МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Т.В. ВОЛКОВА

# ПРОЕКТИРОВАНИЕ И СОЗДАНИЕ БД

Рекомендовано Ученым советом государственного образовательного учреждения высшего профессионального образования «Оренбургский государственный университет» в качестве учебного пособия для студентов, обучающихся по программам высшего профессионального образования по специальности «Программное обеспечение вычислительной техники и автоматизированных систем»

Оренбург 2006

УДК 004.4(075.8)
ББК 32.973.26—018.2я73
В 67

Рецензент
доктор технических наук, профессор Пищухин А.М.

**Волкова Т.В.**
**В 67 Проектирование и создание БД: учебное пособие / Т.В. Волкова — Оренбург: ГОУ ОГУ, 2006 — 140 с.**
**ISBN**

Описано место базы данных в автоматизированной информационной системе, рассмотрены вопросы, связанные с анализом предметной области, решением задач проектирования и создания базы данных, приведено описание методов и моделей данных, используемых на этапах проектирования БД. Центральная тема – построение адекватной модели предметной области, отображение её в физической модели БД. Уделено внимание аспектам создания и администрирования БД.

Учебное пособие предназначено для студентов, обучающихся по программам высшего профессионального образования по специальности 230105, при изучении дисциплин «Проектирование и создание базы данных», «Проектирование информационных систем», «Проектирование автоматизированных систем на основе БД».

ББК 32.973.26—018.2я73

1404000000
В

# Содержание

# Введение

В настоящее время разработка автоматизированных информационных систем (АИС), предназначенных для сбора, хранения и обработки информации является актуальной задачей в области современных информационных технологий.

АИС широко используются в различных сферах деятельности человека, в особой степени в области управления деятельностью предприятия. Под предприятием, в широком смысле слова, может подразумеваться любой современный бизнес, промышленное предприятие, лечебное или учебное заведение, любая финансово—хозяйственная деятельность.

Основой современных автоматизированных информационных систем, как правило, является база данных (БД), предназначенная для хранения большого объема информации, накапливаемой за продолжительный период времени деятельности предприятия.

От того, как спроектирована БД, как создана, как реализованы в ней все присущие предметной области требования и отграничения, зависит успешное функционирование АИС, возможность оперативного ввода и обработки информации, принятие своевременных управленческих решений.

В учебном пособии дается понятие АИС, состав её компонентов, значение информационного обеспечения АИС. В работе представлена архитектура БД, методы и модели, используемые для проектирования БД. Подробно рассмотрены этапы проектирования БД, вопросы по созданию и администрированию базы данных.

Учебное пособие предназначено для студентов, обучающихся по специальности «Программное обеспечение вычислительной техники и автоматизированных систем», при изучении дисциплин «Проектирование и создание базы данных», «Проектирование информационных систем», «Проектирование автоматизированных систем на основе БД». Знание основ проектирования и создания БД является важным для выполнения студентом выпускной квалификационной работы, для дальнейшей практической деятельности.

# 1 Автоматизированные информационные системы

## 1.1 Основные понятия

Автоматизированная информационная система это совокупность организационных, технических, программных и информационных средств, объединенных в единую систему и предназначенных для централизованного накопления и коллективного многоаспектного использования данных. При этом информационная составляющая АИС должна адекватно отражать регистрируемый в рамках этой АИС реальный мир.

АИС могут быть различного назначения: автоматизированные системы управления предприятиям (АСУ); автоматизированные системы управления технологическими процессами (АСУ ТП); системы автоматизированного проектирования (САПР); автоматизированные системы научных исследований (АСНИ). На рынке в настоящее время особенно востребованы экономические информационные системы (ЭИС): банковские, системы фондового рынка, страхования, налоговые и ряд других.

На любом предприятии циркулирует большой объем информационных потоков, состоящих из разного вида нормативных, правовых, распорядительных и отчетных документов, важных для успешной деятельности предприятия. Основное назначение АИС – автоматизация обработки информационных потоков предприятия, и, соответственно, повышение эффективности его управлением.

## 1.2 Экономические информационные системы

Необходимость автоматизации деятельности любого предприятия вызвана интенсивным нарастанием процессов информатизации всех сфер жизни общества, высоким уровнем развития информационных технологий, повышением требований к скорости обработки информации. Развитие информационных технологий делает возможным в настоящее время хранение больших объемов данных за продолжительные периоды времени, создание единого для предприятия центра обработки данных. Немаловажным фактором необходимости автоматизации является и повышение условий обслуживания клиентов предприятия, выпуск высококачественной продукции.

В экономической системе предприятия (системы с управлением) выделяют две подсистемы.

1 Объект управления. Это материальные объекты деятельности предприятия. Например: кадровые ресурсы, сырье, материалы, оборудование, сбыт продукции (для промышленного предприятия), контингент абитуриентов и студентов, организация учебного процесса в учебном заведении (для образовательного учреждения).

2 Систему управления. Это совокупность структурных подразделений предприятия, взаимодействующих между собой и осуществляющих функции

управления. Это, например, дирекция предприятия, плановый отдел, бухгалтерия, отдел кадров, производственные участки.

Функции системы управления:

— функция планирования – определяющая функционирование предприятия на различные периоды времени;

— функция учета, отображающая состояние объекта управления в результате выполнения бизнес процессов предприятия;

— функция контроля – определение отклонения учетных данных от плановых и нормативных;

— функция оперативного управления – мероприятие по реализации исключения возникших отклонений;

— функция анализа – определение тенденций в развитии предприятия и резервов, которые учитываются при планировании деятельности предприятия на следующий период времени.

Целями разработки ЭИС, как и любой другой АИС, являются создание информационной базы, отражающей деятельность предприятия и возможность получения актуальной информации о состоянии объектов управления на всех уровнях (оперативном, стратегическом и тактическом) управления предприятия. Всё это приводит к сокращению времени принятия решений, обеспечению требуемого качества управления.

В зависимости от охвата функций и уровней управления различают корпоративные и локальные АИС. Корпоративные АИС – это системы, которые охватывают (автоматизируют) все функции управления на всех уровнях управления предприятием. Они являются многопользовательскими, функционируют в вычислительной сети предприятий — локальной (ЛВС), или распределенной (РВС). Локальные АИС охватывают отдельные функции управления на отдельных уровнях управления. Локальная АИС, как правило, является однопользовательской и функционирует на отдельном компьютере.

В состав любой АИС входят функциональные и обеспечивающие подсистемы.

Функциональная подсистема АИС представляет собой комплекс задач с высокой степенью информационных обменов (связей) между задачами. Под задачей понимается некоторый процесс обработки информации с четко определенным множеством входной и выходной информации. Например, задачами являются «Ведение организационной структуры предприятия», «Планирование штатного расписания» «Учет прихода материалов». Функциональные подсистемы АИС делятся, как правило, по предметному, функциональному, проблемному или смешанному (предметно—функциональному) принципу. В свою очередь каждая задача состоит из ряда функций, например, функция добавления и обновления данных о структурном подразделении, функция поиска структурного подразделения по дате приказа о его создании и т.п. При проектировании большой системы рекомендуется её разделять на отдельные подсистемы, каждую из которых можно разрабатывать независимо от других.

Обеспечивающие подсистемы АИС являются общими для всех функциональных подсистем, независимо от конкретных функциональных подсистем.

Пример деления информационно—аналитической системы (ИАС) Оренбургского государственного университета, относящейся к классу интегрированных автоматизированных информационных систем управления высшим учебным заведением, на состав функциональных и обеспечивающих подсистем можно посмотреть на сайте ias.osu.ru.

Одними из ключевых требований к разрабатываемой автоматизированной информационной системе являются:

— использование современных информационных технологий;

— возможность функционального развития;

— независимость от роста объема обрабатываемой информации и количества одновременно работающих пользователей;

— обеспечение высокой надежности и устойчивости к сбоям;

— обеспечение непротиворечивости и полноты хранимой информации, её целостности;

— обеспечение надлежащего уровня защиты и конфиденциальности обрабатываемых данных.

## 1.3 Место БД в автоматизированной информационной системе

Основным компонентом АИС является подсистема «Информационное обеспечение». Главная часть информационного обеспечения – база данных. Предшественниками баз данных при решении задач сбора, хранения и обработки данных были файловые системы. В настоящее время они устарели по ряду причин. Рассмотрим эти причины.

1 Описание способа хранения данных и структура данных в файловых системах определяется в коде программ, что приводит, даже при незначительных изменениях этих структур, к перекомпиляции исходного кода.

2 Организация одновременного доступа к двум или более файлам требует определенных трудоемких действий, ведет к написанию дополнительного программного кода с элементами системного программирования.

3 При организации работы по ведению (вводу и обновлению) данных в файлах несколькими пользователями, территориально находящимися в разных помещениях, необходимо осуществлять дублирование данных. Это ведет к нарушению целостности данных, необходимости реализации их программного обновления и синхронизации.

4 Поскольку структура файла определяется кодом приложения, она также зависит от языка программирования, на котором создано это приложение. При обработке нескольких файлов разных форматов необходимо осуществлять их приведение к некоторому общему формату, что также достаточно трудоемко и требует написания отдельных программ.

5 Постоянное увеличение запросов, реализуемых к данным файлов, ведет к увеличению количества вновь разрабатываемых приложений.

Можно сделать вывод, что есть две основные причины, затрудняющие использование АИС на основе файловых систем. Это, во—первых, определение структур данных внутри прикладных программ, что ведет к зависимости данных от приложения. Во—вторых, необходимость разработки в рамках АИС инструментов для создания файлов, их наполнения, корректировки и обработки.

Использование баз данных и систем управления базами данных (СУБД – *DBMS – DataBase Management System*) является эффективным средством разработки и поддержки информационного обеспечения АИС.

База данных (БД – *DB — DataBase*) – это хранилище структурированных, непротиворечивых данных, минимально избыточных и целостных.

Важным преимуществом использования БД является то, что описание структур объектов БД хранится в самой базе данных. Такое описание называется системным каталогом (*system catalog*) или словарем данных (*data dictionery*). Элементы описания объектов базы данных в словаре данных называют метаданными (*meta—data*).

Хранение самоописания данных в БД обеспечивает независимость баз данных от приложений. Для одной БД может быть разработано много прикладных программ, реализованных в разных инструментальных средах, база данных одновременно может использоваться многими пользователями.

Компонентом, располагающимся между собственно физической БД и её пользователями, является СУБД. Это программное обеспечение, основная функция которого — предоставление пользователю БД широких возможностей работы с ней.

СУБД должна поддерживать определенные функции.

1 Создавать БД и объекты базы данных. Это осуществляется с помощью команд языка определения данных (ЯОД) СУБД. ЯОД позволяет задать структуру, тип, логические ограничения на данные, связи между данными, структуру доменов, индексов, триггеров и других объектов БД.

2 Вставлять, обновлять, удалять и извлекать информацию из БД. Эти операции осуществляются с помощью команд языка манипулирования данными (ЯМД) СУБД. Для извлечения данных существуют специальные языки запросов. Стандартными являются языки *SQL (Structure Query Language* – язык структурированных запросов), *QBE (Query By Example* – запрос по образцу). Наличие языка запросов устраняет присущие файловым системам ограничения – сформулированный на нем сложный запрос к БД выражается небольшим по объему кодом.

3 Предоставлять контролируемый доступ к данным с помощью следующих средств:

— системы обеспечения безопасности, предотвращающей несанкционированный доступ к объектам БД со стороны пользователей;

— системы поддержки целостности данных, обеспечивающей непротиворечивое состояние хранимых данных;

— системы управления параллельной работой приложений, контролирующей процессы совместного доступа к БД;

— системы восстановления, позволяющей восстановить БД до предыдущего непротиворечивого состояния, нарушенного в результате сбоя аппаратного или программного обеспечения;

— доступного пользователям каталога (словаря данных), содержащего описание хранимой в БД информации.

Реальный объем функциональных возможностей отличается в разных СУБД. Современные СУБД поддерживают широкий набор функций.

# 2 Методы и средства проектирования БД

## 2.1 Архитектура БД

Основной целью любой СУБД является возможность предложить обычному пользователю базы данных абстрактное представление данных, скрыв от пользователя особенности хранения и управления ими. Поскольку база данных, как правило, разрабатывается как общий ресурс для большого количества пользователей, то каждому пользователю может потребоваться своё, отличное от других пользователей представление о данных, хранимых в БД. Это вызвано следующими причинами:

— каждый пользователь должен иметь право обращаться к общим данным, используя своё представление о них;

— взаимодействие пользователя с БД не должно зависеть от особенностей её физической организации;

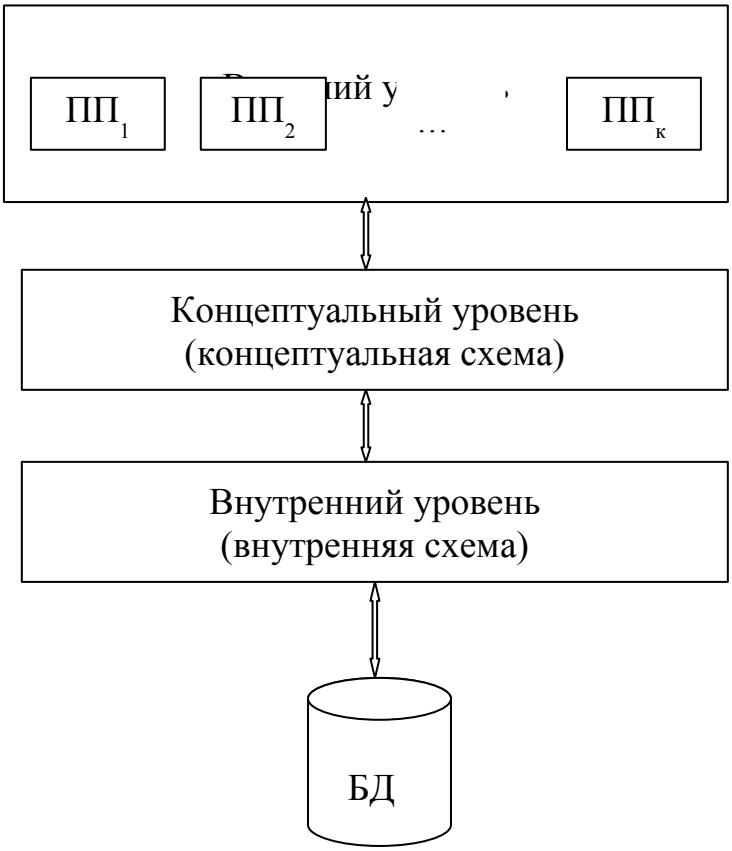
— администратор базы данных (АБД) должен иметь возможность изменять структуру и формат данных, не оказывая влияния на пользовательские представления;

— внутренняя структура БД не должна зависеть от переключения на новое устройство хранения;

— АБД должен иметь возможность изменять концептуальную структуру данных без какого—либо влияния на всех пользователей.

Для удовлетворения этих потребностей архитектура большинства современных коммерческих СУБД, существующих на рынке программного обеспечения, в той или иной мере, строится на базе так называемой архитектуры *ANSI—SPARC*. Название произошло по названию комитета планирования стандартов и норм (*Standards Planning and Requirements Committee SPARC*) национального института стандартизации (*American National Standard Institute— ANSI*) США. Комитет признал необходимость использования трехуровневого подхода к организации БД, отделяющего пользовательские представления базы данных от её физического представления посредством создания независимого уровня.

Архитектура БД представлена на рисунке 1. Внешний уровень – это совокупность индивидуальных представлений БД с точки зрения отдельных пользователей. Пользователи могут быть разные, с разным уровнем подготовки. Каждый пользователь представляет данные в соответствии с формами различных документов, присущих данной предметной области. При этом одни и те же данные могут иметь различную форму представления — тип, длину. Например, сведения о зарплате – их можно увидеть в виде итоговой суммы в записи ведомости, либо в виде перечня составляющих – различных начислений и удержаний.




$ПП_1$ – представление 1—го пользователя, $ПП_к$ – представление к—того пользователя

Рисунок 1 — Трехуровневая архитектура БД

Некоторые представления могут включать производные или вычисляемые данные, которые не хранятся в БД, а создаются по мере необходимости. Индивидуальное внешнее представление называют подсхемой.

Концептуальный уровень отображает обобщенное представление пользователей. Это промежуточный уровень в трехуровневой архитектуре БД, представляющий данные такими, какими они есть на самом деле, а не такими, какими их вынужден видеть пользователь в рамках какого—то инструментального средства или приложения. Концептуальный уровень поддерживает каждое внешнее представление, однако не содержит сведений о методах хранения данных. Этот уровень интересен администратору БД и может быть представлен концептуальными схемами.

Внутренний уровень на языке определения данных выбранной СУБД представляет, в каком виде информация хранится в БД, описывает структуры объектов БД. Внутреннее представление не связано с физическим уровнем, так как в нем не рассматривается организация физических записей (блоков и страниц памяти), физической области хранения данных.

Описание уровней архитектуры БД можно представить в виде некоторого технического описания (в бумажном или электронном виде), разработанного с помощью различных средств и правил.

В соответствии с архитектурой БД, существуют несколько внешних схем или подсхем, каждая из которых соответствует разным представлениям данных, одна концептуальная схема БД и одна внутренняя схема БД. Каждая внешняя схема связана с концептуальной с помощью внешне концептуального отображения. Концептуальная схема связана с внутренней посредством концептуально внутреннего отображения. Схемы и отображения создаются администратором БД средствами СУБД.

Рассмотрим представление различных уровней архитектуры базы данных на примере БД, созданной для решения задачи автоматизации управлением персоналом некоторой фирмы.

Внешний уровень представляется внешней схемой, содержащей несколько подсхем, отображающих внешние представления различных пользователей БД. Во—первых, это представление сотрудника фирмы, выполняющего обязанности кадрового учета. Его функциями являются учет личных данных сотрудников, сведения о приеме их на работу, всех перемещениях и увольнениях. Сотрудник—кадровик представляет данные в таком виде: каждому сотруднику присваивается табельный номер, это 5—значное целое число. Личные данные сотрудника вводятся в базу данных из следующих документов: паспорта, диплома об образовании, договора о найме на работу. Сотрудник—кадровик создает личную карточку сотрудника, в которую, кроме личных данных, вводит сведения о том в какое подразделение и на какую должность принимается сотрудник. Эти данные он берет из приказов о приемах, переводах, увольнениях и отпусках. Сотрудник—кадровик также периодически формирует различные отчеты по качественному составу персонала для руководителя фирмы: список сотрудников, достигших пенсионного возраста на заданную дату, список сотрудников, имеющих более 2 —х несовершеннолетних детей, список сотрудников, не имеющих высшего образования и тому подобное. Второе внешнее представление – это представление сотрудника, выполняющего функции начисления заработной платы. Он ведет учет количества отработанных дней каждого сотрудника, сведения о полагающихся сотруднику надбавках, ежемесячно осуществляет расчет заработной платы, формирует ряд отчетов, как для руководителей фирмы, так и для внешних организаций – налоговой инспекции, пенсионного фонда. Третий пользователь базы данных это руководитель фирмы, который может просматривать личные данные сотрудников, сведения о выплатах по зарплате. Четвертый внешний пользователь — это пользователь сети Интернет, который на сайте фирмы может получить следующие сведения о сотрудниках фирмы – фотографии, фамилии, имена и отчества, названия подразделений, должности, ученые звания, степени и другую общедоступную информацию. Таким образом, можно представить, что каждый пользователь видит «свой» фрагмент данных, при этом данные могут иметь разное представление по длине, по виду отображения. Описание всех представлений пользователей дает

общее описание внешнего уровня БД. Оно обычно представляется в виде технической документации (технического задания), описании входных и выходных документов, в виде схем, рисунков, словесного описания.

Концептуальный уровень базы данных рассматриваемого примера может быть отображен в виде 2—х схем, получаемых последовательно. Первая схема — концептуальная информационно—логическая модели предметной области (ИЛМ), представляющая классы объектов (сущности) предметной области и связи между ними. Вторая схема — концептуальная даталогическая модель базы данных (ДЛМ), отображающая описание предметной области в виде логической структуры данных в терминах выбранной модели данных. Каждая из этих схем обобщает представление всех пользователей базы данных и описывает все данные, которые должны храниться в БД, какие связи между ними существуют. Схемы могут быть отображены на бумаге, либо в электронном формате и являются проектной документацией. Формируют концептуальный уровень проектировщики базы данных, администратор данных и администратор базы данных. Это особая категория пользователей базы данных — её разработчики. Представление концептуального уровня БД в виде ИЛМ и ДЛМ, либо только в виде ДЛМ, зависит от выбранного метода проектирования БД.

Внутренний уровень рассматриваемой БД может быть представлен в виде *SQL*—скриптов объектов БД, если для реализации БД выбрана СУБД, поддерживающая стандарт языка *SQL*, либо в виде описания, содержащего конструкции языка определения данных выбранной СУБД.

Основным назначением трехуровневой архитектуры БД является обеспечение независимости описаний базы данных (схем БД), получаемых на различных уровнях архитектуры БД, друг от друга, что обеспечивает независимость прикладных программ от данных и является одним из основных достоинств базы данных.

Различают логическую и физическую независимость данных. Логическая независимость данных поддерживает защищенность внешних схем от изменений, вносимых на концептуальном уровне. Так, например, при функциональном развитии АИС проводится дообследование предметной области и в концептуальную схему добавляется описание новых данных. При этом некоторые внешние схемы даже не замечают этих изменений, часть пользователей работает с БД в прежнем виде. Физическая независимость поддерживает защищенность концептуальной схемы от изменений, вносимых на физическом уровне. Например, добавление индексов, создание триггеров не требуют внесения изменений в концептуальную схему. Также, возможен перевод физической модели БД на ЯОД другой СУБД без внесения изменений во внешние схемы. Пользователь сможет заметить только увеличение или уменьшение производительности системы.

Для того чтобы создать БД необходимо предварительно осуществить последовательное проектирование схем каждого уровня архитектуры БД.

## 2.2 Модели данных

Одним из важнейших моментов в процессе проектирования БД является понимание концепций, лежащих в основе моделей данных, используемых как в качестве инструмента проектирования, так и в качестве результата проектирования. В сложившейся терминологии БД и инструменты проектирования, и его результаты рассматриваются как модели. Необходимо различать назначение каждой используемой в процессе проектирования модели данных.

В современной трактовке термин «модель данных» обозначает инструмент моделирования. Модель базы данных (схема данных) или модель предметной области являются результатами моделирования.

Модель данных это интегрированный набор понятий для описания данных, связей между ними и ограничений, накладываемых на данные, на предприятии. Для реализации успешного проектирования БД необходимо для каждой модели – инструмента знать три её основных компонента:

— набор правил, определяющих структуру данных;

— определение типов допустимых операций с данными;

— набор ограничений поддержки целостности данных.

Исходя из трехуровневой архитектуры БД, различают три, связанных между собой, видов моделей данных, получаемых в результате проектирования.

1 Внешняя модель данных, отображает обобщенное представление всех пользователей. Эту модель называют описанием предметной области, формируемым на естественном языке. Представить внешнюю модель можно как в формализованном (схемы, рисунки, таблицы), так и в неформализованном (словесное описание на языке проектировщиков) виде.

2 Концептуальные модели. Концептуальная информационно логическая модель предметной области может быть выражена в виде диаграммы, схемы, рисунка, отображающих обобщенное логическое представление информации предметной области. Концептуальная даталогическая модель данных в виде рисунка, схемы, отображающих обобщенное логическое представление данных.

3 Внутренняя модель. Является результатом отображения концептуальной модели средствами языка определения данных выбранной СУБД.

В литературе по БД предложено и опубликовано достаточно много различных моделей данных, используемых при проектировании БД как инструментальное средство. Модели, отображающие уровни архитектуры БД, строятся по правилам этих моделей.

Модели данных, как инструменты, делятся на 3 основные категории.

1 Объектные модели данных. В этих моделях используются такие понятия как: классы объектов (типы сущностей), объекты (экземпляры сущностей), свойства классов объектов (атрибуты сущностей), связи между классами объектов. В скобках приведена исторически более ранняя терминология, используемая в теории баз данных.

Среди объектных моделей выделяют наиболее общие типы:

— семантические модели. Их назначение – обеспечение возможности выражения семантики (смысла) предметной области. Это, например, модели типа "сущность—связь" (*ER*—модели — *Entity Relationship model*), отображающие семантику предметной области в виде *ER*—диаграмм;

— функциональные модели, дающие представление о функциях автоматизируемого предприятия, о распределении ответственности за их выполнение. Результаты использования функциональных моделей могут быть представлены в виде диаграмм бизнес—функций, диаграмм потоков данных;

— объектно—ориентированные модели. Эти модели расширяет определение класса объектов (сущности) предметной области с целью включения в определение не только свойств, описывающих состояние объекта, но и действий, которые с ним связаны, т.е. его поведение. Это, например, модели, основанные на использовании языка *UML (Unified Modeling Language* — унифицированного языка моделирования). Описание предметной области получают в виде различных диаграмм — диаграмм вариантов использования, диаграмм деятельности, диаграмм классов.

В настоящее время для проектирования БД, получения концептуальной инфологической модели предметной области, широко используются семантические модели «сущность—связь».

2 Модели на основе физических записей. Такие модели позволяют описывать логическую структуру БД в виде записей, фиксированного формата. Каждый тип записи определяет фиксированное количество полей, поля имеют фиксированную длину. Существует три основных типа логических моделей данных на основе записей:

— иерархическая (*hierarchical data model*);

— сетевая (*network data model*);

— реляционная (*relational data model*).

3 Физические модели данных. Модель содержит всю информацию, необходимую для реализации конкретной БД в среде выбранной (целевой) СУБД. В физической модели в виде описания содержится информация обо всех объектах БД. В описании объектов БД определяется физический формат данных, реализуются ограничения предметной области, бизнес—логика автоматизируемого предприятия, уровни доступа пользователей. Описание создается на языке определения данных (ЯОД) выбранной (целевой) СУБД. В состав ЯОД входят операторы, позволяющие создать или удалить объект БД, модифицировать его структуру. Физическая модель данных не затрагивает вопросы физического размещения данных на машинных носителях, в настоящее время это максимально реализуется средствами СУБД.

Модели первых двух групп используются для формирования концептуального уровня архитектуры БД, третьей – для описания БД на внутреннем уровне.

Модель данных, полученная в результате проектирования, должна представлять автоматизируемое предприятие в таком виде, который позволит проектировщикам и пользователям БД обмениваться конкретными

недвусмысленными мнениями. Оптимальная модель данных, полученная в результате проектирования, должна удовлетворять следующим критериям:

— обладать структурной достоверностью – способы определения информации в модели данных должны соответствовать организации информации на рассматриваемом предприятии;

— быть относительно простотой, легко понимаемой, как профессионалами в области разработки БД, так и обычными пользователями;

— обладать выразительностью — представлять отличия между разными типами данных, связями и ограничениями;

— обладать отсутствием избыточности, любая часть данных должна быть представлена только один раз;

— обладать возможностью совместного использования, не принадлежать к какому—то особому приложению или технологии;

— быть целостной, согласованной со способами использования и управления информацией внутри предприятия;

— обладать расширяемостью, эволюционировать с целью включения новых требований с минимальным влиянием на уже существующих пользователей;

— иметь возможность представления в виде диаграмм.

Использование моделей данных на этапах проектировании БД представлено в таблице 1.

Таблица 1 — Модели данных, используемые при проектировании БД

| **Уровень архитектуры БД** | **Модель данных, как инструмент, используемый для формирования схемы БД** | **Результат проектирования** |
|---|---|---|
| Внешний уровень | Функциональные модели, модели на основе языка *UML.* | Диаграмма иерархии функций, диаграмма потоков данных и др. |
| Концептуальный уровень | 1. Семантические модели («сущность—связь»)<br><br>2. Модели на основе физических записей | 1. *ER*—диаграмма предметной области – концептуальная информационно—логическая (инфологическая) модель (ИЛМ) предметной области<br>2. Логическая структура БД – концептуальная даталогическая модель (ДЛМ) БД. |
| Внутренний уровень | ЯОД СУБД | 1. Техническое описание объектов БД.<br>2. *SQL*—скрипты объектов БД. |

## 2.3 Жизненный цикл БД

Поскольку база данных является основным компонентом автоматизированной информационной системы, её жизненный цикл непрерывно связан с жизненным циклом АИС, с жизненным циклом программного обеспечения АИС. Жизненный цикл АИС, в целом, определяется как период времени, который начинается с момента принятия решений о необходимости создания АИС и заканчивается завершением её полной эксплуатации. Наиболее распространены две основные модели жизненного цикла программного обеспечения: каскадная и спиральная модели. Жизненный цикл АИС, её программного обеспечения, базы данных, как правило, соответствует спиральной модели.

Рассмотрим некоторые этапы жизненного цикла БД.

1 Планирование разработки БД. Это подготовительные действия, позволяющие с максимально возможной эффективностью реализовать этапы жизненного цикла БД. На данном этапе решают следующие задачи:

— анализ функционирования автоматизируемого предприятия в соответствии с требованиями, предъявляемыми ему — определение бизнес—планов и целей предприятия с последующим выделением потребностей предприятия в информационных технологиях;

— анализ существующих на предприятии автоматизированных информационных систем с целью выявления их сильных и слабых сторон (однопользовательские системы, устаревшее программное обеспечение и т.п.);

— формулирование потребностей в новой АИС (определение всех недостатков существующих на рынке программного обеспечения подобных АИС, их высокая стоимость, высокая сложность их сопровождения и т.п.);

— разработка стандартов, определяющих технологию сбора данных, их форматов, определение состава требуемой технической документации, порядка проектирования и реализации. Четко определенный набор стандартов позволяет создать хорошую основу для последующего обучения персонала, организации контроля качества, гарантировать определение работ по строго определенным образцам. Например, специальные правила могут определять, как присваиваются имена элементам данных, описываемых в словаре данных базы данных.

2 Определение требований к системе. Это этап определения диапазона действия и границ разрабатываемой БД, состава её пользователей, областей применения.

Задачи этапа:

— анализ и выбор направления совершенствования объекта управления в рамках предприятия;

— установление границ исследуемой области;

— связь разрабатываемой системы с существующими на предприятии АИС;

— выбор программно—технических средств;

— определение ограничения ресурсов на разработку;

— определение состава возможных будущих пользователей;

— направлений развития.

3 Сбор и анализ требований пользователей. Сбор и анализ информации о той части предприятия, работа которого будет поддерживаться с помощью создаваемой АИС. Необходимая для проектирования БД информация может быть собрана следующими способами:

— посредством опроса специалистов предприятия в наиболее важных областях его деятельности;

— с помощью наблюдения за деятельностью предприятия;

— посредством изучения документов, особенно тех, которые используются для сбора или представления информации – входных, выходных документов;

— за счет использования опыта проектирования других подобных систем;

— с помощью анкет. Например, проведение анкетного опроса руководителей подразделений, будущих пользователей БД. В связи с разным функциональным назначением подразделений, у руководителей может быть разное видение предметной области.

Требования (это некоторая функция, которая должна быть включена в создаваемую систему) пользователей анализируются с целью выяснения всех необходимых потребностей автоматизируемого предприятия. Объем собранных требований зависит от сути проблемы и действующих бизнес—правил предприятия. Слишком тщательный анализ легко может привести к получению большого объема требований, которые будет трудно проанализировать, а поверхностный анализ – к пустой трате времени и денежных средств на проведение работ, решение может оказаться ошибочным.

На этапе сбора и анализа требований пользователей полезно ответить на шесть следующих вопросов:

а) Что лежит в основе деятельности данного предприятия. Выявляются наиболее важные компоненты, подлежащие автоматизации. Например: подсистема «Управление персоналом» – компоненты: предприятие, отделы, организационно—кадровая структура, личная карта сотрудника, приказ; задача «Почта» – компоненты: отправитель, получатель, почтовое отделение, главпочта; задача «Занятость населения» – компоненты: отдел по трудоустройству, предприятия, учебные заведения, начисление пособий; задача «Учет кассовых операций» – компоненты: касса, бухгалтерия.

б) Как это делается? Определяется список основных процессов, происходящих на данном предприятии, в частности над компонентами. Например, задача «Почта» – обеспечивается ввод, хранение и обработка данных об отправителе, получателе и почтовом отделении. Автоматически рассчитывается сумма оплаты почтовых услуг и выдается квитанция об оплате. Каждый день составляется и выдаются на печать отчеты о принятых и выданных почтовых отправлениях; задача «Занятость населения» – сбор, накопление, обработка и передача данных о безработных, регистрация вакантных рабочих мест, заполнение карточек персонального учета, ведение

истории по безработице, выдача рекомендательных писем; задача «Учет кассовых операций» – ведение журнала регистрации кассовых операций, оформление приходных и расходных, кассовых ордеров и т.п.

в) Где происходят данные процессы? Территориально определяется положение компонентов, решается вопрос оптимального распределения данных (локальная, сетевая БД, БД уровня предприятия, распределенные БД).

г) Кто выполняет эти процессы? Рассматривается организационная структура предприятия – подчинение и зависимость подразделений.

д) Когда выполняется то или иное действие? Проясняется периодичность времени осуществляемых процессов, последовательность выполнения.

е) Почему эти действия выполняются? Определяется мотивация деятельности предприятия. Например: достижение наилучших соотношений "затраты – удобства" для клиентов; обеспечение успешной деятельности персонала; получение приемлемой прибыли; повышение доходов при автоматической обработке данных; получение более эффективного управления.

ж) Требования к техническому или аппаратному обеспечению АИС.

з) Сроки сдачи АИС в эксплуатацию.

В результате полученных ответов на данные вопросы должны быть сформированы, как минимум, следующие документы, на основе которых будет производиться проектирование программной системы:

— название задачи;

— организационная структура предприятия;

— описание существующих на предприятии информационных потоков, выделение автоматизируемых;

— описание структуры входных и выходных документов;

— функции, которые должны быть автоматизированы в рамках данной задачи;

— формализованное описание предметной области – классы объектов (сущности) и связи между ними;

— правила (бизнес – правила, семантические утверждения), ограничивающие предметную область;

— требования к программному обеспечению АИС;

— требования к техническому обеспечению АИС;

— сроки ввода в эксплуатацию АИС.

4 Этап проектирования. Это процесс создания проекта БД, предназначенной для поддержки функционирования АИС. Это сложный процесс отображения словесного и естественного описания предметной области в схему внутренней модели БД.

Основными целями проектирования БД являются:

— представление данных и связей между ними, необходимых для всех областей применения БД и любых существующих групп пользователей;

— создание модели данных, способной поддерживать выполнение требуемой обработки данных;

— разработка предварительного варианта проекта, структура которого позволяет удовлетворить все основные требования, предъявляемые к производительности системы – например, ко времени реакции системы.

Любое проектирование – это поиск способа удовлетворения функциональных требований средствами имеющейся технологии с учетом заданных ограничений. Для каждого проекта существует ряд абсолютных требований, как правило, это:

— максимальное время, отпущенное на проект;

— максимальное количество денег, которое может быть потрачено;

— масса других неудобных требований и ограничений (недостаточная подготовка специалистов, отсутствие технического задания на проектирование, недопонимание сотрудников автоматизируемых подразделений сути автоматизации и т.п.).

Проектирование БД охватывает 3 основные области:

— проектирование конкретных объектов, которые будут реализованы в БД (домены, таблицы, хранимые процедуры, триггеры, индексы);

— проектирование конкретных экранных форм, отчетов, программ, которые будут сопровождать данные в БД, и обеспечивать выполнение запросов к ним;

— при определенных обстоятельствах в процессе проектирования также необходимо учитывать конкретную среду или технологию – например, топологию сети, конфигурацию аппаратных средств, использование архитектуры клиент/сервер, параллельной обработки данных, распределенной архитектуры БД.

Реальное проектирование заключается в достижении компромиссов и строится на информированном принятии решений с учетом различных ограничений (видение разработчиком определенной технологической цепочки обработки информации, а в реальном состоянии дел на предприятии реализовать эту цепочку не получается по каким—либо причинам), развитием информационных технологий, постоянного увеличения требований к разработке, объема знаний разработчиков.

5 Выбор целевой СУБД. Выбор может осуществляться в любой момент разработки базы данных, но строго до этапа проектирования логической структуры БД (до построения концептуальной даталогической модели БД). Выбор должен осуществляться при условии, что имеется вся необходимая информация о таких требованиях как: производительность системы; стратегия реализации ограничений целостности БД; уровень защищенности данных; архитектура вычислительной среды; необходимость параллельной обработки данных.

6 Разработка приложений. Это этап проектирования интерфейсов пользователей и прикладных программ для работы с БД. В жизненном цикле БД этот этап выполняется параллельно с проектированием БД. Между этими фазами должен постоянно происходить информационный обмен, перекрестные проверки между проектируемыми данными и выявленными функциями разрабатываемого приложения. Необходимо убедиться в том, что модель

данных отражает потребности бизнеса, а модель функций использует данные из модели данных.

7 Создание БД. Этап физической реализации БД в среде выбранной СУБД. Включает в себя следующее:

— компиляцию команд ЯОД, описывающих БД и структуры объектов БД — создание пустой схемы БД;

— реализацию прикладных программ с помощью языков программирования (многие СУБД имеют встроенный язык);

— реализацию элементов прикладных программ на языке манипулирования данными (ЯМД) СУБД;

— разработку экранных форм для ввода—вывода информации, отчетов;

— реализацию мероприятий по защите данных, как правило, средствами ЯОД и ЯМД СУБД.

8 Конвертирование и загрузка данных из старой системы. Конвертирование и загрузка данных – перенос любых существующих данных в новую БД и модификация существующих приложений с целью организации совместной работы с новой БД. Этот этап выполняется только в том случае, если новая БД заменяет или включает данные старой, наследуемой БД. Для реализации подобных видов работ разрабатываются программы—конверторы. На этом этапе также осуществляется заполнение справочников БД, необходимых для обеспечения начала работы АИС.

9 Тестирование БД. Тестирование – процесс выполнения функций, объявленных в АИС, с целью выявления ошибок. Продумывается стратегия теста с использованием реальных данных. Если тестирование проведено успешно, оно обязательно вскроет ошибки в прикладных программах и, возможно, в структурах данных. Как правило, тестирование реализуется на отдельном оборудовании и тестовой БД. Если же тестирование осуществляется на реальных данных, то обязательно делается резервная копия БД.

По завершении этого этапа БД готова к эксплуатации.

10 Эксплуатация и сопровождение. На данном этапе проводится наблюдение за системой и поддержка её нормального функционирования. При этом выполняют следующие виды работ:

— контроль производительности системы. Если она падает, возможна дополнительная настройка или реорганизация БД (денормализация), оптимизация *SQL* запросов, создание дополнительных объектов БД (индексов);

— сопровождение и модернизация, в случае необходимости, элементов прикладных программ на языке манипулирования данными (ЯМД) СУБД;

— администрирование БД: проверка эффективности системы блокировок в параллельных процессах; осуществление мониторинга работы системы; создание резервных копий БД и т.п.

Начальные этапы эксплуатации БД должны осуществляться параллельно с бумажной обработкой информации на предприятии, либо параллельно с работой старой системы. Впоследствии возможна некоторое время работа двух (старой и новой) систем параллельно, затем окончательный переход на работу новой системы, модифицированной по результатам эксплуатации.

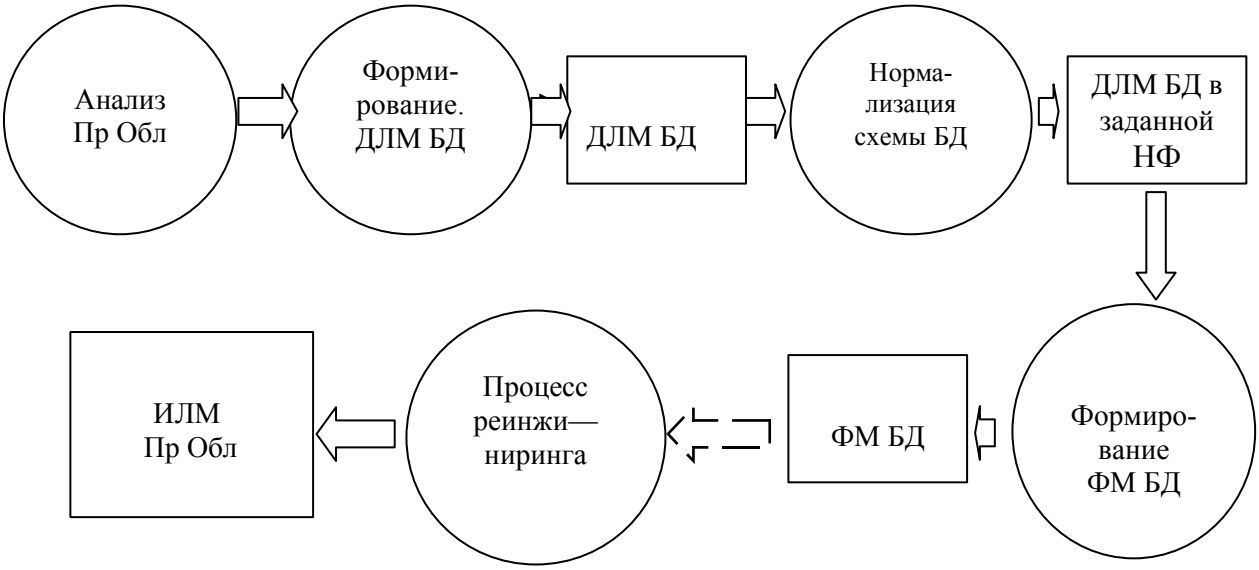
## 2.4 Методы проектирования БД

Целью проектирования БД является адекватное отображение в базе данных сути предметной области, рассматриваемой с точки зрения решения задачи автоматизации.

В теории баз данных существует ряд методов разработки моделей БД, отображающих разные уровни её архитектуры. Распространены два основных подхода к проектированию систем баз данных: "нисходящий" и "восходящий":

При «восходящем» подходе осуществляют структурное проектирование снизу—вверх. Этот процесс называют процессом синтеза, попыткой получения целого, адекватно отображающего описание предметной области, на основе описания составляющих его частей.

Этапы проектирования БД методом «восходящего» проектирования представлены на рисунке 2.




Пр Обл – предметная область; ДЛМ – даталогическая модель; НФ – нормальная форма; ИЛМ – информационно—логическая модель; ФМ – физическая модель.

Рисунок 2 — Этапы проектирования БД методом «восходящего» проектирования

Работа над проектом БД начинается с определения свойств объектов (атрибутов сущностей) предметной области, которые на основе анализа существующих между ними связей группируются в реляционные отношения (таблицы), отображающие эти объекты (в том случае, если мы проектируем структуру реляционной БД). Как правило, получают два — три, связанных между собой, реляционных отношения. Для того чтобы полученная структура БД (ДЛМ) не обладала различными аномалиями при добавлении, обновлении или удалении данных вследствие их избыточности, необходимо осуществить

проверку каждой полученной схемы отношения, как минимум, на соответствие 3НФ. Если схемы отношений не соответствуют этому условию, а они, скорее всего, не будут соответствовать, необходимо проводить процесс нормализации схем отношений.

Для успешного проведения нормализации необходимо на основе анализа предметной области (анализа документов предметной области) для каждой схемы реляционного отношения:

— выявить потенциальные ключи;

— увидеть повторяющиеся группы и не атомарные атрибуты;

— привести схемы отношения к 1НФ;

— определить функциональные зависимости между не ключевыми атрибутами и первичным ключом;

— определить частичные функциональные зависимости;

— осуществить декомпозицию (деление) соответствующих схем отношений для удалений частичных функциональных зависимостей;

— увидеть транзитивные зависимости между не ключевыми атрибутами и первичным ключом;

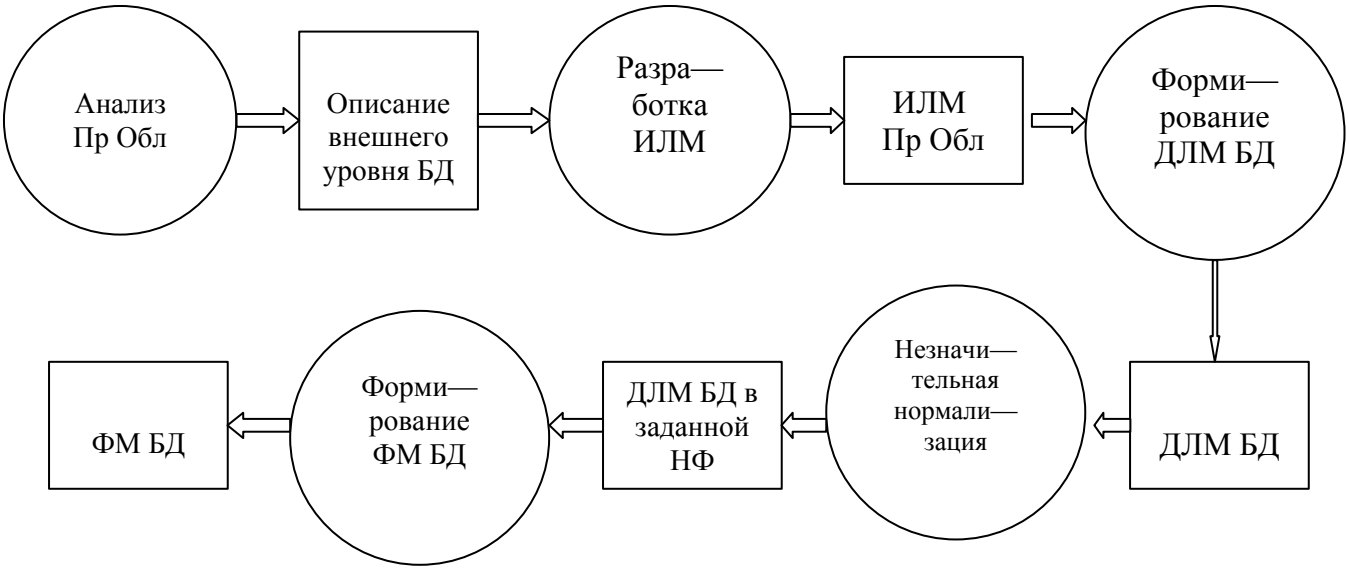
— исключить транзитивные зависимости путем декомпозиции соответствующих схем отношений.

Эти виды работ являются достаточно трудоемкими, и их успех будет определяться хорошим знанием предметной области, теории множеств и предикатной логики. Значительный объем мероприятий по нормализации схем реляционных отношений даже дал второе название методу «восходящего» проектирования. Этот метод часто называют методом «нормализации».

«Восходящее» проектирование – это достаточно сложная и устаревшая методика, которая подходит для проектирования только небольших баз данных.

При «нисходящем» проектировании осуществляется структурное проектирование сверху—вниз («нисходящее» проектирование). Такое проектирование называют анализом – происходит изучение целого (описания предметной области), затем разделение целого на составные части и затем следует последовательное изучение этих частей.

Этапы проектирования БД методом «нисходящего» проектирования представлены на рисунке 3. Для отображения метода использованы следующие обозначения: в кругах описаны названия этапов проектирования, в прямоугольниках – результаты. Проектирование начинается с анализа предметной области и формирования описания внешнего уровня БД, объединяющего представления всех пользователей разрабатываемой БД, выявления классов объектов (сущностей) предметной области, связей между ними (определения, описания предметной области). На основе описания внешнего уровня БД строится концептуальная информационно—логическая модель предметной области (ИЛМ), затем на её основе получают даталогическую модель (ДЛМ) базы данных. ДЛМ является основой для следующего этапа проектирования БД – этапа формирования физической модели базы данных.




Пр Обл – предметная область; ИЛМ – информационно—логическая модель предметной области; ДЛМ – даталогическая модель; НФ – нормальная форма; ФМ – физическая модель.

Рисунок 3 — Этапы проектирования БД методом «нисходящего» проектирования

Метод «нисходящего» проектирования достаточно формализован и используется в *CASE (Computer Aided System /Software Engineering* — компьютерное проектирование программного обеспечения и систем) средствах. Проведение тщательного анализа предметной области, выявление всех присущих ей классов объектов и связей между ними, правильное их отображение в ИЛМ предметной области, ведет к получению высоко нормализованной схемы логической структуры реляционной БД.

Для того чтобы сравнить эти два метода, используемых для проектирования реляционных баз данных, необходимо понимать, что при использовании «восходящего» метода проектирования для реляционной базы данных сразу формируется схема БД. Термины реляционной модели (схемы отношений) не предусматривают возможность описания полного смысла (семантики) предметной области. Неправильное отображение в даталогической модели БД сути предметной области приводит к ошибкам в последующей работе АИС. Установлено, что цена ошибки (стоимость исправления) быстро возрастает с увеличением интервала времени (технологического времени: числа выполненных операций между двумя событиями) между появлением погрешности и её обнаружением. В литературе по базам данных приводятся такие цифры: на интервале от выработки требований на программу до сдачи программного продукта заказчику стоимость расходов на исправление ошибки возрастает в среднем в 80 раз. Большая цена ошибки определяет необходимость тщательной проработки проекта.

Также необходимо отметить, что ручное проектирование, каким является метод «восходящего» проектирования или метод «нормализации», является достаточно трудоемким.

Более удобный, приятный и современный метод проектирования БД – это «нисходящий» метод. Его часто называют методом концептуального проектирования БД. Такое проектирование, прежде всего, связано с попыткой более полного представления семантики предметной области в модели БД. Это стало возможным с появлением семантических моделей данных, которые позволяют описать конкретную предметную область достаточно формальным, но в тоже время понятным и разработчику и заказчику образом, что дает возможность отображения в модели общего понимания сути предметной области.

Предметная область определена, если известны существующие в ней объекты, их свойства и отношения между ними. В концептуальном подходе к проектированию БД выделяют следующие три сферы:

— реальный мир или объектную систему;

— информационную сферу;

— даталогическую сферу.

Основными составляющими объектной системы являются: объект (экземпляр сущности), свойство (атрибут), отношение (связь). Объект в концептуальном подходе – это то, о чем в информационной системе должна накапливаться информация.

Класс объектов может состоять из одного или более объектов. Например, класс объектов ФИЗИЧЕСКОЕ ЛИЦО, отдельные объекты – Иванов, Петров, Сидоров. Каждый класс объектов должен обладать уникальным идентификатором, который однозначно идентифицирует каждый отдельный объект (экземпляр сущности) в классе объектов. Каждый класс объектов должен обладать некоторыми свойствами (атрибутами), количество которых одинаково для каждого объекта в классе объектов, значение же каждого свойства может быть различным в разных объектах. Каждый класс объектов может обладать любым количеством связей с другими классами объектов.

Информационная (инфологическая) сфера представляется понятиями, с помощью которых можно формально описать и проанализировать информацию об объектной системе.

В даталогической сфере рассматриваются вопросы представления предметной области (описанной в информационной сфере) с помощью структур данных, определяемых выбором СУБД. В настоящее время наиболее широко для формирования даталогической сферы используются реляционные СУБД.

В основе концептуального подхода лежит идея установления последовательного соответствия между объектной системой, информационной и далее даталогической сферами. Происходит последовательное преобразование понимания объектов предметной области и связей между ними в формализованное описание логики информации предметной области и

дальнейшее преобразование логики информации предметной области в описание структуры базы данных в терминах выбранных структур данных – построение логики данных. Такое последовательное преобразование позволяет понятным и простым образом осуществлять правильное отображение смысла реального мира в базе данных. Таким образом, концептуальное проектирование БД состоит из следующих последовательных этапов:

— анализ предметной области, выявление классов объектов и связей между ними (формирование внешнего уровня БД);

— концептуальное инфологическое проектирование. Строится концептуальная информационно—логическая модель (ИЛМ) предметной области, формально (в терминах выбранной методологии построения ИЛМ) описывающая классы объектов и связи между ними (формирование концептуального уровня БД);

— концептуальное даталогическое проектирование. На основе ИЛМ в терминах выбранной модели данных строится концептуальная даталогическая модель (ДЛМ) БД (формирование концептуального уровня БД);

— преобразование ДЛМ в физическую модель БД, полученную на ЯОД выбранной СУБД (формирование внутреннего уровня БД).

Современный метод проектирования БД состоит из 3—х основных этапов: инфологическое проектирование, даталогическое проектирование и физическое проектирование. Необходимо отметить, что на втором этапе проектирования ИЛМ можно преобразовать в даталогическую структуру, используя любую модель – иерархическую, сетевую, реляционную, традиционные файлы.

Помимо этих подходов, для проектирования БД могут применяться другие методы. Например, известен метод "изнутри наружу". Он похож на метод «восходящего» проектирования, но отличается от него начальной идентификацией набора основных классов объектов с последующим расширением круга рассматриваемых классов объектов, связей и свойств, которые взаимодействуют с первоначально определенными классами объектов. Известен также подход "смешанной стратегии" — сначала «восходящий» и «нисходящий» методы используются для разных частей модели, после чего все подготовленные фрагменты собираются в единое целое.

## 2.5 CASE − технологии

Проектирование БД может быть автоматизировано. Для этого используются различные *CASE* − средства, особенно эффективно их использование при создании крупных корпоративных АИС большим коллективом разработчиков. Использование современных *CASE* − средств позволяет поддерживать как начальные этапы разработки АИС, так и проектирование, и генерацию баз данных и пользовательских интерфейсов. *CASE* − средства обеспечивают качество принимаемых технических решений и подготовку проектной документации.

*CASE*—средства классифицируются по методологиям проектирования (структурно—ориентированные, объектно—ориентированные, комплексно—ориентированные), по графическим нотациям построения диаграмм, по степени интегрированности (отдельные локальные средства, набор интегрированных средств, охватывающих большинство этапов разработки АИС), по режиму коллективной разработки проекта (режим реального времени, режим объединения проектов) и ряду других. Современный рынок программных средств насчитывает около 300 различных CASE − средств, наиболее популярные приведены в таблице 2.

Таблица 2 – CASE − средства, используемые для проектирования БД.

| Название *CASE*—средства | Фирма—производитель | *URL* |
|---|---|---|
| *Erwin* | *Computer Associates* | *http://www.cai.com/products/alm/erwin.htm* |
| *Designer 2000/ Designer 6i* | *Oracle* | *http://www.oracle.com/tools/designer* |
| *Power Designer* | *Sybase* | *http://www.sybase.com/products/designtools/powerdesigner* |
| *ER/Studio* | *Embarcadero Technologies* | *http://www.embracadero.com/products/Design/erdatasheet.htm* |
| *Visio Enterprise* | *Microsoft* | *http://www.microsoft.com/office/visio* |

Многие из этих продуктов предназначены не только для проектирования баз данных, но и для решения других задач, например, для моделирования потоков данных или бизнес—процессов, функционального моделирования, документирования, управления проектами.

# 3 Проектирование БД

## 3.1 Формирование внешнего уровня БД

Внешний уровень проектируемой БД представляет собой обобщенное представление отдельных представлений пользователей БД. Это обобщенное представление может быть оформлено в виде технической документации (технического задания), различных схем, рисунков, таблиц, словесного описания.

Для формирования внешнего уровня БД необходимо: провести обследование и анализ предметной области; осуществить сбор потребностей предприятия в рамках решения задачи автоматизации; определить размер разрабатываемой АИС, который может охватывать потребности, как отдельного подразделения, так и всего предприятия в целом. Необходимо увидеть и возможную тенденцию функционального развития АИС.

Существуют разные методы сбора материалов в ходе анализа предметной области. Это беседы и консультации с руководителями предприятия по вопросам проблем предприятия, стратегий его развития и управления; опросы исполнителей на рабочих местах по заранее разработанному перечню вопросов (анкетам), позволяющего выяснить порядок работ, отношение конкретных исполнителей к объекту автоматизации. Последовательно изучается весь деловой процесс в целом, далее — его составные части. Помимо всего, проектировщик должен своими глазами увидеть рабочий день исполнителя работ, функции которого автоматизируются. Это даст возможности определить технологию обработки данных, нормативы выполнения отдельных операций.

В ходе анализа предметной области необходимо получить представление об объекте автоматизации, системе управления, выявить какие функции системы управления будут автоматизированы. В настоящее время широким спросом пользуются задачи автоматизации функций учета и контроля состояния объекта управления на предприятии, всё более востребованной является автоматизация функции анализа. Важным является изучение организационной структуры предприятия, определение состава задач и функциональных подсистем, перечня входной и выходной информации в рамках разрабатываемой АИС.

Рассмотрим основные результаты, получаемые в ходе анализа предметной области и с помощью которых можно сформировать внешний уровень базы данных.

### 3.1.1 Обоснование целесообразности создания АИС

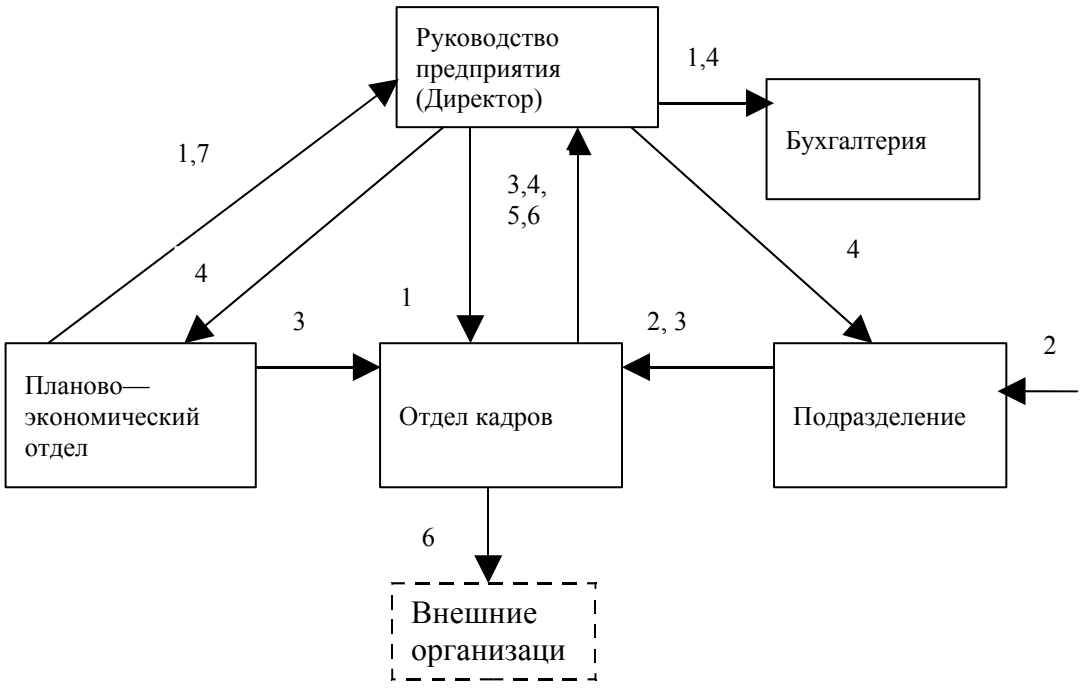
Прежде чем начать разработку автоматизированной информационной системы, необходимо определить какая цель при этом должна быть достигнута. Определяется перечень процессов, которые необходимо реализовать в рамках АИС, обосновывается целесообразность автоматизации этих процессов.

Под целью автоматизации подразумеваются некоторые характеристики, говорящие о достижении определенного экономического эффекта в сфере управления предприятием. Это могут быть: снижение стоимостных и трудовых затрат на осуществление производственного процесса, повышение оперативности его регулирования, повышение оперативности принятия решений и т.п.

Необходимо также сделать предварительный обзор и анализ аналогов подобных программных систем, существующих на огромном рынке современных программных продуктов и предназначенных для автоматизации подобных функций. Анализ должен дать положительный или отрицательный вывод о необходимости новой разработки.

### 3.1.2 Структура предприятия. Информационные потоки

Одна из важных задач при обследовании автоматизируемого предприятия – увидеть его организационно—кадровую структуру, информационные потоки, существующие между структурными единицами предприятия, выделить те, которые будут подлежать автоматизации. Пример структуры предприятия и информационных потоков, выявленных в ходе анализа предметной области в рамках подсистемы «Управление персоналом», представлен на рисунке 4.




1 – штатное расписание подразделения; 2 – заявление о приеме/увольнении; 3 – трудовой договор; 4 – приказ о приеме/увольнении; 5 – отчет о количественном составе контингента сотрудников; 6 – внешний отчет; 7 – отчет об исполнении штатного расписания

Рисунок 4 — Организационная структура предприятия. Информационные потоки

Кроме внутренних, в рамках решаемой задачи выявляются и внешние для предприятия информационные потоки. Это, как правило, нормативно − правовая информация, создаваемая государственными учреждениями в части законодательств, поток информации о конъюнктуре рынка, нормативные документы вышестоящих организаций и т.п. Поступающая на предприятие информация может влиять на обработку данных в АИС. Например, может быть выявлено, что в проектируемой БД должны быть отражены отраслевые стандарты, которые периодически обновляются и на основе которых формируется ряд внутренних распорядительных документов. На каждом предприятии существуют также информационные потоки, направленные от предприятия в различные внешние организации (вышестоящие организации, налоговые, пенсионные органы и т.п.).

### 3.1.3 Описание входных и выходных документов

Анализ информационных потоков позволяет определить состав входных и выходных документов, обрабатываемых и получаемых в рамках проектируемой АИС.

Описание входной и выходной информации состоит из перечня формируемых системой сообщений; структурных единиц информации; описания периодичности возникновения и сроков получения информации; наименований и структур каждого документа; перечней наименований и форматов качественных реквизитов и количественных показателей по каждой форме документа.

С форм входных документов осуществляется ввод информации в БД, формы выходных документов формируются на основе БД. При этом ввод информации может осуществляться механическим способом (ввод с клавиатуры), полуавтоматическим или автоматическим способом (с магнитных носителей, из оперативной памяти).

Например, для задачи «Приказы по перемещению сотрудников» подсистемы «Управление персоналом» входными являются документы: заявление сотрудника, трудовой договор, штатное расписание подразделения — данные с них вводятся в базу данных. Выходными — кадровый приказ, отчет по исполнению штатного расписания.

### 3.1.4 Функциональная структура АИС

Одной из важных задач в ходе анализа предметной области является определение масштаба разработки, выделение функций, объединение их в задачи, задач — в функциональные подсистемы. Состав функциональных подсистем разрабатываемой автоматизированной системы, во многом, определяется характером деятельности предприятия и его размером.

Например, функциональная подсистема «Управление персоналом» автоматизированной информационной системы предприятия может иметь деление на задачи и функции, представленное в таблице 3. В таблице

приведены примеры названий комплексов задач и задач, входящих в эти комплексы. Такая (по объему задач) подсистема может появиться на крупном предприятии. Возможно, на небольшом предприятии количество задач этой подсистемы может уменьшиться, так как, автоматизировать, например, функцию формирования и активизации в БД кадровых приказов будет не совсем целесообразно.

Таблица 3 – Состав функциональной подсистемы «Управление персоналом»

| Комплекс задач | Задача | Функция |
|---|---|---|
| Организационно—кадровая структура предприятия | Ведение структуры | Добавление, обновление<br>Поиск по критериям, просмотр |
| | Аналитические отчеты по структуре | Формирование, просмотр отчета.<br>Передача данных отчета в MS Word для печати |
| Штатное расписание (ШР) подразделений предприятия | Планирование ШР | Ведение справочников окладов, надбавок<br>Ведение справочника должностей<br>Формирование ШР подразделения<br>Добавление, обновление единиц вакансий ШР подразделения<br>Просмотр ШР заданного подразделения<br>Просмотр ШР всего предприятия в целом |
| | Анализ исполнения ШР | Формирование, просмотр отчета об исполнении ШР подразделения с выделением различных отклонений, просмотр.<br>Передача данных отчета в MS Word для печати. |
| Ведение данных сотрудников | Личная карточка сотрудника | Создание личной карточки<br>Добавление, обновление данных личной карточки<br>Поиск сотрудника по критериям, просмотр личной карточки |
| | Приказы по перемещению сотрудников (прием, перевод, увольнение) | Формирование приказа<br>Передача данных отчета в MS Word для печати.<br>Активизация приказа в БД после его подписи.<br>Поиск по критериям, просмотр приказа. |
| | Отчеты по количественному и качественному составу кадров предприятия. | Формирование отчета.<br>Передача данных отчета в MS Word для печати. |

Для каждой отдельной задачи должен быть определен состав входящих в неё функций. Обычно фрагмент БД разрабатывается для решения отдельной конкретной задачи и далее существует либо отдельно в виде локальной БД, либо интегрируется с другими фрагментами БД в рамках единой,

интегрированной базы данных корпоративной информационной системы предприятия.

В рамках одной задачи основная часть функций использует одни и те же данные или фрагменты данных, либо данные преобразуются функциями последовательно.

Удобно иерархическое представление выявленных функций. Иерархия функций позволяет отобразить функциональные зависимости автоматизируемых процессов. Пример функциональной иерархии задачи «Личная карточка сотрудника» приведен на рисунке 5.

| | | | |
|---|---|---|---|
| Соединение с БД | | | |
| Ведение справочных данных | | | |
| | Образование | Добавление/Обновление | Ф1 |
| | | Просмотр | Ф2 |
| | Ученое звание | Добавление/Обновление | Ф3 |
| | | Просмотр | Ф4 |
| | Ученая степень | Добавление/Обновление | Ф5 |
| | | Просмотр | Ф6 |
| | Типы адреса | Добавление/Обновление | Ф7 |
| | | Просмотр | Ф8 |
| | … | | |
| Личная карточка сотрудника | | Добавление/Обновление | Ф18 |
| Поиск сотрудников | (по критериям) | Просмотр/Печать | Ф19 |
| | | | Ф20 |
| Формирование документов | | | |
| | Справка по форме 1 | Формирование | Ф21 |
| | | Просмотр/Печать | Ф22 |
| | Справка по форме 2 | Формирование | Ф23 |
| | | Просмотр/Печать | Ф24 |

Рисунок 5 — Функциональная иерархия

На рисунке 5 все функции пронумерованы, эта нумерация может понадобиться в дальнейшем при проверке соответствия выявленной иерархии функций структуре полученной модели данных.

### 3.1.5 Выявление классов объектов и связей

Одним из важных этапов анализа предметной области является выявление и описание классов объектов (сущностей) и связей между ними. Описание может быть получено в произвольном виде, но для удобства процесса проектирования его формализуют в виде таблиц.

#### 3.1.5.1 Классы объектов

Как выявить в предметной области классы объектов? Класс объектов (тип сущности, сущность) – это значимая вещь, о которой предприятие должно хранить информацию. Признаки класса объектов:

а) объект, представляющий интерес для предприятия;

б) класс, категория, тип какой—то вещи;

в) именованное понятие;

г) существительное;

д) класс объектов есть, если есть реальный значимый объект;

Выявив класс объектов, необходимо дать ему имя. Оно должно быть уникальным. В качестве имени выбираются термины, используемые на предприятии. Имя изобретается, если все остальные возможности исчерпаны, так как придуманные имена могут привести к неправильному пониманию и дублированию. Имя должно быть согласовано с заказчиком. Имя может состоять из более чем одного слова (уточняющие имя слова – прилагательные и прочее). Часто одно и тоже называют одним именем, тогда необходимо выбрать одно главное имя, остальные описать как синонимы. При выявлении класса объектов выявляется группа вещей, состоящих из отдельных элементов (объектов). Класс объектов – это класс или категория вещей. Например, класс объектов «ОТДЕЛ» состоит из конкретных объектов «Учебно—методический отдел», «Отдел главного механика».

Все экземпляры (объекты) выявленного класса объектов должны однозначно определяться, быть идентифицированными. Если объект не может быть однозначно идентифицирован в классе объектов, то это возможно это не класс объектов вообще.

Этапы выявления и моделирования класса объектов:

а) исследование каждого существительного, выявленного в ходе анализа предметной области на предприятии и выявление его значимости;

б) выявление, имеется ли информация об этом существительном, которую необходимо хранить для данного предприятия;

в) присвоение имени классу объектов в единственном числе;

г) проверка, можно ли отличить один объект класса объектов от другого;

д) описание класса объектов для проверки того, что все (разработчики, заказчики) вкладывают в этот термин одно и тоже значение;

Для каждого класса объектов определяют его свойства (атрибуты сущности). Свойство – это конкретный элемент информации. Свойство

описывает класс объектов. Это качественное или количественное описание класса объектов. Свойство может выглядеть следующим образом:

— описательные слова, фразы (название, краткое название);

— предложные конструкции (сумма зарплаты для каждого сотрудника);

— притяжательные существительные и местоимения (признак устаревания, прочие).

Каждое свойство наделяется именем. Имена должны быть понятными и однозначными. Вопросы, которые рекомендуется задавать при выявлении свойств: какую информацию о классе объектов надо хранить; какую информацию о классе объектов надо выводить на экран или печать; нужно ли на самом деле это свойство.

Заказчики часто забывают о своих конкретных потребностях – думают, что поле само появится на экране или отчете, и не видят необходимости упоминать об этом.

Изучая существующую на предприятии документацию, необходимо обращать внимание на устаревшие требования прежних систем, например, старая форма выходного документа – членство в партии, национальность. Необходимо также отмечать производные и агрегированные данные, для каждого класса объектов фиксируются только исходные свойства. Производные и агрегированные свойства описываются отдельно и формируются, как правило, программой на основе значений исходных свойств. Необходимость хранения таких свойств бывает достаточно редкой.

К имени свойства предъявляется ряд требований. Имена должны быть понятными и однозначными, например название свойства "количество" может привести к путанице — возвращенное, поставленное? Необходимо выбирать более конкретные имена: «размер поставки», «объем заказа» и т.п. Если имя состоит из более одного слова, они разделяются пробелами.

Самый распространенный пример – свойство «дата». Если не указано конкретно, что это за дата, она может интерпретироваться как дата рождения, дата найма. Если необходимо хранить и то и другое, добавляется конкретное свойство, например, кроме даты найма необходимо еще хранить дату избрания по конкурсу. Это может быть выявлено на более поздних этапах анализа предметной области.

Выявленное в ходе анализа предметной области свойство необходимо разбить на мельчайшие компоненты, имеющие смысл. Уровень деления зависит от потребностей предприятия. Например, адрес, блочные размеры пиломатериалов (высота, длина, ширина), можно хранить и в виде одного свойства, но более полезно хранить в виде отдельных свойств.

Отличие класса объектов (сущности) от свойства (атрибута) приведено в таблице 4.

Таблица 4 — Отличия между классом объектов и свойством

| Характеристики класса объектов | Характеристики свойства |
|---|---|
| Вещь, о которой необходимо хранить информацию. | Квалифицирует класс объектов. |
| Имеет одно или более свойств. | Собственных свойств не имеет (в противном случае это класс объектов или комбинация свойств, которой дали неудачное название). |
| Может иметь несколько экземпляров и должна иметь значимую связь с другой класс объектов. | Для каждого экземпляра класса объектов имеет только одно значение. |

Определив свойство, необходимо убедиться, что для каждого конкретного объекта свойство может иметь одно единственное значение. Если обнаруживается более одного значения у какого—либо свойства, это свидетельствует о недостающем свойстве у класса объектов или кандидате в новый класс объектов.

Если обнаружится свойство, имеющее собственные свойства, то это не свойство, а класс объектов. Например, выявлен класс объектов «СПЕЦОДЕЖДА», имеющий свойства «номер», «название», «цвет». Если в ходе дальнейшего анализа и изучения соответствующего в предметной области документа, что цвет кроме названия должен иметь артикул, то «цвет» это уже не свойство, а класс объектов «ЦВЕТ» с соответствующими свойствами «название», «артикул».

Для каждого свойства необходимо определить его опциональность. Значение свойства может быть обязательным или необязательным при сохранении в базе данных информации о конкретном объекте. Такая характеристика называется опциональностью свойства. Обязательное значение свойства должно существовать и быть известным для каждого объекта рассматриваемого класса объектов. Необязательные значения свойства могут быть неизвестны (или не существуют) для какого—либо объекта на момент его создания. Например, значение свойства «дата начала работы» известно всегда для работающего сотрудника, а значение свойства «дата окончания работы» может быть неизвестным в заданный момент времени, если у сотрудника контракт бессрочный.

Для каждого свойства также выявляются в предметной области и описываются:

— формат (тип, максимальное длина, средняя длина (обычный размер), место десятичной точки, единица измерения;

— допустимые значения (диапазон, список значений, несколько диапазонов, значения по умолчанию);

При выявлении характеристик свойств могут быть определены и домены. Домен – это набор правил проверки с точки зрения бизнеса, ограничений, относящихся более чем к одному свойству.

С помощью домена можно задать: диапазон значений; список конкретных значений; несколько диапазонов; математическое уравнение;

значение по умолчанию и т.п. Эти правила описываются в БД один раз и применяются для разных свойств. Самый известный домен {да, нет}.

Существует следующая технология работы со свойствами, содержащая шаги:

— выявление кандидата в свойство;

— связывание свойства с классом объектов;

— присвоение имени свойству;

— определение формата свойства;

— определение опциональности свойства;

— определение логических ограничений свойства, накладываемых предметной областью (вхождение значения в диапазон и др.);

— проверка того, что это действительно свойство, а не класс объектов;

— в случае необходимости создание домена.

Для каждого класса объектов должны быть обязательно выявлены уникальные идентификаторы. Уникальный идентификатор – это свойство, совокупность свойств или комбинация свойств и связей, используемых для однозначной идентификации объекта в классе объектов. Свойство, входящее в состав уникального идентификатора должно иметь обязательную опциональность. Количество уникальных идентификаторов в классе объектов может быть каким угодно. И каким угодно может быть количество компонентов (свойств и связей), входящих в состав уникального идентификатора.

Уникальный идентификатор можно определять на любом этапе анализа предметной области, но чтобы приступить к описанию и проектированию класса объектов, необходимо, чтобы каждый класс объектов имел уникальные идентификаторы.

Выбирая способ идентификации объектов класса объектов, необходимо моделировать не технологические потребности разрабатываемой системы, а потребности бизнеса. При использовании в качестве уникального идентификатора числового кода необходимо убедиться, что в предметной области имеется соответствующий документ, в котором такой код отображается. Например, свойства «табельный номер сотрудника», «код подразделения» уже определены существующей на предприятии системой бухгалтерского учета, свойство «код должности» представлено в отраслевом классификаторе должностей и т.п.

На этапе проектирования БД уникальный идентификатор может быть сгенерирован технически, но во время анализа предметной области используются уникальные идентификаторы, используемые предприятием.

Если уникальных идентификаторов несколько, то необходимо определить среди них главный. Таким делается идентификатор, чаще используемый в бизнесе, например, «табельный номер». Либо любой уникальный идентификатор, имеющий наименьшую длину и числовой тип.

Самое большое количество уникальных идентификаторов имеет такой класс объектов, как «ФИЗИЧЕСКОЕ ЛИЦО/ ЧЕЛОВЕК». Каждый объект в таком классе объектов может быть однозначно идентифицирован такими

свойствами: «номер», «серия паспорта», «ИНН», «номер водительского удостоверения», «номер жетона». Для класса объектов «ДОЛЖНОСТЬ» могут быть выявлены следующие уникальные идентификаторы: «код», «название», «краткое название».

Необходимо отметить, что чем больше классов объектов будет выявлено в ходе анализа предметной области, тем более нормализованной затем будет структура реляционной базы данных. Почти любое существительное в предметной области имеет право быть определено как класс объектов, поскольку почти каждое существительное имеет, как минимум, набор из трех свойств: название объекта, краткое название объекта, числовой эквивалент названия объекта (код, номер, шифр).

Увидеть классы объектов предметной области можно подробно изучая информационные потоки предприятия, подлежащие автоматизации. Информационные потоки представлены документами. Любой документ является кандидатом в класс объектов. Документ имеет шапку, в которой, как правило, указано название документа и дата его формирования. Документ имеет информативную часть, в которой находятся качественные и количественные показатели. В нижней части документа находятся фамилии и должности лиц, подписывающих документ. На документе могут также быть расположены название, адрес и другие реквизиты предприятия, выпускающего документ. Таким образом, изучая документ, можно увидеть и выделить следующие классы объектов: «ПРЕДПРИЯТИЕ/ ЮРИДИЧЕСКОЕ ЛИЦО» или «СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ»; «ТИП СТРУКТУРНОЙ ЕДИНИЦЫ»; «АДРЕС»; «НАСЕЛЕННЫЙ ПУНКТ»; «ТИП НАСЕЛЕННОГО ПУНКТА»; «УЛИЦА»; «ТИП УЛИЦЫ» (улица, проспект, переулок, проезд и т.п.); «ДОКУМЕНТ»; «ПОЗИЦИЯ ДОКУМЕНТА»; «ФИЗИЧЕСКОЕ ЛИЦО»; «ДОЛЖНОСТЬ»; «ЗАПИСЬ О РАБОТЕ ФИЗИЧЕСКОГО ЛИЦА» (дата начала, дата окончания); «ТОВАР/ УСЛУГА»; «Объект» (учета).

Что представляет собой класс объектов «ПОЗИЦИЯ ДОКУМЕНТА»? Любой документ обычно имеет несколько позиций (позиции приказа, позиции прайс—листа, позиции счета—фактуры, записи учетной карточки и тому подобное). Таким образом, видна существующая в предметной области связь типа 1:М: «каждый ДОКУМЕНТ должен иметь одну или более ПОЗИЦИЙ»; с обратной стороны связь читается – «каждая ПОЗИЦИЯ ДОКУМЕНТА должна относиться к одному и тому же ДОКУМЕНТУ». Кроме того, каждая позиция документа имеет свои собственные свойства — номер, какие—то количественные показатели (количество учетных единиц, цена за единицу и другие).

Для каждой предметной области можно увидеть обязательный для всех предметных областей перечень классов объектов. Каждая предметная область, в широком смысле слова, отображает работу какого—либо предприятия или организации – производственного предприятия, учебного или лечебного учреждения, торговой организации, склада, пункта проката, домашней экономической сферы и так далее. Название (полное или краткое) предприятия или организации фигурирует в различных выходных документах и отчетах.

Таким образом, в предметной области присутствует класс объектов ПРЕДПРИЯТИЕ или СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ. Кроме того, зачастую необходимо вести учет адреса и телефона этого предприятия. В предметной области обязательно присутствуют физические лица, занимающие те или иные должности, своими подписями фиксирующие учет (приход или расход) какого—либо объекта. При чем, для решения задач анализа данных и принятия затем соответствующих управленческих решений, для предметной области представляет интерес хранения знаний об истории учета состояния того или иного объекта. И ещё одна категория обязательных для каждой предметной области классов объектов – это документы, на основании которых и происходят все процессы в заданной предметной области.

Свести все итоги анализа предметной области в ходе выявления классов объектов и их свойств можно в виде формализованного описания, таблицы. Пример такого описания приведен в таблице 5.

Таблица 5 — Формализованное описание предметной области. Классы объектов, свойства.

| Объект/ Свойство | Уникальный иденти— фикатор | Физические характерис— тики свойства (тип, длина) | Опциональ— ность свойства (Да/ Нет) | Логические ограничения свойства (диапазон значений, прописные, строчные буквы для символьных свойств и т.п.) | Процессы для значений свойств |
|---|---|---|---|---|---|
| ЧЕЛОВЕК | | | | | |
| таб.номер | У1, П | число, 10 | Да | > 0 | Г, Пр |
| ИНН | У2 | число, 12 | Нет | > 0 | Вв, Пр, Об |
| имя | | симв., 25 | Да | Перв. буква заглавн. | Вв, Пр, Об |
| дата рожд | | дата | Нет | ДД.ММ.ГГГГ | Вв, Пр, Об |
| ДОЛЖНОСТЬ | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

В таблице использованы сокращения: У – уникальный идентификатор, П – кандидат в первичный ключ (главный уникальный идентификатор), Г – генерация данных, Вв – ввод данных, Пр – просмотр данных, Об – обновление данных.

### 3.1.5.2 Связи между классами объектов

Поскольку всё в этом мире связано, то параллельным шагом в ходе анализа предметной области, вместе с выявлением классов объектов и их свойств, является шаг выявления связей, ассоциаций, возникающих между

классами объектов. Связи представляют информационные потребности и правила бизнеса на предприятии, их определение можно выразить следующим:

— именованная, значимая ассоциация между двумя классами объектов.

— отношение, которое имеет одна вещь к другой.

Рассматривая связь необходимо видеть её как двустороннюю, двунаправленную.

Например, класс объектов «КАТЕГОРИЯ ДОЛЖНОСТИ» связан с классом объектов «ДОЛЖНОСТЬ». Класс объектов «ДОЛЖНОСТЬ» связан с классом объектов «КАТЕГОРИЯ ДОЛЖНОСТИ».

Каждая связь обладает определенными характеристиками. Одна их характеристик — опциональность связи (минимальное кардинальное число). Это бизнес—правило, указывающее должна ли связь существовать для каждого объекта класса объектов (обязательная связь) или это не требуется (необязательная связь). Например, на предприятии выявлено следующее правило: «каждой конкретной категории должности может соответствовать должность». В некоторый момент времени на предприятии появляется документ о создании новой категории, но нет ещё ни одной должности, ссылающейся на эту категорию. Но с другой стороны есть и правило: «каждая должность на предприятии должна быть отнесена к одной и только одной должности». Таким образом, видно, что между двумя классами объектов («КАТЕГОРИЯ ДОЛЖНОСТИ» и «ДОЛЖНОСТЬ») выявлены две разные ассоциации.

Еще одна важная характеристика связи — мощность (максимальное кардинальное число). Это бизнес правило, указывающее, сколько таких связей существует – одна и только одна, или много. Если обнаружена связь, которая имеет мощность «ноль», эта связь необязательная.

Кроме того, каждая сторона связи имеет имя. Это описание правил бизнеса. Например: «соответствует», «относится к». Имена часто составляют пары: «основан на» – «является основой для»; «приобретается у» – «поставляется»; «отвечает за» – «находится под ответственностью».

Имя имеет большое значение, оно показывает, насколько хорошо понята взаимосвязь информации.

Увидев связь, необходимо убедиться в том, что она имеет смысл. Для этого её необходимо проговорить как обычное предложение в обе стороны (любая связь двусторонняя), используя правило произношения связи (таблица 6).

Таблица 6 – Правило чтения связи

| Часть 1 | Часть 2 | Часть 3 | Часть 4 | Часть 5 | Часть 6 |
|---|---|---|---|---|---|
| Каждый (ая, ое) | Имя первого класса объектов | Опциональность связи (д.б. или м.б.) | Имя связи | Мощность связи (одна или много) | Имя второго класса объектов |

Пример чтения связи: «каждое ФИЗИЧЕСКОЕ ЛИЦО может иметь ноль, одну или более ЗАПИСЕЙ ТРУДОВОЙ КНИГИ»; «Каждая ЗАПИСЬ

ТРУДОВОЙ КНИГИ должна относиться к одному и только одному ФИЗИЧЕСКОМУ ЛИЦУ».

Рассмотрим более подробно существующие типы (мощности) связей.

1 Связь «один_ко_многим» (1:M). Это самый распространенный тип связи, имеющей мощность один и более в одном направлении и один и только один в другом. Классы объектов, находящиеся в этой связи на стороне «один» называют главным или родительским. Класс объектов, находящийся на стороне «много» – подчиненным или потомком.

В большинстве случаев подчиненные классы объектов необязательны, а главные обязательны. То есть, объект главного класса объектов может существовать без подчиненного объекта, а подчиненный без главного нет. С точки зрения базы данных это означает, что сначала в БД создается объект главного класса объектов, а потом объекты подчиненного. Если связь 1:M не обязательная с обеих сторон, объекты могут создаваться произвольно. Связи 1:M, обязательные с обеих сторон, очень редки и означают, что объекты двух классов объектов не могут существовать друг без друга.

Пример связи 1:M: «каждой СТРУКТУРНОЙ ЕДИНИЦЕ ПРЕДПРИЯТИЯ может соответствовать ноль, одна или более ЗАПИСЕЙ ТРУДОВОЙ КНИГИ». С обратной стороны: «Каждая ЗАПИСЬ ТРУДОВОЙ КНИГИ должна относиться к одной и только одной СТРУКТУРНОЙ ЕДИНИЦЕ ПРЕДПРИЯТИЯ».

2 Связь «многие_ко_многим» (M:M или M:N). Это тоже очень распространенный тип связи, особенно на начальных этапах анализа предметной области. Эта связь имеет мощность «один или более» в обоих направлениях. Пример такой связи: «в каждой СТРУКТУРНОЙ ЕДИНИЦЕ ПРЕДПРИЯТИЯ могут работать много ФИЗИЧЕСКИХ ЛИЦ». С обратной стороны: «каждое ФИЗИЧЕСКОЕ ЛИЦО может работать во многих «СТРУКТУРНЫХ ЕДИНИЦАХ ПРЕДПРИЯТИЯ».

Большинство связей M:M необязательны в обоих направлениях, то есть объект одного класса объектов может существовать без привязки к объекту другого класса объектов, любой экземпляр может появиться первым. Связи M:M, обязательные с обеих сторон очень редки – объекты обоих классов объектов должны быть созданы одновременно.

Необходимо заметить, что в любой предметной области нет связей «многие_ко_многим», в каждый момент времени всё определяется однозначно. Появление такой связи в проектной документации показывает, что предметная область не дообследована. Связь M:M может быть «разорвана» каким—либо документом или позицией документа. Такой класс объектов, разрывающий связь M:M называют «сущностью пересечения». Необходимо только увидеть, найти этот класс объектов. Для выше приведенного примера связи M:M таким классом объектов является «ЗАПИСЬ ТРУДОВОЙ КНИГИ». Если мы его выявили, то связи в предметной области уже звучат так: «каждой СТРУКТУРНОЙ ЕДИНИЦЕ ПРЕДПРИЯТИЯ может соответствовать ноль, одна или более ЗАПИСЕЙ ТРУДОВОЙ КНИГИ». С обратной стороны: «каждая ЗАПИСЬ ТРУДОВОЙ КНИГИ должна относиться к одной и только

одной СТРУКТУРНОЙ ЕДИНИЦЕ ПРЕДПРИЯТИЯ». И ещё одна связь: «каждому ФИЗИЧЕСКОМУ ЛИЦУ, работающему на предприятии, может соответствовать ноль, одна или более ЗАПИСЕЙ ТРУДОВОЙ КНИГИ».

3 Связь «один_к_одному» (1:1). Редкая связь, обычно с точки зрения бизнеса это означает, что это не два класса объектов, а один. Эта связь может иметь мощность один и только один в обоих направлениях. Если обнаружится такая связь, следует ещё раз исследовать информационные потоки и может выясниться, что два выявленных класса объектов фактически составляют один.

Пример связи 1:1: «каждый ВЕЛОСИПЕД может использоваться только одним ЧЛЕНОМ КЛУБА». С обратной стороны: «каждый ЧЛЕН КЛУБА может ездить только на одном ВЕЛОСИПЕДЕ»

Связи 1:1, обязательные на обоих концах, когда оба объекта должны появляются одновременно, очень редки.

После выявления любой связи между классами объектов необходимо (для каждой её стороны): установить наличие; выбрать имя; определить мощность; определить опциональность; проверить путем чтения.

Необходимо заметить, что между двумя классами объектов может быть выявлено сколько угодно много связей. Например, между классами объектов «ФИЗИЧЕСКОЕ ЛИЦО» и «АДРЕС» может быть выявлено 2 связи: одна фиксирующая адрес прописки, другая – адрес проживания. Свести итоги выявления связей можно с помощью следующей таблицы (таблица 7):

Таблица 7 — Формализованное описание предметной области. Связи между классами объектов

| Связь | | Опциональность связи со стороны | | Название связи со стороны | | Тип связи со стороны | |
|---|---|---|---|---|---|---|---|
| Главный КО | подчинен ный КО | главного КО | подчинен ного КО | главного КО | подчинен ного КО | главного КО | подчинен ного КО |
| ФИЗИ— ЧЕСКОЕ ЛИЦО | АДРЕС | Д.б. | Д.б. | пропи— сано по | является местом прописки | 1 | 1 |
| ФИЗИ— ЧЕСКОЕ ЛИЦО | ЗАПИСЬ ТРУДО— ВОЙ КНИГИ | М.б. | Д.б. | имеет | соответ— ствует | 1 | М |
| ТРУДО— ВАЯ КНИГА | ЗАПИСЬ ТРУДО— ВОЙ КНИГИ | Д.б. | Д.б. | имеет | соответ— ствует | 1 | М |
| ДОЛ— ЖНОСТЬ | ЗАПИСЬ ТРУДО— ВОЙ КНИГИ | М.б. | Д.б. | имеет | соответ— ствует | 1 | М |

В таблице использованы следующие сокращения: КО – класс объектов; Д.б. – должна быть, М.б. – может быть.

Выявленные связи проверяются путем чтения. Необходимо помнить, что каждая связь двусторонняя!

### 3.1.6 Неформализованное описание предметной области

В процессе анализа также необходимо зафиксировать бизнес правила или семантические (смысловые) утверждения, ограничивающие предметную область в рамках решаемой задачи. Это не функции предприятия, как таковые, а непреложные факты, которым всегда должна подчиняться разрабатываемая автоматизированная информационная система.

Примеры семантических утверждений:

— «На работу принимаются служащие, достигшие 16—ти летнего возраста»;

— «Любой сотрудник не может отвечать одновременно более чем за десять сдаваемых в аренду или продаваемых объектов недвижимости»;

— «Любой сотрудник не имеет права продавать или сдавать в аренду свою собственную недвижимость»;

— «Специальные скидки не распространяются на автомобили возрастом менее одного года»;

— «Общая сумма скидок не может превышать 40% чистой суммы, указанной в счете—фактуре».

Выявленные семантические утверждения записываются на естественном языке и должны быть далее отражены в БД. Как правило, подобные правила реализуются с помощью таких объектов БД, как триггеры, процедуры, просмотры (представления).

### 3.1.7 Уровни доступа пользователей

На этапе анализа предметной области кроме функций системы, классов объектов предметной области, их связей также должны быть определены потенциальные пользователи автоматизированной информационной системы и БД.

Для баз данных выделяют следующие категории пользователей.

1 Проектировщики БД. В функции проектировщиков БД входит анализ предметной области, формирование требований к разрабатываемой БД; выбор методов, моделей, средств для проектирования БД. Сферой их деятельности является разработка информационной модели предметной области, разработка логической и физической структур базы данных.

2 Администратор данных (АД). Администратор данных должен разбираться в данных и понимать нужды предприятия на уровне высшего руководства предприятием. В обязанности администратора данных входит принятие решений о том, какие данные необходимо вносить в БД в первую очередь; обеспечение поддержки порядка при обслуживании данных после их занесения в БД. Например, он должен указывать, кто, при каких условиях, над какими данными и какие операции может выполнять. Таким образом, администратор данных должен формировать политику безопасности данных. Администратор данных работает как управляющий БД, он может не быть

специалистом по техническим вопросам, хотя он должен иметь хорошее представление о возможностях баз данных на техническом уровне.

3 Администратор базы данных (АБД). Это технический специалист, ответственный за реализацию решений администратора данных и проектировщиков БД. АБД должен быть профессиональным специалистом в области информационных технологий. Работа АБД заключается в установке и настройке среды СУБД, создании БД, её администрировании и техническом контроле. Администратор базы данных физически, средствами СУБД реализует политику безопасности данных.

4 Прикладные программисты. Они отвечают за написание прикладных программ, использующих БД. Для этих целей применимы все современные языки высокого уровня. Прикладные программы выполняют над данными все стандартные операции через соответствующие запросы к БД в рамках прав определенных АД и назначенных АБД. Функции прикладных программ — поддержка работы конечного пользователя.

5 Конечные пользователи. Это пользователи, которые непосредственно работают с БД через прикладную программу или в режиме диалога, используя встроенный интерфейс СУБД. Права доступа в рамках решаемой задачи им выделяет прикладной программист. Различают опытных конечных пользователей, которые могут создать и реализовать *SQL* или *QBE* запросы, и наивных конечных пользователей, которые иногда даже и не представляют о том, что работают с базой данных.

При определении уровней доступа пользователей выделяют 3 понятия:

— субъект (пользователь), который выполняет операцию доступа;

— объект операции доступа (данные – объекты БД, в частности таблицы, поля, записи);

— вид операции доступа: операции над данными – чтение *Read (R),* добавление *Insert (I)*, обновление – *Update (U)*, удаление – *Delete (D)*, выполнение хранимых данных процедур – *Execute (E)*.

В теории БД выделяют следующие уровни доступа.

1) для разных категорий пользователей:

а) неограниченный доступ ко всем объектам БД и их поколениям;

б) неограниченный доступ к группе объектам БД и их поколениям;

в) ограниченный доступ к группе объектов БД и их поколениям;

2) на уровне реляционной таблицы различают следующие виды доступа:

а) вся таблица доступна для всех операций;

б) вся таблица закрыта для всех операций;

в) только чтение всей таблицы;

г) чтение всей таблицы, изменение только части;

д) чтение с изменением только одной записи;

е) чтение только одной записи;

ж) чтение и изменение отдельных столбцов;

з) чтение отдельного столбца;

и) одни столбцы читать, другие изменять;

к) вышеуказанные способы доступа с ограничением по времени;

л) вышеуказанные способы доступа при выполнении определенных условий;

м) выполнение вычислительных операций над данными без права их чтения и изменения.

Для разработки небольших баз данных, как правило, определяют следующих пользователей и уровни доступа.

1 Администратор БД. Является фактическим владельцем БД и имеет полный доступ на создание БД и объектов БД; добавление, обновление, удаление, просмотр данных. Его привилегией также является реализация выдачи прав другим пользователям БД.

2 Прикладной программист. Имеет полный доступ к объектам БД на уровне подсхемы БД. Подсхему создает АБД в схеме БД для решения поставленной задачи и выделяет необходимый доступ к подсхеме прикладному программисту. Как правило, это полный доступ к подсхеме. В ней прикладной программист создает необходимые для решения задачи объекты БД.

3 Конечный пользователь. Работает с БД через приложение, разработанное прикладным программистом, либо непосредственно в среде СУБД. Наделяется правами прикладным программистом, если он на это уполномочен АБД, либо непосредственно администратором БД.

Определение уровней доступа пользователей может быть формализовано уже на этапе анализа предметной области. На дальнейших этапах проектирования БД уровни доступа уточняются и затем реализуются АБД средствами СУБД. В таблице 8 приведены примеры доступа пользователей подсистемы «Управление персоналом» на уровне классов объектов.

Таблица 8 — Уровни доступа пользователей подсистемы "Управление персоналом"

| Класс объектов | Пользователи | | | | | |
|---|---|---|---|---|---|---|
| | Прикл—аной програ—ммист | Конечные пользователи | | | | |
| | | Инспектор отдела кадров | Руководи—тель отдела кадров | Сотрудник планового отдела | Руководи—тель планового отдела | Руководи—тель предпри—ятия |
| СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ | *RIUDE* | *R* | *R* | *RIU* | *R* | *R* |
| КАТЕГОРИЯ ДОЛЖНОСТИ | *RIUDE* | *R* | *R* | *RIU* | *R* | *R* |
| ДОЛЖНОСТЬ | *RIUDE* | *R* | *R* | *RIU* | *R* | *R* |
| ФИЗИЧЕСКОЕ ЛИЦО | *RIUDE* | *RIU* | *R* | *R* | *R* | *R* |
| КАДРОВЫЙ ПРИКАЗ | *RIUDE* | *RIU* | *R* | *R* | *R* | *R* |
| ЗАПИСЬ КАДРОВОГО ПРИКАЗА | *RIUDE* | *RIU* | *R* | *R* | *R* | *R* |
| ТИП ПЕРЕМЕЩЕНИЯ | *RIUDE* | *R* | *R* | *RIU* | *R* | *R* |

В таблице использованы сокращения основных операций над данными.

Существуют некоторые особенности определения и реализации уровней доступа пользователей, осуществляющих обработку данных на основе

статистических методов, особенно при получении агрегированных данных о больших подмножествах БД. При применении одной и той же функции несколько раз для разных составов подмножеств БД можно получить интересующие данные путем сравнения результатов запросов. В таких случаях, для более тонкого разграничения уровней доступа отдельных пользователей, получающих права на выполнение чтение данных с использованием различных методов, возможно планирование и реализация следующих мероприятий:

— ограничение на структуру запроса;

— разрешение поиска по части ключа;

— разрешение реализации определенного количества запросов;

— установление минимального числа записей участвующих в запросах;

— ведение протоколов выполнения запросов.

Таким образом, уже на этапе анализа предметной области должны быть определены уровни доступа всех потенциальных пользователей разрабатываемой БД.

## 3.2 Разработка концептуального уровня БД

Целью данного этапа является последовательная разработка концептуальной информационно—логической модели предметной области, отражающей логику информации предприятия и даталогической модели базы данных.

### 3.2.1 Инфологическая модель предметной области

Исходными данными для построения ИЛМ предметной области являются результаты анализа предметной области, представленные в виде описания классов объектов и связей между ними. Чаще всего ИЛМ предметной области представляют в терминах семантической модели данных, в виде *ER* — диаграммы предметной области.

Необходимо отметить, что выявление в предметной области классов объектов, связей, описание и отображение их в диаграмме происходит параллельно.

#### 3.2.1.1 Методологии построения ER—диаграмм

В настоящее время существуют разнообразные методологий (нотации) построения *ER*—модели.

1 Методология Питера Чена. В 1976 году Питером Ченом была предложена семантическая модель "сущность—связь" — *ER*—модель, которая в настоящее время стала самой распространенной.

Соглашения, используемые при изображении диаграммы:

— классы объектов отображаются прямоугольником, свойства эллипсами, связи ромбами;

— уникальный идентификатор (первичный ключ) отображается в виде эллипса, обведенного двойной линией;

— мощность связи «один» отображается линией, «много» — линией со стрелкой.

Особенности этой методологии:

— метод позволяет показать связь между двумя, тремя и более классами объектов (сущностями);

— связь может иметь собственные атрибуты;

— нет возможности отображения взаимоисключающих связей и непереносимости связей;

— взаимоисключающие связи неявно реализуются в виде супертипов и подтипов;

— нельзя выразить опциональность атрибутов и связей.

На рисунке 6 приведен пример фрагмента *ER*—диаграммы в методологии Питера Чена.

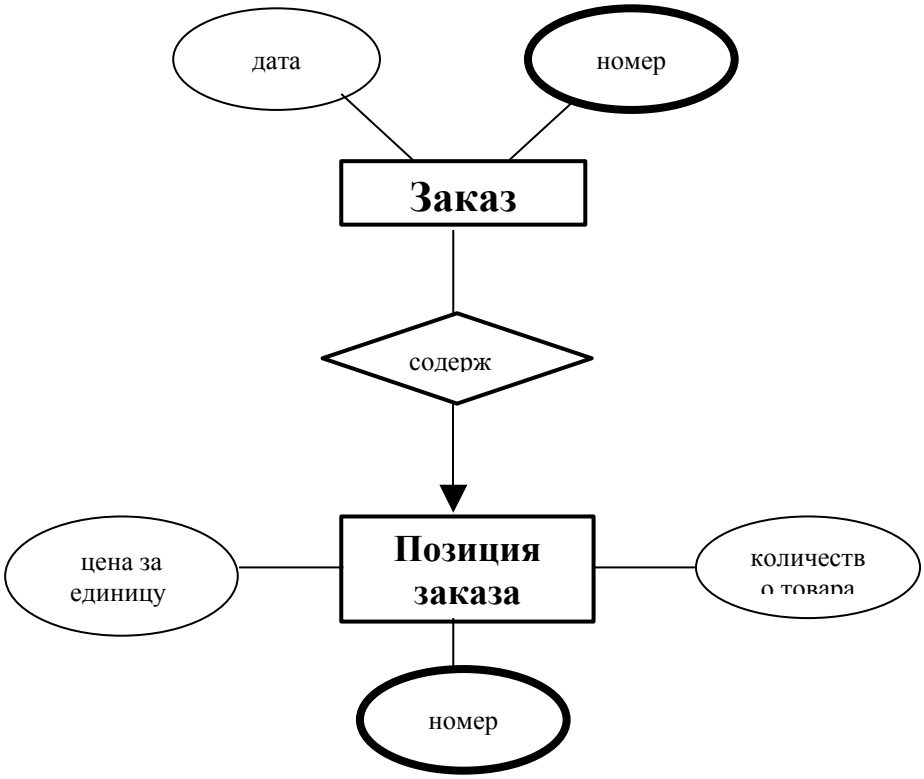



Рисунок 6 — Пример фрагмента *ER*—диаграммы в методологии Питера Чена

На диаграмме отображены следующие бизнес—правила предприятия: «Каждому заказу, имеющему такие свойства как номер и дата, должна соответствовать одна или более позиций заказа, имеющей такие свойства, как номер, цену за единицу товара, количество товара»; с другой стороны – «Каждая позиция товара должна относиться к одному и только одному заказу».

Необходимо отметить, что в примере приведен фрагмент описания предметной области. В ней также должны существовать такие классы объектов, как «Товар», «Единица измерения» и другие.

2 Методология *IDEF1*. Используется в *CASE*—средствах *ERwin, Design/IDEF*. В методологии используются следующие соглашения:

— каждому классу объектов присваивается уникальное имя и номер;

— обязательная связь отображается сплошной линией, необязательная – пунктирной;

— мощность связи "один" отображается линией, "много" – точкой;

— связь может дополнительно определяться с помощью указания мощности (типа) связи. Мощность может принимать следующие значения: *N* – ноль, один или более (принимается по умолчанию); *Z* – ноль или один, *P* – один или более.

— свойства класса объектов отображаются в виде списка имен внутри блока, отображающего класс объектов;

— атрибуты первичного ключа изображаются вверху и отделяются от других.

Пример представления *ER*—диаграммы в методологии *IDEF1* приведен на рисунке 7.

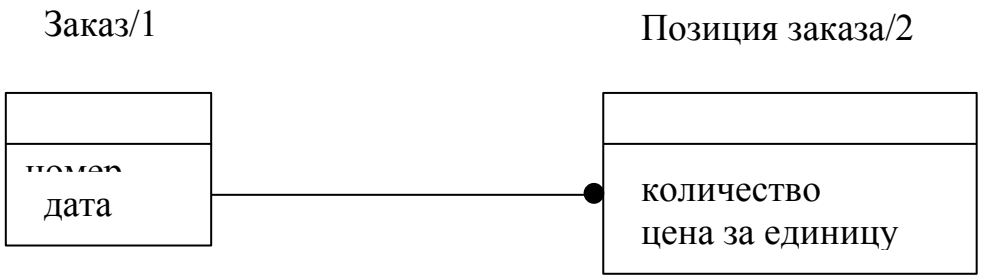



Рисунок 7 — Пример представления *ER*—диаграммы в методологии *IDEF1*.

На рисунке 7 отображена та же ситуация в предметной области, что и на рисунке 6.

3 Методология Ричарда Баркера. Используемые в методологии элементы: класс объектов, свойство класса объектов, уникальные идентификаторы, опциональность свойств, связи, мощность (тип), опциональность и переносимость связей, уникальность объекта из связи, супертипы, подтипы, арки.

Используются следующие соглашения:

— класс объектов отображается в виде четырехугольника с закругленными углами. Имя класса объектов указывается внутри четырехугольника, это имя существительное в единственном числе, отображенное заглавными буквами;

— свойства записываются внутри четырехугольника, отображающего класс объектов строчными буквами, это имя существительное в единственном числе;

— четырехугольник, отображающий класс объектов, можно увеличивать до любых размеров, четырехугольники могут быть разных размеров;

— опциональность свойств помечается: обязательное свойство – звездочкой (*), необязательное – кружочком (о);

— уникальный идентификатор помечается #, если уникальных идентификаторов несколько, тогда каждый помечается номером, указанным в скобках, например, # (1), #(2);

— обязательная связь помечается сплошной линией, необязательная – пунктирной;

— тип (мощность) связи «один» помечается линией, «много» — «вороньей лапой».

Более сложные элементы, используемые в *ER*—диаграмме, построенной по методологии Ричарда Баркера, рассмотрим далее в примерах.

### 3.2.1.2 Шаблоны моделирования

Любая рассматриваемая предметная область имеет свои особенности. Но в тоже время обладает и общими для всех предметных областей элементами. Так, в основной массе решаемых задач автоматизации обязательно фигурируют такие классы объектов, как предприятие (организация), структурная единица предприятия (цех, отдел, факультет, отделение и т.п.), люди, или физические лица, разного рода материальные объекты. Все процессы, происходящие в предметной области и которые необходимо учитывать в базе данных, осуществляются на основе документов, которые в свою очередь фиксируют сбор, перемещение, расход каких—либо данных. Таким образом, многие ситуации можно смоделировать, применяя существующие шаблоны. Мы рассмотрим шаблоны моделирования на примере построения фрагментов *ER* − диаграмм по методологии Ричарда Баркера.

1 Моделирование семейного положения. Например, ситуацию предметной области, описанную следующими предложениями: «каждое ФИЗИЧЕСКОЕ ЛИЦО (мужского пола) может являться супругом другого ФИЗИЧЕСКОГО ЛИЦА (женского пола)» и, с обратной стороны, «каждое ФИЗИЧЕСКОЕ ЛИЦО (женского пола) может являться супругой другого ФИЗИЧЕСКОГО ЛИЦА (мужского пола», можно смоделировать, используя рекурсивную связь. Это связь между объектами одного класса объектов. Такая связь может обладать всеми свойствами, присущими любой другой связи. Пример приведен на рисунке 8. На рисунке 8 изображена рекурсивная связь, имеющая с обеих сторон одинаковый тип («один») и опциональность «необязательная». В методологии Ричарда Баркера на *ER*—диаграмме можно отображать и названия связей.


Рисунок 8 — Пример рекурсивной связи

2 Моделирование иерархии данных. Иерархия данных наблюдается, если в модели предметной области присутствует:

— произвольное число иерархий классов объектов;

— одинаковые свойства у классов объектов, входящих в иерархию;

— связи между такими классами объектов одинаковые.

На рисунке 9 представлен фрагмент *ER*—диаграммы, выполненный по методологии Ричарда Баркера и отображающий пример иерархии данных.

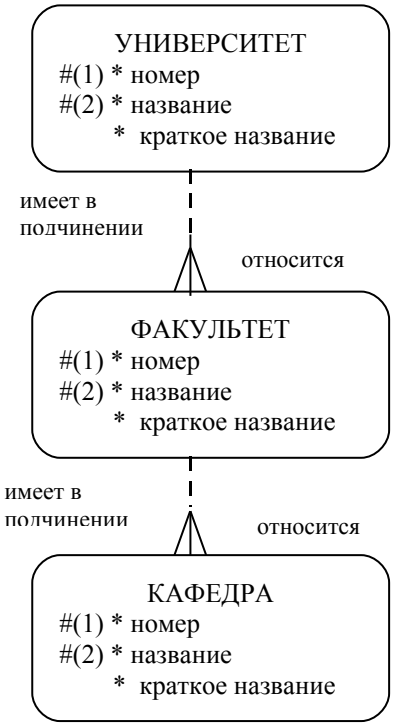



Рисунок 9 — Пример иерархии данных

На любом предприятии организационная структура обычно бывает иерархическая. Так и в этом примере представлена организационная структура высшего учебного заведения. Классы объектов отображают структурные единицы вуза, располагающиеся на разных уровнях иерархической структуры подчинения. Классы объектов имеют одинаковые свойства, между классами объектов, расположенных на разных уровнях иерархии присутствуют одинаковые связи. Прочитав фрагмент такой предметной области, можно убедиться в её адекватности. Однако такое представление имеет некоторый недостаток – при добавлении ещё одного уровня иерархии, например,

добавления структурной единицы «Лаборатория» в подчинение какой—либо кафедре потребует добавления ещё одного класса объектов в модель, то есть любое изменение в организационной структуре предприятия потребует корректировки модели.

Для моделирования подобной иерархии данных можно использовать шаблон модели — рекурсивную связь. При этом рекурсивная связь должна иметь тип 1:M и должна быть необязательной в обоих направлениях. Сторона «один» отображает правило «имеет в подчинении», сторона «много» — «подчиняется». Самый верхний элемент иерархии никому «не подчиняется», самый нижний элемент никого «не имеет в подчинении». Использование шаблона позволяет добавлять и удалять уровни иерархии в соответствии с требованиями предметной области, не меняя базовую модель.

Замена иерархии данных рекурсивной связью осуществляется по следующему алгоритму:

— создается один класс объектов, содержащий свойства, присущие каждому классу объектов в иерархии данных;

— классу объектов присваивается общее имя (в иерархии подчинения подразделений предприятия это может быть, например, класс объектов с именем «СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ»;

— создается дополнительный класс объектов, который будет отображать название для отличия каждого узла иерархии данных, например, класс объектов «ТИП СТРУКТУРНОЙ ЕДИНИЦЫ ПРЕДПРИЯТИЯ».

Как вывод, необходимо сделать следующие замечания:

— шаблон имеет один недостаток: классы объектов, находящиеся на всех уровнях иерархии должны иметь одинаковые свойства;

— иерархия, смоделированная как рекурсивная связь, должна быть необязательной в обоих направлениях. Обязательная ветвь, направленная вверх или вниз, создает бесконечную иерархию, не имеющую применения в реальном мире;

— если при преобразовании модели предметной области происходит слияние частей диаграммы в одну, то необходимо найти в предметной области класс объектов «ТИП», который позволит отобразить уникальность каждой части

Пример использования шаблона, моделирующего иерархию данных, приведен на рисунке 10.

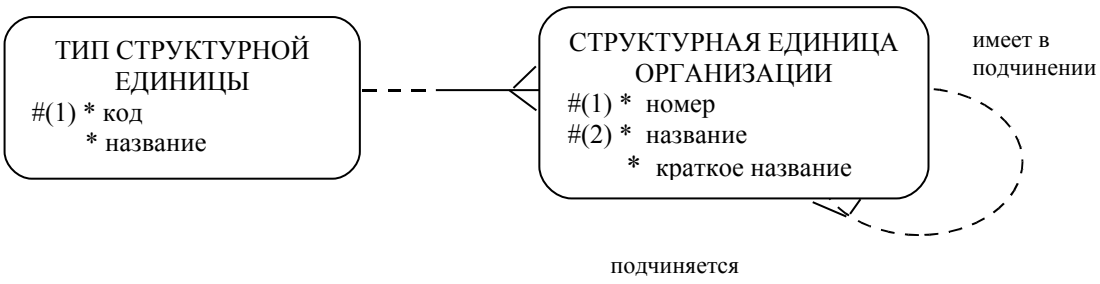



Рисунок 10 – Пример использования шаблон для моделирования иерархии данных.

3 Разрыв связей M:M. Наличие связи M:M в *ER* — диаграмме допустимо, но необходимо помнить, что это не адекватное отображение предметной области, есть предметная область не дообследована. Необходимо найти класс объектов (сущность), который разорвет такую связь. Как правило, это какой—либо документ, или позиция документа. Например, связь «многие ко многим» между классами объектов «ПОСТАВЩИК» и «ТОВАР» («каждый ПОСТАВЩИК может поставлять много ТОВАРОВ» и «каждый ТОВАР может поставляться разными ПОСТАВЩИКАМИ») может быть разорвана с помощью таких классов объектов, как «ПОЗИЦИЯ НАКЛАДНОЙ», «ПОЗИЦИЯ ПРАЙС — ЛИСТА», «ПОЗИЦИЯ ДОГОВОРА» и другие. На рисунке 11 приведен пример разрыва связи M:M. В роли поставщика в примере выступает юридическое лицо.

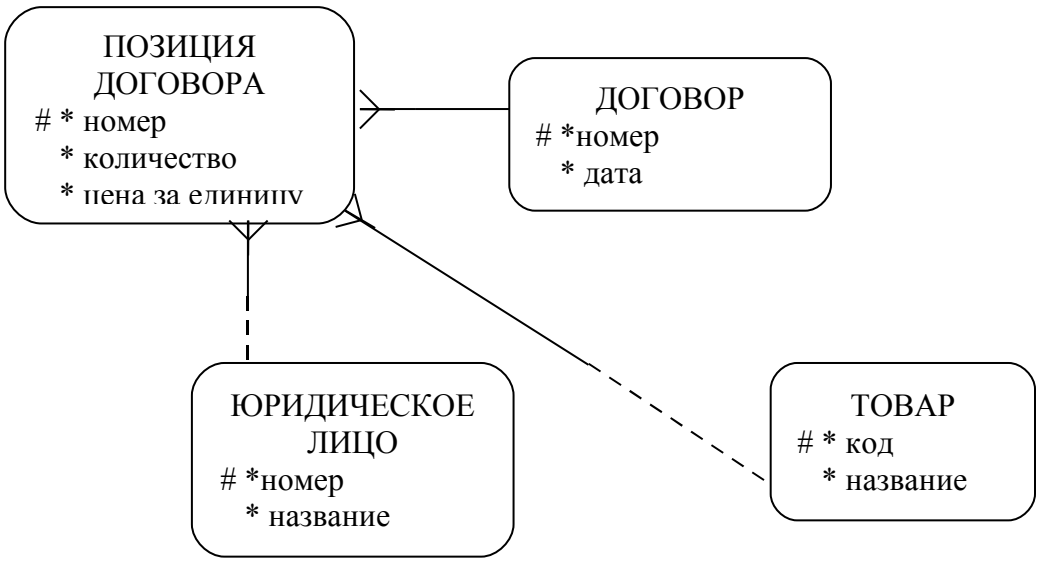



Рисунок 11 — Разрыв связи M:M

Необходимо отметить, что классы объектов, разрывающие связь M:M, как правило, содержат свойства, значения которых динамически меняется. Это такие свойства, как «количество», «цена».

4 Моделирование ролей. Под ролями человека или организации в предметной области понимаются должности, обязанности, прозвища. В разных классах объектов, представляющих роли объекты могут перекрывать, дублировать друг друга. Например, класс объектов «ВРАЧ» и класс объектов «ПАЦИЕНТ» отображают разные роли человека – один лечит, другой лечится. Но в случае, если врач становится пациентом в той же предметной области, то информация о нем должна быть отображена в классе объектов «ПАЦИЕНТ», два объекта в двух классах объектов будут информационно дублировать, перекрывать друг друга. Роли должны моделироваться с помощью связей, необходимо чтобы объект одного и того же класса объектов мог выступать в нескольких ролях.

Пример неправильного моделирования ролей приведен на рисунке 12, правильного – на рисунке 13.

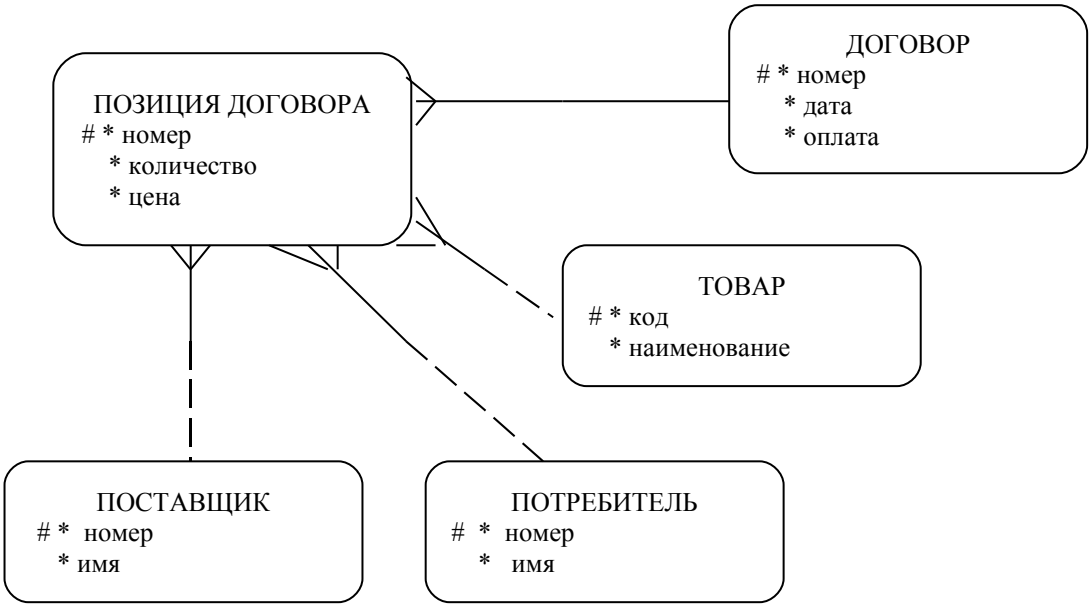



Рисунок 12 — Неправильное моделирование ролей

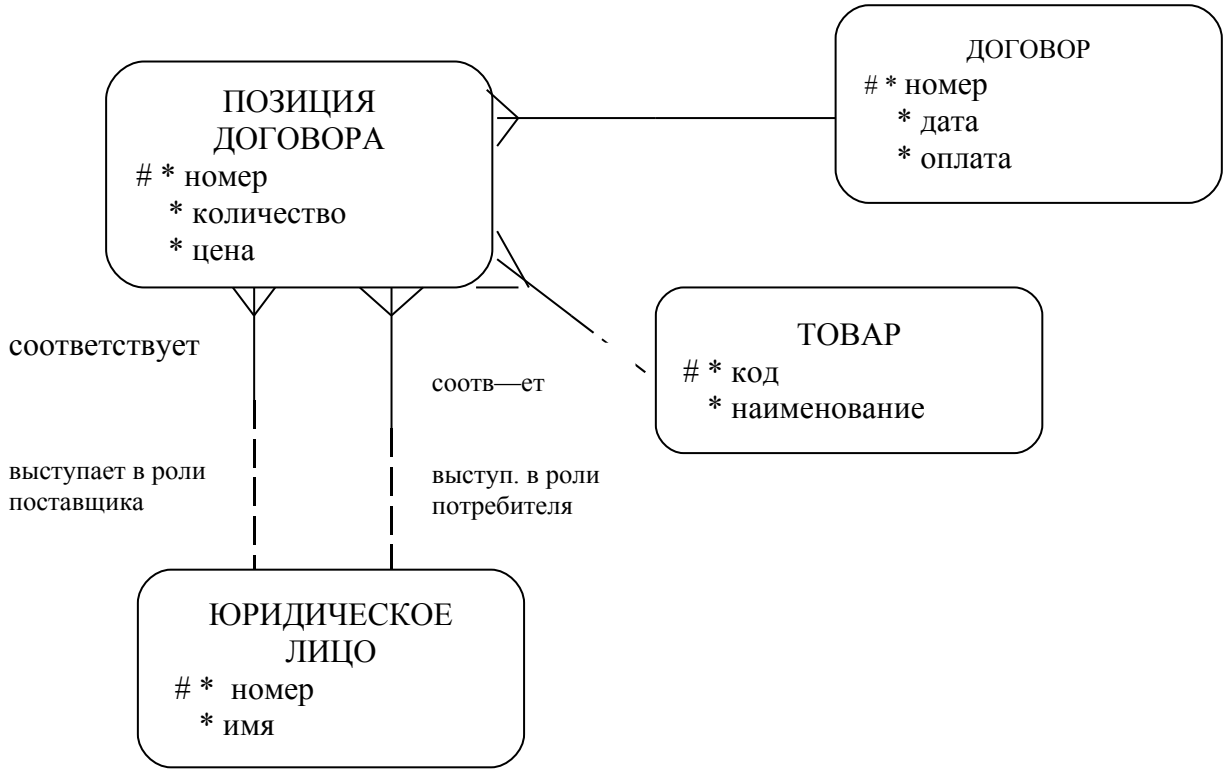



Рисунок 13 — Правильное моделирование ролей

На рисунке 12 классы объектов «ПОСТАВЩИК» и «ПОТРЕБИТЕЛЬ» выделены отдельно. При возникновении ситуации, что какое—то юридическое лицо станет выступать как в роли поставщика, так и в роли потребителя, модель будет неадекватно отображать предметную область – информация будет продублирована. Правильным моделированием ситуации будет выделение одного класса объектов «ЮРИДИЧЕСКОЕ ЛИЦО», а роли «поставщик» и «потребитель» отобразить в виде соответствующих связей (рисунок 13).

Примеры предметных областей, где необходимо моделировать роли, приведены в таблице 9.

Таблица 9 — Примеры моделирования ролей.

| Предметная область | Неправильное моделирование | Правильное моделирование |
|---|---|---|
| Купля—продажа, поставка товара | Классы объектов: ПОКУПАТЕЛЬ, ПРОДАВЕЦ, ПОСТАВЩИК | Классы объектов: ЮРИДИЧЕСКОЕ ЛИЦО или ФИЗИЧЕСКОЕ ЛИЦО. Связи (роли): покупает, продает, поставляет |
| Образовательное учреждение, обучение | Классы объектов: АБИТУРИЕНТ, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, АСПИРАНТ | Классы объектов: ФИЗИЧЕСКОЕ ЛИЦО, РАБОТА ФИЗИЧЕСКОГО ЛИЦА, ОБУЧЕНИЕ ФИЗИЧЕСКОГО ЛИЦА, ТИП ОБУЧЕНИЯ ФИЗИЧЕСКОГО ЛИЦА, ТИП ПЕРЕМЕЩЕНИЯ ФИЗИЧЕСКОГО ЛИЦА. Связи (роли): сдает документы, работает, обучается. |
| Документооборот | Классы объектов: ВХОДЯЩИЙ ДОКУМЕНТ, ИСХОДЯЩИЙ ДОКУМЕНТ, ПРИКАЗ, РАСПОРЯЖЕНИЕ | Классы объектов: ДОКУМЕНТ, ПОЗИЦИЯ ДОКУМЕНТА, ТИП ДОКУМЕНТА, ТИП ПЕРЕМЕЩЕНИЯ ДОКУМЕНТА. Связи (роли): относится (к типу) |

На рисунке 14 приведен фрагмент *ER*—диаграммы, отображающей предметную область «Управление персоналом». Класс объектов «ПОЗИЦИЯ ПРИКАЗА О ПЕРЕМЕЩЕНИИ» отображает сведения о перемещениях сотрудников (физических лиц) на предприятии, класс объектов «ВИД ПЕРЕМЕЩЕНИЯ» — виды кадровых перемещений – прием, перевод, увольнение и тому подобное. Между классами объектов «ПРИКАЗ О ПЕРЕМЕЩЕНИИ» и «ПОЗИЦИЯ ПРИКАЗА О ПЕРЕМЕЩЕНИИ» присутствуют три связи, две из них – моделируют роли:

— «каждый ПРИКАЗ О ПЕРЕМЕЩЕНИИ должен быть подписан одним сотрудником, являющимся начальником отдела кадров, о чем есть соответствующая информация в классе объектов ПОЗИЦИЯ ПРИКАЗА О ПЕРЕМЕЩЕНИИ», начальник отдела кадров может подписывать много приказов»;

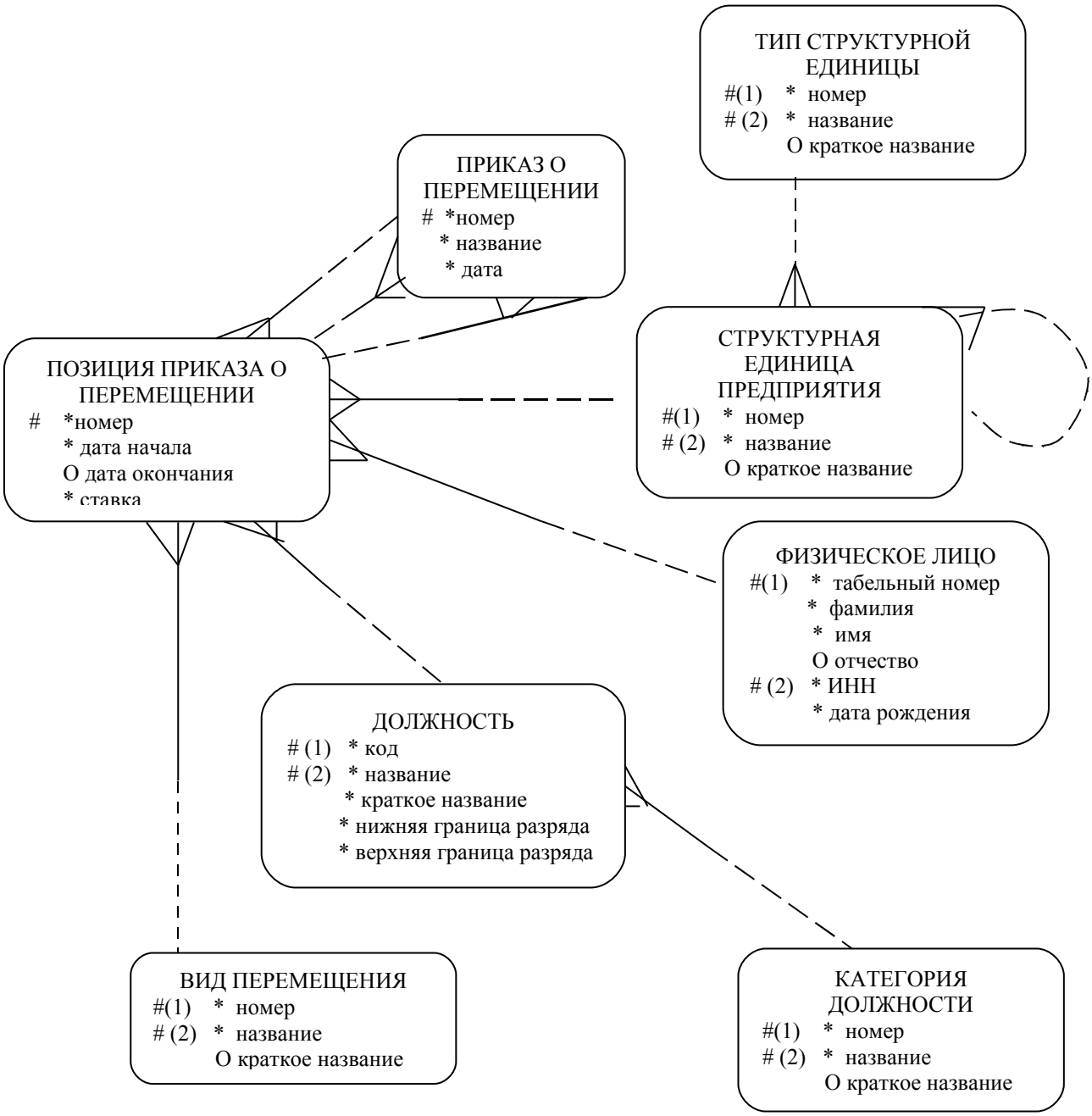



Рисунок 14 — Пример моделирования ролей

— «каждый ПРИКАЗ О ПЕРЕМЕЩЕНИИ должен быть подписан одним сотрудником, являющимся руководителем предприятия, о чем есть соответствующая информация в классе объектов «ПОЗИЦИЯ ПРИКАЗА О ПЕРЕМЕЩЕНИИ», руководитель предприятия может подписывать много приказов».

Представленный на рисунке 14 фрагмент описания предметной области можно назвать шаблоном, который может быть использован для отображения ситуации, когда какие—либо документы подписываются должностными лицами и в базе данных необходимо отслеживать историю – кто и когда из

физических лиц и когда, находясь в той или иной должности, визировал тот или иной документ. Это важно, поскольку документы в предметной области отображают, как правило, перемещение (приход, расход) материальных и не материальных объектов (приход, расход товаров на склад, перемещение кадров, движение контингента больных, учет выпущенных в эфир передач и тому подобное). В данной предметной области должно поддерживаться следующее семантическое утверждение: «Дата подписываемого приказа должна быть более поздней, чем дата приказов, в которых определены роли подписывающих приказ лиц».

### 3.2.1.3 Моделирование сложных структур

Моделирование сложных структур – это моделирование взаимоисключающих классов объектов, связей, моделирование данных во времени.

1 Моделирование взаимоисключающих классов объектов. Взаимоисключающие классы объектов моделируются с помощью супертипов и подтипов. Супертип – это класс объектов, который делится на взаимоисключающие подгруппы меньшего размера. Супертип может иметь собственные свойства или просто использоваться как имя группы. Подтип – это класс объектов, представляющий разбитую группу в рамках супертипа. Каждый подтип неявно наследует все свойства и связи супертипа, кроме того, он может иметь свои собственные свойства и связи. Подтипы должны быть взаимоисключаемыми, то есть объект одного подтипа не может быть объектом другого подтипа. Свойства и связи, общие для подтипов можно описывать на уровне супертипов. Уникальные идентификаторы, выявленные для супертипа, наследуются подтипом. Важное замечание: подтип – это не наследование, это эксклюзивность!

Подтипы могут быть вложенными. Для ясности модели рекомендуется ограничиваться 2—3 уровнями вложения.

Рассмотрим описание фрагмента предметной области «Прокат видеофильмов и видеоигр». Необходимо хранить информацию о продуктах проката. Каждый продукт характеризуется кодом, названием, текстовым описанием. Для фильмов необходимо хранить такие свойства как категория фильма (код, название), длительность. Для игры – возраст ограничения, количество игроков. Кроме того, необходимо учитывать прочие продукты проката, но пока свойства их неизвестны.

Для описания фрагмента данной предметной области можно использовать супертип, объединяющий общие атрибуты и подтипы, отражающие эксклюзивность отдельных продуктов проката.

На рисунке 15 представлено отображение предметной области в виде супертипа «ПРОДУКТ ПРОКАТА» и подтипов «ФИЛЬМ», «ИГРА», «ПРОЧЕЕ». Подтип «ФИЛЬМ» имеет собственную связь с классом объектов «КАТЕГОРИЯ». Каждый подтип, входящий в супертип может быть указан в

позиции документа, с помощью которого фиксируется выдача продуктов проката.

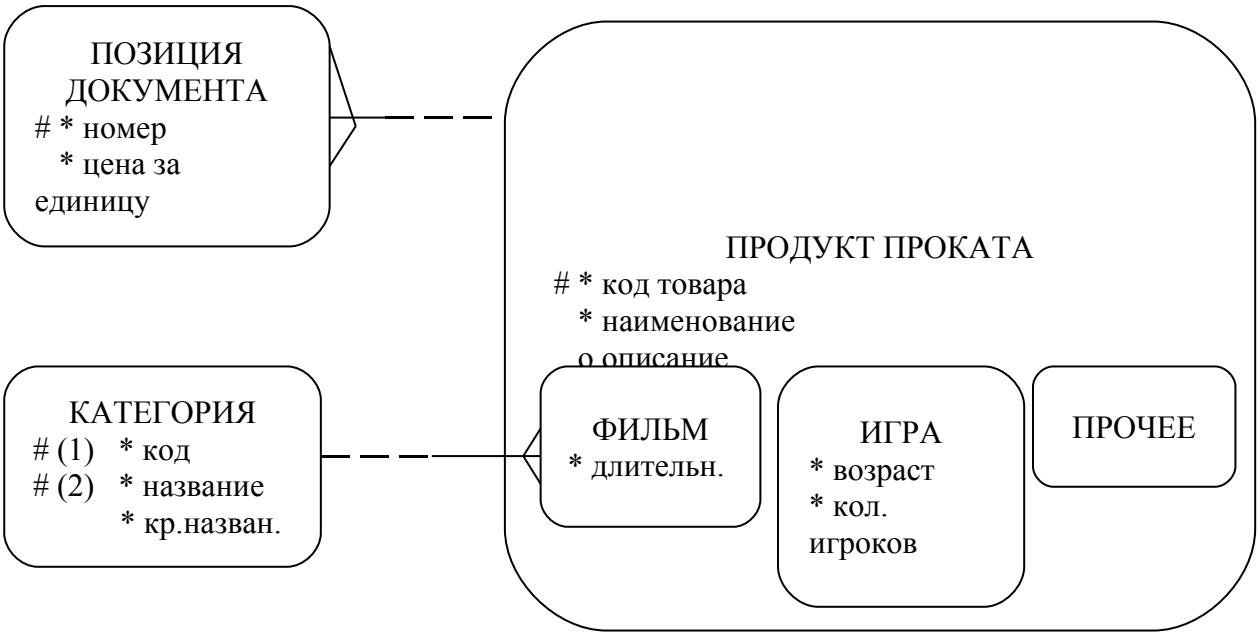



Рисунок 15 — Пример супертипа и подтипов

При чтении *ER*—диаграммы подтипы читаются «или – или»: «в каждой ПОЗИЦИИ ДОКУМЕНТА может быть отражен или ФИЛЬМ или ИГРА или ПРОЧИЙ продукт проката».

2 Моделирование взаимоисключающих связей. Взаимоисключающая связь – это такая ситуация, когда класс объектов имеет связь либо с классом объектов А, либо с классом объектов В. Обе связи могут быть действительными, но в разные моменты времени. Взаимоисключаемость связей моделируется с помощью арка. Арк – это элемент *ER*—диаграммы, построенной по методологии Ричарда Баркера, изображается в виде дуги, пересекающей входящие в арк взимоисключающие связи. Связи, входящие в арк, помечаются кружочком. *CASE* — средство это делает автоматически. Пример использования арка приведен на рисунке 16.

На рисунке изображено типовое представление адреса в предметной области. Связи, входящие в арк читаются с использованием союзов или—или, либо—либо. Например: «каждый АДРЕС должен относиться либо к ФИЗИЧЕСКОМУ, либо к ЮРИДИЧЕСКОМУ ЛИЦУ».

Правила использования арка:

— все концы связей в арке должны иметь одну и ту же опциональность, если это не так, то эксклюзивность связей не различается;

— связь может входить только в один арк;

— количество связей в арке может быть любым;

— связи в арке часто имеют одинаковые имена;

— связи, входящие в арк, должны идти от одного и того же класса объектов.

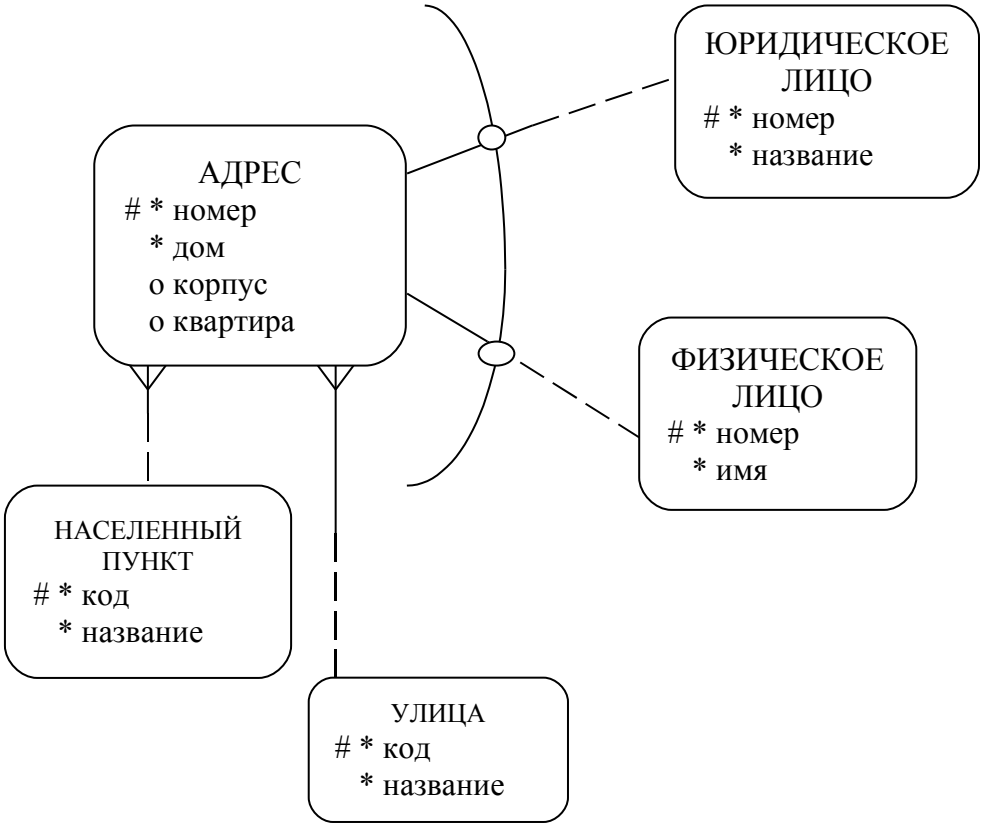



Рисунок 16 — Пример использования арка.

Для моделирования взаимоисключаемости могут быть использованы арки, подтипы, сочетание первого и второго. Например, бизнес правило «каждый Членский БИЛЕТ должен принадлежать либо ОРГАНИЗАЦИИ, либо ФИЗИЧЕСКОМУ ЛИЦУ», отображаемое с помощью арка, может быть несколько интерпретировано и отображено с помощью супертипа и подтипов: «каждый ЧЛЕНСКИЙ БИЛЕТ выдается КЛИЕНТУ, который может быть либо ФИЗИЧЕСКИМ ЛИЦОМ, либо ОГРАНИЗАЦИЕЙ».

Можно пользоваться любым удобным методом при условии, что он позволит правильно отразить потребности автоматизируемого предприятия.

3 Моделирование данных во времени. Для того чтобы в рамках автоматизированной информационной системы можно было выполнять анализ данных, необходимо иметь возможность хранения в БД данных, привязанных к конкретному времени. Для этого в модели предметной области должен быть соответствующий класс объектов, отображающий документ предметной области, имеющий соответствующее свойство «дата».

Пример использования классов объектов для отображения данных во времени приведен в таблице 10. Курсивом выделены классы объектов и их свойства, отображающие данные во времени.

Таблица 10 — Пример классов объектов, отображающих данные во времени

| Предметная область | Классы объектов | Комментарий |
|---|---|---|
| Аренда недвижимости | ФИЗИЧЕСКОЕ ЛИЦО, ЮРИДИЧЕСКОЕ ЛИЦО, ПОМЕЩЕНИЕ, *ДОГОВОР АРЕНДЫ (номер, дата начала, дата окончания)* | Договор аренды заключается либо с физическим, либо с юридическим лицом. |
| Управление персоналом | СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ, ФИЗИЧЕСКОЕ ЛИЦО, ДОЛЖНОСТЬ, ПРИКАЗ, *ЗАПИСЬ ПРИКАЗА О ПЕРЕМЕЩЕНИИ (номер, дата начала, дата окончания, ставка)*, ТИП ПЕРЕМЕЩЕНИЯ | Перемещение – это перемещение сотрудника в организации – прием на работу, перевод, увольнение, избрание по конкурсу, уход на службу в ряды российской армии и т.п.<br>В каждом приказе может быть несколько записей. |
| Складской учет | СТРУКТУРНАЯ ЕДИНИЦА ПРЕДПРИЯТИЯ, ТОВАР, ДОКУМЕНТ, *ПОЗИЦИЯ ДОКУМЕНТА (номер, количество),* ТИП ДОКУМЕНТА, ЕДИНИЦА ИЗМЕРЕНИЯ | Тип документа – это тип, отражающий приход или расход того или иного товара.<br>В каждом документе может быть несколько позиций. |

4 Переносимость связей. Связь между классами объектов отображает связь между конкретными объектами классов объектов. Переносимость связи – это возможность перемещения связи с одного объекта класса объектов на другой объект того же класса объектов. Связи могут быть переносимыми и непереносимыми.

Рассмотрим фрагмент *ER*— диаграммы предметной области «Управление персоналом» — рисунок 17.

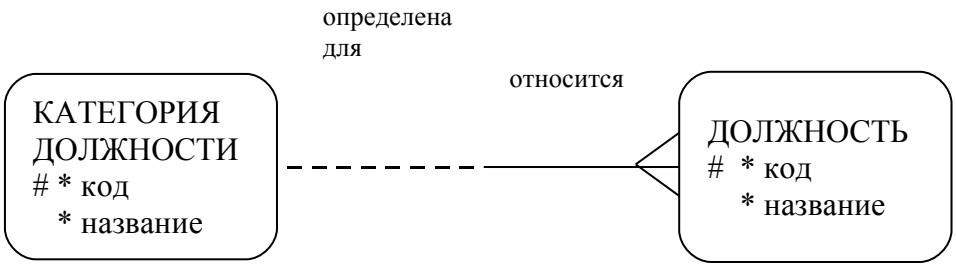



Рисунок 17 — Пример фрагмента *ER*— диаграммы

На рисунке отражена ситуация соответствия должностей какой—либо категории. Например, должность "программист" относилась к категории "учебно—вспомогательный персонал", затем стала относиться к категории "вычислительный центр". Необходимо отразить эту ситуацию, разрешить осуществить этот перенос, при этом хранить информацию о том, когда это произошло не надо.

Переносимыми считаются все связи, за исключением тех, которые специально объявляются непереносимыми. Непереносимость связи – это бизнес – правило, ограничивающее перенос связи – логическое ограничение на связь, определяемое предметной областью.

Фрагмент *ER*—диаграммы с примером непереносимости связи представлен на рисунке 18.

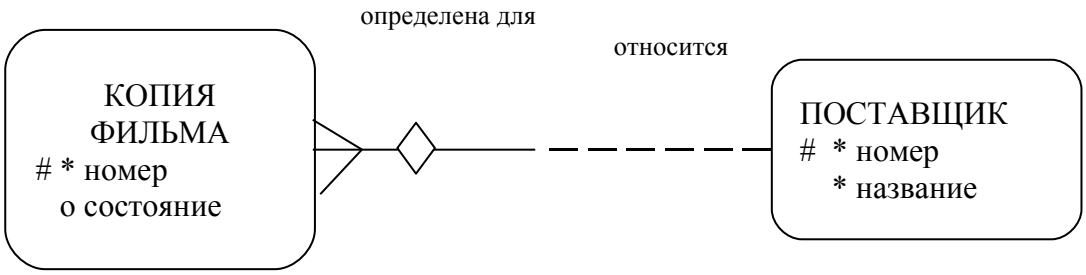



Рисунок 18 — Пример непереносимости связи.

Бизнес правило, отраженное на рисунке 18: «каждая КОПИЯ ФИЛЬМА поставляется конкретным ПОСТАВЩИКОМ. Если КОПИЯ ФИЛЬМА была поставлена, то информация об этом измениться не может».

Непереносимость связи помечается ромбом на стороне подчиненного класса объектов. Физически это означает, что в БД в последующем должно быть запрещено изменение значения внешнего ключа, реализующего эту связь.

Прежде чем объявлять связь непереносимой, необходимо убедиться, что это правило хорошо понимается.

Непереносимыми также считаются связи, уникально идентифицирующие подчиненный класс объектов — рисунок 19. На рисунке представлен фрагмент описания предметной области:

— «каждое ЗАДАНИЕ должно быть поручено только одному СЛУЖАЩЕМУ»;

— «каждый СЛУЖАЩИЙ может выполнять несколько ЗАДАНИЙ»;

— «каждое ЗАДАНИЕ должно входить только в один ПРОЕКТ»;

— «каждый ПРОЕКТ может включать несколько ЗАДАНИЙ».

Класс объектов «ЗАДАНИЕ» имеет свойство "дата назначения", которое входит в уникальный идентификатор. Каждый объект из класса объектов «ЗАДАНИЕ» дополнительно уникально идентифицируется связями с классами объектов «СЛУЖАЩИЙ» и «ПРОЕКТ». Однозначно его можно определить, зная дату назначения, табельный номер служащего и номер проекта.

Уникальная идентификация объекта из связи необходима, если в предметной области не определяется явный уникальный идентификатор класса объектов, например, он не представлен явно ни в одном документе (например, задание было устным), или такой документ пока не найден.

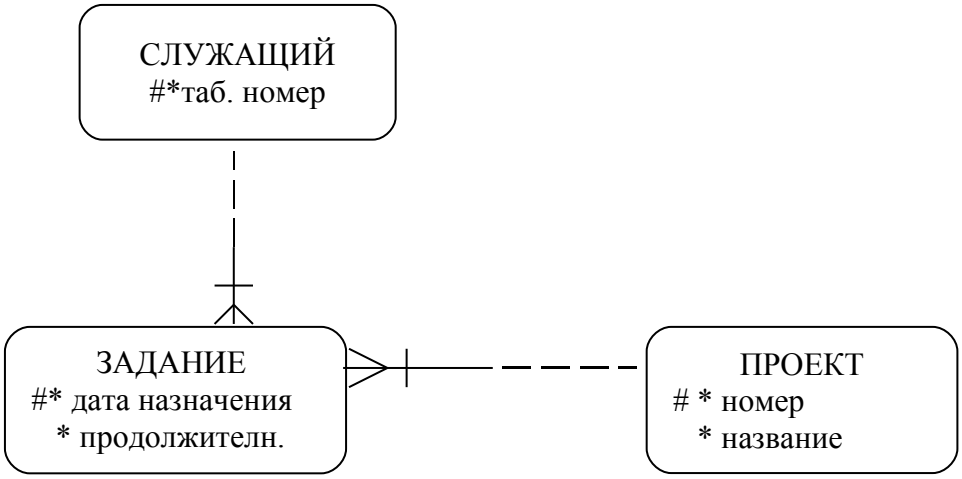



Рисунок 19 — Уникальность класса объекта из связи.

Уникальность класса объектов из связи помечается вертикальной линией на связи.

### 3.2.1.4 Проверка законченности ER—диаграммы

При построении модели с использованием методологии Ричарда Баркера, необходимо, соблюдая четкость и аккуратность, придерживаться следующего:

— размер четырехугольников по вертикали, отображающих классы объектов, сопоставлять с количеством включенных в него свойств;

— линии связи рисовать прямыми и направленными вверх, по горизонтали и по диагонали; для диагональных линий использовать углы в 30 или 60 градусов, это упрощает чтение, если связи пересекаются;

— избегать большого количества параллельных линий, их трудно отслеживать;

— избегать сокращений и жаргонов в названиях классов объектов, свойств, добавлять к названиям прилагательные для облегчения понимания модели;

— имена связей указывать на концах линий с разных сторон от неё;

— для упрощения чтения диаграммы рекомендуется располагать так классы объектов, чтобы "воронья лапа", обозначающая связь «много» была направлена вверх и влево. Таким образом, самые динамичные и объемные классы объектов будут всегда расположены ближе к верхнему, левому углу диаграммы;

— нежелательно вычерчивать диаграмму на сетке.

*ER*—диаграмма должна легко читаться как непрерывное предложение, начиная с любого места, и иметь смысл с точки зрения бизнеса.

Для того чтобы получить законченную и качественную *ER*—диаграмму, построенную по методологии Ричарда Баркера необходимо убедиться в следующем.

1 Для классов объектов: представлены четырехугольниками с закругленными углами; название в единственном числе, заглавными буквами; обязательно имеют уникальные идентификаторы.

2 Для свойств классов объектов: имена записаны строчными буквами и не включают имя класса объектов; имеют одну из меток «*» (обязательное свойство) или «о» (необязательное свойство). Все свойства разбиты на атомарные компоненты.

3 Для подтипов: полностью описывают класс объектов; не перекрывают друг друга; существование каждого подтипа оправдано – имеют разные свойства, разные связи.

4 Для связей: каждая сторона имеет имя (пишется строчными буквами), мощность и опциональность. Для больших моделей необязательно указывать названия связей на диаграмме, но они должны быть обязательно указаны в формализованном описании предметной области. Необходимо проверить опциональность связей: для обязательных связей – действительно ли связь должна существовать, то есть объект не может быть создан без одновременного создания связи; для необязательных связей – действительно ли связь только может существовать, может ли объект существовать без этой связи.

5 Для рекурсивных связей: рекурсивные связи имеют необязательное значение с обеих сторон (кроме рекурсивных подтипов).

6 Для арков: связи, входящие в арк имеют одинаковый тип и опциональность с обеих сторон и помечены кружками.

7 Все ли выявленные в классе объектов свойства имеют одно значение (рисунок 20). В противном случае необходимо дообследовать предметную область и выявить необходимые классы объектов.

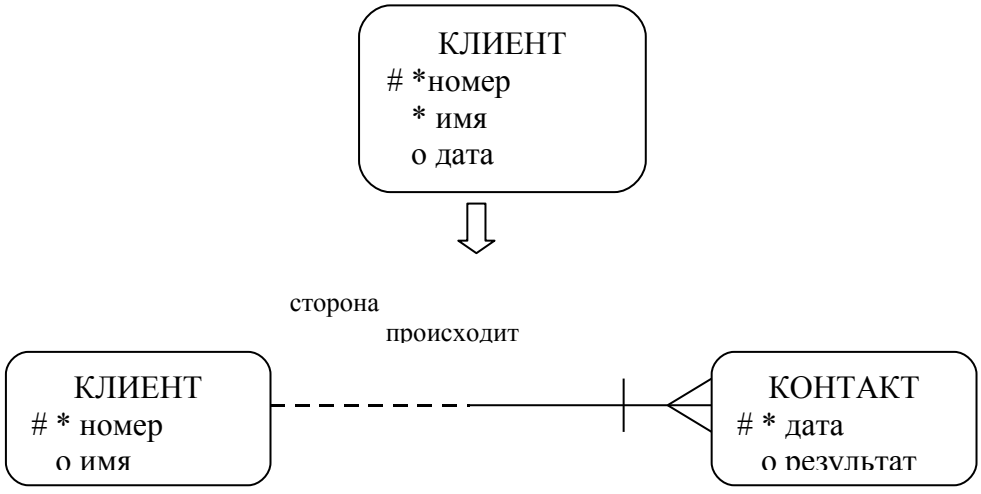



Рисунок 20 — Пример дообследования предметной области

8 Каждое ли свойство класса объектов зависит от всего уникального идентификатора этого класса объектов — рисунок 21. Чем больше отдельных существительных предметной области выделено в виде классов объектов, тем более нормализованной будет схема будущей реляционной БД.

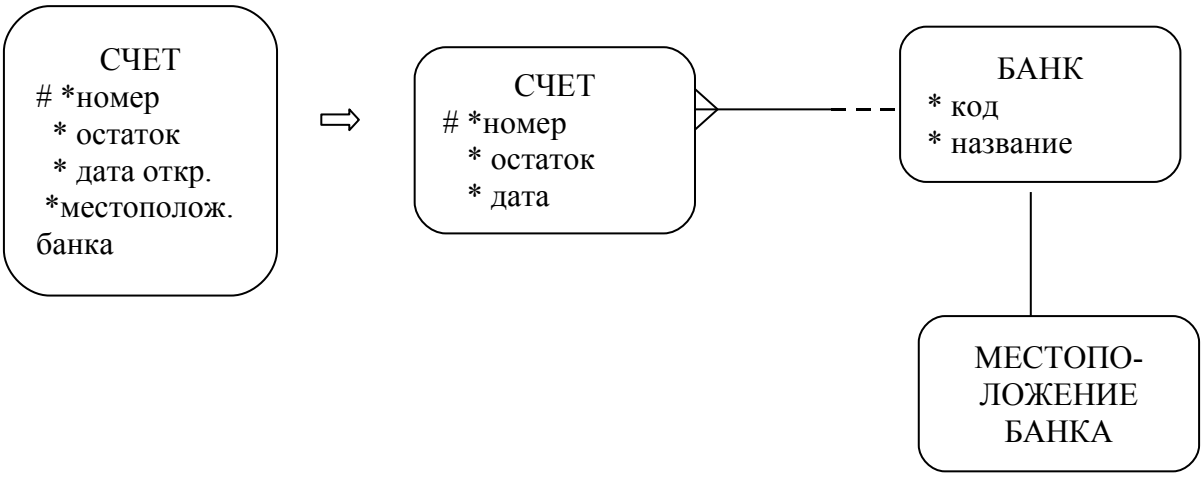



Рисунок 21 — Нормализация на уровне ER—диаграммы

*ER*—диаграмма должна соответствовать формализованному описанию предметной области, представленному в виде таблиц.

### 3.2.1.5 Перекрестная проверка модели данных и иерархии функций

На этапе проектирования автоматизированной информационной системы важно осуществлять проверку на то, что проект базы данных содержит все данные, необходимые для реализации приложения, а проект приложения включает модули, которые используют все проектируемые данные. Такую проверку называют перекрестной. Перекрестная проверка может осуществляется на уровне полученной модели предметной области.

Рассмотрим перекрестную проверку для предметной области «Аренда помещений». На рисунке 22 приведена модель предметной области, представленная в виде *ER*—диаграммы, построенной по методологии Ричарда Баркера. *ER* – диаграмма отображает ситуацию заключения договоров на аренду помещений. Договор может быть заключен либо с юридическим, либо с физическим лицом. На рисунке 23 представлена иерархия функций, функции для удобства проведения проверки пронумерованы.

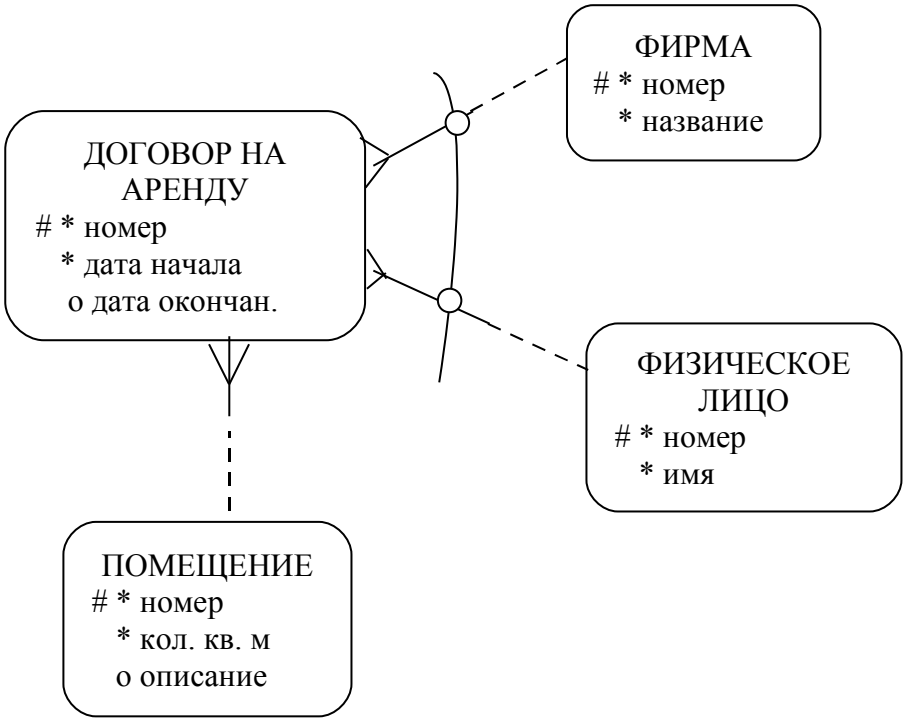



Рисунок 22 — Пример модели предметной области

| | | | |
|---|---|---|---|
| Ведение справочных данных | Фирма | Добавление/ Обновление | Ф1 |
| | | Просмотр | Ф2 |
| | Физ. лицо | Добавление/ Обновление | Ф3 |
| | | Просмотр | Ф4 |
| | Помещение | Добавление/ Обновление | Ф5 |
| | | Просмотр | Ф6 |
| Аренда помещений | Формирование договора | Добавление | Ф7 |
| Отчетные документы | Отчет1. Список арендованных физическими лицами помещений за заданный период времени | Формирование/ Просмотр | Ф8 |
| | Отчет2. Список арендованных фирмами помещений за заданный период времени | Формирование/ Просмотр | Ф9 |

Рисунок 23 — Пример иерархии функций

Перекрестная проверка может быть формализована в виде таблицы (таблица 11).

Таблица 11 — Пример формализации перекрестной проверки

| Функции | Классы объектов | | | |
|---|---|---|---|---|
| | ПОМЕЩЕНИЕ | ДОГОВОР | ФИРМА | ФИЗ.ЛИЦО |
| Ф1 | | | *I, U* | |
| Ф2 | | | *R* | |
| Ф3 | | | | *I, U* |
| Ф4 | | | | *R* |
| Ф5 | *I, U* | | | |
| Ф6 | *R* | | | |
| Ф7 | | *I* | | |
| Ф8 | *R* | *R* | | *R* |
| Ф9 | *R* | *R* | *R* | |

В таблице использованы сокращения названий функций: *I* – добавление данных (*insert*); *U* – обновление данных (*update*); *R* – чтение данных (*read*).

Анализируя таблицу 10, можно отметить, что каждой функции соответствует хотя бы один класс объектов, при этом для реализации функции чтения данных (для формирования отчета) в модели присутствует необходимое количество классов объектов. С другой стороны, каждый класс объектов, отображенный в модели, необходим для реализации хотя бы одной функции.

### 3.2.2 Даталогическая модель БД

Построение даталогической модели базы данных – это следующий этап проектирования базы данных методом «нисходящего» проектирования. Исходными данными для даталогического проектирования является модель предметной области. В результате должна быть получена логическая структура базы данных, описанная в терминах выбранной модели данных на основе физических записей. В настоящее время распространение получила реляционная модель данных, на рынке программного обеспечения наиболее широко представлены реляционные СУБД.

#### 3.2.2.1 Реляционная модель данных

Для того чтобы построить даталогическую модель БД по правилам реляционной модели данных необходимо знать компоненты этой модели.

1 Структура данных. В основе реляционной модели данных (РМД) лежит абстрактная теория данных, основанная на некоторых положениях математики – теории множеств и предикатной логики. Основа РМД – понятие теоретико—множественное отношения, которое представляет собой подмножество декартова произведения.

Рассмотрим некоторые определения.

Доменом называется множество допустимых значений одного или нескольких атрибутов. Пример домена: $D\{а,б,с\}$, где $D$ – название домена, $а,б,с$ – множество допустимых значений.

Декартовым произведением доменов $D_1, D_2, \ldots, D_к$, обозначаемом $D_1 \times D_2 \times \ldots D_к$, называется множество всех кортежей $(V_1, V_2, \ldots V_к)$ длины $к$, таких, что элемент кортежа $V_1$ принадлежит $D_1$, $V_2$ принадлежит $D_2$, $V_к$ принадлежит $D_к$. Кортеж – это множество атрибутов, каждый из которых определен на соответствующем домене.

Реляционным отношением называется некоторое подмножество декартового произведения одного или более доменов, имеющее смысл с точки зрения предметной области.

Удобно представлять отношение как таблицу, где каждая строка есть кортеж и каждый столбец (атрибут) представляет элементы домена. Если столбцам присваиваются имена, то их порядок становится несущественным.

Арность (степень) отношения – это число его атрибутов.

Мощность (кардинальное число) отношения – это число его кортежей.

Список имен атрибутов (столбцов) отношения называется схемой отношения: $R(A_1, A_2, \ldots, A_k)$, где $R$ – имя отношения, $A_1, A_2, \ldots, A_k$ – имена атрибутов.

Каждое реляционное отношение обладает хотя бы одним уникальным ключом. Уникальный ключ это атрибут или совокупность атрибутов отношения, значения которых не повторяются для разных кортежей. Каждый уникальный ключ так же называют потенциальным или возможным ключом.

Реляционное отношение используется для:

— представления класса объектов предметной области – посредством отображения в виде схемы отношения;

— представления связей между классами объектов – посредством потенциальных ключей и их копий. Таким образом, схемы отношений могут быть связаны!

Совокупность связанных между собой схем реляционных отношений, используемых для представления информации, называется схемой реляционной БД, а текущее значение элементов реляционных отношений – реляционной БД.

Схему реляционной базы данных можно представить в виде совокупности $m$ схем отношений:

$$\{R_1(A_{1,1}, A_{1,2}, \ldots, A_{1,k});$$
$$R_2(A_{2,1}, A_{2,2}, \ldots, A_{2,n});$$
$$\ldots$$
$$R_m(A_{m,1}, A_{m,2}, \ldots, A_{m,t})\},$$

где $m$ – количество схем отношений;

$k, n, t$ — арности отношений.

2 Операции манипулирования данными. Реляционной модели данных присущи четыре операции над данными: добавление строки (строк) в таблицу, обновление строки (строк) в таблице, удаление строки (строк) из таблицы, выборка или чтение данных из таблицы (таблиц). Наиболее важной операцией является операция выборки данных, эту операцию также называют

формулировкой запросов. Запросы в реляционной модели данных базируются на трех, равноценных между собой, теоретических языках: реляционной алгебре, реляционном исчислении с переменными — кортежами, реляционном исчислении с переменными — доменами. Запросы на основе реляционной алгебры выражены через операции объединения, разности, декартова произведения, пересечения, соединения и деления двух отношений, а также операций проекции и выборки, производимых над отдельным отношением.

3 Ограничения реляционной модели данных. Модели присущи два ограничения. Первое, основное ограничение, требует невозможности представления в реляционном отношении кортежей дубликатов и формулируется следующим образом: «каждое реляционное отношение имеет первичный ключ, значение первичного ключа должно быть определено». Первичный ключ определяется из совокупности уникальных ключей отношения. Это, как правило, атрибут числового типа и наименьшей длины.

Второе ограничение РМД называется правилом ссылочной целостности, или правилом поддержки целостности связей и формулируется: «каждому значению внешнего ключа отношения должно соответствовать значение, соответствующего первичного ключа (по связи), либо значение внешнего ключа может быть не определено». Основная цель ссылочной целостности заключается в недопустимости наличия "висячих" ссылок из дочерних отношений на родительское отношение. Значение внешнего ключа строки дочернего отношения должно быть равно значению первичного ключа некоторого кортежа родительского отношения, либо значение внешнего ключа может быть неопределенным. Отношение, которое содержит внешний ключ, называется ссылающимся отношением.

### 3.2.2.2 Виды документирования ДЛМ реляционной БД

Документировать процесс получения модели даталогической модели реляционной БД можно разными способами.

1 Формировать математическую запись получаемой схемы БД в виде совокупности описания схем реляционных отношений в виде $R_m(A_{m,1}, A_{m,2}, \ldots, A_{m,k})$. Такое представление удобно для дальнейшего анализа полученных схем отношений на соответствие заданной нормальной формы.

2 Делать формализованное описание каждого реляционного отношения в виде таблицы (таблица 12).

Таблица 12 — Формализованное описание реляционного отношения

| Название атрибута (поля) | Поле 1 | … | Поле к |
|---|---|---|---|
| Тип ключа (ПК— первичный ключ, УК – уникальный ключ, ВК – внешний ключ) | | … | |
| Обязательность (опциональность) значения (Д.б.—должно быть, М.б. – может быть) | | … | |

3 Представлять схему реляционной БД графически – рисунок 24.

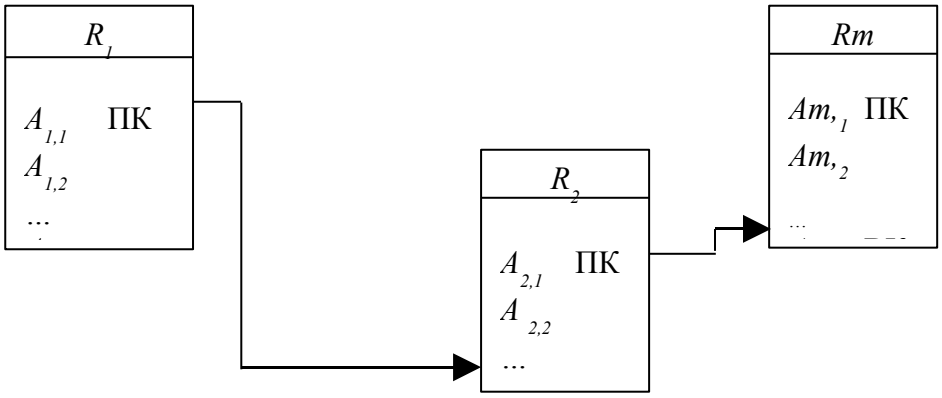



Рисунок 24 — Графическое представление схемы реляционной БД

На рисунке 24 название таблиц выделено. Помечены первичные (ПК) и внешние ключи (ВК). Если внешних ключей несколько, то они нумеруются – ВК1, ВК2. Связи прорисованы линями, идущими горизонтально и вертикально. Линии связи идут от первичного ключа к соответствующему внешнему ключу. Возле атрибута внешнего ключа (на стороне «много») линия связи заканчивается стрелкой. Графическое представление удобно для понимания логики структуры БД, отслеживания связей, моделирования *SQL*—запросов.

Наиболее полно представление о реляционном отношении дает его формализованное описание – появляется возможность подробно представит опциональность всех атрибутов. Это важно, поскольку преобразование в даталогической модели БД обязательной связи, присущей предметной области, не всегда делает опциональность внешнего ключа обязательной.

### 3.2.2.3 Формирование ДЛМ реляционной БД

Рассмотрим последовательно алгоритм формирования логической структуры реляционной базы данных на основе *ER*—диаграммы предметной области.

1 Преобразование простых классов объектов. Это классы объектов, информация о которых первой появляется в предметной области, они не имеют рекурсивных связей и не входят в супертипы, арки. Связи на стороне этих классов объектов имеют тип «один». Такие классы объектов называют родительскими или главными.

Алгоритм преобразования следующий: именем реляционного отношения становится имя класса объектов. Каждое свойство класса объектов становится атрибутом отношения, первичный ключ выделяется, уникальные (потенциальные) ключи помечаются. Все свойства, входящие в состав первичного ключа, должны быть обязательными. Большой составной первичный ключ может быть заменен суррогатным (техническим) первичным ключом. Атрибуты, входящие в состав уникальных ключей могут быть необязательными. Из состава уникальных ключей могут быть выбраны (помечены) ключи, альтернативные первичному ключу, для возможной

реализации связи полученного реляционного отношения с другими отношениями.

На рисунке 25 приведен фрагмент *ER*—диаграммы с простым классом объектов, в таблице 13 – формализованное описание реляционного отношения.

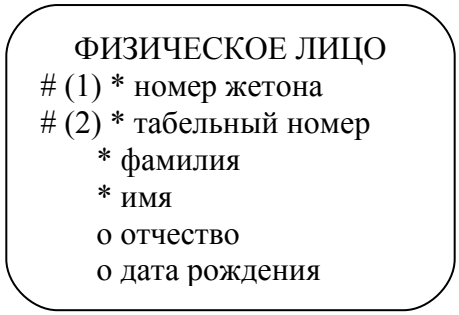



Рисунок 25 – Простой класс объектов

Таблица 13 – Реляционное отношение «Физическое лицо»

| Название поля | Номер жетона | Таб. номер | Фамилия | Имя | Отчество | Дата рождения |
|---|---|---|---|---|---|---|
| Ключ | ПК | УК | | | | |
| Опциональность | Д.б. | Д.б. | Д.б. | Д.б. | М.б. | М.б. |

В таблице использованы сокращения: ПК – первичный ключ, УК – уникальный ключ, Д.б. – обязательное значение поля, М.б. – необязательное значение поля.

2 Преобразование связи 1:М. Связь реализуется копированием первичного ключа из реляционного отношения на стороне «один» в реляционное отношение на стороне "много", из главного отношения в подчиненное. Новому появившемуся атрибуту присваивается уникальное в пределах отношения имя. В имени хорошо использовать имя таблицы, откуда осуществляется копия. Этот вновь появившийся атрибут помечается как внешний ключ. Если на *ER*—диаграмме опциональность связи со стороны «много» была обязательной, то опциональность внешнего ключа также обязательная. В противном случае опциональность внешнего ключа будет иметь значение "м.б.". Если уникальность класса объектов со стороны "много" определялась из связи, то внешний ключ должен входить в состав первичного ключа, эта ситуация соответственно помечается.

На рисунке 26 приведен фрагмент *ER*—диаграммы со связью 1:М. В таблицах 14 и 15 – реализация фрагмента в схеме реляционной БД.

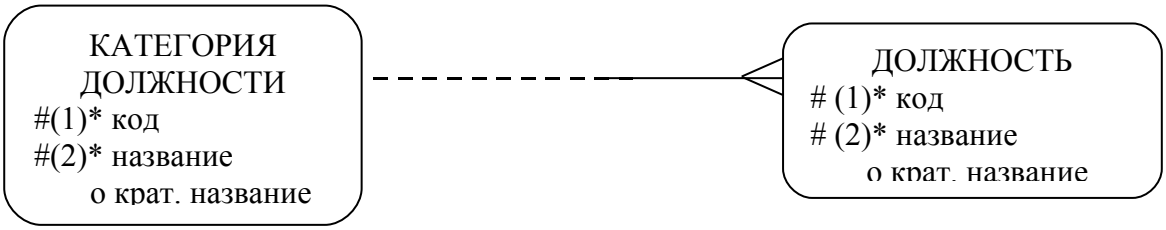



Рисунок 26 – Пример связи 1:М

Таблица 14 – Реляционное отношение «Категория должности»

| Название поля | Код | Название | Крат. название |
|---|---|---|---|
| Ключ | ПК | УК | |
| Опциональность | Д.б. | Д.б. | М.б. |

Таблица 15 – Реляционное отношение «Должность»

| Название поля | Код | Название | Крат. название | Код категории |
|---|---|---|---|---|
| Ключ | ПК | УК | | ВК |
| Опциональность | Д.б. | Д.б. | М.б. | Д.б. |

На рисунке 27 приведен пример фрагмента *ER*—диаграммы с отображением уникальности класса объектов из связи, в таблице 16 реализация части фрагмента в схеме реляционной БД. Каждый объект класса объектов «ЗАПИСЬ ТРУДОВОЙ КНИГИ» уникально определяется совокупностью двух свойств: свойства «дата начала» из класса объектов «ЗАПИСЬ ТРУДОВОЙ КНИГИ» и свойства «таб.номер» из класса объектов «ФИЗИЧЕСКОЕ ЛИЦО». Необходимо отметить, что свойство «дата» само по себе не может являться уникальным идентификатором.

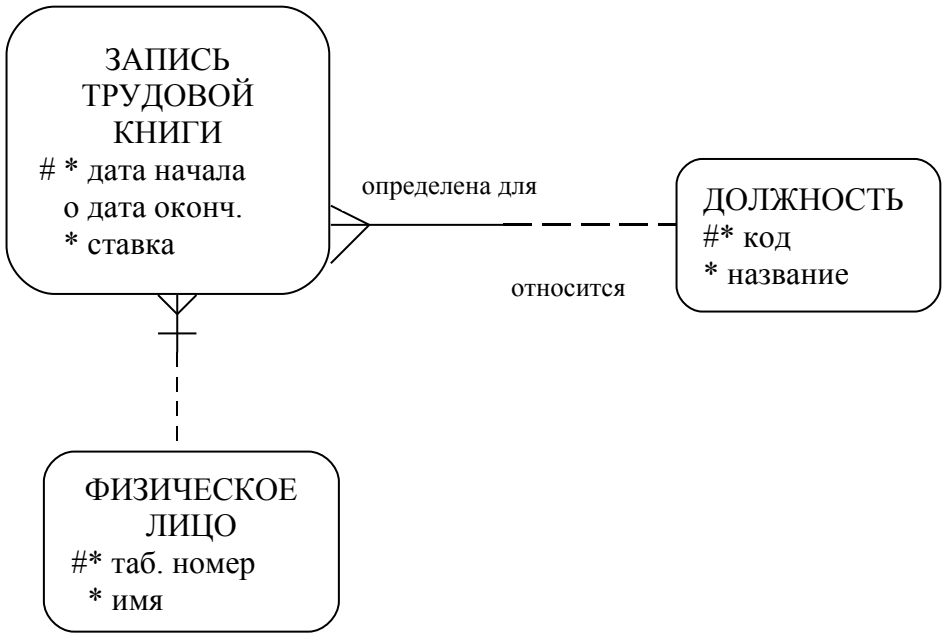



Рисунок 27 – Определение уникальности класса объектов из связи

Таблица 16 – Реляционное отношение «Запись трудовой книги»

| Название поля | Дата начала | Дата окончания | Ставка | Таб. номер ФЛ | Код должности |
|---|---|---|---|---|---|
| Ключ | ПК | | | ПК, ВК1 | ВК2 |
| Опциональность | Д.б. | М.б. | Д.б. | Д.б. | Д.б. |

3 Преобразование связи 1:1. В *ER*—диаграмме связь 1:1 может иметь разную опциональность. От этого зависит её отображение в схеме БД. Если связь "один к одному" обязательна с одной стороны, то поле с внешним ключом добавляется в отношение на обязательной стороне и это отношение становится подчиненным, опциональность внешнего ключа будет обязательной. Если связь 1:1 необязательная или обязательная (что очень редко) в обоих направлениях, необходимо выбрать в какую таблицу будет помещен внешний ключ. Решение принимается в зависимости от времени появления и объема данных:

— если строка в одном отношении создается обычно раньше, чем в другом (это определяется предметной областью), то это отношение назначается главным, а внешний ключ создается в подчиненном отношении;

— если в одном отношении будет меньше строк, чем в другом, то есть его размер будет изменяться менее динамично, тогда это отношение назначается главным, а внешний ключ создается в подчиненном. Внешний ключ создается копированием первичного ключа из главного отношения в подчиненное. Опциональность внешнего ключа определяется опциональностью связи. Замечание: внешний ключ, отображающий такую связь должен быть уникальным, это усиливает связь типа 1:1.

На рисунке 28 приведен пример фрагмента *ER*—диаграммы со связью 1:1, необязательной с одной стороны, в таблице 17 реализация части фрагмента в схеме реляционной БД. Главный класс объектов – ФИЗИЧЕСКОЕ ЛИЦО.

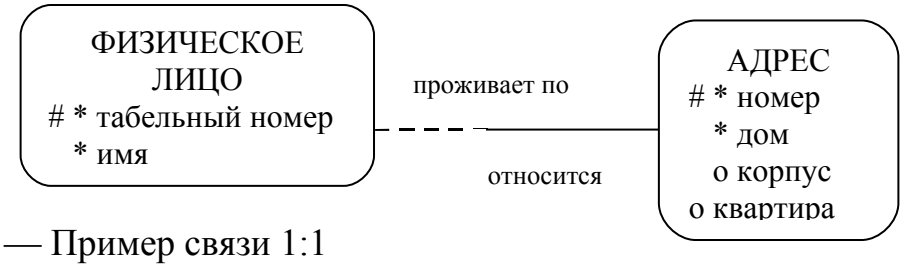



Рисунок 28 — Пример связи 1:1

Таблица 17 – Реляционное отношение «Адрес»

| Название поля | Номер | Дом | Корпус | Квартира | Таб. номер ФЛ |
|---|---|---|---|---|---|
| Ключ | ПК | | | | ВК, УК |
| Опциональность | Д.б. | Д.б. | М.б. | М.б. | Д.б. |

На рисунке 29 приведен пример фрагмента *ER*—диаграммы со связью 1:1, не обязательной с обеих сторон, в таблице 18 реализация части фрагмента в схеме реляционной БД. В качестве главного класса объектов выбран класс объектов ЧЛЕН КЛУБА. Это, наверное, более логично, хотя, если велосипедов гораздо меньше, чем членов клуба, то в виде главного может быть выбран класс объектов ВЕЛОСИПЕД.

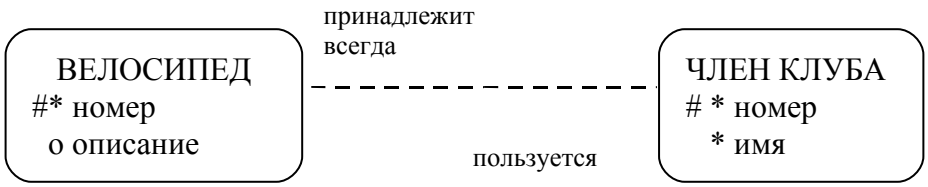



Рисунок 29 – Пример необязательной связи 1:1

Таблица 18 – Реляционное отношение «Велосипед»

| Название поля | Номер | Описание | Номер члена клуба |
|---|---|---|---|
| Ключ | ПК | | ВК, УК |
| Опциональность | Д.б. | М.б. | М.б. |

4 Преобразование рекурсивной связи. Поскольку рекурсивная связь – это связь между объектами одного класса объектов, то внешний ключ создается в том же отношении путем копирования первичного ключа в эту же схему. Дополнительные ограничения рекурсивной связи, такие, например как, объект не должен ссылаться сам на себя (нельзя быть женатым не себе самом, нельзя подчиняться себе самому) реализуются либо в логике приложений, либо в таких объектах БД, как хранимые процедуры, что более предпочтительно. Замечание: для рекурсивной связи 1:1 комбинация значений первичного и внешнего ключа не должна повторяться в кортежах отношения.

На рисунке 30 приведен пример фрагмента *ER*—диаграммы с рекурсивной связью, в таблице 19 реализация части фрагмента в схеме реляционной БД.

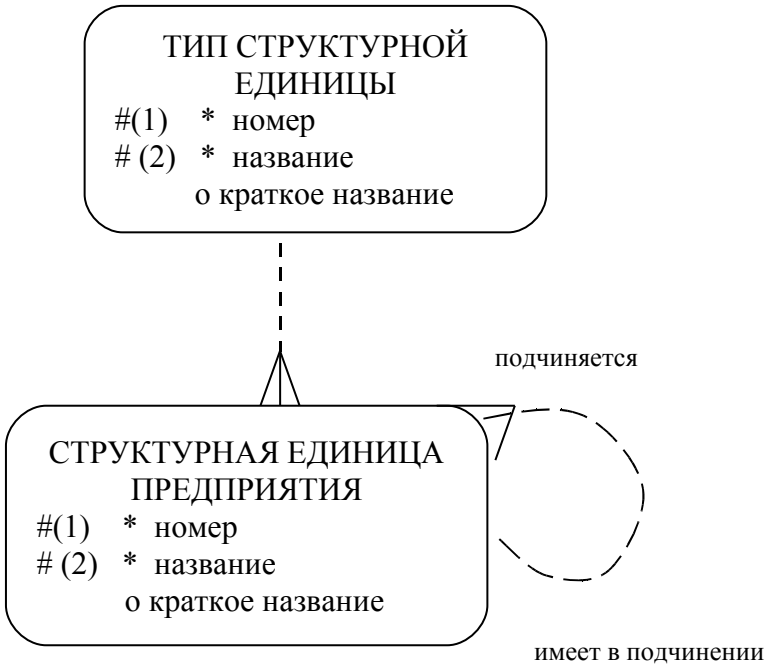



Рисунок 30 – Пример рекурсивной связи

Таблица 19 – Реляционное отношение «Структурная единица предприятия»

| Название поля | Номер | Название | Краткое название | Код родительской структурной единицы | Код типа |
|---|---|---|---|---|---|
| Ключ | ПК | | | ВК1 | ВК2 |
| Опциональность | Д.б. | Д.б. | М.б. | М.б. | Д.б. |

5 Реализация взаимоисключающих связей. Реализация арка в логической структуре может иметь несколько альтернатив. Имеется два способа преобразования арков во внешние ключи: явное проектирование арка и неявное проектирование арка.

При явном проектировании арка создается столбец внешних ключей для каждой связи, входящей в арк. Этот метод используется в том случае, если внешние ключи имеют разные форматы (тип, длину). При этом если связи на *ER*—диаграмме обозначены как обязательные, то столбцы внешних ключей не могут определяться как обязательные, потому что для каждой строки реляционной таблицы имеет значение только один из внешних ключей — должна быть реализована ситуация взаимоисключаемости связей — «или—или». Проверка исключительности каждого внешнего ключа реализуется с помощью хранимых процедур или триггеров.

При неявном проектировании арка создается один столбец внешних ключей и один дополнительный столбец, используемый как индикатор типа. Так как связи взаимоисключают друг друга, для каждой строки таблицы должно существовать только одно значение какого—либо внешнего ключа. Опциональность внешнего ключа определяются опциональностью связи – если она необязательная, то и внешний ключ будет иметь необязательное значение. При неявном проектировании арка все внешние ключи должны быть одного формата.

На рисунке 31 приведен пример фрагмента *ER*—диаграммы с арком, в таблице 20 реализация арка в схеме БД явно, в таблице 21 – реализация арка в схеме БД неявно.

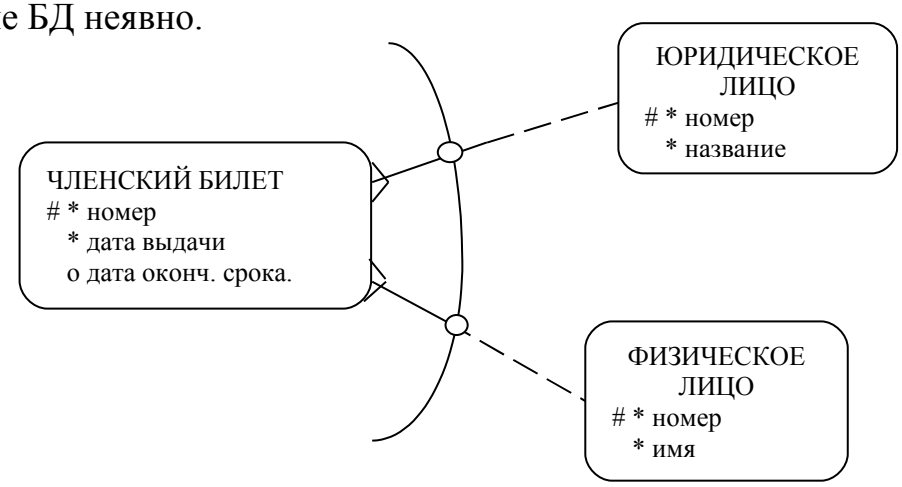



Рисунок 31. Фрагмент *ER*—диаграммы с арком

Таблица 20 – Явная реализация арка. Реляционное отношение «Членский билет»

| Название поля | Номер | Дата выдачи | Дата оконч. срока | Код ЮЛ | Номер ФЛ |
|---|---|---|---|---|---|
| Ключ | ПК | | | ВК1 | ВК2 |
| Опциональность | Д.б. | Д.б. | М.б. | М.б. | М.б. |
| Примеры данных | 268 | 12.05.2005 | 12.05.2006 | 1289 | |
| | 345 | 25.06.2006 | | | 516 |
| | 346 | 25.06.2006 | | 1327 | |

В рассмотренном примере при явной реализации арка опциональность внешних ключей «может быть», несмотря на то, что связи, входящие в арк на *ER*—диаграмме имеют обязательное значение.

Таблица 21 – Неявная реализация арка. Реляционное отношение «Членский билет»

| Название поля | Номер | Дата выдачи | Дата оконч. срока | Код члена клуба | Тип члена клуба |
|---|---|---|---|---|---|
| Ключ | ПК | | | ВК | |
| Опциональность | Д.б. | Д.б. | М.б. | Д.б. | Д.б. |
| Примеры данных | 268 | 12.05.2005 | 12.05.2006 | 1289 | 1 |
| | 345 | 25.06.2006 | | 616 | 2 |
| | 346 | 25.06.2006 | | 1327 | 1 |

Поскольку на *ER*—диаграмме связи входящие в арк имеют обязательное значение на стороне «много», то и внешний ключ при неявном проектировании арка тоже имеет опциональность «должен быть», в противном случае было бы наоборот.

6 Реализация взаимоисключающих классов объектов. Реляционные отношения, отображающие супертипы и подтипы могут быть смоделированы по—разному. Два наиболее используемых способа реализации: в виде одного отношения и в виде нескольких отношений (сколько подтипов – столько отношений).

При реализация подтипов в виде одного отношения все свойства каждого подтипа отображаются в едином отношении супертипа. При этом необходимо также добавить поле «тип» для обозначения того, к какому подтипу относится строка таблицы. Обязательность значений полей таблицы в этом случае должна поддерживаться дополнительно средствами СУБД (ограничениями *check* языка *SQL*, триггерами). Реализация в виде одного отношения рекомендуется в случае, если у подтипов небольшое количество собственных, присущих только им, свойств и связей.

На рисунке 32 приведен пример фрагмента *ER*—диаграммы с супертипами, в таблице 22 приведена реализация супертипа в схеме БД в виде одного отношения.

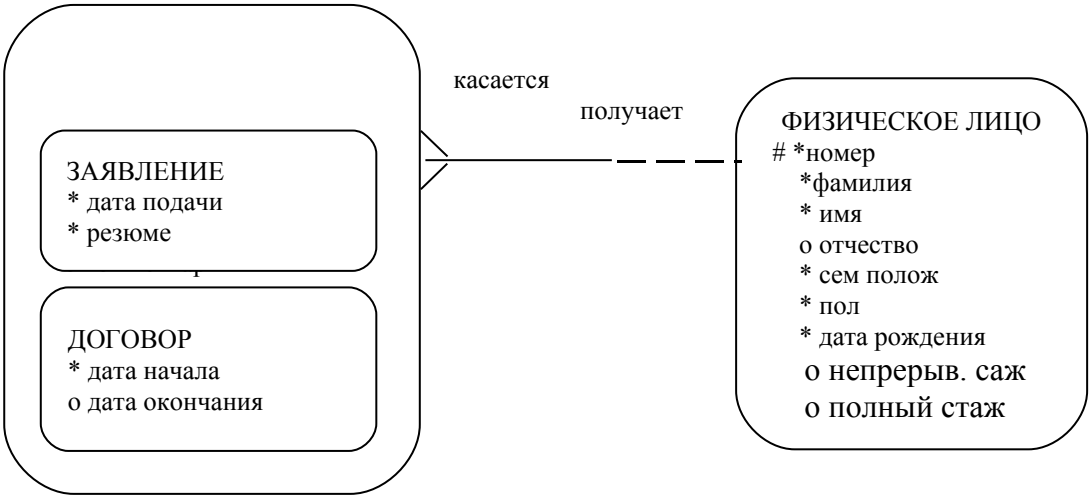



Рисунок 32 – Пример фрагмента *ER*—диаграммы с супертипом

Таблица 22 — Реализация супертипа в виде одного отношения. Реляционное отношение «Назначение»

| Название поля | Номер | Коммен —тарий | *Дата подачи* | *Резюме* | **Дата начала** | **Дата окон** | Номер ФЛ |
|---|---|---|---|---|---|---|---|
| Ключ | ПК | | | | | | ВК |
| Опциональность | Д.б. | М.б. | *М.б.* | *М.б.* | **М.б.** | **М.б.** | Д.б. |
| Примеры данных | 13 | К1 | *12.03.2006* | *Р1* | | | 13289 |
| | 14 | К3 | | | **12.03.2006** | | 13201 |
| | 15 | К3 | *12.03.2006* | *Р2* | | | 13290 |

В таблице данные одного подтипа выделены курсивом, другого – жирным шрифтом. Опциональность всех полей таблицы, принадлежащих подтипам – «может быть».

В случае реализации подтипов в виде нескольких отношений, каждый подтип отображается в отдельном отношении (сколько подтипов – столько отношений). Атрибуты супертипа повторяются в каждом отношении подтипа. Такая реализация рекомендуется в случае, если имеется большое количество свойств или связей, присущих только отдельным подтипам.

В таблицах 23 и 24– представлена реализация супертипа в схеме БД в виде двух отношений (два подтипа).

Таблица 23 – Реализация супертипа в виде нескольких отношений. Реляционное отношение «Заявление»

| Название поля | Номер | Коммен —тарий | Дата подачи | Резюме | Номер ФЛ |
|---|---|---|---|---|---|
| Ключ | ПК | | | | ВК |
| Опциональность | Д.б. | М.б. | Д.б. | Д.б. | Д.б. |
| Примеры данных | 13 | К1 | 12.03.2006 | Р1 | 13289 |
| | 15 | К3 | 12.03.2006 | Р2 | 13290 |

Таблица 24 – Реализация супертипа в виде нескольких отношений. Реляционное отношение «Договор»

| Название поля | Номер | Комментарий | Дата начала | Дата окон | Номер ФЛ |
|---|---|---|---|---|---|
| Ключ | ПК | | | | ВК |
| Опциональность | Д.б. | М.б. | М.б. | М.б. | Д.б. |
| Примеры данных | 14 | К3 | 12.03.2006 | | 13201 |

7 Реализация связей M:M. Наличие в *ER*—диаграмме связей M:M – это не дообследование предметной области и в этом случае модель предметной области не совсем адекватна.

В случае если связь M:M все—таки осталась, то от неё в схеме реляционной БД необходимо избавиться. Реляционные СУБД такую связь не поддерживает.

Для того чтобы разорвать связь M:M, в схеме реляционной БД создается искусственное отношение, в которое включаются копии первичных ключей из отношений на сторонах связи. Эти копии становятся соответствующими внешними ключами. Первичный ключ такого отношения состоит из обеих копий первичных ключей. В такую суррогатную таблицу по желанию проектировщика могут быть перенесены некоторые атрибуты из отношений на сторонах связи M:M. В любом случае будет получена модель, отражающая предметную область неадекватно.

### 3.2.2.4 Анализ схемы реляционной БД на соответствие заданной нормальной форме

Полученную на основе *ER*—диаграммы схему реляционной БД необходимо проверить на соответствие заданной нормальной форме и в случае необходимости осуществить нормализацию отдельных схем отношений. Нормализация схем отношений необходима для устранения избыточности данных в реляционных отношениях, ведущих к аномалиям при добавлении, обновлении и удалении данных в БД. Избыточность данных также может привести к неправильным результатам при обработке данных.

В теории баз данных определено 6 нормальных форм (НФ). Вначале были предложены 1НФ, 2НФ, 3НФ, затем сформулировано более строгое определение 3НФ, которое получило название НФБК (нормальная форма Бойса—Кодда). Все эти формы предназначены для устранения нежелательных функциональных зависимостей между атрибутами отношения. Далее следуют 4НФ и 5НФ, устраняющие многозначные зависимости и зависимости соединения. Нормальные формы обладают свойством вложенности – если схема отношения соответствует какой—то более высокой НФ, то она соответствует и всем более низким НФ. Для нормальной работы базы данных достаточно исследовать структуру БД на соответствие 3НФ, при использовании «восходящего» метода проектирования – до НФБК.

Как правило, при полном обследовании предметной области и правильном выявлении классов объектов и связей между ними получаемая на

основе *ER*—диаграммы логическая структура реляционной БД нормализована и соответствует 3НФ, в ней отсутствуют составные первичные ключи и транзитивные зависимости между не ключевыми атрибутами и первичным ключом.

Нормализация схем отношений сложный процесс, требует дополнительного анализа предметной области, выявления функциональных и транзитивных зависимостей между не ключевыми атрибутами и первичным ключом каждой схемы отношения.

Функциональной зависимостью атрибута *B* (набора атрибутов) отношения *R(A,B,C)* от атрибута (набора атрибутов) *A* отношения *R*, обозначаемой *A —> B*, называется такая связь между атрибутами отношения, когда в предметной области в каждый момент времени каждому значению атрибута (набору атрибутов) *B* соответствует только одно определенное значение атрибута (набора атрибутов) *A*. Однако, для заданного значения атрибута B может существовать несколько различных значений атрибута *A*. Таким образом, если из семантики предметной области нам известно значение атрибута *A*, то в предметной области однозначно можно определить значение атрибута *B*.

Различают частичную и полную функциональные зависимости. Частичная функциональная зависимость – это зависимость не ключевого атрибута от части составного первичного ключа. Полная функциональная зависимость – это зависимость не ключевого атрибута от всего составного первичного ключа. Если в схеме отношения первичный ключ не составной, то можно сказать, что в этой схеме отношения отсутствуют частичные функциональные зависимости.

Атрибут *C* отношения *R* транзитивно зависит от атрибута *A* отношения *R*, если для атрибутов *A*, *B*, *C* отношения *R(A,B,C)* выполняется условие существования следующих функциональных зависимостей: *A —> B*; *B —> C*, при условии, что атрибут *A* функционально не зависит ни от атрибута B, ни от атрибута *C*.

Транзитивная зависимость появляется в случае того, что ранее, при обследовании предметной области было выявлено свойство класса объектов, принимающее несколько значений для одного и того же значения первичного ключа. В примере, приведенном на рисунке 20, если оставить класс объектов «КЛИЕНТ» без изменения, то при дальнейшем отображении его в виде реляционной таблицы, в таблице будет наблюдаться транзитивная зависимость: *Номер—>Дата контакта*; *Имя—>Дата контакта*, следовательно, присутствует транзитивная зависимость *Номер—>Дата контакта*. Для исключения транзитивной зависимости необходимо будет создать новое отношение с атрибутами (Номер клиента, Дата контакта), где «Номер клиента» будет одновременно и первичным и внешним ключом, ссылающимся на свою копию в отношении «Клиент». Полученная таким образом схема реляционной БД не совсем адекватно будет отражать предметную область. Более логично выделить сразу класс объектов «КОНТАКТ» и найти ему адекватный уникальный идентификатор.

Для того чтобы нормализовать схемы реляционных отношений необходимо знать определения нормальных форм.

1 Первая нормальная форма. Схема отношения находится в 1НФ тогда и только тогда, если все атрибуты схемы имеют атомарное значение и в схеме отношений отсутствуют повторяющиеся группы. Повторяющаяся группа – один или более элементов данных, которые имеют более одного значения для одного значения части первичного ключа. Выявление повторяющихся групп осуществляется, если первичный ключ составной. Повторяющимися являются поля, содержащие одинаковые по смыслу значения.

Необходимо отметить, что при правильном анализе предметной области определения составного ключа, как главного ключа класса объектов можно избежать. Почти всегда можно идентифицировать объект в классе объектов с помощью не составного уникального идентификатора, кандидата в первичный ключ. Это какой—то шифр, артикул, нормативный код или номер, номер записи какого—то документа, имеющего в свою очередь свой адекватный номер. Всякий учитываемый контакт в любой предметной области каким—либо образом документируется и нумеруется, это определено правилами делопроизводства.

2 Вторая нормальная форма. Схема отношения находится во 2НФ, если она находится в 1НФ и все не ключевые атрибуты функционально полно зависят от составного первичного ключа. При этом если схема отношения находится в 1НФ и первичный ключ не составной, то схема отношения, по определениям первичного ключа и функциональных зависимостей, находится во 2НФ. Если первичный ключ в схеме отношения составной, тогда необходимо выявлять частичные функциональные зависимости и избавляться от них.

3 Третья нормальная форма. Схема отношения находится в 3НФ, если она находится во 2НФ, и отсутствуют транзитивные зависимости между не ключевыми атрибутами и первичным ключом.

После последовательного выявления частичных функциональных и транзитивных зависимостей, они выносятся в отдельные схемы отношения путем декомпозиции исходных схем отношений. Таким образом, нормализация схемы реляционной БД ведет к увеличению количества схем отношений, при этом полученная результирующая схема БД, практически всегда, неадекватно отображает предметную область. Адекватность может наблюдаться для небольших баз данных.

4 Нормальная форма Бойса—Кодда (НФБК). Ситуация, когда отношение находится в 3НФ и его надо проверить на соответствие НФБК, возникает при условии, что отношение имеет два (или более) уникальных (потенциальных) ключа, которые при этом являются составными и имеют общий атрибут.

Эта нормальная форма вводит дополнительное ограничение по сравнению с 3НФ и определяет, что отношение находится в НФБК, если оно находится в 3НФ и каждый детерминант отношения является потенциальным ключом отношения. Детерминант – это атрибут (совокупность атрибутов),

находящийся в левой части функциональной зависимости. В одном отношении может быть несколько детерминантов.

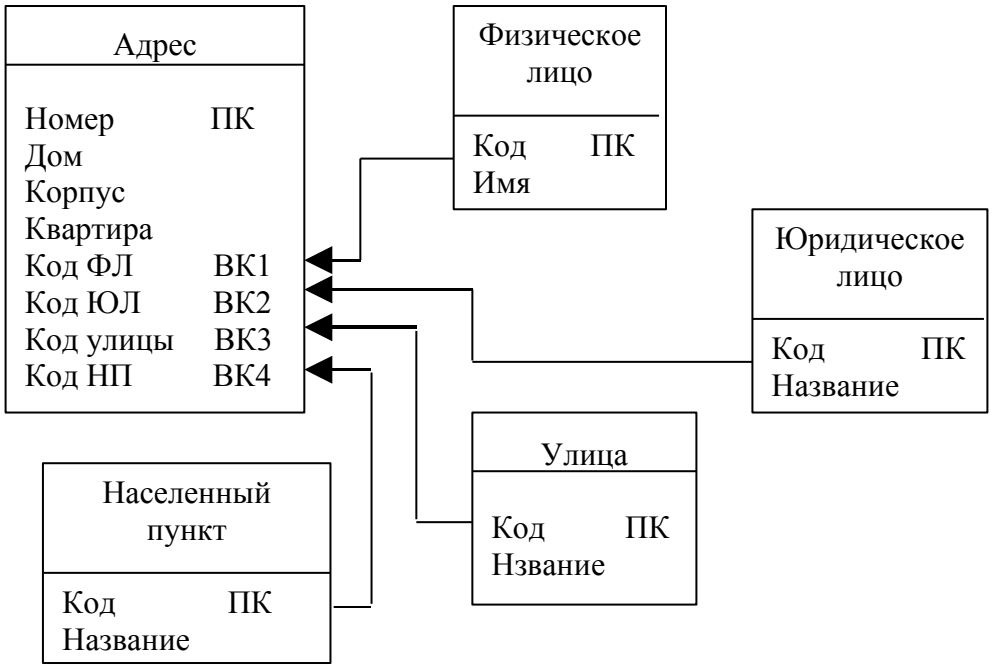
Приведение схем отношений к более высоким нормальным формам обычно требуется для схем отношений, полученных при проектировании БД «восходящим» методом.

Только понимая семантику предметной области можно правильно осуществить процесс нормализации.

### 3.2.2.5 Пример графического представления схемы реляционной БД

На рисунке 33 представлена схема реляционной базы данных, полученная на основе *ER*—диаграммы фрагмента предметной области, представленного на рисунке 16.

Для отображения в логической структуре БД взаимоисключаемости связей использовано явное проектирование арка.




ПК – первичный ключ, ВК – внешний ключ, ФЛ – физическое лицо, ЮЛ – юридическое лицо, НП – населенный пункт.

Рисунок 33 – Пример графического представления ДЛМ реляционной БД

Приведенная схема соответствует, как минимум, 3НФ, поскольку во всех отношениях атрибуты атомарные, первичные ключи не составные и отсутствуют транзитивные зависимости между не ключевыми атрибутами и первичными ключами.

### 3.3 Проектирование внутреннего уровня БД

Проектирование внутреннего уровня БД заключается в преобразовании логической модели данных в такую форму, которая позволит реализовать данный проект в среде выбранной СУБД. Исходными данными для проведения этого этапа является полученная на ранних этапах проектирования логическая структура БД (ДЛМ БД). В результате должна быть получена физическая модель БД, представленная на языке определения данных выбранной СУБД. Чаще всего это *SQL* команды, описывающие объекты БД (*SQL* скрипты).

Разработчик физической модели БД (эту функцию выполняет администратор БД) должен хорошо знать функциональные возможности выбранной СУБД, понимать все её достоинства и недостатки. На данном этапе проектирования БД должны быть определены и реализованы способы размещения данных в среде хранения и способы доступа к этим данным на физическом уровне. Хотя для организации физической модели могут быть использованы и файловые системы, в большинстве случаев для реализации баз данных используется физическая организация файлов, которыми непосредственно управляет СУБД. Как правило, СУБД для платформы *IBM PC* не предоставляют проектировщику никаких возможностей выбора или изменения способа организации файлов БД. Настройка среды хранения и настройка оптимальной работы СУБД, а, следовательно, и БД, является сложной задачей и требует определенных профессиональных знаний и навыков. Мы рассмотрим часть видов работ на этапе физического проектирования БД: выбор СУБД и проектирование ряда объектов БД.

#### 3.3.1 Выбор реляционной СУБД

СУБД значительно различаются по своим характеристикам и функциям. Первые программные продукты такого рода были разработаны для больших ЭВМ в конце 1960—х годов и выполняли достаточно простые операции. С тех пор СУБД постоянно совершенствовались, а функции их расширялись. Усовершенствования касались не только обработки данных, СУБД также снабжались функциями, упрощающими создание приложений БД.

При появлении персональных компьютеров начался бурный рост популярности настольных реляционных СУБД (*dBase, FoxPro, Paradox*), а затем их сетевых многопользовательских версий, позволяющих обрабатывать данные, находящиеся в общедоступном месте в сети. В многопользовательских настольных СУБД появились механизмы блокировок частей данных, что позволило реализовать обращение к одним и тем же данным нескольких пользователей одновременно. Недостатки таких СУБД (низкая производительность и защита данных, ограниченное число пользователей) отсутствуют в СУБД, появившихся на следующих этапах развития данного программного продукта. Это, так называемые, серверные СУБД.

Серверные СУБД предназначены для архитектуры вычислительной среды «клиент—сервер», основанной на централизации хранения и обработки

данных на одном выделенном более мощном компьютере, называемом сервером базы данных. Сервер БД отвечает за работу с файлами БД, поддержку ссылочной целостности, резервное копирование и восстановление БД, параллельную работу с данными большого числа пользователей, обеспечение авторизованного доступа к данным, ведение данных и их обработку. Основным преимуществом серверных СУБД является обеспечение высокой безопасности и надежности данных.

В настоящее на рынке программных продуктов насчитывается несколько сотен реляционных СУБД.

Основным принципом при выборе СУБД является определение такого программного продукта, который в наибольшей степени соответствует требованиям, определенным характером решаемой задачи автоматизации.

Решить эту задачу непросто:

— во—первых, требования к СУБД, по мере освоения пользователями её функциональных особенностей, меняются – требуются новые возможности. Так, например, сначала была реализована локальная версия БД, затем потребовалась сетевая. Либо потребовалось увеличение скорости обработки данных, наиболее полная поддержка ограничения целостности данных, определяемых предметной областью и т.п.;

— во—вторых, СУБД имеют большое число параметров, их трудно сравнивать;

— в—третьих, зачастую используют либо не лицензионные, либо свободно распространяемые программные продукты и поэтому разработчики не имеют доступа к качественной технической документации.

Каждый программный продукт, и СУБД в частности, может сопровождать информация следующего вида:

— сведения разработчиков и рекламная информация продавцов;

— информация, предоставляемая на различных форумах конечными пользователями, разработчиками программного обеспечения для БД, администраторами, имеющих опыт работы с БД;

— информация профессиональных аналитиков и экспертов.

Выбор СУБД удобно проводить в 3 этапа.

1 Этап качественного оценивания предлагаемых показателей пригодности программного продукта, уменьшающий, таким образом, область выбора.

2 Этап оценки технических характеристик выбранных систем более детально.

3 Этап оценки производительности оставшихся продуктов по различным критериям для принятия окончательного решения.

К показателям пригодности относятся следующие характеристики СУБД:

— полнота функциональности СУБД (развитый интерфейс пользователя, позволяющий осуществлять различные операции с БД: создание БД; создание и модификация объектов БД; ввод, обновление информации; реализация запросов и вывод результатов на печать);

— возможность реализации сервера БД;

— наличие встроенных средств для разработки программ для работы с БД (клиентских приложений);

— поддержка объектов БД, позволяющих хранить логику предметной области и логику обработки данных;

— требования к квалификации пользователей;

— удобство и простота использования;

— сложность процедуры установки;

— сложность интерфейса пользователя;

— простота выполнения обычных операций с БД;

— наличие подсказок помощи;

— модель представления данных;

— возможность обеспечения целостности данных на уровне СУБД;

— автоматическая генерация кода пользовательского приложения, разрабатываемого в среде СУБД визуально;

— наличие средств автоматизации проектирования БД;

— качество средств защиты и контроля данных (поддержка внутренних ограничений модели данных и ограничений предметной области; средства для реализации функций резервного копирования и восстановления БД);

— поддержка сетевых протоколов, обеспечивающих работу СУБД в различных сетях;

— наличие средств разграничения прав пользователя;

— поддержка стандартных интерфейсов: *ODBC, SQL* и др.;

— фирма – разработчик;

— наличие документации или книг на рынке;

— высокая уверенность в появлении новой версии;

— стоимость.

К техническим характеристикам СУБД относятся:

— операционная система, под управлением которой работает СУБД;

— потребность в оперативной памяти

— ограничение на максимальный объем БД;

— перечень объектов БД, поддерживаемых СУБД;

— ограничение на количество одновременных подключений пользователей;

— ограничения на операции над данными;

— максимальный размеры поля таблицы, числа полей в таблице, размер строки, количества строк в таблице, число таблиц, которые можно обрабатывать одновременно;

— средства поддержки ограничений целостности БД;

— возможности средств формулировки и выполнения запросов (навигационная обработка, работа с языками запросов *QBE, SQL*);

— возможность сохранения запросов в БД;

— поддержка многопользовательской работы с БД (виды блокировок, средства обработки транзакций);

— наличие различных генераторов (интерфейса, отчетов, выполняемых модулей);

— наличие средств импорта и экспорт данных, в частности в web— формат.

Задача определения производительности СУБД является достаточно сложной. Решить её могут только высокопрофессиональные специалисты, владеющие методиками измерения производительности программных продуктов. Одна из методик – это использование тестов *TPC*, разработанных *Transaction processing Performance Council* — Советом по производительности систем обработки транзакций; тесты, не относящиеся к семейству *TC*P — *Wisconsin Benchmark и AS3AP*.

Производительность СУБД оценивается следующими характеристиками:

— временем выполнения запросов;

— скоростью поиска информации в неиндексированных полях;

— временем выполнения операций импортирования базы данных из других форматов;

— скоростью создания индексов и выполнения таких массовых операций, как обновление, вставка, удаление данных;

— максимальным числом параллельных обращений к данным в многопользовательском режиме;

— временем генерации отчета.

На производительность СУБД также оказывает большое влияние правильное проектирование объектов базы данных.

Сведения о некоторых наиболее популярных СУБД приведены в таблице 25.

Таблица 25 – Современные СУБД

| Продукт | Фирма производитель | URL |
|---|---|---|
| Настольные, для создания локальных БД | | |
| *Visual dBase* | *dBase, Inc.* | *http://www.dbase2000.com* |
| *Paradox* | *Corel* | *http://www.corel.com* |
| *Microsoft Access* | *Microsoft* | *http://www.microsoft.com* |
| *Microsoft FoxPro* | | |
| *Microsoft Visual FoxPro* | | |
| Серверные СУБД | | |
| *Oracle* | *Oracle Corp* | *http://www.oracle.com* |
| *Microsoft SQL Server* | *Microsoft* | *http://www.microsoft.com* |
| *Informix* | *Informix* | *http://www.informix.com* |
| *Sybase* | *Sybase* | *http://www.sybase.com* |
| *IBM DB2* | *IBM* | *http://www—4.ibm.com* |

### 3.3.2 Объекты БД

Рассмотрим некоторые объекты БД, поддерживаемые современными системами управления базами данных (таблица 26).

Таблица 26 – Объекты БД

| Объект | Описание | Комментарий |
|---|---|---|
| *SCHEMA* Схема | Именованный набор (множество) объектов БД, управляемых одним пользователем. | БД может включать набор разных схем. |
| *DOMAIN* Домен | Объект, который может использоваться как альтернатива типу данных для столбца (ов) таблицы, таблиц. Домен определяет тип данных и может также задавать некоторые другие ограничения атрибута, значения по умолчанию. | Создатели реляционной базы данных настоятельно рекомендовали использовать домены. |
| *TAB*LE Таблица | Фундаментальная информационная структура реляционных БД. | Поддерживается всеми реляционными СУБД |
| INDEX Индекс | Хранит последовательность упорядочивания данных в столбце (столбцах) таблицы по возрастанию или убыванию. | Создаются только те индексы, которые реально необходимы |
| *SEQUENCES* Последователь—ность | Объект схемы данных, генератор порядковых номеров, используемых для автоматической генерации значений уникальных и первичных ключей. | Использование последовательности увеличивает конкурентоспособность приложений БД |
| *PROCEDURE* Процедура | Хранимый в БД программный модуль, написанный на процедурном языке СУБД и используемым разными пользователями БД | Используются для реализации бизнес—правил предметной области |
| *TRIGGER* Триггер | Хранимая процедура, которая автоматически выполняется СУБД при внесении изменений в указанной таблице (вставка, обновление, удаление данных). | Всегда связан с конкретной таблицей |
| *VIEW* Представление, просмотр | Виртуальная таблица, которая создается только по вызову и обрабатывается как таблица, затем удаляется. | Хранятся в виде SQL—запросов, используются для удобного представления данных или для реализации разграничения прав пользователей. |
| *ROLE* Роль | Именованный набор прав на множество объектов базы данных. | Применяются для эффективного управления привилегиями пользователей БД. |
| *CURSOR* Курсор | Указатель, используемый для перемещения по наборам записей при их обработке. | Поддерживает текущую позицию данных в БД |
| *USERS* Пользователи | Пользователь БД | Применяется для разграничения прав доступа к объектам БД |

Различные СУБД поддерживают различный набор объектов, хранимых в БД.

Современные серверные СУБД поддерживают все перечисленные в таблице 26 объекты базы данных. Наиболее своеобразной, в плане поддерживаемых объектов БД, является СУБД *Access*. Несмотря на это, средствами СУБД *Access* можно реализовать все предъявляемые к реляционной БД требования.

### 3.3.3 Физическая модель БД

Описание физической модели базы данных включает в себя описание объектов БД. Есть несколько способов, с помощью которых структура объектов БД описывается для СУБД. Чаще всего для этого создается текстовый файл, который описывает эти структуры. Язык, используемый для определения структур объектов БД, называется языком определения данных (ЯОД). Альтернативой текстовому описанию является визуальный (графический) способ задания структур объектов БД, используемый, например, в СУБД *Access*.

Схема БД состоит из набора определений, выраженных на ЯОД. В результате компиляции команд, созданных на ЯОД, создается системный каталог (словарь данных), в котором хранятся метаданные, т.е. данные, описывающие объекты БД. Метаданные упрощают способы доступа и управления объектами БД.

#### 3.3.3.1 Проектирование реляционных таблиц

Каждое отношение схемы реляционной базы данных, полученное на этапе даталогического проектирования, должно быть описано на языке ЯОД СУБД и содержать следующие конструкции:

— имя отношения (таблицы);
— имена атрибутов (полей);
— определение первичных ключей;
— определение уникальных (потенциальных) ключей;
— определение физических характеристик атрибута (тип и длину);
— определение обязательности значения атрибута;
— определение логических ограничений на значение атрибута.

В начале физического проектирования реляционных таблиц удобно создать техническое описание этих таблиц, что затем позволит более эффективно создавать текстовое описание их структур на ЯОД.

Техническое описание можно представить в виде таблицы. Рассмотрим на примере. Допустим, есть следующая схема реляционной БД, содержащая следующие отношения: Категория должности (Код, Название, Краткое название); Должность (Код, Название, Краткое название, Код категории); Подразделение (Номер, Название, Аббревиатура, Код подразделения); Запись о работе сотрудника (Номер, Дата начала, Дата окончания, Ставка, Код

должности, Код подразделения, Код сотрудника); Сотрудник (Табельный номер, Имя, Дата рождения, Пол).

Техническое описание таблицы «Должность» на ЯОД СУБД *Access* приведено в таблице 27, на ЯОД СУБД *InterBase* в таблице 28.

Таблица 27 – Реляционная таблица «Должность»

| Имя поля | Код_Долж | Назв_Долж | Кр_Назв_Д | Код_Катег. |
|---|---|---|---|---|
| Ключ | Ключевое поле | | | |
| Тип, длина | Счетчик | Текстовый, 50 | Текстовый, 16 | Числовой, длинное целое |
| Обязательность значения | Да | Да | Нет | Да |
| Логическое ограничение на поле | | Маска ввода: L<????????? ????? | Маска ввода: а?????????? ???? | |
| Примеры данных | 17 | техник | тех | 3 |
| | 29 | хормейстер | хорм | 3 |
| | 348 | заведую— щий складом | зав.скл. | 12 |

Таблица 28— Реляционная таблица «*Dolgn*»

| Имя поля | *Kod_D* | *N_D* | *Sh_K_D* | *K—Kateg* |
|---|---|---|---|---|
| Ключ | *Primary Key* | | | *Foreign Key* |
| Тип, длина | *Integer* | *VarChar (50)* | *VarChar(16)* | *Integer* |
| Обязательность значения | *Not Null* | *Not Null* | *Null* | *Not Null* |
| Логическое ограничение на поле | *Check (value>0)* | | | *Check (value>0)* |
| Примеры данных | 17 | техник | Тех | 3 |
| | 29 | хормейстер | Хорм | 3 |
| | 348 | заведую— щий складом | зав.скл. | 12 |

Из таблиц 27 и 28 видно, как отличаются языки определения данных в СУБД *Access* и *InterBase*. СУБД *InterBase* использует ЯОД, являющимся диалектом стандарта языка *SQL*.

### 3.3.3.2 Реализация ограничений целостности реляционной базы данных

Создаваемая и эксплуатируемая реляционная база данных должна быть целостной и надежной. Правила поддержки ограничений целостности и надежности определяются используемой моделью данных и предметной областью. Правила должны быть описаны в физическом проекте БД и

реализованы либо средствами СУБД, либо приложения. Целостность данных – это поддержка точности и корректности данных, хранящихся в БД.

Поддержка целостности реляционной БД рассматривается в 3—х аспектах.

1 Целостность таблицы. Обязательно должны поддерживаться:

— уникальность строк таблицы. Должен быть определен первичный ключ таблицы, и значение его должно быть определено;

— все уникальные (потенциальные) ключи, выявленные в ходе анализа предметной области.

Эти ограничения реализуются в командах создания и модификации таблиц. Например, в языке *SQL* это команды *Create Table*, *Alter Table*. В этих командах для описания полей — первичных ключей используется конструкция *Primary Key*, для описания полей – уникальных ключей конструкция *Unique*, обязательность значений полей задается конструкцией *Not Null*.

2 Ссылочная целостность. Каждая таблица проектируемой БД должна быть связана с другими посредством соответствующих первичных и внешних ключей, т.е. быть либо родительской (главной) по отношению к другим таблицам, либо дочерней (подчиненной), либо той и другой для разного уровня связей. Назначение внешнего ключа — связывать каждую строку дочерней таблицы с соответствующей строкой родительской таблицы. Значение внешнего ключа может иметь и пустое значение (*Null*), если он реализует необязательную связь, выявленную в предметной области. В качестве значения внешнего ключа может выступать значение и любого уникального (потенциального) ключа. Чтобы в физическом проекте реализовать поддержку ссылочной целостности, необходимо знать ситуации, когда она может быть нарушена:

1) вставка новой строки в дочернюю таблицу. В этом случае значение атрибута внешнего ключа новой строки должно соответствовать конкретному значению, присутствующему в одной из строк родительской таблицы, либо должно быть равно пустому значению (*Null*). В противном случае целостность будет нарушена;

2) обновление внешнего ключа в строке дочерней таблицы. Ситуация достаточно редкая, должны поддерживаться требования предыдущего пункта;

3) удаление строки из родительской таблицы. Ссылочная целостность будет нарушена, если в дочернем отношении существуют строки, ссылающиеся на удаляемую в родительской таблице строку. В этом случае может быть использована одна из следующих стратегий:

а) *No Action* – удаление строки из родительской таблицы запрещено, если в дочерней таблице есть хотя бы одна ссылающаяся на неё строка;

б) *Cascade* (каскадное взаимодействие) – при удалении строки из родительской таблицы автоматически удаляются все ссылающиеся на нее строки дочерней таблицы. Если при этом любая из удаляемых строк дочерней таблицы выступает в качестве родительской для дочерних таблиц следующего уровня, то операция удаления применяется ко всем строкам дочерней таблицы

этой связи и т.д. – удаление распространяется каскадно на все дочерние таблицы;

в) *Set Null* – при удалении строки из родительской таблицы во всех ссылающихся на неё строках дочерней таблицы в атрибутах внешнего ключа записывается пустое значение (Null);

г) *Set Default* – при удалении строки родительской таблицы значение атрибутов внешнего ключа ссылающейся на неё строки дочерней таблицы автоматически замещаются значениями по умолчанию, определенными при создании дочерней таблицы;

д) *No Check* – при удалении строки из родительской таблицы никаких действий по сохранению ссылочной целостности не предпринимается;

4) обновление первичного ключа в строке родительской таблицы. Редкая ситуация, рассматриваются все возможные стратегии, как и в случае 3).

Связи между таблицами (ссылочная целостность) могут быть заданы либо путем явного описания внешних ключей в структурах таблиц (что является более предпочтительным, как и любое другое явное описание), либо ссылочная целостность может поддерживаться с помощью триггеров. Например, в среде СУБД *InterBase* связь между двумя таблицами можно определить в команде *Create Table* при помощи конструкции *Foreign Key*, задающей явно поле – внешний ключ, ссылающийся на соответствующее поле — первичный ключ (конструкция *References*). В этом случае СУБД *InterBase* запрещает изменять значение первичного ключа, если на нее ссылаются какая —либо строка из дочерней таблицы и удалять запись в родительской таблице, если на неё есть ссылающаяся запись из дочерней таблицы. Таким образом связь, описанная в команде *Create Table*, блокирует каскадные изменения и удаления в родительской и дочерней таблицах. По умолчанию СУБД *InterBase* использует стратегию *No Action*.

С помощью триггеров можно реализовать любую стратегию. Например, триггер, спроектированный для реализации каскадного обновления значений внешних ключей в дочерней таблице, будет автоматически срабатывать на любое обновление в родительской таблице по следующему алгоритму: если старое значение поля, являющегося первичным ключом родительской таблицы не остается равным новому, тогда изменить значение соответствующего поля внешнего ключа дочерней таблицы на новое значение поля первичного ключа.

Еще один пример. Триггер, спроектированный для срабатывания на удаление строки в родительской таблице, будет работать по алгоритму: удалить в дочерней таблице те строки, у которых значения поля внешнего ключа по данной связи равно значению поля первичного ключа удаляемой строки родительской таблицы (каскадное удаление).

3 Декларативные ограничения данных. Так называют ограничения реляционной базы данных, объявленные предметной областью и выявленные в ходе её анализа. Задача проектировщика БД — адекватно отобразить их в БД.

Самые распространенные ограничения предметной области – это ограничения на свойства объекта предметной области, далее атрибута отношения или поля таблицы:

— обязательность значения поля;

— тип, длина, диапазон значения поля (например, значение должно быть целым и положительным), вхождение значения в заданный список и т.п.

Такие ограничения рекомендуется задавать на уровне домена в командах *Create Domain*, *Alter Domain*. Также они могут быть заданы в командах создания и модификации таблиц — *Create Table*, *Alter Table* при описании поля таблицы.

Кроме ограничений предметной области, которые могут быть явно отображены на всех этапах проектирования БД на уровне моделей данных (например, использование семантических возможностей *ER*—диаграммы), существует ряд ограничений, которые выявлены, но не отражены в моделях данных, а описаны на естественном языке проектировщика. Например, это такие ограничения:

— если статус человека «не состоит в браке», то его следующим статусом может быть «в браке», но не в коем случае «разведен (а)»;

— если статус студента был «в академическом отпуске», то следующий статус может быть только «вышел из академического отпуска», но не в коем случае «отчислен»;

— общее количество действующих записей (текущая дата меньше даты окончания работы меньше) о работе человека не должно превышать больше 2 —х (запрет двух совмещений);

— общее количество ставок, занимаемых работающим человеком, не может превышать 1,5 единиц;

— общее количество заключенных договоров на аренду не может уменьшаться со временем.

Такие ограничения называют ограничениями перехода. Ранее рассмотренные ограничения можно отнести к ограничениям состояния.

Ограничения перехода поддерживаются с помощью таких объектов БД, как триггеры и хранимые процедуры, либо в логике приложения. Наиболее эффективной является поддержка целостности средствами БД и СУБД, поскольку такая реализация ограничений описана на уровне БД и хранится в ней и может быть использована в дальнейшем любым пользователем БД.

#### 3.3.3.3 Проектирование индексов

Индексы используются СУБД для быстрого поиска строки в таблице. По своей организации индексы представляют, как правило, структуры данных в виде двоичных деревьев. Индексы позволяют избегать полных просмотров таблиц, снижающих производительность системы. Индекс хранит последовательность упорядочивания данных в поле (полях) таблицы по возрастанию или убыванию в виде физических номеров записей. Эти служебные системные номера генерируются СУБД по мере формирования каждой новой строки таблицы. Используя индекс, СУБД быстрее находит нужные данные, при этом сокращается физическое число перемещений физических устройств считывания данных между блоками памяти. Проектируя

индексы, необходимо помнить, что повальное использование индексов по принципу «чем больше – тем лучше» может затормозить выполнение запросов, оптимизатор запросов СУБД будет использовать первый индекс и, возможно, он не будет лучшим. Необходим здравый подход к созданию индексов.

Индексы необходимы, если:

— часто производится поиск в БД по определенному полю (полям) таблицы;

— часто выполняется операция объединения таблиц по значению поля (полям);

— часто выполняется операция сортировки по значениям поля (полей) в наборе данных, возвращаемых в результате запроса.

Не рекомендуется строить индексы по полям или группам полей, если:

— поля редко используются для выполнения операций поиска, объединения, сортировки;

— значения полей часто меняют свои значения;

— значения полей содержат небольшое число вариантных значений;

— значения полей имеют большие размеры;

— поля содержат большое количество пустых значений.

Для объявленных при создании таблиц первичного и уникальных ключей современные СУБД индекс создают автоматически. Некоторые СУБД, в частности *InterBase*, создают автоматически индексы и для объявленных внешних ключей. При выполнении запроса к БД, в условие поиска которого входит столбец таблицы, для которого создан индекс, поиск значений производится в первую очередь в индексе.

Различают следующие типы индексов:

— простые, строятся на основе только одного поля таблицы;

— составные – строятся по двум и более полям таблицы, при этом последовательность полей в составном индексе может быть не связана с последовательностью полей в таблице. Последовательность влияет на скорость поиска данных. Для создания оптимального составного индекса рекомендуется первым помещать поле, содержащее меньшее количество повторов, то есть содержащее наиболее ограничивающее значение, либо первым помещать поле, содержащее данные, которые наиболее часто задаются в условиях поиска;

— уникальные – используются дополнительно с целью поддержки целостности данных. Так, например, для создаваемой в среде СУБД *FoxPro* базы данных необходимо такие индексы строить для первичных и уникальных ключей, поскольку эта СУБД не поддерживает явно определение таких ключей.

При создании индексов следует поддерживать следующие рекомендации:

— нельзя создавать для таблицы несколько индексов, содержащих одинаковые, но расположенные в другом порядке поля, в случае необходимости индексы следует сокращать. Исключение составляют таблицы, разбивающие связь "многие_ко_многим";

— нельзя допускать, чтобы у нескольких индексов для одной и той же таблицы была одинаковая лидирующая часть.

Если проектировщиком приняты неудачные решения по индексированию, то впоследствии может значительно снизиться производительность системы, вследствие ненужной траты времени работы процессора на лишние операции чтения, записи, ненужную оптимизацию; неэффективного использования дискового пространства. Создание индекса может быть реализовано командой *Create Index*.

При многократном внесении и изменении данных в таблицах индексы, связанные с этой таблицей, могут быть разбалансированы, содержать неадекватный порядок следования записей. Поэтому периодически необходимо осуществлять улучшение производительности индекса, используя команды перестройки (деактивизации) индексов, например команду *Alter Index*.

Для оптимального использования индексов в конкретной СУБД необходимо пользоваться соответствующей технической документацией.

# 4 Создание БД

Перенос разработанной физической модели базы данных в среду конкретной выбранной СУБД называется процессом создания базы данных или её реализацией. Каждая конкретная инструментальная среда, каковой является система управления базами данных, требует особых знаний и навыков. Создание высокопроизводительной БД в выбранной среде потребует изучения соответствующей коммерческой технической документации. Рассмотрим общие подходы и технологии к процессу создания БД без особых подробностей их реализации.

В каждом конкретном случае создания БД средствами выбранной СУБД разработчик (администратор БД) должен уметь выбрать оптимальную стратегию размещения и хранения данных. Существует несколько показателей, которые могут быть использованы для оценки достигнутой эффективности:

— количество выполненных операций в БД (добавление, обновление, удаление, чтение строк таблиц) за заданный интервал времени;

— время ответа – временной промежуток, необходимый для выполнения определенного количества операций в БД;

— объем дискового пространства, необходимого для размещения файлов БД.

Ни один по отдельности из этих факторов не является самодостаточным, как правило, все усилия направлены на достижение оптимального баланса значений этих характеристик. После создания БД могут выполняться дополнительные настройки её конфигурации в процессе эксплуатации.

Основные шаги в ходе создания БД это: подготовка среды хранения, генерация схемы БД; загрузка и корректировка данных из старой БД; ввод и контроль данных в справочные таблицы.

## 4.1 Подготовка среды хранения

Необходимыми системными ресурсами для работы БД являются четыре основных компонента аппаратного обеспечения, которые будут взаимодействовать между собой и влиять на уровень производительности.

1 Процессор. Выполняет пользовательские процессы, управляет задачами других системных ресурсов. У него должна быть высокая тактовая частота. Для организации сервера БД может быть использовано несколько процессоров одновременно.

2 Оперативная память. Чем больше её объем, тем быстрее работают приложения баз данных. Рекомендуют планировать объем, чтобы в процессе работы системы выполнялось условие: всегда есть в наличии свободными 5 процентов от всего объема памяти.

3 Внешняя память. Для определения этого ресурса важным является количество операций ввода/ вывода в секунду. Также на общую производительность системы влияет способ организации в ней хранения данных. Рекомендуется равномерно распределять сохраняемые данные между

всеми доступными в системе устройствами. Существуют следующие принципы распределения данных на дисковых устройствах:

— файлы операционной системы должны быть отделены (располагаться физически на разных носителях) от файлов базы данных;

— основные файлы БД должны быть отделены от индексных файлов;

— журнал восстановления должен быть отделен от остальной части БД.

4 Сеть. При организации сетевого доступа к БД узким местом всей системы может стать чрезмерный рост сетевого трафика. Оптимальная загрузка сети должна быть реализована в клиентских приложениях БД.

Для планирования оптимальных характеристик среды хранения должны быть обсуждены также следующие вопросы.

1 Определение функциональных характеристик транзакций, которые будут выполняться в базе данных.

Транзакция – неделимый с точки зрения СУБД набор действий, выполняемых пользователем БД с целью доступа или изменений содержимого БД.

Изучаются качественные и количественные характеристики присущих проектируемой БД транзакций – ожидаемая частота выполнения; таблицы, поля таблиц и операции над данными транзакции, используемые индексы; ограничения, устанавливаемые на выполнение транзакции. Далее определяются самые «важные» (наиболее активные) транзакции, зависимость их друг от друга, возможные проблемы и конфликты. Разработчики БД должны найти оптимальные способы взаимодействия транзакций, для каждой транзакции определить максимальную производительность.

2 Выбор файловой структуры. Возможны следующие варианты типов файлов: последовательные файлы, хешированные файлы, индексно— последовательные файлы, двоичные деревья. Структура файлов изучается и документируется, обосновывается выбор конкретного варианта.

Необходимо отметить, что современные СУБД не предоставляют разработчику никаких возможностей выбора или изменения способа организации файлов БД.

3 Анализ необходимости введения контролируемой избыточности данных. В том случае, если требования к производительности системы не удается удовлетворить никакими другими способами, снижают требования к уровню нормализации данных. Такую процедуру называют оптимизацией использования. Денормализация целесообразна, если данные в БД обновляются редко, а количество выполняемых запросов велико. В противном случае требования к непротиворечивости данных должны превышать требования к производительности системы.

4 Определение требований к дисковой памяти. Выделяемая для хранения БД память должна иметь размер, позволяющий накапливать данные. Для каждой СУБД необходимо опытным путем определять возможный рост необходимого объема памяти.

Подготовка среды хранения является важной задачей, определяющей производительность дальнейшего использования БД.

## 4.2 Генерация схемы БД

Физически БД состоит из одного или нескольких дисковых файлов. Кроме файлов данных в состав БД могут входить файлы журналов транзакций, отслеживающие следы выполнения транзакций и дающие возможность отменить или восстановить транзакцию; управляющие файлы, которые поддерживают внутреннюю целостность данных; файлы – журналы предупреждающих сообщений и т.д. Совокупность всех файлов определяется выбранной для реализации БД СУБД.

База данных может быть создана различными способами – визуально в режиме диалога с СУБД, выполнением соответствующих программных кодов (*SQL*—скриптов), автоматической генерацией, на основе построенной в инструментальной среде *CASE* — средства модели предметной области.

В любой СУБД первой выполняется операция создания новой базы данных, а затем её объектов, при этом необходимо соблюдать последовательность их создания. Первым создается родительский объект, а затем на него ссылающийся. Так, создание дочерней таблицы первой, по отношению к родительской, вызовет соответствующую реакцию СУБД (сообщение), если связь в дочерней таблице прописана явно. В любом случае, связи создать нельзя, если отсутствуют элементы этой связи.

Самым простым является процесс создания БД, например, в СУБД *Access*. БД можно создать на основе предлагаемого шаблона, с использованием таких средств, как «Мастер» и «Конструктор». Наиболее сложным является процесс создания серверной БД, требующим определенный уровень квалификации разработчика.

## 4.3 Загрузка и корректировка данных из старой БД

Очень важной задачей при разработке новой автоматизированной информационной системы является поддержка накопленных данных, созданных на предприятии другими, унаследованными информационными системами, работавшими ранее или параллельно с разрабатываемой БД. Такая потребность возникает при смене технологий обработки данных, переводе задач на новое аппаратное или программное обеспечение. Кроме того, возможна периодическая загрузка данных в систему из какого—либо источника (например, общегосударственного классификатора), поскольку работать в полной изоляции от окружающего мира могут только редкие АИС. Для переноса данных из старой БД в новую разрабатываются специальные программы—конверторы, которые осуществляют такую выгрузку данных из старой структуры и загрузку в новую этих же данных. При этом формат старых данных преобразуется, как правило, в соответствии с требованиями новой системы. Например, необходимо осуществить загрузку данных из файлов формата *dbf*, полученных ранее в среде какой—либо *dBase* — подобной СУБД, в таблицы БД, созданной средствами серверной СУБД. Для того чтобы осуществить успешную конвертацию данных, во—первых, необходимо понять

смысл этих данных с точки зрения предметной области. Затем необходимо изучить логическую структуру каждой *dbf* таблицы (каждый *dbf* файл представляет собой отдельную таблицу). Поскольку это файл БД (созданный средствами СУБД), то он в себе содержит не только данные, но и описание этих данных. Первые 32 байта *dbf* файла описывают характеристики самой таблицы – дату последней корректировки, число записей, размер записи, наличие Memo полей в таблице. Далее идет описание полей записей. Каждое поле описывается 32 байтами – имя, тип, размер поля. Длина описания всех полей будет равна 32 * *n* байта, где *n* – количество полей таблицы. За описанием данных начинается непосредственно область данных. Таким образом, зная структуру файла, можно последовательно считать данные и затем записать их в файл формата другой СУБД. При этом необходимо помнить, что в *dbf* файлах нет описания как первичных, уникальных, так и внешних ключей. Поэтому при загрузке данных в новую среду необходимо позаботиться о поддержке ограничений целостности новой БД.

Для написания программы—конвертора можно использовать любую среду. Запись в БД должна осуществляться средствами СУБД, в формат которой преобразуются данные. При рассмотрении задач, связанных с выгрузкой и загрузкой данных, проектировщик программы—конвертора должен:

— определить каким системам (передающей, принимающей) нужен интерфейс;

— определить периодичность и объем передаваемых данных;

— установить степень синхронизации данных;

— исследовать и определить методы транспортировки данных (файловые, коммуникационные и др.);

— согласовать с проектировщиками систем (передающей, принимающей) формат данных для обмена;

— определить порядок передачи данных при загрузке и выгрузке;

— установить правила обработки входных данных, которые частично разрушены или утратили целостность;

— для каждой загрузки составить планы перехода в аварийный режим и режим восстановления на случай неудачного и неполного приема данных.

При переносе данных из старой системы в новую может возникнуть проблема несовместимости данных и попытки переноса в новую систему «неочищенных данных». Это возможно, например, в случае, если старая система содержит не полностью нормализованные данные, либо если ограничения целостности не реализованы на уровне базы данных. При переносе данных в новую систему необходимо бороться за их чистоту. Затраты времени на это затем полностью окупаются.

### 4.4 Ввод и контроль данных в справочные таблицы

Эксплуатация БД начинается с ввода в неё учетных данных, только затем возможна реализация операций по их обработке. Учетные данные, как правило, сохраняются в рабочих, дочерних таблицах. Для нормального начала ввода учетных данных необходимо наличие первоначальной информации в справочных, родительских таблицах. Функция первоначального наполнения справочных таблиц возлагается на администратора БД, затем она может быть осторожно, с соблюдением всех требований предметной области, передана пользовательским приложениям.

Информация в ряде справочников может соответствовать различным свободно распространяемым государственным классификаторам – перечень населенных пунктов, улиц, научных специальностей и т.п. В таком случае необходимо реализовать функцию загрузки таких данных из доступных классификаторов в справочники и определить технологию (сроки и порядок) постоянного обновления справочных данных.

### 4.5 Словарь данных

Словарь данных, или системный каталог является хранилищем информации, содержащей данные в базе данных (данные о данных, метаданные). Наличие словаря данных обуславливает независимость разрабатываемых в рамках автоматизированной информационной системы прикладных программ от данных и наоборот.

В зависимости от используемой СУБД, количество информации в системном каталоге и способы её использования могут изменяться. Обычно в словаре данных хранятся описания структур объектов БД:

— имена, типы и размеры элементов данных (таблицы, поля таблиц);
— имена связей;
— имена, описание структур объектов БД;
— накладываемые на данные ограничения целостности;
— имена пользователей, которые имеют права доступа к объектам БД;
— пользовательские ограничения;
— и другое.

Любая СУБД, в которой создается словарь данных, имеет средства (комплект утилит) для просмотра и работы с метаданными. Так, например, работая с локальным сервером *InterBase*, средствами утилиты *IBConcole* можно просмотреть перечень объектов БД и их структуру – пункты меню *Database*, *Veiw Metadata*. СУБД, поддерживающие работу со словарем данных, имеют также таблицу, в которой есть описание структуры и словаря. Это делает возможным просмотр метаданных с использованием команды *Select*.

# 5 Администрирование БД

Любая база данных нуждается в администрировании. Задачи администрирования различаются по степени сложности в разных инструментальных средах, их объем зависит от того, к какой категории относится БД: локальная или многопользовательская.

Одной из основных функций администратора БД является упрощение разработки и использования базы данных. Как правило, это означает поддержку баланса между защитой базы данных и максимизацией её доступности и выгод от её использования.

Рассмотрим основные функции администратора БД

## 5.1 Управление структурой БД

Администратор базы данных принимает активное участие в проектировании и реализации БД и далее, в ходе эксплуатации, следит, контролирует и управляет созданной структурой.

Требования к структуре БД могут меняться в процессе её эксплуатации. Это может быть обусловлено рядом причин: изменением или расширением автоматизируемых бизнес—процессов предприятия, реструктуризацией БД вследствие её дополнительной нормализации или денормализации, реформатизацией БД (переход в среду другой СУБД).

АБД, обладающий полными правами на структуру БД, выдает права на управление схемами (подсхемами) БД прикладным программистам в рамках решаемых ими задач. Прикладной программист, владея схемой, управляет структурой БД в рамках этой схемы. Администратор базы данных участвует в проектировании новых фрагментов БД, согласуя эти новые фрагменты с существующей, основной частью схемы БД. В обязанности АБД входит также документирование процесса управления базой данных. Важно знать, какие модификации были произведены, когда и каким образом. Это может быть реализовано с помощью создания соответствующих объектов БД. Например, триггеров, создающих записи в специальных таблицах в ответ на какие—либо действия с таблицей, для которой создан триггер. Такое фиксирование помогает решить ряд проблем при возникновении конфликтных ситуаций.

## 5.2 Защита данных

Важными факторами, влияющими на работу БД в сети, в которой хранится большое количество совместно используемых данных, являются целостность, стабильность и надежность данных. Обобщение всех вопросов, связанных с этими тремя факторами называется защитой данных. Обеспечение защиты данных – одна из основных задач при проектировании и эксплуатации АИС. Способы обеспечения защиты данных при выполнении операций над данными имеют общее название управление данными. Обеспечение защиты данных является прямой функцией администратора БД

Защита данных рассматривается в 2—х аспектах.

1 Управление доступом – защита данных от несанкционированного (преднамеренного, незаконного) доступа, поддержка функции секретности БД.

2 Управление целостностью – защита данных в БД от неверных, непреднамеренных изменений и разрушений – поддержка функции безопасности БД. Обеспечение целостности данных и целостности всей БД — часть решения этой проблемы

Поддержание целостности состоит в обеспечении правильности БД в любой момент времени и рассматривается, в свою очередь, в следующих аспектах.

1 Обеспечение достоверности. Состоит в предотвращении возможности появления недопустимых значений данных из—за ошибки в нарушении ограничений целостности БД. Задача решается как средствами СУБД, так и приложений.

2 Управление параллелизмом. Управление данными должно осуществляться таким образом, чтобы при одновременном выполнении нескольких операций целостность БД не нарушалась.

3 Восстановление. При возникновении неисправности со стороны программного или аппаратного обеспечения системы необходимо иметь возможность за предельно короткое время восстановить то состояние БД, которое было перед сбоем.

Вопросы и управления доступом и управления целостностью тесно соприкасаются между собой, и во многих случаях для их решения могут быть использованы одни и те же механизмы. Различие между этими понятиями: управление доступом – это защита БД от преднамеренного (поддержка функции секретность) разрушения; управление целостностью — это защита БД от непреднамеренного (поддержка функции безопасности) разрушения. В целом, система безопасности, как и цепь, сильна ровно настолько, насколько сильно ее слабейшее звено.

Существуют компьютерные и некомпьютерные методы и приемы защиты данных. Вначале рассмотрим компьютерные средства контроля.

### 5.2.1 Авторизация пользователей

Авторизация – предоставление прав (или привилегий), позволяющих владельцу иметь законный доступ к системе или к её объектам. Механизм авторизации часто называют подсистемой управления доступом, которая также требует проектирования. Процесс авторизации включает в себя идентификацию и аутентификацию пользователей.

Идентификация — это точное установление личности пользователя на основании различных признаков. Использование пароля – идентификация пользователя по одному признаку. Идентификация позволяет системе установить имя пользователя.

Аутентификация — процедура проверки прав пользователя на доступ к данным и прав выполнения определенных действий с этими данными.

Аутентификация не может определить личность пользователя, зато может точно указать, какими правами он обладает

Администратор БД каждому созданному в БД пользователю присваивает уникальный идентификатор (логин), с каждым идентификатором связывается пароль, известный только пользователю. СУБД проверяет идентификатор и пароль пользователя при соединении пользователя с БД посредством утилит СУБД или приложения, которое настроено на соединение с БД.

Алгоритм реализации процедуры проверки прав доступа пользователей может быть разным. Необходимо использовать все имеющиеся в СУБД средства для реализации точной настройки прав доступа пользователя, но не в коем случае не зашивать пароль пользователя в код прикладной программы.

Более тонкая настройка уровней доступа пользователей возможна средствами языка *SQL*. Для этого можно использовать такие объекты БД как представления (просмотры, *view*).

Представление – это динамический результат одной или нескольких реляционных операций с базовыми (хранимыми в БД) отношениями с целью создания некоторого иного отношения (виртуального), которое реально в БД не существует, но создается по требованию отдельных пользователей. Представление может быть определено на базе нескольких таблиц, после чего пользователю будут предоставлены необходимые привилегии доступа к этому представлению. Это жесткий механизм контроля доступа – пользователь не будет иметь никаких сведений о строках и столбцах таблиц, не включенных в представление.

Привилегия доступа – возможность определенного пользователя (группы пользователей) выполнять определенный вид действия над определенными объектами БД. Привилегии устанавливаются АБД или другим пользователем (прикладным программистом), которому АБД предоставил такое право. *SQL* команда установки привилегий выглядит следующим образом: *Grant <вид привилегии> On <имя таблицы/имя представления> To <объект/список пользователей>*.

Существуют следующие виды привилегий: — *All* – выполнение операций *Select*, *Delete*, *Insert*, *Update*, *Execute*; выполнение операции *Select*; выполнение операции *Delete*; выполнение операции *Insert*; выполнение операции *Update*; выполнение операции *Execute* (обращение к хранимой процедуре).

Объект это: процедура; триггер; представление (просмотр); пользователь и др.

По умолчанию доступ к таблицам и хранимым процедурам имеет только тот пользователь, который их создал. Естественно, АБД имеет доступ ко всем объектам БД. Для предоставления пользователю привилегии доступа к таблице в операторе *Grant* необходимо указать, как минимум, следующие параметры: ключевое слово, обозначающее вид привилегии доступа; имя таблицы; имя пользователя.

Пример 1. Предоставление пользователю *IvanovVV* права на выполнение поиска данных в таблице с названием *Adres*: *Grant Select On Adres To IvanovVV*. Пример команды создания пользователя: *Create User IvanovVV Identified By* <пароль>.

Пример 2. Дать право выполнять все операции всем пользователям в таблице *Adres*: *Grant Select, Insert, Update On Adres To Public*.

Пример 3. Предоставить пользователям *IvanovVV* и *PetrovKL* добавлять и менять значения столбцов *A1* и *A2* в таблице *Adres*: *Grant Update (A1,A2) On Adres To IvanovVV, PetrovKL*.

Команда создания представления выглядит следующим образом: *Create View <имя таблицы [(список столбцов)]> As (<Select >)*. Например, команда *Create View «Представление1» («Название») As Select «Название» From «Должность» Where «Код категории» = 5*, создает динамическую таблицу с именем «Представление1», в которой будет формироваться столбец с названием должностей из таблицы «Должность», поле «Код категории» которых равен 5.

Пример использования представления в команде назначения прав: *Grant Select On «Представление1» To «Пользователь1», «Пользователь2»*. После выполнения команды пользователям «Пользователь1» и «Пользователь2» будут даны права на чтение названий должностей, относящихся к 5 категории.

Данные представления можно обновлять, т.е. применять к этому объекту БД операции добавления, изменения, вставки и удаления записей. Тогда пользователь, имеющий права на работу с этим представлением, сможет вследствие получить права и на операции, определенные для этого представления. При этом некоторые СУБД требуют выполнения трех условий: представление должно формироваться из записей только одной таблицы; в представление каждый столбец должен иметь опциональность *Not Null*; оператор *Select* представления не должен использовать агрегирующих функций, режима *Distinct*, предложения *Having*, операций соединения таблиц, хранимых процедур и функций, определенных пользователем.

### 5.2.2 Управление параллельно работой пользователей

Меры по управлению параллельной обработкой предотвращают непредусмотренное влияние действий одного пользователя на другого. Иногда необходимо, чтобы в условиях параллельной обработки пользователь получил тот же результат, как и в случае, если бы он был единственным пользователем, в других случаях подразумевается, что действия различных пользователей будут влиять друг на друга, но ожидаемым образом.

#### 5.2.2.1 Транзакции

Любая СУБД, поддерживающая работу в многопользовательском режиме, должна иметь механизм поддержки транзакций. Транзакция (transaction) может содержать одну или несколько *SQL* команд и

рассматривается СУБД как единое целое. Транзакция выполняется одним пользователем, который имеет соответствующий доступ. С точки зрения БД выполнение какой—либо программы может расцениваться как серия транзакций.

Пример транзакции: выбор сведений из БД о заданном физическом лице; изменение паспортных данных; подтверждение изменения (сохранение).

Существуют некоторые свойства, которыми должна обладать любая транзакция.

1 Атомарность (*atomicity*). Предполагает, что транзакция должна быть либо завершена, либо не выполнена вовсе. Входящие в транзакцию операции выступают вместе как неделимая единица работы, т.е. либо все операции успешно завершаются, либо отменяются.

2 Непротиворечивость, постоянство (*consistency*). После завершения транзакции система должна находиться в известном состоянии, т.е. транзакция не должна оставлять после себя следов. С точки зрения стороннего наблюдателя, система находится в непротиворечивом состоянии и до начала, и после завершения транзакции.

3 Изолированность (*isolation*). Транзакция должна быть изолирована, т.е. не должна влиять на другие транзакции и зависеть от них. Зависимость вызывает тупиковые ситуации (*deadlocks*), т.е. в ходе выполнения одна транзакция не должна «видеть» изменений, сделанных другими незавершенными транзакциями.

4 Устойчивость, продолжительность, долговечность (*durability*). Если транзакция завершена, и цель её достигнута, то не может быть никаких веских причин для её отката. По этой причине важные операции с таблицей выполняются за одну транзакцию. Должна обеспечиваться независимость сохранности изменений, совершенных транзакцией, и при сбоях аппаратного обеспечения АИС (внезапная перезагрузка, поломка оборудования).

Существуют некоторые действия, которые нельзя выполнить в транзакции. Их либо выполняет АБД, либо они фиксируются в журнале транзакций для возможного дальнейшего дословного восстановления системы. Это действия: изменение структуры объектов БД; создание объектов БД; переконфигурация системы; обновление статистики работы системы. Перед выполнением этих операций предварительно делается резервная копия БД.

В большинстве языках манипулирования данными разных СУБД для указания границ отдельных транзакций используются операторы: начать транзакцию (*Begin Transaction*) – явный запуск транзакции; завершить транзакцию (*Commit*); отменить транзакцию (*Rollback)*.

В стандарте *SQL* указывается, что транзакция запускается любым *SQL* оператором, выполняемым пользователем или программой (*Select*, *Insert*, *Update*), это, так называемый, неявный запуск транзакции. Вся выполняемая программа может рассматриваться СУБД как единая транзакция.

Завершение транзакции может быть выполнено одним из 4—х способов.

1 Ввод оператора *Commit* означает успешное завершение транзакции. После его выполнения, внесенные в БД изменения, приобретают постоянный характер.

2 Ввод оператора *Rollback* означает отказ от завершения транзакции, в результате чего выполняется откат всех изменений в БД, внесенных при выполнении этой транзакции.

3 При внедрении *SQL* команды в текст программы успешное окончание работы программы автоматически вызовет завершение последней запущенной транзакции, даже если оператор Commit не был явно введен.

4 При внедрении *SQL* команды в текст программы аварийное завершение этой программы автоматически вызовет откат последней запущенной этой программой транзакции.

Зафиксированная транзакция не может быть отменена. Если оказалось, что зафиксированная транзакция была ошибочной, потребуется выполнить другую транзакцию, отменяющую действие первой.

Более подробно технологию работы с транзакциями необходимо изучать в технической документации выбранной СУБД.

При работе с БД в многопользовательском режиме необходимо знать о проблемах, которые могут возникнуть в связи с параллелизмом процессов. Термин параллелизм означает, что СУБД имеет возможность одновременно обрабатывать много транзакций, осуществляющих доступ к одним и тем же данным в одно и то же время.

### 5.2.2.2 Проблемы, возникающие при параллельной обработке данных

Пока пользователи читают одновременно данные, никаких проблем не возникает, они начинаются, когда происходит одновременное обращение к одним и тем же данным для их корректировки, т.е. основная проблема при работе с БД в многопользовательском режиме — обеспечение целостности БД при одновременном обращении к одним и тем же данным с корректирующими действиями. Возможно появление взаимного влияния процессов друг на друга, способное привести к несогласованности данных.

1 Проблема утраченного обновления. Параллельно выполняются два процесса обработки одних и тех же данных *Д* пользователем *А* и пользователем *В*, т.е. выполняются две, независимые друг от друга транзакции, реализующие обновление одних и тех же данных *Д* (рисунок 34).

| Пользователь *А* | | Пользователь *В* |
|---|---|---|
| Выборка *Д* | момент времени $т_1$ | |
| | момент времени $т_2$ | Выборка *Д* |
| Обновление данных *Д* | момент времени $т_3$ | |
| | момент времени $т_4$ | Обновление данных *Д* |
| ***! Утраченное обновление*** | момент времени $т_5$ | |

Рисунок 34 – Проблема утраченного обновления

И для пользователя *B*, и для системы в целом, обновление, сделанное пользователем *A*, будет утрачено (пользователь *B* его не увидит и сделает своё обновление, не учитывая обновление, сделанное пользователем *A*). Такая ситуация называется проблемой утраченного обновления.

2 Проблема зависимости от незафиксированных обновлений. Параллельно выполняются два процесса обработки одних и тех же данных *Д* – рисунок 35.

| Пользователь *A* | Пользователь *B* | |
|---|---|---|
| | момент времени $т_1$ | Обновление *Д* |
| Выборка *Д* | момент времени $т_2$ | |
| | момент времени $т_3$ | ***! Откат (возврат к пред. состоянию)*** |

Рисунок 35 – Проблема зависимости от незафиксированного обновления

Пользователь *A*, если ему разрешен просмотр этих же данных, в момент времени $т_2$ увидит данные, которые окончательно не были зафиксированы в системе, и вполне может быть, что они ошибочные. Такая ситуация называется "зависимость от незафиксированных обновлений». Транзакция *A* становится зависимой от не сохраненного обновления данных *Д* в момент $т_2$, т.е. в транзакции *A* принимают участие данные, которые больше не существуют.

3 Проблема несовместимого анализа. Параллельно выполняются два процесса обработки одних и тех же данных *Д*. Один пользователь создает отчет, который содержит как детальные, так и итоговые строки. Второй в это время изменяет данные, влияющие на формирование итоговых строк. Могут наблюдаться противоречия в отчете: сумма детальных строк может быть не равна итоговой цифре — целостность БД не нарушается, но документ неприемлем. Такая ситуация называется "проблемой несовместимого анализа".

Для корректной обработки параллельных транзакций без создания конфликтных ситуаций необходимо использовать некоторый метод управления параллелизмом. Методы могут быть пессимистическими (консервативными), поскольку они откладывают выполнение транзакций, способных в будущем создать конфликтную ситуацию и оптимистическими – строятся на предположениях, что конфликтная ситуация маловероятна, поэтому они допускают асинхронное выполнение транзакций в БД, а проверка на наличие конфликта откладывается на момент их завершения и фиксации данных в БД.

#### 5.2.2.3 Блокировка данных

Блокировка – процедура, используемая для управления параллельным доступом к данным. Механизм блокировки позволяет некоторой транзакции путем захвата необходимого элемента данных (от БД, в целом, до отдельного поля конкретной записи) получить доступ к данным, отклоняя попытки

получения такого же доступа со стороны других транзакций, пока она их не разблокирует. Это пессимистический метод управления параллелизмом.

Существует несколько различных вариантов этого механизма, но все они основаны на одном главном принципе: транзакция должна потребовать выполнить блокировку для чтения (разделяемую) или для записи (эксклюзивную) некоторого элемента данных, перед тем, как она сможет выполнить в БД соответствующую операцию чтения или записи.

Для реализации механизма блокировок БД разбивается структурно на элементы, которые можно блокировать. Выбор характера и размера элементов, подлежащих блокированию, выполняется исходя из специфики решаемой задачи.

Влияние размеров блокируемых элементов на показатели производительности БД представлены в таблице 29.

Таблица 29 – Влияние размеров блокируемых элементов

| Элементы, подлежащие блокировке/ Показатели | Большие по размеру (таблица) | Малые по размеру (запись, поле) |
|---|---|---|
| 1. Накладные расходы по поддержке блокировок | снижаются | увеличиваются |
| 2. Возможность параллельного использования процессов | снижается | расширяются |

Размер блокируемого элемента данных может быть определен в настройках среды разработчика (например, свойство соответствующего компонента *Delphi*), либо с помощью команд СУБД. Существуют разные модели блокировок.

1 Простая модель. Не вводится различие блокировок для операторов чтения и записи элемента данных *Д*. Используются две команды: установить блокировку элемента *Д*; снять блокировку элемента *Д*. При установке блокировки элемента *Д* предотвращается к нему доступ от других транзакций, как для выполнения операции чтения, так и для выполнения операции записи до тех пор, пока элемент *Д* не будет разблокирован. Это эксклюзивная или монопольная блокировка, она, как правило, автоматически устанавливается СУБД при выполнении команд *Insert*, *Update*.

2 Модель с блокировками для чтения и записи. В данной модели определено различие между видами доступа к элементу *Д*: доступ только для чтения; доступ только для записи. В связи с эти различают 2 типа команд блокировки:

а) команда блокировки элемента *Д* по записи. Такая блокировка запрещает любой другой транзакции выполнить запись нового значения элемента *Д*, пока он не будет разблокирован. Допускается параллельные операции по чтению элемента *Д* другими транзакциями;

б) команда блокировки элемента *Д* по чтению и записи. Эта команда соответствует команде блокировки простой модели, т.е. предотвращает доступ к элементу *Д* всем другим транзакциям.

В модели с блокировками по чтению допускается ситуация, когда транзакция может вначале установить блокировку элемента *Д* по записи, а затем блокировку элемента *Д* по чтению и записи.

Все блокировки снимаются командой «снять блокировку».

Любая блокировка осуществляются по следующим правилам.

1 Любая транзакция, которой необходимо получить доступ к элементу данных, должна вначале выполнить блокировку этого элемента. Блокировка может запрашиваться для чтения или для записи (если запись, то, естественно, и чтение).

2 Если элемент ещё не заблокирован какой—либо транзакцией, то блокировка будет выполнена успешно.

3 Если элемент данных уже заблокирован, СУБД анализирует, является ли тип полученного запроса совместимым с типом уже существующего блока. Если запрашивается доступ для чтения к элементу, который заблокирован для чтения, доступ к нему будет разрешен. В противном случае транзакция будет переведена в состояние ожидания, которое будет продолжаться до тех пор, пока существующий блок не снят.

4 Транзакция продолжает удерживать блокировку элемента данных до тех пор, пока она явным образом не освободит его – либо в ходе выполнения транзакции, либо по её окончании (успешном или неуспешном). Только после того, как с элемента данных будет снята блокировка для записи, другие транзакции смогут «увидеть» результаты проведенной операции записи.

Для обеспечения нормальной работы системы необходимо помнить и выполнять некоторые правила.

1 Не захватывать данные на более длительный период, чем объективно необходимо. Например, не полагаться на автоматическое разблокирование, а управлять разблокированием.

2 Открывать БД в режиме разделения, если изменению данных предшествует иное их использование, а захват БД осуществлять позднее.

3 Стремиться к полной или временной замене каждого полного захвата БД захватом отдельных записей или групп записей.

4 Немедленно отменять все свои блокирования, если неуспешна попытка очередного блокирования и осознана бесполезность дальнейшего ожидания.

5 Исследовать вероятность тупиков и обдумывать меры их предотвращения. Это делается в кругу пользователей с участием администратора БД.

6 Отменить блокирование может только владелец. Отмена может быть неявной — новое блокирование отменяет предыдущее.

Политика блокировок может поддерживаться самой СУБД с помощью соответствующих настроек. СУБД автоматически блокирует определенные элементы данных при выполнении определенных операторов. Если разработчика не удовлетворяют блокировки по умолчанию, он может управлять ими сам с помощью команд СУБД.

### 5.2.2.4 Бесконечные ожидания и тупики

Использование блокировок может привести к различным нежелательным эффектам, таким как бесконечные ожидания и тупики.

Пример ситуации бесконечного ожидания приведен на рисунке 36.

| Транзакция *T2* | время | Транзакция *T1* | Транзакция *T3* | Транзакция *T4* |
|---|---|---|---|---|
| | **$T_1$** | Блокиро—вание *Д* Чтение *Д* | | |
| Попытка блокиро—вания Д – отказ ***ожидание..*** | **$T_к$** | | | |
| **.** | **$T_{к+1}$** | Разблоки—рование *Д* | | |
| | | | Блокиро—вание *Д* (раньше, чем *T2*) Чтение *Д* | |
| | | | | Блокиро—вание *Д* – отказ ***ожидание*** **…** |

Рисунок 36 – Пример ситуации бесконечного ожидания

Предположим, что транзакции *T1, T2, T3, T4* исполнения программы, содержащей следующие действия: блокирование *Д*; чтение *Д*; изменение *Д;* подтверждение сохранения *Д*; разблокирование *Д*. Не исключена возможность того, что транзакция **T2** будет бесконечно находиться в состоянии ожидания, тогда как некоторые другие транзакции постоянно осуществляют блокировку *Д*, хотя и существует неограниченное число моментов, когда *T2* имеет шансы заблокировать *Д*. Состояние такого рода называется бесконечным ожиданием. Подобная проблема потенциально может возникнуть в любой обстановке, предусматривающей параллельное исполнение процессов (задача продажи билетов).

Теоретиками предлагаются различные пути решения этой проблемы. Простой способ заключается в том, что система предоставления блокировок должна регистрировать все неудовлетворенные немедленно запросы и предоставлять возможность блокировки элемента *Д* после его разблокирования первой запросившей его транзакции из числа ожидающих. Стратегия «первый вошел – первым обслужился» устраняет бесконечное ожидание, однако она может привести к тупикам.

Пример ситуации тупика приведен на рисунке 37. Есть две транзакции: транзакция *T1*, содержащая команды, работающие с элементами данных *A* и *B*: *LOCK(A); LOCK (B); …; UNLOCK (A); UNLOCK (B)* и транзакция *T2*: LOCK(*B*); *LOCK (A); …; UNLOCK (B); UNLOCK (A)*. При этом не имеет значения, для каких процессов требуются элементы данных *A* и *B* в транзакциях *T1* и *T2* (команда *LOCK* – заблокировать элемент данных, команда *UNLOCK* – разблокировать элемент данных).

| Транзакция *T1* | время | Транзакция *T2* |
|---|---|---|
| *LOCK(A)* | $T_1$ | *LOCK(B)* |
| *LOCK(B)* НЕТ! | $T_2$ | *LOCK(A)* НЕТ! |
| | $T_3$ | |
| ***ожидание*** | | ***ожидание*** |

Рисунок 37 – Пример ситуации тупика

Каждая из транзакций ожидает, пока другая разблокирует требуемый для неё элемент. Ожидание будет бесконечным. Ситуация, при которой каждая из множества двух или более транзакций ожидает, когда ей будет предоставлена возможность заблокировать элемент, заблокированный в данный момент времени какой—либо иной транзакцией из рассматриваемого множества, называется тупиком.

Способы предотвращения тупиков являются объектом исследования в области теории БД. Предлагаются различные методы разрешения тупиков.

1 Выполняется линейное упорядочивание элементов по какому—либо признаку (например, последовательно перечисляются все элементы, подлежащие блокированию) и вводится системное требование на составление запросов (программ): все запросы должны выполнять блокировки в этом порядке.

2 Вводится системное требование, чтобы в каждом запросе (программе) каждой транзакции все требуемые блокировки запрашивались сразу. Это позволяет СУБД управлять транзакциями без тупиковых ситуаций.

3 Никакие системные требования не вводятся. СУБД просто следит за возникновением тупиковых ситуаций. При обнаружении тупика действие одной из транзакций останавливается, все выполненные ею изменения в БД устраняются, транзакция переводится либо в состояние ожидания, либо полностью аннулируется. При этом все данные об этой транзакции СУБД фиксирует в системных журналах для возможного последующего перезапуска.

### 5.2.2.5 Уровни изоляции транзакций

Блокировки предотвращают потерю изменений при параллельной обработке данных. Но существует ряд проблем, которые нельзя решить с помощью блокировок: «грязное чтение», «невоспроизводимое (неповторяемое) чтение», «фантомное чтение».

1 «Грязное чтение» (*dirty reads*). Одна транзакция изменяет некоторые данные, но еще не завершается. Другая транзакция читает эти же данные (с изменениями, внесенными первой транзакцией) и принимает на их основе какие—то решения. Первая транзакция выполняет откат. В результате решение, принятое второй транзакцией будет основано на неверных данных.

2 «Невоспроизводимое (неповторяемое) чтение» (*nonrepeatable reads*). Одна транзакция в ходе своего выполнения несколько раз читает одни и те же данные *Д*. Другая транзакция в интервалах между этими чтениями изменяет данные *Д* и успешно заканчивается. В результате получится, что чтения, осуществляемые первой транзакцией, дают разные результаты.

3 «Фантомное чтение» (*phantoms reads*). Одна транзакция в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Другая транзакция в интервалах между этими выборками добавляет или удаляет строки или изменяет столбцы некоторых строк, используемых в критериях выборки первой транзакции, и успешно заканчивается. В результате получится, что одни и те же выборки в первой транзакции дают разные множества строк.

Для определения допустимости тех или иных проблем можно определить уровни изоляции транзакций. Такие настройки можно сделать как в среде многопользовательской СУБД, так и в среде разработки приложений. Например, в среде *Delphi* установка уровней изоляции транзакций определяется свойством компонента *Tdatabase property TransIsolation: TtransIsolation*. Разработчик может указать желаемый уровень, а СУБД будет управлять транзакциями в соответствии с этими указаниями.

Уровни изоляции транзакций определяют:

— могут ли другие (конкурирующие транзакции) вносить изменения в данные, изменяемые текущей транзакцией;

— может ли текущая транзакция видеть изменения, произведенные конкурирующими транзакциями и наоборот.

В стандарте языка *SQL* от 1992 года определяется четыре уровня изоляции для транзакций. Каждый уровень изоляции определяет действия, которые недопустимы при выполнении параллельных транзакций. Более высокие уровни изоляции включают все ограничения, установленные на более низких уровнях.

1 Уровень изоляции Read *Uncommetted* (*RU*) — незавершенное, грязное чтение. На этом уровне запрещается изменение данных со стороны других транзакций, если эти данные модифицируются еще не окончившейся транзакцией. Иначе говоря, другие транзакции блокируются по записи для этих данных до тех пор, пока не окончится текущая транзакция. Однако другим

транзакциям разрешается считывать еще не подтвержденные данные, что классифицируется как «грязное чтение».

2 Уровень *Read Commetted* (*RC*) — чтение данных. На этом уровне запрещается грязное чтение.

3 *Repeatable Reads* (*RR*) — воспроизводимое чтение. На этом уровне запрещается «грязное чтение» и «невоспроизводимое (неповторяемое) чтение».

4 *Serializable* (*S*) — сериализуемость. На этом уровне запрещается «фантомное чтение».

В таблице 30 приведены уровни изоляции транзакций и проблемы, которые они решают.

Таблица 30 – Уровни изоляции

| Тип Проблемы | Уровень изоляции | | | |
|---|---|---|---|---|
| | *RU* | *RC* | *RR* | *S* |
| Грязное чтение | *Возможно* | *Невозможно* | *Невозможно* | *Невозможно* |
| Невоспроиз—водимое чтение | *Возможно* | *Возможно* | *Невозможно* | *Невозможно* |
| Фантомное чтение | *Возможно* | *Возможно* | *Возможно* | *Невозможно* |

Для использования уровней изоляции транзакций в конкретной СУБД необходимо изучение соответствующей технической документации.

### 5.2.3 Управление восстановлением БД

Восстановление БД это процесс возвращения базы данных в корректное состояние, утраченное в результате сбоя или отказа. Существует множество различных типов отказов: сбои аппаратного и программного обеспечения системы, преднамеренных и непреднамеренных действий пользователей. Например, прикладной программист не реализовал в приложении обработку ситуации временного отключения клиентского приложения от сервера (сетевой сбой), вследствие чего конечный пользователь осуществляет ввод большого объема данных, не зная о том, что данные не будут сохранены на сервере. Причиной любого сбоя или отказа системы возможны такие последствия, как утрата данных в оперативной памяти, утрата данных на диске.

Когда происходит сбой, невозможно просто устранить проблему и продолжить работу БД. При многопользовательском режиме работы состояние системы с того места, как она была прервана, воспроизвести в точности практически невозможно.

Стратегия восстановления БД в случае сбоя или отказа системы должна быть спроектирована и реализована разработчиками БД и отражены в соответствующей технической документации.

### 5.2.3.1 Резервное копирование БД

Современные СУБД содержат набор различных средств, позволяющих делать резервные копии базы данных, а также восстанавливать её в случае необходимости. Существует три стандартных способа резервного копирования БД: экспорт, автономное резервное копирование и оперативное резервное копирование.

Экспорт представляет собой логическое копирование БД, два остальных способа – физическое копирование файлов.

Надежная стратегия резервного копирования опирается и на физическое, и на логическое резервное копирование. Как правило, промышленные БД используют в качестве основного метода физическое резервное копирование, а логическое служит вспомогательным методом. Для небольших БД и БД, где данные перемещаются незначительно, больше подходят операции логического резервного копирования.

Логическое резервное копирование базы данных предполагает чтение её записей и внесение их в файл. Записи считываются независимо от их физического расположения. При этом происходит обращение, как к данным, так и к словарю данных. Можно экспортировать всю БД, конкретные подсхемы или конкретные таблицы. В процессе экспорта можно также решить, следует ли экспортировать связанную с таблицами информацию словаря данных, такую как привилегии, индексы, ограничения. Созданные в процессе экспорта файл будет содержать команды, необходимые для полного воссоздания всех выбранных для экспорта объектов. Экспортированные данные не обязательно должны быть импортированы в ту же самую базу данных или в туже схему. С помощью этого файла можно создать копию экспортированных объектов в другой схеме или в другой БД. При реализации импорта данных возможно определить – все данные будут импортированы или необходимая их часть.

В ходе операций физического резервного копирования файлы БД копируются независимо от их логического содержания. Эти копии называют резервными копиями файловой системы. Различают два типа физического копирования файлов – автономное (холодное копирование) и оперативное (горячее копирование). Автономное копирование выполняется при нормальной остановке БД. После её отключения копируются следующие файлы: все файлы данных; все управляющие файлы; все оперативные журналы и т.д. Получают полный образ БД на момент её останова. Все файлы впоследствии можно извлечь из резервной копии и база данных снова будет работать. Оперативное резервное копирование можно осуществлять для любой БД, работающей в открытом режиме, это необходимо для баз данных, остановка которых невозможна. Оперативное резервное копирование позволяет впоследствии осуществить полное восстановление информации с привязкой ко времени.

### 5.2.3.2 Способы восстановления БД

Одна из основных функций администратора БД – быть готовым к возможному отказу системы. В случае возникновения отказа база данных должна быть восстановлена быстро и с минимально возможными потерями. Процесс восстановления БД требует от АБД:

— определения, какие структуры базы данных затронуты и требуют восстановления;

— выполнения соответствующих шагов по восстановлению;

— рестарта экземпляра БД для восстановления его нормальной работоспособности;

— проверки, что в базе данных не остались некорректные данные, и действия пользователей не пропали.

Цель этих мероприятий – наиболее быстрый возврат к нормальной работе пользователей с БД. В то же время необходимо уберечь пользователей БД от любых проблем, связанных с возможными потерями и необходимостью дублирования работ по ведению данных.

Процесс восстановления зависит от типа отказа и размеров части базы, на которую отказ повлиял. Администратор БД должен предвидеть любой тип отказа и иметь соответствующую стратегию восстановления. При выборе стратегии восстановления необходимо исходить из альтернативы, связанной со степенью "восстановимости" и затратами на защиту БД. В общем случае, чем больше гарантия защиты данных, тем больше затраты на ее реализацию со стороны АБД.

Поскольку обработка данных в многопользовательских системах не может быть возобновлена точно с того места, где она была прервана, то существует возможность отойти назад до некоторой известной точки состояния БД и возобновить работу с БД с неё.

Наиболее простой стратегией является возможность восстановления базы данных из её копии, которая периодически делается администратором БД. Затем при возникновении сбоя база данных восстанавливается, и заново производятся все утерянные транзакции, например, осуществляется повторный ввод данных, не отраженных в ранее сделанной копии. Такая стратегия для баз данных с большим числом пользователей может и не позволить вернуть систему в то состояние, в котором она находилась до сбоя из—за того, что при параллельной работе трудно восстановить и синхронизовать все действия пользователей. Какой—то объем данных будет утрачен.

Существует ещё один подход к восстановлению БД. Он заключается в том, чтобы периодически делать копии базы данных и вести журнал всех изменений, произведенных в базе данных транзакциями со времени последнего копирования. Восстановление в случае сбоя может быть произведено одним из двух методов. Первый – база данных восстанавливается до известного (отраженного в копии) состояния, после чего выполняются все правильные транзакции согласно записям в журнале. Метод называют «откат вперед» (*rolling forward*). Второй метод – в случае сбоя отменяются все ошибочные

транзакции, затем запускаются правильные транзакции, которые выполнялись в момент сбоя – «откат назад», (*rolling back*). Для восстановления данных любым из этих методом требуется ведение журнала результатов транзакций. Современные СУБД имеют такие средства. Журнал транзакций позволяет сохранять действия, произведенных с данными в хронологическом порядке — прежде чем транзакция будет выполнена, она будет записана в журнал. В случае сбоя журнал используется как для отмены, так и для выполнения заданных транзакций. Журнал транзакций рекомендуется размещать на отдельном устройстве, что дает возможность повысить производительность и надежность системы баз данных:

— в случае потери работоспособности носителя данных сохраняется возможность создания резервной копии журнала транзакций;

— повышается скорость записи в журнал транзакций, так как операции ввода/вывода, разделенные между двумя устройствами, выполняются быстрее;

— рост объема журнала транзакций не понизит общую производительность системы;

— создание архивных копий журнала транзакций и данных происходит быстрее, поскольку они создаются отдельно друг от друга.

Размеры журнала транзакций варьируются в зависимости от объема модифицируемых данных и частоты создания архивных копий. Как правило, выделяют от 10 до 25% места, зарезервированного для данных.

Запись действий, производимых тем или иным пользователем БД, может вестись также администратором БД и другими способами, например, посредством использования различных триггеров.

### 5.3 Управление СУБД

К функциям, которые выполняет администратор БД, относятся также сбор и анализ статистики производительности работы СУБД, управление этой производительностью.

Производительность СУБД зависит от многих факторов – от используемой сервером БД операционной системы, от выделенных системных ресурсов, от параметров настройки ядра СУБД, от структуры запросов клиентских приложений к БД (например, производительность повышается при преимущественном использовании простых запросов, хранимых процедур, триггеров) и др.

Способ узнать, насколько эффективно функционирует сервер БД — это осуществление мониторинга состояния и производительности системы в целом с помощью специальных средств, входящих в состав СУБД. К показателям, определяющим производительность системы, относятся показатели статистики работы СУБД такие, как число логических чтений (чтение из кэш—памяти), число физических чтений (чтение с физического носителя), число разборов *sql* – выражений, общее число сортировок (на диске и в памяти), среднее число записей в сортировке, процент откатов в транзакциях, большое количество других показателей, которые соответствуют определенному виду обработки

данных. АБД должен следить за тем, чтобы количественные значения показателей собранной статистики отвечали требуемым, и при обнаружении каких—либо отклонений принимать соответствующие решения.

Важной задачей для АБД является также разбор жалоб пользователей на медленный отклик системы, длительное время выполнения тех или иных запросов. Выполняя запрос, любая современная серверная СУБД строит большое количество планов выполнения данного запроса на основе собранной статистики по таблицам и индексам, используемым в запросе. Для каждого плана СУБД вычисляет его «стоимость». Соответственно, для выполнения запроса выбирается план с наименьшей стоимостью. Если ситуация в БД сильно изменилась (добавился большой объем новых данных), то собранная ранее статистика является уже не актуальной для разбора выполняемого в настоящий момент запроса, запрос выполняется дольше, чем рассчитывал пользователь. Вновь собранная (на другом объеме данных) статистика ускоряет выполнение запроса. Решение о том, выполнять или не выполнять сбор статистики, какой должна быть частота периодичности этой процедуры, АБД должен принимать в зависимости от ситуации.

# 6 Вопросы проектирования приложений БД

## 6.1 Участие администратора БД в разработке приложения

К разработке приложения, работающего с базой данных, предъявляется ряд определенных требований. Первое и одно из главных требований – администратор БД должен участвовать в разработке приложения для того, чтобы правильно спроектировать и создать БД, которая будет поддерживать конечный продукт. Объединенная команда администраторов БД и разработчиков прикладных программ в процессе разработки должна придерживаться следующего:

— наделять администратора БД функциями управления разработкой приложения;

— создавать управляемые приложения – нужного размера и правильно организованные;

— реализовывать ограничения целостности БД максимально средствами СУБД;

— создавать необходимые индексы для максимального повышения производительности запросов;

— определять таблицы и индексы, которые чаще всего будут использоваться приложением;

— обнаруживать и исправлять недостатки в *SQL*—конструкциях, чтобы избежать их воздействия на производительность системы;

— четко определять права доступа пользователей и стремиться максимально реализовывать их средствами СУБД;

— обнаруживать и решать конфликты между распределением ресурсов работающих в оперативном режиме пользователей и пакетных процессов, требующих для своего выполнения значительного времени и другое.

Эффективное взаимодействие администратора БД с разработчиками приложений ведет к эффективному созданию, настройке и дальнейшему управлению БД, используемой различными приложениями.

## 6.2 Виды функций приложений БД

Функции разрабатываемого в рамках автоматизированной информационной системы приложения определяются на этапе анализа предметной области.

Основная функция приложения БД – обработка данных. Для баз данных определены четыре основные функции обработки: добавление и обновление данных, чтение и удаление.

Проектированию форм приложений, предназначенных для ввода и обновления данных, необходимо уделять особое внимание. При реализации на формах приложения этих функций необходимо следить за тем, чтобы приложение максимально поддерживало ограничения целостности, присущие рассматриваемой предметной области: правильно осуществлялось

формирование значений внешних ключей в дочерних таблицах (на основе соответствующих значений первичных ключей родительских таблиц, либо, если связь необязательная в предметной области, разрешать, но только в таком случае, возможность определения необязательного значения внешнего ключа); поддерживались выявленные ранее ограничения свойств классов объектов (например, посредством создания соответствующих масок ввода); формировались необходимые сообщения пользователю в ответ на реакцию СУБД, в случае нарушения ограничений целостности, реализованных на уровне базы данных. Например, если пользователь не ввел данные, предназначенные для поля таблицы, имеющего обязательную опциональность, разработчик приложения должен спроектировать текст соответствующего сообщения с конкретной подсказкой о невыполненных пользователем действий.

Важным моментом при проектировании форм приложений БД является реализация на формах понятной пользователю (и определенной предметной областью) последовательности (технологии) ввода данных. Пользователь должен иметь удобную возможность перемещения по цепочке вводимых данных, как вперед, так и назад. Это позволит осуществлять ему контроль своих действий при вводе и обновлении данных.

Как правило, функции добавления и обновления данных реализуются на одних и тех же интерфейсных формах. Но при этом должны быть реализованы и отличия (с учетом ограничений предметной области). Например, должно быть запрещено обновление тех данных, обновление которых запрещено в предметной области.

Форма приложения, предназначенная для ввода или обновления данных, должна быть спроектирована таким образом, чтобы пользователь мог сознательно реализовать функцию сохранения введенной или обновленной информации в базе данных. Разработчик приложения должен спроектировать диалог, в ходе которого пользователь должен подтвердить сохранение данных, имея при этом возможность ещё раз увидеть и, в случае необходимости, исправить все введенные или обновленные данные.

Операция удаления данных в БД недопустима, поскольку любое удаление наносит изъян представлениям предметной области. Если появилась необходимость удаления ошибочно введенных данных, то это должно послужить сигналом о том, что на уровне приложения неправильно организован контроль вводимых данных, либо неправильно реализована логика обработки данных. Возможно, удаление потребовалось вследствие того, что неправильно спроектирована БД или интерфейс ввода/ обновления данных, не поддерживаются на уровне БД её ограничения.

Функция удаления данных в БД должна быть доступной только администратору БД, который должен периодически анализировать и приводить данные БД в согласованное состояние, если их рассогласование достигнуто по вине некорректно спроектированных компонентов автоматизированной информационной системы. В рассогласованное состояние БД могут привести находящиеся в ней продублированные данные, некорректное состояние известных значений в определенных полях (например, некорректное

сокращение краткого наименования) и т.п. При этом АБД должен оказывать решительное влияние на принятие решения о модификации приложения, в котором неправильно реализованы функции по вводу и обновлению данных.

Функция чтения в БД реализуется, как правило, посредством *SQL*—запроса. АБД должен требовать, чтобы запросы проектировались с учетом их производительности: правильно использовались операции соединения (естественное, внешнее левое, внешнее правое); подзапросы, индексы и другое. При анализе плана выполнения запроса АБД может сделать соответствующие выводы и порекомендовать разработчику запроса изменить его структуру, спроектировать дополнительные индексы. Оптимальным является вариант совместного проектирования запросов администратором БД и прикладным программистом с учетом всех требований, накладываемых СУБД, приложением, вычислительной средой, предметной областью.

Важным для автоматизированной информационной системы является адекватное, с точки зрения конечного пользователя, представление данных на интерфейсных формах ввода/ вывода, в различных документах и отчетах, формируемых прикладной программой. Данные, взятые из БД, могут иметь много различных форматов – каждый конечный пользователь определяет свой формат. Востребованным являются форматы *MS Word*, *Excel*, форматы интернет—страниц.

В функции приложения БД входит и функция реализации контроля доступа к данным. В приложении, разрабатываемом для работы с БД, должна быть реализована процедура авторизация пользователя БД. Механизмы идентификации реализуют, как правило, средствами приложения, а аутентификации – средствами СУБД. Например, в приложении реализован интерфейс ввода имени и пароля пользователя, обработка сообщения об отказе в доступе. В механизме аутентификации используют, как правило, такие объекты БД, как представления и роли. В разрабатываемом приложении проектируется настройка и обработка результатов идентификации и запуск механизма аутентификации посредством использования специальных визуальных компонентов, реализации логики обработки соответствующих событий. Если разрабатывается одно, но многопользовательское приложение, механизм аутентификации может быть реализован частично через элементы интерфейса приложения. Проектируется разделение доступа пользователей (в зависимости от их привилегий) к соответствующим визуальным компонентам форм (меню, кнопки и другое) приложения, при этом разграничивается выполнение тех или иных функций приложения. Например, один пользователь может выполнять все функции приложения, другой – только доступные в заданном пункте меню.

Успешность внедрения и эксплуатации приложения, предназначенного для обработки данных в БД, зависит от реализации в нем определенных ранее требований предметной области. В противном случае приложение либо не будет востребовано, либо должно будет подвергнуться значительной модификации.

# Список использованных источников


1 **Дейт, К.** Введение в системы баз данных/ К. Дейт: пер. с англ. – 6—е изд. — СПб.: Издательский дом «Вильямс», 2000.— 848 с.

2 **Конноли, Т.** Базы данных: проектирование, реализация и сопровождение. Теория и практика: учебное пособие/ Т. Конноли, К. Бегг, А. Страчан: пер. с англ. — М.: Издательский дом «Вильямс», 2000. – 1120 с.

3 **Малыхина, М.П.** Базы данных: основы, проектирование, использование/ М.П. Малыхина. – СПб.: БХВ—Петербург, 2004. – 512 с.

4 **Петров, В.Н.** Информационные системы/ В.Н. Петров. – СПб.: Питер, 2002. – 688 с.

5 **Вендеров, А.М.** Проектирование программного обеспечения экономических информационных систем: учебник/ А.М. Вендеров. – М.: Финансы и статистика, 2000. – 352 с.

6 **Смирнова, Г.Н.** Проектирование экономических информационных систем: учебник/ Г.Н. Смирнова, А.А. Сорокин, Ю.Ф.Тельнов; под. ред. Ю.Ф. Тельнова. – М.: Финансы и статистика, 2001. – 512 с.

7 **Мюллер, Р.Дж.** Базы данных и UML. Проектирование/ Р.Дж. Мюллер: пер. с англ. – М.: Лори, 2002. – 419 с.

8 **Крёнке, Д.** Теория и практика построения баз данных/ Д. Крёнке. 8—е изд. – СПб.: Питер, 2003. – 800 с.

9 **Григорьев, Ю.А.** Банки данных: учебник/ Ю.А. Григорьев, Г.И. Ревунков. – М.: Изд—во МГТУ, 2002. – 320 с.

10 **Федоров, А.Г.** Базы данных для всех/ А.Г. Федоров, Н.З. Елманова. – М.: КомпьютерПресс, 2001. – 256 с.

11 **Верников, Г.** Основы методологии IDEF1X [Электронный ресурс] – Режим доступа: WWW.URL: http://www.citforum.ru/cfin/idef/idef1x.shtml.

12 **Кузнецов, С.Д.** Основы современных баз данных [Электронный ресурс] — Режим доступа: WWW.URL: http://www.citforum.ru/ database/osbd/contents.shtml

13 **Хоменко, Д.А.** Базы данных: учебник для высших учебных заведений/ Д.А. Хоменко, В.М. Цыганков, М.Г. Мальцев; под ред. проф. А.Д.Хоменко. – Спб.: КОРОНА принт, 2000. – 416 с.

14 **Карпова, Т.С.** Базы данных: модели разработка, реализация/ Т.С. Карпова. — СПб.: Питер, 2001. — 304 с.

15 **Голенищев, Э.П.** Информационное обеспечение систем управления/ Э.П. Голенищев, И.В. Клименко. — Ростов н/Д.: Феникс, 2003 – 352 с. (Серия "Учебники и учебные пособия").

16 **Ульман, Дж.Д.** Введение в системы баз данных/ Дж.Д. Ульман, Дж. Уидом: пер. с англ. – М.: Лори, 2000. – 374 с.

17 **Энсор, Д.** Oracle. Проектирование баз данных/ Д. Энсор, Й. Стивенсон: пер. с англ.: — К.: BHV, 1999. – 560 с.

18 **Стивенс, Р.** Программирование баз данных/ Р. Стивенс: пер. с англ.: — М.: Бином—Пресс, 2003. – 384 с.

19 **Горин, С.В.** Применение CASE—средства ERwin 2.0 для информационного моделирования в системах обработки данных/ С.В. Горин// СУБД. — 1995. — N 3. — С. 26—28.

20 **Луни, К.** Oracle 9i. Настольная книга администратора/ К. Луни, М. Терьо: пер. с англ.: М., Лори, 2004. – 745 с.

# Приложение А
*(обязательное)*

## Вопросы для самостоятельной работы

Вопросы к разделу 1

1. Дайте определение автоматизированной информационной системы.
2. Приведите примеры типов АИС.
3. Какие функции системы управления предприятия могут подлежать автоматизации?
4. Приведите пример функционального деления АИС.
5. Приведите примеры обеспечивающих подсистем АИС.
6. Назовите основные отличия использования файловых систем и баз данных в качестве информационного обеспечения АИС.
7. Перечислите функции СУБД.

Вопросы к разделу 2

1. Дайте краткое описание уровней архитектуры современных баз данных.
2. Опишите назначение внешнего уровня архитектуры БД.
3. Опишите назначение концептуального уровня архитектуры БД.
4. Поясните что такое логическая и физическая независимость данных.
5. Назовите модели данных, используемые при проектировании БД.
6. Перечислите основные этапы жизненного цикла БД.
7. Назовите этапы метода «нисходящего» проектирования БД.
8. Почему метод «восходящего» проектирования БД рекомендуется использовать для небольших БД?
9. Какими средствами может быть автоматизировано проектирование БД?

Вопросы к разделу 3

1. Назовите основные результаты, которые необходимо получить во время анализа предметной области.
2. Перечислите основные функции, которые должны быть реализованы в АИС.
3. Дайте определение класса объектов предметной области.
4. Перечислите характеристики, которые должны быть выявлены в ходе анализа предметной области для каждого свойства класса объектов.
5. Какие типы связей могут быть определены между классами объектов предметной области.
6. Для чего надо уметь правильно читать связи? Приведите пример.

7. Приведите примеры классов объектов, которые могут быть выявлены почти в каждой предметной области.

8. Приведите примеры семантических утверждений, ограничивающих предметную область.

9. Опишите функции всех категорий пользователей БД.

10. Перечислите основные методологии, используемые для построения информационно—логической модели (ИЛМ) предметной области на основе модели «объект—отношение», их различия.

11. В каких шаблонах моделирования, используемых для отображения фрагментов ИЛМ предметной области, используется рекурсивная связь?

12. Поясните что такое, с точки зрения предметной области, связь M:M. Какими способами она разрывается.

13. Приведите примеры моделирования ролей человека при построении ИЛМ предметной области.

14. Приведите примеры моделирования взаимоисключающих классов объектов.

15. Приведите примеры моделирования взаимоисключающих связей.

16. Что такое переносимость связей?

17. Приведите пример определения уникальности объекта из связи.

18. Как осуществляется проверка законченности *ER*—диаграммы.

19. для чего нужна перекрестная проверка выявленной иерархии функций и построенной модели данных?

20. Перечислите основные правила (компоненты) реляционной модели данных, которые необходимо знать для реализации этапа даталогического проектирования БД.

21. Перечислите виды документирования, которые могут быть использованы при построении даталогической модели (ДЛМ) БД.

22. Назовите основные действия, осуществляемые на этапе построения ДЛМ БД на основе *ER*—диаграммы предметной области.

23. Какие особенности предметной области необходимо учитывать при реализации в ДЛМ БД связей 1:1?

24. Расскажите об особенностях реализации в ДЛМ рекурсивных связей.

25. Перечислите способы реализации в ДЛМ взаимоисключающих классов объектов и взаимоисключающих связей.

26. Почему схема реляционной БД должна быть нормализованной?

27. Дайте определения 1НФ, 2НФ, 3НФ, НФБК.

28. Какие виды работ осуществляются на этапе формирования внутреннего уровня БД?

29. На основании каких критериев осуществляется выбор СУБД?

30. Приведите примеры современных СУБД, кратко их охарактеризуйте.

31. Назовите объекты БД, поддерживаемые современными СУБД.

32. Что должно быть отражено в техническом описании реляционной таблицы?

33. Для чего необходимы индексы? Особенности их проектирования?

34. Как средствами СУБД реализуются ограничения целостности реляционной БД?

Вопросы к разделу 4

1. Что подразумевается под созданием БД?
2. Какие требования предъявляются к аппаратному обеспечению, на котором будет находиться БД?
3. Назовите способы создания БД, используемые в разных СУБД.
4. Что такое метаданные?
5. Как в новой БД поддержать данные, накопленные в ранее действующих на предприятии информационных системах?

Вопросы к разделу 5

1. Назовите основные функции администратора БД.
2. Что такое идентификация и аутентификация пользователя?
3. Назовите и поясните структуру команд языка *SQL*, используемых для назначения прав доступа пользователям БД.
4. Что такое представление (просмотр) БД? Расскажите о механизме использования представлений БД для ограничения прав доступа пользователей БД.
5. Что такое транзакция? Назовите свойства транзакции.
6. Какие проблемы могут возникнуть при параллельной обработке данных несколькими пользователями?
7. Что такое блокировка данных? К каким проблемам может привести использование блокировок?
8. Какие уровни изоляции транзакций определены в стандарте языка *SQL*?
9. Какие способы восстановления БД существуют?

Вопросы к разделу 6

1. Какие вопросы должны решаться совместно с администратором БД при разработке приложения БД?
2. Какие основные виды функций присущи приложениям БД?

# Приложение Б
*(обязательное)*

## Тесты для контроля знаний

1. Укажите вариант верного утверждения: «Концептуальная инфологическая модель предметной области может быть отображена в виде … диаграммы»

а) *ER*;
б) функциональной;
в) произвольной.

2. Как называют метод проектирования БД, если он начинается с записи одной большой схемы реляционного отношения, отражающей по предположениям проектировщика предметную область, затем эта схема нормализуется

а) смешанный;
б) восходящий;
в) нисходящий?

3. Каким должен быть атрибут реляционного отношения, чтобы схема отношения соответствовала 1НФ

а) составным;
б) уникальным;
в) атомарным?

4. Какие зависимости между атрибутами отношения должны быть выявлены при приведении схемы этого отношения ко 2НФ

а) многозначные;
б) функциональные;
в) транзитивные?

5. Какие зависимости между атрибутами отношения должны быть выявлены при приведении схемы этого отношения к 3НФ

а) множественные;
б) транзитивные;
в) зависимости соединения?

6. Какие зависимости устраняются при приведении схемы реляционного отношения к 2НФ

а) транзитивные;
б) частичные функциональные;
в) множественные?

7. Какие зависимости устраняются при приведении схемы реляционного отношения к 3НФ?

а) транзитивные б) частичные функциональные в) множественные

8. На основе только чего можно выявить зависимости между атрибутами реляционного отношения

а) определении значений атрибутов, входящих в состав потенциальных ключей;

б) определении количества атрибутов отношения;

в) изучении семантики предметной области?

9. Для определений какой нормальной формы выявляются функциональные зависимости между атрибутами реляционного отношения

а) 1НФ;

б) 2НФ;

в) 3НФ?

10. Для определений какой нормальной формы выявляются транзитивные зависимости между атрибутами реляционного отношения

а) 1НФ;

б) 2НФ;

в) 3НФ?

11. Что такое детерминант реляционного отношения

а) правая часть функциональной зависимости;

б) левая часть функциональной зависимости;

в) левая часть транзитивной зависимости?

12. Какая нормальная форма требует того, чтобы все выявленные в схеме отношения детерминанты являлись потенциальными ключами этой схемы

а) 2НФ;

б) 3НФ;

в) НФБК?

13. Какая нормальная форма требует атомарности атрибутов и отсутствия повторяющихся групп в схеме реляционного отношения

а) НФБК;

б) 3НФ;

в) 1НФ?

14. Укажите вариант верного утверждения: «Основной компонент концептуальной инфологической (семантической) модели предметной области

а) свойство;

б) класс объектов;

в) арк».

15. Какой уровень архитектуры базы данных обеспечивает логическую и физическую независимость данных в БД

а) внешний;

б) концептуальный;

в) внутренний?

16. Какое слово пропущено в фразе: «Внешний уровень архитектуры БД это обобщенное … всех пользователей БД»

а) дело;

б) представление;

в) выступление?

17. Укажите вариант пропущенного слова в определении: «Часть реального мира, интересная с точки зрения решаемой задачи автоматизации и отображаемая в базе данных, называется … »

а) темой;

б) информационной системой;

в) предметной областью.

18. Укажите вариант верного утверждения: «На этапе информационного проектирования БД для термина "класс объектов" синонимом является термин

а) «экземпляр сущности »;

б) «сущность»;

в) «запись».

19. Укажите вариант верного утверждения: «На этапе информационного проектирования БД для термина "объект" синонимом является термин

а) «атрибут»;

б) «поле»;

в) «экземпляр сущности».

20. Укажите вариант верного утверждения: «На этапе информационного проектирования БД для термина "свойство" синонимом является термин

а) «атрибут»;

б) «запись»;

в) «экземпляр сущности».

21. Укажите вариант пропущенного слова в предложении: «Каждый класс объектов, выделенный в предметной области, должен иметь хотя бы один .... ключ»

а) уникальный;

б) внешний;

в) составной.

22. Укажите вариант верного утверждения: «На этапе информационного проектирования БД для термина "связь" синонимом является термин

а) «отношение»;

б) «произведение»;

в) «включение».

23. Укажите вариант правильного утверждения: «При выборе первичного ключа класса объектов из совокупности уникальных ключей следует отдавать предпочтение…

а) ключам, содержащим минимальное число целочисленных свойств;

б) ключам, содержащим минимальное число свойств любого типа;

в) ключам, содержащим большие текстовые значения».

24. Укажите вариант правильного утверждения:

а) «Первичный ключ должен иметь уникальное значение»;

б) «Первичный ключ может иметь неопределенное значение»;

в) «Первичный ключ может включать в себя только одно свойство».

25. Какому типу связи соответствует данное определение: «В каждый момент времени каждому объекту класса объектов КО1 может соответствовать один или ноль объектов класса объектов КО2, а каждому объекту класса объектов КО2, в свою очередь, может соответствовать ноль, один или несколько объектов класса объектов КО1".

а) М:1;

б) 1:1;

в) 1:М?

26. Какому типу связи соответствует данное определение: «В каждый момент времени каждому объекту класса объектов КО1 соответствует ноль, один или несколько объектов класса объектов КО2, но каждому объекту класса объектов КО2 соответствует ноль или только один объект класса объектов КО1.

а) 1:М;

б) 1:1;

в) М:М?

27. Какому типу связи соответствует данное определение: «В каждый момент времени отсутствуют все виды ограничений на связи в обоих направлениях между классами объектов КО1 и КО2».

а) 1:М;

б) 1:1;

в) М:М?

28. Какому типу связи соответствует данное определение: "В каждый момент времени каждому объекту класса объектов КО1 соответствует ноль или

только один объект класса объектов КО2 и каждому объекту класса объектов КО2 соответствует ноль или только один объект класса объектов КО1».

а) 1:М;

б) 1:1;

в) М:М?

29. Кто первый предложил использовать для представления модели предметной области *ER*—диаграмму

а) Ричард Баркер;

б) Питер Чен;

в) Джеймс Мартин?

30. Какой пользователь БД обязательно должен знать функциональные особенности СУБД

а) конечный пользователь;

б) администратор БД;

в) администратор данных?

31. Как называется характеристика свойства класса объектов, описывающая, что значение свойства обязательно должно быть определено

а) уникальность;

б) ограниченность;

в) опциональность?

32. Какую опциональность должна иметь рекурсивная связь

а) обязательную с обеих сторон;

б) обязательную с одной стороны и необязательную с другой;

в) необязательную с обеих сторон?

33. Как называется программное обеспечение, которое располагается непосредственно между физической БД и пользователями БД

а) *CASE*—средство;

б) СУБД;

в) операционная система?

34. Какой тип связи не поддерживают реляционные СУБД

а) 1:М;

б) 1:1;

в) М:М?

35. Может ли между двумя классами объектов в предметной области существовать более одной связи

а) Да;

б) Нет?

36. Укажите пропущенное слово в определении: «Множество допустимых значений, на котором может быть определен один или несколько атрибутов одного или нескольких реляционных отношений называется ....»

а) столбцом;

б) доменом;

в) диапазоном.

37. Чем определяются ограничения (тип, длина, опциональность, диапазон значений и т.п.) свойств классов объектов

а) выбранной СУБД;

б) предметной областью;

в) опытом проектировщика БД?

38. Как переводится название *Entity Relationship*

а) «сущность—связь»;

б) «функциональная диаграмма»;

в) «реляционное отношение»?

39. Назовите синоним термина «экземпляр сущности»

а) «объект»;

б) «класс объектов»;

в) «реляционное отношение»?

40. Назовите синоним термина “атрибут сущности"

а) «свойство класса объектов»;

б) «поле таблицы»;

в) «реляционное отношение»?

41. Можно ли в *ER*—диаграмме отразить домен

а) да;

б) нет?

42. С помощью какого графического элемента отображается связь в *ER* —диаграмме, созданной по методологии Питера Чена

а) только линии;

б) ромба и линии;

в) арка?

43. Что необходимо для осуществления перекрестной проверки

а) иерархия функций и соответствующая ей модель данных;

б) *ER*—диаграмма;

в) техническое задание?

44. В виде чего отображается на *ER*—диаграмме, построенной по методологии Ричарда Баркера, взаимоисключающие связи

а) в виде ромба;
б) в виде арка;
в) сплошными линиями?

45. В виде чего отображается на *ER*—диаграмме, построенной по методологии Ричарда Баркера, подтип
а) в виде прямоугольника с закругленными углами;
б) в виде арка;
в) сплошными линиями?

46. В виде чего отображается на *ER*—диаграмме, построенной по методологии Ричарда Баркера, супертип
а) в виде прямоугольника с закругленными углами;
б) в виде арка;
в) сплошными линиями?

47. Сколько подтипов минимально может существовать в супертипе
а) один;
б) два;
в) ограничений нет?

48. Как на *ER*—диаграмме, построенной по методологии Ричарда Баркера, отображается опциональность атрибута
а) крестиком;
б) стрелкой;
в) кружком и звездочкой?

49. Как на *ER*—диаграмме, построенной по методологии Ричарда Баркера, отображается опциональность связи
а) только пунктиром;
б) стрелкой;
в) сплошной линией и пунктиром?

50. Для чего в *ER*—диаграмму вводится класс объектов, который называют «сущность пересечения»
а) для отображения взаимоисключающих связей;
б) для отображения ролей;
в) для разрыва связи M:M?

51. Как на *ER*—диаграмме, построенной по методологии Ричарда Баркера, отображается мощность связи
а) пунктиром;
б) стрелкой;
в) «вороньей лапой»?

52. Где на *ER*—диаграмме рекомендуется располагать наиболее динамичные классы объекты

а) в левом верхнем углу;

б) посередине;

в) внизу?

53. Может ли подтип супертипа иметь свою собственную связь

а) да;

б) нет?

54. Может ли класс объектов иметь несколько рекурсивных связей

а) нет;

б) да?

55. Если уникальный идентификатор главного класса объектов является частью уникального идентификатора подчиненного класса объектов, то как это отмечается на *ER*—диаграмме, построенной по методологии Ричарда Баркера

а) вертикальной чертой на связи возле подчиненного класса объектов;

б) вертикальной чертой на связи возле главного класса объектов;

в) на середине линии связи?

56. С помощью чего на *ER*—диаграмме, построенной по методологии Ричарда Баркера, отображаются разные роли (функции) одного и того же класса объектов

а) отдельной линией связи;

б) это нельзя отобразить в методологии Ричарда Баркера;

в) все связи вместе отображаются одной линией?

57. Может ли связь входить в несколько арков

а) да;

б) нет?

58. Может ли в арк входить боле двух связей

а) да;

б) нет?

59. Связь какого типа называют «фотографией момента»

а) 1:M;

б) 1:1;

в) M:M?

60. Как помечается на *ER*—диаграмме, построенной по методологии Ричарда Баркера, непереносимость связи

а) ромбом на связи возле подчиненного класса объектов;

б) вертикальной чертой на связи возле главного класса объектов;

в) ромбом на середине линии связи?

61. Какого типа связи встречаются наиболее редко в предметной области
а) 1:M;
б) 1:1;
в) M:M?

62. Какого типа связи наиболее присущи любой предметной области
а) 1:M;
б) 1:1;
в) M:M?

63. Какие модели данных могут быть использованы для построения концептуальной информационно—логической модели предметной области
а) объектные;
б) модели на основе физических записей?

64. Могут ли быть вложенные подтипы
а) да;
б) нет?

65. В каком виде на *ER*—диаграмме в методологии Питера Чена отображается свойство
а) в виде ромба;
б) в виде эллипса;
в) в виде прямоугольника?

66. В каком виде на *ER*—диаграмме в методологии Питера Чена отображаются свойства классов объектов
а) в виде ромба;
б) в виде эллипса;
в) в виде прямоугольника?

67. Что отображает *ER*—диаграмма
а) даталогическую модель БД;
б) информационно—логическую модель предметной области;
в) физическую структуру БД?

68. Сколько ограничений накладывает на реляционную БД реляционная модель данных
а) одно;
б) два;
в) три?

69. К какой группе команд языка *SQL* относится команда *Create Table*

а) *DDL*;
б) *DML* ?

70. К какой группе команд языка *SQL* относится команда Select
а) *DDL*;
б) *DML*?

71. Атрибут, кортеж, отношение, база данных. Какая модель данных поддерживает такую структуру
а) сетевая;
б) реляционная;
в) иерархическая?

72. Укажите пропущенное слово в определении: «Элементы реляционного отношения называются ....
а) кортежами;
б) объектами;
в) доменами».

73. Что представляет собой совокупность связанных между собой схем реляционных отношений
а) содержимое базы данных;
б) модель предметной области;
в) схему базы данных?

74. Посредством чего реализуются в реляционной БД связи
а) внешних ключей;
б) функциональных зависимостей;
в) детерминантов?

75. В виде чего представляется класс объектов предметной области в реляционной базе данных
а) внешнего ключа;
б) реляционного отношения;
в) отдельных атрибутов реляционного отношения?

76. Укажите вариант верного утверждения: «Реляционное отношение обязательно должно иметь
а) внешний ключ;
б) первичный ключ;
в) транзитивную зависимость».

77. Что используют операторы реляционной алгебры в качестве операндов
а) реляционные отношения;

б) значения первичных ключей;
в) константы?

78. Что возвращают операторы реляционной алгебры в качестве результата
а) выражение;
б) значение заданной переменной;
в) реляционное отношение?

79. В какой нормальной форме находится схема отношения, если она находится в 1НФ и её первичный ключ не составной
а) 1НФ;
б) 2НФ?

80. К каким проблемам ведет ненормализованная структура БД
а) содержит не актуальные данные;
б) содержит избыточные данные;
в) содержит данные, которые нельзя использовать?

81. Чему равно в схеме реляционного отношения значение внешнего ключа, если оно определено
а) названию родительского отношения;
б) значению первичного ключа родительского отношения;
в) названию дочернего отношения?

82. Когда внешний ключ в схеме реляционного отношения, полученного на основе *ER*—диаграммы, может иметь неопределенное значение
а) если он отображает переносимую связь;
б) если он отображает необязательную связь;
в) если он входит в состав первичного ключа?

83. Чему равно в схеме реляционного отношения значение внешнего ключа, если оно определено и отображает рекурсивную связь
а) названию отношения;
б) значению первичного ключа дочернего отношения;
в) значению первичного ключа этого же отношения?

84. Как реализуются в реляционной БД “супертипы” и “подтипы”
а) только в виде одного отношения;
б) только в виде количества отношений, равного количеству подтипов;
в) возможны разные реализации – и первый, и второй варианты?

85. Укажите вариант верного утверждения: «Графическое представление схемы реляционной БД удобно для
а) нормализации схем отношений;

б) анализа опциональности внешних ключей;
в) формирования *SQL*—запросов».

86. Что не является объектом БД
а) таблица;
б) домен;
в) техническое задание?

87. Какую СУБД рекомендуют для создания только настольных, локальных БД
а) *Oracle*;
б) *Paradox*;
в) *MS SQL*?

88. Какую СУБД рекомендуют для создания серверной БД
а) *FoxPro*;
б) *Paradox*;
в) *MS SQL*?

89. Какие команды языка *SQL* относятся к командам определения данных
а) *Insert*;
б) *Select*;
в) *Alter*?

90. Какие команды языка *SQL* относятся к командам манипулирования данными
а) *Insert*;
б) *Drop*;
в) *Alter*?

91. Какие действия могут привести к нарушению ссылочной целостности
а) удаление строки из дочерней таблицы;
б) добавление строки в родительскую таблицу;
в) удаление строки из родительской таблицы?

92. Какой объект БД можно использовать для поддержки ссылочной целостности
а) таблицу;
б) индекс;
в) триггер?

93. Укажите вариант верного утверждения: «Индексы необходимы, если

а) часто выполняется операция сортировки по значениям столбца (столбцов) в наборе данных, возвращаемых в результате запроса;

б) значения столбцов часто меняют свои значения;

в) значения столбцов больших размеров».

94. Что содержит словарь данных

а) номера строк таблиц;

б) описание структуры объектов БД;

в) запись всех действий, выполняемых в БД за заданный период.

95. Укажите вариант верного утверждения: «Управление целостностью БД это

а) защита данных от несанкционированного (преднамеренного, незаконного) доступа;

б) защита данных в БД от неверных, непреднамеренных изменений и разрушений».

96. Укажите вариант верного утверждения: «СУБД управляет доступом пользователей, если

а) пароль пользователя «зашит» в прикладной программе;

б) в БД пользователь описан и описаны права его доступ;

в) администратор БД лично знает всех пользователей».

97. Выберите правильную последовательность действий при создании БД:

а) создание БД, создание доменов, создание дочерней таблицы, создание родительской таблицы таблиц;

б) создание БД, создание доменов, создание родительской таблицы, создание дочерней таблицы;

в) создание доменов, создание БД, создание дочерних таблиц, создание родительских таблиц;.

98. Укажите вариант верного утверждения: «Аутентификация – это:

а) установление личности пользователя на основании различных признаков;

б) процедура проверки прав пользователя на доступ к данным и прав выполнения определенных действий с этими данными;

в) процедура назначения прав пользователю».

99. Укажите вариант верного утверждения: «Блокировка данных может привести к:

а) «зависанию» процесса выполнения программы;

б) получению не корректного результата запроса;

в) искажению введенных в БД данных».

100. Укажите вариант верного утверждения: «БД входит в состав
а) программного обеспечения АИС;
б) правового обеспечения АИС;
в) информационного обеспечения АИС».

101. Укажите вариант верного утверждения: «Метод «восходящего» проектирования рекомендуется использовать для:
а) БД корпоративных АИС;
б) небольших, локальных БД;
в) БД, проектируемых с использованием CASE—средств».

## Приложение В
*(обязательное)*

### Ответы на тесты

1. а) *ER*.

2. б) восходящий.

3. в) атомарным.

4. б) функциональные.

5. б) транзитивные.

6. б) частичные функциональные.

7. а) транзитивные.

8. в) изучении семантики предметной области.

9. б) 2НФ.

10. в) 3НФ.

11. б) левая часть функциональной зависимости.

12. в) НФБК.

13. в) 1НФ.

14. б) класс объектов.

15. б) концептуальный.

16. б) представление.

17. в) предметной областью.

18. б) «сущность».

19. в) «экземпляр сущности».

20. а) «атрибут».

21. а) уникальный.

22. а) «отношение».

23. а) ключам, содержащим минимальное число целочисленных свойств.

24. б) «Первичный ключ должен иметь уникальное значение»;

25. а) M:1.

26. а) 1:M.

27. в) M:M.

28. б) 1:1.

29. б) Питер Чен.

30. б) администратор БД.

31. в) опциональность.

32. в) необязательную с обеих сторон.

33. б) СУБД.

34. в) M:M.

35. а) Да.

36. б) доменом.

37.б) предметной областью.

38. а) «сущность—связь».

39. а) «объект».

40. а) «свойство класса объектов».

41. б) нет.

42. б) ромба и линии.

43. а) иерархия функций и соответствующая ей модель данных.

44. б) в виде арка.

45. а) в виде прямоугольника с закругленными углами.

46. а) в виде прямоугольника с закругленными углами.

47. в) ограничений нет.

48. в) кружком и звездочкой.

49. в) сплошной линией и пунктиром.

50. в) для разрыва связи M:M.

51. б) «вороньей лапой».

52. а) в левом верхнем углу.

53. а) да.

54. б) да.

55. а) вертикальной чертой на связи возле подчиненного класса объектов.

56. а) отдельной линией связи.

57. б) нет.

58. а) да.

59. б) 1:1.

60. а) ромбом на связи возле подчиненного класса объектов.

61. б) 1:1.

62. а) 1:M.

63. а) объектные.

64. а) да.

65. б) в виде эллипса.

66. в) в виде прямоугольника.

67. б) информационно—логическую модель предметной области.

68. б) два.

69. а) *DDL*.

70. б) *DML*.

71. б) реляционная.

72. а) кортежами.

73. в) схему базы данных.

74. а) внешних ключей.

75. б) реляционного отношения.

76. б) первичный ключ.

77. а) реляционные отношения.

78. в) реляционное отношение.

79. б) 2НФ.

80. б) содержит избыточные данные.

81. б) значению первичного ключа родительского отношения.

82. б) если он отображает необязательную связь.

83. в) значению первичного ключа этого же отношения.

84. в) возможны разные реализации – и первый, и второй варианты.

85. в) формирования *SQL*—запросов».

86. в) техническое задание.

87. б) *Paradox*.

88. в) *MS SQL*.

89. в) *Alter*.

90. а) *Insert.*

91. в) удаление строки из родительской таблицы.

92. в) триггер.

93. а) часто выполняется операция сортировки по значениям столбца (столбцов) в наборе данных, возвращаемых в результате запроса.

94. б) описание структуры объектов БД.

95. б) защита данных в БД от неверных, непреднамеренных изменений и разрушений».

96. б) в БД пользователь описан и описаны права его доступ.

97. б) создание БД, создание доменов, создание родительской таблицы, создание дочерней таблицы.

98. б) процедура проверки прав пользователя на доступ к данным и прав выполнения определенных действий с этими данными.

99. а) «зависанию» процесса выполнения программы.

100. в) информационного обеспечения АИС.

101. б) небольших, локальных БД.