МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА РОССИЙСКОЙ ФЕДЕРАЦИИ
ДЕПАРТАМЕНТ НАУЧНО-ТЕХНОЛОГИЧЕСКОЙ ПОЛИТИКИ
И ОБРАЗОВАНИЯ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ УНИВЕРСИТЕТ»

Моделирование данных с помощью Data Modeler за 7 дней

Красноярск
2020

Моделирование данных с помощью Data Modeler за 7 дней
[Электронный ресурс]**:** методические указания к выполнению лабораторных работ / сост.: Миндалёв И.В. — Красноярск, Краснояр. гос. аграр. ун-т., 2020, 108 с.

Методические указания представляет собой руководство по моделированию данных с помощью CASE-средства Oracle Developer Data Modeler.

Указания содержат описание методов анализа и проектирования моделей данных в объеме, необходимом для практической работы. На конкретном примере рассмотрено применение CASE-средства для автоматизации этапов анализа и проектирования реляционной базы данных.

Предназначено для студентов, изучающих дисциплины «Базы данных», «Теория экономических информационных систем», «Информационные системы в управлении», «Разработка программных приложений», «Практика по получению профессиональных умений и опыта профессиональной деятельности», «Преддипломная практика» по направлению 09.03.03 «Прикладная информатика», «Базы данных», «Интернет программирование» по направлению 01.03.02 «Прикладная математика и информатика».

Рецензент: С.И. Сенашев, д.ф.-м.н., профессор, зав. кафедрой «Информационные экономические системы» СибГАУ.

Печатается по решению редакционно-издательского совета
Красноярского государственного аграрного университета

Введение

Решения о внедрении информационных технологий должны основываться на концепциях и подходах, позволяющих предприятиям в оптимальном объеме овладеть информационными аспектами. Отсюда неуклонный рост требований к профессиональной подготовке будущих специалистов по ИТ, менеджеров и экономистов, к обновлению средств и методов обучения студентов.

Создание современных информационных систем представляет собой сложную задачу, решение которой требует применения специальных инструментов. К ним относятся CASE-средства, которые позволяют систематизировать и автоматизировать все этапы разработки программного обеспечения.

Моделирование данных – это средство формального сбора данных, относящихся к бизнес-процессу конкретного предприятия. Моделирование является одним из главных на сегодняшний день приемов анализа, основанием, на котором строятся реляционные базы данных.

В настоящем пособии рассматривается технология моделирования с помощью CASE-средства – Oracle Developer Data Modeler.

Пакет Oracle SQL Developer Data Modeler представляет собой универсальный, полностью автономный инструмент с поддержкой логического, реляционного, многомерного моделирования и моделирования типов данных. Возможность конструирования моделей данных на разных уровнях позволяет сформировать исчерпывающие концептуальные блок-схемы связей между сущностями ERD (Entity Relationship Diagram) и превратить их в рабочие реляционные модели данных.

Разделение реляционной и физической модели позволяет разработчикам на базе единственной реляционной модели создавать разные физические модели данных для разных целевых серверов СУБД, включая Oracle Database, IBM DB2 V7 и V8 для платформ Linux, UNIX и Windows, мэйнфреймы OS/390, а также Microsoft SQL Server 2000 и 2005.

Пакет Oracle SQL Developer Data Modeler поддерживает весь диапазон физических описаний для баз данных, в том числе логические разделы, роли и табличные пространства, с учетом конкретной версии БД в системах с разными СУБД от разных производителей. Кроме всего прочего, новый пакет полностью интегрируется с популяр-

ной визуальной средой разработки баз данных Oracle SQL Developer, поэтому разработчики могут быстро открыть и просмотреть ранее созданные структуры БД, а также могут выполнить запросы, запустить формирование отчетов из репозитария отчетов и т. д.

Пакет Oracle SQL Developer Data Modeler поддерживает работу со всеми редакциями СУБД Oracle Database 11g в операционных системах Windows, Linux и Mac OS X [14].

Указания включают следующие разделы:

1-й день – посвящен основным понятиям разработки информационных систем.

2-й день – вводит в мир реляционных баз данных.

3-й день – знакомство с технологией разработки информационных систем, предлагаемой компанией Oracle, в том числе даны пошаговые инструкции по установке Oracle SQL Developer Data Modeler, Oracle SQL Developer, Oracle 10g XE.

4-й и 5-й дни – посвящены практической работе по созданию логической модели данных книгоиздательской компании с помощью Oracle SQL Developer.

6-й день – посвящены практической работе по созданию реляционной модели данных книгоиздательской компании с помощью Oracle SQL Developer Data Modeler.

7-й день – посвящены практической работе по созданию физической модели данных книгоиздательской компании с помощью Oracle SQL Developer Data Modeler, Oracle SQL Developer, Oracle 10g XE.

1-й день. Разработка информационных систем

Ель растет перед дворцом, А
под ней хрустальный дом;
Белка там живет ручная,
Да затейница какая!
Белка песенки поет
Да орешки все грызет,
А орешки не простые,
Все скорлупки золотые,
Ядра – чистый изумруд...
А. С. Пушкин. Сказка о царе Салтане

1.1. Абстракции

Разработка информационной системы обязательно включает создание различных моделей. Что такое модель? В широком смысле, модель – есть результат корректного воспроизведения каким-либо способом или средствами различных объектов (в том числе процессов и явлений реального мира или мыслительной деятельности человека). Модели являются, с одной стороны, продуктом изучения свойств соответствующих объектов, с другой, служат инструментом для углубления знаний о них, а также решений прикладных задач.

Модель теснейшим образом связана с абстракцией. Абстракция – означает буквально отвлечение, то есть исключение из рассмотрения чего-то. И всякая модель – это абстракция, то есть отвлечение от тех явлений реального или идеального мира, которые нас в данный момент не интересуют.

Но среди разработчиков информационных систем нередко практикуется «технологически конструктивистский» подход к своей деятельности. Люди увлекаются чисто техническими решениями типа «как получить такой-то эффект в новой версии», «как соединить одно с другим» и т. п. Разработка информационной системы при таком подходе часто видится как подбор комбинации и организация взаимодействия технологических решений, предоставляющих возможность выполнения требуемой задачи. Однако, нужно помнить, что, кроме технической конкретики, не менее важны абстракции [1].

В пользу научных абстракций можно вспомнить, что информатика (computer science) объединяет в себе три составляющие: науку, техно-

логию и практику. Поэтому при работе в компьютерной области отрешение хотя бы одной из этих составляющую нежелательно. В базах данных это утверждение отражается в двойственной природе СУБД: с одной стороны СУБД – сложно устроенный и насыщенный технологическими решениями инженерный продукт, а с другой стороны – инструмент моделирования прикладной области. Без качеств СУБД именно как инструмента моделирования о них можно забыть.

Перечислим некоторые виды абстракций, которые в той или иной степени, явно или неявно, но обязательно присутствуют в разработке конкретной информационной системы: математическая модель, реляционная модель, модель данных SQL, модель сущность-связь, объектная модель.

Понимание абстракций – есть предпосылка грамотной разработки информационных систем.

1.2. Свойства и компоненты информационных систем

Мы определили модель как один из инструментов, используемый в создании информационных систем. А что такое информационная система?

Начнем с истории. Как известно, возникновение компьютеров главным образом стимулировалось необходимостью проведения расчетов для создания ядерного оружия и ракетной техники. Объемы требуемых вычислений просто не позволяли произвести их в приемлемое время традиционным коллективом расчетчиков.

Однако сразу же на появление компьютеров обратили внимание бизнесмены. Как правило, в гражданском бизнесе не требуются большие расчеты. Основной проблемой в нем являются объемы информации, которые необходимо собирать, надежно хранить и оперативно обрабатывать. Появление информационных систем,,основным назначением которых было решение этой проблемы, было ответом компьютерной индустрии на требование мира бизнеса [6].

Информационной системой (ИС) называется программно-аппаратный комплекс, функционирование которого состоит в надежном хранении информации в в памяти компьютера, выполнении специфических для конкретной предметной области преобразований информации и вычислений, предоставлении удобного и легко осваемого интерфейса.

Области применения информационных приложений разнообразны: страхование, транспорт, образование и т. д. Трудно найти область деловой активности, в которой сегодня можно было бы обойтись без использования информационных систем. И конечно, в зависимости от конкретной области применения информационные системы очень сильно различаются по своим функциям, архитектуре, реализации.

Но можно выделить два свойства, которые являются общими для всех информационных систем:

Любая ИС предназначена для сбора, хранения и обработки информации. Поэтому в основе любой ИС лежит среда хранения и доступа к данным. Среда – совокупность ресурсов, предоставляемых в распоряжение пользователя системы. Среда должна обеспечить уровень надежности хранения и эффективность доступа, соответствующие области применения ИС.

ИС ориентируются на конечного пользователя, например, бухгалтера. Такие пользователи могут быть очень далеки от мира компьютеров. Для них персональный компьютер – всего лишь орудие собственной профессиональной деятельности. Поэтому ИС обязана обладать простым, удобным, легко усваиваемым интерфейсом, который должен предоставить конечному пользователю все необходимые для его работы функции, но в то же время не дать ему возможность выполнять какие-то лишние действия. Обычно этот интерфейс является графическим: с меню, кнопками, подсказками и т. п.

Для функционирования ИС необходимы следующие основные компоненты:

база данных;
схема базы данных;
система управления базой данных;
приложения;
пользователи,
технические средства.

Рассмотрим кратко каждый из этих компонентов. Начнем с базы данных. Существует немало ее определений. Вот нестрогое определение БД, которое Дейт (C. J. Date), один из известных экспертов в области баз данных, дает в начале своего учебного курса: «Базу данных можно рассматривать как подобие электронной картотеки, то есть

хранилище для некоторого набора занесенных в компьютер файлов данных» [5].

Тогда получается, что база данных – это просто колоссальный набор данных? Да, многие люди так и думают. Но файл может содержать довольно большое количество данных и не быть базой данных. Важным свойством БД является то, что база данных может себя описать. Можно сказать, что БД обязательно содержит – данные и метаданные. Данные – это данные пользователя или предприятия, использующего систему, и связанные с его деятельностью. Например, данные о продукции, счетах, коровах. Метаданные – это данные о данных или схема базы данных,,которая описывает структуру обычных данных и дает о них фундаментальную информацию. Обычно мы не видим эту схему, потому что она спрятана от нас программными средствами.

Пользователей можно разделить на три большие группы: прикладные программисты, пользователи, администраторы.

Прикладные программисты – отвечают за написание бизнес-приложений, использующих базу данных (например, приложение по автоматизации маркетинга). Приложения выполняют над данными стандартные операции: выборку существующей информации, вставку новой информации, удаление или обновление существующей информации. Все эти функции выполняются через соответствующий запрос к СУБД.

Конечные пользователи (например, менеджер) – работают с информационной системой непосредственно через рабочую станцию или терминал. Пользователь получает доступ к БД, используя одно из приложений.

В связи с тем, что данные – одна из главных ценностей предприятия, администратор данных должен разбираться в данных и понимать нужды предприятия по отношению к данным на уровне управления высшего руководства предприятия. В его обязанности входят: принимать решения, какие данные необходимо вносить в БД, обеспечивать поддержание порядка при использовании их после занесения в базу данных.

Техническим специалистом, ответственным за реализацию решений администратора данных, является администратор БД. Его работа

заключается в создании самой БД и техническом контроле , необходимом для осуществления решений администратора данных.

Между собственно БД (т. е. данными) и пользователями располагается уровень программного обеспечения – система управления базой данных. Все запросы пользователей на доступ к БД обрабатываются СУБД.

СУБД важный, но не единственный компонент программного обеспечения ИС. Среди других – упомянутые выше бизнес-приложения, утилиты, CASE-средства, генераторы отчетов и форм и т.д.

Технические средства информационных систем могут включать:

средства вычислительной техники (серверное оборудование, рабочие станции, принтеры и т.д.);

локальные вычислительные сети;

копировально-множительную аппаратуру;

средства связи (учрежденческие АТС, каналы связи и канальное оборудование, телефоны, факсимильные аппараты, мобильные средства связи).

1.3. Жизненный цикл

Одним из базовых понятий методологии разработки ИС является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО).

Жизненный цикл программного обеспечения – это непрерывный процесс, который начинается с момента принятия решения о необходимости его создания и заканчивается в момент его полного изъятия из эксплуатации.

Основным нормативным документом, регламентирующим ЖЦ ПО, является международный стандарт ISO/IEC 12207. Стандарт определяет структуру ЖЦ, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания ПО.

Структура ЖЦ ПО по стандарту ISO/IEC 12207 базируется на трех группах процессов:

основные процессы: приобретение, поставка, разработка, эксплуатация, сопровождение;

вспомогательные процессы, обеспечивающие выполнение основных процессов: документирование, управление конфигурацией, обеспечение качества, верификация, аттестация, оценка, аудит, решение проблем;

организационные процессы: управление проектами, создание инфраструктуры проекта, определение, оценка и улучшение самого ЖЦ, обучение.

Рассмотрим кратко некоторые из этих процессов [2].

Разработка – включает в себя все работы по созданию ПО в соответствии с заданными требованиями, включая оформление проектной и эксплуатационной документации, подготовку материалов, необходимых для проверки работоспособности и соответствующего качества программных продуктов, материалов, необходимых для организации обучения персонала.

Разработка ПО включает в себя, как правило:

анализ;

проектирование;

реализацию (программирование).

Эксплуатация – включает, в себя работы по внедрению компонентов ПО в эксплуатацию, в том числе конфигурирование базы данных и рабочих мест пользователей, обеспечение эксплуатационной документацией, проведение обучения персонала и непосредственно эксплуатацию, в том числе локализация проблем и устранение причин их возникновения, модификацию ПО в рамках установленного регламента,подготовку предложений по совершенствованию, развитию и модернизации системы.

Обеспечение качества проекта – связано с проблемами верификации, проверки и тестирования ПО.

Верификация – это процесс определения того, отвечает ли текущее состояние разработки, достигнутое на данном этапе, требованиям этого этапа. Проверка позволяет оценить соответствие параметров разработки исходным требованиям. Проверка частично совпадает с тестированием, которое связано с определением различий между действительными и ожидаемыми результатами и оценкой соответствия характеристик ПО исходным требованиям.

Управление проектом – связано с вопросами планирования и организации работ, создания коллективов разработчиков и контроля за сроками и качеством выполняемых работ.

Техническое и организационное обеспечение проекта – включает выбор методов и инструментальных средств для реализации проекта,

определение методов описания промежуточных состояний разработки, разработку методов и средств испытаний ПО, обучение персонала.

Каждый процесс характеризуется определенными задачами и методами их решения, исходными данными, полученными на предыдущем этапе, и результатами. ЖЦ ПО носит итерационный,характер: результаты очередного этапа часто вызывают изменения е проектных решениях, выработанных на более ранних этапах.

1.4. Анализ

Анализ является составной частью разработки информационной системы. Анализ – это подробное исследование бизнес-процессов (функций) и информации, необходимой для выполнения этих функций (сущностей с их атрибутами и связями). Моделирование является основным приемом анализа. Поэтому методы моделирования считаются частью анализа [3].

Аналитики собирают и фиксируют информацию в двух разных, но взаимосвязанных формах. Это:

функции – информация о событиях и процессах, которые происходят в бизнесе;

сущности – информация о «вещах, имеющих значение для предприятия, о которых что-то известно».

Вот два вида классических результатов анализа:

иерархия функций, которая разбивает процесс обработки на функции;

модель сущность-связь, охватывающая все сущности, их атрибуты и связи между ними.

1.5. Проектирование

Проектирование выполняется на основе данных анализа и охватывает три основные области:

Проектирование схемы базы данных – на основании модели, разработанной на этапе анализа. Схема содержит описание конкретных таблиц, представлений, индексов, ограничений, триггеров, которые будут реализованы в базе данных.

Проектирование конкретных экранных форм, отчетов и программ, которые будут сопровождать данные в БД и обеспечивать выполнение запросов к этим данным.

При определенных обстоятельствах в процессе проектирования необходимо учитывать конкретную среду или технологию – топологию сети, конфигурацию аппаратных средств, использование архитектуры клиент-сервер или параллельной обработки, или распределенной архитектуры БД.

Можно сказать, что проектирование – это поиск удовлетворения функциональных требований средствами имеющейся технологии с учетом заданных ограничений [8]. Какие это ограничения? Это может быть максимальное время, отпущенное на проект или максимальное количество денег, которое может быть на него потрачено.

1.6. Реализация

По завершении исходного проектирования следует этап реализации, на котором создаются и тестируются программные модули, определенные при проектировании. Главными результатами этого этапа являются модули исходного кода и объектные модули. После реализации переходят к тестированию системы, а затем к сдаче ее в эксплуатацию [8].

Важно понимать, что вплоть до запуска ИС в эксплуатацию вам придется постоянно возвращаться к этапам проектирования и анализа.

1.7. Зачем нужны CASE-средства

Тенденции развития современных информационных технологий приводят к постоянному возрастанию сложности информационных систем, создаваемых в различных областях бизнеса.

Для успешной реализации проекта объект разработки должен быть прежде всего адекватно описан, должны быть построены полные и непротиворечивые модели ИС. Накопленный к настоящему времени опыт разработки ИС показывает, что это логически сложная, трудоемкая и длительная по времени работа, требующая высокой квалификации участвующих в ней специалистов. Раньше разработка ИС выполнялась в основном на интуитивном уровне с применением ненормализованных методов, основанных на искусстве, практическом опы-

те, экспертных оценках и дорогостоящих экспериментальных проверках качества функционирования ИС. Кроме того, в процессе создания и функционирования ИС информационные потребности пользователей могут изменяться или уточнятся, что еще более усложняет разработку и сопровождение таких систем.

В 70- и 80-х годах XX в. при разработке ИС достаточно широко применялась структурная методология, предоставляющая в распоряжение разработчиков строгие формализованные методы описания ИС и принимаемых технических решений. Она основана на наглядной графической технике: для описания различного рода моделей ИС используются схемы и диаграммы. Наглядность и строгость средств структурного анализа позволяла разработчикам и будущим пользователям (менеджеры, бухгалтеры) с самого начала ненормально участвовать в ее создании, обсуждать и закреплять понимание основных технических решений.

Однако широкое применение этой методологии и следование ее рекомендациям при разработке конкретных ИС встречались достаточно редко, поскольку при ручной разработке это практически невозможно. Действительно, вручную очень трудно разработать и графически представить строгие формальные спецификации системы, проверить их на полноту и непротиворечивость и тем более изменить. Если все же удается создать строгую систему проектных документов, то ее переработка при появлении серьезных изменений практически неосуществима.

Ручная разработка обычно порождала следующие проблемы:

неадекватная спецификация требований;

неспособность обнаруживать ошибки в проектных решениях;

низкое качество документации, снижающее эксплуатационные качества;

затяжной цикл и неудовлетворительные результаты тестирования [2].

Перечисленные факторы способствовали появлению программно-технологических средств специального класса – CASE-средств, реализующих САБЕ-технологию создания и сопровождения информационных систем.

Термин CASE (computer aided software engineering) используется в настоящее время в весьма широком смысле. Первоначальное значение термина САБЕ, ограниченное вопросами автоматизации разработки только лишь программного обеспечения, в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом. Теперь под термином САБЕ-средства понимаются программные средства, поддерживающие процессы создания и сопровождения ИС, включая анализ и формулировку требований, проектирование прикладного ПО (приложений) и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом, а также другие процессы.

CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

1.8. Классификация САЗЕ-средств

Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования ИС: от простых средств анализа и документирования до полномасштабных средств автоматизации, покрывающих весь жизненный цикл ПО.

Наиболее трудоемкими этапами разработки ИС являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации. При этом большую роль играют методы визуального представления информации. Это предполагает построение структурных или иных диаграмм в реальном масштабе времени, использование многообразной цветовой палитры, сквозную проверку синтаксических правил. Графические средства моделирования предметной области позволяют разработчикам в наглядном виде изучать существующую ИС, перестраивать ее в соответствии с поставленными целями и имеющимися ограничениями.

В разряд CASE-средств попадают как относительно дешевые системы для персональных компьютеров с весьма ограниченными возможностями, так и дорогостоящие системы для неоднородных вычислительных платформ и операционных сред.

Обычно к CASE-средствам относят любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного

цикла ПО и обладающее следующими основными характерными особенностями:

мощные графические средства для описания и документирования ИС, обеспечивающие удобный интерфейс с разработчиком и развивающие его творческие возможности;

интеграция отдельных компонент CASE-средств, обеспечивающая управляемость процессом разработки ИС;

использование специальным образом организованного хранилища проектных метаданных (репозитория).

Классификация CASE-средств по типам отражает их функциональную ориентацию на те или иные процессы ЖЦ и включает:

средства анализа, предназначенные для построения и анализа моделей предметной области (Bpwin, Ramus, ArgoUML, Oracle Developer Data Modeler, Dia);

средства анализа и проектирования, поддерживающие наиболее распространенные методологии проектирования и использующиеся для создания проектных спецификаций (Oracle Designer, ARIS, IBM Rational). Выходом таких средств являются спецификации компонентов и интерфейсов системы, архитектуры системы, алгоритмов и структур данных;

средства проектирования баз данных, обеспечивающие моделирование данных и генерацию схем баз данных (как правило, на языке SQL) для наиболее распространенных СУБД. К ним относятся ERwin, MySQL Workbench, Oracle Developer Data Modeler.

средства разработки приложений. К ним относятся средства JDeveloper, Delphi, Lazarus, Eclipse, NetBeans, Microsoft Visual Studio, Oracle Application Express.

средства реинжиниринга, обеспечивающие анализ программных кодов и схем баз данных и формирование на их основе различных моделей и проектных спецификаций.

2-й день. Реляционная модель данных

Мой дядя самых честных правил,
Когда не в шутку занемог,
Он уважать себя заставил
И лучше выдумать не мог.
А. С. Пушкин. Евгений Онегин, гл. 1,1

2.1. Правила Кодда

Создание моделей в среде Data Modeler базируется на теории реляционных баз данных, База данных – это совокупность данных, организованных с определенной целью.

В этом определении слово «организованных» означает, что указанная совокупность включает данные, которые сохраняются, имеют определенный формат (отформатированы), к ним может быть обеспечен доступ (доступны), они могут быть представлены потребителю информации в приемлемом виде [9].

Упоминанием в определении слова «цели» подчеркивается, что состав данных должен соответствовать некой задаче: с одной стороны, в базе не должны содержаться данные, не имеющие отношения к задаче, а с другой – в ней должна содержаться вся информация, касающаяся задачи.

Хорошим примером БД может служить телефонный справочник. Он содержит данные, имеющие прямое отношение к его назначению – имена абонентов, которые открывают доступ к номеру телефона. В справочнике отсутствует избыточная информация (скажем, цвет аппарата абонента), т. е. в этой простейшей базе содержатся сведения, непосредственно связанные с целью ее создания.

Приведем определение СУБД (система управления базой данных). Это есть программный комплекс, обеспечивающий функционирование базы данных. СУБД играет роль кладовщика при данных, отвечает за их сохранность, безопасность, целостность, взаимное соответствие и обеспечивает доступ пользователей к данным.

Реляционный подход к управлению базами данных основан на математической модели, использующей методы реляционной алгебры и реляционного исчисления.

Наличие солидной теоретической базы является одним из преимуществ реляционного подхода. Вот как об этом пишет Дейт: «Реляци-

онная фактически означает различная. Она является различной, поскольку не является специальной. Старые же системы, напротив, имели специальное назначение; они предоставляли решение для определенных задач того времени, но у них не было твердой теоретической базы. А у реляционных систем такая база есть..., а это означает, что [они] надежны как скала... Благодаря этому твердому основанию, поведение реляционных систем отличается предсказуемостью; пользователь (возможно, не осознавая этого) держит в голове простую модель этого поведения, и это позволяет ему предвидеть, что сделает система в той или иной ситуации. Сюрпризов быть не может (или не должно быть). Предопределенность означает, что пользовательский интерфейс прост для понимания, обучения, изучения, использования и запоминания» [5].

Тем не менее большинство действительно необходимых нам определений из области управления базами данных скорее относятся к практической, чем к теоретической стороне этого вопроса.

Кодд (E. F. Codd), автор реляционной модели, разработал список критериев, которым должна удовлетворять реляционная модель. Описание этого списка, называемого правилами Кодда, требует введения сложной терминологии и теоретических выкладок. Мы опишем эти правила в несколько упрощенном виде.

Итак, чтобы считаться соответствующей реляционной модели, СУБД должна [10]:

представлять всю информацию в виде таблиц;

поддерживать логическую структуру данных, независимо от их физического представления;

использовать язык высокого уровня для структурирования, выполнения запросов и изменения информации в БД (обычно для этого используется язык SQL);

поддерживать основные реляционные операции (выбор, проектирование и соединение), а также теоретико-множественные операции, такие, как объединение, пересечение, вычитание и декартово произведение;

поддерживать виртуальные таблицы (представления), обеспечивая пользователям альтернативный способ просмотра данных в таблицах;

различать в таблицах неизвестные значения (null), нулевые значения и пропуски в данных;

обеспечивать механизмы для поддержки целостности, авторизации,транзакции и восстановления данных.

Рассмотрим эти правила, являющиеся основными законами реляционного мира.

2.2. Одни таблицы

Первое правило Кодда гласит, что вся информация в реляционных БД представляется значениями в таблицах. Таблица (см. табл. 1) состоит из горизонтальных строк и вертикальных столбцов. Все данные представляются в табличном формате – другого способа просмотреть информацию в базе данных не существует.

Фамилия	**Адрес**
Аринчин	Мира 130, кв. 24
Деменюк	Академгородок 10, кв. 10
Корякина	Академгородок 20, кв. 46

Таблица 1: Таблица реляционной БД

Набор связанных таблиц образует базу данных. Таблицы в реляционной базе данных разделены, но полностью равноправны. Между ними не существует никакой иерархии.

Каждая таблица состоит из строк и столбцов. Каждая строка описывает отдельную сущность (entity) – автора книг, компанию, торговую сделку и что-нибудь другое. Каждый столбец описывает одну характеристику сущности – фамилию автора, адрес автора и т. п.

Основным условием, накладываемым на строки в реляционной модели, является их уникальность. То есть в таблице не может быть двух одинаковых строк. В каждой таблице должен быть столбец, значения которого будут во всех строках разными. Уникальный признак-столбец называют первичным ключом (primary key).

Каждый элемент данных, или значение (value), определяется пересечением строки и столбца. Чтобы найти требуемый элемент данных, необходимо знать имя содержащей его таблицы, столбец и значение его первичного ключа.

Предположим, вы хотите узнать адрес Деменюка. Чтобы добраться до этой информации, вы приказываете системе извлечь этот адрес из

таблицы под названием Автор из столбца Адрес. При этом имя Деменюк является значением первичного ключа, идентифицирующим эту строку.

В реляционных базах данных существует два типа таблиц – пользовательские таблицы и системные таблицы.

Пользовательские таблицы содержат информацию, для поддержки которой собственно и создаются реляционные базы данных – данные по авторам, заказам и т.д.

Системные таблицы, известные также под названием системный каталог (system catalog), или словарь данных (data dictionary), содержат схему БД – описание базы данных, то есть сведения обо всем, что хранится в базе: наименованиях, структуре, размещении и типах. Как было уже сказано, сведения такого характера представляют собой метаданные (metadata), то есть данные о данных. Весь жизненный путь некоторого фрагмента данных – от создания до уничтожения – отражается в словаре данных так же, как и информация о его логической и физической структуре [9].

Системные таблицы обычно поддерживаются самой СУБД, однако доступ к ним можно получить так же, как и к любым другим таблицам.

2.3. Независимость

В жизни все стремятся к независимости. То же самое при управлении базами данных. Независимость данных позволяет изменять приложение, не изменяя для этого структуру БД, и изменять конструкцию БД, не оказывая при этом влияния на работу приложений. СУБД не должна вынуждать вас выносить окончательное решение о том, какие данные вы должны сохранять, как получать к ним доступ и что будет нужно вашим пользователям. Система не должна становится бесполезной при изменении ваших потребностей [10].

Реляционная модель обеспечивает независимость данных на двух уровнях – физическом и логическом.

Физическая независимость данных означает с точки зрения пользователя, что представление данных абсолютно не зависит от способа их физического хранения. Поэтому физическое перемещение данных не может повлиять на логическую структуру БД и ваше восприятие данных.

Например, при недостатке места для хранения данных может потребоваться установка дополнительных физических носителей. Другой тип независимости – логическая независимость означает, что изменение связей между таблицами, столбцами и строками не влияет на правильное функционирование приложений и текущих запросов. Вы можете разбивать таблицы по строкам или столбцам, а приложения и запросы все равно будут выполняться, как и раньше. Несмотря на изменение логической структуры базы данных, вы всегда можете воспользоваться своими старыми запросами.

2.4. Язык высокого уровня

Правила Кодда требуют, чтобы весь диалог с базой данных велся на едином языке. В мире коммерческих СУБД такой язык – это SQL (structured query language, язык структурированных запросов). Он является одновременно и языком управления данных (data manipulation language, DML), и языком определения данных (data definition language, DDL).

Можно сказать, что SQL используется для манипуляции с данными, определения данных и администрирования данных. Любая подобная операция выполняется с помощью оператора или команды языка SQL.

Имеется две разновидности операций по манипуляции с данными – выборка данных и модификация данных. Операции по выборке, чаще называемые запросами (queries), осуществляют поиск в БД, извлекают затребованную вами информацию и отображают ее. А модификация означает добавление, удаление или изменение данных.

Во всех запросах SQL используется оператор SELECT. Операции по модификации выполняются с помощью операторов INSERT, DELETE, UPDATE.

Например, следующий оператор SELECT покажет вам данные столбцов Фамилия и Адрес из таблицы Автор:

SQL>SELECT Фамилия, Адрес FROM Автор;

Для определения данных и структурированного доступа к ним служат операторы CREATE, ALTER, DROP. Они позволяют задать предложение SQL для определения той или иной реляционной таблицы, входящей в структуру БД, или для доступа к ней.

Например, следующая команда создает таблицу Автор2 с двумя столбцами – Фамилия и Адрес для хранения символьной информации:

SQL>CREATE TABLE Автор2 (Фамилия char (20), Адрес char (40));

Администрирование данных обеспечивают операторы администрирования, которые позволяют координировать совместное использование базы данных и поддерживать ее в наиболее эффективном состоянии.

Например, пользователю с именем Luna разрешается выбирать данные из таблицы Автор:

SQL>GRANT SELECT ON Автор TO Luna;

Учтите, что мы не рассматриваем синтаксис языка SQL. Все это вводные замечания, необходимые для понимания концепций, лежащих в основе моделирования.

2.5. Реляционные операции

В определении СУБД упоминаются три операции по выборке данных: проектирование выбирает столбцы, выбор – строки, операция соединения собирает вместе данные из связанных таблиц.

Логическая и физическая независимость означает, что вам не нужно беспокоиться о физическом расположении данных и о том, как их искать – это проблемы исключительно СУБД. SQL является непроцедурным языком программирования, так как он позволяет вам выразить то, что вы хотите получить, не вдаваясь в детали самого процессе эти операции записываются с использованием оператора SELECT. Например, если вы хотите просмотреть все строки таблицы Автор (табл. 1), содержащей информацию об авторах, но вам нужны только их адреса:

SQL>SELECT Адрес FROM Автор;

результат проектирования:

Адрес

Академгородок 10, кв. 10

Академгородок 20, кв. 46

Мира 130, кв. 24

Опять-таки не смотрите на синтаксис языка. Взгляните на это с абстрактной точки зрения: операция проектирования определяет под-

множество столбцов в таблице. Обратите внимание, что результат выполнения проектирования также отображается в форме таблицы.

Операция выбора позволяет вам получать из таблицы подмножества ее строк. Чтобы указать, какие строки вам нужны, соответствующие условия нужно указать после WHERE. Например, вы хотите получить фамилии писателей, проживающих в Академгородке:

SQL>SELECT Фамилия FROM Автор WHERE Адрес LIKE 'Академгородок%';

Результат выбора:

Фамилия

Деменюк

Корякина

Операция соединения может работать одновременно с одной или несколькими таблицами, соединяя данные таким образом, что вы сможете легко сопоставить или выделить определенную информацию в БД.

Операция соединения обеспечивает реляционную модель и SQL необходимой мощностью и гибкостью. Вы можете легко выявить любую связь, существующую между данными, а не только связи, введенные при конструировании базы.

Когда вы «соединяете» две таблицы, на период действия запроса они как бы становятся единой таблицей. Операция соединения соединяет данные, сравнивая значения в заданных столбцах и отражая результат.

2.6. Представления

Представление (view) – это альтернативный способ просмотра данных из нескольких таблиц. Еще их иногда называют виртуальными таблицами. Таблицы, на основе которых работают представления, называют базовыми. Представление можно рассматривать как перемещаемую по таблицам рамку, через которую вы можете увидеть только необходимую вам часть информации. Представление можно получить из одной или нескольких таблиц БД, используя операции выбора, проектирования и соединения. Представления позволяют вам создавать таблицы для специальных целей. Представления,в отличие от «настоящих», базовых таблиц,физически не хранятся в БД. Но представление – это и не копия некоторых данных, помещаемая в другую

таблицу. Когда вы изменяете данные в представлении, то тем самым изменяете данные в базовых таблицах.

2.7. Null

В реальном мире управления информацией данные часто являются неизвестными или неполными: вы можете забыть узнать адрес автора, автор может скрывать свой адрес и т. п. Такие пропуски информации создают «дыры» в таблицах.

Проблема, конечно, состоит не в простой неприглядности подобных дыр. Опасность в том, что из-за них ваша база данных может стать противоречивой. Чтобы сохранить целостность данных для обработки пропущенной информации, используется понятие нуля (null). Null не означает пустое поле или обычный математический нуль. Он отображает тот факт, что значение неизвестно, недоступно или неприменимо. Существенно, что использование нулей означает переход с двухзначной логики (да/нет) на трехзначную (да/нет/может быть).

2.8. Целостность

Целостность – очень серьезный вопрос при управлении базами данных. Ведь как уже было сказано, данные – одна из главных ценностей предприятия. Поэтому, что будет, если они станут ложными? Например, деньги с вашего счета в Сбербанке по ошибке окажутся на другом счете, в другом банке и в другой стране.

Под термином целостность (integrity) понимается взаимная согласованность отдельных фрагментов данных и их корректность. А согласованность (consistency) означает, что все порций данных в БД должны быть единообразно смоделированы и включены в систему. Квалифицировать данные как корректные можно в том случае, если они достоверны, точны и значимы [9].

Несогласованность между данными может возникнуть по разным причинам. Несогласованность или противоречивость данных может возникнуть вследствие сбоя системы – проблемы с аппаратным обеспечением, ошибки в программном обеспечении и т.п. Реляционная СУБД защищает данные от такого типа несогласованности, гарантируя, что команда SQL либо будет выполнена до конца, либо будет полностью отменена. Этот процесс называют управлением транзакциями. Любая операция манипулирования данными, осуществляемая

операторами SELECT, INSERT, UPDATE, DELETE – это и есть транзакция.

Другой тип целостности, называемый сущностной целостностью, связан с корректным проектированием базы данных. Сущностная целостность требует, чтобы ни один первичный ключ не имел null-значения.

Третий тип целостности называется ссылочной целостностью, означает непротиворечивость между частями информации, повторяющимися в разных таблицах. Важно, чтобы при изменении информации в одном месте, она соответственно изменялась и во всех других местах.

3-й день. Технология ORACLE

Татьяна пред окном стояла,
На стекла хладные дышала,
Задумавшись, моя душа,
Прелестным пальчиком писала
На отуманенном стекле
Заветный вензель О да Е.
А.С. Пушкин. Евгений Онегин, гл.3, XXXVII

3.1. От моделирования данных до приложений

Корпорация Oracle выпускает ряд продуктов и поддерживает технологии, которые ориентированы на разработчиков. Это инструменты и среды разработки, языки и концепции, которые могут быть использованы в процессе проектирования, разработки, тестирования и внедрения программных продуктов.

Направления, в которых развиваются продукты Oracle, можно разделить на следующие группы:

web-разработка (Интернет-приложения);
java-разработка (десктоп-приложения);
SOA-разработка (интеграция и управление);
database-разработка (программирование в СУБД).

Это разделение достаточно условно, так как некоторые продукты можно отнести одновременно к нескольким группам, другие продукты функционально дополняют друг друга в разных группах, т. к. практически все рассматриваемые нами инструменты являются частью единой платформы Oracle Fusion Middleware. Это большой плюс, т. к. взаимодействие различных компонентов конструируемой ИС уже отлажено и описано производителем. Рассмотрим некоторые продукты и технологии Oracle [14].

Упрощенный жизненный цикл разработки приложения включает пять фаз [13]:

моделирование;
разработка;
тестирование;
развёртывание;
мониторинг.

1 фаза. Моделирование данных с помощью SQL Developer Data Modeler

Oracle SQL Developer Data Modeler – это комплексное решение, позволяющее разработчикам проектировать реляционные модели взаимосвязей объектов для последующего преобразования их в полноценные БД. Продукт поддерживает логическое, реляционное, многомерное моделирование и моделирование типов данных, предлагая возможности многоуровневого проектирования и построения концептуальных диаграмм сущностей и связей. Пользователи могут создавать, расширять и модифицировать модели, а также сравнивать их с уже существующими.

Модели данных являются мощными коммуникационными средствами, которые используются при инициации новых проектов, а также при консолидации и обновлении существующих проектов. Data Modeler предлагает множество функциональных возможностей для моделирования данных и баз данных, включая:

Визуальное моделирование взаимосвязей между сущностями – поддерживает нотации Баркера и Бахмана, чтобы разработчики могли переключаться между моделями для удовлетворения потребностей клиентов или для создания и сохранения различных визуальных представлений моделей.

Ускоренное преобразование ERD-моделей в реляционные модели – трансформация всех правил и решений, сделанных на концептуальном уровне, в реляционную модель, детали в которой уточняются и обновляются.

Разделение реляционной и физической моделей – позволяет разработчикам создавать одну реляционную модель для разных версий базы данных или для разных баз данных, включая Oracle Database, IBM DB2 V7 и V8 для платформ Linux, UNIX, Windows и OS/390, а также Microsoft SQL Server 2000 и 2005.

Полный набор физических определений для баз данных – поддерживает такие физические определения, как секции, роли и табличные пространства для конкретных версий базы данных в средах с разными СУБД от разных производителей, обеспечивая большую согласованность и повышение продуктивности разработчиков.

Продукт интегрируется с Oracle SQL Developer – популярным графическим инструментом Oracle для разработки баз данных, – чтобы предоставить разработчикам возможность открывать и просматривать созданные ранее структуры, а также выполнять запросы и формировать отчеты с использованием репозитория отчетов.

Решение Oracle SQL Developer Data Modeler доступно для всех редакций Oracle Database 11g и работает в средах Windows, Linux и Mac OS X. Кроме того, предлагается версия для Oracle Database 10g. Продукт лицензируется по пользователям [14].

2 фаза. Разработка приложения с помощью Oracle SQL Developer и Oracle Application Express

Oracle SQL Developer – бесплатный инструмент для написания SQL-запросов, разработки PL/SQL пакетов, процедур, функций, триггеров и т. п. Этот инструмент написан на языке Java и является кроссплатформенным. Oracle SQL Developer интегрируется с APEX для разработки и администрирования приложений.

Возможности Oracle SQL Developer:

интегрированная среда разработки БД;
облегчённый интерфейс, упрощающий и улучшающий разработку БД;
запуск и настройка SQL;
разработка и отладка PL/SQL;
просмотр объектов БД;
интегрированная утилита миграции БД;
выполнение и создание отчётов;
просмотр, создание и редактирование данных в БД;
интегрированная поддержка управления версиями;
экспорт объектов БД в SQL скрипты;
генерация SQL скриптов из словаря данных;
чтение и форматирование трассировочных файлов;
расширяемость через Java и XML.

APEX является бесплатным продуктом, интегрированным с СУБД Oracle Database.

Изначально APEX предназначался для создания HTML-интерфейса к базе данных. В настоящее время выпущена 4-я версия продукта, который стал полноценной средой проектирования и разработки web-приложений любой сложности с интегрированной БД. На базе APEX и бесплатной редакции Oracle Database eXpress Edition (XE) можно создавать сайты и порталы, которые не требуют затрат на лицензирование.

Характерной особенностью этой среды разработки является то, что для работы с ней не требуется высокой квалификации в web-программировании и HTML-верстке. APEX представляет собой конструктор готовых блоков сайта. Фактически, минимально подготовленный пользователь может создавать рабочие сайты со встроенными средствами аутентификации и безопасности, современным дизайном и интерфейсом. С другой стороны, это гибкий инструмент, и квалифицированный разработчик может создавать страницы и сайты любого дизайна и структуры.

Немаловажным является и то, что работоспособность этого сайта будет поддерживаться мощной и надежной базой данных Oracle Database. Сайты и порталы, разработанные на APEX, способны обслуживать сотни пользователей, т. е. отвечают требованиям, предъявляемым по масштабируемости к Интернет-приложениям [14].

В состав APEX входят следующие четыре основных компонента.

Application Builder – собственно среда разработки web-страниц и бизнес-правил.

SQL Workshop – среда управления объектами базы данных (индексы, таблицы, представления и т. п.). Включает мастер создания SQL запросов для пользователей, которые не обладают знаниями в языке SQL.

Utilities – импорт и экспорт данных, генерация SQL-скриптов на изменение структуры базы данных, отчеты и восстановление удаленных объектов.

Administration – управление пользователями, настройками, правами доступа и просмотр отчетов.

APEX включает в себя следующие возможности:

среда разработки имеет простой и эффективный web-интерфейс, т. е. для начала разработки не требуется специализированных сред, разработка может вестись с любого компьютера с web-браузером;

помощники миграции из настольных баз данных и электронных таблиц;

встроенный мастер генерации отчетов в формате PDF;

инструменты для интеграции и web-сервисами;

большое количество шаблонов пользовательского интерфейса;

интуитивно-понятное управление рабочим пространством;

управление объектами по принципу Drag&Drop;

графический помощник создания SQL-запросов;

защищенность данных сессии после авторизации пользователя;

встроенный редактор PL/SQL;

мастер создания диаграмм и отчетов на сайте;

поддержка более 20 языков, включая русский.

APEX является кросс-платформенной системой, т. е. он успешно работает как на операционной системе Windows, так и на Linux, Solaris, HP-UX, MAC OS и других.

Одним из простейших примеров применения APEX на предприятии является переход от настольных баз данных и электронных таблиц (например, MS Access, MS Excel) к web-представлению этих баз и документов. Это бывает очень полезным, когда необходимо обеспечить одновременный доступ для редактирования одного и того же документа, особенно когда пользователи находятся в территориально удаленных офисах. В APEX встроен инструмент конвертации из таблиц Excel в таблицы Apex. После конвертации эти таблицы становятся доступны на корпоративном Интранет- или Интернет-сайте. Пользователь получает доступ к такой таблице после того, как вводит имя и пароль на сайте. Таким образом, можно организовать совместную работу над документом без пересылки его по электронной почте и т. п.

Разработка в APEX может вестись на нескольких языках: PHP, Java, PL/SQL. При разработке на PL/SQL, внутреннем языке базы данных Oracle Database, можно обойтись без промежуточного звена в виде

web-сервера Apache (Oracle HTTP Server), HTML-код будет выдавать непосредственно СУБД.

Фаза 3. Тестирование с помощью Oracle SQL Developer

Основные возможности [13]:
тестирование SQL и PL/SQL;
проверка результата;
просмотр планов выполнения SQL;
отладка PL/SQL;
производительность;
анализ трассировочных файлов;
диагностическое покрытие кода PL/SQL;
Profiler на уровне строки;
иерархический Profiler.

Фаза 4. Развёртывание

Продукт: Oracle Enterprise Manager.
Основные возможности [13]:
развёртывание;
объекты БД;
начальные данные;
приложения;
патчирование;
использование сравнения схем;
генерация скриптов для патчирования;
Application Express развёртывание приложения;
приложения развёртываются как SQL скрипты;
приложение может включать создание объектов базы данных.

Фаза 5. Мониторинг

Продукт: Oracle Enterprise Manager (OEM).

Oracle Enterprise Manager – комплекс средств для централизованного управления системами, созданными на основе продуктов Oracle, включая базы данных, серверы приложений, HTTP-серверы, Интернет-приложения и т.д.

OEM включает в себя:

Oracle Management Service (OMS) – управляющий сервер, реализующий всю логику работы OEM. Пользователи работают с

OEM через Web browser по протоколу http/https, используя интерфейс, предоставляемый OMS. Управляющий сервер имеет свой репозиторий, где он хранит необходимую для работы информацию о всех управляемых объектах. Репозиторий хранится в БД Oracle.

Oracle Management Agent (OMA) – должен быть установлен и работать на каждом узле, находящемся под управлением OEM. OMA выполняет задания, которые исходят от управляющего сервера. Выполнение этих заданий может происходить в заранее указанные моменты времени или с определенной периодичностью.

Oracle Enterprise Manager Grid Control – средство управления Oracle Grid.

Oracle Enterprise Manager Database Control – облегчённая версия OEM Grid Control, предназначенная для управления только одним экземпляром или одним кластером баз данных Oracle, устанавливается по умолчанию с каждой базой данных Oracle.

Oracle Enterprise Manager 10g Application Server Control – облегчённая версия OEM Grid Control, предназначенная для управления только одним экземпляром или одним кластером серверов приложений Oracle, устанавливается по умолчанию с каждым Oracle Application Server.

Основные возможности [13]:

мониторинг SQL;

Active Session History (ASH);

Active Workload Repository (AWR);

SQL-операторы с высокой нагрузкой (Top SQL);

долго выполняющийся SQL;

мониторинг Application Express приложений;

использование;

производительность;

медленно работающие страниц.

3.2. Установка JDK

Для работы многих программ, в том числе Data Modeler, Oracle SQL Developer необходимы JRE или JDK.

Java, JRE, JDK? Что это такое? Java – объектный язык программирования, разработанный компанией Sun на основе языка C++. Компания предоставляет на своем сайте для свободного доступа спецификацию языка (описывающая лексику, типы данных, основные конструкции) и инструментальные средства разработки приложений – Java Development Kit (JDK).

Главная задача, которую преследовали разработчики – создание надежного платформо-независимого объектного языка, который позволял бы разрабатывать небольшие мобильные приложения для web – аплеты. Технология разработки и использования java-аплетов основана на двух стандартизованных компанией компонентах: на мобильных Java-байт кодах – формате представления результатов компиляции исходного программного кода java-аплета – и на спецификации виртуальной машины Java (JVM) – интерпретаторе мобильных java-байт кодов. Скомпилированные аплеты должны храниться на web-сервере. Их вызов на машину клиента обеспечивается браузером при просмотре HTML-страницы, в которой с помощью специальных тегов встроен вызов аплета с передачей ему фактических параметров. После вызова мобильных java-байт кодов на сторону web-клиента их исполнение осуществляется JVM, встроенной в браузер.

Наряду с использованием java для создания аплетов широко используются системы программирования с входным языком java, основанные на традиционной технологии.

Интерес к Java был также обусловлен появлением таких технологий, как J2EE, включая JSP (Java Server Pages), J2ME сделавших Java наиболее популярной платформой для создания корпоративных решений, поддерживаемой почти всеми производителями ПО. Основная сфера применения Java – это приложения масштаба предприятия и многозвенные распределенные системы, базирующиеся на J2EE-совместимых серверах приложений.

В 1999 г. Sun объявила о разделении развития платформы Java 2 на три направления: Java 2 Platform Standard Edition (J2SE); Java 2 Platform Enterprise Edition (J2EE); Java 2 Platform Micro Edition (J2ME).

J2SE предназначается для использования на рабочих станциях и ПК. Standard Edition – основа технологии Java и прямое развитие JDK (средство разработчика было переименовано в j2sdk).

J2EE содержит все необходимое для создания сложных, высоконадежных, распределенных серверных приложений. Enterprise Edition – это набор мощных библиотек (например, Enterprise Java Beans, EJB) и пример реализации платформы (сервера приложений, Application Server), которая их поддерживает.

J2ME является усечением Standard Edition, чтобы удовлетворять жестким аппаратным требованиям небольших устройств, таких как карманные компьютеры и сотовые телефоны.

В 1997 года Sun начала предлагать Java Runtime Environment (JRE**)** – среду выполнения Java. Это минимальная реализация виртуальной машины, необходимая для исполнения Java-приложений, без компилятора и других средств разработки. Если пользователь хочет только запускать программы, это именно то, что ему нужно.

JDK долгое время было базовым средством разработки приложений. Оно не содержит никаких текстовых редакторов, а оперирует только уже существующими java-файлами. Компилятор представлен утилитой javac (java compiler). Виртуальная машина реализована программой java. Для тестовых запусков апплетов существует специальная утилита appletviewer. Наконец, для автоматической генерации документации на основе исходного кода прилагается средство javadoc.

Где взять? Комплект разработчика JDK бесплатно распространяется с сайта http://java.sun.com/javase/downloads/index.jsp

1. Запустите Java и узнайте ее версию: выберите **Пуск, Программы, Выполнить**, введите команду **cmd**, в открывшейся консоли введите команду **java -version**.Если сообщение как на рис. 1, то необходимо установить java, то есть выполняем пункт 5. Если сообщение примерно как на рис. 2, то java уже установлена.

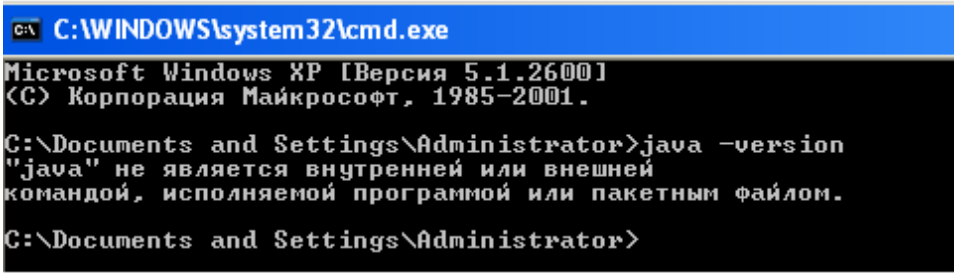

Рис. 1: Java не установлена

2. Запустите установку: выберите **jdk-6u20-ea-bin-b02-windows-i586-01_apr_2010.exe, M2.**

3. Разработчики программы приветствуют вас, выберите **Next**.

4. В окне **License Agreement** предлагается ознакомиться с условиями лицензионного соглашения, выберите **Accept>**.

6. В окне **Custom Setup** предлагается выбрать программы для установки, выберите по умолчанию, то есть все, обратите внимание на каталог **C:\Program Files\Java\jdk1.6.0_20** где будет размещаться приложение**,** нажмите **Next**.

7. После завершения установки: выберите **Finish**.

8. Войдите в каталог **jdk1.6.0_20**, там должны быть следующие папки: bin (здесь находятся основные утилиты, например компилятор и JVM); demo (примеры программ); jre (файлы, имеющие отношение непосредственно к JRE); lib (библиотеки JDK). Кроме того здесь также должен быть файл src.zip. В нем находятся исходные кода библиотек, которые вы при желании можете изучить самостоятельно.

9. Перезагрузите компьютер.

10. Запустите Java и проверьте ее версию: выберите **Пуск | Программы | Выполнить**, наберите **cmd**, в открывшейся консоли введите команду: **java -version**

Если результат примерно такой как на рис. 2, то все установлено правильно.

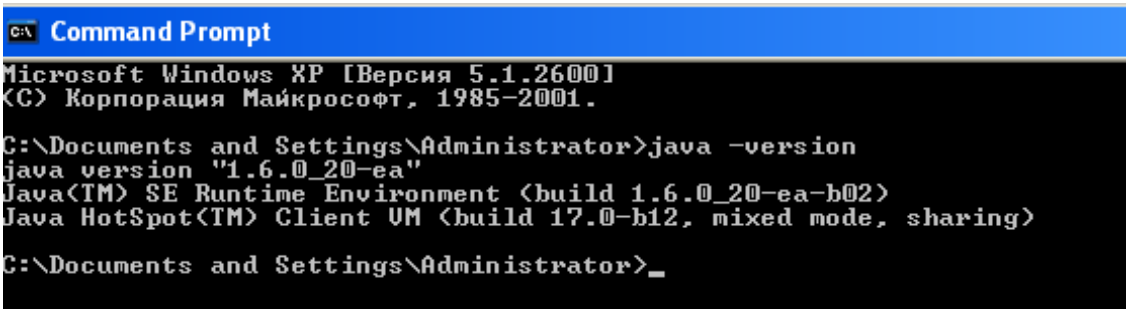

Рис. 2 Java установлена

3.3. Установка JDK (Linux)

Загружаем JDK со страницы http://java.sun.com/javase/downloads/index.jsp

Устанавливаем jdk-6u20-linux-i586-rpm.bin по инструкции http://java.sun.com/javase/6/webnotes/install/jdk/install-linux-self-extracting.html

11. Скопируйте **jdk-6u6-linux-i586-rpm.bin** в каталог **/home/student/Загрузка** в котором student имеет права на запись.

12. Запустите консоль: выберите **Утилиты, Консоль Терминалы**

13. Войдите почти как администратор: введите команду **sudo su**

14. Смените текущий каталог: введите команду

cd /home/student/Загрузка

15. Проверьте, где вы: введите команду **ls**

16. Установите права на выполнение с помощью команды:

chmod a+x jdk-6u6-linux-i586-rpm.bin

Если команда выполнена, то сообщений не должно быть. Если появилось сообщение о ошибке, исправьте и заново выполняйте команду.

17. Запустите установку с помощью команды:

./jdk-6u6-linux-i586-rpm.bin

18. Просмотрите лицензионное соглашение, пролистывая страницы нажитием **Enter**, выберите **Yes**, наблюдайте процесс установки.

19. Смените текущий каталог: введите команду **cd /usr/java/**

20. Просмотрите каталог: введите команду **ls**.

21. Запустите java-программу: введите **три** команды

cd /usr/java/jdk1.6.0_06/demo/jfc/SwingSet2

chmod +x SwingSet2.jar

./SwingSet2.jar

Наблюдайте красоту.

3.4. Установка Data Modeler

Загружаем datamodeler-3.1.4-710.zip со страницы

http://www.oracle.com/technetwork/developertools/datamodeler/downloads/index.html

1. Разархивируйте datamodeler-3.1.4-710.zip в каталог **c:\datamodeler**

2. Запустите CASE-средство: откройте каталог **c:\datamodeler**, выберите **datamodeler.exe**, **M2**.

3.4. Установка Data Modeler (Linux)

Установочный файл datamodeler-2.0.0-584-no-jre.zip не содержит JRE. Поэтому надо установить JRE. При необходимости в файле datamodeler.conf из каталога \datamodeler\bin необходимо указать SetJavaHome. (SetJavaHome /usr/java/jdk1.6.0_20)

1. Разархивируйте **datamodeler-2.0.0-584-no-jre.zip** в каталог **/opt/datamodeler/**

2. Для соединения с Oracle 10g XE редактируем файл /datamodeler/datamodeler/bin/**datamodeler.conf**, добавляем строку

AddVMOption -Duser.region=US

3. Запустите CASE-средство: откройте каталог **/opt/datamodeler/**, выберите **datamodeler.sh**, **M2**

3.5. Установка Oracle SQL Developer

Загружаем sqldeveloper-3.2.20.09.87.zip со страницы http://www.oracle.com/technetwork/developer-tools/sqldeveloper/downloads/index.html

Oracle SQL Developer включает JDK1.6.0_35.

1. Разархивируйте **sqldeveloper-3.2.20.09.87** в каталог **c:\sqldeveloper**

2. Для соединения с Oracle 10g XE редактируем файл /sqldeveloper/sqldeveloper/bin/**sqldeveloper.conf**, добавляем строку

AddVMOption -Duser.region=US

3. Запустите CASE-средство: откройте каталог **c:\sqldeveloper**, выберите **sqldeveloper.exe, M2**

3.6. Установка Oracle SQL Developer (Linux)

Установочный файл sqldeveloper-1.5.54.40-1.noarch.rpm (sqldeveloper-2.1.1.64.45-1.noarch.rpm) не содержит JDK. Поэтому предварительно надо установить JDK.

1. Выполните установку: выберите s**qldeveloper-1.5.54.40-1.noarch.rpm**, **M2**.

2. Выполните установку с помощью команды: r**pm -Uhv sqldeveloper-1.5.54.40-1.noarch.rpm**

3. Для соединения с Oracle 10g XE редактируем файл **sqldeveloper.conf,** добавляем строку

AddVMOption -Duser.region=US

4. При необходимости редактируем файл **sqldeveloper.conf,** добавляем строку

SetJavaHome /usr/java/jdk1.6.0_20

5. Запустите CASE-средство: выполните команды

cd sqldeveloper
sqldeveloper

При запуске может появится ошибка:

Type the full pathname of a J2SE installation (or Ctrl-C to quit), the path will be stored in ~/.sqldeveloper/jdk

6. Для исправления ошибки создайте каталог командой:

mkdir -p ~/.sqldeveloper

7. Создайте пустой файл командой: **touch ~/.sqldeveloper/jdk**

8. Выполните редактирование файла **jdk** добавив в него **/usr/java/jdk1.6.0_20**

3.7. Установка Oracle10g XE

Для установки необходим объем оперативной памяти: 256 megabytes минимум, 512 megabytes рекомендуется.
Для установки необходим объем жесткого диска: 1.6 gigabyte минимум.

Загружаем OracleXEUniv.exe со страницы:
http://www.oracle.com/technetwork/database/express-edition/downloads/index.html

1. Входим в Windows как пользователь с правами администратора.

2. Запускаем установку: выберите **OracleXEUniv.exe, M2**.

3. В окне **Oracle Database XE - Install Wizard welcome** выберите **Next**.

4. В окне **License Agreement** выберите **I accept...** и нажмите **Next**.

5. В окне **Choose Destination Location** выберите каталог по умолчанию, нажмите **Next**.

6. В окне **Specify Database Passwords** введите пароль **student2010** и его подтверждение **student2010**, который используется для логинов базы данных sys и system. Нажмите **Next**.

7. В окне **Summary** просмотрите настройки установки и нажмите **Install**. Наблюдайте процесс установки.

8. В окне **Install Shield Wizard Complete** выберите **Launch the Database homepage**, нажмите **Finish**.

9. В окне браузера на домашней странице войдите в систему: в поле **Username** введите **system**, в поле **Password** введите **student2010**, нажмите **Login**.

10. Активируйте тестовую БД: выберите **Administration, Database Users**, выберите **hr**, в поле **password** введите **hr**, в поле **confirm password** введите введите **hr**, в списке **account status** выберите **unlocked**, нажмите **alter user.**

11. Создайте свою учетную запись: выберите **Administration**, **Database Users**, выберите **create**, в поле **username** введите свой логин, например, **soko**, в поле **password** введите **soko**, в поле **confirm password** введите **soko**, в списке **account status** выберите **unlocked**, в области **roles** выберите права администратора галочкой **connect, resorce, dba**, нажмите **create**.

12. Выйдите из системы: нажмите **Logout**.

13. Настройте СУБД: выберите **Пуск, Панель управления, Администрирование, Службы**, выберите **OracleServiceXE**, выберите **МП**, выберите **Свойства**, в списке **Тип запуска** выберите **Вручную**, нажмите **Ладно**.

14. Перезагрузите компьютер.

15. Запустите СУБД: **Пуск, Программы, Oracle Database 10g Express Edition, Start Database**.

Должно появиться в окне сообщение:

C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleXETNSListener

Затребованная служба уже запущена.

C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleServiceXE

Служба "OracleServiceXE" запускается.....

Служба "OracleServiceXE" успешно запущена.

16. Войдите в систему под своим логином: выберите **Пуск, Программы, Oracle Database 10g Express Edition, Go to Database Home page**, в поле **Username** введите свой логин, например, **soko**, в поле **Password** введите **soko**, нажмите **Login**.

17. Выйдите из системы: нажмите **Logout**.

18. Запустите Oracle SQL Developer: выберите **Пуск | Программы | sqldeveloper.**

19. Создайте соединение: выберите **Connections, МП**, выберите **New connection**, в поле **Connection Name** введите **10hr**, в поле **Name** введите **hr**, в поле **Password** введите **hr**, в поле **hostname** введите **localhost**, в поле **SID** введите **xe**, нажмите **Test**, если сообщения об ошибке нет, то нажмите **Connect**.

20. Просмотрите таблицы схемы hr: в левом окне, в дереве объектов выберите плюсик у Tables, должно быть семь таблиц.

3.8. Установка Oracle10g XE (Linux)

Для установки необходим объем оперативной памяти: 256 megabytes минимум, 512 megabytes рекомендуется.
Для установки необходим объем жесткого диска: 1.5 gigabyte минимум.
Для установка требуются следующие библиотеки: glibc release 2.3.2 libaio, release 0.3.96.

21. Проверяем установленные библиотеки: rpm -qa | grep libaio

Результат должен быть такой:

libaio-static-devel-0.3.107-2mdv2009.1

libaio1-0.3.107-2mdv2009.1

libaio-devel-0.3.107-2mdv2009.1

22. Проверяем установленные библиотеки: rpm -qa | grep glibc

Результат должен быть такой:

glibc-devel-2.9-0.20081113.5.3mnb2

glibc_lsb-2.4.7-4mdv2009.1

glibc-2.9-0.20081113.5.3mnb2

Загружаем oracle-xe-univ-10.2.0.1-1.0.i386.rpm со страницы: http://www.oracle.com/technetwork/database/express-edition/downloads/index.html

23. Входим в систему с правами root.

24. Запускаем установку: **rpm -ivf oracle-xe-univ-10.2.0.1-1.0.i386.rpm**

Начинается установка. *Подготовка пакетов для установки...* Система проверяет объем жесткого диска, оперативной памяти и swap-раздела. При необходимости выводятся рекомендации. Например, *This system does not meet the minimum requirements for swap space. Based onthe amount of physical memory available on the system, Oracle Database 10g Express Edition requires 1024 MB of swap space. This system has 0 MB of swap space. Configure more swap space on the system and retry the installation.*

В конце выводится

Executing Post-install steps...

You must run '/etc/init.d/oracle-xe configure' as the root user to configure the database.

25. Выполните конфигурирование: **/etc/init.d/oracle-xe configure**

26. На запрос **Specify the HTTP port that will be used for Oracle Application Express [8080]:** нажмите **Enter**.

27. На запрос **Specify a port that will be used for the database listener [1521]**: нажмите **Enter**.

28. На запрос **Specify a password to be used for database accounts. Note that the same password will be used for SYS and SYSTEM** введите **student2010**.

29. На запрос **Confirm the password:** повторите ввод **student2010**

30. На запрос **Do you want Oracle Database 10g Express Edition to be started on boot (y/n) [y]** введите **n**

Конфигураирование заканчивается.

Starting Oracle Net Listener...Done

Configuring Database...Done

Starting Oracle Database 10g Express Edition Instance...Done

Installation Completed Successfully.

To access the Database Home Page go to "http://127.0.0.1:8080/apex"

31. Запустите listener и базу данных командой:

/etc/init.d/oracle-xe restart

32. Остановите listener и базу данных командой:

/etc/init.d/oracle-xe stop

33. Настройте Oracle Database XE Server Environment Variables:

откройте каталог

/usr/lib/oracle/xe/app/oracle/product/10.2.0/server/bin/oracle_env.sh

$ . ./oracle_env.sh

34. Установите возможность подключения удаленного клиента.

The Oracle Database XE home page is only available from the local machine, not remotely. If you want to enable access from a remote client, you should be aware that HTTPS cannot be used (only HTTP), so your login credentials are sent in clear text, and are not encrypted, so if you don’t need to set this up, it is more secure to leave it as the default setup.

You can also use SQL*Plus command line to enable access from remote clients. To use SQL*Plus command line to change this setting, log into SQL*Plus as system, and run the following command:

SQL> EXEC DBMS_XDB.SETLISTENERLOCALACCESS(FALSE);

3.9. Установка Apex

Загружаем **4.0.2.zip** со страницы

http://www.oracle.com/technology/products/database/application_express/download.html

1. Запустите СУБД: **Пуск, Программы, Oracle Database 10g Express Edition, Start Database**

C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleXETNSListener
Затребованная служба уже запущена.
Для вызова дополнительной справки наберите NET HELPMSG 2182.
C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleServiceXE
Служба "OracleServiceXE" запускается.....
Служба "OracleServiceXE" успешно запущена.
C:\oraclexe\app\oracle\product\10.2.0\server\BIN>
Обратите внимание, что служба "OracleServiceXE" должна быть успешно запущена.

2. Откройте каталог **C:\oraclexe\app\oracle\product\10.2.0\server\NETWORK\ADMIN**, откройте файл **sqlnet.ora** и закоментируйте строку
SQLNET.AUTHENTICATION_SERVICES = (NTS)
Дистрибутив представляет собой ZIP-архив с текстами SQL-скриптов создания объектов APEX в базе данных. Также в архив входят файлы изображений.

3. Распаковываем **apex_4.0.2.zip** на диск D:. Apex файлы находиться в каталоге **D:\apex_3.1.2\apex**.

4. Откройте каталог **D:\apex_3.1.2\apex** и скопируйте каталог **apex** на диск **c:** Apex файлы должны находиться в каталоге C:\apex.

5. В командной строке введите команду: **sqlplus /nolog**

6. В sqlplus введите команду: **connect sys as sysdba**

7. На запрос пароля введите: **student2010**

8. Создайте табличное пространство:

```
CREATE TABLESPACE APEX datafile
'C:\oraclexe\oradata\XE\APEX.dbf'
SIZE 500M
```

EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

9. Создайте табличное пространство:

CREATE TABLESPACE APEX_FILES datafile
'C:\oraclexe\oradata\XE\APEX_FILES.dbf'
SIZE 500M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

10. Закройте sqlplus

11. Установите рабочий каталог для sqlplus: в командной строке введите cd c:\apex

12. В командной строке введите команду: **sqlplus /nolog**

13. В sqlplus введите команду: **connect sys as sysdba**

14. На запрос пароля введите: **student2010**

15. Запустите установку: **@apexins APEX APEX_FILES TEMP /i/**

Установка идет 15-20 мин. Ждите.

В конце должно появится такое!

Upgrade completed successfully no errors encountered.
-- Upgrade is complete ----------
timing for: Upgrade
Elapsed: 00:01:16.98
...End of install if runtime install
...create null.sql
timing for: Development Installation
Elapsed: 00:23:09.09
Disconnected from Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

16. Установите рабочий каталог для sqlplus: в командной строке введите: cd c:\apex

17. В командной строке введите команду: **sqlplus /nolog**

18. В sqlplus введите команду: **connect sys as sysdba**

19. На запрос пароля введите: **student2010**

20. Запустите копирование картинок и скриптов: **@apxldimg.sql c:**

Должно быть такое!

PL/SQL procedure successfully completed.
old 1: create directory APEX_IMAGES as '&1/apex/images'
new 1: create directory APEX_IMAGES as 'c:/apex/images'
Directory created.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
Commit complete.
timing for: Load Images
Elapsed: 00:02:48.50
Directory dropped.

21. Меняем пароль: **@apxxepwd.sql studentfub**

Session altered.
...changing password for ADMIN
PL/SQL procedure successfully completed.
Commit complete.

3.10. Установка APEX (Linux)

1. Разблокируем учетную запись oracle, которая была создана при установке oracle.
2. В командной строке введите команду: **sqlplus /nolog**
3. В sqlplus введите команду: **connect sys as sysdba**
4. На запрос пароля введите: **student2010**
5. Создаем рабочее пространство:

```
CREATE TABLESPACE APEX datafile
'/usr/lib/oracle/xe/oradata/XE/APEX.dbf'
SIZE 500M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

CREATE TABLESPACE APEX_FILES datafile
'/usr/lib/oracle/xe/oradata/XE/APEX_FILES.dbf'
```

SIZE 500M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO;

6. Копируем каталог apex в каталог пользователя oracle.
7. Открываем каталог **apex** и запускаем **sqlplus /nolog**
8. В sqlplus введите команду: **connect sys as sysdba**
9. Запустите установку: **@apexins.sql APEX APEX_FILES TEMP /i/**
10. Запустите копирование картинок и скриптов:
@apxldimg.sql /usr/lib/oracle/xe

PL/SQL procedure successfully completed.
old 1: create directory APEX_IMAGES as '&1/apex/images'
new 1: create directory APEX_IMAGES as
'/usr/lib/oracle/xe/apex/images'
Directory created.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
PL/SQL procedure successfully completed.
Commit complete.
timing for: Load Images
Elapsed: 00:00:41.53
Directory dropped.

11. Меняем пароль: **@apxxepwd.sql studentfub**

Session altered.
...changing password for ADMIN
PL/SQL procedure successfully completed.
Commit complete.

12. Выключаем систему: *SQL>* **shutdown immediate**

Database closed.
Database dismounted.
ORACLE instance shut down.
13. Запускаем **SQL> startup**
ORACLE instance started.
Total System Global Area 608174080 bytes
Fixed Size 1260316 bytes

Variable Size 184550628 bytes
Database Buffers 419430400 bytes
Redo Buffers 2932736 bytes
Database mounted.
Database opened.

3.11. Установка Oracle11g XE на Windows 7 Professional Edition, 64-bit

Для установки необходим объем оперативной памяти: 256 Мб минимум, 512 Мб рекомендуется.

Для установки необходим объем жесткого диска: 1.5 Гб минимум. Операционная система: Windows 7 x64 - Professional, Enterprise, Ultimate Editions, Windows 8 - Pro and Enterprise Editions, Windows 8.1 - Pro и Enterprise Editions.

Термины 32-разрядный и 64-разрядный описывают, каким образом процессор компьютера (ЦП) обрабатывает информацию. 64-разрядная версия Windows обрабатывает большие объемы оперативной памяти (ОЗУ) более эффективно по сравнению с 32-разрядной версией.

1. Чтобы определить, какая версия Windows запущена на компьютере под управлением Windows 7 выполните: нажмите кнопку **Пуск**, нажмите компонент **Компьютер** правой кнопкой мыши и выберите **Свойства**.

Если рядом с надписью Тип системы указано «64-разрядная операционная система», компьютер работает под управлением 64-разрядной версии Windows 7.

Если рядом с надписью Тип системы указано «32-разрядная операционная система», компьютер работает под управлением 32-разрядной версии Windows 7.

В зависимости от версии установленной ОС загружаем СУБД.

2. Загружаем OracleXE112_Win64.zip со страницы: http://www.oracle.com/technetwork/database/express-edition/downloads/index.html

Oracle 11g XE бесплатная версия СУБД Oracle, которая поддерживает большинство функциональности Standard Edition. Она доступна для ОС Windows и Linux.

Oracle 11g XE имеет следующие ограничения: максимальный размер базы данных составляет 11 ГБ, максимальный объем оперативной памяти XE который может использоваться 1 ГБ, только один экземпляр Oracle XE может быть установлена на одном компьютере,

XE использует только один CPU, так как она не распределяет операции между несколькими процессорами.

Служба контроля учетных записей (UAC) защищает операционную систему от повреждения вирусами и другими вредоносными программами. Делает она это с помощью всплывающего окна с требованием подтвердить какое-либо важное действие. Такие надоедливые всплывающие окна можно отключить, но компания MIcrosoft не рекомендует это делать. Но добрые люди советуют это сделать, чтобы избежать проблем при установки. Кстати, после удачной установки, можно UAC включить.

3. Отключите контроль учетных записей. Для этого нажмите кнопку «**Пуск**» и выберите пункт **Панель управления**. В поле поиска введите **uac** и затем выберите пункт **Изменение параметров контроля учетных записей.**

4. Чтобы отключить контроль учетных записей, переместите ползунок в положение **Никогда не уведомлять** и нажмите кнопку **ОК**. Если отображается запрос на ввод пароля администратора или его подтверждения, укажите пароль или предоставьте подтверждение.

5. Контроль учетных записей будет отключен после перезагрузки компьютера.

6. Отредактируйте файл C:\Windows\System32\drivers\etc**hosts.**

Если вы используете DHCP IP-адрес добавьте следующий код:

127.0.0.1 localhost

::1 localhost

127.0.0.1 workst

7. Если вы используете статический IP-адрес добавьте следующий код, где 192.168.100.56 ваш IP-адрес.

127.0.0.1 localhost

::1 localhost

127.0.0.1 workst

192.168.100.56 workst

Как установить статический адрес см. здесь: http://blog.mclaughlinsoftware.com/2009/11/26/windows-7-static-ip/

Учтите, что hostname (workst) должно быть **строчными** буквами.

8. Создайте имя учетной записи пользователя, которое не имеет пробелов, например, **mindalev,** и назначьте ему права администратора.

9. Входим в Windows как пользователь с правами администратора.

10. Запускаем установку: распаковываем OracleXE112_Win64.zip, запускаем **setup.exe** из **OracleXE112_Win64\DISK1\setup.exe**.

11. В окне **Oracle Database XE - Install Wizard welcome** (рисунок 1) выберите **Next**.

Рисунок 1

12. В окне **License Agreement** (рисунок 2) выберите **I accept the terms in the license agreement** и нажмите **Next**.

Рисунок 2

5. В окне **Choose Destination Location** (рисунок 3) можно принять местоположение по умолчанию или выполнить перенастройку. Если примите расположение по умолчанию (`oraclexe`), нажмите **Next.**

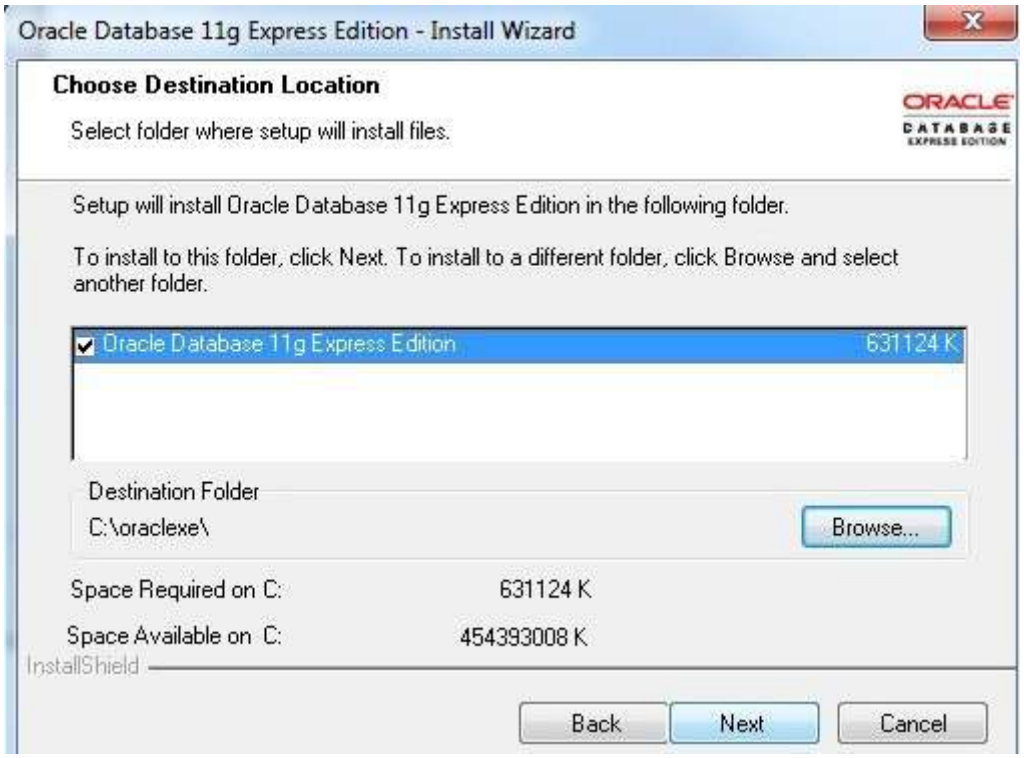

Рисунок 3

13. В окне **Specify Database Passwords** (рисунок 4) введите пароль **student2010** и его подтверждение **student2010**, который используется для логинов базы данных sys и system. Нажмите **Next**.

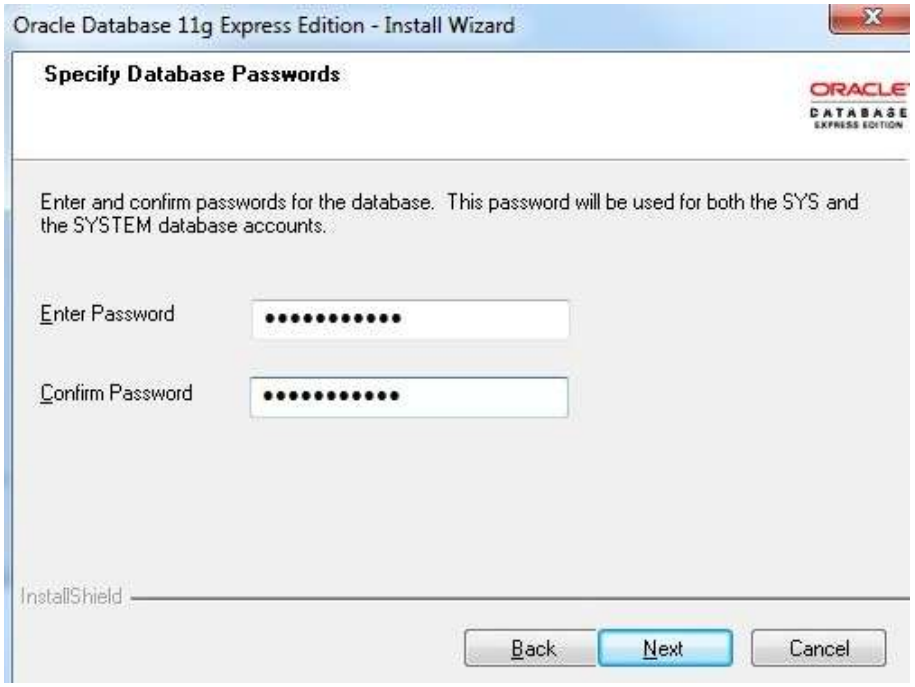

Рисунок 4

Пароль для INTERNAL и ADMIN Oracle Application Express пользовательских аккаунтов будет такой же как и для SYS и SYSTEM административных пользовательских аккаунтов.

Следующие номера портов это значения по умолчанию:

1521: слушатель базы данных Oracle

2030: службы Oracle для сервера Microsoft Transaction Server 8080: HTTP порт для XE Database графического пользовательского интерфейса Oracle

14. В окне **Specify Database Ports** (рисунок 5) если номера портов по умолчанию не используются, то установка использует их автоматически, без запроса. Если они находятся в эксплуатации, то вы можете ввести доступный номер порта. Нажмите **Next.**

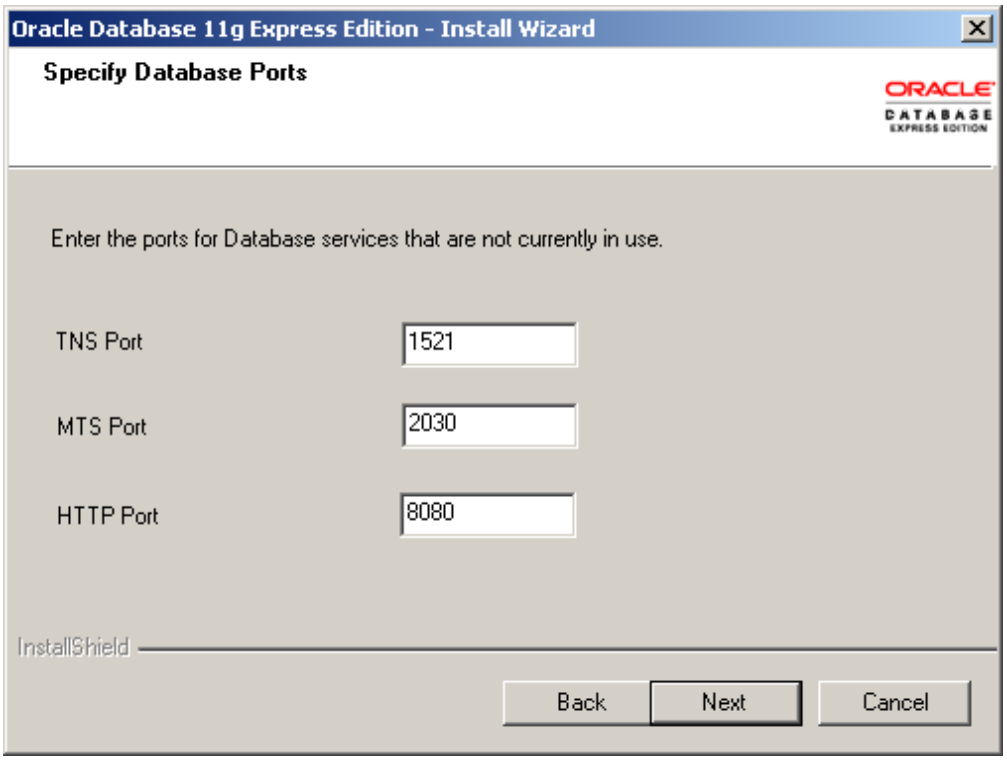

Рисунок 5

15. В окне **Summary** (рисунок 6) просмотрите настройки установки и нажмите **Install**. Наблюдайте процесс установки.

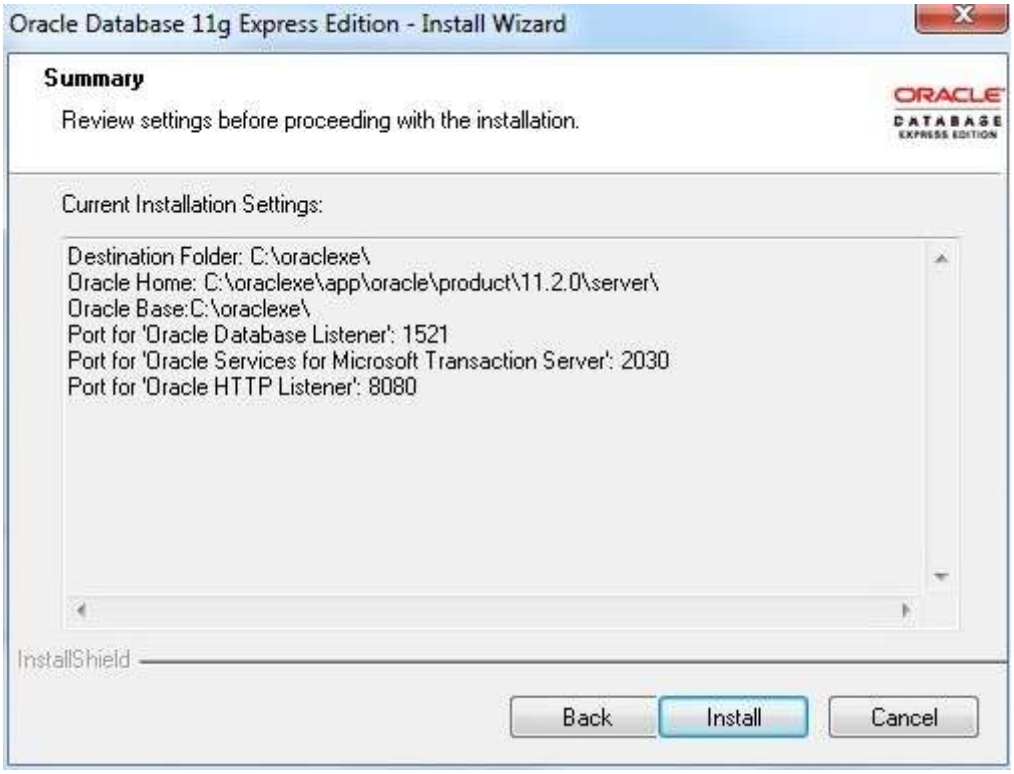

Рисунок 6

Если во время установки произошла ошибка как на рисунке 7. То как лечить читаем здесь:

http://www.hanmiaojuan.com/2013/03/install-oracle-xe-11g-for-windows7-64bits.html

http://www.sibsql.ru/uchebnye_materialy/Oracle-11g-XE-ustanovka-na-kompyutere-s-Windows-64bit

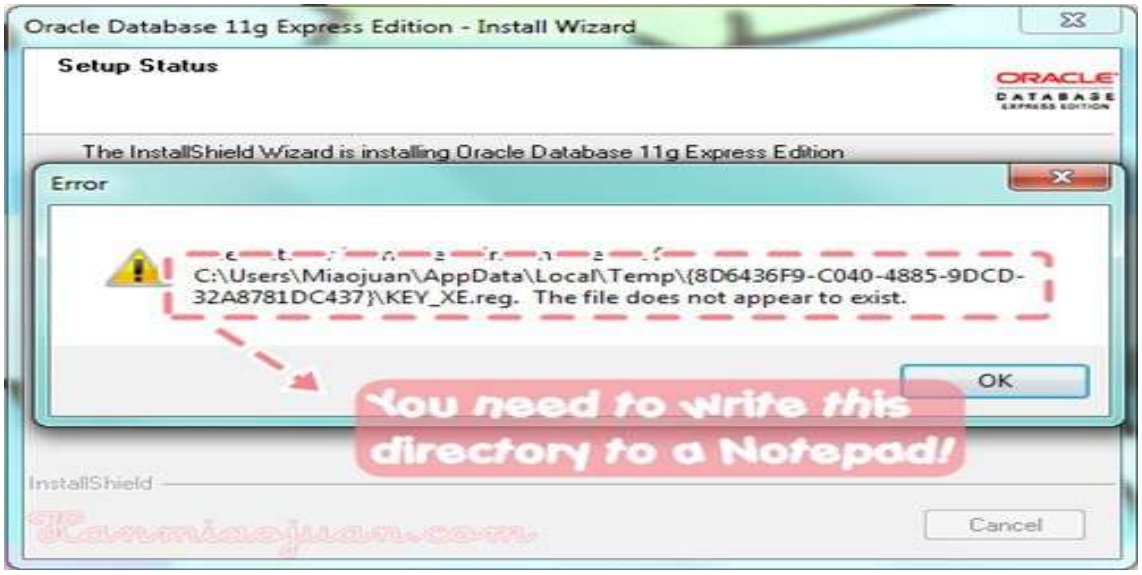

Рисунок 7

16. В окне **InstallShield Wizard Complete** нажмите **Finish**

Смотрим что установили.

17. Просмотрите службы (рисунок 8): **Пуск, Панель управления, Администрирование, Службы.**

OracleJobSchedulerXE		Disabled	Local System
OracleMTSRecoveryService		Manual	Local System
OracleServiceXE	Started	Automatic	Local System
OracleXEClrAgent		Manual	Local System
OracleXETNSListener	Started	Automatic	Local System

Рисунок 8

OracleServiceXE

Это движок базы данных.

OracleXETNSListener

Эта служба для прослушивания входящих соединений и передачи успешных подключений к движку базы данных. Если эта служба не

работает, вы не сможете подключиться к базе данных удаленно. Существующие соединения не будут затронуты.

OracleJobSchedulerXE
This service is used when external jobs are run. By default, it is disabled. If you plan to run external jobs (such as executables, batches, etc.), modify the account the service uses to use proper, low-privileged credentials and start the service.

OracleXEClrAgent
На платформах Windows, Oracle предлагает интеграцию со средой CLR. Так операция CLR выполняется с использованием процесса EXTPROC, это обычно делается с помощью специального (однопоточные) EXTPROC для одного сеанса. Это не может быть оптимальным способом для обработки вызовов CLR. ClrAgent обеспечивает многопоточную механизм так, что один процесс EXTPROC может служить несколько вызовов CLR.

This one is responsible of resolving in-doubt transactions when Oracle is participating in distributed transactions with Microsoft Transaction Server.

18. Настройте СУБД: выберите **Пуск, Панель управления, Администрирование, Службы**, выберите **OracleServiceXE**, выберите **МП**, выберите **Свойства**, в списке **Тип запуска** выберите **Вручную**, нажмите **ОК**.

19. Перезагрузите компьютер.

20. Просмотрите установленные программы: выберите **Пуск, Программы, Oracle Database 11g Express Edition** (рисунок 9).

Рисунок 9

Start and Stop Database Пуск и остановка базы данных используются для управления службой OracleServiceXE

Backup Database и **Restore Database.** Резервное копирование и восстановление баз данных это скрипты для резервного копирования или восстановления данных базы данных, используя программу RMAN.

Run SQL Command Line открывает консоль на базе SQL * Plus, которая может быть использована для выполнения команд SQL или для запуска сценариев к базе данных

Get Started Начать – открывает веб-сайт, используемой для исследования: памяти, объем дискового пространства, используемого табличными сегментами, текущих сеансов и основных сведения о клиентской информации и приложениях, активных SQL командах, показывает текущие значения параметров инициализации.

21. Найдите папку

\oraclexe\app\oracle\oradata\XE

Это папка, где файлы базы данных находятся после установки. Если новые файлы базы данных будут добавлены, то новые файлы могут быть размещены где-то еще, но критические файлы, такие как управляющий файл, файл базы данных системы, и т.д. находятся здесь.

22. Найдите папку

\oraclexe\app\oracle\product\11.2.0\server\network\ADMIN

Этот каталог содержит файлы конфигурации для соединения с базой данных:

listener.ora настраивает слушателя. Например, по умолчанию Oracle слушает порт 1521. Это может быть изменено путем изменения конфигурации в listener.ora и перезапуском службы прослушивателя. Этот файл управляет поведением на стороне сервера.

tnsnames.ora определяет сетевую конфигурацию для клиентских программ, таких как SQL*Plus. Например, когда вы устанавливаете соединение с сервисом XE, XE на самом деле псевдоним, который используется для разрешения конфигурации в реальную сеть,

используя tnsnames.ora. Если вы измените порт слушатель использует, вы должны также отражать эти изменения в tnsnames.ora.

23. Найдите папку
\oraclexe\app\oracle\diag\rdbms\xe\xe

Эта папка содержит различные виды файлов протоколов и трассировки для экземпляра базы данных. В то время как Oracle все еще имеет alert_xe.log под трассировки -Каталога, есть теперь еще один тип файла журнала под оповещения -Каталога называется log.xml. Как следует из названия, это XML файл в формате, содержащий все важные сообщения, что записи в базе данных. Однако, поскольку нет корневой узел в этом файле, большинство из основных редакторов XML не в состоянии показать его содержимое. Oracle представила инструмент под названием ADRCI исследовать его содержимое. Этот инструмент может быть запущен из командной строки.

24. Найдите папку
\oraclexe\app\oracle\diag\tnslsnr\<machinename>\listener

Эта папка похожа на соответствующей папке СУБД. Тем не менее, эта папка содержит войти и трассировки для слушателя. Как реляционными базами данных, файл log.xml можно читать с помощью ADRCI.

Logs for the server component installation are in the
OracleDatabaseXEServerInstall.log file locate2 in the ssstem root 2irectors which is tspicalls c:\WINDOWS.
You can fin2 the 2atabase creation logs in the `install_directory`
\app\oracle\product\11.2.0\server\config\log
directory. (`install_directory` is typically `c:\oraclexe` .)

25. Запустите СУБД: **Пуск, Программы, Oracle Database 10g Express Edition, Start Database**.
Должно появиться в окне сообщение:

Служба "OracleServiceXE" запускается.....

Служба "OracleServiceXE" успешно запущена.

26. Просмотрите службы (рисунок 10): **Пуск, Панель управления, Администрирование, Службы,** службы **OracleServiceXE и OracleXETNSListener** должны работать.

Рисунок 10

27. Запустите Начать: выберите **Пуск, Программы, Oracle Database 11g Express Edition**, выберите **Get Started** (рисунок 11), нажмите кнопку **Application Express**, в поле **Username** введите **system**, в поле **Password** введите **student2010**, нажмите **Login**. Должна запуститься консоль (рисунок (12).

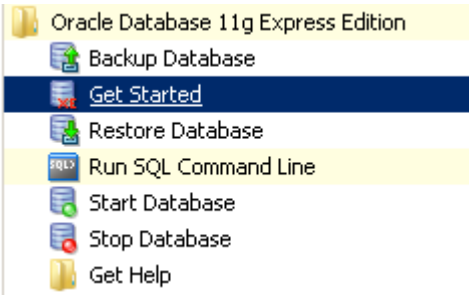

Рисунок 11

Рисунок 12

28. Создайте рабочее пространство для работы в APEX: в поле **Database user** (рисунок 13) выберите **User Existing**, в списке **Database Username** выберите **HR**, в поле **Application Express Username** введите **hr**, в поле **Password** введите **hr**, в поле **Confirm Password** введите **hr**, нажмите **Create Workspace**.

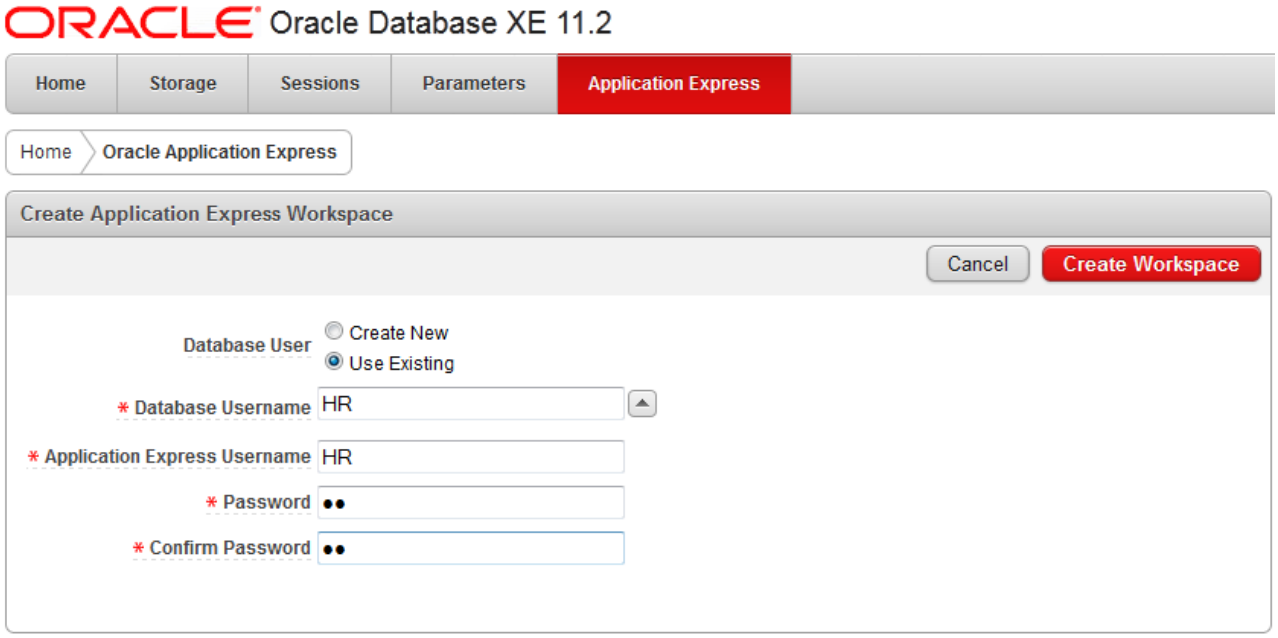

Рисунок 13

29. Войдите в рабочее пространство: введите в браузере http://localhost:8080/apex
в поле workspace введите **hr**, в поле username введите **hr**, в поле password введите **hr**, нажмите **login** (рисунок 14).

Рисунок 14

30. Выберите SQL Workshop, выберите Object Browser, найдите таблице как на рисунке 15.

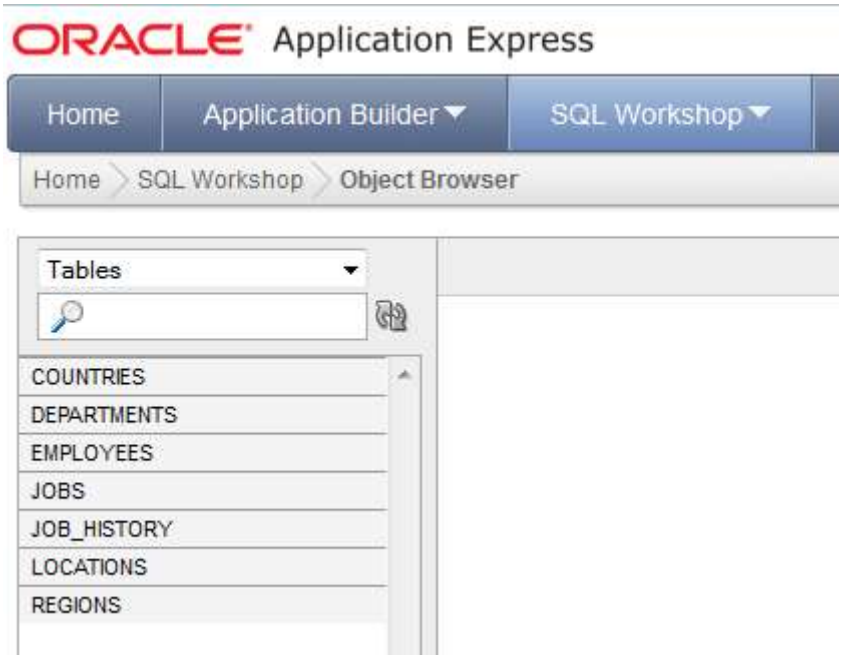

Рисунок 15

31. Выйдите из системы: нажмите **Logout**.

32. Запустите Oracle SQL Developer: выберите **Пуск, Программы, Sqldeveloper**.

33. Создайте соединение: выберите **Connections**, **МП**, выберите **New connection**, в поле **Connection Name** введите **system**, в поле **Name** введите **system**, в поле **Password** введите **student2010**, в поле **hostname** введите **localhost**, в поле **SID** введите **xe**, нажмите **Test**, если сообщения об ошибке нет, то нажмите **Connect**.

34. Активируйте тестовую БД: в окне дерева раскройте **system**, раскройте **Other Users**, выберите **hr**, **МП**, выберите **Edit user**, в окне Edit User выберите вкладку **User**, отключите флажком поля **Account In Locked, Password Expired**, как на рисунке **16**, в поле **New Password** введите **hr**, в поле **Confirm password** введите **hr**, нажмите **Принять**.

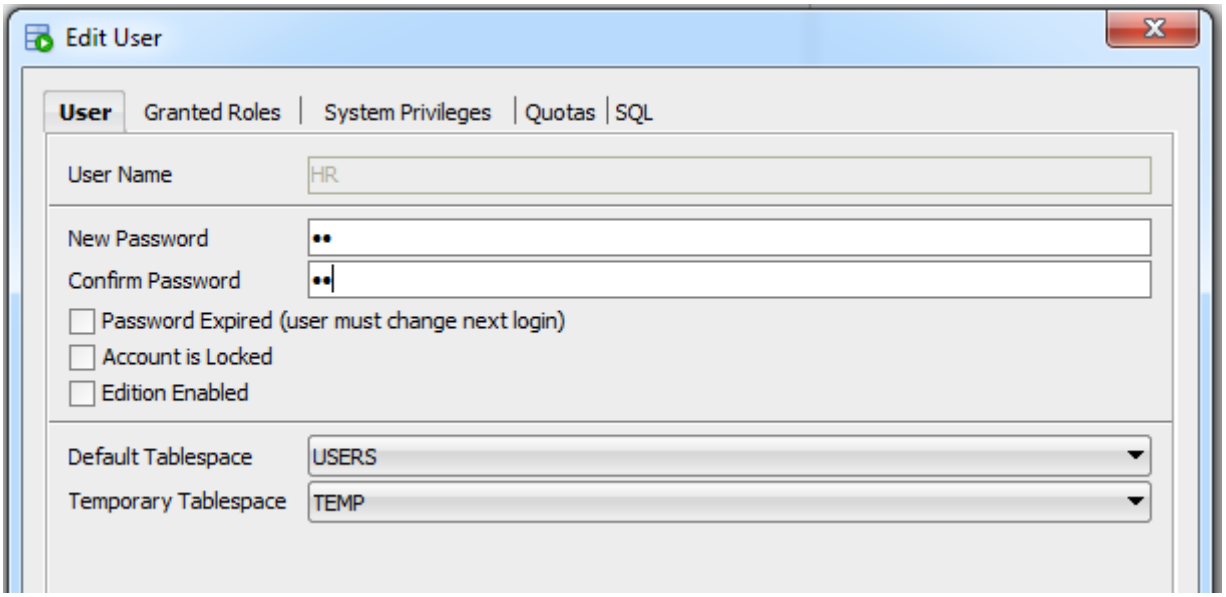

Рисунок 16

35. Должно появиться сообщение как на рисунке 17.

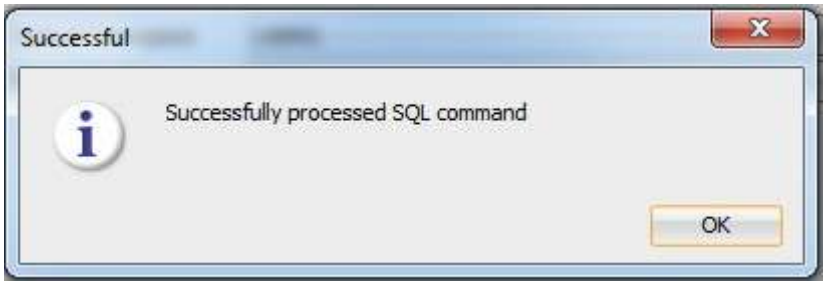

Рисунок 17

36. Создайте соединение: выберите **Connections**, **МП**, выберите **New connection**, в поле **Connection Name** введите **hr**, в поле **Name** введите **hr**, в поле **Password** введите **hr**, в поле **hostname** введите **localhost**, в поле **SID** введите **xe**, нажмите **Test**, если сообщения об ошибке нет, то нажмите **Connect**.

37. Просмотрите таблицы схемы hr: в левом окне, в дереве объектов выберите плюсик у Tables, должно быть семь таблиц.

38. Создайте пользователя: в окне дерева раскройте **system**, раскройте **Other Users, МП**, выберите **Create user**, в окне Edit User выберите вкладку **User**, в поле **User Name** введите **student**, в поле **New Password** введите **student**, в поле **Confirm password** введите **student**, выберите вкладку **Granted Roles**, галочкой выберите Granted **DBA**, нажмите **Принять**.

39. Создайте соединение: выберите **Connections**, **МП**, выберите **New connection**, в поле **Connection Name** введите **student**, в поле **Name** введите **student**, в поле **Password** введите **student**, в поле **hostname** введите **localhost**, в поле **SID** введите **xe**, нажмите **Test**, если сообщения об ошибке нет, то нажмите **Connect**.

40. Просмотрите таблицы схемы student: в левом окне, в дереве объектов выберите плюсик у Tables, должно быть пусто. Здесь можно создавать свои таблицы.

Литература:

1. Installing Oracle 11g XE (Express Edition) http://www.codeproject.com/Articles/254711/Installing-Oracle-g-XE-Express-Edition

2. Oracle 11gR2 on Windows 7 http://blog.mclaughlinsoftware.com/2011/12/29/oracle-11gr2-on-windows-7/

3. https://docs.oracle.com/cd/E17781_01/install.112/e18803.pdf

4.http://blog.mclaughlinsoftware.com/2011/09/13/oracle-11g-xe-install/

4-й день. Логическая модель

> Ничто не скрывается от взоров наблюдательного света. Новая связь графини стала скоро всем известна. Некоторые дамы изумлялись ее выбору, многим казался он очень естественным. Одни смеялись, другие видели с ее стороны непростительную неосторожность.
> *А. С. Пушкин. Арап Петра Великого, гл. I*

4.1. Описание базы данных

Разработка информационной системы начинается с исследования информационной среды, которую вы собираетесь моделировать. Откуда поступает информация и в каком виде? Как она будет вводиться и кто будет этим заниматься? Как часто она изменяется?

Итак, наша цель – это создание базы данных «Облака», которая будет содержать информацию о компании, занимающейся издательской деятельностью и имеющей три дочерних издательства. В базе данных будут представлены данные, которые могут потребоваться редакторам, менеджерам и другим сотрудникам компании – информация о книгах, их авторах, редакторах, о финансовом состоянии компании. На их основе можно получать разнообразные отчеты, например, о текущих продажах, можно узнать, какие редакторы работали с какими авторами и т.д.

Пользователи БД «Облака» могут задавать самые разные вопросы.

– Кто из авторов проживает в Енисейске?

– Какие книги стоят дороже 190 рублей?,

– Кто написал самое больное количество книг?

– Какова средняя стоимость книг по истории?

– Как продаются книги по информатике?

Реляционная модель не требует от вас описания всех возможных связей между данными, вы можете впоследствии задавать вопросы о любых логических связях, содержащихся в базе, а не толь«о о тех, которые планировались первоначально.

Начнем моделирование с определения бизнес-правил оказывающих влияние на данные.

База данных «Облака» должна учитывать следующие бизнес-правила:

автор может написать несколько книг;
книга может быть написана несколькими авторами;
порядок фамилий авторов на первой странице является важной информацией, так ка«влияет на получаемый ими гонорар;
редактор может работать над несколькими книгами, и в каждой книге может быть несколько редакторов;
в заказе на покупку может быть перечислено несколько книг.

1. Установите логический уровень модели: в главной области окна программы выберите вкладку **Logical.**

Логический уровень – это абстрактный взгляд на данные, когда данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Постоянный клиент», «Отдел». Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных может быть построена на основе другой логической модели, например на основе модели процессов. Логическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

4.2. Сущность

Основными понятиями логической модели являются: сущность (entity), атрибут (attribute) и связь (reiationship). На диаграмме сущности изображаются прямоугольниками, возможно соединенными между собой линиями (связями). Например, сущность Автор.

Информация не существует сама по себе, а обязательно связана с какой-то сущностью. Сущность – нечто существуемое и различимое. Можно сказать, что под термином сущность понимается все, что может быть представлено в БД.

Каждая сущность является множеством подобных индивидуальных объектов, называемых экземплярами. Каждый экземпляр сущности должен отличаться от всех остальных экземпляров этой же сущности. Примером экземпляра сущности Автор является Деменюк Андрей, проживающий по адресу Академгородок, 10».

Все экземпляры определенной сущности обладают общими свойствами или атрибутами. Например, атрибутами сущности Автор являются – Автор, Фамилия, Имя, Адрес.

С точки зрения физической модели сущности соответствует таблица, экземпляру сущности – строка в таблице, а атрибуту – столбец таблицы. Сущность должна иметь имя с четким смысловым значением, именоваться существительным в единственном числе, не носить технических наименований. Именование сущности в единственном числе облегчает чтение модели. Фактически имя сущности дается по имени ее экземпляра. Например, сущность Автор, а не Авторы и не AVT_356_TP.

4.3. Определение сущностей

Определим сущности базы данных «Облака»:

авторы, книги которых опубликованы компанией – сущность Автор;

сами книги, изданные компанией – сущность Книга;

редакторы, работающие на компанию – сущность Редактор;

издательства, которыми владеет компания – сущность Издательство;

заказы от книжных магазинов на покупку книг – сущность Заказ.

3. Создайте сущность: на палитре выберите инструмент **New Entity**, выберите мышью на диаграмме место размещения левого угла формы сущности и, не отпуская мышь, проведите диагональ до правого угла формы сущности, отпустите мышь, в появившемся окне свойств сущности выберите вкладку **General**, в поле **Name** введите **Автор**, в поле **Preferred Abbreviation** введите **avtor**, нажмите **ОК.**

4. Создайте сущность **Книга (kniga)** – см. **3.**

5. Создайте сущность **Редактор (redak)** – см. **3.**

6. Создайте сущность **Издательство (izdat)** – см. **3.**

7. Создайте сущность **Заказ (zakaz)** – см. **3.**

Созданные сущности должны быть как на рис. 3.

4.4. Связь

Мы имеем пять сущностей, информация о которых будет храниться в базе данных «Облака». Но еще не определены связи между данными. Например, пока непонятно, как связаны между собой книги и конкретные издательства.

Связь (relationship) – это функциональная зависимость между двумя сущностями (возможна связь сущности с самой собой). Если между некоторыми сущностями существует связь, то экземпляры одной сущности ссылаются или некоторым образом связаны с экземплярами другой сущности. Например, связь между издательствами, с одной стороны, и книгами, в производстве которых они принимают участие, с другой стороны.

На логическом уровне можно установить:

связь многие-ко-многим (M:N Relation)

неидентифицирующую связь один-.ко-многим (1:N Relation).

идентифицирующую связь один-ко-многим (1:N Identifying Relation),

связь один-к-одному (1:1 Relation).

иерархия супертип-подтип

Связь один-к-одному отображает такой характер связей между сущностями, когда каждому экземпляру одной сущности соответствует только один экземпляр другой, и наоборот. Например, муж женат только на одной жене, а жена, в свою очередь, состоит в браке только с одним мужем.

Наиболее часто в реляционной модели используется связь типа один-ко-многим. В таком случае один и только один экземпляр первой сущности связан со многими экземплярами второй сущности. Первая сущность называется родительской. Вторая сущность – дочерней. Например, связь между книгами и издательствами: каждая книга имеет единственного издателя, в то время как издательство может выпустить много книг.

Связь многие-ко-многим представляет ситуацию, где экземпляры одной сущности связаны с одним или большим количеством экземпляров второй сущности, и наоборот, экземпляры второй сущности связаны с одним и большим количество экземпляров первой сущности. Например, как связаны авторы конкретной книги? Каждый автор

может написать много книг, и каждая книга в свою очередь может быть написана несколькими авторами.

Различают зависимые и независимые сущности. Тип сущности определяется ее связью с другими сущностями. Экземпляры независимой сущности могут быть уникально идентифицированы без определения связей с другими сущностями. А зависимая сущность, наоборот, не может быть уникально идентифицирована без определения связей с другими сущностями.

Идентифицирующая связь устанавливается между независимой (родительский конец связи) и зависимой (дочерний конец связи) сущностями. Экземпляр зависимой сущности определяется только через отношение к родительской сущности.

При установлении неидентифицируемой связи дочерняя сущность остается независимой. То есть неидентифицирующая связь служит для связывания независимых сущностей.

Связь должна иметь ими, выражаемое глаголом глагольной фразой, например, «издательство выпускает книги». Глагол («выпускает») выражает некоторое ограничение, или бизнес-правило, и облегчает чтение диаграммы.

Сущность может быть расщеплена на два или большее число взаимно исключающих подтипов, каждый из которых включает общие атрибуты и/или связи. Эти общие атрибуты и/или связи явно определяются один раз на более высоком уровне. В подтипах могут определяться собственные атрибуты и/или связи. В принципе, подтипизация может продолжаться на более низких уровнях, но опыт показывает, что в большинстве случаев оказывается достаточно двух-трех уровней.

Тип сущности, на основе которого определяются подтипы, называется супертипом. Подтипы должны образовывать полное множество, т. е. любой экземпляр супертипа должен относиться к некоторому подтипу [17].

4.5. Определение связей

Определяем связи один-ко-многим:

«Издательство-Книга» – каждое издательство выпускает много книг, а каждая книга выпускается единственным издательством. Это связь – неидентифицирующая один-ко-многим, так как Из-

дательство и Книга независимые сущности. Экземпляр сущности Книга может существовать безотносительно к какому-либо экземпляру сущности Издательство, то есть в БД может существовать книга без указания издательства, которое ее выпустило.

Определяем связи многие-ко-многим:

«Автор-Книга» – некоторые авторы выпустили более одной книги, а некоторые книги написаны несколькими авторами.

«Редактор-Книга» – редактор может работать более чем над одной книгой, а у каждой книги может быть несколько редакторов.

«Заказ-Книга» – каждый заказ может включать различные книги, а каждая книга может быть включена в разные заказы.

8. Создайте связь один-.ко-многим: выберите инструмент **New 1:N Relation**, выберите мышью родительскую сущность **Издательство,** выберите мышью дочернюю сущность **Книга.**

9. Введите имя: выберите связь, **МП, Properties**, выберите вкладку **Cardinality,** в поле **Name on Source** введите **выпускает,** нажмите **Ладно**.

10. Выберите показ имен на диаграмме: Right click on Logical diagram and from context menu select Show->Labels.

11. Создайте связь многие-ко-многим: выберите инструмент **New M:N Relation**, выберите мышью сущность **Автор**, выберите мышью сущность **Книга**.

12. Введите имя связи **написал –** см. **9.**

13. Создайте связи многие-ко-многим: **Редактор-Книга**, **Заказ-Книга** – см. **11**.

14. Введите имена связей: **редактирует, включает –** см. **9**.

Иерархия наследования представляет собой особый тип объединения сущностей, которые разделяют общие характеристики. Например, авторов издаваемых книг можно разделить на современников и классиков. Из их общих свойств можно сформировать обобщенную сущность (родовой предок), супертип Автор, чтобы представить информацию, общую для всех типов авторов. Специфическая для каждого типа авторов информация будет расположена в подтипах (потомках) Современник и Классик.

Обычно иерархию наследования создают, когда несколько сущностей имеют общие по смыслу атрибуты или связи.

15. Создайте сущность **Современник (alive)** – см. **3**

16. Создайте сущность **Классик (classic)** – см. **3**

17. Определите супертип: выберите сущность **Современник**, **МП**, выберите **Properties,** в списке **Super Type** выберите **Автор**, нажмите **Ладно**.

18. Определите супертип: выберите сущность **Классик**, **МП**, выберите **Properties,** в списке **Super Type** выберите **Автор**, нажмите **Ладно**.

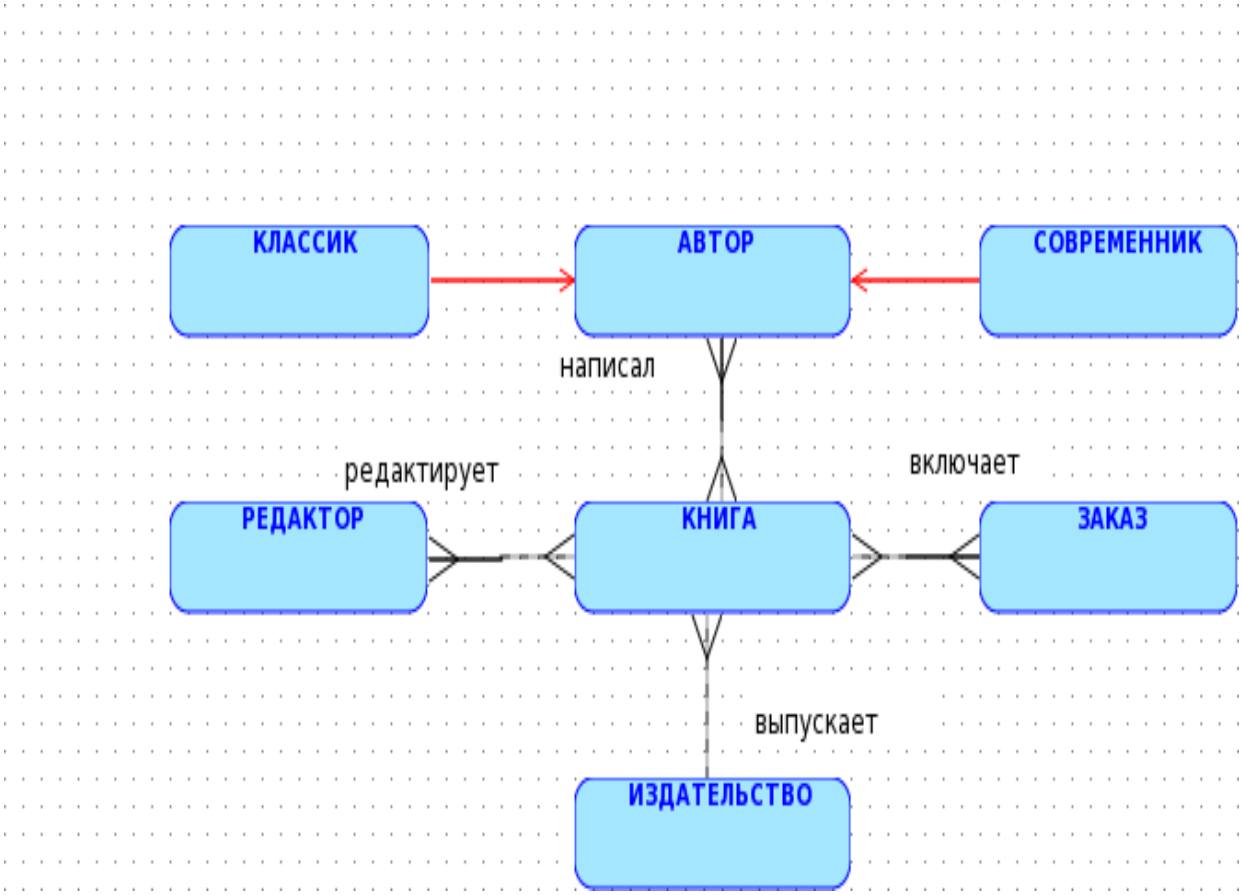

Рис. 3: Логическая модель в нотации Баркера

4.6. Определение доменов

На логическом уровне моделирования среда реализации информационной системы не учитывается. Вы просто определяете атрибут как строку, число или дату, в идеале можно назначить атрибуту соответствующий домен.

Домен можно определить как совокупность значений, из которых берутся значения атрибутов. Каждый атрибут может быть определен только на одном домене, но на каждом домене может быть определено множество атрибутов. В понятие домен входит не только тип данных, но и область значений домена. Например, можно определить домен Фамилия как строковые данные и определить атрибуты Фамилия автора и Фамилия редактора как принадлежащие этому домену.

19. Создайте домен: выберите **Tools, Domains Administration**, выберите **Add**, в поле **Name** введите **vio**, в списке **Logical type** выберите **varchar**, в поле **Size** введите **50**, нажмите **Apply**.

20. Введите остальные домены по табл. 2.

Имя	Тип	Ещё
vio	varchar	Size: 50
kol	varchar	Precision: 10, Scale: 0
zip	varchar	Size: 10
adress	varchar	Size: 50
city	varchar	Size: 30
tel	varchar	Size: 10
title	varchar	Size: 50
tema	varchar	Size: 30
price	numeric	Precision: 10, Scale: 2
rod	date	
numeric_id	numeric	Precision: 4, Scale: 0
book_id	varchar	Size: 20

Таблица 2: Домены

21. После создания доменов нажмите **Save,** нажмите **Close.**

Будет создан файл defaultdomains.xml в каталоге /datamodeler/domain/. Можно скопировать файл defaultdomains.xml в новое место и переименовать например в library_domains.xml. Этот файл можно использовать при последующих проектированиях.

4.7. Атрибут

Каждая сущность будет представлена на физическом уровне таблицей. Каждая из этих сущностей обладает собственными признаками. Каждый признак будет представлен на физическом уровне столбцом.

Экземпляры одной сущности имеют одинаковые наборы признаков (свойств). Например, для каждого автора имеется характеризующий его набор признаков: Фамилия, Имя, Отчество и т. п.

Атрибутом сущности является любой признак, который служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

Каждый признак сущности описывается ровно одним атрибутом в своей сущности. Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение.

Для большинства сущностей полный набор признаков неисчерпаем. Поэтому выбирают только те сведения о сущности, которые необходимы для решения данной практической задачи.

21. Определите атрибуты: выберите **Автор, МП,** выберите **Properties,** выберите вкладку **Attributes,** нажмите **плюс,** в поле **Name** введите **Автор#**, в области **Datatype** выберите **Domain,** в списке **Type** выберите **numeric_id,** нажмите **Apply**, выберите **плюс**, в поле **Name** введите **Фамилия**, в области **Datatype** выберите **Domain,** в списке **Type** выберите **vio,** выберите **Mandatory**, нажмите **Apply,** выберите **плюс,** в поле **Name** введите **Имя,** в области **Datatype** выберите **Domain,** в списке **Type** выберите **vio,** выберите **Mandatory**, нажмите **Apply,** выберите **плюс,** в поле **Name** введите **Отчество,** в области **Datatype** выберите **Domain,** в списке **Type** выберите **vio,** нажмите **Apply,** нажмите **Ладно**.

Имя	Тип данных	Ещё
Автор# (avtor)	Domain numeric_id	Primary UID, уникальный идентификатор автора
Фамилия (famil)	Domain vio	Mandatory
Имя (imja)	Domain vio	Mandatory
Отчество (otche)	Domain vio	
Позиция (pozic)	Logical type: NUMERIC (Precision=2, Scale= 0)	
Разделение гонорара (razde)	Logical type: NUMERIC (Precision=3, Scale= 1)	

Таблица 3: Атрибуты сущности Автор

22. Определите атрибуты сущности **Современник** по табл. 4.

23. Определите атрибуты сущности **Классик** по табл. 5.

Имя	Тип данных	Ещё
Адрес (adres)	Domain adress	
Индекс (Indek)	Domain zip	
Город (gorod)	Domain city	
Телефон (telef)	Domain tel	

Таблица 4: Атрибуты сущности Современник

Имя	Тип данных	Ещё
Дата смерти (smert)	Domain rod	

Таблица 5: Атрибуты сущности Классик

24. Определите атрибуты сущности **Книга** по табл. 6.

Имя	Тип данных	Ещё
Книга# (kniga_id)	Domain book_id	Primary UID, уникальный идентификатор книги
Название (nazva)	Domain title	Mandatory
Тема (tema)	Domain tema	
Цена (cena)	Domain price	
Расходы (rasho)	Domain price	
Дата выхода (vyhod)	Domain rod	

Таблица 6: Атрибуты сущности Книга

24. Определите атрибуты сущности **Редактор** по табл. 7.

Имя	Тип данных	Ещё
Редактор# (redak_id)	Domain numeric_id	Primary UID, уникальный идентификатор редактора
Фамилия (famil)	Domain vio	Mandatory
Имя (imja)	Domain vio	Mandatory
Отчество (otche)	Domain vio	
Адрес (adres)	Domain adress	
Индекс (Indek)	Domain zip	
Город (gorod)	Domain city	
Телефон (telef)	Domain tel	
Специализация (speci)	Domain tema	

Таблица 7: Атрибуты сущности Редактор

26. Определите атрибуты сущности **Заказ** по табл. 8.

Имя	Тип данных	Ещё
Заказ# (zakaz_id)	Domain numeric_id	Primary UID, уникальный идентификатор заказа
Магазин# (magaz)	Domain numeric_id	Mandatory
КоличествоЗаказано (kolza)	Domain kol	Mandatory
КоличествоОтправлено (kolot)	Domain kol	
Дата выполнения (vypol)	Domain rod	

Таблица 8: Атрибуты сущности Заказ

Name	Datatype	Other Information
Издательство# (izdat_id)	Domain numeric_id	Primary UID, уникальный идентификатор автора
Название (nazva)	Domain title	Mandatory
Адрес (adres)	Domain adress	
Индекс (indek)	Domain zip	
Город (gorod)	Domain city	
Телефон (telef)	Domain tel	

Таблица 9: Атрибуты сущности Издательство

27. Определите атрибуты сущности **Издательство** по табл. 9.

Логическая модель после этого шага должна быть как на рис. 4 или 5.

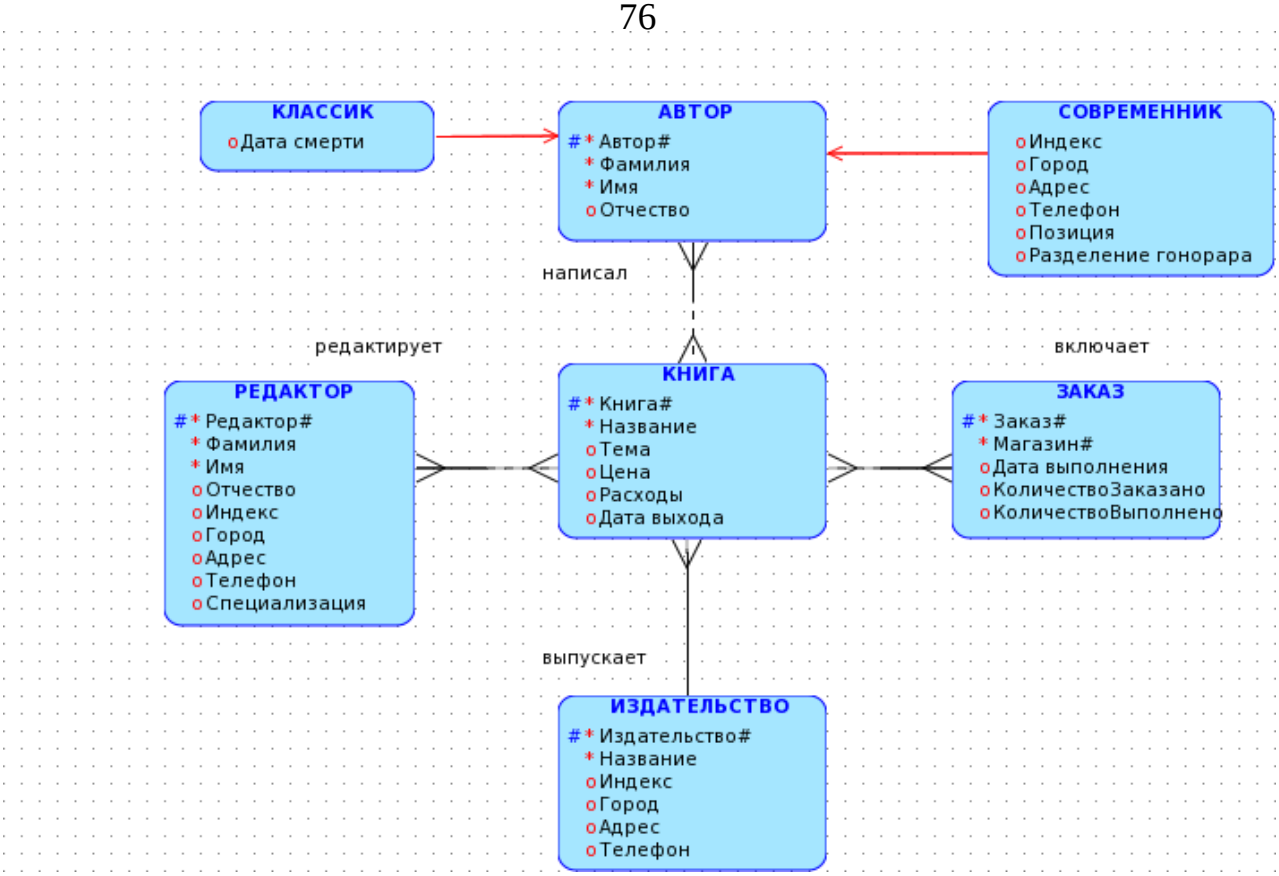

Рис. 4: Атрибуты логической модели в нотации Баркера

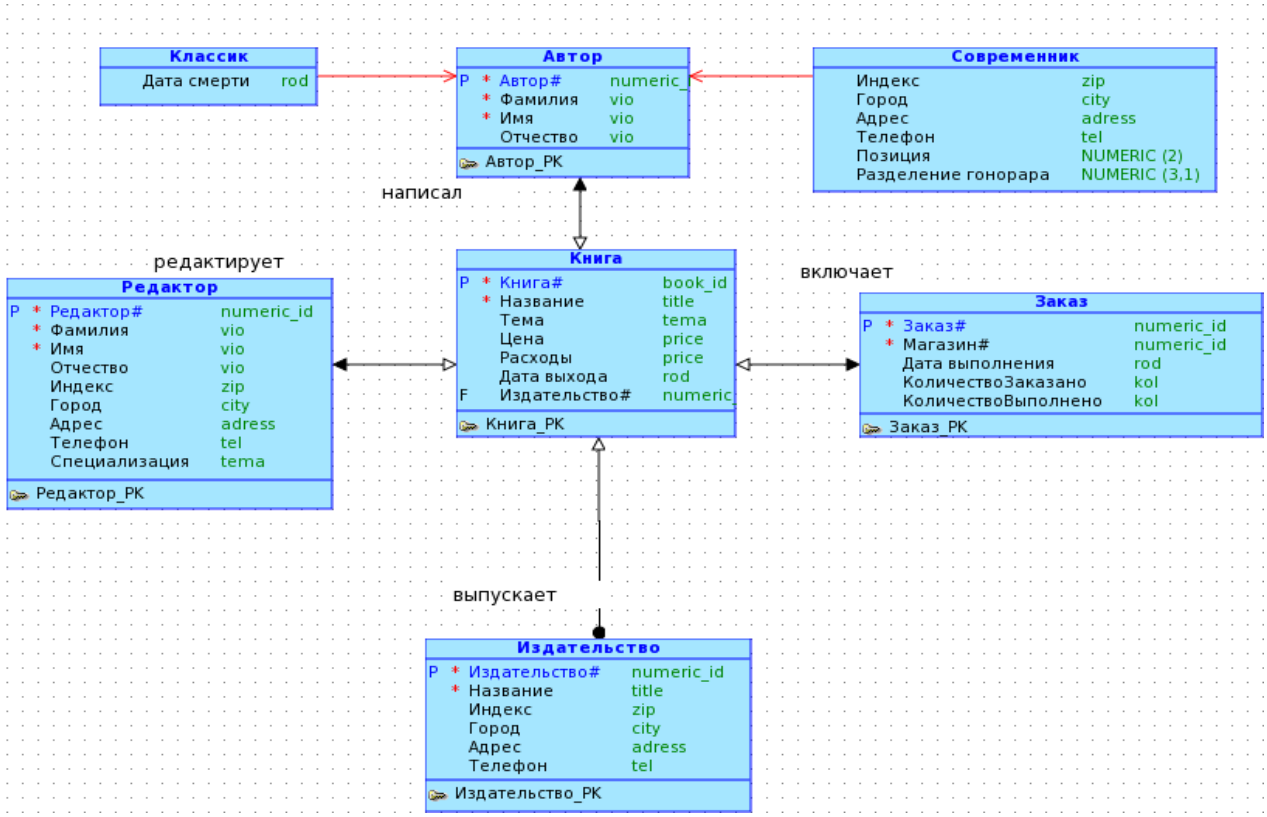

Рис. 5: Атрибуты логической модели в нотации Бахмана

4.8. Определение первичных ключей

Барон:
...Стоять я не могу... мои колени
Слабеют... душно! .. душно! ..
Где ключи? Ключи, ключи мои! ..
А. С. Пушкин. Скупой рыцарь, сцена III

Каждая сущность должна иметь признаки, которые ее описывают, в противном случае она просто не может существовать. Признак сущности, представляющий интерес называется атрибутом. Некоторые атрибуты сущности не только описывают, но и уникальным образом идентифицируют ее. Их называют первичным ключом. Первичный ключ (primary key) – это атрибут или группа атрибутов, однозначно идентифицирующая экземпляр сущности.

Если в сущности своего первичного ключа нет – следует создать дополнительный атрибут. Примером такого атрибута являются код автора, номер зачетной книжки студента. Такие ключи называют суррогатными, т.е. вне базы данных они не имеют

Выбор первичного ключа может оказаться непростой задачей, решение которой может повлиять на эффективность будущей БД. В сущности могут оказаться несколько атрибутов, претендующих на роль первичного ключа. Такие претенденты называются потенциальными ключами.

Рассмотрим кандидатов на первичный ключ сущности Автор. Здесь можно выделить следующие потенциальные ключи: «Автор#», «Номер паспорта», «Фамилия Имя Отчество» (составной).

Ключи могут быть составными, то есть содержащими несколько атрибутов.

Для того чтобы быть первичным, потенциальный ключ должен удовлетворять ряду требований:

Уникальность

Два экземпляра не должны иметь одинаковых значений возможного ключа. Потенциальный ключ «Фамилия Имя Отчество» является плохим кандидатом, так как авторы могут быть полными тезками. Экземпляры же ключей «Автор#» и «Номер паспорта» являются уникальными.

Компактность

Для обеспечения уникальности дополним ключ «Фамилия Имя Отчество» атрибутом «Адрес». Но при выборе первичного ключа предпочтение должно отдаваться более простым ключам, то есть ключам, содержащим меньшее количество атрибутов. Тогда ключи «Автор» и «Номер паспорта» предпочтительнее составного ключа «Фамилия Имя Отчество Адрес».

Атрибуты не должны содержать null-значений.

Если допускается, что автор может, например, не иметь паспорта (например, у Федора Михайловича Достоевского какой номер паспорта?), то ключ «Номер паспорта» не подойдет на роль первичного ключа. Если для обеспечения уникальности необходимо дополнить потенциальный ключ дополнительными атрибутами, то они не должны содержать нулевых значений. Дополняя ключ «Фамилия Ими Отчество» атрибутом «Адрес», нужно убедиться в том, что адреса всех авторов известны.

Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности.

Автор может выйти замуж и сменить фамилию, сменить паспорт. Поэтому «Номер паспорта» и «Фамилия Имя Отчество» не подходят по этому условию на роль первичного ключа.

Итак, наиболее подходящим первичным ключом сущности Автор является суррогатный атрибут «Автор#».

Каждая сущность должна иметь по крайней мере один потенциальный ключ. Многие сущности имеют только один потенциальный ключ. Такой ключ становится первичным. Но если сущность имеет более одного возможного ключа, то один из них становится первичным, а остальные – альтернативными, ключами. Альтернативный ключ – это потенциальный ключ, не ставший первичным.

Внешний ключ (Foreign key) – это атрибут или группа атрибутов одной сущности, которые могут служить в качестве первичного ключа для другой сущности. Говорят также, что внешний ключ одной сущности является ссылкой на первичный ключ другой сущности.

Ссылочная целостность реляционной модели требует, чтобы внешний ключ либо имел значение null, либо соответствовал значению первичного ключа, на которое он ссылается.

При установлении идентифицирующий связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав в первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов.

При установлении неидентифицируемой связи дочерняя сущность остается независимой, а атрибуты первичного ключа родительской сущности мигрируют в состав неключевых компонентов родительской сущности.

28. Задайте первичный ключ: выберите сущность **Автор, МП,** выберите **Properties,** выберите вкладку **Attributes,** в области **Attributes** выберите **Автор#**, выберите флажком **Primary UID**, нажмите **Ладно**.

29. Задайте первичный ключ: выберите сущность **Книга, МП,** выберите **Properties,** выберите вкладку **Attributes,** в области **Attributes** выберите **Книга#**, выберите флажком **Primary UID**, нажмите **Ладно**.

30. Задайте первичный ключ: выберите сущность **Издательство, МП,** выберите **Properties,** выберите вкладку **Attributes,** в области **Attributes** выберите **Издательство#**, выберите флажком **Primary UID**, нажмите **Ладно**.

Обратите внимание, что атрибут первичного ключа Издательство# родительской сущности Издательство должны автоматически мигрировать в дочернюю сущность Книга.

31. Задайте первичный ключ: выберите сущность З**аказ, МП,** выберите **Properties,** выберите вкладку **Attributes,** в области **Attributes** выберите **Заказ#**, выберите флажком **Primary UID**, нажмите **Ладно**.

32. Задайте первичный ключ: выберите сущность **Редактор, МП,** выберите **Properties,** выберите вкладку **Attributes,** в области **Attributes** выберите **Редактор#**, выберите флажком **Primary UID**, нажмите **Ладно**.

4.10. Нотация Баркера

33. Выберите нотацию модели: выберите **View, Logical Diagram Notation**, выберите **Barker Notation**.

Сущность в нотации Баркера представляется прямоугольником любого размера, содержащим внутри себя:

имя сущности;

список имен атрибутов (знак **o** перед именем атрибута);

указатели ключевых атрибутов (знак # перед именем атрибута).;
указатели обязательных атрибутов (знак * перед именем атрибута).

Все связи являются бинарными и представляются линиями с двумя концами (соединяющими сущности), для которых должно быть определено:

имя связи;
степень множественности (один или много объектов участвуют в связи: « 0,1» , « 0,*» , «1» , «1,*»
степень обязательности: обязательная (Mandatory) или необязательная (Optional) связь между сущностями.

Если максимальное кардинальное число отображения равно бесконечности, то линия, представляющая связь, разветвляется, принимая вид «вороньей лапки» и примыкает к прямоугольнику, соответствующему множеству сущностей – области значений, в трех точках; в противном случае линия остается без изменений.

Если минимальное кардинальное число отображения равно нулю, то часть линии-связи, примыкающая к прямоугольнику, изображается при помощи пунктирной линии, в противном случае используется сплошная линия.

Нотация применяется к обоим отображениям, определяемым связью, при этом линия, изображающая связь, делится пополам, и каждая ее часть оформляется в соответствии со значениями кардинальных чисел.

Для указания степени множественности равной 0 в свойствах связи должна быть включена Target optional.

Примеры нотации Баркера на рис. 6, 7, 8, 9.

Идентифицирующая связь изображается поперечной линий на стороне вороньей лапки (рис. 13).

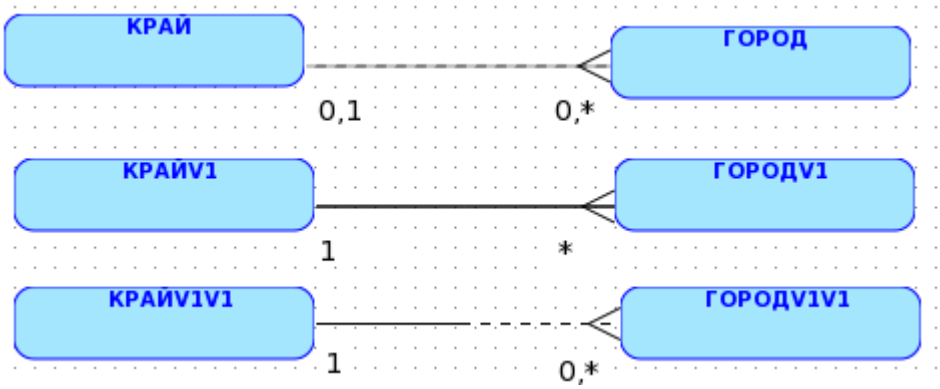

Рис. 6 Связь один-ко-многим в нотации Баркера

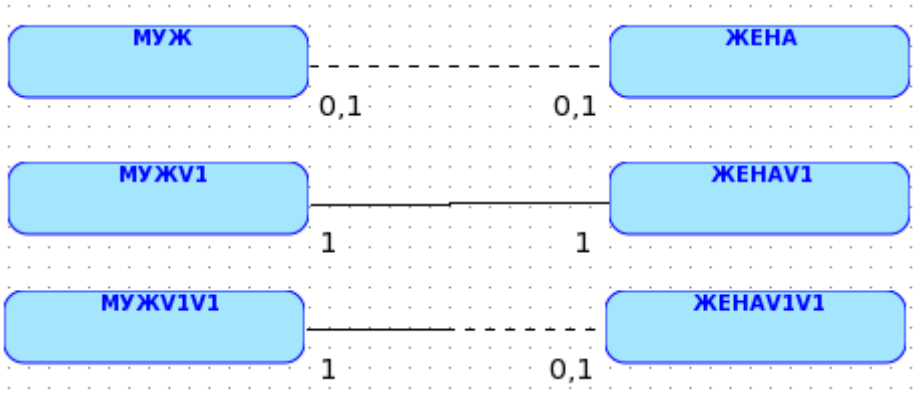

Рис. 7 Связь один-к-одному в нотации Баркера

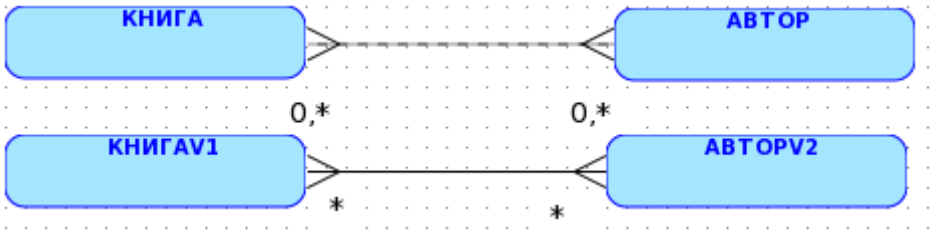

Рис. 8 Связь многие-ко-многим в нотации Баркера

Идентифицирующая связь изображается поперечной линий на стороне вороньей лапки (рис. 9).

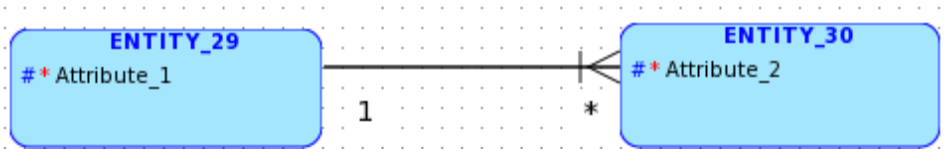

Рис. 9 Идентифицирующая связь в нотации Баркера

4.11. Нотация Бахмана

34. Выберите нотацию модели: выберите **View, Logical Diagram Notation**, выберите **Bachman Notation**.

Сущность представляется прямоугольником любого размера, содержащим внутри себя:

имя сущности;

список имен атрибутов;

указатели ключевых атрибутов (знак P перед именем атрибута);

указатели обязательных атрибутов (знак * перед именем атрибута);

указатели внешних ключей (знак F перед именем атрибута).

Все связи являются бинарными и представляются линиями с двумя концами (соединяющими сущности), для которых должно быть определено

имя;

степень множественности (один или много объектов участвуют в связи, « 0,1» , « 0,*» , « 1» , « 1,*»

Для множественной связи линия присоединяется к прямоугольнику наконечником, а для одиночной связи – кружком. То есть дочерняя сущность обозначается наконечником со стороны линии, а родительская кружком.

Допустимость Null-значений обозначается пустым кружком и пустым наконечником. Недопустимость Null-значений обозначается черным кружком и черным наконечником.

Идентифицирующя связь изображается поперечной линий на стороне наконечника.

Для указания степени множественности 0 в свойствах связи должна быть включена Target optional.

Примеры нотации Бахмана на рис. 10, 11, 12.

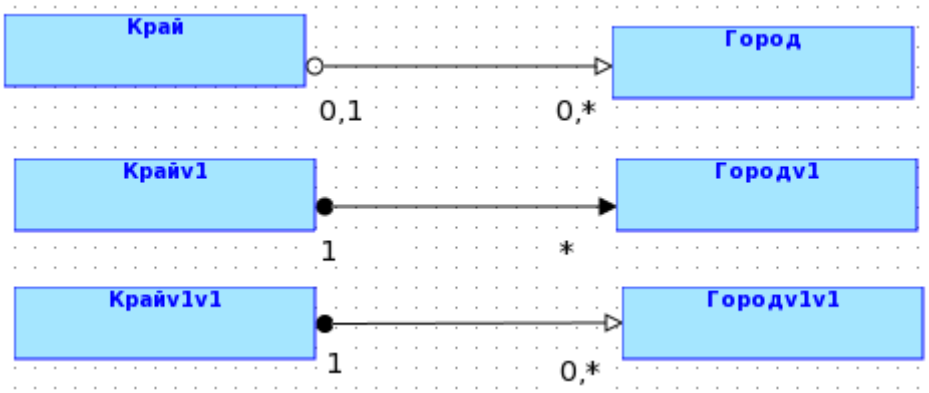

Рис. 10 Связь один-ко-многим в нотации Бахмана

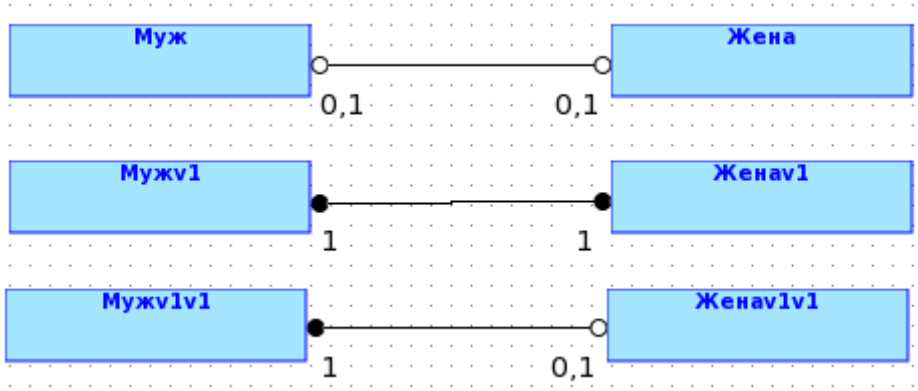

Рис. 11 Связь один-к-одному в нотации Бахмана

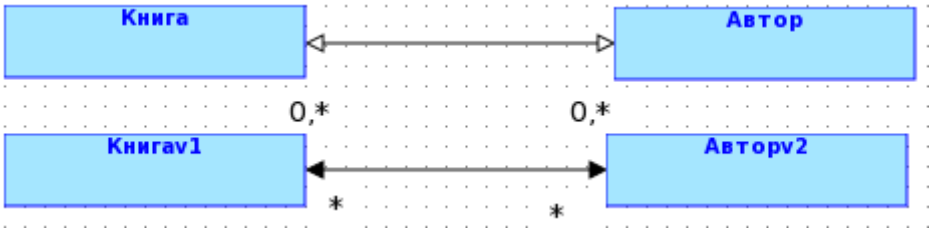

Рис. 12 Связь многие-ко-многим в нотации Бахмана

4.12. Настройки

Data Modeler имеет следующие возможности по настройке моделей:

Для изменения настроек форматирования по умолчанию: выберите Tools, General Options, Diagram, Format, выберите объект.

Изменение общих настроек модели: выберите Tools, General Options, Diagram, имеются следующие параметры: Show Grid – вывод сетки, Diagram Color – управление цветом фона диаграм-

мы, Line Auto Route – управление функцией автоматического маршрута, который включен по умолчанию.

Для изменения количества отображаемых деталей на модели: выберите View, View Details, имеются следующие параметры: All Details, Names Only, Attributes, Datatype, Keys.

Регулировка ширины и высоты объектов модели: выберите сущности предназначенные для регулирования, выберите Edit, Equal Height.

Для ручного управления линиями: выберите МП, отключите Auto Route.

Для управления линиями: выберите линию, МП, имеются следующие параметры: Add Elbow – добавление колена, которое позволяет перенаправлять линию вокруг других форм, Remove Elbow – удаление колена, Straighten Lines – выравнивание линии, удаляя при этом колена.

4.13. Замена связей многие ко-многим

Связь многие-ко-многим возможна только на уровне логической модели. При переходе к реляционной модели Data Modeler автоматически преобразовывает связь многие-ко-многим, добавляя новую таблицу и устанавливая две новые связи один-ко-многим от старой таблице к новой. Объект, который создается в результате такого преобразования связи, называют таблицей пересечения.

Можно сделать эту замену связи вручную, еще на уровне логической модели.

36. Удалите связь между сущностями **Автор** и **Книга.**

37. Создайте сущность **КнигаАвтор (bookavtor)** – см. **3.**

38. Создайте идентифицирующую связь один-.ко-многим**:** выберите инструмент **New 1:N Identifying Relation,** выберите мышью родительскую сущность **Книга,** выберите мышью дочернюю сущность **КнигаАвтор.**

39. Создайте идентифицирующую связь один-.ко-многим**:** выберите инструмент **New 1:N Identifying Relation,** выберите мышью родительскую сущность **Автор,** выберите мышью дочернюю сущность **КнигаАвтор.**

40. Удалите связь между сущностями Заказ и Книга.

41. Создайте сущность **ПунктЗаказа (punkt)** – см. **3.**

42. Создайте идентифицирующую связь один-.ко-многим**:** выберите инструмент **New 1:N Identifying Relation,** выберите мышью родительскую сущность **Книга,** выберите мышью дочернюю сущность **ПунктЗаказа.**

43. Создайте идентифицирующую связь один-.ко-многим**:** выберите инструмент **New 1:N Identifying Relation,** выберите мышью родительскую сущность **Заказ,** выберите мышью дочернюю сущность **ПунктЗаказа.**

Логическая модель после замены связей многие-ко-многим должна быть как на рис. 13.

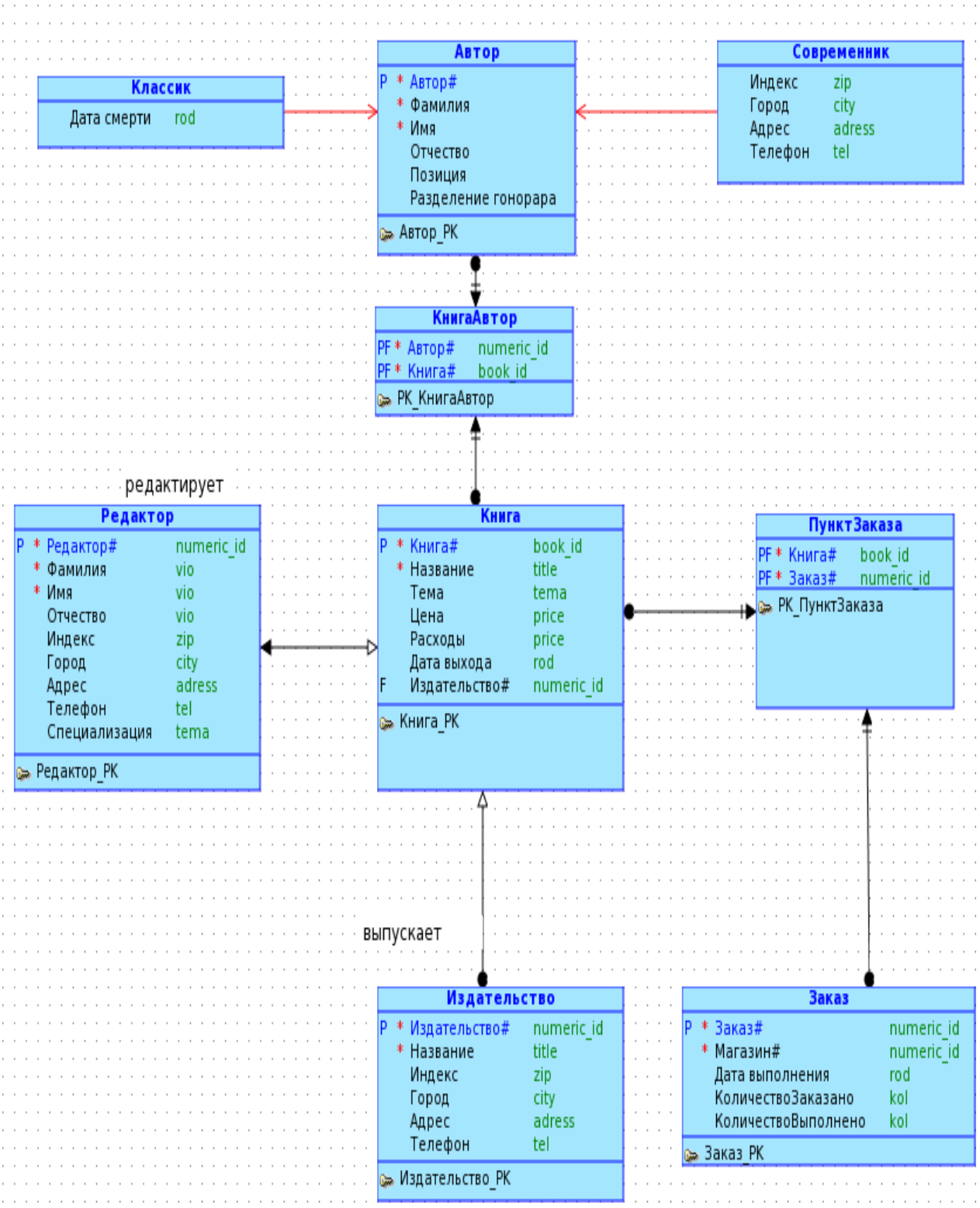

Рис. 13: Логическая модель в нотации Бахмана

5-й день. Нормализация

Бог веселый винограда
Позволяет нам три чаши
Выпивать в пиру вечернем.
А. С. Пушкин. Бог веселый винограда

5.1. Нормализация

Нормализация представляет собой процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных.

В основе нормализации лежит определенный математический аппарат, базирующийся на понятии «функциональной зависимости».

> Атрибут Y сущности R функционально зависит от атрибута X сущности R (символически записывается как X → Y), тогда и только тогда, когда каждое значение X в сущности R связано в точности с одним значением Y в сущности R. Иными словами, X уникально определяет Y.

Наиболее очевидным примером здесь может быть первичный ключ сущности, который однозначно определяет каждый экземпляр этой сущности. Однако могут существовать и другие зависимости, в которые не входят первичные ключи. Главная цель нормализации – избавить сущность от зависимостей, не связанных с первичными ключами.

Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам – формализованным требованиям к организации данным. Нормальные формы изменяются в порядке от первой (1НФ) до пятой (5НФ). Каждая последующая форма удовлетворяет требования предыдущей. На практике обычно ограничиваются приведением данных к третьей нормальной форме.

> Сущность находится в третьей нормальной форме (ЗНФ) тогда и только тогда, когда неключевые атрибуты (если они есть вообще) являются взаимно независимым и полностью зависимыми от первичного ключа.
>
> Два или несколько атрибутов называются взаимно независимыми, если ни один из них не зависит функционально от какой-

либо комбинации остальных атрибутов. Подобная независимость подразумевает, что каждый такой атрибут может быть обновлен независимо от остальных.

Атрибут полностью зависит от первичного ключа, когда он функционально зависит от полного первичного ключа, а не от его отдельных компонентов (атрибутов).

В пользу нормализации обычно приводят следующие три основных довода [9]:

Обеспечение целостности. Одного этого довода вполне достаточно, чтобы заботиться о нормализации. Данные сохраняют корректность и достоверность, поскольку в результате нормализации они будут сохраняться только в одном месте. Другими словами, нормализация должна привести к исключению избыточности данных. В противном случае придется следить за синхронными изменениями всех копий данных, что создает много проблем.

Нормализация обычно приводит к разделению сущности на две или более связанных сущности, которые могут быть соединены вместе с помощью операции объединения. Результатом разделения и является уменьшение избыточности данных.

Создание формальной модели, как можно более независимой от специфики приложения. Другими словами, нормализация способствует тому, что модель опирается на данные, а не на процессы их обработки. На практике это означает, что структура БД остается неизменной даже при изменении процессов обработки.

Снижение требований к объему памяти. Непосредственным следствием этого является повышение скорости поиска данных. Если не обращать внимания на внешние ключи, то полная нормализация приводит к исключению избыточности данных. А избыточность всегда связана с дополнительными объемами памяти для хранения избыточных данных.

5.2. Первая нормальная форма

Сущность находится в первой нормальной форме (1НФ) тогда и только тогда, когда все атрибуты содержат только атомарные значения.

Если значения атомарные, то на пересечении столбца и строки всегда находится только одно значение, а не набор значений.

Особенности 1НФ:

каждая сущность имеет первичный ключ;

повторяющиеся группы данных выделены в самостоятельные сущности.

Примеры нарушений 1НФ рассмотрены в 5.6 и 5.7.

5.3. Вторая нормальная форма

Сущность находится во второй нормальной форме (2НФ) тогда и только тогда, когда она находится в 1НФ.и каждый неключевой атрибут полностью зависит от первичного ключа.

Вторая нормальная форма имеет смысл только для сущностей, имеющих составной первичный ключ.

Особенности 2НФ:

сущность представлена в 1НФ;

если сущность имеет простой первичный ключ, то она представлена в 2НФ;

если сущность имеет составной первичный ключ, то каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа).

Пример нарушения 2НФ рассмотрен в 5.8.

5.4. Третья нормальная форма

Сущность находится в третьей нормальной форме (3НФ) тогда и только тогда, когда она находится в 2НФ, и каждый неключевой атрибут не зависит от другого неключевого атрибута.

Особенности 3НФ:

сущность представлена в 1НФ,

любой неключевой атрибут не зависит от другого неключевого атрибута.

Примеры нарушений 3НФ рассмотрены в 5.9 и 5.10.

5.5. Общие проблемы

Нормальные формы помогают проектировать базы данных, в которых нет ненужных избыточных данных и противоречий, которые могут повлечь за собой проблемы производительности или потерю информации при последующем выполнении операции вставки, обновления и удаления. Можно сказать, что нормальные формы позволяют

избежать искажения данных путем создания ложных данных или разрушения истинных.

Многие общие проблемы проектирования – это результат нарушения одной из нормальных форм. Общие проблемы обычно включают [11]:

повторяющиеся группы данных;
многократное использование одного и того же атрибута;
множественное местонахождение одного и того же факта;
конфликтующие факты;
производные атрибуты.

5.6. Повторяющиеся группы данных

Повторяющиеся группы данных могут быть определены как списки, повторяющиеся элементы, как внутренняя структура внутри атрибута. Эта структура нарушает 1НФ и поэтому должна быть удалена из модели.

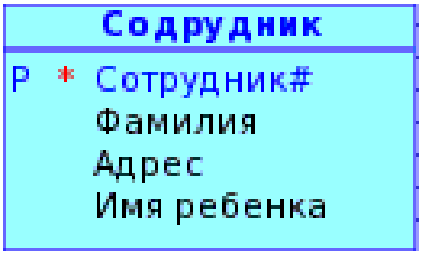

Рис. 14: Сотрудник

Сотрудник#	Фамилия	Адрес	Имя ребенка
E1	Седов	Сурикова, 10	Коля
E2	Куликов	Мира, 23	Маша, Оля

Таблица 10: Экземпляры сущности Сотрудник

Сущность «Сотрудник» содержит повторяющуюся группу данных в атрибуте «Имя ребенка» (рис. 14). Это нарушает 1НФ, которая говорит, что сущность находится в 1НФ, если каждый из атрибутов имеет не больше чем одно значение для каждого экземпляра. Экземпляр атрибута «Имя ребенка» имеет два значения: Маша и Оля (табл. 10).

Это нарушение 1НФ приводит к проблемам. Запись значения атрибута через запятую «Маша, Оля» приводит к тому, что размера столбца может не хватить для хранения данных (ведь сотрудник может

иметь и 4, и 5 детей). Проблема и в том, что по такому атрибуту невозможно построить индекс.

Для приведения сущности «Сотрудник» к 1НФ:

создадим новую сущность «Ребенок»,

перенесем в нее повторяющийся атрибут «Имя ребенка»,

определим атрибут «Ребенок#» в качестве первичного ключа сущности «Ребенок»,

установим идентифицирующую связь от сущности «Сотрудник» к сущности «Ребенок». Первичный ключ «Сотрудник#» сущности «Сотрудник» будет в сущности «Ребенок» внешним ключом.

Результат приведения сущности «Сотрудник» к 1НФ на рис. 15.

Рис. 15: Сотрудник

Сотрудник#	Фамилия	Адрес
E1	Седов	Сурикова, 10
E2	Куликов	Мира, 23

Таблица 11: Экземпляры сущности Сотрудник

Сотрудник#	Ребенок#	Имя ребенка
E1	C1	Коля
E2	C1	Маша
E2	C2	Оля

Таблица 12: Экземпляры сущности Ребенок

5.7. Многократное использование атрибута

Другая проблема – это хранение в одном атрибуте разных по смыслу значений, что нарушает 1НФ.

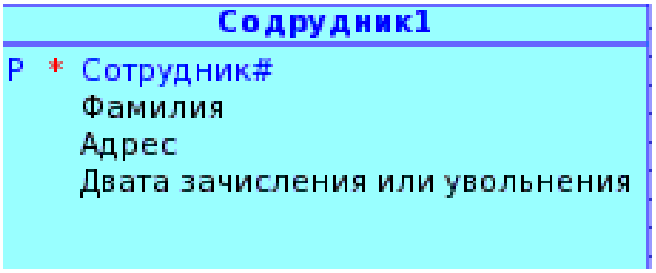

Рис. 16: Сотрудник1

Сотрудник#	Фамилия	Адрес	Дата зачисления или увольнения
El	Седов	Сурикова, 10	01.07.00
E2	Куликов	Мира, 23	07.08.00

Таблица 13: Экземпляры сущности Сотрудник1

Атрибут «Дата зачисления или увольнения» (табл. 13) хранит информацию как о зачислении, так и об увольнении сотрудника. Если хранится только одно значение (например, 01.07.00), то невозможно понять, какая именно дата внесена. Решение проблемы в том, чтобы позволить отдельным атрибутам хранить отдельные факты.

Для приведения сущности «Сотрудник1» к 1НФ: разделим сложный атрибут «Дата зачисления или увольнения» на атомарные атрибуты – «Дата зачисления» и «Дата увольнения». Результат приведения сущности «Сотрудник1» к 1НФ в табл. 14.

Сотрудник#	Фамилия	Адрес	Дата зачисления	Дата увольнения
E1	Седов	Сурикова, 10	01.07.00	
E2	Куликов	Мира, 23	07.08.00	01.09.00

Таблица 14: Экземпляры сущности Сотрудник1

5.8. Множественное местонахождение одного и того же факта

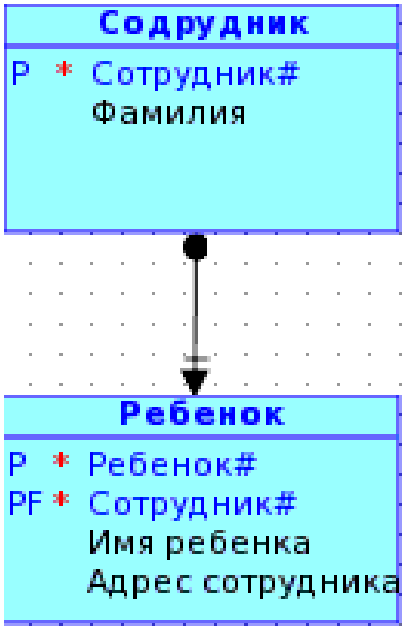

Рис. 17: Ребенок

Сотрудник#	Ребенок#	Имя ребенка	Адрес сотрудника
E1	C1	Коля	Сурикова, 10
E2	C1	Маша	Мира, 23
E2	C2	Оля	Мира, 23

Таблица 15: Экземпляры сущности Ребенок

Одна из целей БД заключается в максимальной поддержке целостности данных, чтобы таким образом гарантировать, что информация, содержащая в БД, правильна, и факты «внутри» БД не находятся в противоречии. Поэтому важно представить каждый факт в БД один и только один раз! Если факт появляется в двух (или более) местах, ошибки могут «проникнуть» в данные. Единственное исключение из

этого правила (один факт в одном месте) – в случае атрибутов первичных ключей.

Включение атрибута «Адрес сотрудника» в сущность «Ребенок» – это ошибка (табл. 15). Теперь, если сотрудник имеет несколько детей, то его адрес будет храниться для каждого ребенка.

Атрибут «Адрес сотрудника» – признак сущности «Сотрудник», но не сущности «Ребенок». Фактически, это нарушение 2НФ, которая говорит: при наличии составного первичного ключа все атрибуты должны зависеть от полного ключа, а не от его частей. Адрес сотрудника не зависит от полного первичного ключа «Сотрудник# Ребенок#», а зависит от его части – атрибута «Сотрудник#».

Для приведения сущности «Ребенок» к 2НФ:

перенесем атрибут «Адрес сотрудника» в сущность «Сотрудник».

Результат приведения сущности «Ребенок» к 2НФ на рис. 15.

5.9. Конфликтующие факты

Конфликты фактов могут происходить по разным причинам, включая нарушение первой, второй или третьей нормальных форм.

Рис. 18: Ребенок

Сотрудник#	Ребенок#	Имя ребенка	Адрес жены
E1	C1	Коля	Сурикова,10

E2	Cl	Маша	Мира, 23
E2	C2	Оля	Ботаническая, 24

Таблица 16: Экземпляры сущности Ребенок

Атрибут «Адрес жены» включен в сущность «Ребенок» (рис. 18), что привело к нарушению 2НФ. Данные на табл. 16 показывают, что сотрудник E2 имеет двух детей – Машу и Олю и два различных адреса жены. Возможно, E2 имеет две жены (одна живет на Мира, 23, другая на Ботанической, 24), или у Оли и Маши разные мамы. Или E2 имеет одну жену, проживающую по разным адресам. Какой вариант является правильным? Проблема в том, что выбрать правильный вариант из имеющихся данных невозможно.

Для приведения к 2НФ:

создадим новую сущность «Жена»,

поместим атрибут «Адрес жены» в сущность «Жена»,

установим идентифицирующую связь от сущности «Сотрудник» к сущности «Жена».

Результат приведения сущности «Ребенок» к 2НФ на рис. 19. Можно увидеть, что сотрудник E2 имеет двух жен, одну нынешнюю (текущая) и одну бывшую (табл. 18).

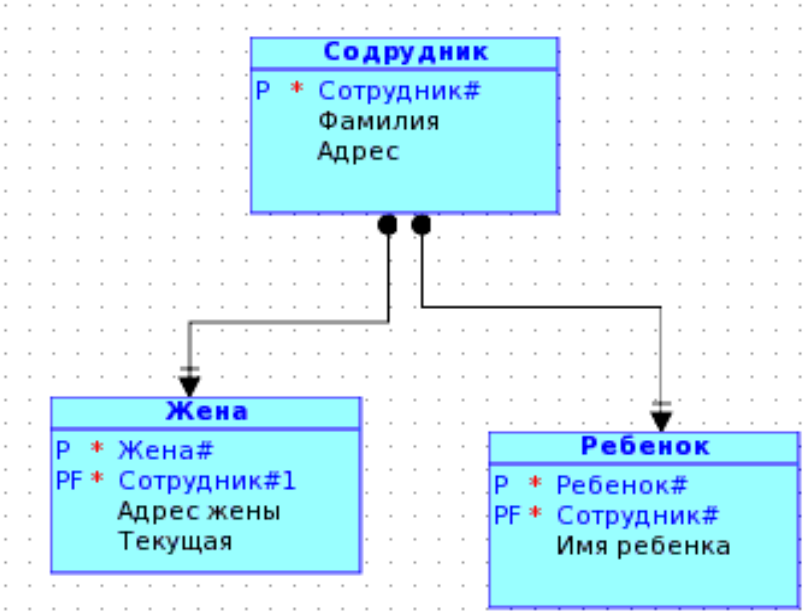

Рис. 19: Ребенок

Сотрудник#	Ребенок#	Имя ребенка
E1	C1	Коля
E2	Cl	Маша
E2	C2	Оля

Таблица 17: Экземпляры сущности Сотрудник

Сотрудник#	Жена#	Адрес жены	Текущая
E1	S1	Сурикова,10	Да
E2	Sl	Мира, 23	Да
E2	S2	Ботаническая, 24	Нет

Таблица 18: Экземпляры сущности Жена

5.10. Производные атрибуты

Другой пример конфликта фактов происходит, когда нарушена 3НФ.

Если включить атрибуты «Дата рождения» и «Возраст» в сущность «Ребенок» (см. табл. 19), то будет нарушена 3НФ, так как атрибут «Возраст» функционально зависит от атрибута «Даты рождения». Зная дату рождения, всегда можно получить возраст ребенка.

Производные атрибуты – это атрибуты, которые могут быть вычислены из других атрибутов (например, «Возраст»), и следовательно, могут не сохраняться в БД. Чтобы быть точными, производные атрибуты должны модифицироваться каждый раз, когда источник их образования модифицируется. Это создает большие непроизводительные затраты в приложении, которое делает модификации.

Сотрудник#	Ребенок#	Имя ребенка	Дата рождения	Возраст
E1	C1	Коля	01.09.00	10
E2	C1	Маша	01.08.98	12
E2	C2	Оля	02.07.05	5

Таблица 19: Экземпляры сущности Ребенок

Цель нормализации состоит в том, чтобы гарантировать, что имеется только один способ знать каждый факт, записанный в БД. Если

же вы можете получить ответ двумя разными способами, возможно, что что два ответа будут различны.

Сущность «Ребенок» имеет атрибуты «Дата рождения» и «Возраст». Предположим, что атрибут «Возраст» модифицируется в конце месяца. Затем вы задаете запрос: какого возраста ребенок? Вы можете непосредственно обратиться к атрибуту «Возраст» и получит ответ. Также вы можете вычесть «Дату рождения» от сегодняшней даты и тоже получить ответ. Если «Возраст» не модифицировался недавно, то вы получите неправильный ответ. То есть при наличии в БД производных атрибутов, в ней всегда имеется потенциал для конфликтующих ответов.

5.11. Приведение к 1НФ

Сущности **Автор, Книга, Издательство, Редактор, КнигаАвтор, ПунктЗаказа** соответствуют 1НФ так как

имеют первичные ключи,

не имеют повторяющихся групп.

Сущность **Заказ** не соответствует 1НФ так как имеет повторяющуюся группу атрибутов **Название, КоличествоЗаказано, КоличествоОтправлено**.

44. Для приведения к 1НФ перенесем повторяющиеся атрибуты **Название, КоличествоЗаказано, КоличествоОтправлено** в сущность **ПунктЗаказа**.

5.12. Приведение к 2НФ

Сущности **Автор, Книга, Издательство, Редактор, Заказ** соответствуют 2НФ так как

представлены в 1НФ,

имеют простой первичный ключ.

Сущность **КнигаАвтор** соответствуют 2НФ так как

представлена в 1НФ

имеют составной первичный ключ.

Не имеют неключевых атрибутов

Сущность **ПунктЗаказа** не соответствуют 2НФ так как

представлена в 1НФ,

имеют составной первичный ключ,

но атрибут **Название** зависит не от полного первичного ключа, а от его части, атрибута **Книга#**

45. Для приведения к 2НФ удалите атрибут **Название** из сущности **ПунктЗаказа.**

5.13. Приведение к 3НФ

Сущности **Автор, Книга, Издательство, Редактор, Заказ, ПунктЗаказа, КнигаАвтор** соответствуют 3НФ так как

представлены в 2НФ,

между неключевыми атрибутами нет взаимосвязей.

Сущность **Автор** не соответствует 3НФ, так как

атрибут **Позиция** (позиция автора в списке авторов) не зависит от первичного ключа Автор#. У каждого автора может быть много книг и в каждой он может занимать разные позиции (1,2, 3 и т.д). На самом деле позиция определяется парой значений Автор# и Книга#.

атрибут **Разделение гонорара** не зависит от первичного ключа Автор**#,** а зависит от пары значений Автор# и Книга#.

46. Для приведения к 3НФ переместите атрибуты **Разделение гонорара** и **Позиция** из сущности **Автор** в сущность **КнигаАвтор.**

5.14. Проверка

47. Выполните проверку логической модели: выберите **Tools, Design Rules**, выберите **Logical, Entity**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

48. Выберите **Attribute**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

49. Выберите **Key**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

50. Выберите **Logical**, выберите **Apply Selected,** ошибки будут красные, предупреждения синие.

51. Логическая модель после нормализации должна быть как на рис. 20.

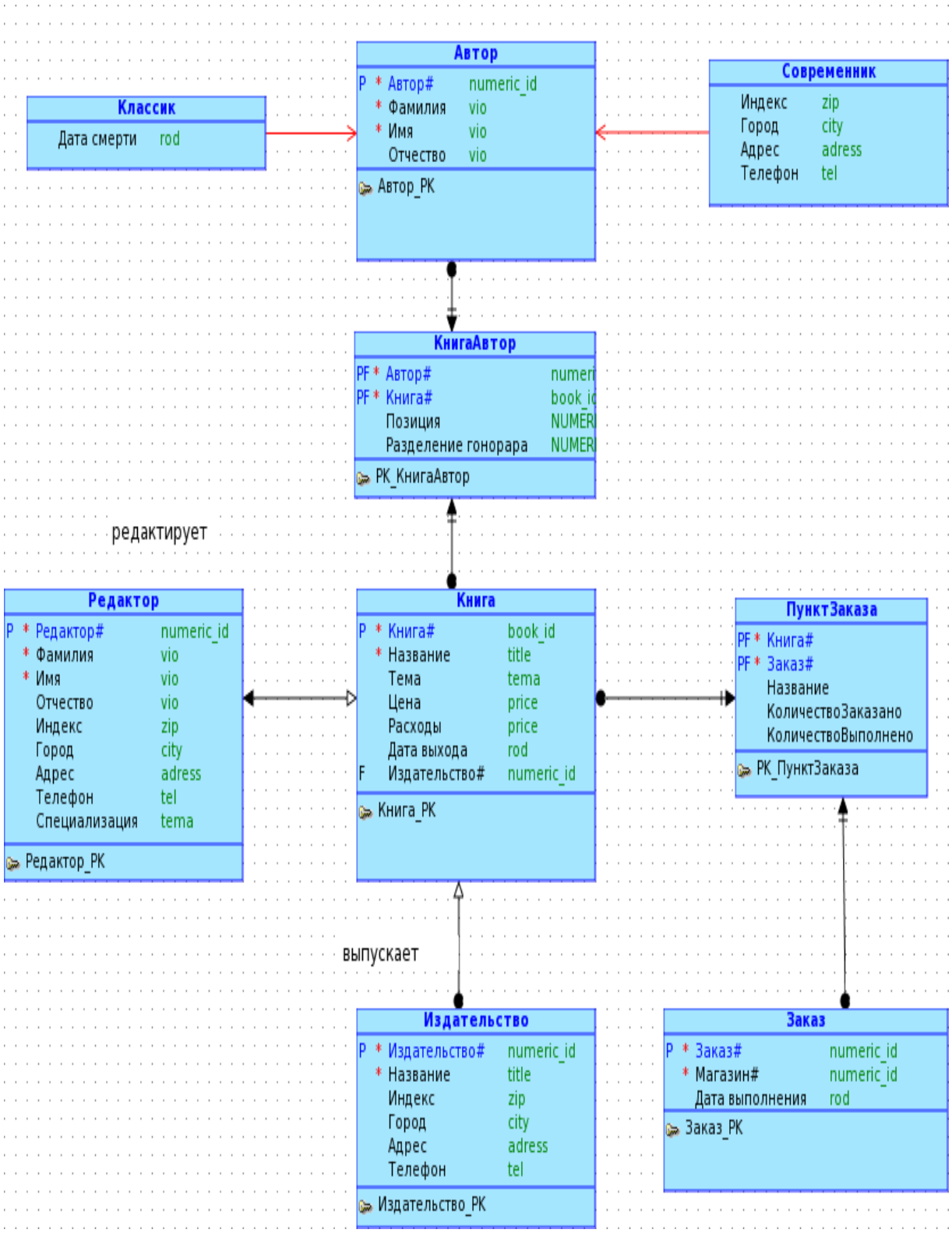

Рис. 20: Нормализованная логическая модель в нотации Бахмана

6-й день. Реляционная модель

Один среди своих владений,
Чтоб только время проводить,
Сперва задумал наш Евгений
Порядок новый учредить.
А.С. Пушкин. Евгений Онегин, гл. 2, IV

6.1. Представление супертипов и подтипов сущности

Если в логической модели присутствуют подтипы, то возможны следующие способы их представления в реляционной схеме:

Single Table: метаданные таблиц подтипа входят в таблицу супертипа.

Table per Child: метаданные таблицы супертипа входят в таблицы подтипа.

Table for each Entity: отдельные таблицы создаются для каждого супертипа и подтипа.

1. Определите стратегию супертипа: выберите сущность **Автор**, **МП**, выберите **Properties, General,** в списке **FWD Engineer Strategy** выберите **Single Table,** нажмите **Ладно**.

6.2. Замена имен

2. Введите сокращенное имя атрибута: выберите **Автор**, **МП,** выберите **Properties, Attributes,** выберите **Автор#**, выберите **Properties, МП,** выберите **Properties,** в поле **Preferred Abbreviation** введите **avtor_id**, нажмите **Ладно**.

3. Введите сокращенные имена для остальных атрибутов сущности Автор по табл. 3 – см. **2.**

4. Введите сокращенные имена атрибутов для всех сущностей, в том числе и для внешних ключей по данным таблиц 3, 4, 5, 6, 7, 8, 9 – см. **2.**

6.3. Генерация

5. Создайте реляционную модель: выберите вкладку **Logical,** выберите **Design**, нажмите **Engineer to Relational Mode**, выберите вкладку **General Option**, флажком выберите **Use Preferred Abbreviation,** выберите **Engineer Coordinates,** выберите **Engineer Formating,** выберите **Apply Name Translation,** выберите **Engineer.**

6.4. Проверка

6. Выполните проверку реляционной модели: выберите **Tools, Design Rules**, выберите **Relational, Table**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

7. Выберите **Column**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

8. Выберите **Index**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

9. Примените правила уникальности: выберите **avtor, МП,** выберите **Properties,** выберите **Naming Rules**, флажками должны быть выбраны все параметры, нажмите **Ладно**, и **Ладно**.

10. Примените правила ко всем таблицам – см. **9**.

11. Выполните проверку реляционной модели: выберите **Tools, Design Rules**, выберите **Relational, Table**, выберите **Apply Selected**, ошибки будут красные, предупреждения синие.

12. Реляционная модель должна быть как на рис. 21.

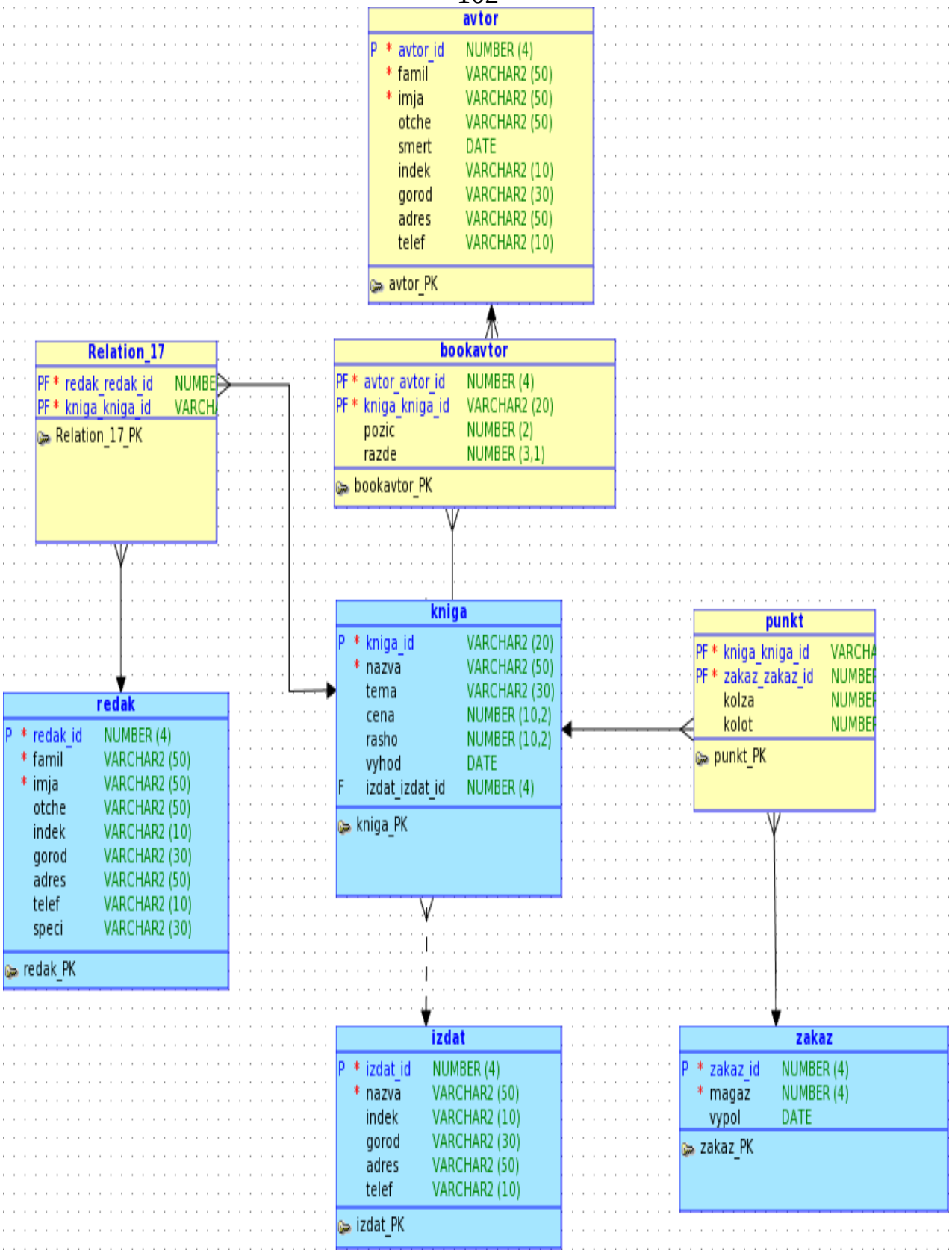

Рис. 21: Реляционная модель

7-й день. Физическая модель

Родила царица в ночь
Не то сына, не то дочь;
Не мышонка, не лягушку,
А неведому зверюшку.
А.С. Пушкин. Сказка о царе Салтане

7.1. Генарация DDL

1. Создайте физическую модель: выберите **Physical, Open Physical Model,** выберите СУБД **Oracle Database 10g,** нажмите **Ладно**. Созданная физическая модель находится в дереве в левой части окна, в узле реляционной модели.

2. Создайте DDL-скрипт: выберите **File**, **Export,** выберите **DDL File,** выберите СУБД **Oracle Database 10g**, нажмите **Generate, в** появившемся окне выберите значения по умолчанию нажатием **Ладно**.

3. Сохраните скрипт: выберите **Save**, введите имя **izdat.sql**

7.2. Запуск СУБД

4. Запустите службу Oracle: **Пуск, Программы, Oracle Database 10g Express Edition, Start Database**

Должно появиться в окне следующие сообщение:

C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleXETNSListener

Затребованная служба уже запущена.

C:\oraclexe\app\oracle\product\10.2.0\server\BIN>net start OracleServiceXE

Служба "OracleServiceXE" запускается.....

Служба "OracleServiceXE" успешно запущена.

7.3. Генерация БД

5. Запустите Oracle SQL Developer: выберите **Пуск, Программы, Sqldeveloper** (или откройте каталог C:\sqldeveloper, выберите sqldeveloper.exe, M2)..

6. Создайте соединение: выберите **Connections**, **МП**, выберите **New connection**, в поле **Connection Name** введите **10hr**, в поле **Name** введите **hr**, в поле **Password** введите **hr**, в поле **hostname** введите

localhost, в поле **SID** введите **xe**, нажмите **Test**, если сообщения об ошибке нет, то нажмите **Connect**.

7. Создайте таблицы: выберите **Tools**, нажмите **SQL Worksheet**, в списке **Connection** выберите, например, **10hr**, откройте с помощью Блокнота файл **izdat.sql**, скопируйте его содержимое в буфер обмена, вставьте содержимое в SQL Worksheet, нажмите **Run Script** (F5). Должны появиться следующие сообщения:

CREATE TABLE succeeded. ALTER TABLE MyMy succeeded. CREATE TABLE succeeded. ALTER TABLE Room succeeded. CREATE TABLE succeeded. ALTER TABLE Type succeeded. ALTER TABLE MyMy succeeded. ALTER TABLE MyMy succeeded

8. Если в процессе генерации появились ошибки, то их необходимо исправить в модели и снова выполнить генерацию DDL.

7.4. Загрузка данных

9. Введите данные: выберите **avtor,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 20, нажмите **Commit**.

AVTOR_ID	FAMIL	IMJA	OTCHE	SMERT	INDEK	GOROD	ADRES	TELEF
1000	Деменюк	Андрей	Фомич		660000	Красноярск	Академгородок, 10	2275419
1001	Астафьев	Виктор	Петрович	29 ноября 2001				
1002	Фаст	Геннадий	Генрихович		663180	Енисейск	Дударева, 20	3911522739

Таблица 20: Avtor

10. Введите данные: выберите **izdat,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 21, нажмите **Commit**.

IZDAT_ID	IMJA	INDEK	GOROD	ADRES	TELEF
2000	КАСС	660000	Красноярск	ул. Маерчака, д. 65 стр.23	(391) 2595960
2001	Запад	660028	Красноярск	ул. Ладо Кецховели, 54	(391) 2436789
2002	Полян	660016	Красноярск	ул. Семафорная, 215	(391) 2368847

Таблица 21: Izdat

11. Введите данные: выберите **kniga,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 22, нажмите **Commit**.

KNIGA_ID	NAZVA	TEMA	CENA	RASHO	VYHOD	IZDAT_IZDAT_ID
3000	Река жизни Виктора Астафьева	Литературоведение	200	150000	26 февраля 2010	2000
3001	Акцент ночи	Стихи	40	2000	1998	2001
3002	Этюды по Ветхому Завету	Богословие	250	80000	2007	2001

Таблица 22: Kniga

12. Введите данные: выберите **bookavtor,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 23, нажмите **Commit**.

AVTOR_AVTOR_ID	KNIGA_KNIGA_ID	POZIC	RAZDE
1000	3001	1	1
1001	3000		
1002	3002	1	1

Таблица 23: Bookavtor

13. Введите данные: выберите **redak,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 24, нажмите **Commit**.

REDAK_ID	FAMIL	IMJA	OTCHE	INDEK	GOROD	ADRES	TELEF	SPECI
4000	Кульчицкая	Валентина	Андреевна	660016	Красноярск	ул. Семафорная, 215	(391) 2368847	
4001	Пантелеева	Антонина	Федоровна	660028	Красноярск	пр. Свободный, 40	(391) 2435677	

Таблица 24: Redak

14. Введите данные: выберите **Relation_17,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 25, нажмите **Commit**.

REDAK_REDAK_ID	KNIGA_KNIGA_ID
4001	3000
4001	3001
4000	3002

Таблица 25: Relation_17

15. Введите данные: выберите **zakaz,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 26, нажмите **Commit**.

ZAKAZ_ID	MAGAZ	VYPOL
5000	6000	14 июля 2010
5001	6000	15 июля 2010

Таблица 26: Zakaz

16. Введите данные: выберите **punkt,** выберите вкладку **Data,** нажмите **плюс**, введите данные по табл. 27, нажмите **Commit**.

KNIGA_KNIGA_ID	ZAKAZ_ZAKAZ_ID	KOLZA	KOLOT
3000	5000	100	100
3002	5000	10	10
3001	5001	20	20

Таблица 27: Punkt

Литература

1. Пржиялковский В. В. Абстракции в проектировании БД // СУБД. – 1998. – № 1-2. С. 90-97.

2. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем . http://citforum.ru/database/case/index.shtml

3. Энсор Дейв, Стивенсон Йен. Oracle. Проектирование баз данных: Пер. с англ. – Киев: ВНЧ, 1999. – 560 с.

5. Дейт К. Дж. Введение в системы баз данных: Пер. с англ. – Киев, М., СПб.: Изд. дом «Вильямс», 1999. – 848 с.

6. Кузнецов С. Введение в информационные системы // СУБД.– 1997. – № 2. С. 83-96.

8. Маклаков С. В. Bpwin и ERwin. CASE-средства разработки информационных систем. – М.: ДИАЛОГ-МИФИ, 1999. – 256 с.

9. Пэйдж Вильям. Использование Oracle8/8i. Специальное издание: Пер. с англ. – М.: Изд. дом «Вильямс», 1999. – 1024 с.

10. Боуман Джудит С., Эмерсон Сандра Л., Дарновски Марси. Практическое руководство по SQL: Пер. с англ. – Киев: Диалектика, 1997. – 320 с.

11. ERwin Version 3.0 Methods Guide.

12. Oracle представила инструментарий проектирования баз данных SQL Developer Data Modeler. http://www.interface.ru/home.asp?artId=21196

13. Разработка приложений баз данных: от моделирования данных до промышленных приложений.
http://www.oracle.com/global/ru/oramag/mar2009/oracle_develop_omre.html

14. Обзор продуктов и технологий Oracle для разработчиков. http://oracle.axoft.ru/fordev/obzor.php

15. Watch this 5 minute online demo
http://download.oracle.com/otn_hosted_doc/datamodeler/demos/IntroDataModel/IntroDataModel.html

16. Sue Harper. Oracle SQL Developer 2.1. Packt Publishing, December 16, 2009, 978-1-847196-26-2, 496 p. http://www.packtpub.com/oracle-sql-developer-2-1/book

17. Кузнецов. С.Д. Наследование типов сущности и типов связи. http://www.intuit.ru/department/database/rdbintro/9/5.html

18. Миндалев И.В. Моделирование с помощью CASE-средства Erwin за 8 дней: Учеб. Пособие / Краснояр. гос. аграр. ун-т. – Красноярск, 2001. – 85 с.

19. Oracle Database Express Edition Installation Guide for Linux.
http://www.oracle.com/technetwork/database/express-edition/downloads/toc-090217.html

20. Oracle® Application Express Installation Guide Release 4.0.
http://download.oracle.com/docs/cd/E17556_01/doc/install.40/e15513/toc.htm

Оглавление