Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования
Московский государственный институт электроники и математики
(Технический университет)

Кафедра вычислительных систем и сетей

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Методические указания

к курсовому проектированию

по курсу "Базы данных"

Москва

2010

Составитель к.т.н., доцент И.П. Карпова

УДК 681.3

Проектирование реляционных баз данных: Метод. указания к курсовому проектированию по курсу "Базы данных" / Московский государственный институт электроники и математики; Сост.: И.П. Карпова. – М., 2010. – 32 с.

Курсовое проектирование посвящено изучению методов проектирования реляционных баз данных, особое внимание уделено этапам инфологического и логического проектирования.

Для студентов III-IV курсов дневных и вечерних отделений технических факультетов вузов, изучающих автоматизированные информационные системы и системы управления базами данных.

Табл. 17. Ил. 7. Библиогр.: 5 назв.

ISBN 5-230-16273-2

СОДЕРЖАНИЕ

ЦЕЛИ РАБОТЫ

Цель курсового проектирования – применение на практике знаний, полученных в процессе изучения курса "Базы данных" [1], и получение практических навыков создания автоматизированных информационных систем (АИС), основанных на базах данных.

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

1.1. Общие положения

Проектирование базы данных (БД) является одной из наиболее сложных и ответственных задач, связанных с созданием АИС.

Проектирование базы данных – это процесс, который подразумевает использование определённой технологии. Никто не сомневается в том, что в случае нарушения технологии изготовления печатной платы, например, эта плата либо вообще не будет работать, либо не будет соответствовать заявленным характеристикам. Но почему-то считается, что соблюдать технологию проектирования БД (и вообще программного обеспечения) совершенно необязательно. И начинают работу по реализации реляционной БД с создания таблиц. Получившаяся в ходе такого "проектирования" база данных будет ненадёжной, неэффективной и сложной в сопровождении. (Исключением могут быть случаи простых предметных областей, которые можно отразить в базе данных, состоящей из 3-4 таблиц). Поэтому при создании базы данных необходимо придерживаться определённой технологии проектирования БД.

Опишем вкратце процесс проектирования реляционной базы данных. (Более подробно этот процесс изложен в [1, 2]).

База данных – это, фактически, модель предметной области (ПрО). Значит, для создания БД надо сначала проанализировать ПрО и создать её модель (это называется **инфологическим проектированием**).

Основой для *анализа предметной области* служат документы, которые отражают ПрО, и информация, которую можно получить от специалистов этой предметной области в процессе общения с ними.

Для анализа берутся те документы, которые имеют отношение к решаемой задаче. Изучение документов позволяет выявить объекты (сущности ПрО) и атрибуты сущностей – данные, которые должны храниться в БД.

Из общения со специалистами необходимо извлечь сведения об особенностях ПрО, которые позволяют установить ограничения целостности, зависимости и связи между объектами (субъектами) предметной области. Также специалисты обладают знаниями о том, каковы алгоритмы обработки данных и какие задачи ставятся перед информационной системой.

Модель ПрО может быть описана любым удобным для разработчика способом (словесное описание, набор формул, диаграмма потоков данных и т.п.). Но, если при проектировании баз данных используется метод сущность–связь, то схема ПрО выполняется в виде ER–диаграммы (entity-relation diagram, диаграмма «сущность-связь»).

После создания модели ПрО определяются **требования к операционной обстановке**: какое аппаратное и программное обеспечение необходимо для реализации БД и АИС в целом. Основные технические параметры (объём оперативной и дисковой памяти, наличие сетевой платы и др.) определяются исходя из планируемого объёма БД, режима работы (локальный или удалённый доступ) и требований к эффективности работы системы (например, ко времени реакции на запрос пользователя или к общей производительности БД). В зависимости от планируемой нагрузки (интенсивности запросов) и требований к надёжности выбирается операционная система. Затем осуществляется **выбор СУБД**, под управлением которой будет работать создаваемая база данных.

На следующем этапе – этапе **логического проектирования** – ER-диаграмма формальным способом преобразуется в схему реляционной базы данных (РБД). На основании схемы РБД и описания сущностей ПрО составляются отношения (таблицы) базы данных. Потом выполняется нормализация отношений. Это необходимо сделать для того, чтобы исключить нарушения логической целостности данных и повысить таким образом надёжность и достоверность данных. В отдельных случаях после нормализации может выполняться денормализация, но причина для этого может быть только одна: повышение эффективности выполнения критических запросов.

В результате всех этих операций создаётся концептуальная схема БД – основной документ для базы данных.

Далее, на этапе **физического проектирования** полученные отношения описываются на языке DDL (Data definition language) – языке определения данных, который поддерживается выбранной СУБД. Также необходимо определить способы хранения данных (кластеризация, хеширование) и способы доступа к данным (индексирование) и создать соответствующие индексы и кластеры (если нужно). Если пользователей АИС можно разделить на группы по характеру решаемых задач, то для каждой группы создаётся свой набор прав доступа к объектам БД.

1.2. Последовательность проектирования базы данных

Итак, процесс проектирования включает в себя следующие шаги:

1. Определение задач, стоящих перед базой данных.
2. Сбор и анализ документов, относящихся к исследуемой предметной области.
3. Описание особенностей ПрО, которые позволяют установить зависимости и связи между объектами (субъектами) предметной области.
4. Создание модели предметной области.
5. Определение групп пользователей и перечня задач, стоящих перед каждой группой.
6. Выбор аппаратной и программной платформы для реализации БД.
7. Выбор СУБД (системы управления базой данных).
8. Создание логической схемы БД.
9. Создание схем отношений, определение типов данных атрибутов и ограничений целостности.
10. Нормализация отношений (до третьей или четвёртой нормальной формы).

11. Определение прав доступа пользователей к объектам БД.
12. Написание текста создания основных объектов базы данных на языке SQL в синтаксисе выбранной СУБД (пользователи, таблицы и др.).
13. Написание текста создания вспомогательных объектов базы данных (представления, индексы, триггеры, роли и т.д.).

Эти шаги можно объединить с 5 этапов:

1. Инфологическое проектирование (1-5).
2. Определение требований к операционной обстановке, в которой будет функционировать информационная система (6).
3. Выбор системы управления базой данных (СУБД) и других инструментальных программных средств (7).
4. Логическое проектирование БД (8-11).
5. Физическое проектирование БД (12-13).

На сегодняшний день не существует формальных способов моделирования реальности, но инфологический подход закладывает основы методологии проектирования базы данных как модели предметной области.

1.2.1. Инфологическое проектирование

Основными задачами этапа инфологического проектирования являются определение предметной области системы и формирование взгляда на неё с позиций сообщества будущих пользователей БД, т.е. информационно-логической 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 ПрО из одного состояния в другое.

Основными подходами к созданию инфологической модели предметной области являются [1]:

1. Функциональный подход к проектированию БД ("от задач").
2. Предметный подход к проектированию БД ("от предметной области").
3. Метод "сущность-связь" (entity–relation, ER–method).

Мы будем использовать метод "сущность–связь" как наиболее распространённый. Приведём основные термины, которыми мы будем пользоваться:

Сущность – это объект, о котором в системе будут накапливаться данные. Для сущности указывается название и тип (сильная или слабая). Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных.

Атрибут – свойство сущности. Различают:

1) *Идентифицирующие и описательные атрибуты*. Идентифицирующие позволяют отличить один экземпляр сущности от другого. Описательные атрибуты заключают в себе интересующие нас свойства сущности.

2) *Составные и простые атрибуты*. Простой атрибут имеет неделимое значение. Составной атрибут является комбинацией нескольких элементов, возможно, принадлежащих разным типам данных (ФИО, адрес и др.).
3) *Однозначные и многозначные атрибуты* (могут иметь соответственно одно или много значений для каждого экземпляра сущности). Например, дата рождения – это однозначный атрибут, а номер телефона – многозначный.
4) *Основные и производные атрибуты*. Значение основного атрибута не зависит от других атрибутов; значение производного атрибута вычисляется на основе значений других атрибутов. Например, возраст вычисляется на основе даты рождения и текущей даты.
5) *Обязательные и необязательные* (первые должны быть указаны при размещении данных в БД, вторые могут не указываться).

Для каждого атрибута необходимо определить название, указать тип данных и описать ограничения целостности – множество значений, которые может принимать данный атрибут.

Связь – это осмысленная ассоциация между сущностями. Для связи указывается название, тип (факультативная или обязательная), кардинальность (1:1, 1:n или m:n) и степень (унарная, бинарная, тернарная или n-арная).

На рис. 1 приведены обозначения, которые мы будем использовать в ER-диаграммах.

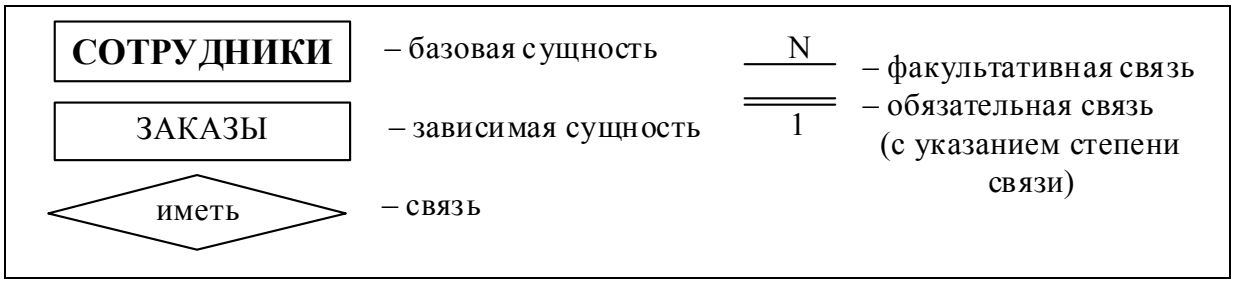

Рис.1. Обозначения, используемые в ER-диаграммах

1.2.2. 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].

1.2.3. Выбор СУБД и других программных средств

Выбор СУБД осуществляется на основании таких критериев, как тип модели данных и её адекватность потребностям рассматриваемой ПрО; характеристики производительности; набор функциональных возможностей; удобство

и надежность СУБД в эксплуатации; стоимость СУБД и дополнительного программного обеспечения [1].

1.2.4. Логическое проектирование реляционной БД

На этапе логического проектирования разрабатывается логическая (концептуальная) структура БД. Для реляционной модели существуют формальные правила, которые позволяют преобразовать инфологическую модель ПрО в виде ER-диаграммы в логическую схему базы данных. Кроме получения схемы БД в целом на этом этапе выполняют создание схем отношений и их нормализацию. Этот этап более подробно рассмотрен в п.2 "Пример проектирования реляционной базы данных".

1.2.5. Физическое проектирование БД

Этап физического проектирования заключается в определении схемы хранения, т.е. физической структуры БД. Схема хранения зависит от той физической структуры, которую поддерживает выбранная СУБД. Физическая структура БД, с одной стороны, должна адекватно отражать логическую структуру БД, а с другой стороны, должна обеспечивать эффективное размещение данных и быстрый доступ к ним. Результаты этого этапа документируются в форме схемы хранения на языке определения данных (DDL, Data Definition Language) выбранной СУБД. Принятые на этом этапе решения оказывают огромное 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 на этапе физического проектирования разрабатывается стратегия резервного копирования. Для защиты от несанкционированного доступа каждому пользователю доступ к данным предоставляется только в соответствии с его правами доступа, набор которых также является составной частью проекта БД.

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 ПрО и неразвитостью средств описания ограничений целостности на уровне модели данных.

Для решения подобных проблем проводится **нормализация отношений**.

Механизм нормализации реляционных отношений разработал Э.Ф. Кодд (E.F. Codd). Этот механизм позволяет по формальным признакам 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.

Первая нормальная форма относится к понятию простого и сложного (составного или многозначного) атрибута (см. п.1.2.1).

Первая нормальная форма (1НФ).

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

Для того чтобы привести к 1НФ отношение, содержащее сложные атрибуты, нужно:

1) разбить составные атрибуты на простые,
2) построить декартово произведение всех многозначных атрибутов с кортежами, к которым они относятся.

Для идентификации кортежа в этом случае понадобится составной ключ, включающий первичный ключ исходного отношения и все многозначные атрибуты.

Вторая нормальная форма основана на понятии *функциональной зависимости*. Пусть X и Y – атрибуты некоторого отношения. Если в любой момент времени каждому значению X соответствует единственное значение Y, то говорят, что Y функционально зависит от X ($X \rightarrow Y$). Атрибут X в функциональной зависимости $X \rightarrow Y$ называется *детерминантом* отношения.

В нормализованном отношении все неключевые атрибуты функционально зависят от ключа отношения. 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 первичного ключа.

(Таким образом, если отношение в 1НФ имеет простой первичный ключ, оно сразу находится во второй нормальной форме).

Для того чтобы привести отношение ко 2НФ, нужно:

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

Третья нормальная форма основана на понятии *транзитивной зависимости*. Пусть X, Y, Z – атрибуты некоторого отношения. При этом $X \rightarrow Y$ и $Y \rightarrow Z$, 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 ($X \rightarrow\rightarrow Z$).

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 проекцию, исключив транзитивно зависящие от ключа атрибуты;
- построить дополнительно одну или несколько проекций на детерминанты исходного отношения и атрибуты, функционально зависящие от них.

Исключение составляют случаи, когда для транзитивной зависимости $X \rightarrow\rightarrow Z$ ($X \rightarrow Y$ и $Y \rightarrow Z$) либо Z зависит от Y, либо Y зависит от X, т.е. между атрибутами X и Y, например, существует связь 1:1. В такой ситуации декомпозиция отношения не производится.

Четвертая нормальная форма основана на понятии *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 $Y \subset X$ или $X \cup Y = R$, 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.

Описание этого метода достаточно подробно изложено в [1], поэтому, чтобы не повторяться, будем рассматривать этот метод на примере проектирования конкретной базы данных.

2. ПРИМЕР ПРОЕКТИРОВАНИЯ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

В качестве примера возьмем базу данных проектной организации. Основной вид деятельности такой организации – выполнение проектов по договорам с заказчиками.

2.1. Инфологическое проектирование

2.1.1. Анализ предметной области

База данных создаётся для информационного обслуживания руководства организации, руководителей проектов и участников проектов. БД должна содержать данные об отделах организации, сотрудниках и проектах.

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

- Каждый сотрудник работает в определённом отделе, в каждом отделе могут работать несколько сотрудников.
- Каждый проект относится к определённому отделу, каждый отдел может отвечать за выполнение нескольких проектов.
- Каждый сотрудник может принимать участие в выполнении нескольких проектов, над каждым проектом может трудиться несколько сотрудников.
- Для каждого проекта назначается руководитель из числа сотрудников того отдела, к которому относится проект.
- Каждый проект должен быть выполнен в заданные сроки, каждый проект может состоять из нескольких этапов. Если проект состоит из одного этапа, то сроки его выполнения должны совпадать со сроками выполнения проекта в целом.
- Оклад сотрудника зависит от занимаемой должности, за участие в проектах сотрудник получает дополнительное вознаграждение.
- Виды участия сотрудников в проектах: руководитель, консультант, исполнитель.
- Каждый отдел занимает одно или несколько помещений (комнат), в каждом помещении может быть один или несколько стационарных телефонов.

Примечание. Описания особенностей ПрО должно быть достаточно для того, чтобы создать ER–диаграмму.

Для создания ER-модели необходимо выделить сущности предметной области:

1) **Отделы**. Атрибуты: название, аббревиатура, комнаты, телефоны.
2) **Сотрудники**. Атрибуты: ФИО, паспортные данные, дата рождения, пол, ИНН (индивидуальный номер налогоплательщика), номер пенсионного страхового свидетельства, адреса, телефоны (рабочий, домашний, мобильный), данные об образовании (вид образования (высшее, среднеспециальное и т.д.), специальность, номер диплома, дата окончания учебного заведения), должность, оклад, логин (имя пользователя).

Примечания: 1. Логин потребуется нам для назначения дифференцированных прав доступа.
2. В нашем задании не предусмотрена полная информационная поддержка сотрудников отдела кадров, поэтому мы не будем отражать в БД такие сведения как дату поступления сотрудника на работу, его переводы с одной должности на другую, уходы в отпуска и т.п.

3) **Проекты**. Атрибуты: номер договора; полное название проекта; сокращённое название проекта; дата подписания договора; заказчик; контактные данные заказчика; дата начала проекта; дата завершения проекта; сумма по проекту; дата реальной сдачи проекта; сумма, полученная по проекту на текущую дату.
4) **Этапы проекта**. Атрибуты: номер по порядку, название, дата начала этапа, дата завершения этапа, форма отчетности, сумма по этапу, дата реальной сдачи этапа; сумма, полученная по этапу на текущую дату.

Исходя из выявленных сущностей, построим ER–диаграмму (рис. 2). Напомним, что пометки у линий означают степень связи: 1:1, 1:N и N:M.

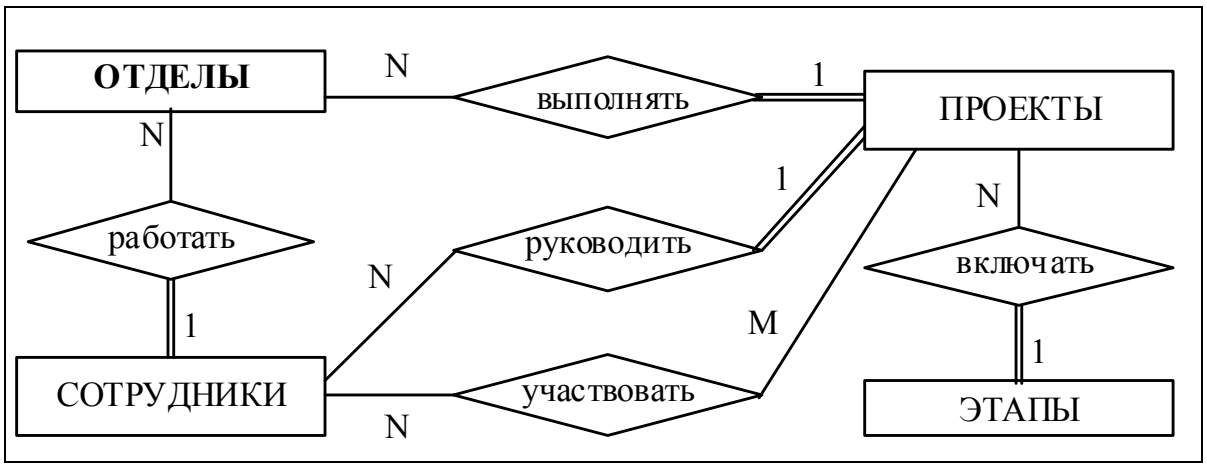

Рис. 2. ER–диаграмма ПрО «Проектная организация»

2.1.2. Анализ информационных задач и круга пользователей системы

Определим группы пользователей, их основные задачи и запросы к БД:

1. Руководители организации:
   - заключение новых договоров;
   - назначение руководителей проектов;
   - получение списка всех участников проектов;
   - изменение должностных окладов и штатного расписания;
   - получение полной информации о проектах;
   - внесение изменений в данные о проектах;
   - архивирование данных по завершённым проектам.

Примечание. Архивирование данных в этом пособии подробно не рассматривается. Это сделано для того, чтобы не перегружать схему БД.

2. Руководитель проекта:
   - назначение участников проекта;
   - получение списка сотрудников, работающих над конкретным проектом;
   - получение полной информации о проекте, руководителем которого он является;
   - получение сведений о сотрудниках, которые могут стать участниками проекта;
   - определение размера дополнительного вознаграждения сотрудников по конкретному проекту;
   - внесение изменений в данные об этапах проекта.
3. Сотрудники отдела кадров:
   - приём/увольнение сотрудников;
   - внесение изменений в данные о сотрудниках.
4. Бухгалтеры:
   - получение ведомости на выплату зарплаты.
5. Сотрудники – участники проектов:
   - просмотр данных о других участниках проекта;
   - просмотр данных о сроках сдачи проекта и форме отчётности.

2.2. Определение требований к операционной обстановке

Для выполнения этого этапа необходимо знать (хотя бы ориентировочно) объём работы организации (т.е. количество проектов и сотрудников), 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 ($М_Д$). Для реальных баз данных обычно наиболее существенным является $М_Д$.

На основе результатов анализа ПрО можно приблизительно оценить объём памяти, требуемой для хранения данных. Примем ориентировочно, что:

- одновременно осуществляется около десяти проектов, работа над проектом продолжается в среднем год (по 1К на каждый проект);
- каждый проект состоит в среднем из четырёх этапов (по 0,5К на этап);
- в компании работают 100 сотрудников (по 0,5К на каждого сотрудника);
- в выполнении каждого проекта в среднем участвуют 10 сотрудников (по 0,2К);
- устаревшие данные переводятся в архив (накапливаются в архиве БД).

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

$$М_д = 2(10*1+10*4*0,5+100*0,5+(10*10*0,2)) = 200\,К,$$

Коэффициент 2 необходим для того, чтобы учесть необходимость выделения памяти под дополнительные структуры (например, индексы). Объём памяти будет увеличиваться ежегодно на столько же при сохранении объёма работы.

Требуемый объём оперативной памяти определяется на основании анализа интенсивности запросов и объёма результирующих данных. Для нашей БД требуемый объём памяти мал, поэтому никаких специальных требований к объёму внешней и оперативной памяти компьютера не предъявляется.

2.3. Выбор СУБД и других программных средств

Анализ информационных задач показывает, что для реализации требуемых функций подходят почти все СУБД для ПЭВМ (MS Access, Firebird, MySQL 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.

Примечание. При выполнении курсового проекта необходимо обосновать выбор конкретной СУБД для реализации проекта и реализовать базу данных под управлением выбранной СУБД.

2.4. Логическое проектирование реляционной БД

2.4.1. 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, содержащую атрибуты сущностей (рис. 3).

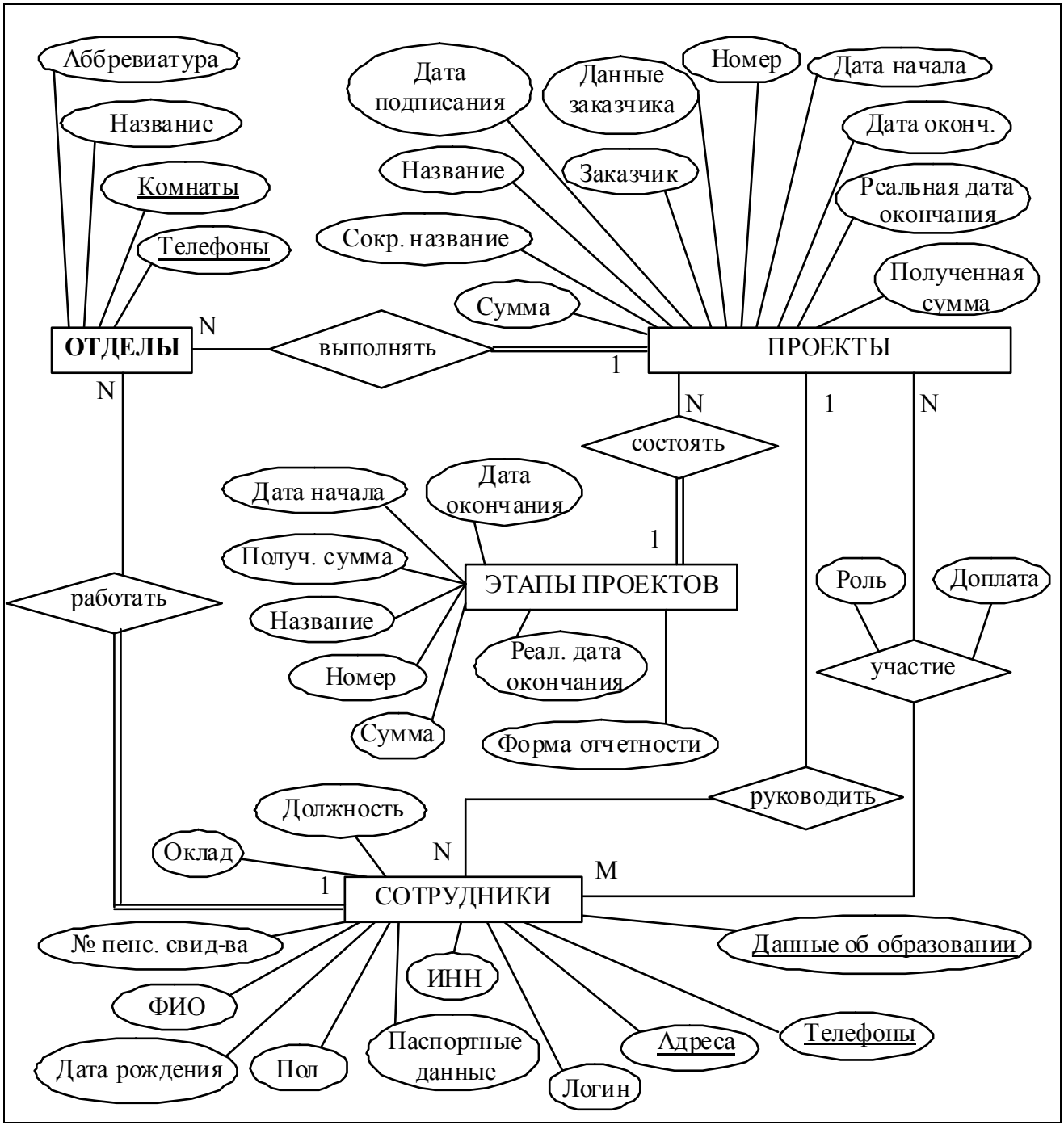

Рис. 3. Уточнённая ER–диаграмма проектной организации

Примечание. Многозначные атрибуты на рисунке выделены подчеркиванием.

Преобразование ER–диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таб-

лицы) БД. Связь типа 1:n (один-ко-многим) между отношениями реализуется через внешний ключ. Ключ вводится для того отношения, к которому осуществляется множественная связь. Внешнему ключу должен соответствовать первичный или уникальный ключ основного (родительского) отношения.

Связь *участвовать* между *ПРОЕКТАМИ* и *СОТРУДНИКАМИ* принадлежит к типу n:m (многие-ко-многим). Этот тип связи реализуется через вспомогательное отношение *Участие*, которое содержит комбинации первичных ключей соответствующих исходных отношений.

Более подробно о принципах преобразования ER-диаграммы в схему БД рассказано в [1].

Для схемы БД будем использовать обозначения, представленные на рис. 4.

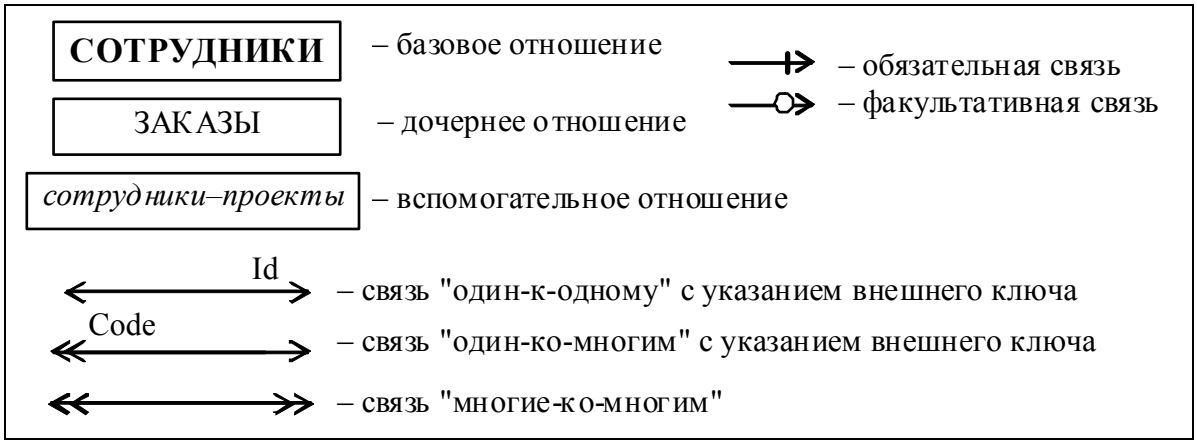

Рис. 4. Обозначения, используемые на схеме базы данных

Полученная схема реляционной базы данных (РБД) приведена на рис. 5.

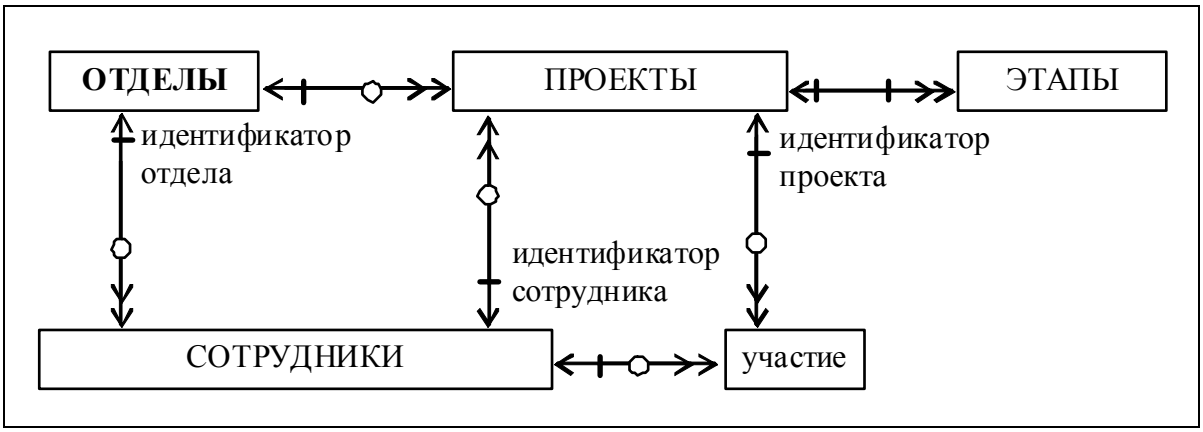

Рис. 5. Схема РБД, полученная из ER–диаграммы проектной организации

Бинарная связь между отношениями не может быть обязательной для обоих отношений. Такой тип связи означает, что, например, прежде чем добавить новый проект в отношение *ПРОЕКТЫ*, нужно добавить новую строку в отношение *ЭТАПЫ*, 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.

Схема на рис. 5 содержит три цикла: "сотрудники–проекты–участие–сотрудники", "отделы–сотрудники–проекты–отделы" и "отделы–сотрудники–участие–проекты–отделы". Цикл допустим только в том случае, если связи, входящие в него, независимы друг от друга. Например, для нашей ПрО справедливо такое правило: сотрудник любого отдела может быть участником (исполнителем или консультантом) проекта любого отдела. Эти связи независимы, поэтому цикл "отделы–сотрудники–участие–проекты–отделы" не будет приводить к нарушению логической целостности данных.

С другой стороны, только сотрудник отдела, отвечающего за выполнение проекта, может быть руководителем проекта. Но система не помешает нам назначить руководителем проекта сотрудника любого отдела. При добавлении проекта с внешним ключом *Руководитель* система проверит только, что такой человек есть в таблице *СОТРУДНИКИ*. А значение внешних ключей *Отдел* в таблицах *СОТРУДНИКИ* и *ПРОЕКТЫ* сравнивать не будет.

Таким образом, остальные циклы могут приводить к возможности нарушения логической целостности данных. Существует несколько подходов для разрешения ситуаций, в которых связи, входящие в цикл, зависят друг от друга.

Рассмотрим эту ситуацию в общем случае. Сначала слегка упростим схему: реализуем связь "руководить" через таблицу *УЧАСТИЕ* – это позволит не отвлекаться на малозначительные детали.

Будем считать, что в выполнении проекта могут участвовать только сотрудники, работающие в том же отделе, к которому относится проект (рис. 6,а). При циклической схеме СУБД не сможет гарантировать логическую целостность данных без использования дополнительных средств.

Один из способов разрешения таких ситуаций – разорвать цикл, исключив одну из связей (рис. 6,б) или введя промежуточное отношение (рис. 6,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, то отношение *ДОГОВОРЫ* отражало бы связь между отделом, сотрудником и проектом.

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. 6,г). Эти ключи проверяются на равенство друг другу с помощью соответствующего ограничения целостности (check). 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 непригодны, логическая целостность контролируется программно или вручную. Если принято решение переложить обязанности по контролю за логической целостностью данных на пользователя, то эти обязанности должны быть отражены в документации (в руководстве пользователя).

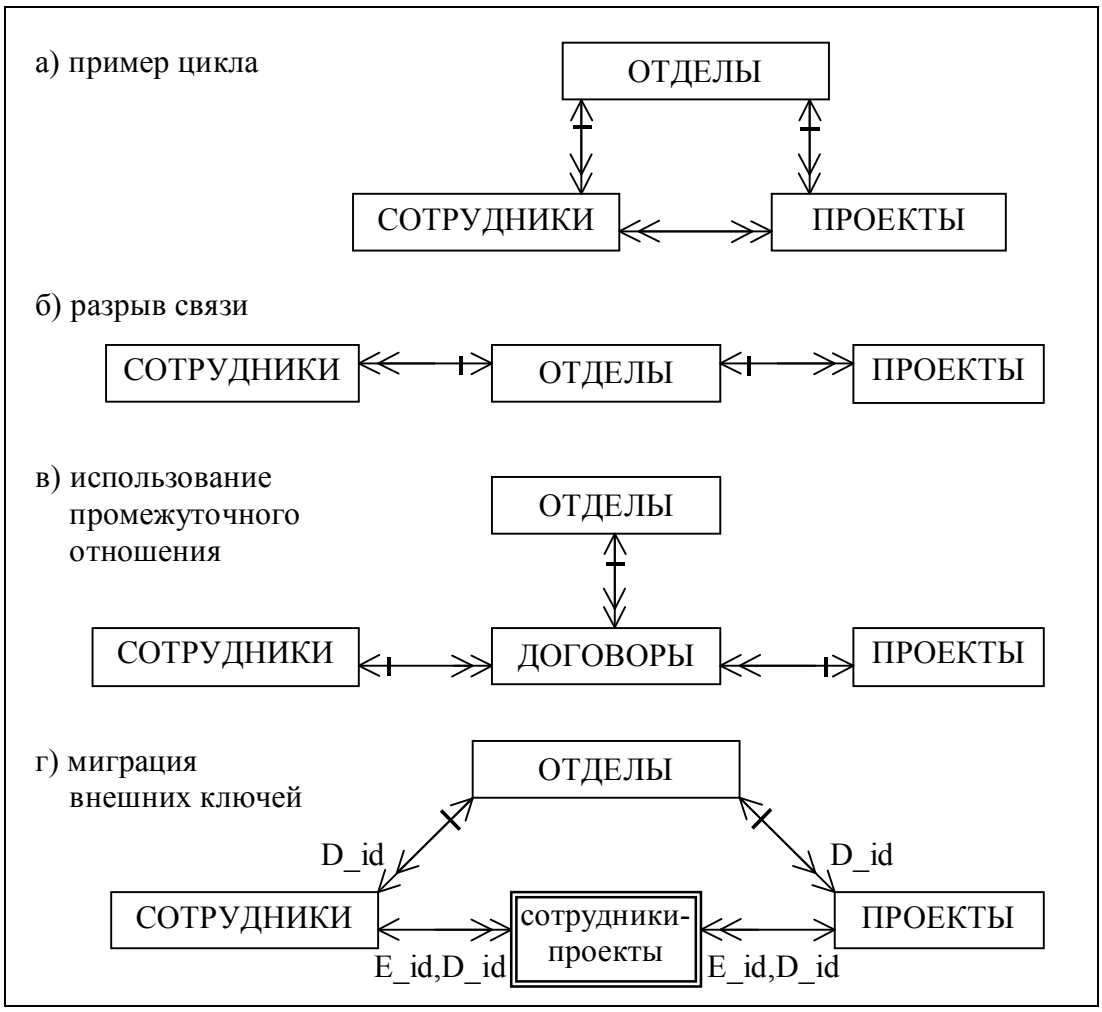

Рис.6. Некоторые способы разрешения циклов в схеме базы данных

Примем для нашей ПрО, что руководитель проекта может одновременно выполнять и другие обязанности в этом проекте, чтобы цикл "сотрудники–проекты–участие–сотрудники" не приводил к возможности нарушения логической целостности данных. Зато цикл "отделы–сотрудники (руководители)–проекты–отделы" включает зависимые связи: руководитель проекта назначается из того отдела, который отвечает за выполнение проекта в целом. Здесь можно было бы применить разрыв связи "отделы–проекты" и определять, к какому отделу относится проект через руководителя (по отделу руководителя проекта). Но такой подход в данном случае имеет существенный недостаток. Заменив руководителя проекта сотрудником другого отдела, можно одновременно изменить отдел, отвечающий за выполнение проекта, т.е. объединить в одно действие два независимых изменения, а это недопустимо.

Исходя из вышесказанного мы не будем разрывать связь, а примем решение реализовать эту проверку программно. Приложение должно будет при назначении руководителя проекта выдавать список сотрудников того отдела, который отвечает за выполнение данного проекта. Руководителя можно будет выбрать только из этого списка, а не вводить вручную.

2.4.2. Составление реляционных отношений

Каждое реляционное отношение соответствует одной сущности (объекту ПрО) и в него вносятся все атрибуты этой сущности. Для каждого отношения определяются первичный ключ и внешние ключи (в соответствии со схемой БД). 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.

Отношения приведены в табл. 1-5. Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: N – числовой, C – символьный тип фиксированной длины, V – символьный тип переменной длины, D – дата (этот тип имеет стандартную длину, зависящую от СУБД, поэтому она не указывается). О правилах выбора типов данных подробно рассказано в [1].

Потенциальными ключами отношения *ОТДЕЛЫ* являются атрибуты *Аббревиатура* и *Название отдела*. Первый занимает меньше места, поэтому мы выбираем его в качестве первичного ключа.

Таблица 1. Схема отношения *ОТДЕЛЫ* (Departs)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Аббревиатура отдела	D_ID	C(10)	**первичный ключ**
Название отдела	D_NAME	V(100)	обязательное поле
Комнаты	D_ROOMS	V(20)	обязательное многозначное поле
Телефоны	D_PHONE	V(40)	обязательное многозначное поле

Потенциальными ключами отношения *СОТРУДНИКИ* являются поля *Паспортные данные*, *ИНН* и *Номер страхового пенсионного свидетельства*. Все они занимают достаточно много места, а паспортные данные кроме того могут меняться. Введём суррогатный первичный ключ *Номер сотрудника*.

Таблица 2. Схема отношения *СОТРУДНИКИ* (Employees)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер	E_ID	N(4)	**суррогатный первичный ключ**
Фамилия, имя, отчество	E_NAME	V(50)	обязательное поле
Дата рождения	E_BORN	D	обязательное поле
Пол	E_SEX	C(1)	обязательное поле, 'м' или 'ж'
Паспортные данные	E_PASP	V(50)	обязательное поле
ИНН	E_INN	C(12)	обязательное уникальное поле
Номер пенсионного страхового свидетельства	E_PENS	C(14)	обязательное уникальное поле
Отдел	E_DEPART	C(10)	внешний ключ (к Departs)
Должность	E_POST	V(30)	обязательное поле
Оклад	E_SAL	N(8,2)	обязательное поле, > 4500 руб.
Данные об образовании	E_EDU	V(200)	обязательное многозначное поле
Адреса	E_ADDR	V(100)	многозначное поле
Телефоны	E_PHONE	V(30)	многозначное поле
Логин	E_LOGIN	V(30)	

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).

В отношении *ПРОЕКТЫ* три потенциальных ключа: *Номер проекта*, *Название проекта* и *Сокращённое названиие*. Меньше места занимает первый из них, но он малоинформативен. Зато сокращённое название, используемое в качестве внешнего ключа в других таблицах, позволит специалисту идентифицировать проект без необходимости соединения с отношением *ПРОЕКТЫ*.

Таблица 3. Схема отношения *ПРОЕКТЫ* (Projects)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер проекта	P_ID	N(6)	обязательное уникальное поле
Название проекта	P_TITLE	V(100)	обязательное поле
Сокращённое название	P_ABBR	C(10)	**первичный ключ**
Отдел	P_DEPART	C(10)	внешний ключ (к Departs)
Заказчик	P_COMPANY	V(40)	обязательное поле
Данные заказчика	P_LINKS	V(200)	обязательное поле
Руководитель	P_CHIEF	N(4)	внешний ключ (к Employees)
Дата начала проекта	P_BEGIN	D	обязательное поле
Дата окончания проекта	P_END	D	обязательное поле, больше даты начала проекта
Реальная дата окончания	P_FINISH	D	
Стоимость проекта	P_COST	N(10)	обязательное поле
Полученная сумма	P_SUM	N(10)	обязательное поле, значение по умолчанию – 0

Потенциальным ключом отношения *ЭТАПЫ* является комбинация внешнего ключа и номера этапа, а потенциальным ключом вспомогательного отношения *УЧАСТИЕ* является комбинация первых трёх полей этого отношения. Можно вообще не вводить первичный ключ для данных отношений, т.к. на них никто не ссылается. Но уникальность этих комбинации является в данном случае ограничением целостности данных, поэтому мы возьмём эти комбинации в качестве первичных ключей соответствующих отношений.

Таблица 4. Схема отношения *ЭТАПЫ ПРОЕКТА* (Stages)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*	
Проект	S_PRO	C(10)	внешний ключ (к Projects)	**составной первичный ключ**
Номер этапа	S_NUM	N(2)		
Название этапа	S_TITLE	V(200)	обязательное поле	
Дата начала этапа	S_BEGIN	D	обязательное поле	
Дата окончания этапа	S_END	D	обязательное поле, > даты начала	
Реальная дата окончания	S_FINISH	D	больше даты начала этапа	
Стоимость этапа	S_COST	N(10)	обязательное поле	
Полученная сумма по этапу	S_SUM	N(10)	обязательное поле, значение по умолчанию – 0	
Форма отчётности	S_FORM	V(100)	обязательное поле	

Таблица 5. Схема отношения *УЧАСТИЕ* (Job)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания* *
Проект	J_PRO	C(10)	внешний ключ (к Projects)
Сотрудник	J_EMP	N(4)	внешний ключ (к Employees)
Роль	J_ROLE	V(20)	обязательное поле
Доплата	J_BONUS	N(2)	

* – в отношении *УЧАСТИЕ* первичный ключ состоит из первых 3-х полей этого отношения.

2.4.3. Нормализация полученных отношений (до 4НФ)

Механизм нормализации подразумевает определённую последовательность преобразования отношений к третьей нормальной форме. Мы не будем чётко придерживаться этой последовательности, т.к. она избыточна, и многозначные атрибуты сразу вынесем в отдельные отношения на первом же этапе.

1НФ. Для приведения таблиц к 1НФ требуется составить прямоугольные таблицы (одно значение атрибута – одна ячейка таблицы) и разбить сложные атрибуты на простые.

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 атрибут *Фамилия, имя, отчество* на два атрибута *Фамилия* и *Имя, отчество*, *Паспортные данные* на *Номер паспорта* (уникальный), *Дата выдачи* и *Кем выдан*, а *Данные об образовании* – на *Вид образования*, *Специальность*, *Номер диплома* и *Год окончания учебного заведения*.

Многозначные атрибуты *Комнаты* и *Телефоны* из отношения *ОТДЕЛЫ* вынесем в отдельное отношение *КОМНАТЫ*, а домашние и мобильные телефоны и адреса сотрудников – в отношение *АДРЕСА-ТЕЛЕФОНЫ*. Так как в комнате может не быть телефона, первичный ключ отношения *КОМНАТЫ* не определен (ПК не может содержать null–значения), но на этих атрибутах можно определить составной уникальный ключ. В отношении *АДРЕСА-ТЕЛЕФОНЫ* также нет потенциальных ключей: оставим это отношение без первичного ключа, т.к. на это отношение никто не ссылается. Данные об образовании сотрудников также вынесем в отдельное отношение.

Что касается рабочих телефонов сотрудников, то один из этих номеров – основной – определяется рабочим местом сотрудника (рассматриваются только стационарные телефоны). Будем хранить этот номер в атрибуте *Рабочий телефон*. Наличие других номеров зависит от того, есть ли в том же помещении (комнате) другие сотрудники, имеющие стационарные телефоны. Добавим в отношение *СОТРУДНИКИ* атрибут *Номер комнаты*, чтобы дополнительные номера телефонов сотрудника можно было вычислить из других кортежей с таким же номером комнаты.

Связь между отношениями *СОТРУДНИКИ* и *КОМНАТЫ* реализуем через составной внешний ключ (*Номер комнаты*, *Рабочий телефон*).

Мы также удалим вычислимый атрибут *Полученная сумма* из отношения *ПРОЕКТЫ*, т.к. он является суммой значений аналогичного атрибута из от-

ношения *ЭТАПЫ ПРОЕКТОВ*. Но атрибут *Стоимость проекта* оставим, т.к. она фигурирует в документации по проекту. А для обеспечения логической целостности данных предусмотрим в приложении проверку того, что сумма по всем этапам совпадает со стоимостью проекта.

2НФ. В нашем случае составные первичные ключи имеют отношения *ЭТАПЫ ПРОЕКТА* и *УЧАСТИЕ*. Неключевые атрибуты этих отношений функционально полно зависят от составных первичных ключей.

3НФ. В отношении *ПРОЕКТЫ* атрибут *Данные заказчика* зависит от атрибута *Заказчик*, а не от первичного ключа, поэтому его 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 *Данные заказчика* и ввести для него суррогатный ПК. Так как с каждым заказчиком может быть связано несколько проектов, связь между отношениями *ПРОЕКТЫ* и *ЗАКАЗЧИКИ* будет 1:n и суррогатный ПК станет внешним ключом для отношения *ПРОЕКТЫ*.

В отношении *СОТРУДНИКИ* атрибут *Оклад* зависит от атрибута *Должность*. Поступим с этой транзитивной зависимостью так же, как в предыдущем случае: создадим отношение *ДОЛЖНОСТИ*, перенесём в него атрибуты *Должность* и *Оклад*, а первичным ключом сделаем название должности.

В отношениях *СОТРУДНИКИ* и *ОБРАЗОВАНИЕ* атрибуты (*Дата выдачи* и *Кем выдан*) и (*Номер диплома* и *Год окончания учебного заведения*) зависят не от первичного ключа, а от атрибутов соответственно *Номер паспорта* и *Специальность*. Но если мы выделим их в отдельное отношение, то получим связи типа 1:1. Следовательно, здесь декомпозиция нецелесообразна.

4НФ. Отношение *АДРЕСА-ТЕЛЕФОНЫ* нарушают 4НФ, т.к. не всякий телефон привязан к конкретному адресу (т.е. мы имеем две многозначных зависимости в одном отношении). Но выделять *Телефоны* в отдельное отношение не стоит, т.к. эти сведения носят справочный характер и не требуется их автоматическая обработка.

Отношения, полученные после нормализации, приведены в табл. 6-15.

Таблица 6. Схема отношения *ОТДЕЛЫ* (Departs)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Аббревиатура отдела	D_ID	V(12)	**первичный ключ**
Название отдела	D_NAME	V(100)	обязательное поле

Таблица 7. Схема отношения *КОМНАТЫ* (Rooms)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Отдел	R_DEPART	V(12)	внешний ключ (к Departs)
Номер комнаты	R_ROOM	N(4)	**составной уникальный ключ**
Телефон	R_PHONE	V(20)	

Таблица 8. Схема отношения *ДОЛЖНОСТИ* (Posts)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Название должности	P_POST	V(30)	**первичный ключ**
Оклад	P_SAL	N(8,2)	обязательное поле, > 4500 руб.

Таблица 9. Схема отношения *СОТРУДНИКИ* (Employees)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Идентификатор сотрудника	E_ID	N(4)	**суррогатный первичный ключ**
Фамилия	E_FNAME	V(25)	обязательное поле
Имя, отчество	E_LNAME	V(30)	обязательное поле
Дата рождения	E_BORN	D	обязательное поле
Пол	E_SEX	C(1)	обязательное поле
Серия и номер паспорта	E_PASP	C(10)	обязательное уникальное поле
Когда выдан паспорт	E_DATE	D	обязательное поле
Кем выдан паспорт	E_GIVEN	V(50)	обязательное поле
ИНН	E_INN	C(12)	обязательное уникальное поле
Номер пенсионного страхового свидетельства	E_PENS	C(14)	обязательное уникальное поле
Отдел	E_DEPART	V(12)	внешний ключ (к Departs)
Должность	E_POST	V(30)	внешний ключ (к Posts)
Номер комнаты	E_ROOM	N(4)	составной внешний ключ (к Rooms)
Рабочий телефон	E_PHONE	V(20)	
Логин	E_LOGIN	V(30)	

Таблица 10. Схема отношения *ОБРАЗОВАНИЕ* (Edu)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Идентификатор сотрудника	U_ID	N(4)	внешний ключ (к Employees)
Вид образования	U_TYPE	V(20)	обязательное поле
Специальность	U_SPEC	V(40)	
Номер диплома	U_DIPLOM	V(15)	
Год окончания учебного заведения	U_YEAR	N(4)	обязательное поле

Таблица 11. Схема отношения *АДРЕСА-ТЕЛЕФОНЫ* (AdrTel)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Идентификатор сотрудника	A_ID	N(4)	внешний ключ (к Employees)
Адрес	A_ADDR	V(50)	
Телефон	A_PHONE	V(30)	

Таблицы ОБРАЗОВАНИЕ и АДРЕСА-ТЕЛЕФОНЫ не имеют потенциальных ключей, но мы не будем вводить суррогатные первичные ключи, т.к. на эти таблицы никто не ссылается.

Таблица 12. Схема отношения *ЗАКАЗЧИКИ* (Clients)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер заказчика	C_ID	N(4)	**суррогатный первичный ключ**
Заказчик	C_COMPANY	V(40)	обязательное поле
Адрес заказчика	C_ADR	V(50)	обязательное поле
Контактное лицо	C_PERSON	V(50)	обязательное поле
Телефон	C_PHONE	V(30)	

Таблица 13. Схема отношения *ПРОЕКТЫ* (Projects)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер проекта	P_ID	N(6)	обязательное уникальное поле
Название проекта	P_TITLE	V(100)	обязательное поле
Сокращённое название	P_ABBR	C(10)	**первичный ключ**
Отдел	P_DEPART	V(12)	внешний ключ (к Departs)
Заказчик	P_COMPANY	N(4)	внешний ключ (к Clients)
Руководитель	P_CHIEF	N(4)	внешний ключ (к Employees)
Дата начала проекта	P_BEGIN	D	обязательное поле
Дата окончания проекта	P_END	D	обязательное поле, больше даты начала проекта
Реальная дата окончания	P_FINISH	D	больше даты начала проекта
Стоимость проекта	P_COST	N(10)	обязательное поле, > 0

Таблица 14. Схема отношения *ЭТАПЫ ПРОЕКТА* (Stages)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*	
Проект	S_PRO	C(10)	внешний ключ (к Projects)	**составной первичный ключ**
Номер этапа	S_NUM	N(2)		
Название этапа	S_TITLE	V(200)	обязательное поле	
Дата начала этапа	S_BEGIN	D	обязательное поле	
Дата окончания этапа	S_END	D	обязательное поле, больше даты начала этапа	
Реальная дата окончания	S_FINISH	D	больше даты начала этапа	
Стоимость этапа	S_COST	N(10)	обязательное поле	
Полученная сумма по этапу	S_SUM	N(10)	обязательное поле, значение по умолчанию – 0	
Форма отчётности	S_FORM	V(100)	обязательное поле	

Таблица 15. Схема отношения *УЧАСТИЕ* (Job)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*	
Проект	J_PRO	C(10)	внешний ключ (к Projects)	**составной ПК**
Сотрудник	J_EMP	N(4)	внешний ключ (к Employees)	
Роль	J_ROLE	V(20)	обязательное поле	
Доплата	J_BONUS	N(2)		

Схема базы данных после нормализации приведена на рис. 7.

2.4.4. Определение дополнительных ограничений целостности

Перечислим ограничения целостности, которые не указаны в табл. 6–15.

1. Атрибут *Вид образования* может принимать одно из следующих значений: 'начальное', 'среднее', 'средне-специальное', 'высшее'.
2. Атрибут *Роль* может принимать одно из двух значений: 'исполнитель' или 'консультант'.
3. В поле *Доплата* хранится величина доплаты сотруднику за участие в проекте (в процентах к его окладу). Значение поля больше либо равно 0.
4. Нумерация в поле *Номер этапа* начинается с 1 и является непрерывной для каждого проекта.

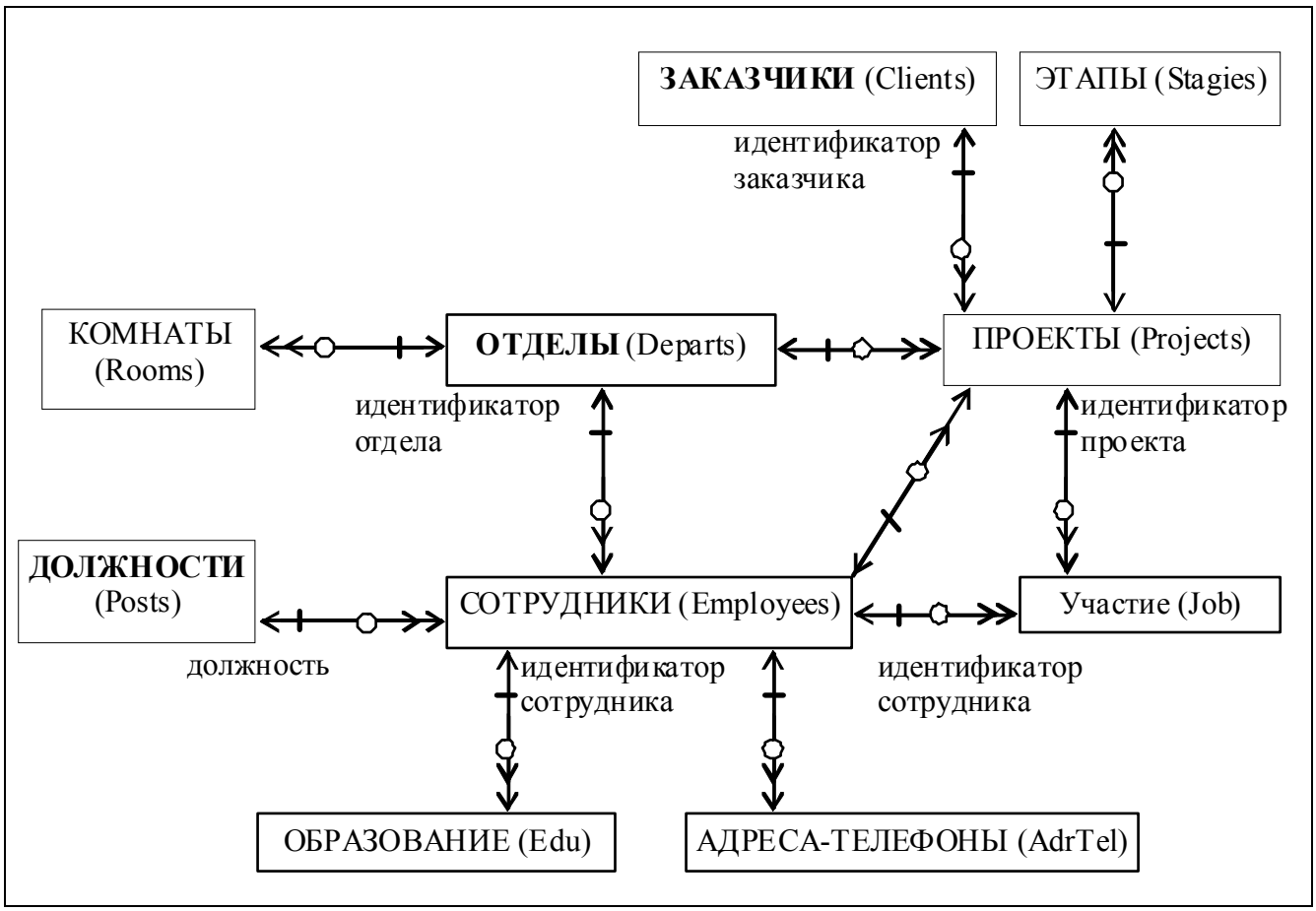

Рис. 7. Окончательная схема БД проектной организации

5. Дата начала первого этапа проекта должна соответствовать началу проекта в целом, дата завершения последнего этапа должна соответствовать завершению проекта в целом. Этапы не должны пересекаться по времени и между ними не должно быть разрывов.
6. Стоимость проекта должна быть равна сумме стоимостей всех этапов этого проекта.

Ограничения 4-6 нельзя реализовать в схеме отношения. В реальных БД подобные ограничения целостности реализуются вручную или программно (через внешнее приложение или специальную процедуру контроля данных – триггер).

Примечание. Вопросы архивирования данных в этом пособии подробно не рассматриваются. Но следует отметить, что обычно архив является частью БД и представляет собой набор отдельных таблиц, которые не связаны с оперативной частью БД внешними ключами. Структура архивных таблиц либо соответствует структуре тех оперативных таблиц, данные которых подлежат архивированию, либо представляет собой денормализованную таблицу, соответствующую декартову произведению оперативных таблиц. Данные в архивные таблицы переносятся специальной программой (или набором запросов) автоматически или по команде пользователя. По истечении периода хранения данные могут удаляться из архива.

2.4.5. Описание групп пользователей и прав доступа

Опишем для каждой группы пользователей права доступа к каждой таблице. Права доступа должны быть распределены так, чтобы для каждого объекта БД был хотя бы один пользователь, который имеет право добавлять и удалять данные из объекта. Права приведены в табл. 16. Используются следующие сокращения:

s – чтение данных (select);
i – добавление данных (insert);
u – модификация данных (update);
d – удаление данных(delete).

Таблица 16. Права доступа к таблицам для групп пользователей

Таблицы	Группы пользователей (роли)				
	Руководители организации	Сотрудники отд. кадров	Руководители проектов	Бухгалтеры	Участники проектов
Отделы	S	SIUD	S	S	
Комнаты	S	SUID	S	S	S
Должности	SIUD			S	
Сотрудники	S	SUID	S	S	
Адреса-телефоны	S	SUID	S	S	
Образование	S	SUID	S	S	
Заказчики	SIUD		S		
Проекты	SIUD		S		
Этапы проектов	SIUD		SUI		
Участие	S		S	S	

Права на изменение данных в таблице *УЧАСТИЕ* будут назначены через представление, т.к. изменять данные этой таблицы может только руководитель проекта. Описание представлений приведено в п.2.5.2. "Создание представлений (готовых запросов)".

Права назначает администратор БД (или администратор безопасности, если система сложная и администраторов несколько).

2.5. Реализация проекта базы данных

Мы условились не привязываться к конкретной СУБД и выполнять описание логической схемы БД на SQL-92. Приведём описание схемы БД на DDL.

2.5.1. Создание таблиц

1. Отношение Departs (отделы):

```
create table departs (
  d_id      varchar(12)   primary key,
  d_name    varchar(100)  not null);
```

2. Отношение Rooms (комнаты):

```
create table rooms (
  d_depart varchar(12)   references departs(d_id),
```

```
  r_room   numeric(4)    not null,
  r_phone  varchar(20),
  unique(r_room, r_phone));
```

3. Отношение Posts (должности):

```
create table posts (
  p_post   varchar(30)   primary key,
  p_salary numeric(8,2)  not null check(p_salary>=4500));
```

4. Отношение Employees (сотрудники):

```
create table employees (
  e_id     numeric(4)    primary key,
  e_fname  varchar(25)   not null,
  e_lname  varchar(30)   not null,
  e_born   date          not null,
  e_sex    char(1)       check(e_sex in ('ж','м')),
  e_pasp   char(10)      not null unique,
  e_date   date          not null,
  e_given  varchar(50)   not null,
  e_inn    char(12)      not null unique,
  e_pens   char(14)      not null unique,
  e_depart varchar(12)   references departs,
  e_post   varchar(30)   references posts,
  e_room   numeric(4)    not null,
  e_phone  varchar(20)   not null,
  e_login  varchar(30),
  foreign key(e_room,e_phone)
    references rooms(r_room,r_phone));
```

(Если внешний ключ ссылается на первичный ключ отношения, его можно не указывать, как в случае ссылок на Departs и Posts).

5. Отношение Edu (образование):

```
create table edu (
  u_id     numeric(4)    references employees,
  u_type   varchar(20)   not null,
  u_spec   varchar(40),
  u_diplom varchar(15),
  u_year   number(4) not null,
  check(u_spec in ('начальное', 'среднее', 'высшее',
      'средне-специальное')));
```

6. Отношение AdrTel (адреса-телефоны):

```
create table adrtel (
  a_id     numeric(4)    references employees,
  a_adr    varchar(50),
  a_phone  varchar(30));
```

7. Отношение Clients (заказчики):

```
create table clients (
  c_id     numeric(4)    primary key,
  c_company varchar(40)  not null,
  c_adr    varchar(50)   not null,
  c_person varchar(50)   not null,
```

```
  c_phone  varchar(30));
```

8. Отношение Projects (проекты):

```
create table projects (
  p_id     numeric(6)    not null unique,
  p_title  varchar(100)  not null,
  p_abbr   char(10)      primary key,
  p_depart varchar(12)   references departs,
  p_company numeric(4)   references clients,
  p_chief  numeric(4)    references employees,
  p_begin  date not null,
  p_end    date not null,
  p_finish date,
  p_cost   numeric(10) not null check(p_cost>0),
  check (p_end>p_begin),
  check (p_finish is null or p_finish>p_begin));
```

9. Отношение Stages (этапы проектов):

```
create table stages (
  s_pro    char(10)      references projects,
  s_num    numeric(2)    not null,
  s_title  varchar(200)  not null,
  s_begin  date          not null,
  s_end    date          not null,
  s_finish date,
  s_cost   numeric(10)   not null,
  s_sum    numeric(10)   not null,
  s_form   varchar(100)  not null,
  check (s_cost>0),
  check (s_end>s_begin),
  check (s_finish is null or s_finish>s_begin));
```

10. Отношение Job (участие):

```
create table job (
  j_pro    char(10)      references projects,
  j_emp    numeric(2)    references employees,
  j_role   varchar(20)   not null,
  j_bonus  numeric(2)    not null,
  check(j_bonus>0),
  check (j_role in ('исполнитель', 'консультант')));
```

2.5.2. Создание представлений (готовых запросов)

Приведём примеры нескольких готовых запросов (представлений):

1. Список всех текущих проектов (`sysdate` – функция, возвращающая текущую дату, определена в СУБД Oracle; в других системах аналогичная функция может называться по-другому, например, `getdate()` в Transact-SQL, `now()` в MS Access, `currdate()` в MySQL и т.д.):

```
 create view curr_projects as
    select *
      from projects
```

```
    where p_begin<=sysdate and sysdate<=p_end;
```

2. Определение суммы по текущим проектам, полученной на текущую дату:

```
create or replace view summ (title, cost, total) as
   select p_title, p_cost, sum(s_sum)
     from curr_projects, stages
    where p_abbr=s_pro
    group by p_title, p_cost;
```

3. Список сотрудников, участвующих в текущих проектах:

```
create view participants (project, name, role) as
   select p_abbr, e_fname||' '||e_lname, 'руководитель'
     from curr_projects, employees
    where p_chief=e_id
   union all
   select p_abbr, e_fname||' '||e_lname, j_role
     from curr_projects, employees, job
     where p_abbr=j_pro and e_id=j_emp
 order by 1, 3 desc;
```

4. Список рабочих телефонов сотрудников:

```
create or replace view worktel (name, room, phone) as
   select e_fname||' '||e_lname, e_room, e_phone
     from employees
     order by 1;
```

5. Форма отчётности и сроки выполнения этапов по текущим проектам:

```
create or replace view reports as
   select s_pro, s_num, s_title, s_begin, s_end, s_form
     from stages
     order by 1, 2;
```

6. Данные о проектах для руководителя проектов:

```
create or replace view my_projects as
   select *
     from projects p
     where exists (select * from employees e
           where e.e_id=p.p_chief and e.e_login=user);
```

   Функция `user` возвращает имя пользователя, выполняющего текущий запрос. Таким образом, каждый пользователь получит данные только о тех проектах, руководителем которых является. Используя аналогичный способ, можно ограничить участника проекта данными только о сотрудниках тех проектов, в которых он сам участвует.

7. Данные об этапах проектов для руководителя проектов:

```
create or replace view my_stages as
   select s.*
     from stages s
     where exists (select *
           from employees e, projects p
```

```
            where e.e_id=p.p_chief and e.e_login=user
            and s.s_pro=p.p_abbr);
```

8. Данные об участниках проектов для руководителя проектов:

```
create or replace view my_staff as
   select j.*
     from job j
     where exists (select *
           from employees e, projects p
           where e.e_id=p.p_chief and e.e_login=user
           and j.j_pro=p.p_abbr);
```

9. Данные о других участниках проекта:

```
create or replace view my_emps as
   select je.j_pro, e.e_fname||' '||e.e_lname e_name,
      e_depart, e_post, e_phone, e_room
     from employees e, job je
     where e.e_id=je.j_emp and exists (select *
           from job jm, employees m
           where m.e_id=jm.j_emp and
           m.e_login=user and je.j_pro=jm.j_pro);
```

Для того чтобы можно было работать с этими представлениями, соответствующим пользователям нужно назначить права доступа к представлениям. Эти права перечислены в табл. 17.

Таблица 17. Права доступа к представлениям

Представления	Группы пользователей (роли)		
	Руководители организации	Руководители проектов	Участники проектов
Текущие проекты (curr_projects)	S	S	
Сумма по текущим проектам (summ)	S	S	
Рабочие телефоны (worktel)	S	S	S
Участники проектов (participants)	S	S	S
Отчетность (reports)	S	S	S
Проекты для руководителя (my_projects)		SIUD	
Стадии проектов (my_stages)		SIUD	
Участники проектов для руководителей (my_staff)		SIUD	
Участники проектов (my_emps)			S

2.5.3. Назначение прав доступа

Права доступа пользователей предоставляются с помощью команды GRANT. Рассмотрим для примера права сотрудника компании ***ok_user***, который является сотрудником отдела кадров. Права доступа к отношениям Departs и Rooms могут быть описаны следующим образом:

```
grant select, insert, update, delete on departs to ok_user;
grant select, insert, update, delete on rooms to ok_user;
```

Права доступа руководителей проектов (сотрудников, `staff`) к представлению `my_projects` могут быть описаны следующим образом:

```
grant select, insert, update, delete on my_projects to staff;
```

Если сотрудник не является руководителем проекта, он не получит данных через этот запрос и не сможет воспользоваться правами доступа к нему.

Права доступа участников проекта (сотрудников, `staff`) к представлению `my_emps` могут быть описаны следующим образом:

```
grant select on my_emps to staff;
```

Если сотрудник не является участником проекта, он не получит данных через этот запрос и не сможет воспользоваться правами доступа к нему.

2.5.4. Создание индексов

Анализ готовых запросов показывает, что для повышения эффективности работы с данными необходимо создать индексы для всех внешних ключей. Приведём примеры создания индексов:

```
create index e_posts on employees(e_post);
create index p_chief on projects(p_chief);
create index e_tel on employees(e_room, e_phone);
```

2.5.5. Разработка стратегии резервного копирования

Интенсивность обновления разработанной базы данных низкая, поэтому для обеспечения сохранности вполне достаточно проводить полное резервное копирование БД раз в день (перед окончанием рабочего дня). Для разработанной БД нет необходимости держать сервер включенным круглосуточно, поэтому можно создать соответствующее задание операционной системы, которое будет автоматически запускаться перед выключением сервера.

3. ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТА

Курсовой проект выполняется по одному из вариантов, приведённых в следующем разделе, или для произвольной предметной области (по согласованию с преподавателем).

Пояснительная записка должна отражать ход выполнения курсового проекта (аналогично разобранному выше примеру). К пояснительной записке прилагаются распечатка программного текста и руководство пользователя.

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.

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

4. ВАРИАНТЫ ЗАДАНИЙ НА КУРСОВОЕ ПРОЕКТИРОВАНИЕ

1. БД книг из домашней библиотеки.
2. БД для домашней видеотеки (БД кинофильмов).
3. БД домашней фонотеки (диски с музыкальными произведениями).
4. БД "Расписание занятий в школе".
5. БД по прокату автомобилей.
6. Городская БД собственников жилья.
7. Городская БД собственников автомобилей.
8. БД страховой компании.
9. БД аптеки.
10. БД жилищно-эксплуатационной компании.
11. БД кинологического клуба.
12. Разработать (найти) и реализовать в виде БД классификацию (одну из предложенных далее):
    - СУБД;
    - интернет-провайдеров;
    - систем контроля знаний;
    - систем искусственного интеллекта;
    - систем поддержки принятия решений;
    - мобильных телефонов;
    - автомобилей;
    - самолётов (вертолётов);
    - садовых растений;
    - лекарственных препаратов;
    - видов спорта;
    - профессий;
    - природных ресурсов;
    - управленческих решений.

Библиографический список

1. Карпова И.П. Базы данных: Учебное пособие по курсу "Базы данных". – М., РИО МГИЭМ, 2009. – 118 с.
2. Коннолли Т., Бегг К. Базы данных: проектирование, реализация, сопровождение. Теория и практика. – 3-е изд.: Пер. с англ.: Уч. пос. – М.: Изд. дом "Вильямс", 2003. – 1440 с.
3. Тиори Т., Фрай Дж. Проектирование структур баз данных: В 2-х кн.: Пер. с англ. – М.: Мир, 1985.
4. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – 2-е изд. – М.: Финансы и статистика, 1989. – 350 с.
5. Грабер М. Введение в SQL. – М.: Лори, 2008. – 378 с.

Учебное издание

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Составитель КАРПОВА Ирина Петровна

Редактор Е.С. Резникова
Технический редактор О.Г. Завьялова

Подписано в печать . Формат 60×84/16.
Бумага офсетная № 2. Ризография. Усл. печ. л.2,0. Уч.-изд.л.1,8.
Изд. № . Тираж 100 экз. Заказ .
Московский государственный институт электроники и математики.
109028, Москва, Б. Трехсвятительский пер., 3.
Отдел оперативной полиграфии
Московского государственного института электроники и математики.
113054, Москва, ул. М. Пионерская, 12.