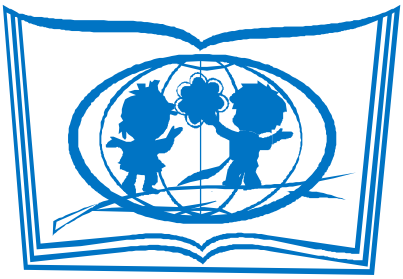
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МИНИСТЕРСТВО ОБРАЗОВАНИЯ СТАВРОПОЛЬСКОГО КРАЯ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
СТАВРОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ПЕДАГОГИЧЕСКИЙ ИНСТИТУТ



В.С. Тоискин, В.В. Красильников, В.В. Малиатаки

АВТОМАТИЗАЦИЯ ПРОЦЕССОВ ПРОЕКТИРОВАНИЯ НА ОСНОВЕ CASE ТЕХНОЛОГИЙ

Учебное пособие

Ставрополь, 2010

УДК 681/518(075.8)
ББК 32.813я73
Т 50

Печатается по решению
редакционно-издательского совета
Ставропольского государственного
педагогического института

Рецензенты:
доктор технических наук, доцент ***И.А.Калмыков*** (ГОУ ВПО Сев-КавГТУ)

кандидат педагогических наук, заместитель начальника
Ставропольского военного института связи Ракетных войск ***Г.Г. Агаджанов***

Тоискин В.С., Красильников В.В., Малиатаки В.В.

Т 50 **Автоматизация процессов проектирования на основе CASE технологий: Учебное пособие**. – Ставрополь: Изд-во СГПИ, 2010. – 131 с.

В учебном пособии раскрываются особенности современных методов и средств проектирования информационных систем, основанных на использовании CASE-технологии, приведено содержание основных понятий методологий структурного анализа и моделирования IDEF0 и IDEF1X.

Описаны правила построения диаграмм и моделей. Рассмотрены приемы работы в инструментальных средах BPwin и Erwin. На конкретных примерах описаны процессы создания моделей.

Предназначено для студентов специальности «Прикладная информатика в экономике». Кроме того, материал может быть использован при проведении факультативного курса для студентов специальности «Учитель информатики».

УДК 681/518(075.8)
ББК 32.813я73

ВВЕДЕНИЕ

Постоянное усложнение программных проектов и увеличение объема работ при их разработке обусловило потребность в универсальном средстве, позволяющем структурировать, упорядочить и даже автоматизировать процесс создания программного обеспечения.

Обеспечить поддержку наиболее трудоемких этапов разработки программного обеспечения – этапов анализа и проектирования – позволяет использование CASE-средств.

Термин CASE используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных информационных систем (ИС) в целом.

CASE-технологии являются естественным продолжением эволюции всей отрасли разработки ПО. Появлению CASE-технологии и CASE-средств предшествовали исследования в области методологии программирования.

Программирование обрело черты системного подхода с разработкой и внедрением языков высокого уровня, методов структурного и модульного программирования, языков проектирования и средств их поддержки, формальных и неформальных языков описаний системных требований и спецификаций и т.д. Кроме того, появлению CASE-технологии способствовали и такие факторы, как:

подготовка аналитиков и программистов, восприимчивых к концепциям модульного и структурного программирования;

широкое внедрение и постоянный рост производительности компьютеров, позволившие использовать эффективные графические средства и автоматизировать большинство этапов проектирования;

внедрение сетевой технологии, предоставившей возможность объединения усилий отдельных исполнителей в единый процесс проектирования путем использования разделяемой базы данных, содержащей необходимую информацию о проекте.

В настоящее время под термином CASE-средства (Computer-Aided Software/System Engineering) понимаются средства, реализующие технологию анализа, проектирования, разработки и сопровождения сложных систем про-

граммного обеспечения на основе взаимоувязанных программных продуктов; набор инструментов и методов для проектирования программного обеспечения, который помогает обеспечить высокое качество разрабатываемых программ, оптимизацию их структуры.

То есть CASE-средства – это программные средства, поддерживающие методологию проектирования информационных систем, а также набор инструментальных средств, поддерживающих процессы создания и сопровождения информационных систем, моделирование предметной области, включая анализ и формулировку требований, проектирование процессов и баз данных, генерацию кода, тестирование, документирование, обеспечение качества, конфигурационное управление и управление проектом.

Использование CASE-технологий позволяет:

– улучшить качество создаваемого программного обеспечения за счет средств автоматического контроля;

– ускорить процесс проектирования и разработки;

– обеспечить поддержку сопровождения разработки;

– обеспечить поддержку технологий повторного использования.

Большинство существующих CASE-средств основано на парадигме методология/метод/нотация/средство и методологиях структурного или объектно-ориентированного анализа и проектирования, использующих спецификации в виде диаграмм или текстов, связей между моделями системы, динамики поведения системы и архитектуры программных средств.

К CASE-средствам принято относить любое программное средство, автоматизирующее ту или иную совокупность процессов жизненного цикла ПО и обладающее следующими характерными особенностями:

– наличие мощных графических средств, используемых для описания и документирования ИС и обеспечивающих удобный интерфейс с разработчиком;

– интеграция отдельных компонентов CASE-средств с целью обеспечения управляемости процессом разработки ИС;

– использование репозитория – специальным образом организованного хранилища проектных метаданных.

В состав интегрированного CASE-средства входят следующие элементы:

– репозиторий, позволяет обеспечить сохранность вариантов проекта и его определенных компонентов, синхронизацию информации от разных разработчиков в процессе групповой разработки, проверка метаданных на полноту и непротиворечивость;

– средства разработки приложений, с использованием языков 4GL и генераторов кодов;

– средства тестирования;

– средства документирования;

– графические средства анализа и проектирования, которые дают возможность создавать и редактировать иерархически связанные диаграммы (DFD, ER-диаграмма и др.), создающие модели информационных систем;

– средства реинжиниринга.

– средства конфигурационного управления;

– средства управления проектом.

При реализации программы интегрированной компьютеризации производства (ICAM) были разработаны принципы повышения эффективности производства за счет внедрения компьютерных технологий.

В соответствии с проектом ICAM было разработано семейство методологий IDEF (ICAM DEFinition), состоящее из ряда самостоятельных методологий моделирования различных аспектов функционирования производственной среды или системы.

Все современные CASE-средства могут быть классифицированы в основном по типам и категориям. Классификация по типам отражает функциональную ориентацию CASE-средств на процессы при проектировании информационных систем. Классификация по категориям определяет степень интегрированности по выполняемым функциям.

В настоящее время существует классификация CASE-средств по следующим признакам:

– по типам – определяет функциональную ориентацию CASE-средств на какие-либо процессы жизненного цикла;

– по категориям – определяет уровень интегрированности по выполняемым функциям (локальные средства, комплект частично интегрированных средств, полностью интегрированные средства);

– по степени интегрированности с системами управления базами данных (СУБД);

– по доступным платформам;

– по применяемым методологиям и моделям систем и БД.

Классифицируя CASE-средства по возможностям можно выделить их следующие типы:

1. Верхние CASE–системы (Upper CASE) – средства анализа, которые используются для построения и анализа моделей предметной области (BPwin Logic Works). Эти системы соответствуют основным понятиям термина CASE, поэтому их также называют нормальными.

2. Средние CASE–системы (Middle CASE) – средства анализа и проектирования, которые поддерживают распространенные методологии проектирования и используются для создания проектных спецификаций (Vantage Team Builder, Designer/2000). Обеспечивают формирование архитектуры системы, составляющих и интерфейсов системы, алгоритмов и устройств данных.

3. Средства разработки приложений (PowerBuilder, 4GL (Uniface (Compuware)), SQL Windows, а также генераторы кодов.

4. Средства реинжиниринга, предназначенные для анализа программных кодов и схем баз данных и создания на их базе моделей и проектных спецификаций (Vantage Team Builder, ERwin).

5. Средства проектирования баз данных – средства моделирования данных, генерирование схем баз данных для распространенных систем управления базами данных (ERwin, DataBase Designer).

Именно BPwin (All Fusion Process Modeler) и ERwin (All Fusion ERwin Data Modeler) на сегодняшний день являются наиболее популярными CASE-средствами, входящими в пакет AllFusion Modeling Suite – интегрированный комплекс CASE-средств, обеспечивающий все потребности компаний-разработчиков программного обеспечения. Данный пакет служит для проектирования и анализа баз данных, бизнес-процессов и информационных систем и включает продукты: AllFusion Process Modeler (BPwin), AllFusion ERwin Data Modeler (ERwin), AllFusion Data Model Validator (инструмент для проверки структуры баз данных и создаваемых в ERwin моделей), AllFusion Model Manager (среда для работы группы проектировщиков на ERwin и BPwin), AllFusion Component Modeler (моделирования компонентов программного обеспечения и генерации объектного кода приложений на основе созданных моделей).

BPwin позволяет моделировать любые бизнес-процессы и проектировать организационную структуру с минимизацией издержек и повышением общей эффективности работы организации. Кроме того, BPwin позволяет значительно облегчить сертификацию на соответствие международному стандарту системы менеджмента качества ISO 9000. Erwin предоставляет возможность создавать многопользовательские информационные модели, автоматизировать процессы сбора и проверки данных. Erwin также позволяет моделировать практически любые

бизнес-процессы предприятия для повышения эффективности работы организации и снижения издержек. В линейку продуктов Erwin входят CASE-средства проектирования, сопровождения и документирования баз данных, функционального моделирования бизнес-процессов и проверки моделей данных.

Правильный выбор и грамотное применение CASE-средств при автоматизации процессов проектирования позволяет произвести оптимизацию информационных систем, значительно повысить их эффективность, снизить вероятность ошибок, а также сократить издержки.

В учебном пособии рассматривается общая характеристика CASE-средств, даются основные положения нотаций IDEF0, IDEF1X и описывается применение этих нотаций в CASE средствах верхнего уровня BPwin и Erwin.

1. ОБЩАЯ ХАРАКТЕРИСТИКА CASE-ТЕХНОЛОГИЙ

Широкое использование вычислительной техники в различных сферах деятельности человека привело к потребности создания соответствующего программного обеспечения (ПО). Однако трудоемкость и наукоемкость разработки программ настолько огромны, что ведутся работы по созданию новых технологий автоматизации проектирования программных средств. Это направление получило название CASE-технология (Computer – Aided Software Engineering, т.е. разработка ПО с помощью компьютера).

В настоящее время под термином CASE-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.

CASE-технологии представляют собой совокупность методологий и инструментарий аналитиков, разработчиков и программистов, предназначенный для автоматизации процессов проектирования и сопровождения ИС на всем ее жизненном цикле.

1.1 Методология CASE-технологий

Основополагающей концепцией является построение логической (не физической) модели системы при помощи графических методов, которые дали бы возможность пользователям, аналитикам и проектировщикам получить ясную и общую картину системы, уяснить как сочетаются между собой компоненты системы и как будут удовлетворены потребности пользователя.

Эта методология предполагает построение системы сверху вниз за счет последовательной детализации: вначале получают диаграмму потоков данных всей системы, далее разрабатывают детализированные диаграммы потоков данных, затем определяют детали структур данных и логики процессов, вслед за этим переходят к проектированию модульной структуры и т.д..

С самого начала CASE-технологии развивались с целью преодоления ограничений ручных применений методологий структурного анализа и проектиро-

вания 60-70-х годов (сложности понимания, большой трудоемкости и стоимости использования, трудности внесения изменений в проектные спецификации и т.д.) за счет их автоматизации и интеграции поддерживающих средств. Таким образом, CASE-технологии не могут считаться самостоятельными методологиями, они только делают более эффективными пути их применения.

Традиционно выделяют шесть периодов, качественно отличающихся применяемой техникой и методами разработки ПО, которые характеризуются использованием в качестве инструментальных средств:

1. ассемблеров, дампов памяти, анализаторов;
2. компиляторов, интерпретаторов, трассировщиков;
3. символьных отладчиков, пакетов программ;
4. систем анализа и управления исходными текстами;
5. CASE-средств анализа требований, проектирования спецификаций и структуры, редактирования интерфейсов (первая генерация CASE-I);
6. CASE-средств генерации исходных текстов и реализации интегрированного окружения поддержки полного жизненного цикла разработки ПО (вторая генерация CASE-II)

CASE-I является первой технологией, адресованной непосредственно системным аналитикам и проектировщикам, и включающей средства для поддержки графических моделей, проектирования спецификаций, экранных редакторов и словарей данных. Она не предназначена для поддержки полного жизненного цикла и концентрирует внимание на функциональных спецификациях и начальных шагах проекта – системном анализе, определении требований, системном проектировании, логическом проектировании БД.

CASE-II отличается значительно более развитыми возможностями, улучшенными характеристиками и исчерпывающим подходом к полному жизненному циклу. В ней, в первую очередь, используются средства поддержки автоматической кодогенерации, а также, обеспечивается полная функциональная поддержка для выполнения графических системных требований и спецификаций проектирования; контроля, анализа и связывания системной информации и информации по управлению проектированием; построение прототипов и моделей системы; тестирования, верификации и анализа сгенерированных программ; генерации документов по проекту; контроля на соответствие стандартам по всем этапам жизненного цикла. CASE-II может включать свыше 100 функциональных компонент, поддерживающих все этапы жизненного цикла, при этом пользователям предоставляется возможность выбора необходимых средств и их интеграции в нужном составе.

Таким образом, CASE-средства являются результатом естественного эволюционного развития отрасли инструментальных (или технологических) средств. CASE-технологии начали развиваться с целью преодоления ограничений методологии структурного программирования.

Эта методология, несмотря на формализацию в составлении программ, характеризуется все же сложностью понимания, большой трудоемкостью и стоимостью использования, трудностью внесения изменений в проектные спецификации. Однако заложенные в ней принципы позволили развивать эту методологию и повысить ее эффективность за счет автоматизации наиболее рутинных этапов (см. табл. 1.1.).

Таблица 1.1.

Сравнение традиционной разработки ПО и разработки с использованием CASE-технологий

№ п/п	Традиционная разработка	CASE-технология
1.	Основные усилия – на кодирование и тестирование	Основные усилия – на анализ и проектирование
2.	«Бумажные» спецификации	Быстрое интерактивное прототипирование
3.	Ручное кодирование	Автоматическая кодогенерация
4.	Ручное документирование	Автоматическая генерация документации
5.	Тестирование кодов	Автоматический контроль проекта
6.	Сопровождение кодов	Сопровождение спецификаций проектирования

CASE-технология базируется на методологии системного анализа. Под системным анализом понимают научную дисциплину, разрабатывающую общие принципы исследования сложных объектов и процессов с учетом их системного характера. Его основная цель – сосредоточить внимание на начальных этапах разработки. В рамках CASE-технологии системный анализ предназначен для отделения проектирования от программирования. В разработке в соответствии с CASE-технологией выделяются построение архитектуры и ее последующая реализация, поэтому системный анализ называют структурным системным анализом или просто структурным анализом.

Важнейшими (базовыми) принципами являются деление (декомпозиция) и последующее иерархическое упорядочение. Они дополняются следующими принципами.

Принцип абстрагирования от несущественных деталей (с их «упрятыванием») с контролем на присутствие лишних элементов.

Принцип формализации.

Принцип концептуальной общности (структурный анализ – структурное программирование – структурное тестирование). Отсюда методология структурного анализа – метод исследования от общего обзора через детализацию к иерархической структуре со все большим числом уровней.

Принцип непротиворечивости – обоснование и согласованность элементов.

Принцип логической и физической независимости данных.

Принцип непосредственного доступа (без программирования) конечного пользователя.

Основными **методологиями**, реализованными в CASE – средствах являются:

SADT (Structured Analysis and Design Technique) – методология структурного анализа и проектирования. Основана на понятиях функционального моделирования. Отражает такие системные характеристики, как управление, обратная связь и исполнитель.

IDEF0 (Integrated Definition Function Modeling) – методология функционального моделирования. Используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, преобразуемые этими функциями. Является подмножеством методологии SADT.

DFD (DataFlow Diagram) – методология моделирования потоков данных. Применяется для описания обмена данными между рабочими процессами.

IDEF1 применяется для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы.

IDEF2 позволяет построить динамическую модель меняющихся во времени поведения функций, информации и ресурсов системы.

IDEF3 – методология моделирования потоков работ. Является более детальной по отношению к IDEF0 и DFD. Позволяет рассмотреть конкретный процесс с учетом последовательности выполняемых операций. С помощью IDEF3 описываются сценарий и последовательность операций для каждого процесса.

IDEF1X (IDEF1 Extended) – методология описания данных. Применяется для построения баз данных. Относится к типу методологий «Сущность-связь» (ER

– Entity-Relationship) и, как правило, используется для моделирования реляционных баз данных, имеющих отношение к рассматриваемой системе.

IDEF4 – объектно-ориентированная методология. Отражает взаимодействие объектов. Позволяет наглядно отображать структуру объектов и заложенные принципы их взаимодействия. Удобна для создания программных продуктов на объектно-ориентированных языках.

IDEF5 – методология онтологического исследования сложных систем. С помощью методологии IDEF5 онтология системы может быть описана при помощи определенного словаря терминов и правил, на основании которых могут быть сформированы достоверные утверждения о состоянии рассматриваемой системы в некоторый момент времени.

ARIS – описывает бизнес-процесс в виде потока последовательно выполняемых работ.

UML – (Unified Modeling Language) язык визуального моделирования, основанный на объектно-ориентированном подходе. UML позволяют описать статическую структуру системы и ее динамическое поведение.

CASE-технологии предлагают новый, основанный на автоматизации подход к концепции жизненного цикла ПО. При использовании CASE изменяются все фазы жизненного цикла, при этом наибольшие изменения касаются фаз анализа и проектирования. На рис. 1.1. приводится простейшая модель жизненного цикла (рис. 1.1.а) и соответствующая ей CASE-модель (рис.1.1.б), в которой фаза прототипирования заменяет традиционную фазу системного анализа. Необходимо отметить, что наиболее автоматизируемыми фазами являются фазы контроля проекта и кодогенерации (хотя все остальные фазы также поддерживаются CASE-средствами).

В основе концептуального построения CASE-средств лежат следующие положения:

Человеческий фактор, определяющий разработку ПО как легкий, удобный и экономичный процесс.

Широкое использование базовых программных средств, получивших массовое распространение в других приложениях (БД и СУБД, компиляторы с различных языков программирования, отладчики, документаторы, издательские системы, оболочки экспертных систем и базы знаний, языки четвертого поколения и др.).

Автоматизированная или автоматическая кодогенерация, выполняющая несколько видов генерации кодов: преобразования для получения документации, формирования БД, ввода/модификации данных, получения выполняемых ма-

шинных кодов из спецификаций ПО, автоматической сборки модулей из словарей и моделей данных и повторно используемых программ, автоматической конверсии ранее используемых файлов в форматы новых требований.

Рисунок 1.1. *– Модель жизненного цикла ПО и соответствующая ей CASE-модель*

Ограничение сложности, позволяющее получать компоненты, поддающиеся управлению, обозримые и доступные для понимания, а также обладающие простой и ясной структурой.

Доступность для разных категорий пользователей.

Рентабельность.

Сопровождаемость, обеспечивающая способность адаптации при изменении требований и целей проекта.

1.2. Структура CASE-средств

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).

В связи с наличием двух подходов к проектированию программного обеспечения существуют CASE-технологии ориентированные на структурный подход, объектно-ориентированный подход, а также комбинированные. Однако сейчас наблюдается тенденция переориентации инструментальных средств, созданных для структурных методов разработки, на объектно-ориентированные методы, что объясняется следующими причинами:

– возможностью сборки программной системы из готовых компонентов, которые можно использовать повторно;

– возможностью накопления проектных решений в виде библиотек классов на основе механизмов наследования;

– простотой внесения изменений в проекты за счет инкапсуляции данных в объектах;

– быстрой адаптацией приложений к изменяющимся условиям за счет использования свойств наследования и полиморфизма;

– возможностью организации параллельной работы аналитиков, проектировщиков и программистов.

– Объектно-ориентированное CASE-средство в идеале должно содержать четыре основных блока: анализ, проектирование, разработка и инфраструктура.

Основные требования к блоку анализа:

– возможность выбора выводимой на экран информации из всей совокупности данных, описывающих модели;

– согласованность диаграмм при хранении их в репозитарии;

– внесение комментариев в диаграммы и соответствующую документацию для фиксации проектных решений;

– возможность динамического моделирования в терминах событий;

– поддержка нескольких нотаций (хотя бы три нотации – Г.Буча, И. Джекобсона и OMT).

Основные требования к блоку проектирования:

– поддержка всего процесса проектирования приложения;

– возможность работы с библиотеками, средствами поиска и выбора;

– возможность разработки пользовательского интерфейса;

– поддержка стандартов OLE, ActiveX и доступ к библиотекам HTML или Java;

– поддержка разработки распределенных или двух- и трехзвенных клиент-серверных систем (работа с CORBA, DCOM, Internet).

Основные требования к блоку реализации:

– генерация кода полностью из диаграмм;

– возможность доработки приложений в клиент-серверных CASE-средствах типа Power Builder;

– реинжиниринг кодов и внесение соответствующих изменений в модель системы;

– наличие средств контроля, которые позволяют выявлять несоответствие между диаграммами и генерируемыми кодами и обнаруживать ошибки как на стадии проектирования, так и на стадии реализации.

Основные требования к блоку инфраструктуры:

– наличие репозитория на основе базы данных, отвечающего за генерацию кода, реинжиниринг, отображение кода на диаграммах, а также обеспечивающего соответствие между моделями и программными кодами;

– обеспечение командной работы (многопользовательской работы и управление версиями) и реинжиниринга.

В составе интегрированного CASE-средства (или комплекса средств, поддерживающих полный жизненный цикл ПО) содержатся следующие **компоненты**:

– репозиторий, являющийся основой CASE-средства. Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль метаданных на полноту и непротиворечивость;

– графические средства анализа и проектирования, обеспечивающие создание и редактирование иерархически связанных диаграмм (DFD, ERD и др.), образующих модели ИС;

– средства разработки приложений, включая языки 4GL и генераторы кодов;

– средства конфигурационного управления;

– средства документирования;

– средства тестирования;

– средства управления проектом;

– средства реинжиниринга.

1.3. Классификация CASE-средств

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

В настоящее время на рынке существует огромное количество CASE-пакетов, при этом все CASE-средства делятся на типы, категории и уровни.

Классификация по типам отражает функциональную ориентацию CASE-средств в технологическом процессе.

Анализ и проектирование. Средства этой группы используются для создания спецификаций системы и ее проектирования; они поддерживают широко известные методологии проектирования. К таким средствам относятся: The Developer (Asyst Technologies), Design Generator (Computer Sciences), Pose (Computer Systems Advises), Analist/ Designer (Jour- don)...

Проектирование баз данных и файлов. Средства обеспечивают логическое моделирование данных, генерацию схем БД и описание форматов файлов: Idef/Leverage (D.Appleton), Chen Toolkit (Chen & Associates), Case+Designer (Orale).

Программирование. Средства поддерживают шаги программирования и тестирования, а также автоматическую кодогенерацию из спецификаций, получая полностью документированную выполняемую программу: Cobol 2/ Workbench (Miero Focus), Decase (DEC), Netron/Cap (Netron)...

Сопровождение и реинженерия. К таким средствам относятся документаторы, анализаторы программ, средства реструктурирования и обратной инженерии: Adpac Case Tools (Adpac), Superstructure (Computer Data Systems)...

Окружение. Средства поддерживающие платформы для интеграции, создания и придания товарного вида CASE-средствам: Multi/ Cam (AGS Management Systems), Sylvia Foondey (Codmare).

Управление проектом. Средства поддерживающие планирование, контроль, руководство, взаимодействие, то есть функции, необходимые в процессе разработки и сопровождения проектов: Projekt Workbench (App- lied Business Technology).

Классификация по категориям определяет уровень интегрированности по выполняемым функциям и включает:

– Вспомогательные программы (Tools), решающие небольшую автономную задачу, принадлежащую проблеме более широкого масштаба.

– Пакеты разработки (Toolkit), представляющие собой совокупность интегрированных средств, обеспечивающих помощь для одного из классов программных задач.

– Инструментальные средства (Workbench) по сравнению с Toolkit обладает более высокой степенью интеграции выполняемых функций, большей самостоятельностью и автономностью использования, а также наличием тесной связи с системными и техническими средствами аппаратно-вычислительной среды, на которой Workbench функционирует. Workbench – это автоматизированная рабочая стадия, используемая как инструментарий для автоматизации всех или отдельных совокупностей работ по созданию ПО АС.

Классификация по уровням связана с областью действия CASE в пределах жизненного цикла ПО.

– Верхние (Upper) CASE часто называют компьютерным планированием. Использование верхних CASE позволяет построить модель ПрО, отражающую всю существующую специфику. Она направлена на понимание общего и частного механизмов функционирования, имеющихся возможностей, ресурсов, целей проекта в соответствии с назначением фирмы. Эти средства позволяют проводить анализ различных сценариев, накапливая информацию для принятия оптимальных решений.

– Средние (Middle) CASE считаются средствами поддержки этапов анализа требований и проектирования спецификаций и структуры АС. Основная выгода от использования среднего CASE состоит в значительном облегчении проектирования систем; проектирование превращается в итеративный процесс, включающий действия: пользователь обсуждает с аналитиком требования к информации; аналитик документирует эти требования, используя диаграммы и словари входных данных; пользователь проверяет эти диаграммы и словари, при необходимости модифицируя их; аналитик отвечает на эти модификации изменяя соответствующие спецификации. Кроме того, средние CASE обеспечивают возможности быстрого документирования требований и прототипирования.

– Нижние (Lower) CASE поддерживают системы разработки ПО АС (при этом может использоваться до 30% спецификаций, созданных средствами среднего CASE). Они содержат системные словари и графические средства, исключающие необходимость разработки физических спецификаций – имеются системные спецификации, которые непосредственно переводятся в программные коды разрабатываемой системы (при этом автоматически генерируется до 80% кодов). Главными преимуществами нижних CASE является: значительное

уменьшение времени на разработку, облегчение модификаций, поддержка возможностей прототипирования (совместно со средними CASE).

Помимо этого, CASE-средства можно классифицировать по следующим признакам:

- применяемым методологиям и моделям систем и БД;
- степени интегрированности с СУБД;
- доступным платформам.

1.4. Результаты использования CASE-средств

Ввиду разнообразной природы CASE-средств было бы ошибочно делать какие-либо утверждения относительно реального удовлетворения тех или иных ожиданий от их внедрения. Можно перечислить следующие факторы, усложняющие определение возможного эффекта от использования CASE-средств:

- широкое разнообразие качества и возможностей CASE-средств;
- относительно небольшое время использования CASE-средств в различных организациях и недостаток опыта их применения;
- широкое разнообразие в практике внедрения различных организаций;
- отсутствие детальных метрик и данных для уже выполненных и текущих проектов;
- широкий диапазон предметных областей проектов;
- различная степень интеграции CASE-средств в различных проектах.

К достоинствам CASE-технологий можно отнести следующие:

1. Единый графический язык.

CASE-технологии обеспечивают всех участников проекта, включая заказчиков, единым строгим, наглядным и интуитивно понятным графическим языком, позволяющим получать обозримые компоненты с простой и ясной структурой. При этом программы представляются двумерными схемами (которые проще в использовании, чем многостраничные описания), позволяющими заказчику участвовать в процессе разработки, а разработчикам – общаться с экспертами предметной области, разделять деятельность системных аналитиков, проектировщиков и программистов, облегчая им защиту проекта перед руководством, а также обеспечивая легкость сопровождения и внесения изменений в систему.

2. Единая БД проекта.

Основа CASE-технологии – использование базы данных проекта (репозитория) для хранения всей информации о проекте, которая может разделяться ме-

жду разработчиками в соответствии с их правами доступа. Содержимое репозитория включает не только информационные объекты различных типов, но и отношения между их компонентами, а также правила использования или обработки этих компонентов. Репозиторий может хранить свыше 100 типов объектов: структурные диаграммы, определения экранов и меню, проекты отчетов, описания данных, логика обработки, модели данных, их организации и обработки, исходные коды, элементы данных и т. п.

3. Интеграция средств. На основе репозитория осуществляется интеграция CASE-средств и разделение системной информации между разработчиками. При этом возможности репозитория обеспечивают несколько уровней интеграции: общий пользовательский интерфейс по всем средствам, передачу данных между средствами, интеграцию этапов разработки через единую систему представления фаз жизненного цикла, передачу данных и средств между различными платформами.

4. Поддержка коллективной разработки и управления проектом.

CASE-технология поддерживает групповую работу над проектом, обеспечивая возможность работы в сети, экспорт-импорт любых фрагментов проекта для их развития и/или модификации, а также планирование, контроль, руководство и взаимодействие, т. е. Функции, необходимые в процессе разработки и сопровождения проектов. Эти функции также реализуются на основе репозитория. В частности, через репозиторий может осуществляться контроль безопасности (ограничения и привилегии доступа), контроль версий и изменений и др.

5. Макетирование.

CASE-технология дает возможность быстро строить макеты (прототипы) будущей системы, что позволяет заказчику на ранних этапах разработки оценить, насколько она приемлема для будущих пользователей и устраивает его.

6. Генерация документации.

Вся документация по проекту генерируется автоматически на базе репозитория (как правило, в соответствии с требованиями действующих стандартов). Несомненное достоинство CASE-технологии заключается в том, что документация всегда отвечает текущему состоянию дел, поскольку любые изменения в проекте автоматически отражаются в репозитории (известно, что при традиционных подходах к разработке ПО документация в лучшем случае запаздывает, а ряд модификаций вообще не находит в ней отражения).

7. Верификация проекта.

CASE-технология обеспечивает автоматическую верификацию и контроль проекта на полноту и состоятельность на ранних этапах разработки, что влияет

на успех разработки в целом – по статистическим данным анализа пяти крупных проектов фирмы TRW (США) ошибки проектирования и кодирования составляют соответственно 64% и 32% от общего числа ошибок, а ошибки проектирования в 100 раз труднее обнаружить на этапе сопровождения ПО, чем на этапе анализа требований.

8. Автоматическая генерация объектного кода.

Генерация программ в машинном коде осуществляется на основе репозитория и позволяет автоматически построить до 85-90% объектного кода или текстов на языках высокого уровня.

9. Сопровождение и реинжиниринг.

Сопровождение системы в рамках CASE-технологии характеризуется сопровождением проекта, а не программных кодов. Средства реинжиниринга и обратного инжиниринга позволяют создавать модель системы из ее кодов и интегрировать полученные модели в проект, автоматически обновлять документацию при изменении кодов и т. п.

Использование CASE-технологии способно принести следующие выгоды:

– высокий уровень технологической поддержки процессов разработки и сопровождения ПО;

– положительное воздействие на некоторые или все из перечисленных факторов: производительность, качество продукции, соблюдение стандартов, документирование;

– приемлемый уровень отдачи от инвестиций в CASE-средства.

Практически невозможно, чтобы в процессе одного внедрения CASE-средств были достигнуты все положительные результаты. Тем не менее любая организация может выработать собственные идеи относительно ожидаемых результатов:

Реалистичные ожидания:

– ускорение и повышение согласованности разработки приложений;

– снижение доли ручного труда в процессе разработки и/или эксплуатации;

– более точное соответствие приложений требованиям пользователей;

– отсутствие необходимости большой переделки приложений для повышения их эффективности;

– улучшение реакции службы эксплуатации на требования внесения изменений и усовершенствований;

– лучшее документирование;

– улучшение коммуникации между пользователями и разработчиками;

– последовательное и постоянное повышение качества проектирования;

– более высокие возможности повторного использования разработок;

– лучшая прогнозируемость затрат.

Нереалистичные ожидания:

– отсутствие воздействия на общую культуру и распределение ролей в организации;

– понимание проектных спецификаций неподготовленными пользователями;

– сокращение персонала, связанного с информационной технологией;

– уменьшение степени участия в проектах высшего руководства и менеджеров, а также экспертов предметной области, уменьшение степени участия пользователей в процессе разработки приложений;

– немедленное повышение продуктивности деятельности организации;

– достижение абсолютной полноты и непротиворечивости спецификаций;

– автоматическая генерация прикладных систем из проектных спецификаций;

– немедленное снижение затрат, связанных с информационной технологией;

– снижение затрат на обучение.

Реализм в оценке ожидаемых затрат имеет особенно важное значение, поскольку позволяет правильно оценить отдачу от инвестиций. Затраты на внедрение CASE-средств обычно недооцениваются, хотя конкретных статей затрат на внедрение потребуют:

– специалисты по планированию внедрения CASE-средств;

– выбор и установка средств;

– учет специфических требований персонала;

– приобретение CASE-средств и обучение;

– настройка средств;

– подготовка документации, стандартов и процедур использования средств;

– интеграция с другими средствами и существующими данными;

– освоение средств разработчиками;

– технические средства;

– обновление версий.

1.5. Критерии оценки и выбора CASE-средств

Эффективность применения CASE-технологий обеспечивается грамотным и обоснованным их выбором и продуманным использованием

К основным критериям оценки и выбора CASE-средств относятся:

– Функциональные характеристики:

– среда функционирования: проектная среда, программное обеспечение/технические средства, технологическая среда;

– функции, ориентированные на фазы жизненного цикла: моделирование, реализация, тестирование;

– общие функции: документирование, управление конфигурацией, управление проектом;

– Надежность;

– Простота использования;

– Эффективность;

– Сопровождаемость;

– Переносимость;

– Общие критерии (стоимость, затраты, эффект внедрения, характеристики поставщика).

Процесс внедрения CASE-средств состоит из следующих этапов:

1. определение потребностей в CASE-средствах;
2. оценка и выбор CASE-средств;
3. выполнение пилотного проекта;
4. практическое внедрение CASE-средств.

Процесс оценки и выбора CASE-средств может преследовать несколько целей и включать:

– оценку нескольких CASE-средств и выбор одного или более из них;

– оценку одного или более CASE-средств и сохранение результатов для последующего использования;

– выбор одного или более CASE-средств с использованием результатов предыдущих оценок.

Переход к практическому использованию CASE-средств начинается с разработки и последующей реализации плана перехода. План перехода должен включать следующее:

Информацию относительно целей, критериев оценки, графика и возможных рисков, связанных с реализацией плана.

Информацию относительно приобретения, установки и настройки CASE-средств.

Информацию относительно интеграции каждого средства с существующими средствами, включая как интеграцию CASE-средств друг другом, так и их интеграцию в процессы разработки и эксплуатации ПО, существующие в организации.

Ожидаемые потребности в обучении и ресурсы, используемые в течение и после завершения процесса перехода.

Определение стандартных процедур использования средств.

Определение потребностей в CASE-средствах можно проиллюстрировать следующей диаграммой (см. рис. 1.2.).

Рисунок 1.2. *– Определение потребностей в CASE-средствах*

Использование CASE-технологий требует постоянного мониторинга использования CASE-средств, обеспечения текущей поддержки, сопровождения и обновления средств по мере необходимости. При этом достигнутые результаты должны периодически подвергаться экспертизе в соответствии с графиком, а план перехода – корректироваться при необходимости. Кроме этого, необходимо уделять внимание удовлетворению потребностей организации и критериям успешного внедрения CASE-средств, а также обучению сотрудников.

При этом стратегия выбора CASE-средств для конкретного применения в общем случае зависит от целей, потребностей и ограничений будущего проекта (включая квалификацию участвующих в процессе проектирования специалистов), которые, в свою очередь, определяют используемые методы проектирования.

2. ОСНОВЫ МЕТОДОЛОГИИ IDEF0

В конце 70-х годов 20 века была реализована Программа интегрированной компьютеризации производства ICAM (Integrated Computer Aided Manufacturing), направленная на увеличение эффективности промышленных предприятий посредством широкого внедрения компьютерных (информационных) технологий.

Реализация программы ICAM потребовала создания адекватных методов анализа и проектирования информационных и производственных систем и способов обмена информацией между специалистами, занимающимися такими проблемами. Для удовлетворения этой потребности в рамках программы ICAM была разработана методология IDEF (ICAM Definition), позволяющая исследовать структуру, параметры и характеристики производственно-технических и организационно-экономических систем. Общая методология IDEF состоит из трех частных методологий моделирования, основанных на графическом представлении систем:

– IDEF0 используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции.

– IDEF1 применяется для построения информационной модели, отображающей структуру и содержание информационных потоков, необходимых для поддержки функций системы;

– IDEF2 позволяет построить динамическую модель меняющихся во времени поведения функций, информации и ресурсов системы.

В настоящее время наибольшее распространение и применение имеют методологии IDEF0 и IDEF1 (IDEF1X), получившие в США статус федеральных стандартов.

Методология IDEF0 основана на подходе, разработанном Дугласом Т. Россом в начале 70-х годов 20 века и получившем название SADT (Structured Analysis & Design Technique – метод структурного анализа и проектирования).

Основу подхода методологии IDEF0 составляет графический язык описания (моделирования) систем, обладающий следующими свойствами:

– графический язык – средство, способное наглядно представлять широкий спектр процессов и операций организации на любом уровне детализации;

– язык обеспечивает точное и лаконичное описание моделируемых объектов, удобство использования и интерпретации этого описания;

– язык облегчает взаимодействие и взаимопонимание разработчиков и персонала изучаемого объекта, то есть служит средством информационного общения;

– язык может генерироваться рядом инструментальных средств машинной графики.

2.1. Сущность структурного подхода

Основу параграфа составляет материал, изложенный в [5].

Сущность структурного подхода к разработке информационных систем заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны. При разработке системы «снизу-вверх» от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов.

Все наиболее распространенные методологии структурного подхода базируются на ряде общих принципов. В качестве основных принципов используются:

– принцип декомпозиции – принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;

– принцип иерархического упорядочивания – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне.

– принцип абстрагирования – заключается в выделении существенных аспектов системы и отвлечения от несущественных;

– принцип формализации – заключается в необходимости строгого методического подхода к решению проблемы;

– принцип непротиворечивости – заключается в обоснованности и согласованности элементов;

– принцип структурирования данных – заключается в том, что данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, иллюстрирующих функции, выполняемые системой и отношения между данными.

Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными из них являются следующие:

– SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;
– DFD (Data Flow Diagrams) диаграммы потоков данных;
– ERD (Entity-Relationship Diagrams) диаграммы «сущность-связь».

На стадии проектирования информационной системы (ИС) модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру программного обеспечения, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание ИС независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

2.2. Нотация IDEF0

Нотация возникла при формализации процесса создания системы, при этом предполагалось, что создание системы включает следующие этапы:

– анализ – определение того, что система будет делать;
– проектирование – определение подсистем и способов их взаимодействия;
– реализация – разработка подсистем по отдельности;
– объединение – соединение подсистем в единое целое;
– тестирование – проверка работы системы;
– установка – введение системы в действие;
– функционирование – использование системы.

Традиционные подходы к созданию систем приводили к возникновению многих проблем. Для решения ключевых проблем традиционного создания систем широкого профиля требовались новые методы, специально предназначенные для использования на ранних стадиях процесса. IDEF0 была создана и опробована на практике в период с 1969 по 1973 г. Эта методология учитывает PLEX, концепции клеточной модели, человек-ориентированные функции Хори, общую теорию систем, технологии программирования.

IDEF0 – это методология, разработанная специально для того, чтобы облегчить описание и понимание искусственных систем, попадающих в разряд средней сложности.

Методология IDEF0 основана на следующих концептуальных положениях [18].

Модель – искусственный объект, представляющий собой отображение (образ) системы и ее компонентов.

М моделирует А, если М отвечает на вопросы относительно А.

М – модель, А – моделируемый объект (оригинал).

Модель разрабатывают для понимания, анализа и принятия решений о реконструкции (реинжиниринге) или замене существующей, либо проектировании новой системы.

Одна и та же схема моделирования может быть использована для моделирования любого выбранного объекта.

Система представляет собой совокупность взаимосвязанных и взаимодействующих частей, выполняющих некоторую полезную работу. Частями (элементами) системы могут быть любые комбинации разнообразных сущностей, включающие людей, информацию, программное обеспечение, оборудование, изделия.

Модель описывает, что происходит в системе, как ею управляют, какие сущности она преобразует, какие средства использует для выполнения своих функций.

Основной концептуальный принцип методологии IDEF – представление любой изучаемой системы в виде набора взаимодействующих и взаимосвязанных блоков, отображающих процессы, операции, действия, происходящие в изучаемой системе.

В IDEF0 все, что происходит в системе и ее элементах, называют функциями. Каждой функции ставится в соответствие блок. На IDEF0 –диаграмме, основном документе при анализе и проектировании систем, блок представляет собой прямоугольник. Интерфейсы, посредством которых блок взаимодействует с другими блоками или с внешней по отношению к моделируемой системе средой, представляются стрелками**,** входящими в блок или выходящими из него. Входящие стрелки показывают, какие условия должны быть одновременно выполнены, чтобы функция, описываемая блоком, осуществилась.

Графический язык позволяет лаконично, однозначно и точно показать все элементы (блоки) системы и все отношения и связи между ними, выявить ошибочные, лишние или дублирующие связи.

Средства IDEF0 позволяют использовать унифицированные средства представления системы. К числу таких средств относятся:

– диаграммы, основанные на графике блоков и стрелок;

– метки на естественном языке для описания блоков и стрелок, а также глоссарий и сопроводительный текст для уточнения смысла элементов диаграммы;

– последовательная декомпозиция диаграмм, строящаяся по иерархическому принципу, при котором на верхнем уровне отображаются основные функции, а затем происходит их детализация и уточнение;

– древовидные схемы иерархии диаграмм и блоков, обеспечивающие обозримость модели в целом и входящих в нее деталей.

Разработка моделей IDEF0 требует соблюдения ряда строгих формальных правил, обеспечивающих преимущества методологии в отношении однозначности, точности и целостности сложных многоуровневых моделей.

Разработка модели в IDEF0 представляет собой пошаговую, итеративную процедуру. На каждом шаге итерации разработчик предлагает вариант модели, который подвергают обсуждению, рецензированию и последующему редактированию, после чего цикл повторяется. Такая организация работы способствует оптимальному использованию знаний аналитика, владеющего методологией и техникой IDEF0, и знаний специалистов – экспертов в предметной области, к которой относится объект моделирования.

При разработке моделей целесообразно не связывать функции исследуемой системы с существующей организационной структурой моделируемого объекта. Это помогает избежать субъективной точки зрения. Организационная структура должна явиться результатом использования (применения) модели. Сравнение результата с существующей структурой позволяет, во-первых, оценить адекватность модели, а во-вторых – предложить решения, направленные на совершенствование этой структуры.

Основу IDEF0 составляет, разработанная Дугласом Т. Россом методология и язык описания систем, названная SADT (Методология структурного анализа и проектирования).

2.3. Нотация модели в IDEF0

Описание системы с помощью IDEF0 называется моделью. В IDEF0-моделях используются как естественный, так и графический языки.

С точки зрения IDEF0 модель может быть сосредоточена либо на функциях системы, либо на ее объектах. IDEF0-модели, ориентированные на функции, принято называть функциональными моделями, а ориентированные на объекты системы – моделями данных. Функциональная модель представляет с требуемой степенью детализации систему функций, которые в свою очередь отражают свои взаимоотношения через объекты системы. Модели данных дуальны к функцио-

нальным моделям и представляют собой подробное описание объектов системы, связанных системными функциями. Полная методология IDEF0 поддерживает создание множества моделей для более точного описания сложной системы.

IDEF0-модель дает полное, точное и адекватное описание системы, имеющее конкретное назначение. Это назначение, называемое целью модели, вытекает из формального определения модели в IDEF0: M есть модель системы A**,** если M может быть использована для получения ответов на вопросы относительно A с некоторой точностью.

Таким образом, целью модели является получение ответов на некоторую совокупность вопросов. Эти вопросы неявно присутствуют в процессе анализа и, следовательно, они являются основой при создании модели и определяют направление моделирования. Это означает, что сама модель должна будет дать ответы на эти вопросы с заданной степенью точности. Если модель отвечает не на все вопросы или ее ответы недостаточно точны, то предполагается, что модель не достигла своей цели. Таким образом IDEF0 определяет основы практического моделирования.

Смысл и трактовка данного определения модели оказали существенное влияние на практические применения IDEF0. Обычно вопросы для IDEF0-модели формулируются на самом раннем этапе проектирования, при этом основная суть этих вопросов должна быть выражена в одной-двух фразах.

В методологии IDEF0 подчеркивается необходимость точного определения границ системы. IDEF0-модель всегда ограничивает свой субъект, то есть модель устанавливает точно, что является и что не является субъектом моделирования, описывая то, что входит в систему, и подразумевая то, что лежит за ее пределами. Ограничивая субъект, IDEF0-модель помогает сконцентрировать внимание именно на описываемой системе и позволяет избежать включения посторонних субъектов. Таким образом, IDEF0-модель должна иметь единственный субъект.

IDEF0 требует, чтобы модель рассматривалась все время с одной и той же позиции. Эта позиция называется «точкой зрения» данной модели. «Точку зрения» лучше всего представлять себе как место (позицию) человека или объекта, на которое надо встать, чтобы увидеть систему в действии.

После того как определены субъект, цель и точка зрения модели, начинается первая интеграция процесса моделирования по методологии IDEF0. Субъект определяет, что включить в модель, а что исключить из нее. Точка зрения определяет выбор нужной информации о субъекте и форму ее представления. Цель становится критерием окончания моделирования. Конечным результатом этого

процесса является набор взаимоувязанных описаний, начиная с описания самого верхнего уровня всей системы и кончая подробным описанием деталей.

Каждое из таких взаимосогласованных описаний называется диаграммой. IDEF0-модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели менее детализированы, чем диаграммы нижних уровней. Другими словами, модель IDEF0 можно представить в виде древовидной структуры диаграмм, где верхняя диаграмма является наиболее общей, а самые нижние наиболее детализированы.

2.4. Синтаксис IDEF0

Набор структурных компонентов языка, их характеристики и правила, определяющие связи между компонентами, представляют собой синтаксис языка.

Компонентами синтаксиса IDEF0 являются блоки, стрелки, диаграммы и правила [15,18].

Блоки представляют функции, определяемые как деятельность, процесс, операция, действие или преобразование.

Стрелки представляют данные или материальные объекты, связанные с функциями.

Правила определяют, как следует применять компоненты; диаграммы обеспечивают формат графического и словесного описания моделей.

Блоки на диаграммах изображаются прямоугольниками. Блок представляет функцию или активную часть системы, поэтому названиями блоков служат глаголы или глагольные обороты. Например, названиями блоков диаграммы выполнить задание являются: определить степень выполнения задания, разработать модель информационной системы, разработать Интранет-портал, разработать автоматизированное рабочее место преподавателя. В общем виде изображение блока имеет вид (рисунок 2.1)

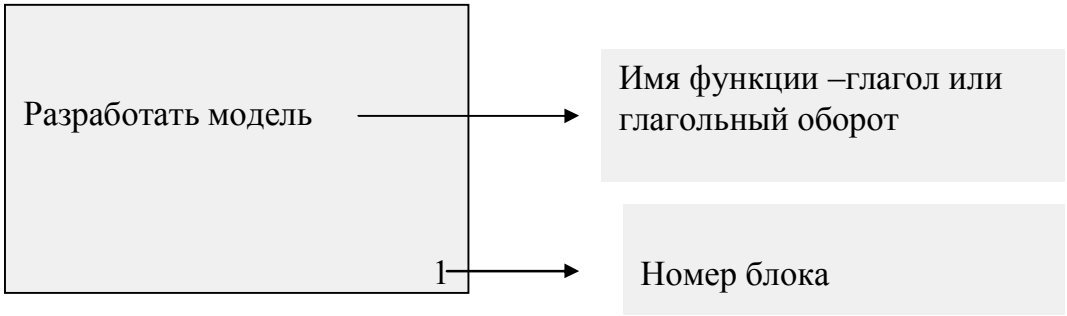


Рисунок 2.1. – Вид блока диаграммы

Кроме того, IDEF0 требует, чтобы в диаграмме было не менее трех и не более шести блоков. Эти ограничения поддерживают сложность диаграмм и модели на уровне, доступном для чтения, понимания и использования.

В IDEF0 каждая сторона блока имеет особое, вполне определенное назначение. Левая сторона блока предназначена для входов, верхняя – для управления, правая – для выходов, нижняя – для механизмов (рисунок 2.2).

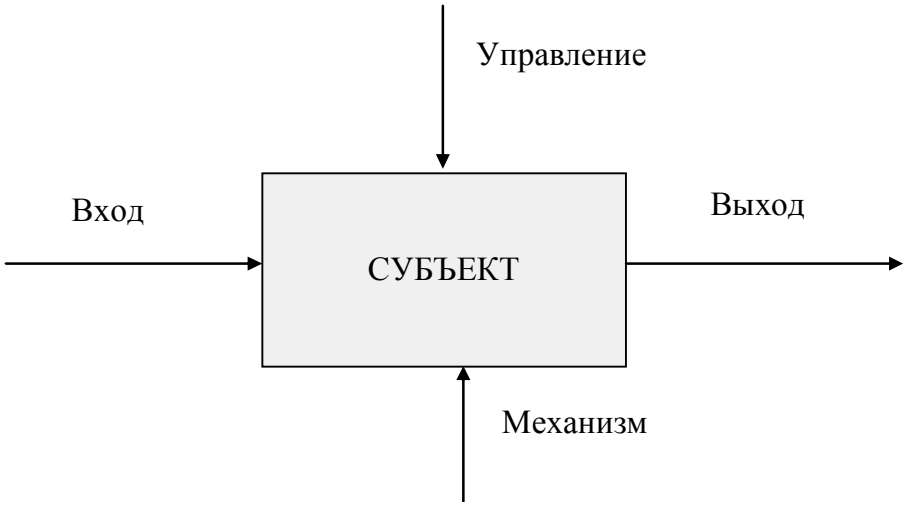


Рисунок 2.2. *– Структура универсального блока*

Вход при наличии *Управления* преобразуется в *Выход с* помощью *Механизма* (исполнителя).

Выходы одного блока могут быть входами или управлениями (или исполнителями) для других блоков. Каждый блок может быть подвергнут декомпозиции, то есть разделен как целое на свои составляющие на более детальной диаграмме. Входы, управления и выходы определяют интерфейсы между блоками, а исполнители позволяют при необходимости в определенной степени объединять объекты. Границы блоков и диаграмм должны быть согласованы, а возникающая иерархическая, взаимосвязанная совокупность диаграмм является моделью. Благодаря объяснению термина «декомпозиция» каждая грамматическая форма имеет свое строго определенное значение.

Такое обозначение отражает определенные системные принципы: входы преобразуются в выходы, управление ограничивает или предписывает условия выполнения преобразований, механизмы показывают, кто, что и как выполняет функция.

Блоки IDEF0 размещаются по степени важности, как ее понимает автор диаграммы. В IDEF0 этот относительный порядок называется доминированием. Доминирование понимается как влияние, которое один блок оказывает на другие блоки диаграммы. Например, самым доминирующим блоком диаграммы может быть либо первый из требуемой последовательности функций, либо планирующая или контролирующая функция.

Наиболее доминирующий блок обычно размещается в верхнем левом углу диаграммы, а наименее доминирующий – в правом нижнем углу. В результате получается "ступенчатая" схема, подобная представленной на рисунке 2.3 для блоков 1, 2, 3,4.

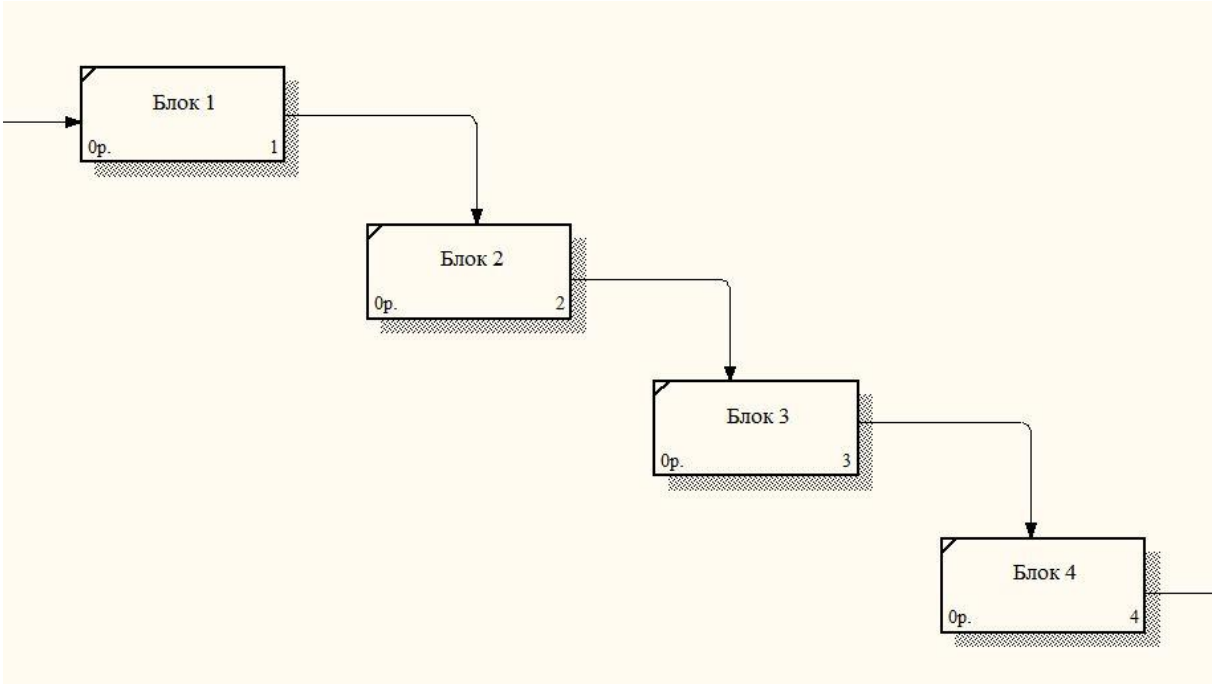


Рисунок 2.3. *– Принцип доминирования в диаграммах*

Расположение блоков на странице отражает авторское определение доминирования. Таким образом, топология диаграммы показывает, какие функции оказывают большее влияние на остальные. Чтобы подчеркнуть это, обычно перенумеровывают блоки в соответствии с порядком их доминирования. Порядок доминирования может обозначаться цифрой, размещенной в правом нижнем углу каждого прямоугольника: 1 будет указывать на наибольшее доминирование, 2 – на следующее после наибольшего, и т.д.

Блоки в IDEF0 должны быть перенумерованы. Номера блоков служат однозначными идентификаторами для системных функций и автоматически органи-

зуют эти функции в иерархию модели. Используя номера блоков и оценивая влияние, которое один блок оказывает на другой можно организовать модель по принципу функционального доминирования. Это позволяет согласовать иерархический порядок функций в модели с уровнем влияния каждой функции на остальную часть системы.

Стрелка формируется из одного или более отрезков прямых и наконечника на одном конце. Сегменты стрелок могут быть прямыми или ломаными; в последнем случае горизонтальные и вертикальные отрезки стрелки сопрягаются дугами, имеющими угол 90^{o}. Стрелки не представляют поток или последовательность событий, они лишь показывают, какие данные или материальные объекты должны поступить на вход функции для того, чтобы эта функция могла выполняться. Примеры стрелок приведены на рисунке 2.4.

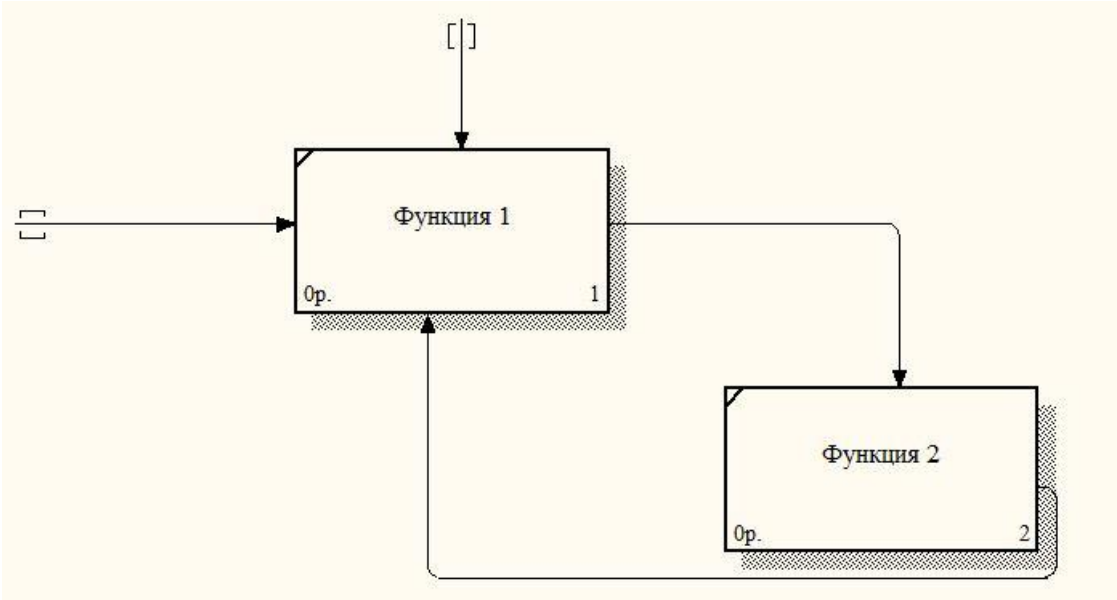


Рисунок 2.4. *– Виды стрелок*

Стрелки на диаграмме IDEF0, представляя данные или материальные объекты, одновременно задают своего рода ограничения (условия). Входные и управляющие стрелки блока, соединяющие его с другими блоками или с внешней средой, по сути описывают условия, которые должны быть выполнены для того, чтобы реализовалась функция, записанная в качестве имени блока.

Каждая стрелка должна быть помечена существительным или оборотом существительного, например: требования, бюджет, стоимость.

Кроме того, в IDEF0 существуют и дополнительное обозначение: стрелки, помещенные в «туннель». Туннель – прямоугольные/круглые скобки в начале

и/или окончании стрелки. Туннельные стрелки означают, что данные, выраженные этими стрелками, не рассматриваются на родительской диаграмме и/или на дочерней диаграмме. Стрелка, помещенная в туннель там, где она присоединяется к блоку, означает, что данные, выраженные этой стрелкой, не обязательны на следующем уровне декомпозиции. Стрелка, помещаемая в туннель на свободном конце означает, что выраженные ею данные отсутствуют на родительской диаграмме.

Итак, IDEF0-диаграмма составлена из блоков, связанных стрелками, которые определяют, как блоки влияют друг на друга. Это влияние может выражаться либо в передаче выходной информации к другой функции для дальнейшего преобразования, либо в выработке управляющей информации, предписывающей, что именно должна выполнять другая функция.

IDEF0-диаграммы не являются ни блок-схемами, ни просто диаграммами потоков данных. Это предписывающие диаграммы, представляющие входные-выходные преобразования и указывающие правила этих преобразований. Стрелки на них изображают интерфейсы между функциями системы, а также между системой и ее окружающей средой.

В методологии IDEF0 используется шесть видов взаимосвязей между блоками для описания их отношений: доминирование, управление, вход-выход, обратная связь по управлению, обратная связь по входу, выход-механизм. Связи по управлению и входу являются простейшими, поскольку они отражают прямые воздействия. Отношение управления возникает тогда, когда выход одного блока непосредственно влияет на блок с меньшим доминированием. Отношение входа возникает тогда, когда выход одного блока становится входом для блока с меньшим доминированием.

Обратная связь по управлению и обратная связь по входу являются более сложными, поскольку они представляют итерацию или рекурсию. А именно выходы из одной функции влияют на будущее выполнение других функций, что впоследствии влияет на исходную функцию. Обратная связь по управлению возникает тогда, когда выход некоторого блока влияет на блок с большим доминированием.

Связь по входной обратной связи имеет место тогда, когда выход одного блока становится входом другого блока с большим доминированием.

Связи «выход-механизм» встречаются нечасто. Они отражают ситуацию, при которой выход одной функции становится средством достижения цели для другой. Примеры связей показаны на рисунке 2.5.

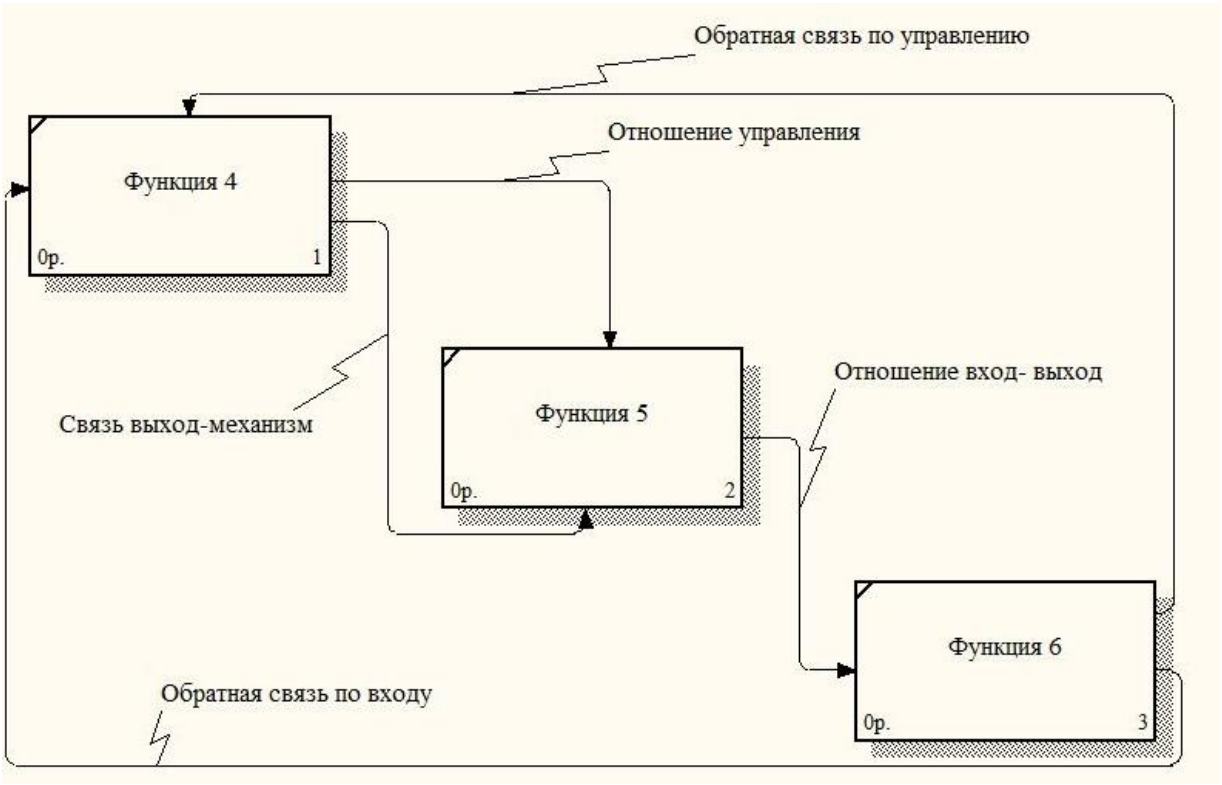


Рисунок 2.5. *– Примеры связей*

Одним из важных моментов при проектировании с помощью методологии IDEF0 является точная согласованность типов связей между функциями. Различают семь типов связывания (по степени значимости от 0 до 6) [5]:

- случайная;
- логическая;
- временная;
- процедурная;
- коммуникационная;
- последовательная;
- функциональная.

Тип случайной связности. Случайная связность возникает, когда конкретная связь между функциями мала или полностью отсутствует. Это относится к ситуации, когда имена данных на IDEF0-стрелках в одной диаграмме имеют малую связь друг с другом.

Тип логической связности. Логическое связывание происходит тогда, когда данные и функции собираются вместе вследствие того, что они попадают в общий класс или набор элементов, но необходимых функциональных отношений между ними не обнаруживается.

Тип временной связности. Связанные по времени элементы возникают вследствие того, что они представляют функции, связанные во времени, когда данные используются одновременно или функции включаются параллельно, а не последовательно.

Тип процедурной связности. Процедурно-связанные элементы появляются сгруппированными вместе вследствие того, что они выполняются в течение одной и той же части цикла или процесса.

Тип коммуникационной связности. Диаграммы демонстрируют коммуникационные связи, когда блоки группируются вследствие того, что они используют одни и те же входные данные и/или производят одни и те же выходные данные.

Тип последовательной связности. На диаграммах, имеющих последовательные связи, выход одной функции служит входными данными для следующей функции, то есть моделируются причинно-следственные зависимости.

Тип функциональной связности. Диаграмма отражает полную функциональную связность, при наличии полной зависимости одной функции от другой. В математических терминах необходимое условие для простейшего типа функциональной связности имеет следующий вид: $C = g(M) = g(f(A))$.

2.5. Классификация функций, моделируемых блоками IDEF0.

Единообразное представление явлений и событий, происходящих в моделируемых системах, в виде функциональных блоков является большим преимуществом графического языка IDEF0.

Практика построения моделей требует введения классификации явлений и событий с целью облегчения построения и интерпретации (понимания) функциональных моделей. Такая классификация облегчает выбор глубины декомпозиции моделируемых систем и способствует выработке единообразных подходов и приемов моделирования в конкретных предметных областях.

В [5,15] предлагается классификация, ориентированная на достаточно широкий круг систем. Классификация делит все функции таких систем на четыре основных и два дополнительных вида.

Основные виды функций.

1. Деятельность – совокупность процессов, выполняемых (протекающих) последовательно или(и) параллельно, преобразующих множество материальных или(и) информационных потоков во множество материальных или(и) информационных потоков с другими свойствами. Деятельность осуществляется в

соответствии с заранее определенной и корректируемой целью, с потреблением различных ресурсов, при выполнении ограничений со стороны внешней среды.

2. Процесс – совокупность последовательно или(и) параллельно выполняемых операций, преобразующая материальный или(и) информационный потоки в соответствующие потоки с другими свойствами. Процесс протекает в соответствии с управляющими директивами, вырабатываемыми на основе целей. В ходе процесса потребляются ресурсы и выполняются ограничения со стороны других процессов и внешней среды.

3. Операция – совокупность последовательно или(и) параллельно выполняемых действий, преобразующих объекты, входящие в состав материального или(и) информационного потока, в соответствующие объекты с другими свойствами. Операция выполняется в соответствии с директивами, вырабатываемыми на основе директив, определяющих протекание процесса, в состав которого входит операция и с потреблением всех видов ресурсов, а также с соблюдением ограничений со стороны других операций и внешней среды.

4. Действие – преобразование какого-либо свойства материального или информационного объекта в другое свойство. Действие выполняется в соответствии с командой, являющейся частью директивы на выполнение операции, с потреблением необходимых ресурсов и с соблюдением ограничений, налагаемых на осуществление операции.

5. Субдеятельность – совокупность нескольких процессов в составе деятельности, объединенная некоторой частной целью (являющейся подцелью деятельности).

6. Подпроцесс – группа операций в составе процесса, объединенная технологически или организационно.

Все функции, входящие в приведенную выше классификацию, находятся между собой в отношениях иерархической подчиненности по принципу «сверху вниз»: деятельность – субдеятельность – процесс – подпроцесс – операция – действие. Согласно методологии IDEF0 каждая функция выполняется посредством механизма. В большинстве систем, анализируемых при помощи функциональных моделей, такими механизмами служат организационно-технические структуры. Одним из концептуальных принципов функционального моделирования является «отделение организации от функций». Вместе с тем анализ показывает, что между иерархией функций (преобразований) и иерархией механизмов существует соответствие.

Таким образом, при корректном построении модели (без априорной привязки к организации) появляется возможность связать ее блоки на разных уровнях

декомпозиции с объектами организационно-технической структуры, выступающими в качестве механизмов. В этом случае организационно-техническая структура становится результатом функционального моделирования.

Один из общих принципов методологии IDEF0 требует, чтобы к каждому блоку на диаграмме должна быть присоединена хотя бы одна управляющая стрелка, отображающая условия правильного функционирования блока. Это требование есть следствие положения системотехники, согласно которому управление есть такое воздействие (преимущественно информационное) на систему, которое стимулирует ее функционирование в направлении достижения некоторой цели. В связи с этим можно сформулировать ряд определений и методических положений, которыми следует руководствоваться при отражении управлений на функциональных моделях.

Управление деятельностью – процесс, состоящий, как минимум, из следующих операций:

– формулирование целей деятельности;

– оценивание ресурсов, необходимых для осуществления деятельности и их сопоставление с имеющимися ресурсами;

– сбор информации об условиях протекания и фактическом состоянии деятельности;

– выработка и принятие решений, направленных на достижение целей; оформление решений в виде директив на управление процессами;

– реализация решений (исполнение директив) и оценка их результатов;

– корректировка (в случае необходимости, например, при нехватке ресурсов) ранее сформулированных целей (самонастройка, адаптация).

2.6. Синтаксис моделей в нотации IDEF0

Одна IDEF0-диаграмма сложна сама по себе, поскольку она содержит от трех до шести блоков, связанных множеством стрелок. Для адекватного описания системы требуется несколько таких диаграмм. Диаграммы, собранные и связанные вместе, становятся IDEF0-моделью. В IDEF0 дополнительно к правилам синтаксиса диаграмм существуют правила синтаксиса моделей. Синтаксис IDEF0-моделей позволяет разработчику определить границу модели, связать диаграммы в одно целое и обеспечить точное согласование между диаграммами.

IDEF0-модель является иерархически организованной совокупностью диаграмм. Диаграммы обычно состоят из трех-шести блоков, каждый из которых

потенциально может быть детализирован на другой диаграмме. Каждый блок может пониматься как отдельный тщательно определенный объект. Разделение такого объекта на его структурные части (блоки и стрелки, составляющие диаграмму) называется декомпозицией.

Декомпозиция формирует границы, и каждый блок в IDEF0 рассматривается как формальная граница некоторой части целой системы. Другими словами, блок и касающиеся его стрелки определяют точную границу диаграммы, представляющей декомпозицию этого блока. Эта диаграмма, называемая диаграммой с потомком, описывает все, связанное с этим блоком и его стрелками, и не описывает ничего вне этой границы.

Декомпозируемый блок называется родительским блоком, а содержащая его диаграмма – соответственно родительской диаграммой. Таким образом, IDEF0-диаграмма является декомпозицией некоторого ограниченного объекта.

Принцип ограничения объекта встречается на каждом уровне. Один блок и несколько стрелок на самом верхнем уровне используются для определения границы всей системы. Этот блок описывает общую функцию, выполняемую системой. Стрелки, касающиеся этого блока, описывают главные управления, входы, выходы и механизмы этой системы. Диаграмма, состоящая из одного блока и его стрелок, определяет границу системы и называется контекстной диаграммой модели. Таким образом, этот блок изображает границу системы: все, лежащее внутри него, является частью описываемой системы, а все, лежащее вне него, образует среду системы.

IDEF0-модели развиваются в процессе структурной декомпозиции сверху вниз. Сначала декомпозируется один блок, являющийся границей модели, на одной диаграмме, которая имеет от трех до шести блоков, затем декомпозируется один (или больше) из этих блоков на другой диаграмме с тремя-шестью блоками и т.д. Название диаграммы совпадает с названием декомпозируемого блока. Результатом этого процесса является модель, диаграмма верхнего уровня которой описывает систему в общих терминах, а диаграммы нижнего уровня описывают очень детализированные аспекты и операции системы.

Примеры декомпозиции приведены на рисунке 2.6.

Можно выделить следующие основные подходы к декомпозиции [4,5,7,18].

1. Функциональная декомпозиции (декомпозиция базируется на функциональных взаимоотношениях действий системы). Предпочтение отдается подробному показу требуемых ограничений на функции системы, а не их последовательности.

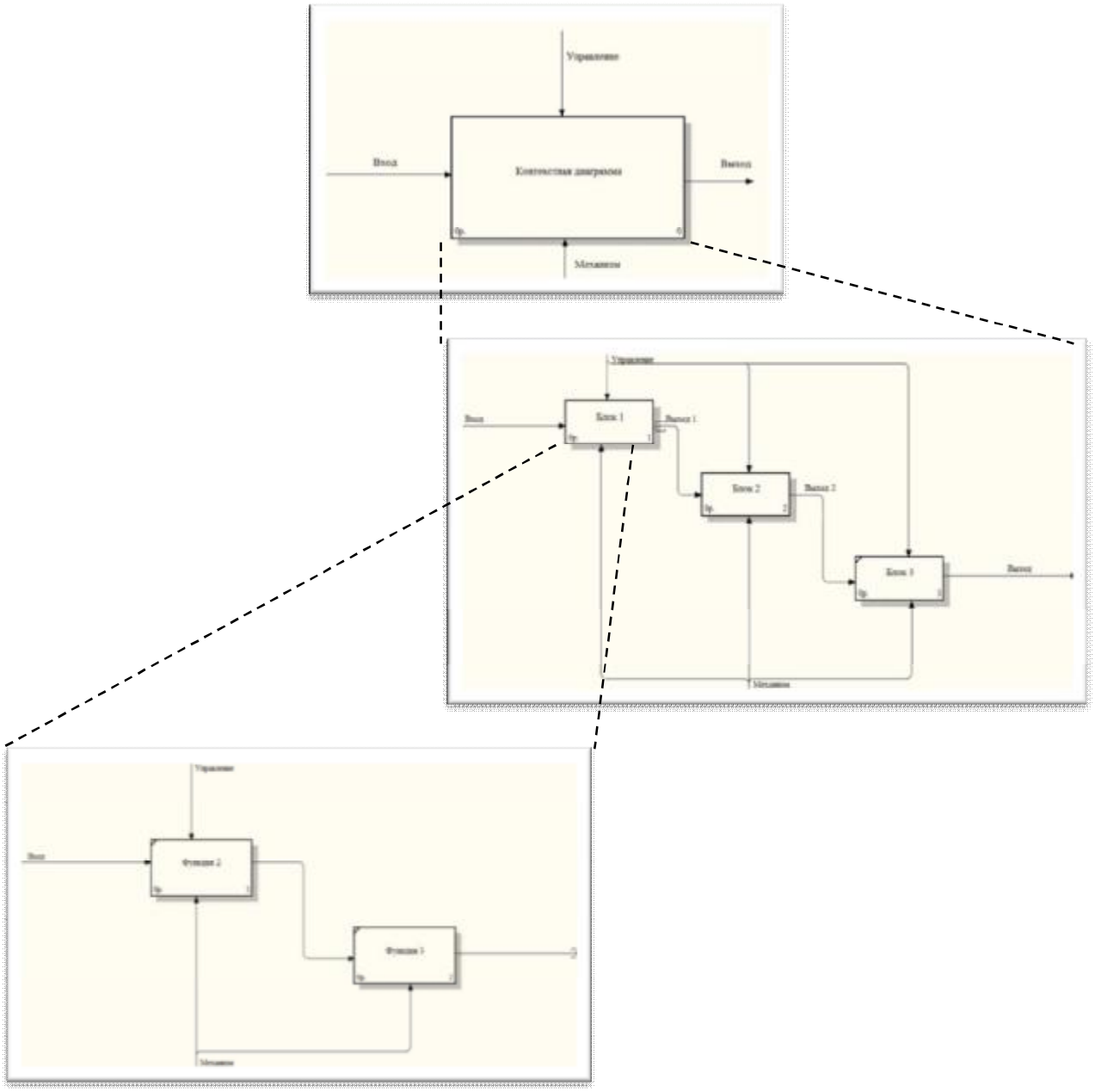


Рисунок 2.6. – *Структурная декомпозиция*

2. Декомпозиция в соответствии с уже известными стабильными подсистемами. Это приводит к созданию набора моделей, по одной модели на каждую подсистему или важную компоненту. Затем для описания всей системы должна быть построена составная модель, объединяющая все отдельные модели.

3. Декомпозиция, основанная на отслеживании «жизненного цикла» системы или ее элементов.

4. Декомпозиция по физическому процессу. Результатом такого сорта декомпозиции будет выделение функциональных стадий, этапов завершения или шагов выполнения.

Таким образом, каждая диаграмма представляет собой некоторую законченную часть всей модели. В методологии IDEF0 идентифицируется каждая диаграмма данной модели посредством того, что называется «номер узла». Номер узла для контекстной диаграммы имеет следующий вид: название модели или аббревиатура, косая черта, заглавная буква А (Activity в функциональных диаграммах), дефис и ноль. Например, номером узла для контекстной диаграммы модели Интранет-портала является ИП/А-0. Номером узла диаграммы, декомпозирующей контекстную диаграмму, является тот же номер узла, но без дефиса (например, ИП/АО). Все другие номера узлов образуются посредством добавления к номеру узла родительской диаграммы номера декомпозируемого блока. Номер узла на первой диаграмме – ИП/АО, а номер узла на второй диаграмме – ИП/А1. Диаграмма ИП/А1 декомпозирует блок 1 диаграммы ИП/АО. (Первый ноль при образовании номера узла принято опускать, поэтому вместо ИП/А01 пишется ИП/А1.)

Для идентификации версий диаграмм используются С-номера, которые служат для связки диаграмм при движении как вверх, так и вниз по иерархии модели. Обычно С-номер диаграммы, декомпозирующей некоторый блок, впервые появляется непосредственно под этим блоком на родительской диаграмме. Это образует «направленную вниз» связь от родительской диаграммы к диаграмме-потомку.

Как только образуется направленная вниз связь, на диаграмме-потомке формируется ссылка на родительскую диаграмму. В области контекста IDEF0-бланка (правый верхний угол) автор изображает каждый блок родительской диаграммы маленькими квадратиками, заштриховывает квадратик декомпозируемого блока и размещает С-номер родительской диаграммы возле заштрихованного квадратика. Это образует «направленную вверх» (к родительской диаграмме) связь. Метод соединения диаграмм посредством однозначно определенных номеров гарантирует, что именно нужная версия диаграммы станет частью модели. Другими словами, при использовании С-номеров осуществляется тщательный контроль за введением новых диаграмм в иерархию модели.

Начало моделирования в IDEF0 означает создание диаграмм А-О и А0, которые затем могут быть отрецензированы. Эти две диаграммы полностью рассказывают все об изучаемой системе с минимальной степенью детализации. Создавая их, разработчик предпринимает начальную попытку декомпозировать систему и затем обобщить полученную декомпозицию. Декомпозиция (диаграмма АО) освещает наиболее важные функции и объекты системы. Объединение (диаграмма А-0) трактует систему как «черный ящик», дает ей название и определяет наиболее важные входы, управления, выходы и, возможно, механизмы.

Рассмотрим некоторые рекомендации по использованию IDEF модели [9, 11, 12, 14].

Прежде чем начать моделирование, разработчик проводит подготовку к нему, собирает информацию, декомпозирует объект и обобщает эту декомпозицию. Подготовка включает выбор цели модели, выбор точки зрения, с которой будет представлена модель, тип создаваемой модели и предполагаемое использование построенной и проверенной модели. Таким образом, подготовка должна максимально облегчить сбор информации.

Сбор информации может включать любую комбинацию следующих видов деятельности: чтение документов, наблюдение за существующими операциями, анкетирование группы экспертов, опрос одного или нескольких экспертов, использование собственных знаний и придуманного описания работы системы, которое впоследствии может быть откорректировано.

Декомпозируя объект, необходимо, прежде всего, обратить внимание на входные и выходные данные для всей системы. Декомпозиция всей системы начинается с составления списка основных типов данных и основных функций. Потом эти списки снабжаются комментариями для указания основных типов, как данных, так и функций системы или их различных сочетаний. Наконец, списки с комментариями используются для создания диаграммы АО, которая затем обобщается с помощью диаграммы А-0.

Цель и точка зрения модели определяются на самой ранней стадии создания модели [9]. Выбор цели осуществляется с учетом вопросов, на которые должна ответить модель, а выбор точки зрения – в соответствии с выбором позиции, с которой описывается система. Иногда цель и точку зрения можно выбрать до того, как будет сделана первая диаграмма. Необходимо как можно раньше определять цель и выбирать точку зрения новой модели.

Иногда оказывается, что определить цель и точку зрения в самом начале моделирования чрезвычайно трудно. В таком случае целесообразно составить списки данных и функций и изобразить диаграмму АО.

Списки объектов системы, создаваемые в ходе моделирования, в IDEF0 принято называть «списками данных». Термин «данные» здесь употребляется как синоним слова «объект». Составление списка данных является начальным этапом создания каждой диаграммы функциональной IDEF0-модели. Правило заключается в том, чтобы вначале составить список данных, а потом список функций.

Начиная с составления списка данных, можно избежать перехода к немедленной функциональной декомпозиции. Списки данных помогут выполнить более глубокий анализ и при этом не концентрироваться на функциях системы.

IDEF0-диаграммы представляют границы функций и ограничения, накладываемые на них, причем ограничения должны присутствовать во всех системах. Без ограничений функциональная IDEF0-диаграмма представляет собой схему потоков данных.

После составления списка данных составляется список функций. При этом несколько различных типов данных может использоваться одной функцией. Для каждой конкретной функции определяется ее отношении к группам данных. Целесообразно объединять функции в «агрегаты». Это позволит привести модель к 3-6 функциональным группировкам. Эти группировки должны иметь один и тот же уровень сложности, содержать примерно одинаковый объем функциональности и функции в каждой из них должны иметь сходные операции и цели.

Исходное содержание диаграммы АО обеспечивают списки данных и функций. Для правильного описания системы содержанию надо придать форму. В IDEF0 это делается посредством построения диаграммы.

Целесообразно придерживаться следующего порядка:

- расположить блоки на странице;
- нарисовать основные стрелки, представляющие ограничения;
- нарисовать внешние стрелки;
- нарисовать все оставшиеся стрелки.

Правильное расположение блоков является самым важным этапом построения диаграммы. Блоки располагаются в соответствии с их доминированием (по степени важности или по порядку следования). Самый доминантный блок обычно располагается в верхнем левом углу, а наименее доминантный – в нижнем правом.

Затем изображают основные стрелки, представляющие ограничения. Это является второй важной частью построения диаграммы АО. Они дают основание для разбиения объекта диаграммы на 3 – 6 системных функций, изображаемых блоками.

Основными стрелками, представляющими ограничения, всегда являются внешние стрелки, то есть стрелки, представляющие данные, поступающие из непосредственного окружения диаграммы.

Следующим шагом в построении диаграммы является размещение остальных внешних стрелок.

Таким образом, все данные, входящие в систему или выходящие из нее, оказываются учтенными на рисунке. Потеря внешней стрелки – это ошибка интерфейса, одна из самых распространенных в системном анализе. Далее изображаются все остальные стрелки, отражающие детали работы системы в целом и обратные связи в потоках данных.

Обобщение является последним важным шагом начального этапа моделирования. Для любой IDEF0-диаграммы есть родительская диаграмма, содержащая ее контекст, где под контекстом понимается блок с набором входных стрелок, стрелок управления и выходных стрелок. Верхняя диаграмма модели (диаграмма АО) не составляет исключения. Контекстом для нее служит диаграмма А-0, представляющая собой обобщение всей модели. Диаграмма А-0 имеет несколько предназначений. Во-первых, она объявляет общую функцию всей системы. Во-вторых, она дает множество основных типов или наборов данных, которые использует или производит система. В-третьих, А-0 диаграмма указывает взаимоотношения между основными типами данных, проводя их разграничение.

Построение диаграммы А-0 свидетельствует об окончании начального этапа моделирования. Общий вид системы, полученный с помощью диаграмм А-0 и АО – основная цель разработчика на начальном этапе построения IDEF0-модели.

Начальный этап моделирования включает определение объекта, цели и точки зрения модели, ограничения, накладываемые на объект, построение диаграммы верхнего уровня и ее обобщение, составление списков данных и функций, объединение функций в блоки, формирование с использованием списка данных взаимоотношений между блоками. Продолжение моделирования основывается на тех же методах и выводит модель на следующий уровень детализации. Для этого требуется создать отдельную диаграмму для, возможно, каждого блока диаграммы верхнего уровня, затем построить диаграммы для всех блоков новых диаграмм, и так до тех пор, пока модель не будет описывать объект с нужной для достижения цели степенью детализации. Таким образом, продолжение моделирования является рекурсивным процессом.

Процесс декомпозиции ограниченного объекта состоит из следующих шагов:

- выбор блока диаграммы;
- рассмотрение объекта, определенного этим блоком;
- создание новой диаграммы;
- выявление недостатков новой диаграммы;
- создание альтернативных декомпозиции;
- корректировка новой диаграммы;
- корректировка всех связанных с ней диаграмм.

Декомпозиция начинается с чтения диаграммы АО и определения самого содержательного блока. Это такой блок, декомпозиция которого выявит многие аспекты диаграммы АО и будет оказывать большое влияние на будущие декомпозиции других блоков этой диаграммы. При выборе самого содержа-

тельного блока следует учитывать доминирование, функциональную сложность и понятность.

2.7. Модели AS-IS и TO-BE

Обычно сначала строится модель существующего объекта – AS-IS (как есть). На основе модели AS-IS достигается консенсус между различными единицами объекта и определяются все данные.

Назначение модели AS IS следующее:

– определить существующие в организации процессы и их взаимную связь;

– получить стоимостные и временные оценки процессов для их анализа и в качестве базы для сравнения с альтернативными вариантами (TO BE) при проведении реинжиниринга;

– определить наиболее критичные процессы, требующие изменения, исключения или поиска альтернативных решений.

Модель AS-IS позволяет выявить существующую структуру с целью правильного ее совершенствования. Модель AS-IS позволяет определить неэффективные места существующего на момент моделирования процесса, оценить, насколько глубоким изменениям необходимо подвергнуть существующую структуру организации системы. Признаками неэффективности существующего процесса могут быть, например, бесполезные работы (в работах отсутствует выход), неуправляемые работы (в работах отсутствует управление) и дублирующиеся работы, отсутствие обратных связей по управлению (на проведение процесса не оказывает влияния его результат), входу (материалы или информация используются нерационально). Анализ функциональной модели позволяет понять, где находятся наиболее слабые места, в чем будут состоять преимущества новых процессов и насколько глубоким изменениям подвергнется существующий объект. Детализация процессов позволяет выявить недостатки организации даже там, где функциональность на первый взгляд кажется очевидной.

Найденные в модели AS-IS недостатки можно исправить при создании модели TO-BE (как будет) – модели новой организации процессов.

При создании модели AS-IS нельзя создавать идеализированные модели, которую нельзя в дальнейшем использовать для анализа. Такая модель называется SHOULD BE (как должно бы быть).

Технология проектирования подразумевает сначала создание модели AS-IS, ее анализ и улучшение процессов, то есть создание модели TO-BE, и только на

основе модели TO-BE строится модель данных, прототип и затем окончательный вариант системы.

Целями разработки модели TO BE являются:

– определить новые процессы предлагаемых альтернативных решений и их влияние на остальные процессы;

– получить стоимостные и временные оценки процессов для их сравнения с базовой AS IS моделью;

– определить наиболее оптимальные альтернативные решения;

– подготовить базу для проведения экономического анализа выбранных альтернативных решений.

Иногда текущая AS-IS и будущая TO-BE модели различаются очень сильно, так что переход от начального к конечному состоянию становится неочевидным. В этом случае необходима третья модель, описывающая процесс перехода от начального к конечному состояния системы, поскольку такой переход – это тоже процесс.

3. ОСНОВЫ МЕТОДОЛОГИИ IDEF1

Для создания моделей данных можно использовать две нотации: методологии проектирования реляционных баз данных IDEF1X и IE (Information Engineering).

Процесс построения информационной модели на основе нотации IDEF1 состоит из следующих основных шагов:

- определение сущностей;
- определение зависимостей между сущностями;
- задание первичных и альтернативных ключей;
- определение атрибутов сущностей;
- приведение модели к требуемому уровню нормальной формы;
- переход к физическому описанию модели: назначение соответствий имя сущности – имя таблицы, атрибут сущности – атрибут таблицы; задание триггеров, процедур и ограничений;
- генерация базы данных.

3.1. Основные понятия реляционной базы данных

База данных (БД) – поименованная совокупность структурированных данных, относящихся к определенной предметной области.

Данные – это набор конкретных значений, параметров, характеризующих объект, условие, ситуацию и др. (результаты тестирования, сдачи промежуточной и итоговой аттестации и т.д.). Данные не обладают определенной структурой, поэтому в явном виде они не несут информационной нагрузки. Только после структуризации, то есть определения смыслового содержания, они становятся информацией (обработка результатов тестирования, определения качества обучения).

Модель данных – это некоторая абстракция, позволяющая систематизировать и преобразовывать данные в соответствии с требуемыми потребностями, и будучи приложена к конкретным данным, позволяет пользователям трактовать их уже как информацию. Конкретный вид модели данных определяет вид системы управления базами данных.

Предметная область – некоторая часть реально существующей системы, функционирующая как самостоятельная единица.

Система управления базами данных (СУБД) – комплекс программных и языковых средств, необходимых для создания базы данных, добавления, модификации, удаления, поиска и отбора информации, представления информации на экране и в печатном виде, разграничения прав доступа к информации, выполнения других операций с базой.

Реляционная БД – основной тип современных баз данных. Состоит из таблиц, между которыми могут существовать связи по ключевым значениям.

Таблица базы данных (table) – регулярная структура, которая состоит из однотипных строк (записей, records), разбитых на столбцы (поля, fields).

В теории реляционных баз данных синоним таблицы – отношение (relation), в котором строка называется кортежем, а столбец называется атрибутом.

Независимо от вида данных на этапе концептуального проектирования основной является модель «сущность–связь». Часто ее называют ER-моделью (Entity – сущность, Relation – связь). В ней моделирование структуры данных предметной области базируется на использовании графических средств – ER-диаграмм (диаграмм «сущность–связь»). Основными понятиями такой модели являются: сущность, атрибут, связь.

Сущность – это некоторый объект предметной области, который может существовать относительно независимо от других объектов.

Сущность имеет экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию.

Атрибут – это свойство сущности.

Пример. Сущность Студент характеризуется такими атрибутами, как возраст, пол, место рождения, место жительства, номер паспорта, семейное положение. Конкретные студенты являются экземплярами сущности Студент. Они отличаются значениями указанных атрибутов и однозначно идентифицируются атрибутом номер паспорта.

Атрибут, который уникальным образом идентифицирует экземпляры сущности, называется ключом. Может быть составной ключ, представляющий комбинацию нескольких атрибутов.

Связь представляет взаимодействие между сущностями. Она характеризуется мощностью, которая показывает, сколько сущностей участвует в связи. Связь между двумя сущностями называется бинарной, а связь между более чем с двумя сущностями – тернарной.

В концептуальной модели реляционной БД аналогом таблицы является сущность (entity), с определенным набором свойств – атрибутов, способных принимать определенные значения (набор допустимых значений – домен).

Ключевой элемент таблицы – такое ее поле (простой ключ) или строковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На практике для использования ключей создаются индексы – служебная информация, содержащая упорядоченные сведения о ключевых значениях. В реляционной теории и концептуальной модели понятие «ключ» применяется для атрибутов отношения или сущности.

Первичный ключ – главный ключевой элемент, однозначно идентифицирующий строку в таблице. Могут также существовать альтернативный и уникальный ключи, служащие также для идентификации строк в таблице.

В реляционной теории первичный ключ – минимальный набор атрибутов, однозначно идентифицирующий кортеж в отношении.

В концептуальной модели первичный ключ – минимальный набор атрибутов сущности, однозначно идентифицирующий экземпляр сущности.

Связь (relation) – функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (parent, родительской) таблице – первичный, второй – внешний ключ – во внешней (child, дочерней) таблице, как правило, первичным не является и образует связи «один-к-одному», «один ко многим» или «многие-ко-многим». Информация о связях сохраняется в базе данных.

Внешний ключ (foreign key) – ключевой элемент подчиненной (внешней, дочерней) таблицы, значение которого совпадает со значением первичного ключа главной (родительской) таблицы.

В реляционной модели определены три основных вида отношений (связей) между двумерными массивами:

- отношение один-к-одному;
- отношение один-ко-многим;
- отношение многие-ко-многим.

Рассмотрим сущности указанных отношений.

Отношение «один-к-одному». Одной строке первой таблицы соответствует одна строка второй таблицы.

Пример. В реляционной модели данные представлены в виде двух плоских таблиц. В первой из них хранятся данные об успеваемости студента (таблица Успеваемость), а во второй таблице – паспортные данные студентов (таблица Студент). Очевидно каждой строчке первой таблицы (например, ФИО студента) можно поставить в соответствие одну строчку второй таблицы (например, номер и се-

рия паспорта студента). Разделение таблиц вызвано тем фактом, что паспортные данные могут потребоваться и для других целей, например, в военный комиссариат, в списки избирателей и т.д. Включение паспортных данных во все требуемые таблицы, во-первых, увеличивает временные затраты, а, во-вторых, существенно увеличивает объем хранимой на компьютере информации.

Отношение «один-ко-многим». Одной строке первой таблицы соответствует несколько строк второй таблицы, но одной строке второй таблицы соответствует только одно строка первой таблицы. Данное отношение является наиболее распространенным в реляционных моделях данных.

Пример. Основной функцией кафедры является учебная работа. Обычно кафедра в соответствии с учебным планом, Государственными образовательными стандартами преподает ряд дисциплин. При этом одна дисциплина может преподаваться по нескольким специальностям, а один преподаватель может проводить занятия по нескольким дисциплинам. При этом перечень дисциплин по специальностям и распределение преподавателей по дисциплинам может изменяться. Поэтому жесткое закрепление Преподаватель → Специальность → Дисциплины явно не является принципиальным. Очевидно, целесообразно составить отдельные таблицы, например: Преподаватели, Дисциплины, Стандарты, Специальность – и потом связать их отношениями. При этом, очевидно, одна строка в таблице преподаватель имеет отношение к нескольким строкам таблицы Дисциплины.

Отношение «многие-ко-многим». Одной строке первой таблицы соответствует несколько строк второй таблицы и одной строке второй таблицы соответствует несколько строк первой таблицы.

Очевидно, чтобы однозначно связать две таблицы, в каждой из них должен быть уникальный столбец, данные в котором однозначно определяют принадлежность к заданному элементу данных (данному преподавателю, студенту с учетом того, что фамилии, имена, отчества могут совпадать). Этот уникальный столбец может либо принадлежать таблице, либо вводится в нее дополнительно. В реляционной модели этот столбец называется ключом. Например, в таблице, хранящей данные о студентах таким столбцом может быть номер по порядку. В ряде случаев уникальность записи может быть получена на основе анализа нескольких столбцов. Тогда мы имеем дело с составным ключом.

Ссылочная целостность данных (referential integrity) – набор правил, обеспечивающих соответствие ключевых значений в связанных таблицах.

Хранимые процедуры (stored procedures) – программные модули, сохраняемые в базе данных для выполнения определенных операций с информацией базы.

Триггеры (triggers) – хранимые процедуры, обеспечивающие соблюдение условий ссылочной целостности данных в операциях изменения первичных ключей (возможно каскадное изменение данных), удалении записей в главной таблице (каскадное удаление в дочерних таблицах) и добавлении записей или изменении данных в дочерних таблицах.

Объект (object) – элемент информационной системы, обладающий определенными свойствами (properties) и определенным образом реагирующий на внешние события (events).

Система – совокупность взаимодействующих между собой и с внешним окружением объектов.

Репликация базы данных – создание копий базы данных (реплик), которые могут обмениваться обновляемыми данными или реплицированными формами, отчетами или другими объектами в результате выполнения процесса синхронизации.

Транзакция – изменение информации в базе в результате выполнения одной операции или их последовательности, которое должно быть выполнено полностью или не выполнено вообще. В СУБД существуют специальные механизмы обеспечения транзакций.

Язык SQL (Structured Query Language) – универсальный язык работы с базами данных, включающий возможности ее создания, модификации структуры, отбора данных по запросам, модификации информации в базе и прочие операции манипулирования базой данных.

Null – значение поля таблицы, показывающее, что информация в данном поле отсутствует. Разрешение на возможность существования значения Null может задаваться для отдельных полей таблицы.

Нормализация – процесс проверки и реорганизации сущностей и атрибутов с целью удовлетворения требований к реляционной модели данных. Нормализация позволяет быть уверенным, что каждый атрибут определен для своей сущности, значительно сократить объем памяти для хранения информации и устранить аномалии в организации хранения данных. В результате проведения нормализации должна быть создана структура данных, при которой информация о каждом факте хранится только в одном месте. Процесс нормализации сводится к последовательному приведению структуры данных к нормальным формам.

Известны шесть нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF);
- третья нормальная форма (3NF);

– нормальная форма Бойса-Кодда (усиленная 3NF);
– четвертая нормальная форма (4NF);
– пятая нормальная форма (5NF).

На практике обычно ограничиваются приведением данных к третьей нормальной форме (полная атрибутивная модель, FA).

Нормальные формы основаны на понятии функциональной зависимости.

Функциональная зависимость. Атрибут B сущности E функционально зависит от атрибута A сущности E тогда и только тогда, когда каждое значение A и E связало с ним точно одно значение B и E, то есть A однозначно определяет B.

Полная функциональная зависимость. Атрибут B сущности E полностью функционально зависит от ряда атрибутов A сущности E тогда и только тогда, когда B функционально зависит от A и не зависит ни от какого подряда A.

Третья нормальная форма (3NF). Сущность находится в третьей нормальной форме, если она находится во второй нормальной форме и никакой неключевой атрибут не зависит от другого неключевого атрибута (не должно быть взаимозависимости между неключевыми атрибутами).

Вторая нормальная форма (2NF). Сущность находится во второй нормальной форме, если она находится в первой нормальной форме, и каждый неключевой атрибут полностью зависит от первичного ключа (не должно быть зависимости от части ключа).

Первая нормальная форма (1NF). Сущность находится в первой нормальной форме тогда и только тогда, когда все атрибуты содержат атомарные значения. Среди атрибутов не должно встречаться повторяющихся групп, то есть несколько значений для каждого экземпляра.

Умение работать с ER-моделью в области образования позволяет структурировать требуемую для решения задачу (моделирование электронного учебника, оценки качества знаний, экспертные оценки, моделирование портала (сайта), проектирование реляционной базы данных и т.д.)

ER-модель в совокупности с наборами атрибутов сущностей может служить примером концептуальной модели предметной области или концептуальной схемы базы данных.

Существующие программные средства (например, ERwin) позволяют строить ER-диаграммы в реальном масштабе времени, что дает возможность наглядно изучать концептуальную модель данных и перестраивать ее соответственно поставленным целям и имеющимся ограничениям.

Общий успех баз данных в сочетании с информационными потребностями образования и исследованиями искусственного интеллекта привел к росту за-

интересованности в превращении систем управления базами данных в системы управления базами знаний, что может рассматриваться как тенденция развития систем управления базами данных

3.2. Нотация IDEF1X

Проблема представления семантики при разработке баз данных давно интересовала разработчиков, и в семидесятых годах было предложено несколько моделей данных, названных семантическими моделями.

К ним можно отнести семантическую модель данных, предложенную Хаммером (Hammer) и Мак-Леоном (McLeon) в 1981 году, функциональную модель данных Шипмана (Shipman), также созданную в 1981 году, модель «сущность-связь», предложенную Ченом (Chen) в 1976 году, и ряд других моделей.

В настоящий момент именно модель Чена «сущность-связь» (Entity Relationship), стала фактическим стандартом при инфологическом моделировании баз данных. Общепринятым стало сокращенное название ER-модель, большинство современных CASE-средств содержат инструментальные средства для описания данных в формате этой модели.

Кроме того, разработаны методы автоматического преобразования проекта БД из ER-модели в реляционную, при этом преобразование выполняется в даталогическую модель, соответствующую конкретной СУБД.

Все CASE-системы имеют развитые средства документирования процесса разработки БД, автоматические генераторы отчетов позволяют подготовить отчет о текущем состоянии проекта БД с подробным описанием объектов БД и их отношений, как в графическом виде, так и в виде готовых стандартных печатных отчетов, что существенно облегчает ведение проекта.

В настоящий момент не существует единой общепринятой системы обозначений для ER-модели и разные CASE-системы используют разные графические нотации.

Метод IDEF1, разработанный Т.Рэмей (T.Ramey), также основан на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. В настоящее время на основе совершенствования методологии IDEF1 создана ее новая версия – методология IDEF1X. IDEF1X разработана с учетом таких требований, как простота изучения и возможность автоматизации.

Под методологией понимают дисциплину проектирования:

– критерии и правила, используемые при выполнении работы;

– графические и текстовые средства, используемые для описания проектируемой системы.

Методология информационного моделирования IDEF1X основана на подходе П.Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

Методология IDEF1X – один из подходов к семантическому моделированию данных, основанный на концепции «сущность-связь». Основное назначение – построения концептуальной схемы реляционной базы данных, которая была бы независимой от программной платформы её конечной реализации.

IDEF1X позволяет на основе простых графических изображений моделировать информационные взаимосвязи и различия между реальными объектами, физическими и абстрактными зависимостями, существующими среди реальных объектов и информацией, относящейся к реальным объектам

При построении информационной модели оперируют с двумя областями, каждая из которых соответствует множеству характерных объектов. Первая область – это реальный мир, или совокупность физических и интеллектуальных объектов, таких, как люди, идеи и т.д., а также все свойства этих объектов и зависимости между ними. Вторая область является информационной. Она включает в себя информационные отображения объектов первой области и их свойств. Методология IDEF1X разработана как инструмент для исследования статического соответствия вышеуказанных областей и установления строгих правил и механизмов изменения объектов информационной области при изменении соответствующих им объектов реального мира.

IDEF1X использует основные понятия реляционных баз данных: сущность, атрибут, отношение и ключ. Кроме того, IDEF1X дополнительно оперирует рядом понятий, правил и ограничений, такими как домены, представления, первичные, внешние и суррогатные ключи и другими.

Стандарт и методология IDEF1X является специализированным инструментом, предназначенным для разработчиков реляционных баз данных.

Основными компонентами IDEF1X- модели являются.

Сущности, представляющие множество реальных или абстрактных предметов (людей, объектов, мест, событий, состояний, идей и т.д). Они изображаются блоками.

Выделяют два вида сущностей:

– независимые от идентификатора сущности;

– зависимые от идентификатора сущности.

Сущность – это класс объектов предметной области, обладающих общими атрибутами или характеристиками, который именуется с помощью существительного в единственном числе. Один из способов идентификации сущностей состоит в поиске объектов предметной области, которые существуют независимо от других или выборка всех существительных, присутствующих в спецификациях на проект.

Основными концептуальными свойствами сущностей в IDEF1 являются:

1. Устойчивость. Информация, имеющая отношение к той или иной сущности постоянно накапливается.

2. Уникальность. Любая сущность может быть однозначно идентифицирована из другой сущности.

Сущность представляет множество реальных или абстрактных предметов. Отдельный элемент этого множества называется экземпляром сущности. Каждая сущность может обладать любым количеством отношений с другими сущностями.

Сущность является независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями.

Сущность называется зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Примеры сущностей приведены на рисунке 3.1.

Сущность обладает одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через отношение.

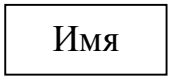
Сущность обладает одним или несколькими атрибутами, которые однозначно идентифицируют каждый образец сущности.

Каждая сущность может обладать любым количеством отношений с другими сущностями модели.

Если внешний ключ целиком используется в качестве первичного ключа сущности или его части, то сущность является зависимой от идентификатора.

И наоборот, если используется только часть внешнего ключа или вообще не используются внешние ключи, то сущность является независимой от идентификатора.

В нотации IDEF1X сущности обозначаются следующими элементами диаграмм:



- независимая сущность

Имя – зависимая сущность

Атрибут – это характеристика или элемент данных, описывающий что-либо в сущности.

Атрибуты могут быть:

- неключевыми;
- первичными ключами;
- альтернативными ключами;
- внешними ключами.

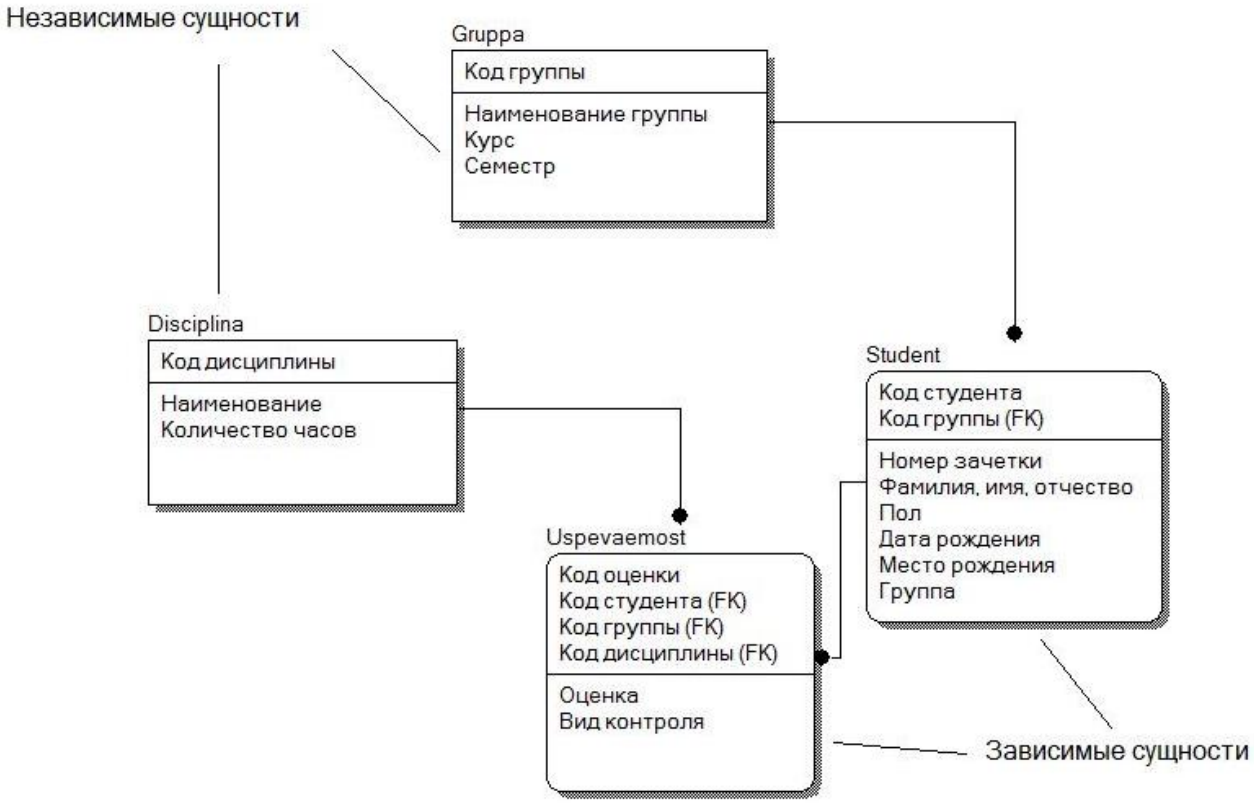


Рисунок 3.1. *– Зависимые и независимые сущности*

Целью определения атрибутов является выявление свойств, и дальнейшее их связывание с соответствующими сущностями или связями (например: сущность Студент имеет атрибут Группа). Различают также свободные атрибуты, то есть не относящиеся к определенным сущностям, но имеющие значение, тогда когда сущности вступают в связь. Рассмотрим пример: сущность «Студент»

не обладает таким атрибутом как «Успеваемость» до тех пор, пока «Студент» не вступит в отношение с сущностью «Тест».

Целью определения доменов атрибутов является определение диапазона всех возможных значений, которые может принять атрибут.

Доменом называется некоторый диапазон значений, элементы которого выбираются для присвоения значений одному или более атрибутам. Домены должны содержать следующие данные:

- набор допустимых значений для атрибута;
- сведения о размере и формате каждого из атрибутов.

Атрибуты могут классифицироваться по принадлежности к одному из трех различных типов: описательные, указывающие, вспомогательные.

Описательные атрибуты представляют факты, внутренне присущие каждому экземпляру сущности.

Указывающие атрибуты используются для присвоения имени или обозначения экземплярам сущности.

Вспомогательные атрибуты используются для связи экземпляра одной сущности с экземпляром другой.

Атрибуты подчиняются строго определенным правилам. Правила атрибутов:

1. Каждый атрибут идентифицируется уникальным именем, одному и тому же имени должно соответствовать одно и то же значение. Одно и то же значение не может соответствовать различным именам.

2. Сущность может обладать любым количеством атрибутов. Каждый атрибут принадлежит только одной сущности.

3. Сущность может обладать любым количеством наследуемых атрибутов, но наследуемый атрибут должен быть частью первичного ключа соответствующей сущности-родителя или общей сущности.

4. Для каждого экземпляра сущности должно существовать значение каждого его атрибута не равное нулю (правило необращения в нуль).

5. Ни один из экземпляров сущности не может обладать более чем одним значением для связанного с ней атрибута (правило неповторения).

6. Сущность должна обладать атрибутом или комбинацией атрибутов, чьи значения однозначно определяют каждый экземпляр сущности. Эти атрибуты образуют первичный ключ сущности.

Например, каждый из атрибутов Номер студенческого билета, Номер паспорта и Номер полиса может однозначно идентифицировать экземпляр сущности Студент, поэтому они могут быть атрибутами первичного ключа. Другие атрибуты сущности являются неключевыми (Фамилия, Имя, Отчество, Дата рождения).

При существовании нескольких возможных ключей один из них обозначается в качестве первичного ключа (атрибут Номер студенческого билета для сущности Студент), а остальные – как альтернативные ключи (например, атрибут Номер паспорта). Альтернативный ключ – это ключ, не являющийся первичным ключом сущности. Каждому альтернативному ключу присваивается уникальный целый номер. Этот ключ указывается с помощью размещения справа от каждого атрибута ключа заключенных в скобки букв АК с номером альтернативного ключа.

Атрибуты изображаются в виде списка имен внутри блока сущности. Атрибуты, определяющие первичный ключ, размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой (рисунок 3.2).

Рисунок 3.2. *– Атрибуты сущности*

Атрибуты, появляющиеся в зависимой сущности и являющиеся также первичным ключом другой сущности (сущности – родителя или общей сущности) называются внешними ключами. Сущность может обладать любым количеством внешних (наследуемых) атрибутов. В диаграммах модели внешние ключи (Foreign Key) изображаются с помощью помещения внутрь блока сущности имен наследуемых атрибутов, после которых в скобках следуют буквы FK. Если наследуемый атрибут принадлежит первичному ключу сущности-потомка, то он помещается выше горизонтальной линии (а сущность изображается с закругленными углами), а если нет, то – ниже.

Процесс перемещения первичного ключа родительской или общей сущности в сущность-потомок или сущность-категорию из соответствующего отношения называется миграцией ключа.

Роль атрибута – это функция, выполняемая атрибутом в описании сущности, включая функцию мигрируемости, принадлежности первичного ключа, альтер-

нативного ключа. Обычно имя роли атрибута назначается для каждого случая, когда отдельный атрибут наследуется более одного раза (то есть, когда сущность-потомок имеет несколько различных отношений с одной и той же сущностью-родителем).

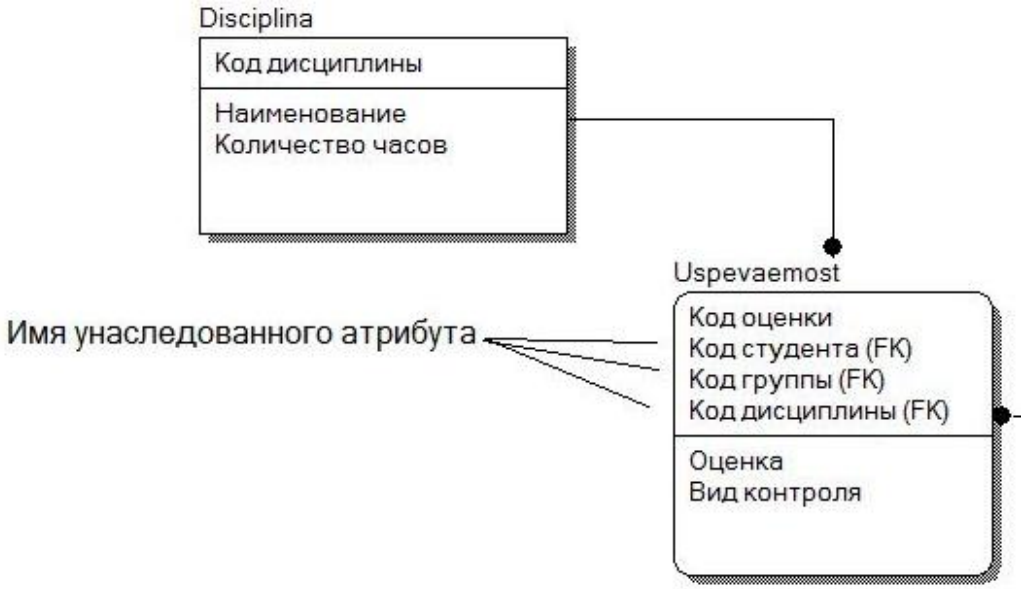


Рисунок 3.3. *– Унаследованные атрибуты*

Например, сущность Студент в качестве сущности-родителя обладает двойным отношением с сущностью Группа. Один и тот же студент в рамках студенческого проекта может быть как руководителем группы, так и исполнителем группы. Имена ролей Руководитель и Исполнитель могут назначаться для различения двух наследуемых атрибутов Студент. Имена ролей могут использоваться и при единственном появлении наследуемого атрибута для более точного выражения его смысла в контексте сущности-потомка, но это не является обязательным.

Правила ключей:

– каждая сущность должна обладать первичным ключом;

– каждая сущность может обладать любым числом альтернативных ключей;

– первичный и альтернативный ключ может состоять из одного атрибута или комбинации атрибутов;

– отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного;

– атрибуты, входящие в первичные или альтернативные ключи сущности, могут быть собственными для сущности или наследоваться через отношение (внешними);

– первичные или альтернативные ключи должны содержать только необходимые для однозначной идентификации атрибуты, то есть при исключении из ключа любого атрибута не все экземпляры сущности могут быть однозначно определены (правило наименьшего ключа);

– если первичный ключ состоит более чем из одного атрибута, то значение любого неключевого атрибута должно функционально зависеть от всего первичного ключа, то есть если первичный ключ известен, то известно значение каждого неключевого атрибута, и значение неключевого атрибута не может быть определено с помощью только части первичного ключа (правило полной функциональной зависимости);

– каждый неключевой атрибут должен функционально зависеть только от первичного и альтернативных ключей, то есть значение неключевого атрибута не может определяться значением другого неключевого атрибута (правило отсутствия транзитивной зависимости).

Выбор первичного ключа для сущности является очень важным шагом, и требует большого внимания. В качестве первичных ключей могут быть использованы несколько атрибутов или групп атрибутов.

Атрибуты, которые могут быть выбраны первичными ключами, называются кандидатами в ключевые атрибуты (потенциальные атрибуты). Кандидаты в ключи должны уникально идентифицировать каждую запись сущности. В соответствии с этим, ни одна из частей ключа не может быть NULL, не заполненной или отсутствующей.

Рекомендации при выборе первичного ключа:

– использование потенциального ключа с минимальным набором атрибутов;

– использование того потенциального ключа, вероятность изменения значений которого минимальна;

– выбор того потенциального ключа, который имеет минимальную вероятность потери уникальности значений в будущем;

– использование потенциального ключа, значения которого имеют минимальную длину (в случае текстовых атрибутов);

– выбор того потенциального ключа, с которым будет проще всего работать (с точки зрения пользователя).

Для наглядного представления о том, как целесообразно выбирать первичные ключи, приведем следующий пример – выберем первичный ключ для знакомой нам сущности «Студент»:

Атрибут «Номер по порядку» является потенциальным ключом, так как он уникален для всех экземпляров сущности Студент.

Атрибут «Имя студента» не подходит для потенциального ключа, так как среди студентов могут быть студенты с одинаковыми именами.

Атрибут «Номер страхового полиса студента» является уникальным, но проблема в том, что студент может не иметь такового.

Комбинация атрибутов «Имя студента» и «Дата рождения студента» может стать потенциальным ключом.

Так как атрибут «Номер студента» имеет самые короткие и уникальные значения, то он лучше других подходит для первичного ключа.

При выборе первичного ключа для сущности, разработчики модели часто используют дополнительный (суррогатный) ключ, то есть произвольный номер, который уникальным образом определяет запись в сущности. Атрибут «Номер студента» является примером суррогатного ключа. Суррогатный ключ лучше всего подходит на роль первичного ключа потому, что является коротким и быстрее всего идентифицирует экземпляры в объекте. К тому же суррогатные ключи могут автоматически генерироваться системой так, чтобы нумерация была сплошной.

Отношение связи, называемое также отношением родитель-потомок –это связь между сущностями, при которой каждый экземпляр сущности-родитель ассоциирован с произвольным (в том числе нулевым) количеством экземпляров сущности-потомком, а – каждый экземпляр сущности-потомка ассоциирован в точности с одним экземпляром сущности-родителя.

Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае – неидентифицирующей.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком с точкой на конце линии у сущности-потомка. Отношения (связи) между сущностями, изображаемые соединяющими блоки линиями.

Выделяют следующие виды отношений:

- отношения, идентифицирующие связи;
- отношения, не идентифицирующие связи;
- отношения категоризации; неспецифические отношения.

Обозначения связей:

 - идентифицирующая связь

 - неидентифицирующая связь

Отношению дается имя, выражаемое грамматическим оборотом глагола. Имя отношения всегда формируется с точки зрения родителя, так что может быть образовано предложение, если соединить имя сущности-родителя, имя отношения, выражение мощности и имя сущности-потомка.

Например: Группа <состоит из> нескольких Студентов

Преподаватель <преподает> нескольких Дисциплин.

Мощность определяет, какое количество экземпляров сущности-потомка может существовать для каждого экземпляра сущности-родителя.

Различают четыре типа мощности:

- общий случай, когда одному экземпляру родительской сущности соответствуют 0, 1 или много экземпляров дочерней сущности не помечается каким-либо символом (эта мощность устанавливается по умолчанию);

- символом P (positive) помечается случай, когда одному экземпляру родительской сущности соответствуют 1 или много экземпляров дочерней сущности (исключено нулевое значение);

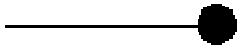
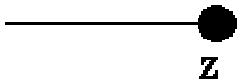
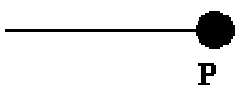
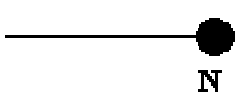
- символом Z (zero) помечается случай, когда одному экземпляру родительской сущности соответствуют 0 или 1 экземпляр дочерней сущности (исключены множественные значения);

- цифрой помечается случай точного соответствия, когда одному экземпляру родительской сущности соответствует заранее заданное число экземпляров дочерней сущности.

Для обозначения типов связей между сущностями в IDEF1X используется понятие кардинальной связи. Обозначение и виды кардинальности связей приведено в таблице 3.1

Таблица 3.1.

Обозначение кардинальных связей

Элемент диаграммы	Обозначает
	1,1
	0,M
	0,1
	1,M
	точно N (N – произвольное число)

Связи в IDEF1X представляют собой ссылки, соединения и ассоциации между сущностями (рисунок 3.4)

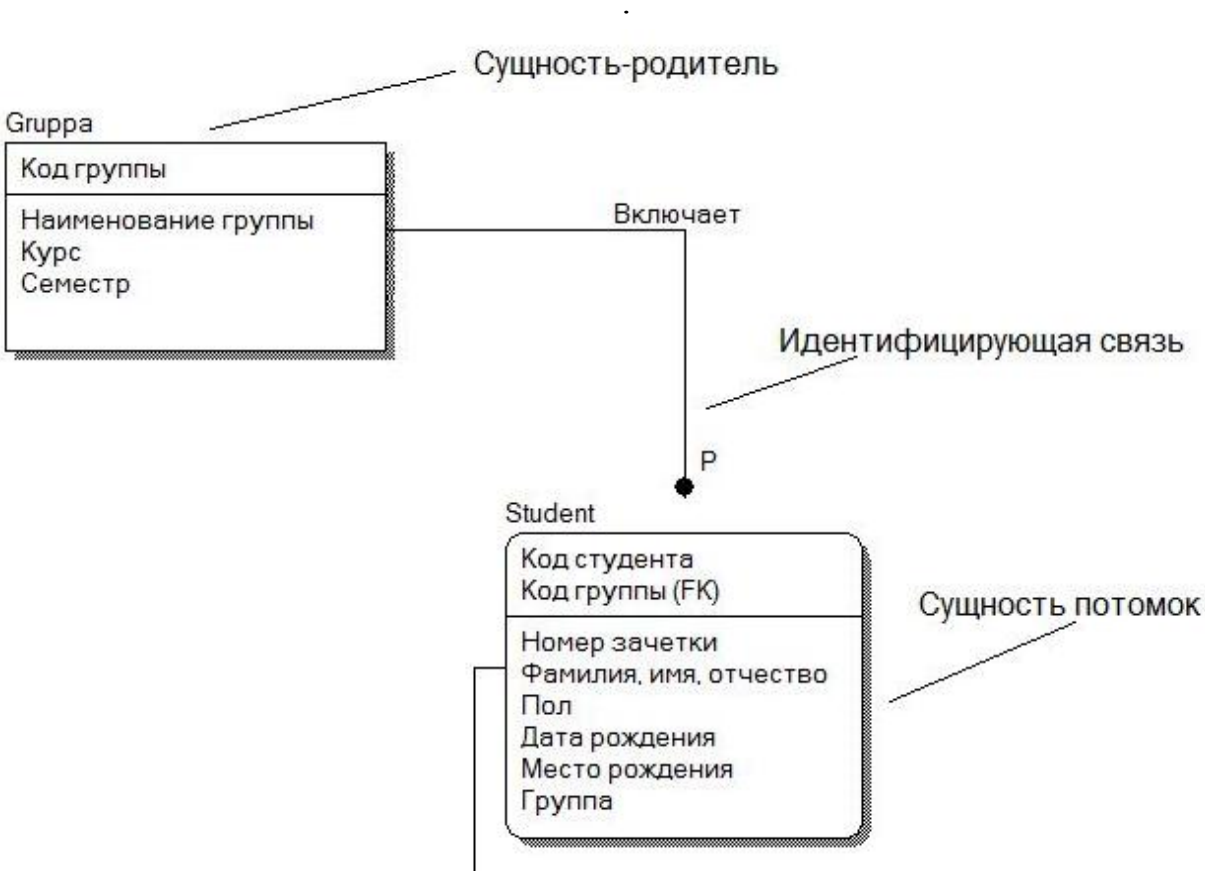


Рисунок 3.4. *– Фрагмент диаграммы*

Сущность-потомок в идентифицирующей связи является зависимой от идентификатора сущностью. Сущность-родитель в идентифицирующей связи может быть как независимой, так и зависимой от идентификатора сущностью (это определяется ее связями с другими сущностями).

Сущность-потомок в неидентифицирующей связи будет независимой от идентификатора, если она не является также сущностью-потомком в какой-либо идентифицирующей связи.

Отношения полной категоризации – это отношение между двумя или более сущностями, в котором каждый экземпляр одной сущности, называющейся общей сущностью, связан в точности с одним экземпляром одной и только одной из других сущностей, называемых сущностями-категориями (рисунок 3.5).

Каждый экземпляр общей сущности и связанный с ним экземпляр одной из категорных сущностей изображают один и тот же предмет реального мира и поэтому обладают одним и тем же уникальным идентификатором.

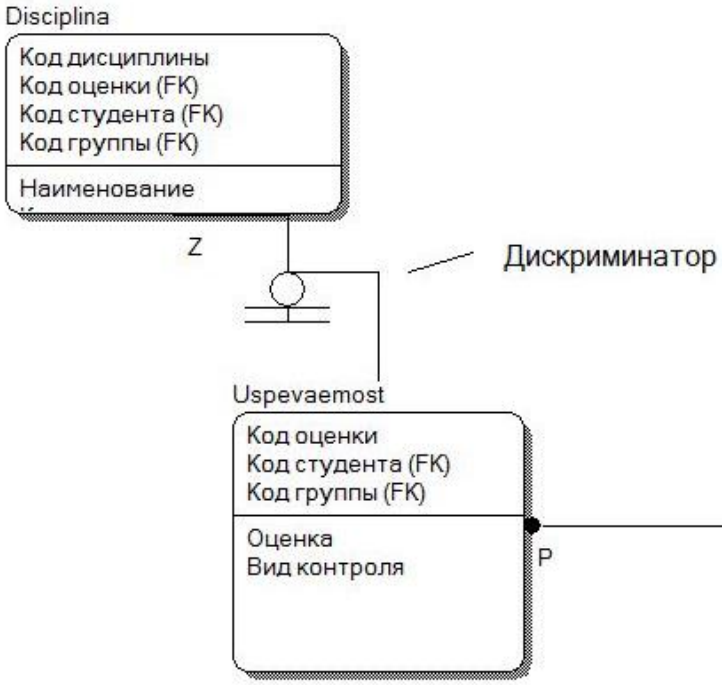


Рисунок 3.5. *– Отношение полной категоризации*

Для сущности-категории мощность не указывается, поскольку она всегда равна нулю или единице. Например, Читатель является общей сущностью, а Иностранная книга и Отечественная книга являются сущностями-категориями. Значение некоторого атрибута в экземпляре общей сущности определяет, с каким из возможных сущностей-категорий он связан. Этот атрибут называется дискриминатором отношения категоризации. В данном примере дискриминатором является Язык книги. Если существует экземпляр общей сущности, не связанный ни с каким экземпляром из сущностей-категорий, то такое отношение называется отношением неполной категоризации. На неполноту множества категорий на диаграмме указывает круг, подчеркнутый один раз.

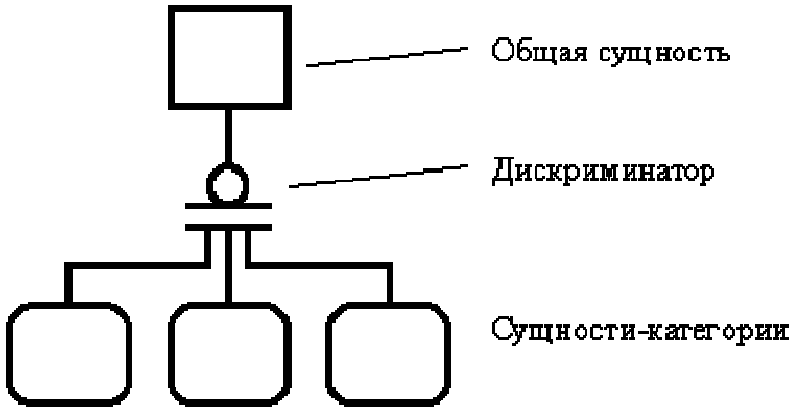
Неспецифическое отношение, называемое также отношением многого ко многому, – это связь между двумя сущностями, при которой каждый экземпляр первой сущности связан с произвольным (в том числе нулевым) количеством экземпляров второй сущности, а каждый экземпляр второй сущности связан с произвольным (в том числе нулевым) количеством экземпляров первой сущности. Например, если студент может быть занят во многих студенческих проектах, а в студенческом проекте может быть занято много студентов, то отношение между сущностями Студент и Проект является неспецифическим отношением. На более поздних стадиях построения модели все неспецифические отношения должны быть детализированы в специфические.

Имя связи – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим идентифицирующей или неидентифицирующей достаточно указать имя, характеризующее отношение от родительской к дочерней сущности. Для связи многие-ко-многим следует указывать имена как от родительской к дочерней сущности так и от дочерней сущности к родительской.

В IDEF1X отношение категоризации, по смыслу эквивалентно иерархической связи между сущностями.

В нотации IDEF1X иерархия категорий может быть двух типов – полная и неполная. Определить, какой тип иерархии категорий представлен в модели данных, можно по стилю отображения дискриминанта: дискриминант с двумя горизонтальными линиями свидетельствует о полноте иерархии категорий, с одной горизонтальной линией – о ее неполноте. Полная иерархия категорий свидетельствует о завершенности анализа. Например, полная иерархия категорий в сущности преподаватель вуза говорит о том, что любой преподаватель является либо штатным сотрудником, либо работающим по совместительству, либо с оплатой по часам и никаких других вариантов быть не может. Если мы изменим иерархию категорий на неполную, это будет свидетельствовать о том, что помимо преподавателей, зачисленных в штат, работающих по совместительству или с почасовой оплатой, могут быть и другие типы преподавателей, но на данном этапе моделирования данных, они еще не выявлены.

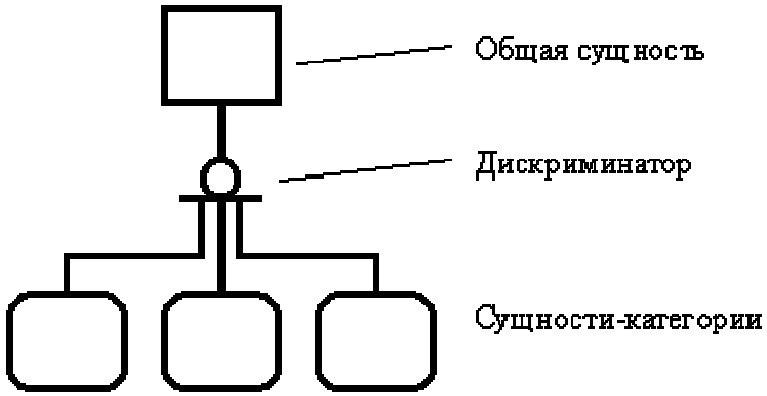
Отношение полной категоризации:



IDEF1X является методом для разработки реляционных баз данных и использует условный синтаксис, специально разработанный для удобного построения концептуальной схемы. Концептуальной схемой мы называем универсальное представление структуры данных в рамках проекта, независимое от конечной реализации базы данных и аппаратной платформы. Будучи статиче-

ским методом разработки, IDEF1X изначально не предназначен для динамического анализа по принципу "AS IS", тем не менее, он иногда применяется в этом качестве, как альтернатива методу IDEF1. Использование метода IDEF1X наиболее целесообразно для построения логической структуры базы данных после того, как все информационные ресурсы исследованы (например, с помощью метода IDEF1) и решение о внедрении реляционной базы данных, как части информационной системы, было принято.

Отношение неполной категоризации:



4. CASE – СРЕДСТВО ВЕРХНЕГО УРОВНЯ BPWIN

4.1. Характеристики BPwin

BPwin это инструмент визуального моделирования информационных систем, позволяющий [3, 13, 14]:

– наглядно описывать, анализировать и совершенствовать сложные процессы, деятельность или структуру в виде модели, что позволяет значительно повысить эффективность работы организации;

– проверить модель на соответствие стандартам ISO9000;

– спроектировать структуру информационных потоков, а соответственно, и модернизировать структуру организации;

– четко выявить факторы, оказывающие влияние на функционирование организации;

– выявить и исключить лишние или неэффективные операции.

BPwin входит в семейство продуктов AllFusion компании Computer Associates под названием AllFusion Process Modeler и предназначен для поддержки всех стадий жизненного цикла разработки информационных систем.

В линейку продуктов AllFusion Modeling Suite кроме BPwin для поддержки всех стадий разработки программного обеспечения, входят CASE-средств ERwin, ModelMart, Paradigm Plus, ERwin Examiner и средства управления проектами. При этом не накладываются ограничения на выбор базовых технологий, методов и платформ разработки. AllFusion Modeling Suite предлагает моделирование и управление процессами, проектами, изменениями, конфигурациями.

BPwin – инструмент моделирования, который используется для анализа и документирования и реорганизации сложных процессов. BPwin соответствует требованиям к инструментам для разработки информационных систем. BPwin является интуитивно понятным визуальным инструментом, позволяющим сформировать модели простых и сложных иерархических структур. Кроме того, BPwin – мощное средство моделирования процессов при создании корпоративных информационных систем.

Основные характеристики BPwin

– поддерживает три стандартные нотации: IDEF0, DFD, IDEF3, что обеспечивает комплексное описание предметной области;

– обладает редакторами для описания операций/функций и связей;

– обеспечивает создание структуры диаграмм, облегчающих последовательное уточнение элементов модели;

– позволяет разрабатывать диаграммы: контекстные (для описания границ системы, области действия, назначения объектов), декомпозиции (для описания особенностей взаимодействия различных процессов), диаграммы FEO (For Exposition Only), позволяющие произвести анализ вариантов, не внося изменений в основную модель, диаграммы Swim Lane – для координации сложных процессов и функциональных ограничений;

– имеет расширенные возможности по представлению диаграмм с возможностью проверки синтаксиса создаваемых моделей и ссылочной целостности;

– обладает средствами объединения для параллельной разработки моделей и разграничения процессов;

– осуществляет экспорт моделей в средства имитационного моделирования;

– обеспечивает интеграцию и связь со средством проектирования баз данных ERwin (методология IDEF1X), которые позволяют сократить время проектирования и разработки сложных ИС;

– обладает удобным интерфейсом пользователя;

– поддерживает свойства, задаваемые пользователем, что позволяет производить расширенное описание моделей, включая мультимедийные документы;

– содержит собственный генератор отчётов, который может создавать отчеты в формате MS Excel и Word для последующей обработки и использования в других приложениях;

– позволяет облегчить сертификацию на соответствие стандартам качества ISO9000.

Для оценки разработанной модели BPwin предоставляет два инструмента – стоимостный анализ, основанный на работах (Activity Based Costing, ABC), и свойства, определяемые пользователем (User Defined Properties, UDP).

ABC является распространенной методикой, используемой организациями для идентификации движителей затрат в организации. Стоимостный анализ представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость процесса. Он основан на модели работ. Обычно ABC применяется для того, чтобы понять происхождение затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Re-engineering, BPR). С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента,

идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь) в каждой из моделей AS-IS и TO-BE.

4.2. Общие принципы построения модели с использованием BPwin

Основу BPwin составляет графический язык описания моделирования бизнес-процессов. Модель в нотации IDEF0 представляется совокупностью иерархически упорядоченных и логически связанных диаграмм. Каждая диаграмма располагается на отдельном листе. Можно выделить четыре типа диаграмм:

– контекстную диаграмму A-0 (в каждой модели может быть только одна контекстная диаграмма);

– диаграммы декомпозиции (в том числе диаграмма первого уровня декомпозиции A0, раскрывающая контекстную);

– диаграммы дерева узлов;

– диаграммы только для экспозиции (FEO).

Контекстная диаграмма является вершиной древовидной структуры диаграмм и представляет собой самое общее описание системы и ее взаимодействия с внешней средой.

После описания системы в целом проводится разбиение ее на крупные фрагменты. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент и взаимодействие фрагментов, называются диаграммами декомпозиции. После декомпозиции контекстной диаграммы (то есть диаграммы A0) проводится декомпозиция каждого блока диаграммы A0 на более мелкие фрагменты и так далее, до достижения нужного уровня подробности описания.

После каждого сеанса декомпозиции проводятся сеансы экспертизы. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколько угодно, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения, либо для специальных целей.

Таким образом, процесс моделирования системы в BPwin начинается с наиболее абстрактного уровня описания системы в целом. В этот уровень входят определение субъекта моделирования, цели и точки зрения на модель. Под субъектом понимается сама система.

На этом этапе важно разделить компоненты системы и внешние воздействия, то есть четко ограничить субъект.

Субъект моделирования. Как уже отмечалось, на определение субъекта системы будет существенно влиять позиция, с которой рассматривается система, и цель моделирования. Другими словами, первоначально необходимо определить область моделирования. Описание области как системы в целом, так и ее компонентов является основой построения модели. Хотя предполагается, что в течение моделирования область может корректироваться, она должна быть в основном сформулирована изначально, поскольку именно область определяет направление моделирования и когда должна быть закончена модель. При формулировании области необходимо учитывать два компонента: определение границ модели и уровня детализации модели (количество уровней декомпозиции). После определения границ модели предполагается, что новые объекты не должны вноситься в моделируемую систему, так как эти элементы будут вносить изменения в существующие взаимосвязи.

Цель моделирования (Purpose). Модель не может быть построена без четко сформулированной цели. Цель должна отвечать на следующие вопросы: необходимость моделирования процесса, что должна показывать модель, где могут быть использованы результаты.

Примерами формулирования цели могут быть следующие утверждения: «Описать структуру контроля качества обучения», «Идентифицировать функции и обязанности пользователей Интранет-портала», «Идентифицировать функции системы контроля остаточных знаний студентов с целью расширения ее функциональных возможностей».

Точка зрения (Viewpoint). Модель должна строиться с единой точки зрения. Точку зрения можно представить как взгляд человека, который видит систему в нужном для моделирования аспекте. Точка зрения должна соответствовать цели моделирования. Как правило, выбирается точка зрения человека, ответственного за моделируемую работу в целом. Часто при выборе точки зрения на модель важно учесть дополнительные альтернативные точки зрения. Для этой цели обычно используют диаграммы FEO (For Exposition Only), которые будут описаны в дальнейшем.

4.3. Построение модели с использованием BPwin

После запуска программы на экране появиться диалоговое окно, в котором следует выбрать режим работы: либо создать новую модель (Create model), либо открыть существующую модель (Open model). При создании новой модели определяется требуемая нотация (в нашем случае IDEF0) (рисунок 4.1).

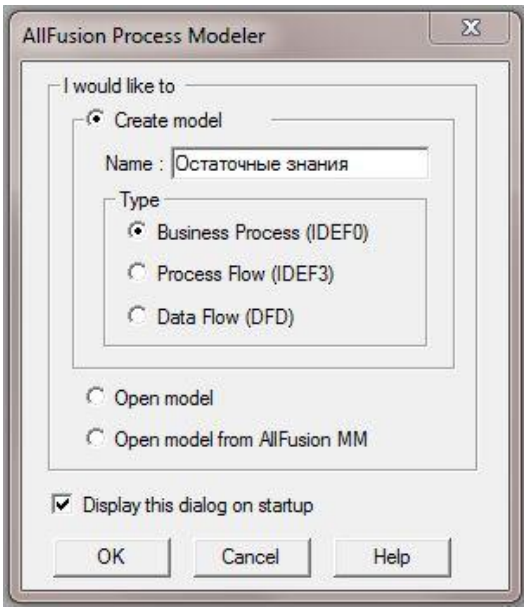


Рисунок 4.1. *– Диалоговое окно начала работы*

После ввода данных автора отображается диаграмма A-0 (рисунок 4.2)

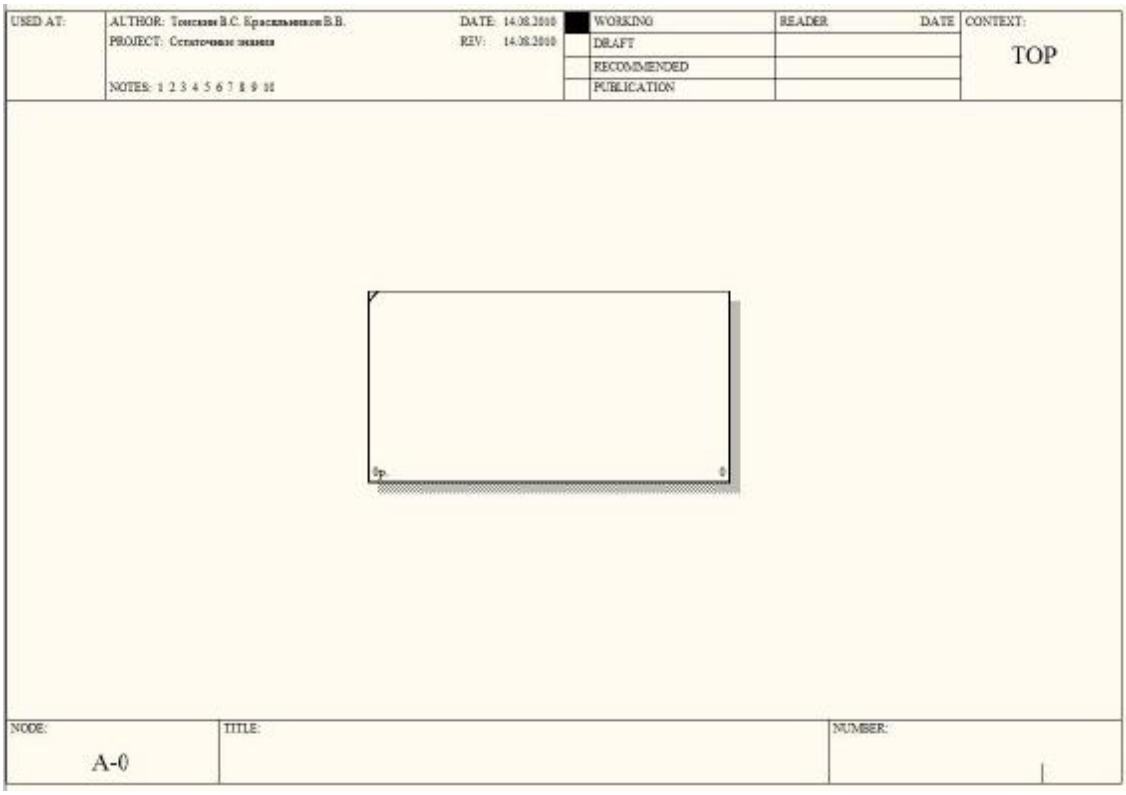


Рисунок 4.2. *– Диаграмма A-0*

Для создания стрелок используется панель инструментов редактора BPwin (рисунок 4.3)

***Рисунок 4.3**. – Панель инструментов редактора BPwin*

Слева направо обозначены.

Pointer Tool – используется для выбора и определения позиции объектов, добавленных в диаграмму.

Activity Box Tool – используется для установки блоков в диаграмме.

Arrow Tool – используется, чтобы устанавливать стрелки в диаграмме.

Squiggle Tool – используется для создания тильды (squiggle), которая соединяет дугу с ее названием.

Text Block Tool – используется для создания текстовых блоков.

Diagram Dictionary Editor – открывает диалоговое окно Diagram Dictionary Editor, с помощью которой можно перейти на имеющуюся диаграмму или создать новую диаграмму.

Go to Sibling Diagram – используется для отображения следующей диаграммы того же уровня.

Go to Parent Diagram – переход на родительскую диаграмму.

Go to Child Diagram – используется, чтобы отобразить диаграмму потомка или разложить выделенный блок на диаграмму потомка.

IDEF0-модель предполагает наличие четко сформулированной цели, единственного субъекта моделирования и одной точки зрения. Для внесения области, цели и точки зрения в модели IDEF0 в BPwin следует выбрать пункт меню Model/Model Properties…, вызывающий диалог Model Properties. Закладка General служит для внесения имени проекта и модели, имени и инициалов автора и временных рамок, модели – AS-IS и TO-BE (рисунок 4.4). В закладке Purpose следует внести цель и точку зрения (рисунок 4.5), а в закладку Definition – определение модели и описание области (рисунок 4.6.).

В закладке **Source** описываются источники информации для построения модели (например, "Опрос экспертов предметной области и анализ документации").

В закладке **Status** можно описать статус модели (черновой вариант, рабочий, окончательный и т.д.), время создания и последнего редактирования (отслеживается в дальнейшем автоматически по системной дате).

Результат описания модели можно получить в отчете Model Report. Диалог настройки отчета по модели вызывается из пункта меню Tools/Reports/ModelReport. В

диалоге настройки следует выбрать необходимые поля (при этом автоматически отображается очередность вывода информации в отчет) (рисунок 4.7)

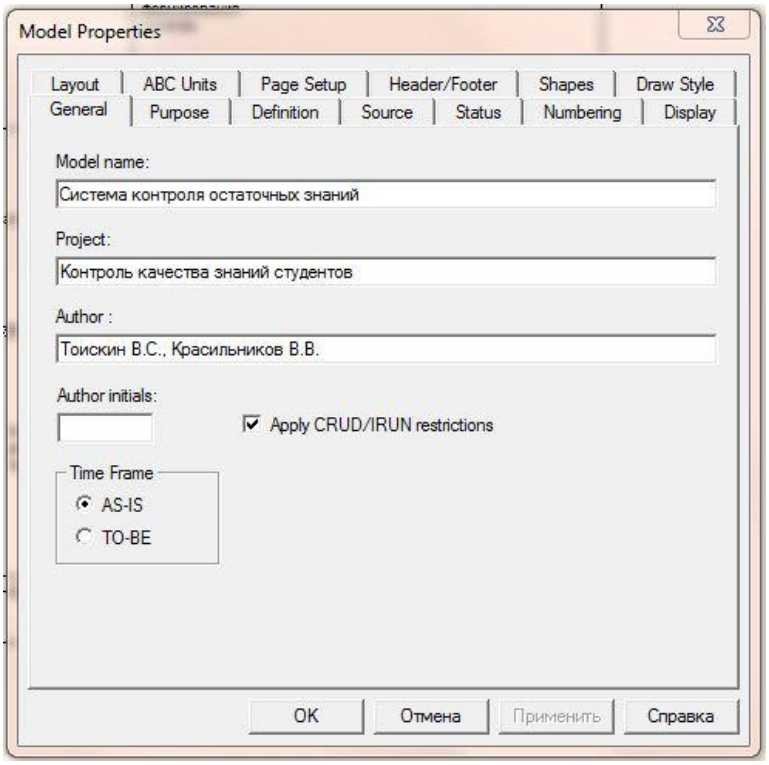


Рисунок 4.4.- *Диалоговое окно свойств модели. Главная*

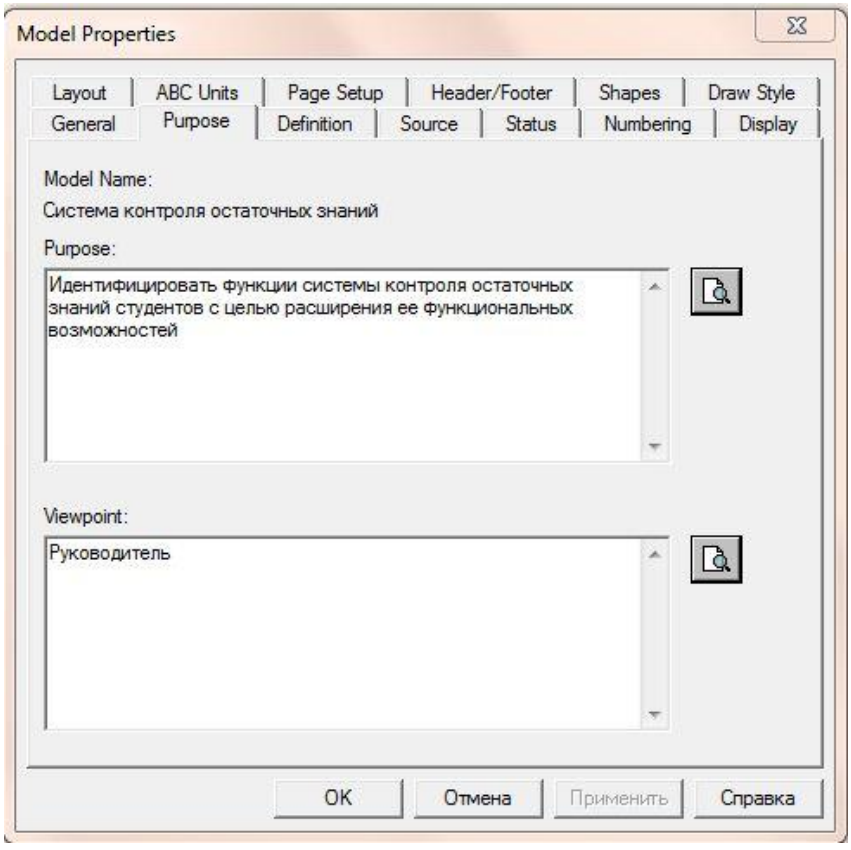


Рисунок 4.5.- *Диалоговое окно свойств модели. Цель и точка зрения.*

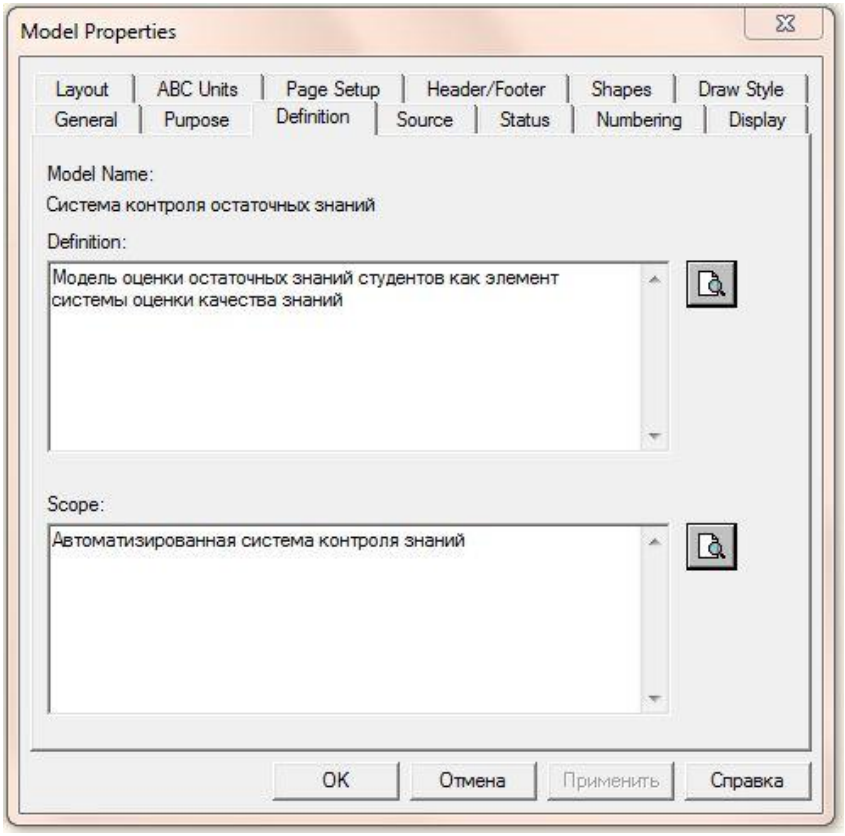


Рисунок 4.6.- *Диалоговое окно свойств модели. Определение модели и описание области.*

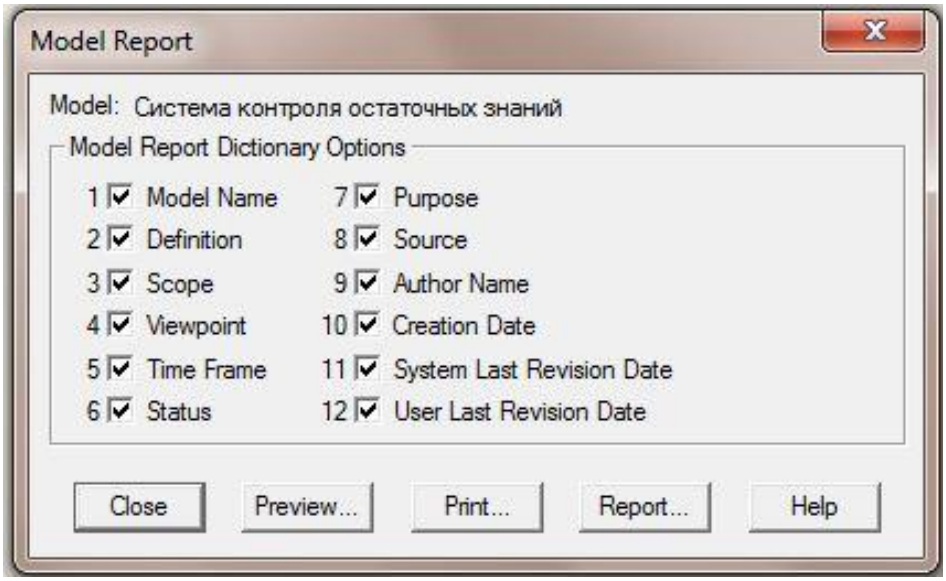


Рисунок 4.7. *– Диалоговое окно Model Report.*

При необходимости просмотреть отчет необходимо использовать кнопку Preview.. данного диалогового окна (рисунок 4.8)

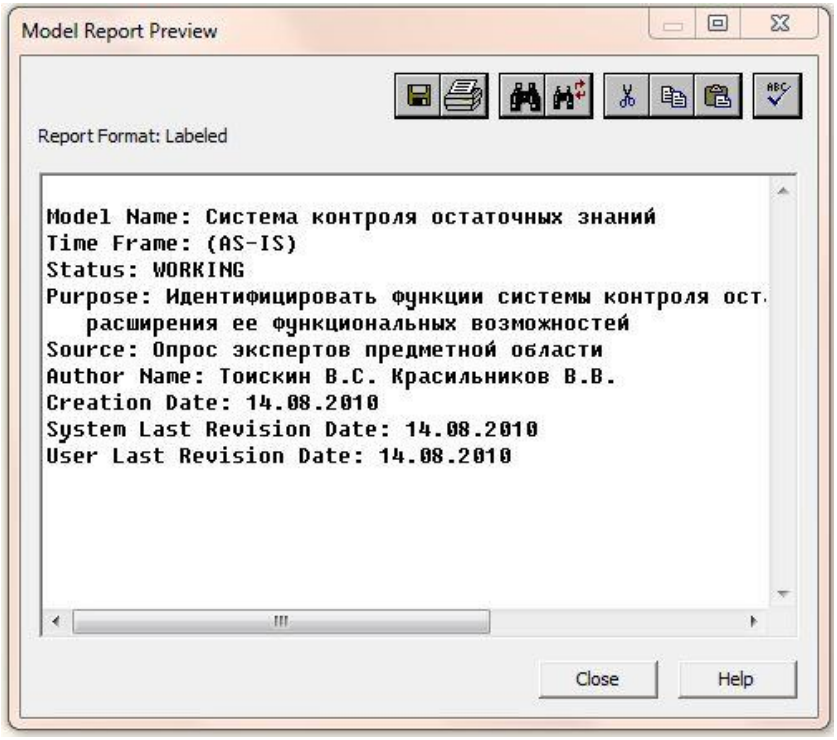


Рисунок 4.8. *– Диалоговое окно Model Report Preview.*

Первая диаграмма в иерархии диаграмм IDEF0 всегда изображает функционирование системы в целом. Такие диаграммы называются контекстными. В контекст входит описание цели моделирования, области и точки зрения (рисунок 4.9).

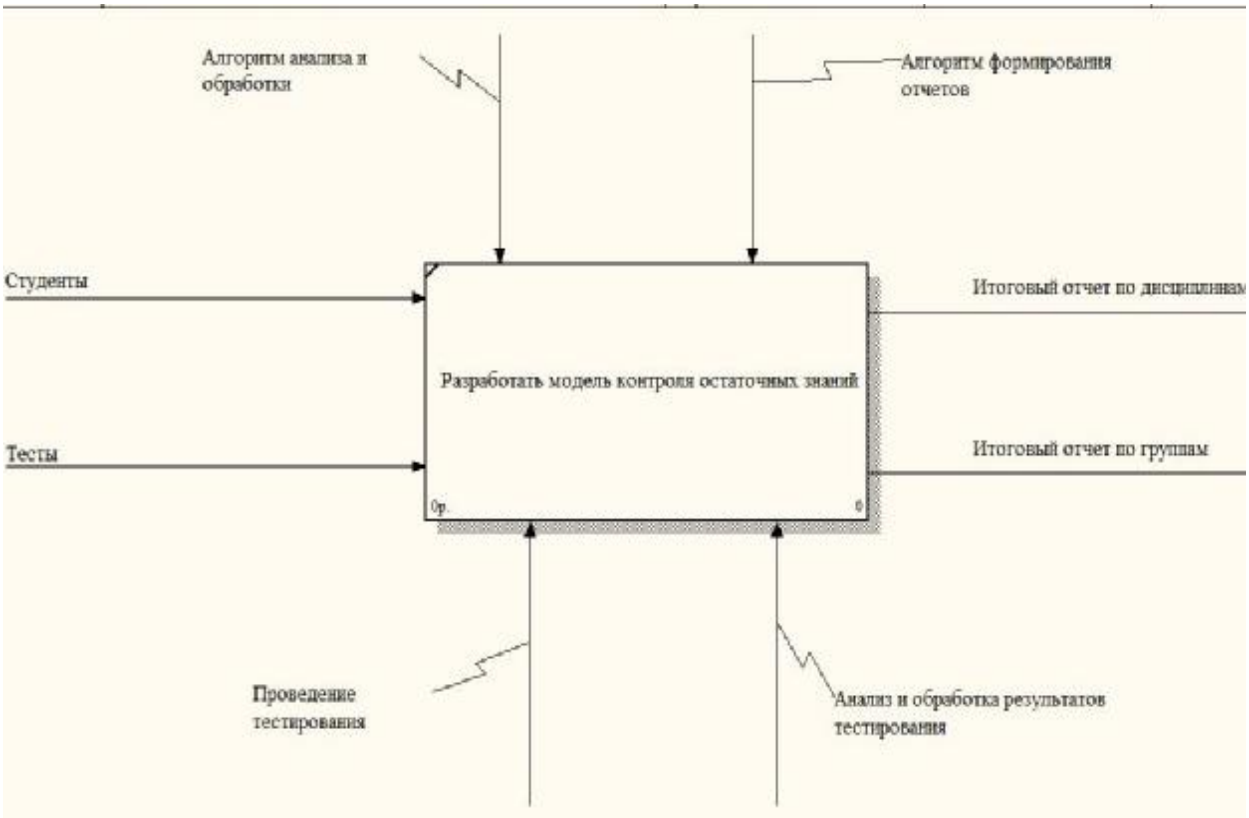


Рисунок 4.9. *– Контекстная диаграмма.*

Каждая диаграмма располагается внутри бланка имеющего несколько информационных полей:

Поля верхней части рамки.

Used At (Используется в) – используется для указания на родительский блок в случае, если на текущую диаграмму ссылались посредством стрелки вызова.

Author (Автор) – имя разработчика диаграммы.

Date (Дата) – дата создания.

Project (Проект) – имя проекта.

Rev (Пересмотрено) – дата последнего редактирования диаграммы.

Notes 12345678910 (Замечания) – используется при проведении сеанса экспертизы. Эксперт должен (на бумажной копии диаграммы) указать число замечаний, вычеркивая цифру из списка каждый раз при внесении нового замечания.

Status (Статус) – статус отображает стадию создания диаграммы, отображая все этапы публикации: Working (Рабочая версия); Draft (Эскиз) – диаграмма прошла первичную экспертизу и готова к дальнейшему обсуждению; Recommended (Рекомендовано) – диаграмма и все ее сопровождающие документы прошли экспертизу; Publication (Публикация) – диаграмма готова к окончательной печати и публикации; Reader (Читатель) – имя читателя (эксперта); Date (Дата) – дата прочтения (экспертизы).

Context (Контекст) – схема расположения работ в диаграмме верхнего уровня. Работа, являющаяся родительской, показана темным прямоугольником, остальные – светлым. На контекстной диаграмме (A-0) показывается надпись TOP. В левом нижнем углу показывается номер по узлу родительской диаграммы.

Поля нижней части рамки

Node (Узел) – номер узла диаграммы (номер родительского блока).

Title (Название) – имя диаграммы. По умолчанию – имя родительского блока.

Number (Номер) – C-номер, уникальный номер версии диаграммы.

Page (Страница) – номер страницы, может использоваться при формировании папки.

В диаграммах процессов по методологии IDEF0 стрелки, как и блоки, имеют свою иерархию. Можно произвести декомпозицию единственной стрелки контекстной диаграммы на множество стрелок в диаграммах декомпозиции. Возможно и обратное действие, при котором производится объединение выходных стрелок диаграмм декомпозиции в единую стрелку на контекстной диаграмме.

Подобно иерархии блоков на диаграммах процессов, в иерархию стрелок могут входить стрелки контекстного уровня, которые не подвергаются декомпо-

зиции и не являются результатом слияния двух или более стрелок диаграмм декомпозиции.

Перед добавлением стрелок в необходимо, прежде всего, определить, какие данные должны быть ассоциированы с конкретной стрелкой на диаграмме.

Правильно использование стрелок обеспечивает единство представления информации в диаграммах любого типа. Используя Model/ Default Arrow Types производится задание стиля для новых стрелок данной модели, как стиль стрелок, устанавливаемый по умолчанию.

В BPwin используются следующие типы стрелок:

Precedence – стрелка, изображаемая сплошной линией. Это наиболее часто используемый тип стрелки для указания связи между двумя объектами диаграммы в любом направлении.

Relational – стрелка изображаемая пунктирной линией. Используется для соединения ссылок в UOW .

Object Flow –связь изображаемая сплошной линией с двойной стрелкой;

Щелчком правой кнопки мыши по стрелки открывается контекстное меню стрелки, содержащее следующие важные опции:

Squiggle – создание тильды, соединяющего стрелку с ее названием

Trim – уменьшение длины стрелки

Arrow Tunnel – создание туннельной стрелки

Off Page Reference – создание стрелки межстраничных ссылок

External Reference – внешняя ссылка

External Definition – внешнее описание

Go to Reference – переход по ссылке

Открыть диалоговое окно Arrow Properties можно из контекстного меню стрелки, выбрав одну из первых девяти опций (рисунок 4.10).

Назначение вкладок.

1. Name – вкладка задания названия стрелки и фамилия автора, включает:

Arrow Name – текстовое поле для задания наименования стрелки.

Replace all occurrences of this arrow name in model – переключатель, обеспечивающий глобальное изменение имени стрелки на новое для всех вхождений данной стрелки в модели

Author – текстовое поле для задания/изменения фамилии автора

2. Style – вкладка задания графического стиля стрелки с помощью опций:

Thickness – зона, в которой содержатся опции задания/изменения толщины стрелок модели

Thickness – список, содержащий применяемые толщины стрелок

Apply thickness to all instances of arrow in the model – переключатель, обеспечивающий применение заданной толщины для всех вхождений стрелки в модели

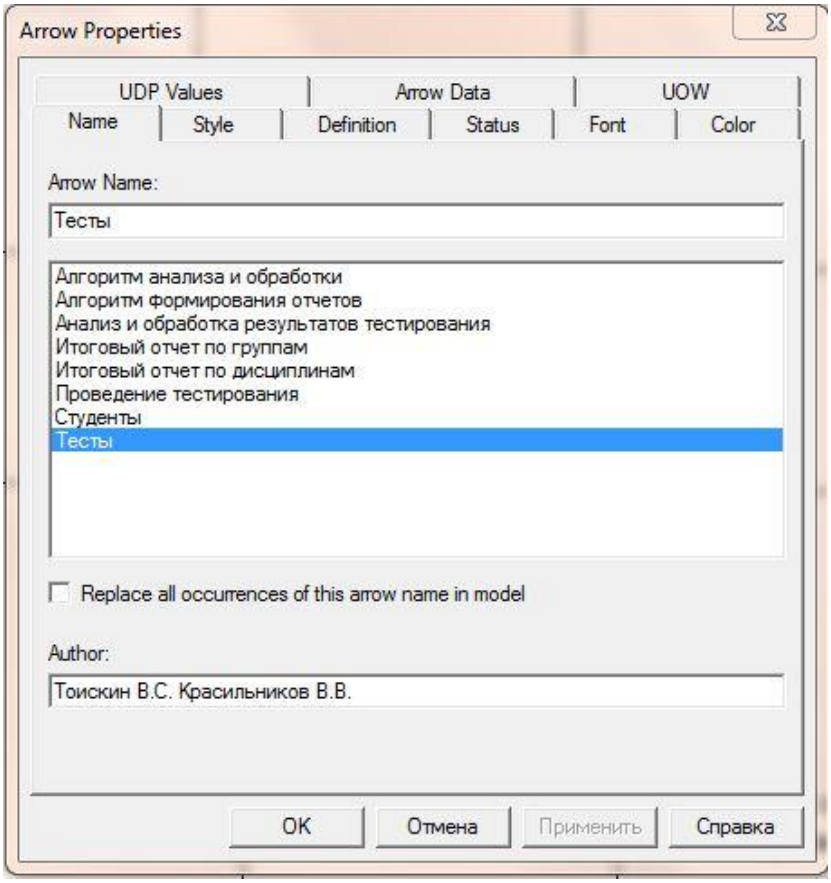


Рисунок 4.10. *– Диалоговое окно Arrow Properties (свойства стрелки)*

Set thickness as default for new arrows – переключатель, обеспечивающий задание принятой толщины стрелки для всех вновь создаваемых стрелок модели

Apply thickness to this arrow segment only – переключатель, обеспечивающий применение выбранной толщины стрелки только для указанного сегмента

Type – зона, содержащая опции задания/изменения типа применяемых стрелок в модели

Precedence –стрелка для указания связи между двумя объектами диаграммы;

Relational – стрелка связи между двумя объектами диаграммы, выполняемая в любых направлениях

Object Flow – стрелка потоков в диаграммах процессов, выполняемая в любом направлении

Apply thickness to all instances of arrow in the model – переключатель, обеспечивающий применение заданного типа стрелки для всех вхождений стрелки в модели

Set thickness as default for new arrows – переключатель, обеспечивающий заданного типа стрелки для всех вновь создаваемых стрелок модели

Apply thickness to this arrow segment only – переключатель, обеспечивающий применение заданного типа стрелки только для указанного сегмента

3. Definition – вкладка, на которой содержатся текстовые поля для задания описания стрелки.

4. Status – вкладка, на которой можно указать статус: Working , Draft , Recommended , Publication или Other

5. Font – вкладка, на которой задаются опции форматирования текстовых надписей.

6. Color – вкладка, на которой задаются опции применяемой цветовой гаммы.

7. UDP Values – вкладка, на которой задаются значения свойств, определяемые пользователем (UDP).

8. Arrow Data – вкладка, опции которой позволяют задать свойства и основные атрибуты стрелки:

Clear – кнопка, удаления всех сущностей и атрибутов, ассоциированных с данной стрелкой.

Migrate – кнопка распространение ассоциированных данных по иерархическому дереву стрелки.

Copy In – кнопка открытия диалогового окна Copy in arrow Data, содержащего перечень стрелок с ассоциированными сущностями и атрибутами, позволяет копировать ассоциированные сущности и атрибуты из одной стрелки в другую

Ent / Att Editor – кнопка открытия диалогового окна Entity and Attribute Dictionary Editor с помощью которого можно добавлять, изменять и удалять сущности и атрибуты:

9. UOW – вкладка, на которой вводится доменная информация, присваиваемая UOW объектам.

Пользователь должен правильно выбирать необходимый тип стрелки для соединения объектов диаграмм, создаваемых в BPwin .

Алгоритм создания стрелки:

– выполнить команду Model/ Default Arrow Types

– выбрать требуемый тип стрелки

– в открытой диаграмме щелкнуть по кнопке Arrow на панели инструментов BPwin

– щелкнуть по блоку или по другому объекту диаграммы – источнику стрелки, и щелкнуть по соответствующей подсвеченной стороне блока или границы диаграммы.

Для разветвления стрелки следует в режиме рисования стрелок щелкнуть по сегменту стрелки, которую нужно разветвить, и затем по соответствующей стороне (входа, управления или механизмов) работы-приемника ветви стрелки.

Для слияния стрелок следует в режиме рисования стрелок щелкнуть по стороне выхода работы-источника ветви стрелки и затем по сегменту стрелки, которую нужно слить с ветвью.

Для удаления блока (стрелки) необходимо его (ее) выделить с помощью мыши и нажать на клавишу Del.

Для отображения кодов ICOM на диаграммах модели необходимо включить опцию ICOM Codes на вкладке Display диалогового окна Model Properties. Данное окно может быть вызвано с помощью меню Model/Model Properties или нажатием правой кнопки мыши в свободной области диаграммы.

После того как контекст описан, проводится построение следующих диаграмм в иерархии. Каждая последующая диаграмма является более подробным описанием (декомпозицией) одной из работ на вышестоящей диаграмме. Пример декомпозиции контекстной работы показан на рис.4.11.

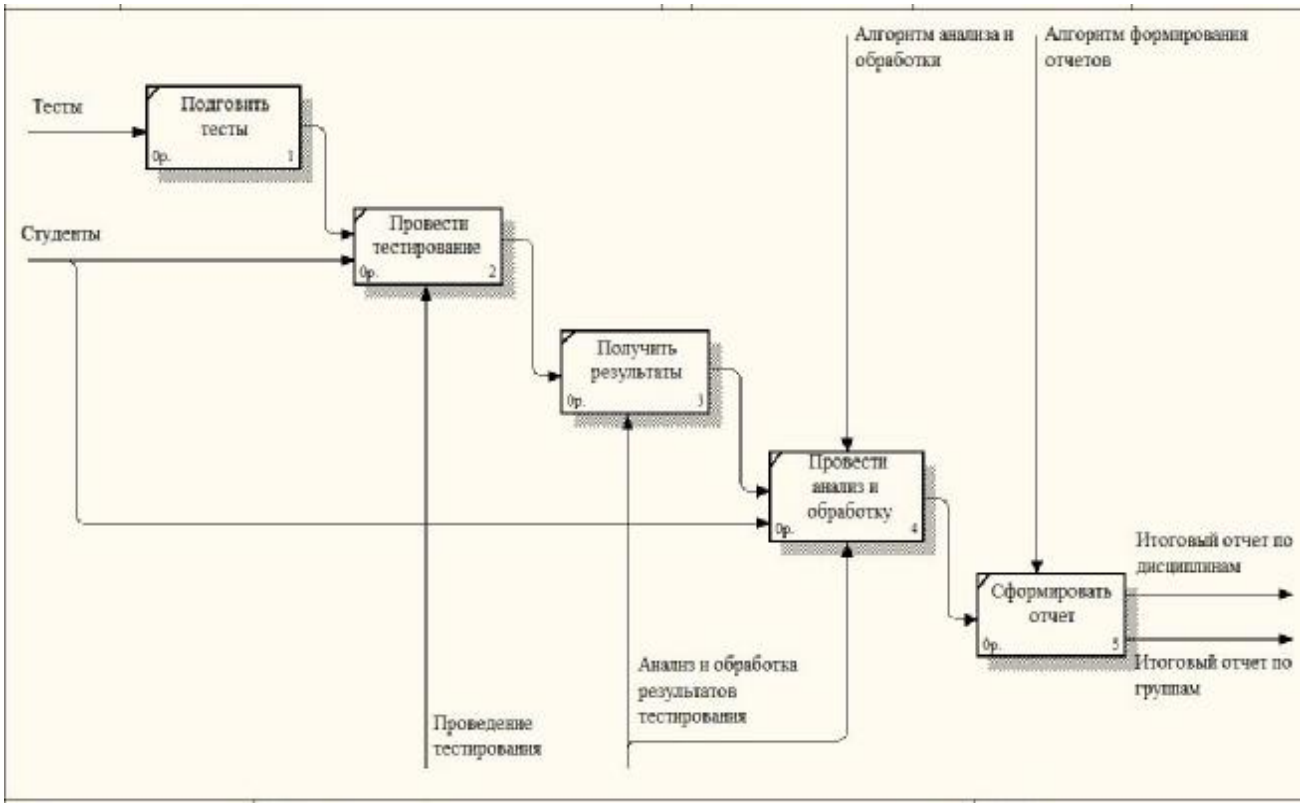


Рисунок 4.11. *– Модель A0*

Описание каждой подсистемы проводится разработчиком совместно с экспертом предметной области. Обычно экспертом является человек, отвечающий

за эту подсистему и, поэтому, знающий все ее функции. Таким образом, вся система разбивается на подсистемы до нужного уровня детализации, и получается модель, аппроксимирующая систему с заданным уровнем точности.

Все работы модели нумеруются. Номер состоит из префикса и числа. Может быть использован префикс любой длины, но обычно используют префикс А. Контекстная (корневая) работа дерева имеет номер А0. Работы декомпозиции А0 имеют номера Al, А2, А3 и т. д. Работы декомпозиции нижнего уровня имеют номер родительской работы и очередной порядковый номер, например работы декомпозиции А3 будут иметь номера А31, А32, А3З, А34 и т. д. Работы образуют иерархию, где каждая работа может иметь одну родительскую и несколько дочерних работ, образуя дерево. Такое дерево называют деревом узлов, а вышеописанная нумерация – нумерацией по узлам. Имеются незначительные варианты нумерации, которые можно настроить во вкладке Numbering диалога Model Properties.

BPwin автоматически поддерживает нумерацию по узлам, то есть при проведении декомпозиции создается новая диаграмма и ей автоматически присваивается соответствующий номер. В результате проведения экспертизы диаграммы могут уточняться и изменяться, следовательно, могут быть созданы различные версии одной и той же (с точки зрения ее расположения в дереве узлов) диаграммы декомпозиции. BPwin позволяет иметь в модели только одну диаграмму декомпозиции в данном узле. Прежние версии диаграммы можно хранить в виде бумажной копии либо как FEO-диаграмму.

В любом случае следует отличать различные версии одной и той же диаграммы. Для этого существует специальный номер – С-номер, который должен присваиваться автором модели вручную. С-номер – это произвольная строка, но рекомендуется придерживаться стандарта, когда номер состоит из буквенного префикса и порядкового номера, причем в качестве префикса используются инициалы автора диаграммы, а порядковый номер отслеживается автором вручную, например TBC00021.

Для рассматриваемого примера целесообразно провести декомпозицию работ А3, А4 и А5.

Диаграммы декомпозиции создаются с помощью диалогового окна Activity box Count , которое открывается щелчком по кнопке Go to Child на панели инструментов BPwin (рисунок 4.12)

Диалоговое окно содержит следующие опции:

– IDEF0, DFD, IDEF3 – переключатели выбора методологии моделирования;

– Include Externals & Data Stores – флажок, указывающий возможность проведения декомпозиции с учетом внешних данных;

– Number of Activities in this Decomposition – список, в котором указывается число блоков декомпозиции. Допустимый интервал числа блоков 3-6.

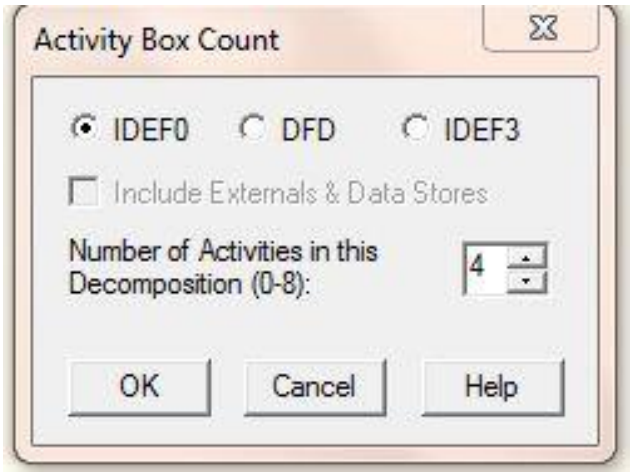


Рисунок 4.12. *– Диалоговое окно процесса декомпозиции*

Если в дальнейшем необходимо добавить блоки на диаграмме, то необходимо выбрать инструмент (Activity Box Tool) и щелкнуть мышью в нужном месте диаграммы.

После декомпозиции необходимо соединить получившиеся блоки стрелками. Для этого необходимо выбрать инструмент Precedence Arrow Tool, щелкнуть мышью по исходной стороне блока затем по конечной стороне следующего блока. Аналогично строятся разветвления и слияния стрелок.

Декомпозиция указанных работ приведена на рисунках 4.13, 4.14 и 4.15 соответственно.

Получив модель, адекватно отображающую текущие процессы (модель AS IS), можно определить недостатки системы контроля. После этого, с учетом выявленных недостатков, можно строить модель новой организации системы контроля остаточных знаний (модель TO BE).

Как уже отмечалось ранее модель, выполненная в BPwin, представляет собой набор иерархически упорядоченных диаграмм (не обязательно сделанных в одной методологии). При размещении на очередной диаграмме некоторого элемента (работы, стрелки…) этот элемент вместе со всеми своими свойствами (которые всегда можно просмотреть или изменить в соответствующем редакторе BPwin) автоматически заносится в словарь BPwin, в результате вместе с графическим изображением моделируемой системы разработчик получает текстовое описание системы.

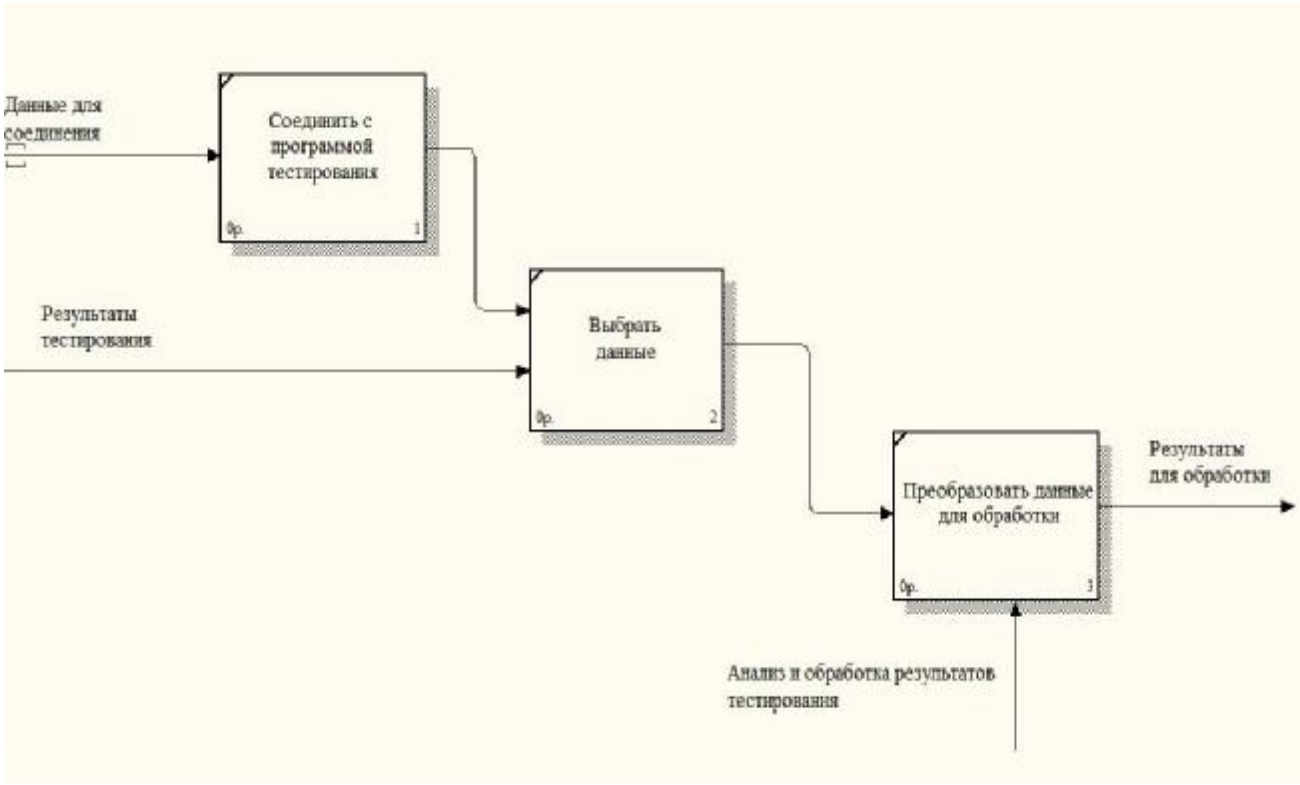


Рисунок 4.13. *– Декомпозиция Работы A3*

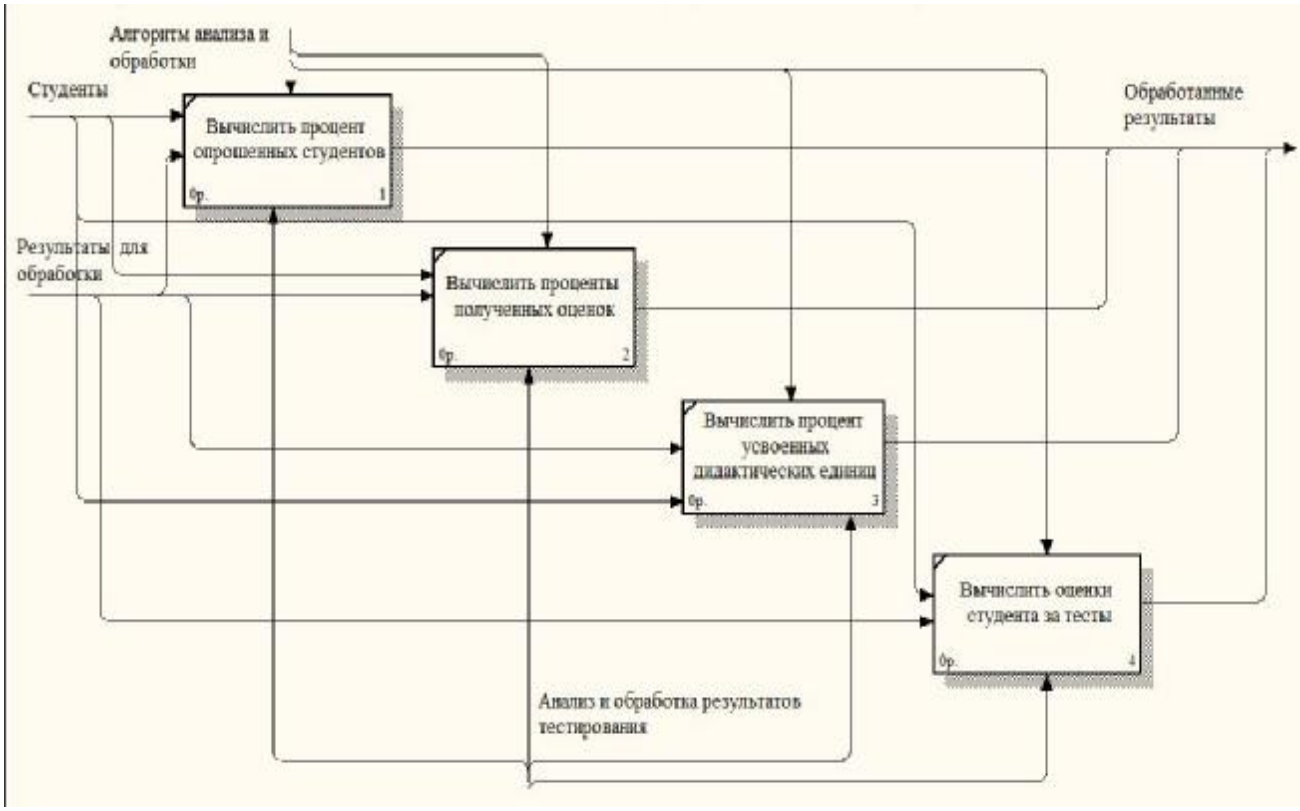


Рисунок 4.14*. – Декомпозиция Работы A4*

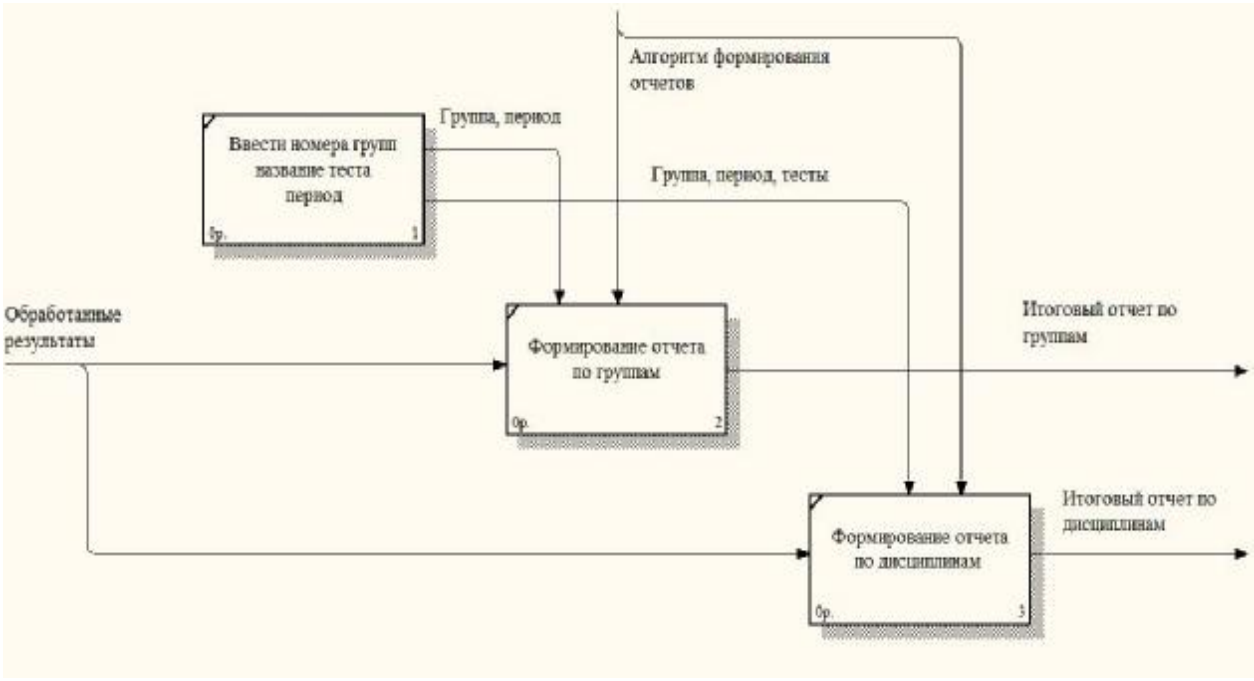


***Рисунок 4.15**. – Декомпозиция Работы A5*

Применение универсальных графических языков моделирования IDEF0, IDEF3 и DFD обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов. Интерактивное выделение объектов обеспечивает постоянную визуальную обратную связь при построении модели. BPwin поддерживает ссылочную целостность, не допуская определения некорректных связей и гарантируя непротиворечивость отношений между объектами при моделировании.

При построении модели BPwin удобным инструментом для навигации по уровням декомпозиции модели является Model Explorer (рисунок 4.16), который по организации очень похож на проводник Windows. Работы IDEF0 показываются в Model Explorer зеленым цветом, DFD - желтым и IDEF3 - синим.

С помощью вкладки объектов можно методом Drag&Drop размещать объекты из словаря на любой диаграмму.

С помощью вкладки диаграмм можно просматривать всю иерархию диаграмм, включая Organization Chart, Node Tree, Swim Lane, FEO, и IDEF3 Scenario.

BPwin имеет мощный инструмент отчетов Report Template Builder, с помощью которого можно легко и быстро создавать различные отчеты о модели. С его помощью можно также создавать шаблоны для отчетов, а также преобразовывать отчеты в формат txt, HTML или RTF.

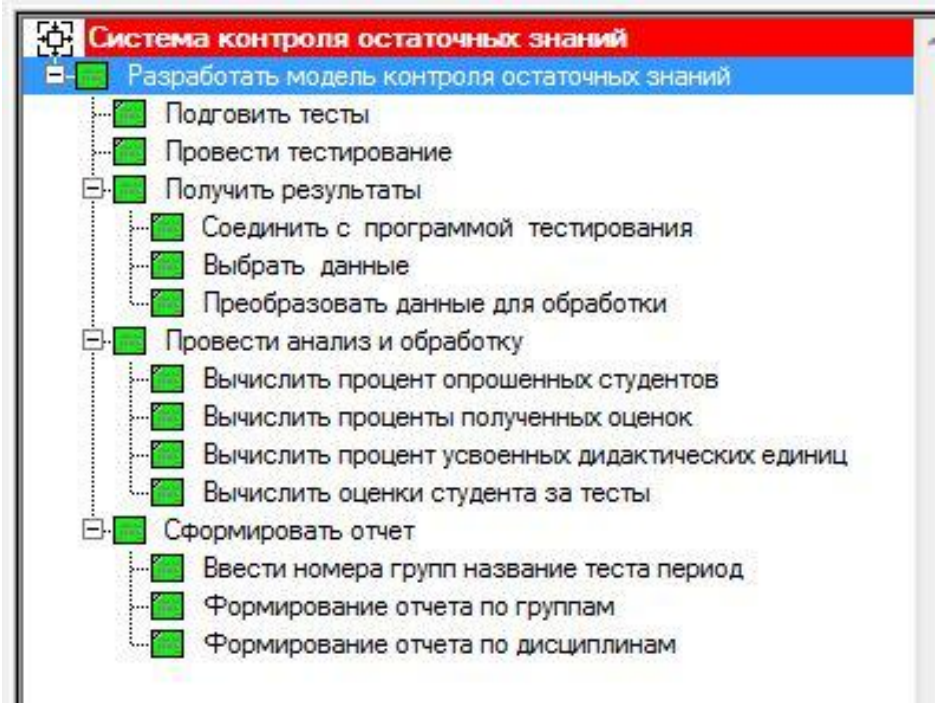


Рисунок 4.16. *– Model Explorer*

В дополнение к диаграммам IDEF0, DFD и IDEF3, BPwin поддерживает еще целый ряд вспомогательных диаграмм таких как:

Диаграммы дерева узлов (Node Tree Diagram). К модели BPwin можно добавлять дерево узлов, которое показывает иерархию всех работ модели на одной диаграмме.

Диаграмма дерева узлов имеет вид традиционного иерархического дерева, где верхний узел (прямоугольник) соответствует работе с контекстной диаграммы, а последующие нижние узлы представляют собой дочерние уровни декомпозиции.

Можно также создать диаграмму дерева узлов лишь для некоторой части модели, тогда верхним узлом диаграммы будет та работа декомпозиции, с которой вы захотите начать.

Для рассматриваемого примера диаграмма дерева узлов имеет вид (рисунок 4.17).

Прямоугольники в дереве узлов сохраняют за собой все свойства соответствующих им работ. Для того, чтобы открыть редактор свойств работы необходимо дважды щелкнуть мышкой по прямоугольнику работы. Если же вы дважды щелкнете мышкой по той части диаграммы, которая не занята работами, откроется редактор свойств самой диаграммы дерева узлов, где можно установить такие свойства диаграммы как ее имя, шрифт и цвет.

Просмотреть диаграмму дерева узлов можно с помощью Model Explorer и вкладки Diagrams.

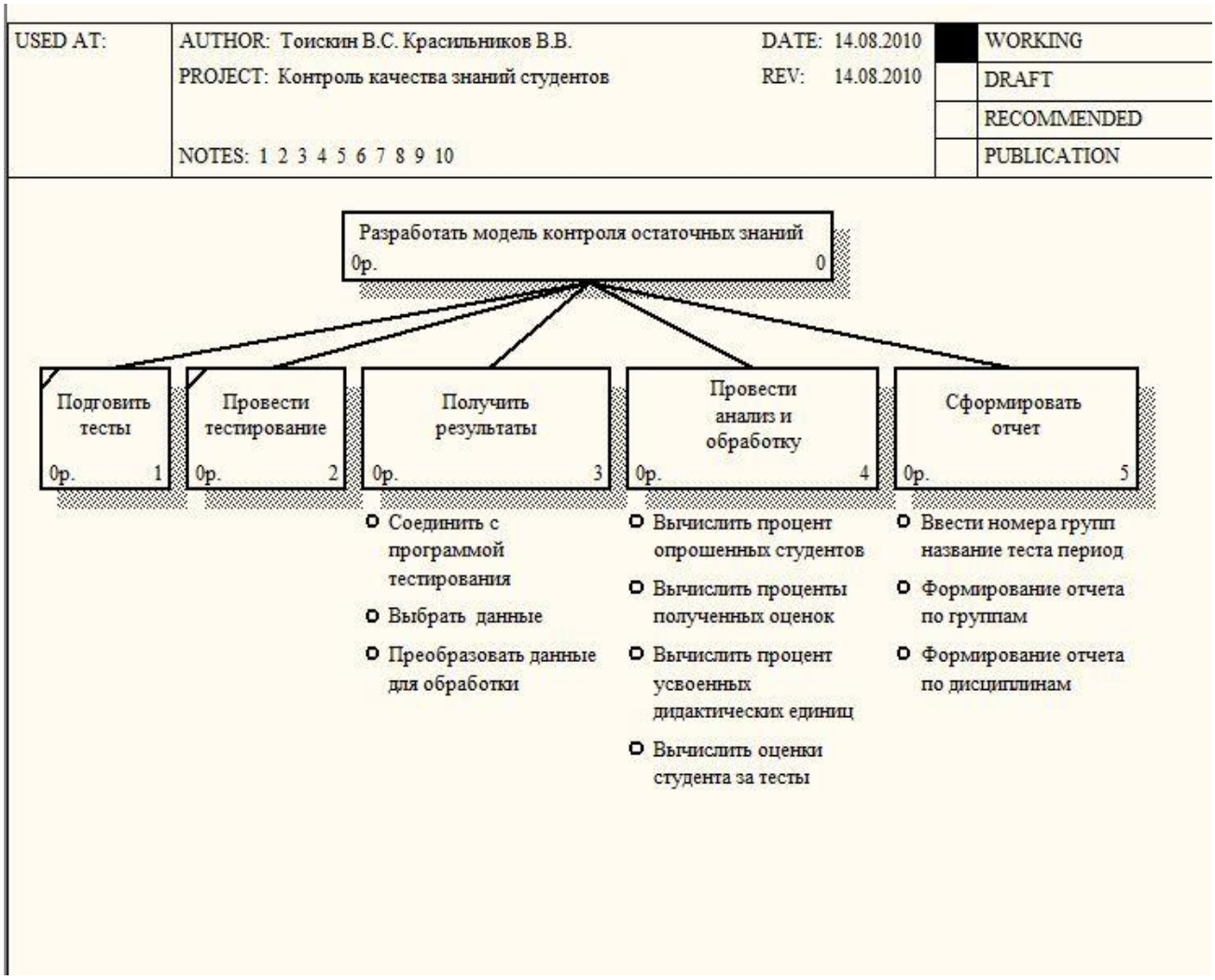


Рисунок 4.17. *– Диаграмма дерева узлов*

Диаграммы «только для экспозиции» (FEO) часто используются в модели для иллюстрации других точек зрения, для отображения отдельных деталей, которые не поддерживаются явно синтаксисом IDEF0. Диаграммы FEO позволяют нарушить любое синтаксическое правило, поскольку по сути являются копиями стандартных диаграмм и не включаются в анализ синтаксиса. Для создания диаграммы FEO следует выбрать пункт меню Diagram/Add FEO Diagram. В возникающем диалоге Add New FEO Diagram следует указать имя диаграммы FEO и тип родительской диаграммы (рисунок 4.18).

Новая диаграмма получает номер, который генерируется автоматически (номер родительской диаграммы по узлу + постфикс F, например A0F).

Отчеты на основе встроенных шаблонов можно создать, выбрав из меню Tools/Reports необходимый тип шаблона. Всего имеется семь типов шаблонов отчетов:

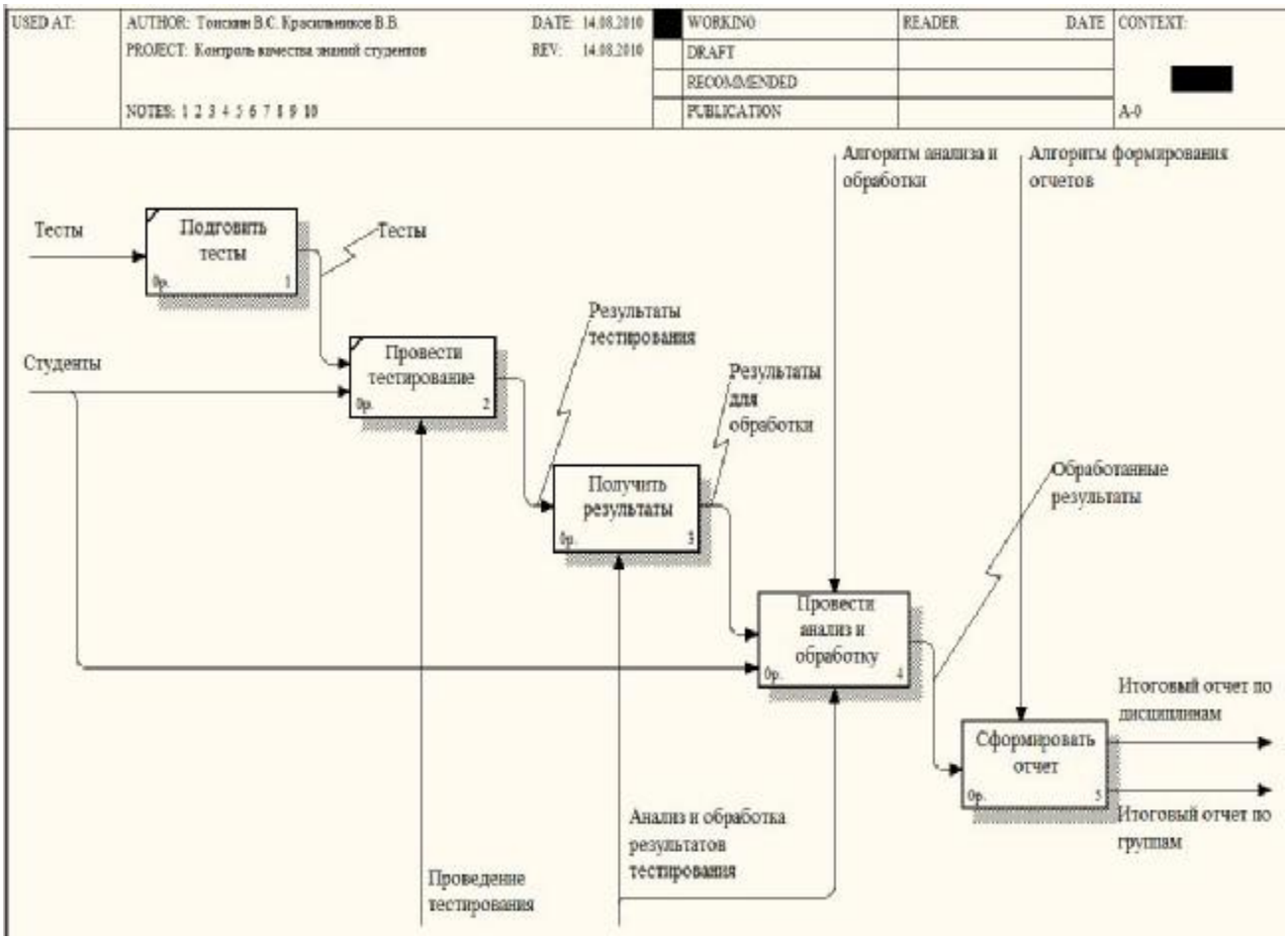


Рисунок 4.18. *– Диаграмма FEO*

1. Model Report. Он включает информацию о контексте модели – имя модели, точку зрения, область, цель, имя автора, дату создания и т.д.

2. Diagram Report. Отчет по конкретной диаграмме. Включает список объектов: работ, стрелок, хранилищ данных, внешних ссылок и т. д.

3. Diagram Object Report. Наиболее полный отчет по модели. Может включать полный список объектов модели: работ, стрелок с указанием их типа и и свойства, определяемые пользователем.

4. Activity Cost Report. Отчет о результатах стоимостного анализа.

5. Arrow Report. Отчет по стрелкам. Содержит информацию из словаря стрелок, информацию о работе-источнике, работе-назначении стрелки и информацию о разветвлении и слиянии стрелок.

6. DataUsage Report. Отчет о результатах связывания модели процессов и модели данных.

7. Model Consistency Report. Отчет, содержащий список синтаксических ошибок модели.

4.4 Модель информационного пространства единого педагогического комплекса

Материал параграфа излагается в соответствии с [25].

Информационное пространство педагогического комплекса есть система, включающая активный объект управления, управляющую систему, имеющиеся ресурсы и окружающую среду. При этом система является активной и рефлексивной, так как имеет собственную адаптивную модель и систему целеполагания и принятия решений.

Под информационным пространством будем понимать систему с целеполаганием и активной свободной волей, поведение которой основано на накоплении информации о себе и окружающей среде, ее анализ, прогноз собственного состояния и состояния окружающей среды, на принятии и реализации решений.

Структура такой системы в общем виде может быть представлена в виде следующей схемы (рисунок 4.19).

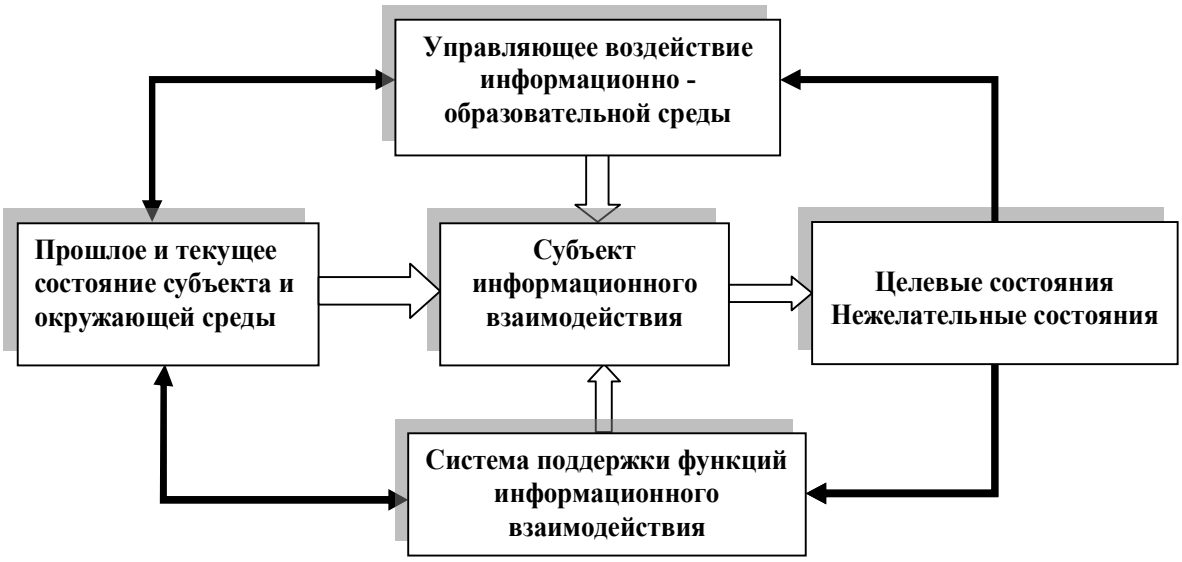


Рисунок 4.19. *– Структура информационно-образовательной среды*

Субъект информационного взаимодействия описывается целевыми адаптивными функциями, позволяющими определить степень достижения цели и возможные нежелательные состояния. Кроме управляющего воздействия информационно-образовательной среды состояние субъекта зависит от его прошлого состояния и воздействия окружающей среды. Цель создания информационно – образовательной среды состоит в разработке адаптивной системы поддержки функ-

ций информационного взаимодействия для достижения заданной цели при минимизации нежелательных последствий с учетом влияния внешних факторов.

Система поддержки функций информационного взаимодействия в общем случае должна включать две подсистемы: информационную и подсистему поддержки коммуникативного общения.

Структуру информационно-образовательной среды целесообразно рассматривать с системных позиций, учитывающих системно-элементный, системно-структурный, системно-функциональный, системно-коммуникационный, системно-интегративный и системно-исторический аспекты. Для облегчения формирования и анализа целей и функций информационно-образовательного пространства на первом этапе предпочтение следует отдать системно-функциональному аспекту исследования системы с учетом формальных методов и неформализованного знания.

При данном подходе появляется возможность оценки функциональности проектируемой среды, то есть оценки степени достижения поставленной цели. Тогда в качестве критерия качества целесообразно использовать функциональность системы при заданной логической сложности и возможности практической реализации.

Для оценки функциональности предлагается использовать экспериментальные методы.

Функции информационно-образовательного пространства могут быть определены исходя из цели создания пространства, точки зрения на это пространство (внешняя цель), имеющихся ресурсов для построения пространства и способов управления им.

В соответствии с [25] целью построения информационно-образовательного пространства является «обеспечение непрерывности и целостности профессионально-личностного образования будущего учителя как современного специалиста «интегрального профиля».

Точка зрения, основанная на исследованиях ведущих ученых [25] и концепциях в сфере образования [25] может быть сформулирована следующим образом: обеспечить профессиональные умения и готовность будущего учителя учить и воспитывать, добиваясь формирования соответствующих качеств в рамках обучения в вузе. При этом будущий учитель должен обладать широкой учебно-воспитательной эрудицией, профессиональной рефлексией, умением принимать оптимальные решения в нестандартных ситуациях и при ограниченной информации, владеть современными знаниями и уметь их практически реализовывать.

Все ресурсы, необходимые для достижения цели можно разделить на внутренние и внешние.

К внутренним ресурсам относятся: учебно-методические, кадровые, финансовые, научно-исследовательские, материально-технические, психологические. К внешним ресурсам будем относить: законодательная база высшего профессионального образования, источники финансирования, сложившийся рынок труда в России и в регионе, профиль вуза, научные исследования в регионе.

Так как цель будет реализовываться на основе информационного взаимодействия определим основные виды педагогических взаимодействий, реализующих антропологический подход:

Субъект-объектные взаимодействия (учебно-дисциплинарная модель).

Субъект-субъектное взаимодействие (личностно-ориентированная модель)

Объект-субъектные взаимодействия (свободное, спонтанное взаимодействие).

Для построения функциональной модели воспользуемся методом синтеза альтернативных вариантов по В.С.Симанкову. Тогда модель А0 может быть представлена в виде (Рисунок 4.20)

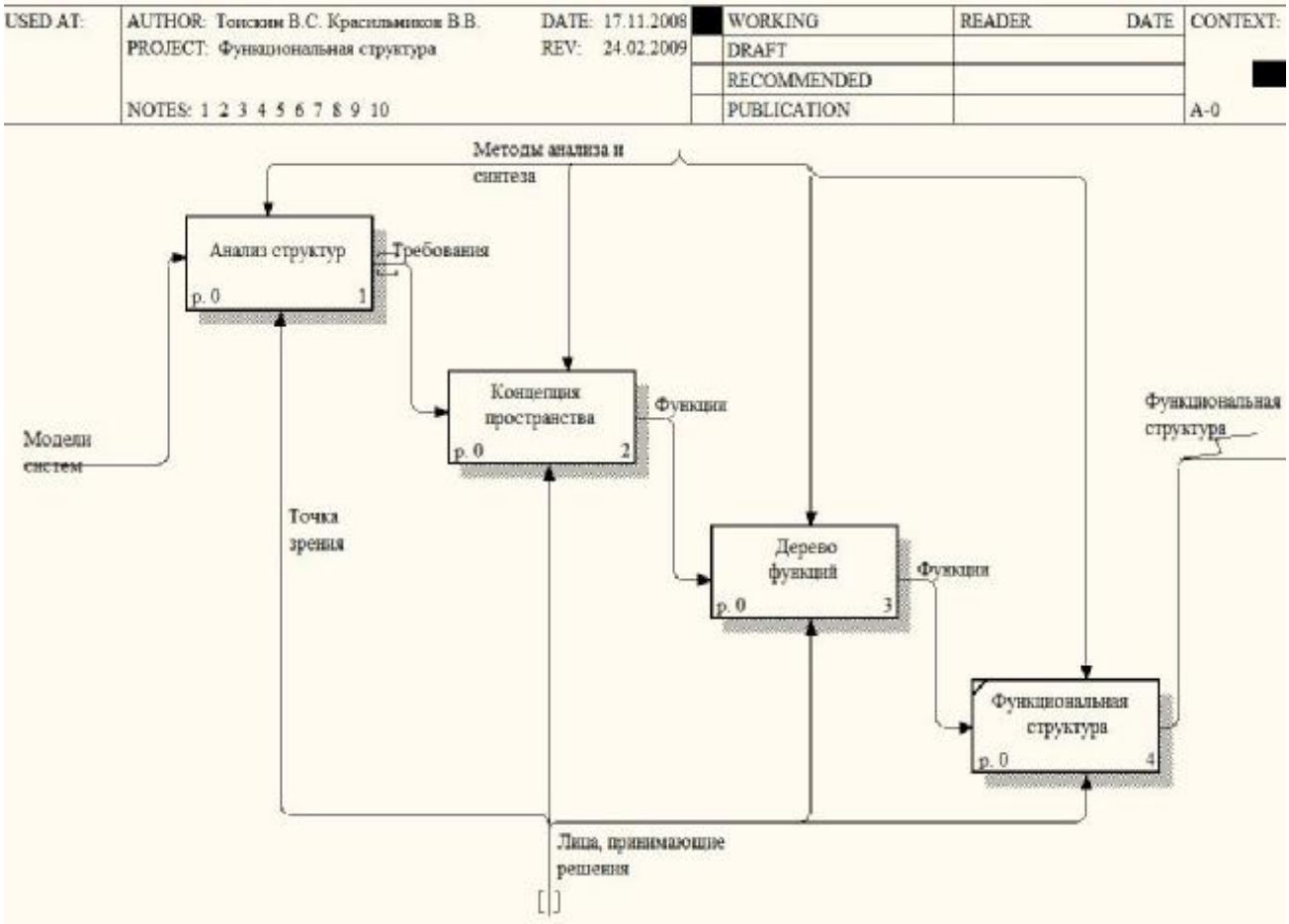


Рисунок 4.20. *– Модель А0.*

К компонентам модели относятся:

– информационные ресурсы педагогического комплекса;

– организационная структура;

– средства информационного взаимодействия.

При этом средства информационного воздействия целесообразно разделить на две группы:

– внутренние, в масштабах педагогического комплекса;

– внешние, в масштабах межвузовского и глобального масштаба.

При раскрытии функциональной структуры будем учитывать и следующие дидактические возможности информационного пространства:

– хранение, обработка передаваемой и получаемой информации;

– доступ к практически неограниченному количеству источников информации;

– доступ к в внешним методическим и дидактическим материалам, широко реализуемых в социальных сетях;

– коллективное, индивидуальное и конфиденциальное общение;

– диалоговое общение в реальном масштабе времени и с задержкой;

– интерактивное взаимодействие с информацией.

С учетом изложенного, можно провести декомпозицию функциональной структуры информационного пространства.

Декомпозиция процесса анализа структур информационных пространств приведена на рисунке 4.21.

Исходя из представленной декомпозиции, при анализе имеющихся подходов к построению информационных пространств необходимо проанализировать реализуемые подходы и функции.

Декомпозиция концепции построения информационного пространства приведена на рисунке 4.22.

Анализ приведенной декомпозиции показывает, что при определении концепции необходимо конкретизировать принципы, требования и этапы построения информационного пространства.

Декомпозиция дерева функций приведена на рисунке 4.23.

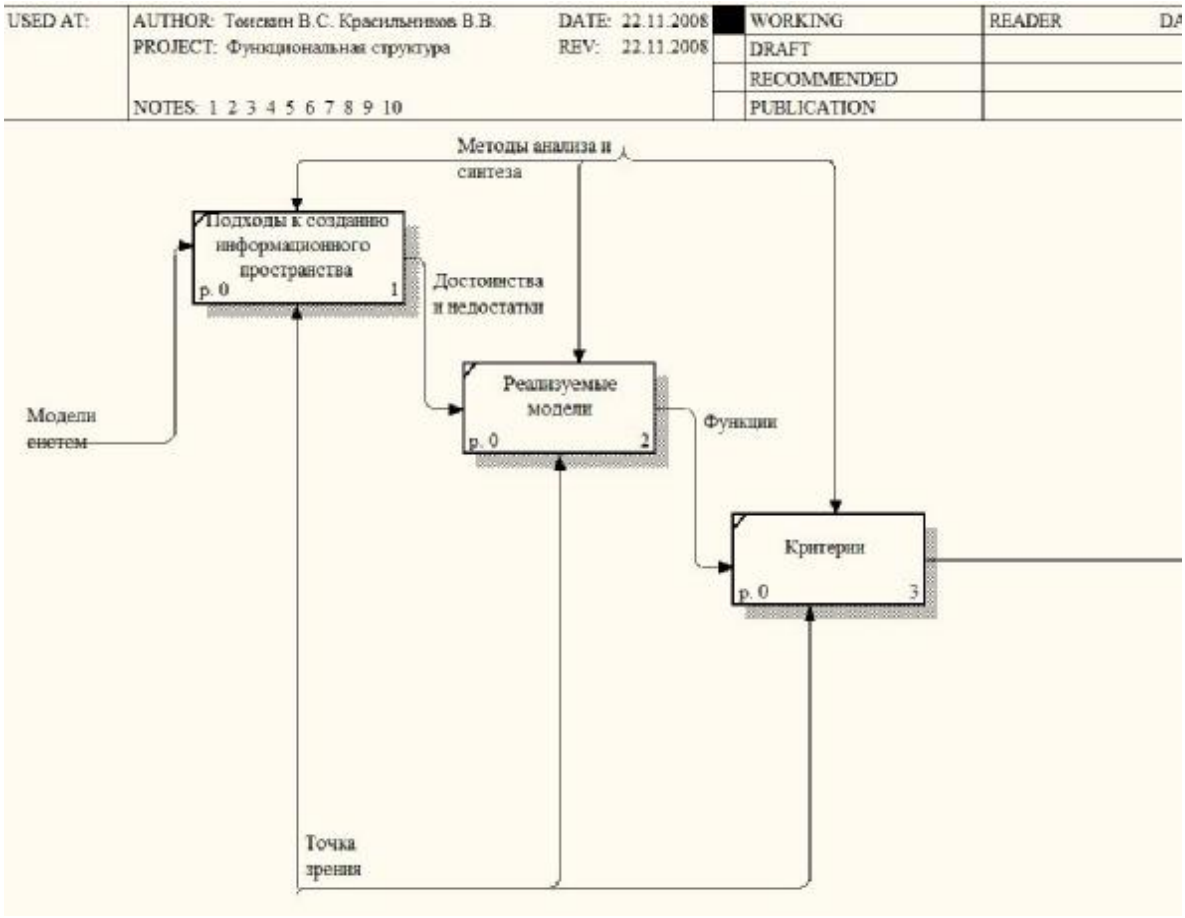


Рисунок 4.21. *– Анализ структур информационных пространств*

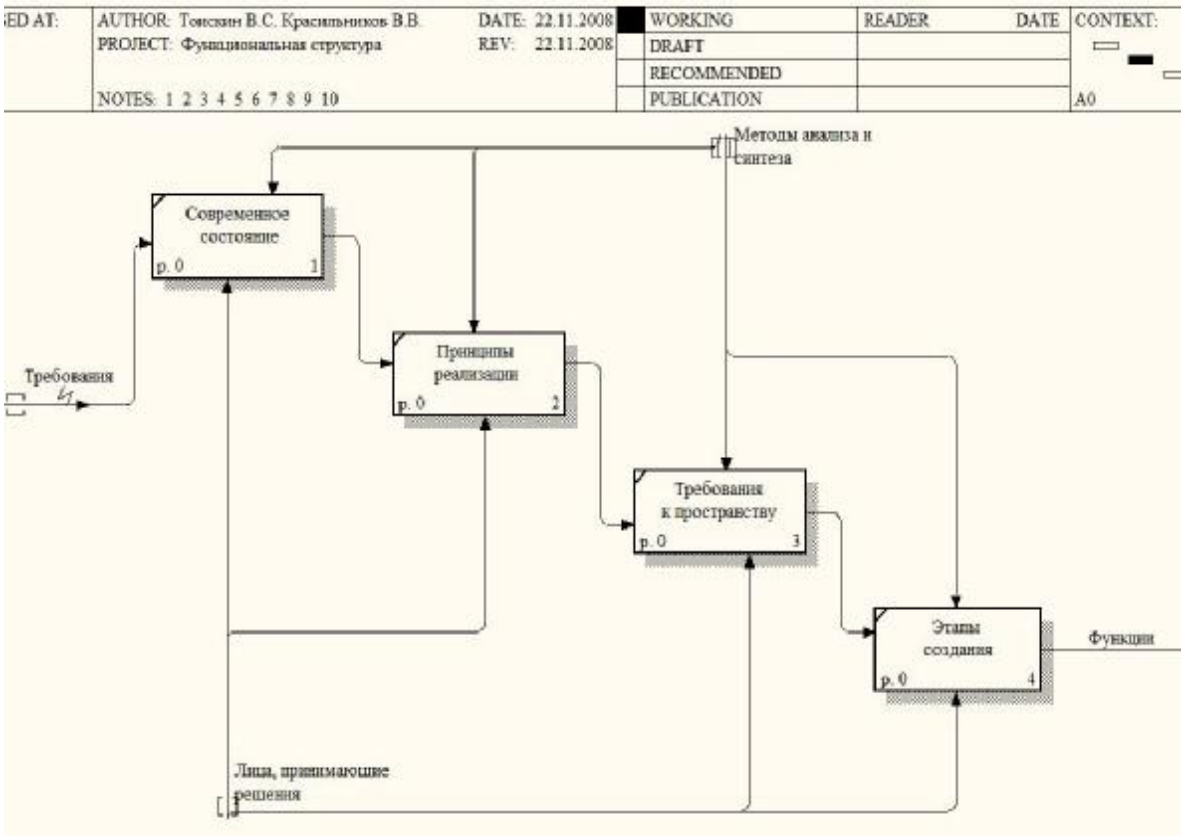


Рисунок 4.22. *– Концепция информационного пространства*

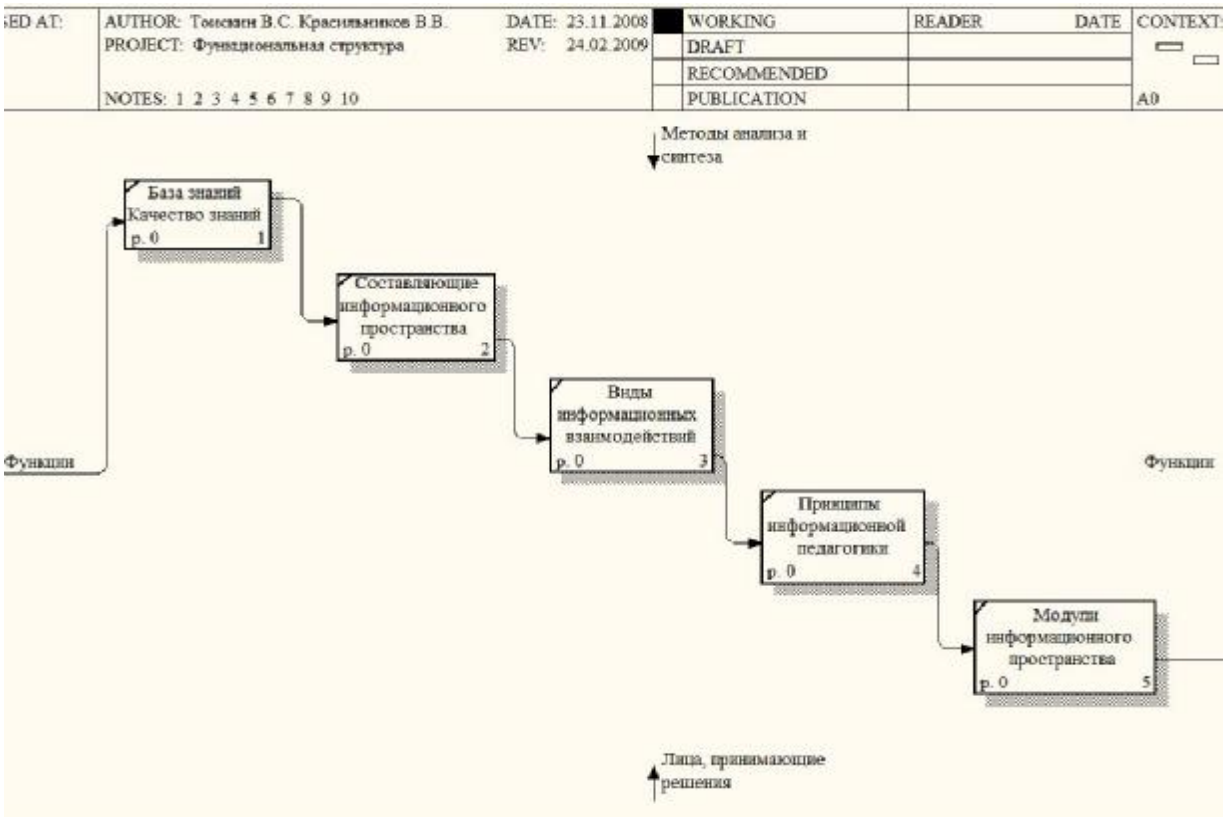


Рисунок 4.23. *– Дерево функций А3*

5. CASE – СРЕДСТВО ВЕРХНЕГО УРОВНЯ ERWIN

5.1. Общие сведения о программном средстве

Erwin обеспечивает выполнение следующих задач:

– поддерживается прямое создание БД на основе модели и генерацию модели по имеющейся базе данных для большинства СУБД;

– поддерживает методологию структурного моделирования SADT и нотации: IDEF1x, IE, Dimensional;

– позволяет повторно использовать компоненты созданных ранее моделей, а также использовать наработки других разработчиков;

– обеспечивает совместную работу над одной моделью;

– позволяет переносить структуру БД из СУБД одного типа СУБД в другой;

– позволяет документировать структуру БД

ERwin обеспечивает следующие возможности.

1. Обеспечивает стандартизацию процессов моделирования и проектирования.
2. Автоматическую генерирует БД с оптимизацией в соответствии физическими характеристиками целевой базы данных.
3. Обеспечивает согласованность логической и физической схем БД.
4. Поддерживает соответствие между серверной базой данных и формами в клиентской части.

ERwin имеет два уровня представления модели – логический и физический.

Логический уровень – это абстрактный взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире, например «Студент», «Кафедра» или «Фамилия студента». Объекты модели, представляемые на логическом уровне, в соответствии с методологией IDEF1X называются сущностями и атрибутами. Логическая модель может быть построена на основе модели процессов (BPwin). Существует возможность автоматического переноса модели из BPwin в Erwin. Логическая модель данных является универсальной и не связана с конкретной реализацией СУБД.

Физическая модель данных зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация обо всех объектах БД. Одной и той же логической модели могут соответствовать несколько разных физических моделей. Если в логической моде-

ли не описывается конкретно тип данных атрибута, то в физической модели важно описать всю информацию о конкретных физических объектах.

Диаграмма ERwin строится из трех основных блоков – сущностей, атрибутов и связей. Если рассматривать диаграмму как графическое представление правил предметной области, то сущности являются существительными, а связи – глаголами.

Выбор между логическим и физическим уровнем отображения осуществляется через линейку инструментов или меню. Внутри каждого из этих уровней есть следующие режимы отображения:

– режим «сущности»; внутри прямоугольников отображается имя сущности (для логической модели) или имя таблицы (для физического представления модели);

– режим «определение сущности» служит для презентации диаграммы;

– режим «атрибуты»; режим является основным при проектировании на логическом и физическом уровнях;

– режим «первичные ключи»; внутри сущностей показываются только атрибуты, составляющие первичный ключ.

– режим «пиктограммы»; каждой таблице может быть поставлена в соответствие пиктограмма;

– режим «показ глагольной фразы»; на дугах связей показываются глагольные фразы, связывающие сущности (для логического уровня) или имена внешних ключей (для физического уровня).

Разделение модели данных на логические и физические позволяет решить несколько важных задач.

1. Документирование модели. Многие СУБД имеют ограничение на именование объектов, что затрудняет понимание используемых обозначений для пользователя. На логическом уровне можно этим объектам дать синонимы – имена, несущие смысловую нагрузку, в том числе с использоваием кириллицы.

2. Масштабирование. На основе **р**азработанной логической модели данных можно сгенерировать физические модели под любую поддерживаемую ERwin СУБД (прямое проектирование). В то же время по содержимому системного каталога или SQL-скрипту имеющейся базы данных сформировать физическую и логическую модель данных (обратное проектирование). На основе полученной логической модели данных можно сгенерировать физическую модель для другой СУБД.

Процесс построения информационной модели в ERwin состоит из следующих шагов:

- выявление сущностей;
- определение связей между сущностями;
- определение атрибутов сущностей;
- определения доменов атрибутов;
- задание первичных и альтернативных ключей;
- приведение модели к требуемому уровню нормальной формы;
- переход к физическому описанию модели.

Целью выявления сущностей является определение основных сущностей, присутствующих в представлении данного пользователя о предметной области приложения (например: Группа, Студент, Успеваемость, Дисциплина).

Целью определения связей является выявление взаимодействий, возникающих между сущностями, выделенными в предыдущем этапе (например: студент учится в группе).

Целью определения атрибутов сущностей является выявление свойств, и дальнейшее их связывание с соответствующими сущностями или связями (например: сущность Студент имеет атрибут Имя).

Целью определения доменов атрибутов является определение диапазона всех возможных значений, которые может принять атрибут.

Целью задания первичных и альтернативных ключей является определение всех потенциальных ключей для каждой сущности и выделение первичного ключа.

Целью нормализации является исключение избыточных атрибутов и связей. Модель, как минимум должна быть приведена к третьей нормальной форме.

Целью переход к физическому описанию модели является описание структуры БД для конкретной СУБД.

5.2. Создание логической модели данных

Различают три уровня логической модели, отличающихся по глубине представления информации о данных:

- диаграмма сущность-связь (Entity Relationship Diagram, ERD);
- модель данных, основанная на ключах (Key Based model, KB);
- полная атрибутивная модель (Fully Attributed model, FA).

Диаграмма сущность-связь представляет собой модель данных верхнего уровня. Она включает сущности и взаимосвязи, отражающие основные правила предметной области.

Диаграмма сущность-связь может включать связи многие-ко-многим и не включать описание ключей. Как правило, ERD используется для презентаций и обсуждения структуры данных с экспертами предметной области.

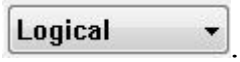
Модель данных, основанная на ключах, включает описание всех сущностей и первичных ключей и предназначена для представления структуры данных и ключей, которые соответствуют предметной области.

Полная атрибутивная модель – наиболее детальное представление структуры данных: представляет данные в третьей нормальной форме и включает все сущности, атрибуты и связи.

Модель состоит из трех основных блоков: сущностей, атрибутов и связей.

Построение модели данных предполагает определение сущностей и атрибутов, то есть необходимо определить, какая информация будет храниться в конкретной сущности или атрибуте. Сущности должны иметь наименование с четким смысловым значением, именоваться существительным в единственном числе, не носить технических наименований. Именование сущности в единственном числе облегчает в дальнейшем чтение модели. Фактически имя сущности дается по имени ее экземпляра.

Основная задача ERD – оценить, какие требования к информации будут достаточными для обеспечения планирования разработки базы данных.

Для внесения сущности в модель необходимо предварительно переключиться на логический уровень проектирования 

.

Щелкнуть на кнопке сущности (Entity) на панели инструментов и затем щелкнуть по тому месту на диаграмме, где необходимо расположить новую сущность (рисунок 5.1).

С использованием контекстного меню сущности (Entity Editor) определяются имя, описание и комментарии сущности (рис. 5.2). При этом ERwin на этапе создания сущности “помечает» этот пункт меню флажком.

Вкладки Note, Note 2, Note 3 , UDP служат для внесения дополнительных комментариев и определений к сущности:

Definition – используется для ввода определения сущности.

Note – можно ввести полезное замечание, описывающее какое-либо бизнес-правило или соглашение по организации диаграммы.

Note 2 – можно задокументировать некоторые возможные запросы, которые, как ожидается, будут использоваться по отношению к сущности в БД.

Note 3 – позволяет вводить примеры данных для сущности (в произвольной форме).

Icon – каждой сущности можно поставить в соответствие изображение, которое будет отображаться в режиме просмотра модели на уровне иконок.

Вкладка Volumetric используется для численного описания сущности с целью генерации сущности в физической модели.

После задания сущностей определяются атрибуты каждой из них, используя контекстное меню сущности (пункт меню будет помечен флажком) (рисунок 5.3).

Для ввода нового атрибута используется кнопка New…(рисунок 5.4)

Следует учесть, что Attribute Name можно вводить с учетом требований к логической модели, а Column Name – с учетом требований к выбранной СУБД. Целесообразно указывать и тип переменной (домен).

Вкладка Datatype позволяет выбрать тип атрибута.

Вкладка Constraint позволяет ввести условия проверки правильности ввода значения атрибута и значение, выводимое по умолчанию.

Вкладка Definition позволяет записывать определения отдельных атрибутов.

Вкладка Note позволяет добавлять замечания об одном или нескольких атрибутах сущности, которые не вошли в определения.

Вкладка UDP служит для задания значений свойств, определяемых пользователем.

Каждый атрибут можно связать с иконкой. При помощи поля со списком Icon в закладке General можно связать иконку с атрибутом.

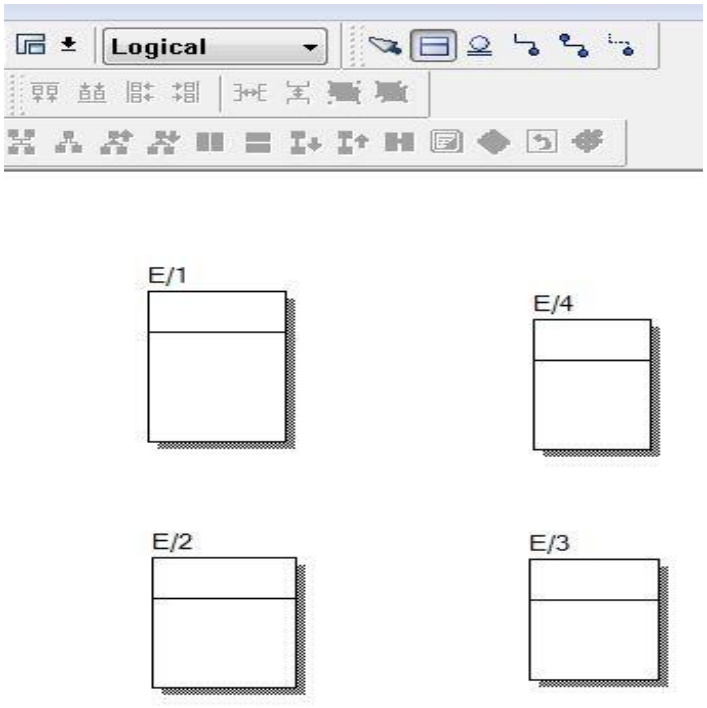


Рисунок 5.1.- *Логическая модель с сущностями E/1 – E/4.*

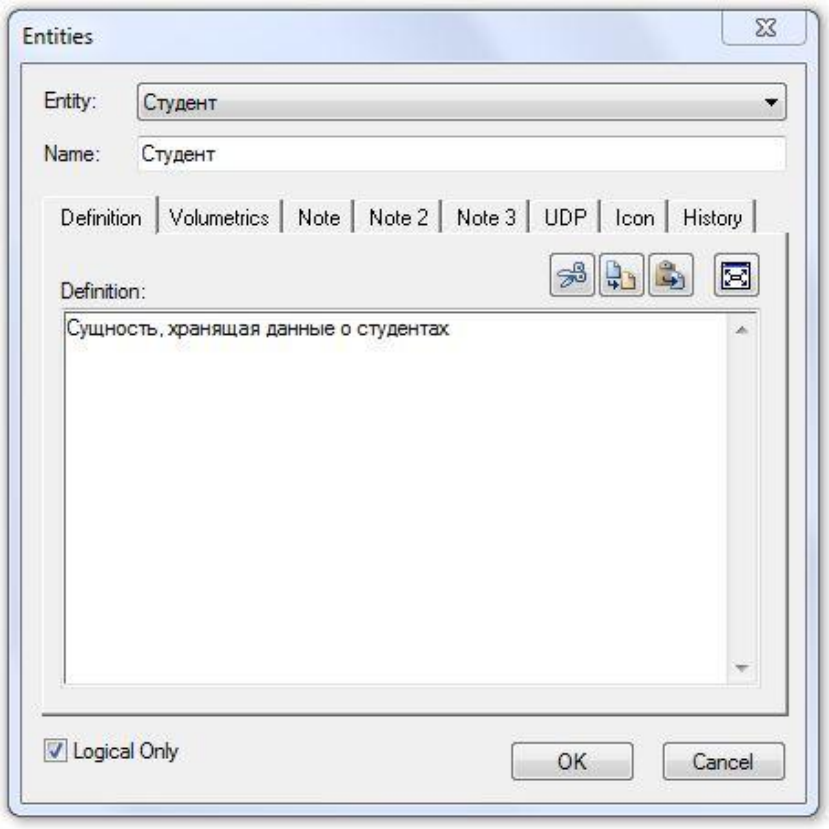


Рисунок 5.2. *– Диалоговое окно Entities*

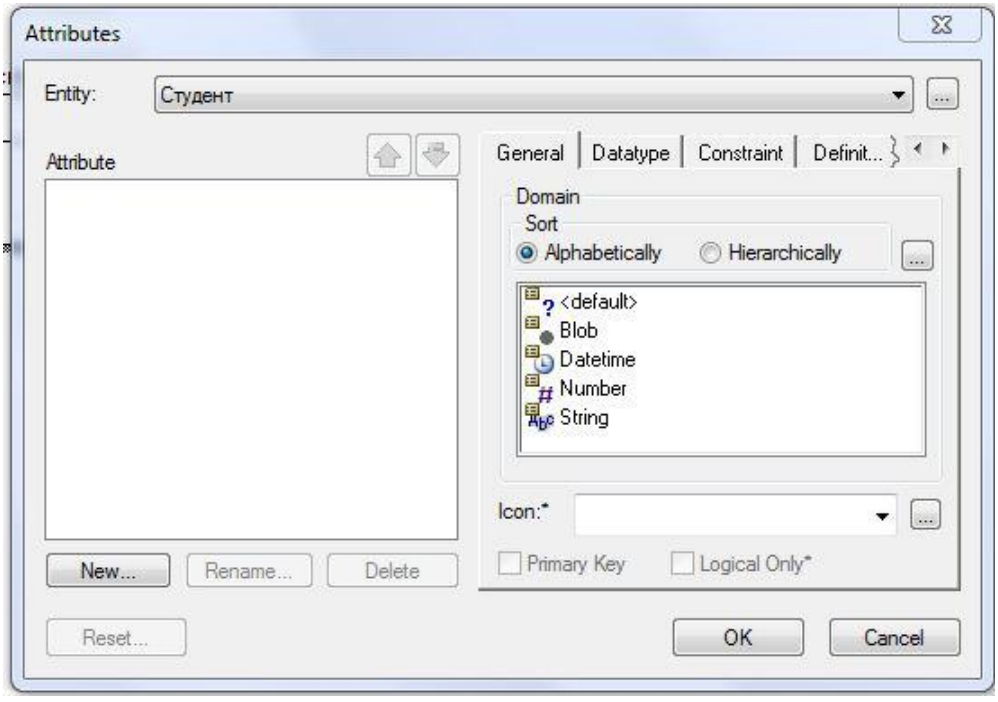


Рисунок 5.3. *– Диалоговое окно ввода атрибутов*

Атрибуты должны именоваться в единственном числе и иметь четкое смысловое значение. Согласно синтаксису IDEF1X имя атрибута должно быть уникально в рамках модели.

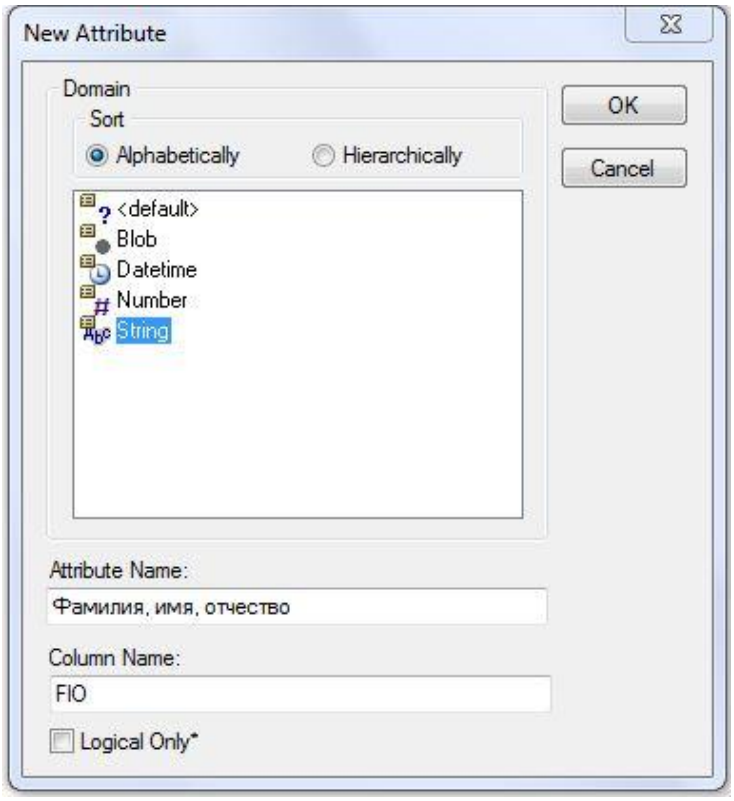


Рисунок 5.4. *– Диалоговое окно ввода нового атрибута*

При задании атрибутов необходимо выбрать первичный ключ.

При необходимости могут быть определены альтернативные и неуникальные ключи. Атрибуты, составляющие альтернативный ключ, однозначно (уникально) идентифицируют экземпляры сущности. В ERwin можно также составлять группы атрибутов, которые не идентифицируют уникально экземпляры сущности, но часто используются для доступа к данным. Для каждой такой группы атрибутов ERwin создает неуникальные индексы.

При выборе первичного ключа предпочтение должно отдаваться более простым ключам, то есть ключам, содержащим меньшее количество атрибутов. Значение атрибутов ключа не должно меняться в течение всего времени существования экземпляра сущности.

Для атрибутов первичного ключа во вкладке General диалога Attributes необходимо установить флажок Primary Key. Пример модели сущностей приведен на рисунке 5.5.

Вторым этапом построения логической модели является установление связей между сущностями. Связь является логическим соотношением между сущностями. Каждая связь должна именоваться глаголом или глагольной фразой.

Рисунок 5.5. *– Логическая модель сущностей*

На логическом уровне можно установить идентифицирующую связь один-ко-многим, связь многие-ко-многим и неидентифицирующую связь один-ко-многим (соответственно это кнопки слева направо в палитре инструментов).

Автоматически происходит отображение независимых и зависимых сущностей. Экземпляр зависимой сущности определяется только через отношение к родительской сущности.

При установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция дополнения атрибутов дочерней сущности при создании связи называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ – (FK) (Рисунок 5.6).

Внешние ключи (Foreign Key) создаются автоматически, когда связь соединяет сущности: связь образует ссылку на атрибуты первичного ключа в дочерней сущности, и эти атрибуты образуют внешний ключ в дочерней сущности (миграция ключа). Атрибуты внешнего ключа обозначаются символом (FK) после своего имени.

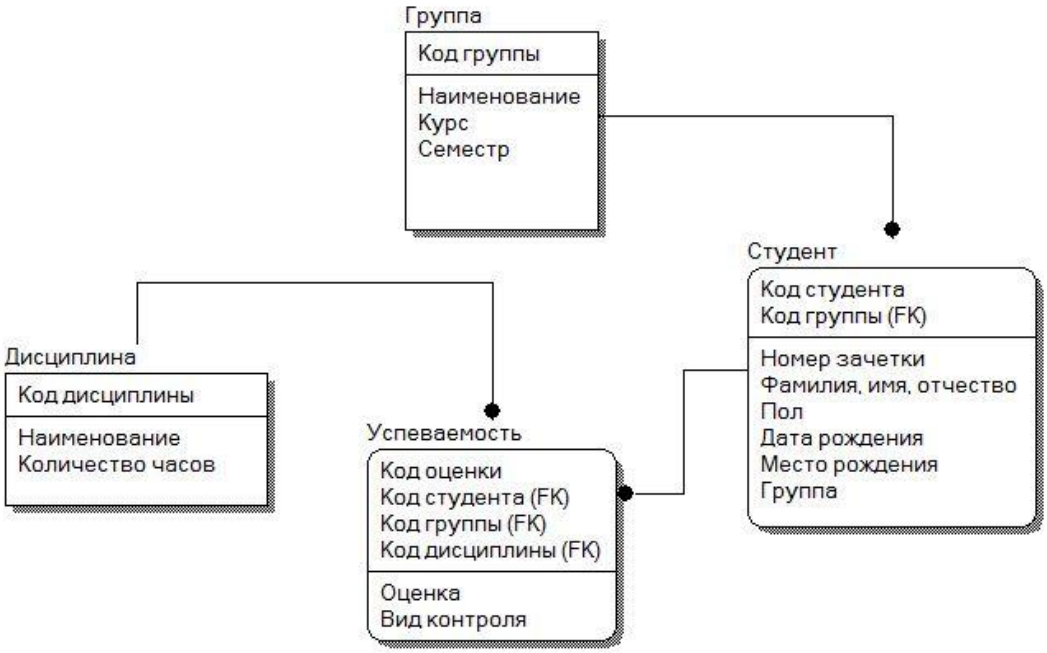


Рисунок 5.6. *– Логическая модель с установленными связями*

Зависимая сущность может иметь один и тот же внешний ключ из нескольких родительских сущностей. Сущность может также получить один и тот же внешний ключ несколько раз от одного и того же родителя через несколько разных связей. Комбинирование или объединение идентичных атрибутов называется унификацией. Унификация производится, поскольку правила нормализации запрещают существование в одной сущности двух атрибутов с одинаковыми именами.

Для установления связи необходимо:

– установить курсор на нужном типе связи в панели инструментов (идентифицирующая или неидентифицирующая) и нажать левую кнопку мыши;

– щелкнуть сначала по родительской, а затем по дочерней сущности.

Для редактирования свойств связи следует выбрать на контекстном меню связи пункт Relationship Properties. Во вкладке General появившегося диалога можно задать мощность, имя и тип связи (рисунок 5.7).

Имя связи (Verb Phrase) – фраза, характеризующая отношение между родительской и дочерней сущностями. Для связи один-ко-многим достаточно указать имя, характеризующее отношение от родительской к дочерней сущности (Parent-to-Child). Для связи многие-ко-многим следует указывать имена как Parent-to-Child так и Child-to-Parent.

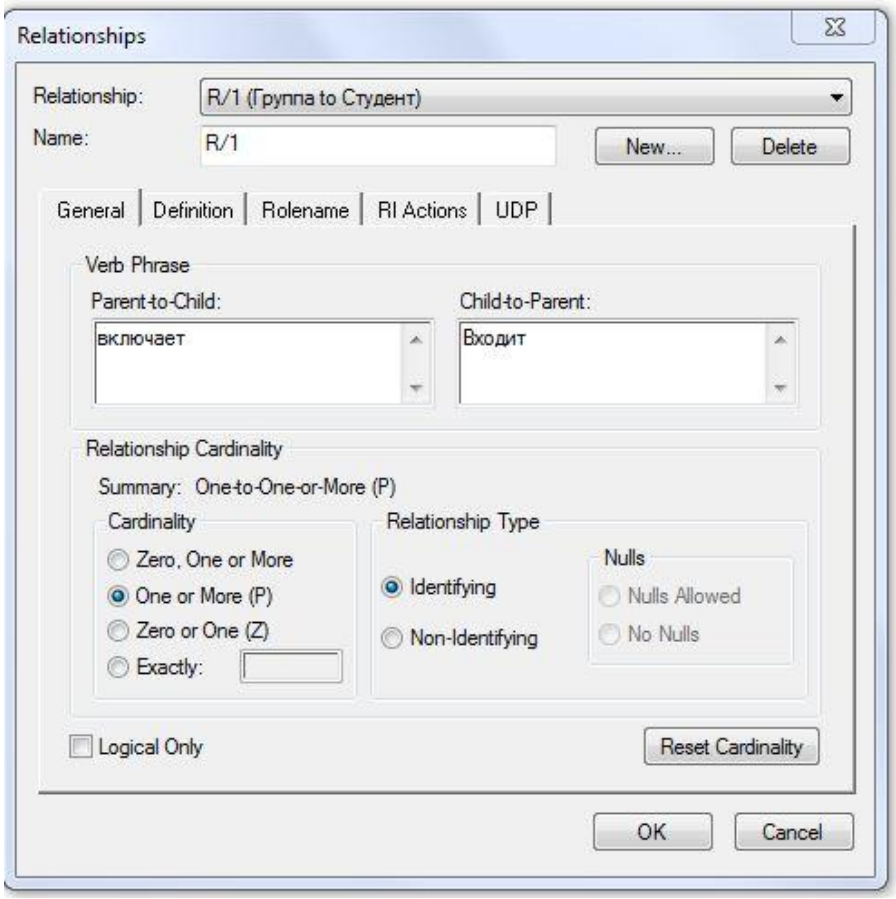


Рисунок 5.7. *– Определение мощности, типа и имени связи*

В закладке Definition можно дать более полное определение связи для того, чтобы в дальнейшем иметь возможность на него ссылаться.

В закладке Rolename можно задать имя роли.

В закладке RI Actions правила ссылочной целостности.

Правила ссылочной целостности (RI – referential integrity) – логические конструкции, которые выражают правила использования данных (правила вставки, замены и удаления). При генерации схемы БД на основе опций логической модели будут сгенерированы правила декларативной ссылочной целостности, которые должны быть предписаны для каждой связи, и триггеры, обеспечивающие ссылочную целостность. Триггеры представляют собой программы, выполняемые всякий раз при выполнении команд вставки, замены или удаления.

ERwin автоматически присваивает каждой связи значение ссылочной целостности, устанавливаемой по умолчанию, прежде чем добавить ее в диаграмму.

В логической модели целесообразно указывать иерархию категорий.

Обычно иерархию категорий создают, когда несколько сущностей имеют общие по смыслу атрибуты, либо когда сущности имеют общие по смыслу связи, либо когда это диктуется бизнес-правилами.

При определении иерархии следует различать следующие типы зависимых сущностей:

1. Характеристическая – зависимая дочерняя сущность, которая связана только с одной родительской и по смыслу хранит информацию о характеристиках родительской сущности.

2. Ассоциативная – сущность, связанная с несколькими родительскими сущностями. Такая сущность содержит информацию о связях сущностей.

3. Именующая – частный случай ассоциативной сущности, не имеющей собственных атрибутов (только атрибуты родительских сущностей, мигрировавших в качестве внешнего ключа).

4. Категориальная – дочерняя сущность в иерархии наследования.

Иерархия наследования (или иерархия категорий) представляет собой особый тип объединения сущностей, которые разделяют общие характеристики.

Для каждой категории можно указать дискриминатор – атрибут родового предка, который показывает, как отличить одну категориальную сущность от другой.

Иерархии категорий делятся на два типа – полные и неполные. В полной категории одному экземпляру родового предка обязательно соответствует экземпляр в каком-либо потомке.

Если категория еще не выстроена полностью и в родовом предке могут существовать экземпляры, которые не имеют соответствующих экземпляров в потомках, то такая категория будет неполной (рисунок 5.8).

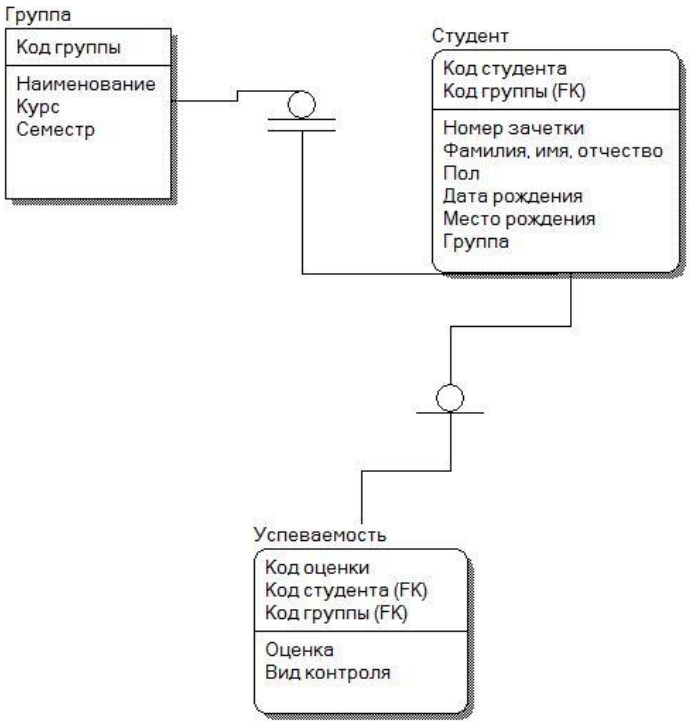


Рисунок 5.8. *– Иерархия категорий*

Важным средством ERwin как и BPwin является браузер, позволяющий осуществлять необходимые операции с элементами логических и физических моделей. Браузер имеет следующий вид (рисунок 5.9).

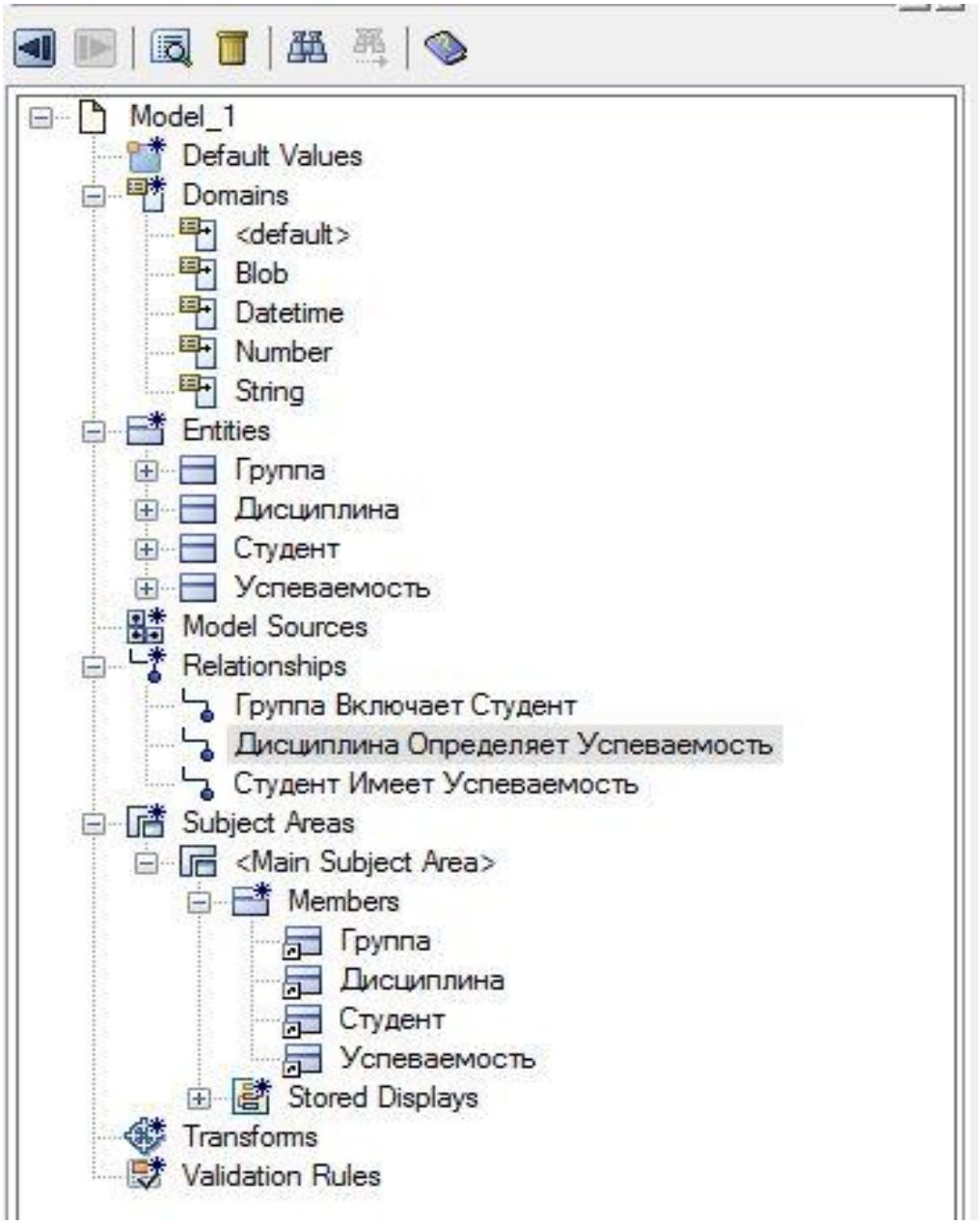


Рисунок 5.9. *– Браузер ERwin*

Использвание панели инструментов Standart  позволяет отобразить построенную модель в трех основных уровнях.

1. Модель Entity level (рисунок 5.10)

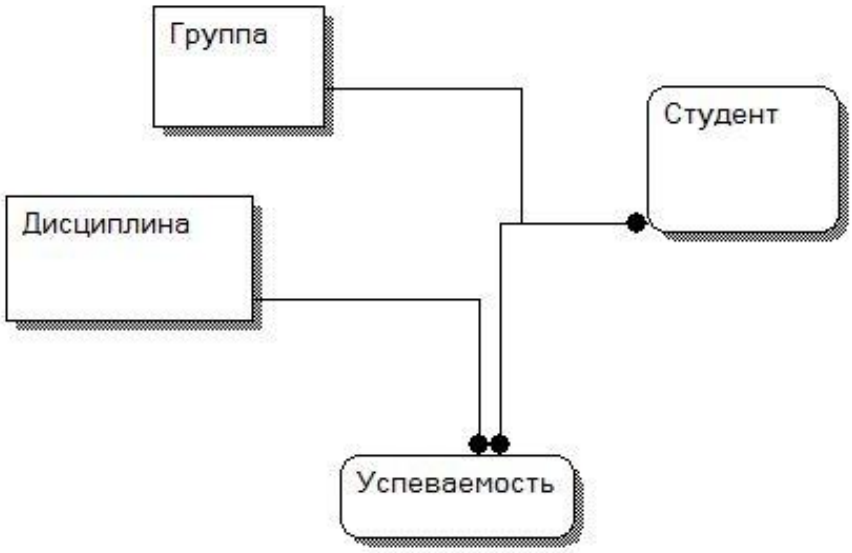


Рисунок 5.10. *– Модель сущности*

2. Модель Attribute level (рисунок 5.11)

Рисунок 5.11. *– Модель атрибутов*

3. Модель Definition level (рисунок 5.12)

Рисунок 5.12. *– Модель описаний*

5.3. Создание физической модели данных

Физические модели отображают всю информацию, необходимую для воплощения логической модели в систему БД.

Физическая модель фактически является отображением системного каталога БД. Создание модели данных, как правило, начинается с создания логической модели. После описания логической модели выбирается необходимую СУБД и ERwin автоматически создаст соответствующую физическую модель. Этот процесс называется прямым проектированием (Forward Engineering).

Обратное проектирование (Reverse Enineer), то есть восстановление информационной модели по существующей базе данных, используется при выборе оптимальной платформы для существующей настольной базы данных или базы данных на mainframe, а также при модификации существующей структуры. Построение модели может быть выполнено как на основании данных каталога базы данных, так и на основании пакета операторов SQL, с помощью которого была создана база данных.

Если в логической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, колонках, индексах, процедурах и т.д.

На основе ключей, описанных на уровне логической модели (поддерживаются первичные, внешние, альтернативные ключи) ERwin генерирует индексы.

Существует два уровня физических моделей:

– модель трансформации;
– DBMS модель.

Модель трансформации является «моделью данных проекта», описывающей отдельную часть всей структуры данных, предназначенную для обеспечения конкретного участка разработки. ERwin поддерживает разделение общей моделью на подобласти, называемые тематическими областями (subject areas).

Основными задачами модели трансформации являются: обеспечение администратора базы данных информацией, необходимой для создания физической базы данных, а также предоставление контекста для определений и записей в словаре данных и записей, образующих базу данных. Эта модель позволяет сравнить полученный проект физической базы данных с начальными требованиями к информации, а также для оценки и корректировки расширяемости и ограничений базы данных.

DBMS модель. Модель трансформации может быть переведена в DBMS модель, которая, в свою очередь, получает определения объектов физической базы данных в схеме RDBMS (или каталоге базы данных).

ERwin напрямую поддерживает эту модель с функцией генерации схемы. Первичные ключи становятся уникальными индексами. Альтернативные ключи и инверсные вхождения также могут стать индексами.

Для созданной логической модели формирование физической модели осуществляется автоматически при переключении на стандартной панели инструментов из логической модели в физическую 

. Физическая модель имеет вид (рисунок 5.13).

В отличие от логической модели наименование сущностей и атрибутов определяются в соответствии с требованиями конкретной базы данных.

Для проверки и корректировки свойств сущностей и атрибутов для конкретной базы данных необходимо выделить соответствующую сущность и в контекстном меню выбрать пункт Columns (рисунок 5.14).

Для внесения новой таблицы в модель на физическом уровне служит кнопка  на палитре инструментов. Связи между таблицами создаются так же, как на логическом уровне.

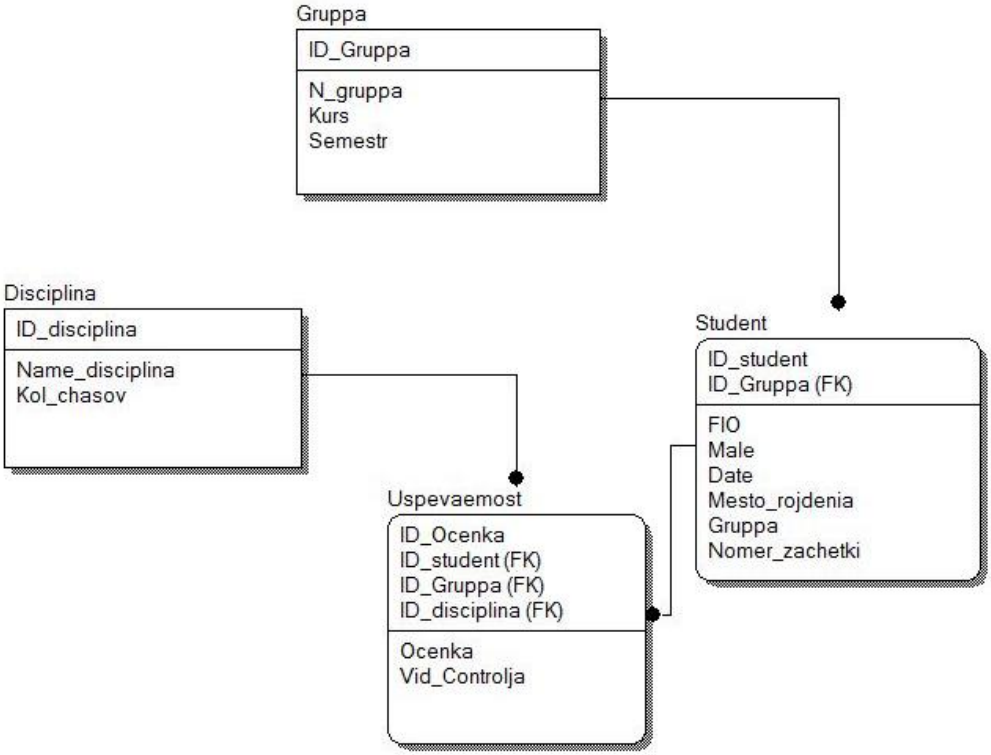


Рисунок 5.13. *– Физическая модель базы данных*

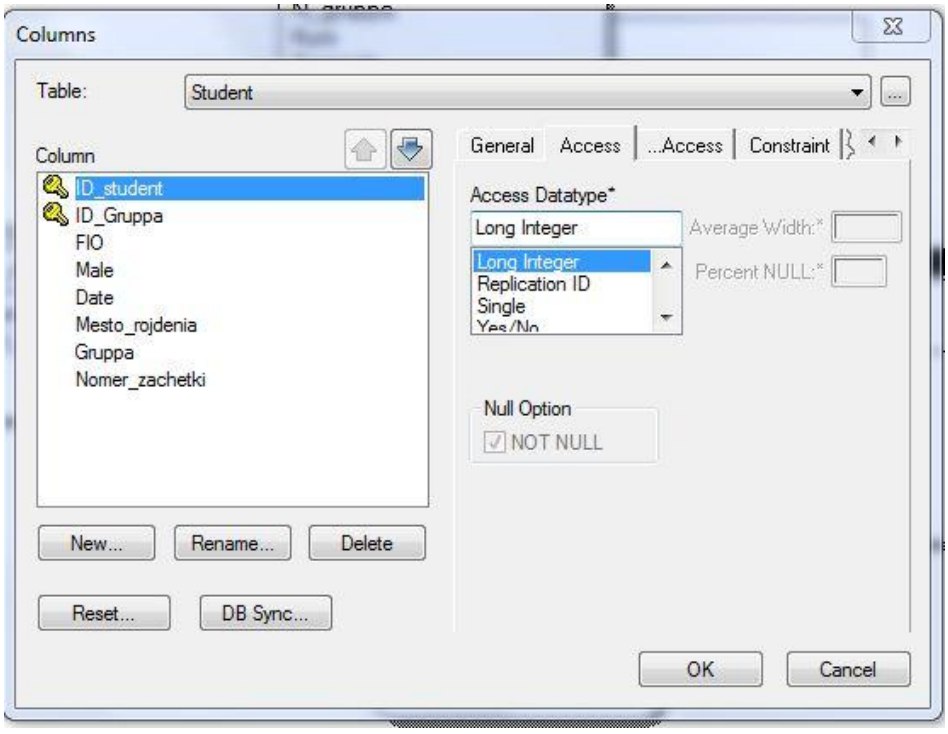


Рисунок 5.14*. – Диалоговое окно Columns*

При генерации имени таблицы или колонки по умолчанию все пробелы автоматически преобразуются в символы подчеркивания, а длина имени обрезается до максимальной длины, допустимой для выбранной СУБД.

ER win позволяет в соответствии с требованиями выбранной базы данных формировать запросы. С этой целью формируются представления. Представления (временные или производные таблицы), представляют собой объекты БД, данные в которых не хранятся постоянно, а формируются динамически при обращении к представлению. Представление не может существовать само по себе, а определяется только в терминах одной или нескольких таблиц.

ERwin имеет специальные инструменты для создания и редактирования независимых таблиц. Палитра инструментов на физическом уровне содер-жит кнопки внесения представлений и установления связей между табли-цами и представлениями. Для внесения представления нужно щелкнуть по кнопке  в палитре инструментов, затем по свободному месту диаграммы. По умолчанию представление получает номер V/n, где n – уникальный по-рядковый номер представления. Для установления связи нужно щелкнуть по кнопке  , затем по родительской таблице и, наконец, по представлению. Связи с представлениями и прямоугольники представлений показываются на диаграмме пунктирными линиями. Для редактирования представления служит контекстное меню Dtabase View представления. Можно сформировать запросы Select, From, Where, SQL запрос (рисунок 5.15).

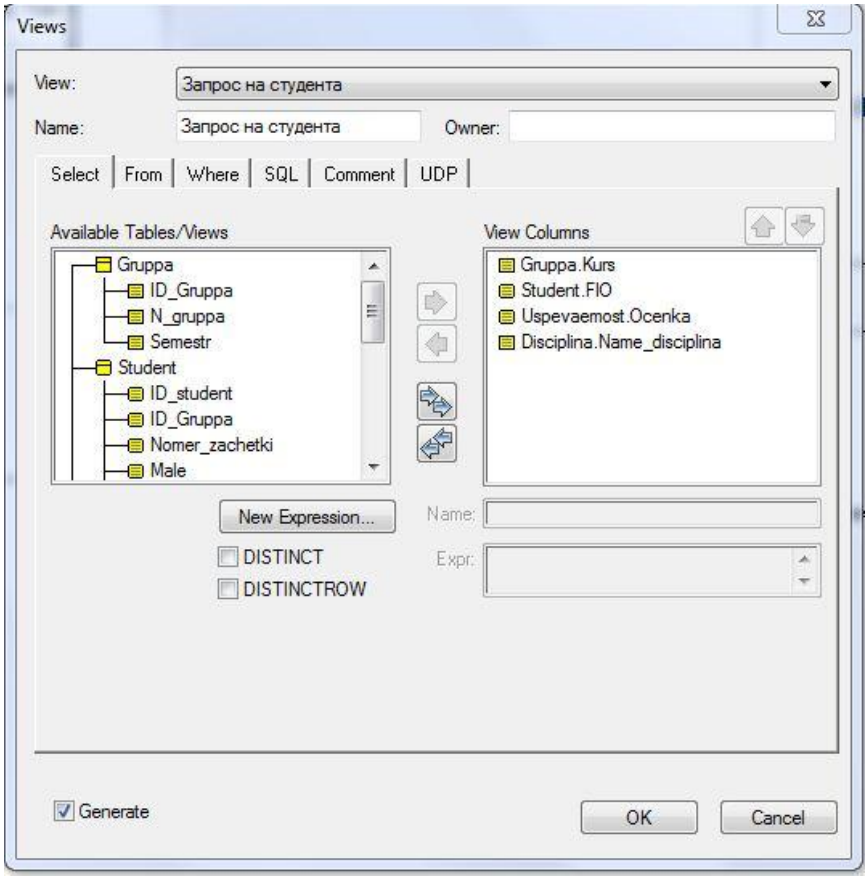


Рисунок 5.15. *– Диалоговое окно формирования запроса*

После сформированного запроса соответствующие изменения вносятся в физическую модель (рисунок 5.16)

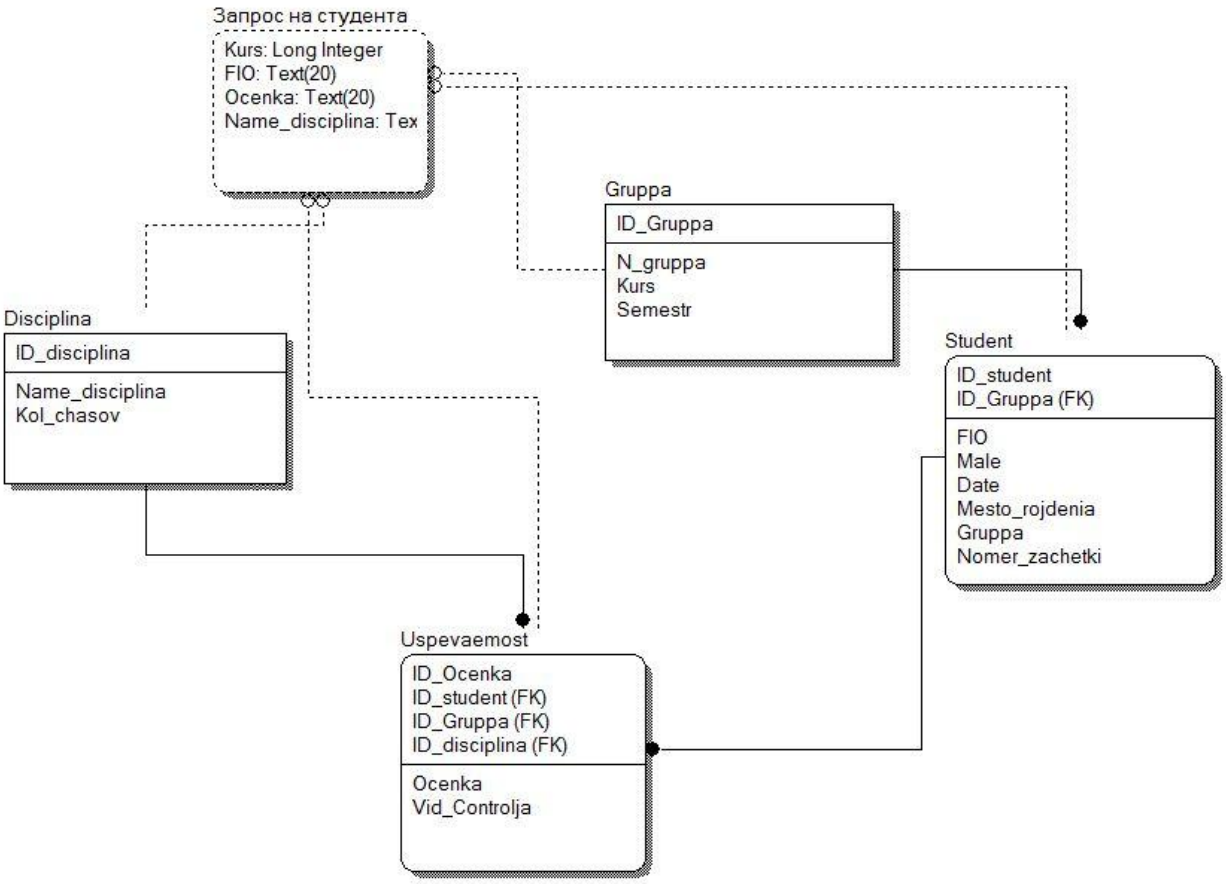


Рисунок 5.16. *– Физическая модель с представлением*

Последним шагом является генерация схемы БД. Для этого в необходимо выбрать пункт меню Tools-Forward Engineer- Schema Generation, откроется диалоговое окно Forward Engineer Schema Generation (рисунок 5.17).

Нажатие кнопки Preview позволяет посмотреть код, который будет создан автоматически ERwin (рисунок 5.16).

Кнопка Report… сохраняет SQL скрипт в ERS или SQL текстовом фай-ле.

Кнопка Generate… запускает процесс генерации схемы. Возникает диалог связи с БД, устанавливается сеанс связи с сервером и начинает выполняться SQL-скрипт. При этом возникает диалог Generate Database Schema.

По завершении работы над информационной моделью, как правило, распечатываются логический и физический уровни диаграммы, а также отчет по соответствиям сущность-таблица, атрибут-имя колонки, сущность-атрибуты.

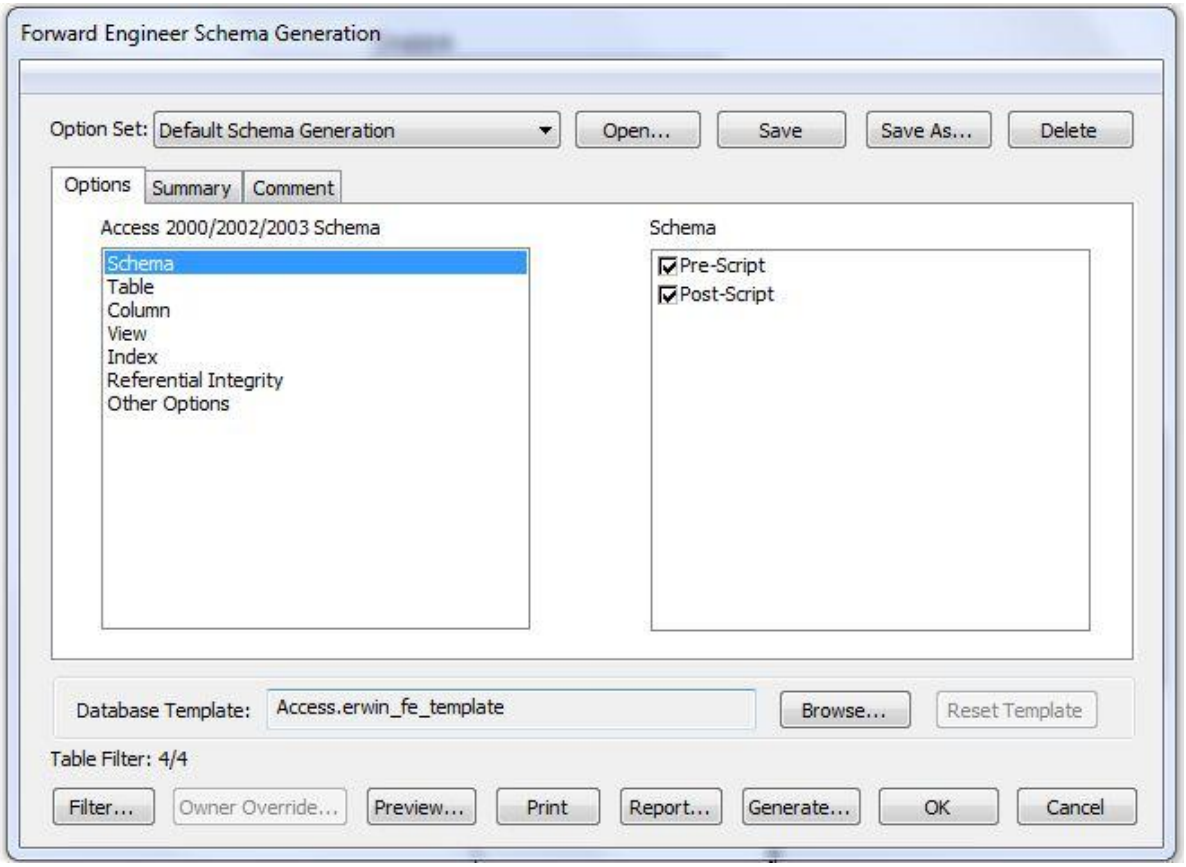


Рисунок 5.17. *– Диалоговое окно генерации схемы.*

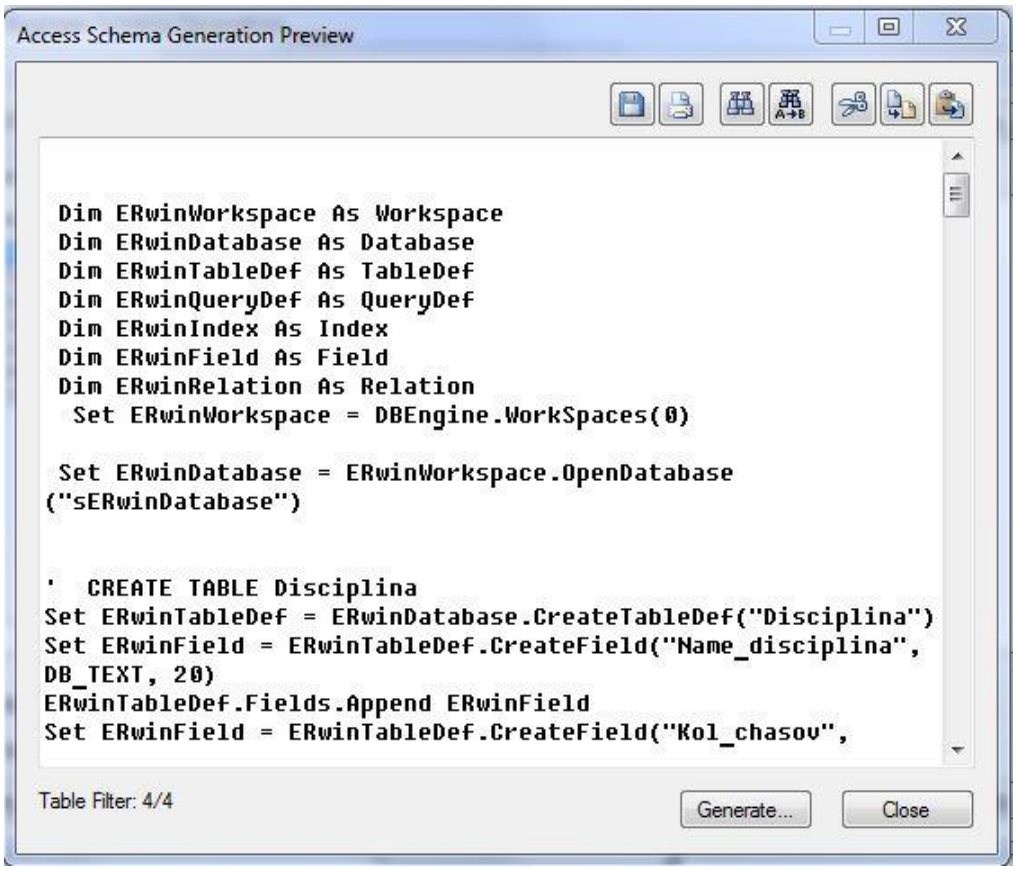


Рисунок 5.18. *– Пример генерации кода базы.*

5.4. Модель успеваемости студентов

Рассмотрим упрощенную модель успеваемости студента. Модель должна обеспечивать хранение сведений о группах и студентах, а также о результатах текущей сессии. Для каждого студента атрибутами являются такие данные, как фамилия, имя отчество студента, номер его зачетной книжки, оценки, полученные в текущей сессии, и отметки о сданных зачетах. Сведения о группе – это номер группы, факультет, кафедра, специальность, к которым она относится, год формирования группы.

В модель могут вноситься следующие изменения

- удаление или добавление в базу студента;
- изменение номера зачетки;
- перевод студента в другую группу;
- ввод оценок, полученных на экзаменах по каждому предмету.

Выходными данными модели являются:

- средний балл студентов каждой группы;
- средний балл по каждому предмету;
- предмет по которому имеется наибольшее количество неудовлетворительных оценок.

При анализе предметной области можно выделить следующие сущности: Студент, Группа, Специальность, Предмет.

Анализ задач, решаемых моделью, позволяет определить атрибуты сущностей.

Студент: фамилия, имя, отчество; номер зачетной книжки; пол; дата рождения.

Группа: номер группы; факультет; название кафедры; год формирования.

Специальность: код специальности, название специальности

Предмет: название предмета.

Между определенными сущностями можно выделить следующий набор связей:

Учится: связь между сущностями Студент и Группа, связь является обязательной для обеих сущностей. Мощность связи «один-ко многим», так как один студент не может обучаться в нескольких группах, в группе обучается несколько студентов, каждый студент должен учиться в какой-нибудь группе.

Специализируется: связь между сущностями Группа и Специальность, связь является обязательной для обеих сущностей. Мощность связи «один-ко-многим», так как группа относится к одной специальности, к одной специальности может относиться несколько групп, каждая группа относится к какой-то специальности.

Сдает экзамен: связь между сущностями Группа и Предмет, связь является обязательной для обеих сущностей. Мощность связи «многие-ко-многим», так как группа может сдавать несколько экзаменов, экзамен могут сдавать несколько групп, группа должна сдавать какие-либо экзамены, экзамены обязательно сдаются группами;

Получает оценки: связь между сущностями Студент и Предмет, связь является обязательной для обеих сущностей. Мощность связи «многие-ко-многим», так как студент может сдавать несколько экзаменов, экзамен могут сдавать несколько студентов.

Используя правила ER-диаграмм получаем следующие отношения.

Из связи *Учится* получаем отношения:

Студент (номер зачетной книжки, ФИО, номер группы, пол, дата рождения)

Группа (номер группы, факультет, название кафедры)

Из связи *Специализируется* получаем отношения.

Группа (номер группы, факультет, название кафедры, код специальности)

Специальность (код специальности, название специальности).

Из связи *Получает оценки* получаем отношения

Студент (номер зачетной книжки, ФИО, пол, дата рождения),

Предмет (Название предмета)

Результаты экзаменов (номер зачетной книжки, название предмета)

Из связи *Сдает экзамен* получаем отношения.

Группа (номер группы, факультет, кафедра)

Предмет (название предмета)

Экзамен (номер группы, название предмета)

Для выделения зачетов введем сущность Зачет и отношение *Сдает зачет.*

Группа (номер группы, факультет, кафедра)

Предмет (название предмета)

Зачет (номер группы, название предмета).

В результате получаем следующие предварительные отношения:

Студент (номер зачетной книжки, фамилия, имя, отчество, номер группы, пол, дата рождения)

Группа (номер группы, факультет, название кафедры, код специальности)

Специальность (код специальности, название)

Предмет (название предмета)

Результаты экзамена (номер зачетной книжки, название предмета, оценка)

Экзамен (номер группы, название предмета, дата экзамена)

Зачет (номер зачетной группы, название предмета, зачет)

Отношение Предмет является избыточным, так как его Название предмета, можно получить получить из отношений Результаты экзаменов, Экзамен и Зачет.

С использованием ERwin спроектируем логическую и физическую модель. После запуска программы определим необходимые параметры (рисунок 5.19).

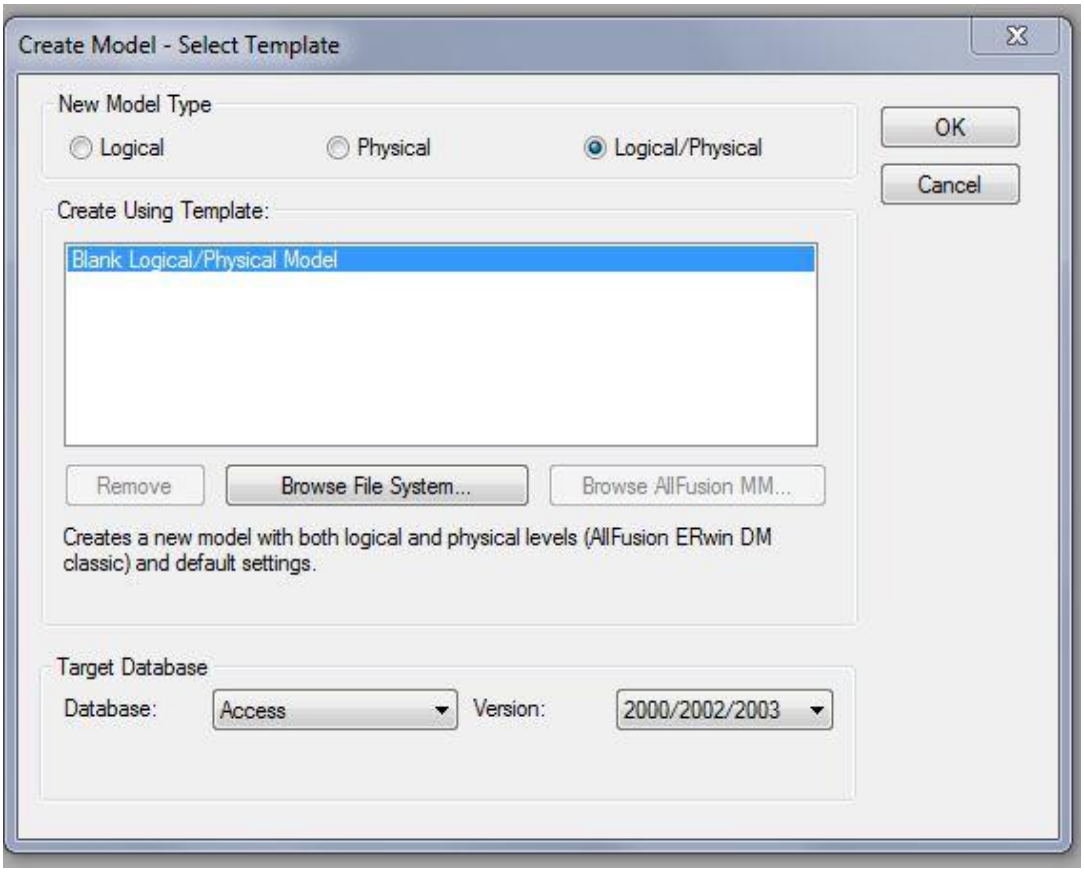


Рисунок 5.19. *– Окно создания новой модели*

В качестве типа модели будем использовать Logical/Physical, в качестве целевой СУБД – Access.

В стандартной панели выбираем логическую модель  . Для создания сущностей и связей используем панель Toolbox  .

Рассмотрим элементы этой панели (слева направо):

– Select – выбор объектов в модели;

– Entity – создание новой сущности;

– Complete sub-category –создание связки-разветвления;

– Identifying relationship – создание идентифицирующей связи «один-ко-многим»;

– Many-to-many relationship – создание связи «многие-ко-многим»;

– Non-identifying relationship – создание неидентифицирующей связи «один-ко-многим».

Для создания сущности надо выбрать соответствующую кнопку, после чего щелкнуть мышью на свободном месте Рабочей области. Создастся новая сущность, при этом активизируется ввод ее имени (имя по умолчанию E/N). После ввода имени и нажатия Enter выделение переходит в часть ввода ключевых атрибутов.

Имена атрибутов также набираются текстом. После ввода имени ключевого атрибута можно либо клавишей Enter добавить еще один ключевой атрибут, либо клавишей Tab перейти к вводу неключевых атрибутов.

Каждый новый неключевой атрибут вводится после нажатия клавиши Enter (рисунок 5.20).

Рисунок 5.20. – Сущность с введенными атрибутами

После ввода имени сущности и атрибутов дополнительно определяем их характеристики.

Для определения свойств сущности вызываем контектсное меню данной сущности и выбираем пункт Entity Properties (рисунок 5.21).

Для задания (корректировки) свойств атрибутов вызываем контектсное меню данной сущности и выбираем пункт Attributes (рисунок 5.22).

При определении атрибутов целесообразно указать описывающий его тип данных (числовой, дата, строковый и т.д)

После определения сущностей и атрибутов переходим к созданию связей между сущностями.

Для создания связи необходимо выбрать тип требуемой связи и последовательно щелкнуть на двух связываемых сущностях. При этом для связей типа «один-ко-многим» первая сущность будет родительской, а вторая – дочерней.

Для задания свойств связи необходимо открыть пункт контекстного меню данной связи Relationship Properties (рисунок 5.23).

Устанавливается вербальное определение связи (Verb Phrase), мощность связи (Relationship Cardinality), тип связи (Relationship Type) и т.д.

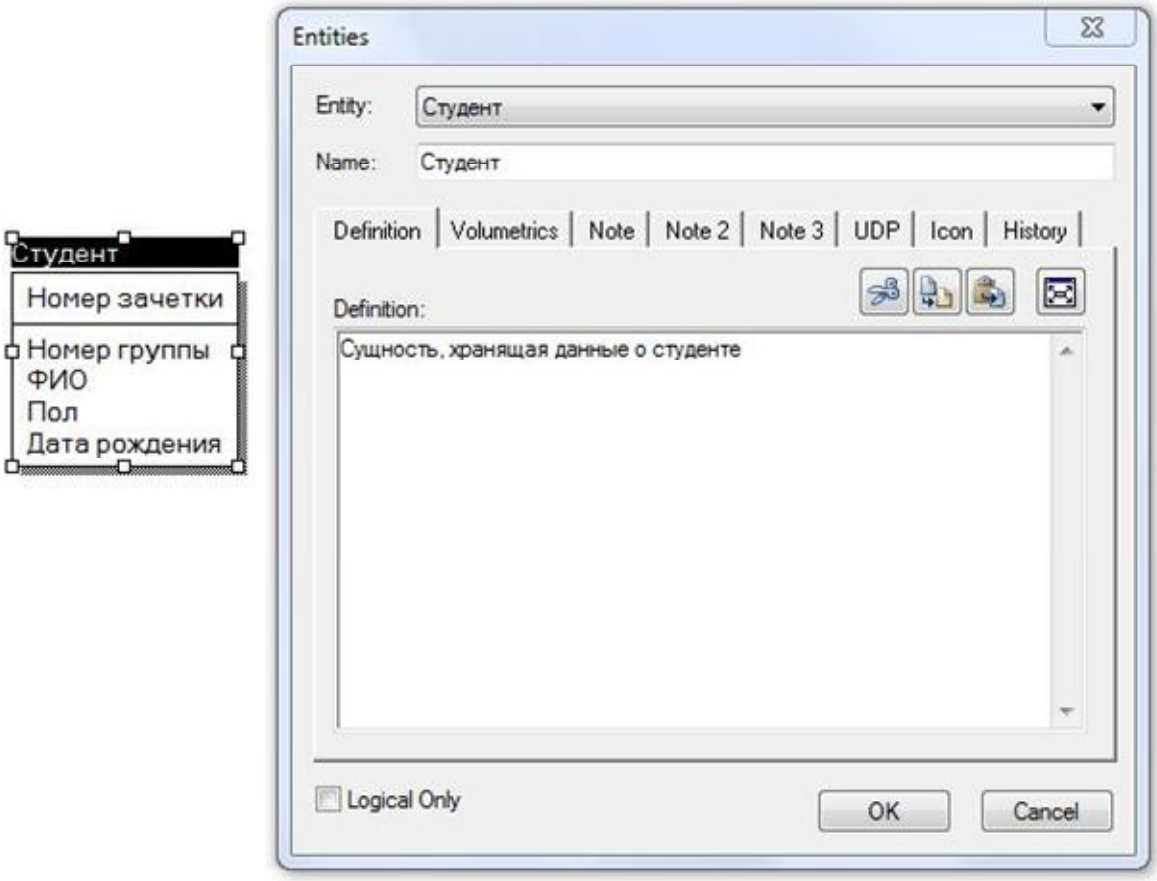


Рисунок 5.21. – *Описание свойств сущности*

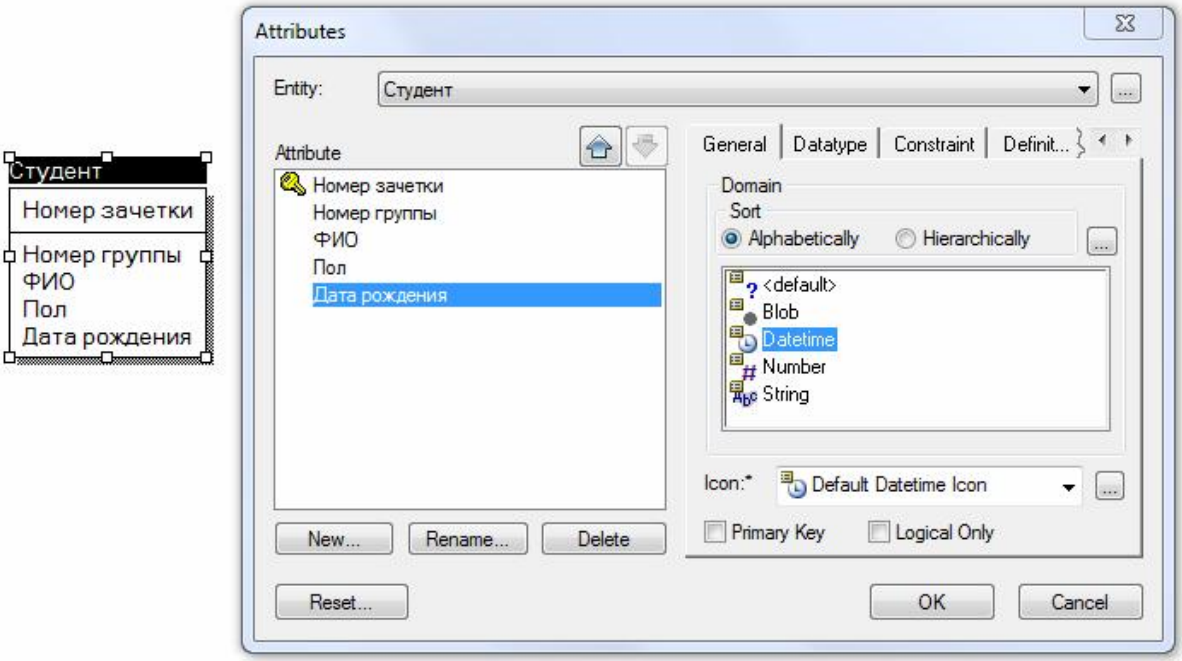


Рисунок 5.22. – *Свойства атрибутов*

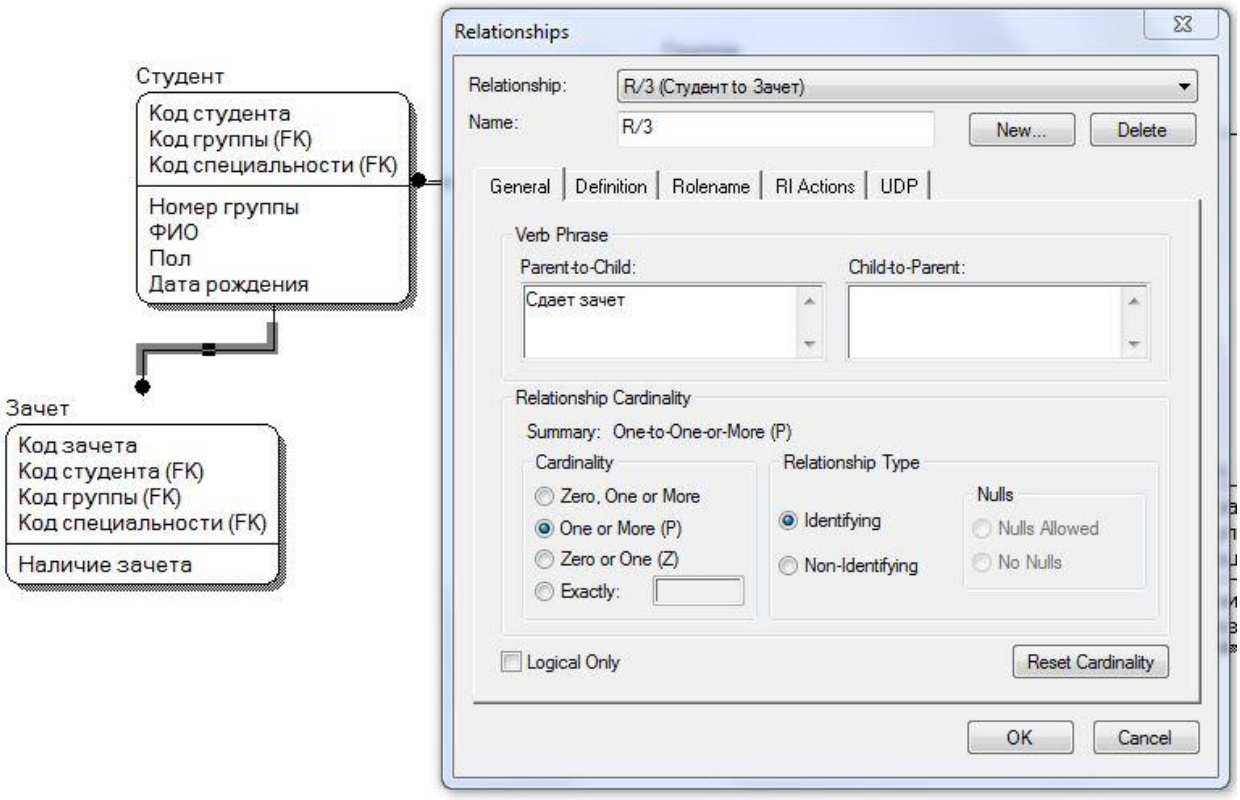


Рисунок 5.23. *– Определение свойств связи*

При создании физической модели имена атрибутов должны соответствовать требованиям соответствующей СУБД. Поэтому при определении свойств связи целесообразно на вкладке «Rolename» указать, имя атрибута, отвечающего за организацию связи в рамках СУБД (рисунок 5.24).

После того, как общий вид модели определен, переходим к редактированию физической модели. Для этого в панели инструментов выбирается из выпадающего списка вид модели выбрать 

.

В спроектированной физической модели имена атрибутов не соответствуют требования, предъявляемым СУБД. При этом физической модели понятию атрибут соответствует понятие колонка. При модификации физической модели рекомендуется придерживаться английских наименований атрибутов и сущностей.

Окончательный вариант логической схемы имеет вид (рисунок 5.25).

Для переименования атрибутов необходимо перейти к пункту Columns контекстного меню сущности, выбрать атрибут из списка, кнопкой «Rename» вызывать диалог переименования атрибута (рисунок 5.26).

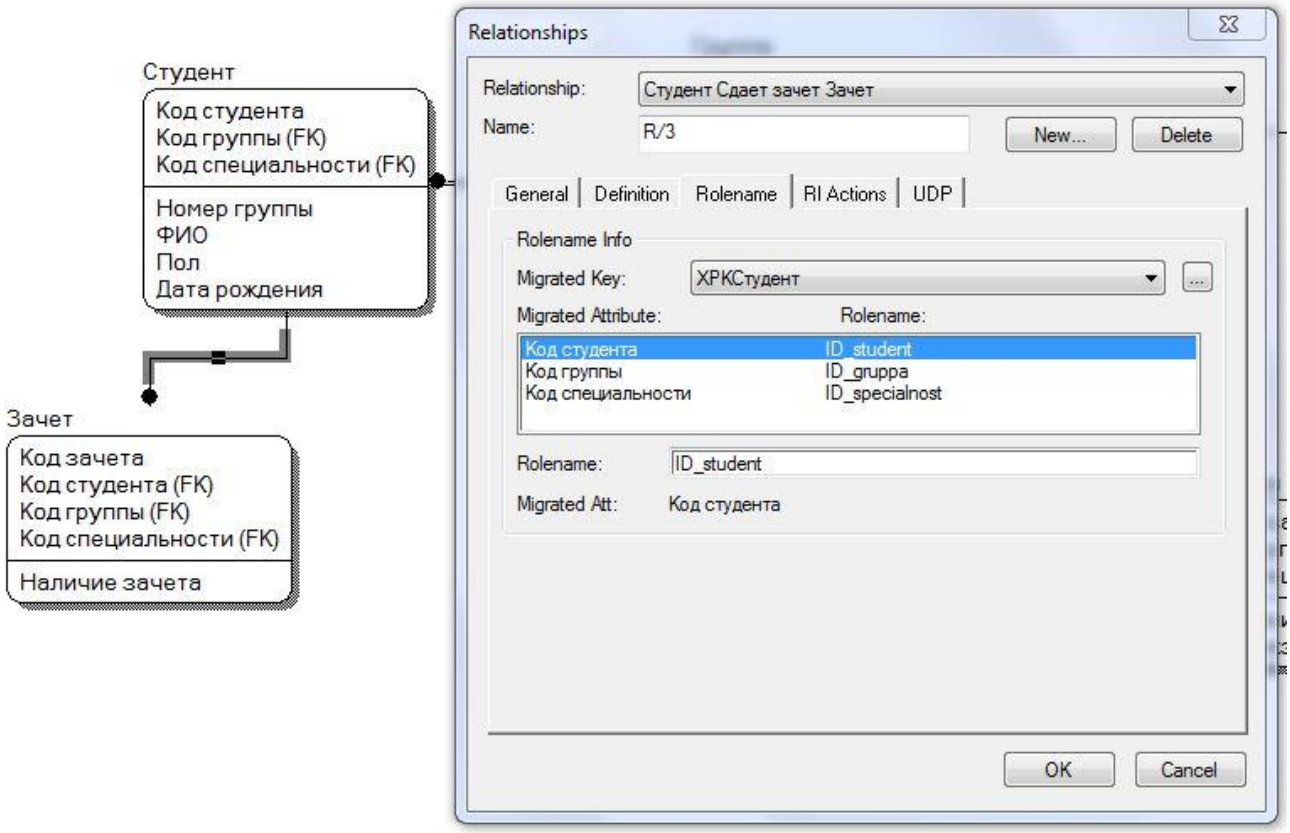


Рисунок 5.24. *– Указание роли внешних ключей*

Рисунок 5.25*. – Логическая модель предметной области*

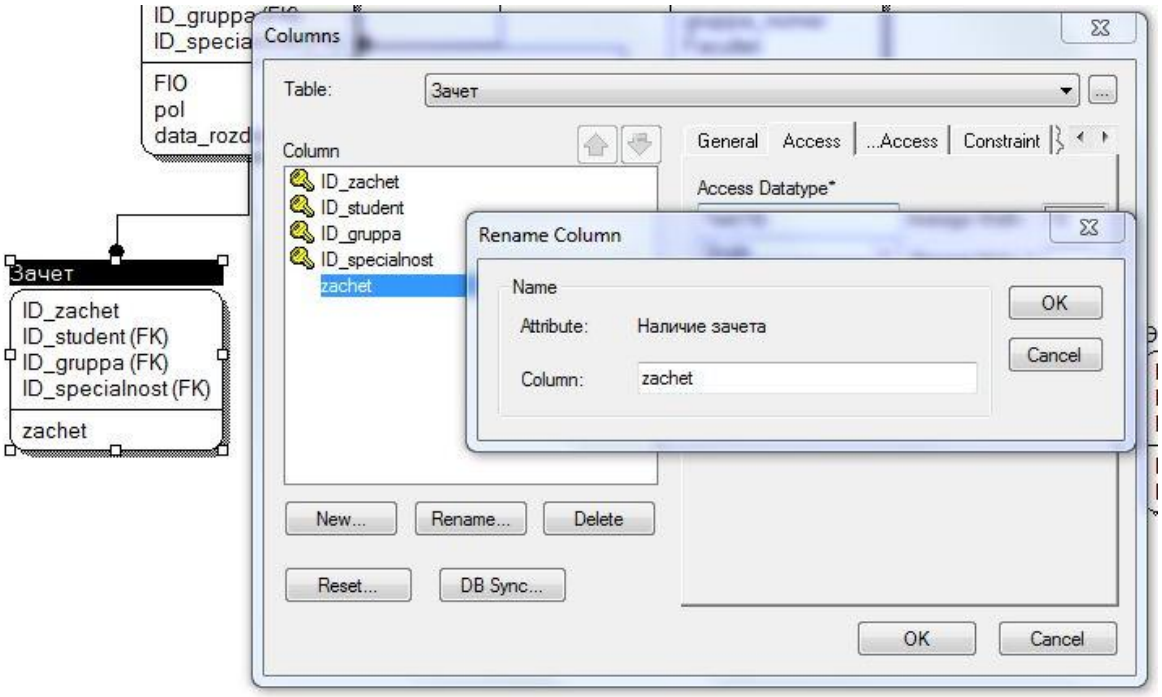


Рисунок 5.26. *– Переименование колонок*

После переименования следует проверить значения типа данных для каждого атрибута. Это делается на вкладке Access диалогового окна Columns (рисунок 5.27).

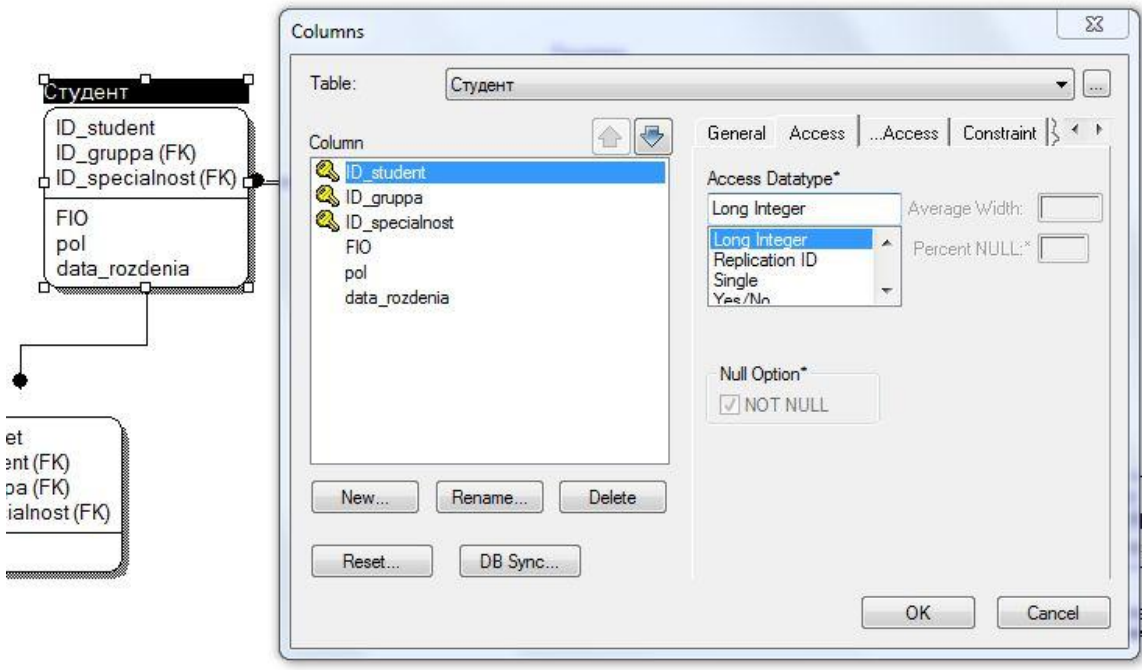


Рисунок 5.27. *– Установка типов данных для выбранной СУБД*

Предлагаемые типы данных зависят от типа данных, установленного для атрибута на вкладке General, установленные при проектировании логической модели.

Если при создании модели указан другой тип сервера, то вкладка с типами данных атрибутов (обычно она идет сразу после вкладки General) будет иметь имя выбранного типа базы данных. Для изменения типа базы данных на стандартной панели выбрать подпункт Choose Database пункта меню Database (рисунок 5.27).

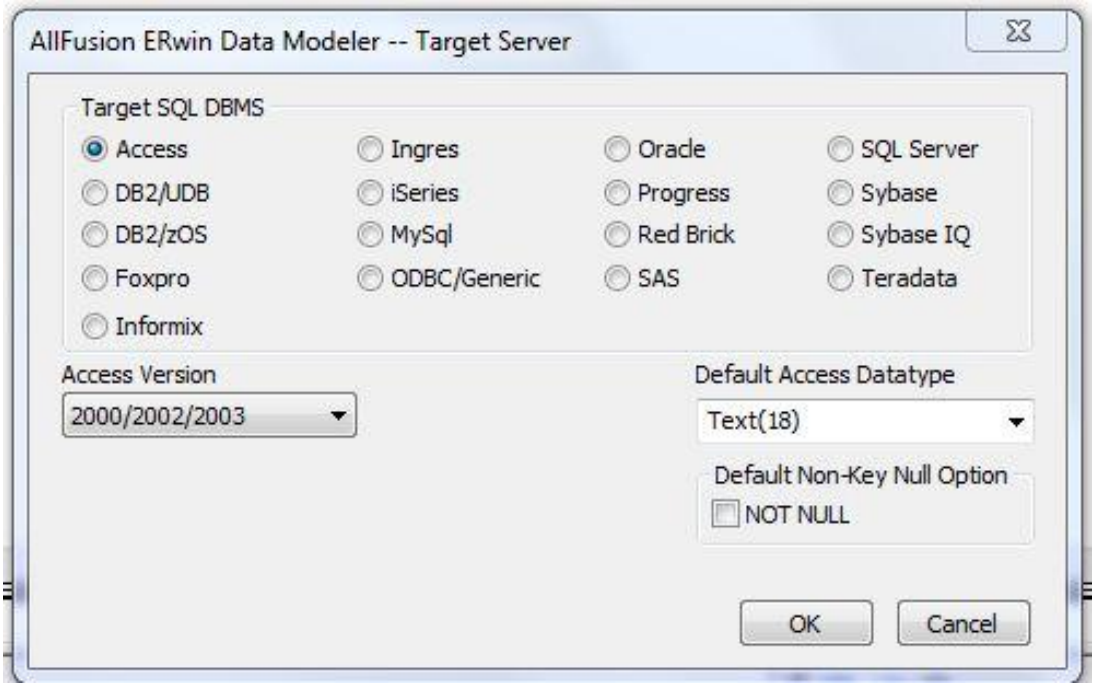


Рисунок 5.28. *– Окно выбора целевого сервера.*

На уровне физической модели целесообразно проверить тип данных для всех атрибутов (колонок) типов атрибутов. В физической модели возможно отображение информации об атрибутах для всех сущностей. Для отображения необходимо вызвать контекстное меню модели (щелкнуть правой кнопкой мыши на свободном месте модели), в появившемся меню выбрать пункт Table Display, и далее установить флажки пунктов Column Datatype, Primary Key Designator, Foreign Key Designator (рисунок 5.29).

Перед завершением работы над моделью следует проверить действия сервера при различных действиях с БД: добавлении, удалении, изменении записей. Для этого следует открыть окно свойств связи (через контекстное меню или двойным щелчком на линии связи) и на вкладке RI Actions установить требуемые ограничения (рисунок 5.30).

Основные действия:

– None, No Action – нет действий;
– Restrict – при наличии связанных записей действие запрещено;
– Cascade – действие распространяется на связанные записи;

– Set Null – установить значение внешнего ключа в NULL.

Разработанную модель целесообразно транслировать в выбранную СУБД. Для этого необходимо: выбрать базу данных и настроить опции генерации таблиц.

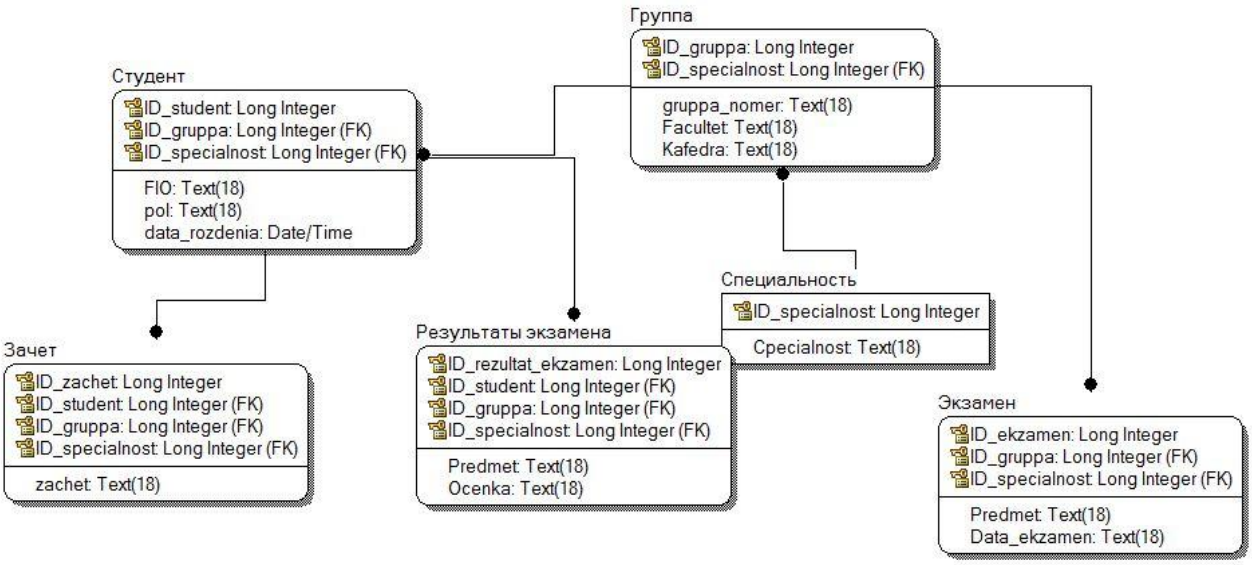


Рисунок 5.29.- *Физическая модель с типами данных*

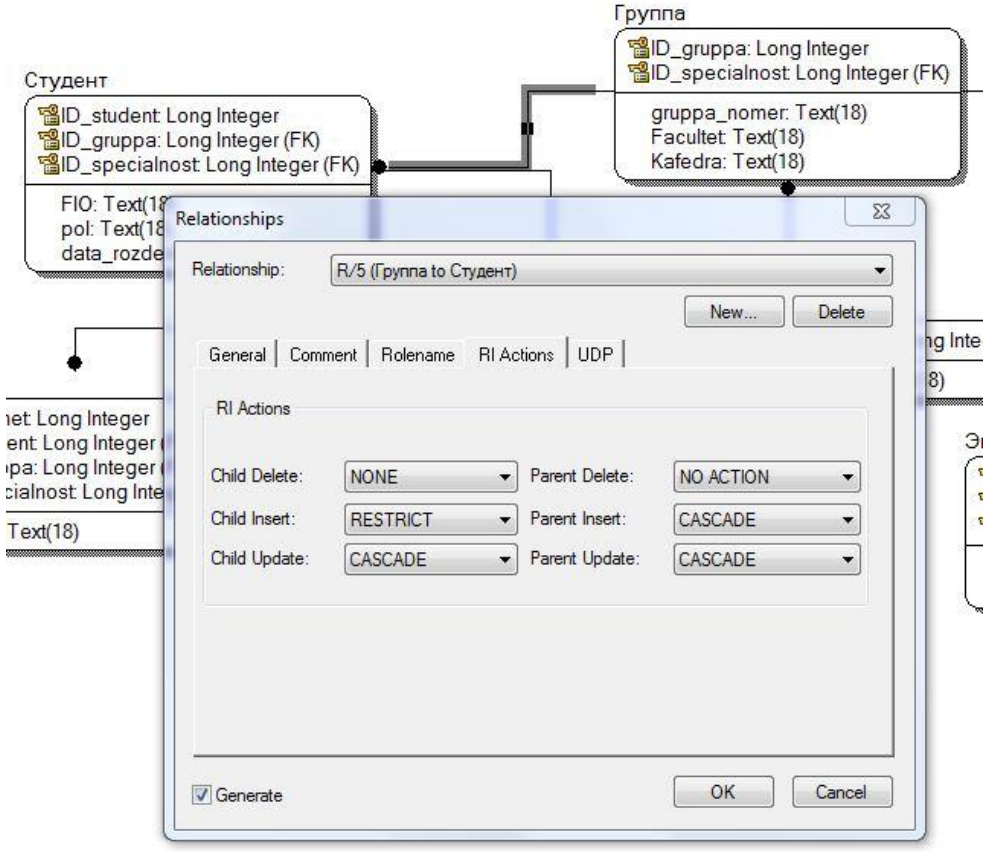


Рисунок 5.30. *– Окно свойств связи*

Для выбора базы данных следует в меню Database выбрать пункт Database Connection (рисунок 5.31).

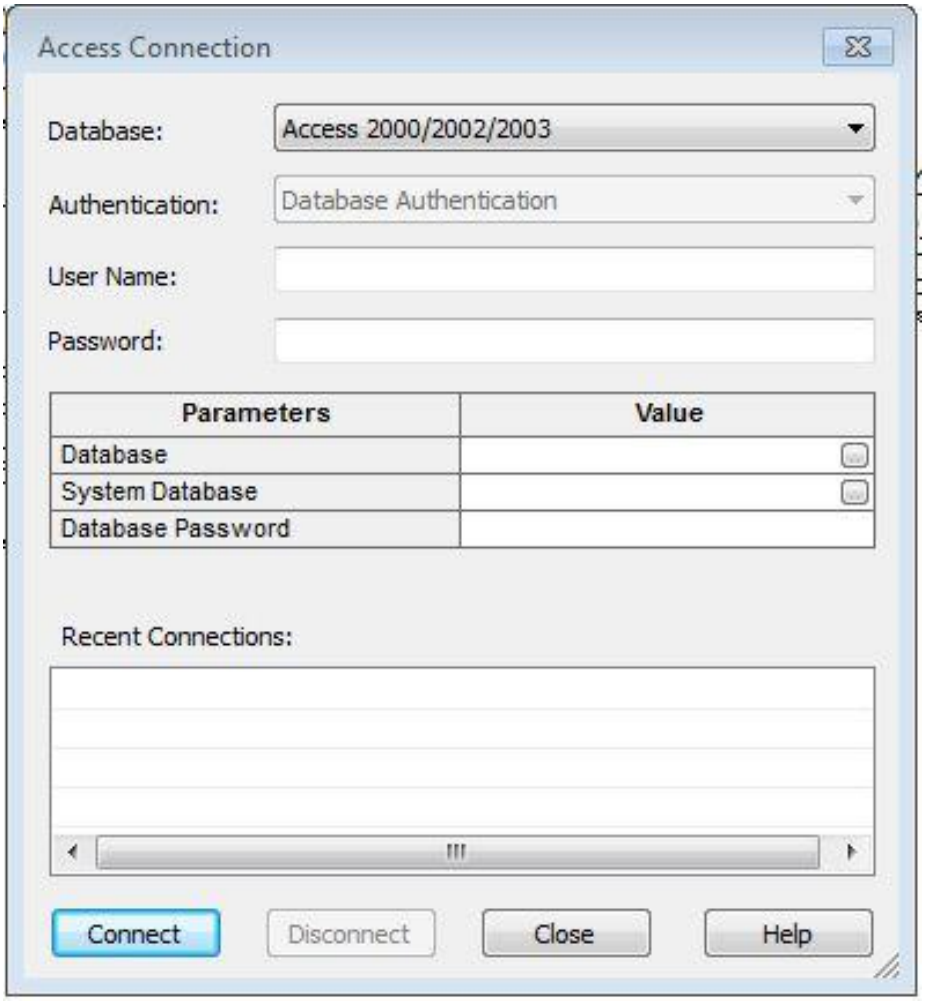


Рисунок 5.31. *– Диалоговое окно соединения с сервером СУБД*

В появившемся диалоге указываются необходимые параметры.

После настройки соединения с базой данных осуществляется трансляция модели данных на сервер БД. Для этого в меню Tools выбирается пункт Forward Engineering/Schema Generation…(рисунок 5.32).

После завершения всех настроек кнопка Generate используется для создания структуры данных по модели на сервере БД.

При необходимости кнопка Preview служить для отображения SQL-кода (рисунок 5.33).

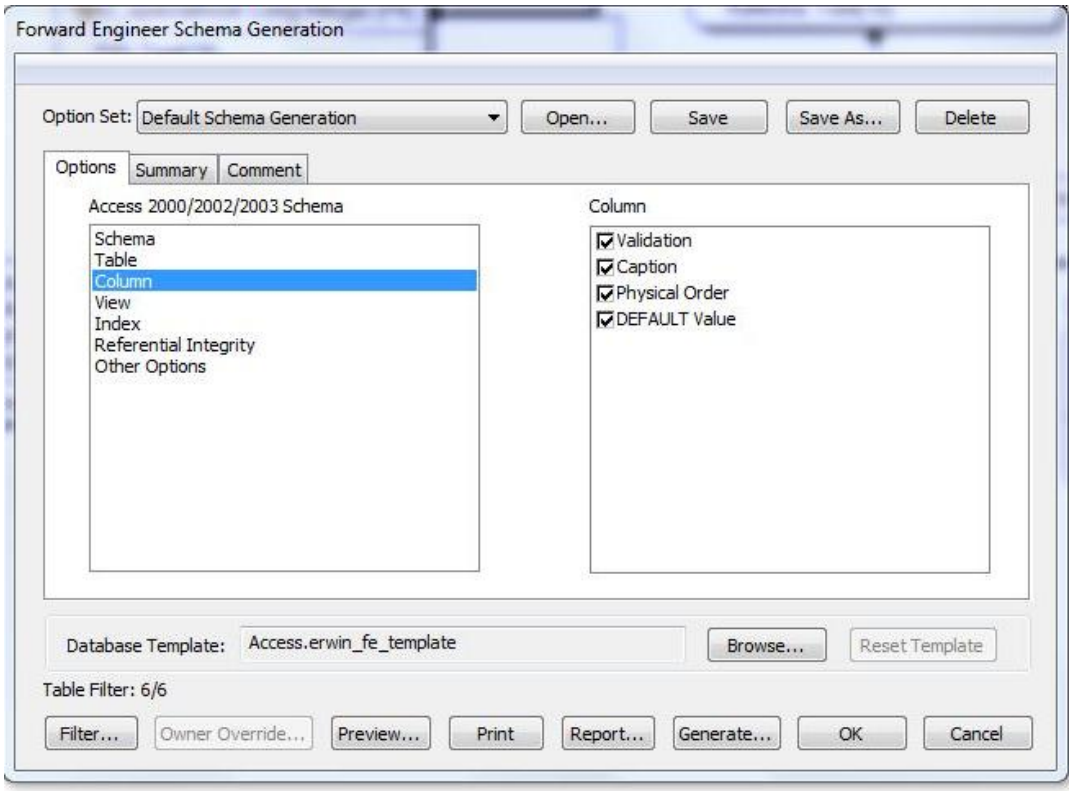


Рисунок 5.32. *– Окно трансляции модели данных на сервер*

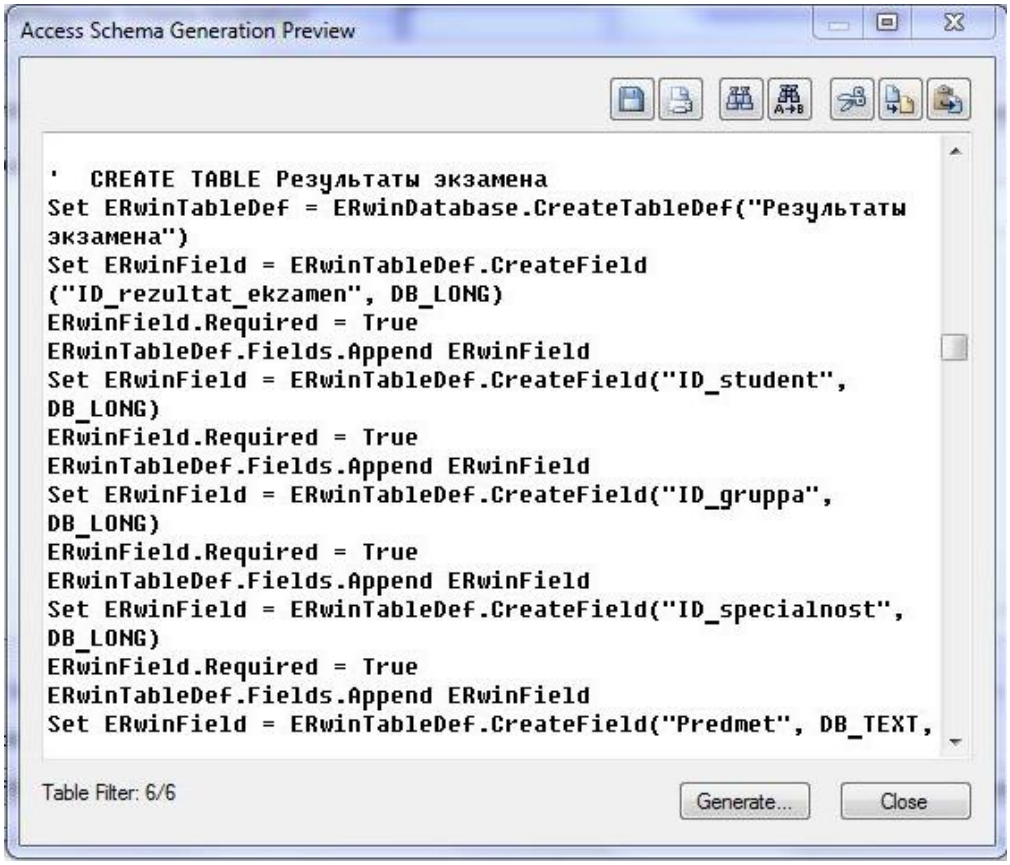


Рисунок 5.33. *– SQL –код создания таблиц модели.*

ЗАКЛЮЧЕНИЕ

Умение использовать инструменты моделирования BPwin и ERwin, является лишь частью моделирования процессов и данных. Кроме этого необходимо понимать, когда решаются задачи моделирования данных и как осуществляется сбор требований к информации, которые должны быть представлены в модели данных. Проведение рабочих исследований обеспечивает благоприятные условия для сбора требований к информации, при этом необходимо включать экспертов предметной области, пользователей и специалистов в области информационных технологий.

Результирующую модель данных необходимо сравнить с корпоративной моделью, если это возможно, для гарантии того, что она не конфликтует с существующими моделями объектов и включает в себя все необходимые объекты.

Модель данных состоит из логической и физической моделей, отображающих требования к информации.

Логическая модель должна быть приведена к третьей нормальной форме. ERwin реализует подход «сущность-связь» (ER) и позволяет создавать объекты для представления требований к информации.

Наиболее трудоемкими этапами разработки информационных систем являются этапы анализа и проектирования, в процессе которых CASE-средства обеспечивают качество принимаемых технических решений и подготовку проектной документации.

CASE-средства – инструментальные системы, покрывающие все стадии жизненного цикла программных систем от анализа требований до сопровождения. Современные CASE-средства охватывают обширную область поддержки многочисленных технологий проектирования информационных систем: от простых средств анализа и документирования до полномасштабных средств автоматизации. Это предполагает построение структурных или иных диаграмм, использование многообразной цветовой палитры, сквозную проверку синтаксических правил.

Основными компонентами CASE-средств являются графические и трансляторные приложения. Графические средства моделирования предметной области позволяют в наглядном виде изучать существующую информационную систему, модернизировать ее в соответствии с поставленными целями и имеющимися ограничениями, обеспечивают удобство разработки. Трансляторы позволяют автоматизировать перевод программ или спецификаций в другое представление.

Современный рынок программных средств насчитывает около 300 различных CASE-средств, наиболее мощные из которых, так или иначе, используются практически всеми ведущими западными фирмами.

Изложенный в пособии материал может быть использован при проведении практических и лабораторных занятий для студентов специальности «Прикладная информатика в экономике». Кроме того, целесообразно использовать Case-средства и при обучении студентов по специальности «Учитель информатики». Это обусловлено следующими факторами.

1. Необходимость разработки информационных сред на основе баз данных и баз знаний, позволяющих осуществить как прямой, так и удаленный доступ к информационным ресурсам.

2. Целесообразность создания универсальной системы компьютерной поддержки научно-методического обеспечения в образовании и последующего применения этого обеспечения в традиционной и дистанционной технологиях.

3. Важность принятия оптимальных решений, связанных с информатизацией научных исследований и разработок в системе образования.

4. Необходимость разработки функционирующих и адекватных заданным требованиям систем контроля качества образования.

Литература

1. AllFusion ERwin Data Modeler проектирование баз данных, хранилищ данных, витрин данных
2. AllFusion® ERwin® Data Modeler. Implementation Guide. r7.2
3. CASE-инструмент BPWIN. www.itstan.ru/content/view/2032/1853/
4. Аксенов К.А., Клебанов Б.И. Работа с CASE-средствами BPwin, ERwin. – Екатеринбург, 2004
5. Вагин В.Н. Искусственный интеллект в case-технологии / Вагин В.Н., Головина Е.Ю., Салапина Н.О.// Программные продукты и системы № 3. – 1996.
6. Вендров А.М.. CASE – технологии. Современные методы и средства проектирования информационных систем, – М.: Финансы и статистика, 1998.
7. Вендров А.М. Ниша и внедрение CASE-средств // Настольный журнал ИТ руководителя / Директор информационной службы №11. – 2000.
8. Ветлугина, И.М. Информационное моделирование в управлении современными экономическими системами. – М.: Компания Спутник+, 2005.
9. Гнатуш А. CASE-технологии: что, когда, как? // CITFORUM http://www.interface.ru/home.asp?artId=987
10. Горин С.В., Тандоев А.Ю. Применение CASE-средства ERwin 2.5 для информационного моделирования в системах обработки данных. – Материалы конференции «Корпоративные базы данных '96» – CITForum.ru
11. Грекул В.И. Проектирование информационных систем. Курс лекций. – Интернет- университет Информационные технологии. – http://www.intuit.ru/department/se/devis/1/
12. Дубейковский В.И. Эффективное моделирование с CA Erwin Process Modeler (AllFusion) PM. – М.: Диалог-МИФИ, 2009. – 384 с: ил.
13. Дэвид А. Марка и Клемент МакГоуэн. Методология структурного анализа и проектирования SADT

14. Кононыхин В. Н. Методическое пособие по созданию моделей бизнес процессов формате IDEF0 и IDEF3 при помощи Bpwin 2.5. - Ростов-на-Дону: Парус-Дон, 2003. – 80 с: ил.
15. Красильникова М.В.. Проектирование информационных систем: Учебное пособие – М.: МИСиС, 2004. – 106 с.
16. Ляхов В. Ф. Прикладная информатика в экономике. Проектирование информационных систем. Часть 1 – 4. – Ставрополь, СевКавГТУ, 2006 г., 560 с.
17. Маклаков С. В. Моделирование бизнес-процессов с BPwin 4.0. – Москва.:ДИАЛОГ-МИФИ, 2002. – 209 с: ил.
18. Маклаков С.В. BPwin и ERwin: CASE-средства для разработки информационных систем. – М.: Диалог-Мифи, 1999. – 295 с.
19. Маклаков С.В. Моделирование бизнес-процессов с ALLFusion PM. – М.: Диалог-МИФИ, 2008. – 224 с: ил.
20. Методология IDEF0 и программный продукт BPwin Учебно-методическое пособие. – Нижний Новгород, 2007
21. Методология IDEF1X и программный продукт ERWin: Учебно-методическое пособие. – Нижний Новгород, 2007
22. Методология функционального моделирования IDEF0. Руководящий документ. Издание официальное. – М.: ИПК Издательство стандартов, 2000, 200 г. – 75 с: ил.
23. Моделирование экономических и производственных процессов предприятий с использованием CASE- средства ERwin: Методические указания к лабораторным работам по курсу «Технико-экономический анализ деятельности предприятий». Сост.: Г.Г. Куликов, Н.О. Никулина, Е.Б. Старцева, А.Ю. Алькин. – Уфа, 2002. – 31 с.
24. Описание стандартов семейства IDEF. www.idef.org
25. Рекомендации по стандартизации. Информационные технологии поддержки жизненного цикла продукции. Методология функционального моделирования. Continuous acquisition and life-cycle support. Methodology of functional modeling. Приняты и введены в действие Постановлением Госстандарта России от 2 июля 2001 г. N 256-ст
26. Рыбанов А.А. Инструментальные средства автоматизированного проектирования баз данных: Учебное пособие и варианты заданий к лабораторным работам по дисциплине «Базы данных» / ВолгГТУ, Волгоград, 2007. – 96 с.

27. Тоискин В.С., Красильников В.В. Антропологический аспект проектирования информационного пространства единого педагогического комплекса. – Ставрополь: Изд-во СГПИ, 2008 -164 с.
28. Федорова Д.Э., Семенов Ю.Д., Чижик К.Н. CASE-технологии. – М.: Горячая линия Телеком, Радио и связь, 2005. – 160 с.
29. Федотова Д.Э., Семенов Ю.Д., Чижик К.Н. CASE-технологии: Практикум. – М.: Горячая линия-Телеком, 2005.-160 с: ил.
30. Чертовской В.Д. Базы и банки данных: Учебное пособие СПб: Изд-во МГУП, 2001.

Оглавление

Публикуется в авторской редакции

Компьютерная верстка П.Г. Немашкалов

Подписано в печать 1.12.10

Формат 60x84 $^{1}/_{16}$ Усл.печ.л. 7,61 Уч.-изд.л. 5,22

Бумага офсетная Тираж 100 экз. Заказ 82

Отпечатано в ООО «РБК-Сервис».