Москва 2021

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью ВКР является – разработка базы данных фотографии.

Актуальность заключается в настоящем времени нет смысла хранить фотографии в печатном формате. Для упрощения хранения разрабатывается база данных фотография.

Объектом исследования является изображение.

Предметом исследования является фотография.

Результатом окончания ВКР будет прототип разработанной базы данных.

Какие задачи включает в себя работа над ВКР:

1)Анализ предметной области;

2)Анализ существующих баз данных;

3)Анализ схемы базы данных;

4)Проектирование базы данных;

5)Разработка базы данных.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Исследование предметной области

1.1.1 Анализ основных понятий

Согласно теме исследования основные понятия предметной области являются:

Фотография — технология записи изображения путём регистрации оптических излучений с помощью светочувствительного фотоматериала или полупроводникового преобразователя.

Фотограф — человек, создающий фотографии при помощи фотоаппарата.

Фотоаппарат — устройство для регистрации неподвижных изображений (получения фотографий).

1.1.2 Анализ основных объектов

Описание объекта фотография представлен в таблице 1.1.

Таблица 1.1 — описание объекта фотография

Название характеристики	**Описание**
Форматы для печати	1) 9 x 13; 2) 10 x 15; 3) 13 x 18; 4) 15 x 21; 5) 20 x 30;
Форматы файлов	RAW, JPEG, TIFF, DNG.
Цвет	Черно-белая или цветная
Принцип получения	Плёночная, цифровая, электрографическая.
Виды деятельности	Научная, публицистическая, художественная.
Жанры	1)Пейзаж, 2)портретные фото, 3)уличная фотография, 4)репортаж, 5)жанровая.

Описание Объекта фотоаппарат представлен в таблице 1.2.

Таблица 1.2 — характеристики фотоаппарата

Название характеристик	**Описание**
Матрица	Это сердце фотокамеры — объединение светочувствительных элементов, отвечающее за преобразование энергии света в электрический заряд, то есть переводящее оптическое изображение в цифровые данные, которые затем последовательно поступают в преобразователь, процессор и на карту памяти.
Объектив	Это оптическая система, состоящая из нескольких линз, расположенных внутри оправы. Линзы могут быть стеклянными или даже пластиковыми. Проходящий сквозь линзы световой поток преломляется и формирует на матрице изображение.
Диафрагма	Это механизм, отвечающий за регулирование потока света, который попадает на матрицу фотокамеры.
Видоискатель	Это вспомогательное устройство, с помощью которого фотограф наблюдает за объектом съемки и определяет границы будущего кадра.

1.1.3 Основные действия с объектами

Основные действия с объектами:

— фотоаппарат:

1) сохранять снимки,

2) снимать видео,

3) хранить фотографии,

— фотограф:

1) фотографировать,

2) редактировать фотографии,

3) просмотр фотографии,

4) удалять фотографии,

5) менять режимы съемки,

— фотография:

1) сохранять историю.

1.1.4 Анализ основных участников предметной области:

Участниками предметной области будут:

Фотограф — человек, в чьи задачи входит подготовка к проведению съёмок (выбор темы, переговоры, получение разрешений и согласований, подбор моделей, реквизита, оборудования, выбор места и т. п.), непосредственно фотосъёмка и последующая работа по обработке и печати фотографий, продажа материала.

Модель — человек, которого снимает фотограф.

1.2 Анализ существующих СУБД

1.2.1 MongoDB

1.2.1.1 Назначение СУБД

MongoDB — это ориентированная на документы база данных NoSQL с открытым исходным кодом, которая использует для хранения структуру JSON. Модель данных MongoDB позволяет представлять иерархические отношения, проще хранить массивы и другие более сложные структур.

1.2.1.2 Основные возможности

Это кроссплатформенная документоориентированная база данных NoSQL с открытым исходным кодом. Она не требует описания схемы таблиц, как в реляционных БД. Данные хранятся в виде коллекций и документов.

Между коллекциями нет сложных соединений типа JOIN, как между таблицами реляционных БД. Обычно соединение производится при сохранении данных путем объединения документов.

Данные хранятся в формате BSON (бинарные JSON-подобные документы)

У коллекций не обязательно должна быть схожая структура. У одного документа может быть один набор полей, в то время как у другого документа — совершенно другой (как тип, так и количество полей).

1.2.1.3 Типы данных

Основные типы данных приведены в таблице 1.3.

Таблица 1.3 — типы данных

Тип данных	описание
String	строковый тип данных
Array (массив)	тип данных для хранения массивов элементов
Binary data	тип для хранения данных в бинарном формате
Boolean	булевый тип данных, хранящий логические значения TRUE или FALSE
Date	хранит дату в формате времени Unix
Double	числовой тип данных для хранения чисел с плавающей точкой
Integer	используется для хранения целочисленных значений

1.2.1.4 Язык запросов

СУБД MongoDB относится к NoSQL базам данных, основной чертой которых является нереляционный характер и соответственно язык запросов, отличный от SQL. В MongoDB в качестве язык запросов используется JavaScript и JSON-структуры. Выбор столь нехарактерного языка запроса объясняется тем, что эта документ-ориентированная СУБД использует JSON-

формат для представления документов и вывода результатов. Физически JSON-структуры хранятся в бинарном BSON-формате. Некоторые примеры запросов изображены на рисунке 1.1, рисунке 1.2.

```
db.mybase.insert({title: "MySQL"})
db.mybase.insert({title: "PostreSQL"})
db.mybase.insert({title: "MongoDB"})
db.mybase.insert([{title: "MS SQL"}, {title: "Oracle"}])
```

Рисунок 1.1 — вставка нового документа

```
db.mybase.find({title: /^M/});
{ "_id" : ObjectId("51f4dae823d2a4ef32d25ec4"), "title" : "MySQL" }
{ "_id" : ObjectId("51f4dae823d2a4ef32d25ec6"), "title" : "MongoDB" }
{ "_id" : ObjectId("51f4dae923d2a4ef32d25ec7"), "title" : "MS SQL" }
```

Рисунок 1.2 — регулярные выражения

1.2.2 Neo4j

1.2.2.1 Назначение СУБД

Neo4j является ведущей в мире графической базой данных с открытым исходным кодом, которая разработана с использованием технологии Java. Он легко масштабируется и не содержит схем (NoSQL).

Граф представляет собой графическое представление набора объектов, где некоторые пары объектов связаны ссылками. Он состоит из двух элементов — узлов (вершин) и отношений (ребер). База данных графиков — это база данных, используемая для моделирования данных в форме графиков. Здесь узлы графа изображают сущности, в то время как отношения изображают ассоциацию этих узлов.

1.2.2.2 Основные возможности

Основные возможности СУБД Neo4j:

1) связанные данные легко представить,

2) получение/просмотр/навигация по большему количеству подключенных данных очень просто и быстро,

3) команды языка запросов Neo4j CQL — это удобный и читаемый формат, очень простой в освоении,

4) он использует простую и мощную модель данных,

5) не требуется сложных подключений для получения связанных данных, потому что легко получить его соседние узлы или детали отношений без подключений или индексов.

1.2.2.3 Типы данных

Основные типы данных: boolean, byte, short, int, long, float, double, char, string,

1.2.2.4 Язык запросов

Cypher является декларативным графовым языком запросов, который позволяет писать выразительные и эффективные запросы на получение данных из хранилища графов и их изменение. Cypher является относительно простым, но весьма мощным языком. Очень сложные запросы к базе данных могут быть легко выражены посредством Cypher. Это позволяет вам сфокусироваться на предметной области, не тратя время на доступ к базе данных.

Язык запросов CQL расшифровывается как Cypher Query Language. Как база данных Oracle имеет язык запросов SQL, Neo4j имеет CQL в качестве языка запросов. Пример запроса CREATE (<node-name>:<label-name>).

1.2.3 MySQL

1.2.3.1 Назначение СУБД

MySQL — система управления реляционными базами данных с открытым исходным кодом. MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

1.2.3.2 Основные возможности

Основные возможности СУБД MySQL является:

1) очень быстрые дисковые таблицы на основе B-деревьев со сжатием индексов,

2) простота в работе — установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД,

3) хеш-таблицы в памяти, используемые как временные таблицы,

4) безопасность — большое количество функций обеспечивающих безопасность, которые поддерживается по умолчанию,

5) скорость — упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность,

6) масштабируемость — MySQL легко работает с большими объемами данных и легко масштабируется.

1.2.3.3 Типы данных

Основные типы данных СУБД MySQL отображены в таблице 1.4.

Таблица 1.4 — типы данных

Символьные типы	Char, varchar…
Числовые типы	Int, decimal, float, double…
Типы для работы с датой и временем	Date, time, datetime, year, timestamp
Составные типы	Enum, set
Бинарные типы	tinyblob, blob, mediumblom, largeblom

1.2.3.4 Язык запросов

Основные операторы, которые используются в запросе представлены в таблице 1.5.

Таблица 1.5 — основные операторы

Команда	**Описание**
CREATE	Создает новую таблицу, представление таблицы или другой объект в БД

Продолжение таблицы 1.5

Команда	**Описание**
ALTER	Модифицирует существующий в БД объект, такой как таблица
DROP	Удаляет существующую таблицу, представление таблицы или другой объект в БД
SELECT	Извлекает записи из одной или нескольких таблиц
INSERT	Создает записи
UPDATE	Модифицирует записи
DELETE	Удаляет записи
GRANT	Наделяет пользователя правами
REVOKE	Отменяет права пользователя

Ниже представлен пример использования запроса(Рисунок 1.3).

```
SELECT `phone_book` FROM `test` ORDER BY `surname`;

INSERT INTO `phone_book` (`id`, `surname`, `name`, `patronymic`, `phone`)
VALUES ('1', 'Абрамов', 'Максим', 'Викторович', '555-55-55');

UPDATE `phone_book`
SET `surname` = 'Абрамович'
WHERE `id`=1;

DELETE FROM `phone_book`
WHERE `id`=1;
```

Рисунок 1.3 — пример запроса

1.3 Обоснование выбора СУБД

База данных обеспечивает хранение информации и представляет собой совокупность данных, организованных по определенным правилам. БД позволяет структурировать, хранить и обрабатывать данные различного типа.

Система управления базами данных (СУБД) — это совокупность языковых и программных средств, предназначенных для создания, ведения и использования БД.

Эффективность функционирования системы, использующей БД, зависит как от выбора архитектуры БД, так и от выбора СУБД. При выборе были рассмотрены следующие СУБД: MS SQL, MySQL и PostgreSQL. Рассмотрим достоинства и недостатки этих СУБД.

MS SQL — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase.

Достоинства:

1) прост в использовании,

2) движок предоставляет возможность регулировать и отслеживать уровни производительности, которые помогают снизить использование ресурсов,

3) масштабируемость и надежность,

4) возможность обработки вычислений в оперативной памяти,

Недостатки:

1) Цена для юридических лиц оказывается неприемлемой для большей части организаций,

2) даже при тщательной настройке производительности корпорация SQL Server способен занять все доступные ресурсы,

3) проблемы с использованием службы интеграции для импорта файлов.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB.

Достоинства:

1) легко использовать,

2) предоставляет большой функционал,

3) безопасность (включает большое количество функций для обеспечения безопасности, причем они поддерживаются по умолчанию),

4) легко масштабируется и подходит для больших баз данных,

5) обеспечивает хорошую скорость и производительность,

6) обеспечивает хорошее управление пользователями и множественный контроль доступа,

Недостатки:

1) наличие ограничений функционала (имеет большинство возможностей SQL, но не все, а иногда они требуются для работы в особо «капризных» приложениях),

2) платную поддержку даже для бесплатной версии,

3) низкая скорость разработки.

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

Достоинства:

1) полная SQL-совместимость,

2) расширяемость. PostgreSQL можно программно расширить за счёт хранимых процедур,

3) объектно-ориентированность. PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД,

4) поддержка БД неограниченного размера,

Недостатки:

1) планы выполнения запросов не кэшируются,

2) производительность: В простых операциях чтения PostgreSQL может уступать своим соперникам,

3) популярность: из-за своей сложности инструмент не очень популярен.

Для дальнейшего выбора СУБД рассмотрим таблицу 1.

Таблица 1 — критерии выбора СУБД.

Критерии	**MS SQL**	**MySQL**	**PostgreSQL**
Производительность	+	+	-
Безопасность	+/-	+	+/-
Мобильность	-	+	+
Масштабируемость	+	+	+
Распространенность СУБД	+	+	+/-
Лёгкость использования	+	+/-	+/-
Качество и полнота документации	+	+	-

Исходя из таблицы 1 может сделать вывод, что СУБД MySQL будет лучше подходит для создания базы данных фотографии.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Формирование требований

2.1.1 Основные сущности базы данных

Основные сущности и их описания отображены на рисунке 2.1.

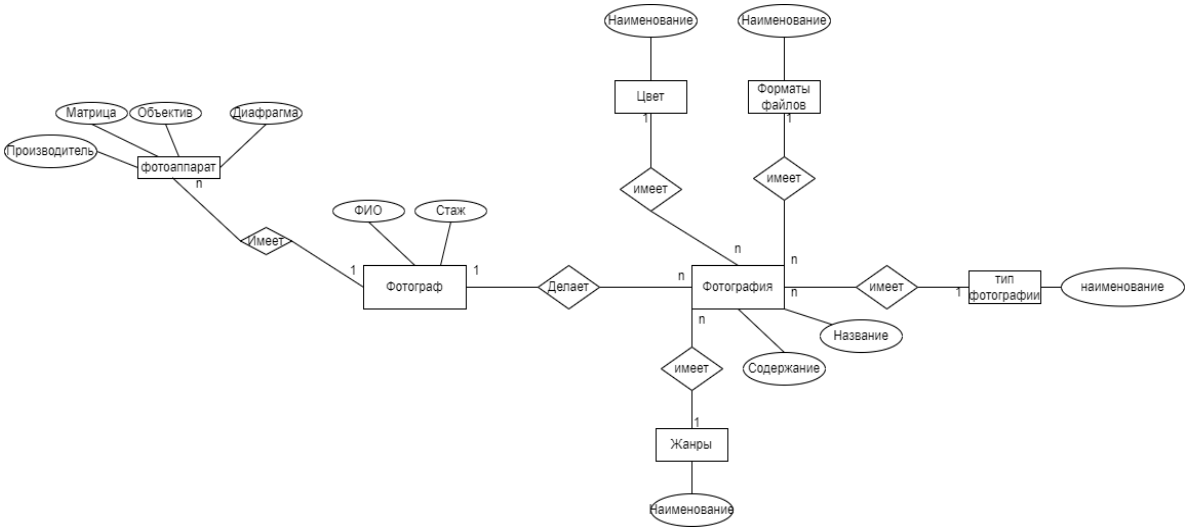


Рисунок 2.4 — диаграмма сущностей

Основные характеристики сущности тип фотографии: плёночная, цифровая, электрографическая.

Основные характеристики сущности цвет: черно-белая, цветная.

Основные характеристики сущности форматы файлов: RAW, JPEG, TIFF, DNG.

Основные характеристики сущности жанры: пейзаж, портретное фото, рекламное фото, уличное фото, репортаж.

2.1.2 Варианты использования

Основные сценарии отображены на рисунке 2.

Рисунок 2.5 — UML диаграмма

2.1.3 Определение API для взаимодействия с базой данных

Основные методы взаимодействия представлены в таблице 2.1.

Таблица 2.1 — описание методов

Название метода	Описание метода
Добавить фотографа	Аргументы: ФИО(строка),стаж(временная петля)
Изменить данные фотографа	Аргументы: код(число), ФИО(строка), новое ФИО(строка) стаж(временная петля), новый стаж(временная петля)
Изменить данные фотографа(ФИО)	Аргументы: код(число), ФИО(строка), новое ФИО(строка)
Изменить данные фотографа(стаж)	Аргументы: код(число), ФИО(строка), стаж(временная петля), новый стаж(временная петля)
Удалить фотографа	Аргументы: код(число)
Просмотр информации о фотографа	Аргументы: код(число), ФИО(строка), стаж(временная петля)
Добавить фотоаппарат	Аргументы: название(строка) производитель(строка), матрица(строка), объектив(строка), диафрагма(строка).
Изменить фотоаппарат	Аргументы: код фотоаппарата(число) название(строка) производитель(строка), матрица(строка), объектив(строка), диафрагма(строка).
Удалить фотоаппарат	Аргументы: код фотоаппарата(число)
Просмотр фотоаппарата	Аргументы: код фотоаппарата(число) название(строка) производитель(строка), матрица(строка), объектив(строка), диафрагма(строка).
Изменить Матрицу фотоаппарата	Аргументы: код фотоаппарата(число), новое название матрицы(строка)
Изменить Название фотоаппарата	Аргументы: код фото(число), новое название(строка)

Продолжение таблицы 2.1

Название метода	Описание метода
Изменить производитель фотоаппарата	Аргументы: код фото(число), новый производитель(строка)
Изменить объектив фотоаппарата	Аргументы: код фото(число), новое название объектива(строка)
Изменить диафрагму фотоаппарата	Аргументы: код фото(число), новое название диафрагмы(строка)
Добавить фотографию	Аргументы: название(строка), содержание(), жанр(строка), цвет(строка), формат файла(строка), тип фото(строка).
Удалить фотографию	Аргументы: код фото(число)
Изменить фотографию	Аргументы: код фото(число), ноыое название(строка), новое содержание(), новый жанр(строка), новый цвет(строка), новый формат файла(строка), новый тип фото(строка).
Вывести фотографию	Аргументы: код фото(число), название(строка), содержание(), жанр(строка), цвет(строка), формат файла(строка), тип фото(строка).
Изменить название фото	Аргументы: код фото(число), новое название фото(строка)
Изменить жанр фотографии	Аргументы: код фотографии(число), новое название жанра(строка)
Изменить формат фото	Аргументы: код фото(число), новый жанр(строка)
Изменить тип фото	Аргументы: код фото(число), новый тип фото(строка)

2.2 Проектирование базы данных

2.2.1 Определение таблиц

Определение основных полей, типы данных и ограничения представлены в таблицах:2.2, 2.3, 2.4, 2.5, 2.6, 2.7.

Таблица 2.2 — photos

Название полей	**Типы данных**	**Ограничения**
id_photo	INT	AI,NN,PK
name_photo	VARCHAR(50)	NN
content	BLOB	NN
size	VARCHAR(25)	NN
color	VARCHAR(10)	NN
id_format	INT	NN,FK
id_type	INT	NN,FK
id_genre	INT	NN,FK
id_ ptgrapher	INT	NN,FK

Таблица 2.3 — formats

Название полей	**Типы данных**	**Ограничения**
id_format	INT	AI,NN,PK
name_format	VARCHAR(10)	NN

Таблица 2.4 — types

Название полей	**Типы данных**	**Ограничения**
id_type	INT	AI,NN,PK
name_type	VARCHAR(15)	NN

Таблица 2.5 — genres

Название полей	**Типы данных**	**Ограничения**
Id_genre	INT	AI,NN,PK
name_genre	VARCHAR(50)	NN

Таблица 2.6 — ptgrapher

Название полей	**Типы данных**	**Ограничения**
id_ptgrapher	INT	AI,NN,PK
first_name	VARCHAR(30)	NN
last_name	VARCHAR(30)	NN

Продолжение таблицы 2.6

Название полей	**Типы данных**	**Ограничения**
middle_name	VARCHAR(30)	NN
work_exp	INT	NN
id_camera	INT	NN,FK

Таблица 2.7 — camera

Название полей	**Типы данных**	**Ограничения**
id_camera	INT	AI,NN,PK
maker	VARCHAR(50)	NN
model	VARCHAR(50)	NN
matrix	VARCHAR(50)	NN
lens	VARCHAR(50)	NN

2.2.1 Логическая схема базы данных

Логическая схема базы данных представлена на рисунке 2.3.

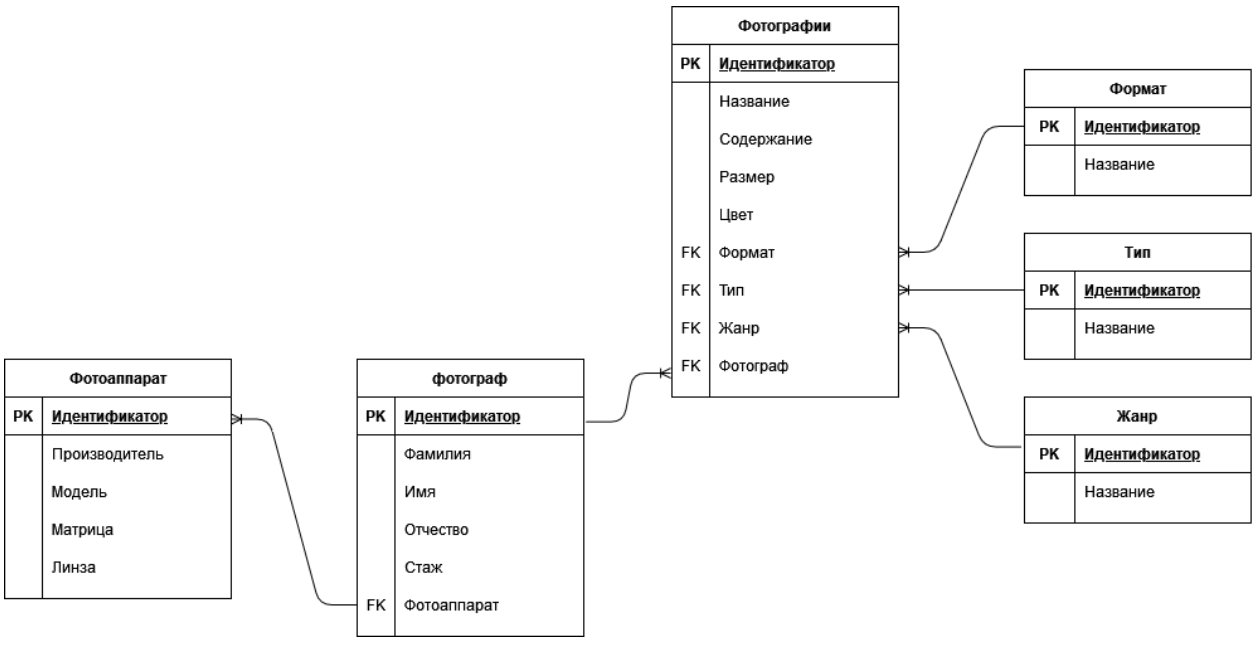


Рисунок 2.3 — логическая модель

2.2.2 Физическая схема базы данных

Физическая схема базы данных представлена на рисунке 2.4.

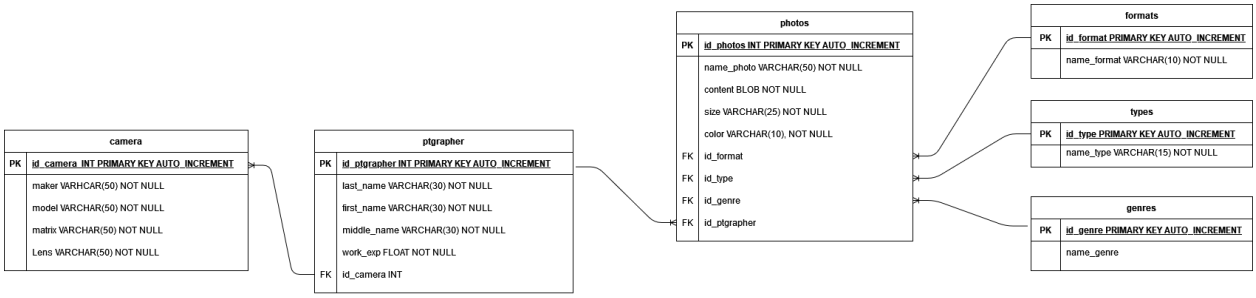


Рисунок 6.4 — физическая модель

2.2.3 Определение типовых запросов к объектам базы данных

Основные запросы к таблице camera представлены в таблице 2.8.

Таблица 2.8 — запросы к таблице camera.

Запрос	Описание
SELECT * FROM camera	Выводит информацию о фотоаппарате
INSERT INTO camera(maker, model, matrix, lens) VALUES(`Nikon`,`D750`,` КМОП`,` AF-S NIKKOR 500mm f/5.6E PF ED VR`)	Добавляет в фотоаппарат в таблицу камера
UPDATE camera set maker=`Nikon` WHERE id=3	Обновляет запись
DELETE FROM camera where id=2	Удаляет данные о фотоаппарате

Основные запросы к таблице ptgrapher представлены в таблице 2.9.

Таблица 2.9 — запросы к таблице ptgrapher

Запрос	Описание
SELECT *FROM ptgrapher WHERE id=5	Выводит информацию о фотографе
INSERT INTO ptgrapher(last_name, first_name, middle_name, work_exp, id_camera) VALUES (`Горшков`,`Сергей`,`Владимирович`,10, 4,5)	Добавляет нового фотографа

Продолжение таблицы 2.9

Запрос	Описание
UPDATE ptgrapher set work_exp=3 WHERE id=4	Обновляет фотографии выбранного фотографа
DELETE FROM ptgrapher WHERE id=5	Удаляет выбранного фотографа

Основные запросы к таблице photos представлены в таблице 2.10.

Таблица 2.10 — запросы к таблице photos

Запрос	Описание
SELECT * FROM photos	Выводит информацию о фотографии
INSERT INTO photo(name_photo,content, size, color, id_format, id_type, id_genre, id_ ptgrapher) VALUES(`cat`,`cat.png`,`34,5кб`, `Цветная`,1,1,7,3)	Добавляет фотографию в таблицу photos
UPDATE photos set name_photo=’dogs’ WHERE id=2	Обновляет запись
DELETE FROM photo WHERE id=3	Удаляет фотографию

2.2.4 Определение процедур и функций API

Основные функции отображены в таблице 2.11 .

Таблица 2.11 — основные функции

Название функции	Описание функции
add_ptgrapher(first_name varchar(50), last_name varchar(50), middle_name varchar(50), work_exp float(11))	Добавляет фотографа в базу данных.
change_attOfptgrapherfirst_name (id int(11),first_name varchar(50))	Изменяет имя фотографа.
change_attOfptgrapherlast_name (id int,last_name varchar(50))	Изменяет фамилию фотографа.
change_attOfptgraphermiddle_name (id int,last_name varchar(50))	Изменяет Отчество фотографа.

Продолжение таблицы 2.12

Название функции	Описание функции
change_attOfptgrapherwork_exp (id int,work_exp float)	Изменяет стаж фотографа.
del_ ptgrapher(id_ptgrapher int(11))	Удаляет фотографа по идентификатору.
get_information_ptgrapher(id int)	Возвращает данные о фотографе.
add_camera(maker varchar(50), model varchar(25), matrix varchar(50), lens varchar(50))	Добавляет фотоаппарат в базу данных.
change_att_camera_maker (id int, maker varchar(50))	Изменяет производителя фотоаппарата.
change_att_camera_model (id int, model varchar(50))	Изменяет модель фотоаппарата.
change_att_camera_matrix (id int, matrix varchar(50))	Изменяет матрицу фотоаппарата.
change_att_camera_lens (id int, lens varchar(50))	Изменяет линзу фотоаппарата.
del_camera(id_camera int(11))	Удаляет выбранный фотоаппарат.
get_information_camera(id_camera int(11))	Возвращает данные о фотоаппарате.
add_photo(name_photo varchar(50), content blob, size varchar(25), color varchar(10), id_format int(11), id_type int(11), id_genre int(11))	Добавляет фотографию в базу данных.
del_photo(id_photo int(11))	Удаляет фотографию.
get_information_photo(id int)	Возвращает данные о фотографии.
change_att_photo_name(id int, name_photo varchar(50))	Изменяет название фотографии
change_att_photo_content(id int, content blob)	Изменение содержимого фотографии
change_att_photo_size(id int, size varchar(25))	Изменяет размер фотографии
change_att_photo_color(id int, color varchar(30))	Изменяет цвет фотографии
change_att_photo_format(id int, id_format int))	Изменяет формат фотографии
change_att_photo_type(id int, id_type int)	Изменяет тип фотографии

2.3 Разработка базы данных

2.3.1 Разработанные таблицы

Разработанные таблицы представлены на рисунке 2.5.

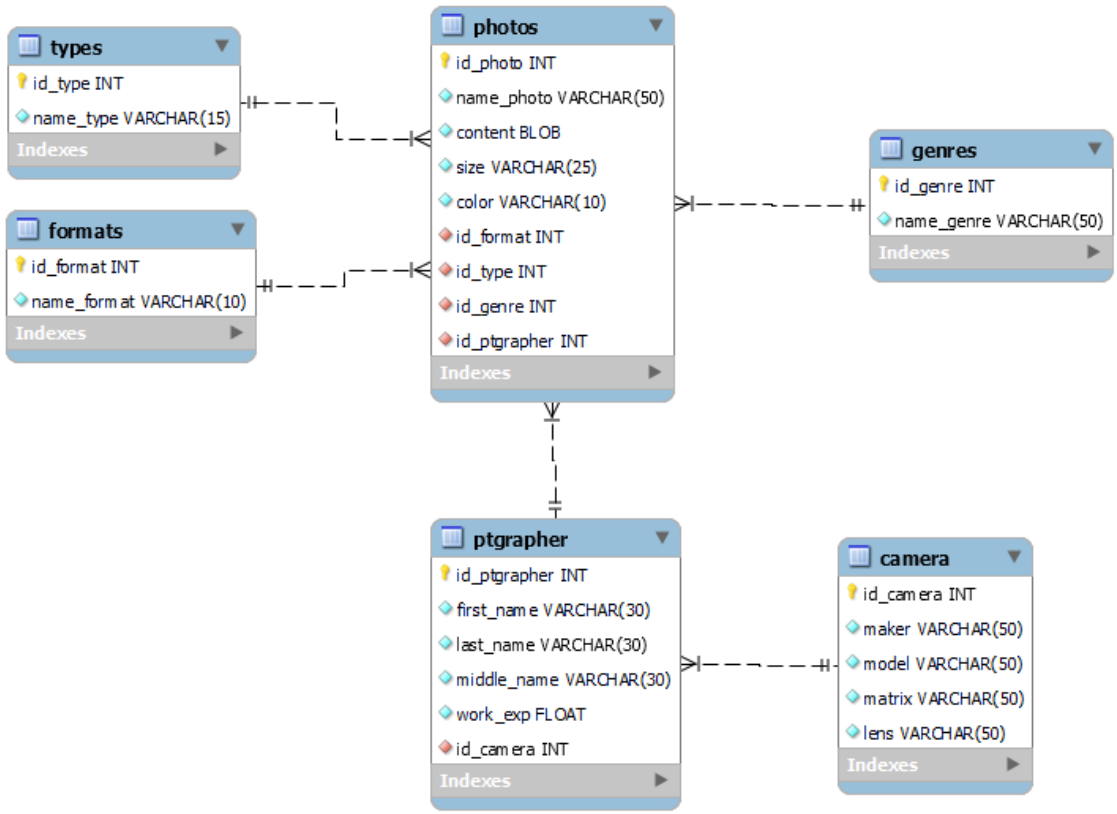
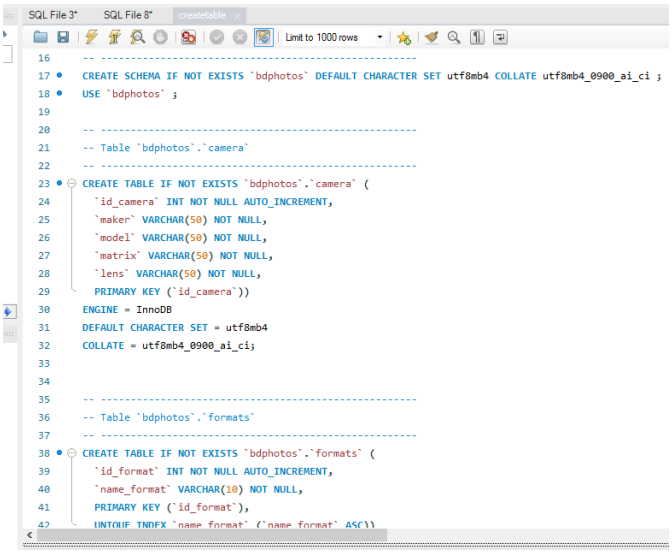


Рисунок 2.5 — модель сущность-связь

Примеры разработки таблиц показаны на рисунке 2.6 и на рисунке 2.7. Полный код отображен в приложении А в таблице А1.



```
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bdphotos` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;
USE `bdphotos` ;

-- -----------------------------------------------------
-- Table `bdphotos`.`camera`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`camera` (
  `id_camera` INT NOT NULL AUTO_INCREMENT,
  `maker` VARCHAR(50) NOT NULL,
  `model` VARCHAR(50) NOT NULL,
  `matrix` VARCHAR(50) NOT NULL,
  `lens` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id_camera`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `bdphotos`.`formats`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`formats` (
  `id_format` INT NOT NULL AUTO_INCREMENT,
  `name_format` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`id_format`),
  UNIQUE INDEX `name_format` (`name_format` ASC))
```

Рисунок 2.6 — создание таблицы camera

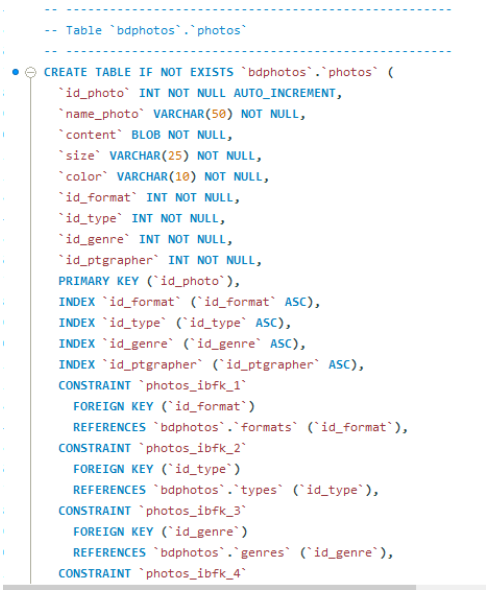


Рисунок 2.7 — создание таблицы photos

2.3.2 Разработанные представления

Создание представления отображен на рисунке 2.8. Полный код находится в приложении А в таблице А3.

Рисунок 2.8 — создания представления

2.3.3 Разработанные процедуры и функции

Пример создания процедур отображен на рисунке 2.9. Полный код находится в приложении А в таблице А2.

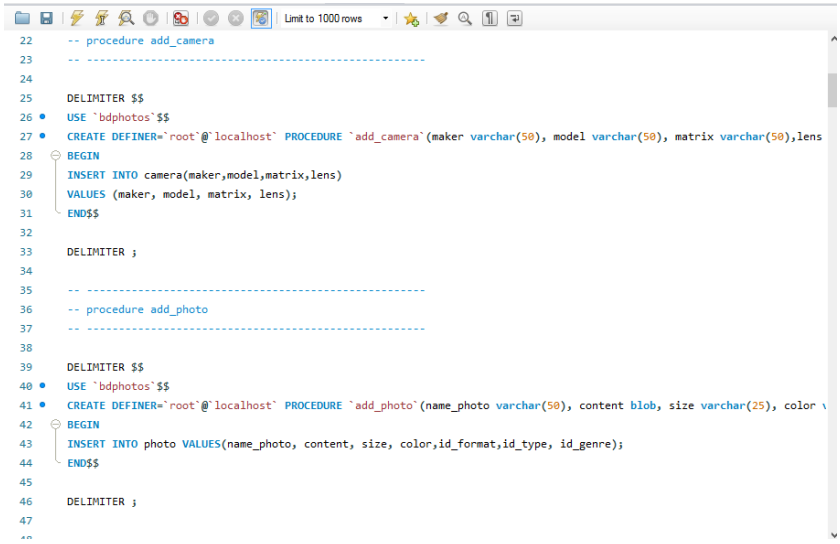


Рисунок 2.9 — создание процедур

3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

(Не трогайте, вам потом самим скажут что сюда вставлять)

ЗАКЛЮЧЕНИЕ

В результате выполнения ВКР была разработана база данных фотографии.

В процессе выполнения ВКР были выполнены следующие задачи:

1) был произведен анализ предметной области,

2) был произведен анализ существующих баз данных,

3) был произведен анализ схемы базы данных,

4) была спроектирована база данных,

5) была разработана база данных.

Достоинства разработанной базы данных:

1) удобный поиск фотографий,

2) быстрота поиска,

Недостатки разработанной базы данных:

1) отсутствие пользователей и ролей,

2) слабая защита данных.

СПИСОК ЛИТЕРАТУРЫ

1) MySQL документация [Электронный ресурс]. — URL: https://dev.mysql.com/doc/ (дата обращения: 17.11.2021);

2) Основная информация об MySQL [Электронный ресурс]. — URL: https://www.opennet.ru/docs/RUS/mysqlcli/glava01.html#Features (дата обращения: 17.11.2021);

3) Документация MongoDB [Электронный ресурс]. — URL: https://docs.mongodb.com/ (дата обращения: 17.11.2021);

4) Синтаксис запросов MongoDB [Электронный ресурс]. — URL: https://salatpower.ru/?p=5 (дата обращения: 17.11.2021);

5) Основная информация об neo4j [Электронный ресурс]. — URL: https://evilinside.ru/neo4j/# (дата обращения: 17.11.2021);

6) Документация neo4j [Электронный ресурс]. — URL: https://neo4j.com/docs/ (дата обращения: 17.11.2021);

7) Фотография [Электронный ресурс]. — URL: 1) https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F (дата обращения: 18.11.2021);

8) Фотоаппарат [Электронный ресурс]. — URL: 2) https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%82%D0%BE%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82#%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D1%8B%D0%B5_%D1%84%D0%BE%D1%82%D0%BE%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D1%8B (дата обращения: 18.11.2021);

9) Фотография [Электронный ресурс]. — URL: 3) https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F (дата обращения: 18.11.2021);

10) Цифровой фотоаппарат: устройство, типы цифровых камер [Электронный ресурс]. — URL: 1) https://www.kp.ru/guide/tsifrovye-fotoapparaty.html (дата обращения: 18.11.2021);

11) Цифровой фотоаппарат: устройство, типы цифровых камер [Электронный ресурс]. — URL: 1) https://www.kp.ru/guide/tsifrovye-fotoapparaty.html (дата обращения: 18.11.2021);

12) Ситифорум. [Электронный ресурс]. — URL: http://citforum.ru/database/articles/criteria/ (дата обращения: 22.11.2021);

13) PostgreSQL или MySQL [Электронный ресурс]. — URL: https://mcs.mail.ru/blog/postgresql-ili-mysql-kakaya-iz-etih-relyacionnyh-subd (дата обращения: 22.11.2021);

14) Сравнение современных СУБД [Электронный ресурс]. — URL: https://drach.pro/blog/hi-tech/item/145-db-comparison (дата обращения: 22.11.2021);

15) ТОП-10 систем управления базами данных в 2019 году [Электронный ресурс]. — URL: https://proglib.io/p/databases-2019 (дата обращения: 22.11.2021);

16) Критерии выбора СУБД при создании информационных систем [Электронный ресурс]. — URL: https://www.internet-technologies.ru/articles/kriterii-vybora-subd-pri-sozdanii-informacionnyh-sistem.html (дата обращения: 22.11.2021);

17) Создание диаграмм [Электронный ресурс]. — URL: https://app.diagrams.net/ (дата обращения: 25.11.2021).

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ОБЪЕКТОВ БАЗЫ ДАННЫХ

Таблица А.1 – Исходный код создания таблиц

example_base.sql

```
-- MySQL Workbench Forward Engineering

SET @OLD UNIQUE CHECKS=@@UNIQUE CHECKS, UNIQUE CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL MODE='ONLY FULL GROUP BY,STRICT TRANS TABLES,NO ZERO IN DATE,NO ZERO DATE,ERROR FOR DIVISION
BY ZERO,NO ENGINE SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema bdphotos
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema bdphotos
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bdphotos` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4 0900 ai ci ;
USE `bdphotos` ;

-- -----------------------------------------------------
-- Table `bdphotos`.`camera`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`camera` (
  `id_camera` INT NOT NULL AUTO_INCREMENT,
  `maker` VARCHAR(50) NOT NULL,
  `model` VARCHAR(50) NOT NULL,
  `matrix` VARCHAR(50) NOT NULL,
  `lens` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id_camera`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4 0900 ai ci;


-- -----------------------------------------------------
-- Table `bdphotos`.`formats`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`formats` (
  `id format` INT NOT NULL AUTO INCREMENT,
  `name format` VARCHAR(10) NOT NULL,
  PRIMARY KEY (`id_format`),
  UNIQUE INDEX `name_format` (`name_format` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4 0900 ai ci;


-- -----------------------------------------------------
-- Table `bdphotos`.`genres`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`genres` (
  `id_genre` INT NOT NULL AUTO_INCREMENT,
  `name_genre` VARCHAR(50) NOT NULL,
  PRIMARY KEY (`id genre`),
  UNIQUE INDEX `name genre` (`name genre` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `bdphotos`.`types`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`types` (
  `id_type` INT NOT NULL AUTO_INCREMENT,
  `name type` VARCHAR(15) NOT NULL,
  PRIMARY KEY (`id type`),
  UNIQUE INDEX `name type` (`name type` ASC))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

```
-- -----------------------------------------------------
-- Table `bdphotos`.`ptgrapher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`ptgrapher` (
  `id ptgrapher` INT NOT NULL AUTO INCREMENT,
  `first_name` VARCHAR(30) NOT NULL,
  `last_name` VARCHAR(30) NOT NULL,
  `middle name` VARCHAR(30) NOT NULL,
  `work exp` FLOAT NOT NULL,
  `id camera` INT NOT NULL,
  PRIMARY KEY (`id ptgrapher`),
  INDEX `id_camera` (`id_camera` ASC),
  CONSTRAINT `ptgrapher_ibfk_1`
    FOREIGN KEY (`id camera`)
    REFERENCES `bdphotos`.`camera` (`id camera`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;


-- -----------------------------------------------------
-- Table `bdphotos`.`photos`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`photos` (
  `id photo` INT NOT NULL AUTO INCREMENT,
  `name photo` VARCHAR(50) NOT NULL,
  `content` BLOB NOT NULL,
  `size` VARCHAR(25) NOT NULL,
  `color` VARCHAR(10) NOT NULL,
  `id format` INT NOT NULL,
  `id type` INT NOT NULL,
  `id genre` INT NOT NULL,
  `id ptgrapher` INT NOT NULL,
  PRIMARY KEY (`id_photo`),
  INDEX `id_format` (`id_format` ASC),
  INDEX `id type` (`id type` ASC),
  INDEX `id genre` (`id genre` ASC),
  INDEX `id ptgrapher` (`id ptgrapher` ASC),
  CONSTRAINT `photos ibfk 1`
    FOREIGN KEY (`id_format`)
    REFERENCES `bdphotos`.`formats` (`id_format`),
  CONSTRAINT `photos ibfk 2`
    FOREIGN KEY (`id type`)
    REFERENCES `bdphotos`.`types` (`id type`),
  CONSTRAINT `photos ibfk 3`
    FOREIGN KEY (`id_genre`)
    REFERENCES `bdphotos`.`genres` (`id_genre`),
  CONSTRAINT `photos ibfk 4`
    FOREIGN KEY (`id ptgrapher`)
    REFERENCES `bdphotos`.`ptgrapher` (`id ptgrapher`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4 0900 ai ci;


SET SQL MODE=@OLD SQL MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Таблица А.2 – Исходный код создания хранимых процедур

example_procedures.sql

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD FOREIGN KEY CHECKS=@@FOREIGN KEY CHECKS, FOREIGN KEY CHECKS=0;
SET @OLD SQL MODE=@@SQL MODE,
SQL MODE='ONLY FULL GROUP BY,STRICT TRANS TABLES,NO ZERO IN DATE,NO ZERO DATE,ERROR FOR DIVISION
BY_ZERO,NO_ENGINE_SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema bdphotos
```

```
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema bdphotos
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bdphotos` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4 0900 ai ci ;
USE `bdphotos` ;
USE `bdphotos` ;

-- -----------------------------------------------------
-- procedure add camera
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `add camera`(maker varchar(50), model varchar(50),
matrix varchar(50),lens varchar(50))
BEGIN
INSERT INTO camera(maker,model,matrix,lens)
VALUES (maker, model, matrix, lens);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure add_photo
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `add_photo`(name_photo varchar(50), content blob,
size varchar(25), color varchar(10),id format int,id type int, id genre int)
BEGIN
INSERT INTO photo VALUES(name photo, content, size, color,id format,id type, id genre);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure add ptgrapher
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `add ptgrapher`(first name varchar(50), last name
varchar(50), middle name varchar(50), work exp float)
BEGIN
iNSERT INTO ptgrapher(first_name, last_name, middle_name, work_exp)
VALUES (first_name, last_name, middle_name, work_exp);
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change attOfptgrapher
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_attOfptgrapher`(id int,first_name
varchar(50), last name varchar(50), middle name varchar(50), work exp float)
BEGIN
UPDATE ptgrapher
set first name=first name,
last_name=last_name,
middle_name=middle_name,
work exp=worrk exp
WHERE id ptgrapher=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change attOfptgrapherfirst name
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change attOfptgrapherfirst name`(id int,first name
varchar(50))
```

```
BEGIN
UPDATE ptgrapher
set first_name=first_name
WHERE id_ptgrapher=id;
END$$

DELIMITER ;

-- ----------------------------------------------------
-- procedure change attOfptgrapherlast name
-- ----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_attOfptgrapherlast_name`(id int,last_name
varchar(50))
BEGIN
UPDATE ptgrapher
set last_name=last_name
WHERE id_ptgrapher=id;
END$$

DELIMITER ;

-- ----------------------------------------------------
-- procedure change_attOfptgraphermiddle_name
-- ----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_attOfptgraphermiddle_name`(id int,last_name
varchar(50))
BEGIN
UPDATE ptgrapher
set middle name=middle name
WHERE id_ptgrapher=id;
END$$

DELIMITER ;

-- ----------------------------------------------------
-- procedure change_attOfptgrapherwork_exp
-- ----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change attOfptgrapherwork exp`(id int,work exp
float)
BEGIN
UPDATE ptgrapher
set work exp=work exp
WHERE id ptgrapher=id;
END$$

DELIMITER ;

-- ----------------------------------------------------
-- procedure change att camera lens
-- ----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att camera lens`(id int, lens varchar(50))
BEGIN
UPDATE camera set lens=lens WHERE id_camera=id;
END$$

DELIMITER ;

-- ----------------------------------------------------
-- procedure change_att_camera_maker
-- ----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_att_camera_maker`(id int, maker varchar(50))
BEGIN
UPDATE camera set maker=maker WHERE id camera=id;
END$$
```

```
DELIMITER ;

-- -----------------------------------------------------
-- procedure change_att_camera_matrix
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_att_camera_matrix`(id int, matrix
varchar(50))
BEGIN
UPDATE camera set matrix=matrix WHERE id camera=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change att camera model
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att camera model`(id int, model varchar(50))
BEGIN
UPDATE camera set model=model WHERE id_camera=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change_att_photo_color
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_att_photo_color`(id int, color varchar(30))
BEGIN
UPDATE photos set color=color WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change att photo content
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_att_photo_content`(id int, content blob)
BEGIN
UPDATE photos set content=content WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change att photo format
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att photo format`(id int, id format int)
BEGIN
UPDATE photos set id format=id format WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change att photo genre
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att photo genre`(id int, id genre int)
BEGIN
UPDATE photos set id_genre=id_genre WHERE id_photo=id;
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure change_att_photo_name
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change_att_photo_name`(id int, name_photo
varchar(50))
BEGIN
UPDATE photos set name photo=name photo WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change att photo size
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att photo size`(id int, size varchar(25))
BEGIN
UPDATE photos set size=size WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure change_att_photo_type
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `change att photo type`(id int, id type int)
BEGIN
UPDATE photos set id_type=id_type WHERE id_photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure del_camera
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `del_camera`(id int)
BEGIN
DELETE FROM camera  WHERE id camera=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure del photo
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `del photo`(id int)
BEGIN
DELETE FROM photo WHERE id photo=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure del ptgrapher
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `del ptgrapher`(id int)
BEGIN
DELETE FROM ptgrapher WHERE id_ptgrapher=id;
END$$

DELIMITER ;
```

```
-- -----------------------------------------------------
-- procedure get information camera
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get information camera`(id int)
BEGIN
SELECT maker, model, matrix, lens FROM camera WHERE id_camera=id;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure get_information_photo
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_information_photo`(id int)
BEGIN
SELECT p.name photo, p.content, p.size, p.color, f.id format,t.id type, g.id genre
FROM photos AS p INNER JOIN formats AS f on f.id format=p.id format
INNER JOIN types AS t on t.id type=p.id type
INNER JOIN genres AS g on g.id_genre=p.id_genre;
END$$

DELIMITER ;

-- -----------------------------------------------------
-- procedure get_information_ptgrapher
-- -----------------------------------------------------

DELIMITER $$
USE `bdphotos`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `get_information_ptgrapher`(id int)
BEGIN
SELECT first name, last name, middle name, work exp WHERE id ptgrapher=id;
END$$

DELIMITER ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Таблица А.3 – Исходный код создания представлений

example_views.sql

```
-- MySQL Workbench Forward Engineering

SET @OLD UNIQUE CHECKS=@@UNIQUE CHECKS, UNIQUE CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE,
SQL MODE='ONLY FULL GROUP BY,STRICT TRANS TABLES,NO ZERO IN DATE,NO ZERO DATE,ERROR FOR DIVISION
BY ZERO,NO ENGINE SUBSTITUTION';

-- -----------------------------------------------------
-- Schema mydb
-- -----------------------------------------------------
-- -----------------------------------------------------
-- Schema bdphotos
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema bdphotos
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `bdphotos` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4 0900 ai ci ;
USE `bdphotos` ;
USE `bdphotos` ;

-- -----------------------------------------------------
-- Placeholder table for view `bdphotos`.`information about photo`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`information_about_photo` (`name_photo` INT, `content` INT,
`size` INT, `color` INT, `name_format` INT, `name_type` INT, `name_genre` INT, `last_name` INT);
```

```
-- -----------------------------------------------------
-- Placeholder table for view `bdphotos`.`information about ptgrapher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `bdphotos`.`information_about_ptgrapher` (`last_name` INT,
`first name` INT, `middle name` INT, `work exp` INT, `maker` INT, `model` INT);

-- -----------------------------------------------------
-- View `bdphotos`.`information_about_photo`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `bdphotos`.`information about photo`;
USE `bdphotos`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW
`bdphotos`.`information about photo` AS select `p`.`name photo` AS `name photo`,`p`.`content` AS
`content`,`p`.`size` AS `size`,`p`.`color` AS `color`,`f`.`name_format` AS
`name_format`,`t`.`name_type` AS `name_type`,`g`.`name_genre` AS `name_genre`,`ptg`.`last_name`
AS `last name` from ((((`bdphotos`.`photos` `p` join `bdphotos`.`formats` `f` on((`p`.`id format`
= `f`.`id format`))) join `bdphotos`.`types` `t` on((`t`.`id type` = `p`.`id type`))) join
`bdphotos`.`genres` `g` on((`g`.`id genre` = `p`.`id genre`))) join `bdphotos`.`ptgrapher` `ptg`
on((`ptg`.`id_ptgrapher` = `p`.`id_ptgrapher`)));

-- -----------------------------------------------------
-- View `bdphotos`.`information about ptgrapher`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `bdphotos`.`information about ptgrapher`;
USE `bdphotos`;
CREATE  OR REPLACE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW
`bdphotos`.`information about ptgrapher` AS select `ptg`.`last name` AS
`last name`,`ptg`.`first name` AS `first name`,`ptg`.`middle name` AS
`middle name`,`ptg`.`work exp` AS `work exp`,`c`.`maker` AS `maker`,`c`.`model` AS `model` from
(`bdphotos`.`ptgrapher` `ptg` join `bdphotos`.`camera` `c` on((`ptg`.`id_camera` =
`c`.`id_camera`)));

SET SQL MODE=@OLD SQL MODE;
SET FOREIGN KEY CHECKS=@OLD FOREIGN KEY CHECKS;
SET UNIQUE CHECKS=@OLD UNIQUE CHECKS;
```