Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

**ПРОГРАММНОЕ СРЕДСТВО ДЛЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ МУЗЫКАЛЬНЫМ МАГАЗИНОМ**

БГУИР КП I–40 01 01 05 ПЗ

Выполнил

студент гр. 051006 Шуляк А. В,

Проверил: Медведев С. А.

Минск 2023

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ПОИТ

(подпись)

Лапицкая Н.В. 2023г.

ЗАДАНИЕ

по курсовому проектированию

Студенту Шуляку Андрею Валерьевичу

1. Тема работы *Программное средство для автоматизации управления музыкальным магазином*

2. Срок сдачи законченной работы *15.12.2023г.*

3. Исходные данные к работе *Система управления базами данных MySQL, GUI for MySQL “MySQL Workbench”*

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

*Введение*

*1 Анализ литературных источников*

*2 Постановка задачи*

*3 Проектирование базы данных*

*4 Описание технических аспектов реализации базы данных*

*5 Тестирование базы данных*

*Заключение*

*Список использованных источников*

*Приложения*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*Схема базы данных в формате А1*

6. Консультант по курсовой работе *Медведев С.А*

7.Дата выдачи задания  *03.09.2023г.*

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объема работы):

*Раздел 1. Введение к 15.09.2023 – 10 % готовности работы;––––––––        –*

*Раздел 2, 3 к 15.10.2023 – 30 % готовности работы;–––––––––––––––       –––*

*Раздел 4, 5 к 15.11.2023 – 60 % готовности работы;*

*Раздел 5, 6, Заключение к 15.12.2023 – 90 % готовности работы; Оформление пояснительной записки и графического материала к 15.12.2023г. – 100 % готовности работы.*

*Защита курсового проекта с 04.12.2023г. по 21.12.2023г.*

РУКОВОДИТЕЛЬ  *Медведев С.А.*

*(подпись)*

Задание принял к исполнению *Шуляк А. В, 03.09.2023*

*(дата и подпись студента)*

Содержание

[Введение 5](#_Toc1)

[1 Анализ литературных источников 6](#_Toc2)

[1.1 1С: Предприятие 6](#_Toc3)

[1.2 SAP Business One 7](#_Toc4)

[1.3 Oracle Net Suite 7](#_Toc5)

[1.4 Microsoft Dynamics 8](#_Toc6)

[1.5 Zoho Inventory 8](#_Toc7)

[2 Постановка задачи 9](#_Toc8)

[3 Проектирование базы данных 10](#_Toc9)

[3.1 Инфологическая модель базы данных 10](#_Toc10)

[3.2 Даталогическая модель базы данных 16](#_Toc11)

[4 Схема базы данных на языке SQL 24](#_Toc12)

[5 Описание технических аспектов реализации базы данных 25](#_Toc13)

[5.1 Процедуры 25](#_Toc14)

[5.2 Триггеры 25](#_Toc15)

[5.3 Представления 26](#_Toc16)

[6 Тестирование Базы данных 27](#_Toc17)

[Заключение 31](#_Toc18)

[Список использованных источников 32](#_Toc19)

[Приложение А 33](#_Toc20)

# Введение

В современных условиях хозяйствования актуальной задачей является эффективное управление большими объемами информации в компаниях. Применение современных информационных технологий позволяет оптимизировать данные процессы, автоматизировать рутинные операции и принимать своевременные управленческие решения.

Одним из важных элементов автоматизации является создание корпоративных баз данных для централизованного хранения информации. Это позволяет систематизировать и структурировать данные, обеспечивает их надежное хранение и защищенность. Базы данных актуальны для любого бизнеса, включая такую сферу как розничная торговля музыкальной продукцией.

В рамках данной работы предлагается разработать базу данных для небольшого музыкального розничного магазина. Это позволит систематизировать всю информацию о товарах, поставщиках, клиентах, финансовых операциях в едином хранилище. База данных будет включать таблицы для различных типов данных, что обеспечит возможность быстрого поиска и анализа информации, а также автоматизацию ряда повторяющихся задач учета, складского хозяйства и отчетности. В дальнейшем база данных сможет служить основой для создания программной системы управления, позволяющей повысить эффективность ведения бизнеса.

# Анализ литературных источников

Разработка базы данных для автоматизации работы музыкального магазина - это сложный и многокомпонентный процесс. Одним из важных этапов этого процесса является анализ аналогов, который помогает определить наиболее оптимальные решения и выявить сильные и слабые стороны конкурентов.

На рынке существует множество баз данных для автоматизации работы магазинов, но не все они обладают необходимыми функциональными возможностями для эффективного управления музыкальным магазином и улучшения работы процессов.

Анализ аналогов позволит определить ключевые функции, которые должны быть включены в базу данных, чтобы обеспечить эффективную автоматизацию работы музыкального магазина. Это в свою очередь поможет создать уникальное и конкурентоспособное предложение на рынке баз данных для магазинов музыкальных инструментов.

Кроме того, анализ аналогов позволит выявить проблемы, с которыми сталкиваются пользователи при использовании существующих баз данных для магазинов. Это поможет улучшить качество разрабатываемой базы данных и предоставить пользователям более удобный и эффективный сервис.

Таким образом, анализ аналогов является необходимым этапом при разработке базы данных для автоматизации работы музыкального магазина. Он позволяет определить ключевые функции, выявить сильные и слабые стороны конкурентов, а также улучшить качество сервиса, предоставляемого пользователям музыкального магазина.

## 1С: Предприятие

1С: Предприятие - это программное обеспечение для автоматизации бизнес-процессов, предназначенное для малых и средних предприятий. Оно обеспечивает управление учётными записями пользователей, информацией о покупках, управлением товарами на складе и более того.

Преимущества:

* Обеспечивает автоматизацию различных аспектов бизнеса, включая учётные записи пользователей, информацию о покупках и управление товарами на складе.
* Возможность интеграции с другими продуктами 1С, такими как 1С: Accounting и 1С: Salaries, для предоставления компрехensive решения для управления всем бизнесом.
* Интуитивно понятный интерфейс и легкость использования, что делает его доступным для широкого круга пользователей.

Недостатки:

* Относительно высокая цена, особенно для маленьких бизнесов.
* Сложный интерфейс и необходимость обучения пользователей.

## SAP Business One

SAP Business One - это программное обеспечение для автоматизации бизнес-процессов, предназначенное для малых и средних предприятий. Оно обеспечивает управление учётными записями пользователей, информацией о покупках, управлением товарами на складе и более того.

Преимущества:

* Обеспечивает автоматизацию различных аспектов бизнеса, включая учётные записи пользователей, информацию о покупках и управление товарами на складе.
* Интуитивно понятный интерфейс и легкость использования, что делает его доступным для широкого круга пользователей.

Недостатки:

* Ограниченная функциональность по сравнению с другими программами, такими как 1С: Предприятия: Торговля.

## Oracle Net Suite

Oracle NetSuite - это программное обеспечение для автоматизации бизнес-процессов, предназначенное для малых и средних предприятий. Оно обеспечивает управление учётными записями пользователей, информацией о покупках, управлением товарами на складе и более того.

Преимущества:

* Обеспечивает автоматизацию различных аспектов бизнеса, включая учётные записи пользователей, информацию о покупках и управление товарами на складе.
* Возможность интеграции с другими продуктами Oracle, такими как Oracle Accounting и Oracle Inventory, для предоставления компрехensive решения для управления всем бизнезом.
* Интуитивно понятный интерфейс и легкость использования, что делает его доступным для широкого круга пользователей.

Недостатки:

* Относительно высокая цена, особенно для маленьких бизнесов.
* Сложный интерфейс и необходимость обучения пользователей.

## Microsoft Dynamics

Microsoft Dynamics 365 - это программное обеспечение для автоматизации бизнес-процессов, предназначенное для малых и средних предприятий. Оно обеспечивает управление учётными записями пользователей, информацией о покупках, управлением товарами на складе и более того.

Преимущества:

* Обеспечивает автоматизацию различных аспектов бизнеса, включая учётные записи пользователей, информацию о покупках и управление товарами на складе.
* Возможность интеграции с другими продуктами Microsoft, такими как Office и Azure, для предоставления компреhensive решения для управления всем бизнесом.

Недостатки:

* Относительно высокая цена, особенно для маленьких бизнесов.
* Сложный интерфейс и необходимость обучения пользователей.

## Zoho Inventory

Zoho Inventory - это программное обеспечение для автоматизации управления запасами, предназначенное для малых и средних предприятий. Оно обеспечивает управление учётными записями пользователей, информацией о покупках, управлением товарами на складе и более того.

Преимущества:

* Обеспечивает автоматизацию различных аспектов управления запасами, включая учётные записи пользователей, информацию о покупках и управление товарами на складе.
* Интеграция с другими продуктами Zoho, такими как Zoho Books, Zoho CRM для предоставления решений по управлению бизнесом целиком

Недостатки:

* Ограниченная функциональность по сравнению с такими программными средствами, как 1С: Предприятия: Торговля and SAP Business One.
* Относительно высокая цена, особенно, для малого бизнеса

# Постановка задачи

База данных для осуществления автоматизации работы музыкального магазина, разрабатываемая в рамках данного курсового проекта, должна обладать функционалом, способным обеспечить следующие возможности:

* управление аккаунтами пользователей программного средства;
* поддержка аутентификации и авторизации для получения данных о текущем пользователе и проверки его прав на выполнение различных действий;
* управление информацией о сотрудниках магазина;
* управление информацией о покупателях;
* управление информацией о покупках товаров в магазине;
* отслеживание статусов своих заказанных товаров покупателями;
* управление информацией об имеющихся в наличии товаров;
* управление информацией о поставщиках товаров и заказах товаров у поставщиков;
* осуществление заказов товаров покупателями;
* создание статистических отчётов о работе магазина;

# Проектирование базы данных

## Инфологическая модель базы данных

Целью инфологического (концептуального) моделирования является создание модели, которая отражает сущности, их атрибуты и связи в предметной области, обеспечение натуральных способов сбора и отображения различной информации. Этот уровень предполагает тщательное изучение предметной области и выражение ее концепций, таких как сущности, атрибуты и связи.

Сложность таких моделей определяется тем, что база может содержать бесконечное количество сущностей и бесконечное количество их связей(таких как многие-ко-многим, один-ко-многим, один-к-одному).

Исследование предметной области включает решение нескольких ключевых задач, в том числе извлечение информации, хранимой в базе данных, глубину исследования и определение границ исследования, а также предоставление естественных для пользователя методов представления и сбора данных.

Первая задача решается путем итеративного выявления требований к базе данных до тех пор, пока не будет собран весь необходимый перечень для проектирования базы данных, соответствующей особенностям предметной области. Последующие задачи решаются сопоставлением с уже полученными требованиями, чтобы охватить максимальное количество деталей и технических аспектов, при этом сохраняя соответствие предметной области.

Представим инфологическую (концептуальную) модель базы данных с учетом составленного перечня требований к проектируемой базе данных в текстовой форме в виде таблицы 3.1.

Таблица 3.1 – Инфологическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Отношение (имя на русском языке и латиницей)** | **Описание** | **Основные атрибуты** | **Краткое описание связей с другими отношениями** |
| Гитары  Guitars | Хранение информации о конкретных гитарах | Id гитары  Характеристики гитары  Рейтинг гитары  Время | FK производителя |
| Отзывы  Reviews | Хранение отзывов пользователей на гитары | Id гитары  Id пользователя-автора  Текст отзыва  Рейтинг | FK гитары  FK пользователя |
| Производители гитар GuitarsManufacturer | Хранение информации о производителях гитар | Id производителя  Описание производителя |  |
| Склад  Storehouse | Таблица для учёта гитар в наличии | Id гитары  Число гитар в наличии  Число проданных гитар  Цена  Дата следующей поставки | FK гитары |
| Поставщики гитар  GuitarProviders | Хранение информации о поставщиках гитар | Id поставщика  Название организации  Комментарий |  |

Продолжение Таблицы 3.1 – Инфологическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Поставляемые гитары | Хранение информации о поставляемых гитарах | Id гитары  Id поставщика  Число предоставляемых гитар  Цена гитары | FK гитары  FK поставщика |
| Заказы поставщикам  ProvidersOrders | Хранение информации о заказах гитар у поставщиков | Id заказа  Дата заказа  Id поставщика  Цена заказа  Id администратора, совершившего заказ  Id директора, подтвердившего заказ  Статус заказа | FK поставщика  FK администратора  FK директора |
| Список позиций заказов поставщикам | Хранение информации о заказанных гитар для каждого заказа поставщикам | Id заказа поставщикам  Id гитары  Количество заказанных гитар | FK заказа поставщикам  FK гитары |
| Аккаунты  Accounts | Хранение записей о всех аккаунтах в системе | Id аккаунта  Роль аккаунта  Логин  Хеш пароля  Адрес электронной почты  Имя пользователя |  |

Продолжение Таблицы 3.1 – Инфологическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Информация аккаунта сотрудника  EmployeeAccountInfo | Таблица расширения записей аккаунтов сотрудников магазина | Id аккаунта  Должность сотрудника  Ставка  Зарплата | FK аккаунта |
| Пасспорт  Passport | Хранение паспортных данных сотрудников | Id пасспорта  Id аккаунта сотрудника  Паспортные данные сотрудника  Активность записи | FK аккаунта сотрудника |
| Информаци аккаунта директора  DirectorsAccountInfo | Таблица расширения записей аккаунтов директоров магазина | Id аккаунта сотрудника | FK аккаунта сотрудника |
| Информаци аккаунта администратора  AdministratorsAccountInfo | Таблица расширения записей аккаунтов администраторов магазина | Id аккаунта сотрудника | FK аккаунта сотрудника |
| Информаци аккаунта курьера  CouriersAccountInfo | Таблица расширения записей аккаунтов курьеров магазина | Id аккаунта сотрудника | FK аккаунта сотрудника |
| Премии сотрудников  EmployeePremiums | Таблица для хранения премий, выписанных сотрудникам | Id премии  Id сотрудника  Размер премии  Причина  Дата | FK сотрудника |

Продолжение Таблицы 3.1 – Инфологическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Штрафы сотрудников  EmployeeFines | Таблица для хранения штрафов, выписанных сотрудникам | Id штрафа  Id сотрудника  Размер штрафа  Причина | FK сотрудника |
| Информация аккаунта покупателя  CustomersAccountInfo | Таблица расширения записей аккаунтов покупателей магазина | Id аккаунта  Дата регистрации  Комментарий | FK аккаунта |
| Избранные гитары  FavouriteGuitars | Таблица для хранения избранных гитар покупателей | Id гитары  Id покупателя | FK покупателя  FK гитары |
| Корзина покупок  CustomerShoppingBasket | Таблица для хранения корзины покупок | Id покупателя  Id гитары  Количество гитар | FK покупателя  FK гитары |
| Заказы покупателей  CustomerOrders | Таблица для хранения информации о заказах покупателей | Id заказа  Id покупателя  Id курьера  Статус заказа  Комментарий  Метод оплаты  Способ оплаты  Адрес доставки  Дата заказа | FK покупателя  FK курьера |

Продолжение Таблицы 3.1 – Инфологическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Список позиций заказов покупателей  CustomerOrderList | Таблица для хранения информации о заказанных гитар по каждому из заказов покупателей | Id заказа  Id гитары  Количество гитар  Цена гитары  Гарантийный срок | FK заказа  FK гитары |
| Архив заказов покупателей  OrderArchive | Таблица для долгосрочного хранения информации о заказах покупателей | Id записи  Id гитары  Количество гитар  Цена гитары  Дата заказа | FK гитары |

## Даталогическая модель базы данных

Данная модель является моделью БД, которая не привязана к определённой СУБД. Она основана на методе сущность-связь. Основной целью даталогического уровня проектирования базы данных является конкретизация инфологической модели и ее трансформация в схему и представление в терминах системы управления базами данных. Данная модель описывает именно данные и их связи. На данном этапе сущности, атрибуты и связи, ранее выделенные, формализуются в соответствии с правилами моделирования, применяемыми для выбранного типа базы данных, часто с учетом конкретной системы управления базами данных. Например, связь устанавливается не не между типами, а между типами, а между объектами(экземплярами) сущностей(виды связей такие же как и в инфологической модели).остей(для такой связи . Этот этап также включает в себя дальнейший анализ предметной области, который может привести к модификациям в инфологической модели. Поскольку даталогический уровень определяет будущую структуру базы данных, все принятые решения на этом этапе, а также возможные ошибки, существенно влияют на адекватность базы данных в контексте предметной области, ее удобство использования, производительность и безопасность данных.

Представим даталогическую модель базы данных в текстовом формате в виде таблицы 3.2.

Таблица 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| **Отношение БД** | **Атрибут отношения БД** | **Тип данных** | **Назначение атрибута** |
| GuitarManufacturer | idGuitarManufacturer | INT | Id производителя гитар |
| manufacturerInfo | VARCHAR(45) | Описание производителя гитар |
| Guitars | idGuitars | INT | Id гитары |
| guitarType | VARCHAR(45) | Тип гитары |
| stringCount | VARCHAR(45) | Число струн гитары |
| bodyShape | VARCHAR(45) | Тип корпуса гитары |
| topDeckMaterial | VARCHAR(45) | Материал верхней деки |
| bodyMaterial | VARCHAR(45) | Материал корпуса |
| colour | VARCHAR(45) | Цвет гитары |
| lacquerCoating | TINYINT(1) | Есть ли лаковое покрытие |
| idManufacturer | INT | Id производителя |
| modelName | VARCHAR(45) | Название модели |
| averageRating | INT | Средний рейтинг |
| ratingVotes | INT | Количество пользовательских оценок |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | iconUrl | VARCHAR(45) | Ссылка на иконку с фотографией |
| photosUrl | VARCHAR(45) | Ссылка на фотографии гитары |
| AccountRoles | idRole | INT | Id роли пользователя системы |
| roleName | VARCHAR(45) | Имя роли пользователя |
| roleTable | VARCHAR(45) | Имя таблицы модуля расширения роли пользователя |
| Accounts | idAccount | INT | Id аккаунта пользователя системы |
| idAccountRole | INT | Id роли аккаунта |
| login | VARCHAR(45) | Логин пользователя |
| passwordHash | VARCHAR(256) | Хеш пароля пользователя |
| email | VARCHAR(45) | Адрес электронной почты пользователя |
| name | VARCHAR(45) | Отображаемое имя пользователя |
| CustomersAccountInfo | idAccount | INT | Id аккаунта |
| comments | VARCHAR(200) | Комментарий к профилю пользователя |
| registrationDate | DATE | Дата регистрации |
| EmployeeRoles | idRole | INT | Id роли сотрудника |
| roleName | VARCHAR(45) | Имя роли сотрудника |
| roleTable | VARCHAR(45) | Таблица расширения для аккаунта сотрудника |
| EmployeeAccountInfo | idAccount | INT | Id аккаунта сотрудника |
| employeeRole | INT | Id роли сотрудника |
| rate | INT | Ставка сотрудника |
| salary | BIGINT | Зарплата сотрудника |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| CouriersAccountInfo | idAccount | INT | Id аккаунта курьера |
| Region | idRegion | INT | Id области |
| name | VARCHAR(45) | Название области |
| City | idCity | INT | Id города |
| name | VARCHAR(45) | Имя города |
| idRegion | INT | Id области |
| Street | idStreet | INT | Id улицы |
| name | VARCHAR(45) | Название улицы |
| Address | idAddress | INT | Id адреса |
| idRegion | INT | Id области |
| idCity | INT | Id города |
| idStreet | INT | Id улицы |
| buildingNumber | INT | Номер здания |
| CustomerOrders | idOrder | INT | Id заказа |
| idCustomer | INT | Id покупателя |
| idCourier | INT | Id курьера |
| status | ENUM('not confirmed', 'confirmed', 'paid', 'delivered') | Статус заказа |
| comment | VARCHAR(200) | Комментарий заказа |
| paymentType | ENUM('upon receipt', 'upon order') | Тип оплаты заказа |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
|  | paymentMean | ENUM('card', 'cash') | Способ оплаты заказа |
| idAddress | INT | Id адреса доставки |
| date | DATE | Дата соавершения заказа |
| FavouriteGuitars | idAccount | INT | Id аккаунта покупателя |
| idGuitar | INT | Id выбранной гитары |
| CustomerShoppingBasket | idAccount | INT | Id аккаунта покупателя |
| idGuitar | INT | Id выбранной гитары |
| guitarCount | INT | Количество гитар выбранного типа |
| GuitarProviders | idProvider | INT | Id поставщика гитар |
| organizationName | VARCHAR(45) | Название организации-поставщика |
| comments | VARCHAR(200) | Комментарий к поставщику гитар |
| ProvidedGuitars | idGuitar | INT | Id гитары, которую могут поставить |
| idProvider | INT | Id поставщика |
| guitarCount | INT | Доступное количество гитар |
| guitarPrice | BIGINT | Цена за 1 гитару данного типа |
| AdministratorsAccountInfo | idAccount | INT | Id аккаунта администратора |
| DirectorsAccountInfo | idAccount | INT | Id аккаунта директора |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| ProvidersOrders | idProviderOrder | INT | Id заказа поставщику |
| date | DATE | Дата заказа |
| idProvider | INT | Id поставщика |
| totalPrice | BIGINT | Стоимость всего заказа |
| idAdministratorAuthor | INT | Id администратора, оформившегося заказ |
| idDirectorThatConfirmed | INT | Id директора, подтвердившего заказ |
| status | ENUM('planning', 'confirmed', 'paid', 'delivered') | Статус заказа |
| ProvidersOrdersList | idProvidersOrders | INT | Id позиции заказа поставщику |
| idGuitar | INT | Id заказанной гитары |
| count | INT | Количество заказанных гитар |
| EmployeeFines | idFine | INT | Id штрафов |
| idEmployee | INT | Id сотрудника |
| fine | BIGINT | Размер штрафа |
| reason | VARCHAR(45) | Причина штрафа |
| date | DATE | Дата |
| EmployeePremiums | idPremium | INT | Id премии |
| idEmployee | INT | Id сотрудника |
| premium | BIGINT | Размер премии |
| reason | VARCHAR(45) | Причина |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Passport | idPassport | INT | Id паспорта |
| idEmployeeAccount | INT | Id аккаунта сотрудника |
| firstName | VARCHAR(45) | Имя сотрудника |
| surname | VARCHAR(45) | Фамилия сотрудника |
| patronymic | VARCHAR(45) | Отчество сотрудника |
| birthDate | DATE | Дата рождения сотрудника |
| serialNumber | VARCHAR(45) | Серийный номер паспорта |
| idNumber | VARCHAR(45) | Id паспорта |
| isActive | TINYINT(1) | Поле активности записи паспорта в системе |
| Reviews | idGuitar | INT | Id описываемой гитары |
| idAccount | INT | Id аккаунта автора |
| text | CARCHAR(400) | Текст отзыва |
| rating | INT | Поставленный рейтинг |
| Storehouse | idGuitar | INT | Id гитары на складе |
| availableCount | INT | Оставшееся количество гитар |
| soldCount | INT | Проданное количество гитар |
| nextSupply | DATE | Дата следующей поставки |
| price | BIGINT | Цена одной гитары |
| CustomerOrderList | idOrder | INT | Id заказа |
| idGuitar | INT | Id гитары в заказе |
| count | INT | Количество заказанных гитар |
| guitarPrice | INT | Цена одной гитары |
| guaranteeExpireDate | DATE | Дата истечения гарантии |

Продолжение Таблицы 3.2 – Даталогическая модель базы данных

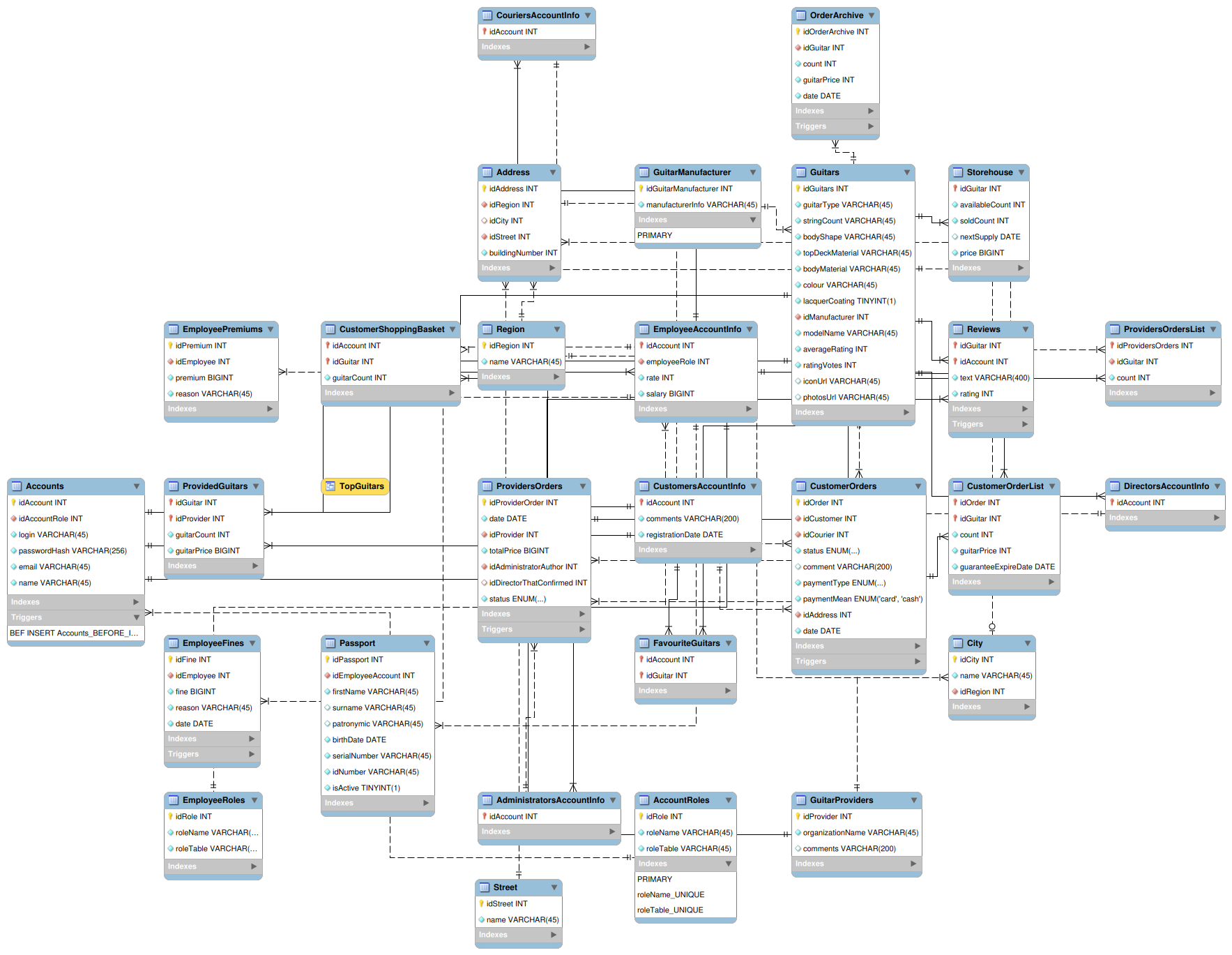
|  |  |  |  |
| --- | --- | --- | --- |
| OrderArchive | idOrderArchive | INT | Id заказа в архиве |
| idGuitar | INT | Id гитары |
| count | INT | Количество гитар в заказе |
| guitarPrice | INT | Цена гитары |
| date | DATE | Дата совершения заказа |

# Схема базы данных на языке SQL

Структура базы данных предоставлена в Приложении А в виде SQL-дампа, который содержит подробную информацию о создании таблиц, установке связей и других ключевых элементах базы данных.

На рисунке 4.1 представлена схема базы данных MySQL.

Рисунок 4.1 – Схема базы данных MySQL



# Описание технических аспектов реализации базы данных

## Процедуры

Таблица 5.1 – Хранимые процедуры базы данных

|  |  |  |
| --- | --- | --- |
| **Название** | **Входные параметры** | **Описание** |
| CreateOrder | idAccount | Формирует запись о новом заказе для указанного покупателя с заданными параметрами |
| idCourier |
| comment |
| paymentType |
| paymentMean |
| idAddress |
| FillOrderFromBasket | orderId | Формирование списка заказа путём копирования всех выбранных в пользовательской корзине позиций, с последующим очищением корзины |

## Триггеры

Таблица 5.2 – Триггеры базы данных

|  |  |
| --- | --- |
| **Название** | **Описание** |
| Accounts\_BEFORE\_INSERT | Проверка корректности логина и пароля создаваемого аккаунта – строка логина имеет как минимум 8 символов, логин и пароль не совпадают |
| CustomerOrders\_BEFORE\_INSERT | Автоматическая установка даты заказа текущей датой |
| CustomerOrders\_BEFORE\_UPDATE | Корректирование даты истечения гарантии при обновлении статуса доставленности заказа – при установке статуса «Доставлен» |
| CustomerOrders\_BEFORE\_DELETE | Проверка возможности удаления аккаунта пользователя – нельзя удалить пользователя с неистёкшей гарантией либо с недоставленными заказами |
| ProvidersOrders\_BEFORE\_INSERT | Автоматическая установка даты заказа текущей датой |
| EmployeeFines\_BEFORE\_INSERT | Автоматическая установка даты заказа текущей датой |
| Reviews\_AFTER\_INSERT | Автоматическое обновление среднего рейтинга гитары при добавлении нового отзыва на неё |
| OrderArchive\_BEFORE\_INSERT | Проверка корректности установки даты для заказа |

## Представления

Таблица 5.3 – Представления базы данных

|  |  |
| --- | --- |
| **Название** | **Описание** |
| TopGuitars | Представляет 10 самых популярных гитар в соответствии с пользовательским рейтингом |

# Тестирование Базы данных

В ходе тестирования был практическим способом проверен функционал разработанной базы данных на корректность работы при нормальных условиях, а также при условии некорректного использования со стороны клиента.

Таблица 6.1 – Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Описание теста | Шаги воспроизведения | Ожидаемый результат |
| 0 | Проверка добавления отзывов | Авторизироваться в системе как покупатель и оставить отзыв на гитару. | Корректное появление отзыва на гитару и изменения ее рейтинга в зависимости от оценки. |
| 1 | Проверка списка любимых гитар | Авторизироваться в системе как покупатель и выбрать любую гитару. Добавить ее в список любимых. | Корректное добавление и отображение гитары в списке. |
| 2 | Попытка создать некорректный аккаунт | Попытаться создать аккаунт с одинаковым паролем и логином. Попытаться создать аккаунт со слишком маленьким логином. Попытаться создать аккаунт с уже существующим логином | Отказ при создании аккаунта. |
| 3 | Просмотр информации о гитаре | Авторизироваться в системе как покупатель . Перейти на страницу любой гитары. | Корректное отображение всех характеристик гитары. |

Продолжение Таблицы 6.1 – Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | Просмотр информации о другом аккаунте | Авторизироваться в системе как покупатель . Перейти на страницу любого человека. | Корректное отображение всех публичных полей. Имя, почта, логин. Не отображение пароля. Отсутствие информации о паспорте. |
| 5 | Просмотр информации о своем аккаунте | Авторизироваться в системе как покупатель . Перейти на свою страницу. | Корректное отображение всех полей. Имя, почта, логин, паспорт, пароль. |
| 6 | Просмотр информации о своем паспорте | Авторизироваться в системе как покупатель . Перейти на страницу своего паспорта. | Корректное отображение всех характеристик паспорта. |
| 7 | Создание аккаунта работником магазина | Создать в системе новый аккаунт. Добавить его в список администраторов с другого администраторского аккаунта | Корректное создание аккаунта и его добавление в список администраторов. |
| 8 | Создание аккаунта курьера | Создать в системе новый аккаунт. Добавить его в список курьеров с администраторского аккаунта | Корректное создание аккаунта и его добавление в список курьеров. |
| 9 | Добавление нового города в систему. | Зайти в систему как администратор. Добавить новый город в систему. | Корректное добавление города в систему и появление его в списке городов. |
| 10 | Добавление нового производителя гитар | Зайти в систему как администратор. Добавить нового производителя в систему. | Корректное добавление города в систему и появление его в списке городов. |

Продолжение Таблицы 6.1 – Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| 11 | Изменение комментариев о производителе гитар | Зайти в систему как администратор. Изменить/добавить комментарий к странице производителя гитар. | Корректное добавление комментария в систему. Видимость комментария для обычных аккаунтов |
| 12 | Проверка работы корзины покупателя | Зайти в систему как покупатель. Добавить в свою корзину несколько гитар разных типов. | Корректное изменение стоимости заказа. Корректное отображение списка гитар в корзине. |
| 13 | Добавление новой улицы в систему. | Зайти в систему как администратор. Добавить новую улицу в систему. | Корректное добавление улицы в систему и появление ее в списке улиц. |
| 14 | Добавление нового региона в систему. | Зайти в систему как администратор. Добавить новый регион в систему. | Корректное добавление региона в систему и появление его в списке регионов. |
| 15 | Добавление нового заказа | Зайти в систему как покупатель. Добавить гитару в корзину и заказать ее. Указать свой адрес. Выбрать тип оплаты. | Корректная обработка заказа. Выбор курьера. Переход заказа в тип обработка. |
| 16 | Обработка заказа | Зайти в систему как работник магазина. Изменить статус заказа. | Корректное изменение статуса. Видимость нового статуса с аккаунта покупателя. |

Продолжение Таблицы 6.1 – Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| 17 | Завершение заказа | Зайти в систему как работник магазина. Изменить статус заказа на доставлен. | Корректное добавление заказа в архив и исчезновение из списка активных заказов. |
| 18 | Тест удаления аккаунта покупателя | Зайти в аккаунт. Удалить свой аккаунт | Корректное удаление аккаунта если у аккаунта нет связи с активным заказом иначе ошибка. |

Выполнение всех вышеуказанных функций закончилось успехом. Данные тесты позволяют сказать, что база данных музыкального магазина корректна.

Заключение

В результате проделанной работы во время создания курсового проекта, была разработана база данных музыкального магазина, служащая для упрощения работы пользователей, поставщиков и сотрудников. В ходе разработки мы получили систему, которая позволяет получать, добавлять и управлять информацией о текущих пользователях системы, а также добавлять новых. Также база обеспечивает отслеживание поставок, продаж и покупок товаров магазина (что, в свою очередь, позволяет создавать статистические отчеты для получения информации о текущем состоянии склада), в том числе и отслеживание промежуточных этапов заказа, что значительно упрощает процесс покупки для заказчика.

База данных работает корректно, в том числе и при некорректном использовании клиентом, что было подтверждено многочисленными проведёнными тестами, описанными выше.

База создана с помощью популярной системы управления базами данных MySQL, которая выделяется своим уровнем безопасности и простотой использования, что отражается в коде, создающем базу данных. Можно отметить высокую скорость создания базы и обработки запросов.

Также нужно отметить, что сам интерфейс базы данных интуитивно понятен и не нуждается в уточнении принимаемых переменными значений, а их названия несут смысловую нагрузку, которая позволяет понять, что в ней хранится.

Список использованных источников

[1] SQLShack [Электронный ресурс] – Режим доступа: https://www.sqlshack.com/ – Дата доступа: 15.12.2023.

[2] MySQL Documentation[Электронный ресурс] – Режим доступа: https://dev.mysql.com/doc/ – Дата доступа: 15.12.2023.

[3] SQL Server Tips, Articles and Training[Электронный ресурс] – Режим доступа: https://www.mssqltips.com/ – Дата доступа: 15.12.2023.

[4] DigitalOcean Documentation[Электронный ресурс] – Режим доступа: https://docs.digitalocean.com/products/databases/mysql/ – Дата доступа: 15.12.2023.

[5] Oracle MySQL online documentation[Электронный ресурс] – Режим доступа: https://docs.oracle.com/cd/E17952\_01/index.html – Дата доступа: 15.12.2023.

[6]SQL QuickStart Guide/ Walter Shields 2019. — 252 с.

[7] SQL Practice Problems 57. beginning, intermediate, and advanced challenges for you to solve using a "learn-bydoing" approach /Sylvia Moestl Vasilik,2016. — 346 c.

[8] Реляционные базы данных в примерах / Святослав Куликов, 2023. – 424 с.

[9] Работа с MySQL, MS SQL Server и Oracle в примерах / Святослав Куликов, 2022. – 529 с

Приложение А

(обязательное)

Схема базы данных на языке SQL

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema guitars\_db

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `guitars\_db` DEFAULT CHARACTER SET utf8 COLLATE utf8\_bin ;

USE `guitars\_db` ;

-- -----------------------------------------------------

-- Table `guitars\_db`.`GuitarManufacturer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`GuitarManufacturer` (

`idGuitarManufacturer` INT NOT NULL,

`manufacturerInfo` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idGuitarManufacturer`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Guitars`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Guitars` (

`idGuitars` INT NOT NULL,

`guitarType` VARCHAR(45) NOT NULL,

`stringCount` VARCHAR(45) NOT NULL,

`bodyShape` VARCHAR(45) NOT NULL,

`topDeckMaterial` VARCHAR(45) NOT NULL,

`bodyMaterial` VARCHAR(45) NOT NULL,

`colour` VARCHAR(45) NOT NULL,

`lacquerCoating` TINYINT(1) NOT NULL,

`idManufacturer` INT NOT NULL,

`modelName` VARCHAR(45) NOT NULL,

`averageRating` INT NOT NULL,

`ratingVotes` INT NOT NULL,

`iconUrl` VARCHAR(45) NULL,

`photosUrl` VARCHAR(45) NULL,

PRIMARY KEY (`idGuitars`),

INDEX `idManufacturer\_idx` (`idManufacturer` ASC) VISIBLE,

INDEX `idxGuitarType` (`guitarType` ASC) VISIBLE,

INDEX `idxModelName` (`modelName` ASC) INVISIBLE,

INDEX `idxRating` (`averageRating` ASC) VISIBLE,

INDEX `idxStringCount` (`stringCount` ASC) INVISIBLE,

INDEX `idxBodyShape` (`bodyShape` ASC) INVISIBLE,

INDEX `idxTopDeckMaterial` (`topDeckMaterial` ASC) INVISIBLE,

INDEX `idxBodyMaterial` (`bodyMaterial` ASC) INVISIBLE,

INDEX `idxColour` (`colour` ASC) INVISIBLE,

INDEX `idxCoating` (`lacquerCoating` ASC) INVISIBLE,

INDEX `idxRatingVotes` (`ratingVotes` ASC) VISIBLE,

CONSTRAINT `Guitars\_idManufacturer`

FOREIGN KEY (`idManufacturer`)

REFERENCES `guitars\_db`.`GuitarManufacturer` (`idGuitarManufacturer`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`AccountRoles`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`AccountRoles` (

`idRole` INT NOT NULL,

`roleName` VARCHAR(45) NOT NULL,

`roleTable` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idRole`),

UNIQUE INDEX `roleName\_UNIQUE` (`roleName` ASC) VISIBLE,

UNIQUE INDEX `roleTable\_UNIQUE` (`roleTable` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Accounts`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Accounts` (

`idAccount` INT NOT NULL,

`idAccountRole` INT NOT NULL,

`login` VARCHAR(45) NOT NULL,

`passwordHash` VARCHAR(256) NOT NULL,

`email` VARCHAR(45) NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idAccount`),

INDEX `idxLogin` (`login` ASC) INVISIBLE,

INDEX `idxEmail` (`email` ASC) VISIBLE,

UNIQUE INDEX `login\_UNIQUE` (`login` ASC) VISIBLE,

UNIQUE INDEX `email\_UNIQUE` (`email` ASC) VISIBLE,

INDEX `accounts\_idRole\_idx` (`idAccountRole` ASC) VISIBLE,

CONSTRAINT `accounts\_idRole`

FOREIGN KEY (`idAccountRole`)

REFERENCES `guitars\_db`.`AccountRoles` (`idRole`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`CustomersAccountInfo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`CustomersAccountInfo` (

`idAccount` INT NOT NULL,

`comments` VARCHAR(200) NOT NULL,

`registrationDate` DATE NOT NULL,

PRIMARY KEY (`idAccount`),

CONSTRAINT `CustomersAccountInfo\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`Accounts` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`EmployeeRoles`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`EmployeeRoles` (

`idRole` INT NOT NULL,

`roleName` VARCHAR(45) NOT NULL,

`roleTable` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idRole`),

UNIQUE INDEX `roleName\_UNIQUE` (`roleName` ASC) VISIBLE,

UNIQUE INDEX `roleTable\_UNIQUE` (`roleTable` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`EmployeeAccountInfo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`EmployeeAccountInfo` (

`idAccount` INT NOT NULL,

`employeeRole` INT NOT NULL,

`rate` INT NOT NULL,

`salary` BIGINT NOT NULL,

PRIMARY KEY (`idAccount`),

INDEX `idRole\_idx` (`employeeRole` ASC) VISIBLE,

CONSTRAINT `EmployeeAccountInfo\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`Accounts` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `EmployeeAccountInfo\_idRole`

FOREIGN KEY (`employeeRole`)

REFERENCES `guitars\_db`.`EmployeeRoles` (`idRole`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`CouriersAccountInfo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`CouriersAccountInfo` (

`idAccount` INT NOT NULL,

PRIMARY KEY (`idAccount`),

CONSTRAINT `CouriersAccountInfo\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Region`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Region` (

`idRegion` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idRegion`),

INDEX `idxName` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`City`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`City` (

`idCity` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

`idRegion` INT NOT NULL,

PRIMARY KEY (`idCity`),

INDEX `idRegion\_idx` (`idRegion` ASC) VISIBLE,

INDEX `idxName` (`name` ASC) VISIBLE,

CONSTRAINT `City\_idRegion`

FOREIGN KEY (`idRegion`)

REFERENCES `guitars\_db`.`Region` (`idRegion`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Street`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Street` (

`idStreet` INT NOT NULL,

`name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idStreet`),

INDEX `idxName` (`name` ASC) VISIBLE)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Address`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Address` (

`idAddress` INT NOT NULL,

`idRegion` INT NOT NULL,

`idCity` INT NULL,

`idStreet` INT NOT NULL,

`buildingNumber` INT NOT NULL,

PRIMARY KEY (`idAddress`),

INDEX `idxRegion` (`idRegion` ASC) VISIBLE,

INDEX `idCity\_idx` (`idCity` ASC) VISIBLE,

INDEX `idStreet\_idx` (`idStreet` ASC) VISIBLE,

CONSTRAINT `Address\_idRegion`

FOREIGN KEY (`idRegion`)

REFERENCES `guitars\_db`.`Region` (`idRegion`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `Address\_idCity`

FOREIGN KEY (`idCity`)

REFERENCES `guitars\_db`.`City` (`idCity`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `Address\_idStreet`

FOREIGN KEY (`idStreet`)

REFERENCES `guitars\_db`.`Street` (`idStreet`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`CustomerOrders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`CustomerOrders` (

`idOrder` INT NOT NULL AUTO\_INCREMENT,

`idCustomer` INT NOT NULL,

`idCourier` INT NOT NULL,

`status` ENUM('not confirmed', 'confirmed', 'paid', 'delivered') NOT NULL,

`comment` VARCHAR(200) NULL,

`paymentType` ENUM('upon receipt', 'upon order') NOT NULL,

`paymentMean` ENUM('card', 'cash') NOT NULL,

`idAddress` INT NOT NULL,

`date` DATE NOT NULL,

PRIMARY KEY (`idOrder`),

INDEX `idCustomer\_idx` (`idCustomer` ASC) VISIBLE,

INDEX `idCurier\_idx` (`idCourier` ASC) VISIBLE,

INDEX `idAddress\_idx` (`idAddress` ASC) VISIBLE,

INDEX `idxStatus` (`status` ASC) VISIBLE,

INDEX `idxDate` (`date` ASC) VISIBLE,

CONSTRAINT `CustomerOrders\_idCustomer`

FOREIGN KEY (`idCustomer`)

REFERENCES `guitars\_db`.`CustomersAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `CustomerOrders\_idCurier`

FOREIGN KEY (`idCourier`)

REFERENCES `guitars\_db`.`CouriersAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `CustomerOrders\_idAddress`

FOREIGN KEY (`idAddress`)

REFERENCES `guitars\_db`.`Address` (`idAddress`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`FavouriteGuitars`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`FavouriteGuitars` (

`idAccount` INT NOT NULL,

`idGuitar` INT NOT NULL,

PRIMARY KEY (`idAccount`, `idGuitar`),

INDEX `idGuitar\_idx` (`idGuitar` ASC) VISIBLE,

CONSTRAINT `FavouriteGuitars\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`CustomersAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `FavouriteGuitars\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`CustomerShoppingBasket`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`CustomerShoppingBasket` (

`idAccount` INT NOT NULL,

`idGuitar` INT NOT NULL,

`guitarCount` INT NOT NULL,

PRIMARY KEY (`idAccount`, `idGuitar`),

INDEX `idGuitars\_idx` (`idGuitar` ASC) VISIBLE,

CONSTRAINT `CustomerShoppingBasket\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`CustomersAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `CustomerShoppingBasket\_idGuitars`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`GuitarProviders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`GuitarProviders` (

`idProvider` INT NOT NULL,

`organizationName` VARCHAR(45) NOT NULL,

`comments` VARCHAR(200) NULL,

PRIMARY KEY (`idProvider`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`ProvidedGuitars`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`ProvidedGuitars` (

`idGuitar` INT NOT NULL,

`idProvider` INT NOT NULL,

`guitarCount` INT NOT NULL,

`guitarPrice` BIGINT NOT NULL,

PRIMARY KEY (`idGuitar`, `idProvider`),

INDEX `idProvider\_idx` (`idProvider` ASC) VISIBLE,

CONSTRAINT `providerGuitars\_idProvider`

FOREIGN KEY (`idProvider`)

REFERENCES `guitars\_db`.`GuitarProviders` (`idProvider`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `providerGuitars\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`AdministratorsAccountInfo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`AdministratorsAccountInfo` (

`idAccount` INT NOT NULL,

PRIMARY KEY (`idAccount`),

CONSTRAINT `AdministratorsAccountInfo\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`DirectorsAccountInfo`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`DirectorsAccountInfo` (

`idAccount` INT NOT NULL,

PRIMARY KEY (`idAccount`),

CONSTRAINT `DirectorsAccountInfo\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`ProvidersOrders`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`ProvidersOrders` (

`idProviderOrder` INT NOT NULL,

`date` DATE NOT NULL,

`idProvider` INT NOT NULL,

`totalPrice` BIGINT NOT NULL,

`idAdministratorAuthor` INT NOT NULL,

`idDirectorThatConfirmed` INT NULL,

`status` ENUM('planning', 'confirmed', 'paid', 'delivered') NOT NULL,

PRIMARY KEY (`idProviderOrder`),

INDEX `idProvider\_idx` (`idProvider` ASC) VISIBLE,

INDEX `idAdministrator\_idx` (`idAdministratorAuthor` ASC) VISIBLE,

INDEX `idDirector\_idx` (`idDirectorThatConfirmed` ASC) VISIBLE,

INDEX `idxStatus` (`status` ASC) INVISIBLE,

INDEX `idxDate` (`date` ASC) VISIBLE,

CONSTRAINT `ProvidersOrders\_idProvider`

FOREIGN KEY (`idProvider`)

REFERENCES `guitars\_db`.`GuitarProviders` (`idProvider`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `ProvidersOrders\_idAdministrator`

FOREIGN KEY (`idAdministratorAuthor`)

REFERENCES `guitars\_db`.`AdministratorsAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `ProvidersOrders\_idDirector`

FOREIGN KEY (`idDirectorThatConfirmed`)

REFERENCES `guitars\_db`.`DirectorsAccountInfo` (`idAccount`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`ProvidersOrdersList`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`ProvidersOrdersList` (

`idProvidersOrders` INT NOT NULL,

`idGuitar` INT NOT NULL,

`count` INT NOT NULL,

PRIMARY KEY (`idProvidersOrders`),

INDEX `idGuitar\_idx` (`idGuitar` ASC) VISIBLE,

CONSTRAINT `ProvidersOrdersList\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `ProvidersOrdersList\_idProvidersOrders`

FOREIGN KEY (`idProvidersOrders`)

REFERENCES `guitars\_db`.`ProvidersOrders` (`idProviderOrder`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`EmployeeFines`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`EmployeeFines` (

`idFine` INT NOT NULL,

`idEmployee` INT NOT NULL,

`fine` BIGINT NOT NULL,

`reason` VARCHAR(45) NOT NULL,

`date` DATE NOT NULL,

PRIMARY KEY (`idFine`),

INDEX `idEmployee\_idx` (`idEmployee` ASC) VISIBLE,

INDEX `idxDate` (`date` ASC) VISIBLE,

CONSTRAINT `EmployeeFines\_idEmployee`

FOREIGN KEY (`idEmployee`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`EmployeePremiums`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`EmployeePremiums` (

`idPremium` INT NOT NULL,

`idEmployee` INT NOT NULL,

`premium` BIGINT NOT NULL,

`reason` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idPremium`),

INDEX `idEmployee\_idx` (`idEmployee` ASC) VISIBLE,

CONSTRAINT `EmployeePremiums\_idEmployee`

FOREIGN KEY (`idEmployee`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Passport`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Passport` (

`idPassport` INT NOT NULL,

`idEmployeeAccount` INT NOT NULL,

`firstName` VARCHAR(45) NOT NULL,

`surname` VARCHAR(45) NULL,

`patronymic` VARCHAR(45) NULL,

`birthDate` DATE NOT NULL,

`serialNumber` VARCHAR(45) NOT NULL,

`idNumber` VARCHAR(45) NOT NULL,

`isActive` TINYINT(1) NOT NULL,

PRIMARY KEY (`idPassport`),

INDEX `idEmployeeAccount\_idx` (`idEmployeeAccount` ASC) VISIBLE,

INDEX `idxIsActive` (`isActive` ASC) VISIBLE,

CONSTRAINT `Passport\_idEmployeeAccount`

FOREIGN KEY (`idEmployeeAccount`)

REFERENCES `guitars\_db`.`EmployeeAccountInfo` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Reviews`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Reviews` (

`idGuitar` INT NOT NULL,

`idAccount` INT NOT NULL,

`text` VARCHAR(400) NOT NULL,

`rating` INT NOT NULL,

INDEX `idGuitar'\_idx` (`idGuitar` ASC) VISIBLE,

INDEX `idAccount\_idx` (`idAccount` ASC) VISIBLE,

PRIMARY KEY (`idGuitar`, `idAccount`),

INDEX `idxRating` (`rating` ASC) VISIBLE,

CONSTRAINT `Reviews\_idGuitars`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT,

CONSTRAINT `Reviews\_idAccount`

FOREIGN KEY (`idAccount`)

REFERENCES `guitars\_db`.`Accounts` (`idAccount`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`Storehouse`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`Storehouse` (

`idGuitar` INT NOT NULL,

`availableCount` INT NOT NULL,

`soldCount` INT NOT NULL,

`nextSupply` DATE NULL,

`price` BIGINT NOT NULL,

PRIMARY KEY (`idGuitar`),

INDEX `idxCountGreaterZero` (`availableCount` ASC) VISIBLE,

INDEX `idxAvailableCount` (`availableCount` ASC) VISIBLE,

INDEX `idxPrice` (`price` ASC) VISIBLE,

CONSTRAINT `Storehouse\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE CASCADE

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`CustomerOrderList`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`CustomerOrderList` (

`idOrder` INT NOT NULL,

`idGuitar` INT NOT NULL,

`count` INT NOT NULL,

`guitarPrice` INT NOT NULL,

`guaranteeExpireDate` DATE NOT NULL,

PRIMARY KEY (`idOrder`, `idGuitar`),

INDEX `idGuitar\_idx` (`idGuitar` ASC) VISIBLE,

INDEX `idxGuarantee` (`guaranteeExpireDate` ASC) VISIBLE,

CONSTRAINT `CustomerOrderList\_idOrder`

FOREIGN KEY (`idOrder`)

REFERENCES `guitars\_db`.`CustomerOrders` (`idOrder`)

ON DELETE CASCADE

ON UPDATE RESTRICT,

CONSTRAINT `CustomerOrderList\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE RESTRICT

ON UPDATE RESTRICT)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `guitars\_db`.`OrderArchive`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`OrderArchive` (

`idOrderArchive` INT NOT NULL AUTO\_INCREMENT,

`idGuitar` INT NOT NULL,

`count` INT NOT NULL,

`guitarPrice` INT NOT NULL,

`date` DATE NOT NULL,

PRIMARY KEY (`idOrderArchive`),

INDEX `idxDate` (`date` ASC) VISIBLE,

INDEX `idGuitar\_idx` (`idGuitar` ASC) VISIBLE,

CONSTRAINT `OrderArchive\_idGuitar`

FOREIGN KEY (`idGuitar`)

REFERENCES `guitars\_db`.`Guitars` (`idGuitars`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

USE `guitars\_db` ;

-- -----------------------------------------------------

-- Placeholder table for view `guitars\_db`.`TopGuitars`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `guitars\_db`.`TopGuitars` (`idGuitars` INT, `modelName` INT, `averageRating` INT, `ratingVotes` INT);

-- -----------------------------------------------------

-- procedure CreateOrder

-- -----------------------------------------------------

DELIMITER $$

USE `guitars\_db`$$

CREATE PROCEDURE CreateOrder(

idAccount INT,

idCourier INT,

comment VARCHAR(200),

paymentType ENUM('upon receipt', 'upon order'),

paymentMean ENUM('card', 'cash'),

idAddress INT)

BEGIN

INSERT INTO `CustomerOrders` (idCustomer, idCourier, status, comment, paymentType, paymentMean, idAddress, date)

VALUES (idAccount, idCourier, 'not confirmed', comment, paymentType, paymentMean, idAddress, DATE(NOW()));

END;$$

DELIMITER ;

-- -----------------------------------------------------

-- procedure FillOrderFromBasket

-- -----------------------------------------------------

DELIMITER $$

USE `guitars\_db`$$

CREATE DEFINER=`root`@`localhost` PROCEDURE `FillOrderFromBasket`(orderID INT)

BEGIN

DECLARE user INT DEFAULT 0;

SELECT idCustomer

INTO user

FROM CustomerOrders

WHERE `idOrder` = orderID;

INSERT INTO `CustomerOrderList` (`idOrder`, `idGuitar`, `count`, `guitarPrice`, `guaranteeExpireDate`)

SELECT orderID, storehouse.idGuitar, `guitarCount`, `price`, '9999-12-31'

FROM `CustomerShoppingBasket`

JOIN Storehouse ON CustomerShoppingBasket.idGuitar = Storehouse.idGuitar

WHERE idAccount = user;

DELETE FROM CustomerShoppingBasket

WHERE idAccount = user;

END$$

DELIMITER ;

-- -----------------------------------------------------

-- View `guitars\_db`.`TopGuitars`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `guitars\_db`.`TopGuitars`;

USE `guitars\_db`;

CREATE OR REPLACE VIEW `TopGuitars` AS

SELECT idGuitars, modelName, averageRating, ratingVotes

FROM Guitars

ORDER BY averageRating DESC, ratingVotes DESC

LIMIT 10;

USE `guitars\_db`;

DELIMITER $$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`Accounts\_BEFORE\_INSERT` BEFORE INSERT ON `Accounts` FOR EACH ROW

BEGIN

IF LENGTH(NEW.login) < 8 OR (SHA2(NEW.login, 256) = NEW.passwordHash) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Invalid credentials';

END IF;

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`CustomerOrders\_BEFORE\_INSERT` BEFORE INSERT ON `CustomerOrders` FOR EACH ROW

BEGIN

SET NEW.date = DATE(NOW());

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`CustomerOrders\_BEFORE\_UPDATE` BEFORE UPDATE ON `CustomerOrders` FOR EACH ROW

BEGIN

IF NEW.status = 'delivered' THEN

UPDATE `CustomerOrderList` SET `guaranteeExpireDate` = DATE(NOW())

WHERE `idOrder` = OLD.idOrder;

END IF;

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`CustomerOrders\_BEFORE\_DELETE` BEFORE DELETE ON `CustomerOrders` FOR EACH ROW

BEGIN

DECLARE countUnexpired INT;

SELECT COUNT(\*) INTO countUnexpired

FROM CustomerOrderList

WHERE idOrder = OLD.idOrder AND guaranteeExpireDate >= DATE(NOW());

IF countUnexpired > 0 THEN

SIGNAL SQLSTATE "45000"

SET MESSAGE\_TEXT = "Cannot delete user with unexpired guarantees";

END IF;

IF OLD.status = "delivered" THEN

INSERT INTO OrderArchive (idGuitar, count, guitarPrice, date)

SELECT idGuitar, count, guitarPrice, OLD.date

FROM CustomerOrderList

WHERE idOrder = OLD.idOrder;

END IF;

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`ProvidersOrders\_BEFORE\_INSERT` BEFORE INSERT ON `ProvidersOrders` FOR EACH ROW

BEGIN

SET NEW.date = NOW();

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`EmployeeFines\_BEFORE\_INSERT` BEFORE INSERT ON `EmployeeFines` FOR EACH ROW

BEGIN

SET NEW.date = NOW();

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`Reviews\_AFTER\_INSERT` AFTER INSERT ON `Reviews` FOR EACH ROW

BEGIN

DECLARE rating INT;

DECLARE votes INT;

SELECT averageRating, ratingVotes INTO rating, votes

FROM Guitars

WHERE idGuitars = NEW.idGuitar;

SET rating = rating \* votes;

SET votes = votes + 1;

SET rating = (rating + NEW.rating) / votes;

UPDATE Guitars

SET averageRating = rating, ratingVotes = votes

WHERE idGuitars = NEW.idGuitar;

END$$

USE `guitars\_db`$$

CREATE DEFINER = CURRENT\_USER TRIGGER `guitars\_db`.`OrderArchive\_BEFORE\_INSERT` BEFORE INSERT ON `OrderArchive` FOR EACH ROW

BEGIN

IF NEW.date > NOW() THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot use specified date for done order';

END IF;

END$$

DELIMITER ;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

-- -----------------------------------------------------

-- Data for table `guitars\_db`.`EmployeeRoles`

-- -----------------------------------------------------

START TRANSACTION;

USE `guitars\_db`;

INSERT INTO `guitars\_db`.`EmployeeRoles` (`idRole`, `roleName`, `roleTable`) VALUES (0, 'Director', 'DirectorsAccountInfo');

INSERT INTO `guitars\_db`.`EmployeeRoles` (`idRole`, `roleName`, `roleTable`) VALUES (1, 'Administrator', 'AdministratorsAccountInfo');

INSERT INTO `guitars\_db`.`EmployeeRoles` (`idRole`, `roleName`, `roleTable`) VALUES (2, 'Courier', 'CouriersAccountInfo');

COMMIT;

**ВЕДОМОСТЬ ДОКУМЕНТОВ**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 027 ПЗ | | | | Пояснительная записка | | | | 47 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 051006 027 ПД | | | | Схема данных | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 027 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Программное средство для автоматизации управления музыкальным магазином |  | | | | Лист | Листов |
| Разраб. | | Шуляк А. В. |  | 11.12.23 | Т |  | |  | 47 | 47 |
| Пров. | | Медведев С. А. |  | 15.12.23 | Кафедра ПОИТ  гр. 051006 | | | | | |
|  | |  |  |  |
|  | |  |  |  |
|  | |  |  |  |