Содержание

[1. Типовой процесс разработки базы данных. 4](#_Toc135767936)

[1.1. Концептуальное моделирование предметной области 4](#_Toc135767937)

[1.2. Логическое проектирование базы данных на основе реляционной модели 8](#_Toc135767938)

[1.3. Физическое проектирование реляционной базы данных и манипулирование данными 9](#_Toc135767939)

[2. Результаты анализа предметной области 11](#_Toc135767940)

[2.1. Общее описание деятельности 11](#_Toc135767941)

[2.2. Описание данных 12](#_Toc135767942)

[3. Результаты разработки базы данных 13](#_Toc135767943)

[3.1. Результаты концептуального моделирования 13](#_Toc135767944)

[3.2. Результаты логического проектирования базы данных 15](#_Toc135767945)

[3.3. Результаты физического проектирования базы данных 18](#_Toc135767946)

[3.4. Типовые запросы к базе данных 22](#_Toc135767947)

[4. Заключение 25](#_Toc135767948)

# Типовой процесс разработки базы данных.

# 1.1. Концептуальное моделирование предметной области

Концептуальное проектирование базы данных – это процедура конструирования информационной модели предприятия, не зависящей от каких-либо физических условий реализации.

В построении общей концептуальной модели данных выделяют ряд этапов:

* Выделение локальных представлений, соответствующих обычно

относительно независимым данным. Каждое такое представление проектируется как подзадача.

* Формулирование сущностей, описывающих локальную предметную область проектируемой БД, и описание атрибутов, составляющих структуру каждой сущности.
* Выделение ключевых атрибутов.
* Спецификация связей между сущностями. Удаление избыточных связей.
* Анализ и добавление неключевых атрибутов.
* Объединение локальных представлений.

Созданная концептуальная модель данных предприятия является

источником информации для фазы логического проектирования базы

данных.

База данных — совместно используемый набор логически связанных данных (и описание этих данных), предназначенный для удовлетворения информационных потребностей организации.

Предметная область базы данных — часть реального мира, представляющая информационный интерес для пользователей, данные о которой хранятся в базе данных.

Тип сущности (сущность) — тип реального объекта (процесса, явления, понятия, события, ситуации), данные о котором хранятся в базе данных.

Экземпляр сущности — конкретный реальный объект, относящийся этой сущности, заданный как набор значений атрибутов.

Атрибут [сущности] — характеристика, признак описания сущности, представляющие информационный интерес для пользователей базы данных.

Связи [между сущностями] — семантические (смысловые) взаимоотношения между сущностями предметной области.

Вид (тип) связи – соотношение количества связываемых экземпляров сущностей. Вид связи определяется исходя из фактов и ограничений предметной области.

Концептуальное описание — создание наиболее абстрактной информационной модели предметной области, не зависящей от каких-либо условий её физической реализации.

Диаграмма «сущность-связь» (ER-диаграмма) – графическая модель предметной области, включающая сущности, их атрибуты и связи между сущностями.

Кратность связи — обозначение количества экземпляров сущностей, участвующих в связи. Указывается для каждой сущности.

Возможные кратности:

0..1 — ноль или один экземпляр сущности;

1..1 — строго один экземпляр сущности;

0..\* — ноль и более экземпляров сущности;

1..\* — один или более экземпляров сущности;

n1..n2 — строго от n1 до n2 экземпляров сущности

# 1.2. Логическое проектирование базы данных на основе реляционной модели

Реляционная модель данных основана на математическом аппарате реляционной алгебры, центральным понятием которого является «отношение» (relation). Строгое математическое обоснование модели позволяет:

• создать достаточно простое представление предметной области с учетом

всех видов связей;

• решить проблемы управления (оперирования) данными;

• решить проблемы непротиворечивости и избыточности данных.

Представлением математического понятия «отношение» для пользователя

реляционной модели данных является реляционная таблица, состоящая из столбцов и строк. Столбцы реляционной таблицы называются атрибутами или полями. Строки реляционной таблицы (за исключением строки имен столбцов) называются кортежами или записями.

Свойства реляционной таблицы:

• таблица имеет уникальное в пределах базы данных имя;

• каждый атрибут имеет уникальное в пределах таблицы имя;

• каждая ячейка таблицы имеет атомарное (элементарное, неделимое) значение;

• значения каждого атрибута относятся к одному домену (типу данных);

• порядок следования атрибутов и записей в таблице произволен;

• каждая запись является уникальной в пределах таблицы, то есть таблица

не может содержать полностью одинаковых записей.

Свойство уникальности записей обеспечивается первичным ключом.

Первичный ключ — атрибут (набор атрибутов), однозначно идентифицирующий запись в таблице.

Первичный ключ, состоящий из нескольких атрибутов, называется составным.

Реляционная модель данных допускает и поощряет использование искусственных (суррогатных) первичных ключей.

Для моделирования связей используется механизм «внешний ключ / первичный ключ».

Внешний ключ — атрибут таблицы, значение которого соответствует значению первичного ключа другой таблицы.

Для пары связываемых таблиц выполняется следующая процедура: первичный ключ первой таблицы (родительской) передается (добавляется) во вторую таблицу (дочернюю) и используется в качестве внешнего ключа.

В этой паре таблиц связанными считаются записи, имеющие одинаковые

значения внешнего и первичного ключей. Внешний ключ дочерней таблицы ссылается на первичный ключ родительской таблицы.

Процедуры логического проектирования базы данных на основе реляционной модели:

1. Определение отношений (реляционных таблиц)

Отношения (реляционные таблицы) представляют в базе данных сущности предметной области. Соответственно набор отношений однозначно определяется исходя из концептуальной модели и записывается следующим образом:

имя\_отношения (атрибут\_1, атрибут\_2, атрибут\_3, …)

Первичный ключ атрибут\_1

1. Определение связей В реляционной модели данных связи между сущностями определяются при помощи механизма «первичный ключ/ внешний ключ». Для моделирования связей используются различные процедуры в зависимости от вида связи.
2. Моделирование связи «один ко многим» Для каждой двухсторонней связи «один ко многим» сущность, находящаяся на стороне «один», определяется как родительская, сущность на стороне связи «многие» – как дочерняя. Для реализации этой связи копия первичного ключа родительской сущности передается в отношение дочерней сущности для использования в качестве внешнего ключа.
3. Моделирование связи «один к одному»

При моделировании связи «один к одному» формально определить родительскую и дочернюю сущности невозможно. Поэтому используется анализ обязательности участия сущностей в связи. Об обязательности участия сущности в связи можно судить по левому числу ограничения кратности при противоположной сущности (1 – обязательное участие, 0 –необязательное участие).

1 вариант. Обязательное участие обеих сторон

В этом случае две связанные сущности объединяются в одно отношение. В качестве первичного ключа нового отношения используется любой из первичных ключей объединяемых отношений.

2 вариант. Обязательное участие одной из сторон

Сущность, которая характеризуется необязательным участием в связи, обозначается как родительская. Сущность, которая должна обязательно участвовать в связи, обозначается как дочерняя. На основании такого обозначения осуществляется передача копии первичного ключа родительской сущности в дочернюю (аналогично моделированию связи «один ко многим»).

3 вариант. Необязательное участие обеих сущностей

В этом случае родительская и дочерняя сущности выбираются произвольно. На основании такого выбора осуществляется передача копии первичного ключа родительской сущности в дочернюю (аналогично моделированию связи «один ко многим»).

1.3. Физическое проектирование реляционной базы данных и манипулирование данными

Язык SQL (Structured Query Language – язык структурированных запросов) представляет собой стандартный высокоуровневый язык описания данных и манипулирования ими в системах управления базами данных, построенных на основе реляционной модели данных.

Язык SQL был разработан фирмой IBM в конце 70-х годов. Первый международный стандарт языка был принят международной стандартизирующей организацией ISO в 1989 г., а новый (более полный) – в 1992 г. В настоящее время все производители реляционных СУБД поддерживают с различной степенью соответствия стандарт SQL92.

Все команды SQL (упрощенно) делятся на две части: команды, предназначенные для создания базы данных и определения ее схемы (состава и структуры реляционных таблиц, связей между ними) относятся к DDL (Data Definition Language – язык определения данных), а команды, предназначенные для выполнения с данными различных действий, таких как добавление, удаление, изменение, извлечение данных относятся к DML (Data Manipulation Language – язык манипулирования данными).

Физическое проектирование реляционной базы данных основано на использовании команд SQL.

Операторы языка DDL предназначены для создания, удаления, изменения основных объектов модели данных реляционных СУБД: таблиц, представлений, индексов.

CREATE TABLE <имя> - создание новой таблицы в базе данных.

DROP TABLE <имя> - удаление таблицы из базы данных.

Операторы языка DML предназначены для манипуляции данными. С их помощью меняется наполнение таблиц. Они позволяют изменять значение строк, столбцов и прочих атрибутов. Такие операторы SQL, например, позволяют удалить информацию о сотруднике, который больше не работает в компании, или исправить данные действующих специалистов. Эти операторы SQL представлены следующими командными словами:

* SELECT — позволяет выбрать данные в соответствии с необходимым условием.
* INSERT — осуществляют добавление новых данных.
* UPDATE — производит замену существующих данных.
* DELETE — удаление информации.

Основные команды DDL языка SQL:

1. Создание базы данных: CREATE DATABASE <имя базы данных>;
2. Удаление базы данных DROP DATABASE <имя базы данных>;
3. Создание таблицы: CREATE TABLE <имя таблицы>

(

<имя атрибута 1> <тип данных> NOT NULL,

<имя атрибута 2> <тип данных>,

<имя атрибута 3> <тип данных>,

…

PRIMARY KEY (<имя атрибута 1>),

FOREIGN KEY (<имя атрибута 3>) REFERENCES <имя родительской таблицы> (<имя атрибута-первичного ключа родительской таблицы>)

);

* 1. Виды типов данных: INT – целое число; FLOAT – вещественное число; TEXT – текстовое значение; DATE – дата.

1. Удаление таблицы: DROP TABLE <имя таблицы> ;

Основные команды DML языка SQL:

1. Добавление данных: INSERT INTO <имя таблицы>

VALUES (<значение>,<значение>, …),

…

(<значение>,<значение>, …);

1. Удаление данных:

Удаление всех данных (всех записей) таблицы: DELETE FROM <имя таблицы>;

Удаление отдельных записей таблицы: DELETE FROM <имя таблицы>

WHERE <имя атрибута> = <значение>;

1. Изменение данных: UPDATE <имя таблицы>

SET <имя атрибута> = <значение>

WHERE <имя атрибута> = <значение>;

1. Извлечение данных: SELECT <список атрибутов>

FROM <список таблиц>

WHERE <условие отбора>;

# Результаты анализа предметной области

# 2.1. Общее описание деятельности

В автосервисе ремонтируют и обслуживают легковые автомобили разного класса и разных производителей. Для того чтобы записаться в автосервис, нужно позвонить в автосервис по телефону сообщить какому узлу автомобиля требуется ремонт, затем выбрать ближайшую дату когда авто возможно будет отдать на ремонт, далее в автосервисе осуществляется ремонт автомобиля, с помощью запчастей имеющихся в автосервисе. Клиент может отдать в автосервис сразу несколько своих автомобилей. В автосервисе есть мастера специализирующиеся на конкретных узлах автомобиля. У мастеров есть специализации: подвеска, двигатель, коробки передач и автоэлектрика, один мастер может иметь несколько специализаций. У автосервиса есть поставщики запчастей. Закупка запчастей происходит каждые две недели.

# 2.2. Описание данных

Ремонт характеризуется датой и временем, узлом авто требующим ремонта и стоимостью.  
Ремонт выполняется мастерами, у мастеров есть ФИО, паспортные данные, номер телефона, специализация.  
Также есть определённые специализации характеризующиеся наименование специализации.  
Клиент при сдаче авто на ремонт предоставляет следующие данные о себе: ФИО, номер телефона.  
Также об автомобиле клиенты предоставляют следующие данные: Марка автомобиля, модель автомобиля, регистрационный номер авто.  
У автосервиса хранятся следующие данные о поставщиках: Номер телефона, почта, закупаемые позиции.  
Ещё автосервис хранит следующие данные о закупаемых запчастях: Наименование запчасти.

# Результаты разработки базы данных

# 3.1. Результаты концептуального моделирования

Таблица 1.

Сущности предметной области

|  |  |
| --- | --- |
| Сущность | Описание |
| Ремонт | содержит информацию о ремонте |
| Мастер | содержит информацию о мастере |
| Клиент | информация о клиентах автосервиса |
| Поставщик | содержит информацию о поставщиках |
| Автомобиль | содержит информацию об автомобиле клиента |
| Специализация | информация о специализации мастера |
| Запчасти | Содержит информацию о запчастях |

Таблица 2.

Связи между сущностями предметной области

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность | Имя связи | Сущность | Вид связи |
| Поставщик | Поставляет | Запчасти | Многие ко многим |
| Клиент | Имеют | Автомобиль | Один ко многим |
| Мастер | Имеет | Специализацию | Один ко многим |
| Осуществляет | Ремонт | Многие к одному |
| Автомобиль | Подлежит | Ремонт | Многие к одному |
| Запчасти | Используются | Ремонт | Многие к одному |

Таблица 3.

Атрибуты

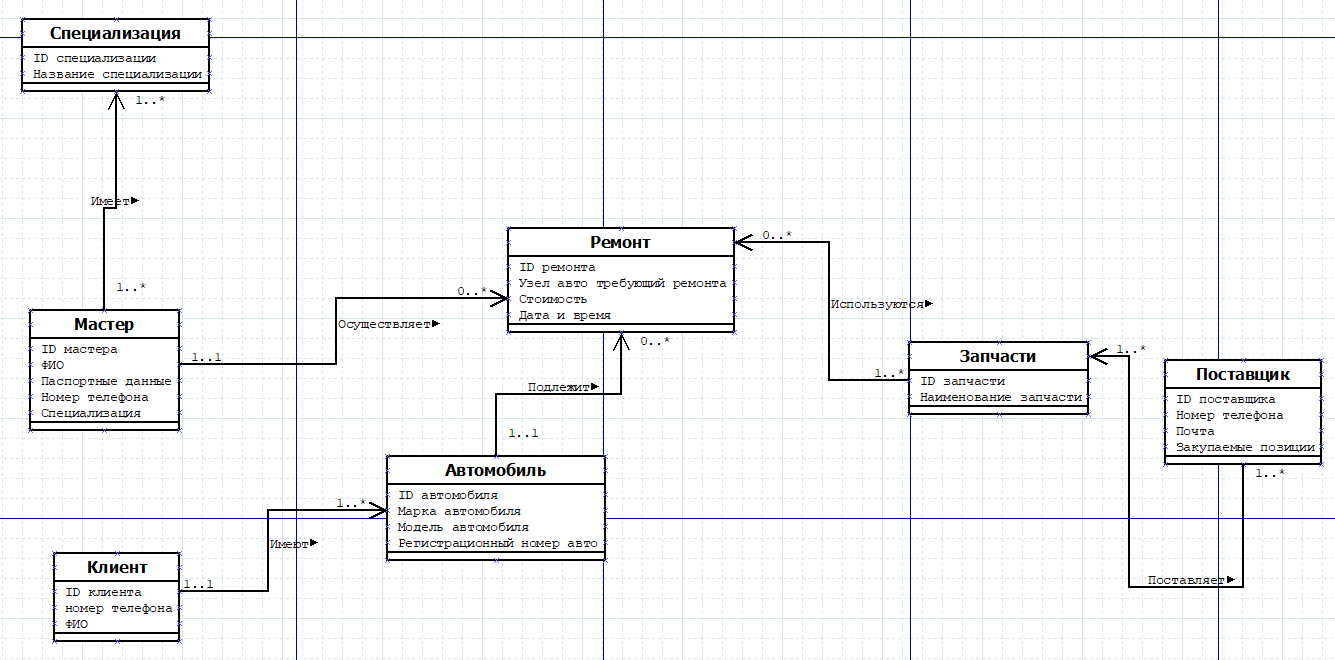
|  |  |  |
| --- | --- | --- |
| Сущность | Атрибута | Домен |
| Ремонт | ID ремонта | Числовой |
| Дата и время | Числовой |
| Узел требующий ремонта | Текстовый |
| Стоимость | Числовой |
| Мастер | ID мастера | Числовой |
| ФИО | Текстовый |
| Паспортные данные | Числовой |
| Номер телефона | Числовой |
| Специализация | Текстовый |
| Специализация | ID специализации | Числовой |
| Наименование специализации | Текстовый |
| Автомобиль | *ID* автомобиля | Числовой |
| Марка автомобиля | Текстовый |
| Модель автомобиля | Текстовый |
| Регистрационный номер авто | Текстовый |
| Запчасти | ID запчасти | Числовой |
| Наименование запчасти | Текстовый |
| Клиент | ID клиента | Числовой |
| ФИО | Текстовый |
| Номер телефона | Числовой |
| Поставщик | *ID* поставщика | Числовой |
| Почта | Текстовый |
| Закупаемые позиции | Текстовый |
| Номер телефона | Числовой |

Таблица 4.

Первичные ключи сущности

|  |  |
| --- | --- |
| Сущность | Первичный ключ |
| Автосервис | ID автосервиса |
| Мастер | ID мастера |
| Автомобиль | ID автомобиля |
| Клиент | ID Клиент |
| Поставщик | ID поставщика |
| Запчасти | ID Запчасти |

Диаграмма "сущность-связь"



*Рисунок 1. Диаграмма "Сущность-связь" в нотации UML с полным представлением атрибутов сущностей*

# 3.2. Результаты логического проектирования базы данных

Автомобиль (IDавтомобиля, марка автомобиля,модель автомобиля, регистрационный номер авто, ID Клиента).

Первичный ключ ID автомобиля

Внешний ключ ID клиента ссылается на ID клиента отношения *клиент*.

Мастер (IDмастера, ФИО, паспортные данные, номер телефона, специализация, ID специализации*).*

Первичный ключ ID Мастера

Внешний ключ ID специализации ссылается на ID д специализации отношения специализация.

Клиент (ID Клиента, ФИО, номер телефона).

Первичный ключ ID клиента

Специализация (ID специализации, специализация.)

Первичный ключ ID специализации

Ремонт (ID ремонта, ID мастера, ID автомобиля, ID запчасти дата и время,узел требующий ремонта, стоимость).

Первичный ключ ID ремонта

Внешний ключ ID автомобиля ссылается на ID автомобиля отношения *автомобиль*.

Внешний ключ ID мастера ссылается на ID мастера отношения мастер.

**Поставщик** (ID поставщика, почта, номер телефона, закупаемые позиции).

Первичный ключ ID поставщика.

**Запчасть** *(*ID запчасти, наименование запчасти, ID поставщика).

Первичный ключ ID запчасти.

Внешний ключ ID поставщика ссылается на ID поставщика отношения *поставщик*.

**Ремонт-Запчасти**

Первичный ключ ID ремонт-запчасть

Внешний ключ ID запчасти ссылается на ID запчасти отношениязапчасть.

Внешний ключ ID ремонта ссылается на ID ремонта отношения *ремонт*.

**Запчасти-поставщик**

Первичный ключ ID запчасти-поставщик

Внешний ключ ID запчасти ссылается на ID запчасти отношениязапчасть.

Внешний ключ ID поставщика ссылается на ID поставщика отношения *поставщик*.

Мастер-специализация

Первичный ключ ID мастер-специализация

Внешний ключ ID мастера ссылается на ID мастера отношения мастер.

Внешний ключ ID специализации ссылается на ID д специализации отношения специализация.

# Результаты физического проектирования базы данных

CREATE TABLE `client` (

`client\_id` INTEGER(11) NOT NULL,

`Name` TEXT COLLATE utf8\_general\_ci,

`Lastname` TEXT COLLATE utf8\_general\_ci,

`Middlename` TEXT COLLATE utf8\_general\_ci,

`Nom\_tel` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`client\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `auto` (

`auto\_id` INTEGER(11) NOT NULL,

`Marka\_auto` TEXT COLLATE utf8\_general\_ci,

`Model\_auto` TEXT COLLATE utf8\_general\_ci,

`Reg\_nom\_auto` TEXT COLLATE utf8\_general\_ci,

`client\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`auto\_id`),

KEY `client` USING BTREE (`client\_id`),

CONSTRAINT `client` FOREIGN KEY (`client\_id`) REFERENCES `client` (`client\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `specialization` (

`specialization\_id` INTEGER(11) NOT NULL,

`specialization` TEXT COLLATE utf8\_general\_ci,

PRIMARY KEY USING BTREE (`specialization\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `master` (

`master\_id` INTEGER(11) NOT NULL,

`Name` TEXT COLLATE utf8\_general\_ci,

`Lastname` TEXT COLLATE utf8\_general\_ci,

`Middlename` TEXT COLLATE utf8\_general\_ci,

`Passport\_dan` TEXT COLLATE utf8\_general\_ci,

`Nom\_tel` INTEGER(11) DEFAULT NULL,

`specialization` TEXT COLLATE utf8\_general\_ci,

`specialization\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`master\_id`),

KEY `specialization\_1` USING BTREE (`specialization\_id`),

CONSTRAINT `specialization\_1` FOREIGN KEY (`specialization\_id`) REFERENCES `specialization` (`specialization\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `master\_specialization` (

`master\_specialization\_id` INTEGER(11) NOT NULL,

`master\_id` INTEGER(11) DEFAULT NULL,

`specialization\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`master\_specialization\_id`),

KEY `master` USING BTREE (`master\_id`),

KEY `specialization` USING BTREE (`specialization\_id`),

CONSTRAINT `master` FOREIGN KEY (`master\_id`) REFERENCES `master` (`master\_id`),

CONSTRAINT `specialization` FOREIGN KEY (`specialization\_id`) REFERENCES `specialization` (`specialization\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `postavshik` (

`postavshik\_id` INTEGER(11) NOT NULL,

`email` TEXT COLLATE utf8\_general\_ci,

`Nom\_tel` INTEGER(11) DEFAULT NULL,

`Zak\_poz` TEXT COLLATE utf8\_general\_ci,

PRIMARY KEY USING BTREE (`postavshik\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `spare\_part` (

`Spare\_part \_id` INTEGER(11) NOT NULL,

`Name\_sp` TEXT COLLATE utf8\_general\_ci,

`postavshik\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`Spare\_part \_id`),

KEY `postavshik\_1` USING BTREE (`postavshik\_id`),

CONSTRAINT `postavshik\_1` FOREIGN KEY (`postavshik\_id`) REFERENCES `postavshik` (`postavshik\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `postavshik–spare\_part` (

`postavshik–spare\_part\_id` INTEGER(11) NOT NULL,

`postavshik\_id` INTEGER(11) DEFAULT NULL,

`Spare\_part\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`postavshik–spare\_part\_id`),

KEY `postavshik` USING BTREE (`postavshik\_id`),

KEY `spare\_part` USING BTREE (`Spare\_part\_id`),

CONSTRAINT `postavshik` FOREIGN KEY (`postavshik\_id`) REFERENCES `postavshik` (`postavshik\_id`),

CONSTRAINT `spare\_part` FOREIGN KEY (`Spare\_part\_id`) REFERENCES `spare\_part` (`Spare\_part \_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `remont` (

`remont\_id` INTEGER(11) NOT NULL,

`Data\_time` TEXT COLLATE utf8\_general\_ci,

`Yzel\_tr\_remont` TEXT COLLATE utf8\_general\_ci,

`price` INTEGER(11) DEFAULT NULL,

`auto\_id` INTEGER(11) DEFAULT NULL,

`master\_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`remont\_id`),

KEY `auto` USING BTREE (`auto\_id`),

KEY `master\_1` USING BTREE (`master\_id`),

CONSTRAINT `auto` FOREIGN KEY (`auto\_id`) REFERENCES `auto` (`auto\_id`),

CONSTRAINT `master\_1` FOREIGN KEY (`master\_id`) REFERENCES `master` (`master\_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

CREATE TABLE `remont–spare\_part` (

`remont–spare\_part\_id` INTEGER(11) NOT NULL,

`remont\_id` INTEGER(11) DEFAULT NULL,

`Spare\_part \_id` INTEGER(11) DEFAULT NULL,

PRIMARY KEY USING BTREE (`remont–spare\_part\_id`),

KEY `remont` USING BTREE (`remont\_id`),

KEY `spare\_part\_1` USING BTREE (`Spare\_part \_id`),

CONSTRAINT `remont` FOREIGN KEY (`remont\_id`) REFERENCES `remont` (`remont\_id`),

CONSTRAINT `spare\_part\_1` FOREIGN KEY (`Spare\_part \_id`) REFERENCES `spare\_part` (`Spare\_part \_id`)

) ENGINE=InnoDB

ROW\_FORMAT=DYNAMIC CHARACTER SET 'utf8' COLLATE 'utf8\_general\_ci'

;

# 

# 3.4. Типовые запросы к базе данных

1. Сформировать список всех автомобилей марки «BMW».

SELECT `auto`.\*

FROM `auto`

WHERE `auto`.`Marka\_auto`='BMW'

1. Cформировать список марок автомобилей сумма ремонта которых составила «2300р».

SELECT `auto`.`Marka\_auto`

FROM `auto`, `remont`

WHERE `remont`.`auto\_id`=`auto`.`auto\_id`

AND `remont`.`price`='2300'

1. Сформировать список фамилий мастеров, специализирующихся на ремонте «Автоэлектрики».

SELECT `master`.`Lastname`

FROM `master`,`specialization`,`master\_specialization`

WHERE `master`.`master\_id`=`master\_specialization`.`master\_id`

AND `specialization`.`specialization\_id`=`master\_specialization`.`specialization\_id`

AND `specialization`.`specialization`='Автоэлектрика'

1. Сформировать список поставщиков двигателей.

SELECT `postavshik`.\*

FROM `postavshik`, `spare\_part`, `postavshik–spare\_part`

WHERE `postavshik–spare\_part`.`postavshik\_id`=`postavshik`.`postavshik\_id`

AND `spare\_part`.`Spare\_part \_id`=`postavshik–spare\_part`.`Spare\_part\_id`

AND `spare\_part`.`Name\_sp`='двигатель'

1. Узнать марку автомобиля клиента по фамилии «Конев».

SELECT `auto`.`Marka\_auto`

FROM `auto`,`client`

WHERE `auto`.`client\_id`=`client`.`client\_id`

AND `client`.`Lastname`='Конев'

1. Узнать какой ремонт требуется автомобилю «Оксаны Дубовой».

SELECT `remont`.`Yzel\_tr\_remont`

FROM `auto`,`client`,`remont`

WHERE `auto`.`client\_id`=`client`.`client\_id`

AND `remont`.`auto\_id`=`auto`.`auto\_id`

AND `client`.`Lastname`='Дубова'

1. Сформировать список деталей, которые требуются для ремонта автомобиля «Lada».

SELECT `spare\_part`.`Name\_sp`

FROM `auto`,`remont`,`remont–spare\_part`,`spare\_part`

WHERE `remont–spare\_part`.`remont\_id`=`remont`.`remont\_id`

AND `remont–spare\_part`.`Spare\_part \_id`=`spare\_part`.`Spare\_part \_id`

AND `remont`.`auto\_id`=`auto`.`auto\_id`

AND `auto`.`Marka\_auto`='Lada'

1. Узнать специализацию мастера по фамилии «Петров».

SELECT `specialization`.`specialization`

FROM `specialization`,`master`, `master\_specialization`

WHERE `master\_specialization`.`master\_id`=`master`.`master\_id`

AND `master\_specialization`.`specialization\_id`=`specialization`.`specialization\_id`

AND `master`.`Lastname`='Петров'

1. Сформировать список ремонтных работ за «12.10.2023»

SELECT `remont`.\*

FROM `remont`

WHERE `remont`.`Data\_time`='12.10.2023'

1. Узнать номер телефона поставщика «Двигателей».

SELECT `postavshik`.`Nom\_tel`

FROM `postavshik`

WHERE `postavshik`.`Zak\_poz`='двигатели'

1. Заключение

В данной курсовой работе была создана база данных "Автосервис", которая способна удовлетворить запросы автосервиса об автомобилях, находящихся на ремонте, нужных запчастях, данных клиентов и мастеров.

В ходе выполнения данной курсовой работы были выполнены все пункты задания:

1. Анализ предметной области;
2. Выделение сущностей и атрибутов;
3. Установление связей между сущностями;
4. Моделирование логического и физического уровней;
5. Реализация базы данных в СУБД «SQL Manager Lite for MySQL»;
6. Заполнение и тестирование полученной Базы Данных.

При моделировании использовалось свободное программное обеспечение – программа для построения диаграмм Dia.

Моделирование предметной области осуществлялось с помощью диаграммы «сущность-связь».

Создание базы данных помогло мне закрепить свои навыки в проектировании и работе с базами данных, а также навыки работы с реляционной СУБД MySQL и языком программирования SQL.