**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc88816885)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 4](#_Toc88816886)

[1.1 Исследование предметной области 4](#_Toc88816887)

[1.2 Анализ существующих СУБД 4](#_Toc88816888)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 5](#_Toc88816889)

[2.1 Формирование требований 5](#_Toc88816890)

[2.2 Проектирование базы данных 5](#_Toc88816891)

[2.3 Разработка базы данных 6](#_Toc88816892)

[3 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ 8](#_Toc88816893)

[ЗАКЛЮЧЕНИЕ 9](#_Toc88816894)

[СПИСОК ЛИТЕРАТУРЫ 10](#_Toc88816895)

[ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ОБЪЕКТОВ БАЗЫ ДАННЫХ 11](#_Toc88816896)

# 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 х 13;  2) 10 х 15;  3) 13 х 18;  4) 15 х 21;  5) 20 х 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.

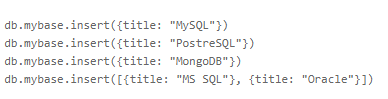


Рисунок 1. — вставка нового документа

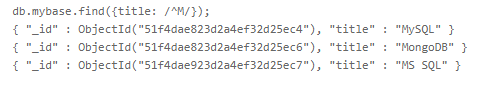


Рисунок 1. — регулярные выражения

**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 запросовCQL 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 В-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).

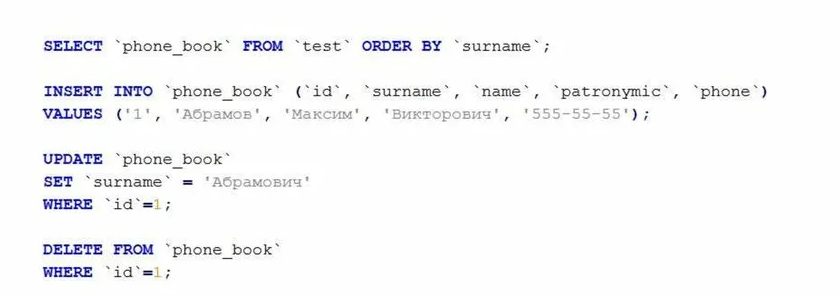


Рисунок 1. — пример запроса

**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.

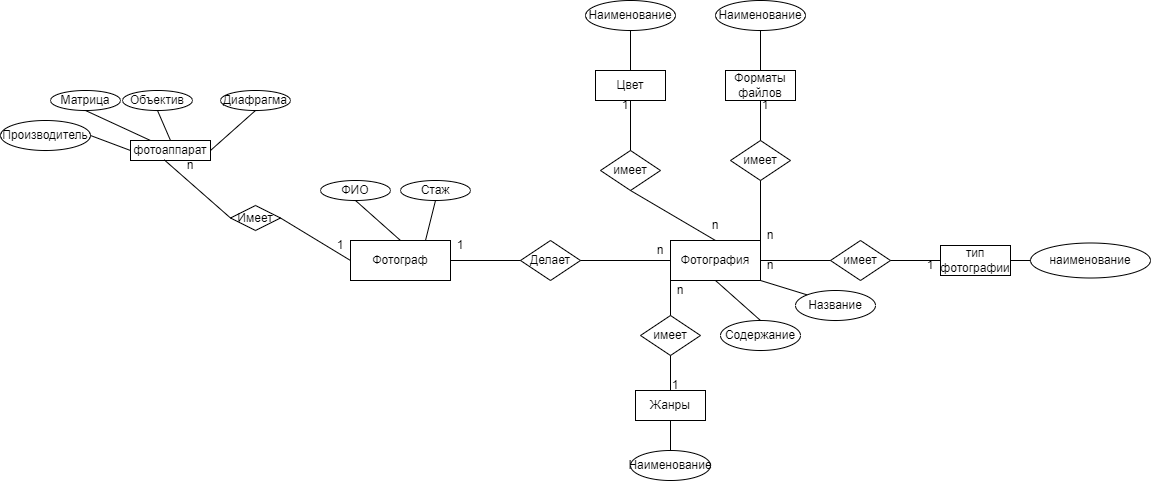


Рисунок 2. — диаграмма сущностей

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.

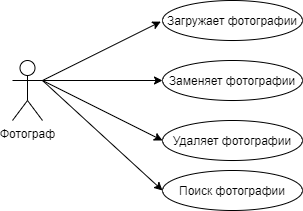
****

Рисунок 2. — 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.

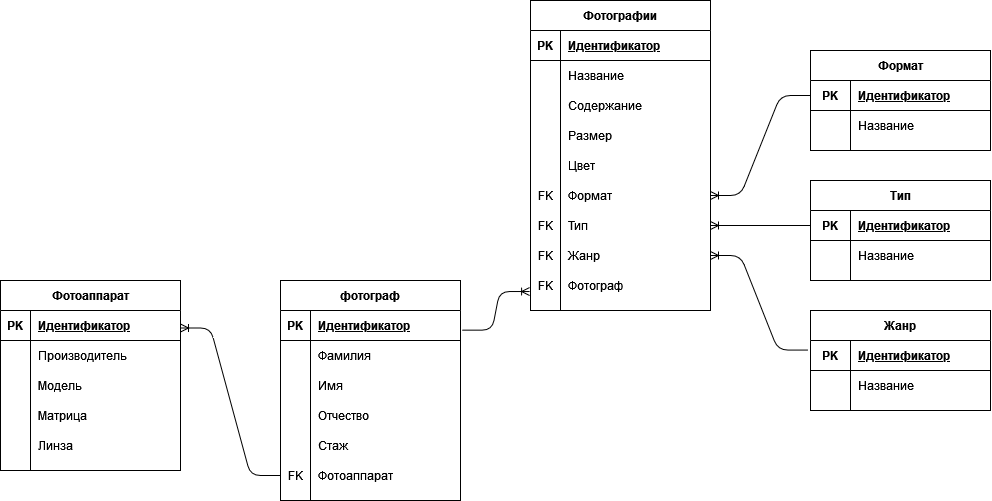


Рисунок 2.3 — логическая модель

### 2.2.2 Физическая схема базы данных

Физическая схема базы данных представлена на рисунке 2.4.



Рисунок .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. |
| сhange\_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.

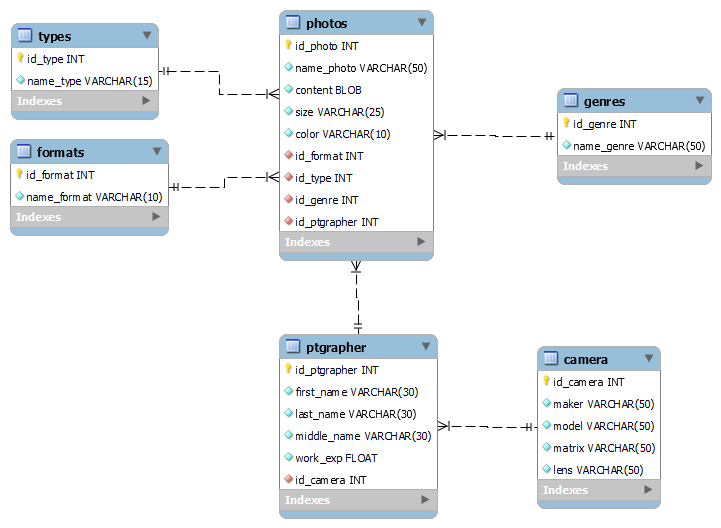


Рисунок 2.5 — модель сущность-связь

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

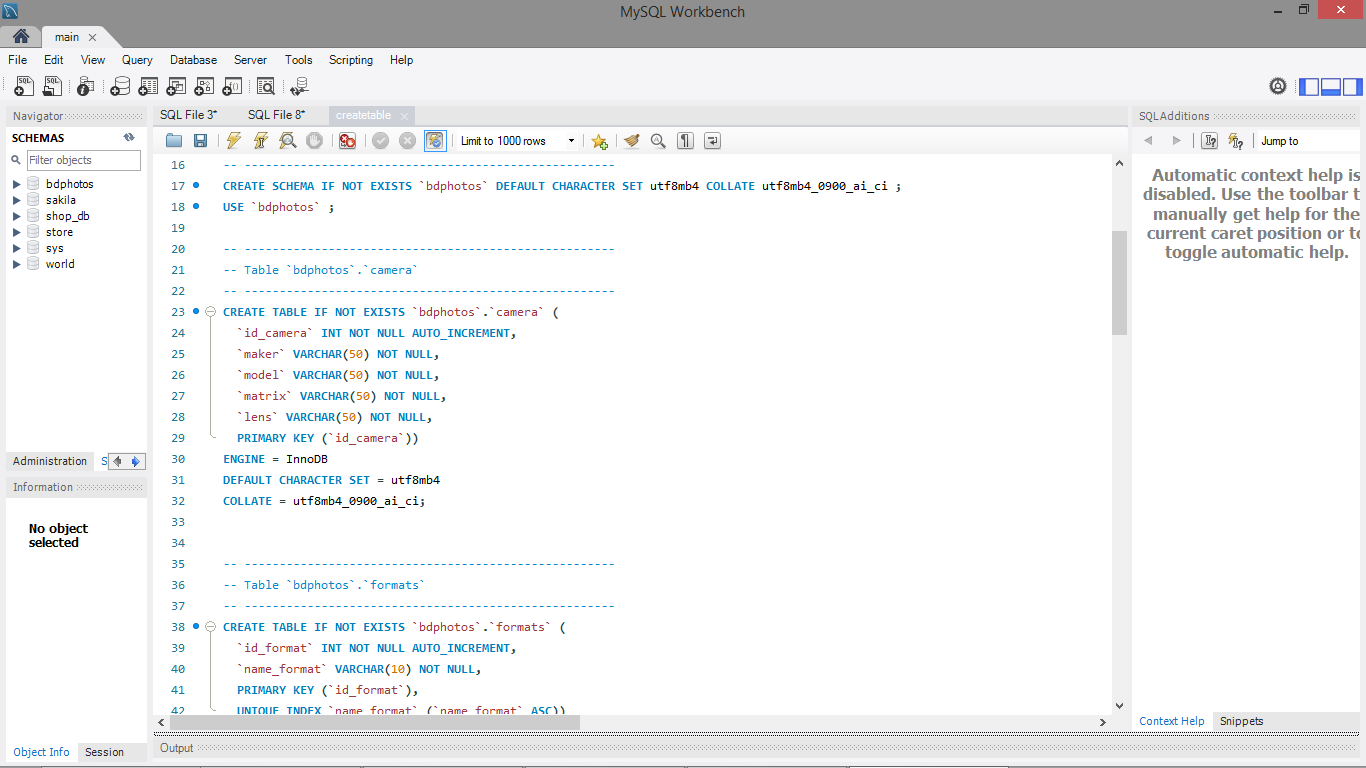


Рисунок 2.6 — создание таблицы camera

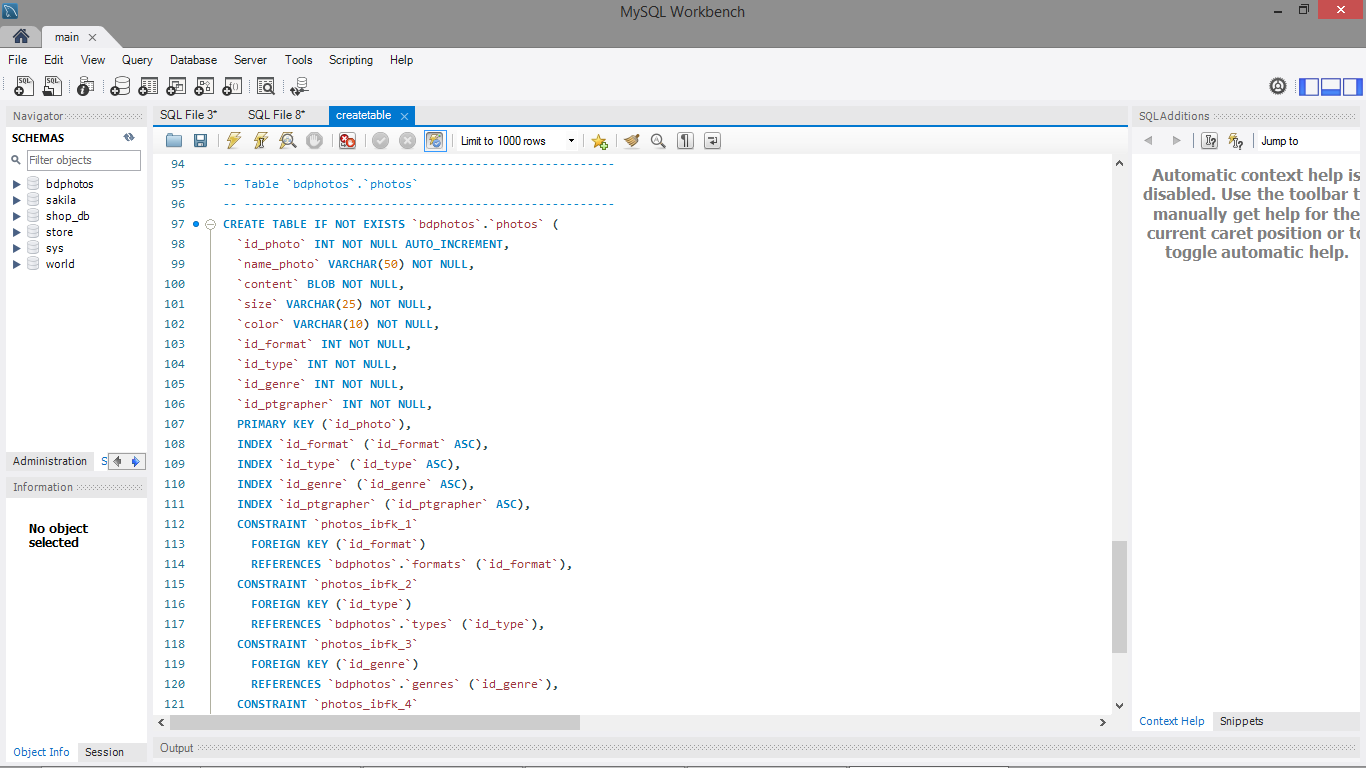


Рисунок 2.7 — создание таблицы photos

### 2.3.2 Разработанные представления

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

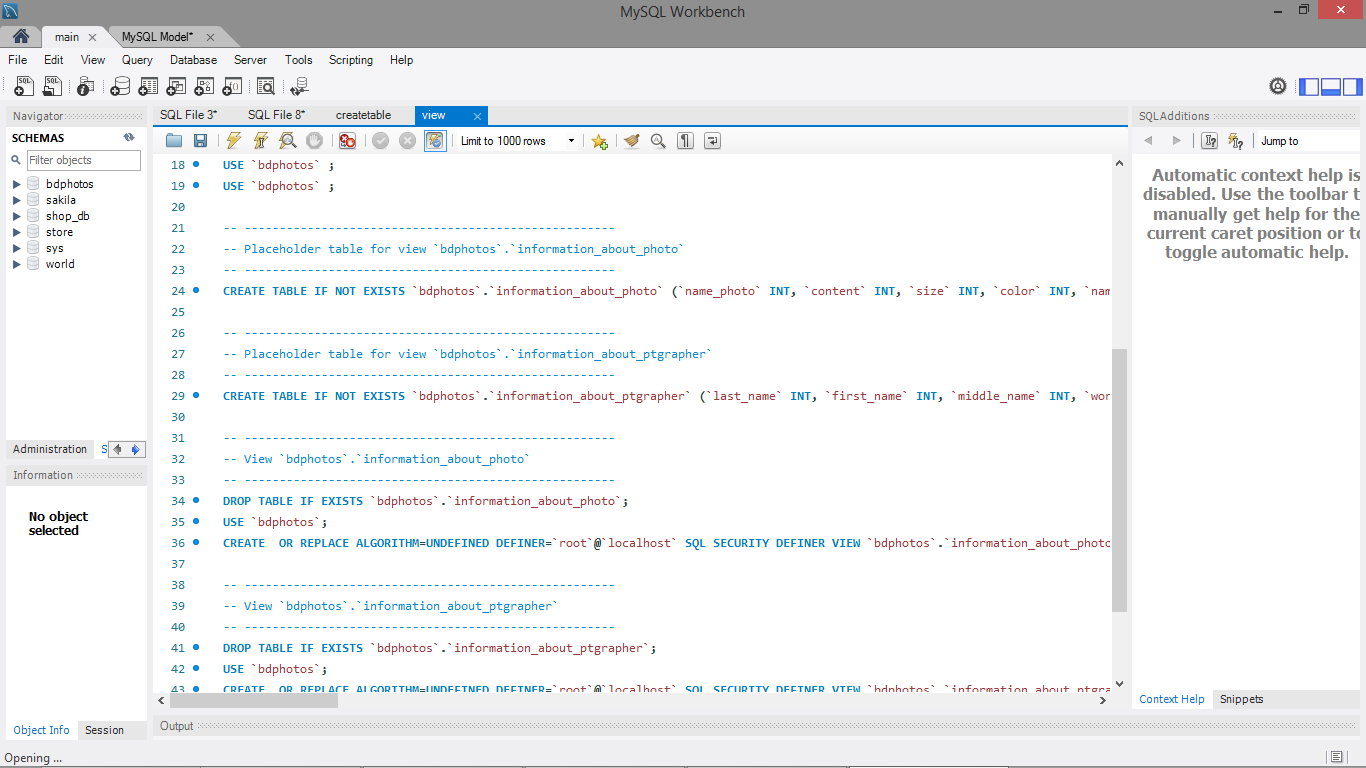


Рисунок 2.8 — создания представления

### 2.3.3 Разработанные процедуры и функции

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

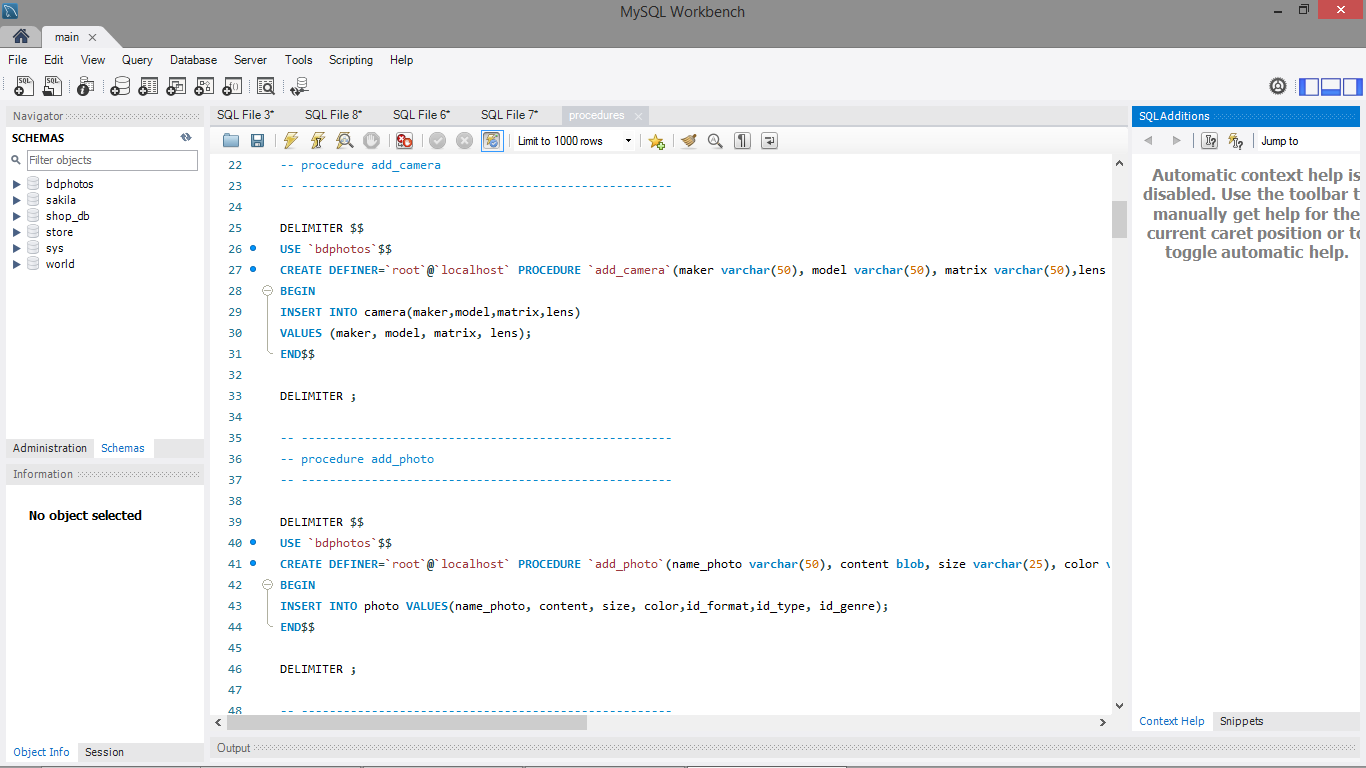


Рисунок 2.9 — создание процедур

# 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/#](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; |