

ЛАБОРАТОРНАЯ РАБОТА 1. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

Цель: спроектировать БД для выбранной предметной области согласно примеру, представленном в методическом указании. Провести нормализацию (до 3 нормальной формы).

ЗАДАНИЕ

- 1) Описать предметную область
- 2) Выделить ключевые объекты системы
- 3) Провести инфологическое проектирование
 - a. Составить и прокомментировать ER-диаграмму
 - b. Составить и прокомментировать уточненную ER-диаграмму (с атрибутами)
- 4) Провести логическое проектирование
- 5) Провести нормализацию (до 3 нормальной формы)
- 6) Описать ключевые ограничения

Примечание:

Для проектирования рекомендуется использовать приложение Oracle SQL Developer Data Modeler

(<http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html>)

или Astah Professional

(<http://astah.net/features/er-diagram>).

Также вы можете осуществить проектирование при помощи векторного графического редактора, редактора диаграмм и блок-схем – Microsoft Visio. Детальное описание установки редактора вы можете найти в данном методическом указании.

После нормализации количество таблиц должно не превышать 7, желательно 5 таблиц.

ХОД РАБОТЫ

1. Выбрать вариант задания
2. Провести инфологическое проектирование проанализировав предметную область согласно варианту задания. Разработать ER-диаграмму сущностей
3. Осуществить процесс логического проектирования, подробно расписав процесс преобразования ER-диаграммы в схему отношений. Учитывая знания, полученные по нормализации отношений.
4. Провести нормализацию схемы, полученной в результате выполнения лабораторной работы. В результате у Вас должны получиться схемы отношений, представленные в табличном виде (как в примере таблицы с 1.8 по 1.17).
5. Подготовить отчет о проделанной работе. Структура отчета:
 - титульный лист;
 - задание;

– описание процесса проектирования (инфологическое проектирование и ER-модель, аналогично примеру, представленному в данном методическом указании.);

– заключение;

Номер варианта задания определяется по последним двум цифрам номера зачетной книжки. Если образуемое ими число больше 24, то следует взять сумму этих цифр

ВАРИАНТЫ ЗАДАНИЙ

1 музей;	2 минимаркет;	3 поликлиника;
4 пиццерия;	5 прокат;	6 гостиница;
7 документооборот;	8 строительная компания;	9 спортивный клуб;
10 завод по изготовлению автомобильных деталей;	11 транспортная компания;	12 туристическая компания;
13 картинная галерея;	14 товары-почтой;	15 автомастерская;
16 книжный склад;	17 авиакомпания;	18 аудио коллекция;
19 компания по сбыту лекарственных препаратов;	20 фирма по ремонту;	21 касса театра.
22 кулинария;	23 деканат;	24 поликлиника;

ОГЛАВЛЕНИЕ

ЗАДАНИЕ	1
ХОД РАБОТЫ	1
ВАРИАНТЫ ЗАДАНИЙ.....	2
1. КРАТКАЯ ТЕОРИЯ.....	4
1.1. Основные понятия.....	5
1.2. Этапы проектирования базы данных:	7
1.3. Инфологическое проектирование.....	7
1.3.1. Entity-Relationship Diagrams	8
1.3.2. Логическое проектирование.....	9
1.3.3. Физическое проектирование	10
1.4. Пример проектирования реляционной базы данных	10
1.4.1. Инфологическое проектирование	10
1.4.2. Логическое проектирование реляционной БД	12
1.5. Нормализация полученных отношений (до 3НФ)	19
1.5.1. Первая нормальная форма.....	19
1.5.2. Вторая нормальная форма	19
1.5.3. Третья нормальная форма	20
1.6. Нормализация отношений для БД «Издательская компания».....	21
1.6.1.1. 1 НФ (нормальная форма)	21
1.6.1. 2 НФ (нормальная форма)	22
1.6.2. 3 НФ (нормальная форма)	22
1.6.3. Окончательные схемы отношений	24
1.7. Пример нормализации отношение №3.....	26
1.7.1. 1 НФ (нормальная форма)	26
1.7.2. Определение функциональной зависимости	27
1.7.3. 2 НФ (нормальная форма)	27
1.7.4. 3НФ (нормальная форма)	29
1.7.5. Вывод.....	30
1.7.6. Определение дополнительных ограничений целостности	30
2. ЗАКЛЮЧЕНИЕ	31

1. КРАТКАЯ ТЕОРИЯ

Перед созданием базы данных разработчик должен определить, из каких таблиц должна состоять база данных, какие данные нужно поместить в каждую таблицу, как связать таблицы. Эти вопросы решаются на этапе проектирования базы данных.

В результате проектирования должна быть определена логическая структура базы данных, то есть состав реляционных таблиц, их структура и межтабличные связи.

Перед созданием базы данных необходимо располагать описанием выбранной предметной области, которое должно охватывать реальные объекты и процессы, определить все необходимые источники информации для удовлетворения предполагаемых запросов пользователей и определить потребности в обработке данных.

На основе такого описания на этапе проектирования базы данных определяются состав и структура данных предметной области, которые должны находиться в БД и обеспечивать выполнение необходимых запросов и задач пользователей. Структура данных предметной области может отображаться информационно-логической моделью. На основе этой модели легко создается реляционная база данных.

1.1. Основные понятия

Понятие реляционный (англ. Relation – отношение) связано с разработками известного английского специалиста в области систем баз данных Эдгара Кодда (Edgar Codd).

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все ячейки в столбце таблицы однородные, то есть все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.);
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Основными понятиями реляционных баз данных являются тип данных, домен, атрибут, кортеж, первичный ключ и отношение.

Для начала покажем смысл этих понятий на примере отношения СОТРУДНИКИ, содержащего информацию о сотрудниках некоторой организации как показано на рисунке 1.1.

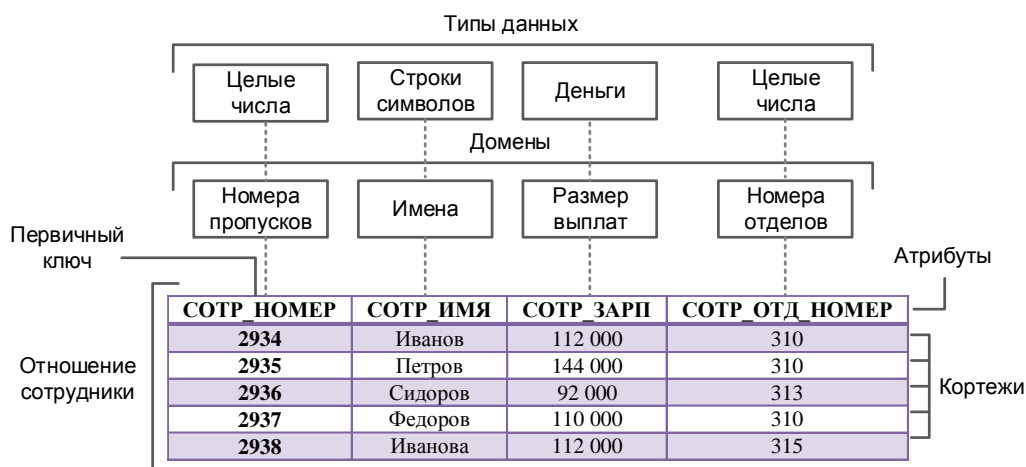


Рисунок 1.1 – Схема реляционного отношения

Тип данных в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых данных (таких как "деньги"), а также специальных "темпоральных" данных (дата, время, временной интервал).

Понятие **домена** более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, домен "Имена" в схеме, показанной выше, определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, такие строки не могут начинаться с мягкого знака).

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение – это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

В свою очередь отношения имеют два важных свойства:

арность – число атрибутов в отношении;

мощность – это кардинальное число отношения, т.е. число кортежей (строк) в отношении.

Каждое реляционное отношение соответствует одной сущности (объекту предметной области) и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи.

Ключ или **потенциальный ключ** – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна, но не обязана обладать хотя бы одним возможным ключом. Другими словами **ключ – это поле или набор полей, однозначно идентифицирующий запись.**

Значение первичного ключа в таблице БД должно быть уникальным, то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа.

Первичные ключи облегчают установление связей между таблицами. Поскольку первичный ключ должен быть уникальным, для него могут использоваться не все поля таблицы.

В том случае, если базовое отношение не имеет потенциальных ключей, вводится **суррогатный первичный ключ**, который не несёт смысловой нагрузки и служит только для идентификации записей (например ID записи).

Примечание: суррогатный первичный ключ также может вводиться в тех случаях, когда потенциальный ключ имеет большой размер (например, длинная символьная строка) или является составным (не менее трёх атрибутов).

Потенциальными ключами отношения АВТОРЫ являются атрибуты Паспортные данные и ИНН. Первый хранится как длинная строка, а последний по условиям предметной области не является обязательным. Поэтому для авторов необходимо ввести суррогатный ключ – A_id. Книги можно идентифицировать по атрибуту Контракт: его номер обязателен и уникален. Потенциальные ключи отношения СОТРУДНИКИ – атрибуты ИНН, Паспортные данные, Табельный номер, причём все они обязательные. Табельный номер занимает меньше памяти, чем ИНН, поэтому он и будет первичным ключом. Кортежи отношения ЗАКАЗЫ можно идентифицировать ключом Номер заказа.

Потенциальными ключами вспомогательных отношений являются комбинации первичных ключей соответствующих базовых отношений.

Отношения приведены в таблице 1.1-1.7. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: **N** – **числовой**, **C** – **символьный**, **D** – **дата** (последний имеет стандартную длину, зависящую от СУБД, поэтому она не указывается).

1.2. Этапы проектирования базы данных:

- инфологическое проектирование;
- определение требований к операционной обстановке, в которой будет функционировать информационная система;
- выбор системы управления базой данных (СУБД) и других инструментальных программных средств;
- логическое проектирование БД;
- физическое проектирование БД.

1.3. Инфологическое проектирование

Инфологическое проектирование – построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создается без ориентации на какую-либо конкретную СУБД и модель данных. Конкретный вид и содержание концептуальной модели базы данных определяется выбранными для этого

формальным аппаратом. Обычно используют графические нотации, подобные ER-диаграммам.

Чаще всего инфологическая (концептуальная) модель базы данных включает в себя:

– описание информационных объектов или понятий предметной области и связей между ними;

– описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

Пример инфологического проектирования показан на рисунке 1.



Рисунок 1 – Инфологическая модель

1.3.1. Entity-Relationship Diagrams

Имеется целый ряд методик создания информационно-логических моделей. Одна из наиболее популярных в настоящее время методик при разработке моделей использует **ERD** (Entity-Relationship Diagrams). В русскоязычной литературе эти диаграммы называют «объект – отношение» либо «сущность – связь». Модель ERD была предложена Питером Пин Шен Ченом в 1976 г. К настоящему времени разработано несколько ее разновидностей, но все они базируются на графических диаграммах, предложенных Ченом. Диаграммы конструируются из небольшого числа компонентов. Благодаря наглядности представления они широко используются в CASE-средствах (Computer Aided Software Engineering).

Рассмотрим используемую терминологию и обозначения.

Сущность (Entity) – реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению.

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа (сущности).

Каждая сущность должна обладать некоторыми свойствами:

– иметь уникальное имя; причем к этому имени должна всегда применяться одна и та же интерпретация (определение сущности). И наоборот: одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;

– обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются ею через связь;

– обладать одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности.

Связь (Relationship) – поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области. Одна из участвующих в связи сущностей – независимая, называется родительской сущностью, другая – зависимая, называется дочерней или сущностью-потомком. Как правило, каждый экземпляр

родительской сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров дочерней сущности. Каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя. Таким образом, экземпляр сущности-потомка может существовать только при существовании сущности-родителя.

Связи дается имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи.

Атрибут – любая характеристика сущности, значимая для рассматриваемой предметной области. Он предназначен для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик (свойств), ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т. д.)

Уникальный идентификатор – это атрибут или совокупность атрибутов и/или связей, однозначно характеризующая каждый экземпляр данного типа сущности. В случае полной идентификации экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в идентификации участвуют также атрибуты другой сущности – родителя.

При построении ER-модели используются следующие принципы:

- сущности на диаграмме представляются прямоугольниками;
- каждый прямоугольник может иметь различные визуальные атрибуты;
- каждой сущности должно быть присвоено уникальное имя;
- имена сущностей необходимо задавать в единственном числе;
- связи на диаграмме представляются линиями, идущими от одной сущности (таблицы) к другой;
- каждой связи присваивается уникальное имя;
- связанные таблицы разделяют на родительские и дочерние;
- родительские таблицы отображаются прямоугольниками с прямыми углами, дочерние – со скругленными.

1.3.2. Логическое проектирование

Логическое проектирование – создание схемы базы данных на основе конкретной модели базы данных, например, реляционной модели данных. Для реляционной модели данных даталогическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Пример логической модели представлен на рисунке 2.

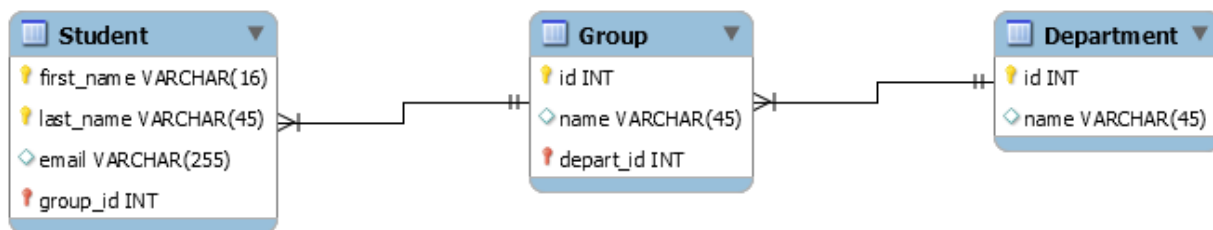


Рисунок 2 – Логическая модель

1.3.3. Физическое проектирование

Физическое проектирование – создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.

Результатом физического проектирования, например, может стать скрипт на языке SQL.

1.4. Пример проектирования реляционной базы данных

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью.

1.4.1. Инфологическое проектирование

Анализ предметной области

База данных создаётся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчёты.


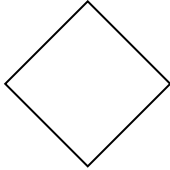

В соответствии с предметной областью система строится с учётом следующих особенностей:

- каждая книга издаётся в рамках контракта;
- книга может быть написана несколькими авторами;
- контракт подписывается одним менеджером и всеми авторами книги;
- каждый автор может написать несколько книг (по разным контрактам);
- порядок, в котором авторы указаны на обложке, влияет на размер гонорара;
- если сотрудник является редактором, то он может работать одновременно над несколькими книгами;
- у каждой книги может быть несколько редакторов, один из них – ответственный редактор;
- каждый заказ оформляется на одного заказчика;
- в заказе на покупку может быть перечислено несколько книг.

Для инфологического проектирования воспользуемся методом «сущность-связь». Для того, чтобы представить, как устроена предметная область нужно задать множество объектов реального мира (главная проблема что считать объектом). Объект – семантическое понятие, которое может быть полезно при обсуждении устройств реального мира. Сущность реального мира – объекты – не обязательно материальны – важно понятие существенно и различимо для других. Между объектами могут возникать связи трех видов:

- один к одному 1:1 (пациент: место в палате);
- один к многим 1:n и многие к одному n:1;
- многие ко многим n:n (пациент : хирург).

При построении моделей используются следующие геометрические фигуры:

Элемент ER-модели	Условно графическое представление
Объект	
Связь	
Атрибут	

В настоящее время существует большое множество прикладных программ для создания графического представления структуры БД. При этом могут быть использованы как специализированные средства (например, Visio), так и средства построения графических образов (Umbrello, OO Draw). Либо использовать облачный сервис <https://www.draw.io/>.

Выделим базовые сущности этой предметной области:

- Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.
- Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.
- Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.
- Контракты будем рассматривать вместе с книгами, т.к. каждая книга связана непосредственно с одним контрактом.

Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

ER–диаграмма сущностей издательской компании приведена на рисунке 1.1

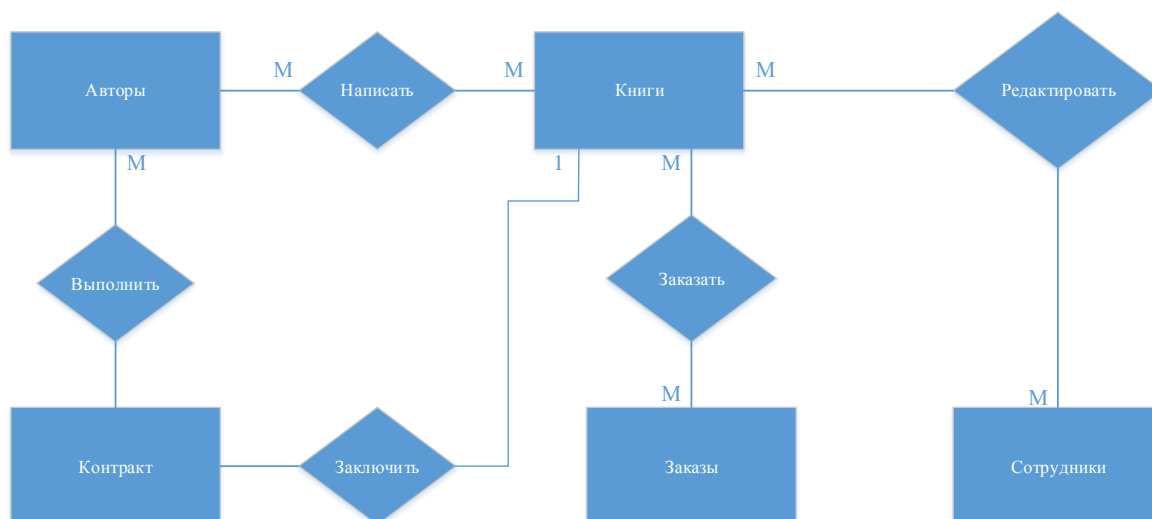


Рисунок 1.1 – ER–диаграмма издательской компании

1.4.2. Логическое проектирование реляционной БД

Преобразование ER–диаграммы в схему базы данных

Связь типа 1:n (один-ко-многим) между отношениями реализуется через внешний ключ. Ключ вводится для того отношения, к которому осуществляется множественная связь (КНИГИ).

Связь редактировать между отношениями КНИГИ и ЗАКАЗЫ принадлежит к типу n:m (многие-ко-многим). Этот тип связи реализуется через вспомогательное отношение, которое является соединением первичных ключей соответствующих отношений.

Для создания логической модели необходимо «уточнить» созданную ранее инфологическую модель. Уточнить – значит конкретизировать, добавить атрибуты в ранее созданные сущности. Уточняя, мы отмечаем в каждой сущности какие именно данные, относящиеся к этой сущности, мы будем хранить далее в базе данных.

Уточнённая схема реляционной БД издательской компании приведена на рисунке 1.2.

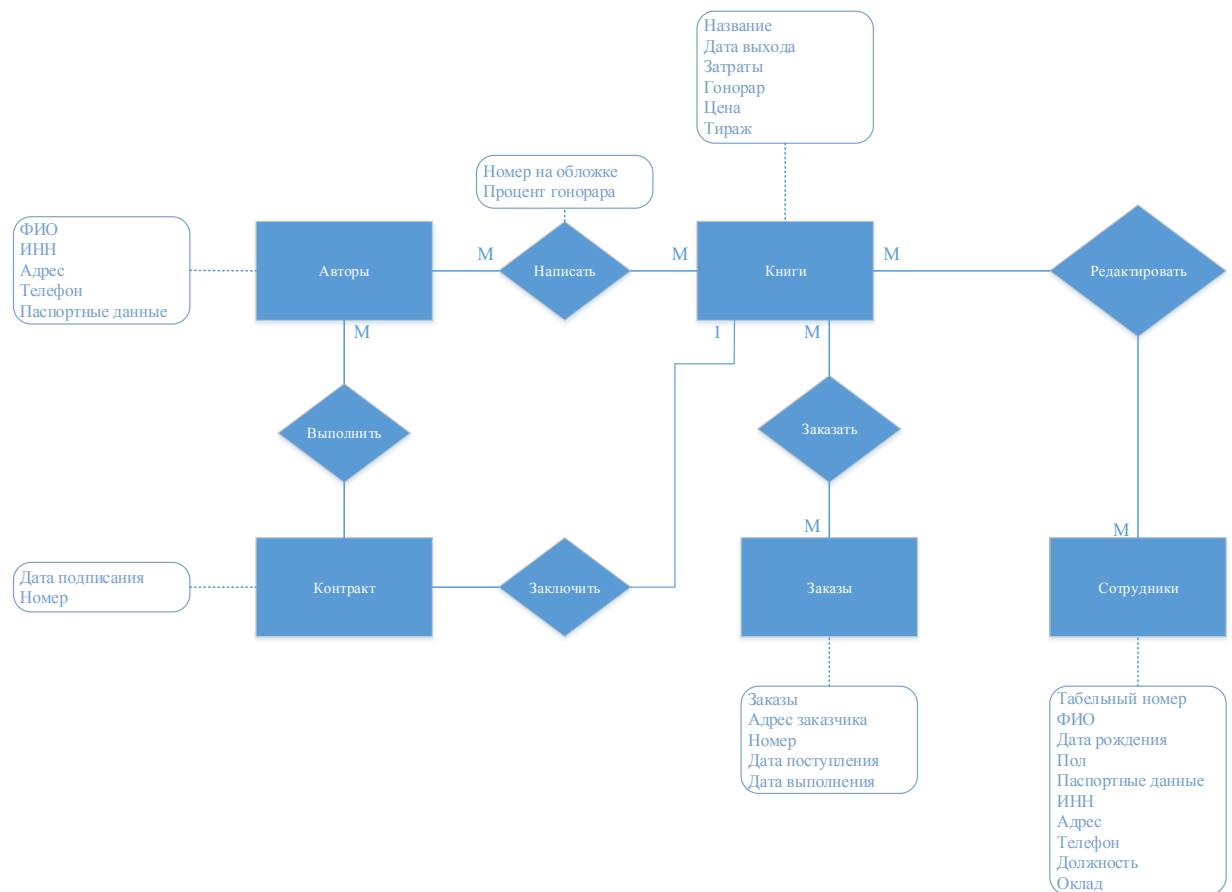


Рис.1.2 – Уточнённая ER–диаграмма издательской компании

Тогда логическая модель БД будет выглядеть как показано на рисунке 1.3.

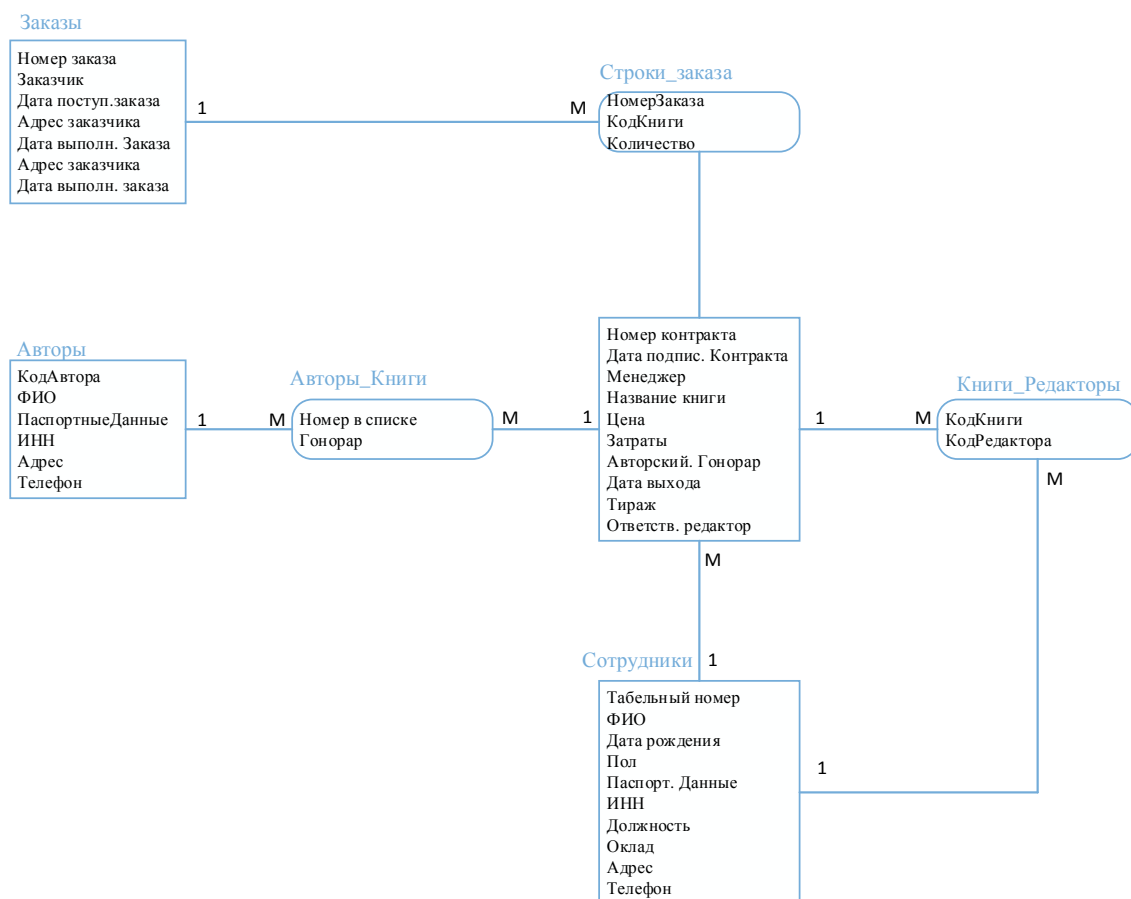


Рисунок 1.3 – Логическая модель БД

В этой модели были добавлены дополнительные таблицы:

– **Книги_Редакторы.** Эта сущность включена в диаграмму, во-первых, для устранения связи многие-ко-многим между редакторами и книгами, т.к. каждую книгу может редактировать множество редакторов и каждый редактор может редактировать множество книг. Во-вторых, т.к. и редакторы входят в число сотрудников и менеджеры, то каждый контракт (книгу) из числа сотрудников может подписывать только один менеджер и для этого установлена прямая связь один-ко-многим между сотрудниками и книгами.

– **Строки_заказа.** Эта сущность определена для устранения связи многие-ко-многим между заказами и книгами т.к. в одном заказе может быть множество книг. Также в этой сущности добавлен атрибут «Количество» для обозначения количества книг.

– **Авторы_Книги.** Эта сущность определена для устранения связи многие-ко-многим между авторами и книгами т.к. у одного автора может быть несколько книг.

Составление реляционных отношений

Понятие реляционный (англ. relation – отношение) связано с разработками известного английского специалиста в области систем баз данных Эдгара Кодда (Edgar Codd).

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы – один элемент данных;
- все ячейки в столбце таблицы однородные, то есть все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.);
- каждый столбец имеет уникальное имя;
- одинаковые строки в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным.

Основными понятиями реляционных баз данных являются тип данных, домен, атрибут, кортеж, первичный ключ и отношение.

Для начала покажем смысл этих понятий на примере отношения СОТРУДНИКИ, содержащего информацию о сотрудниках некоторой организации как показано на рисунке 1.4.

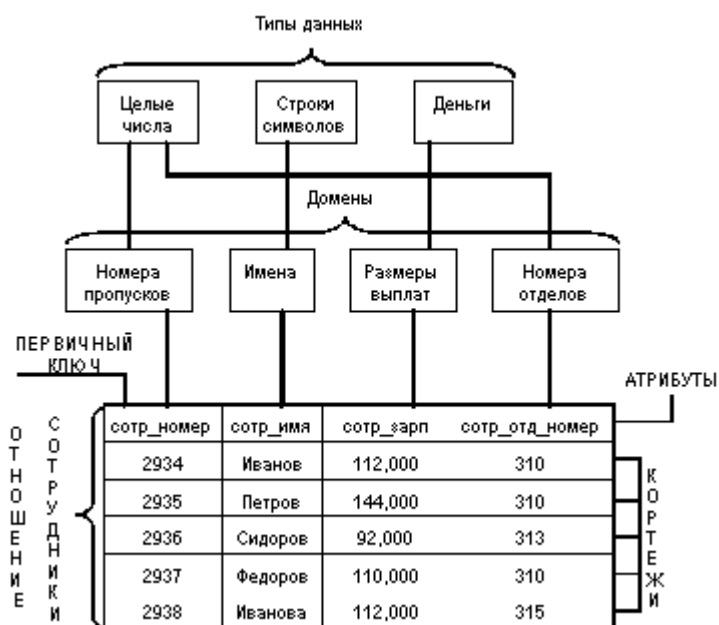


Рисунок 1.4 – Схема реляционного отношения

Тип данных в реляционной модели данных полностью адекватно понятию типа данных в языках программирования. Обычно в современных реляционных БД допускается хранение символьных, числовых данных, битовых строк, специализированных числовых

данных (таких как "деньги"), а также специальных "темпоральных" данных (дата, время, временной интервал).

Понятие **домена** более специфично для баз данных, хотя и имеет некоторые аналогии с подтипами в некоторых языках программирования. В самом общем виде домен определяется заданием некоторого базового типа данных, к которому относятся элементы домена, и произвольного логического выражения, применяемого к элементу типа данных. Если вычисление этого логического выражения дает результат "истина", то элемент данных является элементом домена.

Наиболее правильной интуитивной трактовкой понятия домена является понимание домена как допустимого потенциального множества значений данного типа. Например, домен "Имена" в схеме, показанной выше, определен на базовом типе строк символов, но в число его значений могут входить только те строки, которые могут изображать имя (в частности, такие строки не могут начинаться с мягкого знака).

Кортеж, соответствующий данной схеме отношения, - это множество пар {имя атрибута, значение}, которое содержит одно вхождение каждого имени атрибута, принадлежащего схеме отношения. "Значение" является допустимым значением домена данного атрибута (или типа данных, если понятие домена не поддерживается). Тем самым, степень или "арность" кортежа, т.е. число элементов в нем, совпадает с "арностью" соответствующей схемы отношения. Попросту говоря, кортеж - это набор именованных значений заданного типа.

Отношение - это множество кортежей, соответствующих одной схеме отношения. Иногда, чтобы не путаться, говорят "отношение-схема" и "отношение-экземпляр", иногда схему отношения называют заголовком отношения, а отношение как набор кортежей - телом отношения. На самом деле, понятие схемы отношения ближе всего к понятию структурного типа данных в языках программирования. Было бы вполне логично разрешать отдельно определять схему отношения, а затем одно или несколько отношений с данной схемой.

В свою очередь отношения имеют два важных свойства:

арность – число атрибутов в отношении;

мощность - это кардинальное число отношения, т.е. число кортежей (строк) в отношении.

Каждое реляционное отношение соответствует одной сущности (объекту предметной области) и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи.

Ключ или **потенциальный ключ** – это минимальный набор атрибутов, по значениям которых можно однозначно выбрать требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждая сущность должна, но не обязана обладать хотя бы одним возможным ключом. Другими словами **ключ – это поле или набор полей, однозначно идентифицирующий запись.**

Значение первичного ключа в таблице БД должно быть уникальным, то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа.

Первичные ключи облегчают установление связей между таблицами. Поскольку первичный ключ должен быть уникальным, для него могут использоваться не все поля таблицы.

В том случае, если базовое отношение не имеет потенциальных ключей, вводится **суррогатный первичный ключ**, который не несёт смысловой нагрузки и служит только для идентификации записей (например ID записи).

Примечание: суррогатный первичный ключ также может вводиться в тех случаях, когда потенциальный ключ имеет большой размер (например, длинная символьная строка) или является составным (не менее трёх атрибутов).

Потенциальными ключами отношения АВТОРЫ являются атрибуты Паспортные данные и ИНН. Первый хранится как длинная строка, а последний по условиям предметной области не является обязательным. Поэтому для авторов необходимо ввести суррогатный ключ – A_id. Книги можно идентифицировать по атрибуту Контракт: его номер обязателен и уникален. Потенциальные ключи отношения СОТРУДНИКИ – атрибуты ИНН, Паспортные данные, Табельный номер, причём все они обязательные. Табельный номер занимает меньше памяти, чем ИНН, поэтому он и будет первичным ключом. Кортежи отношения ЗАКАЗЫ можно идентифицировать ключом Номер заказа.

Потенциальными ключами вспомогательных отношений являются комбинации первичных ключей соответствующих базовых отношений.

Отношения приведены в таблице 1.1-1.7. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: **N** – числовой, **C** – символьный, **D** – дата (последний имеет стандартную длину, зависящую от СУБД, поэтому она не указывается).

Таблица 1.1 – Схема отношения СОТРУДНИКИ (Employees)

Содержание поля	Имя поля	Тип, длина	Примечания
Табельный номер	E_ID	N(4)	первичный ключ
Фамилия, имя, отчество	E_NAME	C(50)	обязательное поле
Дата рождения	E_BORN	D	
Пол	E_SEX	C(1)	обязательное поле
Паспортные данные	E_PASSP	C(50)	обязательное поле
ИНН	E_INN	N(12)	обязательное уникальное поле
Должность	E_POST	C(30)	обязательное поле
Оклад	E_SALARY	N(8,2)	обязательное поле
Адрес	E_ADDR	C(50)	
Телефоны	E_TEL	C(30)	многозначное поле

Таблица 1.2 – Схема отношения КНИГИ (Books)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер контракта	B_CONTRACT	N(6)	первичный ключ
Дата подписания контракта	B_DATE	D	обязательное поле
Менеджер	B_MAN	N(4)	внешний ключ (к Employees)
Название книги	B_TITLE	N(40)	обязательное поле
Цена	B_PRICE	N(6,2)	цена экземпляра книги
Затраты	B_ADVANCE	N(10,2)	общая сумма затрат на книгу
Авторский гонорар	B_FEE	N(8,2)	общая сумма гонорара
Дата выхода	B_PUBL	D	
Тираж	B_CIRCUL	N(5)	
Ответственный редактор	B_EDIT	N(4)	внешний ключ (к Employees)

Таблица 1.3 – Схема отношения АВТОРЫ (Authors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код автора	A_ID	N(4)	суррогатный первичный ключ
Фамилия, имя, отчество	A_NAME	C(50)	обязательное поле
Паспортные данные	A_PASSP	C(50)	обязательное поле
ИНН	A_INN	N(12)	уникальное поле
Адрес	A_ADDR	C(50)	обязательное поле
Телефоны	A_TEL	C(30)	многозначное поле

Таблица 1.4 – Схема отношения ЗАКАЗЫ (Orders)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер заказа	O_ID	N(6)	первичный ключ
Заказчик	O_COMPANY	C(40)	обязательное поле
Дата поступления заказа	O_DATE	D	обязательное поле
Адрес заказчика	O_ADDR	C(50)	обязательное поле
Дата выполнения заказа	O_READY	D	

Таблица 1.5 – Схема отношения КНИГИ–АВТОРЫ (Titles)

Содержание поля	Имя поля	Тип, длина	Примечания
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Код автора	A_ID	N(4)	внешний ключ (к Authors)
Номер в списке	A_NO	N(1)	обязательное поле
Гонорар	A_FEE	N(3)	процент от общего гонорара

Таблица 1.6 – Схема отношения КНИГИ–РЕДАКТОРЫ (Editors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Код редактора	E_ID	N(4)	внешний ключ (к Employees)

Таблица 1.7 – Схема отношения СТРОКИ ЗАКАЗА (Items)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер заказа	O_ID	N(6)	внешний ключ (к Orders)
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Количество	B_COUNT	N(4)	обязательное поле

1.5. Нормализация полученных отношений (до 3НФ)

Нормализация – это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

1.5.1. Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице. Для примера возьмем следующую таблицу 1.8.

Таблица 1.8. Машины - модели

Фирма	Модели
BMW	M5, X5M, M1
Nissan	GT-R

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным. Преобразуем таблицу к 1НФ как показано в таблице 1.9.

Таблица 1.9. – Преобразование отношения

Фирма	Модели
BMW	M5
BMW	X5M
BMW	M1
Nissan	GT-R

1.5.2. Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК).

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость. (простыми словами: все не ключевые атрибуты должны полностью зависеть от первичного ключа и не должно быть атрибутов, которые зависят от части ПК).

Приведем пример в таблице 1.10.

Таблица 1.10 –Пример работы с атрибутами для приведения отношения ко 2НФ

Первичный ключ

Модель	Фирма	Цена	Скидка
M5	BMW	5000000	5%
X5M	BMW	9000000	5%
M1	BMW	2500000	5%
GT-R	Nissan	5000000	10%

Таблица находится в первой нормальной форме, но не во второй. Цена машины зависит от модели и фирмы (здесь *составной первичный ключ*). Скидка зависит от фирмы, то есть зависимость от первичного ключа неполная. Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК. Нормализованные таблицы представлены под номером 1.11 и 1.12.

Таблица 1.11 – Нормализованное отношение «Модель-Фирма»

Модель	Фирма	Цена
M5	BMW	5000000
X5M	BMW	9000000
M1	BMW	2500000
GT-R	Nissan	5000000

Таблица 1.12 – Нормализованное отношение «Фирма»

Фирма	Скидка
BMW	5%
Nissan	10%

1.5.3. Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут *нетранзитивно* 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.

Например, возьмем таблицу 1.13.

Таблица 1.13 – Отношение во 2НФ

Модель	Магазин	Телефон
BMW	Риал-авто	87-33-98
Audi	Риал-авто	87-33-98
Nissan	Некст-авто	94-54-12

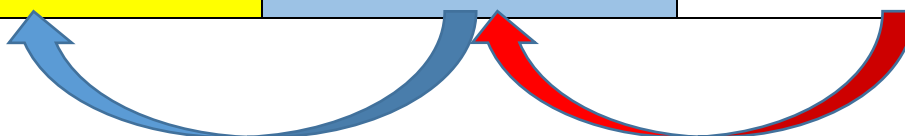


Таблица 1.13 находится во 2НФ, но не в 3НФ.

В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина. Таким образом, в отношении существуют следующие функциональные зависимости:

Модель → Магазин, Магазин → Телефон, Модель → Телефон. Зависимость «Модель → Телефон» является транзитивной, следовательно, отношение не находится в 3НФ. В результате разделения исходного отношения получаются два отношения, находящиеся в 3НФ. Полученные отношения представлены в таблицах 1.14 и 1.15

Таблица 1.14 – Отношение «Модель» в 3НФ

Модель	Магазин
BMW	Риал-авто
Audi	Риал-авто
Nissan	Некст-авто

Таблица 1.15 – Отношение «Магазин» в 3НФ

Магазин	Номер
Риал-авто	87-33-98
Некст-авто	94-54-12

1.6. Нормализация отношений для БД «Издательская компания».

1.1.1. 1 НФ (нормальная форма)

Для приведения таблиц к 1НФ требуется составить прямоугольные таблицы (один атрибут – один столбец) и разбить сложные атрибуты на простые, а многозначные атрибуты вынести в отдельные отношения.

Примечание. В реальных БД сложные атрибуты разбиваются на простые, если:

- этого требует внешнее представление данных;
- в запросах поиск может осуществляться по отдельной части атрибута.

Разделим атрибуты «Фамилия, имя, отчество» на три атрибута «Фамилия» «Имя», «Отчество» и «Паспортные данные» на атрибуты «Номер паспорта (уникальный)», «Дата выдачи» и «Кем выдан».

Многозначный атрибут «Телефоны» для сотрудников компании следует сначала разделить на два – «Домашние телефоны» и «Рабочие телефоны». (Для авторов мы не будем различать домашние и рабочие телефоны). Затем нужно создать отдельные отношения с (нерабочими) телефонами для сотрудников (ТЕЛЕФОНЫ СОТРУДНИКОВ) и для авторов (ТЕЛЕФОНЫ АВТОРОВ).

Атрибут «Рабочие телефоны» отношения СОТРУДНИКИ имеет неоднородные значения. Один из номеров телефонов – основной – определяется рабочим местом сотрудника (рассматриваются только стационарные телефоны). Наличие других номеров зависит от того, есть ли в том же помещении (комнате) другие сотрудники, имеющие стационарные телефоны. Можно добавить в отношении СОТРУДНИКИ атрибут Номер комнаты, а в атрибуте Рабочие телефоны хранить номер того телефона, который стоит на рабочем месте сотрудника. Дополнительные номера телефонов можно будет вычислить из других кортежей с таким же номером комнаты. Но в случае увольнения сотрудника мы потеряем сведения о номере рабочего телефона.

Поэтому создадим новое отношение КОМНАТЫ и включим в него атрибуты Номер комнаты и Телефон. Так как в комнате может не быть телефона, первичный ключ нового отношения не определен (ПК не может содержать null-значения), но на этих атрибутах можно определить составной уникальный ключ. Связь между отношениями СОТРУДНИКИ и КОМНАТЫ реализуем через составной внешний ключ (Номер комнаты, Телефон). Значение внешнего ключа для каждого сотрудника будем брать из того кортежа, в котором хранится основной рабочий телефон этого сотрудника.

1.6.1. 2 НФ (нормальная форма)

В нашем случае составные первичные ключи имеют отношения СТРОКИ ЗАКАЗА, КНИГИ-АВТОРЫ и КНИГИ-РЕДАКТОРЫ. Не ключевые атрибуты этих отношений функционально полно зависят от первичных ключей.

1.6.2. 3 НФ (нормальная форма)

В отношении ЗАКАЗЫ атрибут Адрес заказчика зависит от атрибута Заказчик, а не от первичного ключа, поэтому адрес следует вынести в отдельное отношение ЗАКАЗЧИКИ. Но при этом первичным ключом нового отношения станет атрибут Заказчик, т.е. длинная символьная строка. Целесообразнее перенести в новое отношение атрибуты Заказчик и Адрес заказчика и ввести для него суррогатный ПК. Так как каждый заказчик может сделать несколько заказов, связь между отношениями ЗАКАЗЧИКИ и ЗАКАЗЫ будет 1 : n и суррогатный ПК станет внешним ключом для отношения ЗАКАЗЫ.

В отношении СОТРУДНИКИ атрибут Оклад зависит от атрибута Должность. Поступим с этой транзитивной зависимостью так же, как в предыдущем случае: создадим

новое отношение ДОЛЖНОСТИ, перенесём в него атрибуты Должность и Оклад и введём суррогатный первичный ключ.

В отношениях СОТРУДНИКИ и АВТОРЫ атрибуты Дата выдачи и Кем выдан зависят от атрибута Номер паспорта, а не от первичного ключа. Но если мы выделим их в отдельное отношение, то получившиеся связи будут иметь тип 1:1. Следовательно, декомпозиция нецелесообразна.

В реальных базах данных после нормализации может проводиться денормализация. Она проводится с одной целью – повышение производительности БД. Рассмотрим некоторые запросы к нашей базе данных.

Например, запрос на получение списка телефонов авторов или домашних телефонов сотрудников потребует в нормализованной БД соединения отношений. Пользователю безразлична форма представления этого списка: номера телефонов через запятую или в столбец. Поэтому мы откажемся от создания отдельных отношений с номерами телефонов, и вернёмся к варианту с многозначными полями. (Это не касается рабочих телефонов сотрудников).

Другой запрос: как определяется, можно ли выполнить очередной заказ? Для каждой позиции заказа нужно просуммировать количество книг по выполненным заказам, получить остаток (тираж минус полученная сумма) и сравнить остаток с объёмом заказа. Такой расчёт может потребовать много времени, поэтому предлагается добавить в отношение КНИГИ производный атрибут Остаток тиража. Значение этого атрибута должно автоматически пересчитываться при установлении даты выполнения заказа.

После проведённых преобразований схема БД выглядит так (рисунок 1.3):

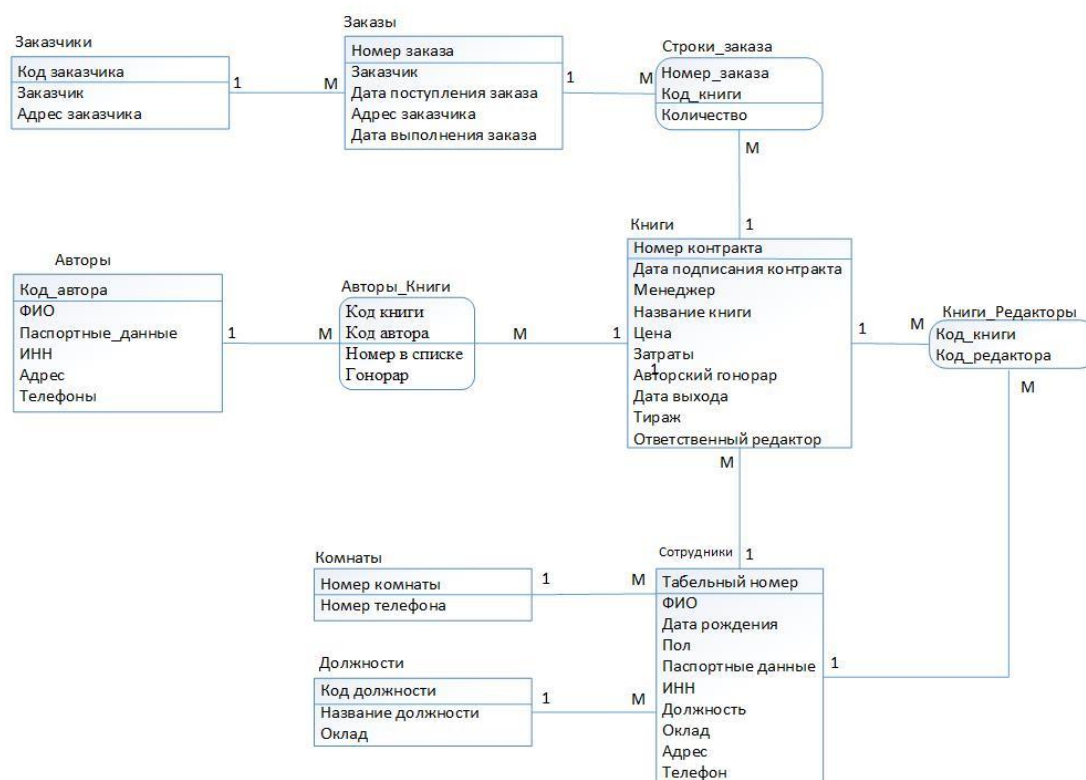


Рис. 1.3 – Окончательная схема РБД издательской компании

1.6.3. Окончательные схемы отношений

Окончательные схемы отношений базы данных с указанием ключей и других ограничений целостности приведены в табл. 1.8–1.17.

Таблица 1.8 – Схема отношения ДОЛЖНОСТИ (Posts)

Содержание поля	Имя поля	Тип, длина	Примечания
Код должности	P_ID	N(3)	суррогатный первичный ключ
Название должности	P_POST	C(30)	обязательное поле
Оклад	P_SAL	N(8,2)	обязательное поле

Таблица 1.9 – Схема отношения КОМНАТЫ (Rooms)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер комнаты	R_NO	N(3)	составной первичный ключ
Номер телефона	R_TEL	C(10)	

Таблица 1.10 – Схема отношения СОТРУДНИКИ (Employees)

Содержание поля	Имя поля	Тип, длина	Примечания
Табельный номер	E_ID	N(4)	первичный ключ
Фамилия	E_FNAME	C(20)	обязательное поле
Имя, отчество	E_LNAME	C(30)	обязательное поле
Дата рождения	E_BORN	D	
Пол	E_SEX	C(1)	обязательное поле
Код должности	E_POST	N(3)	внешний ключ (к Posts)
Номер комнаты	E_ROOM	N(3)	составной внешний ключ (к Rooms)
Номер телефона	E_TEL	C(10)	
ИНН	E_INN	C(12)	обязательное поле
Номер паспорта	E_PASSP	C(12)	обязательное поле
Кем выдан паспорт	E_ORG	C(30)	обязательное поле
Дата выдачи паспорта	E_PDATE	D	обязательное поле
Адрес	E_ADDR	C(50)	

Таблица 1.11 – Схема отношения ЗАКАЗЧИКИ (Customers)

Содержание поля	Имя поля	Тип, длина	Примечания
Код заказчика	C_ID	N(4)	суррогатный первичный ключ
Заказчик	C_NAME	C(30)	обязательное поле
Адрес заказчика	C_ADDR	C(50)	обязательное поле

Таблица 1.12 – Схема отношения АВТОРЫ (Authors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код автора	A_ID	N(4)	суррогатный ключ
Фамилия	A_FNAME	C(20)	обязательное поле
Имя, отчество	A_LNAME	C(30)	обязательное поле

ИНН	A_INN	C(12)	
Номер паспорта	A_PASSP	C(12)	обязательное поле
Кем выдан паспорт	A_ORG	C(30)	обязательное поле
Дата выдачи паспорта	A_PDATE	D	обязательное поле
Адрес	A_ADDR	C(50)	обязательное поле
Телефоны	A_TEL	C(30)	многозначное поле

Таблица 1.13 – Схема отношения КНИГИ (Books)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер контракта	B_CONTRACT	N(6)	первичный ключ
Дата подписания контракта	B_DATE	D	обязательное поле
Менеджер	B_MAN	N(4)	внешний ключ (к Employees)
Название книги	B_TITLE	N(40)	обязательное поле
Цена	B_PRICE	N(6,2)	цена экземпляра книги
Затраты	B_ADVANCE	N(10,2)	общая сумма затрат на книгу
Авторский гонорар	B_FEE	N(8,2)	общая сумма гонорара
Дата выхода	B_PUBL	D	
Тираж	B_CIRCUL	N(5)	
Ответственный редактор	B_EDIT	N(4)	внешний ключ (к Employees)
Остаток тиража	B_REST	N(5)	производное поле

Таблица 1.14 – Схема отношения ЗАКАЗЫ (Orders)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер заказа	O_ID	N(6)	первичный ключ
Код заказчика	O_COMPANY	N(4)	внешний ключ (к Customers)
Дата поступления заказа	O_DATE	D	обязательное поле
Дата выполнения заказа	O_READY	D	

Таблица 1.15 – Схема отношения КНИГИ–АВТОРЫ (Titles)

Содержание поля	Имя поля	Тип, длина	Примечания
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Код автора	A_ID	N(4)	внешний ключ (к Authors)
Номер в списке	A_NO	N(1)	обязательное поле
Гонорар	A_FEE	N(3)	процент от общего гонорара

Таблица 1.16 – Схема отношения СТРОКИ ЗАКАЗА (Items)

Содержание поля	Имя поля	Тип, длина	Примечания
Номер заказа	O_ID	N(6)	внешний ключ (к Orders)
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Количество	B_COUNT	N(4)	обязательное поле

Таблица 1.17 – Схема отношения КНИГИ–РЕДАКТОРЫ (Editors)

Содержание поля	Имя поля	Тип, длина	Примечания
Код книги (№ контракта)	B_ID	N(6)	внешний ключ (к Books)
Код редактора	E_ID	N(4)	внешний ключ (к Employees)

1.7. Пример нормализации отношение №3

1.7.1. 1 НФ (нормальная форма)

В ходе логического моделирования на первом шаге предложено хранить данные в одном отношении, имеющем следующие атрибуты:

СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ (*H_COTR*, ФАМ, H_ОТД, ТЕЛ, *H_ПРО*, ПРОЕКТ, H_ЗАДАН)

Где:

H_COTR - табельный номер сотрудника

ФАМ - фамилия сотрудника

H_ОТД - номер отдела, в котором числится сотрудник

ТЕЛ - телефон сотрудника

H_ПРО - номер проекта, над которым работает сотрудник

ПРОЕКТ - наименование проекта, над которым работает сотрудник

H_ЗАДАН - номер задания, над которым работает сотрудник

Т.к. каждый сотрудник в каждом проекте выполняет ровно одно задание, то в качестве потенциального ключа отношения необходимо взять пару атрибутов {*H_COTR*, *H_ПРО*}.

В текущий момент состояние предметной области отражается следующими фактами:

- **Сотрудник Иванов**, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 1 и во втором проекте "Климат" задание 1.
- **Сотрудник Петров**, работающий в 1 отделе, выполняет в первом проекте "Космос" задание 2.
- **Сотрудник Сидоров**, работающий во 2 отделе, выполняет в первом проекте "Космос" задание 3 и во втором проекте "Климат" задание 2.

Это состояние отражается в таблице (курсивом выделены ключевые атрибуты):

Таблица 1.18 – Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ

<i>H_COTR</i>	ФАМ	H_ОТД	ТЕЛ	<i>H_ПРО</i>	ПРОЕКТ	H_ЗАДАН
1	Иванов	1	11-22-33	1	Космос	1
1	Иванов	1	11-22-33	2	Климат	1
2	Петров	1	11-22-33	1	Космос	2
3	Сидоров	2	33-22-11	1	Космос	3
3	Сидоров	2	33-22-11	2	Климат	2

1.7.2. Определение функциональной зависимости

Для правильного проектирования модели данных применяется метод нормализации отношений. Нормализация основана на понятии функциональной зависимости атрибутов отношения.

Пусть R - отношение. Множество атрибутов Y **функционально зависит** от множества атрибутов X (X **функционально определяет** Y) тогда и только тогда, когда для любого состояния отношения R для любых кортежей $r_1, r_2 \in R$ из того, что $r_1.X = r_2.X$ следует что $r_1.Y = r_2.Y$ (т.е. во всех кортежах, имеющих одинаковые значения атрибутов X , значения атрибутов Y также совпадают в любом состоянии отношения R). Символически функциональная зависимость записывается

$$X \rightarrow Y.$$

Множество атрибутов X называется **детерминантом функциональной зависимости**, а множество атрибутов Y называется **зависимой частью**.

Замечание. Если атрибуты X составляют потенциальный ключ отношения R , то любой атрибут отношения R функционально зависит от X .

В отношении **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ** можно привести следующие примеры функциональных зависимостей:

Зависимость атрибутов от ключа отношения:

$$\{H_СОТР, H_ПРО\} \rightarrow ФАМ$$

$$\{H_СОТР, H_ПРО\} \rightarrow H_ОТД$$

$$\{H_СОТР, H_ПРО\} \rightarrow ТЕЛ$$

$$\{H_СОТР, H_ПРО\} \rightarrow ПРОЕКТ$$

$$\{H_СОТР, H_ПРО\} \rightarrow H_ЗАДАН$$

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

$$H_СОТР \rightarrow ФАМ$$

$$H_СОТР \rightarrow H_ОТД$$

$$H_СОТР \rightarrow ТЕЛ$$

Зависимость наименования проекта от номера проекта:

$$H_ПРО \rightarrow ПРОЕКТ$$

Зависимость номера телефона от номера отдела:

$$H_ОТД \rightarrow ТЕЛ$$

1.7.3. 2 НФ (нормальная форма)

Отношение **СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ** не находится в 2НФ, т.к. есть атрибуты, зависящие от части сложного ключа:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника является зависимостью от части сложного ключа:

$H_СОТР \rightarrow \text{ФАМ}$

$H_СОТР \rightarrow \text{Н_ОТД}$

$H_СОТР \rightarrow \text{ТЕЛ}$

Зависимость наименования проекта от номера проекта является зависимостью от части сложного ключа:

$H_ПРО \rightarrow \text{ПРОЕКТ}$

Для того, чтобы устранить зависимость атрибутов от части сложного ключа, нужно произвести *декомпозицию* отношения на несколько отношений. При этом *те атрибуты, которые зависят от части сложного ключа*, выносятся в отдельное отношение.

Отношение СОТРУДНИКИ_ОТДЕЛЫ_ПРОЕКТЫ декомпозируем на три отношения - СОТРУДНИКИ_ОТДЕЛЫ, ПРОЕКТЫ, ЗАДАНИЯ.

Отношение СОТРУДНИКИ_ОТДЕЛЫ ($H_СОТР$, ФАМ, Н_ОТД, ТЕЛ):

Функциональные зависимости:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

$H_СОТР \rightarrow \text{ФАМ}$

$H_СОТР \rightarrow \text{Н_ОТД}$

$H_СОТР \rightarrow \text{ТЕЛ}$

Зависимость номера телефона от номера отдела:

$\text{Н_ОТД} \rightarrow \text{ТЕЛ}$

Таблица 1.19 – Отношение СОТРУДНИКИ_ОТДЕЛЫ

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Отношение ПРОЕКТЫ ($H_ПРО$, ПРОЕКТ):

Функциональные зависимости:

$H_ПРО \rightarrow$ ПРОЕКТ

Таблица 1.20 – Отношение ПРОЕКТЫ

Н_ПРО	ПРОЕКТ
1	Космос
2	Климат

Отношение ЗАДАНИЯ ($H_СОТР$, $H_ПРО$, $H_ЗАДАН$):

Функциональные зависимости:

$\{H_СОТР, H_ПРО\} \rightarrow H_ЗАДАН$

Таблица 1.21 – Отношение ЗАДАНИЯ

Н_СОТР	Н_ПРО	Н_ЗАДАН
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

1.7.4. ЗНФ (нормальная форма)

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** не находится в ЗНФ, т.к. имеется функциональная зависимость неключевых атрибутов (зависимость номера телефона от номера отдела):

$H_ОТД \rightarrow$ ТЕЛ

Для того, чтобы устранить зависимость неключевых атрибутов, нужно произвести декомпозицию отношения на несколько отношений. При этом *те неключевые атрибуты, которые являются зависимыми*, выносятся в отдельное отношение.

Отношение **СОТРУДНИКИ_ОТДЕЛЫ** декомпозируем на два отношения - **СОТРУДНИКИ, ОТДЕЛЫ**.

Отношение СОТРУДНИКИ ($H_СОТР$, ФАМ, $H_ОТД$):

Функциональные зависимости:

Зависимость атрибутов, характеризующих сотрудника от табельного номера сотрудника:

$H_СОТР \rightarrow \text{ФАМ}$

$H_СОТР \rightarrow H_ОТД$

$H_СОТР \rightarrow \text{ТЕЛ}$

Таблица 1.22 – Отношение СОТРУДНИКИ

H_СОТР	ФАМ	H_ОТД
1	Иванов	1
2	Петров	1
3	Сидоров	2

Отношение **ОТДЕЛЫ** ($H_ОТД$, ТЕЛ):

Функциональные зависимости:

Зависимость номера телефона от номера отдела:

$H_ОТД \rightarrow \text{ТЕЛ}$

Таблица 1.23 – Отношение ОТДЕЛЫ

H_ОТД	ТЕЛ
1	11-22-33
2	33-22-11

Обратим внимание на то, что атрибут **H_ОТД**, *не являвшийся ключевым* в отношении **СОТРУДНИКИ_ОТДЕЛЫ**, *становится потенциальным ключом* в отношении **ОТДЕЛЫ**. Именно за счет этого устраняется избыточность, связанная с многократным хранением одних и тех же номеров телефонов.

1.7.5. Вывод

Реляционная модель, состоящая из четырех отношений **СОТРУДНИКИ**, **ОТДЕЛЫ**, **ПРОЕКТЫ**, **ЗАДАНИЯ**, находящихся в третьей нормальной форме, является адекватной описанной модели предметной области, и требует наличия только тех триггеров, которые поддерживают ссылочную целостность. Такие триггеры являются стандартными и не требуют больших усилий в разработке.

1.7.6. Определение дополнительных ограничений целостности

Перечислим ограничения целостности, которые не указаны в таблицах 8–17.

- Значения всех числовых атрибутов – больше 0 (или null, если атрибут необязателен).
- Область значений атрибута Sex отношения EMPLOYEES – символы «м» и «ж».
- Отношение ROOMS не имеет первичного ключа, но комбинация значений (R_no, Tel) уникальна.
- В отношении TITLES порядковые номера авторов на обложке одной книги должны идти подряд, начиная с 1.
- В отношении TITLES сумма процентов гонорара по одной книге равна 100.

Ограничения (4,5) нельзя реализовать в схеме отношения. В реальных БД подобные ограничения целостности реализуются программно (через внешнее приложение или специальную процедуру контроля данных).

2. ЗАКЛЮЧЕНИЕ

В данной лабораторной работе была дана краткая теоретическая информация проектирования баз данных. Были представлены этапы моделирования и описание реляционных отношений. В результате выполнения данной лабораторной работы студент получит практические навыки по проектированию БД и нормализации реляционных отношений.