Министерство науки и высшего образования
Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Воронежский государственный технический университет»

Кафедра компьютерных интеллектуальных технологий
проектирования

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине
«Архитектура информационных систем» для студентов
направления 09.03.02 «Информационные системы и
технологии» всех форм обучения

Воронеж 2022

УДК 004.9

Составитель:
Филимонова А.А.

Методические указания к выполнению лабораторных работ по дисциплине «Теория информационных процессов и систем» для студентов направления 09.03.02 «Информационные системы и технологии» всех форм обучения / ГОУВПО «Воронежский государственный технический университет»; сост. А.А. Филимонова. Воронеж, 2022. 37 с.

Методические указания содержат теоретические сведения, необходимые для выполнения лабораторных работ по дисциплине «Архитектура информационных систем».

Предназначены для студентов 2 курса.

Методические указания подготовлены в электронном виде и содержатся в файле arcis.pdf.

Ил. 7. Табл. 2. Библиогр.: 3 назв.

УДК 621.3.049.7.002 (075)
ББК 38.54

Рецензент - А. В. Кузовкин, д-р техн. наук, проф., зав. кафедрой графики, конструирования и информационных технологий в промышленном дизайне ВГТУ

Издается по решению редакционно-издательского совета

Воронежского государственного технического университета

Содержание

Лабораторная работа №1
«Выделение высокоуровневых требований к программной системе»

Цель работы: выявить и описать высокоуровневые требования к информационной системе, выделить актеров и варианты использования, построить диаграмму вариантов использования.

1. Теоретическая часть

1.1. Классификация и уровни требований

Существует несколько определений понятия «Требования к информационной системе»

Требование — это условие или возможность, которой должна соответствовать система.

В IEEE Standard Glossary of Software Engineering Terminology (1990) данное понятие трактуется шире. Требование — это:

1 условия или возможности, необходимые пользователю для решения проблем или достижения целей;

2 условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;

3 документированное представление условий или возможностей для пунктов 1 и 2.

Требования — это исходные данные, на основании которых проектируются и создаются автоматизированные информационные системы.

Существует значительное количество различных методов классификации требований.

Требования к продукту. В своей основе требования — это то, что формулирует заказчик. Цель, которую он преследует — получить хороший конечный продукт: функциональный и удобный в использовании. Поэтому требования к продукту

являются основополагающим классом требований.

Требования к проекту. Вопросы формулирования требований к проекту, т.е. к тому, как Разработчик будет выполнять работы по созданию целевой системы, казалось бы, не лежат в компетенции Заказчика. Без регламентации процесса Заказчиком легко можно было бы обойтись, если бы все проекты всегда выполнялись точно и в срок. Однако, к сожалению, мировая статистика результатов программных проектов говорит об обратном. Заказчик, вступая в договорные отношения с Разработчиком, несет различные риски, главными из которых является риск получить продукт с опозданием, либо ненадлежащего качества. Основные мероприятия по контролю и снижению риска — регламентация процесса создания программного обеспечения и его аудит.

Обычно выделяют три уровня требований.

- На верхнем уровне представлены так называемые бизнес-требования (business requirements). Примеры бизнес-требования: система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.
- Следующий уровень — уровень требований пользователей (user requirements). Пример требования пользователя: система должна представлять диалоговые средства для ввода исчерпывающей информации о заказе, последующей фиксации информации в базе данных и маршрутизации информации о заказе к сотруднику, отвечающему за его планирование и исполнение. Требования пользователей часто бывают плохо структурированными, дублирующимися, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.
- Третий уровень — функциональный (functional

requirements). Пример функциональных требований (или просто функций) по работе с электронным заказом: заказ может быть создан, отредактирован, удален и перемещен с участка на участок.

Функциональные требования регламентируют функционирование или поведение системы (behavioral requirements). Функциональные требования отвечают на вопрос «что должна делать система» в тех или иных ситуациях. Функциональные требования устанавливают цели, задачи и сервисы, предоставляемые системой Заказчику.

Функциональные требования обычно записываются в форме предписывающих правил: «система должна позволять кладовщику формировать приходные и расходные накладные». Другим способом являются так называемые варианты использования (uses cases) — популярный и весьма продуктивный способ представления требований.

Это основной, определяющий вид требований

Нефункциональные требования, соответственно, регламентируют внутренние и внешние условия или атрибуты функционирования системы. Выделяют следующие основные группы нефункциональных требований:

- Внешние интерфейсы (External Interfaces),
- Атрибуты качества (Quality Attributes),
- Ограничения (Constraints).

Среди внешних интерфейсов в системах дистанционного обучения наиболее важным является интерфейс пользователя (User Interface, UI). Кроме того, выделяются интерфейсы с внешними устройствами (аппаратные интерфейсы), программные интерфейсы и интерфейсы передачи информации (коммуникационные интерфейсы).

В спецификациях Rational Unified Process при классификации требований используется модель FURPS+.

Аббревиатура FURPS обозначает следующие категории требований:

- Functionality (Функциональность);
- Usability (Применимость);

- Reliability (Надежность);
- Performance (Производительность);
- Supportability (эксплуатационная пригодность).

Символ «+» расширяет FURPS-модель, добавляя к ней:

- ограничения проекта;
- требования выполнения;
- требования к интерфейсу;
- физические требования.

Кроме того, в спецификациях RUP выделяются такие категории требований, как

- требования, указывающие на необходимость согласованности с некоторыми юридическими и нормативными актами;
- требования к лицензированию,
- требования к документированию.

1.2. Свойства требований

Основные свойства требований к программным системам:

- полнота — свойство, означающее, что текст требования не требует дополнительной детализации, то есть в нем предусмотрены все необходимые нюансы, особенности и детали;
- ясность (недвусмысленность, определенность, однозначность спецификаций) — сходность восприятия требования всеми совладельцами системы;
- корректность и согласованность — непротиворечивость требованиям своего уровня иерархии и требованиям «родительского» уровня;
- верифицируемость — пригодность требования к проверке;
- необходимость — свойства, без которых невозможно, либо затруднено выполнение автоматизированных бизнес-функций пользователей;
- полезность при эксплуатации — любые свойства,

повышающие эргономические качества продукта;

- осуществимость (выполнимость) — на практике определяется разумным балансом между ценностью (степенью необходимости и полезности) и требуемыми ресурсами;
- модифицируемость — возможность переработки требований и, при необходимости, поддерживание истории изменений для каждого положения;
- трассируемость — возможностью отследить связь между требованием и другими объектами информационной системы;
- упорядоченность по важности и стабильности — количественная оценка степени значимости (важности) требования и прогнозная оценка неизменности требований во времени;
- наличие количественной метрики (например, запрос должен отрабатываться не более чем ___ секунд).

1.3. Анализ требований

Анализ требований — один из основных рабочих потоков (workflow) программной инженерии.

В процессе анализа требований выделяют следующие составляющие:

- Requirements Elicitation (Извлечение требований);
- Requirements Analysis (Анализ требований в узком смысле);
- Requirements Specification (Специфицирование требований);
- Requirements Validation (Проверка требований).

RUP предлагает выделить в основном потоке анализа требований такие компоненты, как:

- Analyze the Problem (Анализ проблемы);
- Understand Stakeholder Needs (Понимание потребностей совладельцев);
- Define the System (Определение системы);

- Manage the Scope of the System (Управление контекстом системы);
- Refine the System Definition (Уточнение определения системы).

1.4. Модель вариантов использования

В настоящее время существует множество методик, языков, визуальных представлений, позволяющих моделировать требования к системе. При создании информационных систем стандартом де-факто является универсальный язык моделирования (UML).

Самым популярным и весьма эффективным способом повышения информативности требований является оформление их в виде вариантов использования (use case), предложенный И.Якобсоном.

Актер — это некто или нечто, обладающее активностью по отношению к программной системе. Актером может быть пользователь, работающий с системой. Помимо пользователя в качестве актора может рассматриваться другая программная система, аппаратное устройство, в ряде случаев — активная компонента самой системы. Поиск актеров корпоративной информационной системы обычно сводится к анализу ролей различных пользователей.

Вариант использования в первом приближении можно рассматривать, просто, как функцию, реализуемую системой.

Актер обозначается значком человечка, а вариант использования — овалом. Дополнительно в диаграммы могут быть добавлены комментарии.

Диаграмма задумана так, чтобы дать наиболее общее представление о функциональности системы (ее компоненты), не вдаваясь в детали взаимосвязей функций. Поэтому основной вид отношения, используемый в диаграмме — ассоциация между актером и вариантом использования.

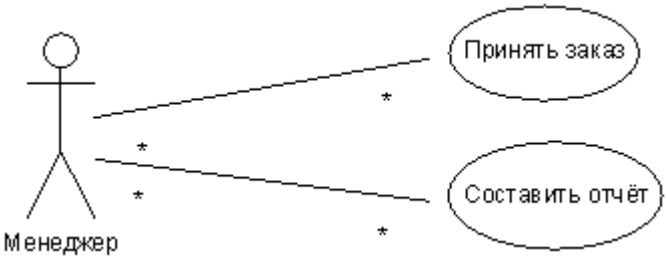

Рис. 1.1. Пример ассоциации между актером и вариантом использования

Другие виды отношений — отношение включения (include), расширения (extend) и обобщения/генерализации.

Включение служит для обозначения подчиненных вариантов использования (когда один или более вариантов использования содержат вызовы одной и той же функциональности).

Рис. 1.2. Отношение включения

Расширение соответствует точке расширения, используемой при описании варианта использования — задание альтернатив, привязанных к шагу основного сценария.

Рис. 1.3. Отношение расширения

Отношение обобщения может применяться как к актерам, так и к вариантам использования, с целью указания специализации одних относительно других.

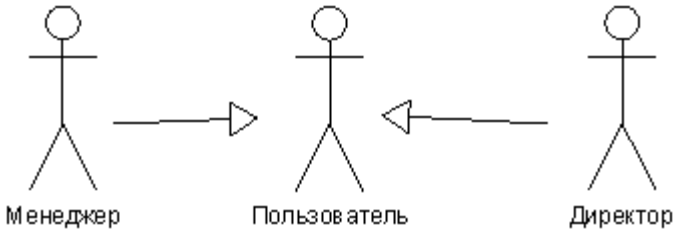

Рис. 1.4. Отношение обобщения

1.7. Вопросы для самоконтроля

1. Что такое требование?
2. Что относится к требованиям к продукту?
3. Что относится к требованиям к проекту?
4. Какие существуют уровни требований?
5. Что такое функциональные требования?
6. Что такое нефункциональные требования?
7. Классификация требований в спецификациях RUP.
8. Категории требований по RUP.
9. Какими основными свойствами обладают требования?
10. Основные составляющие процесса анализа требований.
11. Основное назначение модели вариантов использования.
12. Какие элементы включает диаграмма вариантов

использования?

13. Что представляет собой вариант использования?
14. Какие существуют виды отношений между элементами вариантов использования?

2. Практическая часть

2.1. Задание на лабораторную работу

Необходимо провести анализ и выявить высокоуровневые требования к системе в соответствие с вариантом задания на курсовой проект, разработать документ «Видение» для проектируемой системы и диаграмму вариантов использования.

2.2. Порядок выполнения работы

2.2.1. Создание документа «Видение».

Шаги, которые необходимо пройти для формирования документа «Видение»:

- Формулировка проблем;
- Идентификация совладельцев;
- Определение границ системы;
- Идентификация ограничений;
- Формулировка постановки задач;
- Определение возможностей системы;
- Оценка результатов.

Для описания проблем можно использовать следующий шаблон.

Таблица 1.1. Описание шаблона проблемы

Проблема	(описание проблемы)
Затрагивает	(совладельцы, затрагиваемые проблемой).
Ее следствием является	(каково влияние проблемы).
Успешное решение	(список некоторых ключевых преимуществ от успешного решения).

Описание возможностей системы представляет собой формулировку высокоуровневых требований.

Шаблон документа «Vision» RUP содержит следующие основные разделы:

1. Введение
2. Позиционирование
3. Описания совладельцев и пользователей
4. Краткий обзор изделия
5. Возможности продукта
6. Ограничения
7. Показатели качества
8. Старшинство и приоритеты
9. Другие требования к изделию
10. Требования к документации
11. Приложение.

Во введении описываются цель документа, его контекст (связь и взаимовлияние с различными проектами), определения, аббревиатуры и сокращения, ссылки на другие документы, краткое содержание.

В разделе «позиционирование» помещается определение решаемой проблемы (проблем), указывается целевой заказчик и исследуются деловые преимущества изделия перед аналогичными на рынке.

В описании совладельцев и пользователей, помимо собственно описания этих двух групп, исследуется демография рынка: целевые рыночные сегменты, размер и темпы роста рынка, существующие конкурентные предложения на рынке, репутация Разработчика на рынке.

Краткий обзор изделий содержит резюме изделия, описание его перспектив и ключевых возможностей, предположения и зависимости, указывается стоимость и ее калькуляция, рассматриваются вопросы лицензирования и инсталляции.

В разделе, посвященном возможностям продукта, они описываются более подробно, каждая — в отдельном

параграфе.

В раздел «Ограничения» следует выносить существующие технические, технологические и др. обстоятельства, которые необходимо учитывать на данной стадии.

Раздел «Показатели качества» содержит описание наиболее существенных нефункциональных требований к системе (эффективности, надежности, отказоустойчивости и др.).

Раздел «Старшинство и приоритеты» ранжирует сформулированные ранее требования и возможности системы по степени важности, очередности реализации и т.п.

Раздел «Другие требования к изделию» описывает применяемые стандарты, системные требования, эксплуатационные требования, требования к окружающей среде.

В требованиях к документации приводятся ключевые характеристики руководства пользователя, интерактивной справки, руководства по установке и конфигурированию, файла Read Me.

В приложение выносятся атрибуты возможностей. RUP рекомендует следующий набор атрибутов: статус, выгода, объем работ, риск, стабильность, целевой выпуск, назначение, причина.

2.2.2. Выявление актеров. Необходимо выявить максимально возможное количество актеров (это могут быть потенциальные пользователи системы, а также внешние системы). Осуществить классификацию актеров, разбить их по группам; сформулировать окончательный реестр актеров. Каждому актеру сопоставить краткое (в один абзац) описание.

2.2.3. Выявление варианты использования. Для каждого из актантов необходимо выявить максимально возможное количество вариантов использования. Каждому из них сопоставить краткую (в одно предложение) формулировку.

2.2.4. Составление диаграммы вариантов использования.

2.3. Контрольные вопросы

1. Для чего предназначен документ «Видение»?
2. Какие этапы включает формирования документа «Видение»?
3. Каким образом осуществляется описание проблемы?
4. Какие основные проблемы были сформулированы для проектируемой системы?
5. Какие основные разделы включает документ «Видение»?
6. В каком разделе документы «Видение» описываются нефункциональные требования?
7. В каком разделе документа «Видение» описываются функциональные требования?
8. В каком разделе документа «Видение» перечисляются необходимые стандарты открытых систем?
9. Как осуществляется выделение и описание актеров?
10. Как выявляются и описываются варианты использования?
11. Какое программное средство применялось для построения диаграммы вариантов использования и почему?

Лабораторная работа №2
«Построение диаграммы потоков данных»

Цель работы: провести анализ входной и выходной информации, выявить внешние сущности, процессы, накопители данных и потоки данных, построить диаграмму потоков данных.

1. Теоретическая часть

1.1. Диаграмма потоков данных

Диаграммы потоков данных (DFD) являются основным средством моделирования функциональных требований

проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств — продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами. Диаграммы потоков данных используются для описания документооборота и обработки информации.

DFD описывает:

- функции обработки информации (работы, activities);
- документы (стрелки, arrows), объекты, сотрудников или отделы, которые участвуют в обработке информации;
- внешние ссылки (external references), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;
- таблицы для хранения документов (хранилище данных, data store).

1.2. Детализация процессов и контекстная диаграмма

Декомпозиция DFD осуществляется на основе процессов: каждый процесс может раскрываться с помощью DFD нижнего уровня.

Важную специфическую роль в модели играет специальный вид DFD — контекстная диаграмма, моделирующая систему наиболее общим образом. Контекстная диаграмма отражает интерфейс системы с внешним миром, а именно, информационные потоки между системой и внешними сущностями, с которыми она должна быть связана. Она идентифицирует эти внешние сущности, а также, как правило, единственный процесс, отражающий главную цель или природу системы насколько это возможно. И хотя контекстная диаграмма выглядит тривиальной, несомненная ее полезность заключается в том, что она устанавливает границы

анализируемой системы. Каждый проект должен иметь ровно одну контекстную диаграмму, при этом нет необходимости в нумерации единственного ее процесса.

DFD первого уровня строится как декомпозиция процесса, который присутствует на контекстной диаграмме.

Построенная диаграмма первого уровня также имеет множество процессов, которые в свою очередь могут быть декомпозированы в DFD нижнего уровня. Таким образом строится иерархия DFD с контекстной диаграммой в корне дерева. Этот процесс декомпозиции продолжается до тех пор, пока процессы могут быть эффективно описаны с помощью коротких (до одной страницы) миниспецификаций обработки (спецификаций процессов).

При таком построении иерархии DFD каждый процесс более низкого уровня необходимо соотнести с процессом верхнего уровня. Обычно для этой цели используются структурированные номера процессов. Так, например, если мы детализируем процесс номер 2 на диаграмме первого уровня, раскрывая его с помощью DFD, содержащей три процесса, то их номера будут иметь следующий вид: 2.1, 2.2 и 2.3. При необходимости можно перейти на следующий уровень, т.е. для процесса 2.2 получим 2.2.1, 2.2.2. и т.д.

1.3. Нотации изображения диаграммы потоков данных и основные элементы

Для изображения DFD традиционно используются две различные нотации: Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson).

Основные символы DFD изображены на рис.2.1.

Поток данных определяет информацию, передаваемую через некоторое соединение (кабель, почтовая связь, курьер) от источника к приемнику. На DFD диаграммах потоки данных изображаются линиями со стрелками, показывающими их направление. Каждому потоку данных присваивается имя, отражающее его содержание. Иногда информация может двигаться в одном направлении, обрабатываться и возвращаться

назад в ее источник. Такая ситуация может моделироваться либо двумя различными потоками, либо одним - двунаправленным.

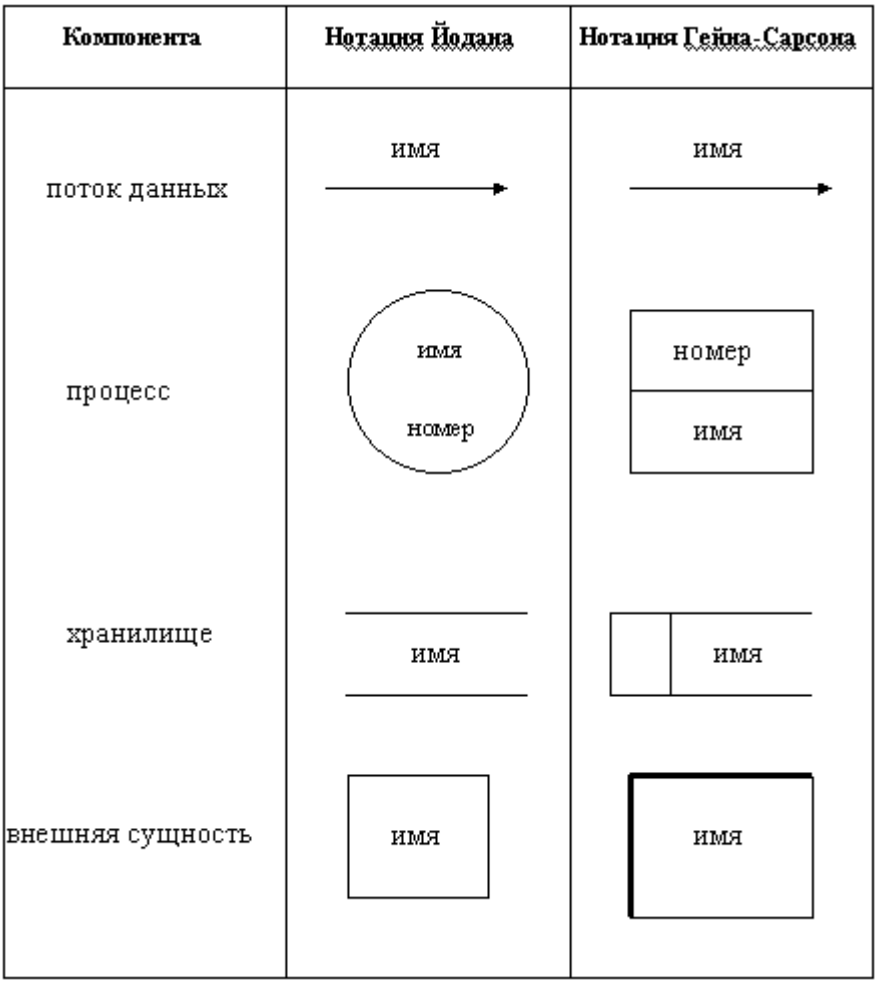

Рис. 2.1. Элементы DFD

Процессы представляют собой преобразование входных потоков данных в выходные в соответствии с определенным алгоритмом. В реальной жизни процесс может выполняться

некоторым подразделением организации, выполняющим обработку входных документов и выпуск отчетов, отдельным сотрудником, программой, установленной на компьютере, специальным логическим устройством и тому подобное. Имя процесса должно содержать глагол в неопределенной форме с последующим дополнением (например, ВЫЧИСЛИТЬ МАКСИМАЛЬНУЮ ВЫСОТУ). Кроме того, каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер может использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Накопители данных (хранилища) предназначены для изображения неких абстрактных устройств для хранения информации, которую можно туда в любой момент времени поместить или извлечь, безотносительно к их конкретной физической реализации. Накопители данных являются неким прообразом базы данных информационной системы организации. Имя хранилища должно идентифицировать его содержимое и быть существительным. В случае, когда поток данных входит или выходит в/из хранилища, и его структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.

Под внешней сущностью (External Entity) понимается материальный объект, являющийся источником или приемником информации. В качестве внешней сущности на DFD диаграмме могут выступать заказчики, поставщики, клиенты, склад, банк и другие. Ее имя должно содержать существительное, например, СКЛАД ТОВАРОВ. Предполагается, что объекты, представленные такими узлами, не должны участвовать ни в какой обработке.

1.4. Вопросы для самоконтроля

1. Что представляет собой диаграмма потоков данных?
2. Что такое контекстная диаграмма?
3. Как формируется иерархия DFD?
4. Какие нотации применяются для изображения DFD?

5. Что такое «поток данных»?
6. Как изображается «поток данных»?
7. Что такое «процесс»?
8. Как изображается «процесс»?
9. Что такое «хранилище»?
10. Как изображается «хранилище»?
11. Что такое «внешняя сущность»?
12. Как изображается «внешняя сущность»?

2. Практическая часть

2.1. Задание на лабораторную работу

В лабораторной работе требуется провести анализ входной и выходной информации, выявить потоки данных, процессы, хранилища и внешние сущности, построить диаграмму потоков данных в необходимой детализацией в соответствии с вариантом на курсовой проект

2.2. Описание входной и выходной информации

Анализ входной и выходной информации является важным этапом проектирования программной системы. На данном этапе описываются исходные данные, основные источники исходных данных, а также источники данных, используемые для корректировки. Входные данные можно описать с использованием следующего шаблона.

Таблица 2.1. Входные данные

(источник данных)	(входной параметр)	(тип данных)	(ограничения)
	(входной параметр)	(тип данных)	(ограничения)
	(……)	(……)	(……)

Также на данном этапе производится описание выходной информации, форма и периодичность отчетов, выдаваемых проектируемой системой.

2.3. Построение диаграммы потоков данных

Главная цель построения иерархического множества DFD заключается в том, чтобы сделать требования ясными и понятными на каждом уровне детализации, а также разбить эти требования на части с точно определенными отношениями между ними. Для достижения этого целесообразно пользоваться следующими рекомендациями:

1. Размещать на каждой диаграмме от 3 до 6-7 процессов. Верхняя граница соответствует человеческим возможностям одновременного восприятия и понимания структуры сложной системы с множеством внутренних связей, нижняя граница выбрана по соображениям здравого смысла: нет необходимости детализировать процесс диаграммой, содержащей всего один или два процесса.
2. Не загромождать диаграммы несущественными на данном уровне деталями.
3. Декомпозицию потоков данных осуществлять параллельно с декомпозицией процессов; эти две работы должны выполняться одновременно, а не одна после завершения другой.
4. Выбирать ясные, отражающие суть дела, имена процессов и потоков для улучшения понимания диаграмм, при этом стараться не использовать аббревиатуры.
5. Однократно определять функционально идентичные процессы на самом верхнем уровне, где такой процесс необходим, и ссылаться к нему на нижних уровнях.
6. Пользоваться простейшими диаграммными техниками: если что-либо возможно описать с помощью DFD, то это и необходимо делать, а не использовать для описания более сложные объекты.
7. Отделять управляющие структуры от обрабатывающих структур (т.е. процессов), локализовать управляющие структуры.

В соответствии с этими рекомендациями процесс

построения модели разбивается на следующие этапы:

1. Разделение множества требований и организация их в основные функциональные группы.
2. Идентификация внешних объектов, с которыми система должна быть связана.
3. Идентификация основных видов информации, циркулирующей между системой и внешними объектами.
4. Предварительная разработка контекстной диаграммы, на которой основные функциональные группы представляются процессами, внешние объекты - внешними сущностями, основные виды информации - потоками данных между процессами и внешними сущностями.
5. Изучение предварительной контекстной диаграммы и внесение в нее изменений по результатам ответов на возникающие при этом изучении вопросы по всем ее частям.
6. Построение контекстной диаграммы путем объединения всех процессов предварительной диаграммы в один процесс, а также группирования потоков.
7. Формирование DFD первого уровня на базе процессов предварительной контекстной диаграммы.
8. Проверка основных требований по DFD первого уровня.
9. Декомпозиция каждого процесса текущей DFD с помощью детализирующей диаграммы или спецификации процесса.
10. Проверка основных требований по DFD соответствующего уровня.
11. Добавление определений новых потоков в словарь данных при каждом их появлении на диаграммах.
12. Параллельное (с процессом декомпозиции) изучение требований (в том числе и вновь поступающих), разбиение их на элементарные и идентификация процессов или спецификаций процессов, соответствующих этим требованиям.

13. После построения двух-трех уровней проведение ревизии с целью проверки корректности и улучшения понимаемости модели.
14. Построение спецификации процесса (а не простейшей диаграммы) в случае, если некоторую функцию сложно или невозможно выразить комбинацией процессов.

2.4. Контрольные вопросы

1. Из каких этапов состоит процесс построения DFD?
2. Сколько процессов рекомендуется размещать на DFD?
3. В какой последовательности осуществляется декомпозиция потоков данных и процессов?
4. По какому принципу выбираются имена процессов и потоков?
5. Какие потоки данных, процессы, внешние сущности и хранилища были выделены в проектируемой системе?
6. Сколько уровней DFD было выделено для проектируемой системы?

Лабораторная работа №3
«Проектирование информационного обеспечения с использованием семантических моделей»

Цель работы: выделить основные сущности и атирибуты предметной области, построить диаграмму «Сущность–связь» и осуществить генерацию схемы базы данных.

1. Теоретическая часть

1.1. Семантические модели данных

Потребность проектировщиков баз данных в более удобных и мощных средствах моделирования предметной области вызвала к жизни направление семантических моделей данных. Хотя любая развитая семантическая модель данных, как и реляционная модель, включает структурную, манипуляционную и целостную части, главным назначением

семантических моделей является обеспечение возможности выражения семантики данных.

Чаще всего на практике семантическое моделирование используется на первой стадии проектирования базы данных. При этом в терминах семантической модели производится концептуальная схема базы данных, которая затем вручную преобразуется к реляционной (или какой-либо другой) схеме. Этот процесс выполняется под управлением методик, в которых достаточно четко оговорены все этапы такого преобразования. Основным достоинством данного подхода является отсутствие потребности в дополнительных программных средствах, поддерживающих семантическое моделирование. Требуется только знание основ выбранной семантической модели и правил преобразования концептуальной схемы в реляционную схему.

Преимущества семантического моделирования:

1. построение мощной и наглядной концептуальной схемы БД позволяет более полно оценить специфику моделируемой предметной области и избежать возможных ошибок на стадии проектирования схемы реляционной БД.
2. на этапе семантического моделирования производится важная документация (хотя бы в виде вручную нарисованных диаграмм и комментариев к ним), которая может оказаться очень полезной не только при проектировании схемы реляционной БД, но и при эксплуатации, сопровождении и развитии уже заполненной БД.

История систем автоматизации проектирования баз данных (CASE-средств) началась с автоматизации процесса рисования диаграмм, проверки их формальной корректности, обеспечения средств долговременного хранения диаграмм и другой проектной документации. Наличие электронного архива проектной документации помогает при эксплуатации, администрировании и сопровождении базы данных.

Подавляющее большинство современных CASE-средств,

представленных на рынке, обеспечивает автоматизированное преобразование диаграммных концептуальных схем баз данных, представленных в той или иной семантической модели данных, в реляционные схемы, специфицированные чаще всего на языке SQL. В типичной схеме SQL-ориентированной БД могут содержаться определения многих объектов (ограничений целостности общего вида, триггеров и хранимых процедур и т. д.), которые невозможно сгенерировать автоматически на основе концептуальной схемы. Поэтому на завершающем этапе проектирования реляционной схемы снова требуется ручная работа проектировщика.

Как правило, CASE-средства, автоматизирующие преобразование концептуальной схемы БД в реляционную, производят реляционную схему базы данных в третьей нормальной форме. Нормализация более высокого уровня усложняет программную реализацию и редко требуется на практике.

1.2 Диаграммы «Сущность-связь»

Диаграммы «Сущность–связь» (ER-диаграммы, Entity–Relationship Diagram, ERD) позволяют строить модели логической структуры данных предметной области, а так же производить моделирование физической структуры систем хранения данных.

Различают концептуальные и физические ER-диаграммы. Концептуальные диаграммы не учитывают особенностей конкретных СУБД. Физические диаграммы строятся по концептуальным и представляют собой прообраз конкретной базы данных. Сущности, определенные в концептуальной диаграмме становятся таблицами, атрибуты становятся колонками таблиц (при этом учитываются допустимые для данной СУБД типы данных и наименования столбцов), связи реализуются путем миграции ключевых атрибутов родительских сущностей и создания внешних ключей.

При правильном определении сущностей, полученные таблицы будут сразу находиться в 3НФ. Основное достоинство

метода состоит в том, модель строится методом последовательных уточнений первоначальных диаграмм.

Логическая модель данных предметной области обеспечивает разработчикам понимание структур данных. После ее разработки следует приступать к моделированию физической структуры систем хранинения выявленных объектов данных, то есть к разработке физической модели данных.

По одной логической модели может быть построено несколько физических моделей — по одной для каждой поддерживаемой СУБД. Построение физической модели данных состоит из двух этапов:

- Нормализация модели данных.
- Денормализация модели данных.

Существует ряд правил организации структур данных, называемых нормальными формами. Нормализация — процесс приведения модели структуры данных к некоторой нормальной форме. Как правило, используется третья нормальная форма. Она обеспечивает эффективное и неизбыточное хранение данных.

В процессе нормализации анализируется структура БД и выявляются элементы, противоречащие определенной нормальной форме. После этого разработчик меняет структуру так, чтобы устранить выявленные несоответствия.

Денормализация — процесс, обратный нормализации. Он заключается во внесении в структуру БД изменений, нарушающих требования нормальных форм. Основной целью денормализации является повышение производительности БД.

1.3. Основные компоненты диаграммы «Сущность–связь»

Основными компонентами ER–диаграмм являются:

- сущности — важные для предметной области объекты;
- атрибуты — их свойства;
- связи — отношения друг с другом.

Нотация ERD была впервые введена П. Ченом и получила дальнейшее развитие в работах Баркера.

Сущность (Entity) — реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области, информация о котором подлежит хранению (рис. 3.1).

Рис. 3.1. Графическое обозначение сущности

Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности. Каждая сущность должна обладать некоторыми свойствами:

- каждая сущность должна иметь уникальное имя, и к одному и тому же имени должна всегда применяться одна и та же интерпретация. Одна и та же интерпретация не может применяться к различным именам, если только они не являются псевдонимами;
- сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через связь;
- сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый экземпляр сущности;
- каждая сущность может обладать любым количеством связей с другими сущностями модели.

Следующим шагом моделирования является идентификация связей.

Связь (Relationship) — поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной

области. Связь — это ассоциация между сущностями, при которой, как правило, каждый экземпляр одной сущности, называемой родительской сущностью, ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности, называемой сущностью-потомком, а каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности–родителя. Таким образом, экземпляр сущности–потомка может существовать только при существовании сущности родителя.

Связи может даваться имя, выражаемое грамматическим оборотом глагола и помещаемое возле линии связи. Имя каждой связи между двумя данными сущностями должно быть уникальным, но имена связей в модели не обязаны быть уникальными. Имя связи всегда формируется с точки зрения родителя, так что предложение может быть образовано соединением имени сущности–родителя, имени связи, выражения степени и имени сущности–потомка.

Степень связи и обязательность графически изображаются следующим образом (рис. 3.2).

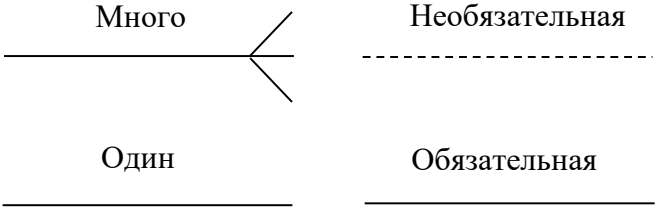

Рис. 3.2. Графическое обозначение связей

Последним шагом моделирования является идентификация атрибутов.

Атрибут — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Атрибут представляет тип характеристик или

свойств, ассоциированных с множеством реальных или абстрактных объектов (людей, мест, событий, состояний, идей, пар предметов и т.д.). Экземпляр атрибута - это определенная характеристика отдельного элемента множества. Экземпляр атрибута определяется типом характеристики и ее значением, называемым значением атрибута. В ER–модели атрибуты ассоциируются с конкретными сущностями. Таким образом, экземпляр сущности должен обладать единственным определенным значением для ассоциированного атрибута.

Атрибут может быть либо обязательным, либо необязательным (рис. 3.3). Обязательность означает, что атрибут не может принимать неопределенных значений (null values). Атрибут может быть либо описательным (т.е. обычным дескриптором сущности), либо входить в состав уникального идентификатора (первичного ключа).

Уникальный идентификатор — это атрибут или совокупность атрибутов и/или связей, предназначенная для уникальной идентификации каждого экземпляра данного типа сущности. В случае полной идентификации каждый экземпляр данного типа сущности полностью идентифицируется своими собственными ключевыми атрибутами, в противном случае в его идентификации участвуют также атрибуты другой сущности-родителя (рисунок 3.4).

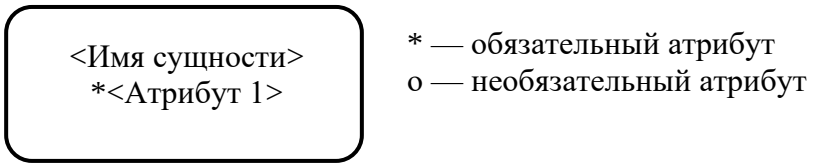

Рис. 3.3. Графическое обозначение атрибутов

Каждый атрибут идентифицируется уникальным именем, выражаемым грамматическим оборотом существительного, описывающим представляемую атрибутом характеристику. Атрибуты изображаются в виде списка имен внутри блока ассоциированной сущности, причем каждый атрибут занимает

отдельную строку. Атрибуты, определяющие первичный ключ, размещаются наверху списка и выделяются знаком «#».

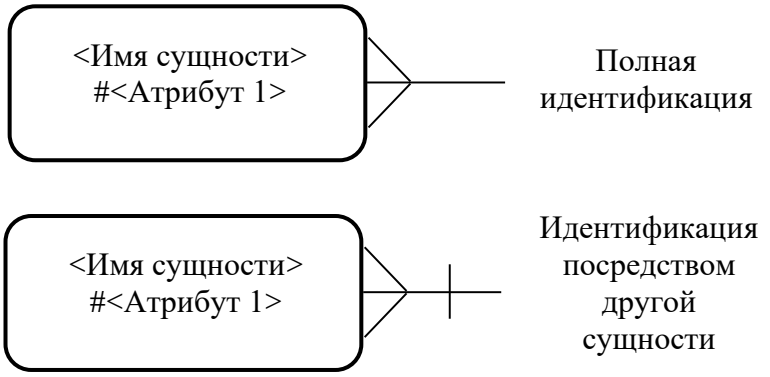

Рис. 3.4. Изображение уникальных идентификаторов

Каждая сущность должна обладать хотя бы одним возможным ключом. Возможный ключ сущности — это один или несколько атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа, а остальные — как альтернативные ключи.

Помимо перечисленных основных конструкций модель данных может содержать ряд дополнительных.

Подтипы и супертипы: одна сущность является обобщающим понятием для группы подобных сущностей.

Взаимно исключающие связи: каждый экземпляр сущности участвует только в одной связи из группы взаимно исключающих связей.

Рекурсивная связь: сущность может быть связана сама с собой.

Неперемещаемые (non-transferrable) связи: экземпляр сущности не может быть перенесен из одного экземпляра связи в другой.

1.4. Вопросы для самоконтроля

1. Что такое семантическая модель базы данных?
2. Какие можно выделить преимущества семантического моделирования?
3. Какие основные возможности предоставляют CASE–средства моделирования баз данных?
4. Что представляет собой диаграмма «Сущность–связь»?
5. Что такое концептуальная ER–диаграмма?
6. Что такое физическая ER–диаграмма?
7. Что такое «Сущность»?
8. Что такое «Связь»?
9. Что такое «Атрибут»?
10. Что такое «Уникальный идентификатор»?
11. Что такое «Подтип» и «Супертип»?
12. Что такое «Взаимно исключающие связи»?
13. Что такое «Рекурсивная связь»?
14. Что такое «Неперемещаемые связи»?
15. Что такое «Возможный ключ сущности»?
16. Как графически отображаются сущности и атрибуты?
17. Как графически отображаются связи между сущностями?
18. Как графически отображается степень связи?

2. Практическая часть

2.1. Задание на лабораторную работу

В лабораторной работе требуется провести анализ предметной области с целью выделения основных сущностей, их атрибутов, а также связей между сущностями. В отчете перечисляются выделенные сущности, для каждой сущности приводится краткое ее описание, а также список атрибутов.

После проведения анализа необходимо выполнить построение ER–модели с использованием любого средства моделирования баз данных и выполнить генерацию схемы базы данных. В отчет выносятся построенная диаграмма и SQL–скрипт создания базы данных.

2.2. Разработка ER–модели

При разработке ER-моделей необходимо получить следующую информацию о предметной области:

- Список сущностей предметной области.
- Список атрибутов сущностей.
- Описание взаимосвязей между сущностями.

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы.

ER-диаграмма должна подчиняться следующим правилам:

- каждая сущность, каждый атрибут и каждая связь должны иметь имя (связь супертипа или ассоциативная связь может не иметь имени);
- имя сущности должно быть уникально в рамках модели данных;
- имя атрибута должно быть уникально в рамках сущности;
- имя связи должно быть уникально, если для нее генерируется таблица БД;
- каждый атрибут должен иметь определение типа данных;
- сущность в необязательной связи должна иметь ключевой атрибут. То же самое относится к сильной сущности в слабой связи, супертипу в связи «супертип–подтип» и необязательной сущности в обязательной (полной) связи;
- подтип в связи «супертип–подтип» не может иметь ключевой атрибут;
- в ассоциативной или слабой связи может быть только одна ассоциативная (слабая) сущность;
- связь не может быть одновременно обязательной, «супертип–подтип» или ассоциативной.

2.3. Прямая и обратная инженерия базы данных информационной системы

Большинство современных систем автоматизации моделирования позволяют генерировать структуру базы данных по построенной модели. Этот процесс называется прямым проектированием (прямая инженерия). После генерация схемы рекомендуется выполнить ее проверку и доработку. Процедура преобразования ER–диаграммы в реляционную (точнее SQL–ориентированную) схему базы данных является многошаговой.

Каждый простой тип сущности превращается в таблицу. (Простым типом сущности называется тип сущности, не являющийся подтипом и не имеющий подтипов.) Имя сущности становится именем таблицы. Экземплярам типа сущности соответствуют строки соответствующей таблицы.

Каждый атрибут становится столбцом таблицы с тем же именем; может выбираться более точный формат представления данных. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, — не могут.

Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы. Если имеется несколько возможных уникальных идентификаторов, для первичного ключа выбирается наиболее характерный. Если в состав уникального идентификатора входят связи, к числу столбцов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (этот процесс может продолжаться рекурсивно, и в общем случае может привести к зацикливанию). Для именования этих столбцов используются имена концов связей и/или имена парных типов сущностей.

Связи «многие к одному» (и «один к одному») становятся внешними ключами, т. е. образуется копия уникального идентификатора сущности на конце связи «один», и соответствующие столбцы составляют внешний ключ таблицы, соответствующей типу сущности на конце связи «многие».

Необязательные связи соответствуют столбцам внешнего ключа, допускающим наличие неопределенных значений; обязательные связи — столбцам, не допускающим неопределенных значений. Если между двумя типами сущности A и B имеется связь «один к одному», то соответствующий внешний ключ по желанию проектировщика может быть объявлен как в таблице A, так и в таблице B. Чтобы отразить в определении таблицы ограничение, которое заключается в том, что степень конца связи должна равняться единице, соответствующий (возможно, составной) столбец должен быть дополнительно специфицирован как возможный ключ таблицы (в случае использования языка SQL для этого служит спецификация UNIQUE).

Для поддержки связи «многие ко многим» между типами сущности A и B создается дополнительная таблица AB с двумя столбцами, один из которых содержит уникальные идентификаторы экземпляров сущности A, а другой — уникальные идентификаторы экземпляров сущности B. Используя таблицы A, B и AB, с помощью стандартных реляционных операций можно найти все пары экземпляров типов сущности, участвующих в данной связи.

Индексы создаются для первичного ключа (уникальный индекс), внешних ключей и тех атрибутов, на которых предполагается в основном базировать запросы.

Если в концептуальной схеме (ER–диаграмме) присутствуют подтипы, то возможны два способа их представления в реляционной схеме:

- собрать все подтипы в одной таблице;
- для каждого подтипа образовать отдельную таблицу.

При применении первого способа таблица создается для максимального супертипа (типа сущности, не являющегося подтипом), а для подтипов могут создаваться представления. Таблица содержит столбцы, соответствующие каждому атрибуту (и связям) каждого подтипа. В таблицу добавляется, по крайней мере, один столбец, содержащий код ТИПА; он

становится частью первичного ключа. Для каждой строки таблицы значение этого столбца определяет тип сущности, экземпляру которого соответствует строка. Столбцы этой строки, которые соответствуют атрибутам и связям, отсутствующим в данном типе сущности, должны содержать неопределенные значения.

При использовании второго метода для каждого подтипа первого уровня (для более глубоких уровней применяется первый метод) супертип воссоздается с помощью представления UNION (из всех таблиц подтипов выбираются общие столбцы — столбцы супертипа).

У каждого способа есть свои достоинства и недостатки. К достоинствам первого способа (одна таблица для супертипа и всех его подтипов) можно отнести следующее:

- соответствие логике супертипов и подтипов; поскольку любой экземпляр любого подтипа является экземпляром супертипа, логично хранить вместе все строки, соответствующие экземплярам супертипа;
- обеспечение простого доступа к экземплярам супертипа и не слишком сложный доступ к экземплярам подтипов;
- возможность обойтись небольшим числом таблиц.

Недостатки первого метода:

- прикладная программа, имеющая дело с одной таблицей супертипа, должна включать дополнительную логику работы с разными наборами столбцов (в зависимости от значения столбца ТИП) и разными ограничениями целостности (в зависимости от особенностей связей, определенных для подтипа);
- общая для всех подтипов таблица потенциально может стать узким местом при многопользовательском доступе по причине возможности блокировки таблицы целиком;
- для индивидуальных столбцов подтипов должна

допускаться возможность содержать неопределенные значения; таким образом, потенциально в общей таблице будет содержаться много неопределенных значений, что при использовании некоторых реляционных СУБД может потребовать значительного объема внешней памяти.

Достоинства метода второго состоят в следующем:

- действуют более понятные правила работы с подтипами (каждому подтипу соответствует одноименная таблица);
- упрощается логика приложений; каждая программа работает только с нужной таблицей.

Недостатки второго метода:

- в общем случае требуется слишком много отдельных таблиц;
- работа с экземплярами супертипа на основе представления, объединяющего таблицы супертипов, может оказаться недостаточно эффективной;
- поскольку множество экземпляров супертипа является объединением множеств экземпляров подтипов, не все реляционные СУБД могут обеспечить выполнение операций модификации экземпляров супертипа.

Существуют два способа формирования схемы реляционной БД при наличии взаимно исключающих связей (имеются в виду связи «один ко многим», причем конец связи «многие» находится на стороне сущности, для которой связи являются взаимно исключающими):

- общее хранение внешних ключей;
- раздельное хранение внешних ключей.

Если имеются взаимно исключающие связи, то в таблице, соответствующей сущности, для которой связи являются взаимно исключающими, необходимо хранить внешние ключи. Если внешние ключи всех потенциально связанных таблиц

имеют общий формат, то можно применить первый способ, т. е. создать два столбца: идентификатор связи и идентификатор сущности (возможно, составной). Столбец идентификатора связи используется для различения связей, покрываемых дугой исключения. В столбце (столбцах) идентификатора сущности хранятся значения уникального идентификатора сущности на дальнем конце соответствующей связи.

Если результирующие внешние ключи не относятся к одному домену, то приходится прибегать к использованию второго способа, т. е. создавать для каждой связи, покрываемой дугой исключения, явные столбцы внешних ключей; все эти столбцы могут содержать неопределенные значения.

Преимущество первого подхода состоит в том, что в таблице, соответствующей сущности, появляется всего два дополнительных столбца. Очевидным недостатком является усложнение выполнения операции соединения: чтобы воспользоваться для соединения одной из альтернативных связей, нужно сначала произвести ограничение таблицы в соответствии с нужным значением столбца, содержащего идентификаторы связей.

При использовании второго подхода соединения являются явными (и естественными). Недостаток состоит в том, что требуется иметь столько столбцов, сколько имеется альтернативных связей. Кроме того, в каждом из таких столбцов будет содержаться много неопределенных значений, хранение которых потенциально может привести к серьезным накладным расходам внешней памяти.

Функция обратной инженерии представляет собой генерацию модели по БД и синхронизацию модели с БД.

2.4. Контрольные вопросы

1. Какая информация о предметной области необходима для построения ER–модели?
2. Каким правилам подчиняется ER–диаграмма?
3. Как преобразуются простые типы сущностей в схему БД?

4. Как преобразуются связи в схему БД?
5. Как преобразуются связи «многие ко многим» в схему БД?
6. Как преобразуются атрибуты сущности в схему БД?
7. Как преобразуются подтипы в схему БД?
8. Как преобразуются взаимно исключающие связи в схему БД?
9. Как преобразуются уникальные идентификаторы сущностей в схему БД?

Список литературы

1. Килина А.А. Открытые системы дистанционного образования: учеб. пособие / А.А. Килина, Е.Д. Федорков. Воронеж: ГОУВПО «Воронежский государственный технический университет, 2008. 97 с.
2. Маглинец Ю.А. Анализ требований к автоматизированным информационным системам. Интернет-университет информационных технологий - ИНТУИТ.ру, БИНОМ. Лаборатория знаний, 2008 г. 200 стр.
3. Кузнецов С.Д. Основы баз данных. Интернет-университет информационных технологий - ИНТУИТ.ру, 2005 г., 488 стр.

Выходные данные

АРХИТЕКТУРА ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к выполнению лабораторных работ по дисциплине «Архитектура информационных» для студентов направления 09.03.02 «Информационные системы и технологии» всех форм обучения

Составитель:
Филимонова Анастасия Анатольевна

Компьютерный набор А.А. Филимоновой

Подписано к изданию _____.
Уч.-изд. л. _____.

ГОУВПО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14