государственное бюджетное
образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

С. В. Рындина

**ЦИФРОВЫЕ ТЕХНОЛОГИИ УПРАВЛЕНИЯ
ПОЛУЧЕНИЕМ, ХРАНЕНИЕМ, ПЕРЕДАЧЕЙ
И ОБРАБОТКОЙ БОЛЬШИХ ДАННЫХ:
MYSQL**

Учебно-методическое пособие

ПЕНЗА 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Пензенский государственный университет» (ПГУ)

С. В. Рындина

Цифровые технологии управления получением, хранением, передачей и обработкой больших данных: MySQL

Учебно-методическое пособие

Пенза
Издательство ПГУ
2024

УДК 004.6(075)
Р95

Р е ц е н з е н т
кандидат технических наук, доцент,
индивидуальный предприниматель
А. А. Масленников

Рындина, Светлана Валентиновна.

Р95

Цифровые технологии управления получением, хранением, передачей и обработкой больших данных : учеб.-метод. пособие / С. В. Рындина. – Пенза : Изд-во ПГУ, 2024. – 118 с.

Рассмотрены возможности языка составления запросов к реляционной базе данных SQL на основе системы управления базами данных MySQL. Приведены задания для лабораторных работ по созданию запросов к базе данных для решения бизнес-задач. Содержание соответствует программе дисциплины «Цифровые технологии управления получением, хранением, передачей и обработкой больших данных». Работа может быть использована и при изучении дисциплины «Бизнес-аналитика на основе больших данных», а также при написании выпускной работы бакалавра.

Издание подготовлено на кафедре «Цифровая экономика» ПГУ и предназначено для обучающихся по направлению подготовки 38.03.05 «Бизнес-информатика».

УДК 004.6(075)

© Пензенский государственный
университет, 2024

СОДЕРЖАНИЕ

Введение	4
1. Настройка программного окружения	5
1.1. Система управления базами данных MySQL	5
1.2. Графический клиент MySQL Workbench:	
Разработка баз данных	16
1.3. Графический клиент MySQL Workbench:	
Проектирование и моделирование баз данных	42
1.4. Консольные клиенты MySQL Command Line Client	
и MySQL Shell.....	55
1.5. Менеджер баз данных DBeaver.....	61
2. Работа с данными	73
2.1. Типы данных и комментарии.....	74
2.2. Операции DDL	78
2.3. Операции DML: INSERT, UPDATE, DELETE	86
2.4. Операции DML: SELECT	91
2.4.1. Условие WHERE, операторы сравнения LIKE и IN.....	92
2.4.2. Исключение дублей, сортировка, группировка	
и агрегации	98
2.4.3. Соединения	106
3. Лабораторные работы	113
3.1. Лабораторная работа «Настройка программного	
обеспечения»	113
3.2. Лабораторная работа «Операции DDL».....	113
3.3. Лабораторная работа «Операции DML»	114
3.4. Лабораторная работа «Бизнес-задачи».....	115
Список литературы.....	117

Введение

В соответствии с учебным планом студенты четвертого курса направления подготовки 38.03.05 «Бизнес-информатика» изучают дисциплину «Цифровые технологии управления получением, хранением, передачей и обработкой больших данных».

Важным элементом аналитики является получение данных из различных источников для последующего анализа и интерпретации.

В издании рассматривается, как выполнять запросы к базе данных под управлением системы управления базами данных (СУБД) MySQL, как настраивать программное окружение, как решать бизнес-проблемы с помощью запросов и переходить от бизнес-задач к формулированию последовательности запросов, а при получении результата интерпретировать его с точки зрения полученного решения бизнес-проблемы.

При написании учебно-методического пособия использовались источники [1–6].

1. Настройка программного окружения

Для того, чтобы работать с данными, необходимо их откуда-то получать, где-то хранить. Одним из решений являются базы данных MySQL.

1.1. Система управления базами данных MySQL

Установим Database Management System (DBMS) или СУБД. В пособии используется СУБД MySQL8 (версия 8.0.36 для Windows). На сайте для загрузки установочных пакетов: <https://dev.mysql.com/downloads/installer/> необходимо выбрать операционную систему (рис. 1.1) и перейти к скачиванию нужной версии установщика, выбрав вариант скачивания без регистрации (рис. 1.2). Выбираем стандартный установщик MySQL, который включает полный комплект приложений MySQL, в отличие от меньшей web-community версии, которая при установке подгружает только выбираемые к установке продукты MySQL, что не очень подходит для первой установки и начала работы в новой СУБД [1].

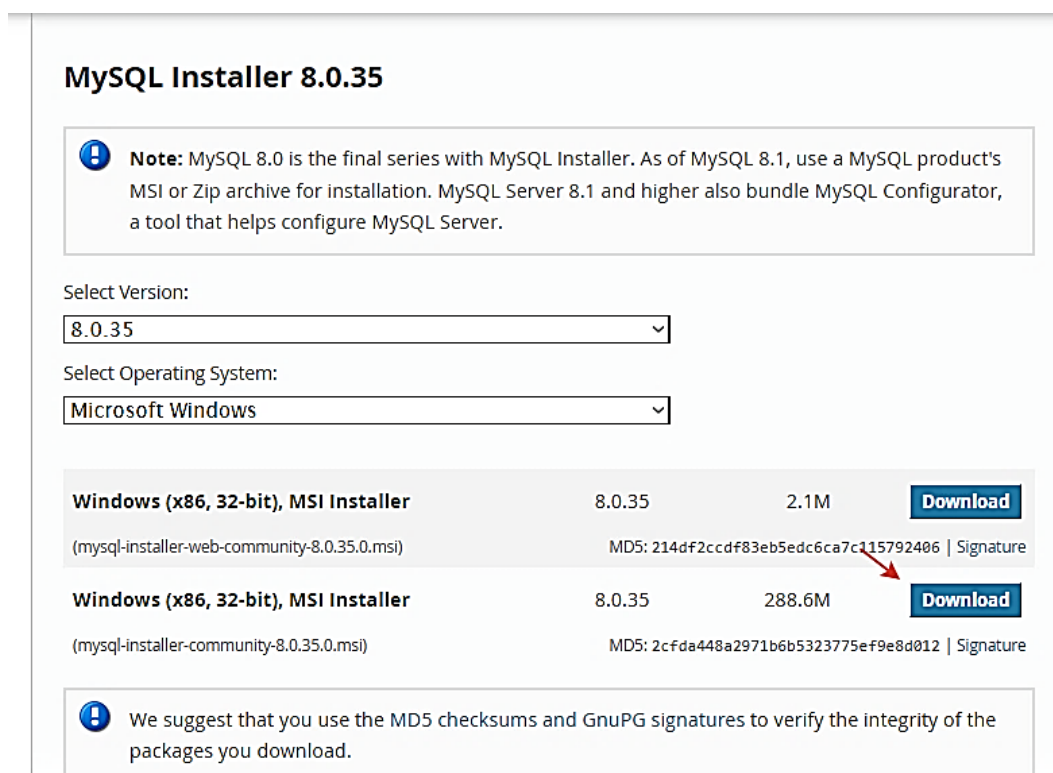


Рис. 1.1. Страница для выбора версии установщика MySQL Installer 8.0.35

MySQL Community Downloads

Login Now or Sign Up for a free account.

An Oracle Web Account provides you with the following advantages:

- Fast access to MySQL software downloads
- Download technical White Papers and Presentations
- Post messages in the MySQL Discussion Forums
- Report and track bugs in the MySQL bug system

Login »
using my Oracle Web account

Sign Up »
for an Oracle Web account

MySQL.com is using Oracle SSO for authentication. If you already have an Oracle Web account, click the Login link. Otherwise, you can signup for a free account by clicking the Sign Up link and following the instructions.

No thanks, just start my download.

Рис. 1.2. Выбор варианта скачивания MySQL Installer без регистрации

После запуска установщика можно выбрать тип установки (рис. 1.3). Полный вариант включает выбранную версию MySQL Server и другие инструменты MySQL, например, MySQL Workbench.

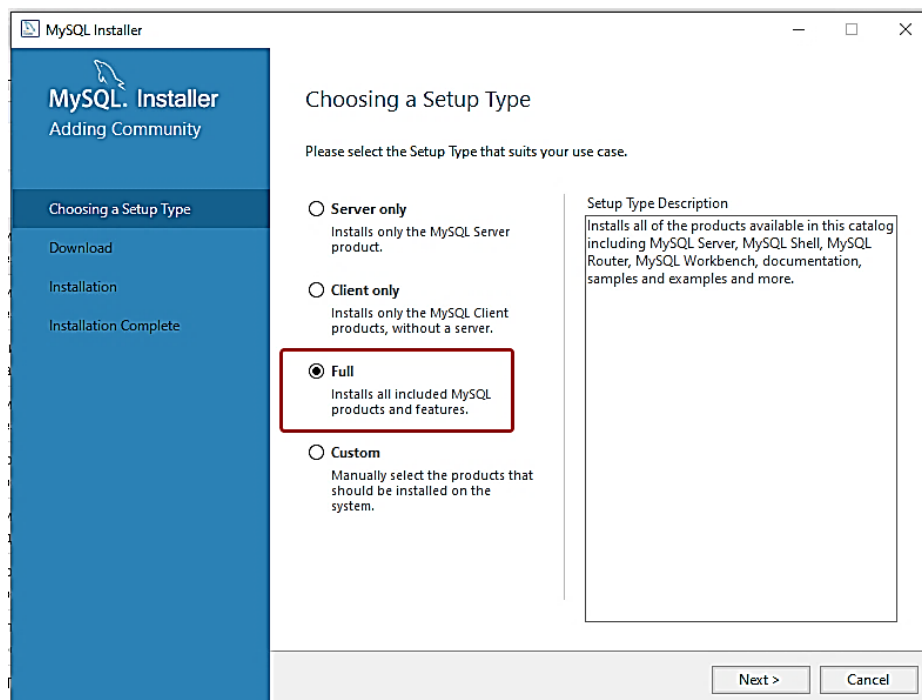


Рис. 1.3. Выбор типа установки MySQL

Для запуска приложений MySQL может потребоваться Microsoft Visual C++ Redistributable (Microsoft Visual C++ Reditributable – пакет, устанавливающий компоненты среды выполнения библиотек Visual C++, необходимых для запуска приложений, разработанных

с помощью Visual C++, на компьютере без установленной среды Visual C++ 2015, 2017 и 2019). После его установки (рис. 1.4), если нужной версии среды Visual C++ не было, пользователь переходит к выбору компонент. Если необходимости в установке дополнительного ПО не было, то сразу с помощью кнопки Execute выбирает и устанавливает нужные компоненты (рис. 1.5–1.7).

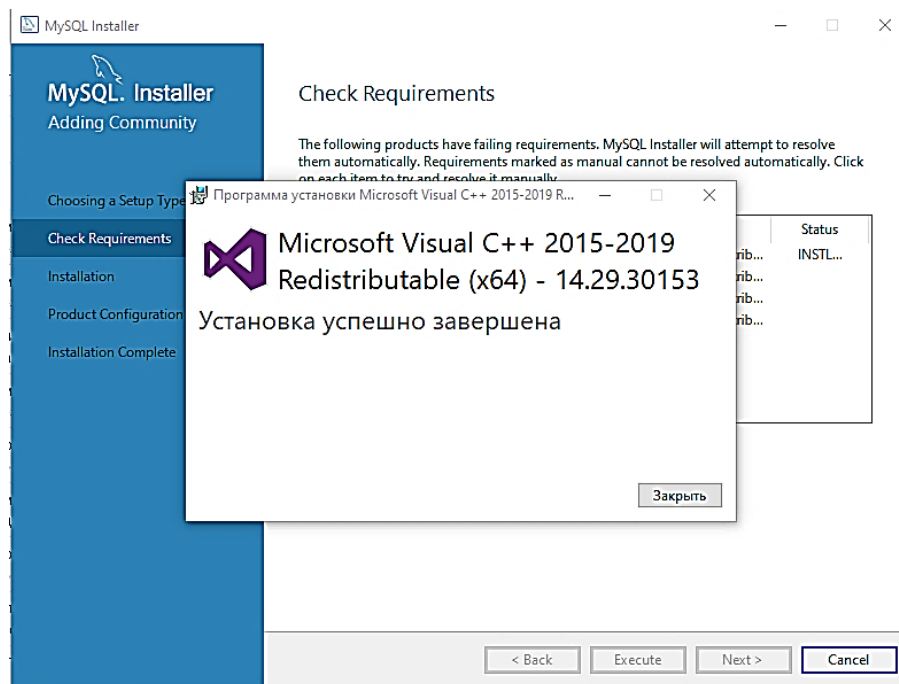


Рис. 1.4. Установка Microsoft Visual C++ Redistributable

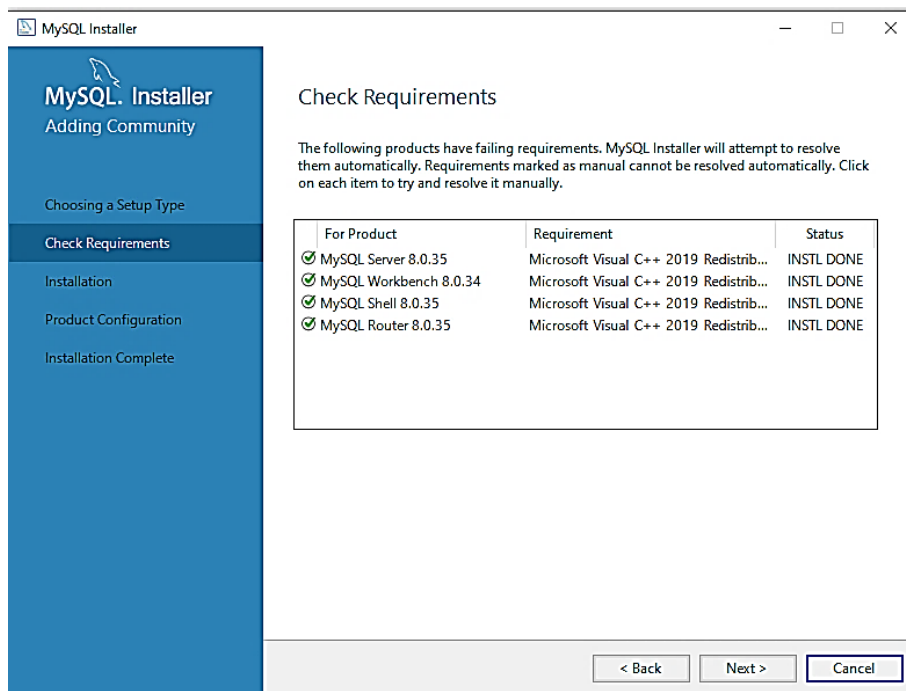


Рис. 1.5. Переход к следующему окну настройки установки

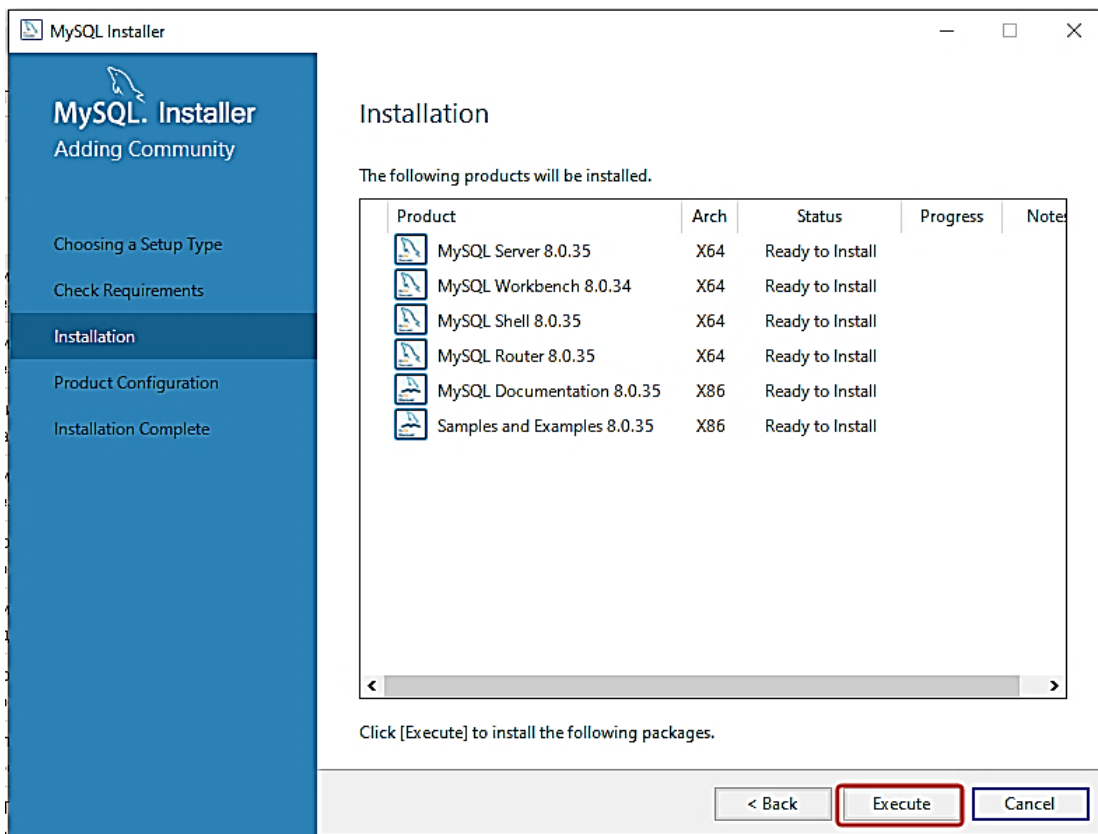


Рис. 1.6. Выбор компонентов для установки

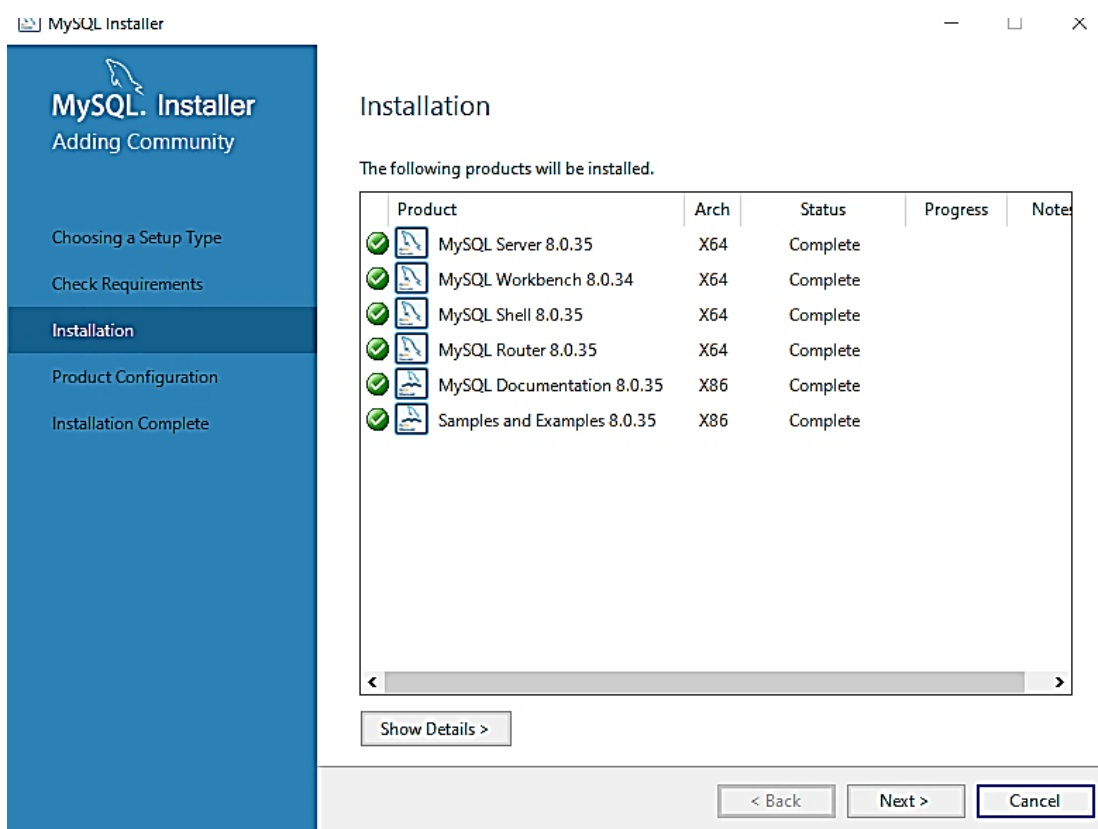


Рис. 1.7. Загрузка всех необходимых файлов

Далее установочная программа предложит настроить конфигурацию некоторых компонентов (рис. 1.8).

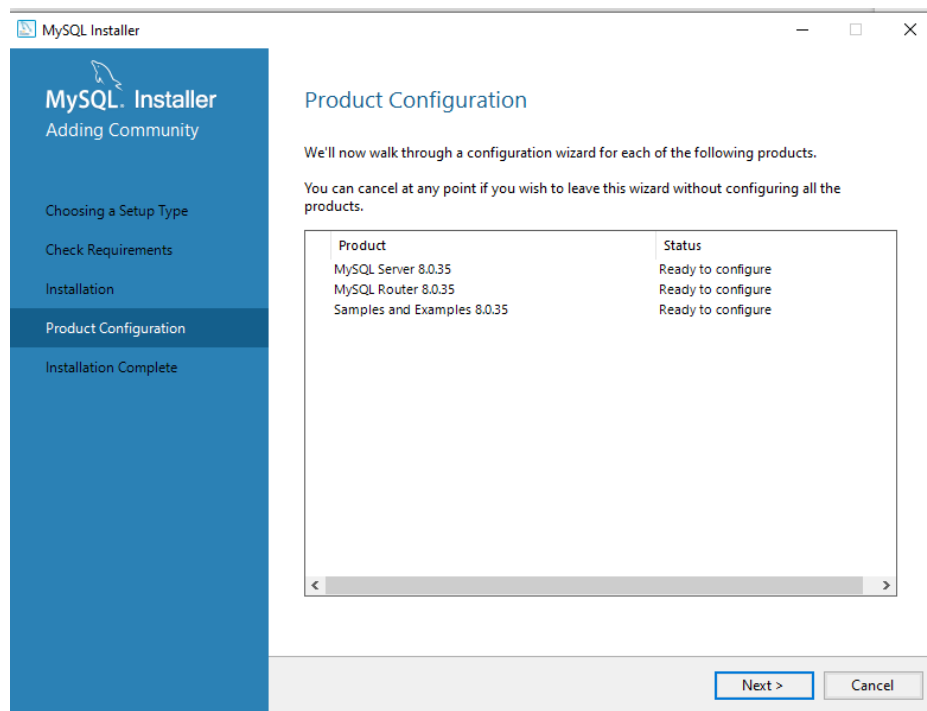


Рис. 1.8. Настройка конфигурации компонентов после установки

Для MySQL Server в настройках его конфигурации уже выставлены все нужные параметры по умолчанию (рис. 1.9) и установлен рекомендованный тип аутентификации (рис. 1.10).

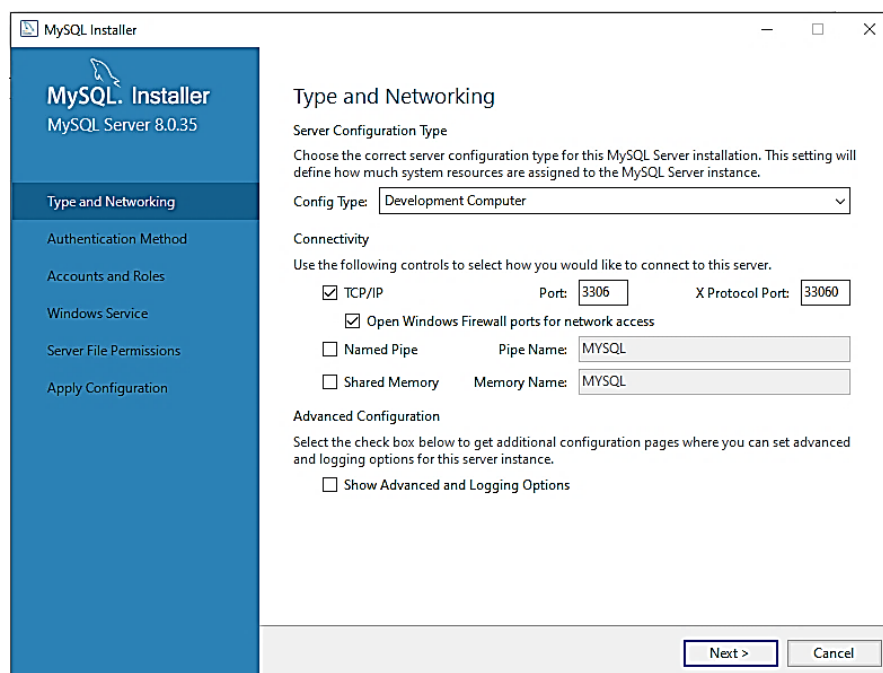


Рис. 1.9. Конфигурация MySQL Server

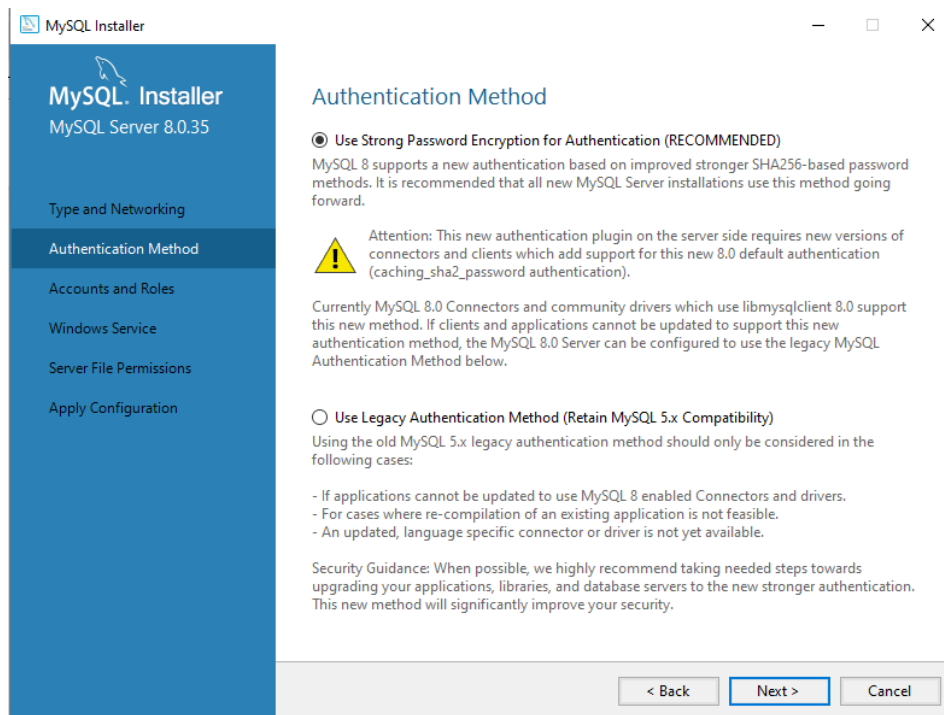


Рис. 1.10. Рекомендованный тип аутентификации

В следующем окне зададим пароль суперпользователя root (рис. 1.11), также можно добавить других пользователей с помощью кнопки Add User и заполнить необходимые поля.

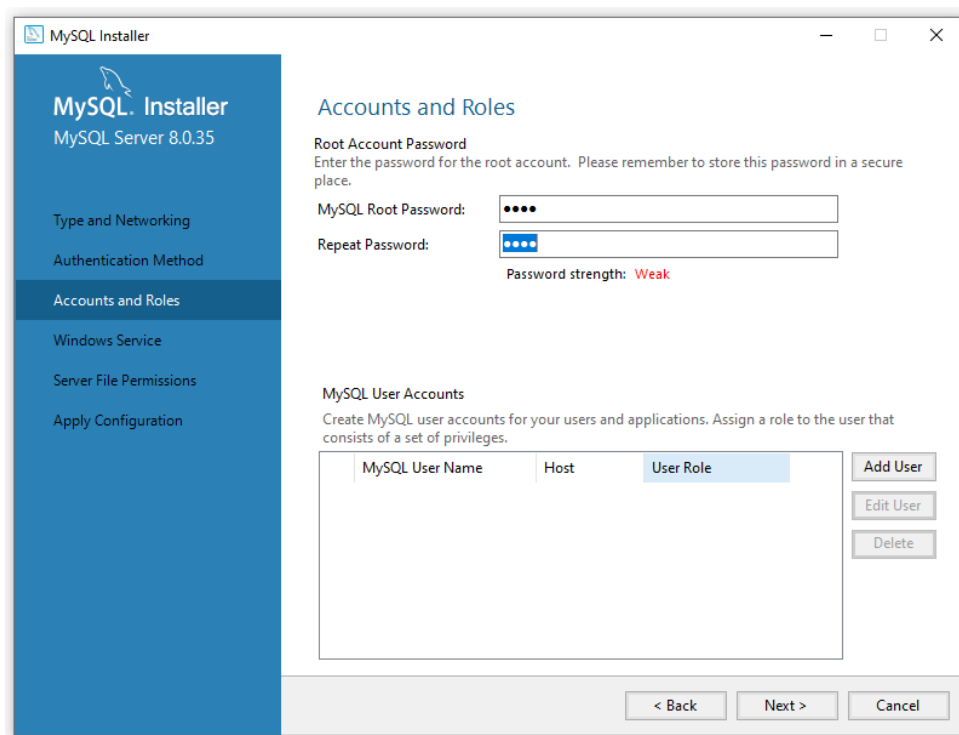


Рис. 1.11. Настройка пароля для суперпользователя root

В следующем окне предлагается настроить запуск MySQL Server как Windows-сервис, если снять галочку, то запускать его нужно будет из командной строки (рис. 1.12).

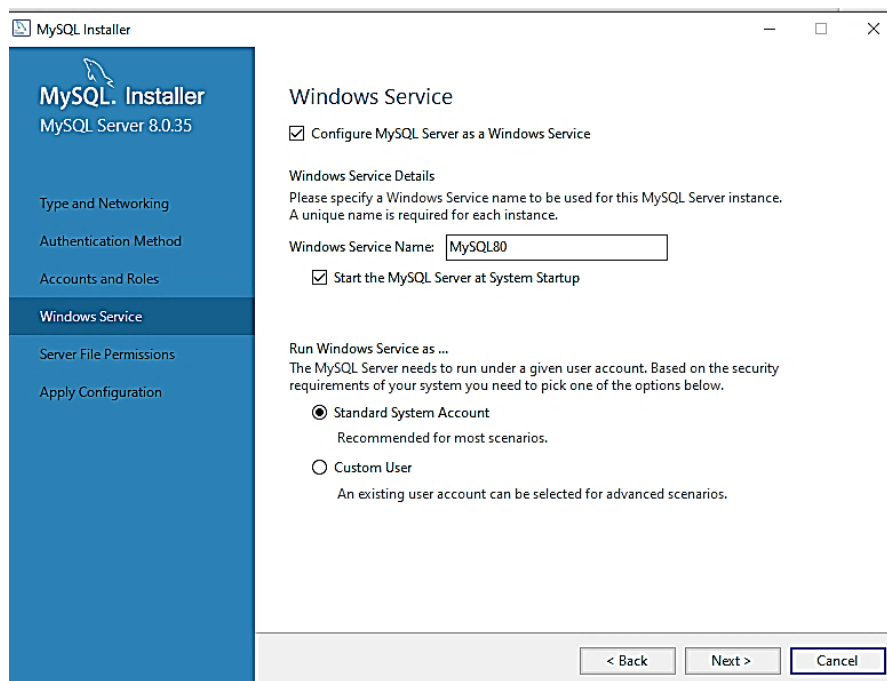


Рис. 1.12. Настройки для запуска MySQL Server как Windows-сервиса

Применяем настроенные параметры конфигурации (рис. 1.13, 1.14).

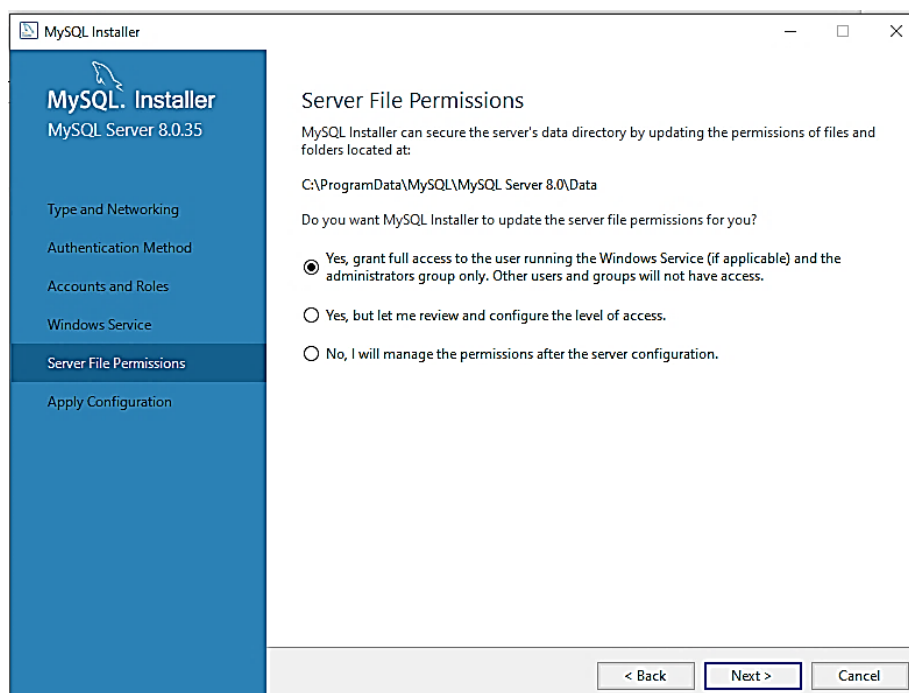


Рис. 1.13. Разрешения на доступ к файлам сервера

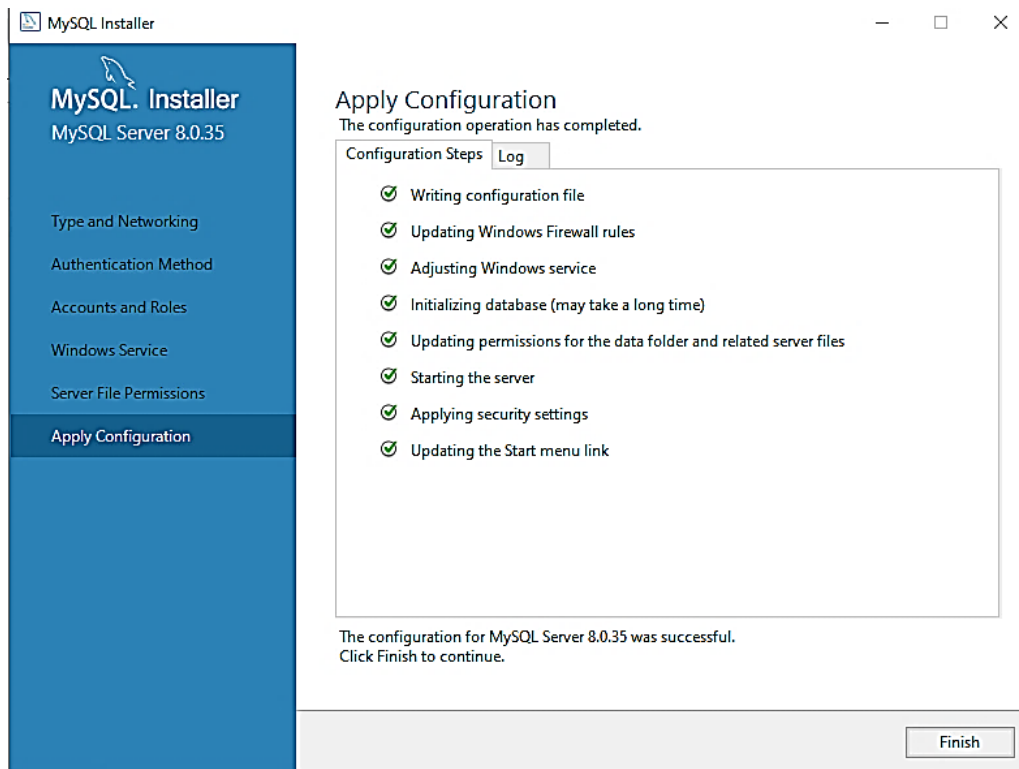


Рис. 1.14. Применение настроек конфигурации

Статус конфигурации MySQL Server помечен как завершённый (рис. 1.15).

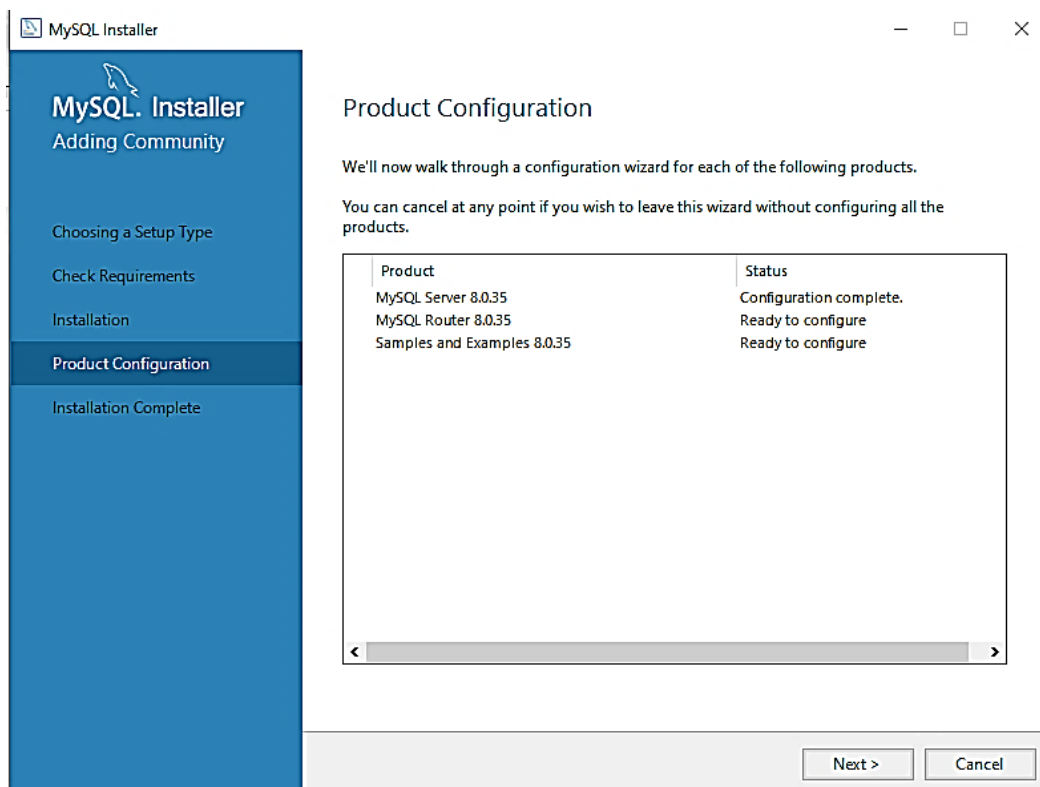


Рис. 1.15. Настройка конфигурации отдельных компонентов

Далее происходит настройка конфигурации MySQL Router с рекомендованными значениями параметров для распределения нагрузки между программами MySQL в кластере (рис. 1.16).

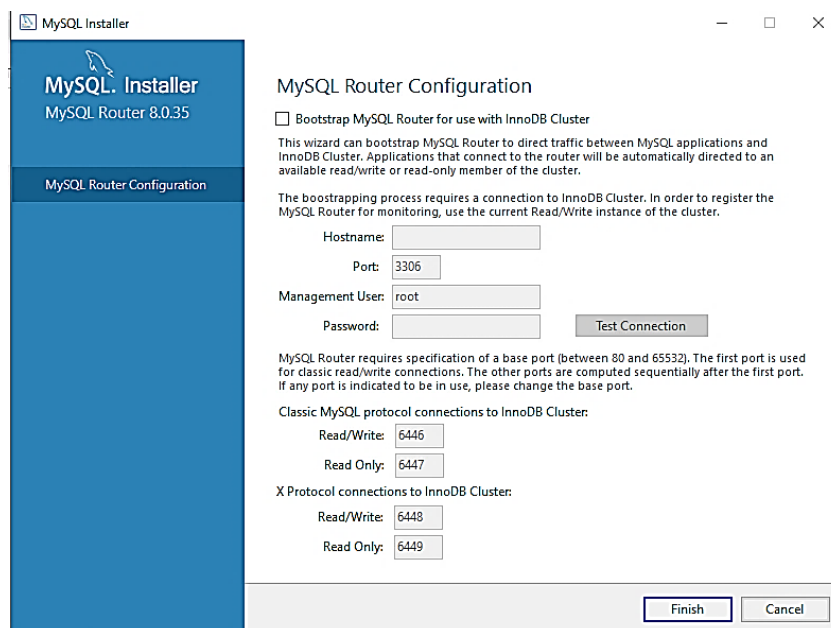


Рис. 1.16. Настройки для MySQL Router

Далее для настроек Samples and Examples необходимо проверить соединение с MySQL Server, заполнив логин root и ранее введенный для суперпользователя пароль и нажать кнопку Check для проверки соединения (рис. 1.17).

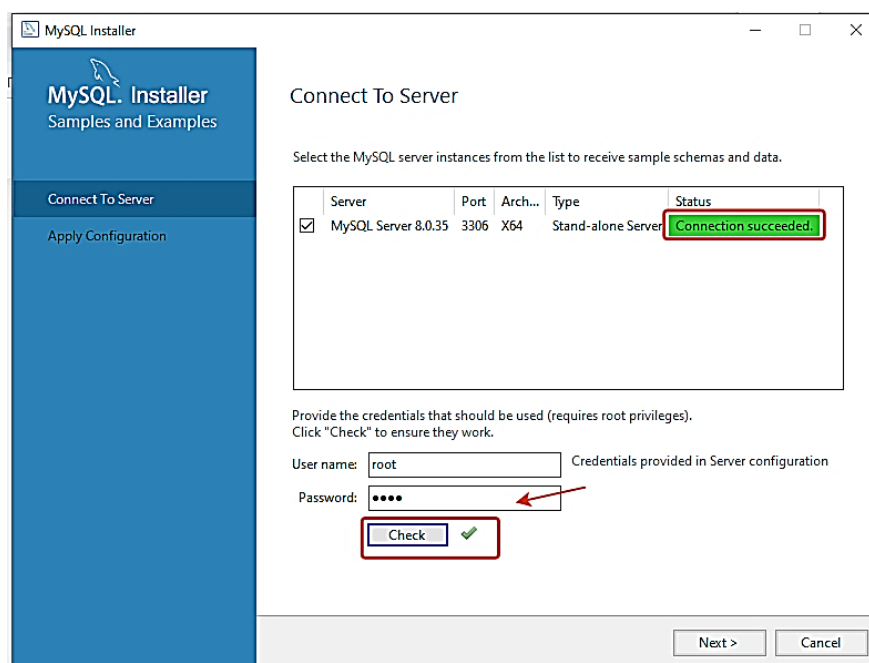


Рис. 1.17. Проверка соединения с MySQL Server

На следующем шаге кнопкой Execute выбираем все параметры конфигурации (рис. 1.18) и необходимые компоненты раздела Samples and Examples устанавливаются на сервер СУБД.

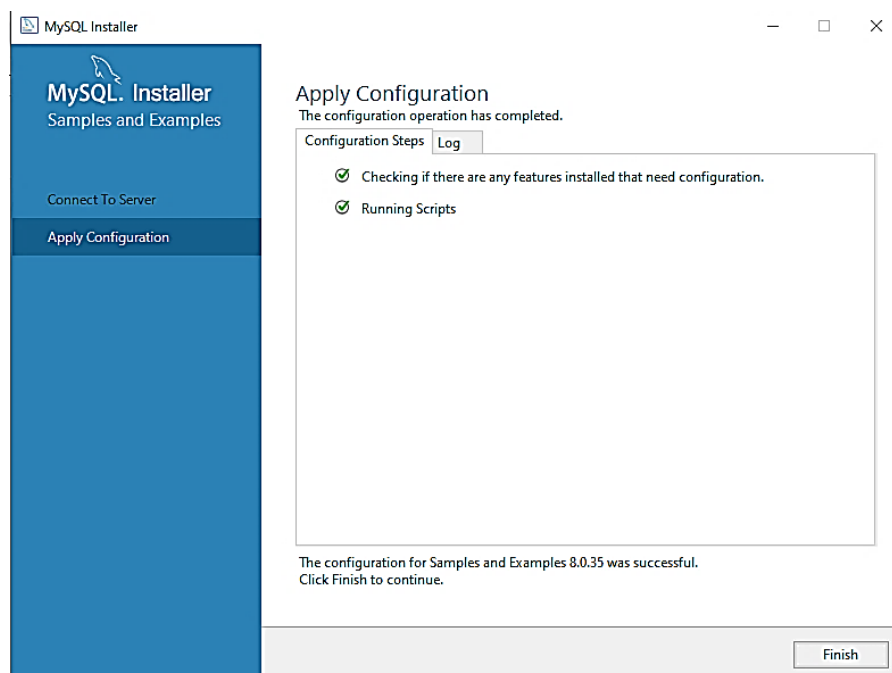


Рис. 1.18. Установка Samples and Examples

Осталось установить MySQL Workbench и MySQL Shell (рис. 1.19).

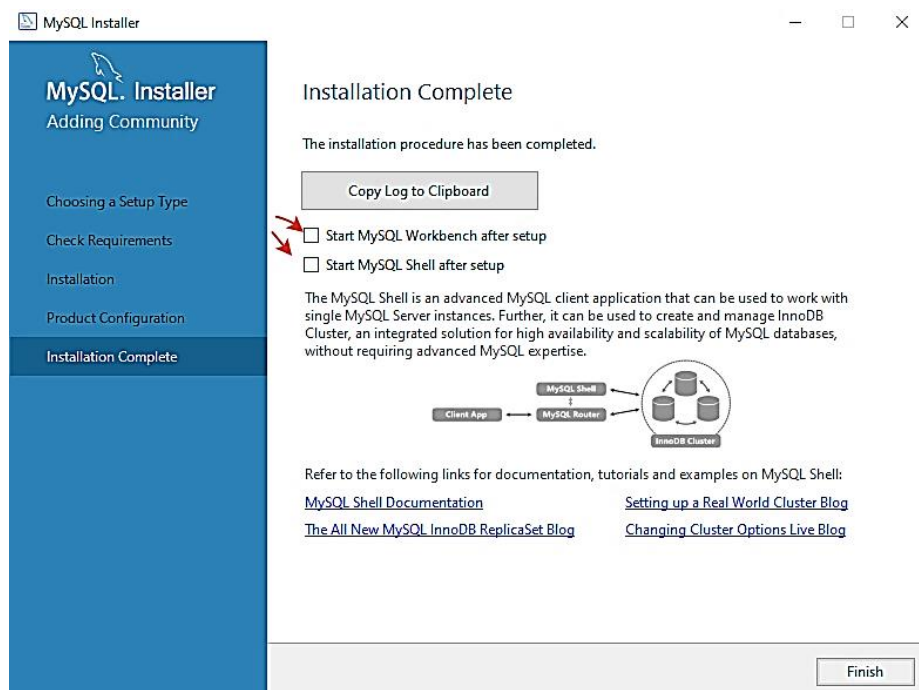


Рис 1.19. Установка MySQL Workbench и MySQL Shell

В списке программ появятся установленные компоненты СУБД MySQL (рис. 1.20).

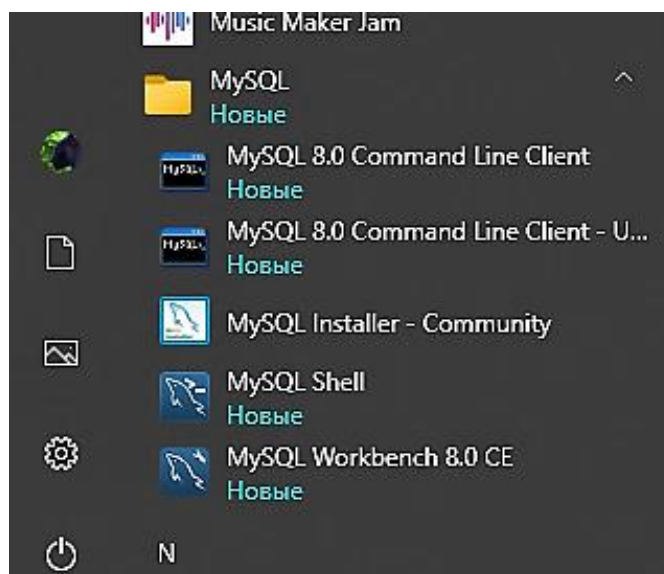


Рис. 1.20. Кнопка Пуск в операционной системе Windows

Для согласованной работы все установленные компоненты должны принадлежать одной версии. После выхода новой версии можно обновить компоненты, повторно запустив MySQL Installer и выбрав опцию Upgrade (рис. 1.21). Так все компоненты первоначальной установки были обновлены автором пособия до версии 8.0.36, которая стала актуальна почти сразу после установки версии 8.0.35.

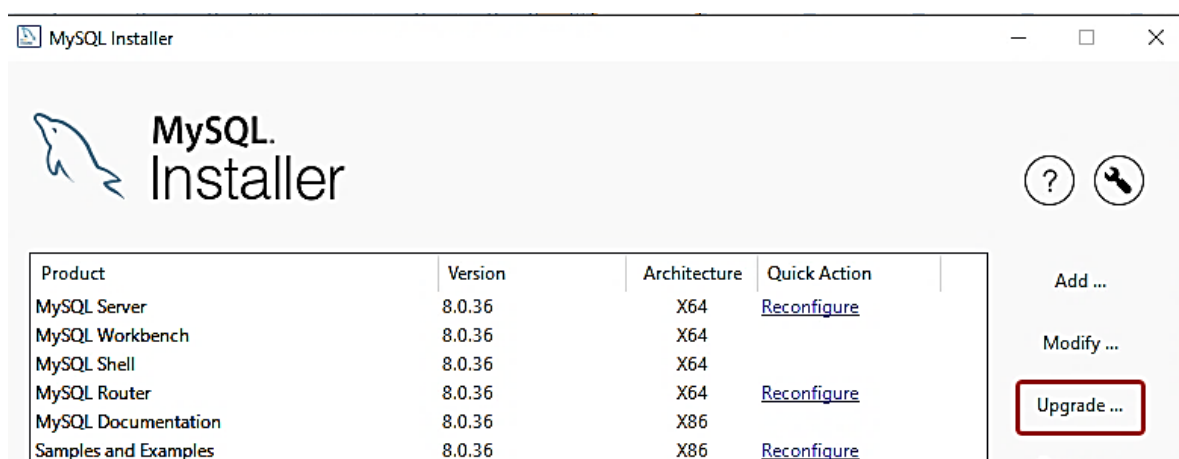


Рис. 1.21. Обновление компонентов MySQL

На этом установку завершили и приступаем к работе с СУБД MySQL.

1.2. Графический клиент MySQL Workbench: Разработка баз данных

MySQL Workbench – это унифицированный визуальный инструмент для архитекторов баз данных, разработчиков и администраторов баз данных. MySQL Workbench обеспечивает моделирование данных, разработку SQL и комплексные инструменты администрирования для настройки сервера, администрирования пользователей, резервного копирования и многого другого [5].

Запустим MySQL Workbench. При щелчке на единственное доступное соединение потребуется ввести пароль суперпользователя root (рис. 1.22).

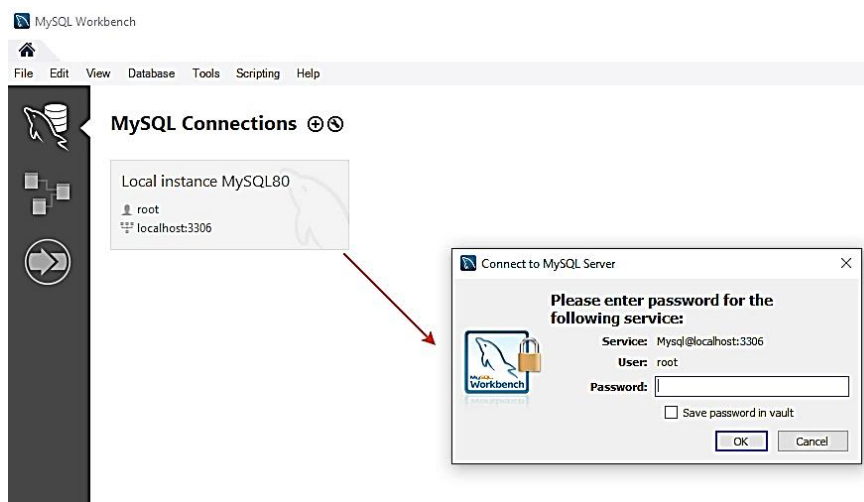


Рис. 1.22. Соединение с сервером

Последующие соединения можно создавать с выбором других пользователей, если они ранее были настроены.

В навигаторе выберем для установленного соединения верхнюю опцию Server Status (состояние сервера) и проверим соединение (рис. 1.23).

После обращения к командам управления сервером можно столкнуться с ошибкой выполнения (невозможно выполнить команду `chsr`) (рис. 1.24). Чтобы исправить эту ошибку, необходимо выполнить следующее. Во-первых, добавить путь `C:\Windows\System32` к переменной среды. Для этого используем командную строку, найдя соответствующее приложение по кнопке Пуск (рис. 1.25) или вызвав приложение горячими клавишами `Win+R` на клавиатуре. В диалоговом окне командной строки ввести `sysdm.cpl` (рис. 1.26) и выполнить команду.

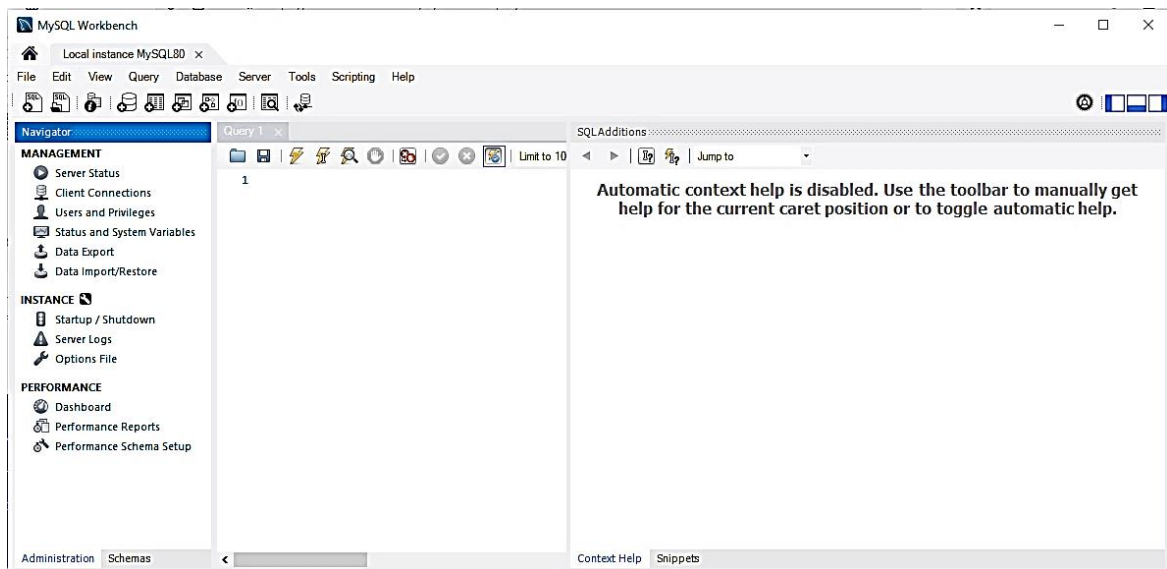


Рис. 1.23. Проверка состояния сервера

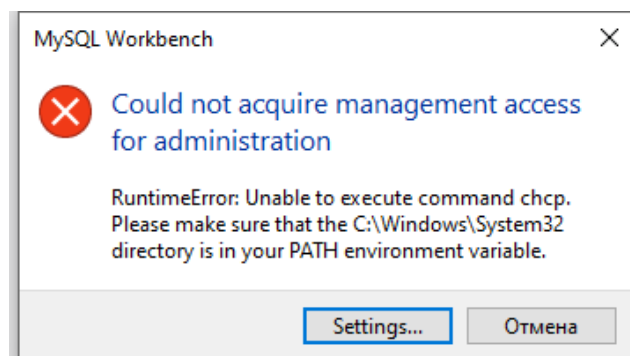


Рис. 1.24. Ошибка выполнения при обращении к управлению сервером

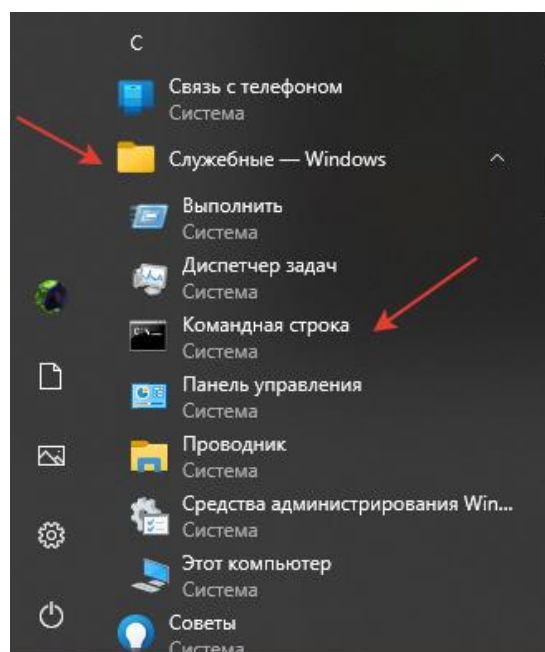


Рис. 1.25. Службные приложения Windows

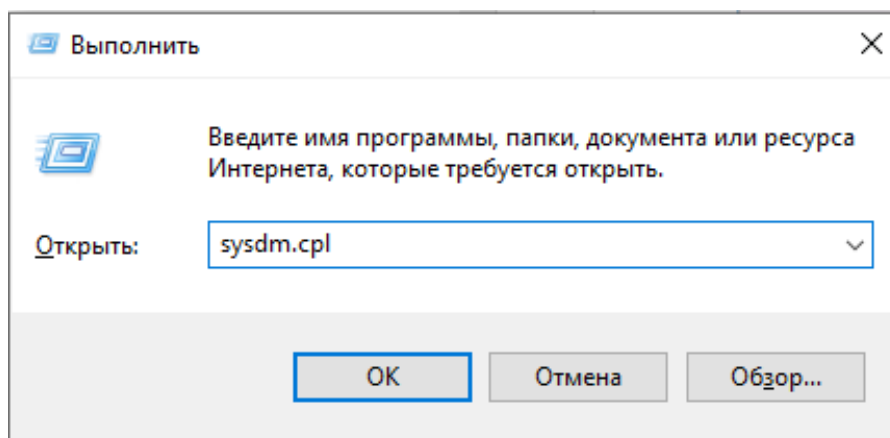


Рис. 1.26. Приложение Командная строка (Win+R)

Далее выбираем вкладку Дополнительно (рис. 1.27) и по кнопке Переменные среды переходим в соответствующее диалоговое окно (рис. 1.28).

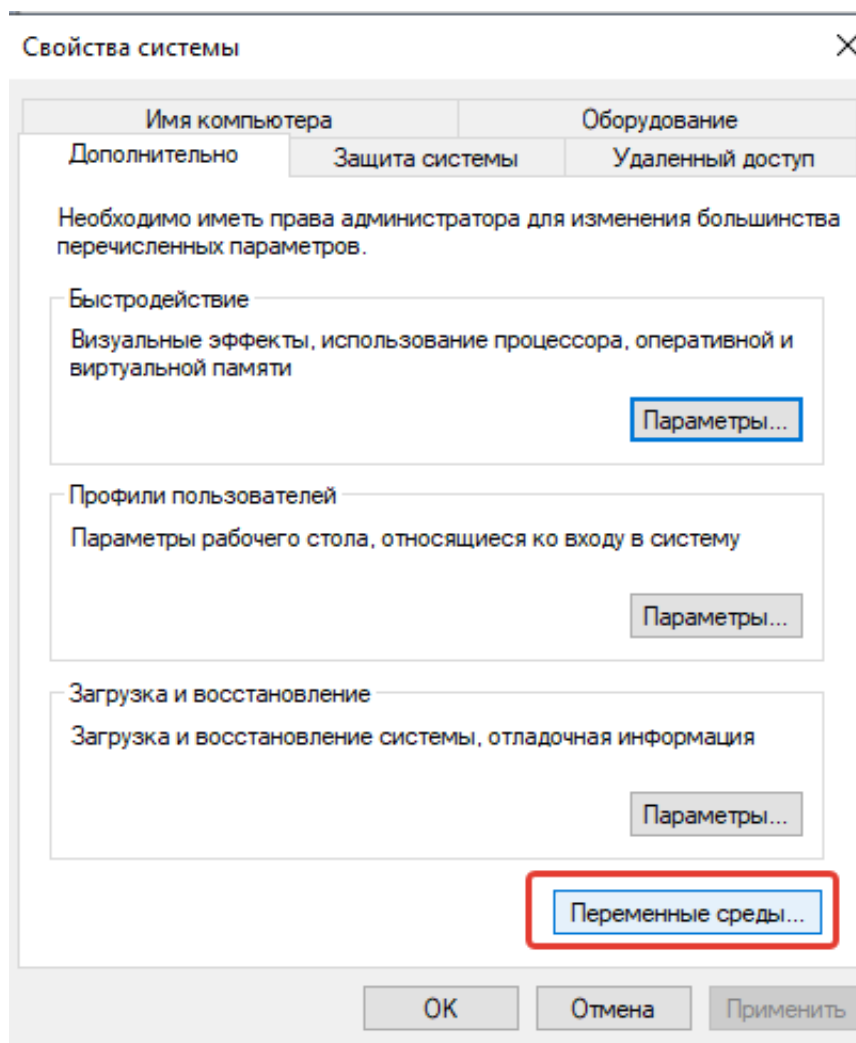


Рис. 1.27. Свойства системы

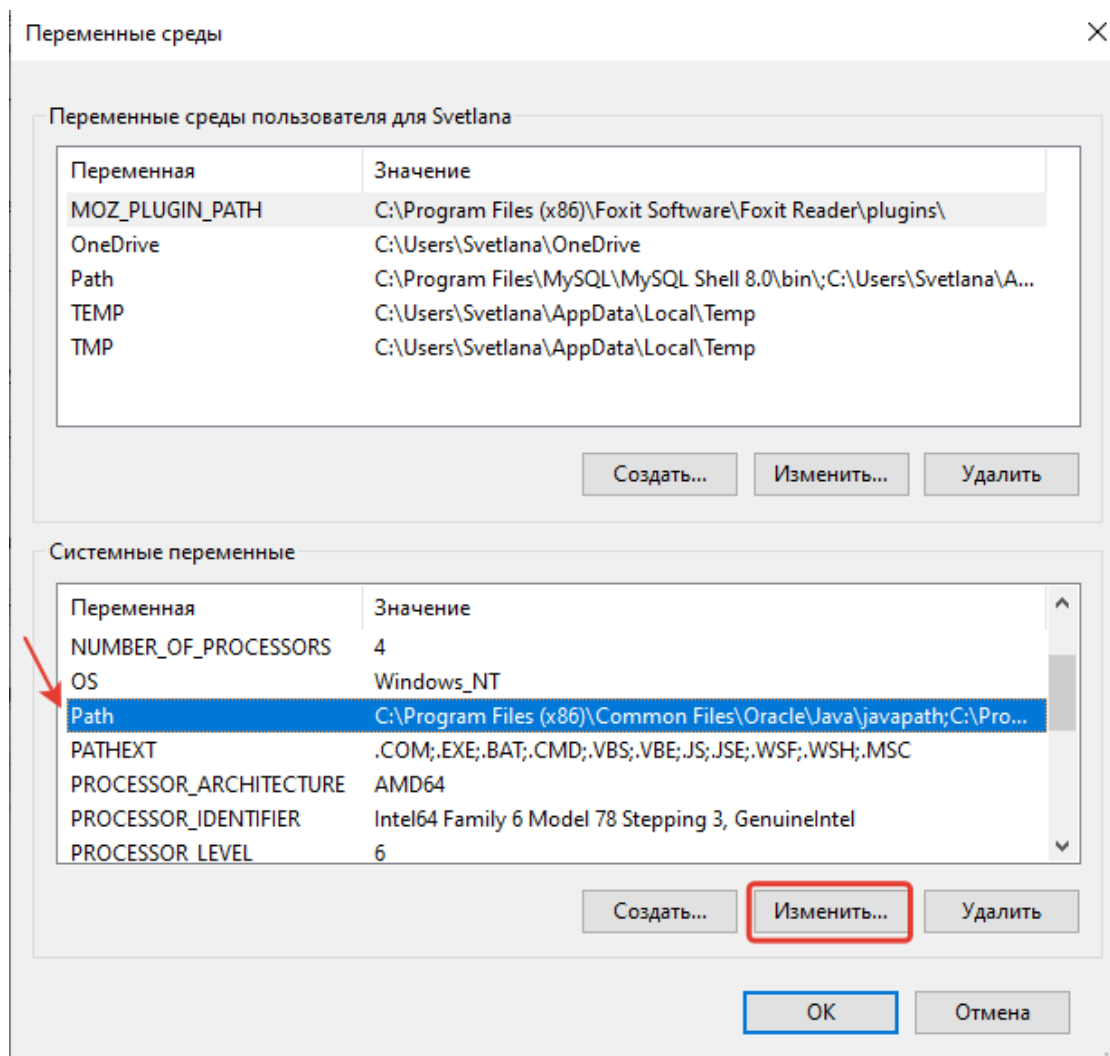


Рис. 1.28. Изменение системных переменных

Создаем новую переменную C:\Windows\System32 (рис. 1.29).

В версиях MySQL Installer позднее 8.0.20 в unicode python, если русский язык – это основной язык, настроенный для пользователей операционной системы Windows, то при недоступности кодировки utf-8 выбирается именно он, что приводит к проблемам с кодировкой кириллицы. Так как производимые для устранения этой проблемы изменения вступают в силу только после немедленной перезагрузки системы, то необходимо подготовить рабочее пространство, сохранив все важные документы и закрыв программы. Зайдем в параметры операционной системы (рис. 1.30) и выберем категорию Время и язык, в которой перейдем в настройки языка (рис. 1.31).

Перейдем в Административные языковые параметры (рис. 1.32) и изменим язык программ, не поддерживающих Юникод, на Английский (США). Прежде чем изменения вступят в силу, потребуется перезагрузка операционной системы.

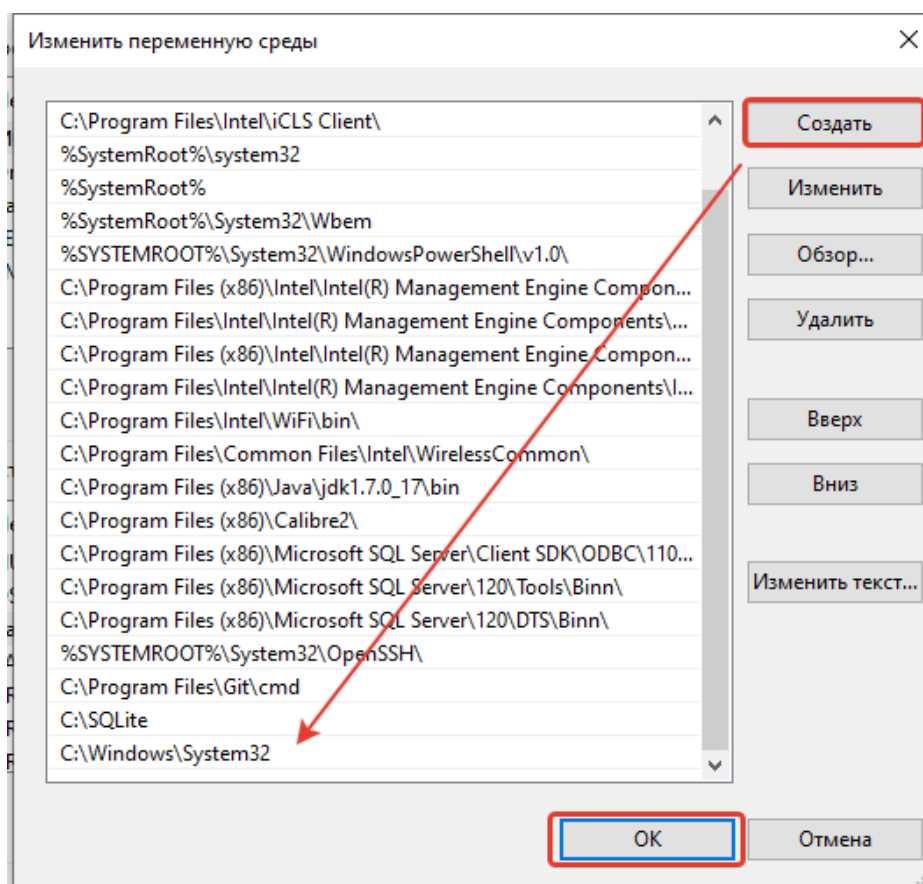


Рис. 1.29. Создание новой системной переменной

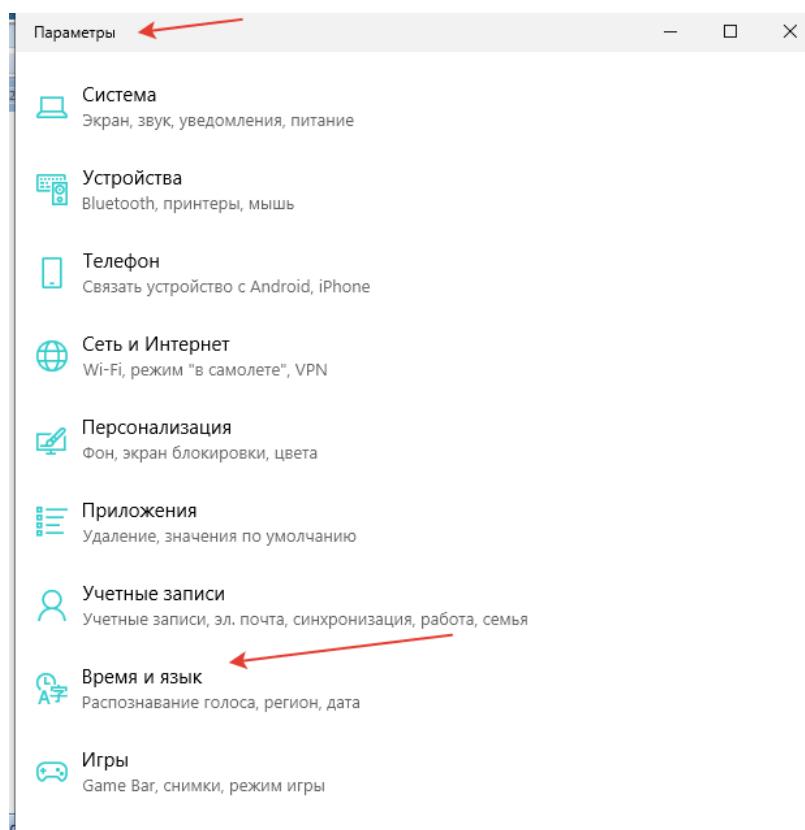


Рис. 1.30. Параметры операционной системы

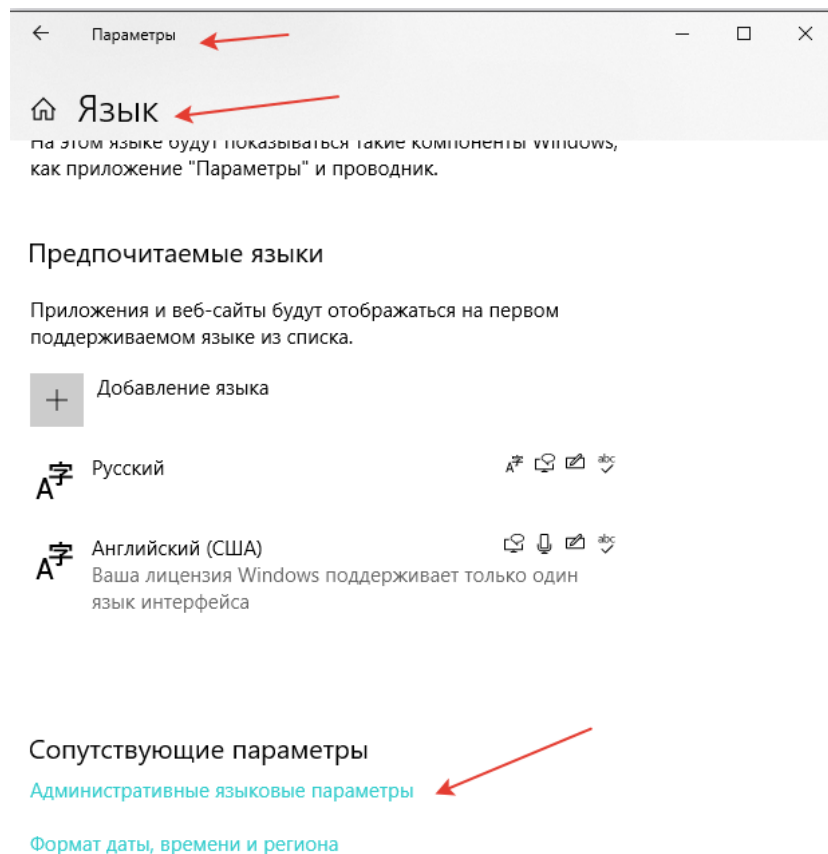


Рис. 1.31. Настройки для языка

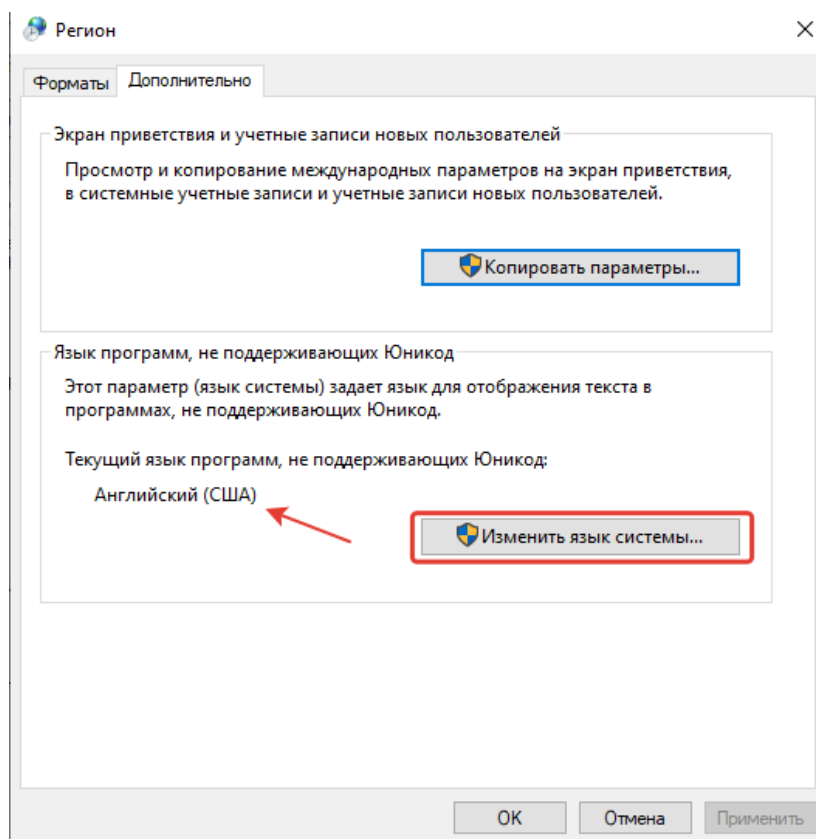


Рис. 1.32. Изменение языка для программ

Еще раз запустим MySQL Workbench, подключимся к серверу и запросим его состояние (рис. 1.33).

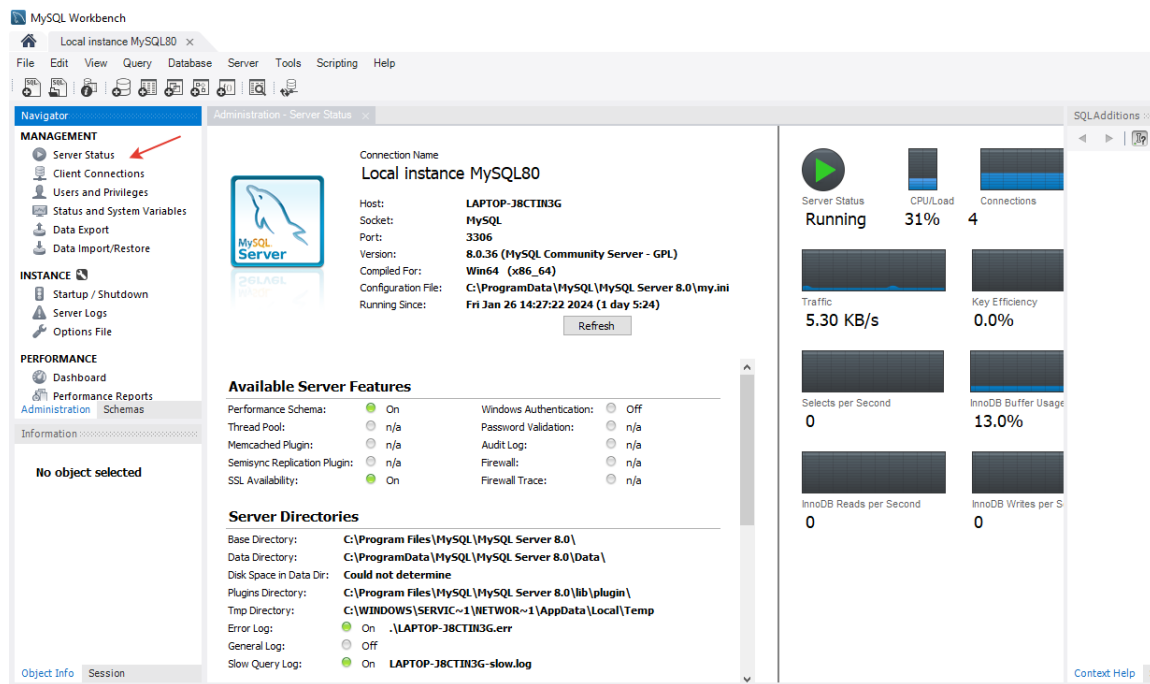


Рис. 1.33. Состояние сервера

Перейдем в панели Navigator на вкладку Schemas (рис. 1.34). Раскроем содержимое БД sakila (она входит в установленные Samples and Examples) и выведем на экран Inspector для таблицы actor (рис. 1.35).

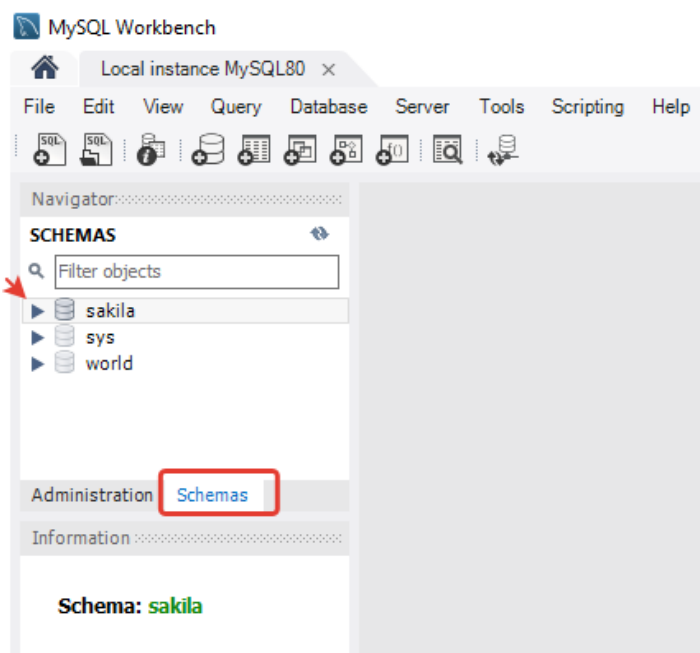


Рис. 1.34. Вкладка Schemas

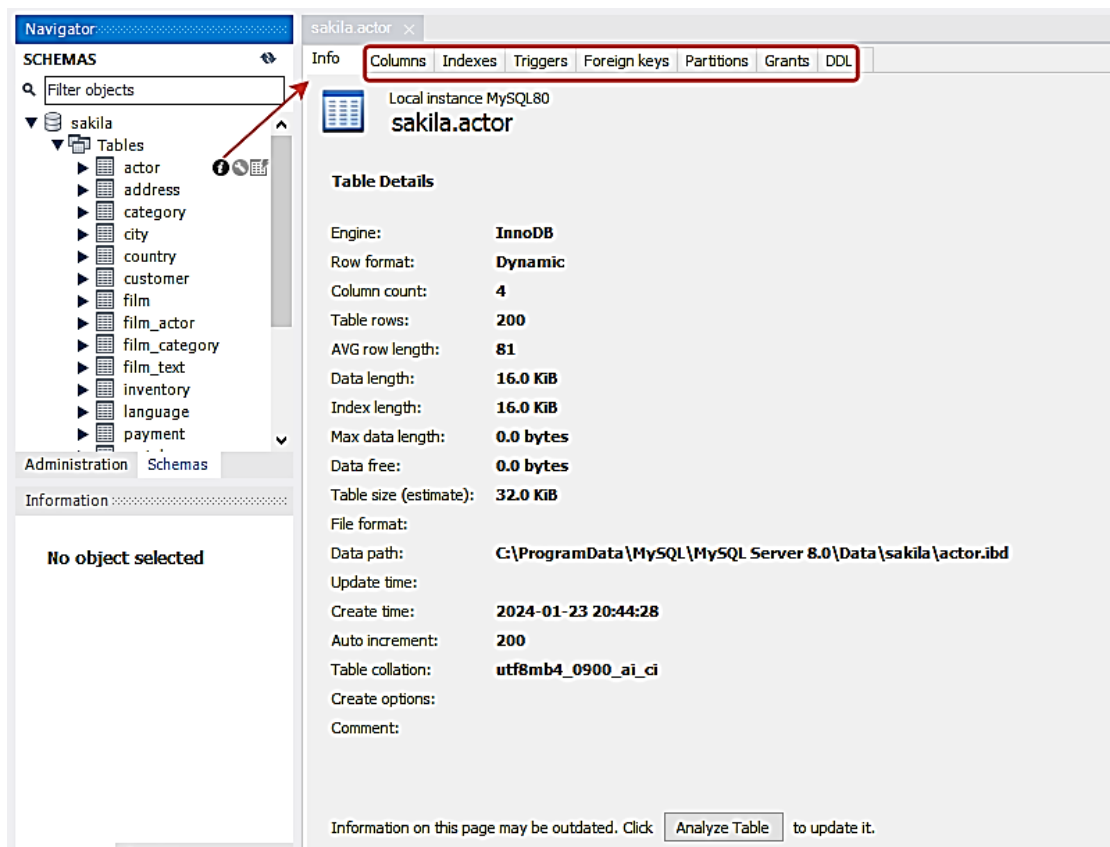


Рис. 1.35. Информация о таблице actor БД sakila

Обратите внимание, что обращение к таблице происходит через указание БД, к которой она принадлежит: `sakila.actor`, так как таблица с таким именем может быть создана и в другой БД.

Можно познакомиться с содержимым других вкладок. Например, можно просмотреть запрос на языке DDL для создания таблицы `actor` (рис. 1.36).

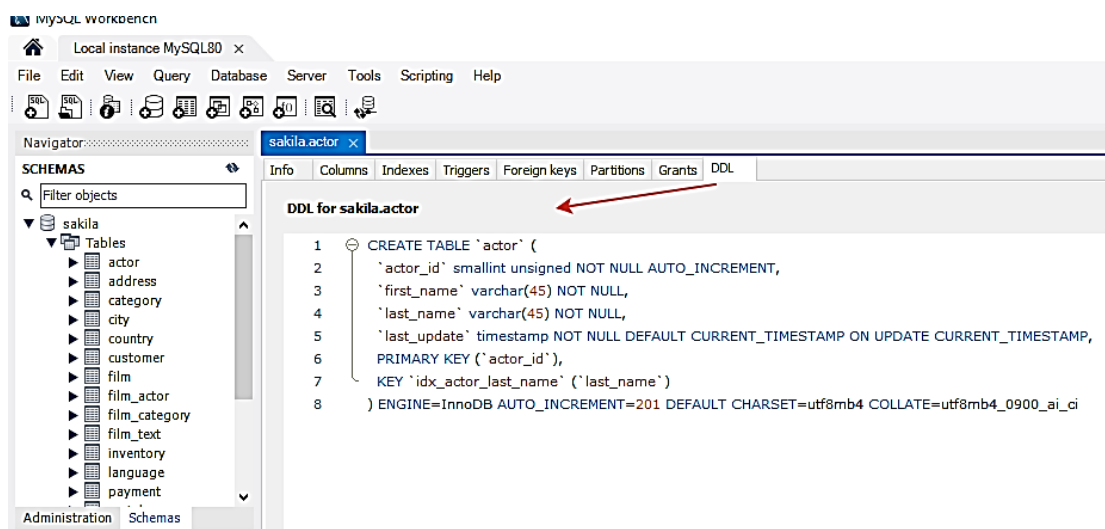


Рис. 1.36. Запрос SQL для создания таблицы actor

Содержимое таблицы actor с помощью третьей пиктограммы в виде таблицы рядом с именем таблицы в навигаторе (рис. 1.34) выводится как выполнение запроса SQL (рис. 1.37). Result Grid – это таблица результатов, а Action Output – содержит историю выполнения команд (при успешном выполнении слева от команды зеленая галочка, при ошибках – красный крестик).

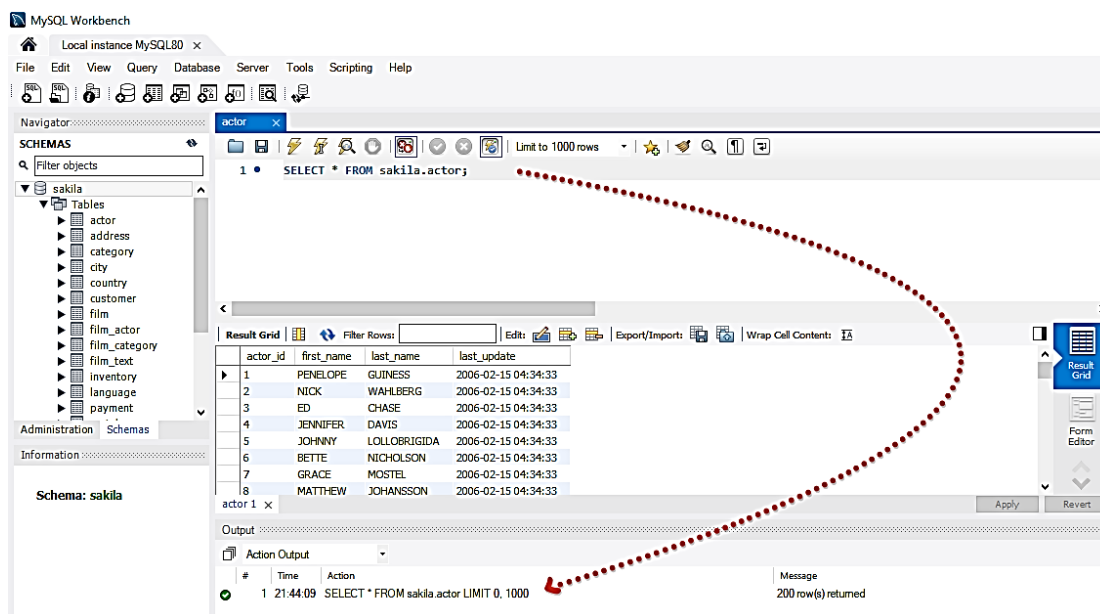


Рис. 1.37. Результат выполнения запроса

Новую вкладку для создания запросов к доступным БД можно организовать с помощью командного меню File > New Query Tab (рис. 1.38).

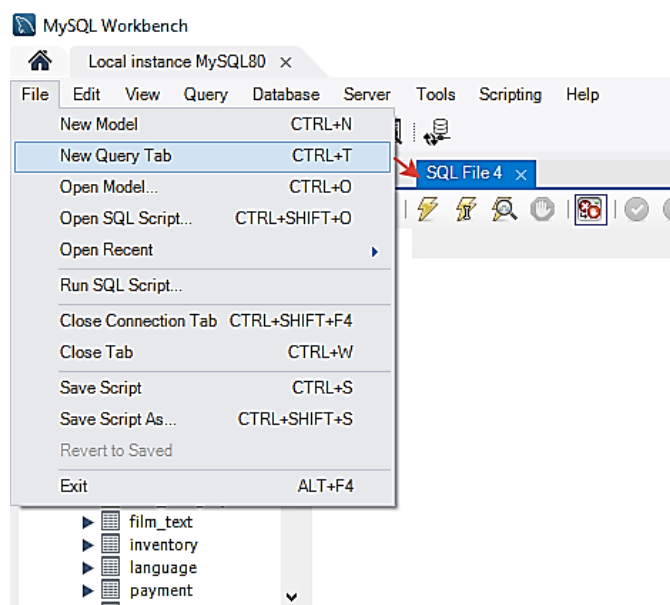


Рис. 1.38. Новая вкладка с файлом сценария SQL (редактором запросов)

В редакторе запросов при наборе команд всплывает окно автозаполнения (рис. 1.39), однако в нем пока все ключевые слова в нижнем регистре.

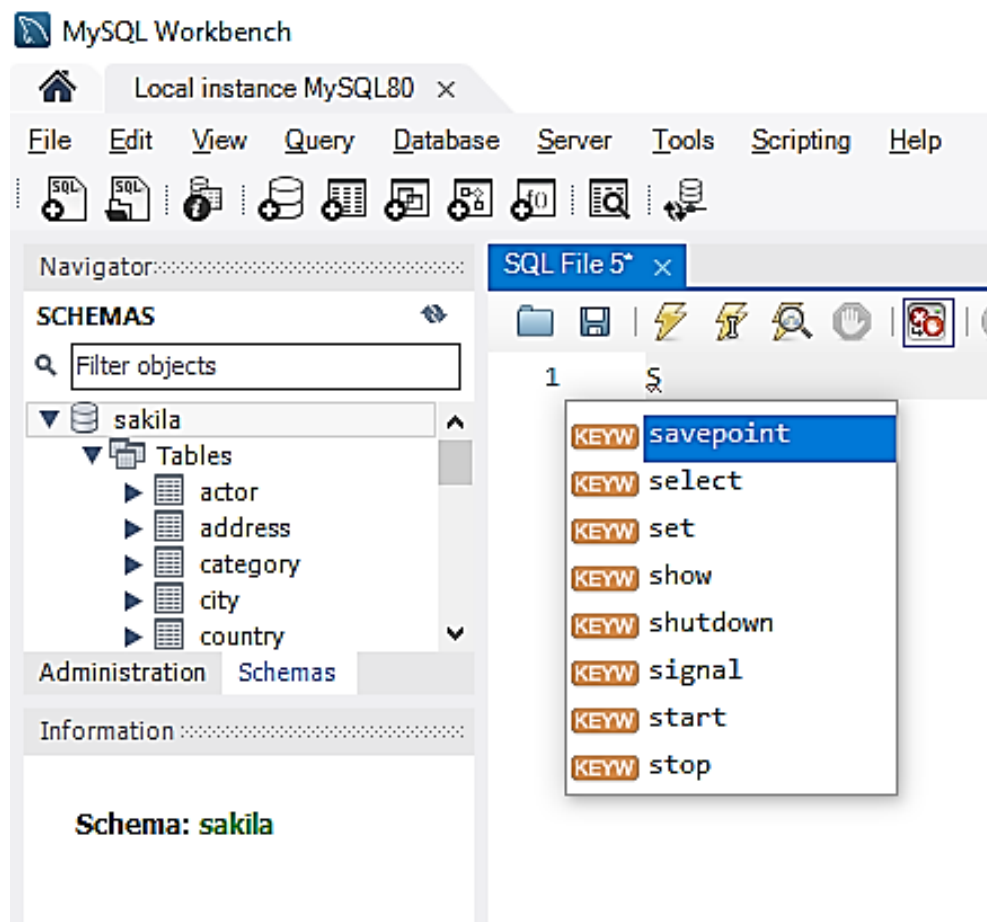


Рис. 1.39. Окно автозаполнения

По негласному соглашению, которое никак не влияет на исполнимость запросов, но повышает читабельность кода, все ключевые слова принято набирать в верхнем регистре, а названия полей, таблиц и тому подобное – в нижнем. Произведем необходимые настройки с помощью командного меню: Edit > Preferences > Query Editor > Use UPPERCASE keywords on completion (рис. 1.40).

Протестируем полученный результат (рис. 1.41).

Попробуем выполнить аналогичный запрос (см. рис. 1.37), но без указания БД, обращаясь к этой таблице только по ее имени. И получим ошибку. Исправим в запросе обращение к таблице, начав с имени БД, и увидим, что автозаполнение срабатывает не только для ключевых слов SQL, но и для схемы БД: имени БД, таблиц, столбцов и т.д. Выполним корректный запрос повторно (рис. 1.42).

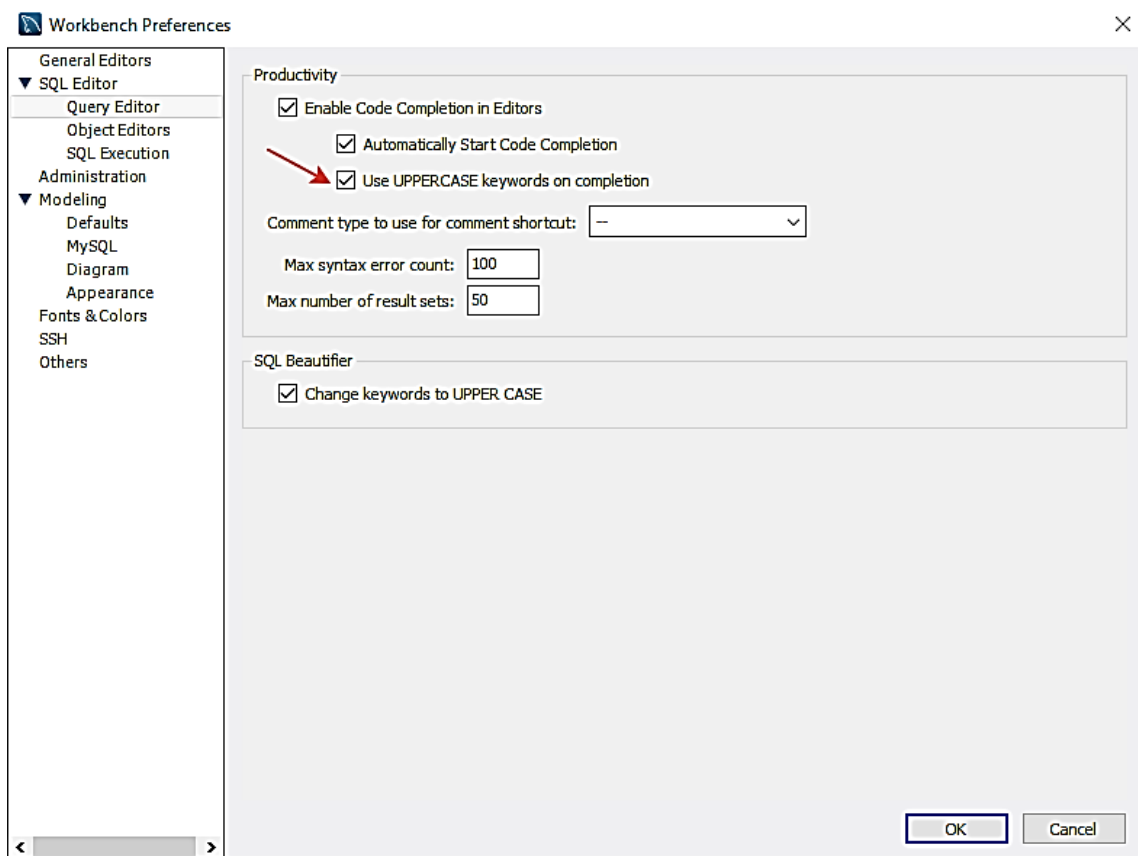


Рис. 1.40. Настройка верхнего регистра для подсказок и автозаполнения ключевых слов

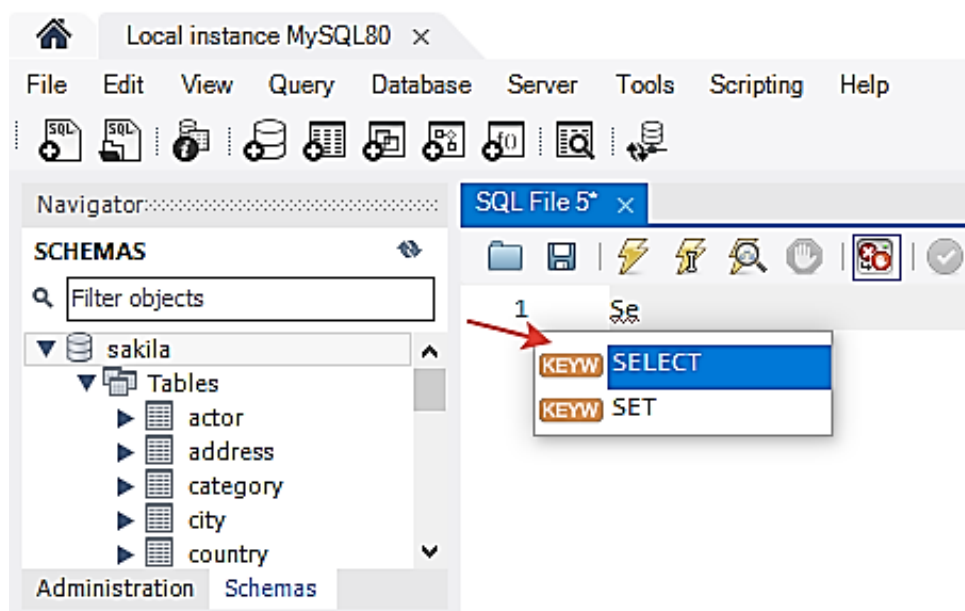


Рис. 1.41. Подсказки в верхнем регистре

Справа сверху (рис. 1.42) есть кнопки отключения ненужных вкладок – справоч по командам (SQL Additions), навигации (Navigator) и панели вывода (Action Output).

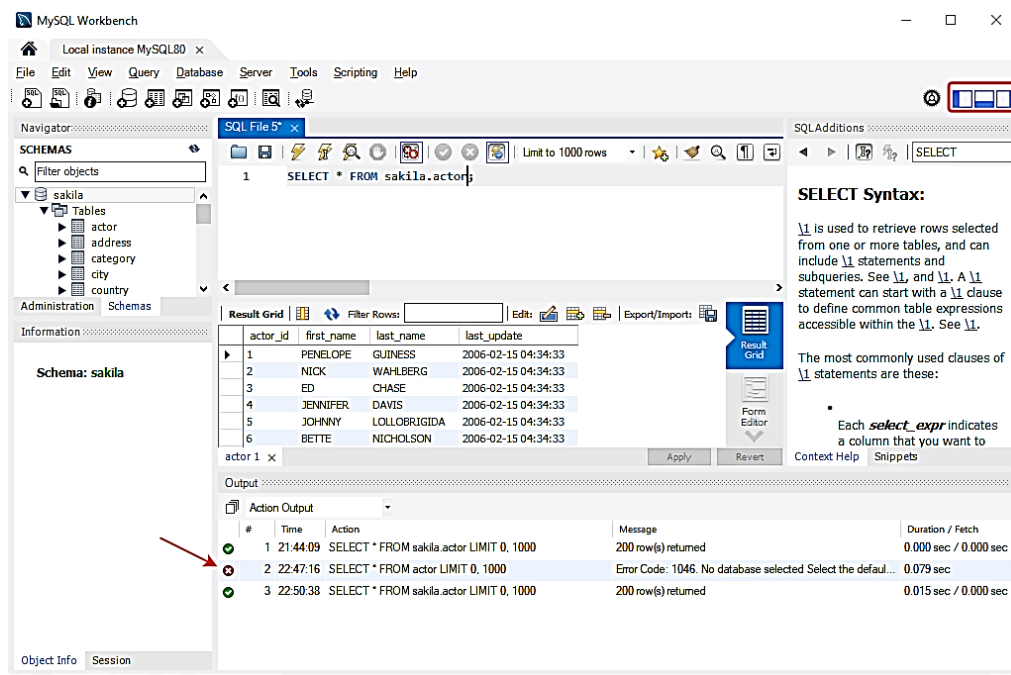


Рис. 1.42. Выполнение исправленного запроса

При подключении к серверу открывается дополнительная вкладка SQL-запроса (рис. 1.43, 1.44), включающая в себя область редактора запросов и панель инструментов. Введенные операторы можно сохранить в файл, также можно сохранить фрагмент кода для дальнейшего использования с помощью пиктограммы на панели инструментов (табл. 1.1, 12 пиктограмма панели инструментов).

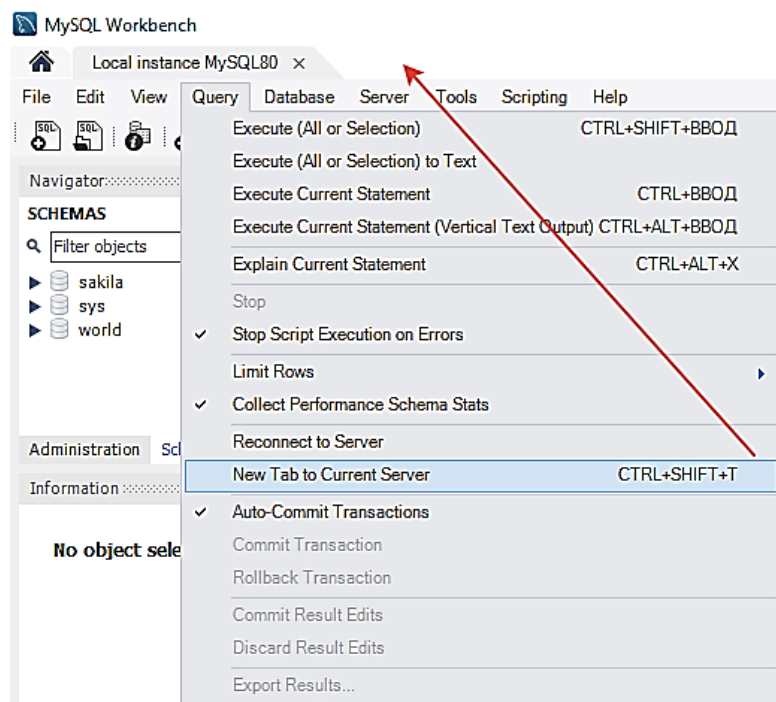


Рис. 1.43. Новая вкладка для текущего сервера

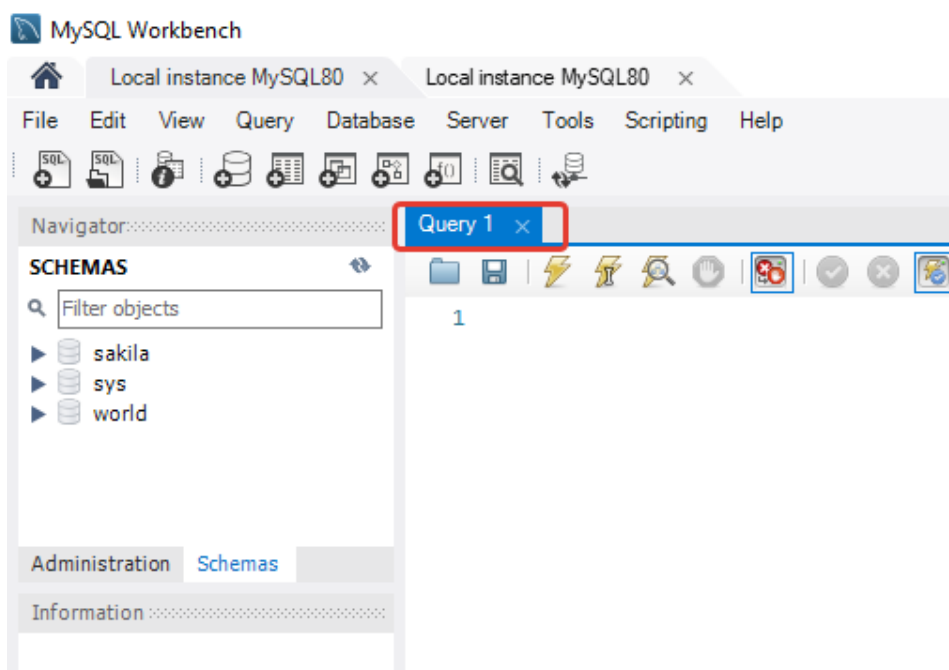
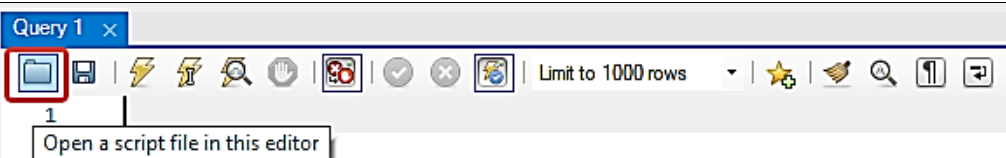
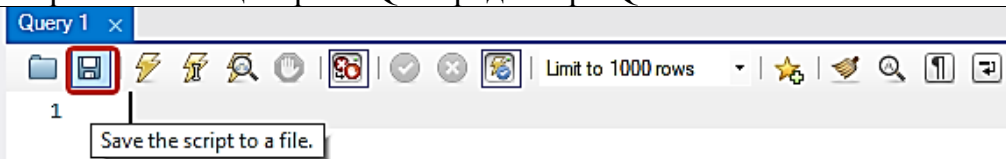
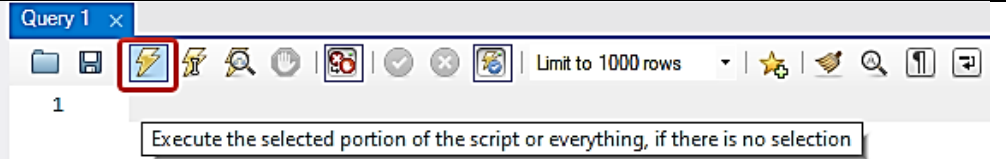


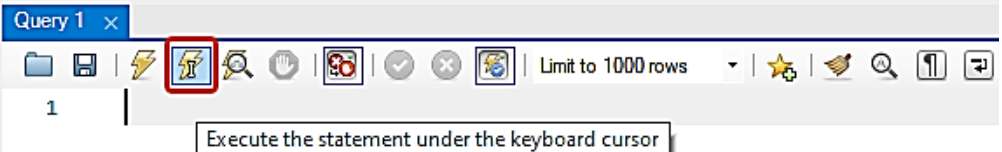
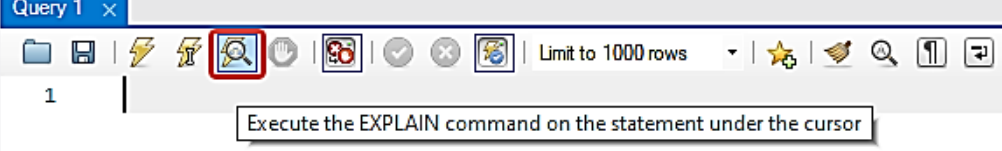
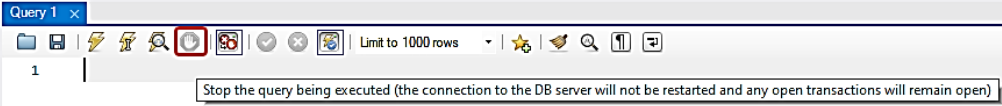
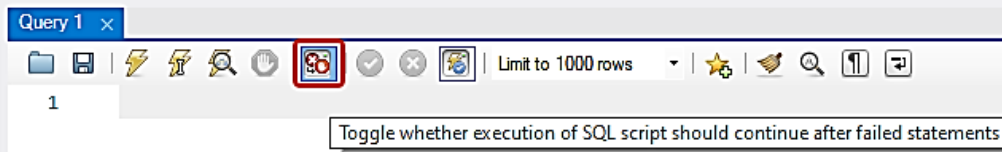
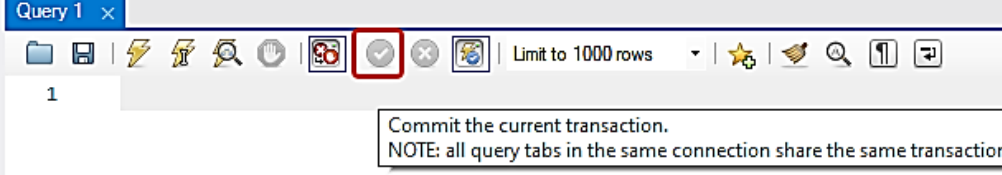
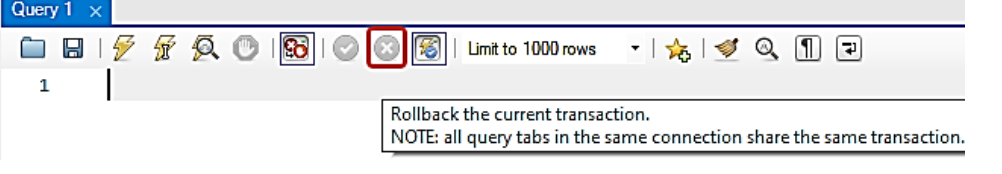
Рис. 1.44. Вкладка SQL-запроса

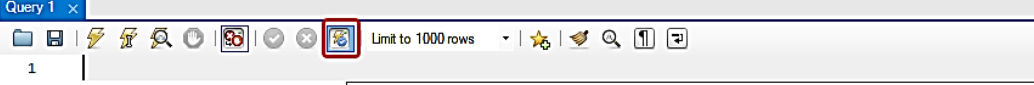
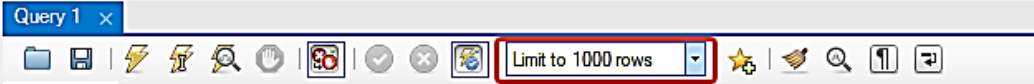
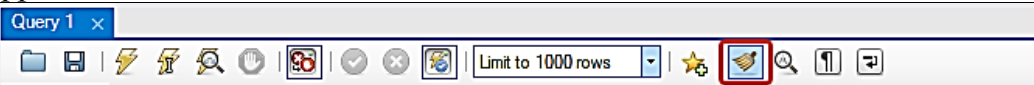
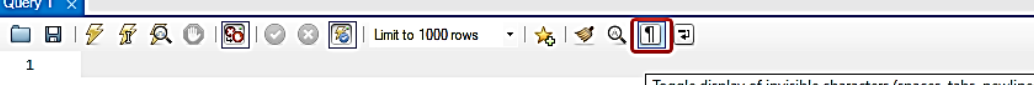
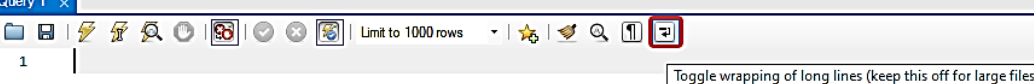
Все операции панели инструментов также представлены в табл. 1.1.

Таблица 1.1

Инструменты SQL-запросов

№ п/п	Пиктограмма панели инструментов и ее описание
1	 Открыть файл сценария в этом редакторе: загружает содержимое сохраненного сценария SQL в редактор SQL
2	 Сохранить сценарий в файл: позволяет сохранить текущее содержимое редактора SQL в файл
3	 Выполнить выбранную часть сценария или все, если выбор отсутствует: предоставляет простой способ выполнить весь запрос или его подмножество

№ п/п	Пиктограмма панели инструментов и ее описание
4	 Выполнить оператор под курсором клавиатуры: используется положение курсора клавиатуры для идентификации и выполнения запроса
5	 Выполнить команду EXPLAIN для оператора под курсором клавиатуры: используется положение курсора клавиатуры для идентификации запроса, а затем выполняется EXPLAIN
6	 Остановить выполнение запроса: останавливает выполнение текущего сценария SQL. Соединение с сервером базы данных не перезапускается, и все открытые транзакции остаются открытыми
7	 Переключение режимов после выявления ошибки: если переключатель неактивен, то выполнение сценария SQL остановится на команде, содержащей ошибку, отображается красный кружок (breakpoint, точка останова), если переключатель нажат, то выполнение продолжится и после ошибочной команды. По умолчанию переключатель активен
8	 Зафиксировать текущую транзакцию. Все вкладки запросов в одном соединении используют одни и те же транзакции. Чтобы иметь независимые транзакции, необходимо открыть новое соединение
9	 Откатить текущую транзакцию. Все вкладки запросов в одном соединении используют одни и те же транзакции. Чтобы иметь независимые транзакции, необходимо открыть новое соединение

№ п/п	Пиктограмма панели инструментов и ее описание
10	 Toggle autocommit mode. When enabled, each statement will be committed immediately. NOTE: all query tabs in the same connection share the same transaction. To have independent transactions, <i>Переключить режим автофиксации: если этот переключатель нажат, все транзакции фиксируются автоматически. Чтобы иметь независимые транзакции, необходимо открыть новое соединение</i>
11	 Set limit for number of rows returned by queries. <i>Установить ограничение на количество строк, возвращаемых запросами</i>
12	 Save current statement or selection to the snippet list. <i>Сохранить текущий оператор или выбранный фрагмент в списке фрагментов</i>
13	 Beautify/reformat the SQL script <i>Переформатировать сценарий SQL: по умолчанию ключевые слова SQL преобразуются в верхний регистр</i>
14	 Show the Find panel for the editor <i>Отобразить панель поиска для редактора</i>
15	 Toggle display of invisible characters (spaces, tabs, newlines) <i>Переключить отображение невидимых символов: если этот флажок установлен, отображаются невидимые символы, такие как символы новой строки, пробелы и табуляции</i>
16	 Toggle wrapping of long lines (keep this off for large files) <i>Переключить перенос длинных строк: если этот параметр выбран, длинные строки в редакторе SQL переносятся, чтобы исключить необходимость прокрутки</i>

Воспользуемся 13-й кнопкой панели инструментов для форматирования кода (см. табл. 1.1). Для этого намеренно наберем ключевые слова в нижнем регистре (рис. 1.45), после чего используем операцию форматирования кода. Полученный результат на рис. 1.46.

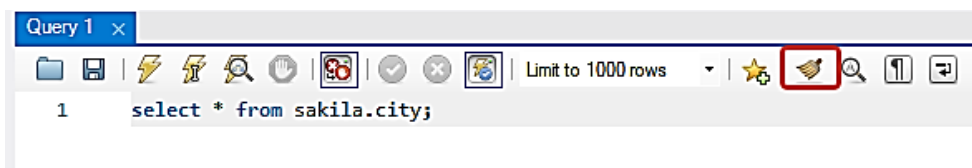


Рис. 1.45. Запрос с ключевыми словами в нижнем регистре

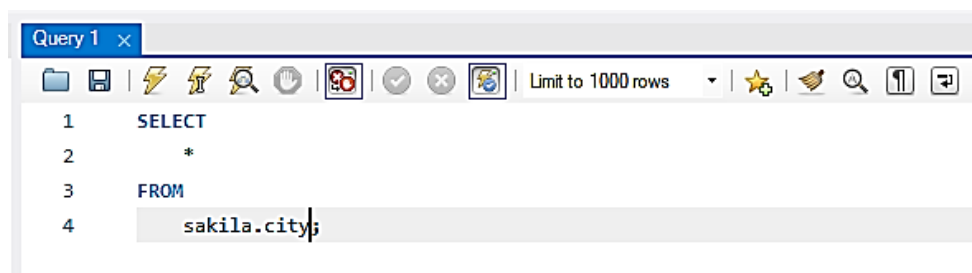


Рис. 1.46. Результат форматирования кода

Рассмотрим также командное меню Query (табл. 1.2) и Edit (табл. 1.3). Меню Query (рис. 1.47) содержит основные команды для работы с запросами.

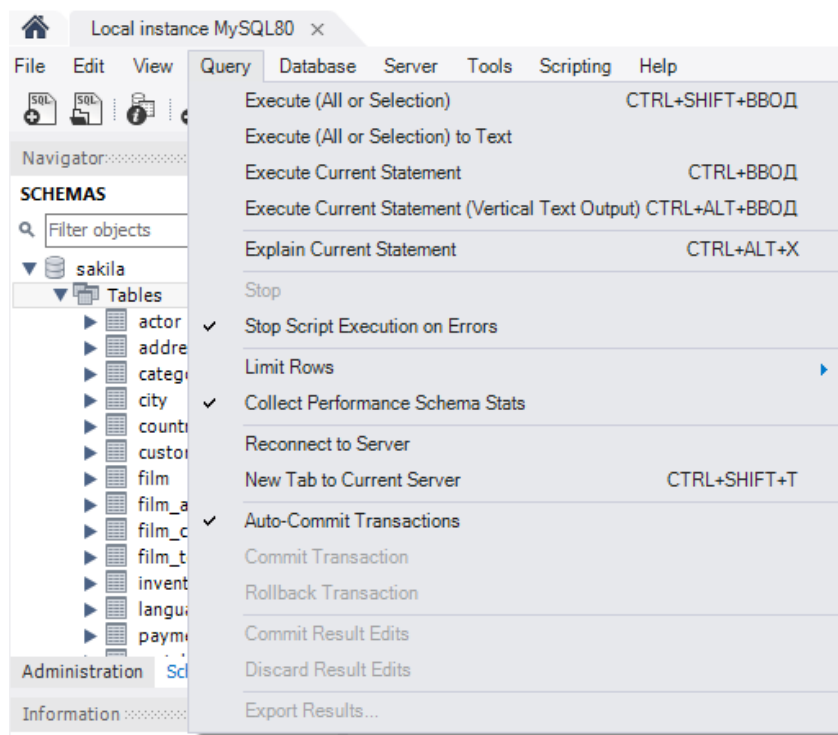


Рис. 1.47. Меню Query

Меню Query

Команда	Описание
Execute (All or Selection)	<i>Выполнить (все или выбор)</i> : выполняет все инструкции в области SQL-запросов или только выбранные инструкции
Execute (All or Selection) to Text	<i>Выполнить (все или выбор) в тексте</i> : выполняет все операторы в области SQL-запроса или только выбранные операторы и отображает их в виде обычного текста, как стандартная консоль командной строки MySQL
Execute Current Statement	<i>Выполнить текущий оператор</i> : выполняет текущий оператор SQL
Execute Current Statement (Vertical Text Output)	<i>Выполнить текущий оператор (вертикальный текстовый вывод)</i> : выполняет все операторы в области SQL-запроса или только выбранные операторы и отображает их в виде обычного текста
Explain Current Statement	<i>Объяснение текущего оператора</i> : описывает текущий оператор с помощью оператора MySQL EXPLAIN
Visual Explain Current Statement	<i>Визуальное объяснение текущего утверждения</i> : MySQL Workbench анализирует выходные данные EXPLAIN (JSON) с сервера MySQL и выводит визуальное представление
Stop	<i>Стоп</i> : останавливает выполнение текущего сценария
Stop Script Execution On Errors	<i>Остановить выполнение сценария при ошибках</i> : если этот параметр включен, MySQL Workbench прекращает выполнение запроса, если обнаружены ошибки
Limit Rows	<i>Ограничить количество строк</i> : по умолчанию количество возвращаемых строк (LIMIT) равно 1000, число варьируется от 10 до 50 000 и есть опция «Не ограничивать»
Collect Performance Schema Stats	<i>Сбор статистики схемы производительности</i> : предоставляет данные в представление набора результатов статистики запросов, которое включает в себя информацию, специфичную для оператора, о времени, обработанных строках, временных таблицах, соединениях по типам, сортировке и использовании индекса
Collect Resultset Field Metadata	<i>Сбор метаданных поля набора результатов</i> : предоставляет данные для представлений набора результатов редактора форм и типов полей

Команда	Описание
Reconnect to Server	<i>Повторное подключение к серверу:</i> повторное подключение к серверу MySQL
New Tab to Current Server	Новая вкладка на текущем сервере: создает дубликат текущей вкладки редактора SQL
Auto-Commit Transactions	<i>Автоматическое подтверждение транзакций:</i> включает автоматическое подтверждение транзакций
Commit Transaction	<i>Зафиксировать транзакцию:</i> фиксирует транзакцию базы данных
Rollback Transaction	<i>Откат транзакции:</i> откатывает транзакцию базы данных
Commit Result Edits	<i>Зафиксировать изменения результатов:</i> фиксирует любые изменения
Discard Result Edits	<i>Отменить изменения результатов:</i> отменяет все внесенные вами изменения
Export Results	<i>Экспортировать результаты:</i> экспортирует наборы результатов в файл

Таблица 1.3

Меню Edit подменю Format

Команда	Описание
Beautify Query	<i>Украсить запрос:</i> переформатирует запрос, выбранный на вкладке запросов, и размещает его с красивыми отступами
UPCASE Keywords	<i>Ключевые слова UPCASE:</i> преобразует ключевые слова в верхний регистр в текущем запросе на вкладке запросов
lowercase Keywords	<i>Ключевые слова в нижнем регистре:</i> преобразует ключевые слова в нижний регистр в текущем запросе на вкладке запросов
Un/Comment Selection	<i>Выбор без/с комментариями:</i> комментирует строки, выбранные в данный момент на вкладке запроса. Если строки уже прокомментированы, эта операция удаляет комментарии
Auto-complete	<i>Автозаполнение:</i> запускает мастер автозаполнения (по умолчанию включено)

Меню Edit содержит подменю Format (рис. 1.48), включающее команды для работы с запросом.

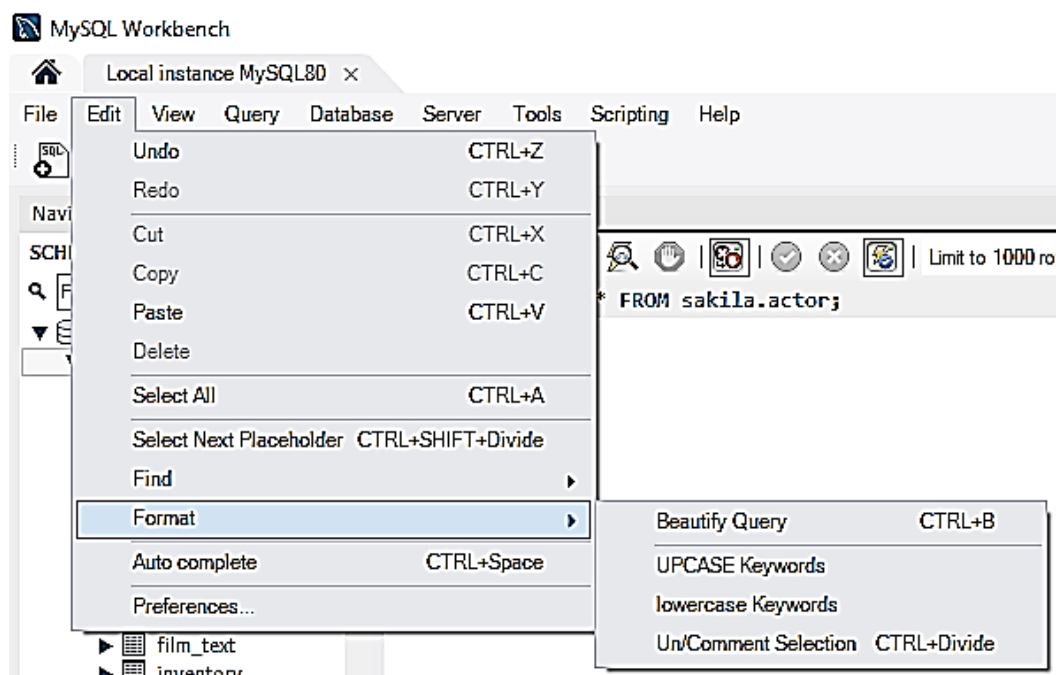


Рис. 1.48. Меню Edit подменю Format

С каждым объектом связано контекстное меню, которое вызывается щелчком правой кнопки мыши на объекте. Так, для работы с таблицей результатов запроса можно вызвать контекстное меню, щелкнув правой кнопкой мыши на любой записи (рис. 1.49).

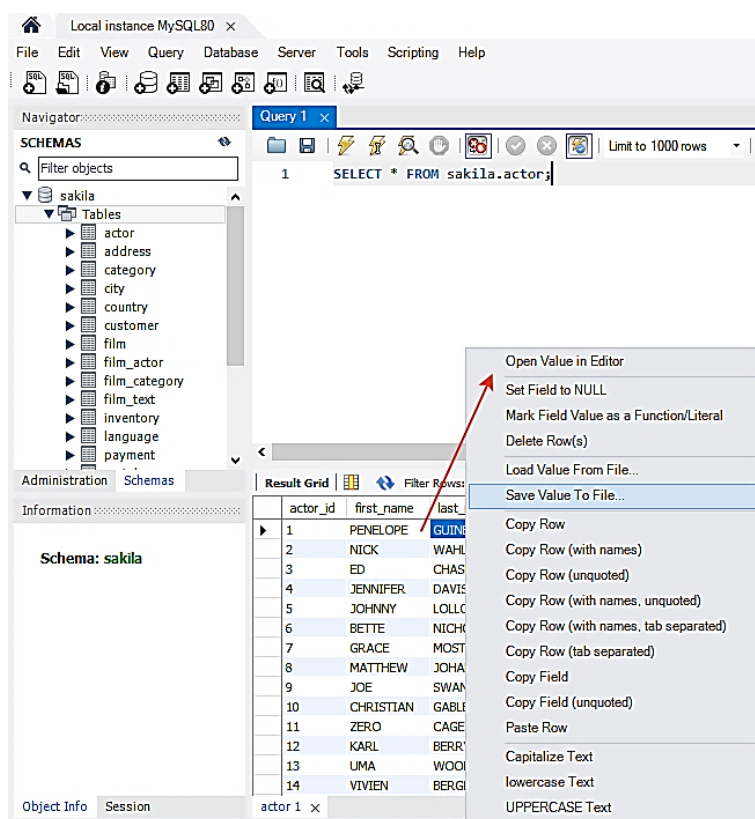


Рис. 1.49. Контекстное меню для Result Grid

А с таблицей БД, щелкнув правой кнопкой мыши на любой таблице (рис. 1.50).

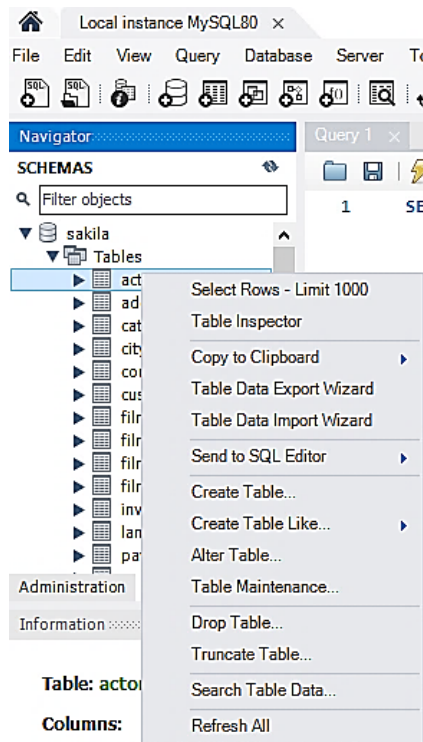


Рис. 1.50. Контекстное меню для таблиц БД

Создадим новую БД health, воспользовавшись панелью инструментов (рис. 1.51).

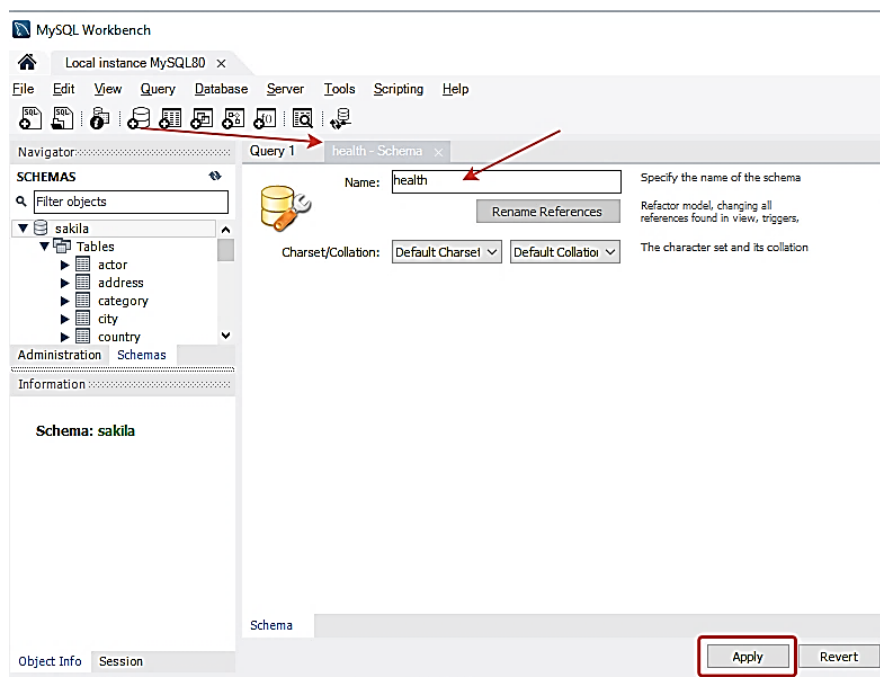


Рис. 1.51. Создание БД health

Запустится мастер для создания БД, генерирующий код (рис. 1.52, 1.53).

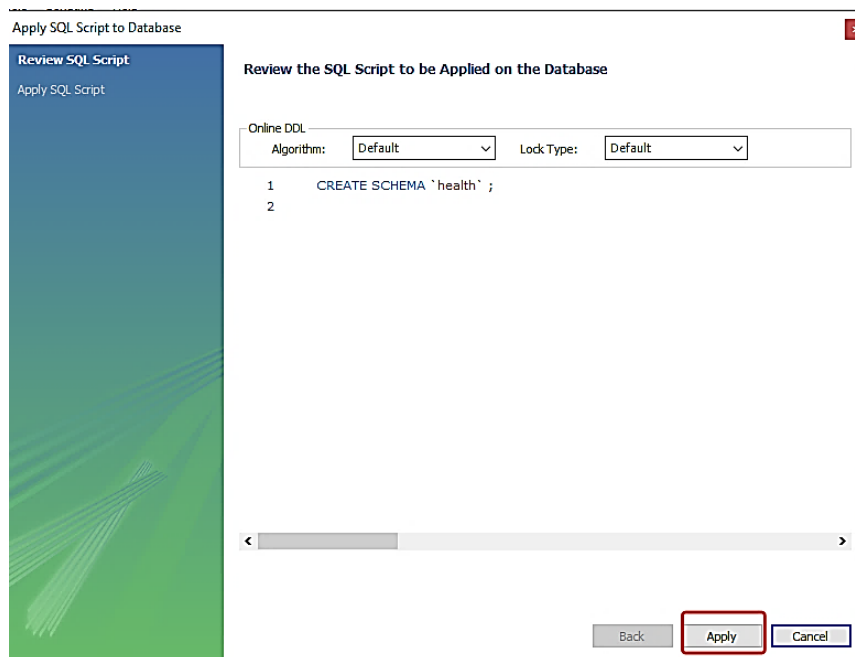


Рис. 1.52. Скрипт создания БД

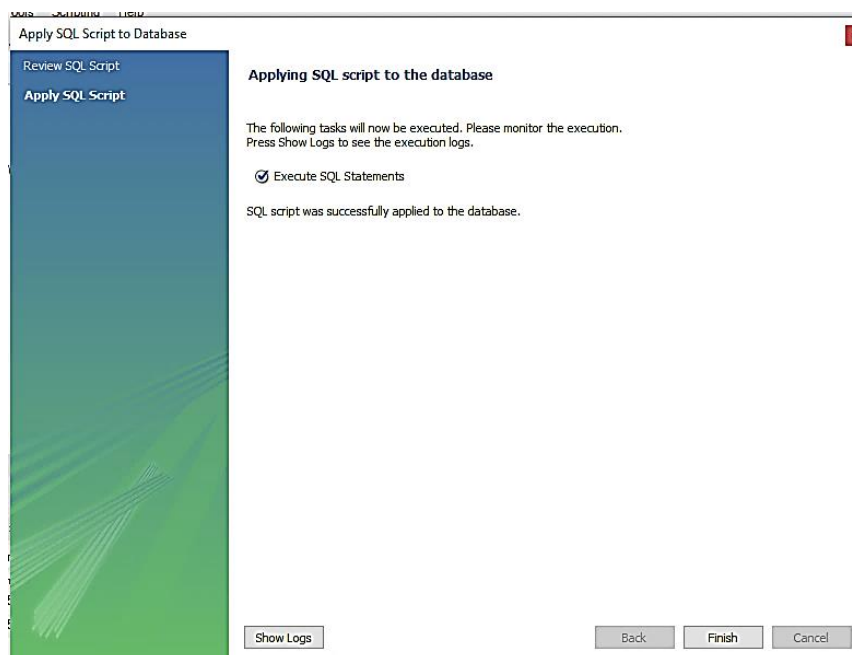


Рис. 1.53. Применение (запуск) скрипта создания БД

В созданной базе health создадим таблицу insurance на основе импорта файла с данными. Данные: <https://www.kaggle.com/mirichoi0218/insurance> – расходы на медицинское обслуживание тех, кто имеет медицинскую страховку.

Описание переменных набора:

- age: возраст основного бенефициара;
- sex: пол застрахованного;
- bmi: индекс массы тела;
- children: число детей, охваченных медицинским страхованием / число иждивенцев;
- smoker: курит ли застрахованный;
- region: жилой район получателя;
- charges: индивидуальные медицинские расходы, оплачиваемые страховкой.

Вызовем контекстное меню для БД health и выберем опцию Table Data Import Wizard (рис. 1.54).

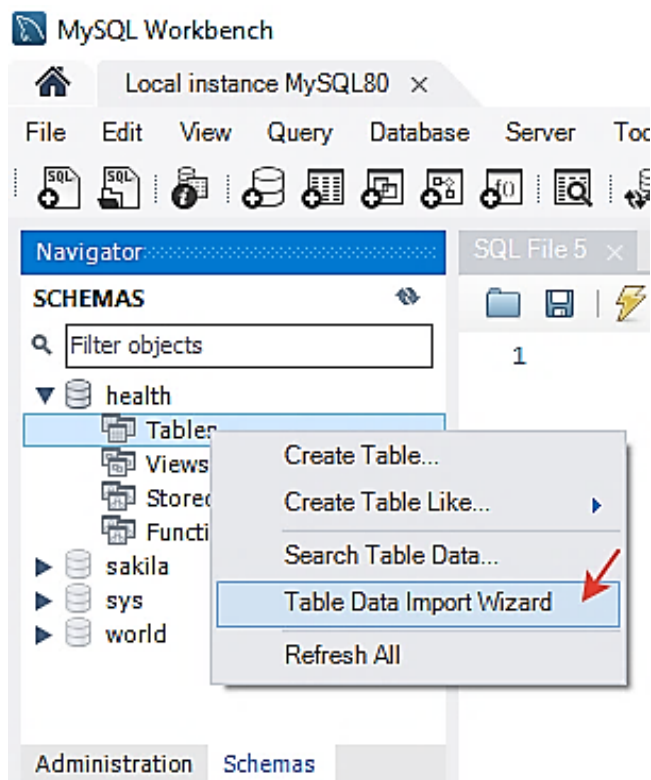


Рис. 1.54. Запуск Мастера импорта данных таблицы

Запустится мастер импорта данных, в котором необходимо выбрать: местоположение файла (рис. 1.55), таблицу, в которую будут импортированы данные (рис. 1.56), принять или скорректировать настройки импорта (рис. 1.57) и выполнить импорт (рис. 1.58). Важно, что, хотя файл содержит менее полутора тысяч записей, импорт происходил крайне медленно (рис. 1.59, 1.60). К этому нужно быть готовым или пользоваться другими инструментами.

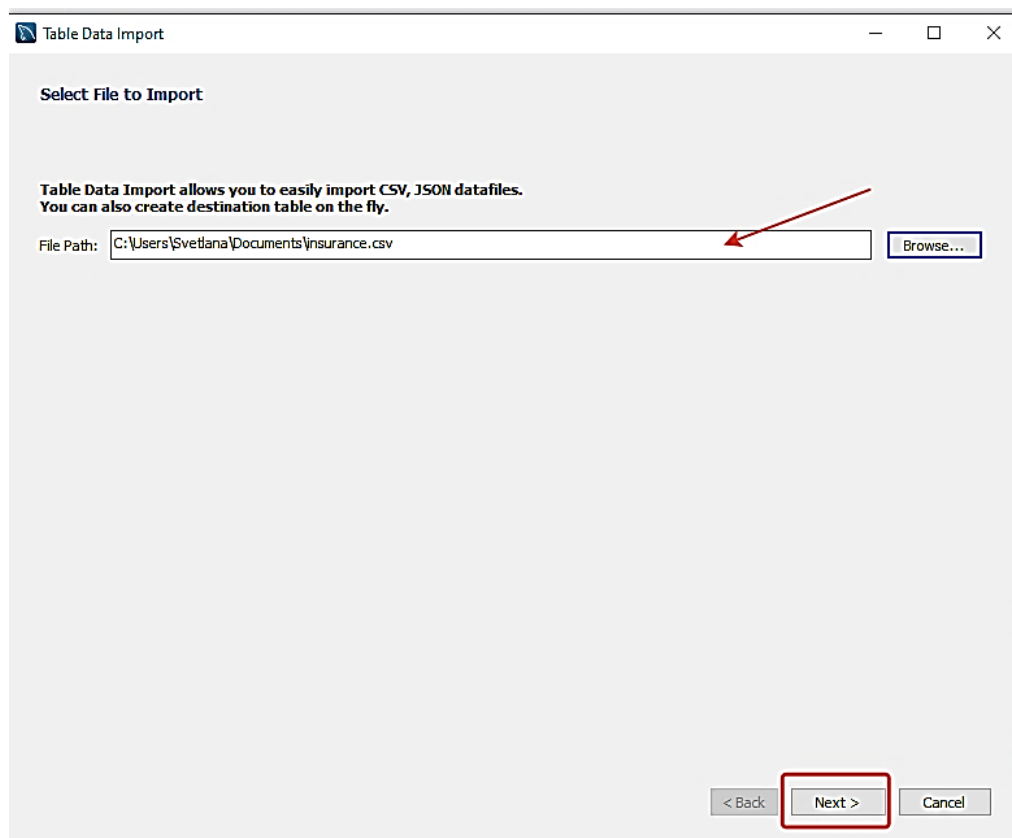


Рис. 1.55. Выбор файла для импорта данных

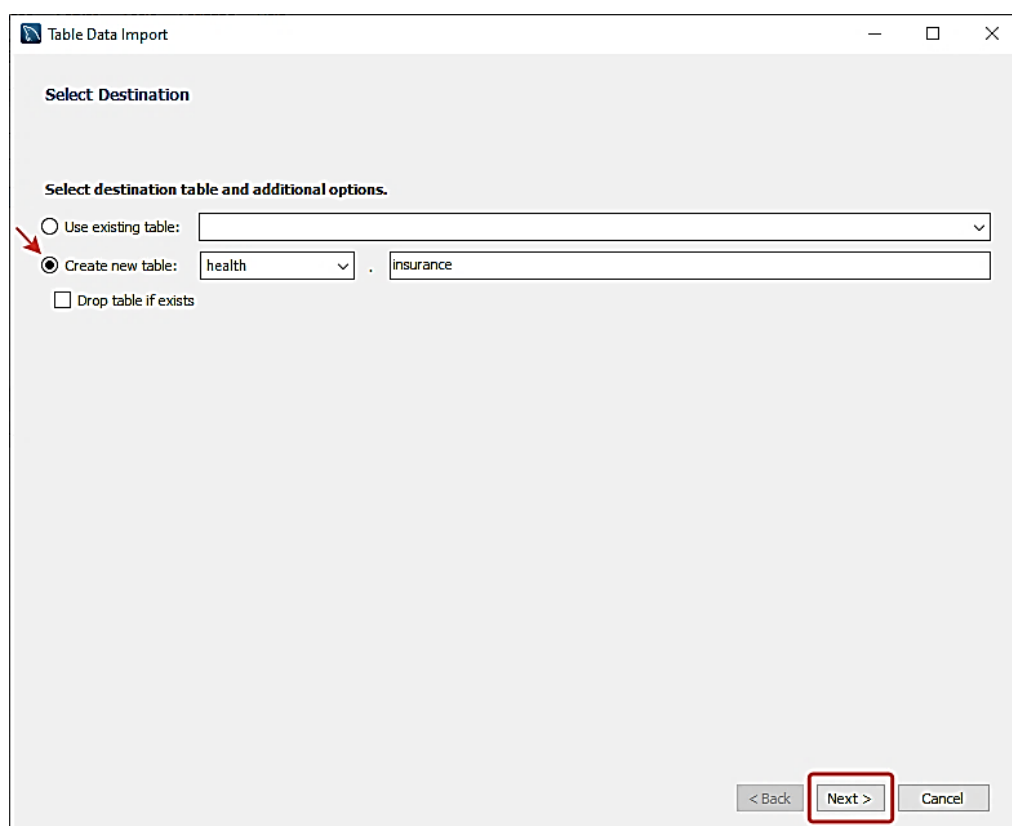


Рис. 1.56. Выбор или создание таблицы, в которую будут импортированы данные

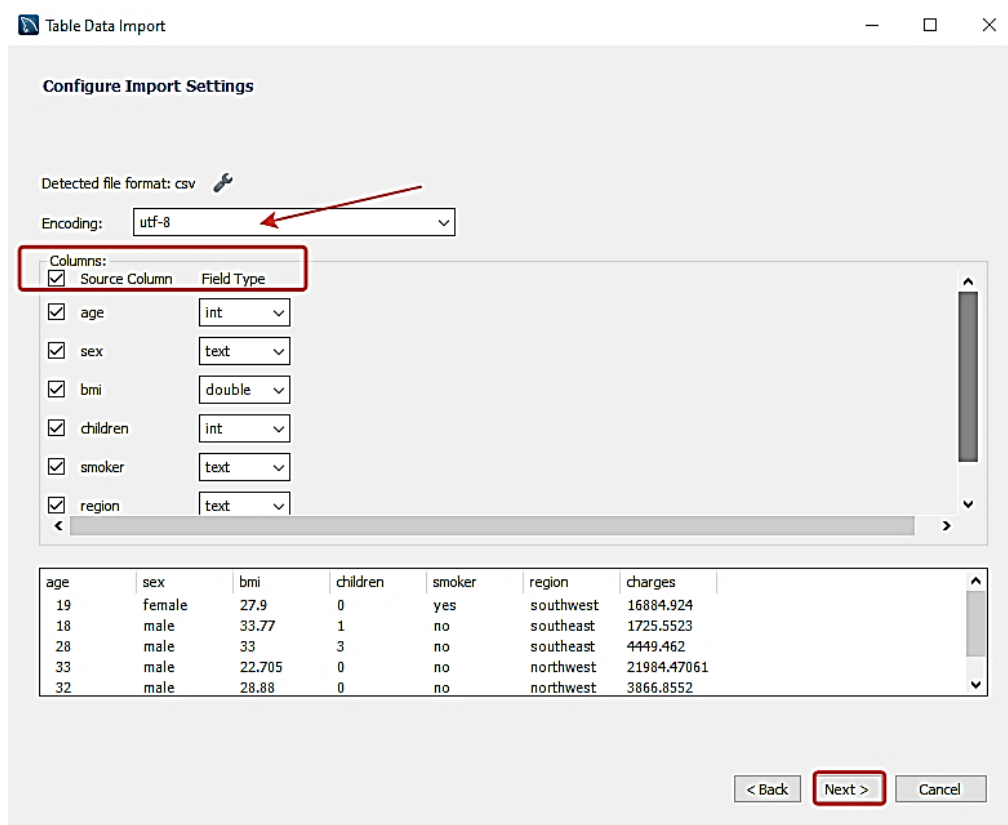


Рис. 1.57. Настройки импорта (имена столбцов, если в файле есть строка заголовков, тип данных, кодировка)

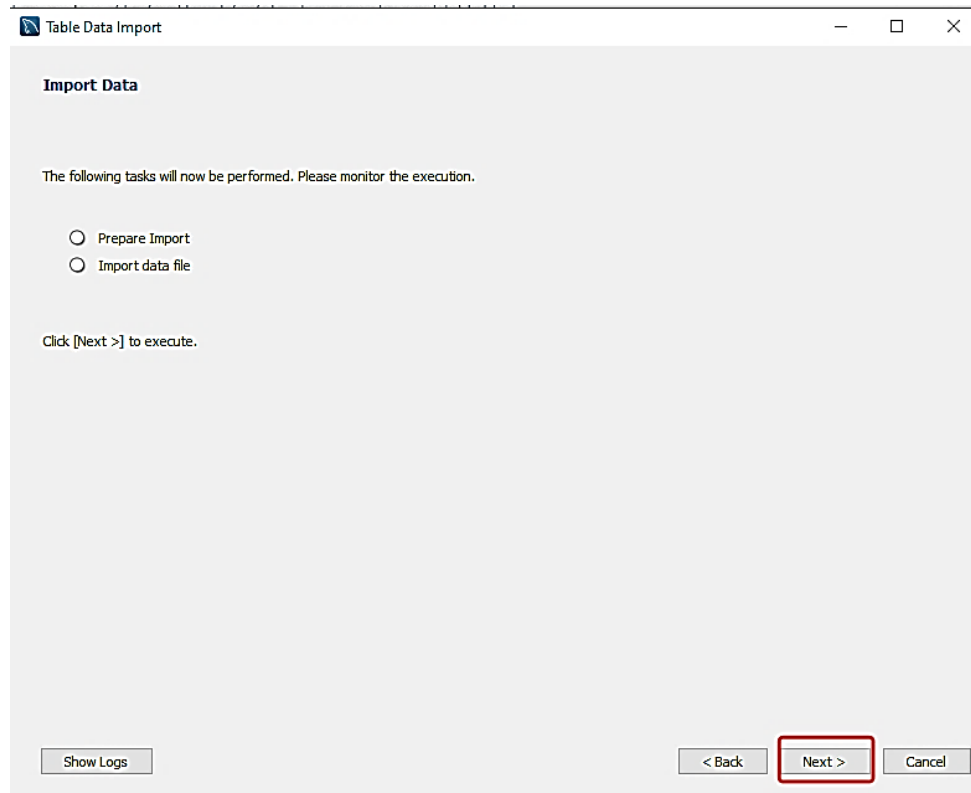


Рис. 1.58. Выполнение импорта данных

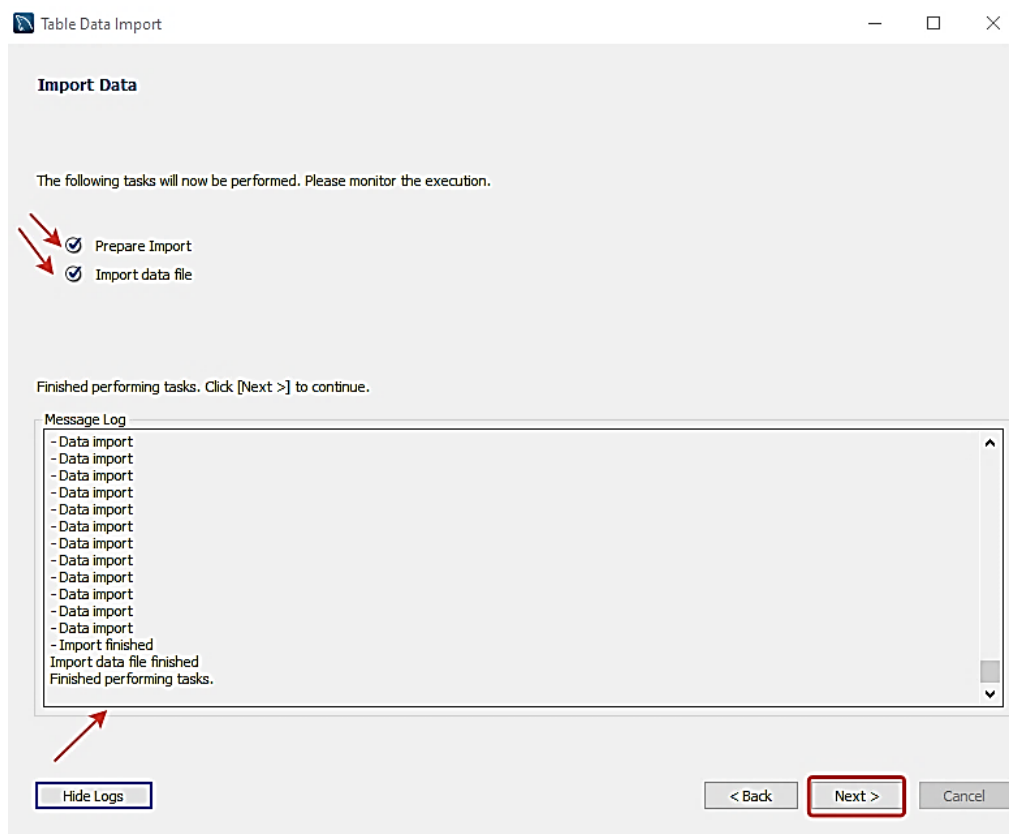


Рис. 1.59. Сводка по импорту данных (логи)

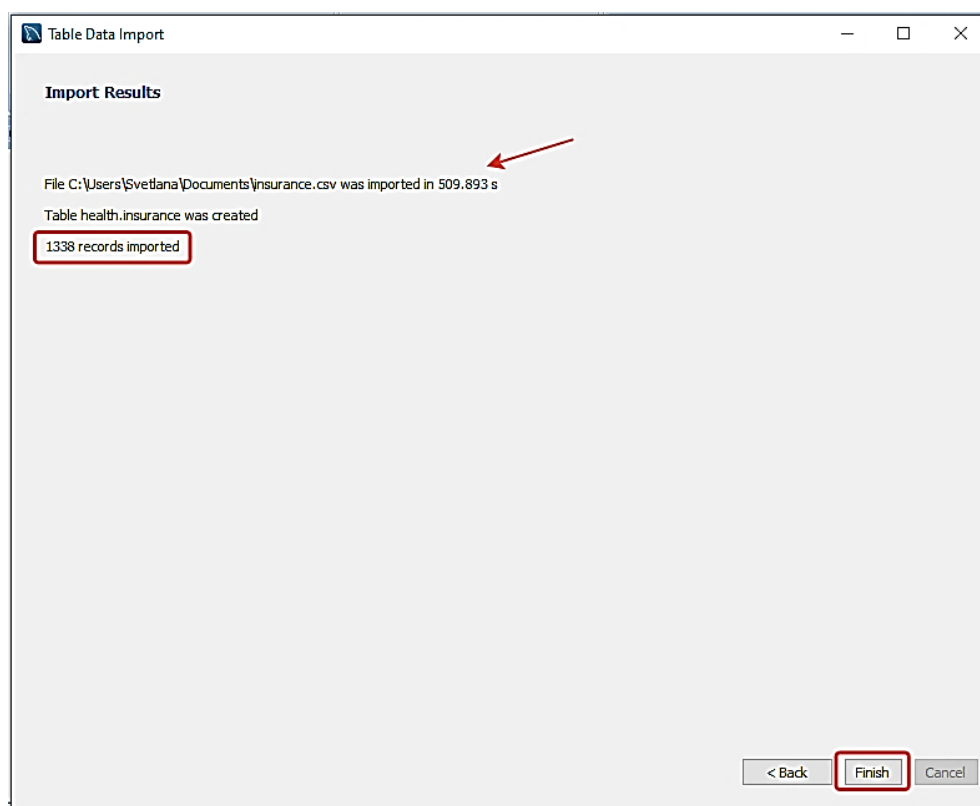


Рис. 1.60. Сводка по импорту данных
(время и количество импортированных записей)

Обновим панель навигации и раскроем перечень таблиц в БД health (рис. 1.61) и выведем Inspector для таблицы insurance (рис. 1.62). На вкладке DDL представлен код, который был сгенерирован приложением для создания таблицы.

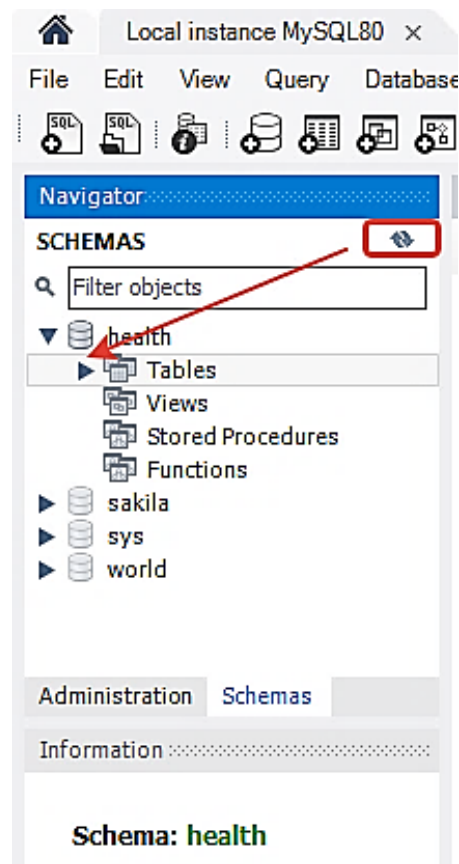


Рис. 1.61. Доступ к созданной таблице insurance

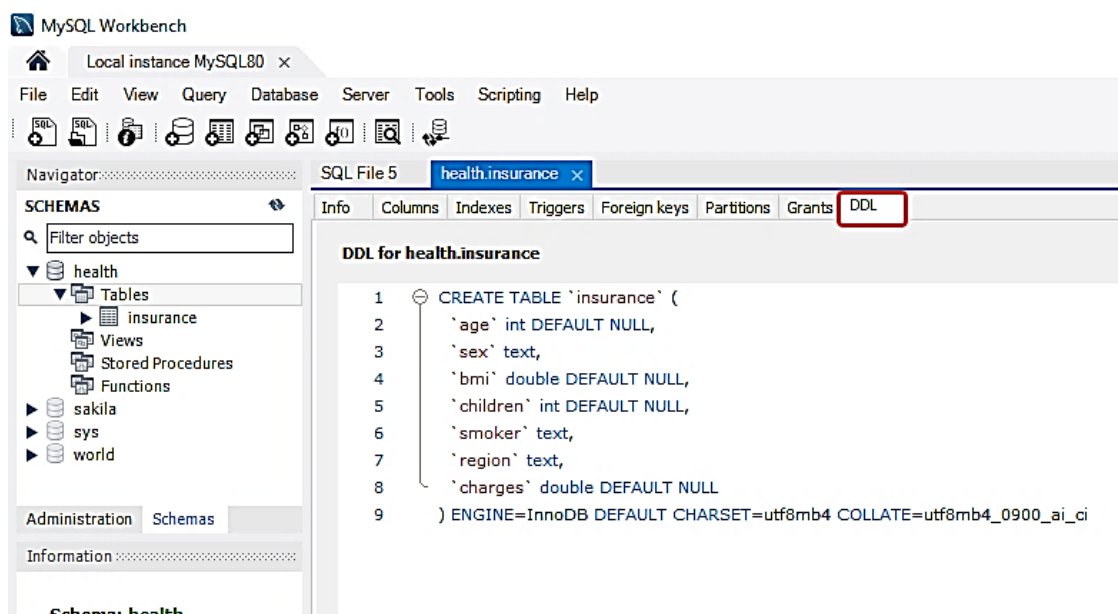


Рис. 1.62. Inspector для таблицы insurance

Выведем также на экран содержимое таблицы insurance (рис. 1.63).

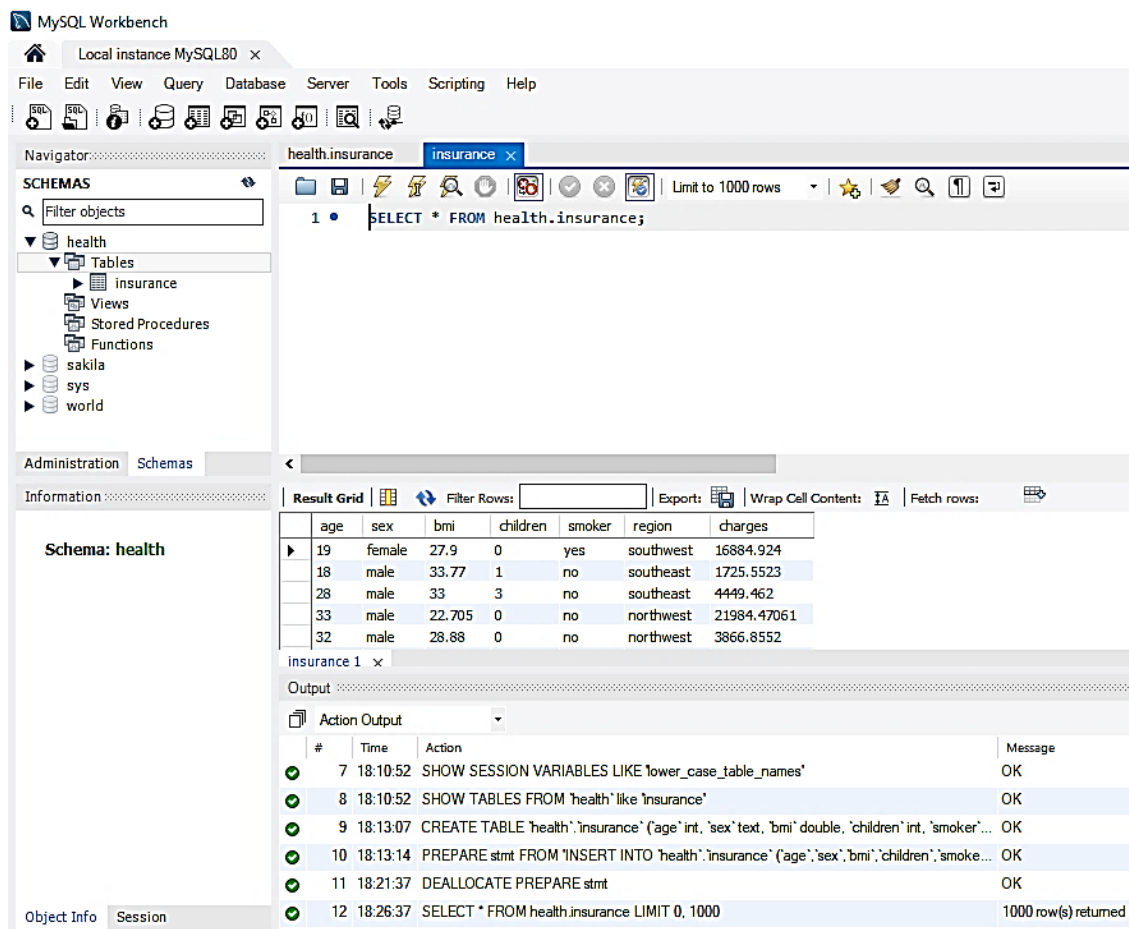


Рис. 1.63. Запрос данных таблицы insurance

Рассмотрим далее возможности MySQL Workbench для проектирования БД.

1.3. Графический клиент MySQL Workbench: Проектирование и моделирование баз данных

Моделирование упрощает проектирование и обслуживание БД, позволяя визуализировать требования.

Создадим с помощью MySQL Workbench модель уже имеющейся БД sakila, этот процесс называется обратным проектированием. Для этого вернемся на главную страницу и перейдем в меню слева на страницу создания моделей (рис. 1.64).

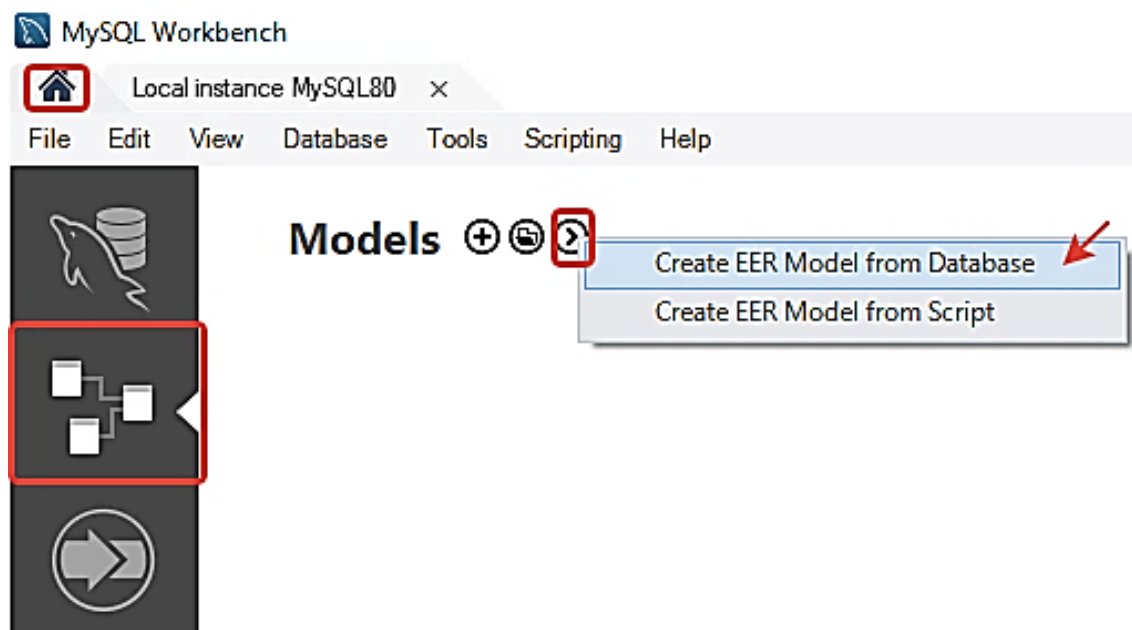


Рис. 1.64. Создание модели БД

Диалоговые окна по созданию модели включают следующие шаги:

- Connection Options – Параметры подключения (рис. 1.65);
- Connect to DBMS – Подключение к СУБД;
- Select Schemas – Выбор схем (рис. 1.66);
- Retrieve Objects – Извлечение объектов;
- Select Objects – Выбор объектов (рис. 1.67);
- Reverse Engineer – Обратное проектирование.
- Result – Результат (рис. 1.68).

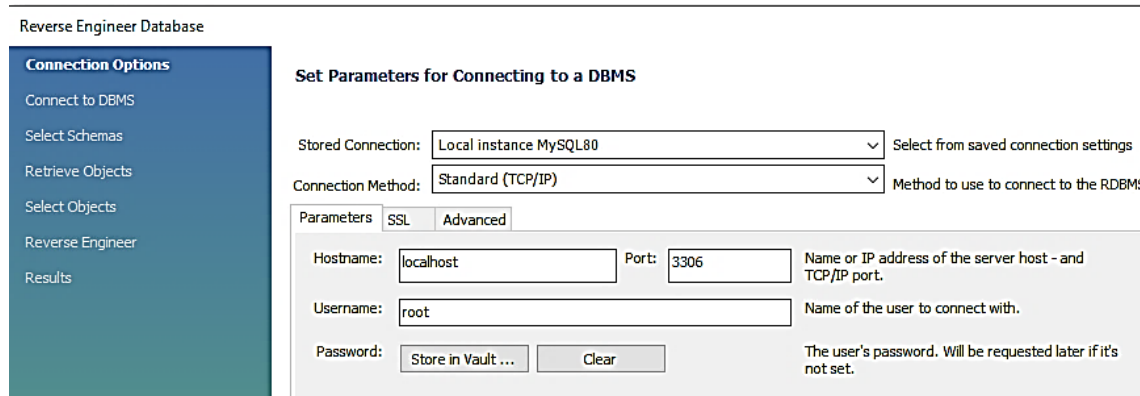


Рис. 1.65. Параметры подключения

После выполнения проектирования созданная модель откроется в новой вкладке MySQL Model (рис. 1.69). Звездочка после имени вкладки означает, что есть несохраненные изменения.

Вкладка MySQL Model включает следующие разделы:

- Расширенные диаграммы «сущность-связь» (EER);
- Физические схемы;
- Привилегии схемы;
- SQL-скрипты;
- Примечания к модели.

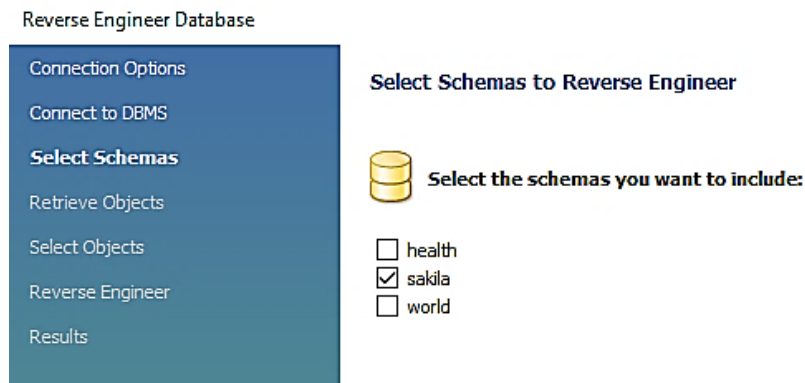


Рис. 1.66. Выбор схем, имеющих в текущем подключении БД

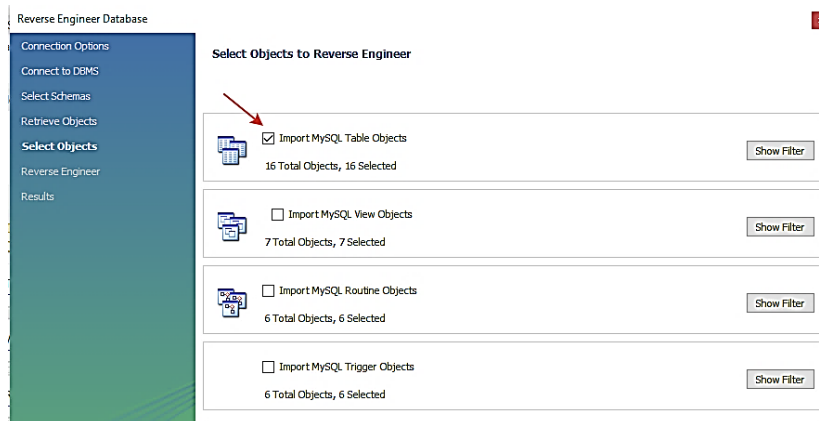


Рис. 1.67. Выбор объектов для проектирования модели

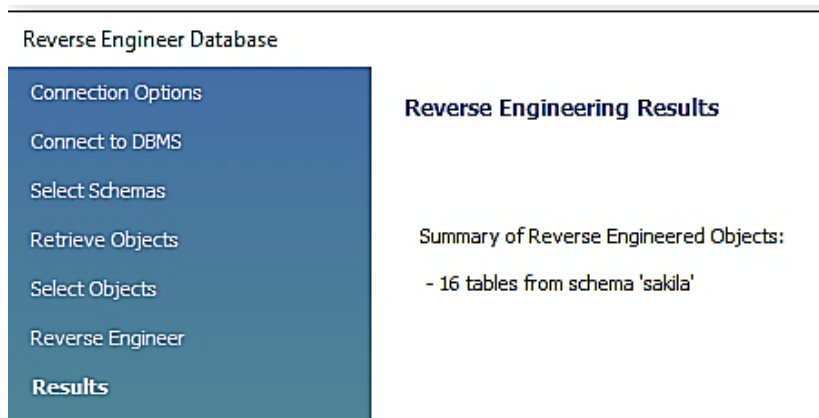


Рис. 1.68. Результат проектирования

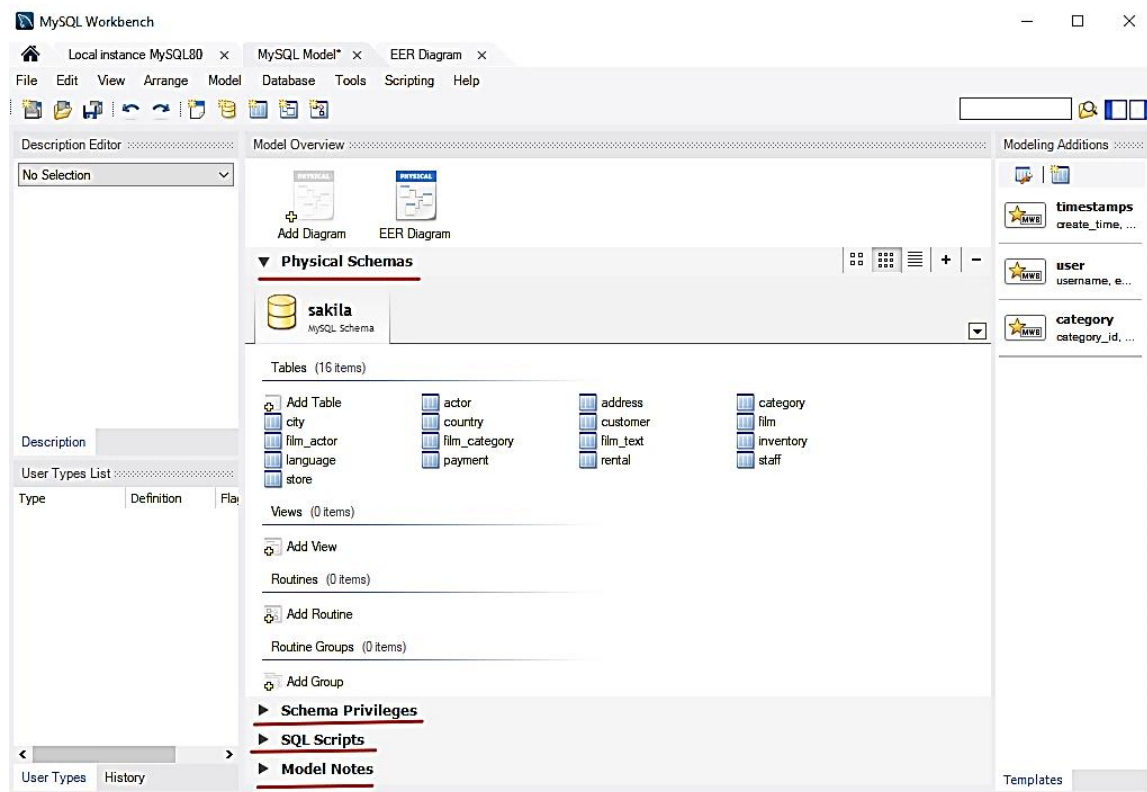


Рис. 1.69. Вкладка MySQL Model

Открывается дополнительная вкладка EER Diagram (рис. 1.70).

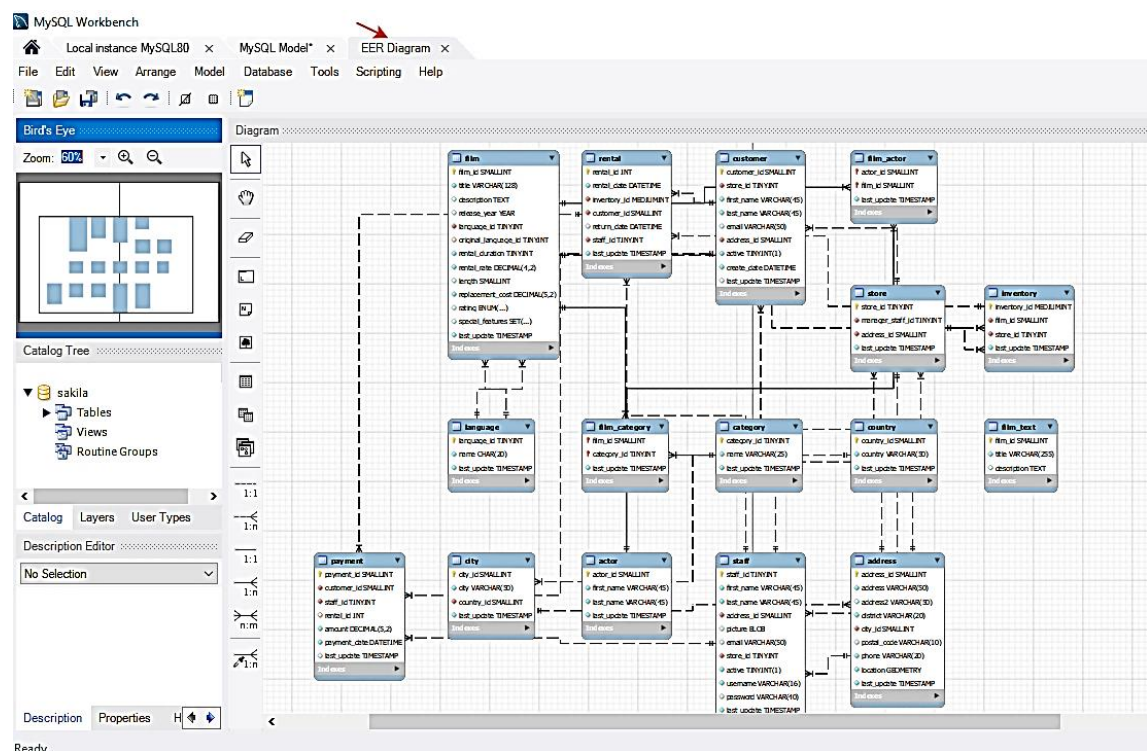


Рис. 1.70. Вкладка EER Diagram для созданной модели

Диаграммы Enhanced Entity-Relationship (EER, расширенная связь между сущностями) являются частью интерфейса моделирования в MySQL Workbench и обеспечивают визуальное представление связей между таблицами в модели.

Сохраним созданную модель в папку на диске (у всех моделей расширение файлов .mwb) с именем sakila (рис. 1.71).

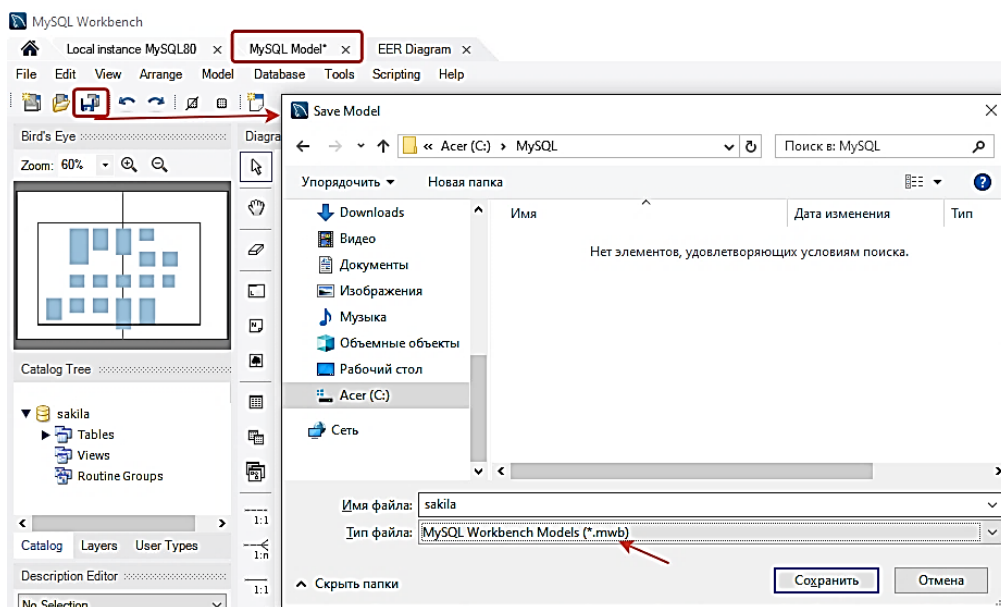


Рис. 1.71. Сохранение модели

Теперь эту модель можно увидеть на главной странице моделей MySQL Workbench (рис. 1.72).

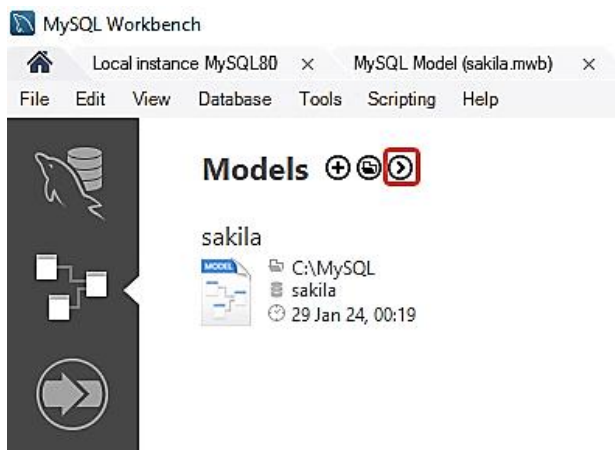


Рис. 1.72. Отображение созданной и сохраненной модели на главной странице

Аналогично создадим через обратное проектирование и модель для БД world.

Для модели sakila.mwb добавим еще одну диаграмму (рис. 1.73). Можно для создания новой диаграммы использовать не только панель инструментов, но и виджет Add Diagram на вкладке MySQL Model (см. рис. 1.69).

Разместим в рабочей области три таблицы: film, film_actor и language. При наведении курсора мыши на таблицу film, она окажется в фокусе и выделятся все связи этой таблицы. Одновременно будут подсвечены поля, по которым была установлена связь.

С таблицей language у таблицы film две связи, а с таблицей film_actor – одна (рис. 1.73).

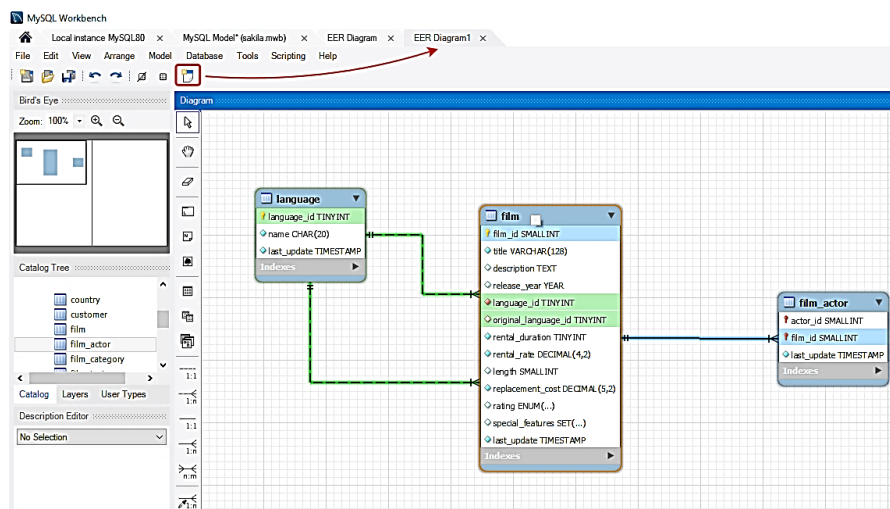


Рис. 1.73. Создание новой диаграммы

Нотацию для диаграммы можно изменить: Model > Relationship Notation (рис. 1.74, 1.75).

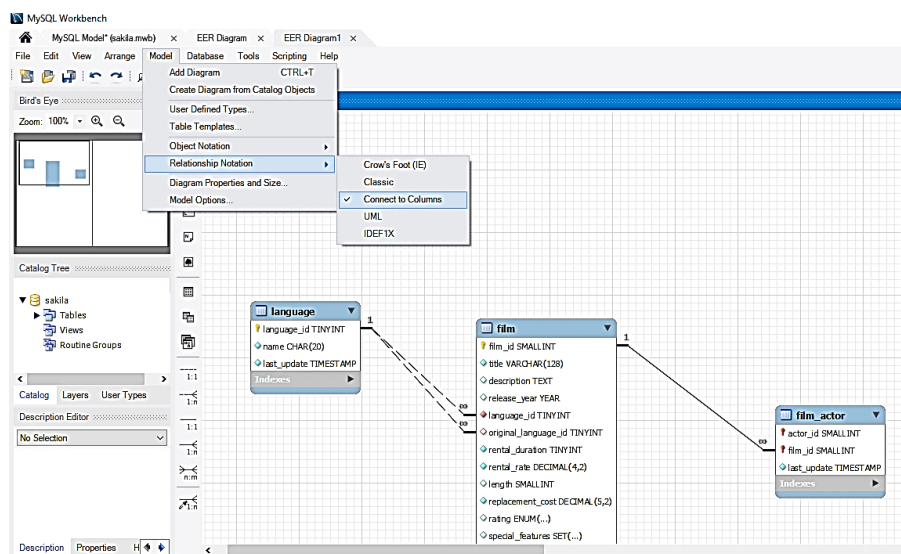


Рис. 1.74. Диаграмма в нотации Connect to Columns

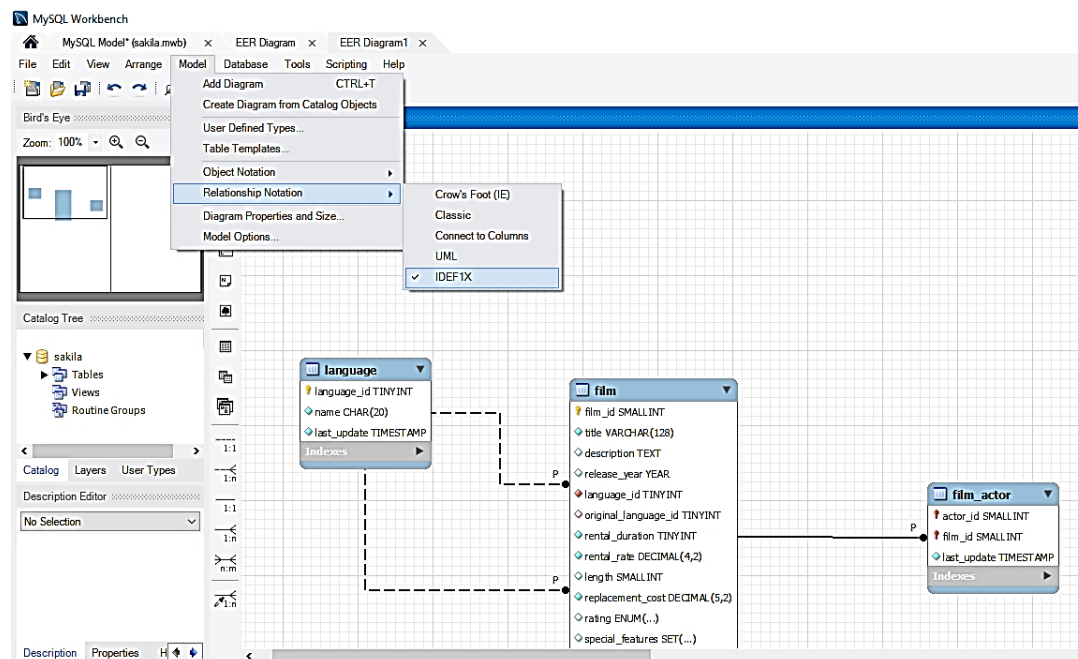


Рис. 1.75. Диаграмма в нотации IDEF1X

Создадим новую модель БД retail. При проектировании будем опираться на реальные данные. Скачаем два связанных файла данных с ресурса kaggle: данные о транзакциях клиентов и данные реакции клиентов на рекламные взаимодействия (https://www.kaggle.com/datasets/regivm/retailtransactiondata?select=Retail_Data_Transactions.csv и https://www.kaggle.com/datasets/regivm/retailtransactiondata?select=Retail_Data_Response.csv).

Файл Retail_Data_Response.csv содержит следующие данные:

- customer_id – идентификатор клиента (текстовая/символьная переменная);
- response – наличие/отсутствие отклика на рекламную компанию (бинарная переменная отклика).

Файл Retail_Data_Transactions.csv содержит следующие данные:

- customer_id – идентификатор клиента (текстовая/символьная переменная);
- trans_date – дата транзакции;
- tran_amount – сумма покупки.

В файле Retail_Data_Response.csv поле customer_id – это первичный ключ (PRIMARY KEY), все значения в этом поле ненулевые и уникальные. В Retail_Data_Transactions.csv первичного ключа нет, так как данные каждого из трех полей могут повторяться, а поле customer_id выступает внешним ключом (FOREIGN KEY), так как идентификаторы клиентов, совершающих транзакции, те же, что и в файле Retail_Data_Response.csv.

На главной странице запустим создание новой модели (рис. 1.76).

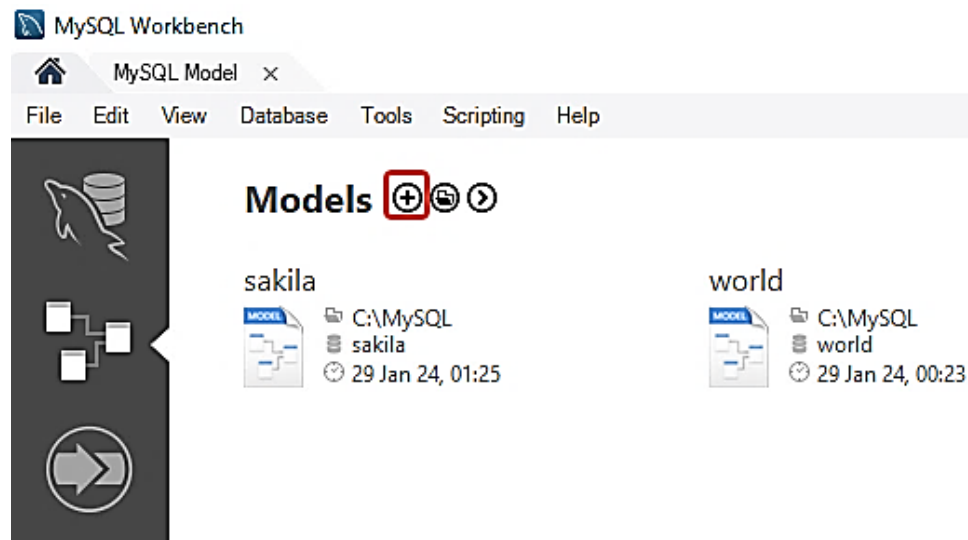


Рис. 1.76. Создание новой модели

Изменим название БД (рис. 1.77) на retail и добавим две таблицы transactions и response с указанными столбцами (рис. 1.78, 1.79).

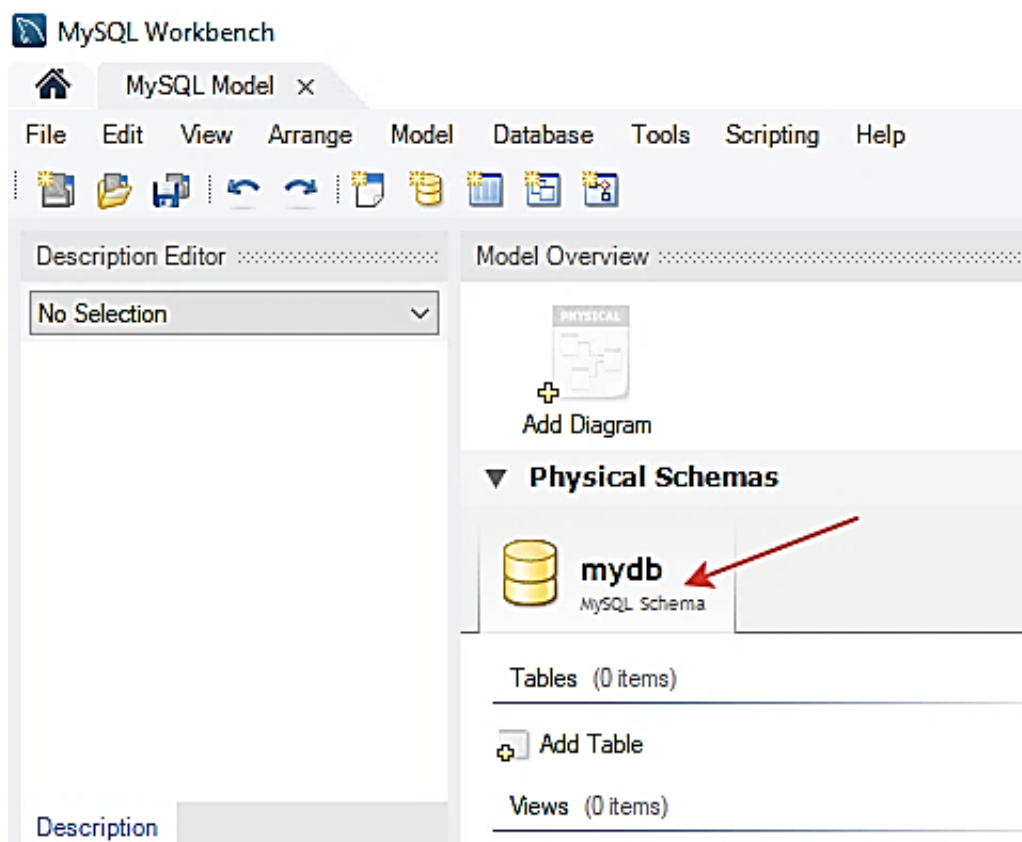


Рис. 1.77. Вкладка созданной модели новой БД

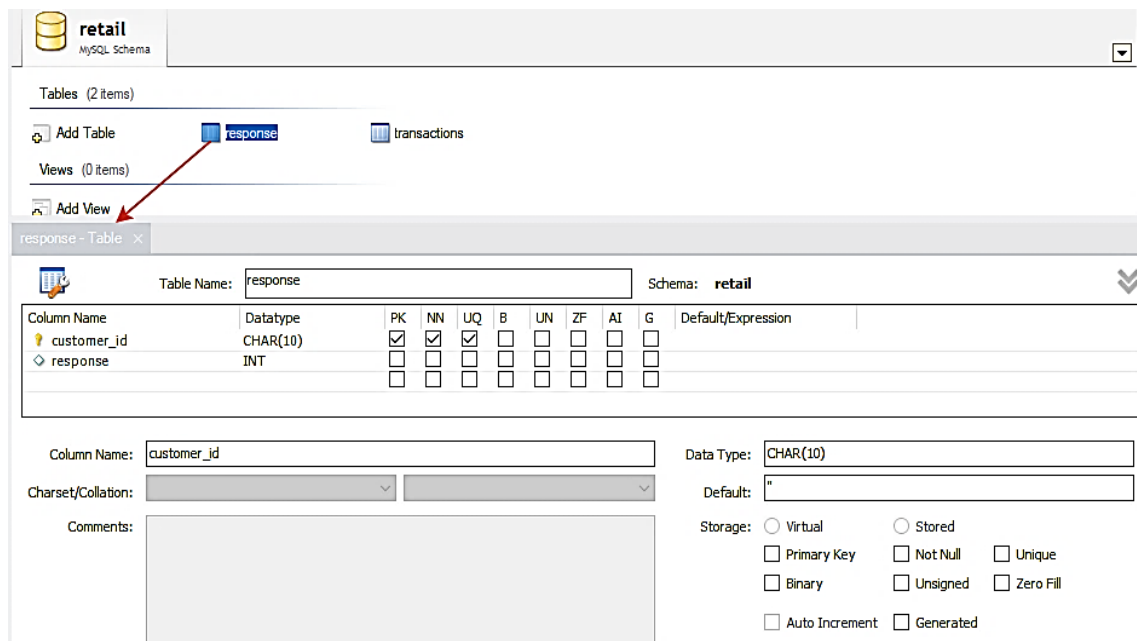


Рис. 1.78. Настройки для таблицы response

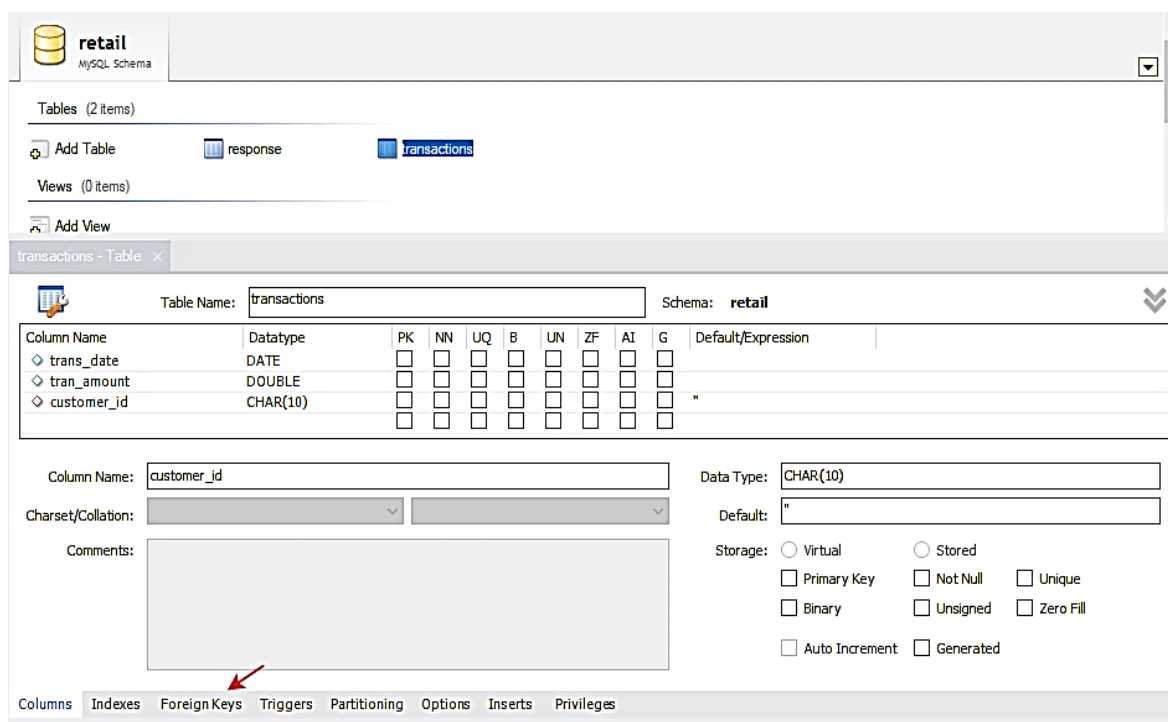


Рис. 1.79. Настройки для таблицы transactions

В настройках таблицы transactions перейдем на вкладку Foreign Keys и настроим связь между таблицами по полю customer_id (рис. 1.80, 1.81).

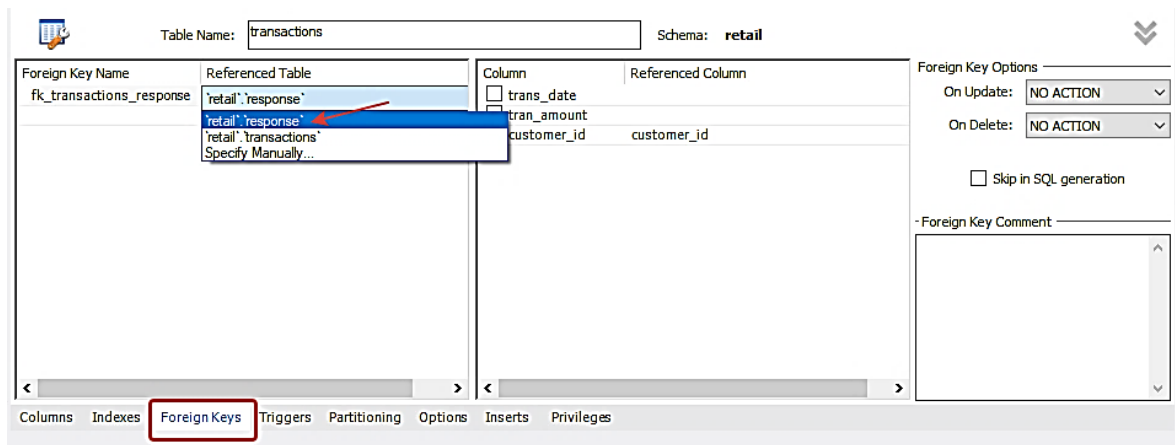


Рис. 1.80. Выбор Referenced Table

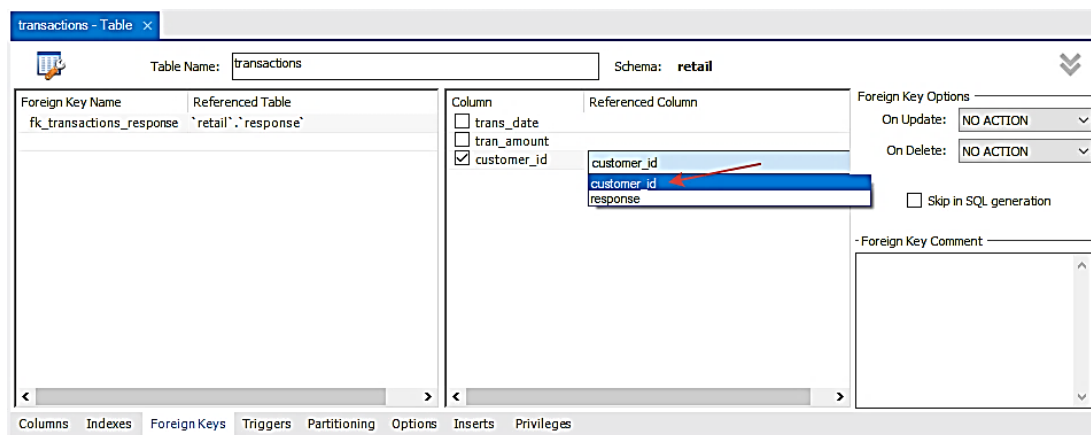


Рис. 1.81. Выбор Referenced Column

Добавим диаграмму (рис. 1.82).

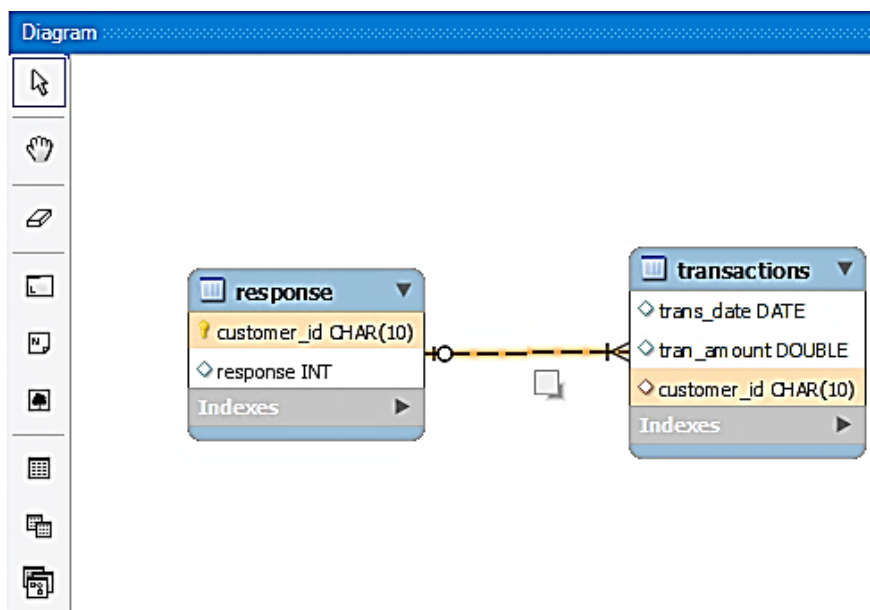


Рис. 1.82. EER diagram

И изменим нотацию диаграммы для наглядности (рис. 1.83).

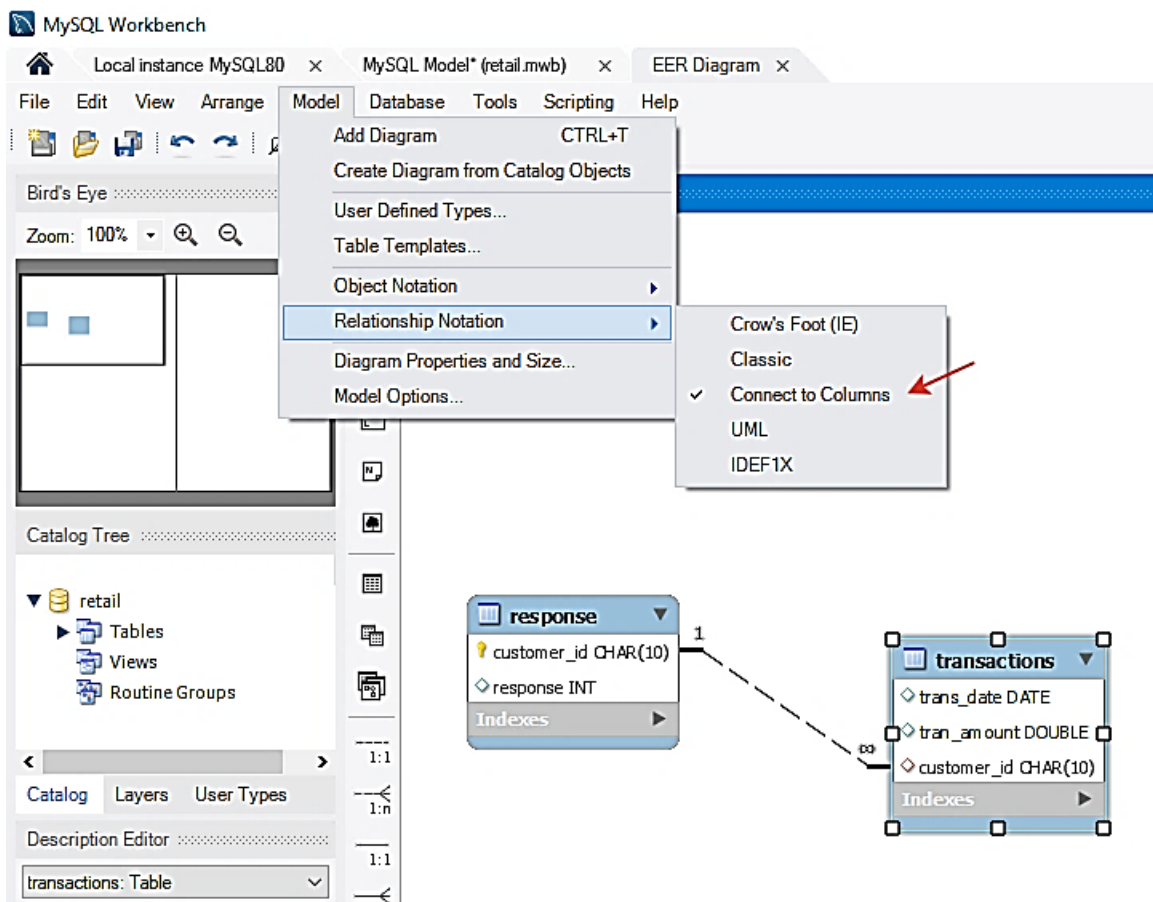


Рис. 1.83. Диаграмма в нотации Connect to Columns

Создадим на основе спроектированной модели БД retail (рис. 1.84).

Запустится мастер передачи проекта схемы на работающий сервер:

- Connection Options – Установка параметров подключения к СУБД;
- Options – Параметры;
- Select Objects – Выбор объектов;
- Review SQL Script – Просмотр сценария SQL (рис. 1.85);
- Commit Progress – Зафиксировать прогресс.

В текущем подключении к серверу после обновления появится новая БД retail, для таблиц которой загрузим Inspector (рис. 1.86).

В дальнейшем эта БД будет создана с использованием другого инструмента, поэтому удалим ее, используя контекстное меню (рис. 1.87).

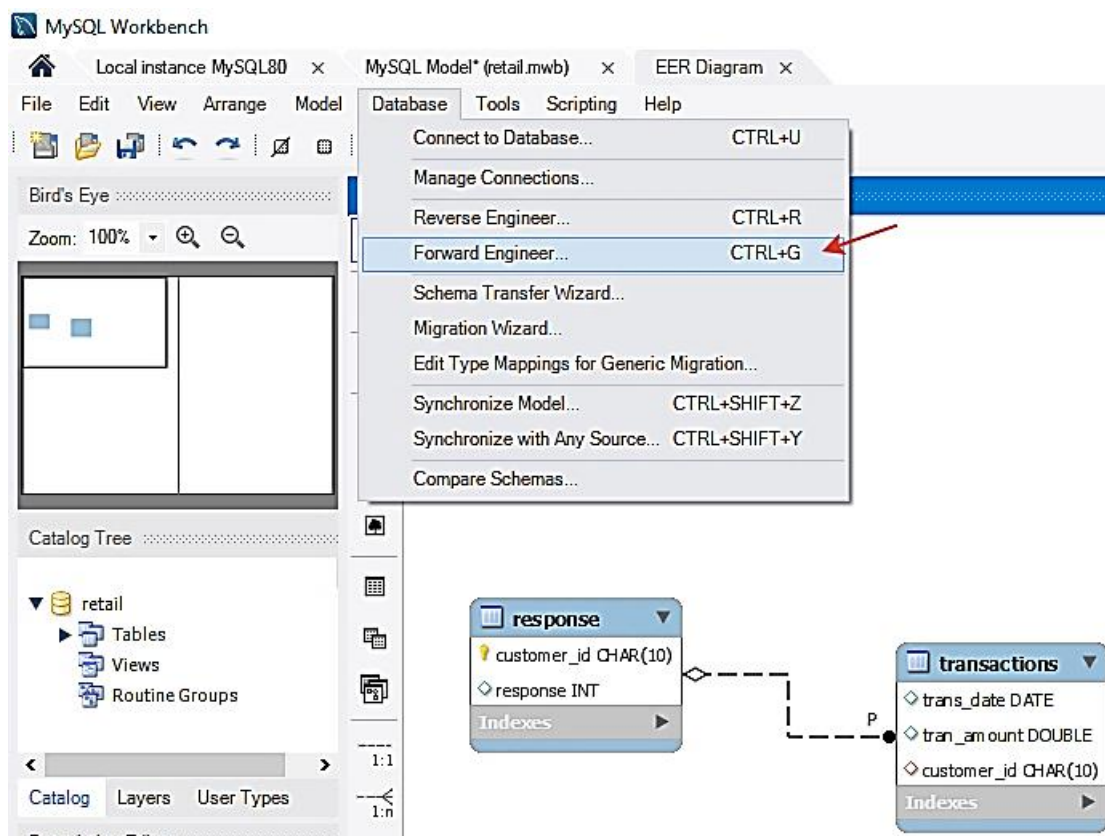


Рис. 1.84. Создание БД на основе модели

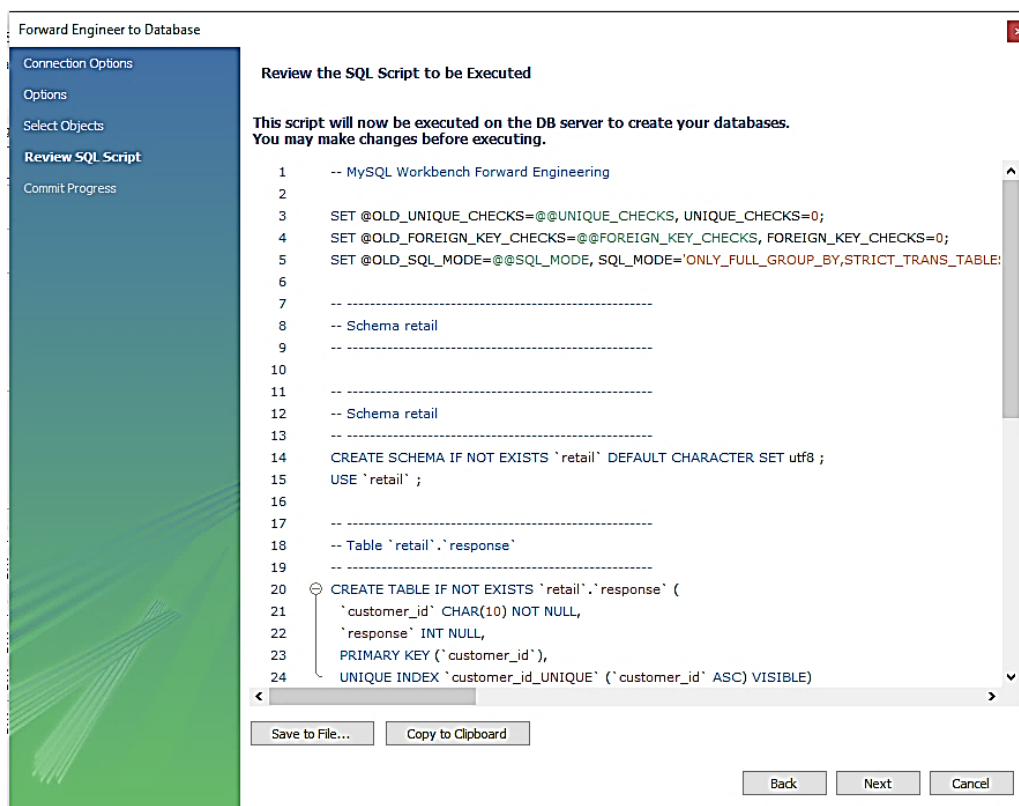


Рис. 1.85. Просмотр сценария SQL

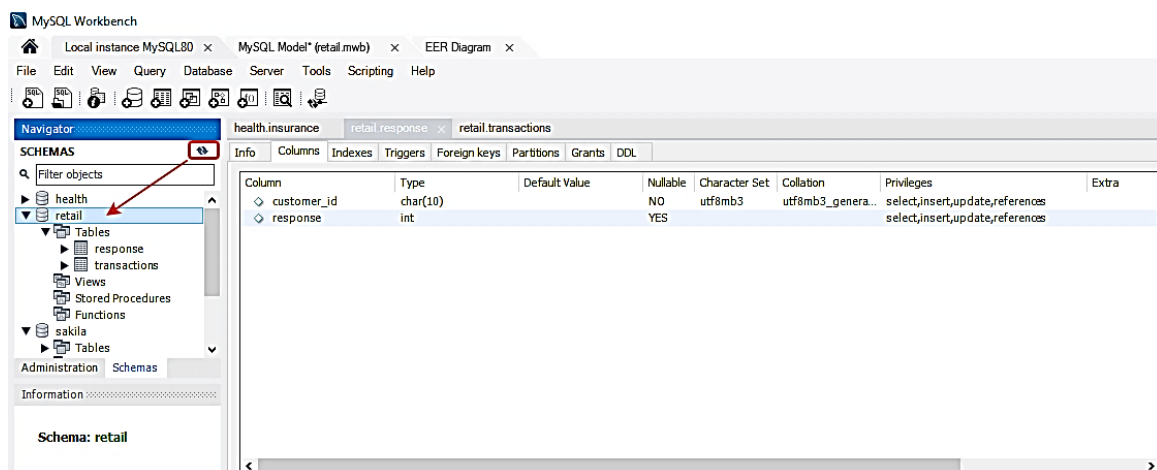


Рис. 1.86. Инспектор для таблиц БД retail

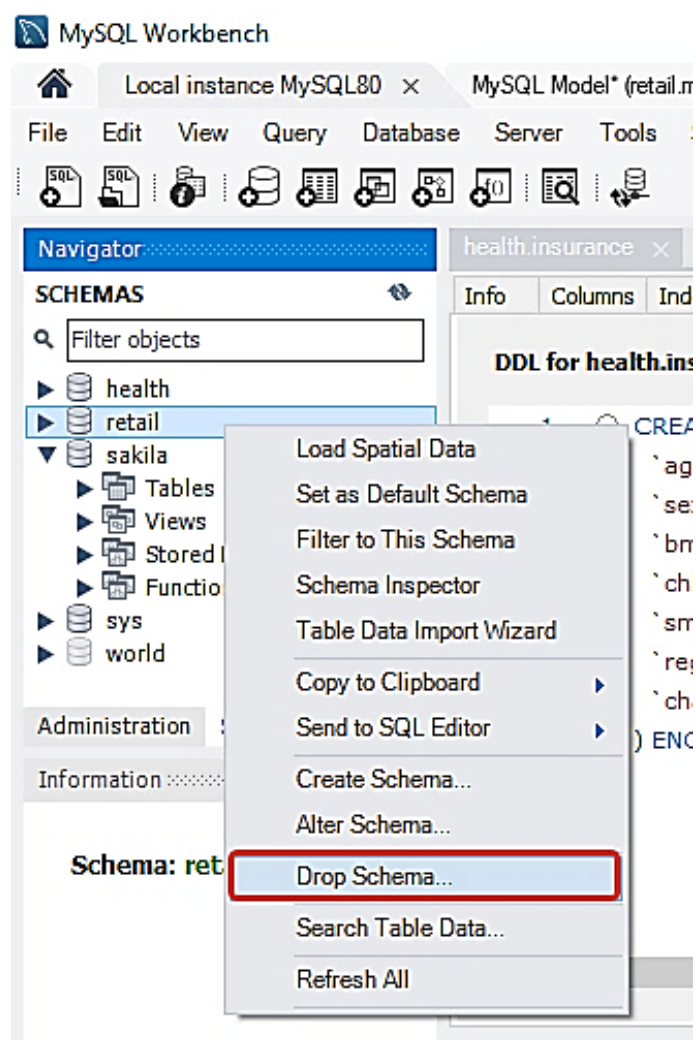


Рис. 1.87. Удаление БД retail

Рассмотрим другие инструменты работы с MySQL.

1.4. Консольные клиенты

MySQL Command Line Client и MySQL Shell

Консольный клиент MySQL Command Line Client при выборе полного варианта установки и с поддержкой Unicode и без (рис. 1.88).

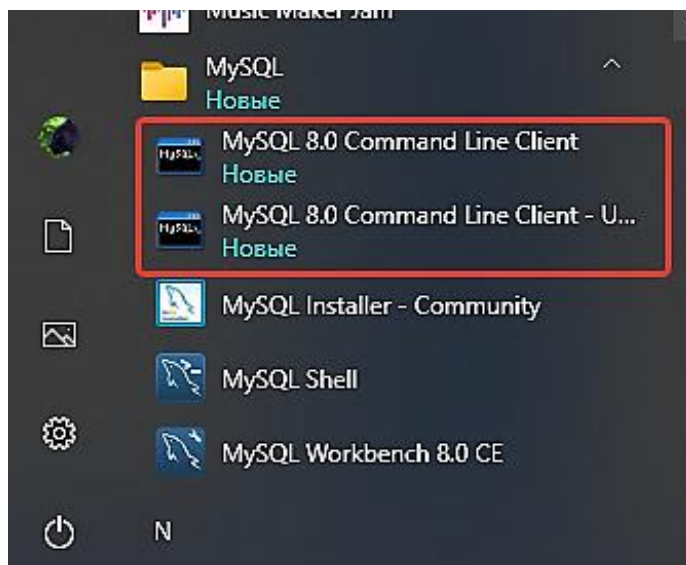


Рис. 1.88. Перечень установленных программ для MySQL

Запустим MySQL 8.0 Command Line Client – Unicode. После ввода пароля появится информация о подключении к серверу (рис. 1.89).

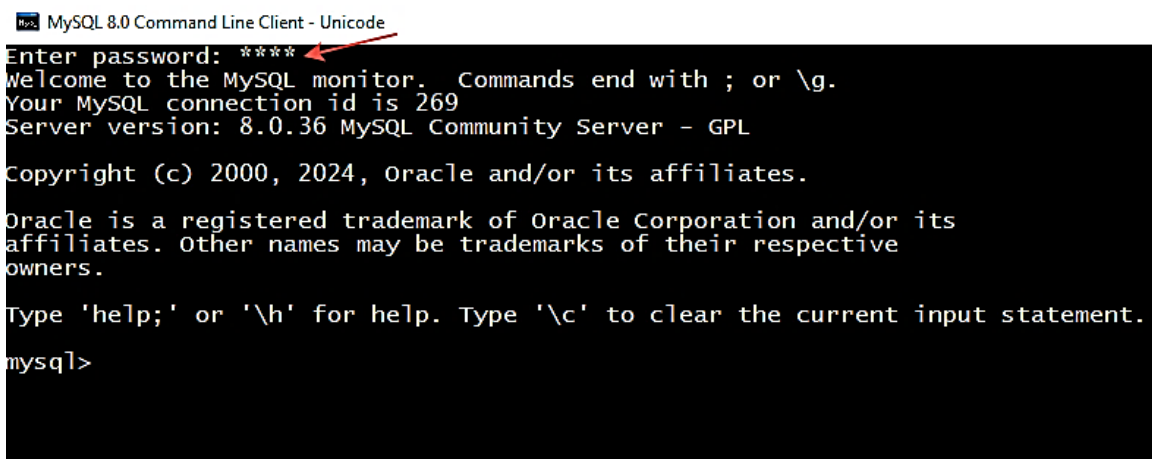


Рис. 1.89. Консольный клиент MySQL

Выведем на экран все базы данных, которые есть на текущем сервере (рис. 1.90):

```
show databases;
```

```
mysql> show databases;
+-----+
| Database |
+-----+
| health   |
| information_schema |
| mysql    |
| performance_schema |
| sakila   |
| sys      |
| world    |
+-----+
7 rows in set (1.52 sec)

mysql> _
```

Рис. 1.90. Вывод списка БД

Как видим на сервере по умолчанию уже есть ряд БД, которые выполняют административные функции. Есть БД, которые были установлены (MySQL Samples and Examples), и есть БД, кото-рая была создана в разделе 1.2. – health.

Создадим новую БД retail:

```
CREATE DATABASE retail;
```

и повторно запросим список БД (рис. 1.91).

```
mysql> CREATE DATABASE retail;
Query OK, 1 row affected (4.50 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| health   |
| information_schema |
| mysql    |
| performance_schema |
| retail   |
| sakila   |
| sys      |
| world    |
+-----+
8 rows in set (0.03 sec)

mysql>
```

Рис. 1.91. Создание БД retail

Подключимся к БД health и запросим первые 10 строк из таблицы insurance (рис. 1.92):

```
use health
SELECT * FROM insurance LIMIT 10;
```

```
mysql> use health;
Database changed
mysql> SELECT * FROM insurance LIMIT 10;
```

age	sex	bmi	children	smoker	region	charges
19	female	27.9	0	yes	southwest	16884.924
18	male	33.77	1	no	southeast	1725.5523
28	male	33	3	no	southeast	4449.462
33	male	22.705	0	no	northwest	21984.47061
32	male	28.88	0	no	northwest	3866.8552
31	female	25.74	0	no	southeast	3756.6216
46	female	33.44	1	no	southeast	8240.5896
37	female	27.74	3	no	northwest	7281.5056
37	male	29.83	2	no	northeast	6406.4107
60	female	25.84	0	no	northwest	28923.13692

```
10 rows in set (0.10 sec)

mysql>
```

Рис. 1.92. Подключение к БД health и вывод 10 строк с данными таблицы insurance

Вызвать консольного клиента MySQL можно и из командной строки Windows (см. рис. 1.25) или набрав в диалоговом окне, открытом клавишами Win+R, команду cmd (рис. 1.93).

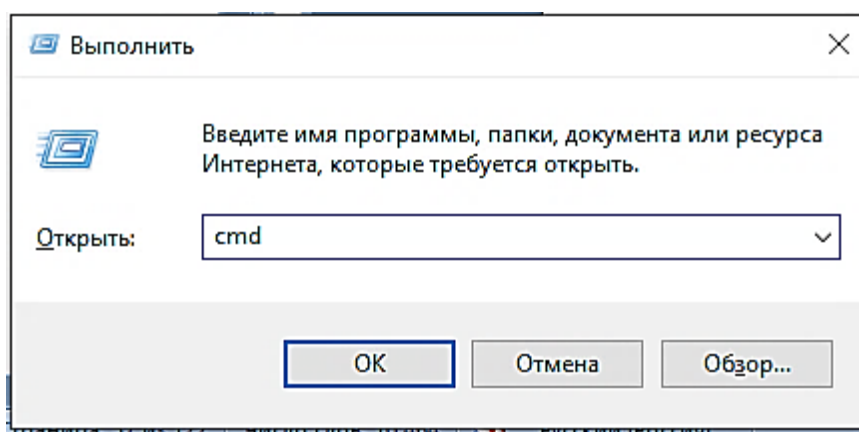


Рис. 1.93. Запуск командной строки через диалоговое окно (Win+R)

В командной строке, прежде всего, нужно изменить директорию (команда `cd`) на ту, в которой установлен консольный клиент:

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin
```

а затем запустить MySQL 8.0 Command Line Client

```
mysql -u root -p
```

`-u root` – это имя пользователя (суперпользователь `root`), `-p` – запрос на ввод пароля, после чего появится приглашение для ввода пароля и при успешной авторизации произойдет подключение к серверу (рис. 1.94).

```

Командная строка - mysql -u root -p
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Svetlana>cd C:\Program Files\MySQL\MySQL Server 8.0\bin

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 280
Server version: 8.0.36 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

Рис. 1.94. Подключение к серверу
через консольного клиента MySQL

Запросим 10 строк из таблицы film, подключившись к БД sakila
(рис. 1.95).

```

Командная строка - mysql -u root -p
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

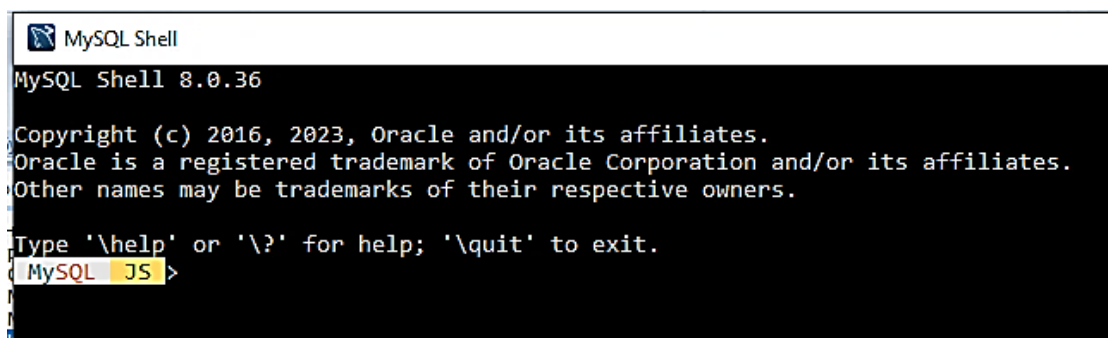
mysql> use sakila;
Database changed
mysql> SELECT * FROM film LIMIT 10;
+-----+-----+-----+-----+-----+-----+-----+-----+
| film_id | title | description | release_year | language_id | original_language_id | rental_duration | rental_rate | length |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | ACADEMY DINOSAUR | A Epic Drama of a Feminist And a Mad Scientist who must Battle a Teacher in The Canadian Rockies | 2006 | 1 | NULL | 6 | 0.99 | 8 |
| 2 | ACE GOLDFINGER | A Astounding Epistle of a Database Administrator And a Explorer who must Find a Car in Ancient China | 2006 | 1 | NULL | 3 | 4.99 | 4 |
| 3 | ADAPTATION HOLES | Trailers,Deleted Scenes | 2006-02-15 05:03:42 | 1 | NULL | 7 | 2.99 | 5 |
| 4 | AFFAIR PREJUDICE | A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monkey in A Shark Tank | 2006 | 1 | NULL | 5 | 2.99 | 11 |
| 5 | AFRICAN EGG | Commentaries,Behind the Scenes | 2006-02-15 05:03:42 | 1 | NULL | 6 | 2.99 | 13 |

```

Рис. 1.95. Запрос строк из таблицы film,
после подключения к БД sakila

Работать в консольном клиенте менее удобно, чем в специализированных продуктах с графическим интерфейсом.

Протестируем также работу в MySQL Shell. После запуска MySQL Shell откроется без подключения к серверу, по умолчанию в режиме JavaScript (рис. 1.96).



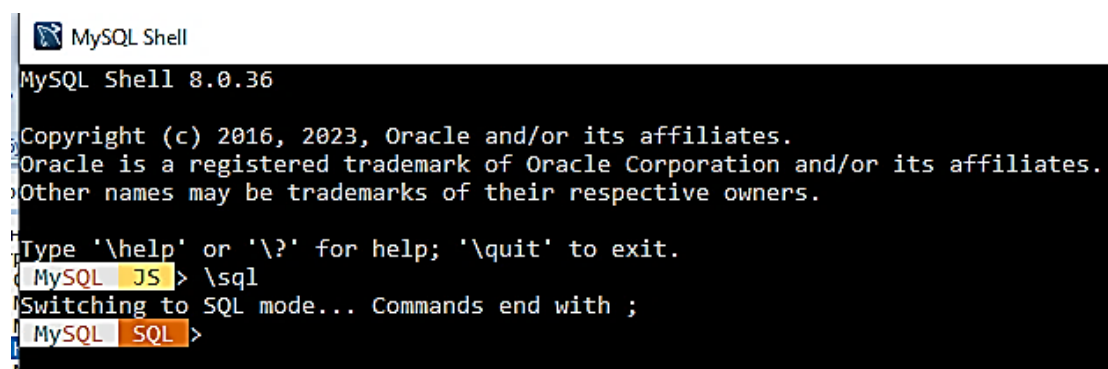
```
MySQL Shell 8.0.36

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS >
```

Рис. 1.96. Запуск MySQL Shell

MySQL Shell поддерживает ввод команд на трех языках: JavaScript, Python и SQL, меняется режим с помощью команд `\sql`, `\py` и `\js`. Переключимся на язык SQL, введя команду `\sql` (рис. 1.97).



```
MySQL Shell 8.0.36

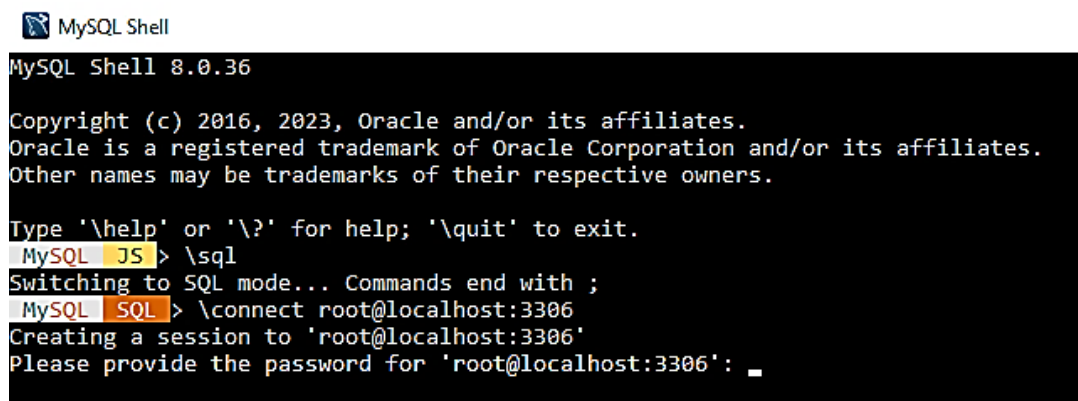
Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \sql
Switching to SQL mode... Commands end with ;
MySQL SQL >
```

Рис. 1.97. Переключение на язык SQL

Далее выполним подключение к серверу (рис. 1.98) с помощью команды `\connect`, указав данные о сервере в формате имя_пользователя@хост:порт (суперпользователь `root`, локальный хост и порт `localhost:3306`):

```
\connect root@localhost:3306
```



```
MySQL Shell 8.0.36

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS > \sql
Switching to SQL mode... Commands end with ;
MySQL SQL > \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Please provide the password for 'root@localhost:3306':
```

Рис. 1.98. Подключение к серверу MySQL

Введем пароль и укажем, что пароль сохранять не нужно (рис. 1.99).

```
MySQL Shell
MySQL Shell 8.0.36

Copyright (c) 2016, 2023, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
Other names may be trademarks of their respective owners.

Type '\help' or '\?' for help; '\quit' to exit.
MySQL JS> \sql
Switching to SQL mode... Commands end with ;
MySQL SQL> \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Please provide the password for 'root@localhost:3306': ****
Save password for 'root@localhost:3306'? [Y]es/[N]o/Ne[v]er (default No): N
```

Рис. 1.99. Ввод пароля

Подключимся к БД health и запросим первые 10 строк из таблицы insurance (рис. 1.100):

```
use health
SELECT * FROM insurance LIMIT 10;
```

```
MySQL JS> \sql
Switching to SQL mode... Commands end with ;
MySQL SQL> \connect root@localhost:3306
Creating a session to 'root@localhost:3306'
Please provide the password for 'root@localhost:3306': ****
Save password for 'root@localhost:3306'? [Y]es/[N]o/Ne[v]er (default No): N
Fetching global names for auto-completion... Press ^C to stop.
Your MySQL connection id is 291
Server version: 8.0.36 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
MySQL localhost:3306 ssl SQL> use health;
Default schema set to `health`.
Fetching global names, object names from `health` for auto-completion... Press ^C to stop.
MySQL localhost:3306 ssl health SQL> SELECT * FROM insurance LIMIT 10;
+-----+-----+-----+-----+-----+-----+-----+
| age | sex  | bmi  | children | smoker | region  | charges |
+-----+-----+-----+-----+-----+-----+-----+
| 19 | female | 27.9 | 0 | yes | southwest | 16884.924 |
| 18 | male   | 33.77 | 1 | no  | southeast | 1725.5523 |
| 28 | male   | 33    | 3 | no  | southeast | 4449.462  |
| 33 | male   | 22.705 | 0 | no  | northwest | 21984.47061 |
| 32 | male   | 28.88 | 0 | no  | northwest | 3866.8552 |
| 31 | female | 25.74 | 0 | no  | southeast | 3756.6216 |
| 46 | female | 33.44 | 1 | no  | southeast | 8240.5896 |
| 37 | female | 27.74 | 3 | no  | northwest | 7281.5056 |
| 37 | male   | 29.83 | 2 | no  | northeast | 6406.4107 |
| 60 | female | 25.84 | 0 | no  | northwest | 28923.13692 |
+-----+-----+-----+-----+-----+-----+-----+
```

Рис. 1.100. Подключение к БД health и вывод 10 строк с данными таблицы insurance

На этом завершим знакомство с консольными клиентами.

1.5. Менеджер баз данных DBeaver

Существует достаточно универсальный менеджер баз данных DBeaver, который умеет работать, в том числе, и с MySQL [6]. Скачаем его инсталлятор с официального сайта <https://dbeaver.io/> (рис. 1.101).

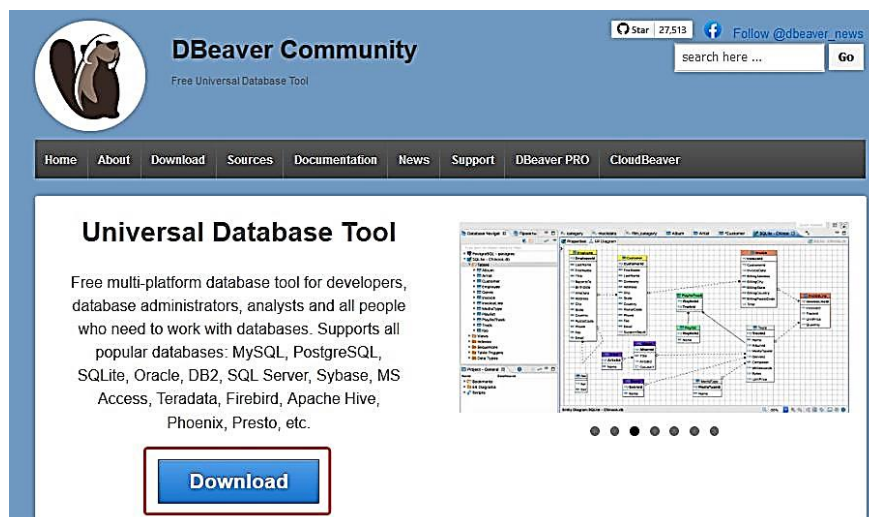


Рис. 1.101. Переход на страницу выбора версии установочных файлов

С учетом операционной системы выберем файл для скачивания (рис. 1.102).

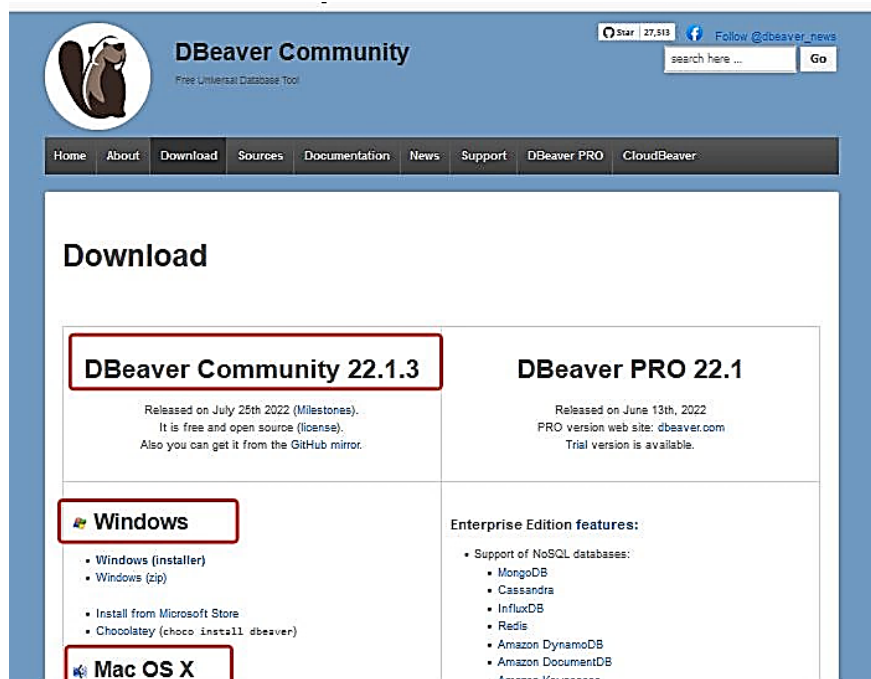


Рис. 1.102. Скачивание установочных файлов

При запуске инсталлятора необходимо выбрать язык для установщика. В мастере установки необходимо принять лицензионное соглашение и выбрать вариант установки (для всех пользователей или для одного, под которым происходит установка программного обеспечения (ПО)). После завершения работы мастера установки запустим менеджер баз данных DBeaver (рис. 1.103).

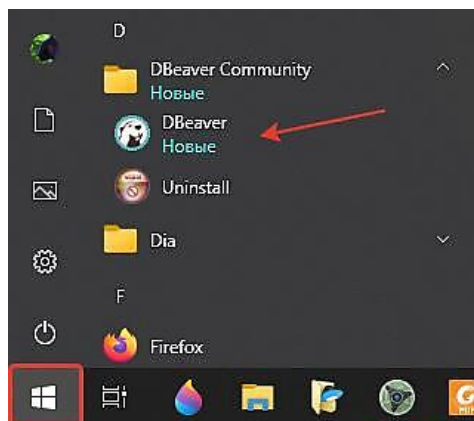


Рис. 1.103. Запуск менеджера DBeaver

В командном меню менеджера создаем новый файл (рис. 1.104).

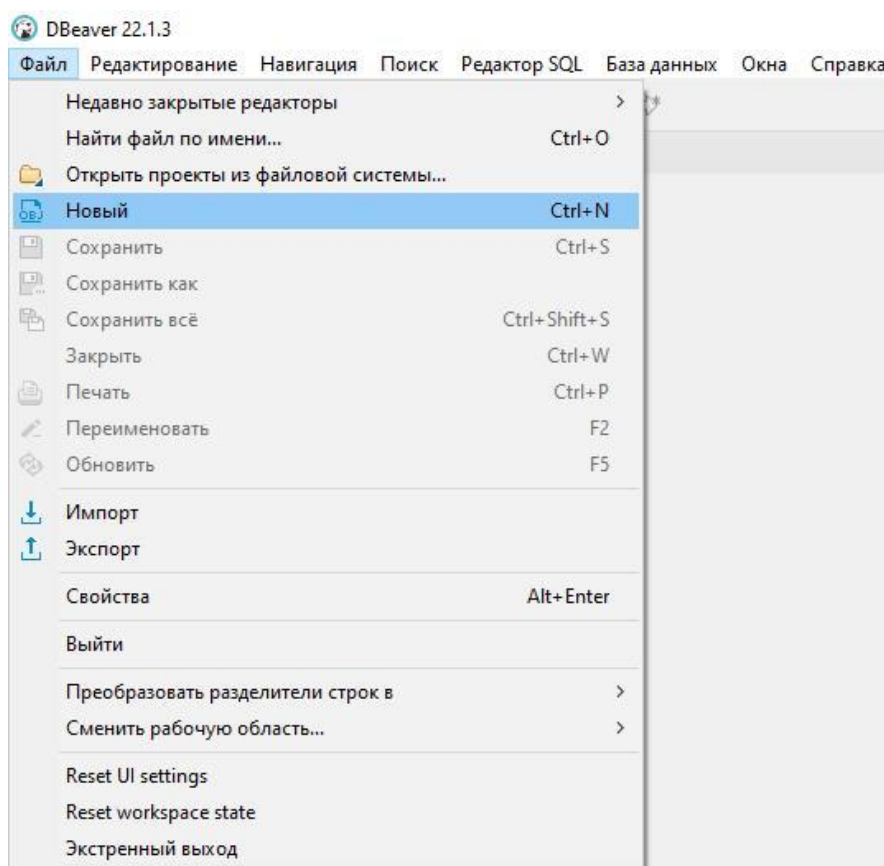


Рис. 1.104. Создание файла

Выбираем тип нового соединения (рис. 1.105).

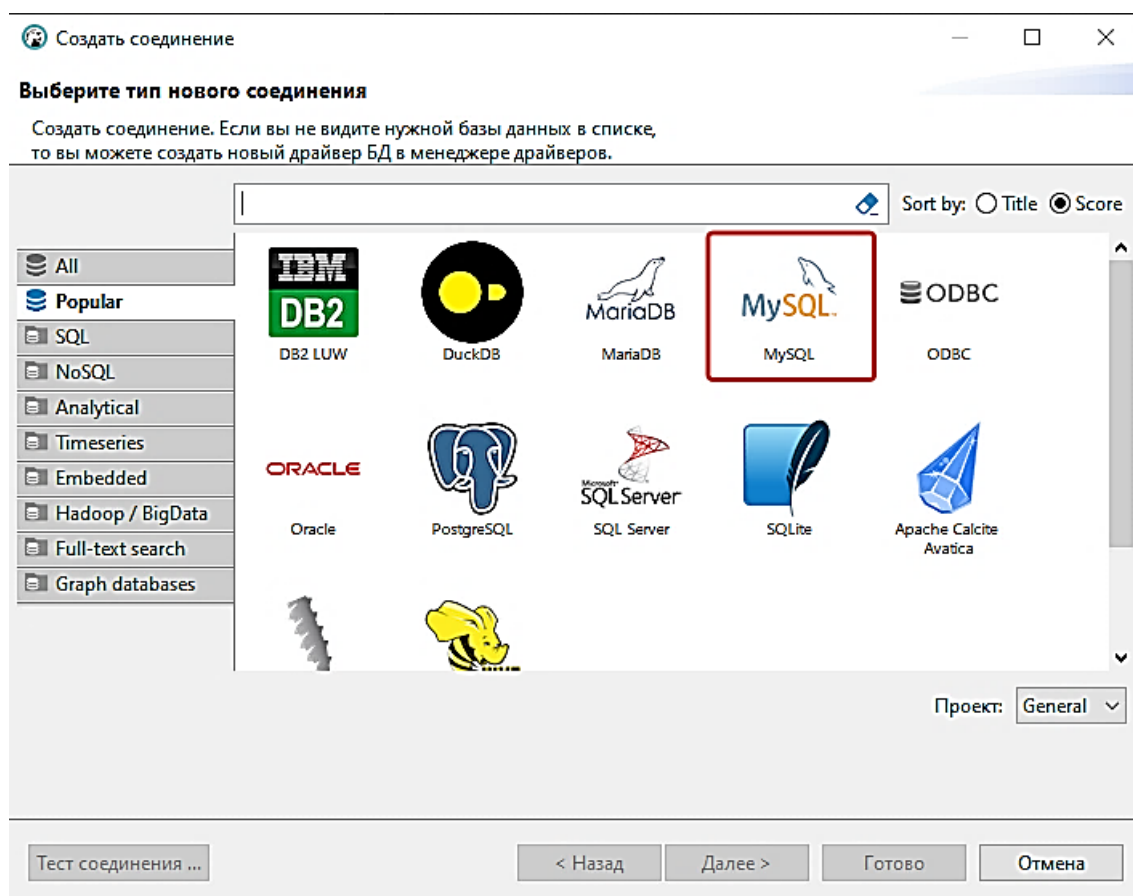


Рис. 1.105. Выбираем базу данных для соединения

Если менеджер DBeaver ранее использовался и есть текущие подключения, то создаем новое соединение нужного типа (рис. 1.106).

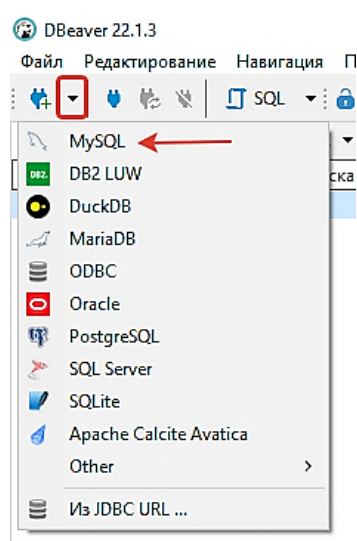


Рис. 1.106. Добавление нового подключения

Настраиваем соединение с сервером MySQL (рис. 1.107).

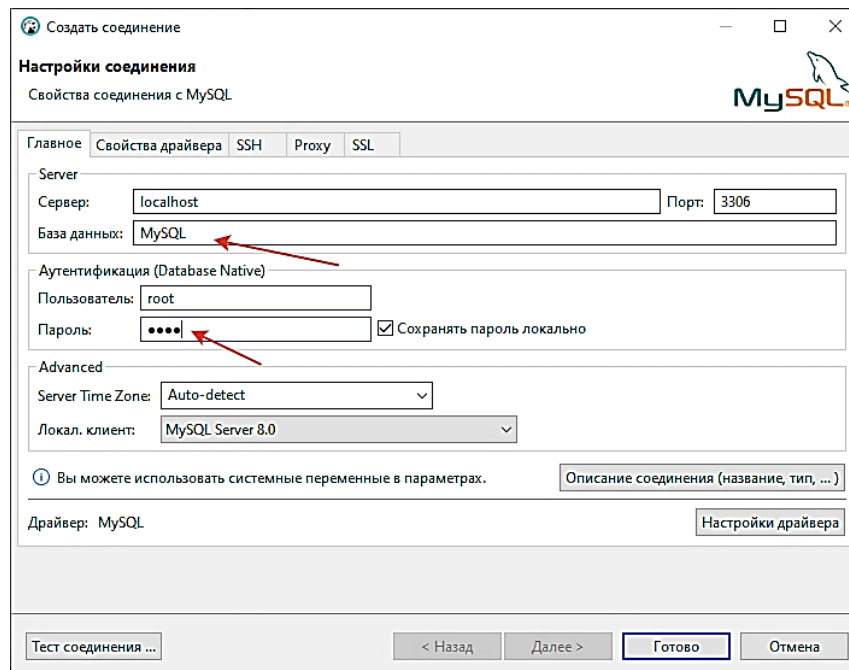


Рис. 1.107. Настройки для соединения с сервером MySQL

Состояние сервера MySQL, к которому выполнено подключение, будет отображаться слева (рис. 1.108). Сразу откроем SQL-скрипт (рис. 1.109).

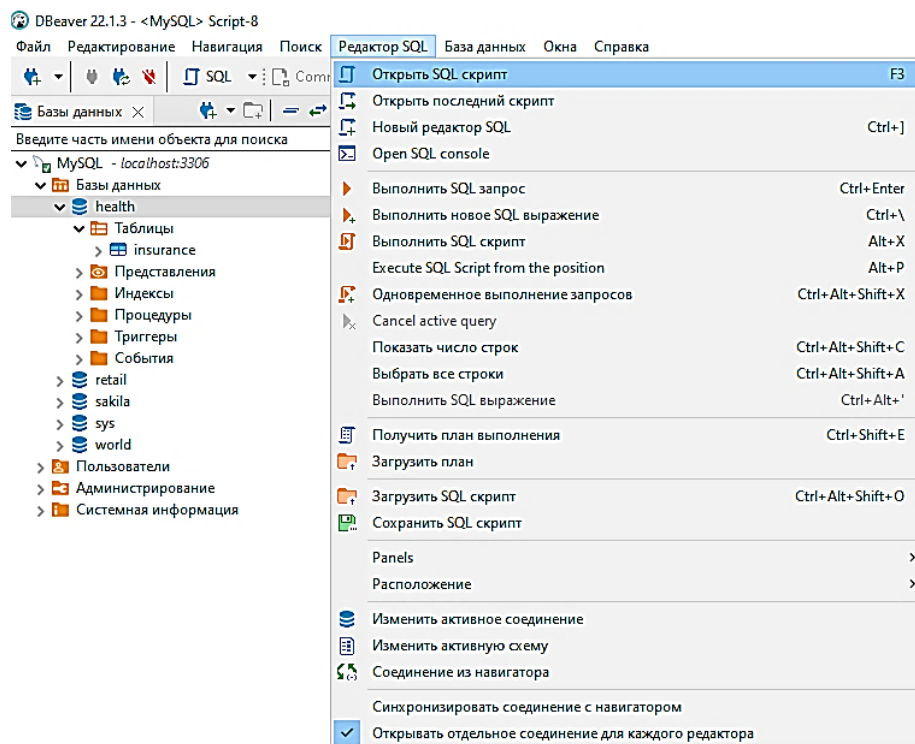


Рис. 1.108. Подключение к серверу MySQL

Для работы со скриптами SQL выполним настройки для DBeaver, так как есть рекомендация по служебным (ключевым) словам SQL – использовать для них верхний регистр, то в командном меню Окна выберем опцию Настройки (рис. 1.109).

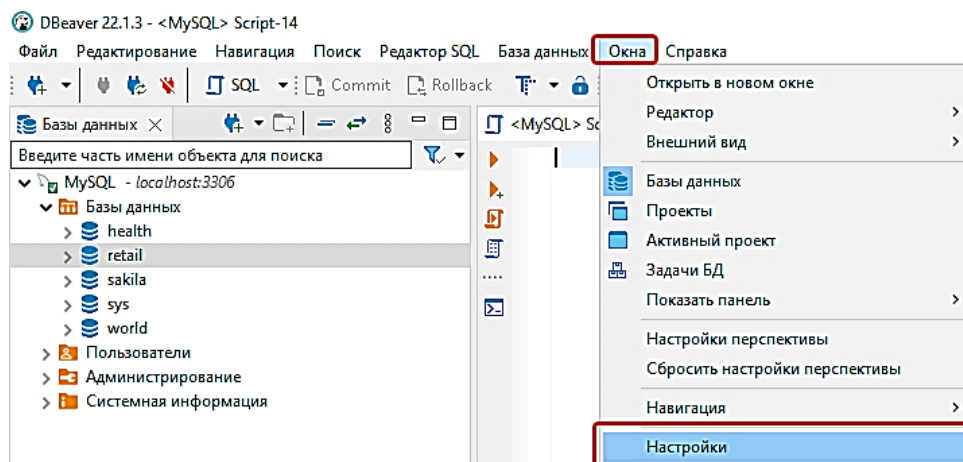


Рис. 1.109. Переход к настройкам DBeaver

Для редактора SQL выберем в навигаторе слева опцию Форматирование и настроим переключатель для Регистра ключевых слов в позицию Upper (рис. 1.110). Теперь автозаполнение по подсказкам команд (клавиша ENTER для автозаполнения) будет происходить в верхнем регистре.

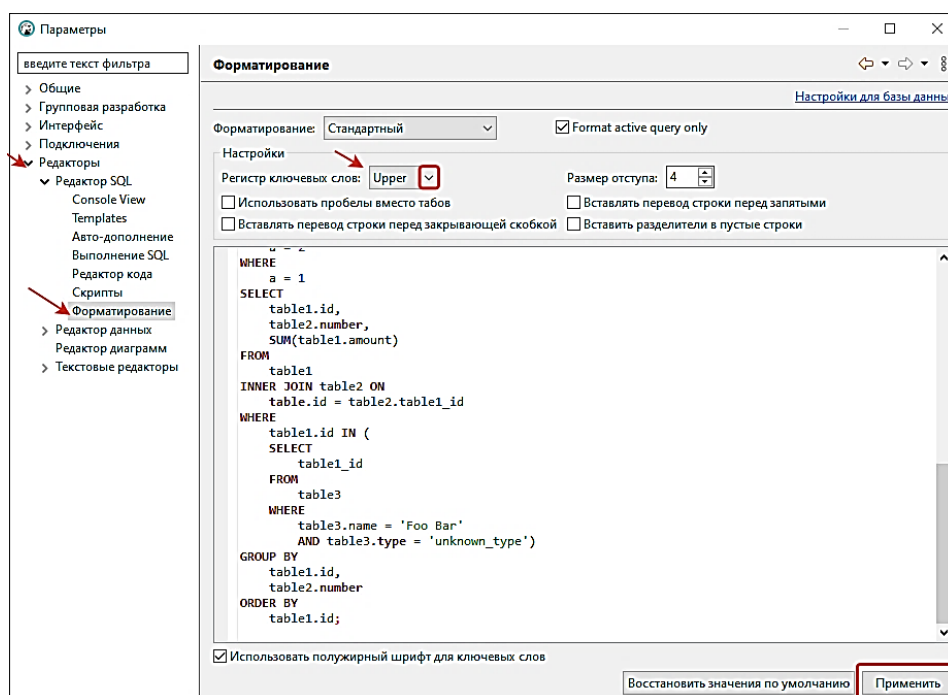


Рис. 1.110. Настройка верхнего регистра для ключевых слов в редакторе SQL

Редактор SQL позволяет обращаться к объектам базы, относительно которой он запущен, без ее упоминания.

Текущий скрипт открыт для базы health, она отображается в настройках редактора и к таблице health.insurance можно обратиться напрямую по имени таблицы (рис. 1.111):

```
SELECT * FROM insurance LIMIT 10
```

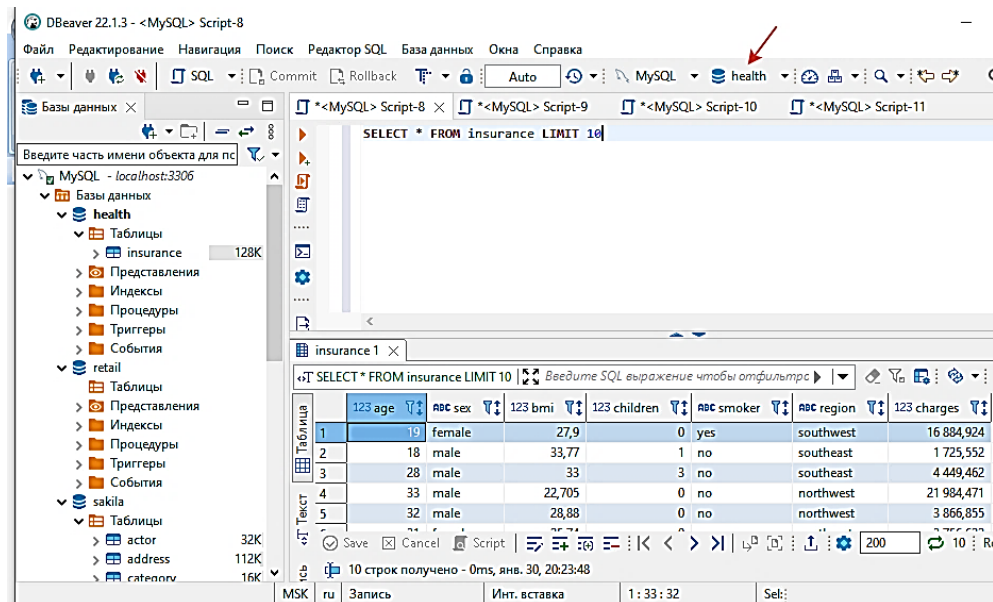


Рис. 1.111. Выполнение запроса на выбор данных из таблицы insurance

Аналогичный запрос к таблице sakila.film без упоминания БД sakila приведет к ошибке (рис. 1.112):

```
SELECT * FROM film LIMIT 10  
SELECT * FROM sakila.film LIMIT 10
```

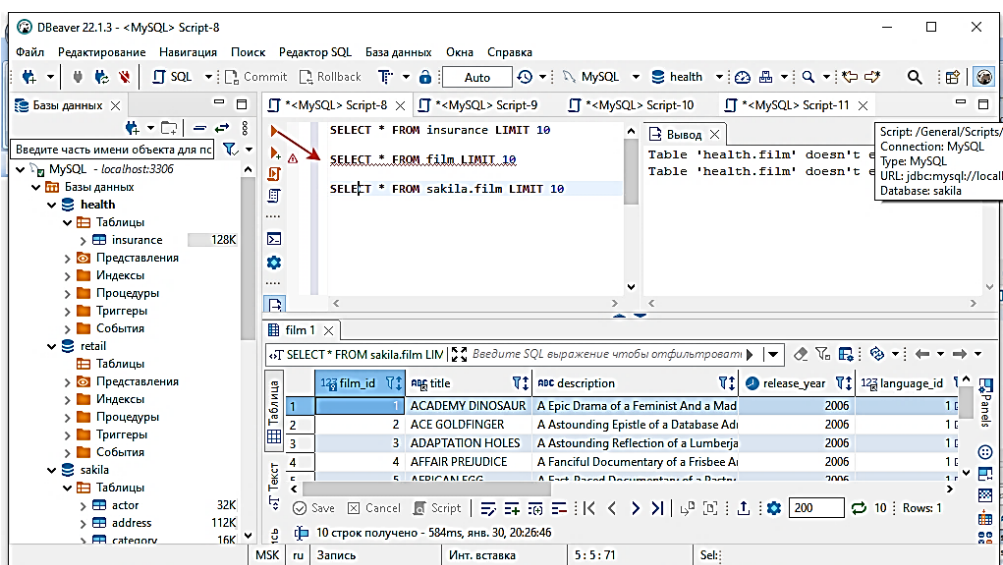


Рис. 1.112. Запрос к таблице sakila.film

Но стоит сменить в настройках редактора активную БД (рис. 1.113) и повторное выполнение ранее ошибочного запроса даст результат.

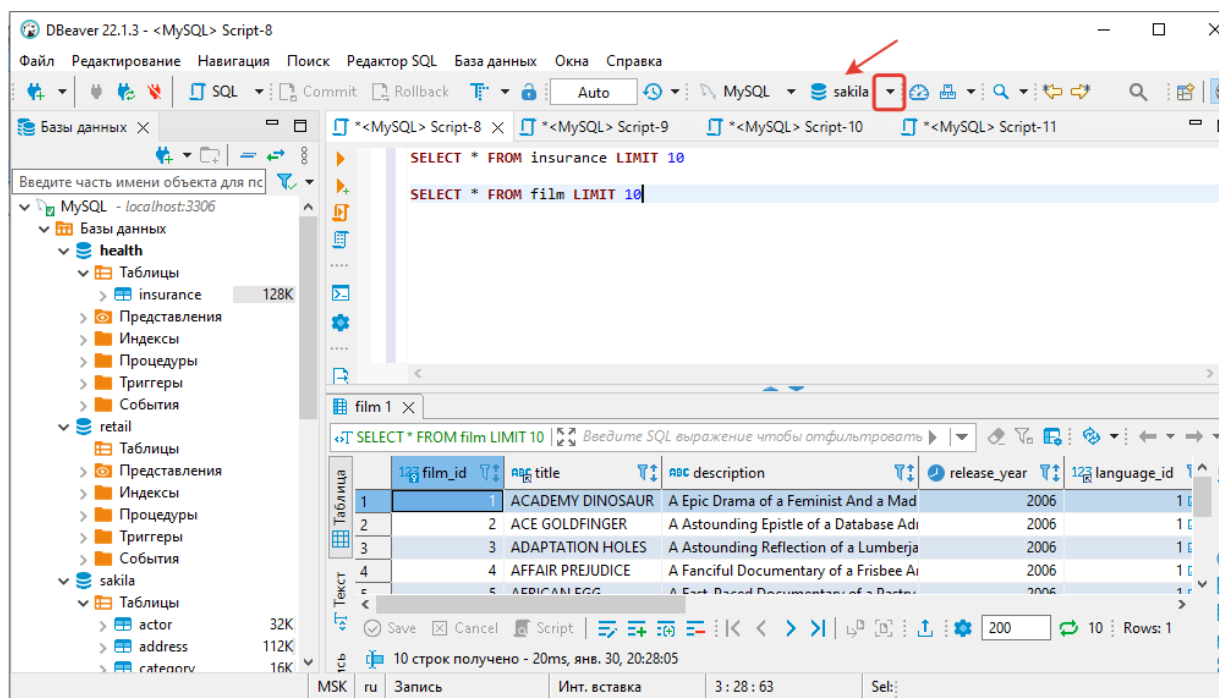


Рис. 1.113. Запрос к таблице sakila.film с новыми настройками редактора

Ранее с помощью MySQL Workbench была создана БД health, в которую были импортированы данные из файла с созданием таблицы insurance. Проведем аналогичные операции и в DBeaver.

В ранее созданную базу данных retail из файла импортируем данные из файла scanner_data.csv (доступен по ссылке <https://www.kaggle.com/datasets/marian447/retail-store-sales-transactions>).

Контекст данных: данные сканера о продажах в розничном магазине.

Данные файла scanner_data.csv:

- Date – дата сделки купли-продажи;
- Customer_ID – идентификатор покупателя;
- Transaction_ID – идентификатор транзакции;
- SKU_Category – категория товара;
- SKU – идентификатор товара;
- Quantity – количество проданного товара;
- Sales_Amount – стоимость проданного товара.

Просмотр данных возможен с помощью текстового редактора например, Блокнот (Стандартные программы ОС Windows) или

с помощью редактора кода Notepad++ (дистрибутив можно скачать по ссылке <https://notepad-plus-plus.org/downloads/>).

В первой строке имена столбцов, разделитель между данными – запятая (рис. 1.114).

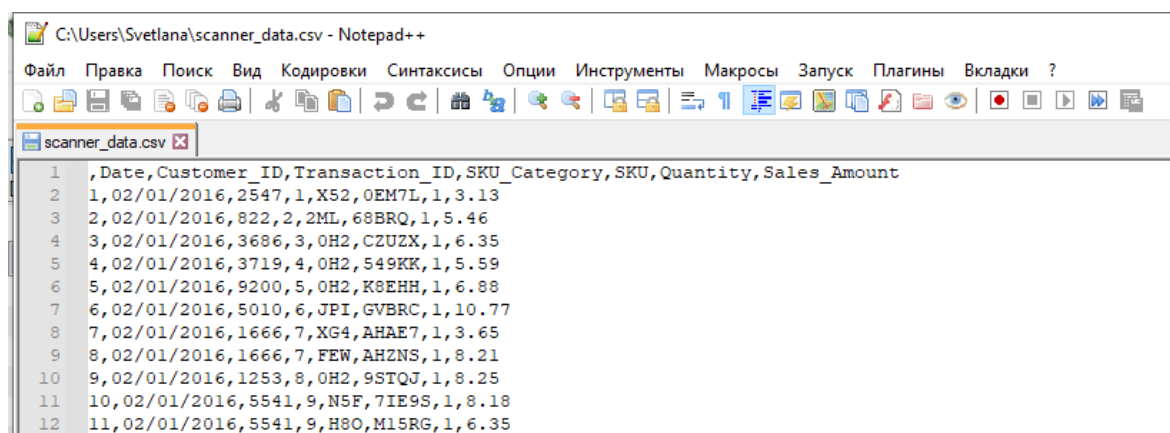


Рис. 1.114. Содержимое файла scanner_data.csv

В контекстном меню для БД retail выбираем опцию Импорт данных (рис. 1.115).

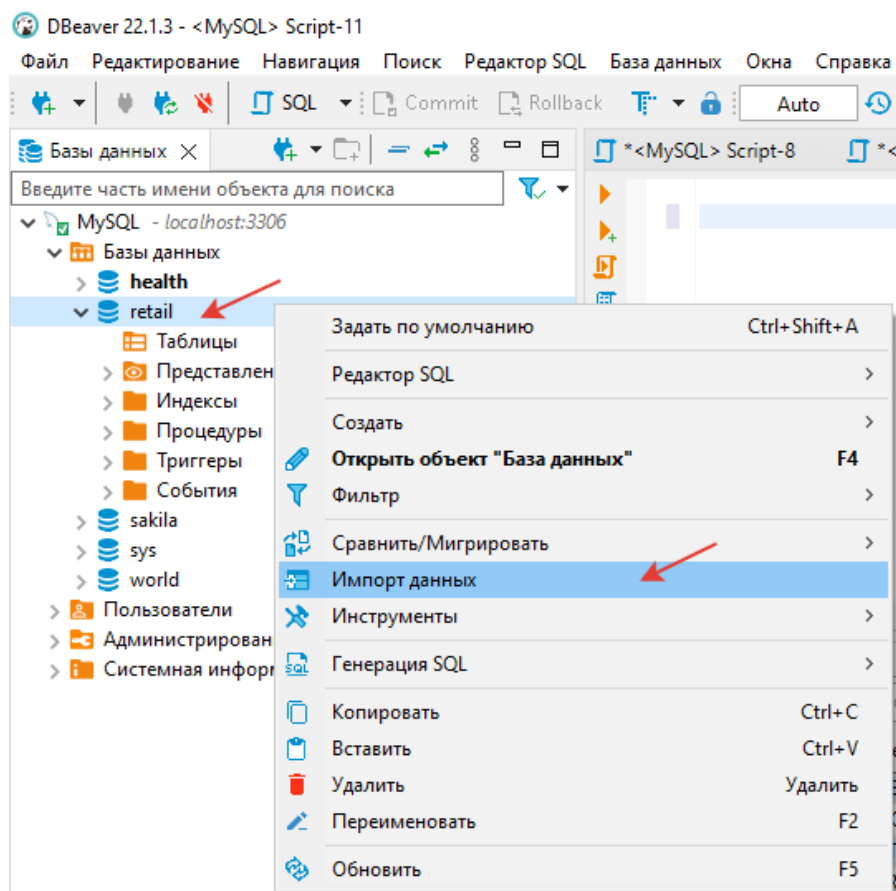


Рис. 1.115. Импорт данных в БД retail

Импорт происходит в БД retail (рис. 1.116), если бы в БД уже была создана таблица, то данные можно было импортировать в нее.

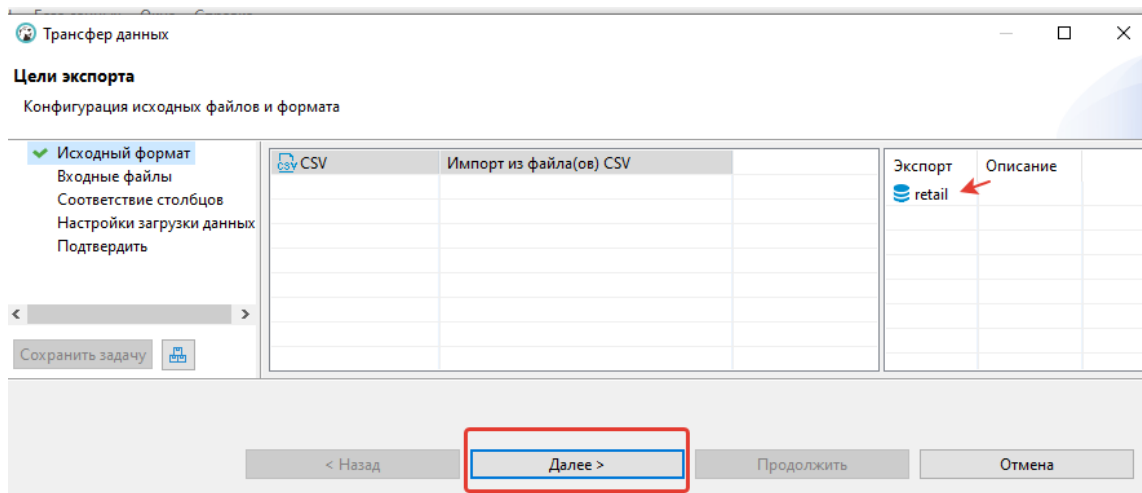


Рис. 1.116. Импорт данных (начало)

Выбирается файл для импорта (рис. 1.117).

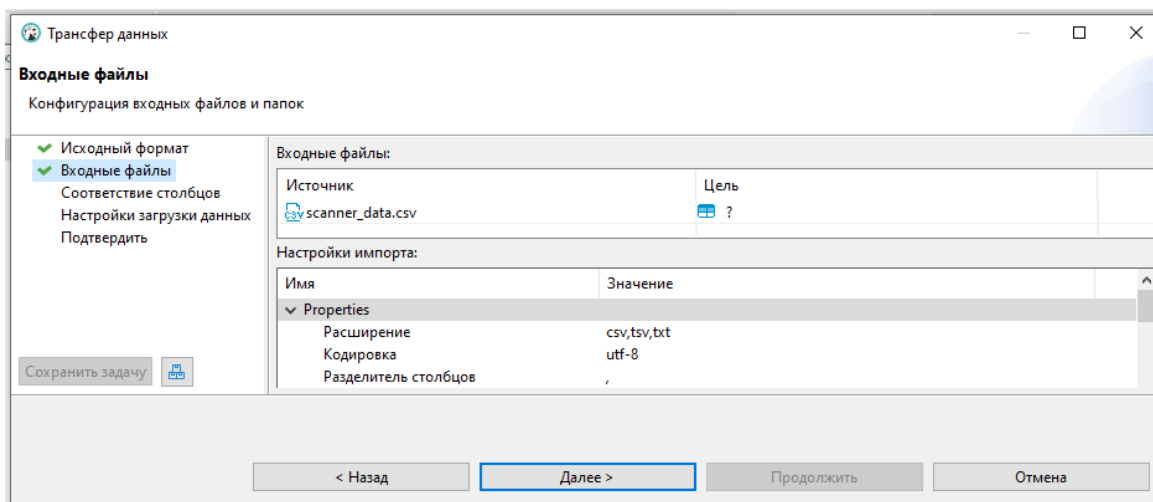


Рис. 1.117. Выбор файла

Именем создаваемой таблицы становится имя файла, но его можно скорректировать (рис. 1.118).

В настройках для импортируемых столбцов (рис. 1.119) выберем для первого столбца, являющегося номером строки, значение – skip (пропустить). Отметим также, что имя столбца Date, взято в кавычки. Это произошло по причине того, что Date зарезервированное, служебное слово, обозначающее тип данных. Изменим название целевого столбца и понизим регистр для имен всех остальных столбцов. Изменим также тип данных для столбцов customer_id

и transaction_id, так как идентификаторы, даже будучи числами, не могут участвовать в арифметических операциях и т.п.

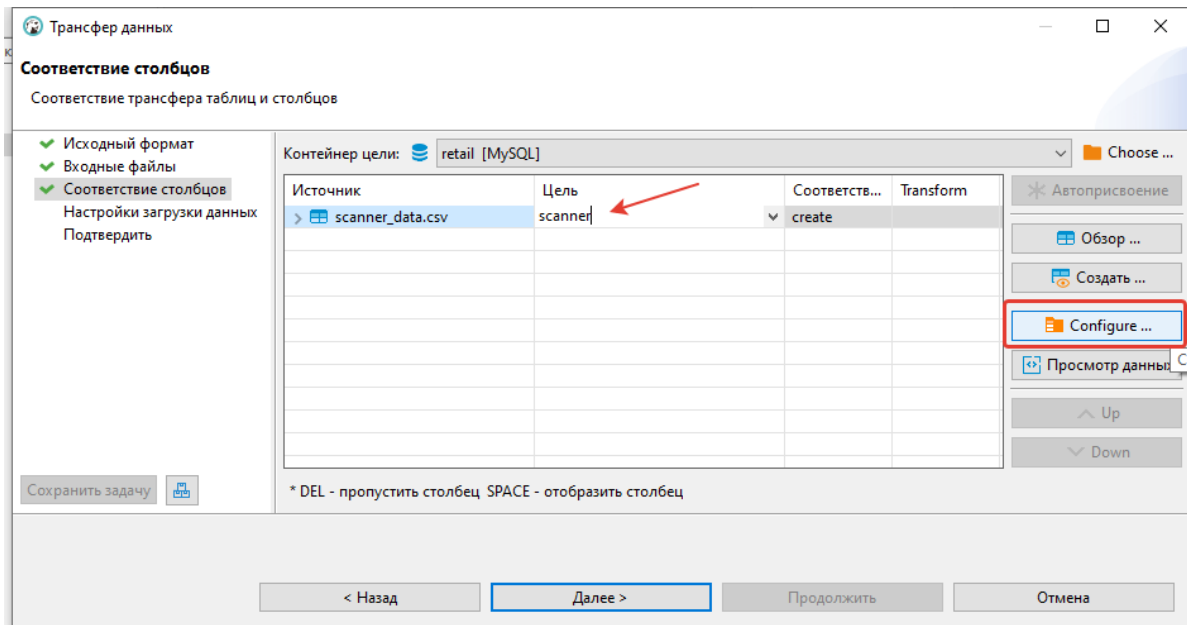


Рис. 1.118. Имя создаваемой таблицы

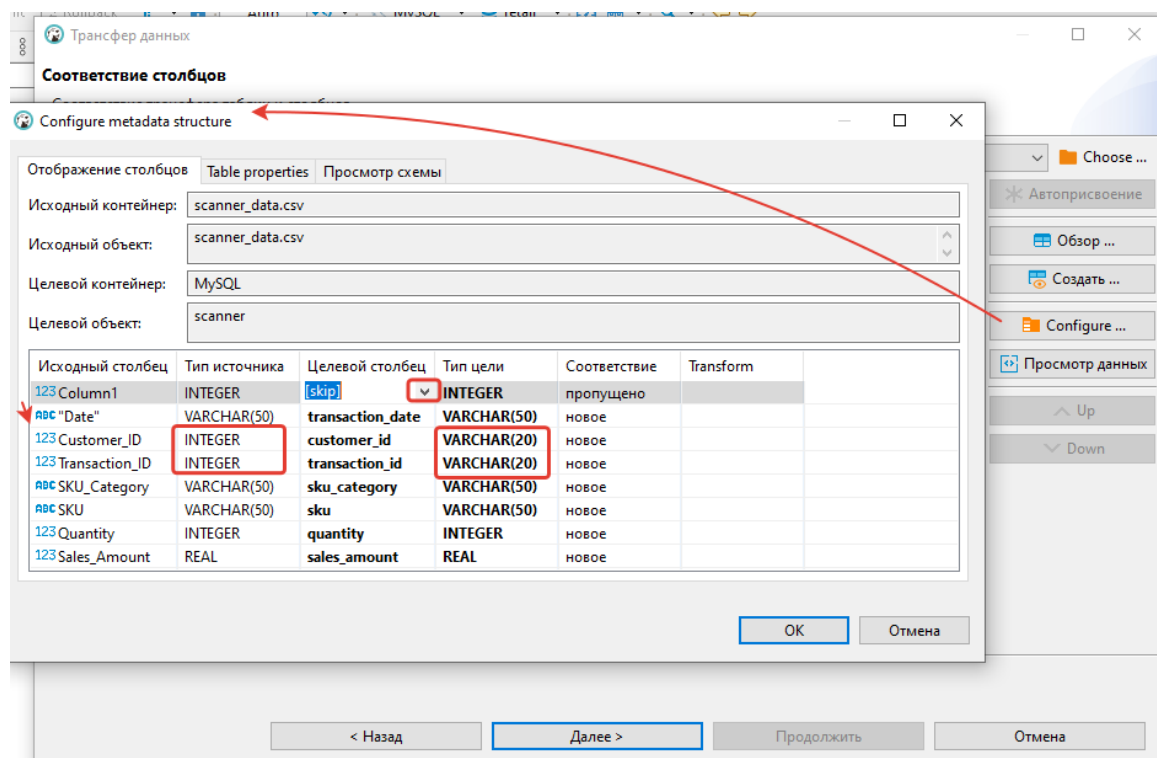


Рис. 1.119. Настройки для импорта столбцов

Проверим настройки загрузки и подтвердим импорт (рис. 1.120, 1.121).

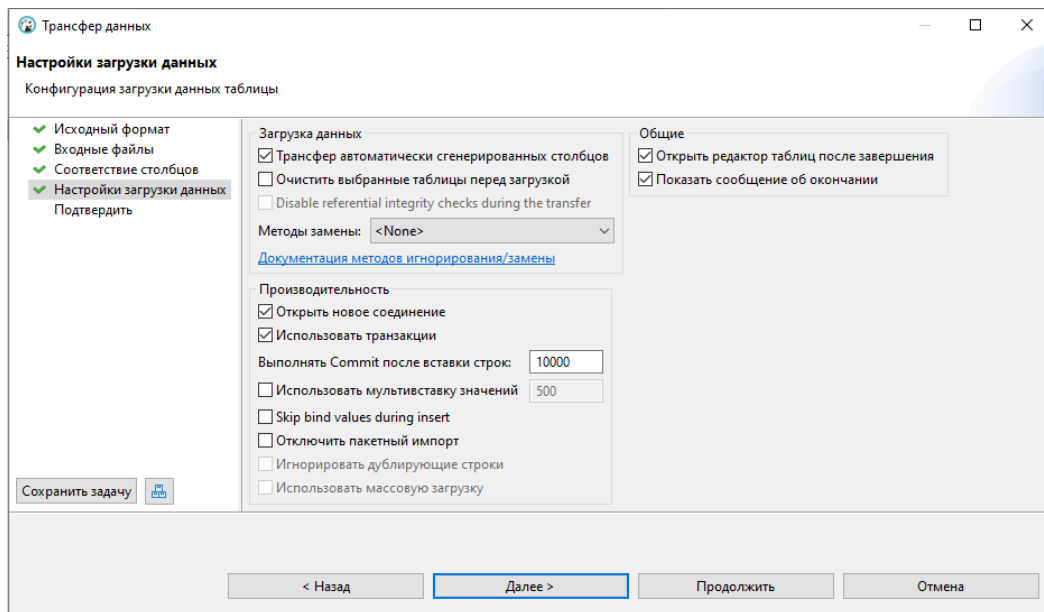


Рис. 1.120. Настройки загрузки данных

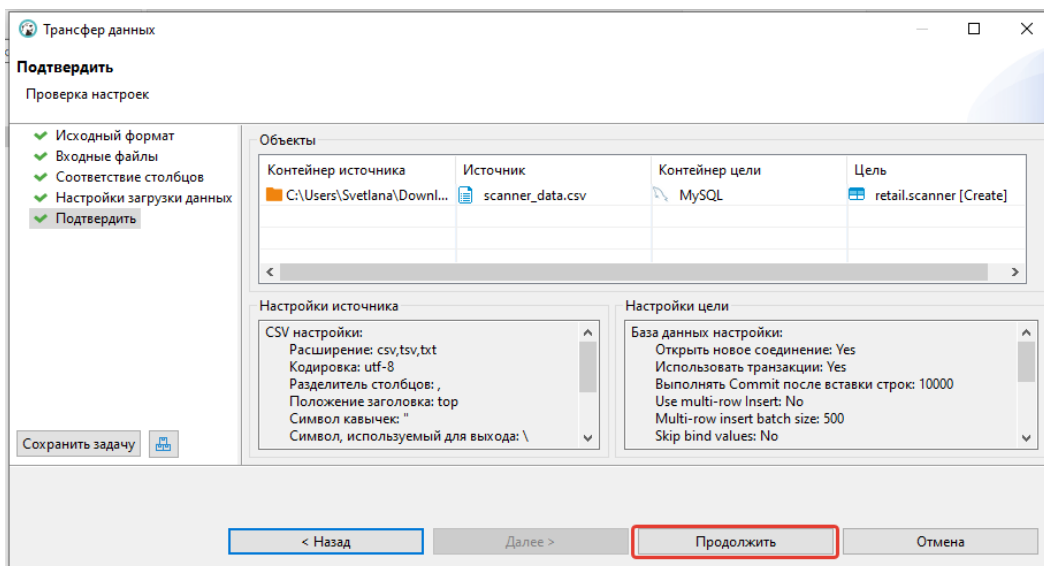


Рис. 1.121. Проверка настроек и подтверждение

Данные импортировались корректно и тип данных соответствует настройкам (рис. 1.122).

	transaction_date	customer_id	transaction_id	sku_category	sku	quantity	sales_amount
1	02/01/2016	2547	1	X52	0EM7L	1	3,13
2	02/01/2016	822	2	2ML	68BRQ	1	5,46
3	02/01/2016	3686	3	0H2	CZUZK	1	6,35
4	02/01/2016	3719	4	0H2	549KK	1	5,59
5	02/01/2016	9200	5	0H2	K8EHK	1	6,88

Рис. 1.122. Таблица scanner (результат импорта данных)

Для столба `transaction_date` тип данных не соответствует содержанию столбца. Далее это будет исправлено. При замене числового типа данных на более общий символьный тип потери данных не произойдет. А вот изменение символьного типа на тип `date` на этапе импорта может привести к потере данных, если данные столбца не будут распознаны как даты в связи с нюансами их представления. Такое преобразование типов лучше выполнять после импорта данных средствами SQL.

2. Работа с данными

В реляционных системах управления базами данных (РСУБД) в качестве высокоуровневого средства выполнения запросов используется язык SQL, который содержит набор операций, необходимых для выполнения действий с базой данных.

Стандарты SQL предусматривают подразделение средств языка на несколько категорий:

– язык определения/описания данных (Data Definition Language, DDL), команды которого определяют структуру базы данных (в том числе логическую) и позволяют работать с объектами этой базы (например, таблицами или представлениями), среди них:

- CREATE – создание нового объекта БД;
- ALTER – модификация/изменение существующего в БД объекта;
- DROP – удаление существующего в БД объекта;
- TRUNCATE – удаление (усечение) всех записей из таблицы или набора таблиц;

– язык манипулирования данными (Data Manipulation Language, DML), команды которого обеспечивают управление данными внутри схемы, позволяют добавлять, изменять, удалять и выгружать данные из базы, среди них:

- SELECT – выборка данных;
- INSERT – вставка новых данных;
- UPDATE – обновление данных;
- DELETE – удаление данных;
- MERGE – слияние данных;

– язык управления данными (Data Control Language, DCL), команды которого обеспечивают управление доступом к данным, т.е. управления привилегиями, разрешающими или запрещающими выполнение определенных операций над объектами базы данных, среди них:

- GRANT – предоставление пользователям привилегии доступа БД;
- REVOKE – отзыв привилегии, предоставленной командой GRANT;
- DENY – задание запрета, имеющего приоритет над разрешением;

– язык управления транзакциями (Transaction Control Language, TCL). Транзакция – это команда или блок DML команд,

которые рассматриваются как единое целое, т.е. в успешной транзакции фиксируются все внесенные изменения, а при сбое в какой-либо команде транзакции все изменения, внесенные командами транзакции, будут отменены. Команды TCL используются для управления изменениями, сделанными транзакцией, среди них:

- BEGIN – определение начала транзакции;
- COMMIT – сохранение изменений;
- SAVEPOINT – определение точки транзакции, к которой можно будет откатиться;
- ROLLBACK – возвращение БД к состоянию на момент последней команды.

Использование верхнего регистра в названиях команд SQL – это всего лишь соглашение, большинство СУБД нечувствительны к регистру.

В пособии будут рассмотрены только конструкции DDL и DML, часть которых уже использовалась в первой части при подготовке программного окружения. Теперь рассмотрим их синтаксис и особенности более системно.

2.1. Типы данных и комментарии

Начнем со знакомства с типами данных. MySQL поддерживает стандартные типы SQL: bigint, int, smallint, real, char(n), varchar(n), date, time, boolean (табл. 2.1–2.4). Эти типы можно считать встроенными.

Таблица 2.1

Числовые типы данных SQL

Имя	Описание	Размер
TINYINT	Очень маленькое целочисленное значение. Значения варьируются от 0 до 255	1 байт
TINYINT UNSIGNED	Значения варьируются от –128 до 127	1 байт
SMALLINT	Малое целочисленное значение. Значения варьируются от 0 до 65535	2 байта
SMALLINT UNSIGNED	Значения варьируются от –32768 до 32767	2 байта
MEDIUMINT	Среднее целочисленное значение. Значения варьируются от 0 до 16777215	3 байта
MEDIUMINT UNSIGNED	Подписанные значения варьируются от –8388608 до 8388607.	3 байта

Имя	Описание	Размер
INT синоним INTEGER	Стандартное целочисленное значение. Значения варьируются от 0 до 4294967295	4 байта
INT UN- SIGNED	Значения варьируются от –2147483648 до 2147483647	4 байта
BIGINT	Большое целочисленное значение. Значения варьируются от 0 до 18446744073709551615	8 байт
BIGINT UN- SIGNED	Значения варьируются от –9223372036854775808 до 9223372036854775807	8 байт
DECI- MAL(preci- sion, scale) синонимы DEC NUMERIC FIXED	Числа с фиксированной точностью. Данный тип может принимать два параметра precision и scale. Параметр precision представляет максимальное количество цифр, которые может хранить число. Это значение должно находиться в диапазоне от 1 до 65. Параметр scale представляет максимальное количество цифр, которые может содержать число после запятой. Это значение должно находиться в диапазоне от 0 до значения параметра precision. По умолчанию оно равно 0	Размер данных в байтах зависит от хранимого значения
FLOAT(m,d)	Число с плавающей точкой одинарной точности. Значения от $-3.4028 * 10^{38}$ до $3.4028 * 10^{38}$. Параметр m – общее количество цифр, d – количество цифр после запятой	4 байта
DOU- BLE(m,d) синонимы DOUBLE PRECI- SION(m,d) REAL(m,d)	Число с плавающей точкой двойной точности. Значения от $-1.7976 * 10^{308}$ до $1.7976 * 10^{308}$. Параметр m – общее количество цифр, d – количество цифр после запятой	8 байт

Для исключения вставки NULL, в определение создаваемого столбца указанного типа добавляется ограничение NOT NULL, а для исключения дублирующихся значений – ограничения UNIQUE или PRIMARY KEY.

Таблица 2.2

Символьные типы данных MySQL

Имя	Описание	Размер
CHAR(size)	Строки с фиксированной длиной. Пространство справа дополняется пробелами до нужного числа символов size. Size – количество сохраняемых символов	Размер до 255 символов
VARCHAR(size)	Строка переменной длины, будет сохранять любое количество символов до size, плюс дополнительный байт, который будет хранить длину строки. Size – максимальное количество сохраняемых символов	Размер до 255 символов
TINYTEXT(size)	Текст неопределенной длины. Size – максимальное количество сохраняемых символов	Размер до 255 символов
TEXT(size)	Текст неопределенной длины. Size – максимальное количество сохраняемых символов	Размер до 6 Кб, или 65535 символов
MEDIUMTEXT(size)	Текст неопределенной длины. Size – максимальное количество сохраняемых символов	Размер до 16 Мб, или 16777215 символов
LONGTEXT(size)	Текст неопределенной длины. Size – максимальное количество сохраняемых символов	Размер до 4 Гб, или 4294967295 символов

Для времени предусмотрены типы дат (date), времени суток (time) и отметок времени (timestamp), которые включают и дату, и время. Для типов, содержащих время (т.е. time и timestamp), имеются разновидности, включающие информацию о часовом поясе.

Тип interval дополнительно позволяет ограничить набор сохраняемых полей указанием на поле: YEAR; MONTH; DAY; HOUR; MINUTE; SECOND или на их комбинацию: YEAR TO MONTH; DAY TO HOUR; HOUR TO MINUTE; MINUTE TO SECOND.

Таблица 2.3

Типы даты/времени MySQL

Имя	Описание	Размер
DATE	Значения варьируются от '1000-01-01' до '9999-12-31'. По умолчанию для хранения используется формат 'yyyy-mm-dd'	3 байта

Имя	Описание	Размер
TIME	Значения варьируются от '-838: 59: 59' до '838: 59: 59'. По умолчанию для хранения времени применяется формат 'hh:mm:ss'	3 байта
DATETIME	Объединяет время и дату, значения варьируются от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. Для хранения по умолчанию используется формат 'yyyy-mm-dd hh:mm:ss'	8 байт
TIMESTAMP	Хранит дату и время, но в другом диапазоне. Значения варьируются от '1970-01-01 00:00:01' UTC до '2038-01-19 03:14:07' UTC	4 байта
YEAR	Значение года как 4 цифры. Диапазон доступных значений от 1901 до 2155	1 байт

Таблица 2.4

Логический тип данных MySQL

Имя	Описание	Размер
BOOL синоним BOOLEAN	Фактически не представляет отдельный тип, а является лишь псевдонимом для типа TINYINT(1) и может хранить два значения 0 и 1. Может также в качестве значения принимать встроенные константы TRUE (представляет число 1) и FALSE (представляет число 0)	1 байт

Для состояния «истина» возможно использование следующих альтернатив: true, yes, on, 1, для состояния «ложь»: false, no, off, 0.

В скриптах SQL для повышения читабельности кода принято добавлять комментарии: для оформления однострочного комментария используется удвоение тире --, многострочный заключается в комбинацию двух символов /* в начале и */ в конце:

```
-- однострочный комментарий
/*
  многострочный
  комментарий
*/
```

2.2. Операции DDL

Рассмотрим основные операторы языка DDL: CREATE, ALTER, DROP.

С оператором CREATE уже сталкивались при создании таблиц баз данных: с помощью MySQL Workbench в п. 1.2 в БД health была создана таблица insurance, а с помощью DBeaver в п. 1.5 в БД retail была создана таблица scanner. Так как эти таблицы создавались при импорте данных из файла с помощью мастера импорта, то оператор CREATE присутствовал в сгенерированном коде, а не использовался напрямую.

Также в п. 1.1 были установлены Samples and Example, в составе которых три демонстрационные БД: sakila, sys и world.

Создадим в DBeaver новую базу данных retail_new (рис. 2.1, 2.2), в которую далее импортируем два файла с данными.

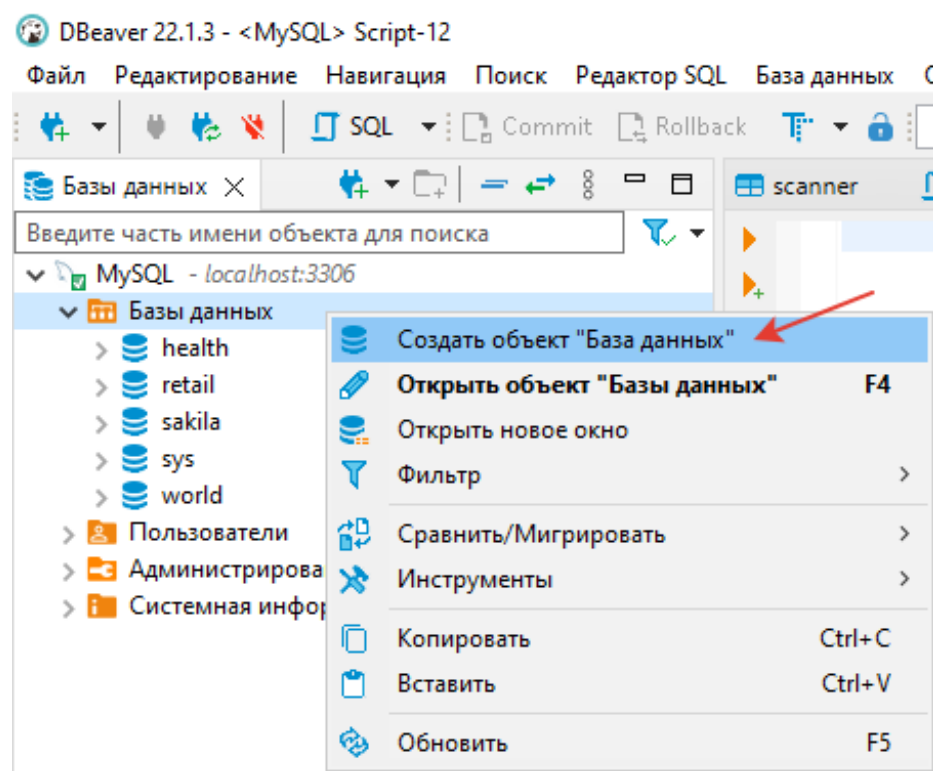


Рис. 2.1. Создание БД

В п. 1.3 при проектировании БД уже использовали как основу два связанных файла данных с ресурса kaggle: данные о транзакциях клиентов и данные реакции клиентов на рекламные взаимодействия (https://www.kaggle.com/datasets/regivm/retailtransactiondata?select=Retail_Data_Transactions.csv и <https://www.kaggle.com/datasets/regivm/>

retailtransactiondata?select=Retail_Data_Response.csv). Каждый из этих файлов будет импортироваться в отдельную таблицу БД retail_new: response и transactions, структуру которых создадим не в момент импорта, а заранее.

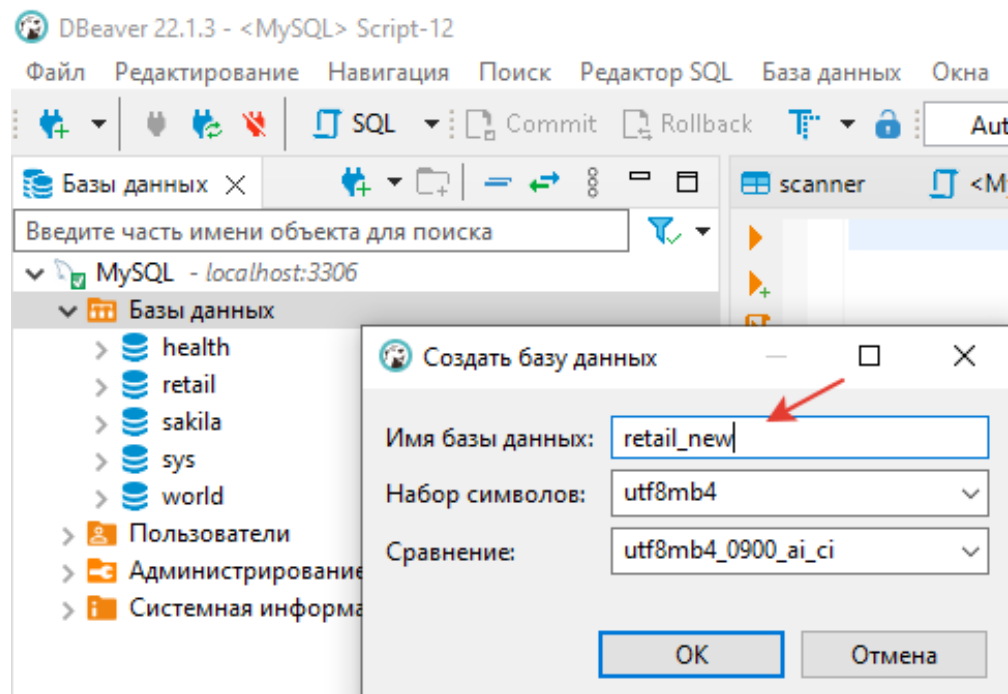


Рис. 2.2. Имя БД

При создании таблицы можно задать не только имена и типы столбцов, но и наложить на определенные столбцы дополнительные, необязательные ограничения и модификаторы. Ограничения столбца влияют только на столбец, для которого они определены. Ограничения, которые включают два или более столбца, должны быть определены как ограничения таблицы.

Ограничения для столбца:

- NOT NULL – запрещает значения NULL, т.е. пропуски в данных столбца;
- UNIQUE – требует уникального значения в соответствующем поле для каждой строки.

Так как NULL не считается значением, то ограничение UNIQUE не гарантирует отсутствие пропусков в данных, т.е. в поле с ограничением UNIQUE могут быть значения NULL и даже не один NULL.

Если в столбце должны быть уникальные значения без пропусков, то у такого столбца должны быть указаны оба ограничения.

Произвольные ограничения задаются пользователем с помощью CHECK (expression).

В роли ограничения также выступает назначение столбца или набора столбцов на роль первичного ключа. Если набор столбцов обозначен как PRIMARY KEY, то первичный ключ составной. Каждая таблица может иметь только один первичный ключ.

Обозначение PRIMARY KEY подразумевает ограничение UNIQUE, так как значения первичного ключа должны быть уникальными. В случае составного первичного ключа требуется уникальность комбинации значений из набора полей в одной строке.

Первичный ключ также не должен содержать пропусков. Если первичный ключ – это столбец с целочисленными значениями, т.е. столбец INT PRIMARY KEY, то такой столбец может быть дополнительно помечен AUTO_INCREMENT, т.е. первичный ключ в таком столбце при вставке нового значения будет генерироваться автоматически с соблюдением условия уникальности и отсутствия пропусков.

Помимо PRIMARY KEY, столбцы также могут быть помечены как FOREIGN KEY. Внешний ключ (FOREIGN KEY) – это способ обеспечить ссылочную целостность в базе данных MySQL. Таблица, на поле которой ссылается внешний ключ называется родительской таблицей, а таблица с внешним ключом, называется дочерней таблицей. Внешний ключ в дочерней таблице обычно ссылается на первичный ключ (PRIMARY KEY) в родительской таблице.

В определении таблиц базы данных retail_new, зададим имена колонок/полей и тип данных в них, и первичный ключ в первой таблице response.

Можно задать связь поля второй таблицы transactions с первичным ключом первой, создав таким образом внешний ключ. Однако данные забираются из источника, который, возможно, не выдержит проверку ограничений внешнего ключа, поэтому внешний ключ будет установлен уже после импорта. В поле trans_date данные также не удовлетворяют формату поля типа Date, который используется в мастере импорта данных. Поэтому для данного поля установим более общий строковый формат, а уже после импорта выполним необходимые преобразования.

Откроем скрипт SQL для базы данных retail_new (рис. 2.3).

Создадим на основе данных скачанных файлов две таблицы: response и transactions с соответствующими полями и типами данных (рис. 2.4).

```
CREATE TABLE response (  
    customer_id CHAR(10) NOT NULL,
```

```

response INT NULL,
PRIMARY KEY(customer_id)
);

CREATE TABLE transactions (
customer_id CHAR(10) NOT NULL,
trans_date VARCHAR(10),
tran_amount DOUBLE
);

```

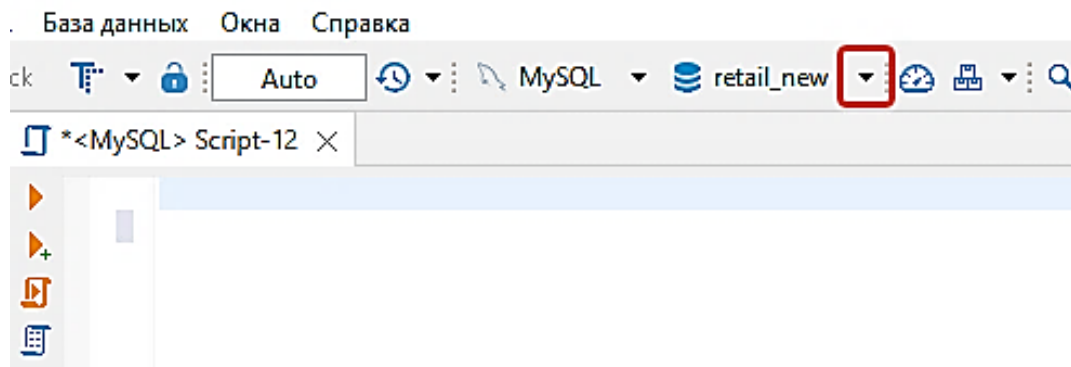


Рис. 2.3. Редактор SQL для базы данных retail_new

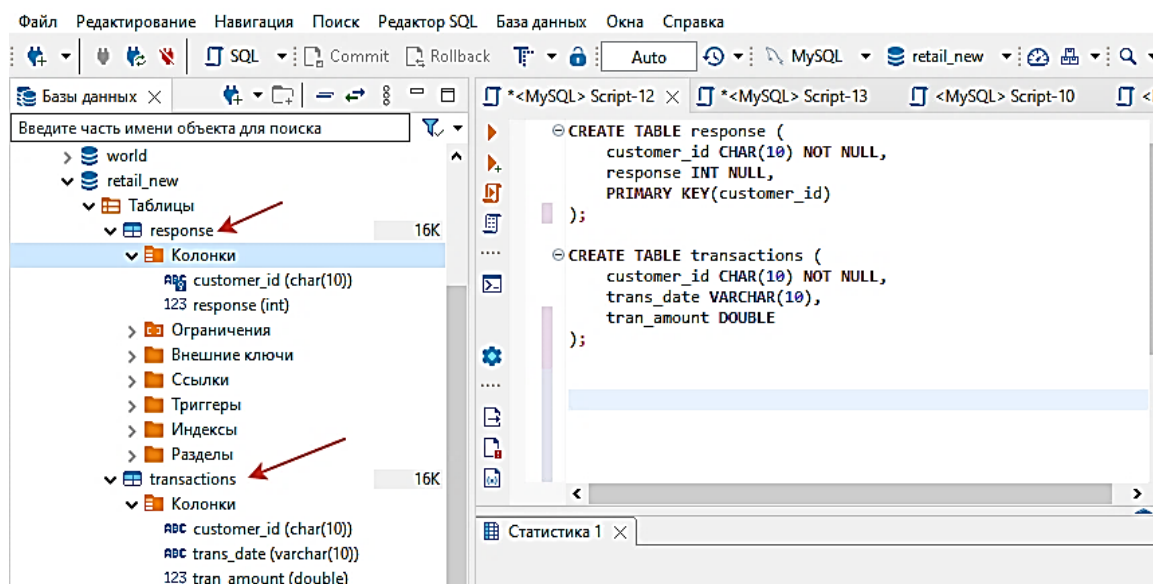


Рис. 2.4. Создание таблиц

После выполнения кода будут созданы две таблицы, в которые импортируем данные из файлов (см. п. 1.5), каждый в соответствующую таблицу. И выведем содержимое этих таблиц на экран (рис. 2.5).

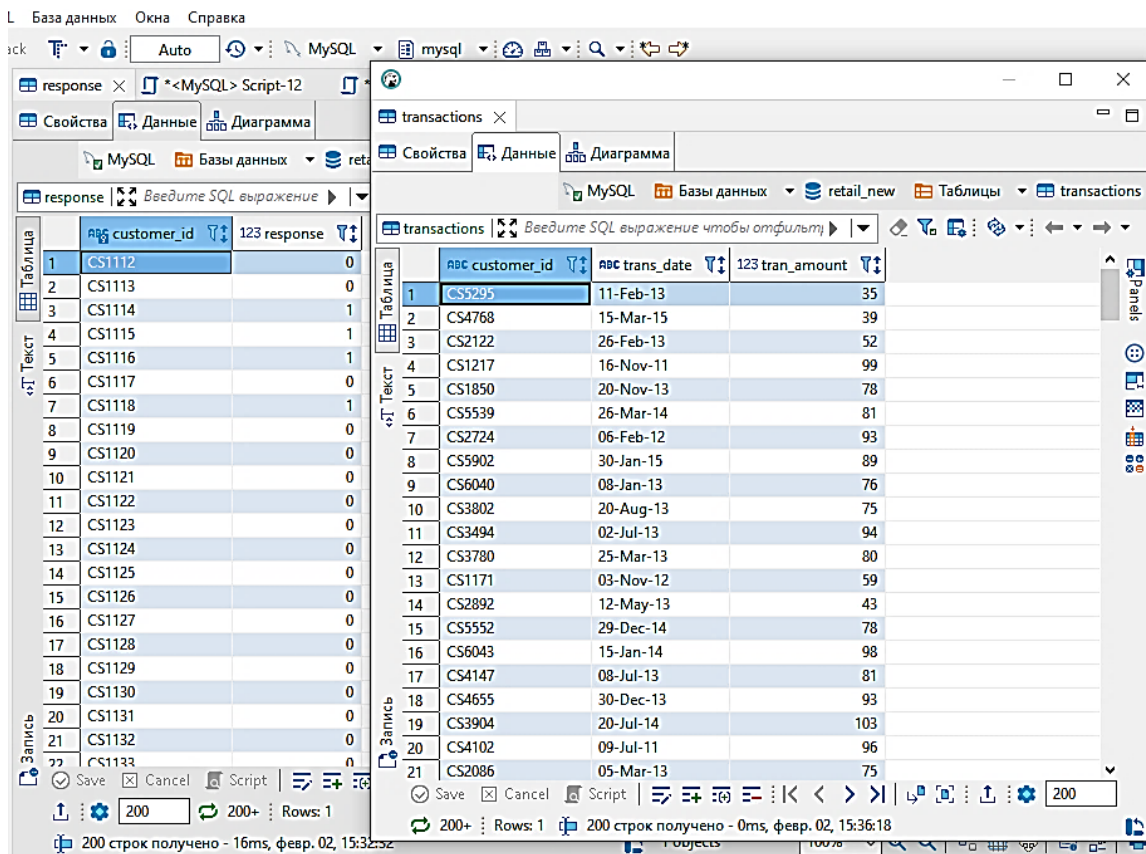


Рис. 2.5. Параллельный просмотр содержимого двух таблиц базы данных retail_new

В MySQL можно использовать оператор ALTER TABLE для добавления столбца или добавления ограничений.

Возможная проблема с проверкой ограничений внешнего ключа не позволила сразу при создании таблицы transactions использовать код для установления связи с таблицей response.

FOREIGN KEY (customer_id) REFERENCES response(customer_id)

Если в таблице response нет каких-то клиентов, которые совершали транзакции, то внешний ключ не может пройти проверку: родительская таблица response не одержит значения в ключевом поле response.customer_id, которые есть в связанной таблице transactions в поле .transactions.customer_id.

Используем оператор ALTER TABLE для добавления ограничения внешнего ключа к уже созданной таблице.

Отключаем сопоставление внешнего ключа с другими таблицами:

SET FOREIGN_KEY_CHECKS=0;

Только после успешного выполнения этой команды приступаем к установлению внешнего ключа в таблице transactions:

```
ALTER TABLE transactions  
ADD FOREIGN KEY (customer_id) REFERENCES response(customer_id);
```

После завершения выполнения команды подключаем проверку условия внешнего ключа:

```
SET FOREIGN_KEY_CHECKS=1;
```

Важно выполнять эти команды по отдельности. Схема для таблиц БД retail_new представлена на рис. 2.6.

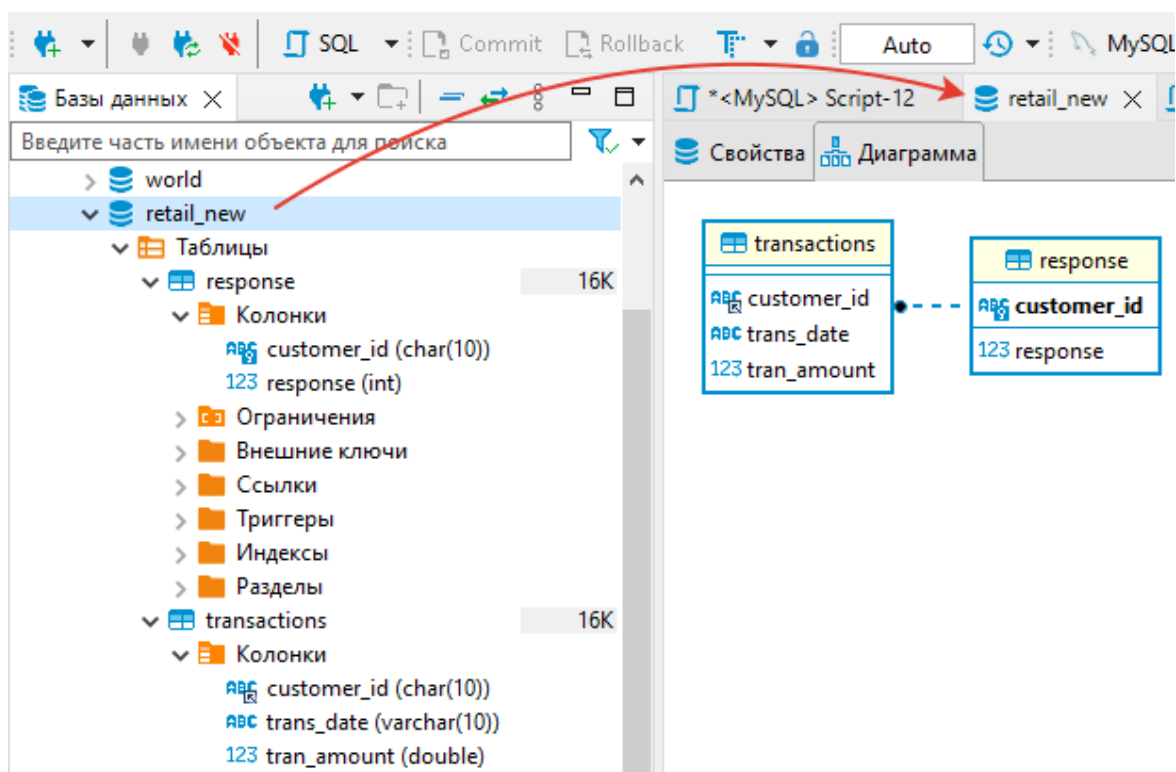


Рис. 2.6. Схема двух таблиц базы данных retail_new

Выполним преобразование для типа данных столбца, содержащего дату (trans_date). Добавим в таблицу transactions новый столбец transaction_date с типом данных DATE:

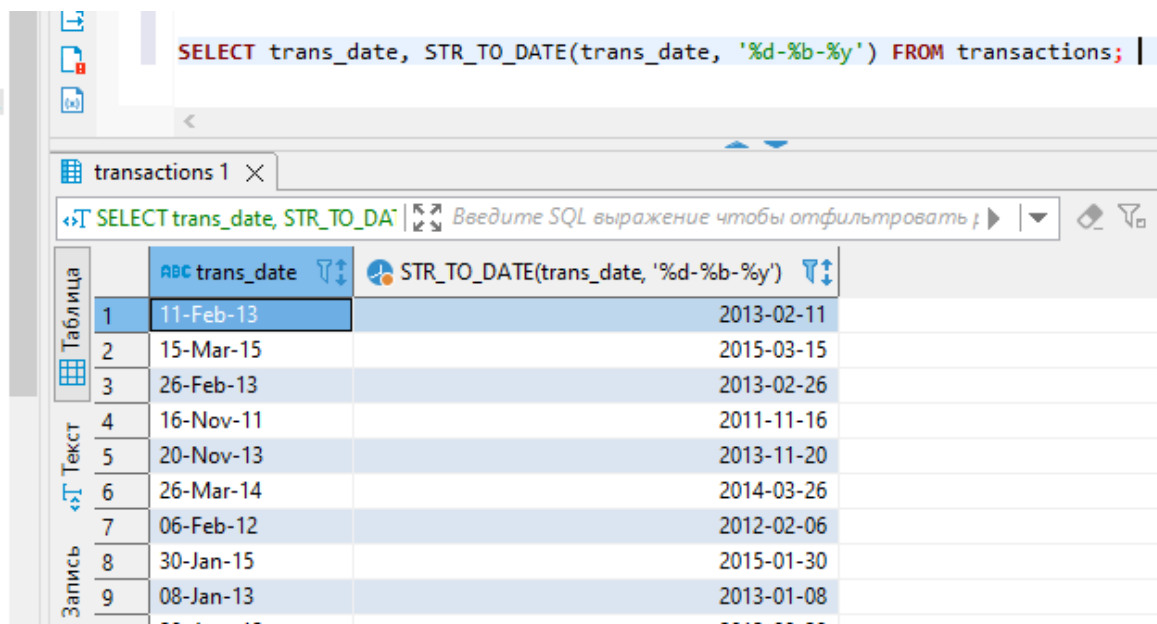
```
ALTER TABLE transactions  
ADD transaction_date DATE NULL;
```

Используем функцию STR_TO_DATE (), чтобы преобразовать строку в дату. Аргументом той функции выступает формат представления даты в строке, а преобразованный вид будет уууу-mm-dd.

В столбце trans_date шаблон для даты '%d-%b-%y' (день, сокращенное название месяца и последние две цифры года, разделенные тире).

Выведем на просмотр результат преобразования этой функцией данных столбца trans_date (рис. 2.7):

```
SELECT trans_date, STR_TO_DATE(trans_date, '%d-%b-%y') FROM transactions;
```

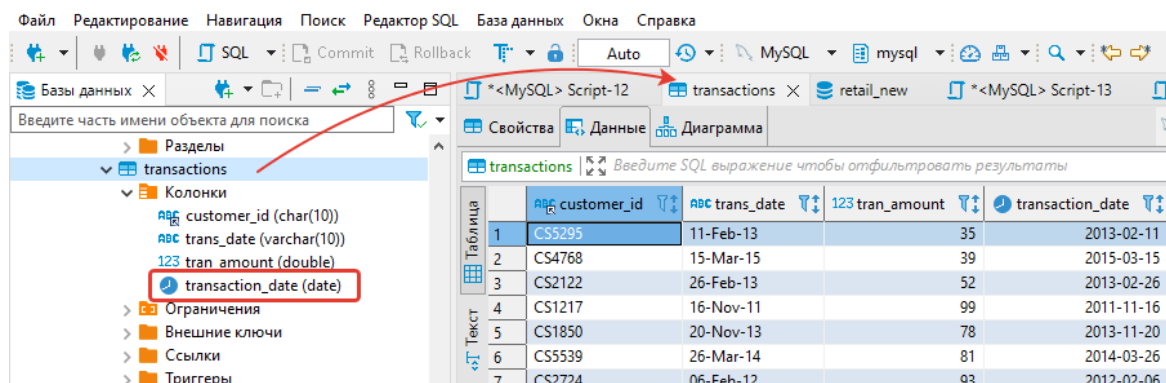


	trans_date	STR_TO_DATE(trans_date, '%d-%b-%y')
1	11-Feb-13	2013-02-11
2	15-Mar-15	2015-03-15
3	26-Feb-13	2013-02-26
4	16-Nov-11	2011-11-16
5	20-Nov-13	2013-11-20
6	26-Mar-14	2014-03-26
7	06-Feb-12	2012-02-06
8	30-Jan-15	2015-01-30
9	08-Jan-13	2013-01-08

Рис. 2.7. Результат выполнения запроса на преобразование в дату

Запишем результат применения функции STR_TO_DATE () к столбцу trans_date в столбец transaction_date (рис. 2.8).

```
UPDATE transactions  
SET transaction_date = STR_TO_DATE(trans_date, '%d-%b-%y');
```



	customer_id	trans_date	tran_amount	transaction_date
1	CS5295	11-Feb-13	35	2013-02-11
2	CS4768	15-Mar-15	39	2015-03-15
3	CS2122	26-Feb-13	52	2013-02-26
4	CS1217	16-Nov-11	99	2011-11-16
5	CS1850	20-Nov-13	78	2013-11-20
6	CS5539	26-Mar-14	81	2014-03-26
7	CS2724	06-Feb-12	93	2012-02-06

Рис. 2.8. Сохранение преобразованных в дату значений в столбец transaction_date

Удалим столбец trans_date:

```
ALTER TABLE transactions DROP COLUMN trans_date;
```

Результат всех преобразований для таблицы transactions на рис. 2.9.

The screenshot shows the MySQL Workbench interface. On the left, the 'Свойства' (Properties) tab is active for the 'transactions' table. It lists columns: 'customer_id' (char(10)), 'tran_amount' (double), and 'transaction_date' (date). Below, it shows a primary key on 'customer_id'. On the right, the 'Данные' (Data) tab shows a table with 12 rows. The columns are 'customer_id', 'tran_amount', and 'transaction_date'.

	customer_id	tran_amount	transaction_date
1	CS5295	35	2013-02-11
2	CS4768	39	2015-03-15
3	CS2122	52	2013-02-26
4	CS1217	99	2011-11-16
5	CS1850	78	2013-11-20
6	CS5539	81	2014-03-26
7	CS2724	93	2012-02-06
8	CS5902	89	2015-01-30
9	CS6040	76	2013-01-08
10	CS3802	75	2013-08-20
11	CS3494	94	2013-07-02
12	CS2780	80	2012-02-25

Рис. 2.9. Таблица transactions

Откроем скрипт SQL для БД retail. В этой БД есть таблица scanner, в которую были импортированы данные из файла и также для столбца с датой был выбран строковый тип данных. Проведем аналогичные преобразования (рис. 2.10) для таблицы scanner:

```
ALTER TABLE scanner
```

```
ADD transaction_date_new DATE NULL;
```

```
SELECT STR_TO_DATE(transaction_date, '%d/%m/%Y') FROM scanner;
```

```
UPDATE scanner
```

```
SET transaction_date_new = STR_TO_DATE(transaction_date, '%d/%m/%Y');
```

```
ALTER TABLE scanner DROP COLUMN transaction_date;
```

Добавим также в таблицу scanner столбец price, как результат отношения sales_amount – стоимости проданного товара к quantity – количеству проданного товара (рис. 2.11):

```
ALTER TABLE scanner
```

```
ADD price DOUBLE AS (sales_amount / quantity);
```

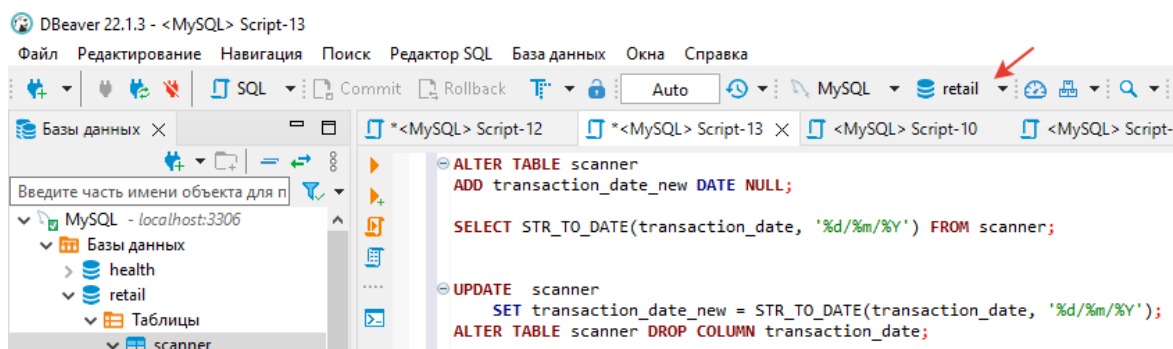


Рис. 2.10. Преобразование строковых данных в даты для таблицы scanner

	customer_id	transaction_id	sku_category	sku	quantity	sales_amount	transaction_date_new	price
43	3071	26	N8U	UNIKW	2	4,23	2016-01-02	2,115
44	2065	27	T80	LP9DJ	1	6,63	2016-01-02	6,63
45	8566	28	SFC	XHB4M	1	2,13	2016-01-02	2,13
46	8303	29	FEW	QWSWG	1	5,12	2016-01-02	5,12
47	4776	30	YMU	3CUG7	1	5,38	2016-01-02	5,38
48	6459	31	EPM	DFFBD	1	1,5	2016-01-02	1,5
49	6459	31	JKC	OTLU5	2	2,13	2016-01-02	1,065
50	2546	32	O38	O1RNE	1	3,63	2016-01-02	3,63
51	1595	33	Z4O	8ZJKU	1	2	2016-01-02	2
52	4022	34	OH2	CZUZX	1	6,35	2016-01-02	6,35
53	4022	34	OH2	LQ555	1	7,57	2016-01-02	7,57
54	2438	35	P4P	5V2V7	1	4,57	2016-01-02	4,57

Рис. 2.11. Схема обновленной таблицы scanner

Новые столбцы предпочтительнее добавлять в представление БД, т.е. в сохраненный результат запроса на вывод данных таблицы с вычисленными значениями в новом столбце.

2.3. Операции DML: INSERT, UPDATE, DELETE

MySQL поддерживает основные операторы языка DML: SELECT, INSERT, UPDATE, DELETE.

Начнем знакомство с DML с INSERT, UPDATE, DELETE. Это команды изменения содержимого таблиц: обновление, вставка или удаление строк.

Оператор UPDATE использовался в п. 2.2. для обновления содержимого созданных столбцов с типом данных DATE.

Для демонстрации работы операторов INSERT, UPDATE, DELETE используем базу данных retail_new и таблицу transactions, которая была изменена в предыдущем пункте.

Синтаксис для оператора INSERT:

```
INSERT INTO table
(column1, column2, ... )
```

```
VALUES  
(expression1, expression2, ... ),  
(expression1, expression2, ... ),  
...;
```

Вставим строку с фиктивными данными в таблицу transactions: ('CS4768', 67, '2015-01-01'). Если не перечисляются имена столбцов, в которые записываются данные, то вставляемые значения должны соответствовать порядку перечисления имен столбцов при создании таблицы.

Однако если в любом произвольном порядке и необязательно в полном составе указать имена столбцов, в которые планируется добавить данные, то вставляемые значения должны соблюдать тот же порядок следования.

```
INSERT INTO transactions  
VALUES ('CS4768', 67, '2015-01-01');
```

Запросим все транзакции этого заказчика (рис. 2.12).

```
SELECT *  
FROM transactions  
WHERE customer_id = 'CS4768';
```

Синтаксис для оператора UPDATE:

```
UPDATE table  
  SET column1 = expression1 | DEFAULT,  
      column2 = expression2 | DEFAULT,  
      ...  
  [WHERE conditions];
```

Обновим только что внесенную запись, изменив дату на '2016-01-03'. Обратите внимание, что при обновлении обязательно указывается условие для отбора записей, в которых произойдет обновление (WHERE <условие>), иначе изменение будет применено для этого столбца/столбцов во всех записях. Условий для определения нужных для обновления записей может быть несколько. Так, если не добавить кроме идентификатора заказчика еще и дату транзакции, то при отборе записей во всех записях для этого заказчика дата транзакции будет обновлена на новую.

```
UPDATE transactions  
SET transaction_date = '2016-01-03'  
WHERE customer_id = 'CS4768' AND transaction_date = '2015-01-01';
```

Выполним также запрос на вывод транзакций заказчика с этим customer_id (рис. 2.13):

```
SELECT *  
FROM transactions  
WHERE customer_id = 'CS4768';
```

The screenshot shows a MySQL IDE interface. The top menu bar includes 'ск', 'Редактор SQL', 'База данных', 'Окна', and 'Справка'. Below the menu, there are buttons for 'Commit', 'Rollback', and a status bar showing 'Auto'. The database name 'retail_new' is selected in the top right. The main editor window displays two SQL scripts: an INSERT statement and a SELECT statement. The SELECT statement is highlighted, and its results are shown in a table view below. The table has three columns: 'customer_id', 'tran_amount', and 'transaction_date'. The results show 14 rows of data for customer_id 'CS4768'. A red arrow points to the last row, which has a transaction date of '2015-01-01'.

customer_id	tran_amount	transaction_date
CS4768	56	2012-03-26
CS4768	75	2014-05-17
CS4768	62	2012-01-01
CS4768	48	2014-08-27
CS4768	43	2013-03-22
CS4768	35	2013-03-25
CS4768	105	2013-06-02
CS4768	71	2011-06-18
CS4768	68	2013-09-04
CS4768	78	2014-10-16
CS4768	40	2014-05-16
CS4768	37	2013-08-20
CS4768	89	2013-10-24
CS4768	86	2013-07-25
CS4768	67	2015-01-01

Рис. 2.12. Запрос на вывод внесенной записи

Commit Rollback Auto MySQL retail_new

*<MySQL> Script-12 x *<MySQL> Script-13 transactions <MySQL>

```

UPDATE transactions
SET transaction_date = '2016-01-03'
WHERE customer_id = 'CS4768' AND transaction_date = '2015-01-03';

SELECT *
FROM transactions
WHERE customer_id = 'CS4768';

```

transactions 1 x

SELECT * FROM transactions WHERE

Таблица	customer_id	tran_amount	transaction_date
13	CS4768	56	2012-03-26
14	CS4768	75	2014-05-17
15	CS4768	62	2012-01-01
16	CS4768	48	2014-08-27
17	CS4768	43	2013-03-22
18	CS4768	35	2013-03-25
19	CS4768	105	2013-06-02
20	CS4768	71	2011-06-18
21	CS4768	68	2013-09-04
22	CS4768	78	2014-10-16
23	CS4768	40	2014-05-16
24	CS4768	37	2013-08-20
25	CS4768	89	2013-10-24
26	CS4768	86	2013-07-25
27	CS4768	67	2016-01-03

Введите SQL выражение чтобы отфильтровать

Рис. 2.13. Результат обновления записи

Синтаксис оператора DELETE:

```
DELETE FROM table
[WHERE conditions];
```

Удалим тренировочную запись из исходного набора данных. Так же, как и в случае обновления, чтобы не произошло удаление всех записей таблицы, необходимо указать условие для отбора нужных записей:

```
DELETE FROM transactions
WHERE customer_id = 'CS4768' AND transaction_date = '2016-01-03';
```

И снова выполним запрос на вывод записей этого заказчика, чтобы убедиться, что удаление произошло (рис. 2.14):

```
SELECT *
FROM transactions
WHERE customer_id = 'CS4768';
```

The screenshot shows a MySQL IDE interface. The top toolbar includes buttons for Commit, Rollback, and Auto. The script editor displays four SQL statements: an UPDATE statement (commented out), a SELECT statement (commented out), a DELETE statement (highlighted with a red box and an arrow pointing to it), and another SELECT statement (commented out). Below the script editor, the 'transactions 1' query result is shown in table format.

	customer_id	tran_amount	transaction_date
16	CS4768	48	2014-08-27
17	CS4768	43	2013-03-22
18	CS4768	35	2013-03-25
19	CS4768	105	2013-06-02
20	CS4768	71	2011-06-18
21	CS4768	68	2013-09-04
22	CS4768	78	2014-10-16
23	CS4768	40	2014-05-16
24	CS4768	37	2013-08-20
25	CS4768	89	2013-10-24
26	CS4768	86	2013-07-25

Рис. 2.14. Результат удаления записи

Далее более подробно остановимся на самом часто используемом операторе SELECT.

2.4. Операции DML: SELECT

Команда SELECT используется для извлечения данных из базы данных. Любая операция чтения из базы данных на языке SQL задается оператором SELECT.

В своей простейшей форме оператор SELECT имеет синтаксис:

```
SELECT expressions  
FROM tables  
[WHERE conditions];
```

Оператор SELECT получает строки из множества таблиц (возможно, пустого). В предложении FROM перечисляются одна или несколько таблиц, служащих источниками данных для SELECT. Если указано несколько источников, результатом будет декартово произведение (перекрестное соединение) всех их строк. В предложении FROM указывается имя таблицы или представления и очень часто алиас (alias, псевдоним), т.е. альтернативное имя для элемента списка FROM, оно используется для сокращения записи и полностью скрывает настоящее имя элемента.

Псевдонимы (алиасы) SQL можно использовать для создания временного имени для столбцов или таблиц.

Псевдонимы столбцов используются для облегчения чтения заголовков столбцов в результирующем наборе.

Псевдонимы таблиц используются для сокращения SQL-кода, чтобы его было легче читать.

Синтаксис для задания псевдонимов столбцов/таблиц:

```
column_name [ AS ] alias_name  
table_name [ AS ] alias_name
```

Ключевое слово AS не является обязательным в MySQL, его можно опустить.

Откроем новый скрипт для retail_new (создана в п. 1.4) и выполним запрос к таблице transactions, имя которой заменим на псевдоним t, также на псевдонимы заменим и имена столбцов и выставим условие, что сумма должна быть больше 100 (рис. 2.15):

```
SELECT customer_id, transaction_date AS d, tran_amount AS a  
FROM transactions AS t  
WHERE tran_amount>100;
```

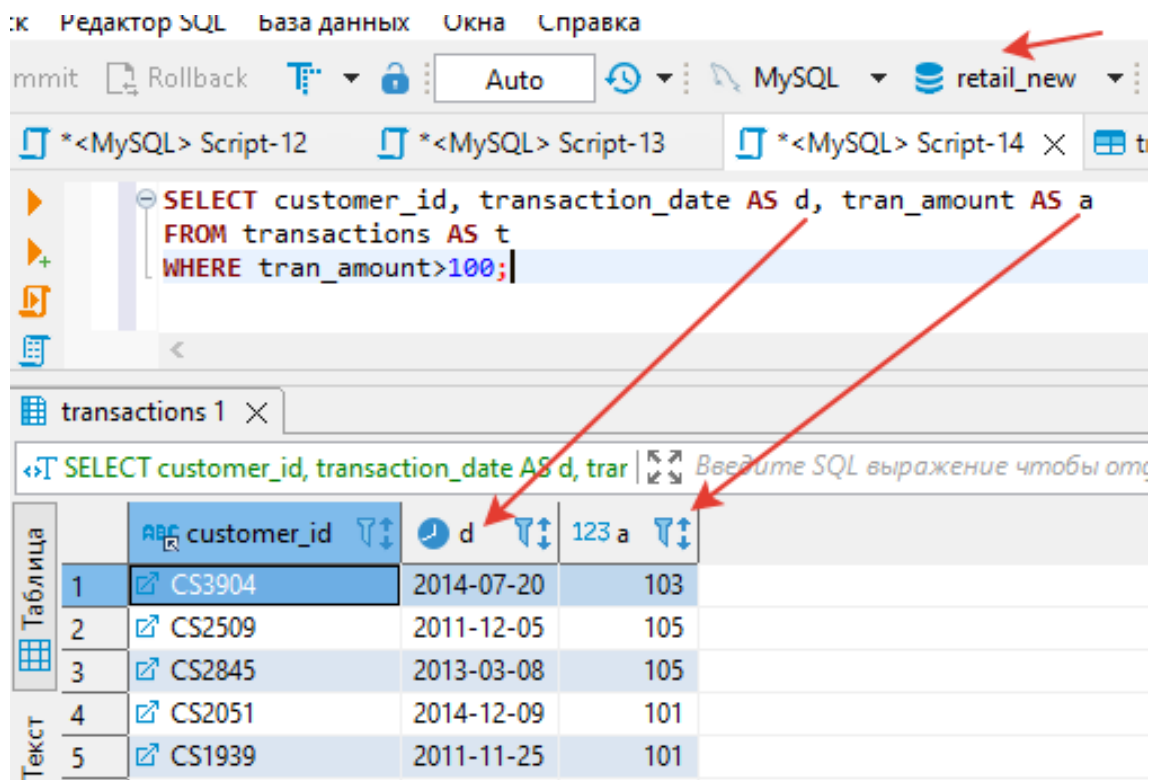


Рис. 2.15. Использование псевдонимов при запросе на записи с суммой больше 100 из таблицы transactions

Просмотреть следующую порцию записей, если количество строк в запросе большое, можно по кнопке  внизу окна с результатом запроса, а кнопка  позволяет подсчитать точное число строк.

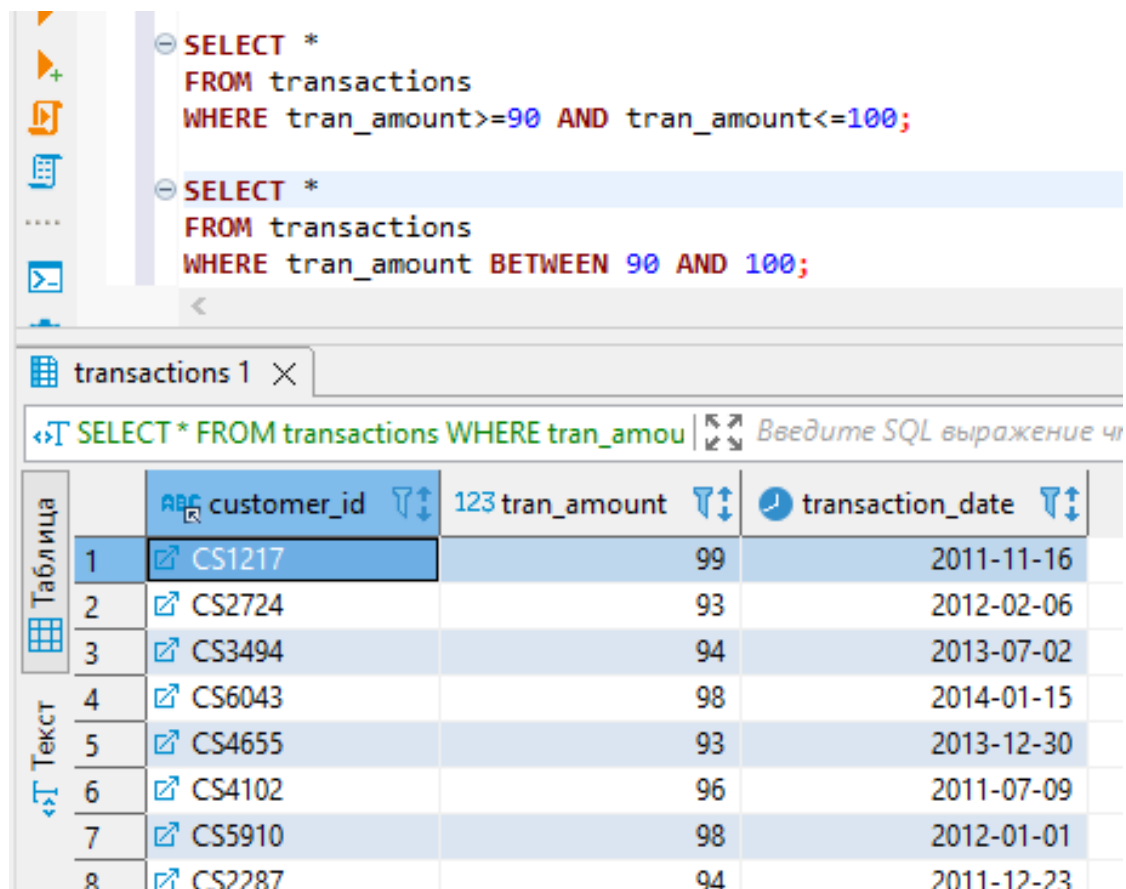
2.4.1 Условие WHERE, операторы сравнения LIKE и IN

Необязательное предложение WHERE содержит список условий (выражений), выдающих результат типа boolean (т.е. логический тип данных TRUE/FALSE, но также возможно и значение NULL). Строки, не удовлетворяющие комбинации условий, исключаются из результата.

Рассмотрим множественные условия в запросах. Например, выберем из таблицы transactions тех заказчиков, у которых значение суммы в транзакции находится в интервале от 90 до 100 (с включением границ интервала). Две альтернативные конструкции с использованием ключевого слова BETWEEN и без него дают идентичный результат (рис. 2.16):

```
SELECT *
FROM transactions
WHERE tran_amount >= 90 AND tran_amount <= 100;
```

```
SELECT *
FROM transactions
WHERE tran_amount BETWEEN 90 AND 100;
```



The screenshot shows a database query tool interface. At the top, two SQL queries are listed in a panel:

```
SELECT *
FROM transactions
WHERE tran_amount >= 90 AND tran_amount <= 100;
```

```
SELECT *
FROM transactions
WHERE tran_amount BETWEEN 90 AND 100;
```

Below the queries, a tab labeled "transactions 1" is active. The SQL editor shows the query: `SELECT * FROM transactions WHERE tran_amo`. Below the editor, a result table is displayed with the following data:

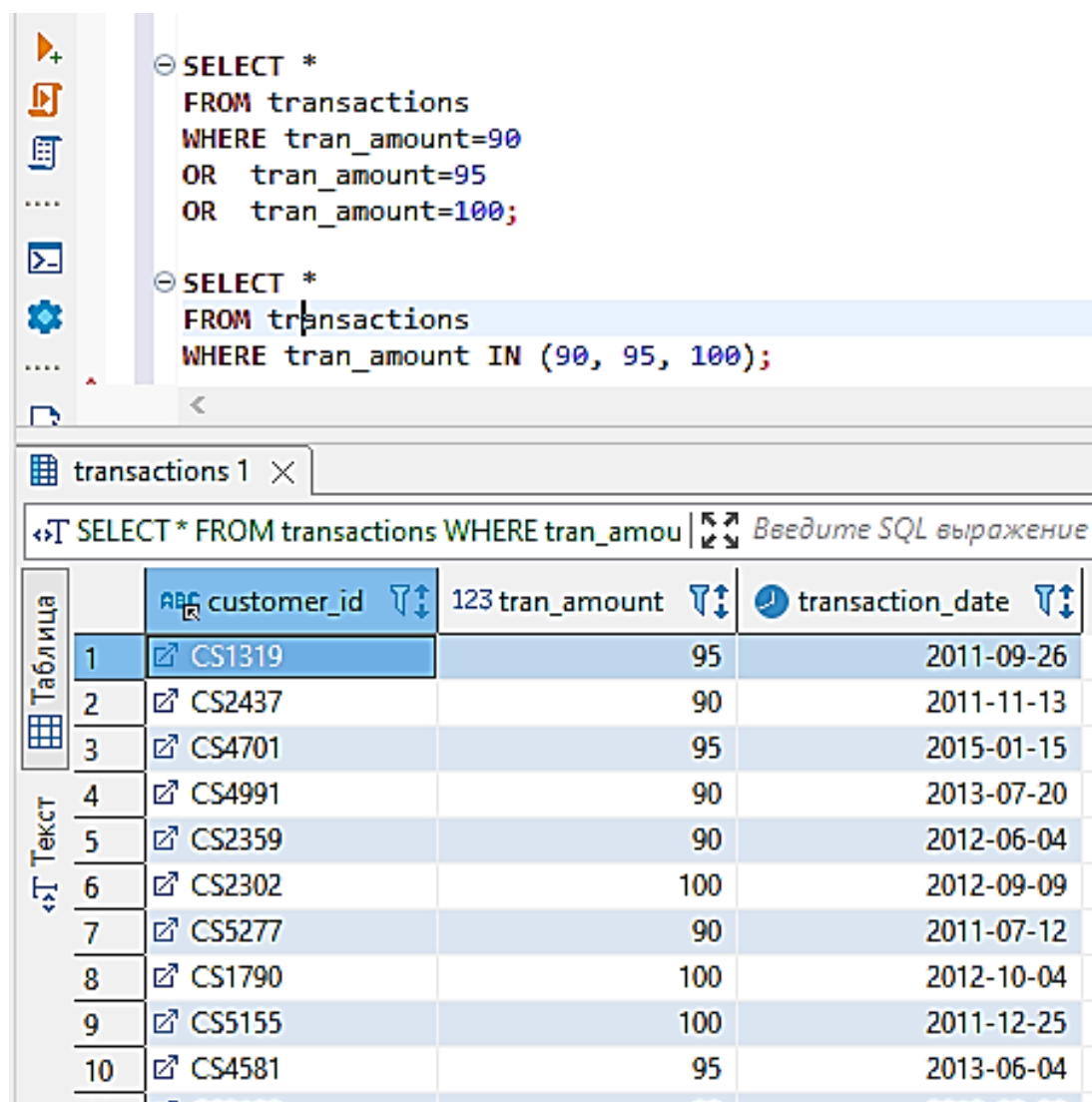
Таблица	customer_id	tran_amount	transaction_date
1	CS1217	99	2011-11-16
2	CS2724	93	2012-02-06
3	CS3494	94	2013-07-02
4	CS6043	98	2014-01-15
5	CS4655	93	2013-12-30
6	CS4102	96	2011-07-09
7	CS5910	98	2012-01-01
8	CS2287	94	2011-12-23

Рис. 2.16. Результат выполнения запроса
(сумма транзакции в интервале от 90 до 100)

Если же необходимо выбрать все записи, в которых поле `tran_amount` принимает несколько конкретных значений, например 90, 95 и 100, то можно использовать отдельные равенства, соединив их OR, или использовать IN перечислив в списке все нужные значения (рис. 2.17):

```
SELECT *
FROM transactions
WHERE tran_amount=90
OR tran_amount=95
OR tran_amount=100;
```

```
SELECT *
FROM transactions
WHERE tran_amount IN (90, 95, 100);
```



```

SELECT *
FROM transactions
WHERE tran_amount=90
OR tran_amount=95
OR tran_amount=100;

SELECT *
FROM transactions
WHERE tran_amount IN (90, 95, 100);

```

Таблица	customer_id	tran_amount	transaction_date
1	CS1319	95	2011-09-26
2	CS2437	90	2011-11-13
3	CS4701	95	2015-01-15
4	CS4991	90	2013-07-20
5	CS2359	90	2012-06-04
6	CS2302	100	2012-09-09
7	CS5277	90	2011-07-12
8	CS1790	100	2012-10-04
9	CS5155	100	2011-12-25
10	CS4581	95	2013-06-04

Рис. 2.17. Результат выполнения запроса
(сумма транзакции принимает одно из значений 90, 95 и 100)

Для поиска заданного шаблона в предложении WHERE используется оператор LIKE. В шаблоне кроме искоемых символов можно использовать подстановочные знаки: знак процента «%» используется вместо любого числа символов от 0 до фактически максимально возможного в данном поле, а знак нижнего подчеркивания «_» заменяет собой один любой символ.

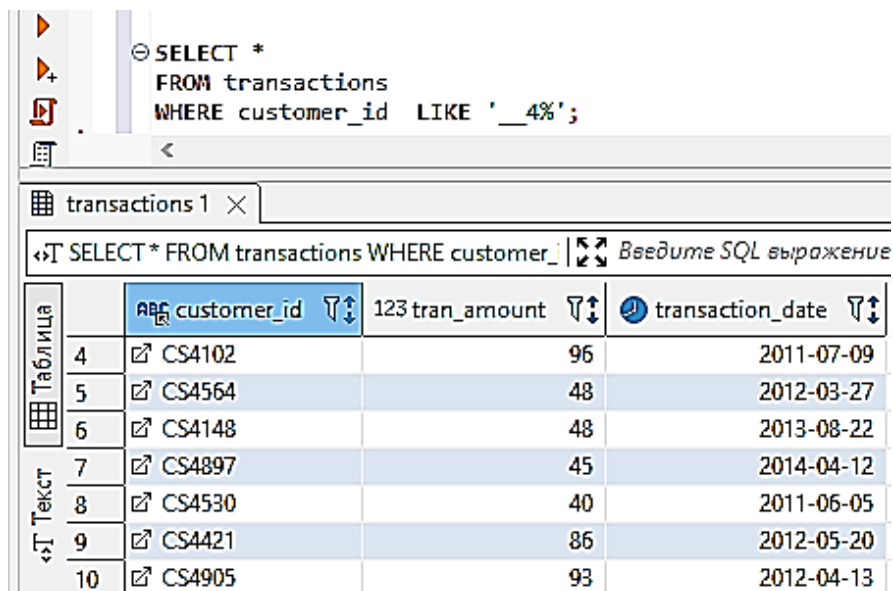
Прочитаем из таблицы transactions те записи, для которых идентификатор заказчика содержит цифру 4 на третьей позиции.

Шаблон для поиска '__4%', альтернативный вариант 'CS4%'. В результате получим 19818 записей (рис. 2.18):

```

SELECT *
FROM transactions
WHERE customer_id LIKE '__4%';

```



```

SELECT *
FROM transactions
WHERE customer_id LIKE '__4%';

```

	customer_id	tran_amount	transaction_date
4	CS4102	96	2011-07-09
5	CS4564	48	2012-03-27
6	CS4148	48	2013-08-22
7	CS4897	45	2014-04-12
8	CS4530	40	2011-06-05
9	CS4421	86	2012-05-20
10	CS4905	93	2012-04-13

Рис. 2.18. Результат выполнения запроса
(customer_id соответствует шаблону '__4%')

Возможные операторы сравнения перечислены в табл. 2.5.

Таблица 2.5

Операторы сравнения

Операторы сравнения	Описание
=	Равно
<>	Не равно
!=	Не равно
>	Больше чем
>=	Больше или равно
<	Меньше чем
<=	Меньше или равно
IN ()	Соответствует значению в списке
NOT	Отрицает условие
BETWEEN	В пределах диапазона (включительно)
IS NULL	Значение NULL
NOT NULL	Не NULL значение
LIKE	Сопоставление с шаблоном % и _

Запросим все транзакции с 1 марта 2014 г. по 31 марта 2014 г. Это можно сделать несколькими способами, например, указав промежуток, к которому принадлежат даты или месяц и год, за который нужна информация или даже шаблон, в соответствии с которым будут отобраны записи (рис. 2.19):

```

SELECT *
FROM transactions

```

```
WHERE transaction_date BETWEEN '2014-03-01' AND '2014-03-31';
```

```
SELECT *  
FROM transactions  
WHERE MONTH(transaction_date) = 3 AND YEAR(transaction_date)  
= 2014;
```

```
SELECT *  
FROM transactions  
WHERE transaction_date LIKE '2014-03-__';
```

The screenshot shows a SQL query editor with three queries listed. The first query is selected and highlighted in blue. Below the queries, a results table is displayed with the following data:

	customer_id	tran_amount	transaction_date
4	CS4103	40	2014-03-02
5	CS3927	77	2014-03-25
6	CS4292	98	2014-03-04
7	CS6056	104	2014-03-22
8	CS1397	77	2014-03-26
9	CS6066	102	2014-03-17
10	CS6009	39	2014-03-13
11	CS5750	48	2014-03-25
12	CS3180	65	2014-03-09
13	CS4341	87	2014-03-17
14	CS1453	45	2014-03-23
15	CS5152	53	2014-03-30
16	CS5043	85	2014-03-22

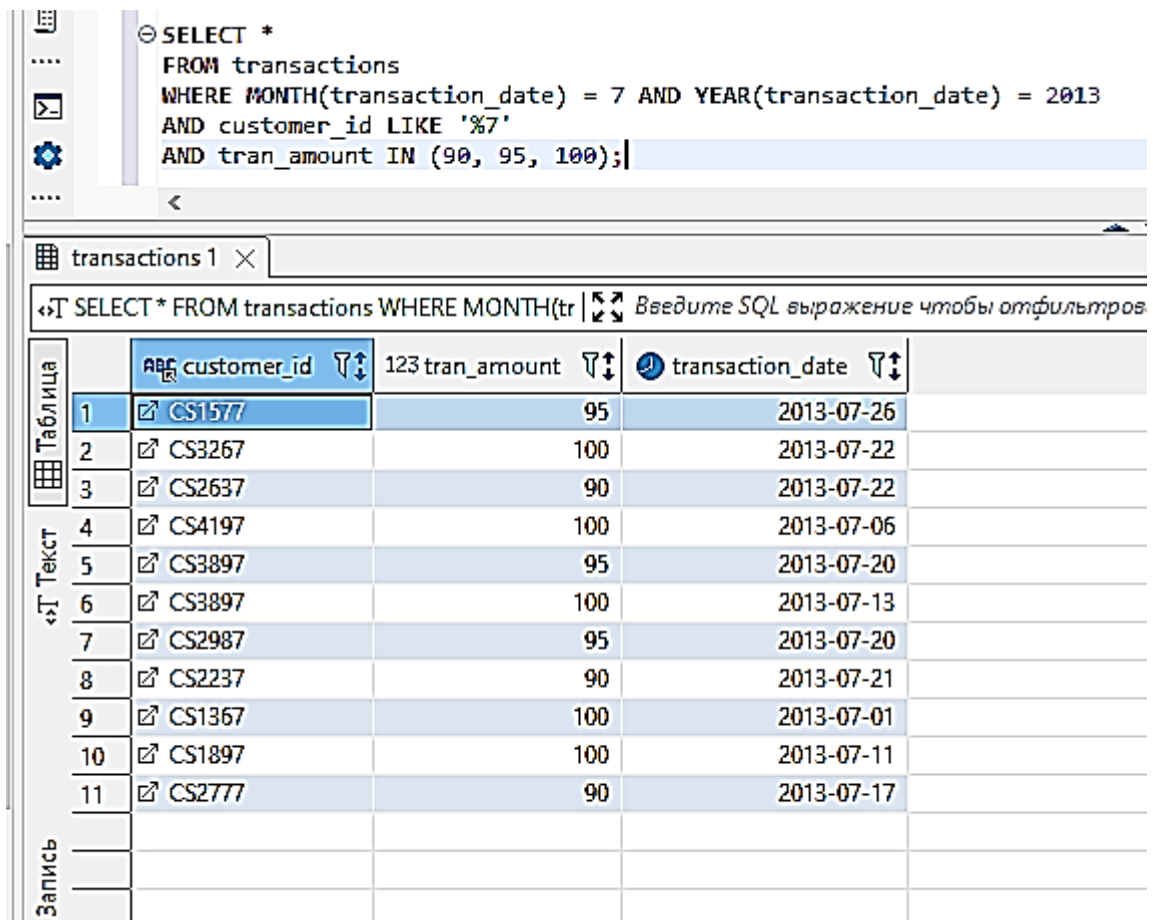
At the bottom of the screenshot, there is a status bar showing "200 строк получено - 239ms (69ms получ.), февр. 02, 20:26:12".

Рис. 2.19. Список транзакций с 1 марта 2014 г. по 31 марта 2014 г.

Ответим на вопрос: сколько заказчиков, чей идентификатор заканчивается на цифру 7 в июле 2013 г. совершили транзакции на сумму 90, 95 или 100? Для этого нужно составить условия отбора для всех трех полей таблицы и соединить их оператором AND (рис. 2.20):

```
SELECT *  
FROM transactions
```

```
WHERE MONTH(transaction_date) = 7 AND YEAR(transaction_date) = 2013
AND customer_id LIKE '%7'
AND tran_amount IN (90, 95, 100);
```



	customer_id	tran_amount	transaction_date
1	CS1577	95	2013-07-26
2	CS3267	100	2013-07-22
3	CS2637	90	2013-07-22
4	CS4197	100	2013-07-06
5	CS3897	95	2013-07-20
6	CS3897	100	2013-07-13
7	CS2987	95	2013-07-20
8	CS2237	90	2013-07-21
9	CS1367	100	2013-07-01
10	CS1897	100	2013-07-11
11	CS2777	90	2013-07-17

Рис. 2.20. Заказчики, чей идентификатор заканчивается на цифру 7, которые в июле 2013 г. совершили транзакции на сумму 90, 95 или 100

В DBeaver на просмотр выводится не более 200 записей запроса, последующие порции дозагружаются по кнопке , но можно изначально ограничить количество выводимых записей, добавив в конец запроса оператор LIMIT с указанием количества записей:

```
SELECT *
FROM transactions
WHERE transaction_date BETWEEN '2014-03-01' AND '2014-03-31'
LIMIT 10;
```

От простых запросов перейдем к более детальным.

2.4.2. Исключение дублей, сортировка, группировка и агрегации

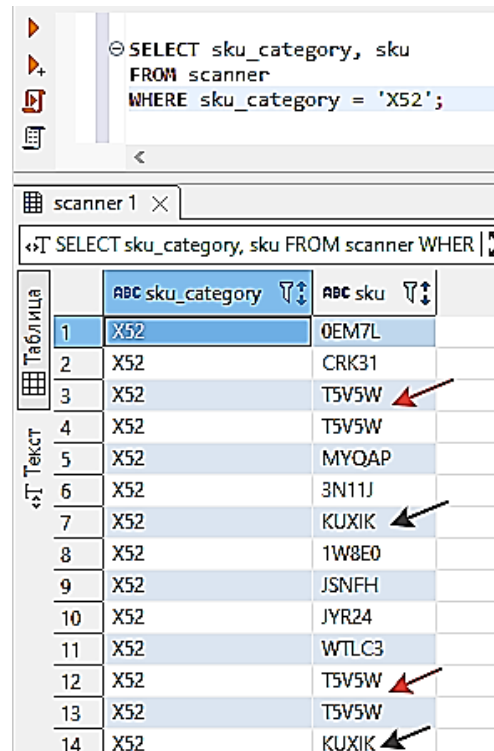
Полный синтаксис оператора SELECT:

```
SELECT [DISTINCT]
      expressions
FROM tables
[WHERE conditions]
[GROUP BY expressions]
[HAVING condition]
[ORDER BY expression [ ASC | DESC | USING operator ]]
[LIMIT [ number_rows | ALL]]
```

Если указано SELECT DISTINCT, то все повторяющиеся строки исключаются из результирующего набора, поведение по умолчанию SELECT ALL сохраняет все строки, ALL можно опустить.

Откроем редактор SQL для БД retail и сделаем выборку по идентификаторам товарных позиций (артикулам) для одной из товарных категорий (рис. 2.21):

```
SELECT sku_category, sku
FROM scanner
WHERE sku_category = 'X52';
```



The screenshot shows a SQL editor window with the following query:

```
SELECT sku_category, sku
FROM scanner
WHERE sku_category = 'X52';
```

Below the query, the results are displayed in a table view. The table has two columns: 'sku_category' and 'sku'. The results are as follows:

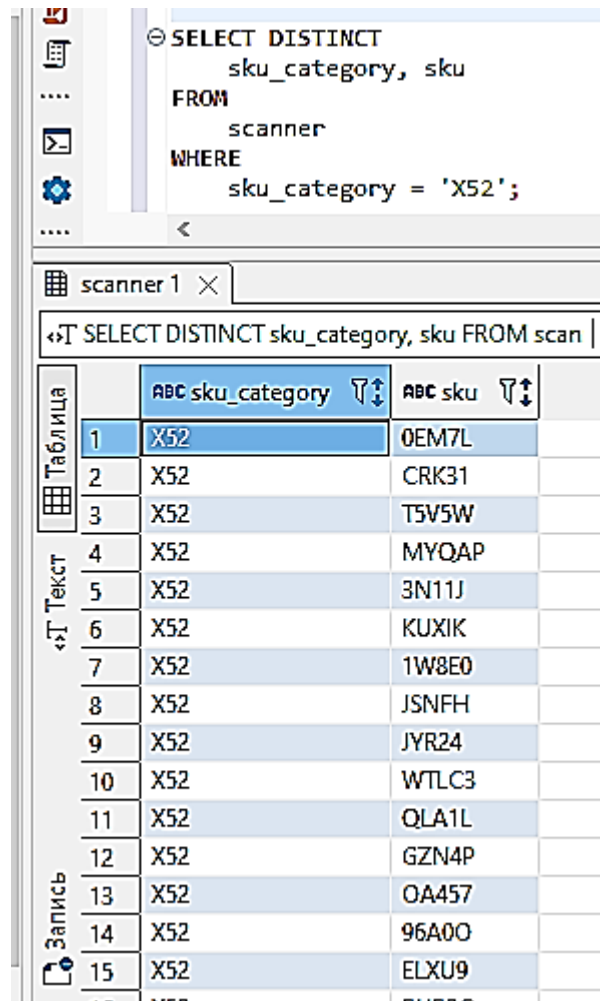
	sku_category	sku
1	X52	0EM7L
2	X52	CRK31
3	X52	T5V5W
4	X52	T5V5W
5	X52	MYQAP
6	X52	3N11J
7	X52	KUXIK
8	X52	1W8E0
9	X52	JSNFH
10	X52	JYR24
11	X52	WTLC3
12	X52	T5V5W
13	X52	T5V5W
14	X52	KUXIK

Red arrows point to the 'T5V5W' entries in rows 3, 12, and 13. A black arrow points to the 'KUXIK' entry in row 7.

Рис. 2.21. Результат выполнения запроса (артикул проданных товаров в категории 'X52')

Как видим, таблица содержит дубли, так как одни и те же товары встречаются в разных транзакциях. Добавим в SELECT предложение DISTINCT (рис. 2.22):

```
SELECT DISTINCT
    sku_category, sku
FROM
    scanner
WHERE
    sku_category = 'X52';
```



The screenshot shows a database query tool interface. At the top, the SQL query is entered: `SELECT DISTINCT sku_category, sku FROM scanner WHERE sku_category = 'X52';`. Below the query editor, the results are displayed in a table. The table has two columns: 'sku_category' and 'sku'. The 'sku_category' column contains the value 'X52' for all 15 rows. The 'sku' column contains 15 different values, representing unique products within that category.

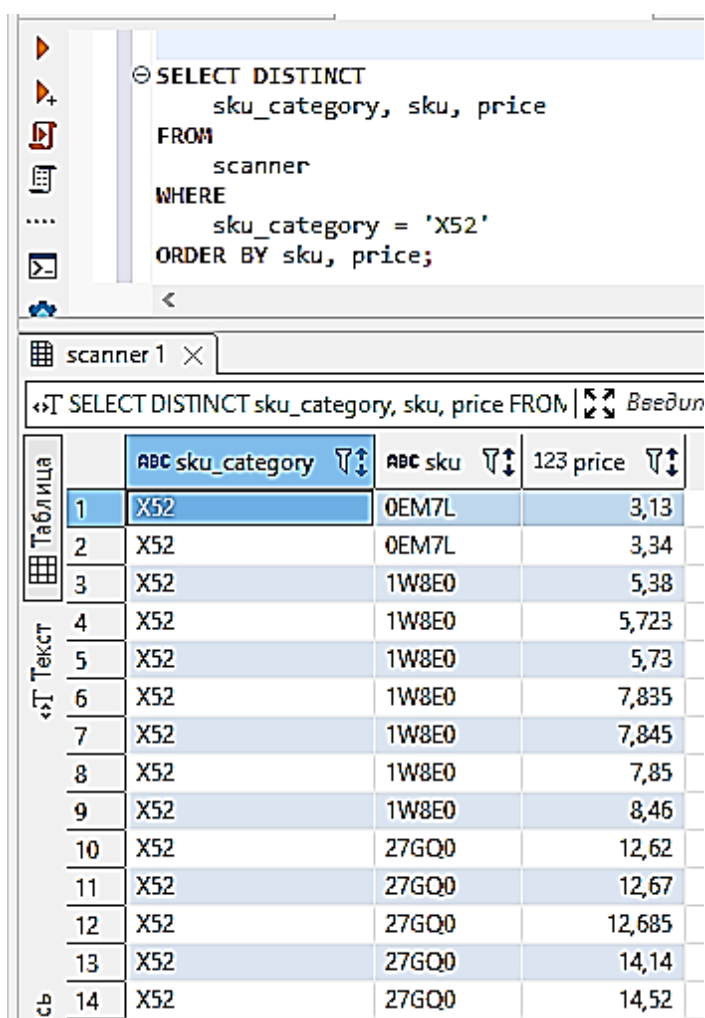
	sku_category	sku
1	X52	OEM7L
2	X52	CRK31
3	X52	T5V5W
4	X52	MYQAP
5	X52	3N11J
6	X52	KUXIK
7	X52	1W8E0
8	X52	JSNFH
9	X52	JYR24
10	X52	WTLC3
11	X52	QLA1L
12	X52	GZN4P
13	X52	OA457
14	X52	96A00
15	X52	ELXU9

Рис. 2.22. Результат выполнения запроса
(артикул проданных товаров в категории 'X52' без дубликатов)

Добавим в вывод информацию о цене товара. Увидим, что количество записей увеличилось. Это означает, что в отдельных транзакциях цена одного и того же товара была различной. Чтобы проанализировать диапазон разброса цен на один и тот же товар, необходимо упорядочить записи по полю артикул товара sku.

Для сортировки записей в результирующем наборе используется оператор ORDER BY с модификаторами ASC (сортировка в порядке возрастания, вариант по умолчанию – можно не указывать), DESC (сортировка по убыванию). Проведем сортировку результирующего набора вначале по артикулу товара, а потом по цене (рис. 2.23):

```
SELECT DISTINCT
    sku_category, sku, price
FROM
    scanner
WHERE
    sku_category = 'X52'
ORDER BY sku, price;
```



	ABC sku_category	ABC sku	123 price
1	X52	0EM7L	3,13
2	X52	0EM7L	3,34
3	X52	1W8E0	5,38
4	X52	1W8E0	5,723
5	X52	1W8E0	5,73
6	X52	1W8E0	7,835
7	X52	1W8E0	7,845
8	X52	1W8E0	7,85
9	X52	1W8E0	8,46
10	X52	27GQ0	12,62
11	X52	27GQ0	12,67
12	X52	27GQ0	12,685
13	X52	27GQ0	14,14
14	X52	27GQ0	14,52

Рис. 2.23. Результат выполнения запроса (артикул проданных товаров в категории 'X52' без дубликатов с учетом цены)

Значение цены единицы товара в четвертой строке выглядит сомнительно. Однако вспомним, что поле price – это поле с вычис-

лимыми значениями, которое было добавлено в таблицу как отношение суммы за товарную позицию в транзакции к количеству товара в этой позиции. И в продажах часто встречаются акции, когда четыре товара продаются по цене трех. Возможны и более сложные ценовые правила, например, учитывается персональная скидка конкретного покупателя и т.п.

Определим количество проданных товаров каждого артикула в этой категории, а также их общую стоимость. Оператор GROUP BY используется в операторе SELECT для сбора данных по нескольким записям и группировки результатов по одному или нескольким столбцам. В запрос необходимо включить агрегатную функцию (табл. 2.6), которая подсчитает количество проданных единиц товара каждого артикула, и общую сумму, вырученную за этот товар. Для столбцов, выводящих результат применения функций к значениям столбцов, необходимо указать псевдоним, иначе названием столбца станет имя самой функции.

Таблица 2.6

Агрегатные функции

Функция	Описание
avg()	находит среднее значение
sum()	находит сумму значений
min()	находит наименьшее значение
max()	находит наибольшее значение
count()	находит количество строк в запросе

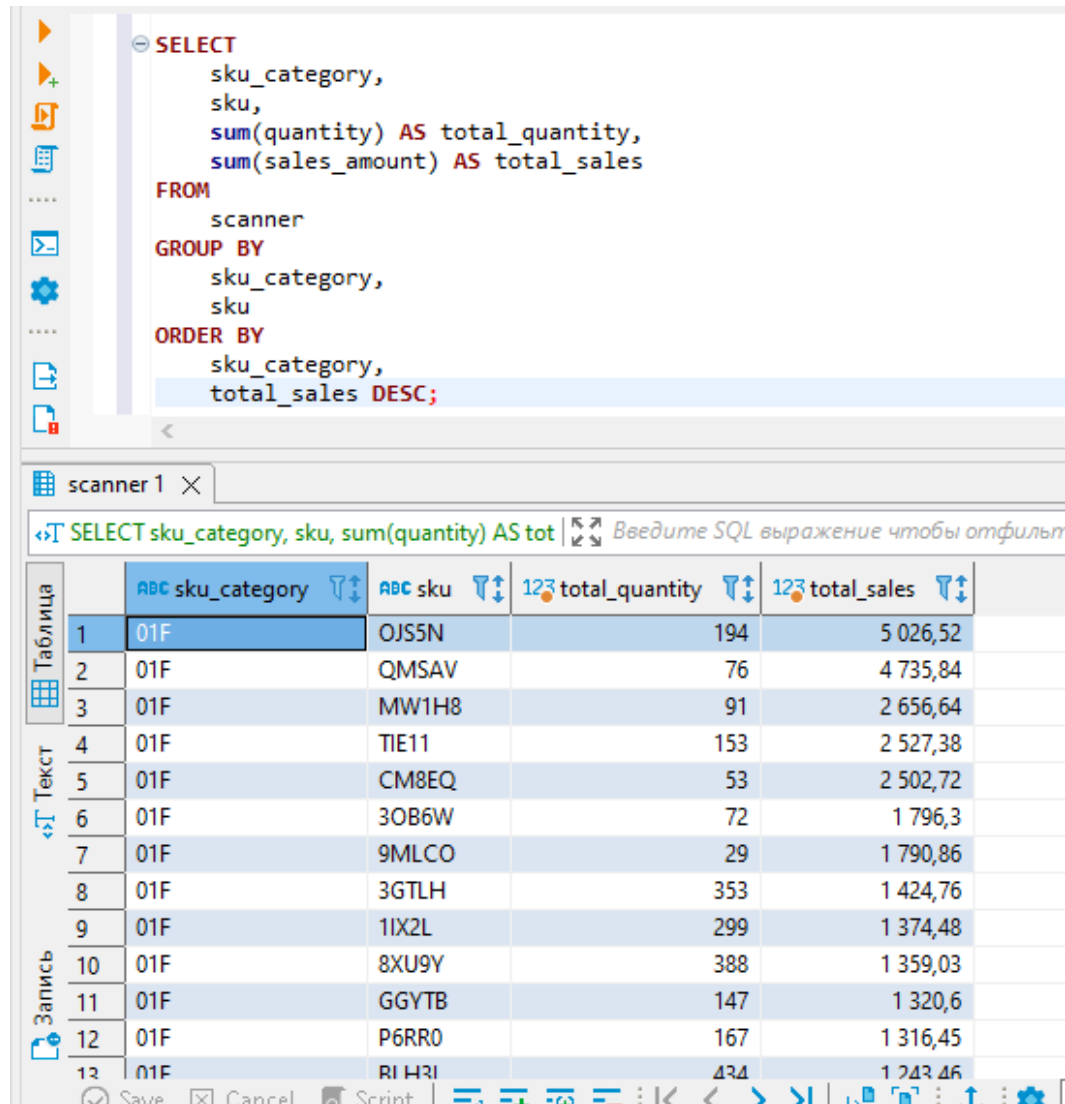
В запросе используется для группировки поле артикул товара SKU и сразу два поля quantity и sales_amount для вычисления значений агрегатных функций sum() (рис. 2.24). Для упрощения анализа данных также проводится сортировка по категории товара (чтобы товары одной категории выводились в одном фрагменте таблицы) и по убыванию общей суммы продаж товара конкретного артикула из категории:

```
SELECT
    sku_category,
    sku,
    sum(quantity) AS total_quantity,
    sum(sales_amount) AS total_sales
FROM
    scanner
GROUP BY
```

```

sku_category,
sku
ORDER BY
sku_category,
total_sales DESC;

```



SELECT

```

    sku_category,
    sku,
    sum(quantity) AS total_quantity,
    sum(sales_amount) AS total_sales
FROM
    scanner
GROUP BY
    sku_category,
    sku
ORDER BY
    sku_category,
    total_sales DESC;

```

scanner 1 X

SELECT sku_category, sku, sum(quantity) AS tot | Введите SQL выражение чтобы отфильтр

	ABC sku_category	ABC sku	123 total_quantity	123 total_sales
1	01F	OJS5N	194	5 026,52
2	01F	QMSAV	76	4 735,84
3	01F	MW1H8	91	2 656,64
4	01F	TIE11	153	2 527,38
5	01F	CM8EQ	53	2 502,72
6	01F	3OB6W	72	1 796,3
7	01F	9MLCO	29	1 790,86
8	01F	3GTLH	353	1 424,76
9	01F	1IX2L	299	1 374,48
10	01F	8XU9Y	388	1 359,03
11	01F	GGYTB	147	1 320,6
12	01F	P6RR0	167	1 316,45
13	01F	RI H3I	434	1 243,46

Save Cancel Script

Рис. 2.24. Результат выполнения запроса (общее количество проданных единиц и сумма продаж каждого артикула товара)

Введем дополнительное условие отбора строк: оставим только одну категорию sku_category 'X52' (рис. 2.25). Так как теперь нет вариации записей по категории товара, то это поле исключим из запроса на вывод и из сортировки, оставив его только в условии отбора записей:

```

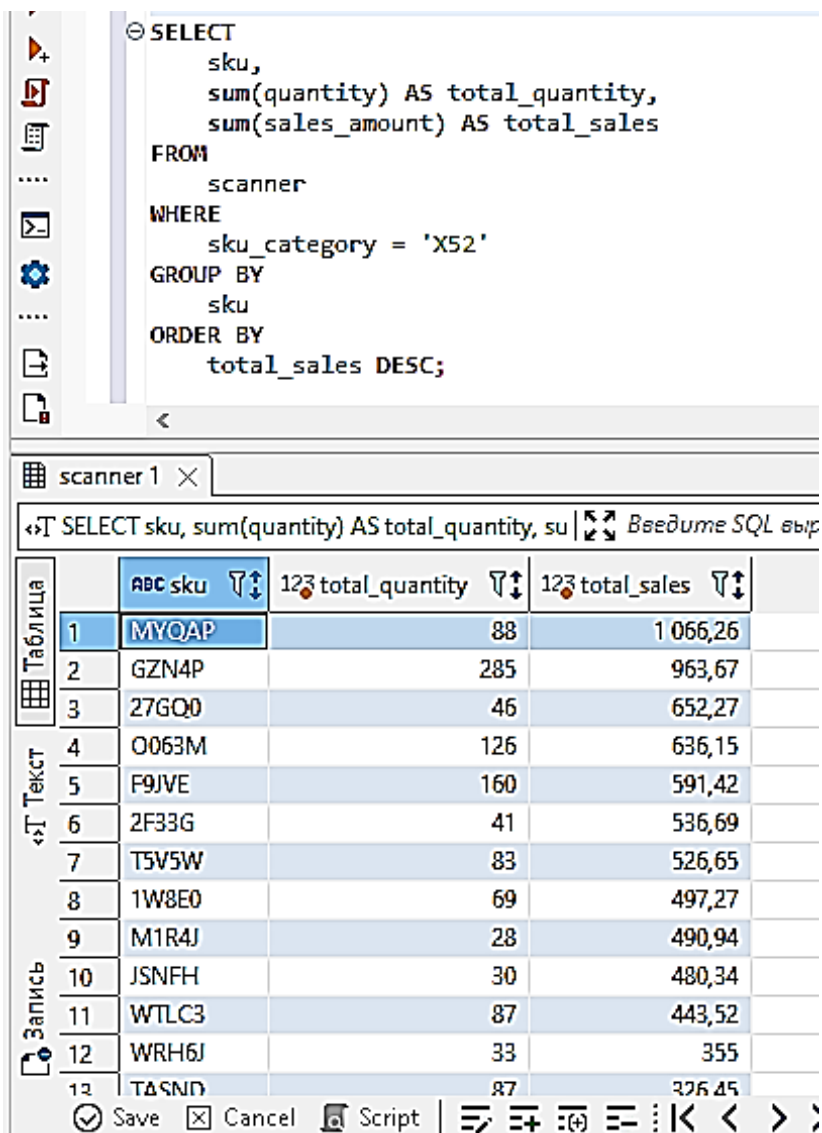
SELECT
    sku,

```

```

sum(quantity) AS total_quantity,
sum(sales_amount) AS total_sales
FROM
    scanner
WHERE
    sku_category = 'X52'
GROUP BY
    sku
ORDER BY
    total_sales DESC;

```



The screenshot shows a database query tool interface. The top panel displays the SQL query: `SELECT sku, sum(quantity) AS total_quantity, sum(sales_amount) AS total_sales FROM scanner WHERE sku_category = 'X52' GROUP BY sku ORDER BY total_sales DESC;`. Below the query editor, a tab labeled 'scanner 1' is active. The results are displayed in a table view with columns: 'sku', 'total_quantity', and 'total_sales'. The table contains 13 rows of data, sorted by total sales in descending order. The first row is for 'MYQAP' with a total quantity of 88 and total sales of 1,066.26. The last row is for 'TASND' with a total quantity of 87 and total sales of 326.45. The interface includes a toolbar on the left with icons for query execution, a bottom toolbar with 'Save', 'Cancel', and 'Script' buttons, and a status bar at the bottom.

	sku	total_quantity	total_sales
1	MYQAP	88	1 066,26
2	GZN4P	285	963,67
3	27GQ0	46	652,27
4	O063M	126	636,15
5	F9JVE	160	591,42
6	2F33G	41	536,69
7	T5V5W	83	526,65
8	1W8E0	69	497,27
9	M1R4J	28	490,94
10	JSNFH	30	480,34
11	WTLC3	87	443,52
12	WRH6J	33	355
13	TASND	87	326,45

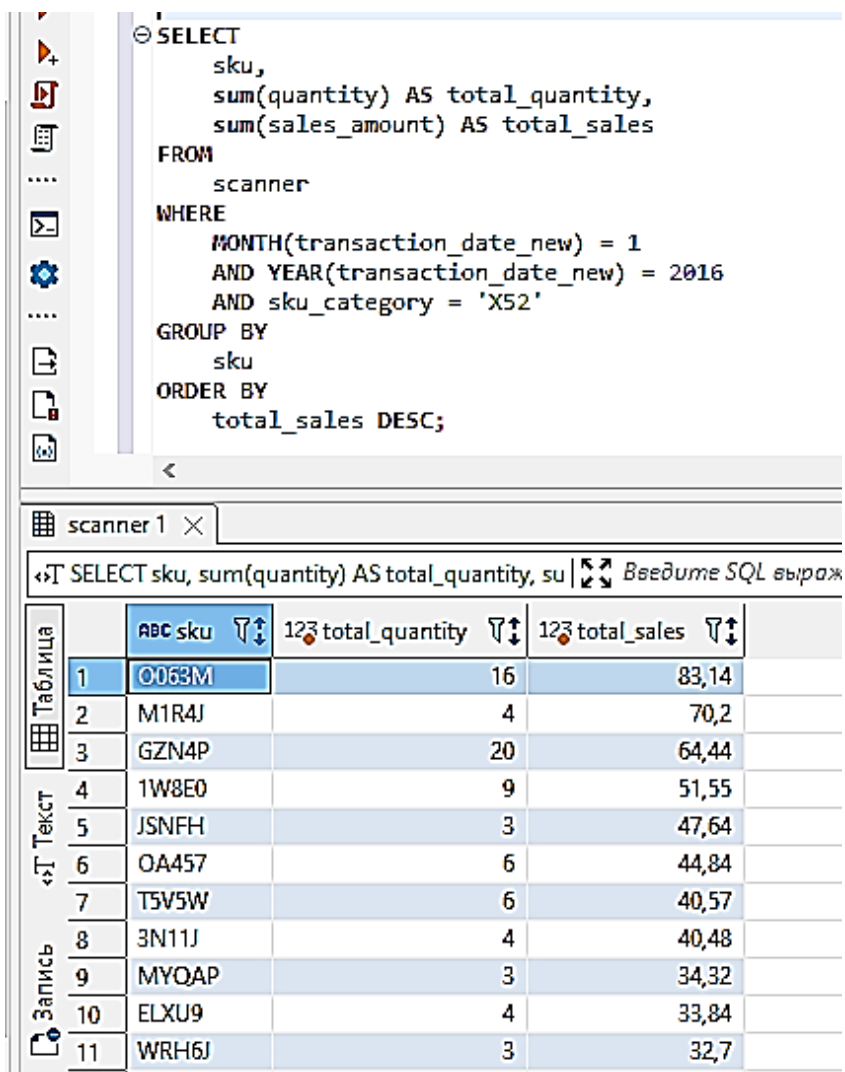
Рис. 2.25. Результат выполнения запроса для категории товаров 'X52'

Добавим дополнительное условие по временному интервалу продаж и узнаем количество проданных товаров и суммы продаж различных артикулов товаров в этой категории за январь 2016 г. (рис. 2.26):

```

SELECT
    sku,
    sum(quantity) AS total_quantity,
    sum(sales_amount) AS total_sales
FROM
    scanner
WHERE
    MONTH(transaction_date_new) = 1
    AND YEAR(transaction_date_new) = 2016
    AND sku_category = 'X52'
GROUP BY
    sku
ORDER BY
    total_sales DESC;

```



SQL Query:

```

SELECT
    sku,
    sum(quantity) AS total_quantity,
    sum(sales_amount) AS total_sales
FROM
    scanner
WHERE
    MONTH(transaction_date_new) = 1
    AND YEAR(transaction_date_new) = 2016
    AND sku_category = 'X52'
GROUP BY
    sku
ORDER BY
    total_sales DESC;

```

Results Table:

Таблица	sku	total_quantity	total_sales
1	O063M	16	83,14
2	M1R4J	4	70,2
3	GZN4P	20	64,44
4	1W8E0	9	51,55
5	JSNFH	3	47,64
6	OA457	6	44,84
7	T5V5W	6	40,57
8	3N11J	4	40,48
9	MYQAP	3	34,32
10	ELXU9	4	33,84
11	WRH6J	3	32,7

Рис. 2.26. Результат выполнения запроса для категории товаров 'X52' в январе 2016 г.

Оставим в исходном запросе (до уточнения временного интервала и категории товаров) в рассмотрении только те артикулы, для которых общая сумма продаж за весь период превосходит 2500. Для этого будем использовать оператор HAVING, который исключает из результата строки групп, не удовлетворяющих условию.

HAVING выполняет ту же роль, что и WHERE, но WHERE фильтрует отдельные строки до применения GROUP BY, а HAVING фильтрует строки групп, созданных предложением GROUP BY.

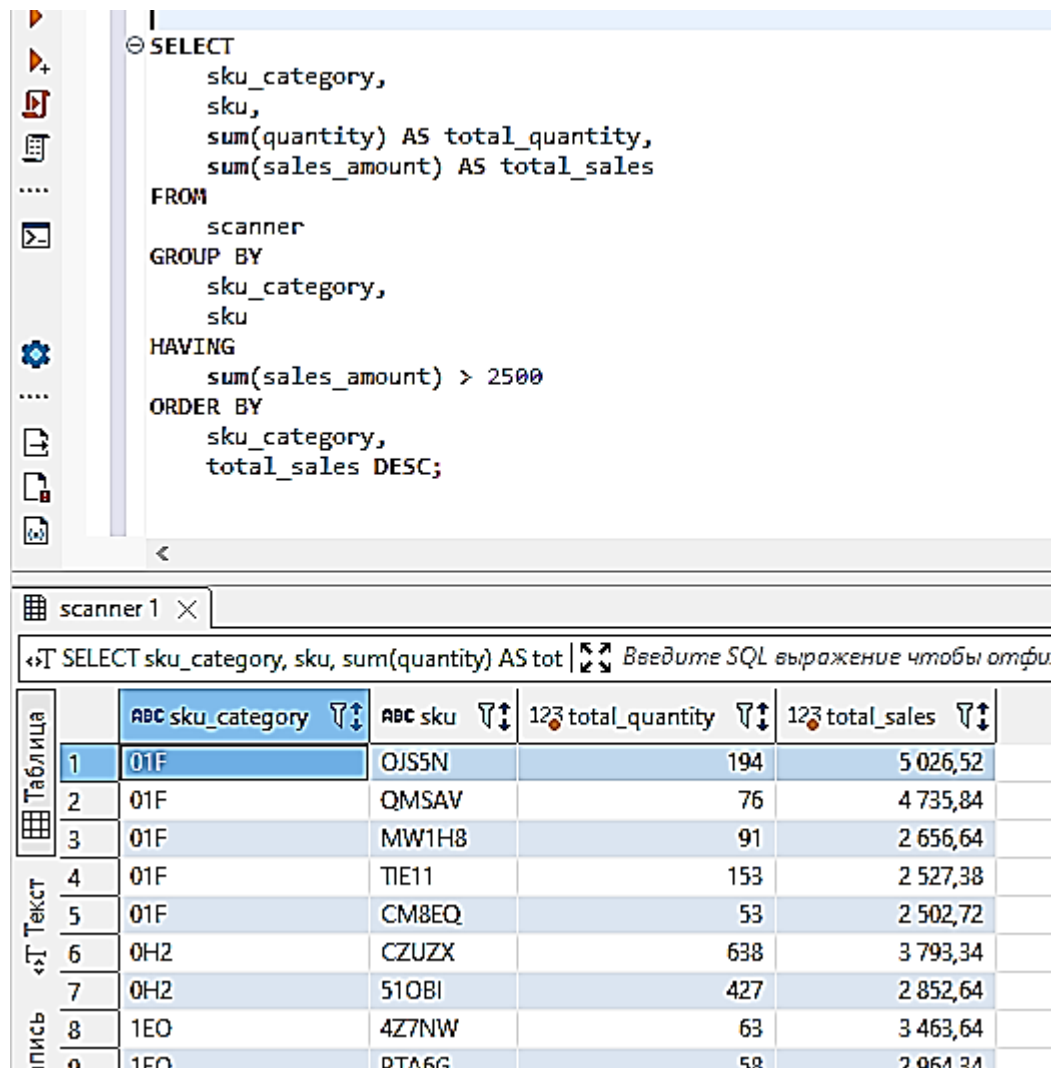
В запрос нужно добавить условие отбора по значению больше 2500 для столбца с агрегатной функцией sum(sales_amount) с псевдонимом total_sales. Однако обращаться по псевдониму total_sales к результату агрегатной функции нельзя, это связано с порядком выполнения операторов:

```
FROM
JOIN
WHERE
GROUP BY
HAVING
SELECT
DISTINCT
ORDER BY
```

К моменту выполнения оператора SELECT уже известны таблицы и представления, из которых требуется забрать столбцы, а также их алиасы (оператор FROM), а вот к моменту выполнения оператора HAVING о псевдониме для столбца с результатом агрегатной функции (оператор SELECT) еще неизвестно. То же и с ORDER BY, к моменту выполнения оператора столбцу с результатом агрегатной функции уже присвоен псевдоним (оператор SELECT), поэтому обращаться к нему можно только по алиасу (рис. 2.27).

```
SELECT
    sku_category,
    sku,
    sum(quantity) AS total_quantity,
    sum(sales_amount) AS total_sales
FROM
    scanner
GROUP BY
    sku_category,
    sku
HAVING
    sum(sales_amount) > 2500
ORDER BY
```

```
sku_category,
total_sales DESC;
```



The screenshot shows a database query tool interface. The top pane displays an SQL query: `SELECT sku_category, sku, sum(quantity) AS total_quantity, sum(sales_amount) AS total_sales FROM scanner GROUP BY sku_category, sku HAVING sum(sales_amount) > 2500 ORDER BY sku_category, total_sales DESC;`. The bottom pane shows the results in a table view for 'scanner 1'. The table has 5 columns: 'sku_category', 'sku', 'total_quantity', and 'total_sales'. The results are ordered by 'sku_category' and 'total_sales' in descending order.

	sku_category	sku	total_quantity	total_sales
1	01F	OJS5N	194	5 026,52
2	01F	QMSAV	76	4 735,84
3	01F	MW1H8	91	2 656,64
4	01F	PIE11	153	2 527,38
5	01F	CM8EQ	53	2 502,72
6	0H2	CZUZX	638	3 793,34
7	0H2	510BI	427	2 852,64
8	1EO	4Z7NW	63	3 463,64
9	1EO	PTA6G	58	2 964,34

Рис. 2.27. Результат выполнения запроса для отбора артикулов товаров продажи которых были за весь период на сумму свыше 2500

2.4.3. Соединения

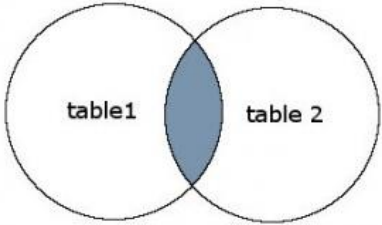
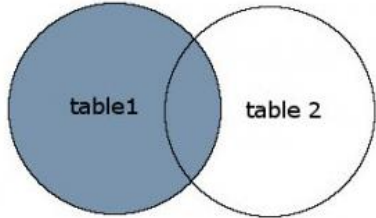
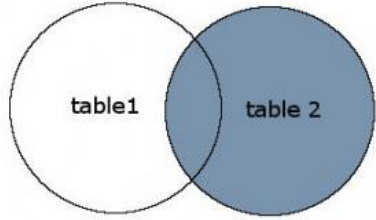
Если для получения результата требуется информация из нескольких таблиц, то в операторе FROM используют JOIN (соединение) таблиц (и любых иных объектов базы данных). Типы доступных в MySQL соединений представлены в табл. 2.7. Предложение JOIN дополняется [ON conditions], определяющим условие соединения таблиц.

Задаваемое условие соединения (ON conditions) представляет собой выражение, выдающее значение типа boolean (как в предложе-

нии WHERE), которое определяет, какие строки считаются соответствующими при соединении.

Таблица 2.7

Типы соединений

Тип соединения	Описание	Визуализация
INNER JOIN JOIN	Простое соединение, которое возвращает все строки декартова произведения, ограниченные условием соединения	
LEFT OUTER JOIN LEFT JOIN	Левостороннее соединение, которое возвращает все строки ограниченного декартова произведения: все объединенные строки, удовлетворяющие условию соединения, и все строки в таблице слева, для которых не находится строк в таблице справа, удовлетворяющих условию (дополняются до полной ширины объединённой таблицы значениями NULL в столбцах таблицы справа)	
RIGHT OUTER JOIN RIGHT JOIN	Правостороннее соединение (практически не используется, так как можно получить аналогичный результат, если поменять порядок следования таблиц и использовать левостороннее соединение)	

Синтаксис операторов соединения JOIN:

```
SELECT columns
FROM table1
[INNER] JOIN table2
ON table1.column = table2.column;
```

```
SELECT columns  
FROM table1  
LEFT [OUTER] JOIN table2  
ON table1.column = table2.column;
```

```
SELECT columns  
FROM table1  
RIGHT [OUTER] JOIN table2  
ON table1.column = table2.column;
```

Откроем скрипт для базы данных retail_new. Найдем общую сумму покупок клиентов, которые откликнулись на рекламное взаимодействие. Транзакции оплаты покупок клиентов находятся в таблице transactions, а ответ клиентов на рекламное взаимодействие – в таблице response, т.е. нужны сведения из двух таблиц. Соединение пройдет по ключевому полю customer_id, которое включено в каждую из двух соединяемых таблиц.

В этом случае становится очевидной польза от псевдонимов для таблиц, так как обе таблицы содержат столбец с одним и тем же именем, и нужна явная отсылка при указании имени столбца, к какой из таблиц он относится. А указывать перед именем столбца одну букву (псевдоним) или полное имя таблицы – это разные затраты времени на кодирование.

Нам нужно также вычислить агрегирующую функцию sum() для платежей сгруппированных по клиентам customer_id и отсортировать по вычисляемому полю по убыванию оплаченной суммы за весь период (рис. 2.28).

```
SELECT  
    t.customer_id,  
    sum(t.tran_amount) AS total_amount,  
    r.response  
FROM  
    transactions AS t  
JOIN  
    response AS r  
ON  
    t.customer_id = r.customer_id  
WHERE  
    r.response = 1  
GROUP BY  
    t.customer_id  
ORDER BY  
    total_amount DESC;
```

Commit Rollback Auto MySQL retail_new

*<MySQL> Script-14 x *<MySQL> Script-12 *<MySQL> Script-13

```

SELECT
  t.customer_id,
  sum(t.tran_amount) AS total_amount,
  r.response
FROM
  transactions AS t
JOIN
  response AS r
ON
  t.customer_id = r.customer_id
WHERE
  r.response = 1
GROUP BY
  t.customer_id
ORDER BY
  total_amount DESC;

```

transactions(+) 1 x

SELECT t.customer_id, sum(t.tran_amount) AS t Введите SQL выражение чтобы отф

	customer_id	total_amount	response
1	CS4320	2 647	1
2	CS2951	2 382	1
3	CS4540	2 361	1
4	CS4746	2 337	1
5	CS1580	2 329	1
6	CS3362	2 303	1

Рис. 2.28. Результат выполнения запроса на вывод клиентов, которые откликнулись на рекламное взаимодействие, и потраченные ими суммы

Отметим, что поле response лишнее в выдаче, так как для всех записей в запросе оно принимает одно и тоже значение и можно было обойтись без него.

Оператор простого соединения можно не использовать, перечислив обе таблицы в операторе FROM и добавив условие соединения таблиц в оператор WHERE (рис. 2.29). Как видим, результат аналогичен, также опущено необязательное ключевое слово AS при назначении псевдонимов таблицам (для столбцов по негласному соглашению его все же используют) и не запрашивается столбец response:

```

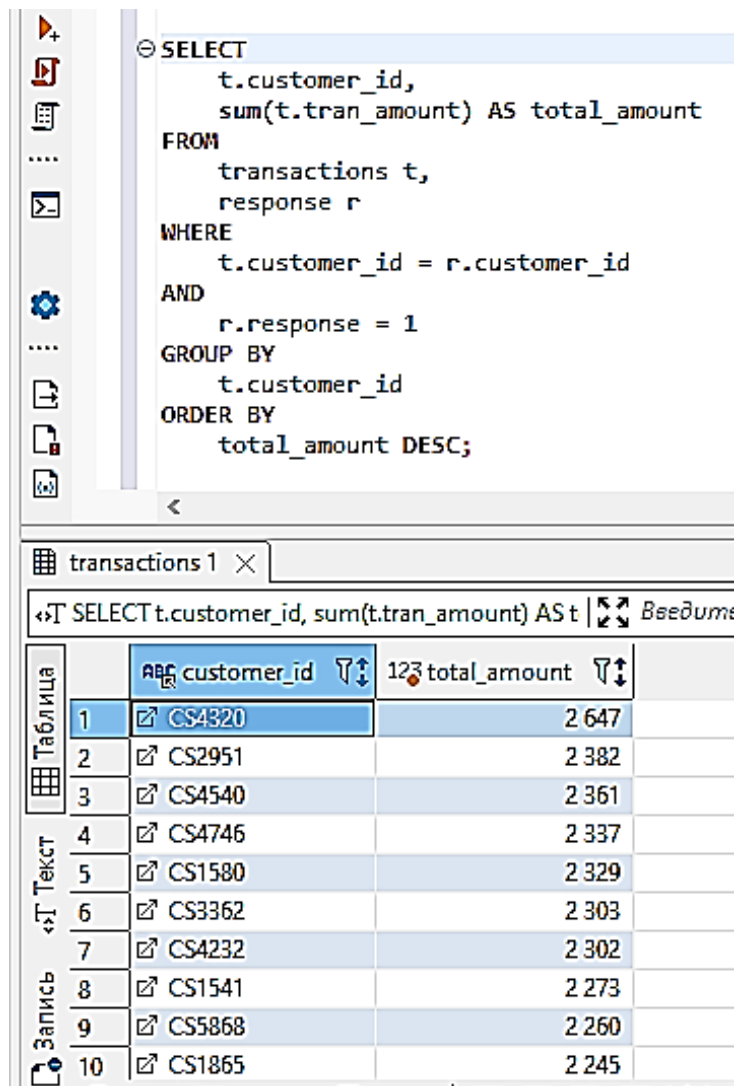
SELECT
  t.customer_id,
  sum(t.tran_amount) AS total_amount
FROM
  transactions t,

```

```

        response r
WHERE
        t.customer_id = r.customer_id
AND
        r.response = 1
GROUP BY
        t.customer_id
ORDER BY
        total_amount DESC;

```



The screenshot shows a database query editor with a SQL query in the top pane and its results in the bottom pane. The query is a SELECT statement that joins the 'transactions' table (t) and the 'response' table (r), filters for responses that are 1, groups by customer_id, and orders by total_amount in descending order. The results pane shows a table with 10 rows, each representing a customer and their total transaction amount.

	customer_id	total_amount
1	CS4320	2 647
2	CS2951	2 382
3	CS4540	2 361
4	CS4746	2 337
5	CS1580	2 329
6	CS3362	2 303
7	CS4232	2 302
8	CS1541	2 273
9	CS5868	2 260
10	CS1865	2 245

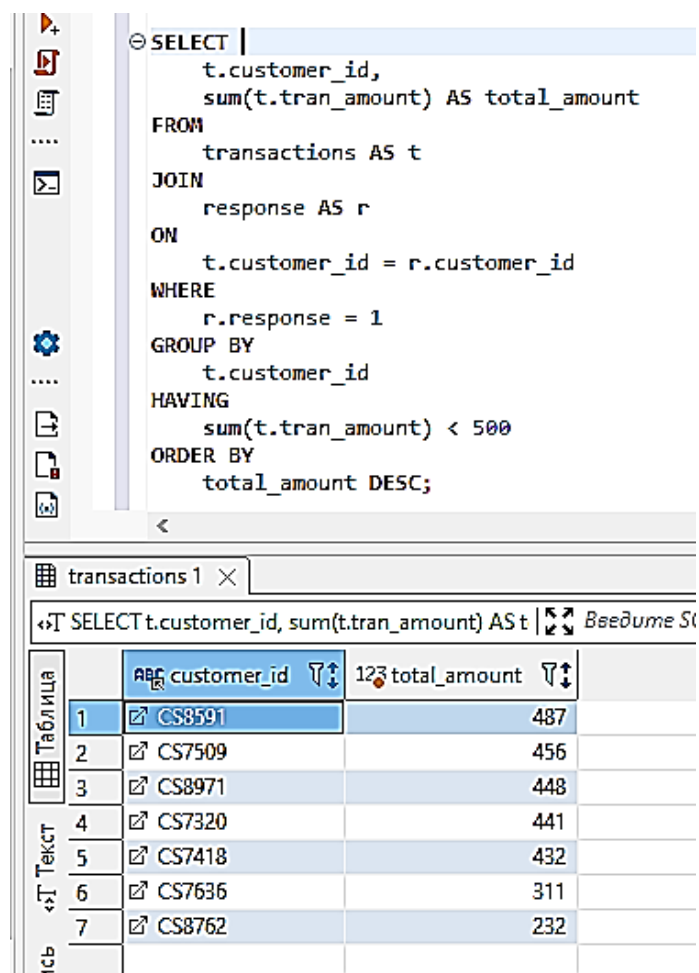
Рис. 2.29. Результат выполнения альтернативной версии запроса (клиенты, которые откликнулись на рекламное взаимодействие, и потраченные ими суммы)

Оставим только те записи в предыдущем запросе, в которых клиенты имеют общую сумму оплаты товаров меньше 500, т.е. используем оператор HAVING для отбора записей по условию (рис. 2.30):

```

SELECT
    t.customer_id,
    sum(t.tran_amount) AS total_amount
FROM
    transactions AS t
JOIN
    response AS r
ON
    t.customer_id = r.customer_id
WHERE
    r.response = 1
GROUP BY
    t.customer_id
HAVING
    sum(t.tran_amount) < 500
ORDER BY
    total_amount DESC;

```



The screenshot shows a database management tool interface. The top pane displays the SQL query. The bottom pane shows the results of the query in a table format. The table has two columns: 'customer_id' and 'total_amount'. The results are ordered by 'total_amount' in descending order.

	customer_id	total_amount
1	CS8591	487
2	CS7509	456
3	CS8971	448
4	CS7320	441
5	CS7418	432
6	CS7636	311
7	CS8762	232

Рис. 2.30. Результат выполнения запроса на вывод откликнувшихся на рекламное взаимодействие клиентов с потраченными суммами меньше 500

Проверим, обо всех ли клиентах, которые совершали транзакции, известна информация об их отклике на рекламную компанию. Запросим информацию обо всех клиентах, про которых неизвестно об их отклике на рекламную компанию (рис. 2.31):

```
SELECT DISTINCT
    t.customer_id
FROM
    transactions AS t
LEFT JOIN
    response AS r
ON
    t.customer_id = r.customer_id
WHERE
    r.response IS NULL;
```

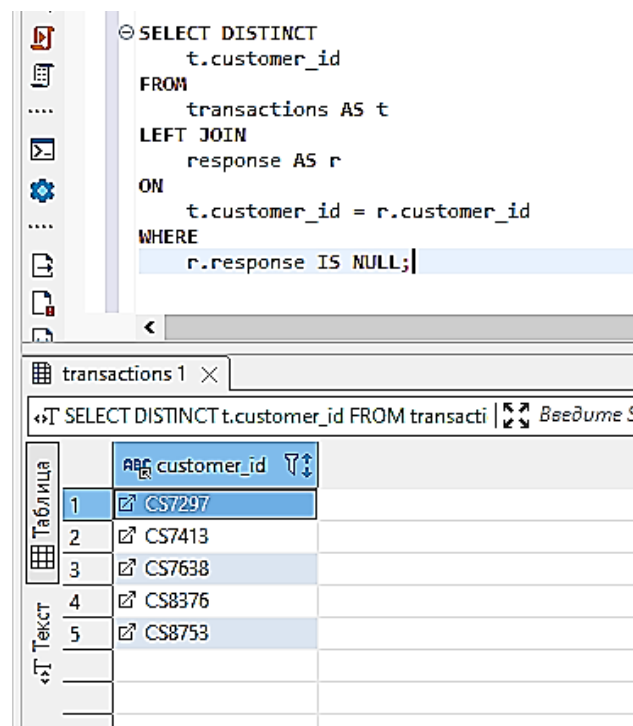


Рис. 2.31. Клиенты, чьи идентификаторы есть в первой таблице transactions, но отсутствуют во второй response

Именно эти клиенты и мешали проверке условия соединения по внешнему ключу.

Однако избавляться от этих записей в таблице transactions было бы некорректно, так как они содержат важные данные о покупках этих клиентов. Удалив из запроса ключевое слово DISTINCT мы получим общее количество записей с такими клиентами в таблице transactions – 31.

3. Лабораторные работы

3.1. Лабораторная работа «Настройка программного обеспечения»

Цель работы: научиться устанавливать программное обеспечение и настраивать его работу, научиться импортировать файлы баз данных с расширением .CSV.

Формируемые знания, умения и навыки: знать, как происходит установка СУБД MySQL, уметь запускать консольных клиентов MySQL, графического клиента MySQL Workbench и менеджер БД DBeaver.

Необходимо:

- 1) используя материал настоящего пособия и вспомогательные ресурсы, произвести установку на домашний компьютер требуемого программного обеспечения;
- 2) выполнить импорт демонстрационных БД (Samples and Examples).

Контрольные вопросы и задания

1. С чего начинается работа в MySQL Command Line Client, MySQL Shell?
2. Как создать новую БД?
3. Как импортировать в таблицу БД файл с данными с расширением .CSV?
4. Как проектируется БД в MySQL Workbench?
5. Как настроить подключение MySQL Server в менеджере БД DBeaver?

3.2. Лабораторная работа «Операции DDL»

Цель работы: научиться работать с объектами БД, научиться создавать новые объекты, модифицировать и удалять существующие.

Формируемые знания, умения и навыки: знать основные типы данных, синтаксис операций языка DDL, уметь составлять запросы с использованием конструкций языка DDL.

Необходимо:

- 1) создать базу данных, создать таблицу в этой БД, добавить столбец в таблицу, удалить столбец из таблицы;
- 2) удалить таблицу, удалить БД.

Контрольные вопросы и задания

1. Как типы данных используются в MySQL?
2. Как создать БД?
3. Как создать таблицу?
4. Как переименовать таблицу?
5. Как переименовать столбец?
6. Как удалить таблицу?
7. Как удалить БД?

3.3. Лабораторная работа «Операции DML»

Цель работы: научиться переходить от бизнес-задач, т.е. вопросов на естественном языке, включающих запрос необходимых бизнес-данных, к запросам на языке SQL.

Формируемые знания, умения и навыки: знать синтаксис операций языка DML, уметь составлять запросы с использованием конструкций языка DML.

Необходимо:

- 1) для таблиц и представлений демонстрационной БД sakila самостоятельно сформулировать запросы с использованием описанных в пособии конструкций и проанализировать результат. Кроме формулировки запроса на естественном языке необходимо привести запрос на языке SQL и проанализировать результат его исполнения;
- 2) уточнить ранее сформулированные запросы по каким-либо условиям;
- 3) привести примеры запросов с использованием агрегатных функций.

Создать новую БД, в которой создать две таблицы, добавить в них данные согласно рис. 3.1. Выполнить различные варианты соединений и пояснить полученный результат (особенно важно уделить внимание проверке условий, если в сравниваемых столбцах имеются значения NULL).

Staff table

LastName	DepartmentID
Иванов	31
Сидорова	33
Смирнов	33
Матвеев	34
Максимова	34
Яковлев	NULL

Department table

DepartmentID	DepartmentName
31	Продажи
33	Делопроизводство
34	Закупки
35	Маркетинг
31	Бухгалтерия

Рис. 3.1. Таблицы – образец заполнения данными

Контрольные вопросы и задания

1. Синтаксис операции SELECT.
2. Какие операторы сравнения можно использовать в условиях?
3. Объясните разницу между использованием операторов WHERE и HAVING.
4. Приведите пример использования агрегатной функции в запросе.
5. Как исключить дубли из запроса?
6. Какие типы соединений вы знаете? Чем они отличаются?
7. Какие особенности имеет сравнение столбцов, в которых есть значения NULL?

3.4. Лабораторная работа «Бизнес-задачи»

Цель работы: научиться формулировать бизнес-запросы и обосновывать их ценность для компании.

Формируемые знания, умения и навыки: уметь работать с данными с точки зрения data-driven подхода (управление на основе данных), уметь формулировать бизнес-проблемы и переводит их на язык запросов, владеть навыком применения результатов запросов для принятия бизнес-решений (обосновывать ценность результатов запросов для принятия бизнес-решений).

Необходимо:

- 1) на примере демонстрационной БД или данных в формате .CSV самостоятельно найденных на ресурсе kaggle (<https://www.kaggle.com/>).

kaggle.com/) сформулировать бизнес-проблемы, для решения которых необходим анализ данных. Перевести бизнес-проблемы на язык запросов к БД. Описать, как будут использоваться полученные результаты при решении проблем и ценность полученных результатов в контексте принятия решений в бизнесе;

2) результаты работы оформить в виде эссе или статьи.

Список литературы

1. MySQL Installer 8.0.35 : офиц. сайт MySQL. URL: <https://dev.mysql.com/downloads/installer/> (дата обращения: 11.01.2024).
2. MySQL 8.0 Reference Manual : офиц. документация MySQL. URL: <https://dev.mysql.com/doc/refman/8.0/en/introduction.html> (дата обращения: 11.01.2024).
3. Введение в MySQL : сайт о программировании. URL: <https://metanit.com/sql/mysql/1.1.php?ysclid=lrqo4wq4p3741608750> (дата обращения: 11.01.2024).
4. MySQL учебник : документация от Oracle по MySQL. URL: <https://oracleplsql.ru/mysql-manual.html> (дата обращения: 11.01.2024).
5. MySQL Workbench Manual : офиц. документация MySQL [Workbench]. URL: <https://dev.mysql.com/doc/workbench/en/wb-intro.html> (дата обращения: 11.01.2024).
6. MySQL Shell 8.0 : офиц. документация MySQL Shell. URL: <https://dev.mysql.com/doc/mysql-shell/8.0/en/mysql-shell-features.html> (дата обращения: 11.01.2024).

Учебное издание

Рындина Светлана Валентиновна

**Цифровые технологии управления получением, хранением,
передачей и обработкой больших данных: MySQL**

Редактор *Е. А. Дергунова*
Технический редактор *М. Б. Жучкова*
Компьютерная верстка *М. Б. Жучковой*

Подписано в печать 07.05.2024.
Формат 60×84¹/₁₆. Усл. печ. л. 6,85.
Тираж 6. Заказ № 164.

Издательство ПГУ.
440026, г. Пенза, ул. Красная, 40.
Тел.: (8412) 66-60-49, 66-67-77; e-mail: iic@pnzgu.ru

