**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 5](#_Toc36152406)

[1 Нормализация базы данных, создание и заполнение таблиц, получение исходной таблицы из нормализованной базы данных 7](#_Toc36152407)

[1.1 Условия задания варианта № 17 7](#_Toc36152408)

[1.2 Нормализация данных 8](#_Toc36152409)

[1.2.1 Первая нормальная форма 9](#_Toc36152410)

[1.2.2 Вторая нормальная форма 10](#_Toc36152411)

[1.2.3 Третья нормальная форма 12](#_Toc36152412)

[1.3 Создание базы данных “Книжный интернет-магазин” 12](#_Toc36152413)

[1.4 Получение исходной таблицы из нормализованных таблиц 14](#_Toc36152414)

[2 Создание, заполнение таблиц базы данных и выполнение запросов на языке SQL 15](#_Toc36152415)

[2.1 Условия задания варианта № 17 15](#_Toc36152416)

[2.2 Создание таблиц 17](#_Toc36152417)

[2.3 Создание ER – диаграммы в Workbench 18](#_Toc36152418)

[2.4 Заполнение таблиц данными 19](#_Toc36152419)

[2.5 Выборка данных из базы данных 20](#_Toc36152420)

[3 Хранимые процедуры и функции, триггеры и назначение прав доступа 23](#_Toc36152421)

[3.1 Условия задания варианта № 17 23](#_Toc36152422)

[3.2 Создание базы данных, таблиц и заполнения 25](#_Toc36152423)

[3.3 Создание хранимых процедур, хранимых функций и триггеров 27](#_Toc36152424)

[3.4 Назначение прав доступа к базам данных 30](#_Toc36152425)

[ЗАКЛЮЧЕНИЕ 32](#_Toc36152426)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 34](#_Toc36152427)

# Введение

Базы данных – это программы, которые позволяют сохранять и получать большие объемы связанной информации. Базы данных состоят из таблиц, которые содержат информацию. Создавая базу данных, необходимо подумать о том, какие таблицы нужно создать и какие связи существуют между информацией в таблицах. Хороший проект базы данных обеспечит целостность данных и простоту их обслуживания.

Язык запросов к базам данных был придуман для этих целей и был назван «Структурированный язык запросов» или «SQL».

Задачами курсовой работы являются:

1. Изучить модели базы данных;
2. Получение представления о языке SQL;
3. Освоить СУБД MySQL;
4. Разработать базу данных с использованием запросов SQL;

Следующие компетенции должны быть освоены в результате выполнения курсовой работы:

* ОПК-1 способностью инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем:
* знать реляционную модель данных;
* уметь инсталлировать программное и аппаратное обеспечение для СУБД;
* владеть навыками работы с программным и аппаратным обеспечением СУБД.
* ОПК-2 Способностью осваивать методики использования программных средств для решения практических задач:
* знать типы и свойства полей, основные объекты баз данных;
* знать операции, выполняемые с базами данных;
* знать язык запросов SQL;
* уметь формировать запросы к базе данных;
* владеть навыками проектирования баз данных.

# 1 Нормализация базы данных, создание и заполнение таблиц, получение исходной таблицы из нормализованной базы данных

## Условия задания варианта № 17

Создать базу данных, содержащую таблицы, удовлетворяющие условиям нормализации (3НФ).

Книжный интернет-магазин:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ISBN | ФИО автора | Название книги | Год издания | Цена, руб. | № заказа | Адрес доставки | Дата заказа | Дата [[1]](#footnote-1)выполнения заказа |
| 978-5-388-00003 | Иванов Сергей Степанович | Самоучитель JAVA | 2012 | 300 | 123456 | Малая Арнаутская ул., д.9, кв.16 | 20.09.2013 | 22.09.2013 |
| 978-5-699-58103 | Сидорова Ольга Юрьевна | JAVA за 21 день | 2013 | 600 | 123456 | Малая Арнаутская ул., д.9, кв.16 | 20.09.2013 | 22.09.2013 |
| 978-5-388-00003 | Иванов Сергей Степанович | Самоучитель JAVA | 2012 | 300 | 222334 | Курчатов бульвар, д.33, кв.9 | 21.09.2013 |  |
| 675-3-423-00375 | Петров Иван Петрович | Физика | 2013 | 450 | 222334 | Курчатов бульвар, д.33, кв.9 | 21.09.2013 |  |
| 758-3-004-87105 | Петров Иван Петрович | Сопромат | 2013 | 350 | 432152 | Нахимовский проспект, д.12, кв.89 | 21.09.2012 | 23.09.2012 |

1. Определить типы и размерности полей, необходимых для сохранения и последующего использования данных из представленной ведомости.
2. Определить количество таблиц, необходимых для хранения всех представленных данных. Таблицы должны находиться в 3 нормальной форме.
3. Записать на стандартном SQL языке запросы на создание и заполнение этих таблиц.
4. Запросы должны содержать описания типов и размерностей полей, возможные ограничения на значения полей, описания первичных и внешних ключей, каскадные операции с записями подчиненных таблиц.
5. Используя таблицы, полученные в предыдущих пунктах задания, создать запрос, выдающий данные максимально соответствующие ведомости.

## Нормализация данных

Процессом разделения по отдельным связанным таблицам называется нормализацией. Нормализация позволяет легко работать с большим объемом данных и тем самым избежать нарушения целостности данных при их изменении.

В проектировании баз данных нормализация преимущественно применяется при восходящем подходе, то есть данные, которые нам нужно сохранить в базу данных, группируем по сущностям, из которых создаем таблицы. При нисходящем подходе нормализация применяется, к примеру, для проверки корректности спроектированных таблиц.

Также нормализация предполагает использование нормальных форм. Всего существует 7 нормальных форм. Каждая из них подразумевает, что к данным уже была применена предыдущая нормальная форма. И что более важно: база данных считается нормализованной, если к ней применяется третья нормальная форма и выше.

Наиболее важные на практике нормальные формы отношений основываются на фундаментальном в теории реляционных баз данных понятии функциональной зависимости.

В отношении R атрибут Y функционально зависит от атрибута X (X и Y могут быть составными) в том и только в том случае, если каждому значению X соответствует в точности одно значение Y: R.X (r) R.Y.

В нашем случае функционально зависят:

* атрибут «ФИО автора» от атрибута «ISBN»
* атрибуты «Название книги», «Год издания», «Цена» от атрибута «ISBN»
* атрибуты «Адрес доставки», «Дата заказа», «Дата выполнения заказа» от атрибута «№ заказа»

Функциональная зависимость R.X (r) R.Y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X.

В нашем случае полная функциональная зависимость:

* атрибут «ФИО автора» от атрибута «ISBN»
* атрибуты «Название книги», «Год издания», «Цена» от атрибута «ISBN»
* атрибуты «Адрес доставки», «Дата заказа», «Дата выполнения заказа» от атрибута «№ заказа»

«Не ключевым» атрибутом называется любой атрибут отношения, не входящий в состав первичного ключа (в частности, первичного).

В нашем случае «не ключевые» атрибуты:

* «ФИО автора»
* «Название книги»
* «Год издания»
* «Цена»
* «Адрес доставки»
* «Дата заказа»
* «Дата выполнения заказа»

### Первая нормальная форма

Первая нормальная форма (1NF) предполагает, что сохраняемые данные на пересечении строк и столбцов должны представлять скалярное значение, а таблицы не должны содержать повторяющихся строк.

Есть два похода к переходу от ненормализованной таблицы к первой нормальной форме. Первый способ называется выравниванием или flattaning. Он предполагает декомпозицию строки с повторяющимися группами данных, при котором для каждой повторяющейся группы создается своя строка. Полученная в результате таблица будет содержать атомарные значения для каждого из атрибутов. Хотя в то же время этот подход увеличит избыточность данных.

Второй подход предполагает, что один атрибут или группа атрибутов назначаются ключом ненормализованной таблицы, а затем повторяющиеся группы удаляются из таблицы и помещаются в отдельную таблицу вместе с копиями ключа из исходной таблицы.

### Вторая нормальная форма

Во второй нормальной форме каждый столбец в таблице, который не является ключом, должен зависеть от ключа.

Эта форма применяется к тем таблицам, которые имеют составной первичный ключ, то есть где первичный ключ состоит из нескольких атрибутов. Если в таблице несоставной первичный ключ, то в этом случае считается, что все остальные атрибуты автоматически находятся в полной функциональной зависимости от первичного ключа.

Вторая нормальная форма применяется только к тем таблицам, которые находятся в первой нормальной форме. После применения второй формы все столбцы таблицы зависят от первичного ключа.

Выполним нормализацию БД «Книжный интернет-магазин» до 2НФ. Так как поле «ФИО автора» зависит от ключевого поля «№ автора», то получаем новую таблицу «Авторы»(Таблица 1):

|  |  |
| --- | --- |
| № автора | ФИО автора |
| 1 | Иванов Сергей Степанович |
| 2 | Сидорова Ольга Юрьевна |
| 3 | Петров Иван Петрович |

Таблица 1 – Таблица «Авторы»

Повторяющиеся записи в этой таблице удаляем, иначе таблица не будет находиться в 1NF.

Аналогично создаем еще одну таблицу «Книги» (Таблица 2). Повторяющиеся записи в этой таблице так же удалены:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ISBN | № автора | Название книги | Год издания | Цена |
| 978-5-388-00003 | 1 | Самоучитель JAVA | 2012 | 300 |
| 978-5-699-58103 | 2 | JAVA за 21 день | 2013 | 600 |
| 675-3-423-00375 | 3 | Физика | 2013 | 450 |
| 758-3-004-87105 | 3 | Сопромат | 2013 | 350 |

Таблица 2 – Таблица «Книги»

Таблица «Заказы» будет содержать следующие поля (Таблица 3):

|  |  |  |  |
| --- | --- | --- | --- |
| № заказа | Адрес доставки | Дата заказа | Дата выполнения заказа |
| 123456 | Малая Арнаутская ул., д.9, кв.16 | 2013.09.20 | 2013.09.22 |
| 222334 | Курчатов бульвар, д.33, кв.9 | 2013.09.21 |  |
| 432152 | Нахимовский проспект, д.12, кв.89 | 2012.09.21 | 2012.09.23 |

Таблица 3 – Таблица «Заказы»

И имеем последнюю таблицу «Книги-заказов»(Таблица 4):

|  |  |
| --- | --- |
| № заказа | ISBN |
| 123456 | 978-5-388-00003 |
| 123456 | 978-5-699-58103 |
| 222334 | 978-5-388-00003 |
| 222334 | 675-3-423-00375 |
| 432152 | 758-3-004-87105 |

Таблица 4 – Таблица «Книги-заказов»

### 1.2.3 Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

В нашем случае, после приведения таблиц в 2NF, таблицы соответствуют и 3NF, поэтому дополнительных преобразований больше не требуется.

На практике третья нормальная форма схем отношений достаточна в большинстве случаев, и приведением к третьей нормальной форме процесс проектирования реляционной базы данных обычно заканчивается. Однако иногда полезно продолжить процесс нормализации.

## 1.3 Создание базы данных “Книжный интернет-магазин”

Следующим этапом задания является:

* определение типов и размерностей полей, необходимых для сохранения данных;
* создание полученных в предыдущем разделе таблиц на языке SQL;
* задание внешних ключей, поддерживающих связи между таблицами и задающими правила выполнения запросов на создание, удаление и обновление записей таблицах;
* заполнение полученных таблиц данными с помощью оператора языка SQL – INSERT;
* и как результат работы, подтверждающий правильность созданных таблиц, получение исходной (заданной в задании) таблицы.

Ниже представлено создание базы данных и нормализованных таблиц (см. подраздел 1.2.2) на языке SQL:

DROP DATABASE work\_1;

CREATE DATABASE work\_1;

SET foreign\_key\_checks = 0;

USE work\_1;

DROP TABLE IF EXISTS `Книги-заказов`;

DROP TABLE IF EXISTS `Заказы`;

DROP TABLE IF EXISTS `Книги`;

DROP TABLE IF EXISTS `Авторы`;

CREATE TABLE `Авторы` (

`№ автора` INT NOT NULL,

`ФИО автора` VARCHAR(36) NOT NULL,

PRIMARY KEY(`№ автора`)

) ENGINE=InnoDB;

CREATE TABLE `Книги` (

`ISBN` VARCHAR(22) NOT NULL,

`№ автора` INT NOT NULL,

`Название книги` VARCHAR(48) NOT NULL,

`Год издания` INT NOT NULL,

`Цена` INT NOT NULL,

PRIMARY KEY(`ISBN`),

FOREIGN KEY(`№ автора`)

REFERENCES `Авторы`(`№ автора`)

ON DELETE NO ACTION ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Заказы` (

`№ заказа` INT NOT NULL,

`Адрес доставки` VARCHAR(100) NOT NULL,

`Дата заказа` DATE NOT NULL,

`Дата выполнения заказа` DATE NULL,

PRIMARY KEY(`№ заказа`)

) ENGINE=InnoDB;

CREATE TABLE `Книги-заказов` (

`№ заказа` INT NOT NULL,

`ISBN` VARCHAR(22) NOT NULL,

PRIMARY KEY(`№ заказа`, `ISBN`),

FOREIGN KEY(`№ заказа`)

REFERENCES `Заказы`(`№ заказа`)

ON DELETE NO ACTION ON UPDATE CASCADE,

FOREIGN KEY(`ISBN`)

REFERENCES `Книги`(`ISBN`)

ON DELETE NO ACTION ON UPDATE CASCADE

) ENGINE=InnoDB;

INSERT INTO `Авторы`

(`№ автора`, `ФИО автора`)

VALUES

(1, 'Иванов Сергей Степанович'),

(2, 'Сидорова Ольга Юрьевна'),

(3, 'Петров Иван Петрович');

INSERT INTO `Книги`

(`ISBN`, `№ автора`, `Название книги`, `Год издания`, `Цена`)

VALUES

('978-5-388-00003', 1, 'Самоучитель JAVA', 2012, 300),

('978-5-699-58103', 2, 'JAVA за 21 день', 2013, 600),

('675-3-423-00375', 3, 'Физика', 2013, 450),

('758-3-004-87105', 3, 'Сопромат', 2013, 350);

INSERT INTO `Заказы`

(`№ заказа`, `Адрес доставки`, `Дата заказа`, `Дата выполнения заказа`)

VALUES

(123456, 'Малая Арнаутская ул., д.9, кв.16', '2013.09.20', '2013.09.22'),

(222334, 'Курчатов бульвар, д.33, кв.9', '2013.09.21', NULL),

(432152, 'Нахимовский проспект, д.12, кв.89', '2012.09.21', '2012.09.23');

INSERT INTO `Книги-заказов`

(`№ заказа`, `ISBN`)

VALUES

(123456, '978-5-388-00003'),

(123456, '978-5-699-58103'),

(222334, '978-5-388-00003'),

(222334, '675-3-423-00375'),

(432152, '758-3-004-87105');

## Получение исходной таблицы из нормализованных таблиц

Ниже представлен запрос, который предоставляет исходную таблицу, послужившая основой для нормализации таблиц:

SELECT `Книги`.`ISBN`,

`Авторы`.`ФИО автора`,

`Книги`.`Название книги`,

`Книги`.`Год издания`,

`Книги`.`Цена`,

`Заказы`.`№ заказа`,

`Заказы`.`Адрес доставки`,

`Заказы`.`Дата заказа`,

`Заказы`.`Дата выполнения заказа`

FROM (((`Авторы` INNER JOIN `Книги` ON

`Авторы`.`№ автора`=`Книги`.`№ автора`) LEFT JOIN `Книги-заказов` ON

`Книги`.`ISBN`=`Книги-заказов`.`ISBN`) LEFT JOIN `Заказы` ON

`Книги-заказов`.`№ заказа`=`Заказы`.`№ заказа`) ORDER BY `№ заказа`;

# 2 Создание, заполнение таблиц базы данных и выполнение запросов на языке SQL

## 2.1 Условия задания варианта № 17

Создать в MySQL базу данных «Такси», состоящую из 4-х связанных таблиц, представленных ниже:

Марки автомобилей:

|  |  |  |
| --- | --- | --- |
| Модель автомобиля | Стоимость (руб.) | |
| минуты простоя | километра проезда |
| Kia Rio[[2]](#footnote-2) | 5 | 20 |
| Toyota Camry | 8 | 30 |
| Ford Focus | 5 | 20 |

Водители:

|  |  |  |  |
| --- | --- | --- | --- |
| Гос. номер | ФИО водителя | Телефон | Модель автомобиля |
| С734ХК750 | Иванов Петр Васильевич | 907-891-78-92 | Kia Rio |
| С865МР750 | Петров Андрей Иванович | 928-742-87-34 | Kia Rio |
| М777КМ777 | Бендер Остап Ибрагимович | 916-758-34-90 | Toyota Camry |
| С654РР7150 | Фролов Виктор Валерьевич | 967-456-12-18 | Ford Focus |

Поездки:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Гос. номер | Дата | Время вызова | Время завершения | Время ожидания у клиента | Расстояние |
| С734ХК750 | 02.02.2020 | 12-20 | 13-10 | 2 | 90 |
| С734ХК750 | 02.02.2020 | 14-45 | 15-50 | 5 | 50 |
| М777КМ777 | 03.02.2020 | 18-30 | 20-20 | 5 | 70 |
| С865МР750 | 02.02.2020 | 10-00 | 10-50 | 2 | 30 |

Поездки1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Гос. номер | Дата | Время вызова | Время завершения | Время ожидания у клиента | Расстояние |
| С865МР750 | 02.02.2020 | 12-00 | 13-45 | 5 | 80 |
| С865МР750 | 03.02.2020 | 10-30 | 11-45 | 10 | 45 |
| С865МР750 | 03.02.2020 | 23-40 | 01-10 | 12 | 95 |

Для этого необходимо на языке SQL написать и выполнить следующие запросы:

1. Создать представленные выше таблицы, индексы по первичным и родительским ключам.
2. С помощью Workbench MySQL по полученным таблицам создать ER диаграмму.
3. INSERT INTO – заполнить полученные таблицы данными.
4. INSERT INTO – заполнить таблицу «Поездки» данными из таблицы «Поездки 1».
5. UPDATE – увеличить в поле «Стоимость (руб.) километра проезда» таблицы «Марки автомобилей» стоимость на 5%.
6. DELETE – удалить данные по гос.номеру М777КМ777 из всех таблиц.
7. DROP – удалить таблицу «Поездки1».
8. SELECT – вывести на экран записи, содержащие следующие поля: Гос. номер, Дата и Время вызова, для которых расстояние поездки было от 50 до 80 км.
9. SELECT – после задания Гос. номера вывести на экран записи, содержащие следующие поля: Гос. номер, ФИО водителя, Модель автомобиля, Стоимость (руб.) километра проезда, Дата и Время вызова, Расстояние по каждой поездке на этом автомобиле.
10. SELECT – вывести на экран записи, содержащие следующие поля: Гос. номер, ФИО водителя, Модель автомобиля для всех водителей, которые ни разу не выходили на работу.
11. SELECT – вывести на экран записи, содержащие следующие поля: Гос. номер, ФИО водителя, Дата вызова, если количество поездок не меньше трех.
12. SELECT – вывести на экран запись – среднее расстояние одной поездки и суммарную выручку по всем автомобилям за все поездки (Стоимость (руб.) километра проезда \* Расстояние).
13. SELECT – вывести на экран записи, содержащие следующие поля: Модель автомобиля, Количество таких автомобилей, Суммарная выручка за весь период по данной модели (Время ожидания у клиента \* Стоимость (руб.) минуты простоя + Стоимость (руб.) километра проезда \* Расстояние).
14. SELECT – вывести на экран записи, содержащие следующие поля: Гос. номера, ФИО водителей, Моделей автомобилей, Даты и их выручка за день, у которых выручка была максимальной среди всех водителей за все дни.

## 2.2 Создание таблиц

В нашей базе данных используются четыре таблицы, поэтому последовательно создадим их:

DROP DATABASE work\_2;

CREATE DATABASE work\_2;

/\* -1-2- \*/

USE work\_2;

DROP TABLE IF EXISTS `Поездки`;

DROP TABLE IF EXISTS `Поездки1`;

DROP TABLE IF EXISTS `Водители`;

DROP TABLE IF EXISTS `Марки автомобилей`;

CREATE TABLE `Марки автомобилей` (

`Модель автомобиля` CHAR(35) NOT NULL,

`Минуты простоя` INT NOT NULL,

`Километра проезда` INT NOT NULL,

PRIMARY KEY(`Модель автомобиля`)

) ENGINE=InnoDB;

CREATE TABLE `Водители` (

`Гос.номер` VARCHAR(12) NOT NULL,

`ФИО Водителя` VARCHAR(36) NOT NULL,

`Телефон` CHAR(16) NOT NULL,

`Модель автомобиля` CHAR(35) NOT NULL,

PRIMARY KEY(`Гос.номер`),

FOREIGN KEY(`Модель автомобиля`)

REFERENCES `Марки автомобилей`(`Модель автомобиля`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Поездки` (

`Гос.номер` VARCHAR(12) NOT NULL,

`Дата` DATE NOT NULL,

`Время вызова` TIME NOT NULL,

`Время завершения` TIME NOT NULL,

`Время ожидания у клиента` INT NOT NULL,

`Расстояние` INT NOT NULL,

PRIMARY KEY(`Гос.номер`, `Дата`, `Время вызова`),

FOREIGN KEY(`Гос.номер`)

REFERENCES `Водители`(`Гос.номер`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Поездки1` (

`Гос.номер` VARCHAR(12) NOT NULL,

`Дата` DATE NOT NULL,

`Время вызова` TIME NOT NULL,

`Время завершения` TIME NOT NULL,

`Время ожидания у клиента` INT NOT NULL,

`Расстояние` INT NOT NULL,

PRIMARY KEY(`Гос.номер`, `Дата`, `Время вызова`),

FOREIGN KEY(`Гос.номер`)

REFERENCES `Водители`(`Гос.номер`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

## 2.3 Создание ER – диаграммы в Workbench

ER-модель (от англ. entity-relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

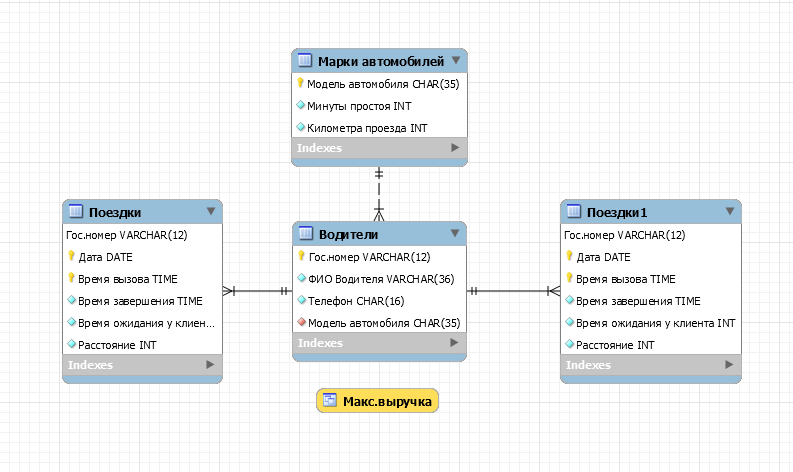


Рисунок 5 – ER-диаграмма связей между сущностями

## 2.4 Заполнение таблиц данными

После создания таблиц нужно заполнить их данными. Для этого используется оператор INSERT INTO:

INSERT INTO `Марки автомобилей`

(`Модель автомобиля`, `Минуты простоя`, `километра проезда`)

VALUES

('Kia Rio', 5, 20),

('Toyota Camry', 8, 30),

('Ford Focus', 5, 20);

INSERT INTO `Водители`

(`Гос.номер`, `ФИО водителя`, `Телефон`, `Модель автомобиля`)

VALUES

('C734XK750', 'Иванов Петр Васильевич', '908-891-78-92', 'Kia Rio'),

('C865MP750', 'Петров Андрей Иванович', '928-742-87-34', 'Kia Rio'),

('M777KM777', 'Бендер Остап Ибрагомович', '916-758-34-90', 'Toyota Camry'),

('C654PP7150', 'Фролов Виктор Валерьевич', '967-456-12-18', 'Ford Focus');

INSERT INTO `Поездки`

(`Гос.номер`, `Дата`, `Время вызова`, `Время завершения`, `Время ожидания у клиента`, `Расстояние`)

VALUES

('C734XK750', '2020.02.02', '12:20', '13:10', 2, 90),

('C734XK750', '2020.02.02', '14:45', '15:50', 5, 50),

('M777KM777', '2020.02.03', '18:30', '20:20', 5, 70),

('C865MP750', '2020.02.02', '10:00', '10:20', 2, 30);

INSERT INTO `Поездки1`

(`Гос.номер`, `Дата`, `Время вызова`, `Время завершения`, `Время ожидания у клиента`, `Расстояние`)

VALUES

('C865MP750', '2020.02.02', '12:20', '13:45', 5, 80),

('C865MP750', '2020.02.03', '10:30', '11:45', 10, 45),

('C865MP750', '2020.02.03', '23:40', '01:10', 12, 88);

## 2.5 Выборка данных из базы данных

С помощью оператора SELECT осуществляется выборка данных из баз данных. Стоит заметить, что в отличии от других операторов, SELECT имеет наиболее сложную структуру:

/\* -4- \*/

INSERT INTO `Поездки`

SELECT \* FROM `Поездки1`;

SELECT \* FROM `Поездки`;

/\* -5- \*/

UPDATE `Марки автомобилей`

SET `Километра проезда`=`Километра проезда` \* 1.05;

/\* -6- \*/

START TRANSACTION;

DELETE FROM `Поездки` WHERE `Гос.номер`='M777KM777';

DELETE FROM `Поездки1` WHERE `Гос.номер`='M777KM777';

DELETE FROM `Водители` WHERE `Гос.номер`='M777KM777';

COMMIT;

/\* -7- \*/

DROP TABLE IF EXISTS `Поездки1`;

/\* -8- \*/

SELECT `Поездки`.`Гос.номер`, `Дата`, `Время вызова`

FROM `Поездки`

WHERE `Расстояние`

BETWEEN 50 AND 80;

/\* -9- \*/

SET @INDEX='C865MP750';

SELECT `Водители`.`Гос.номер`,

`Водители`.`ФИО водителя`,

`Водители`.`Модель автомобиля`,

`Марки автомобилей`.`Километра проезда`, /\* Стоимость \*/

`Поездки`.`Дата`,

`Поездки`.`Время вызова`,

`Поездки`.`Расстояние`

FROM `Водители`

INNER JOIN `Марки автомобилей`

ON `Марки автомобилей`.`Модель автомобиля`=`Водители`.`Модель автомобиля`)

INNER JOIN `Поездки`

ON `Поездки`.`Гос.номер`=`Водители`.`Гос.номер`

WHERE `Водители`.`Гос.номер`=@INDEX;

/\* -10- \*/

SELECT `Гос.номер`, `ФИО водителя`, `Модель автомобиля`

FROM `Водители`

WHERE NOT EXISTS

(SELECT \* FROM `Поездки`

WHERE `Водители`.`Гос.номер`=`Гос.номер`);

/\* -11- \*/

SELECT `Поездки`.`Гос.номер`,

`Водители`.`ФИО водителя`,

`Поездки`.`Дата`,

COUNT(\*) AS `Количество поездок`

FROM `Поездки`

INNER JOIN `Водители`

ON `Водители`.`Гос.номер`=`Поездки`.`Гос.номер`

GROUP BY 1,2,3 HAVING COUNT(\*)>=2 ;

/\* -12- \*/

SELECT AVG(`Расстояние`) AS `Среднее расстояние`,

SUM(`Расстояние` \* `Километра проезда`) AS `Сумм.стоимость`

FROM `Водители`

INNER JOIN `Марки автомобилей`

ON `Марки автомобилей`.`Модель автомобиля` = `Водители`.`Модель автомобиля`

LEFT JOIN `Поездки`

ON `Поездки`.`Гос.номер` = `Водители`.`Гос.номер`;

/\* -13- \*/

SELECT `Водители`.`Модель автомобиля`,

COUNT(DISTINCT `Водители`.`Гос.номер`) AS `Кол-во автомобилей`,

SUM(`Время ожидания у клиента` \* `Минуты простоя` + `Километра проезда` \* `Расстояние`)

AS `Выручка`

FROM `Марки автомобилей`

INNER JOIN `Водители`

ON `Водители`.`Модель автомобиля` = `Марки автомобилей`.`Модель автомобиля`

LEFT JOIN `Поездки`

ON `Поездки`.`Гос.номер` = `Водители`.`Гос.номер`

GROUP BY `Водители`.`Модель автомобиля`;

/\* -14- \*/

/\* сначала находим выручку по каждому гос.номеру за один день \*/

DROP VIEW IF EXISTS `Макс.выручка`;

CREATE VIEW `Макс.выручка` AS

SELECT `Водители`.`Гос.номер`,

`Водители`.`ФИО водителя`,

`Водители`.`Модель автомобиля`,

SUM(`Время ожидания у клиента` \* `Минуты простоя` + `Километра проезда` \* `Расстояние`) AS `Выручка`

FROM `Водители`

INNER JOIN `Марки автомобилей`

ON `Марки автомобилей`.`Модель автомобиля` = `Водители`.`Модель автомобиля`

LEFT JOIN `Поездки`

ON `Поездки`.`Гос.номер` = `Водители`.`Гос.номер`

GROUP BY `Водители`.`Гос.номер`, `Дата`;

/\* потом находим самую большую выручку среди всех гос.номеров \*/

SELECT \* FROM `Макс.выручка`

WHERE `Выручка` = (SELECT MAX(`Выручка`) FROM `Макс.выручка`);

# 3 Хранимые процедуры и функции, триггеры и назначение прав доступа

## 3.1 Условия задания варианта № 17

Необходимо в СУБД MySQL создать:

1. Базу данных «Такси», состоящую из 3-х связанных таблиц, представленных ниже:

Марки автомобилей:

|  |  |  |
| --- | --- | --- |
| Модель автомобиля | Стоимость (руб.) | |
| минуты простоя | километра проезда |
| Kia Rio[[3]](#footnote-3) | 5 | 20 |
| Toyota Camry | 8 | 30 |

Водители:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Гос. номер | ФИО водителя | Телефон | Модель автомобиля | Итоговая выручка (руб.) |
| С734ХК750 | Иванов Петр Васильевич | 907-891-78-92 | Kia Rio | 0 |
| С865МР750 | Петров Андрей Иванович | 928-742-87-34 | Kia Rio | 0 |
| М777КМ777 | Бендер Остап Ибрагимович | 916-758-34-90 | Toyota Camry | 0 |

Поездки:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Гос. номер | Дата | Время вызова | Время завершения | Время ожидания у клиента | Расстояние |
| С734ХК750 | 02.02.2020 | 12-20 | 13-10 | 2 | 90 |
| С734ХК750 | 02.02.2020 | 14-45 | 15-50 | 5 | 50 |
| М777КМ777 | 03.02.2020 | 18-30 | 20-20 | 5 | 70 |
| С865МР750 | 02.02.2020 | 10-00 | 10-50 | 2 | 30 |
| С865МР750 | 02.02.2020 | 12-00 | 13-45 | 5 | 80 |
| С865МР750 | 03.02.2020 | 10-30 | 11-45 | 10 | 45 |
| С865МР750 | 03.02.2020 | 23-40 | 01-10 | 12 | 95 |

1. Хранимую функцию, которая использует данные из таблицы «Поездки» и подсчитывает итоговую выручку в рублях (зависящую от модели автомобиля, времени ожидания и пройденного расстояния) за все время работы указанного водителя.
2. Хранимую процедуру, которая использует хранимую функцию, созданную в предыдущем пункте, для заполнения полей в столбце «Итоговая выручка (руб.)» по всем записям таблицы «Водители».
3. Хранимую процедуру, которая реализует задание пункта 3, но с использование курсора.

Если таблица «Поездки» будет содержать большой объем данных, то для получения итоговых значений в таблице «Водители» с помощью выполнения хранимой функции, созданной в предыдущем задании, потребуется достаточно продолжительное время. В этом случае использование триггеров может решить указанную проблему. Требуется создать триггеры, которые срабатывают при изменении данных в таблице «Поездки», а именно:

1. при удалении записи из таблицы «Поездки» требуется уменьшить значение в соответствующем поле столбца «Итоговая выручка (руб.)» таблицы «Водители» на значение суммы в рублях (зависящей от модели автомобиля, времени ожидания и пройденного расстояния удаляемой записи таблицы «Поездки»);
2. при добавлении записи в таблицу «Поездки» требуется значения полей «Время ожидания у клиента (мин.)» и «Расстояние (км)» создаваемой записи таблицы «Поездки» использовать для добавления в значение соответствующего поля столбца «Итоговая выручка (руб.)» таблицы «Водители» суммы в рублях, зависящей от модели автомобиля, времени ожидания и пройденного расстояния;
3. при обновлении записи в таблице «Поездки» требуется выполнить пункт 5 для необновленной (старой) записи таблицы «Водители», а затем выполнить пункт 6 для обновленной (новой) записи таблицы «Поездки»;

Одной из важных задач администрирования системы является создание пользователей и назначение им прав доступа к БД. Требуется создать пользователей и назначить им права доступа, согласно перечисленным ниже требованиям:

1. Создать пользователей: administrator (администратор), director (директор), worker (работник) и visitor (посетитель).
2. Назначить пользователю administrator все права доступа, в том числе создания новых таблиц и пользователей, их модификации и удаления, кроме создания новых и удаления существующих баз данных.
3. Назначить пользователю director все права доступа ко всем существующим таблицам, кроме создания новых баз данных, таблиц и пользователей, их модификации и удаления.
4. Назначить пользователю worker следующие права доступа:

* к таблице «Марки автомобилей» по созданию, просмотру и обновлению записей, кроме полей «Стоимость (руб.) минуты простоя и километра проезда», доступ к которому ограничить созданием и просмотром;
* к таблице «Водители» по созданию и просмотру записей;
* к таблице «Поездки» по созданию и просмотру записей, в том числе по обновлению полей «Время ожидания у клиента (мин.)» и «Расстояние (км)»;

1. Создать представление (виртуальную таблицу), содержащую следующие поля: «Модель автомобиля», «Гос. номер», «ФИО водителя», «Дата» и «Время вызова».
2. Назначить права доступа visitor только к данному представлению на просмотр.

## 3.2 Создание базы данных, таблиц и заполнения

/\* -1- \*/

CREATE TABLE `Марки автомобилей` (

`Модель автомобиля` CHAR(35) NOT NULL,

`Минуты простоя` INT NOT NULL,

`Километра проезда` INT NOT NULL,

PRIMARY KEY(`Модель автомобиля`)

) ENGINE=InnoDB;

CREATE TABLE `Водители` (

`Гос.номер` VARCHAR(12) NOT NULL,

`ФИО водителя` VARCHAR(36) NOT NULL,

`Телефон` CHAR(16) NOT NULL,

`Модель автомобиля` CHAR(35) NOT NULL,

`Итоговая выручка` INT NOT NULL,

PRIMARY KEY(`Гос.номер`),

FOREIGN KEY(`Модель автомобиля`)

REFERENCES `Марки автомобилей`(`Модель автомобиля`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

CREATE TABLE `Поездки` (

`Гос.номер` VARCHAR(12) NOT NULL,

`Дата` DATE NOT NULL,

`Время вызова` TIME NOT NULL,

`Время завершения` TIME NOT NULL,

`Время ожидания у клиента` INT NOT NULL,

`Расстояние` INT NOT NULL,

PRIMARY KEY(`Гос.номер`, `Дата`, `Время вызова`),

FOREIGN KEY(`Гос.номер`)

REFERENCES `Водители`(`Гос.номер`)

ON DELETE RESTRICT ON UPDATE CASCADE

) ENGINE=InnoDB;

INSERT INTO `Марки автомобилей`

(`Модель автомобиля`, `Минуты простоя`, `километра проезда`)

VALUES

('Kia Rio', 5, 20),

('Toyota Camry', 8, 30);

INSERT INTO `Водители`

(`Гос.номер`, `ФИО водителя`, `Телефон`, `Модель автомобиля`, `Итоговая выручка`)

VALUES

('C734XK750', 'Иванов Петр Васильевич', '908-891-78-92', 'Kia Rio', 0),

('C865MP750', 'Петров Андрей Иванович', '928-742-87-34', 'Kia Rio', 0),

('M777KM777', 'Бендер Остап Ибрагомович', '916-758-34-90', 'Toyota Camry', 0);

INSERT INTO `Поездки`

(`Гос.номер`, `Дата`, `Время вызова`, `Время завершения`, `Время ожидания у клиента`, `Расстояние`)

VALUES

('C734XK750', '2020.02.02', '12:20', '13:10', 2, 90),

('C734XK750', '2020.02.02', '14:45', '15:50', 5, 50),

('M777KM777', '2020.02.03', '18:30', '20:20', 5, 70),

('C865MP750', '2020.02.02', '10:00', '10:20', 2, 30),

('C865MP750', '2020.02.02', '12:20', '13:45', 5, 80),

('C865MP750', '2020.02.03', '10:30', '11:45', 10, 45),

('C865MP750', '2020.02.03', '23:40', '01:10', 12, 95);

## 3.3 Создание хранимых процедур, хранимых функций и триггеров

Хранимые процедуры представляют собой набор команд SQL, которые могут компилироваться и храниться на сервере. Таким образом, вместо того, чтобы хранить часто используемый запрос, клиенты могут ссылаться на соответствующую хранимую процедуру. Это обеспечивает лучшую производительность, поскольку данный запрос должен анализироваться только однажды и уменьшается трафик между сервером и клиентом. Концептуальный уровень можно также повысить за счет создания на сервере библиотеки функций.

Триггер представляет собой хранимую процедуру, которая активизируется при наступлении определенного события. Например, можно задать хранимую процедуру, которая срабатывает каждый раз при удалении записи из транзакционной таблицы - таким образом, обеспечивается автоматическое удаление соответствующего заказчика из таблицы заказчиков, когда все его транзакции удаляются.

/\* -2- \*/

DELIMITER $$

CREATE FUNCTION getSummary(car\_number VARCHAR(12))

RETURNS INT

DETERMINISTIC

BEGIN

DECLARE SUMMARY INT DEFAULT 0;

SELECT SUM(`Время ожидания у клиента` \* `Минуты простоя` + `Километра проезда` \* `Расстояние`) AS `Суммарная выручка`

INTO SUMMARY

FROM `Марки автомобилей`

INNER JOIN `Водители`

ON `Водители`.`Модель автомобиля`=`Марки автомобилей`.`Модель автомобиля`

LEFT JOIN `Поездки`

ON `Поездки`.`Гос.номер`=`Водители`.`Гос.номер`

WHERE `Водители`.`Гос.номер`= car\_number

GROUP BY `Водители`.`Гос.номер`;

RETURN IFNULL(SUMMARY, 0);

END$$

DELIMITER ;

SELECT \* FROM `Поездки`;

SELECT getSummary('C734XK750') AS `Суммарная выручка`;

/\* -3- \*/

/\* если водитель не выйдет на линию, то его "Итоговая выручка" будет равна 0 \*/

DELIMITER $$

CREATE PROCEDURE setSummary()

BEGIN

UPDATE `Водители` SET `Итоговая выручка` = getSummary(`Гос.номер`);

END$$

DELIMITER ;

CALL setSummary();

SELECT \* FROM `Водители`;

/\* -4-5- \*/

DELIMITER $$

CREATE PROCEDURE `setCursor`()

BEGIN

DECLARE car\_number VARCHAR(12);

DECLARE SUMMARY, base INT DEFAULT 0;

/\* здесь тоже самое что и при реализации функции \*/

DECLARE cursor1 CURSOR FOR SELECT `Водители`.`Гос.номер`,

SUM(`Время ожидания у клиента` \* `Минуты простоя` + `Километра проезда` \* `Расстояние`) AS `Суммарная выручка`

FROM `Марки автомобилей`

INNER JOIN `Водители`

ON `Водители`.`Модель автомобиля` = `Марки автомобилей`.`Модель автомобиля`

INNER JOIN `Поездки`

ON `Поездки`.`Гос.номер` = `Водители`.`Гос.номер`

GROUP BY `Водители`.`Гос.номер`;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET base = 1;

UPDATE `Водители`

SET `Итоговая выручка` = 0;

OPEN cursor1;

WHILE base = 0 DO

FETCH cursor1 INTO car\_number, SUMMARY;

UPDATE `Водители` SET `Итоговая выручка` = SUMMARY

WHERE `Гос.номер` = car\_number;

END WHILE;

CLOSE cursor1;

END$$

DELIMITER ;

CALL setCursor();

SELECT \* FROM `Водители`;

DELIMITER $$

CREATE TRIGGER `deleteTrack`

AFTER DELETE ON `Поездки` FOR EACH ROW

BEGIN

UPDATE `Водители`

SET `Итоговая выручка` = getSummary(`Гос.номер`)

WHERE `Гос.номер` = OLD.`Гос.номер`;

END$$

DELIMITER ;

DELETE FROM `Поездки` WHERE `Гос.номер` = 'C734XK750'

AND `Дата` = '2020.02.02'

AND `Время вызова` = '12:20';

SELECT \* FROM `Поездки`;

SELECT \* FROM `Водители`;

/\* -6- \*/

DELIMITER $$

CREATE TRIGGER `insertTrack`

AFTER INSERT ON `Поездки` FOR EACH ROW

BEGIN

UPDATE `Водители`

SET `Итоговая выручка` = getSummary(`Гос.номер`)

WHERE `Гос.номер` = NEW.`Гос.номер`;

END$$

DELIMITER ;

INSERT INTO `Поездки`

(`Гос.номер`, `Дата`, `Время вызова`, `Время завершения`, `Время ожидания у клиента`, `Расстояние`)

VALUES

('M777KM777', '2020.02.03', '18:35', '20:25', 6, 75);

SELECT \* FROM `Поездки`;

SELECT \* FROM `Водители`;

/\* -7- \*/

DELIMITER $$

CREATE TRIGGER `updateTrack`

AFTER UPDATE ON `Поездки` FOR EACH ROW

BEGIN

if OLD.`Гос.номер` != NEW.`Гос.номер` THEN

UPDATE `Водители` SET `Итоговая выручка` = getSummary(`Гос.номер`)

WHERE `Гос.номер` = OLD.`Гос.номер`;

END IF;

UPDATE `Водители` SET `Итоговая выручка` = getSummary(`Гос.номер`)

WHERE `Гос.номер` = NEW.`Гос.номер`;

END$$

DELIMITER ;

UPDATE `Поездки`

SET `Время ожидания у клиента` = 5, `Расстояние` = 45

WHERE `Гос.номер` = 'C734XK750' AND `Дата` = '2020.02.02'

AND `Время вызова` = '12:20';

SELECT \* FROM `Поездки`;

SELECT \* FROM `Водители`;

## 3.4 Назначение прав доступа к базам данных

Ниже представлен короткий список некоторых возможных вариантов прав доступа, которые могут получить пользователи:

* ALL PRIVILEGES - это даст пользователю MySQL полный доступ к заданной базе данных (если база данных не указана, то ко всем), кроме назначения прав доступа другим пользователям.
* CREATE - позволяет создавать новые таблицы или базы данных.
* DROP - позволяет удалять таблицы или базы данных.
* DELETE - позволяет удалять строки из таблиц.
* INSERT - позволяет добавлять строки в таблицу.
* ALTER - используется для добавления, удаления или модификации колонки в уже существующей таблице.
* SELECT - позволит использовать команду SELECT для чтения из баз данных.
* UPDATE - позволит редактировать строки таблиц.
* GRANT OPTION - позволит назначать или удалять права доступа для других пользователей.

Имеем:

/\* -8- \*/

DROP USER 'administrator'@'localhost';

DROP USER 'director'@'localhost';

DROP USER 'worker'@'localhost';

DROP USER 'visitor'@'localhost';

CREATE USER 'administrator'@'localhost' IDENTIFIED BY 'password';

CREATE USER 'director'@'localhost' IDENTIFIED BY 'password';

CREATE USER 'worker'@'localhost' IDENTIFIED BY 'password';

CREATE USER 'visitor'@'localhost';

/\* -9- \*/

GRANT ALL PRIVILEGES ON `WORK\_3`.\*

TO 'administrator'@'localhost' WITH GRANT OPTION;

FLUSH PRIVILEGES;

/\* -10- \*/

GRANT ALL PRIVILEGES ON `WORK\_3`.\*

TO 'director'@'localhost';

REVOKE CREATE, ALTER, DROP ON `WORK\_3`.\*

FROM 'director'@'localhost';

FLUSH PRIVILEGES;

/\* -11- \*/

GRANT INSERT, SELECT ON `WORK\_3`.`Марки автомобилей`

TO 'worker'@'localhost';

GRANT UPDATE(`Модель автомобиля`) ON `WORK\_3`.`Марки автомобилей`

TO 'worker'@'localhost';

GRANT INSERT, SELECT ON `WORK\_3`.`Водители`

TO 'worker'@'localhost';

GRANT UPDATE(`Время ожидания у клиента`, `Расстояние`), INSERT, SELECT ON `WORK\_3`.`Поездки`

TO 'worker'@'localhost';

FLUSH PRIVILEGES;

/\* -12- \*/

DROP VIEW IF EXISTS `Просмотр`;

CREATE VIEW `Просмотр` AS

SELECT `Водители`.`Модель автомобиля`,

`Водители`.`Гос.номер`,

`Водители`.`ФИО водителя`,

`Поездки`.`Дата`,

`Поездки`.`Время вызова`

FROM `Водители`

INNER JOIN `Поездки`

ON `Поездки`.`Гос.номер`=`Водители`.`Гос.номер`;

/\* -13- \*/

GRANT SELECT ON `WORK\_3`.`Просмотр` TO

'visitor'@'localhost';

FLUSH PRIVILEGES;

SHOW GRANTS FOR 'administrator'@'localhost';

SHOW GRANTS FOR 'director'@'localhost';

SHOW GRANTS FOR 'worker'@'localhost';

SHOW GRANTS FOR 'visitor'@'localhost';

# Заключение

Данные в базе данных являются интегрированными и общими, поэтому является наиболее важным ее преимуществом.

Интеграция данных подразумевает возможность представить базу данных как объединение нескольких отдельных файлов данных, полностью или частично не перекрывающихся.

Общие данные подразумевают возможность использования отдельных областей в базе данных несколькими различными пользователями.

Между физической базой данных и пользователями системы располагается уровень программного обеспечения – диспетчер базы данных или система управления базой данных (СУБД). Основная функция СУБД – предоставление возможности пользователю баз данных работать с ней, не вникая в детали на уровне аппаратного обеспечения.

В результате выполнения курсовой работы были выполнены следующие мероприятия:

1. Создание нормализованной базы данных.
2. Создание, заполнение таблиц базы данных и выполнение запросов на языке SQL.
3. Создание хранимых процедур, хранимых функций и триггеров, а также создание пользователей и назначение прав доступа к базе данных.

Следующие компетенции были освоены в результате выполнения курсовой работы:

* ОПК-1 способностью инсталлировать программное и аппаратное обеспечение для информационных и автоматизированных систем:
* знать реляционную модель данных;
* уметь инсталлировать программное и аппаратное обеспечение для СУБД;
* владеть навыками работы с программным и аппаратным обеспечением СУБД.
* Способностью осваивать методики использования программных средств для решения практических задач:
* знать типы и свойства полей, основные объекты баз данных;
* знать операции, выполняемые с базами данных;
* знать язык запросов SQL;
* уметь формировать запросы к базе данных;
* владеть навыками проектирования баз данных.

# Список использованной литературы

1. Затонский А.В. Информационные технологии: разработка информационных моделей и систем: Учебное пособие [Электронный ресурс]/ А.В.Затонский - М.: ИЦ РИОР: НИЦ ИНФРА-М, 2014 - 344с. // ЭБС «Znanium». – URL: http://znanium.com/catalog.php?bookinfo=400563 (Дата обращения 26.03.2020)
2. Дадян Э.Г. Методы, модели, средства хранения и обработки данных: Учебник / Э.Г. Дадян, Ю.А. Зеленков. — М.: Вузовский учебник: ИНФРА-М, 2017//ЭБС «Znanium».-URL: http://znanium.com/catalog.php?bookinfo=543943 (Дата обращения 26.03.2020)
3. Голицына О.Л. Информационные системы: Учебное пособие [Электронный ресурс]/ О.Л. Голицына, Н.В. Максимов, И.И. Попов. - 2-e изд. - М.: Форум: НИЦ ИНФРА-М, 2014. - 448 с. // ЭБС «Znanium». – URL: http://znanium.com/catalog.php?bookinfo=435900 (Дата обращения 26.03.2020)
4. Аллен, Г. Тейлор SQL для чайников / Аллен Г. Тейлор. - М.: Диалектика, Вильямс, 2015. - 416 c. (Дата обращения 26.03.2020)
5. Базы данных: Учеб. для вузов / Под ред. А. Д. Хомоненко. СПб.: КОРОНА принт, 2015. 416 с. (Дата обращения 26.03.2020)
6. Donald Bales. Beginning Oracle PL/SQL. Second edition. - New York: Apress, 2015. - 492 p. (Дата обращения 26.03.2020)
7. Глушаков, С.В. Базы данных / С.В. Глушаков, Д.В. Ломотько. - М.: Харьков: Фолио, 2016. - 504 c. (Дата обращения 26.03.2020)
8. Дженнингс Р. Использование Access: Пер. с англ. 2-е спец. изд. М.; Спб.; Киев: Вильяме, 2017. - 944 с. (Дата обращения 26.03.2020)
9. Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2017. - 288 c. (Дата обращения 26.03.2020)
10. Чаллавала Шаббир, Лакхатария Джадип, Мехта Чинтан. MySQL 8 для больших данных.-М: ДМК-Пресс, 2019.- 220с. (Дата обращения 26.03.2020)
11. Дейт, К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL / К.Дж. Дейт. - М.: Символ-плюс, 2016. - 0 c. (Дата обращения 26.03.2020)
12. Дунаев, В. В. Базы данных. Язык SQL для студента / В.В. Дунаев. - М.: БХВ-Петербург, 2017. - 288 c. (Дата обращения 26.03.2020)
13. Карвин, Билл Программирование баз данных SQL. Типичные ошибки и их устранение / Билл Карвин. - М.: Рид Групп, 2018. - 336 c. (Дата обращения 26.03.2020)
14. Кригель, А. SQL. Библия пользователя / А. Кригель. - М.: Диалектика / Вильямс, 2016. - 0 c. (Дата обращения 26.03.2020)
15. Майкл, Мак-Локлин Oracle Database 11g. Программирование на языке PL/SQL / Мак-Локлин Майкл. - М.: ДМК Пресс, 2017. - 0 c. (Дата обращения 26.03.2020)
16. Пржиялковский, В. В. Введение в Oracle SQL / В.В. Пржиялковский. - М.: Бином. Лаборатория знаний, Интернет-университет информации-онных технологий, 2016. - 320 c. (Дата обращения 26.03.2020)
17. Проектирование и реализация баз данных Microsoft SQL Server 2000. Учебный курс MCSE (+ CD-ROM). - М.: Русская Редакция, 2017. - 664 c.
18. Селко, Джо SQL для профессионалов. Программирование / Джо Селко. - М.: ЛОРИ, 2015. - 464 c. (Дата обращения 26.03.2020)
19. Форта, Бен. Освой самостоятельно SQL за 10 минут / Бен Форта. - М.: Вильямс, 2015. - 288 c. (Дата обращения 26.03.2020)
20. Л.И. Шустова, О.В. Тараканов. Базы данных. Учебник. – М.: Инфра-М, 2018. – 304 с. (Дата обращения 26.03.2020)
21. Карвин, Билл Программирование баз данных SQL. Типичные ошибки и их устранение / Билл Карвин. - М.: Рид Групп, 2018. - 336 c. (Дата обращения 26.03.2020)
22. Л.Г. Гагарина. Разработка и эксплуатация автоматизированных информационных систем. Учебное пособие. – М.: Форум, Инфра-М, 2015. – 384 с. (Дата обращения 26.03.2020)
23. Кумскова И.А. Базы данных. Учебник. – М.: КноРус, 2016. – 400 с. (Дата обращения 26.03.2020)
24. Грабер, Мартин SQL для простых смертных / Мартин Грабер. - М.: ЛОРИ, 2014. - 378 c. (Дата обращения 26.03.2020)

1. Отсутствие в поле записи «Дата исполнения заказ» означает, что этот заказ не выполнен [↑](#footnote-ref-1)
2. Ключевые поля подчеркнуты [↑](#footnote-ref-2)
3. Ключевые поля подчеркнуты [↑](#footnote-ref-3)