МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский технологический университет"
МИРЭА

Институт информационных технологий
Кафедра инструментального и прикладного программного обеспечения (ИППО)

КУРСОВОЙ ПРОЕКТ
по дисциплине
«Управление данными»

Тема курсового проекта: «Проектирование и апробация базы данных музыкального магазина»

Студент группы ИСБОп-01-14 *Карих Д.С.*

Руководитель курсового проекта работы *Матчин В.Т.*
ст.преп.

Рецензент *Матчин В.Т.*
ст.преп.

Работа представлена к защите «__»_____201__ г.

«Допущен к защите»
«__»_____201__ г.

Москва 2017

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"Московский технологический университет"
МИРЭА

Институт информационных технологий
Кафедра инструментального и прикладного программного обеспечения (ИППО)

Утверждаю

Заведующий кафедрой _____

«____» _____201___ г.

ЗАДАНИЕ
на выполнение курсового проекта по дисциплине «Управление данными»

Студент Карих Дмитрий Степанович Группа ИСБОп-01-14
Фамилия, имя, отчество

1. **Тема «Проектирование и апробация базы данных музыкального магазина»**
2. **Исходные данные:** Система автоматизации развёртывания ПО *Docker*, реляционная система управления базами данных *MariaDB*, веб-интерфейс администрирования СУБД MySQL и совместимых *phpMyAdmin*, исходные данные для заполнения БД.
3. **Перечень вопросов, подлежащих разработке, и обязательного графического материала:** *Анализ предметной области; Разработка структуры базы данных; Построение запросов к базе данных.*
4. **Срок утверждения на курсовой проект: до 31 марта 2017 г.**
5. **Срок представления курсового проекта на первичное рецензирование: до 30 апреля 2017г.**
6. **Срок представления к защите курсового проекта: до 15 мая 2017г.**

Задание на курсовую работу выдал «____» _____201___ г. *Матчин В.Т.*

Задание на курсовую работу получил «____» _____201___ г. *Карих Д.С.*

Реферат

Курсовой проект содержит XX страниц отчёта, XX примеров кода, XX иллюстраций и 11 использованных источников литературы.

Целью курсового проекта является реализация базы данных музыкального магазина, включающая в себя анализ предметной области, разработку структуры и построение запросов к базе данных, находящейся под управлением СУБД MariaDB.

Во Введении, на основе исходных данных, формулируются задачи проектирования базы данных.

В разделе «Анализ предметной области» производится разработка логической и физической модели проектируемой базы данных.

В разделе «Разработка структуры базы данных» физическая модель реализуется в виде запросов к СУБД MariaDB на языке SQL.

В разделе «Реализация информационной системы» сначала производится настройка серверного прогрмааного обеспечения, а затем разрабатывается пользовательский веб-интерфейс на базе языка программирования PHP.

В разделе «Построение запросов к базе данных» приводятся примеры комплексных запросов, отображающих содержимое сразу нескольких таблиц, а также проведено сравнение производительности СУБД MariaDB при 100, 200 и 500 тысячах строк.

В Заключении дана оценка достигнутых результатов в разработке и проектировании.

В Приложении 1 приведён листинг исходного кода пользовательского интерфейса, созданного в разделе «Реализация информационной системы».

Список определений и сокращений

1. База данных (БД) — совокупность материалов, систематизированных для автоматизированного поиска и обработки информации при помощи электронно-вычислительной машины . [1]
2. Система управления базами данных (СУБД) — совокупность программных и лингвистических средств, обеспечивающих управление созданием и использованием баз данных. [2]
3. PHP: Hypertext Preprocessor, PHP: препроцессор гипертекста — скриптовый язык общего назначения, широко применяемый для разработки веб-приложений. [3]
4. PHP-FPM — реализация протокола FastCGI, предназначенная для выполнения кода PHP. [3]
5. HTML (HyperText Markup Language — «Язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. [4]
6. NGINX — веб-сервер и почтовый прокси-сервер, работающий на Unix-подобных операционных системах. [5]

Оглавление

Введение

Ещё до появления электронно-вычислительных машин аналоги баз данных использовались во многих областях. Такой подход позволял организовывать и систематизировать большие объемы данных, доступные для обращения, когда это необходимо.

Повсеместная компьютеризация и объединение устройств в единую сеть привели к появлению гораздо более продвинутых хранилищ информации — электронных баз данных (БД). Основное назначение таких БД заключается в автоматизированной организации информации в заранее заданном формате, а также быстрому доступу к ней посредством особых запросов.

Целью данного курсового проекта является проектирование и создание базы данных на примере музыкального магазина. На сегодняшний день создание баз данных является актуальной темой в связи с повсеместной автоматизацией и нарастающими потребностями в хранении и обработке информации.

Для достижения цели курсового проекта необходимо провести анализ существующих решений в области музыкальных БД и БД, используемых в торговле; разработать структуру БД; установить необходимое программное обеспечение и реализовать структуру БД на его базе.

1. Анализ предметной области

В данном разделе создаётся и описывается логическая структура базы данных музыкального магазина. Этот этап необходим для разработки любой информационной системы и помогает представить её итоговую структуру и принцип действия.

Перед началом работы необходимо ознакомиться с принципами организации коллекции музыки. Для этого проведем анализ публичной музыкальной базы данных MusicBrainz [6].

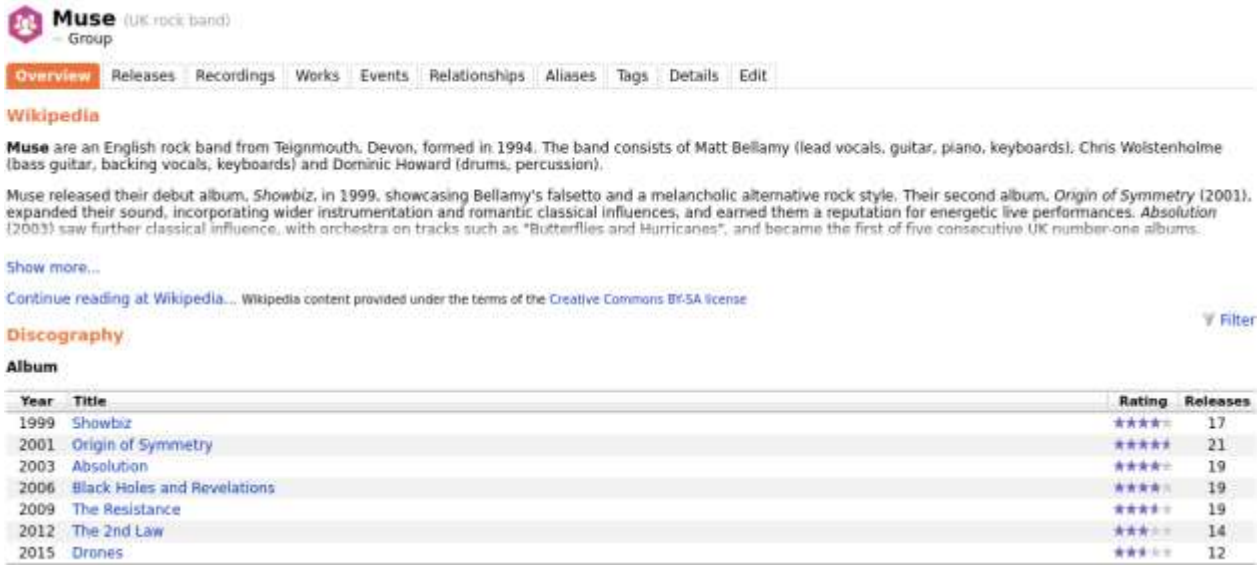


Рис. 1 — Страница группы Muse в БД MusicBrainz (список альбомов)

1.1. Выявленные базовые объекты

В результате анализа базы данных MusicBrainz удалось выделить следующие базовые объекты, которые станут основной нашей БД.

1. Релиз
    - Номер;
    - Название;
    - Исполнитель альбома;
    - Жанр;
    - Тип: альбом, мини-альбом (EP), сингл;
    - Формат: CD-диск, DVD-диск, кассета, виниловая пластинка;
    - Количество треков;

- Продолжительность;
- Дата создания записи;
- Студия звукозаписи (лейбл);
- Страна релиза.

2. Исполнитель
    - Номер;
    - Название;
    - Количество участников: одиночка, группа (> 1);
    - Дата начала деятельности;
    - Страна начала деятельности;

Таким же образом определим объекты, непосредственно описывающие работу магазина: сотрудников, логистику и т.п.

3. Работник
    - Номер;
    - Фамилия;
    - Инициалы;
    - Должность;
    - Заработная плата;
4. Товар
    - Релиз;
    - Количество;
    - Цена.
5. Операция
    - Номер;
    - Товар;
    - Работник;
    - Дата и время;
    - Количество.

1.2. Описание объектов

Релиз

Описание физического носителя с музыкой, находящегося в продаже. Не имеет уникальных полей, за исключением порядкового номера, специфичного для конкретной базы данных.

Исполнитель

Не имеет общепринятого уникального идентификатора, поэтому, аналогично релизу, не обладает уникальными полями. В одной базе может существовать несколько исполнителей с одинаковым названием.

Работник

Описывает работника магазина. В качестве ключа использует порядковый номер записи в таблице. Не ссылается на другие объекты, т.е. является самостоятельной сущностью.

Товар

Связывает товар и факт его наличия на складе. В качестве ключа использует поле «Номер» из таблицы «Релиз», что обеспечивает непосредственную связь двух таблиц и предотвращает нежелательные ошибки в номере товара.

Операция

Описывает любое изменение в таблице «Товар». В качестве ключа использует порядковый номер операции. Характер операции (поступление товара или продажа) зависит от знака в поле «Количество» (отрицательное число — продажа, положительное — поступление).

1.3. Связи объектов

Релиз и Исполнитель

Релиз может ссылаться только на одного Исполнителя. При этом на одного Исполнителя может ссылаться несколько Релизов.

Релиз и Товар

Для каждого Релиза может быть только одна запись в таблице «Товар».

Товар и Операция

На один Товар может ссылаться несколько Операций.

Операция и Работник

Работник может относиться к нескольким Операциям, но каждая операция относится исключительно к одному Работнику.

1.4. ER-диаграмма

Подведём итог анализа предметной области, составив концептуальную и физическую модели нашей базы данных. Концептуальная модель отображает отношения между сущностями, не учитывая при этом особенности конкретной СУБД.

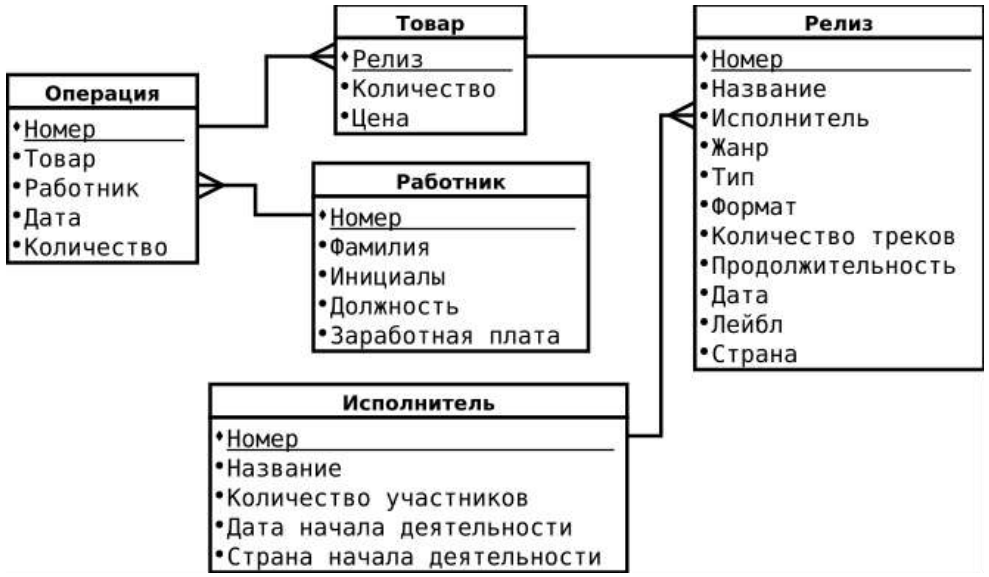


Рис. 2 — Концептуальная модель базы данных

Физическая модель учитывает требования конкретной СУБД к названиям таблиц и полей, а также её возможности по созданию связей между таблицами. Так как данный курсовой проект выполняется в СУБД MySQL, в названиях будут использоваться только латинские буквы и подчёркивания, а для соединения таблиц будет использоваться механизм внешнего ключа.

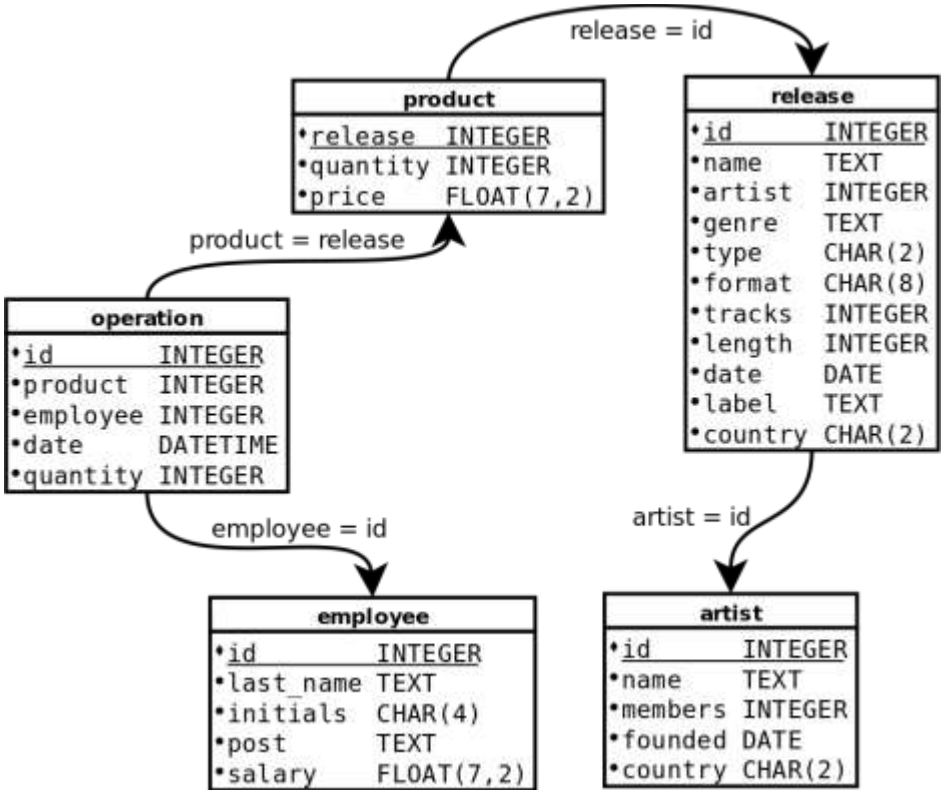


Рис. 3 — Физическая модель базы данных

Данную модель можно использовать для непосредственной реализации структуры базы данных на основе любой СУБД, совместимой с MySQL (MariaDB, Percona Server и т.п.).

2. Разработка структуры базы данных

После разработки физической модели можно приступать к непосредственному созданию таблиц посредством специальных запросов на языке программирования SQL.

SQL (англ. structured query language — язык структурированных запросов) — язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных. [7]

2.1. Создание таблиц базы данных

Запрос на создание таблицы в СУБД SQLite (частично совместимой с MySQL) имеет следующий синтаксис:

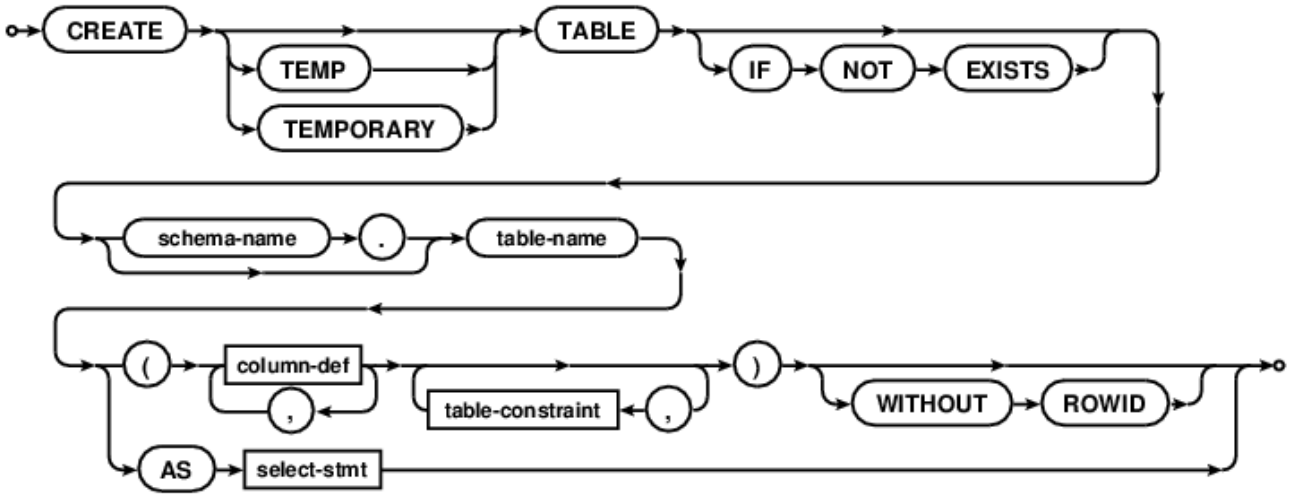


Рис. 4 — синтаксис запроса CREATE TABLE (SQLite) [8]

Для нас наиболее важны параметр table-name (название таблицы) и структура полей, задаваемая в скобках (column-def).

```
-- Структура таблицы `artist`

CREATE TABLE IF NOT EXISTS `artist` (
     `id` INTEGER NOT NULL AUTO_INCREMENT,
     `name` TEXT NOT NULL,
     `members` INTEGER,
     `founded` DATE,
     `country` CHAR(2),

     PRIMARY KEY (`id`)
);
```

```
-- Структура таблицы `release`

CREATE TABLE IF NOT EXISTS `release` (
     `id` INTEGER NOT NULL AUTO_INCREMENT,
     `name` TEXT NOT NULL,
     `artist` INTEGER NOT NULL,
     `genre` TEXT,
     `type` CHAR(2) NOT NULL,
     `format` CHAR(8) NOT NULL,
     `tracks` INTEGER NOT NULL,
     `length` INTEGER,
     `date` DATE,
     `label` TEXT,
     `country` CHAR(2),

     PRIMARY KEY (`id`),
     FOREIGN KEY (`artist`) REFERENCES `artist`(`id`)
);


-- Структура таблицы `product`

CREATE TABLE IF NOT EXISTS `product` (
     `release` INTEGER NOT NULL,
     `quantity` INTEGER NOT NULL,
     `price` FLOAT(7,2) NOT NULL,

     PRIMARY KEY (`release`),
     FOREIGN KEY (`release`) REFERENCES `release`(`id`)
);


-- Структура таблицы `employee`

CREATE TABLE IF NOT EXISTS `employee` (
     `id` INTEGER NOT NULL AUTO_INCREMENT,
     `last_name` TEXT NOT NULL,
     `initials` CHAR(4) NOT NULL,
     `post` TEXT NOT NULL,
     `salary` FLOAT(7,2) NOT NULL,

     PRIMARY KEY (`id`)
);
```

```
-- Структура таблицы `operation`

CREATE TABLE IF NOT EXISTS `operation` (
     `id` INTEGER NOT NULL AUTO_INCREMENT,
     `product` INTEGER NOT NULL,
     `employee` INTEGER NOT NULL,
     `date` DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
     `quantity` INTEGER NOT NULL,

     PRIMARY KEY (`id`),
     FOREIGN KEY (`product`) REFERENCES
`product`(`release`),
     FOREIGN KEY (`employee`) REFERENCES `employee`(`id`)
);
```

Данные запросы, будучи выполненными в командной строке MySQL или в окне запроса в phpMyAdmin, создадут таблицы и связи между ними в соответствии с физической моделью базы данных. PhpMyAdmin также поддерживает визуализацию полученной структуры (см. Рис. 5). Данная структура полностью совпадает с физической моделью (см. Рис. 3).

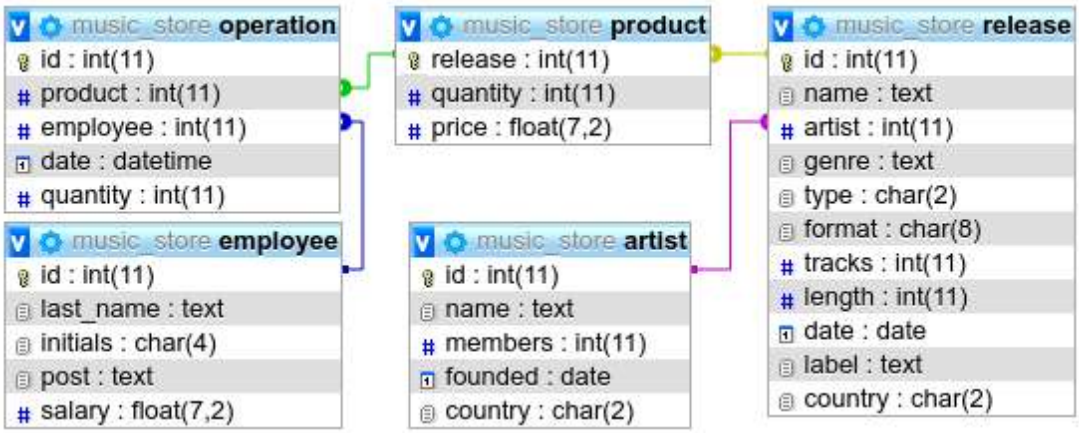


Рис. 5 — Представление структуры базы данных в phpMyAdmin

2.2. Описание таблиц базы данных

Таблица `artist` — исполнители альбомов

№	Имя столбца	Тип данных	Комментарий
1	id	INTEGER(11)	Порядковый номер
2	name	TEXT	Названиие
3	members	INTEGER(11)	Количество участников
4	founded	DATE	Дата основания
5	country	CHAR(2)	Страна

Таблица `release` — релизы (записи)

№	Имя столбца	Тип данных	Комментарий
1	id	INTEGER(11)	Порядковый номер
2	name	TEXT	Название
3	artist	INTEGER(11)	Исполнитель альбома
4	genre	TEXT	Преобладающий жанр
5	type	CHAR(2)	Тип: LP (альбом), EP, SP (сингл)
6	format	CHAR(8)	Физический носитель: CD, DVD
7	tracks	INTEGER(11)	Количество композиций
8	length	INTEGER(11)	Общая продолжительность
9	date	DATE	Дата релиза
10	label	TEXT	Студия звукозаписи (лейбл)
11	country	CHAR(2)	Страна релиза

Таблица `product` — наличие физических копий релиза на складе

№	Имя столбца	Тип данных	Комментарий
1	release	INTEGER(11)	Идентификатор релиза
2	quantity	INTEGER(11)	Количество носителей
3	price	FLOAT(7,2)	Цена

Таблица `employee` — работники магазина

№	Имя столбца	Тип данных	Комментарий
1	id	INTEGER(11)	Порядковый номер
2	last_name	TEXT	Фамилия
3	initials	CHAR(4)	Инициалы
4	post	TEXT	Должность
5	salary	FLOAT(7,2)	Зарплата

Таблица `operation` — изменения на складе

№	Имя столбца	Тип данных	Комментарий
1	id	INTEGER(11)	Порядковый номер
2	product	INTEGER(11)	Идентификатор релиза на складе
3	employee	INTEGER(11)	Идентификатор работника
4	date	DATETIME	Дата и время операции
5	quantity	INTEGER(11)	Изменение количества копий

2.3. Занесение первичной информации

Запрос на добавление информации в таблицу в СУБД SQLite имеет следующий синтаксис:

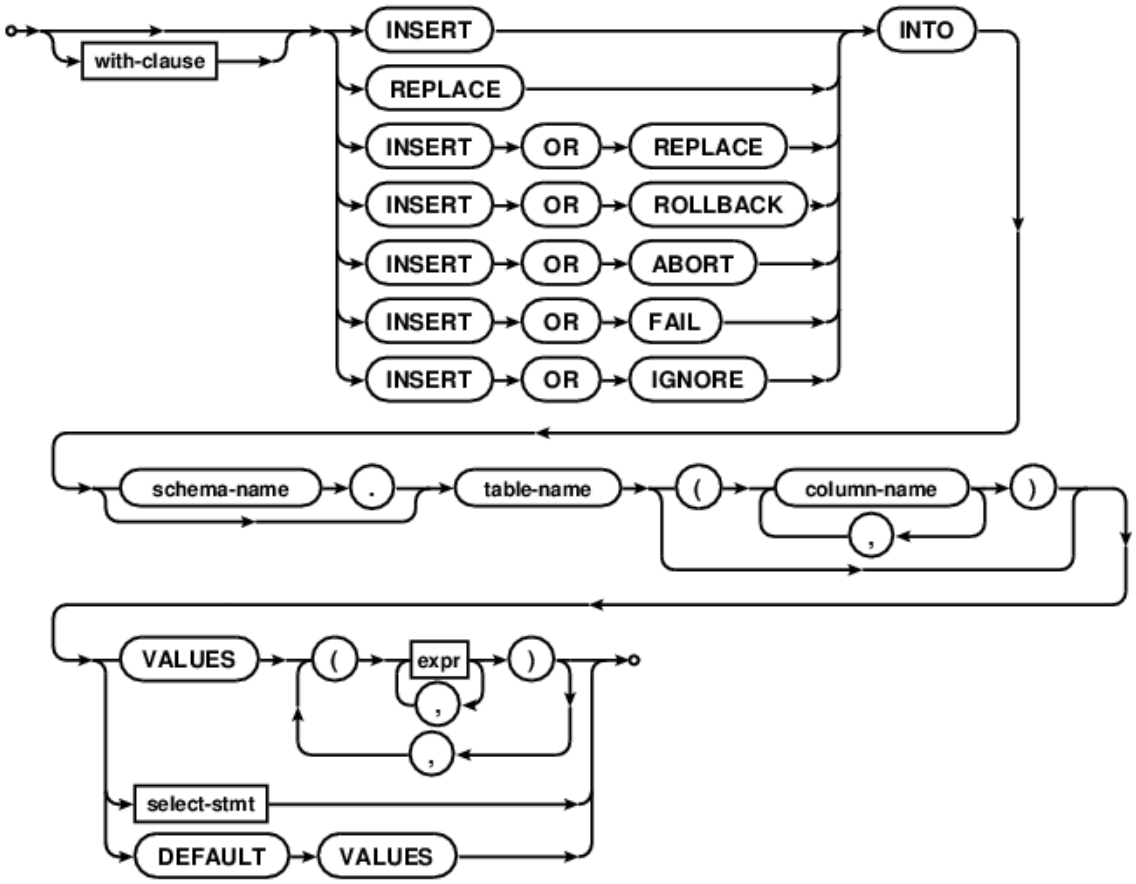


Рис. 6 — синтаксис запроса INSERT (SQLite) [10]

Следующие запросы показывают пример внесения данных в каждую таблицу. Помимо внесения одной строки эти запросы также позволяют вносить сразу несколько строк, если указать несколько наборов данных через запятую.

```
INSERT INTO `artist` (
    `name`, `members`, `founded`, `country`
) VALUES (
    'Muse', 3, '1994-01-01', 'UK'
);

INSERT INTO `release` (
    `name`, `artist`, `genre`, `type`, `format`,
    `tracks`, `length`, `date`, `label`, `country`
) VALUES (
    'Origin of Symmetry', 1, 'Rock', 'LP', 'CD',
    12, 3331, '2001-06-13', 'cutting edge', 'JP'
);
```

```
INSERT INTO `product` (
    `release`, `quantity`, `price`
) VALUES (
    1, 100, 5.0
);

INSERT INTO `employee` (
    `last_name`, `initials`, `post`, `salary`
) VALUES (
    'Айлбибеков', 'Т.А.', 'Кассир', 8500.00
);

INSERT INTO `operation` (
    `product`, `employee`, `quantity`
) VALUES (
    1, 1, 100
);
```

3. Реализация информационной системы и разработка интерфейса

Для обеспечения максимальной совместимости информационной системы с различными платформами разумнее всего будет использовать модель клиент-сервер, где в качестве клиента выступает любой веб-браузер, а исполняемой средой сервера является скриптовый язык программирования PHP.

3.1. Описание интерфейса

Разрабатываемый интерфейс рассчитан на использование сотрудниками магазина, но также может быть расширен для использования покупателями путём разграничения прав.

Интерфейс должен позволять вносить следующие изменения в таблицы:

1. Вносить исполнителей в таблицу «artists»;
2. Вносить релизы со ссылкой на исполнителей в таблицу «release»;
3. Регистрировать поступления и продажу товаров в таблице «operation» с учётом текущего количества товара в таблице «product»;
4. Добавлять новых работников в таблицу «employee».

Такой интерфейс достаточно просто реализовать на языке PHP с использованием HTML таблиц и форм.

3.2. Программная реализация информационной системы

В качестве исполняемой среды для создаваемой информационной системы были выбраны следующие компоненты:

1. Система управления контейнерами Docker;
2. СУБД MariaDB (аналог MySQL);
3. phpMyAdmin;
4. Сервер PHP-FPM версии 7;
5. Веб-сервер NGINX.

Docker используется для быстрого развёртывания остальных серверов и соединения их в единую сеть. Такой подход позволяет отвязать информационную систему от конкретного оборудования и версий программного обеспечения.

3.2.1. Подготовка программного обеспечения

Воспользуемся Docker для быстрого развёртывания всех необходимых серверов. Для запуска каждого сервера достаточно ввести по одной команде в терминал. (Предполагается, что Docker уже установлен на компьютер)

```
# СУБД MariaDB
docker run -d --name=db -v db:/var/lib/mysql \
    --expose=3306 --env="MYSQL_ROOT_PASSWORD=root" \
    --restart=always mariadb:latest

# phpMyAdmin
docker run -d --name pma --publish=127.0.0.1:8080:80 \
    --env="PMA_HOST=db" --link=db --restart=always \
    phpmyadmin/phpmyadmin:latest

# PHP-FPM
docker run -d --name=php --volume=$(pwd):/var/www/html \
    --expose=9000 --link=db --restart=always \
    php:7-fpm-alpine
```

```
# Драйвер MySQLi для PHP
docker exec php docker-php-ext-install mysqli
docker restart php

# NGINX
docker run -d --name=nginx --link=php --restart=always \
    --volume=$(pwd):/var/www/html \
    --publish=127.0.0.1:80:80 \
    --env="FPM=true" --env="FPM_HOST=php" \
    thedrhax/nginx-stateless:v0.1
```

После выполнения данной серии команд Docker загрузит, установит и запустит 4 контейнера с необходимыми серверами. Два из них, phpMyAdmin и NGINX, будут ожидать установки соединения через браузер на локальных портах 8080 и 80 соответственно.

3.2.2. Обзор возможностей интерфейса

Рис. 7 — Главная страница

Главная страница содержит ссылки на все остальные разделы и не выполняет каких-либо действий.

Рис. 8 — Раздел «Исполнители»

Данный раздел позволяет просматривать и вносить изменения в таблицу «artists». Правая колонка содержит ссылки, нажатие на которые приводит к

удалению данной строки из базы данных. Последняя строка предназначена для внесения новых записей в таблицу.

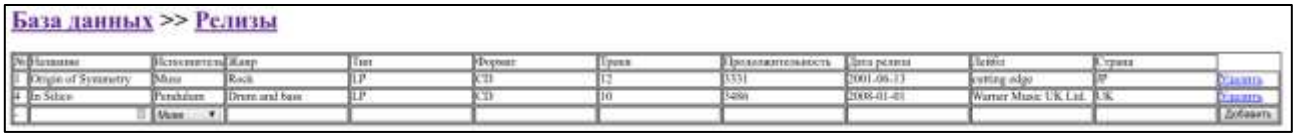


Рис. 9 — Раздел «Релизы»

Данный раздел предназначен для внесения новых релизов в базу данных. Немного отличается от предыдущего тем, что поле ввода «Исполнитель» имеет вид выпадающего списка с перечислением всех существующих исполнителей.

Рис. 10 — Раздел «Работники»

Страница, позволяющая вносить новые и удалять старые записи из таблицы «employee». Полностью аналогична по функциональности разделу «Исполнители».

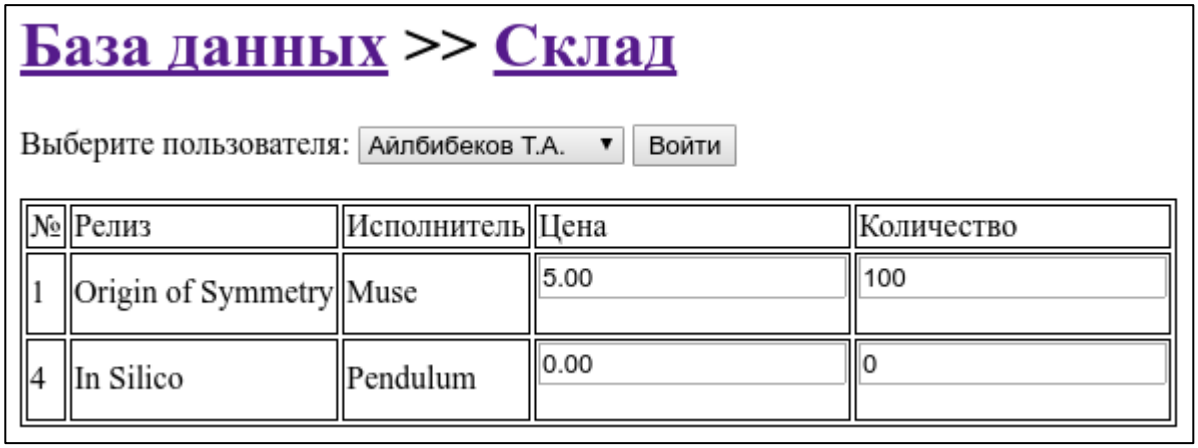
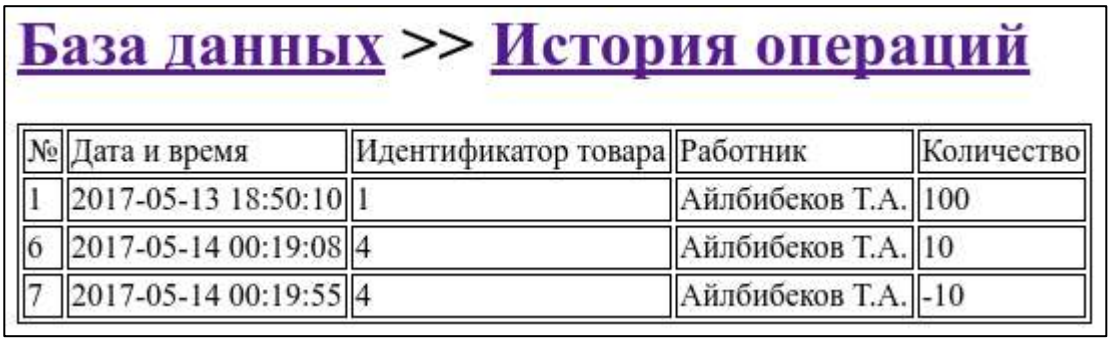


Рис. 11 — Раздел «Склад»

Данная страница предназначена для прямого внесения изменений в таблицу «product». Для изменения колонки «Количество» необходимо выбрать пользователя из выпадающего списка над таблицей. Выпадающий список перечисляет всех работников магазина, указанных в таблице «employee».



База данных >> История операций

№	Дата и время	Идентификатор товара	Работник	Количество
1	2017-05-13 18:50:10	1	Айлбибеков Т.А.	100
6	2017-05-14 00:19:08	4	Айлбибеков Т.А.	10
7	2017-05-14 00:19:55	4	Айлбибеков Т.А.	-10

Рис. 12 — Раздел «История операций»

Данная страница показывает журнал изменений на складе. Информация доступна только для чтения, так как полностью зависит от других разделов.

Исходный код интерфейса этой информационной системы представлен в Приложении 1.

4. Построение запросов к базе данных

4.1. Создание запросов

В данном разделе представлены примеры различных по сложности запросов, применимых к созданной базе данных.

1. Выборка всех исполнителей из таблицы "artist" с подсчётом количества релизов, указанных в таблице "release", с сортировкой по столбцу `id` (по возрастанию).

```
SELECT `id`, `name`, `members`, YEAR(`founded`),
       `country`, COALESCE(`count`, 0)
FROM `artist`
LEFT OUTER JOIN (
    SELECT `artist`, COUNT(`artist`) AS `count`
    FROM `release` GROUP BY `artist`
) AS `release`
ON `artist`.`id` = `release`.`artist`
ORDER BY `id` ASC;
```

Функция YEAR() извлекает год из поля типа DATE. Функция COALESCE() возвращает 0 для всех исполнителей, для которых не указано ни одного релиза. LEFT OUTER JOIN используется для объединения таблиц `artist` и `release` с сохранением даже тех исполнителей, для которых релизы не указаны.

id ▲ 1	name	members	YEAR(`founded`)	country	COALESCE(`count`, 0)
1	Muse	3	1994	UK	1
2	Pendulum	5	2002	AU	1

Рис. 13 — результат выполнения запроса 1

2. Выборка всех релизов с подстановкой названия исполнителя вместо его номера и с сортировкой по возрастанию номера релиза.

```
SELECT `release`.`id`, `release`.`name`,
       `artist`.`name` AS `artist`, `genre`, `type`,
       `format`, `tracks`, `length`, `date`, `label`,
       `release`.`country`
FROM `release`
LEFT OUTER JOIN `artist`
ON `release`.`artist` = `artist`.`id`
ORDER BY `id` ASC;
```

id ▲ 1	name	artist	genre	type	format	tracks	length	date	label	country
1	Origin of Symmetry	Muse	Rock	LP	CD	12	3331	2001-06-13	cutting edge	JP
4	In Silico	Pendulum	Drum and bass	LP	CD	10	3486	2008-01-01	Warner Music UK Ltd.	UK

Рис. 14 — результат выполнения запроса 2

3. Выборка всех товаров со склада с подстановкой названия релиза и исполнителя с сортировкой по возрастанию идентификатора релиза.

```
SELECT `release`.`id`, `release`.`name`, `artist`.`name`,
       COALESCE(`product`.`price`, 0),
       COALESCE(`product`.`quantity`, 0)
FROM `release`
LEFT OUTER JOIN `artist`
    ON `release`.`artist` = `artist`.`id`
LEFT OUTER JOIN `product`
    ON `product`.`release` = `release`.`id`
ORDER BY `release`.`id` ASC;
```

id ▲ 1	name	name	COALESCE(`product`.`price`, 0)	COALESCE(`product`.`quantity`, 0)
1	Origin of Symmetry	Muse	5.00	100
4	In Silico	Pendulum	0.00	0

Рис. 15 — результат выполнения запроса 3

4. Отображение истории всех операций на складе с подстановкой ФИО работника, проводившего операцию.

```
SELECT `operation`.`id`, `date`, `product`,
       CONCAT(`employee`.`last_name`, ' ',
              `employee`.`initials`), `quantity`
FROM `operation`
LEFT OUTER JOIN `employee`
    ON `employee` = `employee`.`id`
```

Использует функцию CONCAT() для объединения двух текстовых полей и одной строковой константы в одно поле.

id	date	product	CONCAT(`employee`.`last_name`, '', `employee`.`initials`)	quantity
1	2017-05-13 18:50:10	1	Айлбибеков Т.А.	100
6	2017-05-14 00:19:08	4	Айлбибеков Т.А.	10
7	2017-05-14 00:19:55	4	Айлбибеков Т.А.	-10

Рис. 16 — результат выполнения запроса 4

5. Расчет средней стоимости релиза для каждого из исполнителей с сортировкой по убыванию цены.

```
SELECT `artist`.`id`, `artist`.`name` AS `artist`,
       `release`.`avg_price` AS `price`
FROM `artist`
JOIN (
    SELECT `release`.`artist`,
           AVG(`product`.`price`) AS `avg_price`
    FROM `release`
    JOIN `product`
        ON `product`.`release` = `release`.`id`
    GROUP BY `release`.`artist`
) AS `release` ON `release`.`artist` = `artist`.`id`
ORDER BY `release`.`avg_price` DESC;
```

id	artist	price
1	Muse	5.000000
2	Pendulum	4.000000

Рис. 17 — результат выполнения запроса 5

6. Вывод всех сотрудников с общим количеством продаж на складе и суммарным количеством проданных товаров с сортировкой по последнему полю.

```
SELECT `id`, `last_name`, `initials`,
       COALESCE(`operation`.`count`, 0),
       COALESCE(`operation`.`sum`, 0)
FROM `employee`
LEFT OUTER JOIN (
    SELECT `employee`, COUNT(`id`) AS `count`,
           -SUM(`quantity`) AS `sum`
    FROM `operation`
    WHERE `quantity` < 0
    GROUP BY `employee`
) AS `operation`
ON `employee`.`id` = `operation`.`employee`
ORDER BY `operation`.`sum` DESC;
```

id	last_name	initials	COALESCE(`operation`.`count`, 0)	COALESCE(`operation`.`sum`, 0)
1	Айлбибеков	T.A.	7	23
4	Тестов	T.T.	1	10

Рис. 18 — результат выполнения запроса 6

4.2. Тестирование производительности

Для тестирования производительности создадим новую таблицу, которую затем заполним случайными числами.

```
CREATE TABLE `performance` (
    `id` INTEGER NOT NULL AUTO_INCREMENT,
    `data` INTEGER NOT NULL,

    PRIMARY KEY (`id`)
);
```

Чтобы быстро заполнить таблицу нужным количеством данных, воспользуемся возможностями командной оболочки Bash. Ниже приведён пример однострочного скрипта, отправляющего 100000 случайных чисел в базу данных посредством консольного клиента.

```
(for i in {1..100000}; do echo 'INSERT INTO
`music_store`.`performance`(`data`) VALUES (' $RANDOM
');'; done) | docker exec -i db mysql -uroot -proot
music_store
```

Примечание: Предполагается, что MySQL сервер запущен в Docker согласно описанному ранее методу. В противном случае вторая часть команды (после символа «|») может выглядеть иначе.

Были протестированы следующие запросы для трёх наборов данных:

```
1. SELECT AVG(`data`) FROM `performance`;
2. SELECT * FROM `performance` WHERE `data` BETWEEN 100
   AND 200;
3. SELECT `data` FROM `performance` GROUP BY `data`;
4. SET @avg = (SELECT AVG(`data`) FROM `performance`)
```

	AVG	BETWEEN	GROUP	SET
100000	53 ms	5.6 ms	119.8 ms	158.3 ms
200000	110.5 ms	4.9 ms	222.7 ms	317.5 ms
500000	258 ms	5.2 ms	505.3 ms	738 ms

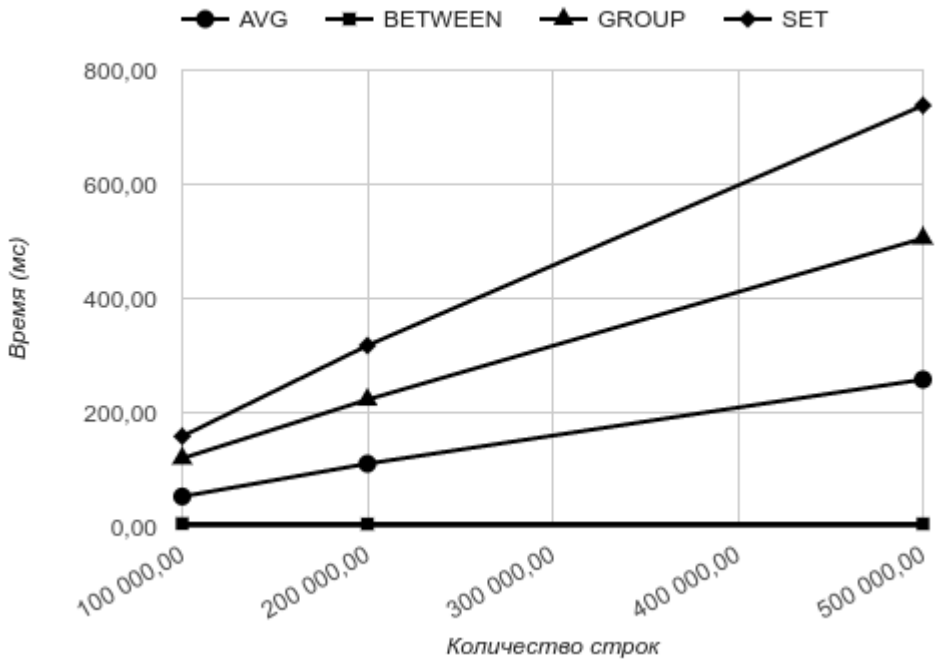


Рис. 19 — зависимость времени выполнения запроса от количества строк

Как можно заметить, длительность запроса линейно возрастает в зависимости от количества строк и даже при 500 тысячах записей составляет, большинстве случаев, меньше половины секунды. При использовании более производительной машины это значение может стать ещё ниже за счёт большего использования оперативной памяти и кэша (что поддерживается движком InnoDB).

Заключение

В результате выполнения данного курсового проекта была создана информационная система, обеспечивающая работу простейшего музыкального магазина.

Созданная система полностью основана на открытом и свободном программном обеспечении, поэтому является крайне выгодной при внедрении и использовании фирмой.

Использование языка программирования PHP для разработки пользовательского интерфейса позволило абстрагироваться от клиентского оборудования: в 99% случае веб-браузеры работают одинаково на всех платформах, поэтому отдельная поддержка различных платформ не требуется.

Система управления базами данных MariaDB является высоко масштабируемой и подходит как для частного применения, так и для компаний средних и крупных размеров.

Docker, являющийся относительно новым инструментом, оказался очень полезен для быстрого развёртывания инфраструктуры. Созданный нами проект может быть упакован и быстро перенесён на любое оборудования без проблем с совместимостью.

Список используемых источников литературы

1. Википедия: База данных // https://ru.wikipedia.org/wiki/База_данных (дата обращения: 11.05.17)
2. Википедия: Система управления базами данных // https://ru.wikipedia.org/wiki/Система_управления_базами_данных (дата обращения: 11.05.17)
3. Википедия: PHP // https://ru.wikipedia.org/wiki/PHP (дата обращения: 11.05.17)
4. Википедия: HTML // https://ru.wikipedia.org/wiki/HTML (дата обращения: 11.05.17)
5. Официальный сайт NGINX // http://nginx.org/ (дата обращения: 11.05.17)
6. Сайт проекта MusicBrainz // https://musicbrainz.org (дата обращения: 11.05.17)
7. Википедия: SQL // https://ru.wikipedia.org/wiki/SQL (дата обращения: 12.05.17)
8. Документация SQLite: CREATE TABLE // https://www.sqlite.org/lang_createtable.html (дата обращения: 12.05.17)
9. Документация MySQL: CREATE TABLE // https://dev.mysql.com/doc/refman/5.5/en/create-table.html (дата обращения: 12.05.17)
10. Документация SQLite: INSERT // https://sqlite.org/lang_insert.html (дата обращения: 12.05.17)
11. Официальный сайт Docker // https://www.docker.com/ (дата обращения: 14.05.17)

Приложение 1

Исходный код интерфейса информационной системы

Файл index.php

```
<h1>База данных музыкального магазина</h1
<b>Выберите раздел:</b>
<ul>
    <li><a href="/artists.php">Исполнители</a></li>
    <li><a href="/releases.php">Релизы</a></li>
    <li><a href="/employees.php">Работники</a></li>
    <li><a href="/products.php">Склад</a></li>
    <li><a href="/operations.php">История операций</a></li>
</ul>
```

Файл config.php

```
<?php
    $mysqli = new mysqli("db", "root", "root", "music_store");
    $mysqli->set_charset("utf8");
?>
```

Файл style.css

```
table, th, td {
    border: 1px solid black;
}
```

Файл artists.php

```
<html>
<head>
    <title>База данных >> Исполнители</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body><h1><a href="/">База данных</a> >>
<a href="/artists.php">Исполнители</a></h1>

<?php
    include 'config.php';

    if ($_GET['action'] === "add") {
        $stmt = $mysqli->prepare(
            "INSERT INTO
`artist`(`name`,`members`,`founded`,`country`)
             VALUES (?, ?, ?, ?)"
        );
        $stmt->bind_param(
            'sdss',  $_GET['name'], $_GET['members'],
            $s = $_GET['year'] . '-01-01', $_GET['country']
        );
        $stmt->execute();
    }
```

```
    if ($_GET['action'] === "delete") {
        $stmt = $mysqli->prepare("DELETE FROM `artist` WHERE `id`
= ?");
        $stmt->bind_param('d', $_GET['id']);
        $stmt->execute();
    }
?>

<table>
    <tr>
        <td>№</td> <td>Название</td> <td>Участники</td>
<td>Год</td>
        <td>Страна</td> <td>Количество релизов</td>
<td>Действия</td>
    </tr>

    <?php
        $result = $mysqli->query(
            "SELECT `id`, `name`, `members`, YEAR(`founded`),
                    `country`, COALESCE(`count`, 0)
             FROM `artist`
             LEFT OUTER JOIN (
                 SELECT `artist`, COUNT(`artist`) AS `count`
                 FROM `release` GROUP BY `artist`
             ) AS `release`
             ON `artist`.`id` = `release`.`artist`
             ORDER BY `id` ASC"
        );

        while ($row = mysqli_fetch_array($result)) {
            echo "<tr>";
            for ($i = 0; $i < 6; $i++) {
                echo "<td>" . $row[$i] . "</td>";
            }
            echo "<td><a href=\"/artists.php?action=delete&id=" .
$row['id']
                . "\">Удалить</a></td></tr>";
        }
        $result->close();
    ?>

    <tr><form method="GET" action="/artists.php">
        <td><input type="hidden" name="action" value="add" />-
</td>
        <td><input type="text" name="name" /></td>
        <td><input type="text" name="members" /></td>
        <td><input type="text" name="year" /></td>
        <td><input type="text" name="country" /></td>
        <td>-</td>
        <td><input type="submit" value="Добавить" /></td>
    </form></tr>
</table></body></html>
```

Файл releases.php

```
<html>
<head>
    <title>База данных >> Релизы</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body><h1><a href="/">База данных</a> >> <a
href="/releases.php">Релизы</a></h1>

<?php
    include 'config.php';

    if ($_GET['action'] === "add") {
        $stmt = $mysqli->prepare(
            "INSERT INTO `release`(`name`, `artist`, `genre`,
`type`, `format`,
                                    `tracks`, `length`, `date`,
`label`, `country`)
             VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"
        );
        $stmt->bind_param(
            'sdsssddsss',  $_GET['name'], $_GET['artist'],
$_GET['genre'],
            $_GET['type'], $_GET['format'], $_GET['tracks'],
$_GET['length'],
            $_GET['date'], $_GET['label'], $_GET['country']
        );
        $stmt->execute();
    }

    if ($_GET['action'] === "delete") {
        $stmt = $mysqli->prepare("DELETE FROM `release` WHERE `id`
= ?");
        $stmt->bind_param('d', $_GET['id']);
        $stmt->execute();
    }
?>

<table>
    <tr>
        <td>№</td> <td>Название</td> <td>Исполнитель</td>
<td>Жанр</td>
        <td>Тип</td> <td>Формат</td> <td>Треки</td>
<td>Продолжительность</td>
        <td>Дата релиза</td> <td>Лейбл</td> <td>Страна</td>
    </tr>

    <?php
        $result = $mysqli->query(
            "SELECT `release`.`id`, `release`.`name`,
`artist`.`name`,
```

```
                    `genre`, `type`, `format`, `tracks`, `length`,
`date`,
                    `label`, `release`.`country`
             FROM `release` LEFT OUTER JOIN `artist`
             ON `release`.`artist` = `artist`.`id`
             ORDER BY `id` ASC"
        );

        while ($row = mysqli_fetch_array($result)) {
            echo "<tr>";
            for ($i = 0; $i < 11; $i++) {
                echo "<td>" . $row[$i] . "</td>";
            }
            echo "<td><a href=\"/releases.php?action=delete&id=" .
$row['id']
                . "\">Удалить</a></td></tr>";
        }

        $result->close();
    ?>

    <tr><form method="GET" action="/releases.php">
        <td><input type="hidden" name="action" value="add" />-
</td>
        <td><input type="text" name="name" /></td>
        <td><select name="artist">
        <?php
            $result = $mysqli->query("SELECT `id`, `name` FROM
`artist`");
            while ($row = mysqli_fetch_array($result)) {
                echo "<option value=" . $row['id'] . ">" .
$row['name'] . "</option>";
            }
            $result->close();
        ?>
        </select></td>
        <td><input type="text" name="genre" /></td>
        <td><input type="text" name="type" /></td>
        <td><input type="text" name="format" /></td>
        <td><input type="text" name="tracks" /></td>
        <td><input type="text" name="length" /></td>
        <td><input type="text" name="date" /></td>
        <td><input type="text" name="label" /></td>
        <td><input type="text" name="country" /></td>
        <td><input type="submit" value="Добавить" /></td>
    </form></tr>
</table></body></html>
```

Файл employees.php

```
<html>
<head>
    <title>База данных >> Работники</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body><h1><a href="/">База данных</a> >> <a
href="/employees.php">Работники</a></h1>

<?php
    include 'config.php';

    if ($_GET['action'] === "add") {
        $stmt = $mysqli->prepare(
            "INSERT INTO `employee`(`last_name`, `initials`,
`post`, `salary`)
             VALUES (?, ?, ?, ?)"
        );
        $stmt->bind_param(
            'sssd',  $_GET['last_name'], $_GET['initials'],
$_GET['post'],
            $_GET['salary']
        );
        $stmt->execute();
    }

    if ($_GET['action'] === "delete") {
        $stmt = $mysqli->prepare("DELETE FROM `employee` WHERE
`id` = ?");
        $stmt->bind_param('d', $_GET['id']);
        $stmt->execute();
    }
?>

<table>
    <tr>
        <td>№</td> <td>Фамилия</td> <td>Инициалы</td>
<td>Должность</td>
        <td>Зарплата</td> <td>Действия</td>
    </tr>

    <?php
        $result = $mysqli->query("SELECT * FROM `employee` ORDER
BY `id` ASC");

        while ($row = mysqli_fetch_array($result)) {
            echo "<tr>";
            for ($i = 0; $i < 5; $i++) {
                echo "<td>" . $row[$i] . "</td>";
            }
            echo "<td><a href=\"/employees.php?action=delete&id="
. $row['id']
```

```
                . "\">Удалить</a></td></tr>";
        }

        $result->close();
    ?>

    <tr><form method="GET" action="/employees.php">
        <td><input type="hidden" name="action" value="add" />-
</td>
        <td><input type="text" name="last_name" /></td>
        <td><input type="text" name="initials" /></td>
        <td><input type="text" name="post" /></td>
        <td><input type="text" name="salary" /></td>
        <td><input type="submit" value="Добавить" /></td>
    </form></tr>
</table></body></html>
```

Файл products.php

```
<?php
    if (!empty($_GET['user'])) {
        setcookie("user", $_GET['user']);
        header('Location: ' . '/products.php');
        exit();
    }
?>

<html>
<head>
    <title>База данных >> Склад</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body><h1><a href="/">База данных</a> >> <a
href="/products.php">Склад</a></h1>

<?php
    include 'config.php';

    if ($_GET['action'] === "price") {
        if ($mysqli->query("SELECT * FROM `product` WHERE
`release` = " . $_GET['id'])) {
            $stmt = $mysqli->prepare("UPDATE `product` SET
`price`=? WHERE `release`=?");
        } else {
            $stmt = $mysqli->prepare("INSERT INTO
`product`(`price`, `release`) VALUES (?, ?)");
        }
        $stmt->bind_param('dd', $_GET['value'], $_GET['id']);
        $stmt->execute();
    }

    if ($_GET['action'] === "quantity") {
```

```
        if (empty($_COOKIE['user']) || $_COOKIE['user'] <= 0) {
            print "<p>Вы должны выбрать пользователя для этой
операции!</p>";
        } else {
            $old = $mysqli->query(
                "SELECT `quantity` FROM `product` WHERE `release`
= " . $_GET['id']
            )->fetch_object()->quantity;

            if ($mysqli->query("SELECT * FROM `product` WHERE
`release` = " . $_GET['id'])) {
                $stmt = $mysqli->prepare("UPDATE `product` SET
`quantity`=? WHERE `release`=?");
            } else {
                $stmt = $mysqli->prepare("INSERT INTO
`product`(`quantity`, `release`) VALUES (?, ?)");
            }
            $stmt->bind_param('dd', $_GET['value'], $_GET['id']);
            $stmt->execute();

            $stmt = $mysqli->prepare(
                "INSERT INTO
`operation`(`product`,`employee`,`quantity`) VALUES (?, ?, ?)"
            );
            $stmt->bind_param('ddd', $_GET['id'],
$_COOKIE['user'], $s = $_GET['value'] - $old);
            $stmt->execute();
        }
    }
?>

<p><form>Выберите пользователя: <select name="user">
<option value="-1">Вход не выполнен</option>
<?php
    $result = $mysqli->query("SELECT `id`, `last_name`, `initials`
FROM `employee`");
    while ($row = mysqli_fetch_array($result)) {
        echo "<option value=" . $row['id'];
        if (!empty($_COOKIE['user']) && $_COOKIE['user'] ==
$row['id']) {
            echo " selected";
        }
        echo ">" . $row['last_name'] . " " . $row['initials'] .
"</option>";
    }
    $result->close();
?>
</select> <input type="submit" value="Войти" /></form></p>

<table>
    <tr>
        <td>№</td> <td>Релиз</td> <td>Исполнитель</td>
<td>Цена</td>
```

```
        <td>Количество</td>
    </tr>

    <?php
        $result = $mysqli->query(
            "SELECT `release`.`id`, `release`.`name`,
`artist`.`name`,
                    COALESCE(`product`.`price`, 0),
                    COALESCE(`product`.`quantity`, 0)
             FROM `release`
             LEFT OUTER JOIN `artist` ON `release`.`artist` =
`artist`.`id`
             LEFT OUTER JOIN `product` ON `product`.`release` =
`release`.`id`"
        );

        while ($row = mysqli_fetch_array($result)) {
            echo "<tr>";
            echo "<td>" . $row[0] . "</td>";
            echo "<td>" . $row[1] . "</td>";
            echo "<td>" . $row[2] . "</td>";
            echo "<td><form>
                    <input type=\"hidden\" name=\"action\"
value=\"price\" />
                    <input type=\"hidden\" name=\"id\" value=\"" .
$row['id'] . "\" />
                    <input type=\"text\" name=\"value\" value=\""
. $row[3] . "\">
                </form></td>";
            echo "<td><form>
                    <input type=\"hidden\" name=\"action\"
value=\"quantity\" />
                    <input type=\"hidden\" name=\"id\" value=\"" .
$row['id'] . "\" />
                    <input type=\"text\" name=\"value\" value=\""
. $row[4] . "\">
                </form></td>";
            echo "</tr>";
        }

        $result->close();
    ?>
</table></body></html>
```

Файл operations.php

```
<html>
<head>
    <title>База данных >> История операций</title>
    <link rel="stylesheet" type="text/css" href="style.css" />
</head>

<body><h1><a href="/">База данных</a> >> <a
href="/operations.php">История операций</a></h1>

<table>
    <tr>
        <td>№</td> <td>Дата и время</td> <td>Идентификатор
товара</td>
        <td>Работник</td> <td>Количество</td>
    </tr>

    <?php
        include 'config.php';

        $result = $mysqli->query(
            "SELECT `operation`.`id`, `date`, `product`,
                    CONCAT(`employee`.`last_name`, ' ',
`employee`.`initials`),
                    `quantity` FROM `operation`
             LEFT OUTER JOIN `employee` ON `employee` =
`employee`.`id`"
        );

        while ($row = mysqli_fetch_array($result)) {
            echo "<tr>";
            for ($i = 0; $i < 5; $i++) {
                echo "<td>" . $row[$i] . "</td>";
            }
        }

        $result->close();
    ?>
</table></body></html>
```