ЦЕЛИ И ЗАДАЧИ РАБОТЫ

Цель курсового проектирования – применение на практике знаний, полученных в процессе изучения курса "Базы данных", и получение практических навыков создания автоматизированных информационных систем (АИС), основанных на базах данных.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Общие положения

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

1. Корректность схемы БД, т.е. база данных должна быть как можно более полным и адекватным образом моделируемой предметной области (ПрО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных.
2. Обеспечение ограничений (на объёмы внешней и оперативной памяти и другие ресурсы вычислительной системы).
3. Эффективность функционирования (соблюдение ограничений на время реакции системы на запрос и обновление данных).
4. Защита данных (от аппаратных и программных сбоев и несанкционированного доступа).
5. Простота и удобство эксплуатации.
6. Гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Удовлетворение требований 1–4 обязательно для принятия проекта.

1.2. Этапы проектирования базы данных

Процесс проектирования включает в себя следующие этапы:

1. Инфологическое проектирование.
2. Обоснование выбора системы управления базой данных (СУБД)
3. Даталогическое проектирование БД.
4. Физическое проектирование БД.

Инфологический подход не предоставляет формальных способов моделирования реальности, но он закладывает основы методологии проектирования баз данных.

1.2.1. Инфологическое проектирование

Основными задачами инфологического проектирования являются определение предметной области системы и формирование взгляда на ПрО с позиций сообщества будущих пользователей БД, т.е. инфологической модели ПрО.

Инфологическая модель ПрО представляет собой описание структуры и динамики ПрО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПрО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое. На первом этапе инфологического проектирования происходит выбор подхода к проектированию БД и анализ предметной области в соответствии с индивидуальным заданием.

Рассмотрим основные подходы к созданию инфологической модели предметной области.

Функциональный подход к проектированию БД

Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаётся рассматриваемая БД.

Предметный подход к проектированию БД

Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чёткое представление о самой ПрО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПрО и наиболее адекватному её отображению в БД с учётом самого широкого спектра информационных запросов к ней.

Проектирование с использованием метода "сущность-связь"

Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПрО. Проектировщик разбивает её на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПрО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Сущность – это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, СОТРУДНИК или АВТОМОБИЛЬ), так и абстрактные (например, ЭКЗАМЕН или ДИАГНОЗ).

Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств.

Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчинёнными (дочерними), а сильные – базовыми (основными, родительскими).

Для каждой сущности выбираются свойства (**атрибуты**). Различают:

1. *Идентифицирующие и описательные атрибуты*. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются *потенциальными ключами.* Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.
2. *Составные и простые атрибуты*. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
3. *Однозначные и многозначные атрибуты* (могут иметь соответственно одно или много значений для каждого экземпляра сущности).
4. *Основные и производные атрибуты*. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут.

Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику). Различают связи типа "сущность-сущность", "сущность-атрибут" и "атрибут-атрибут" для отношений между атрибутами, которые характеризуют одну и ту же сущность или одну и ту же связь типа "сущность-сущность".

Каждая связь характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если вновь порождённый объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной.

По типу различают множественные связи "один к одному" (1:1), "один ко многим" (1:n) и "многие ко многим" (m:n). ER–диаграмма, содержащая различные типы связей, приведена на рис. 1. Обратите внимание, что обязательные связи на рис. 1 выделены двойной линией.

Рис.1. ER–диаграмма с примерами типов множественных связей

Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи – связь между отделом и сотрудниками, которые в нём работают. Примером тернарной связи является связь типа *экзамен* между сущностями *ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ*. Из последнего примера видно, что связь также может иметь атрибуты (в данном случае это *Дата проведения* и *Оценка*). Пример ER–диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 2.

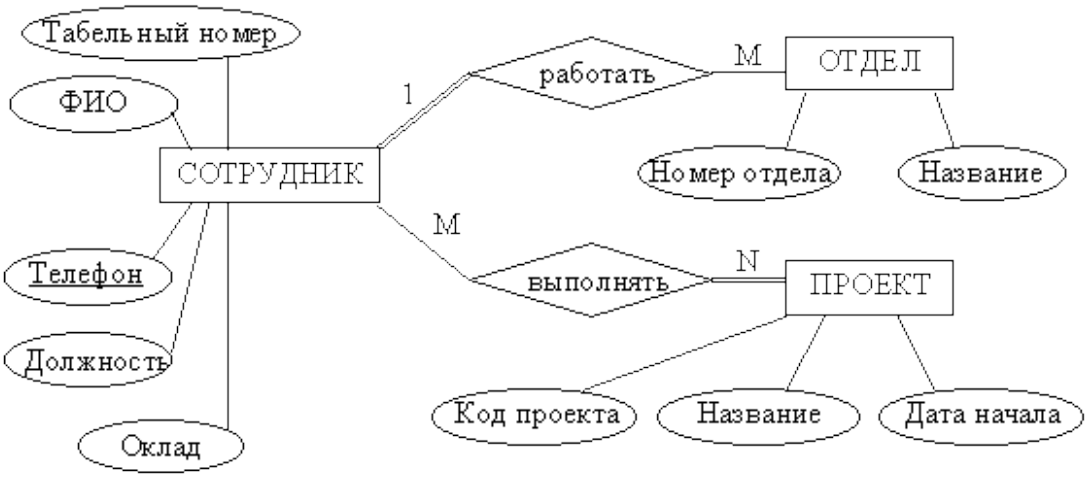

Рис.2. Пример ER–диаграммы с однозначными и многозначными атрибутами

После того, как созданы локальные представления, выполняется их объединение. При небольшом количестве локальных областей (не более пяти) они объединяются за один шаг. В противном случае обычно выполняют бинарное объединение в несколько этапов.

При объединении проектировщик может формировать конструкции, производные по отношению к тем, которые были использованы в локальных представлениях. Такой подход может преследовать следующие цели:

- объединение в единое целое фрагментарных представлений о различных свойствах одного и того же объекта;
- введение абстрактных понятий, удобных для решения задач системы, установление их связи с конкретными понятиями, использованными в модели;
- образование классов и подклассов подобных объектов (например, класс "изделие" и подклассы типов изделий, производимых на предприятии).

На этапе объединения необходимо выявить и устранить все противоречия. Например, одинаковые названия семантически различных объектов или связей или несогласованные ограничения целостности на одни и те же атрибуты в разных приложениях. Устранение противоречий вызывает необходимость возврата к этапу моделирования локальных представлений с целью внесения в них соответствующих изменений.

По завершении объединения результаты проектирования являют собой концептуальную инфологическую модель предметной области. Модели локальных представлений – это внешние инфологические модели.

1.2.2. Обоснование выбора СУБД и других программных средств

Выбор СУБД является одним из важнейших моментов в разработке проекта БД, так как он принципиальным образом влияет на весь процесс

проектирования БД и реализацию информационной системы. Теоретически при выборе СУБД нужно принимать во внимание десятки факторов. Но практически разработчики руководствуются лишь собственной интуицией и несколькими наиболее важными критериями, к которым, в частности, относятся:

- тип модели данных, которую поддерживает данная СУБД, её адекватность потребностям рассматриваемой предметной области;
- характеристики производительности системы;
- запас функциональных возможностей для дальнейшего развития ИС;
- степень оснащённости системы инструментарием для персонала администрирования данными;
- удобство и надежность СУБД в эксплуатации;
- стоимость СУБД и дополнительного программного обеспечения.

Таким образом, в разделе нужно не выбрать СУБД из множества возможных, а, опираясь на критерии, приведенные выше, обосновать свой выбор СУБД для реализации проектируемой БД.

1.2.3. Определение требований к операционной обстановке

На этом этапе производится оценка требований к вычислительным ресурсам, необходимым для функционирования системы, определение типа и конфигурации конкретной ЭВМ, выбор типа и версии операционной системы. Объём вычислительных ресурсов зависит от предполагаемого объёма проектируемой базы данных и от интенсивности их использования. Если БД будет работать в многопользовательском режиме, то требуется подключение её к сети и наличие соответствующей многозадачной операционной системы.

1.2.4. Даталогическое проектирование БД

На этом этапе разрабатывается логическая структура БД, соответствующая логической модели ПО. Решение этой задачи существенно зависит от модели данных, поддерживаемой выбранной СУБД.

Результатом выполнения этого этапа являются схемы БД концептуального и внешнего уровней архитектуры, составленные на языках определения данных (DDL, Data Definition Language), поддерживаемых данной СУБД.

1.2.5. Физическое проектирование БД

Этап физического проектирования заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, т.е. отображении логической структуры БД в структуру хранения. Решается вопрос размещения хранимых данных в пространстве памяти, выбора эффективных методов доступа к различным компонентам "физической" БД. Результаты этого этапа документируются в форме схемы хранения на языке

определения данных (DDL). Принятые на этом этапе решения оказывают определяющее влияние на производительность системы.

Одной из важнейших составляющих проекта базы данных является разработка средств защиты БД. Защита данных имеет два аспекта: защита от сбоев и защита от несанкционированного доступа. Для защиты от сбоев разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа.

1.3. Особенности проектирования реляционной базы данных

Проектирование реляционной базы данных проходит в том же порядке, что и проектирование БД других моделей данных, но имеет свои особенности.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления. При неправильно спроектированной схеме БД могут возникнуть аномалии модификации данных. Они обусловлены отсутствием средств явного представления типов множественных связей между объектами ПО и неразвитостью средств описания ограничений целостности на уровне модели данных.

Для решения подобных проблем проводится нормализация отношений.

1.3.1. Нормализация отношений

В рамках реляционной модели данных Э.Ф. Коддом (E.F. Codd) был разработан аппарат нормализации отношений и предложен механизм, позволяющий любое отношение преобразовать к третьей нормальной форме.

Нормализация схемы отношения выполняется путём декомпозиции схемы. **Декомпозицией** схемы отношения R называется замена её совокупностью схем отношений A_i таких, что

$$R = \bigcup_i A_i ,$$

и не требуется, чтобы отношения A_i были непересекающимися.

Введём понятие простого и сложного атрибута. Простой атрибут – это атрибут, значения которого атомарны (т.е. неделимы). Сложный атрибут может иметь значение, представляющее собой конкатенацию нескольких значений одного или разных доменов. Аналогом сложного атрибута может быть агрегат или повторяющийся агрегат данных.

Первая нормальная форма (1НФ).

Отношение приведено к 1НФ, если все его атрибуты простые.

Для того чтобы привести к 1НФ отношение, содержащее повторяющиеся атрибуты (агрегаты), нужно построить декартово произведение всех повторяющихся агрегатов с кортежами, к которым они относятся. Для идентификации кортежа в этом случае понадобится составной ключ, включающий первичный ключ исходного отношения и идентифицирующие атрибуты агрегатов.

Введём понятие функциональной зависимости. Пусть X и Y – атрибуты некоторого отношения. Если в любой момент времени каждому значению X соответствует единственное значение Y, что Y функционально зависит от X (X→Y). Атрибут (группа атрибутов) X называется **детерминантом**.

В нормализованном отношении все неключевые атрибуты функционально зависят от ключа отношения. Говорят, что неключевой атрибут функционально полно зависит от составного ключа, если он функционально зависит от ключа, но не находится в функциональной зависимости ни от какой части составного ключа.

Вторая нормальная форма (2НФ).

> Отношение находится во 2НФ, если оно приведено к 1НФ и каждый неключевой атрибут функционально полно зависит от составного ключа.

Для того чтобы привести отношение ко 2НФ, нужно:

- построить его проекцию, исключив атрибуты, которые не находятся в функционально полной зависимости от составного ключа;
- построить дополнительно одну или несколько проекций на часть составного ключа и атрибуты, функционально зависящие от этой части ключа.

Рассмотрим понятие транзитивной зависимости. Пусть X, Y, Z – атрибуты некоторого отношения. При этом X→ Y и Y→ Z, но обратное соответствие отсутствует, т.е. Z не зависит от Y или Y не зависит от X. Тогда говорят, что Z транзитивно зависит от X (X→→Z).

Третья нормальная форма (3НФ).

> Отношение находится в 3НФ, если оно находится во 2НФ и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Для того чтобы привести отношение к 3НФ, нужно:

- построить его проекцию, исключив транзитивно зависящие от ключа атрибуты;

- построить дополнительно одну или несколько проекций на детерминанты исходного отношения и атрибуты, функционально зависящие от них.

Введём понятие **многозначной зависимости**. Многозначная зависимость существует, если заданным значениям атрибута X соответствует множество, состоящее из нуля (или более) значений атрибута Y (X–>>Y).

Различают тривиальные и нетривиальные многозначные зависимости. **Тривиальной** называется такая многозначная зависимость X–>>Y, для которой Y ⊂ ∈X или X ∪ Y = R, где R – рассматриваемое отношение. Тривиальная многозначная зависимость не нарушает 4НФ. Если хотя бы одно из двух этих условий не выполняется, то такая зависимость называется **нетривиальной**.

Четвертая нормальная форма (4НФ).

> Отношение находится в 4НФ, если оно находится в 3НФ и в нём отсутствуют нетривиальные многозначные зависимости.

Для того чтобы привести отношение к 4НФ, нужно построить две или более проекции исходного отношения, каждая из которых содержит ключ и одну из многозначных зависимостей.

Нормализация отношений позволяет сократить дублирование данных, но появление новых отношений порождает проблему поддержки семантической целостности данных.

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Методические указания к курсовому проектированию по курсу "Базы данных". Часть II

СОДЕРЖАНИЕ

3.2. Составление реляционных отношений

3.3. Нормализация полученных отношений (до 3НФ)

3.3. Определение дополнительных ограничений целостности

3.4. Описание групп пользователей и прав доступа

4. Физическое проектирование БД

5. Реализация пользовательских запросов

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

1. Инфологическое проектирование

1.1. 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.

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 как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники.

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 издательской компании приведена на рис. 3 (базовые сущности на рисунках выделены полужирным шрифтом).

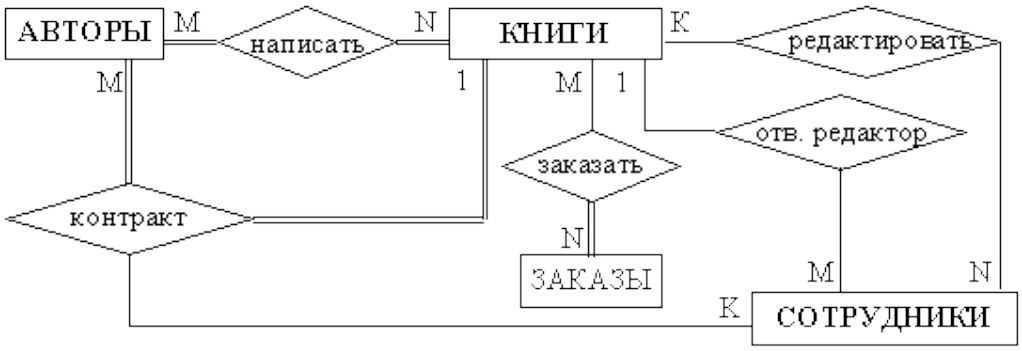

Рис.3. ER–диаграмма издательской компании

1.2. Анализ информационных задач и круга пользователей системы

Система создаётся для обслуживания следующих групп пользователей:

- администрация (дирекция);
- менеджеры;
- редакторы;
- сотрудники компании, обслуживающие заказы.

Определим границы информационной поддержки пользователей:

1) Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление в архив);
- обеспечение логической непротиворечивости БД;

- обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

2) Готовые запросы:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
- получение списка редакторов, работающих над книгами;
- получение полной информации о книге (проекте);
- получение сведений о конкретном авторе (с перечнем всех книг);
- получение информации о продажах (по одному или по всем проектам);
- определение общей прибыли от продаж по текущим проектам;
- определение размера гонорара автора по конкретному проекту.

2. Выбор СУБД и других программных средств

Анализ информационных задач показывает, что для реализации требуемых функций подходят почти все СУБД для ПЭВМ (FoxPro, Clipper, MS Access и др.). Все они поддерживают реляционную модель данных и предоставляют разнообразные возможности для работы с данными.

Объём внешней и оперативной памяти, требующийся для функционирования СУБД, обычно указывается в сопроводительной документации.

Для того чтобы в учебном примере не привязываться к конкретной СУБД, выполним описание логической схемы БД на SQL-92.

2.1. Определение требований к операционной обстановке

Для выполнения этого этапа необходимо знать (хотя бы ориентировочно) объём работы издательства (т.е. количество книг, авторов и заказчиков), а также иметь представление о характере и интенсивности запросов.

Объём внешней памяти, необходимый для функционирования системы, складывается из двух составляющих: память, занимаемая модулями СУБД (ядро, утилиты, вспомогательные программы), и память, отводимая под данные ($M_Д$). Наиболее существенным обычно является $M_Д$. Объём памяти $M_Д$, требуемый для хранения данных, можно приблизительно оценить по формуле

$$Mc = 2\sum_{i=1}^{n} l_i * (N_i + N_{ai}),$$

где l_i – длина записи в i-й таблице (в байтах), N_i – примерное (максимально возможное) количество записей в i-й таблице, N_a – количество записей в архиве

i-й таблицы. Коэффициент 2 перед суммой нужен для того, чтобы выделить память для хранения индексов, промежуточных данных, для выполнения объёмных операций (например, сортировки) и т.п.

Посчитаем приблизительно, какой объём внешней памяти потребуется для хранения данных. Примем ориентировочно, что:

- одновременно осуществляется около пятидесяти проектов, работа над проектом продолжается в среднем два месяца (по 0,3К);
- в компании работает 100 сотрудников (по 0,2К на каждого сотрудника);
- издательство сотрудничает с тридцатью авторами (по 0,2К);
- в день обслуживается порядка двадцати заявок (по 0,1К);
- устаревшие данные переводятся в архив.

Тогда объём памяти для хранения данных за первый год примерно составит:

$M_c = 2(100*0,2+6(50*0,3)+30*0,2+250(20*0,1)) = 1232\,К \approx 1,2\,М,$

где 250 – количество рабочих дней в году, а 12 мес./2 мес. = 6. Объём памяти будет увеличиваться ежегодно на столько же при сохранении объёма работы.

Объём памяти, занимаемый программными модулями пользователя, обычно невелик по сравнению с объёмом самих данных, поэтому может не учитываться. Требуемый объём оперативной памяти определяется на основании анализа интенсивности запросов и объёма результирующих данных.

3 Даталогическое проектирование реляционной БД

3.1. Преобразование ER–диаграммы в схему базы данных

База данных создаётся на основании схемы базы данных. Для преобразования ER–диаграммы в схему БД приведём уточнённую ER–диаграмму, содержащая атрибуты сущностей (рис. 4).

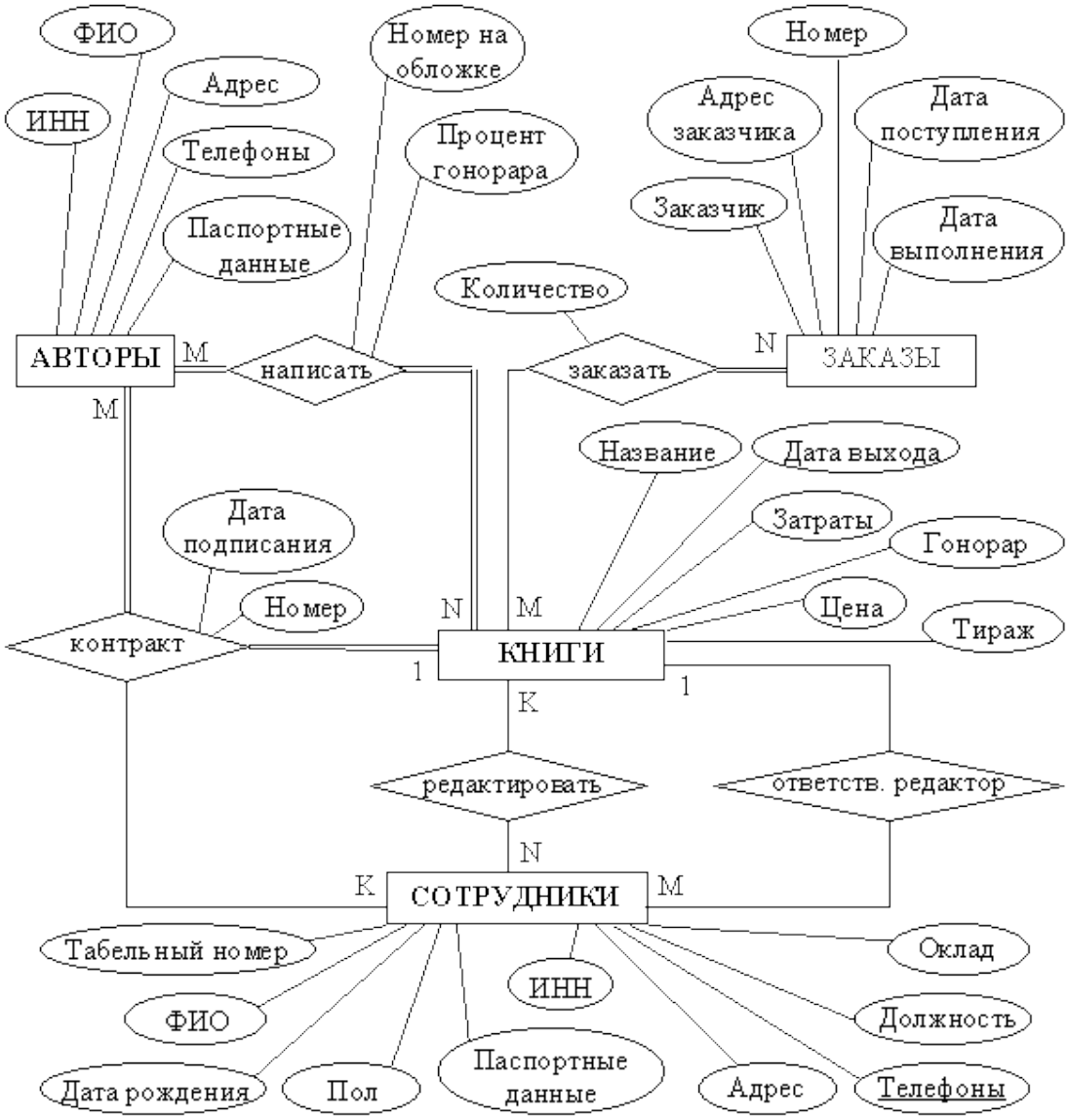

Рис.4. Уточнённая ER–диаграмма издательской компании

Примечание: многозначные атрибуты на рисунке выделены подчеркиванием.

Преобразование ER–диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таблицы БД). Будем использовать обозначения, представленные на рис. 5.

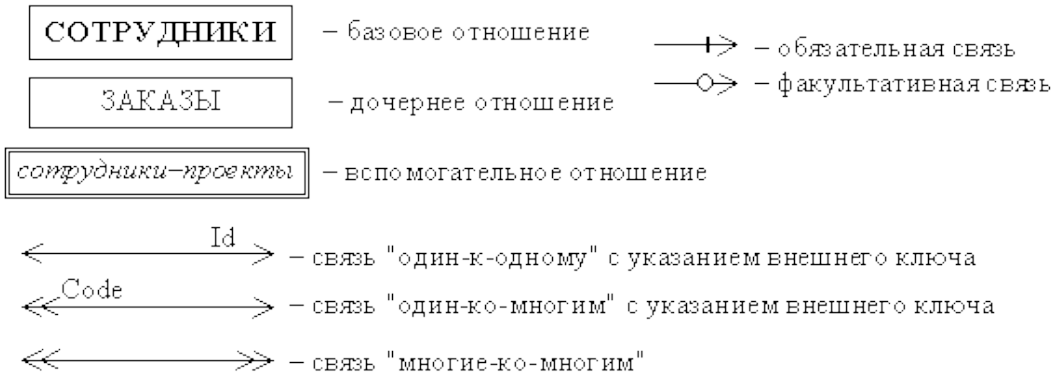

Рис.5. Обозначения, используемые на схеме базы данных

Полученная схема реляционной базы данных (РБД) приведена на рис. 6.

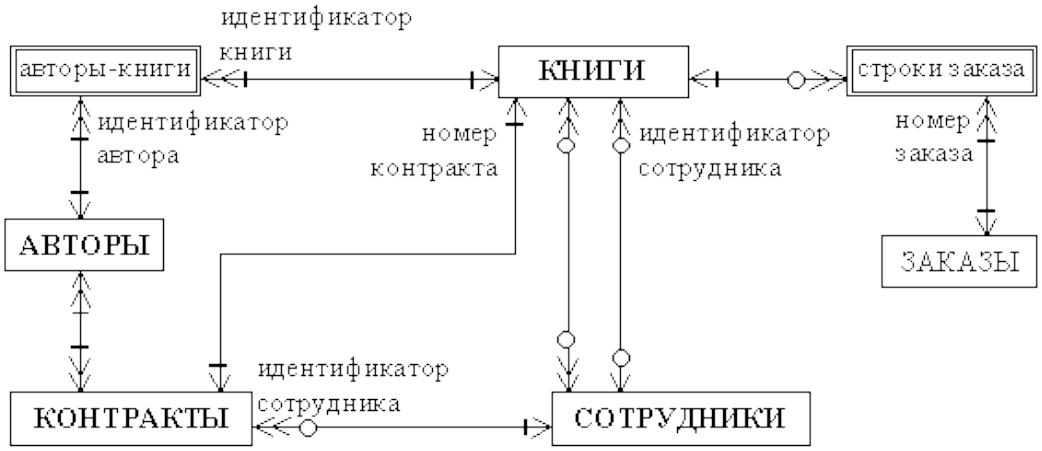

Рис.6. Схема РБД, полученная из ER–диаграммы издательской компании

На схеме (рис. 6) есть связь типа 1:1 – обязательная связь между *КНИГАМИ* и *КОНТРАКТАМИ*. Такие отношения следует объединять в одно. Дополнительный эффект от объединения этих отношений – слияние связей авторы–контракты и авторы–книги: ведь в нашем случае контракт заключается именно для написания книги.

> **Примечание:** исключение для связи типа 1:1 составляют ситуации, когда для увеличения производительности системы в отдельную таблицу выделяются редко используемые данные большого объёма.

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 *СОТРУДНИКИ* принадлежит к типу n:m (многие-ко-многим). Этот тип связи реализуется через вспомогательное отношение, которое является соединением первичных ключей соответствующих отношений.

Бинарная связь между отношениями не может быть обязательной для обоих отношений. После объединения сущностей *КНИГИ* и *КОНТРАКТЫ* остаётся три связи, обязательные для всех участников: между авторами и книгами и между заказами и строками заказов. Такой тип связи означает, что, например, прежде чем добавить новый заказ в отношение *ЗАКАЗЫ*, нужно добавить новую строку в отношение *СТРОКИ ЗАКАЗА*, и наоборот. Поэтому для такой связи необходимо снять с одной стороны условие обязательности. Так как все эти связи будут реализованы с помощью внешнего ключа, снимем условие обязательности связей для отношений, содержащих первичные ключи.

Уточнённая схема РБД издательской компании приведена на рис. 7.

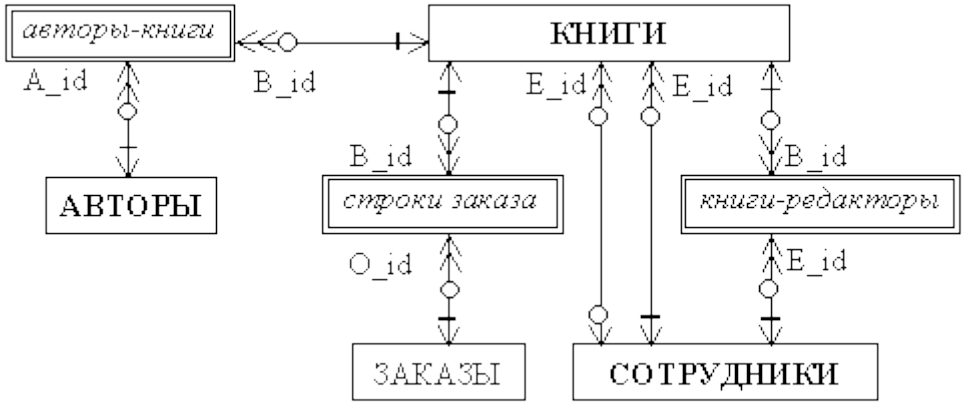

Рис.7. Уточнённая схема РБД издательской компании

Схема на рис. 7 содержит цикл "сотрудники–книги–сотрудники". Цикл допустим только в том случае, если связи, входящие в него, независимы друг от друга. Примем для нашей ПО, что ответственный редактор книги может являться также просто редактором этой же книги или не входить в число редакторов. При этом цикл не приводит к нарушению логической целостности данных.

> **Примечание.** Существует несколько подходов для разрешения ситуаций, в которых связи, входящие в цикл, зависят друг от друга. Рассмотрим пример цикла "отделы–проекты–сотрудники–отделы" (рис. 8,а). Будем считать, что в выполнении проекта могут участвовать только сотрудники, работающие в том же отделе, к которому относится проект. При циклической схеме СУБД не сможет гарантировать логическую целостность данных без использования дополнительных средств.
>
> Один из способов – разорвать цикл, исключив одну из связей (рис. 8,б) или введя промежуточное отношение (рис. 8,в). В нашем случае можно было бы разорвать связь "сотрудники–проекты", если бы каждый сотрудник участвовал во всех проектах своего отдела. Промежуточное отношение можно было бы использовать, если бы

существовала общая связь между сущностями, входящими в цикл. Например, если бы каждый сотрудник заключал договор с отделом на выполнение работ в рамках проекта. Тогда сущность *ДОГОВОРЫ* отражала бы связь между отделами, сотрудниками и проектами.

Другой способ разрешения цикла заключается в том, что в промежуточное отношение *СОТРУДНИКИ– ПРОЕКТЫ*, которое реализует связь многие-ко-многим, добавляются (мигрируют) внешние ключи *Код отдела* (*D_id*) из отношений *СОТРУДНИКИ* и *ПРОЕКТЫ* (рис. 8,г). Эти ключи проверяются на равенство друг другу с помощью соответствующего ограничения целостности. Использование этого способа возможно в том случае, когда соответствующие связи (*отдел–проект* и *отдел–сотрудник*) имеют тип один-ко-многим и являются обязательными.

В тех ситуациях, когда все эти способы не пригодны, логическая целостность контролируется программно или вручную.

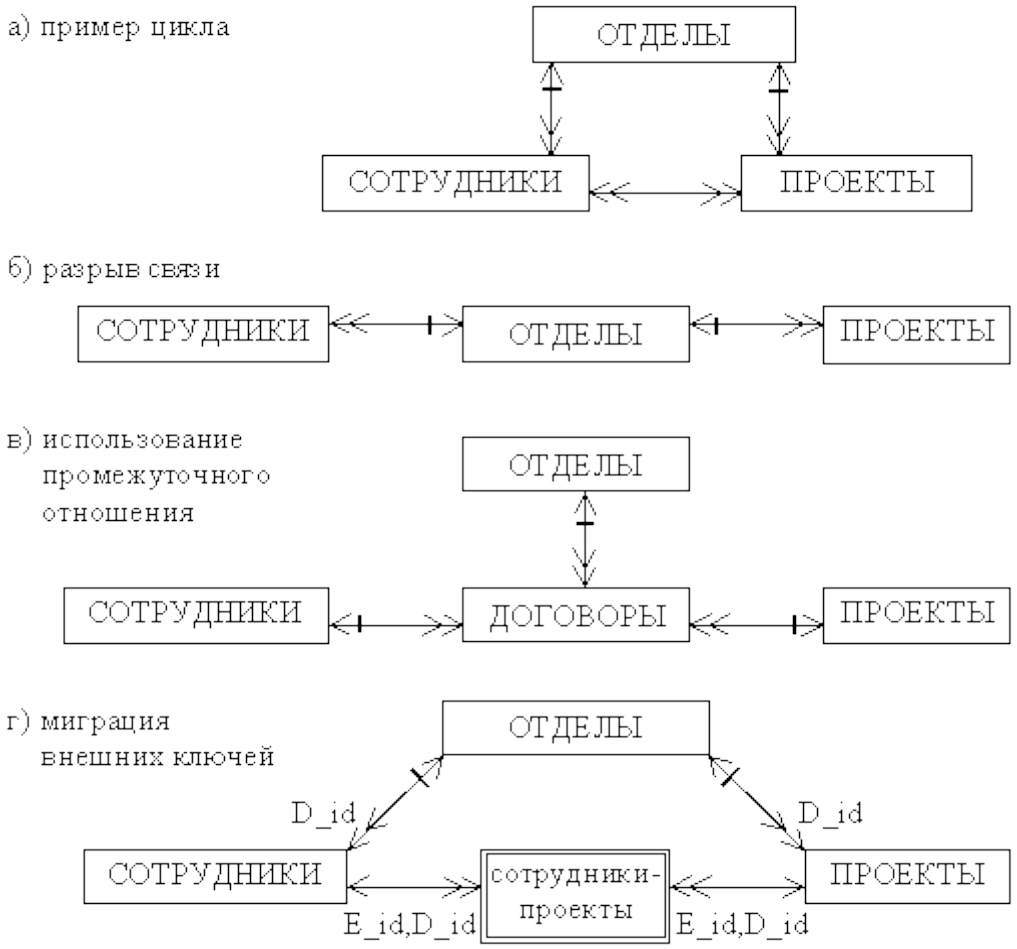

Рис.8. Некоторые способы разрешения циклов в схеме базы данных

3.2. Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту ПО) и в него вносятся все атрибуты сущности. Для каждого отношения необходимо определить первичный ключ и внешние ключи (если они есть). 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, счетчик).

> **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 табл. 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. 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 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 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 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 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 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 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

3.3. Нормализация полученных отношений (до 4НФ)

1НФ. 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:
>
> а) этого требует внешнее представление данных;
>
> б) в запросах поиск может осуществляться по отдельной части атрибута.

Разделим атрибуты *Фамилия, имя, отчество* на два атрибута *Фамилия* и *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 отношение *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.

2НФ. В нашем случае составные первичные ключи имеют отношения *СТРОКИ ЗАКАЗА*, *КНИГИ–АВТОРЫ* и *КНИГИ–РЕДАКТОРЫ*. Неключевые атрибуты этих отношений функционально полно зависят от первичных ключей.

3НФ. 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.

4НФ. Отношения данного примера не нарушают 4НФ, т.к. не содержат нетривиальных многозначных зависимостей.

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 (рис. 9):

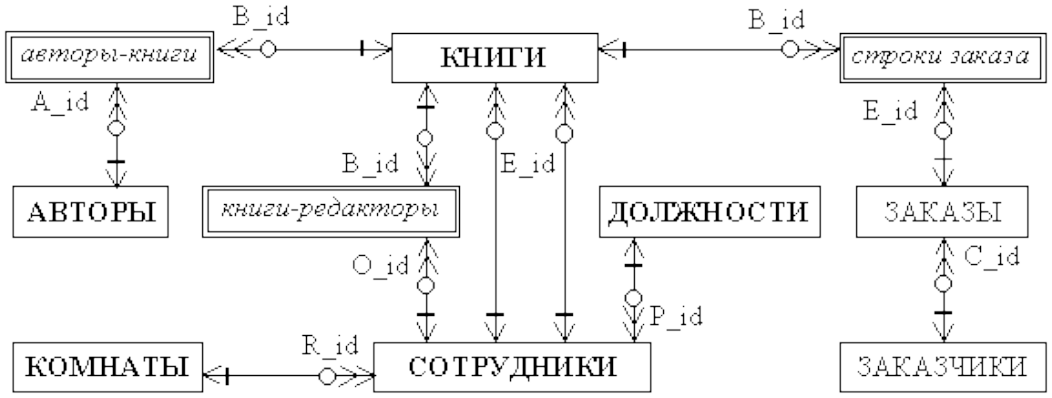

Рис.9. Окончательная схема РБД издательской компании

Окончательные схемы отношений базы данных с указанием ключей и других ограничений целостности (или лингвистическое описание) приведены в табл. 8–17.

Таблица 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 9. Схема отношения *КОМНАТЫ* (Rooms)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер комнаты	R_NO	N(3)	обязательное поле
Номер телефона	R_TEL	C(10)	

Таблица 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 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 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 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 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 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 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 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)

3.3. Определение дополнительных ограничений целостности

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

1. Значения всех числовых атрибутов – больше 0 (или null, если атрибут необязателен).
2. Область значений атрибута *Sex* отношения *EMPLOYEES* – символы 'м' и 'ж'.
3. Отношение *ROOMS* не имеет первичного ключа, но комбинация значений (*R_no*, *Tel*) уникальна.
4. В отношении *TITLES* порядковые номера авторов на обложке одной книги должны идти подряд, начиная с 1.
5. 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).

3.4. Описание групп пользователей и прав доступа

Опишем для каждой группы пользователей права доступа к каждой таблице и к каждому полю (атрибуту).

1. Администратор БД: имеет доступ ко всем данным (по записи), может изменять структуру базы данных и связи между отношениями. Устанавливает права доступа для всех остальных групп.
2. Представители администрации компании: имеют доступ по чтению ко всем данным и доступ по записи к отношениям *POSTS*, *ROOMS* и *EMPLOYEES*.
3. Менеджеры: имеет доступ по чтению ко всем данным, кроме отношения *POSTS*. Имеют доступ по записи к отношениям *AUTHORS*, *CUSTOMERS*, *BOOKS*, *EDITORS*, *TITLES*,*ORDERS*, *ITEMS*.
4. Редакторы: имеют доступ по чтению к следующим отношениям:
    - *AUTHORS*, кроме полей *A_passp*, *A_org*, *A_pdate*, *A_INN*(паспортные данные и ИНН).
    - *BOOKS*, кроме полей *B_advance*, *B_fee* (затраты и гонорар).
    - *EDITORS*.
    - *TITLES*.

1. Сотрудники, принимающие и выполняющие заказы: имеют доступ по записи к отношениям *CUSTOMERS*, *ORDERS*, *ITEMS* и по чтению к полям *B_title*, *B_circul*, *B_price*и *B_rest*отношения *BOOKS* (название, тираж, цена, непроданный остаток тиража).

4. Физическое проектирование БД

Мы условились не привязываться к конкретной СУБД и выполнять описание логической схемы БД на SQL-92. Следует заметить, что в некоторых СУБД использование базового SQL-92 для реализации схемы данных затруднительно.

Если операторы SQL, такие как chec, numeric и пр. не срабатывают, и СУБД выдает ошибку описания, нужно переключить настройки СУБД на использование стандарта ANSI SQL -92, или заменить эти операторы на другие, аналогичные. Если же использовать SQL-92 для реализации БД в выбранной Вами СУБД не представляется возможным, рекомендуется создать в режиме создания таблиц все спроектированные отношения и связать их в режиме "Схема данных"(как, например, в Access). Описание всех отношений на SQL -92 в пояснительной записке КП обязательно.

Приведём фрагмент описания схемы БД на DDL:

1. Отношение *POSTS* (должности):

```
create table posts (
p_id numeric(3) primary key,
p_post varchar(30) not null,
p_salary numeric(8,2) not null check(p_salary > 0));
```

2. Отношение *ROOMS* (комнаты):

```
create table rooms (
r_no numeric(3) primary key,
r_tel varchar(10),
unique(r_no, r_tel));
```

3. Отношение *EMPLOYEES* (сотрудники):

```
create table employees (
e_id numeric(4) primary key,
e_fname varchar(20) not null,
e_lname varchar(30) not null,
e_born date,
e_sex char(1) not null check(e_sex in ('ж','м')),
e_post numeric(3) references posts,
e_room numeric(3),
```

```
e_tel varchar(10),

e_inn char(12) not null,

e_passp char(12) not null,

e_org varchar(30) not null,

e_pdate date not null,

e_addr varchar(50),

foreign key(e_room,e_tel) references
rooms(r_no,r_tel));
```

Другие отношения описываются аналогично. В результате реализации в среде СУБД MS Access-2007 получается схема данных, представленная на рисунке 10.

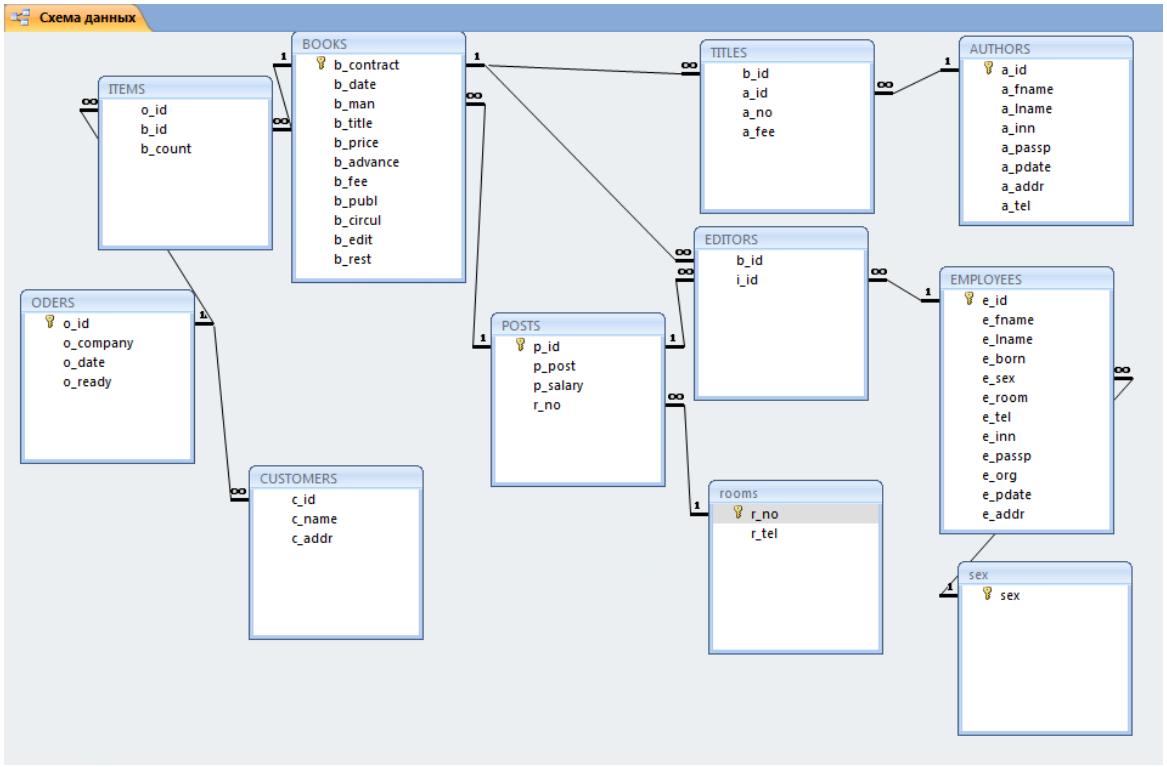

Рис.10 Схема данных в СУБД MS Access-2007

Права доступа пользователей, описанные в п. 2.4.4, предоставляются с помощью команды GRANT. Рассмотрим для примера права сотрудника компании ***user1***, который принимает и обслуживает заказы. Права доступа к отношениям *CUSTOMERS*, *ORDERS*, *ITEMS* могут быть описаны следующим образом:

```
grant insert, update on customers to user1;
```

```
grant insert, update, delete on orders to user1;

grant insert, update, delete on items to user1;
```

Для реализации частичного доступа к отношению *BOOKS* следует создать соответствующее представление и предоставить доступ к этому представлению:

```
create view goods (id, title, circul, price, rest)

as select b_contract, b_title, b_circul, b_price, b_rest

from books;

grant select on goods to user1;
```

Приведём примеры нескольких готовых запросов:

1. Список всех текущих проектов (книг, находящихся в печати и в продаже):

   ```
   create view edits as

   select b_contract, b_title, b_date

   from books

   where b_rest is null or b_rest > 0;
   ```

2. Список редакторов, работающих над книгами:

   ```
   create view edits (title, fname, lname) as

   select b_title, e_fname, e_lname /*ответственный редактор*/

   from books, employees e

   where b_edit=e_id and

   (b_publ is null or b_publ > sysdate);

   union /*sysdate – текущая дата*/

   select b_title, a_fname, a_lname

   from books, employees e, editors d

   where b.b_contract=d.b_id and d.e_id=a.e_id and

   (b_publ is null or b_publ > sysdate)
   ```

```
order by 1;
```

3. Определение общей прибыли от продаж по текущим проектам:

```
create view edits (title, total) as
    select b_title, (circul-rest)*price-advance
        from books
    where b_rest is null or b_rest > 0;
```

Анализ готовых запросов показывает, что для повышения эффективности работы с данными необходимо создать индексы для всех внешних ключей (и всех первичных ключей, если выбранная СУБД не создаёт их автоматически). Приведём примеры создания индексов:

```
create index e_posts on employees(e_post);

create index b_editors on books(b_edit);

create unique index r_tel on rooms(r_no,r_tel);
```

4. Реализация пользовательских запросов

В разделе приводятся:

- наименование запроса
- листинг инструкции SELECT
- скриншот результата запроса.

ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

Пояснительная записка должна как можно более подробно отражать ход выполнения курсового проекта. К пояснительной записке прилагаются распечатка программного текста и руководство пользователя (если разрабатываются экранные формы).

Реализация базы данных выполняется с помощью выбранной СУБД (или языка программирования, включающего функции работы с БД). Минимальная реализация системы подразумевает создание базы данных и запросов на SQL, осуществляющих выполнение тех функций, которые оговорены в задании. Обязательны скриншоты результатов запросов.

В том случае, если система реализуется не полностью, например, отсутствуют некоторые ограничения целостности или функциональные возможности, это должно быть указано в пояснительной записке.

В базу должен быть помещен тестовый набор данных.

ВАРИАНТЫ ЗАДАНИЙ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

3. БД "Плановый отдел".

Задача – информационная поддержка деятельности планового отдела (выбрать конкретное производство).

БД должна осуществлять:

- ведение плановой документации по основному и вспомогательному производствам (план и факт);
- составление заказов на поставку сырья и комплектующих (в соответствии с планом выпуска продукции);
- составление планов работы вспомогательных производств для обеспечения потребностей основного производства;
- подсчёт энергозатрат;
- определение соответствия результатов работы плану (в процентах).

Библиографический список (рекомендованный список использованных источников)

1. *Кузин А.В., Левонисова С.В*. Базы данных: учебное пособие для вузов. - 6-е изд., стер. - М. : Академия, 2016. - 315 с.
2. *Советов, Б.Я. Цехановский В.В., Чертовской В.Д.* Базы данных: учебник. — М.: Издательство Юрайт, 2017. — 463 с.
3. *Илюшечкин, В. М.* Основы использования и проектирования баз данных : учебник для СПО. — М. : Издательство Юрайт, 2017. — 213 с
4. *Цехановский В.В., Чертовской В.Д.* Управление данными: учебник для вузов – СПб.: Лань, 2015. - 432 с.
5. *Воробьева Е.Е.* Базы данных: конспект лекций. – СПб: БГТУ, 2015 (электронный ресурс). - elr2409.pdf.
6. *Маркин, А. В.* Программирование на SQL в 2 ч. Часть 1: учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М.: Издательство Юрайт, 2017. — 362 с.
7. *Маркин, А. В.* Программирование на SQL в 2 ч. Часть 2 : учебник и практикум для бакалавриата и магистратуры / А. В. Маркин. — М : Издательство Юрайт, 2017. — 292 с.
8. *Фаронов В.В.* Delphi. Программирование на языке высокого уровня: учебник для вузов. – СПб.: Питер, 2003. – 639 с.
9. *Верхолат А.М., Гаврилов В.А.* Проектирование структуры базы данных. Пособие по курсовому проектированию./ Балт. Гос. Техн. ун-т. – СПб., 2007. - 46 с.