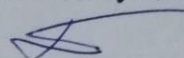


МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

Работа допущена к защите


 Руководитель

« 19 » 12 20 18 г.

КУРСОВАЯ РАБОТА

по дисциплине «Базы данных»

на тему: «Разработка и реализация базы данных для онлайн магазина по
продаже музыкальных композиций»

Студент  Шорин В.Д.

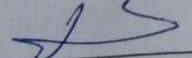
Шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Промышленная разработка программного
обеспечения

Группа 71-ПП

Руководитель  Рыженков Д.В.

Оценка: « 5.0 »

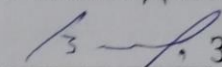
Дата 27 12 2018

Орел 2018

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

УТВЕРЖДАЮ:

 Зав. кафедрой

« 10 » 10 2018 г.

ЗАДАНИЕ
на курсовую работу

по дисциплине «Базы данных»

Студент Шорин В.Д.

Шифр 171406

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Промышленная разработка программного обеспечения

Группа 71-ПП

1 Тема курсовой работы

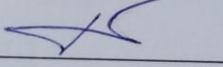
«Разработка и реализация базы данных для онлайн магазина по продаже музыкальных композиций»

2 Срок сдачи студентом законченной работы «17» декабря 2018


3 Исходные данные
В качестве предметной области выбран учет музыкальных композиций в онлайн магазине

4 Содержание курсовой работы
Описание предметной области процесса учета музыкальных композиций в онлайн магазине
Проектирование схемы базы данных по учету музыкальных композиций в онлайн магазине
Реализация базы данных по учету музыкальных композиций в онлайн магазине

5 Отчетный материал курсовой работы
Пояснительная записка курсовой работы; презентация

Руководитель _____  Рыженков Д.В.

Задание принял к исполнению: «8» октября 2018

Подпись студента _____ 

Содержание

Введение.....	4
1 Описание предметной области онлайн магазина по продаже музыкальных композиций	5
2 Проектирование схемы базы данных для онлайн магазина по продаже музыкальных композиций.....	6
2.1 Проектирование логической схемы базы данных для онлайн магазина по продаже музыкальных композиций	6
2.2 Нормализация разработанной логической схемы	9
2.3 Построение физической схемы базы данных для онлайн магазина по продаже музыкальных композиций	15
3 Реализация базы данных для онлайн магазина по продаже музыкальных композиций	18
3.1 Реализация и пробное наполнение базы данных для онлайн магазина по продаже музыкальных композиций	18
3.2 Разработка триггеров к базе данных для онлайн магазина по продаже музыкальных композиций.....	21
3.3 Разработка хранимых процедур для базы данных для онлайн магазина по продаже музыкальных композиций	22
3.4 Построение запросов на извлечение информации из базы данных для онлайн магазина по продаже музыкальных композиций	22
Заключение	24
Список использованных источников	25
Приложение А(обязательное)Скрипты создания	26
Приложение Б(обязательное)Скрипты изменения	28
Приложение В(обязательное)Скрипты наполнения.....	29
Приложение Г(обязательное)Скрипты триггеров	31
Приложение Д(обязательное)Скрипты процедур.....	37
Приложение Е(обязательное)Скрипты запросов.....	38

Введение

Одной из наиболее актуальных задач в сфере IT в настоящее время является разработка баз данных.

Целью курсовой работы является построение базы данных для онлайн магазина по продаже музыкальных композиций.

Задачами курсовой работы являются:

- 1) Изучение предметной области процесса продажи музыкальных композиций в онлайн магазине;
- 2) Разработка модели базы данных для онлайн магазина по продаже музыкальных композиций;
- 3) Выбор системы управления базой данных(СУБД);
- 4) Реализация базы данных для онлайн магазина по продаже музыкальных композиций;
- 5) Построение триггеров к базе данных для онлайн магазина по продаже музыкальных композиций;
- 6) Разработка процедур для базы данных для онлайн магазина по продаже музыкальных композиций;
- 7) Построение запросов на извлечение информации из базы данных для онлайн магазина по продаже музыкальных композиций

1 Описание предметной области онлайн магазина по продаже музыкальных композиций

Онлайн магазин — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или мобильном устройстве, сформировать заказ на покупку, выбрать способ оплаты и получения заказа. В данной работе будет рассмотрен онлайн магазин по продаже музыкальных композиций. Онлайн магазин является своего рода витриной, на которой в качестве товара выставляются музыкальные композиции. Музыкальные композиции делятся по жанру, исполнителю.

Страница определенной музыкальной композиции содержит следующую информацию о ней: название, исполнитель, длительность, жанр, принадлежность к альбому, комментарии пользователей, тип файла, дата выхода, цена, язык текста песни.

Просто просматривать сайт, искать музыкальные композиции и просматривать о них информацию может каждый незарегистрированный пользователь. Однако, если он захочет что-либо скачать, ему придется зарегистрироваться, указав следующие свои данные: никнейм, электронную почту, фамилию, имя, отчество, дату рождения, город проживания и один мобильный телефон, а также придумать пароль.

Только зарегистрированный пользователь может оставлять комментарии к композициям. Комментарий содержит следующую информацию: никнейм пользователя, который оставил комментарий, оценка пользователя, содержание комментария.

Когда зарегистрированный пользователь добавил в заказ все необходимые ему музыкальные композиции, он переходит на страницу подтверждения заказа для оплаты товара. В заказе о каждой музыкальной композиции хранится все ее информация. Как только пользователь оплатил заказ, ему становятся доступны ссылки для скачивания каждой композиции.

2 Проектирование схемы базы данных для онлайн магазина по продаже музыкальных композиций

2.1 Проектирование логической схемы базы данных для онлайн магазина по продаже музыкальных композиций

Изучив предметную область, можно выделить следующие объекты: музыкальная композиция, пользователь, заказ.

На основании выделенных объектов, можно выделить следующие сущности: музыкальная композиция, жанр, исполнитель, пользователь, заказ, комментарий.

Рассмотрим связи между данными сущностями.

Поскольку один исполнитель может создавать много различных музыкальных композиций, но композицию не могут исполнять разные исполнители, то между ними будет неидентифицирующая связь «один ко многим».

Музыкальная композиция может относиться только к одному жанру, но к одному жанру могут относиться множество различных композиций. Поэтому между ними будет неидентифицирующая связь «один ко многим».

Пользователь может делать много заказов, но один заказ может принадлежать только одному пользователю, поэтому между ними будет неидентифицирующая связь «один ко многим».

Один пользователь может оставлять множество комментариев, но один конкретный комментарий может быть оставлен только одним пользователем, поэтому между ними будет идентифицирующая связь «один ко многим».

Музыкальная композиция может быть добавлена во множество различных заказов, также и один заказ может содержать множество различных композиций, поэтому связь «многие ко многим».

К музыкальной композиции может быть оставлено множество комментариев, но один комментарий может принадлежать только к одной композиции, поэтому между ними будет идентифицирующая связь «один ко многим».

Рассмотрим сами сущности и их атрибуты.

Сущность «Музыкальная композиция» содержит следующие атрибуты:

- Код композиции
- Название
- Исполнитель
- Жанр
- Длительность
- Цена
- Альбом
- Язык песни
- Комментарии
- Тип файла
- Дата выхода
- Дата начала действия цены
- Дата окончания действия цены

В качестве первичного ключа будет служить поле «Код композиции».

Сущность «Жанр» содержит в себе атрибуты:

- Код жанра
- Название

В качестве первичного ключа будет служить поле «Код жанра».

Сущность «Исполнитель» содержит в себе следующие атрибуты:

- Код исполнителя
- Название
- Состав

— Профессия

Первичным ключом будет служить поле «Код исполнителя».

Сущность «Пользователь» содержит в себе следующие атрибуты:

— Код пользователя

— Никнейм

— Почта

— Фамилия

— Имя

— Отчество

— Дата рождения

— Мобильный телефон

— Пароль

Первичным ключом данной сущности является «Код пользователя».

Сущность «Заказ» содержит следующие атрибуты:

— Код заказа

— Дата

— Статус заказа

Первичным ключом будет являться поле «Код заказа».

Сущность «Комментарий» содержит следующие атрибуты:

— Код комментария

— Оценка

— Содержание

Первичным ключом является «Код комментария».

На рисунке 1 представлена логическая схема базы данных.

Однако, полученная логическая схема базы данных не соответствует первым трем нормальным формам. Исходя из этого, мы не можем построить физическую схему и перейти к реализации базы данных. Поэтому в следующем пункте будет рассмотрен процесс нормализации базы данных, в

конце которого мы должны получить нормализованную логическую схему базы данных, которая соответствует третьей нормальной форме.

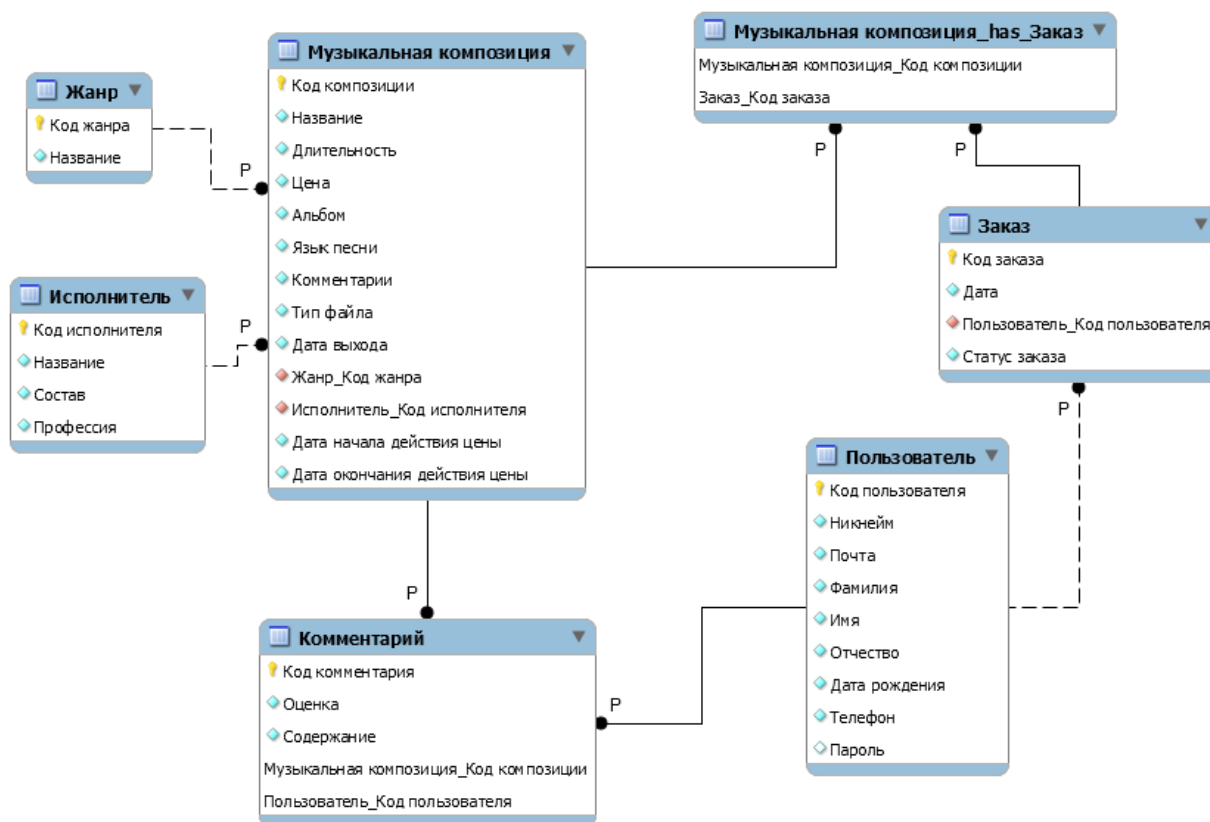


Рисунок 1 – Логическая схема базы данных

2.2 Нормализация разработанной логической схемы

Приведем нашу схему к первой, второй и третьей нормальным формам [5].

Первая нормальная форма (1НФ) говорит, что каждый атрибут отношения должен хранить атомарное значение, каждое отношение (строка в таблице) должно содержать одинаковое количество атрибутов (столбцов), т.е.

— запрещает повторяющиеся столбцы (содержащие одинаковую по смыслу информацию)

- запрещает множественные столбцы (содержащие значения типа списка и т.п.)

- требует определить первичный ключ для таблицы, то есть тот столбец или комбинацию столбцов, которые однозначно определяют каждую строку [1].

Вторая нормальная форма (2НФ) говорит, что отношение находится во второй нормальной форме, если оно находится в 1НФ, и при этом все не ключевые атрибуты зависят только от первичного ключа, т.е.

- Вторая нормальная форма требует, чтобы не ключевые столбцы таблиц зависели от первичного ключа в целом, но не от его части.

- Если таблица находится в первой нормальной форме и первичный ключ у нее состоит из одного столбца, то она автоматически находится и во второй нормальной форме [2].

Отношение находится в **третьей нормальной форме (3НФ)**, если оно находится во второй нормальной форме и каждый не ключевой атрибут зависит только от первичного ключа и не зависят друг от друга [4].

Для того, чтобы привести получившуюся логическую схему к первой нормальной форме, сделаем следующее:

Выделим следующие сущности с соответствующими атрибутами:

«Музыкант»:

- Код музыканта
- Фамилия
- Имя
- Отчество
- Дата рождения

Первичным ключом будет являться поле «Код музыканта».

«Профессия»:

- Код профессии
- Название

Первичным ключом будет являться поле «Код профессии».

«Язык»:

— Код языка

— Название

Первичным ключом будет являться поле «Код языка».

«Альбом»:

— Код альбома

— Название

— Дата выхода

Первичным ключом будет являться поле «Код альбома».

«Исполнитель_has_Музыкант»:

— Исполнитель_Код исполнителя

— Музыкант_Код музыканта

— Профессия_Код профессии

— Дата начала

— Дата конца

«Музыкальная композиция_has_цена»:

— Код цены

— Музыкальная композиция_Код композиции

— Цена

— Дата начала

— Дата конца

— Первичным ключом будет являться поле «Код цены»

Обозначим связи между сущностями:

«Музыкант» - «Исполнитель»: «многие ко многим», поскольку исполнителем может являться группа, в которую входит много музыкантов, и музыкант в разное время может принадлежать к разным группам, поэтому вводится ассоциативная сущность «Исполнитель_has_Музыкант» с

соответствующими атрибутами; идентифицирующая, поскольку первичные ключи будут однозначно определять запись в данной сущности.

«Профессия» - «Исполнитель_has_Музыкант»: «один ко многим», поскольку, если исполнитель является группой, то в ней может быть несколько людей с одинаковыми профессиями; не идентифицирующая, поскольку сущность «Исполнитель_has_Музыкант» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Тип файла»: «один ко многим», поскольку к одному типу файла могут принадлежать разные композиции, но у композиции может быть только один тип файла; не идентифицирующая, поскольку сущность «Музыкальная композиция» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Язык»: «один ко многим», поскольку на одном языке могут быть написаны разные композиции, но одна композиция может быть написана на одном языке; не идентифицирующая, поскольку сущность «Музыкальная композиция» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Исполнитель»: «один ко многим», поскольку одна уникальная композиция может принадлежать только одному исполнителю, но у исполнителя может быть много разных композиций; не идентифицирующая, поскольку сущность «Музыкальная композиция» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Альбом»: «один ко многим», поскольку одна композиция может входить в один альбом, но в альбом входит много разных композиций; не идентифицирующая, поскольку

сущность «Музыкальная композиция» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Жанр»: «один ко многим», поскольку композиция может принадлежать только к одному жанру, но к одному жанру могут относиться разные композиции; не идентифицирующая, поскольку сущность «Музыкальная композиция» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Цена»: «один ко многим», поскольку композиции может быть только одна цена в данный момент времени, но одна цена может быть у разных композиций; не идентифицирующая, поскольку сущность «Цена» уже имеет первичный ключ и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Комментарий»: «один ко многим», поскольку у одной композиции может быть много комментариев, но один комментарий принадлежит только к одной композиции; не идентифицирующая, поскольку сущность «Комментарий» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Музыкальная композиция» - «Заказ»: «многие ко многим», поскольку одна композиция может принадлежать разным заказам, и в один заказ могут входить разные композиции, поэтому вводится дополнительная сущность «Заказ_has_Музыкальная композиция», в которой будут храниться первичные ключи данных сущностей; идентифицирующая, поскольку внешние ключи данных двух сущностей образуют первичный составной ключ в сущности «Заказ_has_Музыкальная композиция» и позволяют однозначно определить в ней запись.

«Пользователь» - «Заказ»: «один ко многим», поскольку у одного пользователя может быть много разных заказов, но один заказ может принадлежать только одному пользователю; не идентифицирующая,

поскольку сущность «Заказ» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

«Пользователь» - «Комментарий»: «один ко многим», поскольку пользователь может оставлять много комментариев, но один комментарий принадлежит только одному пользователю; не идентифицирующая, поскольку сущность «Комментарий» уже имеет первичный ключ, и ввод еще одного ключа может привести к функциональной зависимости.

Проанализировав полученную логическую схему, можно сделать вывод, что она находится во второй нормальной форме (так как она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо зависит от первичного ключа).

Продолжив изучать полученную схему, можно заметить, что она также находится и в третьей нормальной форме, поскольку она находится во второй нормальной форме и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа.

Опираясь на совокупность всех вышеупомянутых и перечисленных фактов, получим нормализованную логическую схему, изображенную на рисунке 2.

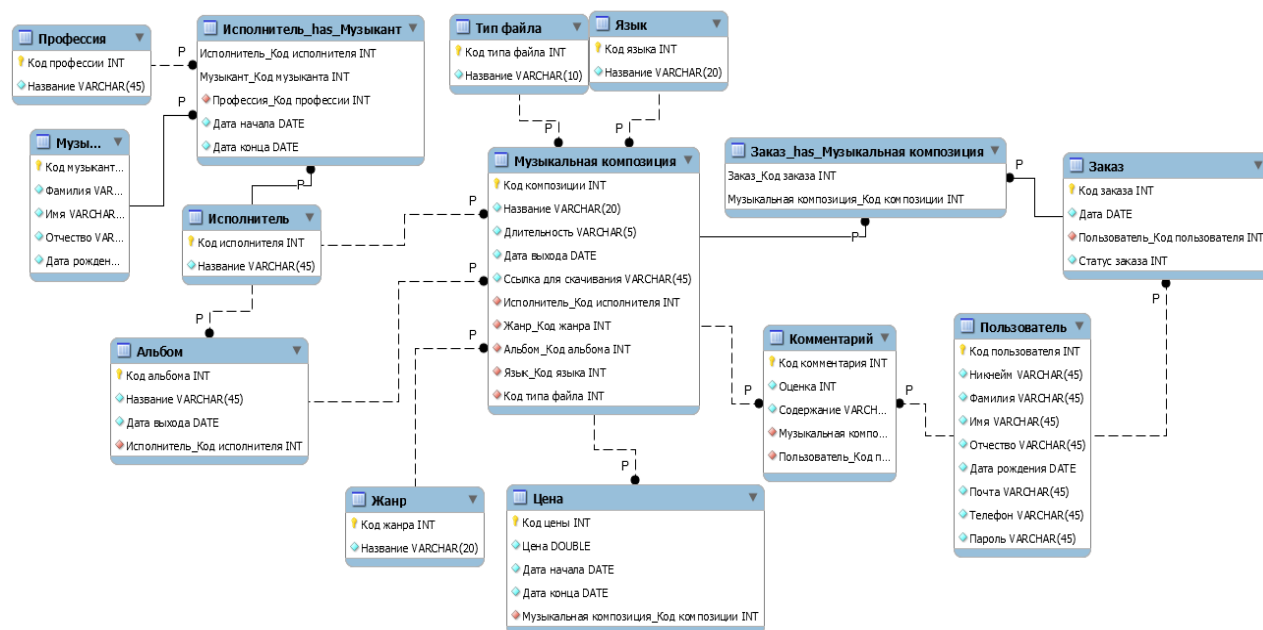


Рисунок 2 – Нормализованная логическая схема базы данных

2.3 Построение физической схемы базы данных для онлайн магазина по продаже музыкальных композиций

MySQL обладает большим количеством преимуществ перед другими системами:

1. СУБД MySQL является одной из самых быстрых баз данных среди имеющихся на современном рынке.

2. СУБД MySQL является высокопроизводительной и относительно простой в использовании СУБД, которую значительно проще установить и администрировать, чем многие другие большие системы.

3. СУБД MySQL распространяется бесплатно для домашнего использования.

4. MySQL понимает команды SQL, поддерживает интерфейс ODBC, протокол интерфейса с базами данных, разработанный компанией Microsoft.

5. MySQL отлично работает как под управлением разных версий UNIX, так и под управлением других систем: Windows и OS/2. При этом система работает как на мощных серверах, так и на домашних ПК.

Если сравнивать MySQL с, к примеру, PostgreSQL, то, не смотря на все плюсы второй, первая работает быстрее при простых операциях чтения [3].

В контексте выбранной СУБД физическая схема нашей базы данных будет выглядеть следующим образом (рисунок 3).

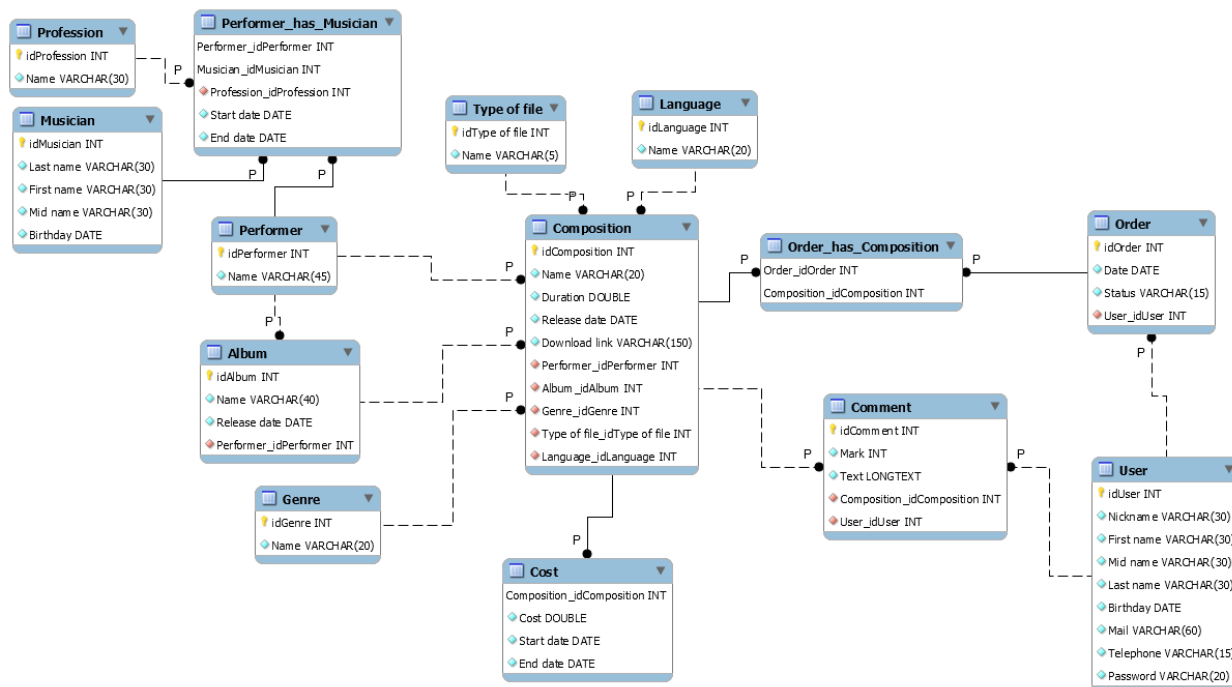


Рисунок 3 – физическая схема базы данных

Целостность базы данных – соответствие имеющейся в базе данных информации предметной области.

В реляционной модели данных определяют две базовые категории обеспечения целостности:

- Целостность ссылок
- Целостность сущностей

Целостность ссылок — необходимое качество реляционной базы данных, заключающееся в отсутствии в любом её отношении внешних ключей, ссылающихся на несуществующие кортежи.

Целостность сущностей — каждый кортеж любого отношения должен отличаться от любого другого кортежа этого отношения (т.е. любое отношение должно обладать первичным ключом).

Учитывая то, что наша схема удовлетворяет первым трем нормальным формам, можно сделать вывод, что целостность данных соблюдена.

Бизнес-правила определяют реакцию системы на добавление, изменение или удаление данных, обеспечивая непротиворечивость и

ссылочную целостность БД. Многие бизнес-правила реализуются с помощью триггеров. В нашей базе данных необходимо будет реализовать следующие бизнес-правила:

- Дата окончания действия цены должны быть позже даты начала, либо NULL
- Оценка композиции должна быть в диапазоне 1 – 5.

3 Реализация базы данных для онлайн магазина по продаже музыкальных композиций

3.1 Реализация и пробное наполнение базы данных для онлайн магазина по продаже музыкальных композиций

Для реализации базы данных будут использованы следующие языки:

DDL - (Data Defenition Language) предложения для определения структуры базы данных или схемы.

DML - (Data Manipulation Language) предложения для управления данными.

К DDL относятся такие команды, как:

- CREATE - создает объекты базы данных (таблицы, представления и т.д.)
- ALTER - Изменяет структуру и объекты базы данных
- DROP - Удаляет объекты базы данных
- TRUNCATE - Удаляет все записи из таблицы
- COMMENT - Добавляет комментарии в словарь данных
- RENAME - Переименовывает объект (alter table <old_name> rename to <new_name>)

К DML относятся такие команды, как:

- SELECT - Возвращает данные из базы данных
- INSERT - Вставляет данные в таблицу
- UPDATE - Обновляет существующие данные в таблице
- DELETE - Удаляет все записи в таблице
- MERGE - UPSERT операция (insert или update)
- CALL - вызов подпрограммы PL/SQL или Java
- EXPLAIN PLAN - Предоставляет план запроса
- LOCK TABLE - Управление параллелизмом

Рассмотрим некоторые скрипты реализации базы данных для онлайн магазина по продаже музыкальных композиций.

Ниже представлены скрипты для создания таблиц «Composition» и «User». Скрипты по созданию остальных таблиц представлены в Приложении А.

```
create table if not exists kyrsmusicshop.Composition (
    idComposition int not null auto_increment,
    Name VARCHAR(20) not null,
    Duration double not null,
    ReleaseDate DATE,
    DownLoadLink varchar(150),
    primary key (idComposition)
)
```

```
create table if not exists kyrsmusicshop.user(
    idUser int not null auto_increment,
    Nickname VARCHAR(30) not null unique,
    LastName varchar(30) not null,
    FirstName varchar(30) not null,
    MidName varchar(30) not null,
    Birhday DATE not null,
    Mail varchar(60) not null,
    Telephone varchar(15) not null,
    Password varchar(20) not null,
    primary key (idUser)
)
```

Для формирования внешних ключей в таблицах будем использовать команду ALTER TABLE. Ниже приведены скрипты по созданию внешних ключей для таблиц «Composition» и «Album». Скрипты по изменению остальных таблиц представлены в Приложении Б.

```

alter table kyrsmusicshop.composition
    add column Performer_idPerformer int not null,
    add column Album_idAlbum int not null,
    add column Genre_idGenre int not null,
    add column TypeOfFile_idTypeOfFile int not null,
    add column Language_idLanguage int not null,
    add foreign key (Performer_idPerformer) references
kyrsmusicshop.performer(idPerformer),
    add foreign key (Album_idAlbum) references
kyrsmusicshop.album(idAlbum),
    add foreign key (Genre_idGenre) references
kyrsmusicshop.genre(idGenre),
    add foreign key (TypeOfFile_idTypeOfFile) references
kyrsmusicshop.typeoffile(idTypeOfFile),
    add foreign key (Language_idLanguage) references
kyrsmusicshop.language(idLanguage)
alter table kyrsmusicshop.album
    add column Performer_idPerformer int not null,
    add foreign key (Performer_idPerformer)
references kyrsmusicshop.performer(idPerformer)

```

Для пробного наполнения базы данных используем команду INSERT INTO. Ниже представлены скрипты для наполнения таблиц «Performer» и «Genre». Остальные скрипты по наполнению таблиц представлены в Приложении В.

```

insert into performer (Name)
    values ('Imagine dragons'),
    ('Twenty One Pilots'),
    ('Queen'),
    ('Король и шут')
insert into genre (Name)

```

```

values ('Инди-рок'),
      ('Альтернативный рок'),
      ('Поп-рок'),
      ('Электро-поп'),
      ('Хард-рок'),
      ('Панк-рок')

```

После реализации и пробного наполнения базы данных для онлайн магазина по продаже музыкальных композиций можно приступить к разработке триггеров и хранимых процедур.

3.2 Разработка триггеров к базе данных для онлайн магазина по продаже музыкальных композиций

Триггер - это код SQL, который запускается непосредственно перед (BEFORE) или сразу (AFTER) после того, как событие INSERT, UPDATE или DELETE происходит в конкретной таблице базы данных.

Триггер представляет собой специальный тип хранимых процедур, запускаемых сервером автоматически при попытке изменения данных в таблице, с которой этот триггер связан.

Ниже представлен пример триггера, который реализует одно из бизнес-правил, названных выше, а именно проверку оценки композиции на значения от 1 до 5. Остальные скрипты триггеров представлены в приложении Г.

```

create trigger InsertValueDateCost
before INSERT
on comment
for each row
begin
  if(NEW.Mark > 0 and NEW.Mark < 6)then

```



```

        set @Mark = NEW.Mark;
    else
        signal sqlstate '45000' set message_text = 'Field "Mark" should be more
than 0 and less than 6!';
    end if;
end;

```

3.3 Разработка хранимых процедур для базы данных для онлайн магазина по продаже музыкальных композиций

Хранимая процедура - это способ инкапсуляции повторяющихся действий. В хранимых процедурах можно объявлять переменные, управлять потоками данных, а также применять другие техники программирования. Причина их создания ясна и подтверждается частым использованием.

Ниже представлена хранимая процедура с одним входным параметром, которая принимает название композиции и выводит альбом, к которому она принадлежит. Остальные процедуры представлены в приложении Д.

```

create procedure getAlbumOfComposition(IN inName varchar(40))
begin
    select album.Name from album
        inner join composition c on album.idAlbum = c.Album_idAlbum
        where lower(c.Name) = lower(inName);
end;

```

3.4 Построение запросов на извлечение информации из базы данных для онлайн магазина по продаже музыкальных композиций

При работе с таблицами можно в любой момент выбрать из базы данных необходимую информацию с помощью запросов. Запрос - это

обращение к БД для поиска или изменения в базе данных информации, соответствующей заданным критериям.

В языке SQL запрос представлен командой SELECT, которая может выводить значения из всех таблиц, нескольких выборочных или только одной.

К примеру, нам необходимо вывести на экран информацию обо всех песнях, входящих в какой-то конкретный альбом (в данном случае, название альбома «Origins»). Данные, полученные в результате работы данного скрипта, представлены на рисунке 4. Скрипты остальных запросов к базе данных представлены в Приложении Е.

```
select c.Name, Duration, c.ReleaseDate, c.DownLoadLink from composition c
inner join album a on c.Album_idAlbum = a.idAlbum
where lower(a.Name) = lower('Origins')
```

	Name	Duration	ReleaseDate	DownLoadLink
1	Natural	3.09	2018-07-17	http://zaycev.net/pages/75297/7529763.shtml
2	Digital	3.21	2018-11-09	http://zaycev.net/pages/85202/8520207.shtml
3	Bullet in the gun	3.24	2018-11-09	http://zaycev.net/pages/85202/8520206.shtml
4	Machine	3.01	2018-10-31	http://zaycev.net/pages/85107/8510784.shtml

Рисунок 4 – Результат запроса к базе данных

Заключение

В результате выполнения курсовой работы была достигнута поставленная цель и выполнены все поставленные задачи:

- изучен процесс продажи музыкальных композиций в онлайн магазине
- разработана модель базы данных, выбрана СУБД
- реализована база данных для онлайн магазина по продаже музыкальных композиций
 - построены триггеры
 - разработаны хранимые процедуры
 - построены запросы.

Список использованных источников

1. Дейт К. Дж. Введение в системы баз данных [Текст] / Дж. К. Дейт — 8-е изд. — М.: Вильямс, 2005. — 1328 с.
2. Коннолли Т., Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. — 3-е изд. [Текст] / Т. Коннолли, К. Бегг. — М.: Вильямс, 2003. — 1436 с.
3. Кузнецов Максим, Симдянов Игорь. MySQL на примерах. [Текст] / Максим Кузнецов, Игорь Симдянов — Спб.: «БХВ-Петербург», 2008. — С. 952.
4. Кузнецов С. Д. Основы баз данных. — 2-е изд. — [Текст] / С.Д. Кузнецов.- М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с.
5. Учебно-методический комплекс по дисциплине «Базы данных. Системы управления базами данных» - StudFiles.net [Электронный ресурс].- Режим доступа: <https://studfiles.net/preview/5150865/page:14/>, свободный

Приложение А

(обязательное)

Скрипты создания

```

create table if not exists kyrsmusicshop.TypeOfFile (
  idTypeOfFile int not null auto_increment, Name varchar(5) not null ,
  primary key (idTypeOfFile)
)
create table if not exists kyrsmusicshop.Language(
  idLanguage int not null auto_increment, Name VARCHAR(20),
  primary key (idLanguage)
)
create table if not exists kyrsmusicshop.Performer(
  idPerformer int not null auto_increment, Name varchar(45) not null,
  primary key (idPerformer)
)
create table if not exists kyrsmusicshop.Musician(
  idMusician int not null auto_increment,
  LastName varchar(30) not null,
  FirstName varchar(30) not null,
  MidName varchar(30) not null,
  Birhday DATE not null,
  primary key (idMusician)
)
create table if not exists kyrsmusicshop.profession(
  idProfession int not null auto_increment,
  Name varchar(30) not null,
  primary key (idProfession)
)
create table if not exists kyrsmusicshop.album(
  idAlbum int not null auto_increment,
  Name varchar(40) not null,
  ReleaseDate DATE,
  primary key (idAlbum)
)
create table if not exists kyrsmusicshop.genre(
  idGenre int not null auto_increment,
  Name varchar(20)not null,
  primary key (idGenre)
)
create table if not exists kyrsmusicshop.cost(
  Cost double not null,
  StartDate DATE not null,
  EndDate DATE not null
)
create table if not exists kyrsmusicshop.order(
  idOrder int not null auto_increment,
  Date date not null,
  Status VARCHAR(15) not null,
  primary key (idOrder))
create table if not exists kyrsmusicshop.order_has_composition(
  Order_idOrder int not null,
  Composition_idComposition int not null
)

```

```
create table if not exists kyrsmusicshop.comment(  
  idComment int not null auto_increment,  
  Mark int not null,  
  Text text not null,  
  primary key (idComment)  
)  
create table if not exists kyrsmusicshop.performer_has_musician(  
  StartDate Date not null,  
  EndDate DATE null  
)
```

Приложение Б

(обязательное)

Скрипты изменения

```

alter table kyrsmusicshop.performer_has_musician
  add column Performer_idPerformer int not null,
  add column Musician_idMusician int not null,
  add column Profession_ifProfession int not null,
  add foreign key (Performer_idPerformer) references kyrsmusicshop.performer(idPerformer),
  add foreign key (Musician_idMusician) references kyrsmusicshop.musician(idMusician),
  add foreign key (Profession_ifProfession) references kyrsmusicshop.profession(idProfession),
  add primary key (Performer_idPerformer,Musician_idMusician)
alter table kyrsmusicshop.cost
  add column Composition_idComposition int not null,
  add foreign key (Composition_idComposition)
  references kyrsmusicshop.composition(idComposition),
  add primary key (Composition_idComposition)
alter table kyrsmusicshop.order_has_composition
  add foreign key (Order_idOrder) references kyrsmusicshop.`order`(idOrder),
  add          foreign          key          (Composition_idComposition)          references
kyrsmusicshop.composition(idComposition),
  add primary key (Order_idOrder, Composition_idComposition)
alter table kyrsmusicshop.comment
  add column Composition_idComposition int not null,
  add column User_idUser int not null,
  add foreign key (Composition_idComposition) references kyrsmusicshop.composition(idComposition),
  add foreign key (User_idUser) references kyrsmusicshop.user(idUser)
alter table kyrsmusicshop.`order`
  add column User_idUser int not null,
  add FOREIGN KEY (User_idUser) references kyrsmusicshop.user(idUser)

```


Приложение В

(обязательное)

Скрипты наполнения

```

insert into kyrsmusicshop.profession(Name)
  values ('Рэпер'), ('Гитарист'), ('Барабанщик'), ('Певец'), ('Клавишник'), ('Бас-гитарист')
insert into kyrsmusicshop.musician (LastName, FirstName, MidName, Birthday)
  values ('Рейнольдс', 'Дэниел', 'Колтер', '1987-07-14'),
    ('Сермон', 'Дэниел', 'Уэйн', '1984-06-15'),
    ('Платцман', 'Дэниэл', 'Джеймс', '1986-09-28'),
    ('Артур', 'Макки', 'Бенджамин', '1985-04-7'),
    ('Джозеф', 'Тайлер', 'Роберт', '1988-12-1'),
    ('Дан', 'Джошуа', 'Уильям', '1988-06-18'),
    ('Меркьюри', 'Фредди', 'Фаррух', '1946-09-05'),
    ('Дикон', 'Джон', 'Ричард', '1951-08-19'),
    ('Мэй', 'Брайан', 'Гарольд', '1947-07-19'),
    ('Тейлор', 'Роджер', 'Мэдоувс', '1949-07-26'),
    ('Горшенёв', 'Михаил', 'Юрьевич', '1973-08-07'),
    ('Князев', 'Андрей', 'Сергеевич', '1973-02-06'),
    ('Балунов', 'Александр', 'Валентинович', '1973-03-19'),
    ('Щиголев', 'Александр', 'Анатольевич', '1973-04-08')
insert into genre (Name)
  values ('Инди-рок'), ('Альтернативный рок'), ('Поп-рок'), ('Электро-поп'), ('Хард-рок'), ('Панк-рок')
insert into performer_has_musician (StartDate, EndDate, Performer_idPerformer, Musician_idMusician,
  Profession_idProfession)
  values ('2008-07-10', NULL, '1', '3', '4'), ('2008-07-10', NULL, '1', '4', '2'),
    ('2011-09-15', NULL, '1', '5', '3'), ('2009-06-21', NULL, '1', '6', '6'),
    ('2009-03-05', NULL, '2', '7', '4'), ('2011-08-07', NULL, '2', '8', '3'),
    ('1970-04-19', NULL, '3', '9', '4'), ('1971-02-11', NULL, '3', '10', '6'),
    ('1970-04-19', NULL, '3', '11', '5'), ('1970-04-19', NULL, '3', '12', '3'),
    ('1988-05-23', '2013-07-13', '4', '13', '4'), ('1990-01-15', '2011-12-16', '4', '14', '4'),
    ('1988-05-23', NULL, '4', '15', '6'), ('1988-05-23', NULL, '4', '16', '3')
insert into typeoffile (Name)
  values ('.mp3'), ('.mp4');
insert into language (Name)
  values ('English'), ('Русский')
insert into album (Name, ReleaseDate, Performer_idPerformer)
  VALUES ('Origins', '2018-11-09', '1'), ('Evolve', '2017-06-23', '1'),
    ('Blurryface', '2015-05-19', '2'), ('Trench', '2018-10-05', '2'),
    ('A Night at the Opera', '1975-12-02', '3'), ('The Game', '1980-06-30', '3'),
    ('Жаль, нет ружья', '2002-10-10', '4'), ('Герои и Злодеи', '2000-04-13', '4')
insert into user (Nickname, LastName, FirstName, MidName, Birthday, Mail, Telephone, Password)
  values ('User1', 'Шорин', 'Владислав', 'Дмитриевич', '2000-05-27', 'kniger33@gmail.com', '89961618691',
    '123456'),
    ('User2', 'Иванов', 'Петр', 'Геннадьевич', '1998-06-16', 'gena123@yandex.ru', '88005553535', 'asdqwe'),
    ('User3', 'Сидоров', 'Николай', 'Анатольевич', '1995-08-31', 'sidr152@mail.ru', '89516286485', 'qwerty'),
    ('User4', 'Кузнецов', 'Андрей', 'Валерьевич', '1999-02-11', 'kyzya666@mail.ru', '83692581479', 'zxcvbn')
insert into composition (Name, Duration, ReleaseDate, DownloadLink, Performer_idPerformer, Album_idAlbum,
  Genre_idGenre, TypeOfFile_idTypeOfFile, Language_idLanguage)
  VALUES ('Natural', '3.09', '2018-07-17', 'http://zaycev.net/pages/75297/7529763.shtml', '1', '1', '1', '1', '1'),
    ('Digital', '3.21', '2018-11-09', 'http://zaycev.net/pages/85202/8520207.shtml', '1', '1', '6', '1', '1'),
    ('Bullet in the gun', '3.24', '2018-11-09', 'http://zaycev.net/pages/85202/8520206.shtml', '1', '1', '2', '1', '1'),
    ('Machine', '3.01', '2018-10-31', 'http://zaycev.net/pages/85107/8510784.shtml', '1', '1', '4', '1', '1'),
    ('Whatever it takes', '3.21', '2017-05-09', 'http://zaycev.net/pages/45603/4560393.shtml', '1', '2', '4', '1', '1'),
    ('Thunder', '3.07', '2017-04-27', 'http://zaycev.net/pages/67571/6757194.shtml', '1', '2', '1', '1', '1'),
    ('Believer', '3.24', '2017-02-01', 'http://zaycev.net/pages/56230/5623021.shtml', '1', '2', '2', '1', '1'),
    ('Stressed Out', '3.22', '2015-11-10', 'http://megapesni.me/rock_pesni/52435-twenty-one-pilots-stressed-
out.html', '2', '3', '4', '2', '1'),
    ('Ride', '3.34', '2016-08-28', 'http://megapesni.me/popsa/53114-twenty-one-pilots-ride.html', '2', '3', '1', '2', '1'),

```

```

('Heavydirtysoul', '3.54', '2016-12-09', 'https://muzlo.me/song/47828698', '2', '3', '3', '2', '1'),
('Jumpsuit', '3.58', '2018-07-11', 'https://muzlo.me/song/57296550', '2', '4', '2', '2', '1'),
('Morph', '4.18', '2018-10-05', 'https://muzlo.me/song/59446257', '2', '4', '6', '2', '1'),
('Nico and The Niners', '3.47', '2018-07-11', 'https://muzlo.me/song/57296552', '2', '4', '1', '1', '1'),
('Bohemian Rhapsody', '5.55', '1975-10-31', 'http://luxmp3.net/music_track/queen-bohemian-rhapsody', '3', '5',
'5', '1', '1'),
('I'm in Love with My Car', '03.04', '1975-11-21', 'http://zaycev.net/pages/168/16820.shtml', '3', '5', '5', '1', '1'),
('39', '03.30', '1976-05-18', 'http://zaycev.net/pages/168/16821.shtml', '3', '5', '1', '1', '1'),
('Another One Bites the Dust', '3.32', '1980-08-22', 'https://muzlo.me/song/47828496', '3', '6', '5', '2', '1'),
('Play the Game', '3.30', '1980-05-30', 'https://muzlo.me/song/47828508', '3', '6', '5', '2', '1'),
('Save Me', '3.43', '1980-01-25', 'https://muzlo.me/song/47828503', '3', '6', '5', '1', '1'),
('Мёртвый Анархист', '4.07', '2002-10-10', 'http://zaycev.net/pages/7927/792736.shtml', '4', '7', '5', '2', '2'),
('Волосокрад', '4.27', '2002-10-10', 'http://zaycev.net/pages/15261/1526194.shtml', '4', '7', '3', '2', '2'),
('Смешной совет', '4.05', '2002-10-10', 'http://zaycev.net/pages/12609/1260994.shtml', '4', '7', '5', '1', '2'),
('Дед на свадьбе', '4.46', '2000-04-13', 'http://zaycev.net/pages/15260/1526051.shtml', '4', '8', '3', '2', '2'),
('Запрет отца', '3.31', '2000-04-13', 'http://zaycev.net/pages/16938/1693884.shtml', '4', '8', '5', '1', '2'),
('Кузнец', '2.59', '2000-04-13', 'http://zaycev.net/pages/16938/1693865.shtml', '4', '8', '5', '1', '2')
insert into cost (Cost, StartDate, EndDate, Composition_idComposition)
VALUES ('20.90', '2018-10-01', NULL, '1'), ('21.90', '2018-11-02', NULL, '2'),
('22.90', '2018-12-03', NULL, '3'), ('23.90', '2018-10-04', NULL, '4'),
('24.90', '2018-11-05', NULL, '5'), ('23.90', '2018-12-06', NULL, '6'),
('22.90', '2018-10-07', NULL, '7'), ('21.90', '2018-11-08', NULL, '8'),
('20.90', '2018-12-09', NULL, '9'), ('10.90', '2018-10-10', NULL, '10'),
('11.90', '2018-11-11', NULL, '11'), ('12.90', '2018-12-12', NULL, '12'),
('13.90', '2018-10-13', NULL, '13'), ('14.90', '2018-11-14', NULL, '14'),
('15.90', '2018-12-15', NULL, '15'), ('16.90', '2018-10-16', NULL, '16'),
('15.90', '2018-11-17', NULL, '17'), ('14.90', '2018-12-18', NULL, '18'),
('13.90', '2018-10-19', NULL, '19'), ('12.90', '2018-11-20', NULL, '20'),
('11.90', '2018-12-21', NULL, '21'), ('10.90', '2018-10-22', NULL, '22'),
('18.90', '2018-11-23', NULL, '23'), ('19.90', '2018-12-24', NULL, '24'),
('17.90', '2018-10-25', NULL, '25')
insert into comment (Mark, Text, Composition_idComposition, User_idUser)
VALUES ('5', 'Супер!', '1', '1'), ('4', 'Классно!', '2', '2'),
('3', 'Неплохо', '9', '3'), ('4', 'Замечательно', '4', '4'),
('5', 'Отличная песня!!!', '5', '4'), ('2', 'Такое себе', '16', '3'),
('1', 'Вообще ни о чем', '21', '2'), ('5', 'Шедевр!!!', '14', '1'),
('5', 'Лучше не слышал ничего!!!', '7', '2')
insert into `order` (Date, Status, User_idUser)
VALUES ('2018-12-09', 'Оплачен', '1'), ('2018-12-08', 'Не оплачен', '2'),
('2018-12-07', 'Оплачен', '3'), ('2018-12-06', 'Не оплачен', '4'),
('2018-12-05', 'Оплачен', '3'), ('2018-12-04', 'Оплачен', '2'),
('2018-12-03', 'Оплачен', '1')
insert into order_has_composition (Order_idOrder, Composition_idComposition)
values ('1', '1'), ('2', '2'), ('3', '3'), ('4', '4'), ('5', '5'), ('6', '6'), ('7', '7'), ('1', '8'), ('1', '9'),
('2', '10'),
('3', '11'), ('4', '12'), ('5', '13'), ('6', '14'), ('7', '15'), ('2', '16'), ('1', '17'), ('2', '18'), ('3', '19'), ('4',
'20'), ('5', '21'), ('6', '22'), ('7', '23'), ('3', '24'), ('1', '8'), ('2', '7'), ('3', '6'), ('4', '5'), ('5',
'4'),
('6', '3'), ('7', '2'), ('4', '1')

```

Приложение Г

(обязательное)

Скрипты триггеров

```

create trigger InsertValidAlbumName
before INSERT
on album
for each row
begin
  if NEW.Name regexp '^[a-zA-Za-яA-Я]+$' != " then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Name isn\'t valid';
  end if;
end;
create trigger UpdateValidAlbumName
before UPDATE
on album
for each row
begin
  if NEW.Name regexp '^[a-zA-Za-яA-Я]+$' != " then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Name isn\'t valid';
  end if;
end;
create trigger InsertValueDateCost
before INSERT
on cost
for each row
begin
  if(NEW.Cost > 0 and NEW.Mark < 6)then
    set @Mark = NEW.Mark;
  else
    signal sqlstate '45000' set message_text = 'Field "Mark" should be more than 0 and less than 6!';
  end if;
end;
create trigger InsertValueOfMark
before insert
on comment
for each row
begin
  if(NEW.Mark > 0 and NEW.Mark < 6)then
    set @Mark = NEW.Mark;
  else
    signal sqlstate '45000' set message_text = 'Field "Mark" should be more than 0 and less than 6!';
  end if;
end;
create trigger InsertValidCompositionDuration
before INSERT
on composition
for each row
begin
  if NEW.Duration > 0 then
    set @Duration = NEW.Duration;
  else
    signal sqlstate '45000' set message_text = "Duration cannot be negative!";
  end if;
end;

```

```

create trigger InsertValidCompositionName
before INSERT
on composition
for each row
begin
    if NEW.Name regexp '^[a-zA-Za-яA-Я]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;
create trigger UpdateValidCompositionDuration
before UPDATE
on composition
for each row
begin
    if NEW.Duration > 0 then
        set @Duration = NEW.Duration;
    else
        signal sqlstate '45000' set message_text = "Duration cannot be negative!";
    end if;
end;
create trigger UpdateValidCompositionName
before UPDATE
on composition
for each row
begin
    if NEW.Name regexp '^[a-zA-Za-яA-Я]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;
create trigger InsertValidCostDate
before INSERT
on cost
for each row
begin
    if NEW.EndDate != null then
        if datediff(New.EndDate, NEW.StartDate) >= 0 then
            set @StartDate = NEW.StartDate;
            set @EndDate = NEW.EndDate;
        else
            signal sqlstate '45000' set message_text = 'Value isn\'t valid!';
        end if;
    else
        set @StartDate = NEW.StartDate;
    end if;
end;
create trigger UpdateValidCostDate
before UPDATE
on cost
for each row
begin
    if NEW.EndDate is null then
        set @EndDate = NEW.EndDate;
    else
        if datediff(New.EndDate, NEW.StartDate) >= 0
        or datediff(New.EndDate, OLD.StartDate) >= 0 then
            set @StartDate = NEW.StartDate;
            set @EndDate = NEW.EndDate;
        else

```

```

        signal sqlstate '45000' set message_text = 'Value isn\'t valid!';
    end if;
end if;
end;
create trigger InsertValidGenre
before INSERT
on genre
for each row
begin
    if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Name isn\'t valid';
    end if;
end;
create trigger UpdateValidGenre
before UPDATE
on genre
for each row
begin
    if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Name isn\'t valid';
    end if;
end;
create trigger InsertValidLanguage
before INSERT
on language
for each row
begin
    if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Name isn\'t valid';
    end if;
end;
create trigger UpdateValidLanguage
before UPDATE
on language
for each row
begin
    if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != " then
        set @Name = NEW.Name;
    else
        signal sqlstate '45000' set message_text = 'Name isn\'t valid';
    end if;
end;
create trigger InsertValidMusicianName
before INSERT
on musician
for each row
begin
    if NEW.LastName regexp '[a-zA-Za-яA-Я-]+$' != "
    and NEW.FirstName regexp '[a-zA-Za-яA-Я-]+$' != "
    and NEW.MidName regexp '[a-zA-Za-яA-Я-]+$' != " then
        set @LastName = NEW.LastName;
        set @FirstName = NEW.FirstName;
        set @MidName = NEW.MidName;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;

```

```

end;
create trigger UpdateValidMusicianName
before UPDATE
on musician
for each row
begin
  if NEW.LastName regexp '[a-zA-Za-яA-Я]+' != "
    and NEW.FirstName regexp '[a-zA-Za-яA-Я]+' != "
    and NEW.MidName regexp '[a-zA-Za-яA-Я]+' != " then
    set @LastName = NEW.LastName;
    set @FirstName = NEW.FirstName;
    set @MidName = NEW.MidName;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid';
  end if;
end;
create trigger InsertValidOrderStatus
before INSERT
on `order`
for each row
begin
  if NEW.Status regexp '[a-zA-Za-яA-Я]+' != " then
    set @Status = NEW.Status;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid';
  end if;
end;
create trigger UpdateValidOrderStatus
before UPDATE
on `order`
for each row
begin
  if NEW.Status regexp '[a-zA-Za-яA-Я]+' != " then
    set @Status = NEW.Status;
  else
    signal sqlstate '45000' set message_text = 'Status isn\'t valid';
  end if;
end;
create trigger InsertValidPerformerName
before insert
on performer
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я]+' != " then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid';
  end if;
end;
create trigger UpdateValidPerformerName
before update
on performer
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я]+' != " then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid';
  end if;
end;
create trigger InsertValidStartDate
before INSERT

```

```

on performer_has_musician
for each row
begin
  if datediff(New.EndDate, NEW.StartDate) >= 0 then
    set @StartDate = NEW.StartDate;
    set @EndDate = NEW.EndDate;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid!';
  end if;
end;
create trigger UpdateValidStartDate
before UPDATE
on performer_has_musician
for each row
begin
  if datediff(New.EndDate, NEW.StartDate) >= 0 then
    set @StartDate = NEW.StartDate;
    set @EndDate = NEW.EndDate;
  else
    signal sqlstate '45000' set message_text = 'Value isn\'t valid!';
  end if;
end;
create trigger InsertValidProfession
before INSERT
on profession
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != '' then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Name isn\'t valid';
  end if;
end;
create trigger UpdateValidProfession
before UPDATE
on profession
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != '' then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Name isn\'t valid';
  end if;
end;
create trigger InsertValidTypeOfFile
before INSERT
on typeoffile
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != '' then
    set @Name = NEW.Name;
  else
    signal sqlstate '45000' set message_text = 'Name isn\'t valid';
  end if;
end;
create trigger UpdateValidTypeOfFile
before UPDATE
on typeoffile
for each row
begin
  if NEW.Name regexp '[a-zA-Za-яA-Я-]+$' != '' then
    set @Name = NEW.Name;

```



```

        else
            signal sqlstate '45000' set message_text = 'Name isn\'t valid';
        end if;
    end;
create trigger InsertValidUserName
before INSERT
on user
for each row
begin
    if NEW.LastName regexp '[a-zA-Za-яA-Я]+' != "
        and NEW.FirstName regexp '[a-zA-Za-яA-Я]+' != "
        and NEW.MidName regexp '[a-zA-Za-яA-Я]+' != " then
        set @LastName = NEW.LastName;
        set @FirstName = NEW.FirstName;
        set @MidName = NEW.MidName;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;
create trigger InsertValidUserTelephone
before INSERT
on user
for each row
begin
    if NEW.Telephone regexp '[0-9]+' then
        set @Telephone = NEW.Telephone;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;
create trigger UpdateValidUserName
before UPDATE
on user
for each row
begin
    if NEW.LastName regexp '[a-zA-Za-яA-Я]+' != "
        and NEW.FirstName regexp '[a-zA-Za-яA-Я]+' != "
        and NEW.MidName regexp '[a-zA-Za-яA-Я]+' != " then
        set @LastName = NEW.LastName;
        set @FirstName = NEW.FirstName;
        set @MidName = NEW.MidName;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;
create trigger UpdateValidUserTelephone
before UPDATE
on user
for each row
begin
    if NEW.Telephone regexp '[0-9]+' then
        set @Telephone = NEW.Telephone;
    else
        signal sqlstate '45000' set message_text = 'Value isn\'t valid';
    end if;
end;

```

Приложение Д

(обязательное)

Скрипты процедур

```

create procedure deleteOldCosts()
begin
  declare dateNow date default now();
  declare finished int default 0;
  declare startDate date;
  declare id int;
  declare costCursor cursor for select idCost, StartDate from cost;
  declare continue handler for not found set finished = 1;
  open costCursor;
  getEndDate: loop
    fetch costCursor into id, startDate;
    if datediff(dateNow, startDate) >= 3500 then
      delete from cost where idCost = id;
    end if;
    if finished = 1 then
      leave getEndDate;
    end if;
  end loop;
  close costCursor;
end;

create procedure getAlbumOfComposition(IN inName varchar(40))
begin
  select album.Name from album
  inner join composition c on album.idAlbum = c.Album_idAlbum
  where lower(c.Name) = lower(inName);
end;

create procedure getNumberOfSongsInAlbum(in inName varchar(40), out outCount int)
begin
  set outCount = (
    select count(composition.Name) from composition
    inner join album a on composition.Album_idAlbum = a.idAlbum
    where lower(a.Name) = lower(inName)
  );
end;

create procedure getCommentsBySongId (in inId int)
begin
  select idUser, Nickname, LastName, FirstName, MidName, Birhday, Mail, Telephone from user
  inner join comment c on user.idUser = c.User_idUser
  inner join composition c2 on c.Composition_idComposition = c2.idComposition
  where c2.idComposition = inId;
end;

```

Приложение Е
(обязательное)
Скрипты запросов

```
select * from composition;
```

```
select LastName, FirstName, MidName, Birhday, StartDate, EndDate from musician  
inner join performer_has_musician phm on musician.idMusician = phm.Musician_idMusician  
inner join performer p on phm.Performer_idPerformer = p.idPerformer  
where p.Name = 'Imagine Dragons'
```

```
select idOrder, Date, Status, Nickname, LastName, FirstName, MidName, Birhday, Mail, Telephone from  
`order`  
inner join order_has_composition ohs on `order`.idOrder = ohs.Order_idOrder  
inner join composition c on ohs.Composition_idComposition = c.idComposition  
inner join user u on `order`.User_idUser = u.idUser  
where c.Name = 'Natural'
```

```
select * from user
```

```
select genre.Name from genre  
inner join composition c on genre.idGenre = c.Genre_idGenre  
inner join performer p on c.Performer_idPerformer = p.idPerformer  
where p.Name = 'Imagine Dragons'
```