Федеральное государственное автономное образовательное учреждение
высшего образования
"Национальный исследовательский университет
"Высшая школа экономики"

Московский институт электроники и математики им. А.Н Тихонова

Департамент компьютерной инженерии

ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Методические указания
по выполнению домашнего задания
по курсу "Базы данных"

Москва
2020

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

УДК 681.3

Проектирование реляционной базы данных: Метод. указания по выполнению домашнего задания по курсу "Базы данных" / Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ; Сост.: И.П. Карпова. – М., 2020. – 33 с.

Домашнее задание посвящено изучению методов проектирования реляционных баз данных, особое внимание уделено этапам инфологического, логического и физического проектирования.

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

Табл. 22. Ил. 7. Библиогр.: 6 назв.

ISBN 5-230-16273-2

СОДЕРЖАНИЕ

Цели работы

Цель домашнего задания – 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 реляционной базе данных, состоящей из 3-4 таблиц). Поэтому, если вас интересует результат, при создании базы данных необходимо придерживаться правил технологии проектирования БД.

Опишем вкратце процесс проектирования реляционной базы данных. (Более подробно этот процесс изложен в [2, 3]).

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 чаще всего при проектировании баз данных используется метод сущность–

связь, и модель 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 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 технический 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), 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 кода создания основных объектов базы данных на языке SQL в синтаксисе выбранной СУБД (пользователи, таблицы и др.).
13. Написание кода 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.

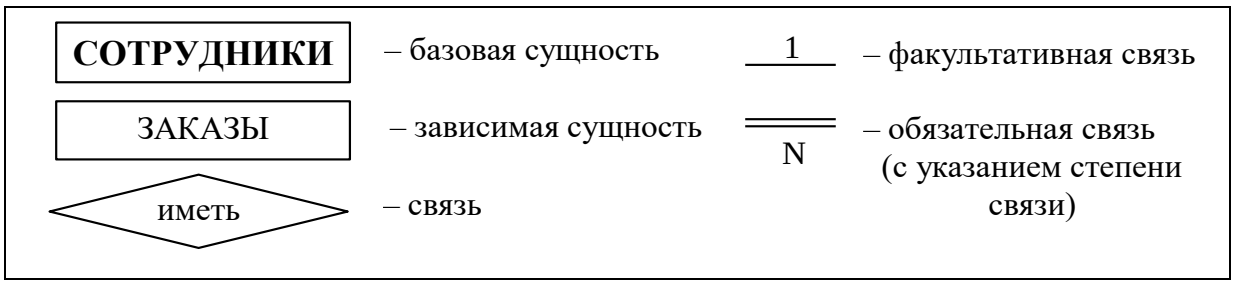

Рис. 1. Обозначения, используемые в ER-диаграммах

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 по атрибутам.

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 ($X \to Y$). Атрибут X в функциональной зависимости $X \to Y$ 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 $X \to Y$ и $Y \to Z$,

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 первичного 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.

Формальное описание нормализации достаточно подробно изложено в [1], поэтому, чтобы не повторяться, будем рассматривать этот метод на примере проектирования конкретной базы данных. Нормализацию будем проводить в **два этапа**:

I. 1) Разобьем составные атрибуты на простые. 2) Вынесем многозначные атрибуты в отдельные отношения. II. Найдем транзитивные зависимости и вынесем их в отдельные отношения.

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 и указать их атрибуты. **Идентифицирующие** атрибуты мы выделим полужирным шрифтом, *многозначные* – курсивом, составные подчеркнем:

1) **Отделы**. Атрибуты: **название**, **аббревиатура**, *комнаты*, *телефоны*.
2) **Сотрудники**. Атрибуты: ФИО, **паспортные данные**, дата рождения, пол, **ИНН** (индивидуальный номер налогоплательщика), **СНИЛС** (страховой номер индивидуального лицевого счета), *адреса*, *телефоны* (мобильный, рабочий, домашний), *данные об образовании* (вид образования (высшее, средне-специальное и т.д.), специальность, номер диплома, дата окончания учебного заведения), должность, оклад, **логин** (имя пользователя).

   Каждый сотрудник работает в определенном отделе, может руководить проектами и участвовать в их выполнении.

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 переводы сотрудника с одной должности на другую, уходы в отпуска, больничные 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.

   За каждый проект отвечает определенный отдел, у каждого проекта есть руководитель из числа сотрудников этого отдела.

4) **Этапы проекта**. Атрибуты: номер по порядку, название, дата начала этапа, дата завершения этапа, *форма отчетности* (перечень технической документации), сумма по этапу, дата реальной сдачи этапа; сумма, полученная по этапу на текущую дату.

   Каждый этап относится к определенному проекту.

Обратите внимание! У сущности нет атрибутов других сущностей, это реализуется через связь. Связи как устойчивые ассоциации между сущностями являются неотъемлемой частью предметной области. Просто отражаются они не с помощью атрибутов сущностей, а с помощью специальных элементов – **связей**. Например, каждый сотрудник работает в определенном отделе, но это не значит, что номер отдела является его личной характеристикой: это его связь с сущностью *Отделы*, которая, кстати, может измениться – он может перейти в другой отдел. Метод сущность-связь не зависит от модели данных, на основе ER-диаграммы можно спроектировать не только реляционную БД, но и любую другую. Но только в реляционной базе данных связи реализуются с помощью отдельных полей таблицы – так называемых внешних ключей. А в сетевой модели данных или в объектно-ориентированной, например, связь – это физическая ссылка от одной записи к другой, и никаких "чужих" атрибутов в этих записях нет.

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

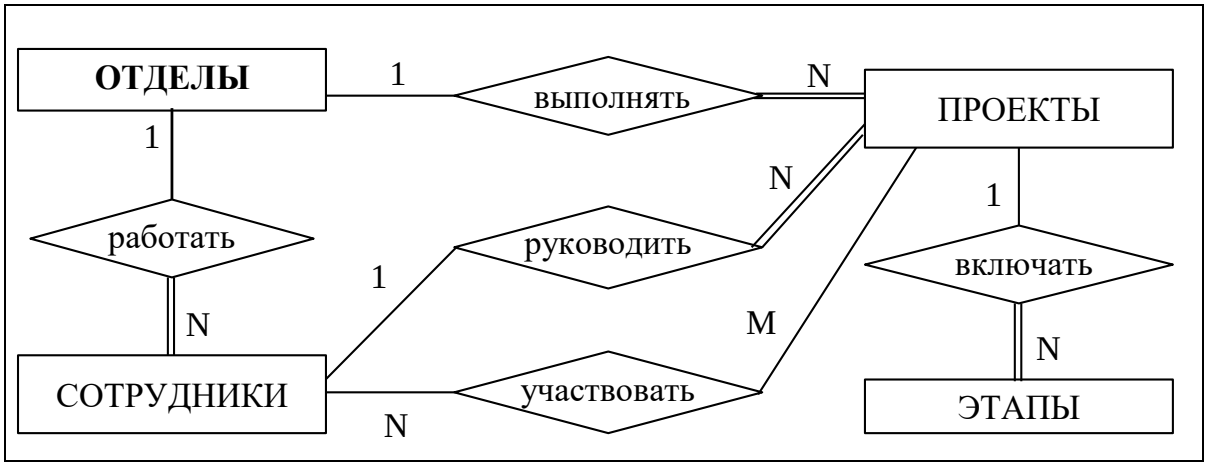

Рис. 2. ER–диаграмма ПрО «Проектная организация»

2.1.2. Анализ информационных задач и круга пользователей системы

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

1. Руководитель организации:
   - заключение новых договоров;
   - назначение руководителей проектов;
   - получение списка всех участников проектов;
   - изменение должностных окладов;
   - получение полной информации о проектах;
   - внесение изменений в данные о проектах.
2. Заместитель руководителя организации:
   - изменение штатного расписания;
   - проверка и исправление данных о заключенных договорах;
   - формирование и актуализация справочных таблиц.
3. 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.
4. Сотрудники отдела кадров:
   - приём/увольнение сотрудников;
   - внесение изменений в данные о сотрудниках.
5. Бухгалтеры:
   - получение ведомости на выплату зарплаты.
6. 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);
- объем технической документации составляет 100М на каждый проект.

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

$$M_д = 2(10*1+10*4*0,5+100*0,5+(10*10*0,2)+10*20000) \approx 1\ G,$$

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 невелик, 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 (PostgreSQL, 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.

2.4. Логическое проектирование реляционной БД

2.4.1. Преобразование ER–диаграммы в схему базы данных

База данных создаётся на основании схемы базы данных. Преобразование ER–диаграммы в схему БД выполняется путем сопоставления каждой сущности и каждой связи, имеющей атрибуты, отношения (таблицы) БД. Связь типа 1:n (один-ко-многим) между отношениями реализуется через внешний ключ. Ключ вводится для дочернего отношения. Внешнему ключу должен соответствовать первичный или уникальный ключ основного (родительского) отношения.

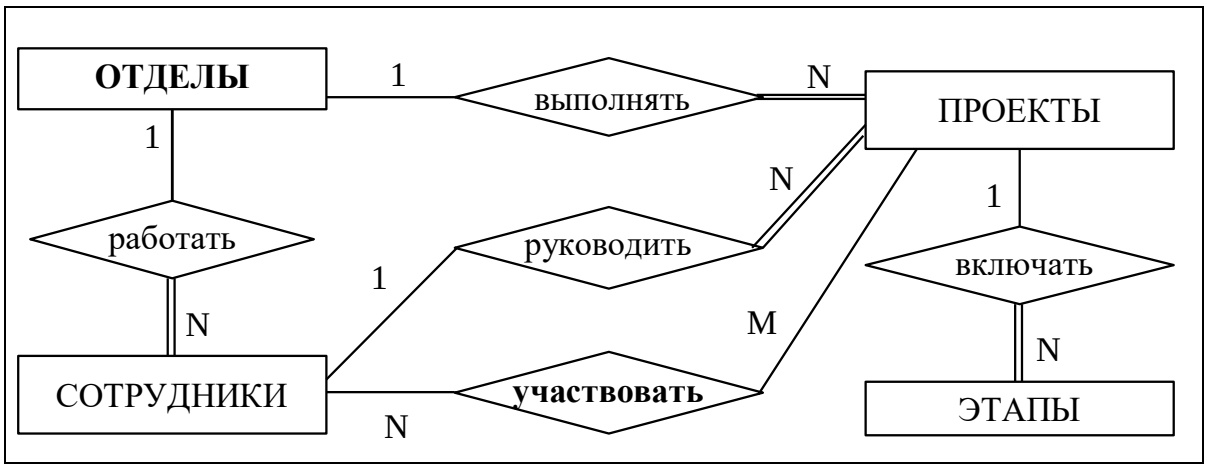

Рис. 3. ER–диаграмма ПрО «Проектная организация»

Связь *участвовать* между *ПРОЕКТАМИ* и *СОТРУДНИКАМИ* принадлежит к типу N:M (Рис. 3). 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.

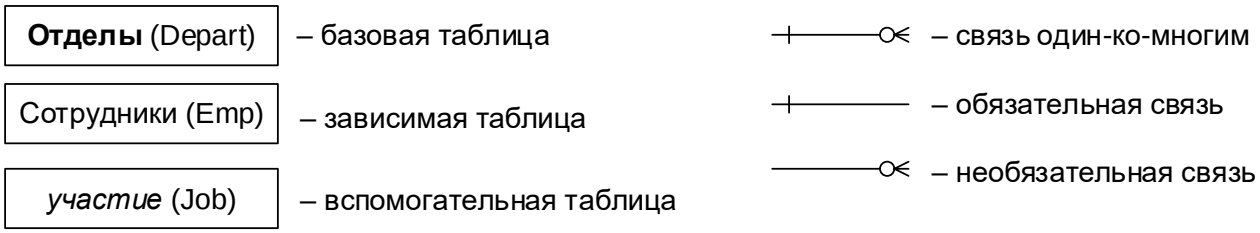

Рис. 4. Обозначения, используемые на схеме базы данных

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

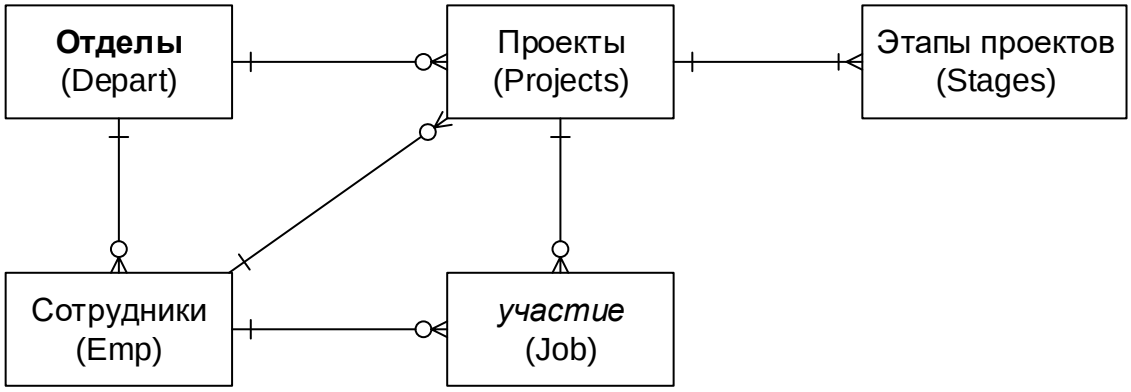

Рис. 5. Схема РБД, полученная из ER–диаграммы проектной организации

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 ПРОЕКТОВ*, 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 (Рис. 6). Кроме того, сотрудник может принимать участие не во всем проекте, а в его отдельных этапах; поэтому мы изменим связь между таблицами *ПРОЕКТЫ* и *УЧАСТИЕ* на связь между таблицами *ЭТАПЫ ПРОЕКТОВ* и *УЧАСТИЕ*.

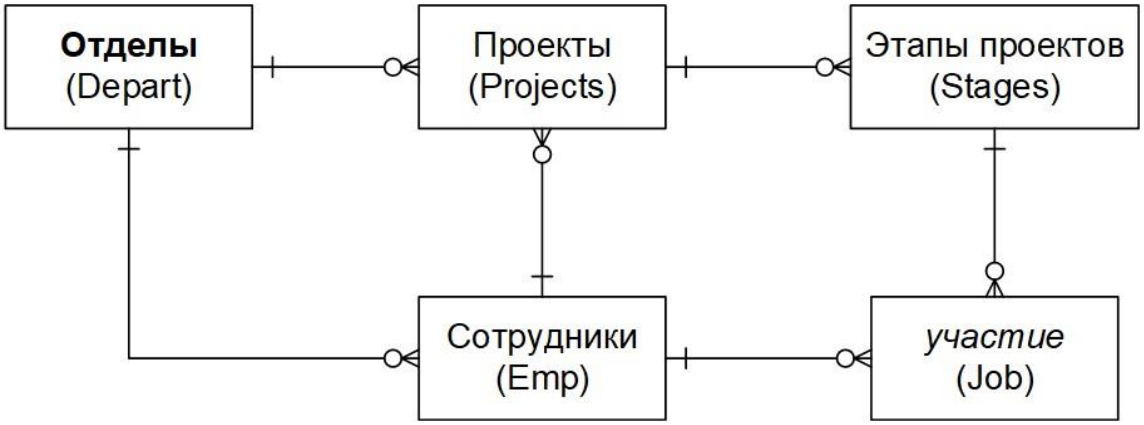

Рис. 6. Схема РБД без нереализуемых связей

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 **Таблица 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 *СНИЛС*. 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_GENDER	C(1)	обязательное поле, 'м' или 'ж'
Паспортные данные	E_PASP	V(50)	обязательное поле
ИНН	E_INN	C(12)	обязательное уникальное поле
СНИЛС	E_SNILS	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 (искусственный) 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(300)	обязательное многозначное поле	

В состав формы отчётности входит техническая документация, которая занимает большой объём памяти. Её можно хранить в базе данных (типы LONG, BLOB), но мы будем хранить только ссылки на файлы с документами.

Таблица 5. Схема отношения *УЧАСТИЕ* (Job)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Проект	J_PRO	C(10)	составной внешний ключ (к Stages)
Номер этапа	J_NUM	N(2)	
Сотрудник	J_EMP	N(4)	внешний ключ (к Employees)
Роль	J_ROLE	V(20)	'консультант' или 'исполнитель'
Доплата	J_BONUS	N(2)	>=0, по умолчанию 0
Начало участия	J_BEGIN	D	обязательное поле
Окончание участия	J_END	D	

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.

Для приведения таблиц к **1НФ** разобьем сложные (составные) атрибуты на простые и вынесем многозначные атрибуты в отдельные отношения.

Примечание. В реальных БД составные атрибуты разбиваются на простые в следующих случаях:

а) необходимо установить ограничения целостности на отдельные части составного атрибута;

б) этого требует внешнее представление данных;

б) в запросах поиск может осуществляться по отдельной части атрибута.

***ОТДЕЛЫ*.** Многозначные атрибуты *Комнаты* и *Телефоны* вынесем в отдельное отношение *КОМНАТЫ*. Так как в комнате может не быть стационарно-

го телефона, первичный ключ отношения *КОМНАТЫ* не определен (ПК не может содержать null-значения), но на этих атрибутах можно определить составной уникальный ключ. После нормализации отношение *ОТДЕЛЫ* превратится в два отношения, которые приведены в Таблица 6-7.

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)	

СОТРУДНИКИ. Разделим составной атрибут *Фамилия, имя, отчество* на два атрибута *Фамилия* и *Имя, отчество*, составной атрибут *Паспортные данные* на *Серия и номер паспорта* (уникальный), *Дата выдачи* и *Кем выдан*. Серию и номер паспорта не имеет смысла хранить в разных полях, т.к. оба они числовые, и система тратит меньше времени на поддержание уникальности одного поля, а не комбинации двух полей.

Составной многозначный атрибут *Данные об образовании* вынесем в отдельное отношение и разделим на *Вид образования*, *Специальность*, *Номер диплома* и *Год окончания учебного заведения*.

Домашние и мобильные телефоны и адреса сотрудников вынесем в отношение *АДРЕСА-ТЕЛЕФОНЫ*. В отношении *АДРЕСА-ТЕЛЕФОНЫ* нет потенциальных ключей: оставим это отношение без первичного ключа, т.к. на него никто не ссылается. Что касается рабочих телефонов сотрудников, то один из этих номеров – основной – определяется рабочим местом сотрудника (мы учитываем 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 можно было ссылаться на отношение *КОМНАТЫ*, а 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*).

Для приведения отношения ***СОТРУДНИКИ*** к **3НФ** найдем транзитивные зависимости. Атрибут *Оклад* зависит от атрибута *Должность*. Создадим отношение *ДОЛЖНОСТИ*, перенесём в него атрибуты *Должность* и *Оклад*, а первичным ключом сделаем название должности (Таблица 8).

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, > 16000 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_GENDER	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
СНИЛС	E_SNILS	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)	

* – серию и номер паспорта храним в символьном поле, т.к. серия может начинаться с 0, а в числовом поле ведущий ноль не хранится и не будет отображаться при выводе. Для проверки того, что поле содержит только цифры, в Oracle, например, можно использовать check(to_number(e_pasp)>100000000).

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 отдельные отношения, то получим связи типа 1:1. Следовательно, эта декомпозиция нецелесообразна.

Также мы создадим справочную таблицу ***ВИДЫ ОБРАЗОВАНИЯ***, Эта таблица обеспечит одинаковое написание значений соответствующего поля таблицы ***ОБРАЗОВАНИЕ***, и возможность добавлять и изменять их при необходимости. Отношения, получившиеся при вынесении многозначных полей из отношения ***СОТРУДНИКИ***, приведены в Таблица 10-12.

Таблица 10. Схема отношения *ВИДЫ ОБРАЗОВАНИЯ* (Types)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Название вида образования	T_NAME	V(20)	**первичный ключ**

Таблица 11. 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, внешний ключ (к Types)
Специальность	U_SPEC	V(40)	
Номер диплома	U_DIPLOM	V(15)	
Год окончания учебного заведения	U_YEAR	N(4)	обязательное поле

Таблица 12. 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)	

Отношение *АДРЕСА-ТЕЛЕФОНЫ* нарушает **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 *ОБРАЗОВАНИЕ* и *АДРЕСА-ТЕЛЕФОНЫ* не имеют потенциальных ключей, но мы не будем вводить суррогатные первичные ключи, т.к. на эти таблицы никто не ссылается. Но для таблицы АДРЕСА-ТЕЛЕФОНЫ введем ограничение, в соответствии с которым или адрес, или телефон должен быть указан.

ПРОЕКТЫ. Удалим вычислимый атрибут *Полученная сумма*, т.к. он является вычислимым – это сумма значений аналогичного атрибута из отношения ***ЭТАПЫ ПРОЕКТОВ***. Но атрибут *Стоимость проекта* оставим, т.к. она фигурирует в документации по проекту. Для обеспечения логической целостности данных необходимо предусмотреть в приложении проверку того, что сумма стоимостей по всем этапам совпадает с общей стоимостью проекта. Атрибут *Данные заказчика* зависит от атрибута *Заказчик*, а не от первичного ключа, поэтому их следует вынести в отдельное отношение *ЗАКАЗЧИКИ*. Но при этом первичным ключом нового отношения станет атрибут *Заказчик*, т.е. длинная символьная строка. Целесообразнее ввести 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*** (Таблица 13-14).

Таблица 13. 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 14. 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

ЭТАПЫ ПРОЕКТОВ. Многозначный составной атрибут *Формы отчетности* вынесем в отдельное отношение *ОТЧЁТЫ* и разделим на поля *Номер ГОСТа*, *Название отчёта*, *Ссылка на файл отчёта*.

В отношении ***ОТЧЁТЫ*** атрибут *Название отчёта* зависит от *Номера ГОСТа*, поэтому вы создадим справочную таблицу *ФОРМЫ ОТЧЕТНОСТИ*, в которой *Номер ГОСТа* будет первичным ключом.

Таблица 15. Схема отношения *ФОРМЫ ОТЧЕТНОСТИ* (Forms)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер ГОСТа	F_GOST	V(20)	**первичный ключ**
Название	F_TITLE	V(150)	обязательное поле
Ссылка на файл с ГОСТом	F_FILE	V(250)	обязательное поле

Таблица 16. 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	

Таблица 17. Схема отношения *ОТЧЁТЫ* (Reports)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Проект	R_PRO	C(10)	составной внешний ключ (к Stages)
Номер этапа	R_NUM	N(2)	
Форма отчётности	R_FORM	V(20)	внешний ключ к Forms
Ссылка на файл с отчётом	R_FILE	V(250)	
Дата создания отчёта	R_DATE	D	

Таблица 18. Схема отношения *РОЛИ* (Roles)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Название роли участника	R_NAME	V(20)	**первичный ключ**

Таблица 19. Схема отношения *УЧАСТИЕ* (Job)

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Проект	J_PRO	C(10)	составной внешний ключ (к Stages)
Номер этапа	J_NUM	N(2)	
Сотрудник	J_EMP	N(4)	внешний ключ (к Employees)
Роль	J_ROLE	V(20)	внешний ключ (к Roles)

Начало участия	J_BEGIN	D	обязательное поле
Окончание участия	J_END	D	
Доплата	J_BONUS	N(2)	обязательное поле, по умолчанию 0

Схема базы данных после нормализации приведена на Рис. 7.

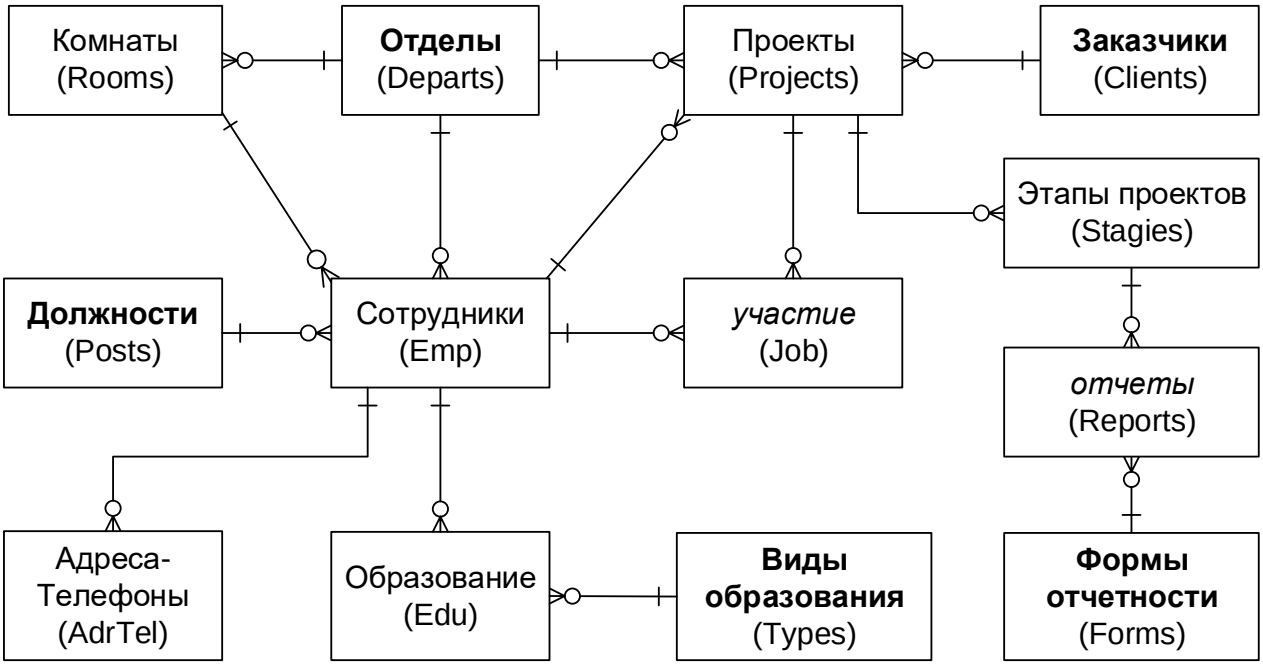

Рис. 7. Окончательная схема БД проектной организации

2.4.4. Определение дополнительных ограничений целостности

Перечислим ограничения целостности, которые не указаны в табл. 6–19.

1. В поле *Доплата* хранится величина доплаты сотруднику за участие в проекте (в процентах к его окладу). Значение поля больше либо равно 0.
2. Нумерация в поле *Номер этапа* начинается с 1 и является непрерывной для каждого проекта.
3. 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.
4. Стоимость проекта должна быть равна сумме стоимостей всех этапов этого проекта. Вычислимые поля обычно не хранятся в БД. Но для проектов общая стоимость является базовым атрибутом, а стоимости этапов – зависимым, поэтому мы оставим это поле.
5. Периоды участия в проекте не должны выходить за границы периодов выполнения этапов проекта. Например, если сотрудник участвует в 1-м этапе проекта, то дата начала участия должна быть не меньше, чем дата начала 1-го этапа, а дата окончания – не больше, чем дата завершения этого этапа.

Ограничения 2-5 нельзя реализовать в схеме отношения. В реальных БД подобные ограничения целостности реализуются вручную или программно (через внешнее приложение или специальную процедуру контроля данных – триггер).

2.4.5. Описание групп пользователей и прав доступа

Опишем для каждой группы пользователей права доступа к каждой таблице (Таблица 21). 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. Используются следующие сокращения:

Таблица 20. Обозначения прав доступа

Обозначение	Расшифровка	Альтернативное обозначение
I (insert)	добавление данных	C (create)
S (select)	чтение данных	R (read)
U (update)	модификация данных	U (update)
D (delete)	удаление данных	D (delete)

Права руководителей проектов на изменение данных в таблицах *ЭТАПЫ ПРОЕКТОВ*, *УЧАСТИЕ* и *ОТЧЁТЫ* будут назначены через представления, т.к. руководитель проекта может изменять данные только о своих проектах. 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).

Таблица 21. Права доступа к таблицам для групп пользователей

Таблицы	Группы пользователей (роли)					
	Руководитель организации	Зам. руководителя	Сотрудники отд. кадров	Руководители проектов	Бухгалтеры	Участники проектов
Отделы	S	SUID	SUI	S	S	S
Комнаты	S	S	SUID	S	S	S
Должности	SUID	SU	S		S	
Сотрудники	S	S	SUID	S	S	
Адреса-телефоны	S	S	SUID	S	S	
Виды образования	S	SIU	SUI	S		
Образование	S	SU	SUID	S		
Заказчики	SUID	SIU		S		
Проекты	SUID	SUI		SU	S	
Формы отчётности	SUID	SUI		SU		
Этапы проектов	SUID	SUI		S	S	
Отчёты	SUID	SU		S		
Роли	SUID	SUI		S		
Участие	S	S		S	S	

2.5. Реализация проекта базы данных

Мы условились не привязываться к конкретной СУБД и выполнять описание логической схемы БД на SQL-92 [5]. Приведём код создания БД.

2.5.1. Создание таблиц

Сначала создаются справочные и базовые таблицы, затем – подчиненные.

1. Отношение Types (виды образования):

```
create table types (
  t_name   varchar(20)   primary key);
```

2. Отношение Roles (роли участников):

```
create table roles (
  r_name   varchar(20)   primary key);
```

3. Отношение Forms (формы отчетности):

```
create table forms (
  f_gost   varchar(20)   primary key,
  f_title  varchar(150)  not null,
  f_file   varchar(250));
```

4. Отношение Posts (должности):

```
create table posts (
  p_post   varchar(30)   primary key,
  p_salary numeric(8,2)  not null,
  constraint check_salary check(p_salary>=16000));
```

5. Отношение Departs (отделы):

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

6. Отношение Rooms (комнаты):

```
create table rooms (
  d_depart varchar(12)   not null
  constraint fk_rooms_departs references departs(d_id),
  r_room   numeric(4)    not null,
  r_phone  varchar(20),
  constraint uniq_room_phone unique(r_room, r_phone));
```

7. 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          not null
       constraint check_sex check(e_sex in ('ж','м')),
  e_pasp   char(10)      not null
       constraint uniq_pasp unique
       constraint check_pasp check(to_number(e_pasp)>100000000),
  e_date   date          not null,
  e_given  varchar(50)   not null,
  e_inn    char(12)      not null
       constraint uniq_inn unique,
  e_snils  char(14)      not null
       constraint uniq_snils unique,
  e_depart varchar(12)   not null
       constraint fk_emp_departs references departs,
  e_post   varchar(30)   not null
```

```
      constraint fk_emp_posts references posts,
    e_room   numeric(4)    not null,
    e_phone  varchar(20),
    e_login  varchar(30),
    constraint fk_emp_rooms 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).

8. Отношение Edu (образование):

```
create table edu (
  u_id     numeric(4)    not null
      constraint fk_edu_emp references employees,
  u_type   varchar(20)   not null
      constraint fk_edu_types references types,
  u_spec   varchar(40),
  u_diplom varchar(15),
  u_year   numeric(4)    not null);
```

9. Отношение AdrTel (адреса-телефоны):

```
create table adrtel (
  a_id     numeric(4)    not null
      constraint fk_adrtel_emp references employees,
  a_adr    varchar(100),
  a_phone  varchar(20),
  constraint check_adrtel check(a_adr is not null
      or a_phone is not null));
```

10. 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));
```

11. Отношение Projects (проекты):

```
create table projects (
  p_id     numeric(6)    not null
      constraint uniq_p_id unique,
  p_title  varchar(100)  not null,
  p_abbr   char(10)      primary key,
  p_depart varchar(12)   not null
      constraint fk_pro_departs references departs,
  p_company numeric(4)   not null
      constraint fk_pro_clients references clients,
  p_chief  numeric(4)    not null
      constraint fk_pro_emp references employees,
  p_begin  date          not null,
  p_end    date          not null,
  p_finish date,
```

```
  p_cost   numeric(10)   not null,
  constraint check_pro_cost check(p_cost>0),
  constraint check_pro_end_begin check (p_end>p_begin),
  constraint check_pro_finish check (p_finish is null
      or p_finish>p_begin));
```

12.Отношение Stages (этапы проектов):

```
create table stages (
  s_pro    char(10)      not null
      constraint fk_stages_projects references projects,
  s_num    numeric(2),
  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,
  primary key(s_pro, s_num),
  constraint check_stages_cost check(s_cost>0),
  constraint check_stages_end_begin check(s_end>s_begin),
  constraint check_stages_finish check(s_finish is null
      or s_finish>s_begin));
```

13.Отношение Reports (отчеты):

```
create table reports (
  r_pro    char(10)      not null,
  r_num    numeric(2)    not null,
  r_form   varchar(100)  not null
  constraint fk_reports_forms references forms,
  r_file   varchar(250),
  r_date   date,
  constraint fk_reports_stages foreign key(r_pro, r_num)
      references stages);
```

14.Отношение Job (участие):

```
create table job (
  j_pro    char(10)      not null,
  j_num    numeric(2)    not null,
  j_emp    numeric(4)    not null
      constraint fk_job_emp references employees,
  j_role   varchar(20)   not null
      constraint fk_job_roles references roles,
  j_begin  date          not null,
  j_end    date,
  j_bonus  number(2)     default 0 not null,
  constraint fk_job_stagse foreign key(j_pro, j_num)
      references stages,
  constraint check_job_end_begin check (j_end>j_begin),
  constraint check_job_bonus check(j_bonus>=0));
```

Начальный состав данных для **справочных таблиц** (обратите внимание – только для справочных таблиц. Содержание других таблиц не входит в проект БД и будет вводиться по мере поступления данных):

1) Таблица *Виды образования*:
```
insert into Types values('начальное');
insert into Types values('среднее');
insert into Types values('средне-специальное');
insert into Types values('высшее');
```
2) Таблица *Роли*:
```
insert into Roles values ('исполнитель');
insert into Roles values ('консультант');
```
3) Таблица *Формы Отчетности* [6]:
```
insert into Forms(f_gost, f_title) values('ГОСТ 19.005-85', 'ЕСПД. Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения');
insert into Forms(f_gost, f_title) values('ГОСТ 19.201-78', 'ЕСПД. Техническое задание. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.202-78', 'ЕСПД. Спецификация. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.301-79', 'ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.401-78', 'ЕСПД. Текст программы. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.402-78', 'ЕСПД. Описание программы');
insert into Forms(f_gost, f_title) values('ГОСТ 19.404-79', 'ЕСПД. Пояснительная записка. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.501-78', 'ЕСПД. Формуляр. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.502-78', 'ЕСПД. Описание применения. Требования к содержанию и оформлению');
insert into Forms values('ГОСТ 19.503-79', 'ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.504-79', 'ЕСПД. Руководство программиста. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.505-79', 'ЕСПД. Руководство оператора. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.508-79', 'ЕСПД. Руководство по техническому обслуживанию. Требования к содержанию и оформлению');
insert into Forms(f_gost, f_title) values('ГОСТ 19.701-90', 'ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения');
```

2.5.2. Создание представлений (готовых запросов)

Представления являются хранимыми запросами и не содержат данных. Они необходимы для того, чтобы пользователь мог быстро извлечь данные в удобном для себя виде. В ходе анализа ПрО была определена потребность в наличии следующих представлений:

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

```
create view curr_projects as
   select *
     from projects
    where current_date between p_begin and p_end;
```

2. Список всех текущих этапов проектов:

```
create view curr_stages as
   select *
     from stages
    where current_date between s_begin and
      COALESCE(s_end, current_date);
```

Функция `COALESCE()` возвращает первый аргумент, отличный от null. В данном случае функция вернет текущую дату, если дата завершения участия сотрудника в проекте не определена.

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

```
create or replace view summa (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;
```

4. 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 j_pro, e_fname||' '||e_lname, j_role
     from employees, job
     where e_id=j_emp and current_date between j_begin
     and COALESCE(j_end, current_date)
    order by 1, 3 desc;
```

5. 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;
```

6. Форма отчётности и сроки выполнения текущих этапов по проектам:

```
create or replace view curr_reports as
   select s_pro, s_num, s_title, s_begin, s_end, r_form,
          r_file, r_date
     from stages, reports
     where s_pro=r_pro and s_num=r_num
          and current_date between s_begin
```

```
                and COALESCE(s_finish, s_end)
     order by 1, 2;
```

7. Данные о проектах (будут доступны только руководителям 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)
      with grant option;
```

Функция **user** возвращает имя пользователя, выполняющего текущий запрос. Таким образом, **права доступа к этому представлению можно дать всем пользователям (PUBLIC)**, но 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.

8. Данные о текущих этапах проектов (доступны руководителям проектов):

```
create or replace view my_stages as
   select s.*
     from curr_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) with grant option;
```

9. Данные об участниках текущих проектов:

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

Обратите внимание: таблицы *УЧАСТИЕ* и *ПРОЕКТЫ* не связаны напрямую внешним ключом; но наличие в таблице *УЧАСТИЕ* поля "Аббревиатура проекта" (j_pro) позволяет соединять эти таблицы непосредственно, без обращения к таблице *ЭТАПЫ ПРОЕКТА*.

10. Данные о других участниках проекта (доступны участникам проектов):

```
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 работы с этими представлениями соответствующим пользователям нужно определить права доступа к представлениям (Таблица 22). Представления 7-9 являются обновляемыми, поэтому для них указывается `with grant option` и можно пердоставить права на их редактирование.

Таблица 22. Права доступа к представлениям

Представления	Группы пользователей (роли)		
	Руководители организации	Руководители проектов	Все сотрудники (PUBLIC)
Текущие проекты (curr_projects)	S	S	
Текущие этапы (curr_stages)	S	S	
Сумма по текущим проектам (summa)	S	S	
Рабочие телефоны (worktel)			S
Участники проектов (participants)			S
Отчетность (curr_reports)			S
Проекты для руководителя (my_projects)			SIUD
Стадии проектов (my_stages)			SIUD
Участники проектов (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 отдела кадров (роль ***ok_user***). Права доступа к отношениям Departs и Rooms могут быть описаны следующим образом:

```
grant ALL on departs to ok_user;
grant ALL on rooms to ok_user;
```

Права доступа руководителей проектов (роль `staff`) к представлению `summa` могут быть описаны следующим образом:

```
grant select on summa to staff;
```

Права доступа всех сотрудников к представлению `my_projects` могут быть описаны следующим образом:

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

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

Права доступа участников проекта к представлению `my_emps` могут быть описаны следующим образом:

```
grant select on my_emps to PUBLIC;
```

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

Примечание: в проекте БД необходимо привести полный перечень команд создания пользователей (ролей) и назначения прав доступа. Но в отчете по ДЗ можно ограничиться несколькими примерами таких команд (3-5 примеров).

2.5.4. Создание триггеров

Периоды участия в проекте не должны выходить за границы периодов выполнения этапов проекта. Например, если сотрудник участвует в 1-м этапе проекта, то дата начала участия должна быть не меньше, чем дата начала 1-го этапа, а дата окончания – не больше, чем дата завершения этого этапа. Реализуем эту проверку с помощью триггера уровня строки:

```
create or replace function check_period() returns trigger as $$
declare
     cnt integer;
begin
     select count(*) into cnt
          from  stages s
          where s_pro = new.j_pro and
               s_num = new.j_num and
               new.j_begin between s_begin and s_end and
               (new.j_end is null or
               new.j_end between s_begin and s_end);
     if cnt=0 then
          raise exception 'Период участия сотрудника с номером %
          не совпадает с периодом выполнения этапа', new.j_emp;
     else return NEW;
     end if;
end;
$$ LANGUAGE plpgsql;
```

После создания триггерной функции можно создать сам триггер:

```
create trigger tr_check_period
     before insert or update on Job
     for each row
     execute procedure check_period();
```

Примечание. Не забудьте проверить работоспособность создаваемых вами триггеров с помощью ввода команд, которые являются событиями триггера.

2.5.5. Создание индексов

Анализ готовых запросов показывает, что для повышения эффективности работы с данными необходимо создать индексы для всех внешних ключей (в тех системах, которые не создают такие индексы автоматически). Приведём примеры создания индексов:

```
create index ind_e_posts on employees(e_post);
create index ind_p_chief on projects(p_chief);
create index ind_e_room2 on employees(e_room, e_phone);
```

Также есть запросы, в которых учитывается роль участника проекта; добавим это поле к составному индексу на поля *Сотрудник* и *Роль*. Возможно, полезным окажется составной индекс по ФИО сотрудника. А для ускорения поиска текущих участников проекта нужен индекс по полям "начало" и "окончание участия" в таблице Участие. Создадим эти индексы:

```
create index ind_j_role on job(j_emp, j_role);
create index ind_e_name on employees(e_lname, e_fname);
```

```
create index ind_e_period on job(j_begin, j_end);
```

2.5.6. 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.

Примечание. Для ПрО с высокой интенсивностью обновлений или повышенными требованиями к надежности данных более рациональным может являться более частное инкрементное или частичное резервное копирование.

3. Выполнение домашнего задания

Домашнее задание выполняется бригадами по 2 человека по одному из вариантов, приведённых в следующем разделе, или для произвольной предметной области (по согласованию с преподавателем).

Отчет должен отражать ход выполнения домашнего задания (аналогично разобранному выше примеру, раздел 2). Реализация базы данных подразумевает набор команд создания 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 отчете.

4. Варианты предметных областей для домашнего задания

1. БД домашней фонотеки (БД музыкальных произведений).
2. БД для домашней видеотеки (БД кинофильмов).
3. Городская БД собственников автомобилей.
4. Городская БД собственников жилья.
5. БД книг из домашней библиотеки.
6. БД по прокату автомобилей.
7. БД кинологического клуба.
8. БД страховой компании.
9. БД кинотеатра.
10. БД вакансий.
11. БД аптеки.
12. Разработать (найти) и реализовать в виде БД классификацию (например, одну из предложенных ниже):
    - 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. Курс лекций и материалы для практических занятий: Учеб. пособие. – СПб., "Питер", 2013. – 240 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", 2017. – 1439 с.
3. Кузнецов С.Д. Базы данных: учебник для студ. учреждений высшего проф. образования. М.: Академия, 2012. – 496 с. – https://gdztest.com/2734-bazy-dannyh-kuznecov-sd.html (Дата обращения: 01.06.2023)
4. Introduction to Relational Database Design (based on Lecture by Tom Grayson). – http://web.mit.edu/11.521/www/lectures/lecture10/lec_data_design.html (Дата обращения: 01.06.2023)
5. Грабер М. Введение в SQL. – М.: Лори, 2008. – 378 с.
6. ФГУП "СтандартИнформ". Единая система программной документации. – http://www.standards.ru/collection.aspx?control=40&search=&sort=%20ASC&catalogid=temat-sbor&id=868075&page=1 (Дата обращения: 01.06.2023)