Министерство образования Российской Федерации

Международный образовательный консорциум «Открытое образование»

Московский государственный университет экономики, статистики и информатики

АНО «*Евразийский открытый институт*»

С.М. Диго

Базы данных

Часть 1. Введение в банки данных Методология проектирования

Учебно-практическое пособие

Москва 2004

УДК 004.65
ББК 32.973.202
Д 44

Дигo С.М. **Базы данных. – Ч. 1. Введение в банки данных. Методология проектирования:** Учебно-практическое пособие / Московский государственный университет экономики, статистики и информатики. – М., 2004. – 156 с.

ISBN 5-7764-0308-1

Оглавление

Глава 1. ВВЕДЕНИЕ В БАНКИ ДАННЫХ

1.1. ПОНЯТИЕ БАНКА ДАННЫХ

Основные понятия

Банк данных (БнД) является современной формой организации хранения и доступа к информации. «*Банк данных* – это система специальным образом организованных данных (*баз данных*), программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных» [11].

В данном выше определении БнД, с одной стороны, подчеркивается, что банк данных является сложной системой, включающей в себя все обеспечивающие подсистемы, необходимые для функционирования любой системы автоматизированной обработки данных.

С другой стороны, в этом определении также обозначены и основные *отличительные особенности банков данных:*

- Базы данных создаются обычно не для решения какой-либо одной задачи для одного пользователя, а для многоцелевого использования.
- Базы данных отражают определенную часть реального мира. Надо стремиться, чтобы вся информация, описывающая предметную область, фиксировалась в базе данных однократно, накапливалась и поддерживалась в актуальном состоянии централизовано, а все пользователи, которым эта информация нужна, должны иметь возможность работать с ней.
- Базы данных – это специальным образом организованные данные. Эти особенности в организации данных заключаются, прежде всего, в том, что БД представляют собой системы взаимосвязанных данных, единство и целостность которых поддерживается специальными программными средствами.
- Для функционирования БнД необходимо наличие специальных языковых и программных средств (называемых *СУБД – Система Управления Базами Данных*), облегчающих для пользователей выполнение всех операций, связанных с организацией хранения данных, их корректировки и доступа к ним.

Нельзя сказать, что в рассматриваемой нами сфере установилось терминологическое единство. Так, в англоязычной литературе понятие «банк данных» используется редко. В некоторых из этих источников используется понятие «система баз данных» (database system), которое по своему содержанию близко введенному понятию банка данных (система баз данных включает базу данных, систему управления базами данных, соответствующее оборудование и персонал) [2]. Согласно семантике русского языка «система баз данных» воспринимается у́же, чем то, что это понятие обозначает в действительности. Поэтому слово «банк» является в этом смысле лучше, так как «банк» привычно обозначает не только то, что хранится в нем, но и всю инфраструктуру (вспомните хотя бы понятие «банк» как финансовое учреждение – это ведь не просто «куча денег»). Очевидно, что нельзя отождествлять понятие «база данных» и «банк данных», как это иногда происходит в некоторых литературных источниках.

Терминологические различия наблюдаются и при определении других понятий в области БнД. Особенно это касается терминов, используемых в конкретных программных системах. В связи с тем, что терминология конкретных СУБД сильно различается, нельзя описать общие принципы построения БнД, пользуясь терминологией какой-либо одной из них. В учебном пособии будут введены термины, которые, по мнению автора, в наибольшей степени соответствуют отображаемым ими понятиям.

Следует отметить, что использование тех или иных терминов зависит от аспекта рассмотрения изучаемой проблемы. Так, например, в [10] под базой данных понимается практически любая совокупность данных, которая может быть обработана с помощью ЭВМ. И это оправдано, так как права собственности и иные права не могут зависеть от того, при помощи какого программного средства созданы файлы, и какой у них способ организации. Но такое широкое толкование термина БД в курсе «Проектирование баз данных» приведет к нивелированию особенностей банков данных как особой информационной технологии.

Преимущества БнД

Особенности «банковской» организации данных определяют их основные преимущества перед «небанковской» организацией.

Наличие единого отображения определенной части реального мира позволяет обеспечить непротиворечивость и целостность информации, возможность обращаться к ней не только при решении заранее предопределенных задач, но и с нерегламентированными запросами. Интегрированное хранение сокращает избыточность хранимых данных, что приводит к сокращению затрат не только на создание и хранение данных, но и на поддержание их в актуальном состоянии.

Использование БнД при правильной его организации должно существенно изменить деятельность организации, где он внедряется: привести к обеспечению большей доступности данных для всех категорий сотрудников, сокращению документооборота, возможности получения разнообразных по форме и содержанию документов, перераспределению функций между сотрудниками и изменению характера выполняемых функций и, как следствие, улучшить всю систему управления предприятием.

Централизованное управление данными также дает целый ряд преимуществ. Использование СУБД обеспечивает высокое качество выполнения функций по управлению данными и облегчает процесс создания информационных систем (ИС).

Выделение специальной группы сотрудников, выполняющих функции по проектированию и развитию БнД (администраторов БД), и освобождение от этих функций всех остальных пользователей не только приводит к снижению требований к остальным участникам процесса создания и функционирования БнД, но и повышает качество разработок, так как вопросами организации данных занимается небольшое число профессионалов в этой области.

Преимуществом банков данных является также то, что они обеспечивают возможность более полной реализации принципа независимости прикладных программ от данных, чем это возможно при организации локальных файлов.

Пользователи БнД

В процессе создания и эксплуатации БнД с ним взаимодействуют пользователи разных категорий (рис. 1.1). Базы данных создаются для удовлетворения потребностей *конечных пользователей*. Чаще всего – это специалисты конкретных предметных областей, использующие БД для выполнения своих профессиональных обязанностей. В последнее время БД все чаще используются и для удовлетворения непроизводственных информационных потребностей. Конечные пользователи – наиболее многочисленная группа пользователей. Нельзя недооценивать важности этой группы пользователей и не понимать их специфических особенностей для каждой из категорий конечных пользователей БнД.

Специфическими пользователями БнД являются сотрудники информационных служб. Они пользуются, в основном, метаинформацией. Часто бывает желательным, чтобы другая информация была для них закрыта. Кроме того, они используют и другие ресурсы БнД для выполнения своих функций.

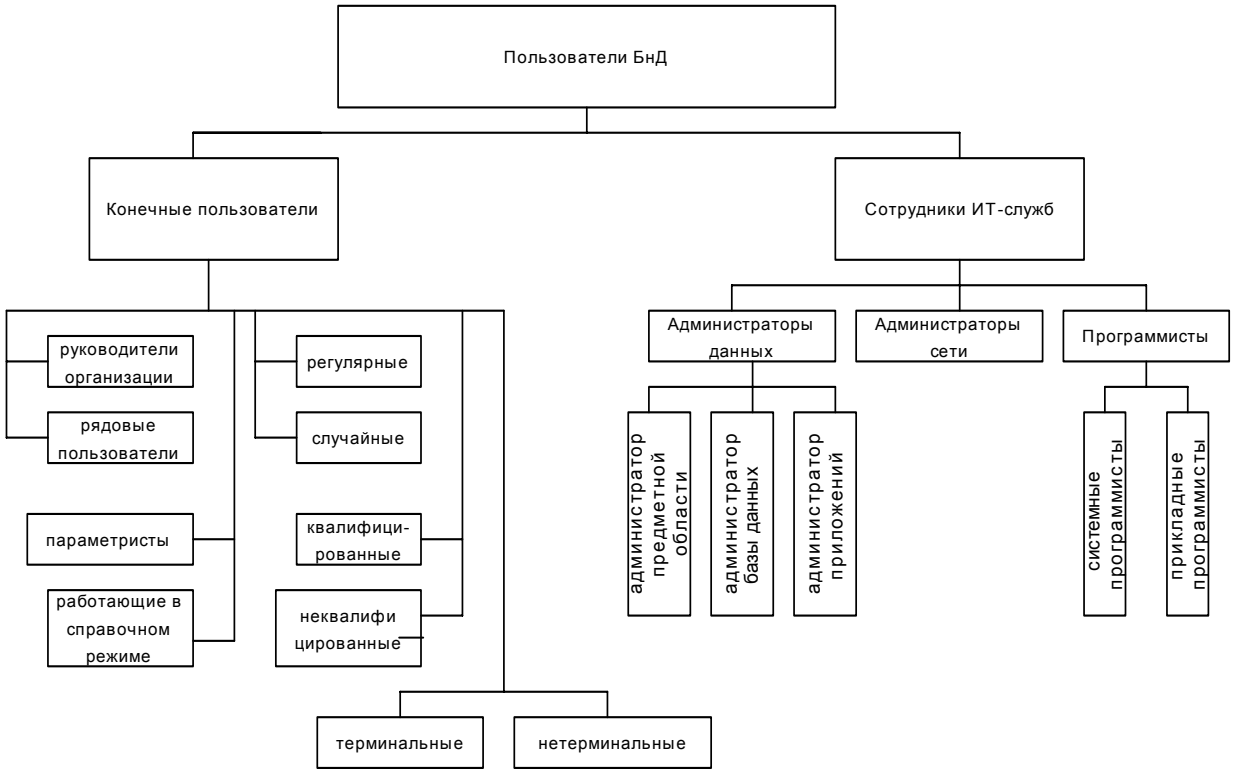


Рис. 1.1. Пользователи БнД

Категория «Конечные пользователи» неоднородна: конечные пользователи различаются широтой информационных потребностей, квалификацией, режимами взаимодействия с БнД и др. Это могут быть случайные пользователи, обращающиеся к базе данных время от времени, а могут быть и регулярные пользователи. Конечные пользователи могут отличаться друг от друга и степенью владения вычислительной техникой. От конечных пользователей не должно требоваться каких-то специальных знаний в области вычислительной техники и языковых средств.

При создании БнД важно не только построение классификационной схемы, но и распределение реальных конечных пользователей по группам, так как от характеристики пользователей будут зависеть принимаемые проектные решения.

В связи с тем, что использование БнД оказывает влияние на все аспекты деятельности организации, особую роль играют руководители организации. Именно они должны обеспечить проведение единой информационной политики и организацию взаимодействия различных подразделений через общую базу данных. Они должны создавать подразделения, отвечающие за создание и функционирование БнД, определять функциональные обязанности сотрудников, которые существенно изменятся с внедрением БнД. Кроме того, руководители организации выступают в качестве конечных пользователей с наиболее высоким приоритетом.

Отдельные пользователи в процессе работы с базой данных могут менять содержание БД – это так называемые пользователи-параметристы. Другие могут только использовать хранящуюся в БД информацию.

Пользователи могут взаимодействовать с БД как непосредственно (терминальные пользователи), так и через посредников.

Понятием «Конечные пользователи» определяется не только отдельное лицо или группа лиц, но и вычислительные процессы / задачи, а иногда и целые системы, взаимодействующие с БнД.

В зависимости от особенностей создаваемого банка данных круг его конечных пользователей может существенно различаться.

Категория «сотрудники информационных служб» также является неоднородной. В рамках курса «Базы данных» наибольший интерес для нас представляют «*Администраторы БнД*» – лица, ответственные за создание БнД и его надежное функционирование, за соблюдение регламента доступа к хранимым за развитие БнД.

Наличие в составе СУБД средств, ориентированных на разные категории пользователей, делает возможной работу с базой данных не только профессионалов в области обработки данных, но практически любого пользователя, причем это использование может быть как для их профессиональных целей, так и для удовлетворения потребности в информации в быту и т.п.

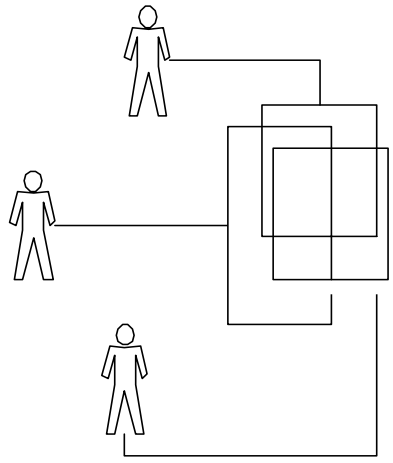


Рис. 1.2. Пересечение информационных потребностей пользователей

Предпосылки широкого использования БнД.

Очевидные преимущества БнД и объективные предпосылки их создания привели к широкому их использованию. К числу предпосылок применения БнД относятся следующие:

- объекты реального мира находятся в сложной взаимосвязи между собой. Это приводит к необходимости, чтобы их информационное отражение также представляло единое взаимоувязанное целое;
- информационные потребности различных пользователей существенно пересекаются, что делает целесообразным использование единых баз данных и обеспечение доступа к ним разных пользователей (рис.1.2);
- функции создания и ведения информационного фонда и предоставления нужных данных тем или иным процессам являются универсальными, общими при решении разнообразных задач. Создание специализированных программных средств для управления данными приводит к повышению уровня выполнения этих функций и сокращению трудоемкости создания информационных систем;

- современный уровень развития технического и программного обеспечения, а также теории и практики построения информационных систем позволяют создавать эффективные БнД.

Требования к БнД

Особенности «банковской» организации данных позволяют сформулировать основные требования, предъявляемые к БнД:

- адекватность отображения предметной области (полнота, целостность и непротиворечивость данных, актуальность информации, т.е. ее соответствие состоянию отображаемой реальной системы на данный момент времени);
- возможность взаимодействия пользователей разных категорий и в разных режимах; обеспечение высокой эффективности доступа для разных приложений;
- дружелюбность интерфейсов и малое время на освоение системы, особенно для конечных пользователей;
- обеспечение секретности и конфиденциальности для некоторой части данных; определение групп пользователей и их полномочий;
- обеспечение взаимной независимости программ и данных;
- обеспечение надежности функционирования БнД; защита данных от случайного и преднамеренного разрушения; возможность быстрого и полного восстановления данных в случае их разрушения; технологичность обработки данных;
- приемлемые характеристики функционирования БнД (стоимость обработки, время реакции системы на запросы, требуемые машинные ресурсы и др.).

Недостатки БнД

Недостатки БнД вытекают из их достоинств. Создание интегрированной системы, естественно, сложнее, чем создание множества локальных систем. Как следствие, предъявляются высокие требования к квалификации разработчиков БнД. В результате интеграции возможна некоторая потеря эффективности отдельных приложений (но общая эффективность всей системы будет выше). Для управления данными требуется специализированное программное обеспечение, которое, в зависимости от класса системы, может быть сравнительно дорогим, предъявляющим повышенные требования к техническим средствам. Эксплуатация распределенных корпоративных БнД – процесс сложный и дорогостоящий.

Но, несмотря на некоторые недостатки, присущие такой форме организации данных, преимущества БнД значительно превосходят их недостатки. Кроме того, имеется очень широкий круг СУБД разных классов и технологий их использования. Правильный выбор системы позволит свести отрицательные последствия к минимуму.

1.2. КОМПОНЕНТЫ БАНКА ДАННЫХ

Банк данных является сложной человеко-машинной системой, включающей различные взаимосвязанные и взаимозависимые компоненты (рис. 1.3).

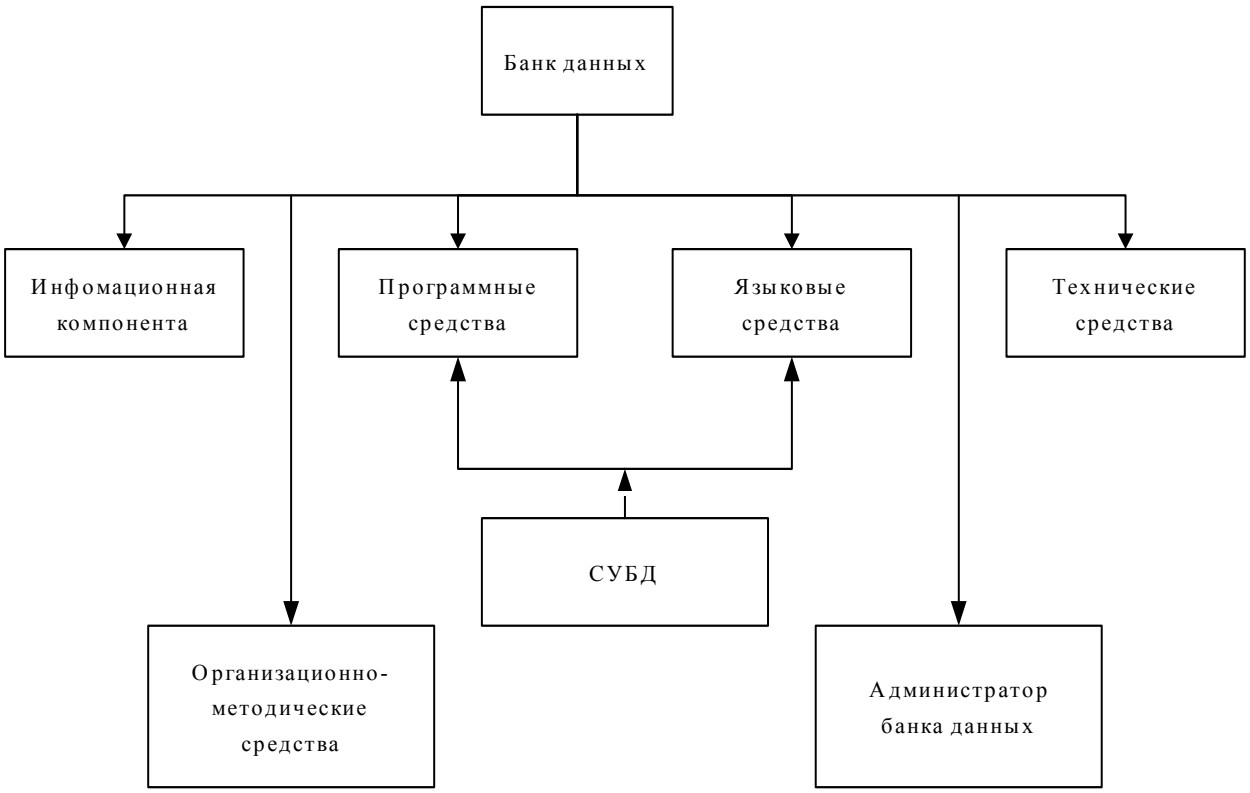


Рис. 1.3. Компоненты банка данных

Информационная компонента

Ядром БнД является база данных. *База данных* – это поименованная совокупность взаимосвязанных данных, находящихся под управлением СУБД.

Существует множество определений базы данных. Некоторые из них имеют право на существование. Другие устарели и не соответствуют современным представлениям о БД. Так, в ранних определениях баз данных указывалось на их не избыточность, отсутствие дублирование данных в них. На самом деле это не так. В базах данных может наблюдаться дублирование информации. Оно может быть вызвана спецификой используемой модели данных, не позволяющей полностью устранить дублирование, или технологическими причинами (обеспечение большей надежности, сокращение времени реакции системы и др.). Но это должна быть управляемая избыточность, причины и цели возникновения которой известны администратору базы данных и управляются как им, так и СУБД.

В настоящее время действует закон «О правовой охране программ для электронных вычислительных машин и баз данных» [10]. В этом законе дается следующее определение базы данных: «База данных – это объективная форма представления и организации совокупности данных (например, статей, расчетов), систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны с помощью ЭВМ» [10, ст.1]. Учитывая назначение этого закона, вполне естественно, что сделан иной акцент, чем в данном нами определении БД; определение, используемое в тексте данного Закона, является более широким, чем приведенное нами.

Мы в качестве рабочего будем пользоваться данным нами в п. 1.1 определением, и не всякие файлы будем считать базами данных.

В технической документации некоторых СУБД, а также в некоторых литературных источниках в состав БД включаются не только собственно хранимые данные о предметной области, но и описания БД. Более правильно описания баз данных считать самостоятельными компонентами БнД, даже если они и хранятся вместе с самими данными.

Описания баз данных относятся к *метаинформации*, т.е. информации об информации. Описание баз данных часто называют *схемой*. Кроме того, в БнД могут присутствовать описания отдельных частей базы данных с точки зрения конкретных пользователей. Такое описание называется *подсхемой*.

Кроме описания баз данных в состав метаинформации, хранимой в БнД, может включаться информация о предметной области, необходимая для проектирования автоматизированной информационной системы, о пользователях БнД, о проектных решениях и некоторая другая информация.

Централизованное хранилище метаинформации называется *словарем данных*. В литературе используются также термины словарь-справочник, энциклопедия, репозиторий. В некоторых источниках выявляются различия между этими терминами, в других они используются как синонимы. Для данного уровня рассмотрения для нас эти различия несущественны.

Роль словарной системы особенно возрастает при использовании средств автоматизированного проектирования информационных систем. Для большинства из них репозиторий является ядром всей системы. Кроме того, роль репозитория особо значима в распределенных системах.

К банку данных не относятся немашинные документы, служащие источниками информации, вводимой в БД, файлы входной и выходной информации, архивные файлы, выходные документы. Однако многие СУБД включают в свой состав языковые средства для описания этих компонент. В этом случае сами описания, используемые в процессе функционирования БнД, будут входить в его состав.

Как уже отмечалось выше, терминология, используемая в разных системах и разных литературных источниках, существенно различается. Так, в ранних версиях многих «настольных» реляционных СУБД вообще не использовался «механизм» базы данных (т.е. создавались фактически отдельные реляционные таблицы, каждая из которых запоминалась в отдельном файле базы данных). Во многих литературных источниках каждый такой файл стали называть базой данных, что не правильно.

В некоторых системах, например, Access, под БД понимают совокупность разных объектов: таблиц, запросов, форм, отчетов, макросов и модулей, т.е. понятие базы данных расширено и включает в себя практически все информационные компоненты, созданные для конкретного приложения. В других системах, в частности, в Paradox, для обозначения подобной совокупности взаимосвязанных объектов используется понятие «семейство», что, очевидно, терминологически более правильно.

При работе с конкретной системой надо, прежде всего, уточнить терминологию, используемую в ней.

Программные средства БнД

Программные средства БнД представляют собой сложный комплекс, обеспечивающий взаимодействие всех частей информационной системы при ее функционировании (рис 1.4).

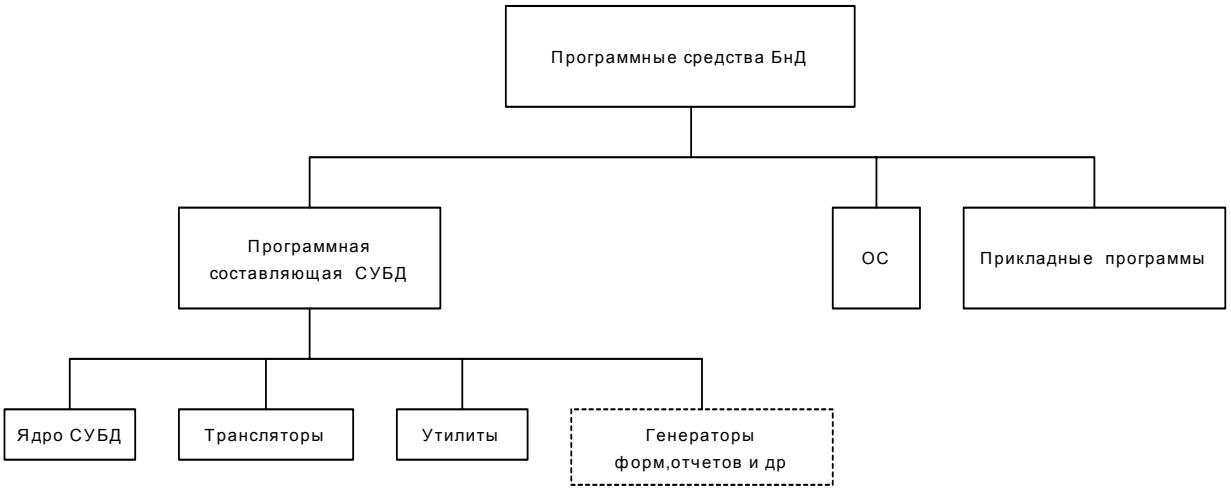


Рис.1.4. Программные средства БнД

Основу программного обеспечения БнД представляют программные компоненты СУБД. Среди них можно выделить ядро СУБД, обеспечивающее создание БД, организацию ввода, обработки и хранения данных, т.е. именно то, что называется «управлением данными», а также другие компоненты, обеспечивающие настройку системы, средства тестирования, утилиты, обеспечивающие выполнение вспомогательных функций, таких как восстановление баз данных, сбор статистики о функционировании БнД и др. Важной компонентой СУБД являются трансляторы или компиляторы для используемых ею языковых средств.

В состав большинства СУБД включены программные компоненты, позволяющие автоматизировать проектирование систем обработки информации (генераторы отчетов, меню и др.). Строго говоря, эти функции не являются непосредственно функциями по управлению данными, но большинство современных программных средств, которые продолжают называться СУБД, выходят за названные рамки и фактически являются мощными комплексными инструментальными средствами, позволяющими автоматизировать процесс создания информационных систем.

Подавляющее большинство СУБД работает в среде универсальных операционных систем (ОС) и взаимодействует с ОС при обработке обращений к БнД. Поэтому можно считать, что ОС также входит в состав БнД.

Для удовлетворения конкретных потребностей пользователей пишутся соответствующие программы[1], которые представляют прикладное программное обеспечение БнД.

В зависимости от используемых технологий создания и функционирования систем могут появляться те или иные дополнительные компоненты. Так, при использовании Case-технологий будут присутствовать соответствующие программные компоненты, поддерживающие проектирование и перепроектирование системы.

При работе в архитектуре клиент-сервер программные средства будут подразделяться на соответствующие компоненты: клиентская часть, обеспечивающая интерфейс пользователя с системой, серверная часть, реализующая обработку запроса на сервере, и связная часть, обеспечивающая взаимодействие элементов в сети.

Программные средства, используемые при создании и эксплуатации БнД, будут также зависеть от масштаба БнД, требований, предъявляемых к нему.

[1] Понятие «программа» здесь трактуется широко. Это могут быть, например, и экранные формы, созданные с использованием визуальных средств, и запросы, написанные на любом языке запросов.

Языковые средства БнД

Языковые средства СУБД являются важнейшей компонентой банков данных, так как, в конечном счете, они обеспечивают интерфейс пользователей разных категорий с банком данных. Набор используемых языков средств широк и разнообразен. Языковые средства, используемые в БнД, можно классифицировать по разным признакам (рис. 1.5).

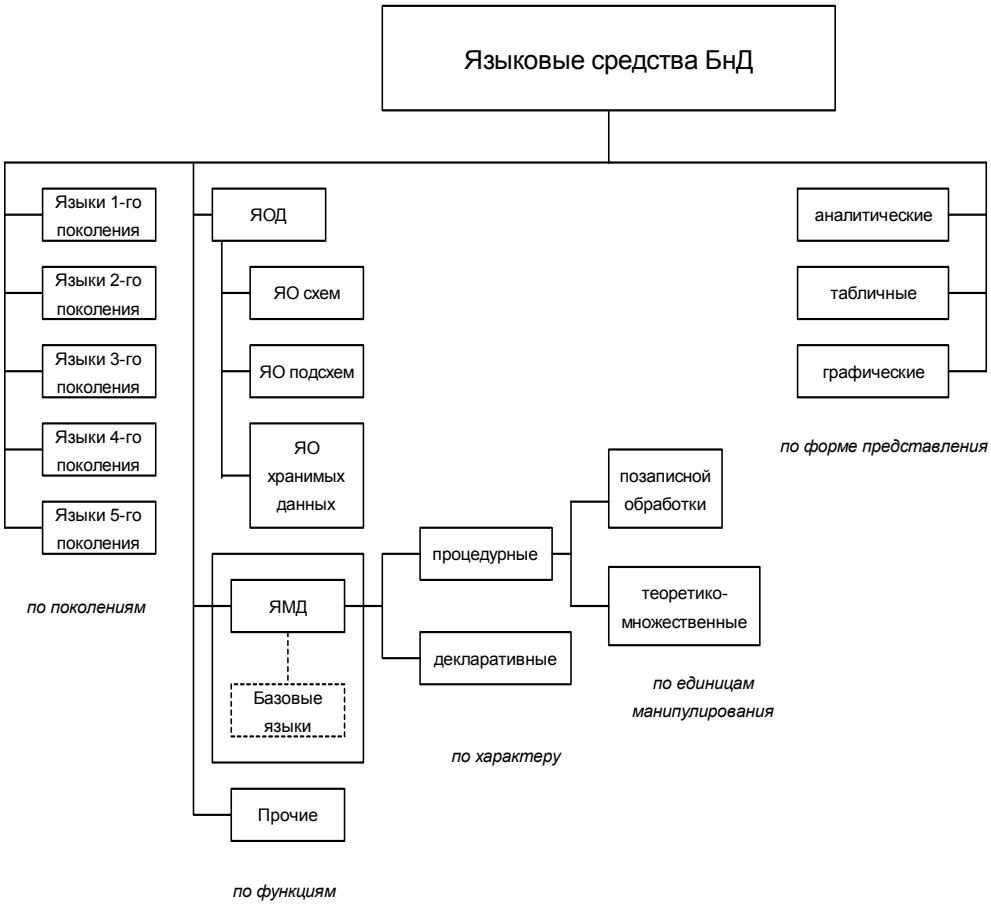


Рис. 1.5. Классификация языковых средств БнД

Языковые средства большинства современных СУБД относятся к *языкам четвертого поколения* (к *первому поколению* языков относят машинные языки, ко *второму* – символические языки Ассемблера, к *третьему* – алгоритмические языки типа PL, COBOL и т.п., которые в 60-е годы назывались языками высокого уровня, но уровень которых гораздо ниже, чем у языков четвертого поколения. Имеются еще и языки *пятого* поколения, к которому относят языки систем искусственного интеллекта).

Языки четвертого поколения создавались по принципу: «люди стоят дороже, чем машины» [1]. При их проектировании использовались следующие принципы:

1. Принцип минимума работы: язык должен обеспечить минимум усилий, чтобы «заставить» машину работать.
2. Принцип минимума мастерства: работа должна быть так проста, как только это возможно; она не должна быть уделом избранных и быть понятной лишь посвященным.
3. Принцип естественности языка, упразднения «инородного» синтаксиса и мнемоники. Язык не должен требовать от пользователей значительных усилий в изучении синтаксиса или содержать много мнемонических или иных обозначений, которые быстро забываются.

4. Принцип минимума времени. Язык должен позволять без существенной задержки реализовывать возникающие потребности в доступе к информации и ее обработке.
5. Принцип минимума ошибок. Технология должна быть спроектирована таким образом, чтобы минимизировать ошибки человека, а уж если они возникли, то, по возможности, «выловить» их автоматически.
6. Принцип минимума поддержки. Механизм языков четвертого поколения должен позволять легко вносить изменения в имеющиеся приложения.
7. Принцип максимума результата. Языки четвертого поколения предоставляют пользователям мощный инструмент для решения разнообразных задач.

На рис. 1.6 представлены компоненты языка четвертого поколения. Как мы видим, здесь представлены все основные «генераторы», наличие которых уже стало традиционным для СУБД разных классов.

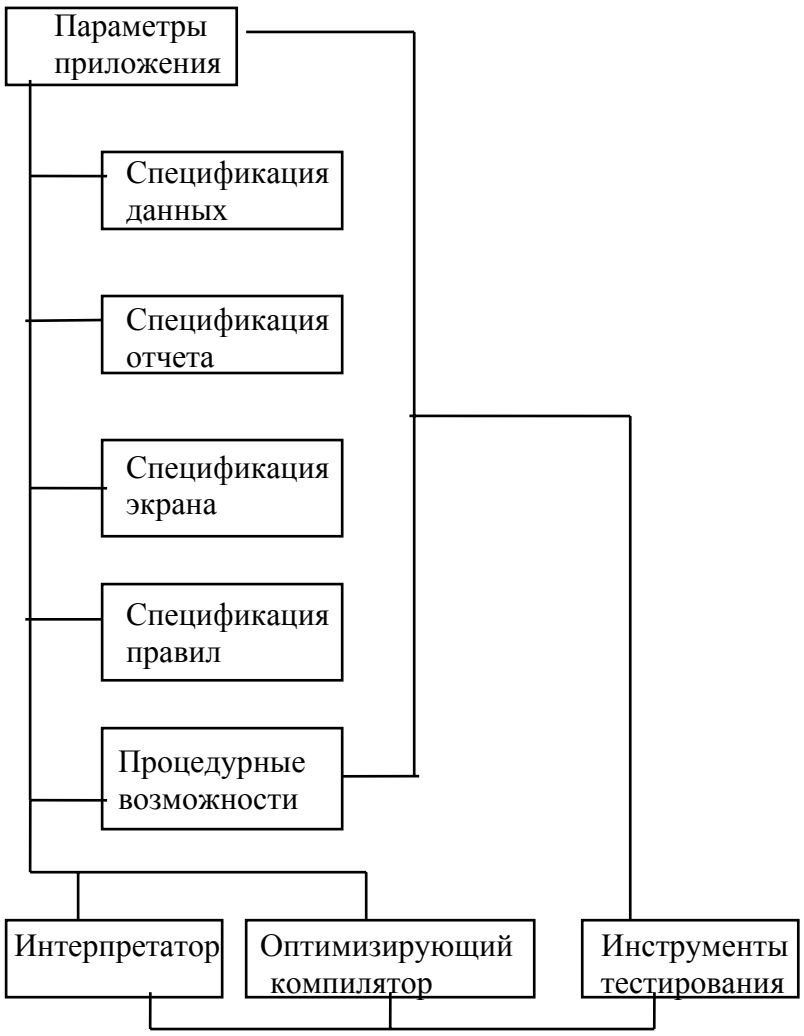


Рис. 1.6. Компоненты языка четвертого поколения

Можно выделить две концепции развития языковых средств: концепцию разделения и концепцию интеграции. При использовании концепции разделения различают *языки описания данных* (ЯОД) и *языки манипулирования данными* (ЯМД). Назначение каждого из этих подклассов ясно из их названия.

Иногда в особую группу выделяют *языки запросов* (ЯЗ). Первоначально под языками запросов понимали языки высокого уровня, ориентированные на конечного пользователя, предназначенные для формирования запросов к БД (в такой трактовке их можно считать одной из разновидностей ЯМД). Однако сейчас ЯЗ понимается шире (см. гл. 3); многие ЯЗ включают в себя еще и возможности описания данных и корректировки БД.

В составе языков описания данных в зависимости от особенностей СУБД поддерживаются все или некоторые из следующих языков: *язык описания схем (ЯОС), язык описания подсхем (ЯОПС), язык описания хранимых данных (ЯОХД), языки описания внешних данных* (входных, выходных). В некоторых СУБД и сами эти разновидности языков, и создаваемые с их помощью элементы ИС являются самостоятельными компонентами, в других – некоторые из них могут объединены.

Языки манипулирования данными разделяются на две большие группы: *процедурные* и *непроцедурные*. При пользовании процедурными языками надо указать, какие действия и над какими объектами необходимо выполнить, чтобы получить результат. В непроцедурных языках указывается, что надо получить в ответе, а не как этого достичь.

Процедурные языки могут различаться по основным информационным единицам, которыми они манипулируют. Это могут быть языки, ориентированные на позаписную обработку данных, и языки, ориентированные на операции над множеством записей. Так, операции реляционной алгебры оперируют целиком отношением, а не каждой его записью.

Примерами непроцедурных языков являются языки, основанные на реляционном исчислении. Представителем языков, основанных на реляционном исчислении кортежей, является широко используемый язык запросов SQL. Табличный язык QBE также является непроцедурным языком.

Языковые средства предназначаются для пользователей разных категорий: конечных пользователей, системных аналитиков, профессиональных программистов. Повышение уровня языковых средств, их дружелюбности приводит к тому, что все большее число функций выполняется пользователями-непрограммистами самостоятельно, без посредников.

По своим ***функциональным возможностям*** выделяют следующие категории языков [1]:

1. Языки, обеспечивающие только возможности запросов. Они обеспечивают вывод требуемых данных на экран или печать в нужном формате. В настоящее время используются редко.
2. Комплексные языки запросов / обновлений. Более развитые языки, которые позволяют формулировать сложные запросы, относящиеся к нескольким взаимосвязанным записям, а также обновлять данные также легко, как и формулировать запросы. Используя их, пользователи могут создавать свои собственные файлы.
3. Генераторы отчетов. Они позволяют выбирать нужные данные из файлов или баз данных и форматировать их в виде требуемых форм документов.
4. Графические языки. Использование графических средств в настоящее время постоянно расширяется. Они позволяют выводить данные в виде различных графиков и диаграмм, а также использовать другие изобразительные возможности. Как и генераторы отчетов, графические языки позволяют осуществлять отбор информации из файлов или баз данных по различным критериям, а также выполнять арифметические и логические манипуляции с данными.
5. Инструментальные средства поддержки решений. Языки этого типа предназначены для создания систем принятия решений. Это могут быть системы типа «что-если», системы, выполняющие временной или трендовый анализ и другие. Возможно использование как универсальных, так и проблемно-ориентированных средств.

6. Генераторы приложений. Языковые средства, предназначенные для генерации приложений, обеспечивают возможность описания непроцедурным путем требуемой обработки информации и дальнейшей автоматической генерации программ.
7. Машино-ориентированные языки спецификаций. Фактически являются генераторами приложений, дальнейшим их развитием. В отличие от генераторов приложений языки спецификаций более универсальны и позволяют специфицировать приложения разных типов.
8. Языки очень высокого уровня. В большинстве случаев приложения строятся при помощи непроцедурных языков. Однако некоторые языки являются процедурными (например, NOMAD), но программирование на них значительно короче, чем, например, на COBOLe.
9. Параметризированные пакеты прикладных программ. Эта категория программных средств известна давно, и «четвертое поколение» относится к таким ППП, которые допускают легкую модификацию самого пакета, позволяют пользователям генерировать собственные отчеты, запросы к базе данных и т.д.
10. Языки приложений. Многие языки четвертого поколения являются универсальными языками. Другие – спроектированы для специфических приложений. Примерами таких языков являются языки для управления финансами, управления работой станков с программным управлением и т.д.

По форме представления различают *аналитические, табличные и графические* языковые средства. Классификация языковых средств по форме представления относится как к языкам описания данных, так и к языкам манипулирования данными. Так, описание таблицы с использованием команды CREATE TABLE языка SQL является примером аналитической формы ЯОД, а описание такой же таблицы в Access и большинстве других настольных СУБД – примером табличной формы описания. В качестве примеров табличной и аналитической формы задания запросов можно привести языки QBE и SQL соответственно.

Достаточно часто бывает, что в рамках одной СУБД для одних и тех же целей могут использоваться языки разных типов. Так, например, во многих СУБД (dBase, FoxPro, и др.) для манипулирования данными могут использоваться:

1. Табличный язык запросов типа QBE.
2. Язык SQL – аналитический ЯЗ, относящийся к классу языков исчисления кортежей.
3. Процедурный язык программирования (для указанных выше систем dBase, FoxPro – это язык xBase, часть операторов которого реализуют операции реляционной алгебры, а другая часть, более значительная по количеству операторов и функций, представляет собой нереляционные операции, обеспечивающие позаписную обработку файлов, организацию циклической и условной обработки, ввод-вывод данных, корректировку, возможность работы с переменными памяти и другие возможности).

Описание данных в этих системах может быть представлено в табличном виде, либо, если определение данных происходит средствами SQL или с использованием операторов языка xBase, – в аналитическом виде.

Кроме упомянутых языковых средств, эти системы включают в себя генераторы экранных форм, отчетов и приложений, а также язык разветвленной иерархической системы «меню», позволяющей пользователю выбрать те действия, которые он желает выполнить.

Часто СУБД обеспечивают автоматическое преобразование «текстов» с одного языка на другой. Так, например, многие СУБД, такие как Access, FoxPro и др., используют языки запросов табличного типа не только для непосредственной реализации запросов, но и как средство для более простого описания запроса и последующего автоматического преобразования его на язык SQL.

Технические средства БнД

В качестве технических средств для банков данных (рис. 1.7) чаще всего используются универсальные ЭВМ, периферийные средства для ввода информации в базу данных и отображения выводимой информации. Иногда используются дополнительные технические средства для хранения больших объемов данных на внешних носителях. Если банк данных реализуется в сети, то необходимы соответствующие технические средства для обеспечения ее работы.

Состав и тип технических средств, на которых реализуются БнД, зависит от многих факторов, основными из которых являются: технические характеристики оборудования, используемые технологии обработки данных, масштаб системы, временные ограничения на время реакции системы, сложность обработки, стоимостные характеристики и др.

Первоначально БнД реализовывались в основном на больших ЭВМ, а для доступа к БД использовались терминалы. В связи со значительным и постоянным улучшением характеристик персональных ЭВМ появилась возможность реализовать банки данных и на машинах этого класса. Но сначала характеристики персональных ЭВМ были недостаточными, чтобы в полной мере реализовать идеологию банков данных. Стала наблюдаться некоторая раздробленность информационных систем, что, в свою очередь, привело к бурному развитию сетевых технологий и использованию соответствующих технических средств.

Существуют и специализированные технические средства, предназначенные для создания и эксплуатации банков данных (машины баз данных), но они не нашли широкого распространения.

В последние годы некоторые фирмы (Oracle, Sun) активно развивают идею использования так называемых «сетевых компьютеров». Эти компьютеры представляют собой дешевые рабочие станции без дисковых накопителей[2], которые будут работать в сети и использовать и программные средства, и данные, которые находятся на сервере.

Использование сетевых компьютеров предполагает обязательное применение мощных ЭВМ в качестве серверов, предъявляет высокие требования к организации хранения данных, к качеству каналов связи. При этом во многом становится предопределенной технология обработки данных (особенно в части распределения функций между клиентом и сервером). Использование сетевых компьютеров обусловлено не столько тем, чтобы сэкономить за счет использования более дешевых компьютеров, сколько желанием упорядочить использование программных средств, упростить систему обработки информации в целом, облегчить и удешевить поддержку системы.

Недостатками такого подхода являются:

- очень большая зависимость от «центральной» системы, потеря самостоятельности конечными пользователями;
- уязвимость системы;
- невозможность / неэффективность обеспечения потребности всех пользователей таким образом (хотя потребности пользователей и пересекаются, но степень пересечения может быть разной; кроме того, не исключается наличие сугубо персональных данных; хранение таких данных на отдаленном сервере приводит к непроизводительным расходам);
- очень высокие требования к серверной части системы.

Другим новым явлением является использование карманных ПК в качестве коммуникационных устройств для доступа к корпоративным данных в глобальных сетях.

[2] Позднее появились предложения создавать сетевые компьютеры с собственными накопителями.

Характеристики карманных компьютеров существенно улучшаются. Для них создается соответствующее программное обеспечение, позволяющее использовать их для мобильных пользователей, работающих в общей системе (тиражирование и синхронизация данных). Эти компьютеры являются более легкими, что немало важно для мобильных пользователей, а также стоят дешевле, чем переносные ПК. Ведущие производители СУБД приспосабливают свои крупномасштабные серверные системы для доступа из карманных ПК.

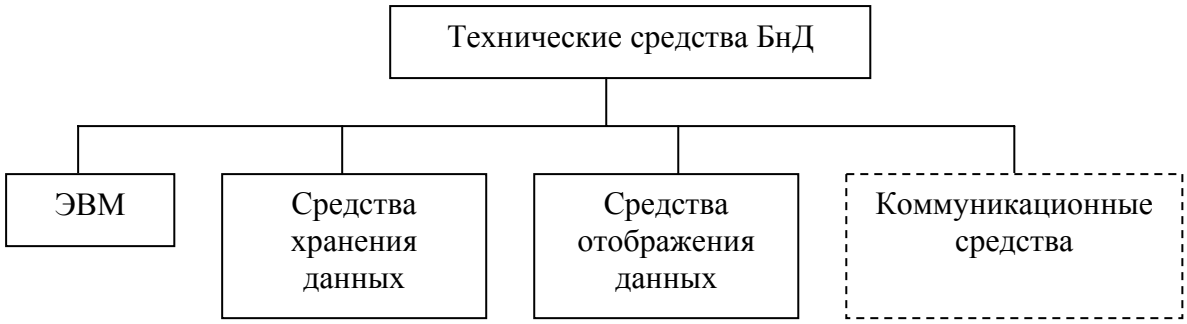


Рис. 1.7. Технические средства БнД

Тип используемых ЭВМ будет зависеть от масштаба создаваемой системы (рис. 1.8). В настоящее время в подавляющем большинстве случаев БнД реализуются в сетевой среде с использованием множества разнотипных ЭВМ, причем их состав постоянно меняется в процессе эксплуатации банка данных.

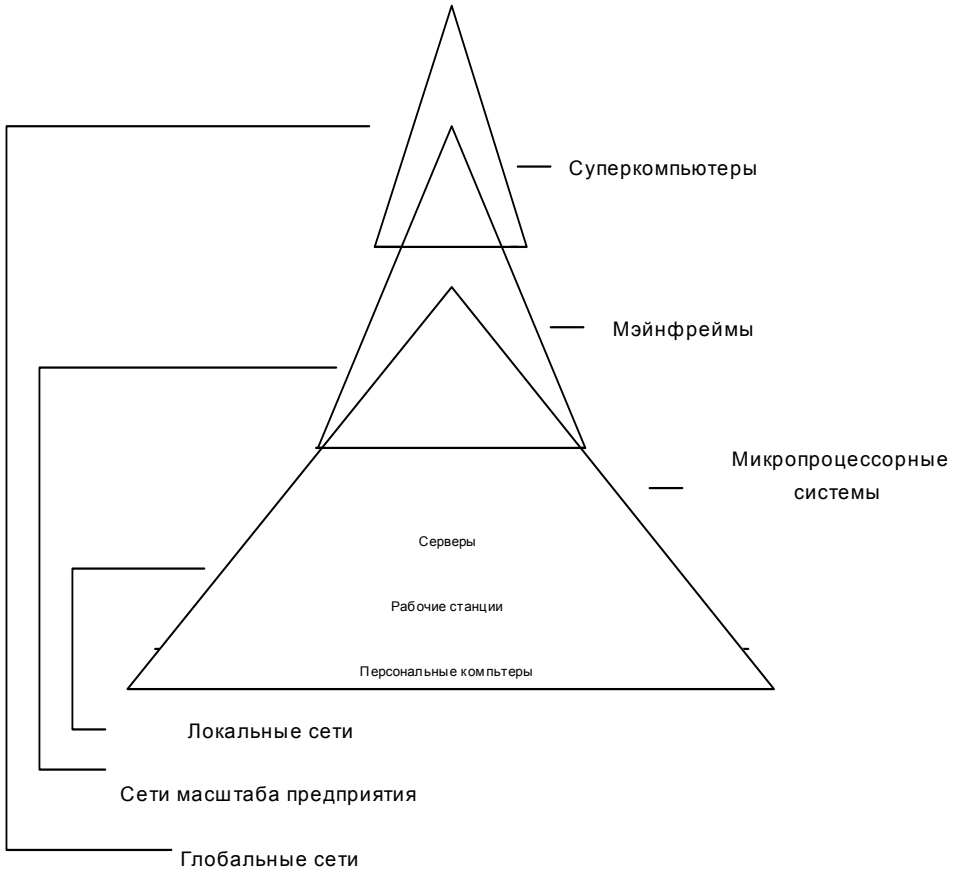


Рис. 1.8. Взаимосвязь используемых ЭВМ и технологии организации информационной системы

Технические средства БнД не ограничиваются только ЭВМ. Сюда входит весь комплекс технических средств хранения, отображения и передачи информации. Особую роль для обеспечения эффективного и надежного функционирования банка данных играют средства хранения информации. Память в БнД обычно организуется в виде многоуровневой системы. Необходимо обращать внимание не только на выбор запоминающих устройств для организации хранения данных, предназначенных для оперативного доступа к ним, но и хранения архивных данных.

В банках данных, также как и во всех других информационных системах, выполняются операции по вводу, хранению, обработке и выводу информации. При выполнении каждой из этих операций могут использоваться различные технологии и, как следствие, различные технические и программные средства для их поддержания.

Организационно-методические средства

Организационно-методические средства банка данных представляют собой различные инструкции, методические и регламентирующие материалы, предназначенные для пользователей разных категорий, взаимодействующих с банком данных. Это могут быть инструкции конечным пользователям по работе с базой данных, документы, определяющие права доступа и регламент работы; сюда же отнесем и методики проектирования баз.

Администраторы банка данных

Функционирование БнД невозможно без участия специалистов, обеспечивающих создание, функционирование и развитие БнД. Такая группа специалистов называется администратором банка данных (АБД). Эта группа специалистов считается составной частью банка данных.

В зависимости от сложности и объема банка данных, от особенностей используемой СУБД служба администрации банка данных может различаться как по составу и квалификации специалистов, так и по количеству работающих в этой службе.

Функции администратора банка данных. АБнД выполняют работы по созданию и обеспечению функционирования БнД на прояжениии всех этапов жизненного цикла системы. В составе группы администраторов банка данных можно выделить различные подгруппы в зависимости от выполняемых ими функций. Численность группы администрации, выполняемые ими функции будут в значительной степени зависеть от масштаба банка данных, специфики хранимой в нем информации, типа банка данных, особенностей используемых программных средств и некоторых других факторов.

В составе администрации БнД должны быть системные аналитики, проектировщики структур данных и внешнего по отношению к банку данных информационного обеспечения, проектировщики технологических процессов обработки данных, системные и прикладные программисты, операторы, специалисты по техническому обслуживанию. Если речь идет о коммерческом банке данных, то важную роль здесь будут играть специалисты по маркетингу.

Администраторы банка данных выполняют большой круг разнообразных функций. Дальше в учебном пособие мы будем подробно рассматривать некоторые из них. Сейчас же просто перечислим основные из этих функций:

1. Анализ предметной области: описание предметной области, выявление ограничений целостности, определение статуса информации, определение потребностей пользователей, определение статуса пользователей, определение соответствия «данные – пользователь», определение объемно-временных характеристик обработки данных.

2. Проектирование структуры базы данных: определение состава и структуры информационных единиц, составляющих базу данных, задание связей между ними, выбор методов упорядочения данных и методов доступа к информации, описание структуры БД на ЯОД.
3. Задание ограничений целостности при описании структуры базы данных и процедур обработки БД: задание ограничений целостности, присущих предметной области, определение ограничений целостности, вызванных структурой базы данных, разработка процедур обеспечения целостности БД при вводе и корректировке данных, обеспечение ограничений целостности при параллельной работе пользователей в многопользовательском режиме.
4. Первоначальная загрузка и ведение базы данных: разработка технологии первоначальной загрузки и ведения (изменения, добавления, удаления записей) БД, проектирование форм ввода, создание программных модулей, подготовка исходных данных, ввод и контроль ввода.
5. Защита данных от несанкционированного доступа.
   5.1. Обеспечение парольного входа в систему: регистрация пользователей, назначение и изменение паролей.
   5.2. Обеспечение защиты конкретных данных: определение прав доступа групп пользователей и отдельных пользователей, определение допустимых операций над данными для отдельных пользователей, выбор / создание программно-технологических средств защиты данных; шифрование информации с целью защиты данных от несанкционированного использования.
   5.3. Тестирование средств защиты данных.
   5.4. Фиксация попыток несанкционированного доступа к информации.
   5.5. Исследование возникающих случаев нарушения защиты данных и проведение мероприятий по их предотвращению.
6. Защита данных от разрушений. Одним из способов защиты от потери данных является резервирование. Используется как при физической порче файла, так и в случае, если в БД внесены нежелательные необратимые изменения.
7. Обеспечение восстановления БД: разработка программно-технологических средств восстановления БД, организация ведения системных журналов.
8. Анализ обращений пользователей к БД: сбор статистики обращений пользователей к БД, ее хранение и анализ (кто из пользователей, к какой информации, как часто обращался, какие выполнял операции, время выполнения запросов, анализ причин безуспешных (в т. ч. и аварийных) обращений к БД.
9. Анализ эффективности функционирования БнД и развитие системы: анализ показателей функционирования системы (время обработки, объем памяти, стоимостные показатели), реорганизация и реструктуризация баз данных, изменение состава баз данных, развитие программных и технических средств.
10. Работа с пользователями: сбор информации об изменениях в предметной области, об оценке пользователями работы БнД, определение регламента работы пользователей с БнД, обучение и консультирование пользователей.
11. Подготовка и поддержание системных программных средств: сбор и анализ информации о СУБД и других ПП, приобретение программных средств, их установка, проверка работоспособности, поддержание системных библиотек, развитие программных средств.
12. Организационно-методическая работа: выбор или создание методики проектирования БД, определение целей и направлений развития системы, планирование этапов развития БнД, разработка и выпуск организационно-методических материалов.

Связи администратора банка данных. В процессе своей деятельности администратор БнД взаимодействует с другими категориями пользователей банка данных, а также и с «внешними» специалистами, не являющимися пользователями БнД (рис.1.9).

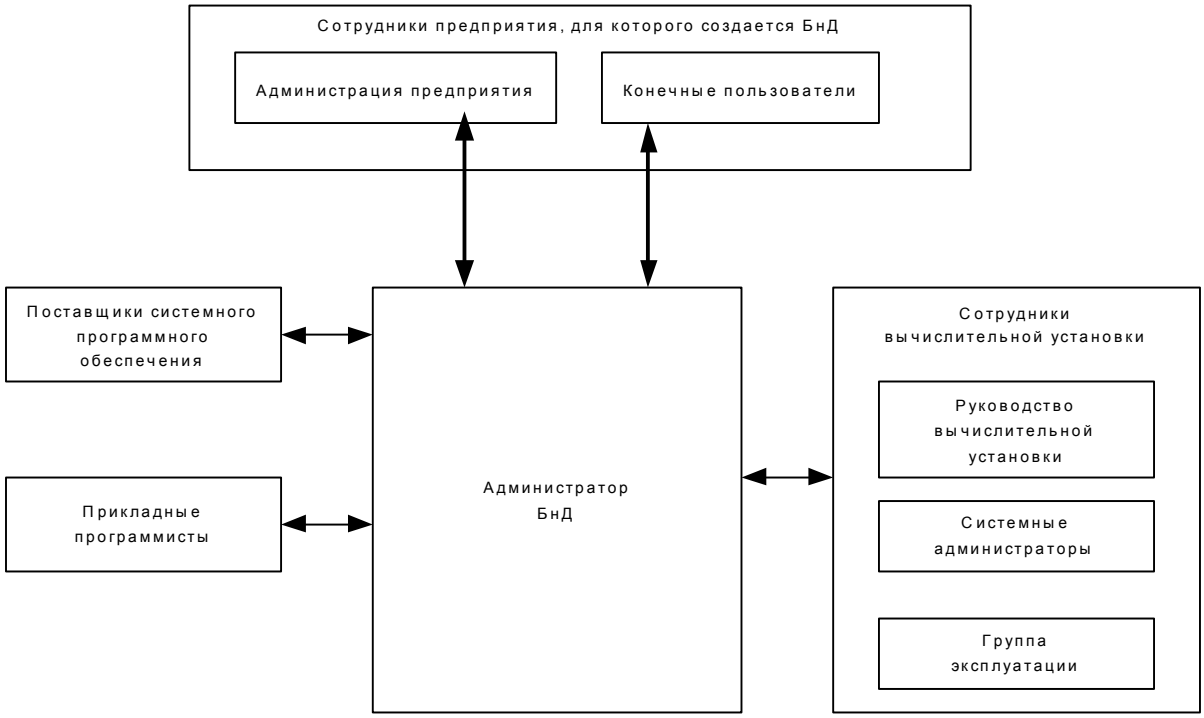


Рис. 1.9. Взаимодействие АБнД с другими категориями пользователей

Прежде всего, если банк данных создается для информационного обслуживания какого-либо предприятия или организации, то необходимы контакты с администрацией этой организации. Как указывалось выше, внедрение БнД приводит к большим изменениям не только системы обработки данных, но и всей системы управления организацией. Естественно, что такие большие проекты не могут быть выполнены без активного участия и поддержки руководителей организации. Руководство организации должно быть ознакомлено с возможностями, предоставляемыми БнД, проинформировано об их преимуществах и недостатках, а также проблемах, вызываемых созданием и функционированием БнД.

Т.к. база данных является динамическим информационным отображением предметной области, то желательно, чтобы администратор БнД в свою очередь был своевременно информирован о перспективах развития объекта, для которого создается информационная система.

Руководством организации и администратором БнД должны быть согласованы цели, основные направления и сроки создания БнД и его развития, очередность подключения пользователей.

Очень тесная связь у АБД на всех этапах жизненного цикла БнД наблюдается с конечными пользователями. Это взаимодействие начинается на начальных стадиях проектирования системы, когда изучаются потребности пользователей, уточняются особенности предметной области, и постоянно поддерживается как процессов проектирования, так и функционирования системы.

Следует отметить, что в последнее время наблюдается активное перераспределение функций между конечными пользователями и администраторами банка данных. Это, пре-

жде всего, связано с развитием языковых и программных средств, ориентированных на конечных пользователей. Сюда относятся простые и одновременно мощные языки запросов, а также средства автоматизации проектирования.

Если банк данных функционирует в составе какой-либо включающей его автоматизированной информационной системы (например, в АСУ), то АБД должен работать в контакте со специалистами по обработке данных в этой системе.

Администраторы БнД взаимодействуют и с внешними по отношению к нему группами специалистов и, прежде всего, поставщиками СУБД и ППП, администраторами других БнД.

БнД часто создаются специализированными проектными коллективами на основе договора на разработку информационной системы в целом или БнД как самостоятельного объекта проектирования. В этом случае служба администрации БнД должна создаваться как в организации-разработчике, так и в организации-заказчике.

Средства администратора современных СУБД. На эффективность работы БнД оказывают влияние множество внешних и внутренних факторов. Возрастание сложности и масштабов БнД, высокая «цена» неправильных или запоздалых решений по администрированию БД, высокие требования к квалификации специалистов делают актуальной задачу использования развитых средствах автоматизированного (или даже автоматического) администрирования БнД.

Средства администрирования включены в состав всех СУБД. Особенно развиты эти средства в корпоративных СУБД. Кроме того, появился целый класс специализированного программного обеспечения: средства DBA (DataBase Administration – администрирование базы данных).

Типичные функции средств DBA

Мониторинг работы БД, реакция на нештатные ситуации	Наблюдение за объектами БД, анализ, сопоставление характеристик	Оптимизация хранения данных, оптимизация работы сервера	Сопровождение БД, файлов, табличных пространств, откатных сегментов
Слежение за использованием ресурсов, выдача статистики	Планирование необходимых вычислительных мощностей	Анализ свободного пространства, устранение дефрагментации	Перенос таблицы на новое пространство, в другую СУБД, на другой компьютер
Обнаружение и исправление возникающих неполадок	Задание пороговых значений для слежения за нужными объектами	Наблюдение за параметрами, влияющими на производительность БнД	Перенос содержимого базы данных в другую СУБД

Взаимодействие компонентов БнД

На рис. 1.10 представлена упрощенная схема взаимодействия компонентов БнД в процессе создания и эксплуатации системы. Создание БД начинается с проектирования БД и ее описания на ЯОД (1). На этапе проектирования структуры БД могут использоваться как методики «ручного» проектирования, так и CASE-средства, автоматически генерирующие описания БД. Полученные описания должны быть введены в БнД и запомнены в соответствии с требованиями конкретной СУБД (2,3). После того, как описание базы данных сохранено, в базу данных могут вводиться данные (4). При этом СУБД использует метаинформацию, зафиксированную в словаре данных. Заполненная БД может использо-

ваться для извлечения из нее нужной пользователям информации (5). При формулировании запросов используется информация, содержащаяся в схемах и подсхемах. В результате выполнения запроса выходные данные в том или виде выдаются пользователю (6). Кроме собственно затребованных данных при выполнении операций с БнД часто выдается та или иная диагностическая информация (7). Для обеспечения надежности функционирования БнД необходимо выполнять соответствующие процедуры, в частности осуществлять журнализацию (8) выполняемых действий с БД, регулярно архивировать данные (9).

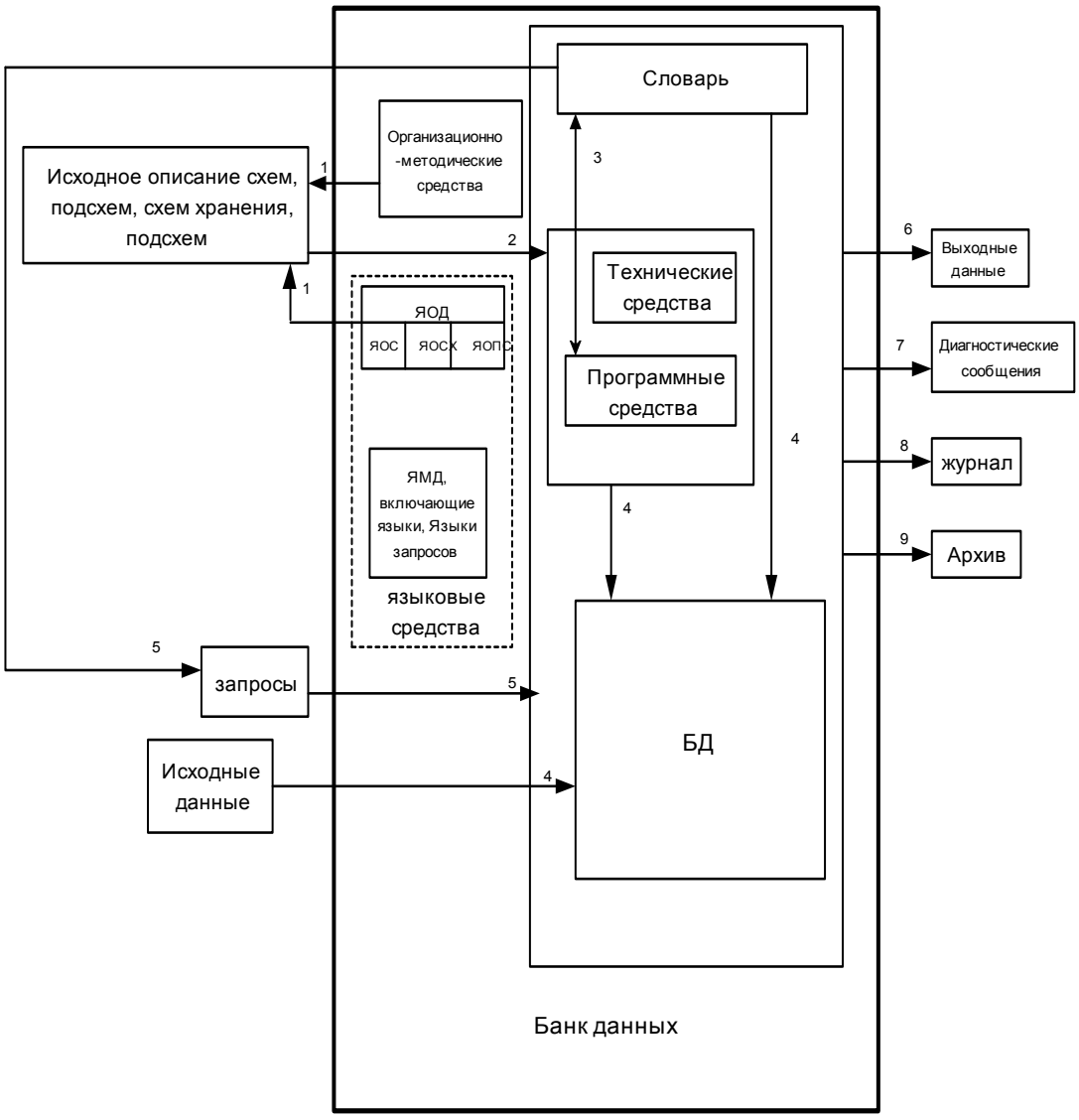


Рис. 1.10. Схема взаимодействия компонентов БнД

1.3. КЛАССИФИКАЦИЯ БАНКОВ ДАННЫХ

Банки данных являются сложными системами, и их классификация может быть произведена как для всего банка данных в целом, так и для каждой его компоненты отдельно; классификация для каждый компоненты может быть проведена по множеству разных признаков (рис. 1.11).

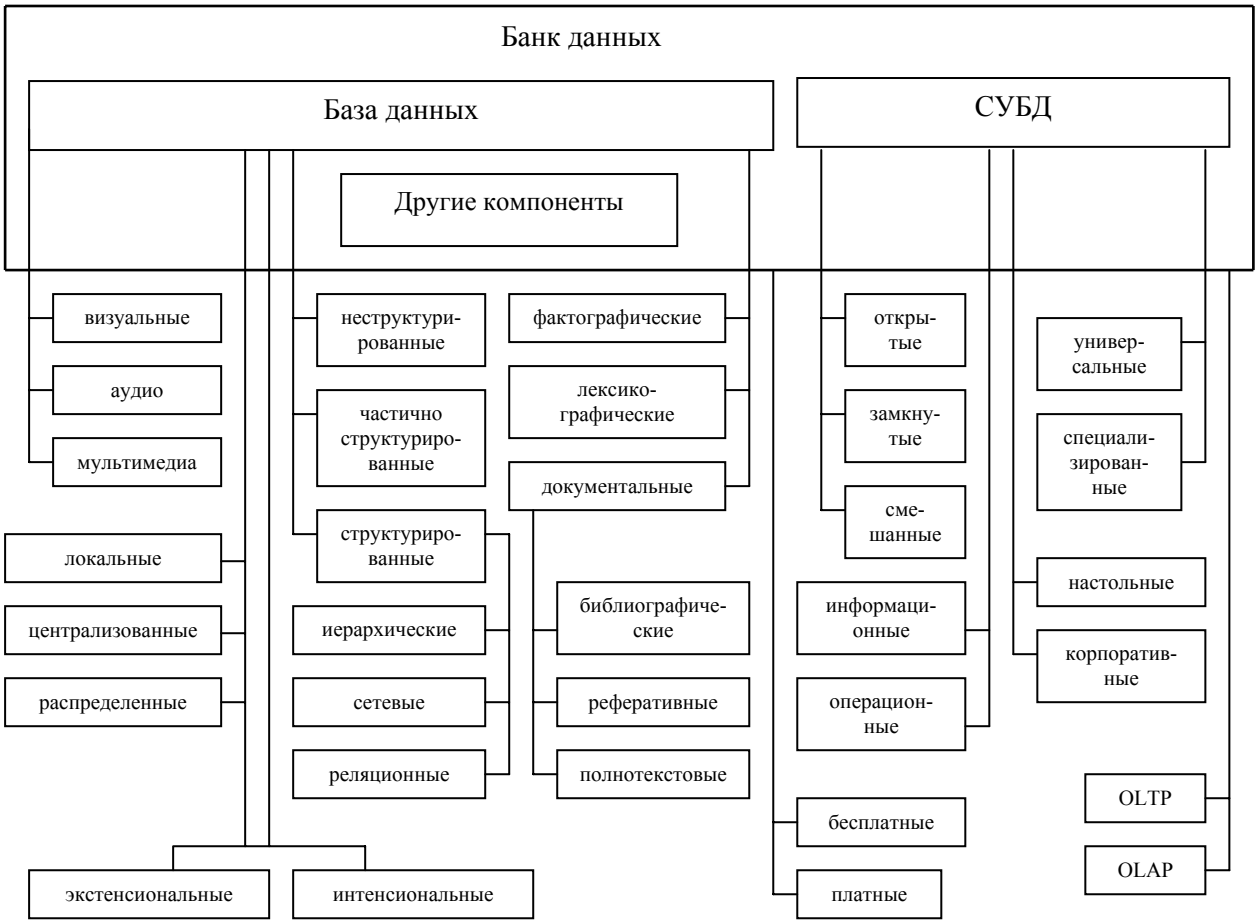


Рис. 1.11. Классификация БнД

Классификация баз данных

Центральной компонентой банка данных является база данных, и большинство классификационных признаков относятся именно к ней. ***По форме представления информации*** различают *визуальные* – и *аудио* системы, а также системы *мультимедиа*. Эта классификация показывает, в каком виде информация храниться в БД и выдается из баз данных пользователям: в виде изображения, звука или имеется возможность использования разных форм отображения информации. Понятие «изображение» здесь используется в широком смысле: это может быть символьный текст, неподвижное графическое изображение (рисунки, чертежи и т.п.), фотографии, географические карты, движущие изображения. Классификация способов представления информации являет собой самостоятельную проблему и здесь не рассматривается.

По характеру организации данных БД могут быть разделены на *неструктурированные*, *частично структурированные* и *структурированные*. Этот классификационный признак относится к информации, представленной в символьном виде. К неструктурированным БД могут быть отнесены базы, организованные в виде семантических сетей. Частично структурированными можно считать базы данных в виде обычного текста или гипертекстовые системы. Структурированные БД требуют предварительного проектирования и описания структуры БД. Только после этого базы данных такого типа могут быть заполнены данными.

Структурированные БД, в свою очередь, ***по типу используемой модели*** делятся на *иерархические, сетевые, реляционные, смешанные* и *мультимодельные*.

Классификация по типу модели распространяется не только на базы данных, но и на СУБД.

В структурированных БД обычно различают несколько уровней информационных единиц, входящих одна в другую. Число этих уровней может быть различным даже для систем, относящихся к одному и тому же классу. Большинство структурированных систем поддерживают уровень поля, записи и файла. Эти информационные единицы могут называться в разных системах по-разному, но суть остается одной и той же, а именно: *полю* соответствует наименьшая семантическая единица информации; совокупность полей или и иных, более сложных информационных единиц, если они допустимы в конкретной СУБД, образуют *запись*, а множество однотипных записей представляют *файл базы данных*. В последнее время большинство СУБД в явном виде поддерживают и уровень *базы данных*, как совокупности взаимосвязанных файлов БД.

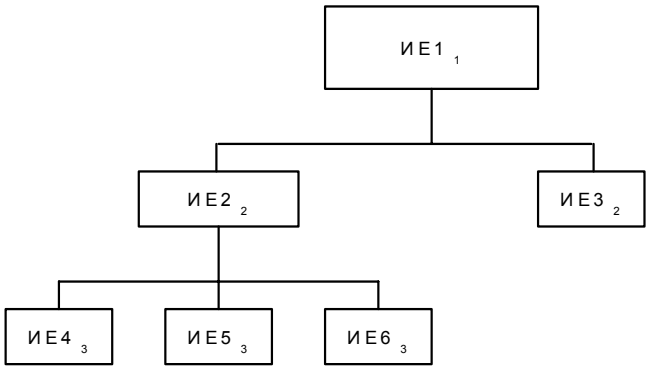


Рис. 1.12. Схема иерархической модели

В иерархических (рис. 1.12) и сетевых (рис. 1.13) моделях между информационными единицами (записями разных файлов) могут задаваться связи. Как видно из приведенных схем, графическое представление иерархической модели представляет собой граф типа «дерево». В такой модели имеется одна вершина – корень дерева, являющаяся входом в структуру. Каждая вершина, отличная от корня, может иметь только одну исходную вершину и, в общем случае, сколько угодно порожденных вершин.

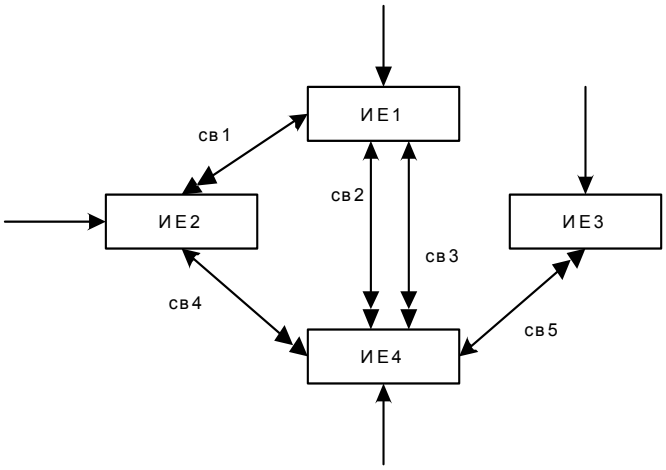


Рис. 1.13. Схема сетевой модели

Графическое представление сетевой модели представляет собой граф типа «сеть». Входом в такую структуру может являться любая вершина. Каждая вершина может иметь как несколько порожденных, так и несколько исходных вершин. Между парой вершин может быть объявлено несколько связей. Подавляющее большинство СУБД поддерживает простые сетевые структуры, т.е. между каждой парой типов записей поддерживается отношение 1 : М. Направление и характер связи в сетевых моделях не являетюся очевидными, как в случае иерархической модели, поэтому при изображении структуры БД направление связи должно быть указано.

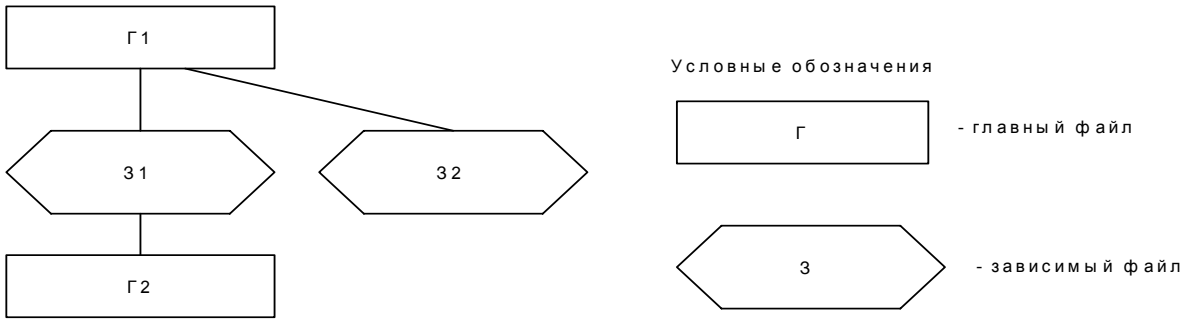


Рис. 1.14. Схема сетевой модели с разнотипными файлами

Кроме сетевых моделей с равноправными файлами существуют сетевые модели с разнотипными файлами. В таких моделях различают главные (основные) и зависимые файлы (рис. 1.14). Вход в структуру возможен только через главные файлы. Связываться между собой могут только записи разных типов.

Связи в иерархических и сетевых моделях описываются при проектировании БД. Чаще всего эти связи при хранении данных в БД передаются посредством адресных указателей. Иерархические и сетевые модели БД не накладывают ограничения на тип внутризаписной структуры. В принципе она может быть любой, как простой линейной (т.е. состоять только из простых полей, следующих в записи последовательно друг за другом), так и сложной иерархической, включающей в себя различные составные единицы информации (векторы, повторяющиеся группы и т.п.). Конкретные же СУБД накладывают ограничения на допустимые в них информационные единицы, характер связей между ними, порядок их расположения в записи, а также часто имеют и различные количественные ограничения.

Особое место среди структурированных систем занимают системы, построенные на использовании *инвертированных файлов*. Особенность организации данных в них состоит в том, что собственно хранимые данные и информация о связях между ними логически и физически отделены друг от друга. Основные данные в этих системах хранятся в файлах, записи которых могут иметь сложную структуру. Вся управляющая информация сосредоточена в ассоциаторе. Логическая связь между файлами устанавливается посредством компонента ассоциатора, называемого сетью связи. На рис. 1.15 схематически представлен принцип установления связей в таких системах. Реально, связи устанавливаются не непосредственно с элементами связи, как это изображено на рисунке, а через преобразователь адреса. В системах, построенных на инвертированных файлах, можно передавать связь типа М : М между записями файлов (что не позволяют никакие другие системы). Отделение ассоциативной информации от собственно хранимых данных позволяет изменять связи, не изменяя при этом самих файлов.

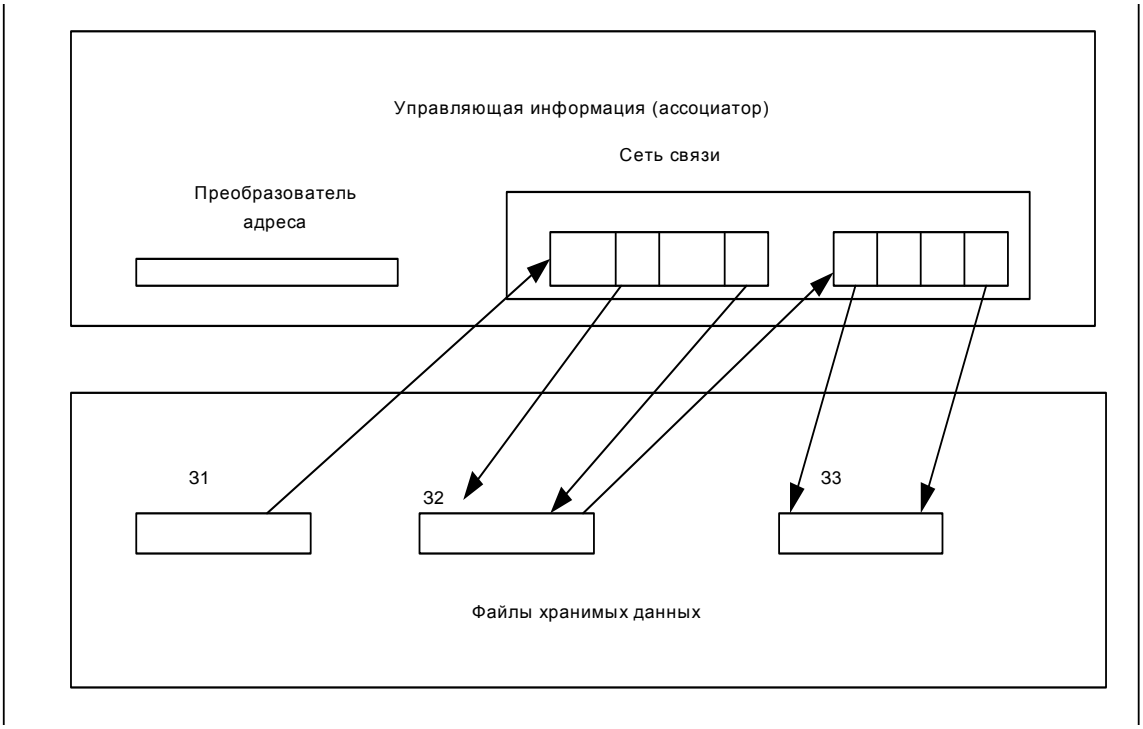


Рис. 1. 15. Схема организации данных в системах основанных на инвертированных файлах

В реляционных моделях (в отличие от иерархических и сетевых) связи между файлами БД определяются динамически в момент выполнения запроса. Эти связи определяются по равенству значений соответствующих полей (полей связи), содержащихся в каждом из связанных файлов.

Другой отличительной чертой реляционных моделей является ограничение на внутризаписную структуру: записи имеют линейную структуру и могут содержать только простые поля (рис. 1.16).

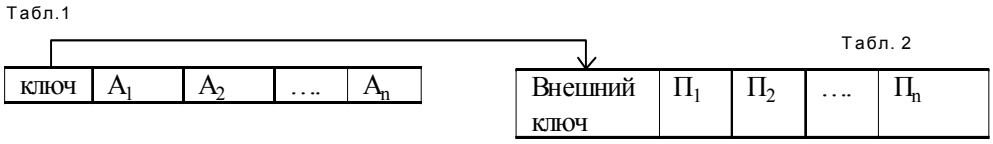


Рис. 1.16. Схема реляционной модели

Эти отличительные особенности играют решающую роль при проектировании структуры БД.

Восьмидесятые годы были временем интенсивного развития реляционных систем. В 1992 году [3] уровень продаж реляционных СУБД впервые превысил уровень продаж нереляционных СУБД. Но до 90% данных предприятий хранилось к этому моменту в нереляционных базах данных на мэйнфреймах.

[3] Первая коммерческая реляционная СУБД была выпущена фирмой Oracle в 1979 году.

По ***типу хранимой информации*** БД делятся на *документальные, фактографические* и *лексикографические*. Среди документальных баз различают *библиографические, реферативные* и *полнотекстовые*. К лексикографическим базам данных относятся различные словари (классификаторы, многоязычные словари, словари основ слов и т. п.).

В системах фактографического типа в БД хранится информация об интересующих пользователя объектах предметной области в виде «фактов» (например, биографические данные о сотрудниках, данные о выпуске продукции производителями и п. т.); в ответ на запрос пользователя выдается требуемая ему информация об интересующем его объекте / объектах или сообщение о том, что искомая информация отсутствует в БД.

В документальных БД единицей хранения является какой-либо документ (например, текст закона или статьи), и пользователю в ответ на его запрос выдается либо ссылка на документ, либо сам документ, в котором он может найти интересующую его информацию.

БД документального типа могут быть организованы по-разному: без хранения и с хранением самого исходного документа на машинных носителях. К системам первого типа можно отнести библиографические и реферативные БД, а также БД-указатели, «отсылающие» к источнику информации. Системы, в которых предусмотрено хранение полного текста документа, называются *полнотекстовыми.*

В системах документального типа целью поиска может быть не только какая-то информация, хранящаяся в документах, но и сами документы. Так, возможны запросы типа «сколько документов было создано за определенный период времени» и т.п. Часто в критерий поиска в качестве признаков включаются «дата принятия документа», «кем принят» и другие «выходные данные» документов.

Специфической разновидностью баз данных являются *базы данных форм документов*. Они обладают некоторыми чертами документальных систем (ищется документ, а не информация о конкретном объекте, форма документа имеет название, по которому обычно и осуществляется ее поиск), так и специфическими особенностями (документ ищется не с целью извлечь из него информацию, а с целью использования его в качестве «шаблона»).

В последние годы активно развивается объектно-ориентированный подход к созданию информационных систем. Объектные базы данных организованы как объекты и ссылки к объектам. Объект представляет собой данные и правила, которые оперируют этими данными. Объект включает метод, который является частью определения объекта и запоминается вместе с объектом. В объектных базах данных данные запоминаются как объекты, классифицированные по типам классов и организованные в иерархическое семейство классов. Класс – коллекция объектов с одинаковыми свойствами. Объекты принадлежат классу. Классы организованы в иерархии.

По ***характеру организации хранения*** данных *и обращения* к ним различают *локальные* (персональные), *общие* (интегрированные, централизованные) и *распределенные* базы данных (рис. 1.17).

Персональная база данных – это база данных, предназначенная для локального использования одним пользователем. Локальные БД могут создаваться каждым пользователем самостоятельно, а могут извлекаться из общей БД.

Интегрированные и распределенные БД предполагают возможность одновременного обращения нескольких пользователей к одной и той же информации (многопользовательский, параллельный режим доступа). Это привносит специфические проблемы при их проектировании и в процессе эксплуатации БнД. *Распределенные БД* кроме этого имеют характерные особенности, связанные с тем, что физически разные части БД могут быть расположены на разных ЭВМ, а логически, с точки зрения пользователя, они должны представлять собой единое целое.

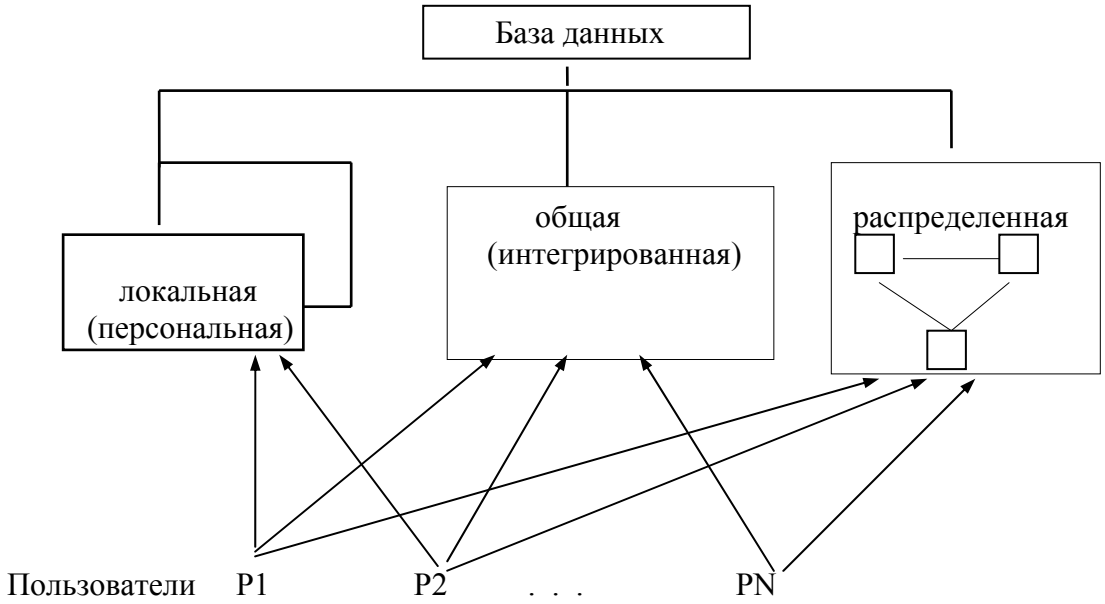


Рис. 1.17. Классификация БнД по характеру хранения данных и обращения к ним

Технологии, которые, на первый взгляд, вроде бы находятся на разных концах спектра (локальная и распределенная обработка), на самом деле очень близки и различаются практически тем, как поддерживается связь между отдельными частями БД. В случае локальных систем поддержание этой связи не является централизованной, а в случае распределенных БнД – должна поддерживаться СУБД. Технологией, позволяющей совмещать идеи локальной работы и централизованного поддержания единой БД, является технология тиражирования, при которой средства СУБД позволяют тиражировать отдельные части общей БД, локально использовать их, а потом «согласовывать» отдельные фрагменты БД в рамках единой базы данных.

Концепции централизованной и распределенной обработки данных также не так сильно различаются между собой, как кажется на первый взгляд. Так называемые клиент-серверные системы с «тонким клиентом» очень близки к централизованным базам данных.

Банк данных является сложной человеко-машинной системой, и распределяться по узлам сети могут не только БД, но и другие компоненты БнД. Причем сама БД при этом может быть и не распределенной (например, при обеспечении многопользовательского доступа к централизованной БД в сети). Поэтому будем различать два понятия: распределенные БД и распределенные БнД. При этом под *распределенным БнД* будем понимать банк данных, в котором распределена хотя бы одна любая из его компонент.

В [15] различают *экстенсиональные* (ЭБД) и *интенсиональные* базы данных. Интенсиональная база данных строится с помощью правил, определяющих ее содержание, а не с помощью явного хранения данных в БД, как в экстенсиональных БД.

Например, пусть имеется ЭБД, содержащая таблицу ЛИЧНОСТЬ (PERSON), которая содержит сведения о личности и среди полей которой есть поля ФАМИЛИЯ _ ИМЯ _ ОТЧЕСТВО (FIO), ПОЛ (SEX). Мы можем построить в этой ЭБД вторую таблицу РОДИТЕЛЬ (PARENT), которая содержит поля ФАМИЛИЯ _ ИМЯ _ ОТЧЕСТВО родителя (FIO) и ИМЯ_РЕБЕНКА (CHILD). С помощью правил мы можем определить, например, отношение ОТЕЦ (FATHER), просто указав, что отец – это родитель, у которого пол – мужской. На ПРОЛОГе это отношение можно определить следующим образом:

father(X, Y): = person (X, male), parent(X, Y).

Если выполнить это правило, то получится отношение, которое содержит подмножество кортежей таблицы PARENT, таких, для которых верно указанное условие. Пользователю эти данные выдадутся в виде обычного отношения.

Данное определение ЭБД и ИБД можно расширить и на другой (не реляционный) тип БД, и на другой способ задания правил. В более общем виде можно сказать, что информацию можно передать и в виде данных, и в виде программ (строго говоря, программы тоже являются данными, но в русском языке нет подходящего термина, который можно было бы здесь употребить вместо слова «данные»).

БД классифицируются ***по объему***. Особое место здесь занимают так называемые *очень большие базы данных*. Это вызвано тем, что для больших баз данных по-иному стоят вопросы обеспечения эффективности хранения информации и обеспечения ее обработки.

Классификация СУБД

Рассмотрим теперь ряд классификационных признаков, относящихся к СУБД. ***По языкам общения*** СУБД делятся на *открытые*, *замкнутые* и *смешанные*. *Открытые системы* – это системы, в которых для обращения к базам данных используются универсальные языки программирования. *Замкнутые системы* имеют собственные языки общения с пользователями БнД.

По числу уровней в архитектуре различают одноуровневые, двухуровневые, трехуровневые системы. В принципе возможно выделение и большего числа уровней. Под архитектурным уровнем СУБД понимают функциональный компонент, механизмы которого служат для поддержки некоторого уровня абстракции данных (логический и физический уровень, а также «взгляд» пользователя – внешний уровень).

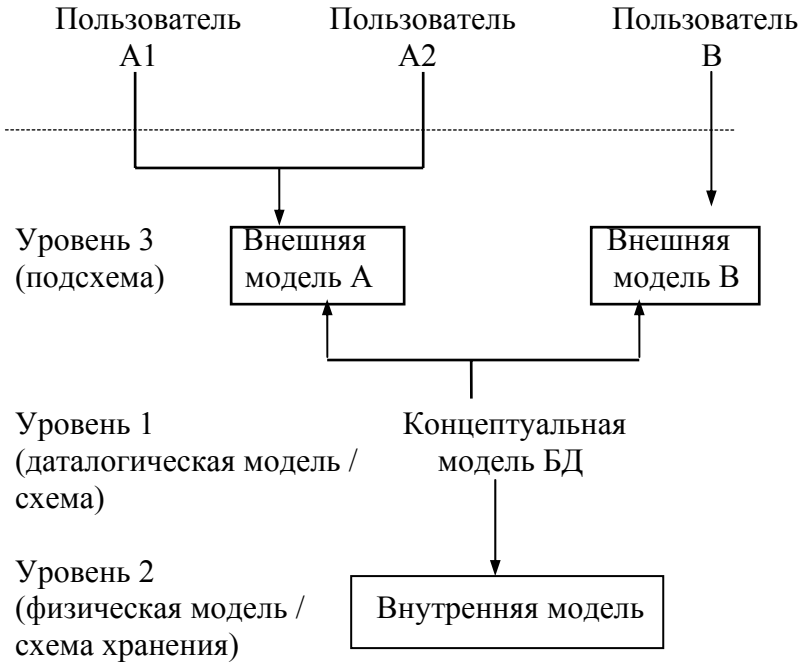


Рис 1.18. Классификация СУБД по числу уровней в архитектуре (пример трехуровневой архитектуры)

На рис. 1.18 сделана попытка совместить терминологию, встречающуюся в разных литературных источниках. В литературе широко используются понятия «внешняя», «концептуальная» и «внутренняя» модель / уровень (см., например, [9], и др.), а также «логический» и «физический»[4] уровень [15], а кроме того «внешняя схема», «подсхема», «схема хранения», просто «схема» и проч. Понятие схема с тем или иным уточнением обычно относится к описанию соответствующего уровня описания данных.

Нумерация уровней на рисунке условна, но тем нее менее отражает их значимость (внутренняя модель может быть построена только на основе концептуальной; эти два уровня могут быть совмещены, но поддерживаются СУБД всегда; внешний уровень в архитектуре СУБД может отсутствовать).

По выполняемым функциям СУБД делятся на *информационные* и *операционные*. Информационные СУБД позволяют организовать хранение информации и доступ к ней. Для выполнения более сложной обработки необходимо писать специальные программы. Операционные СУБД выполняют достаточно сложную обработку, например, автоматически позволяют получать агрегированные показатели, не хранящиеся непосредственно в базе данных, могут изменять алгоритмы обработки и т.д.

По сфере возможного применения различают *универсальные* и *специализированные*, обычно проблемно-ориентированные СУБД.

Системы управления базами данных поддерживают разные типы данных. Набор типов данных, допустимых в разных СУБД, различен. Кроме того, ряд СУБД позволяет разработчику добавлять новые типы данных и новые операции над этими данными. Такие системы называются *расширяемыми* системами баз данных (РСБД).

Дальнейшим развитием концепции РСБД являются *системы объектно-ориентированных баз данных (СООБД)*, обладающие достаточно мощными выразительными возможностями, чтобы непосредственно моделировать сложные объекты.

По «мощности» СУБД делятся на *«настольные»* и «*корпоративные»*. Характерными чертами настольных СУБД являются сравнительно невысокие требования к техническим средствам, ориентация на конечного пользователя, низкая стоимость.

Корпоративные СУБД обеспечивают работу в распределенной среде, высокую производительность, поддержку коллективной работы при проектировании систем, имеют развитые средства администрирования и более широкие возможности поддержания целостности.

В связи с выше перечисленными чертами корпоративных СУБД очевидно, что эти системы сложны, дороги, требуют значительных вычислительных ресурсов.

[4] Во многих современных Case-средствах концептуальной моделью называется ER-модель предметной области, а физической – модель, поддерживаемая конкретной СУБД. Если первое еще можно считать удачным использованием термина (так как ER-модель действительно отражает общую «концепцию» системы, то второе – крайне неудачно, так как ни о какой «физике» речь здесь не идет.

Таблица 1.1

Сравнение «настольных» и «корпоративных» СУБД

Критерий	Настольные	Корпоративные
Простота использования	+	
Стоимость программного обеспечения	+	
Стоимость эксплуатации	+	
Функциональные возможности, в т.ч.: • возможности администрирования • возможности работы с Интернет / Интранет и др.		+
Надежность функционирования		+
Поддерживаемые объемы данных		+
Быстродействие		+
Возможности масштабирования		+
Работа в гетерогенной среде		+

Системы обоих классов интенсивно развиваются, причем некоторые тенденции развития присущи каждому из этих классов. Прежде всего, это использование высокоуровневых средств разработки приложений (что раньше было присуще, в основном, настольным системам), рост производительности и функциональных возможностей, работа в локальных и глобальных сетях и др.

Наиболее известными из корпоративных СУБД являются Oracle, Informix, Sybase, MS SQL Server, Progress и некоторые другие.

Наблюдается связь между классом СУБД и используемой операционной системой. Системы под UNIX позиционируются как корпоративные распределенные системы. Сейчас в этот сектор «пробивается» Windows NT и заменяющая ее Windows 2000.

По ориентации на преобладающую категорию пользователей можно выделить СУБД *для разработчиков* и *для конечных пользователей*. Системы, относящиеся к первому классу, должны иметь качественные компиляторы и позволять создавать «отчуждаемые» программные продукты, обладать развитыми средствами отладки, включать средства документирования проекта и обладать другими возможностями, позволяющими создавать эффективные сложные системы. Основными требованиями, предъявляемыми к системам, ориентированным на конечного пользователя, являются: удобство интерфейса, высокий уровень языковых средств, наличие интеллектуальных модулей подсказок, повышенная защита от непреднамеренных ошибок («защита от дурака») и т.п.

Классификационные группировки, относящиеся к БнД в целом

Следующая группа признаков классификации связана с **банком данных в целом**. ***По условиям предоставления услуг*** различают *бесплатные* и *платные* банки данных. Платные БД в свою очередь делятся на *бесприбыльные* и *коммерческие*. Бесприбыльные банки данных функционируют на принципе самоокупаемости и не ставят своей целью получение прибыли. Это обычно БнД социально значимой информации, имеющей широкий круг пользователей, или научной, библиотечной информации. Основной целью создания коммерческих банков данных является получение прибыли от информационной деятельности.

Информационные системы различаются ***по характеру преобладающей обработки информации***. В одних в основном реализуется большое число достаточно простых запросов (такие системы получили название *OLTP* (On-Line Transaction Processing) – *системы*

оперативной обработки транзакций). В других, напротив, требуется сложная аналитическая обработка данных (для такого класса систем стал использоваться термин *OLAP* (On-line Analytical Processing)).

Термин OLAP является сравнительно новым и в разных литературных источниках трактуется иногда по-разному. Этот термин часто отождествляют с поддержкой принятия решений (DSS (Decision Support Systems) – системы поддержки принятия решения). А в качестве синонима для последнего термина используют Data Warehousing – хранилища (склады) данных, понимая под этим набор организационных решений, программных и аппаратных средств для обеспечения аналитиков информацией на основе данных из систем обработки транзакций нижнего уровня и других источников.

«Склады данных» позволяют обрабатывать данные, накопленные за длительные периоды времени. Эти данные являются разнородными (и не обязательно структурированными). Для «складов данных» присущ многомерный характер запросов. Огромные объемы данных, сложность структуры как данных, так и запросов требует использования специальных методов доступа к информации.

В других источниках понятие Системы Поддержки Принятия Решений (СППР) считается более широким. Хранилища данных и средства оперативной аналитической обработки могут служить одними из компонентов архитектуры СППР.

Иногда различают «OLAP в узком смысле» – это системы, которые обеспечивают только выборку данных в различных разрезах, и «OLAP в широком смысле», или просто OLAP, включающие в себя:

- поддержку нескольких пользователей, редактирующих БД;
- функции моделирования, в том числе вычислительные механизмы получения производных результатов, а также агрегирования и объединения данных;
- прогнозирование, выявление тенденций и статистический анализ.

Естественно, что каждый из этих типов ИС требует специфической организации данных, а также специальных программных средств, обеспечивающих эффективное выполнение стоящих задач.

Таблица 1.2

Сравнение OLTP и OLAP

Характеристика	OLTP	OLAP
Преобладающие операции	Ввод данных, поиск	Анализ данных
Характер запросов	Много простых транзакций	Сложные транзакции
Хранимые данные	Оперативные, детализированные	Охватывающие большой период времени, агрегированные
Вид деятельности	Оперативная, тактическая	Аналитическая, стратегическая
Тип данных	Структурированные	Разнотипные

Для обеспечения быстрой обработки данных при их анализе используются разнообразные приемы. Одним из них является организация данных в виде так называемых многомерных БД (MDD). Информация в MDD хранится не в виде индексированных записей в таблицах, а в форме логически упорядоченных массивов. Единой общепризнанной многомерной модели хранения данных не существует. В MDD отсутствует стандартизованный метод доступа к данным, и они могут отвечать требованиям специфической аналитической обработки данных.

Хранилища данных могут быть разбиты на два типа: корпоративные хранилища данных (enterprise data warehouses) и киоски данных (data marts).

Корпоративные хранилища данных содержат информацию, относящуюся ко всей корпорации и собранную из множества оперативных источников для консолидированного анализа. Обычно такие хранилища охватывают целый ряд аспектов деятельности корпорации и используются для принятия как тактических, так и стратегических решений.

Киоски данных содержат подмножество корпоративных данных и строятся для отделов или подразделений внутри организации. Киоски данных часто строятся силами самого отдела и охватывают конкретный аспект, интересующий сотрудников данного отдела. Киоск данных может получать данные из корпоративного хранилища (зависимый киоск), или, что более распространено, данные могут поступать непосредственно из оперативных источников (независимый киоск).

Киоски и хранилища данных строятся по сходным принципам и используют практически одни и те же технологии.

По степени доступности БнД делятся на *общедоступные* и *с ограниченным кругом пользователей.*

По охвату БнД могут классифицироваться в свою очередь в разных «разрезах»:

- *территориальный:*
  - *всемирный*
  - *...*
  - *страна*
  - *...*
  - *город*
  - *...*
- *временной;*
- *ведомственный;*
- *проблемный (тематический).*

Территориальный и ведомственный признаки классификации могут относиться не только к информации, хранящейся БД, но и к кругу обслуживаемых пользователей.

По характеру взаимодействия с пользователями (кто инициализирует действия) БнД делятся на:

- активные БнД;
- пассивные БнД.

В пассивных БнД ведущая роль принадлежит пользователю. В активных – система может самостоятельно менять поведение. В последнее время термин «активная база данных» стал часто использоваться для систем, использующих тригерры.

По форме собственности БнД делятся на:

- государственные;
- негосударственные:
  - частные;
  - групповые;
  - личные.

В литературе встречаются и другие аспекты классификации банков данных, но названные являются наиболее значимыми.

1.4. УРОВНИ МОДЕЛЕЙ И ЭТАПЫ ПРОЕКТИРОВАНИЯ БД

Уровни моделей

В базе данных отражается информация об определенной предметной области. *Предметной областью* называется часть реального мира, представляющая интерес для данного исследования.

В автоматизированных информационных системах отражение предметной области обеспечивается посредством информационной модели. Мы будем рассматривать далее вопросы проектирования баз данных для СУБД, поддерживающих структурированные модели данных. В зависимости от аспекта рассмотрения (уровня абстракции) различают модели данных нескольких уровней. Число реально выделенных и самостоятельно поддерживаемых уровней моделей будет зависеть от особенностей СУБД.

Чаще всего выделяют три уровня моделей: логический, физический и внешний.

Даталогическая (datalogical) модель (ДЛМ) базы данных является моделью логического уровня и представляет собой отображение логических связей между элементами данных безотносительно к среде хранения. Эта модель строится в терминах информационных единиц, допустимых в той конкретной СУБД, в среде которой мы проектируем базу данных. Этап создания ДЛМ называется *даталогическим проектированием*. Описание логической структуры базы данных на языке СУБД называется *схемой.*

Для привязки даталогической модели к среде хранения используется модель данных физического уровня (для краткости часто называемая *физической моделью*). Эта модель определяет используемые запоминающие устройства, способы физической организации данных в среде хранения. Модель физического уровня также строится с учетом возможностей, предоставляемых СУБД. Описание физической структуры базы данных называется *схемой хранения*. Соответствующий этап проектирования БД называется физическим проектированием. СУБД обладают разными возможностями по физической организации данных, в связи с чем сложность и трудоемкость физического проектирования, набор выполняемых шагов различаются для конкретных систем. К числу работ, выполняемых на этапе физического проектирования, относятся: выбор типа носителя, способа организации данных, методов доступа, определение размера физического блока, управление размещением данных на внешнем носителе, управление свободной памятью, определение целесообразности сжатия данных и используемых методов сжатия, оценка физической модели данных. К физическому проектированию относятся и проблемы, связанные с буферизацией.

Независимо от того, поддерживаются ими в явном виде отдельно модели логического и физического уровня, с точки зрения методологии все равно можно выделить эти уровни моделей и соответствующие им этапы проектирования баз данных.

В некоторых СУБД, помимо описания общей логической структуры базы данных, имеется возможность описать логическую структуру БД с точки зрения конкретного пользователя. Такая модель называется *внешней*, а ее описание называется *подсхемой*. Если СУБД «поддерживает» схему, схему хранения и подсхему, то она является СУБД с трехуровневой архитектурой.

Внешняя модель не всегда является точным подмножеством схемы. Некоторые СУБД допускают различия в типах данных, определенных в схеме и подсхеме, и обеспечивают их преобразование, позволяют задавать различный логический порядок следования элементов в схеме и подсхеме, обеспечивают введение в подсхему виртуальных полей и т.д. Если определена подсхема, то пользователь имеет доступ только к тем данным, которые отражены в соответствующей подсхеме, что является одним из способов защиты информации от несанкционированного доступа.

В подсхемах часто задается не только логическая структура части базы данных с точки зрения конкретного пользователя (приложения), но и допустимые режимы обработки в рамках этой подсхемы, что служит дополнительным механизмом защиты информации от разрушения.

Использование аппарата подсхем облегчает работу пользователя, так как он должен знать структуру не всей базы данных, а только той ее части, которая имеет непосредственное отношение к нему.

В тех случаях, когда СУБД в явном виде не поддерживает подсхемы, перечисленные функции могут выполнять другие компоненты системы. Близким к понятию подсхемы является понятие view (взгляд), которое в настоящее время широко используется в англоязычной литературе по реляционным СУБД.

Выше мы говорили о трех уровнях моделей, которые поддерживаются СУБД. Но для того, чтобы спроектировать структуру базы данных, необходима исходная информация о предметной области. Желательно, чтобы эта информация была представлена в формализованном виде. Такое формализованное описание предметной области будем называть *инфологической (infological) моделью предметной области (ИЛМ)*[5] *или концептуальной моделью (КМ)*. Информация, требуемая для проектирования БД, мало зависит от особенностей СУБД. Более того, для проектирования ИС с «небанковской» организацией (но использующей структурированное представление данных) обычно требуется та же исходная информация. Поэтому *концептуальная схема* представляет собой описание предметной области, выполненное без жесткой ориентации на используемые в дальнейшем программные и технические средства. Концептуальная схема должна отражать специфику предметной области, а не структуру БД. Иногда в концептуальную схему добавляют информацию, отображающую чисто языковые характеристики, такие как наличие синонимов, длина реквизитов и др. Это, скорее всего, вызвано следующими основными причинами: 1) нежеланием вводить еще один уровень моделей, 2) трудностью отделения языковых проблем от других, так как анализируемая предметная область обычно представлена в какой-либо знаковой системе, и анализу обычно подвергается именно это представление, а не непосредственно сама ПО.

Начальным шагом проектирования ИС является построение инфологической модели предметной области. Предварительная инфологическая модель строится еще на предпроектной стадии и затем уточняется на более поздних стадиях проектирования. Затем на ее основе строится даталогическая модель. Физическая и внешняя модель после этого могут строиться в любой последовательности по отношению друг к другу, в том числе и параллельно. На рис. 1.19 изображена взаимосвязь этапов проектирования БД. Как видно из рисунка, при проектировании БД возможен возврат на предыдущие уровни. При этом возможны два вида возвратов: первый тип обусловлен необходимостью пересмотра результата проектирования (например, для улучшения полученных характеристик, «обхода» ограничений и т.п.), второй тип вызван необходимостью уточнения предыдущей модели (обычно – инфологической) с целью получения дополнительной информации для проектирования или при выявлении противоречий в модели.

[5] В предыдущем издании учебника [о] для этого уровня моделирования использовался термин «инфологическое». Последнее время в литературных источниках этот термин используется редко. В данном учебном пособии термины «инфологическое» и «концептуальное» моделирование используются как синонимы.

Взаимосвязь этапов проектирования БД

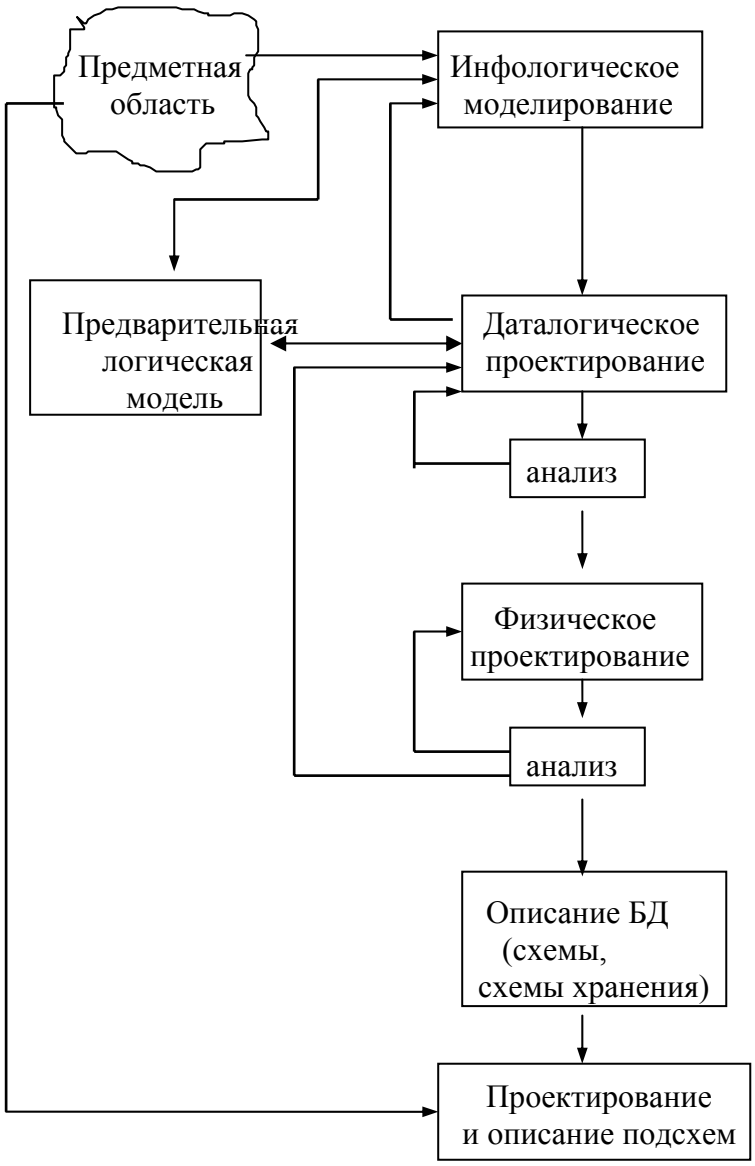


Рис. 1.19. Взаимосвязь этапов проектирования БД

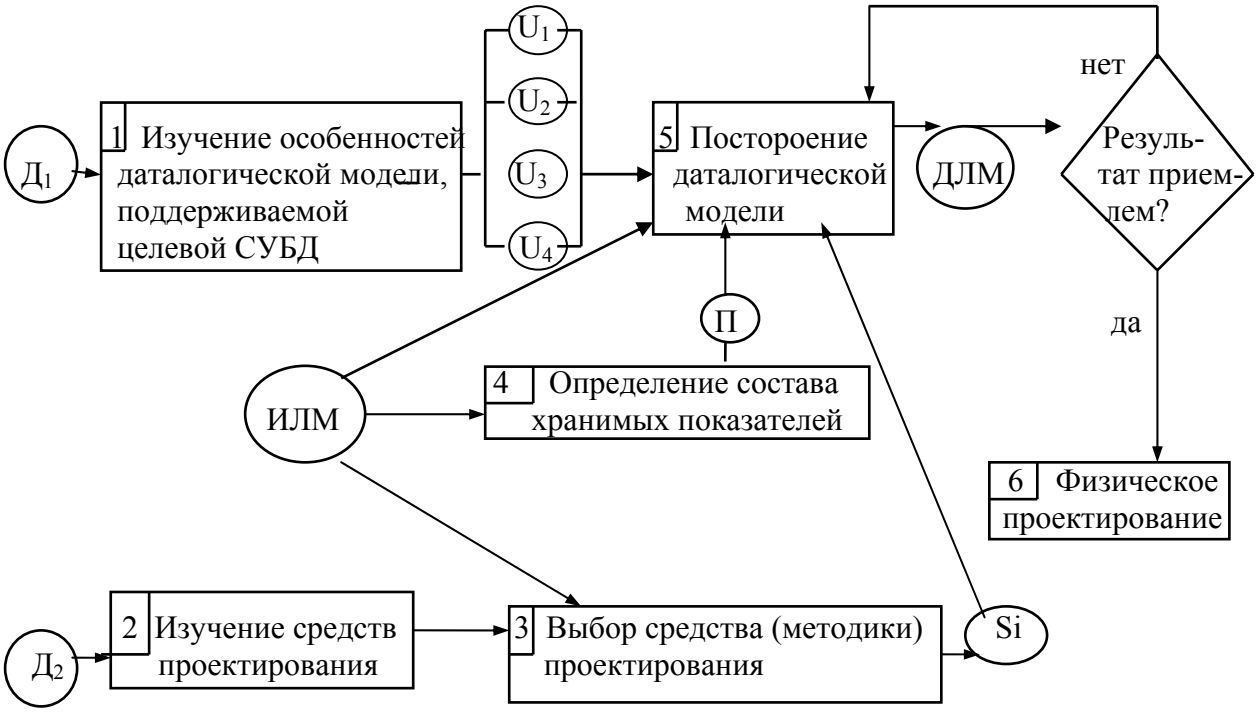
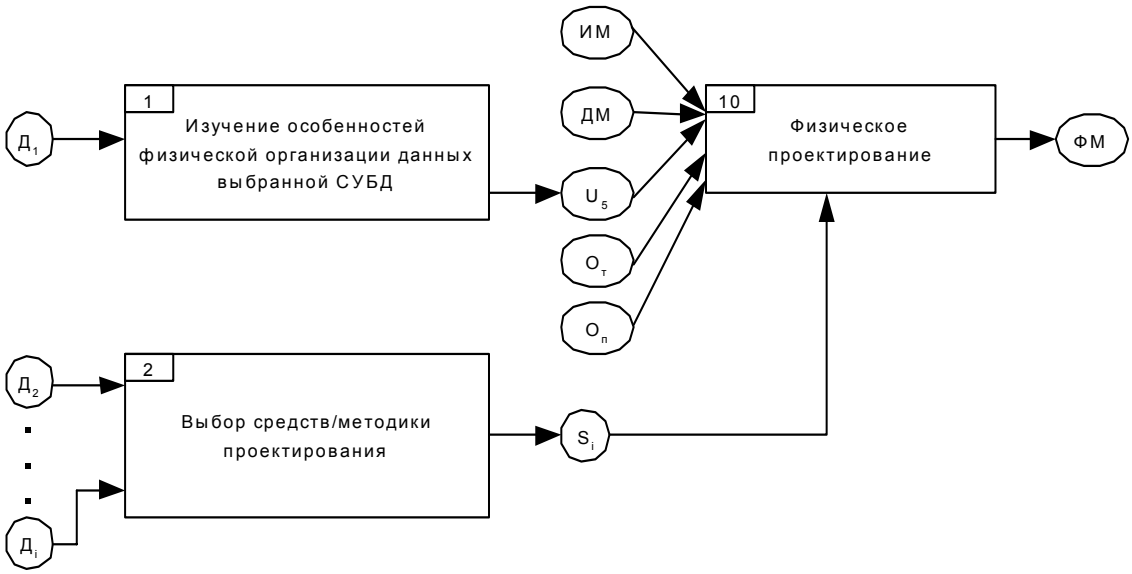


Рис. 1.20. Технологическая сеть проектирования для этапа даталогического проектирования

Д1 – документация по СУБД;
Д2 – документация по средствам проектирования;
U1 – набор допустимых даталогических конструкций;
U2 – операторы ЯМД;
U3 – ограничения, налагаемые СУБД на ДЛМ;
U4 – возможности физической организации данных;
ДЛМ – даталогическая модель;
ИЛМ – инфологическая модель;
П – перечень хранимых показателей;
Si – выбранное средство проектирования.

На рис. 1.20 и 1.21 изображены укрупненные технологические сети проектирования для этапов даталогического и физического проектирования 6. Как видно из этих рисунков, результат предыдущего этапа проектирования используется на входе следующего этапа.

[6] Следует обратить внимание на различие терминологии, используемой как в литературных источниках, так и в конкретных CASE-системах. Так, во многих CASE-системах [...] ER-модель предметной области называется концептуальной схемой, а представление логической структуры целевой базы данных – физической моделью.



$Д_1$ – документация по СУБД;
$Д_2 – Д_1$ – документация по средствам проектирования;
ДЛМ – даталогическая модель;
ИЛМ – инфологическая модель;
Si – выбранное средство проектирования;
$O_т$ – ограничения на используемые технические средства;
$O_п$ – ограничения со стороны пользователей / процессов

Рис. 1.21. Технологическая сеть проектирования для этапа физического проектирования

Факторы, влияющие на проектирование БД

Как было отмечено выше, на стадии инфологического моделирования должна быть собрана и представлена в надлежащем виде вся информация, необходимая и достаточная для дальнейшего проектирования БнД. Для того, чтобы было понятно, какая информация должна фиксироваться при описании предметной области, перечислим основные из факторов, оказывающих влияние на проектирование структуры БД:

1. Специфика предметной области:
   1.1. особенности отображаемых объектов, характер связи между объектами предметной области;
   1.2. «размер» системы (объем хранимых данных).
2. Особенности требуемой обработки информации:
   2.1. характеристика запросов (критерий поиска, частота запроса; состав реквизитов, выдаваемых в ответ, упорядоченность ответа, частота совместного использования реквизитов и т.п.);
   2.2. требования к защите информации;
   2.3. ограничения по времени реакции системы на каждый из запросов, что в свою очередь, определяется несколькими факторами, такими как: режим выполнения запроса (интерактивный, пакетный, в реальном масштабе времени), статус запроса и др.
3. Характеристика пользователей системы:
   3.1. важность / статус, приоритеты;

3.2. число пользователей;
3.3. распределение функций между пользователями; степень пересечения информационных потребностей пользователей;
3.4. приоритеты пользователей в оценке значимости факторов, влияющих на проектирование БД;
3.5. Технология обработки данных;
3.6. возможность / необходимость работы в распределенной среде, в том числе необходимость поддерживать связь с «мобильными» компьютерами;
3.7. «доступные» технологии обработки данных.

4. Состояние существующей системы обработки информации:
4.1. наличие существующей автоматизированной системы обработки информации;
4.2. объем имеющихся «наработок»;
4.3. наличие технических и программных средств, их состояние;
4.4. соотношение объемов «существующей» и «новой» частей проектируемой системы;
4.5. затраты на перевод имеющейся системы на новую основу.

5. Возможности, предоставляемые используемыми (выбранными для реализации проекта) техническими и программными средствами:
5.1. поддерживаемые структуры данных; ограничения, накладываемые программным обеспечением;
5.2. ограничения по объему памяти;
5.3. быстродействие технических средств;
5.4. «производительность» Программного Обеспечения;
5.5. особенности языков манипулирования данными.

6. Трудоемкость проектирования.
7. Финансовые возможности.
8. Квалификация кадров:
8.1. разработчиков;
8.2. пользователей.

9. Используемые методики проектирования:
9.1. наличие средств автоматизации проектирования;
9.2. используемый алгоритм проектирования.

10. Субъективные факторы:
10.1. мода;
10.2. привычки и предпочтения.

Более подробно влияние некоторых из перечисленных выше факторов будет рассмотрено далее, по мере изложения вопросов проектирования БД.

Контрольные вопросы

1. Дайте определение банка данных.
2. Назовите основные преимущества банков данных.
3. Назовите основные недостатки банков данных.
4. Каковы предпосылки создания БнД?
5. Какие требования предъявляются к банкам данных?
6. Какие компоненты включаются в состав банка данных?
7. Что называется Системой Управления Базой данных?
8. Что называется базой данных?
9. Дайте классификацию языковых средств СУБД.

10. Какие поколения языковых средств Вы знаете? Дайте краткую характеристику языковым средствам каждого из поколений.
11. Назовите принципы, по которым построены языки четвертого поколения.
12. Перечислите компоненты языка четвертого поколения.
13. Приведите примеры процедурных и непроцедурных языков. В чем основные отличия между языками этих классов?
14. Назовите основные отличительные особенности банков данных.
15. Какие технические средства необходимы для реализации банка данных?
16. Какие типы ЭВМ чаще всего используются для реализации банков данных?
17. Перечислите основные признаки классификации банков данных.
18. В чем разница между системами со структурированными и неструктурированными базами данных?
19. Охарактеризуйте основные классы СУБД.
20. СУБД каких классов являются в настоящее время наиболее распространенными?
21. Каковы основные тенденции развития СУБД наблюдаются в настоящее время?
22. Сравните системы типа OLTP и OLAP.
23. Перечислите основные отличия корпоративных и настольных СУБД.
24. Сравните локальные, интегрированные и распределенные БД.
25. Перечислите этапы проектирования баз данных.
26. Что называется схемой, подсхемой и внешней схемой?
27. Какую роль выполняет подсхема? Какие преимущества дает ее использование?
28. Что называется словарем данных, репозиторием?
29. Охарактеризуйте взаимодействие компонентов БнД при работе с системой.
30. Что называется инфологической моделью?
31. Является ли инфологическое моделирование этапом, присущим только проектированию баз данных?
32. Какая информация является исходной для построения концептуальной модели?
33. Кто должен создавать концептуальную модель и почему?
34. Какие требования предъявляются к инфологической модели?
35. Что называется даталогической моделью?
36. Какая информация является исходной для построения даталогической модели?
37. Какие вопросы решаются на стадии даталогического моделирования?
38. Изобразите технологическую сеть проектирования для стадии даталогического моделирования.
39. Что называется физической моделью?
40. Какая информация является исходной для построения физической модели?
41. Какие вопросы решаются на стадии физического моделирования?
42. Какие факторы влияют на проектирование БД?
43. Перечислите основные категории пользователей банков данных.
44. Кого называют конечными пользователями?
45. Кого называют администраторами банка данных?
46. Перечислите основные функции администратора банка данных.
47. В каком порядке должны выполнятся этапы проектирования БнД?

Глава 2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Концептуальное проектирование является центральной частью, ядром всего процесса проектирования баз данных. Подходы к концептуальному проектированию, излагаемые в разных литературных источниках и реализованные в разнообразных CASE-системах, отличаются друг от друга. Многие из подходов имеют существенные недостатки, что не позволяет рекомендовать именно их как основные. В связи с этим в данной главе изложена некоторая базовая модель, с которой производятся сравнения других систем.

Так как в настоящее время CASE-систем достаточно много, то неизвестно, с какой именно из систем придется проектировщику столкнуться на практике. Поэтому в данной главе даны некоторые критерии, по которым следует сравнивать CASE-системы, и приведены обобщенные рекомендации по построению ER-моделей в зависимости от доступных изобразительных средств и алгоритмов проектирования логической структуры базы данных.

В качестве примеров рассмотрен процесс концептуального моделирования в среде Design / IDEF и ERWin. Эти системы достаточно широко известны, приемлемы по стоимости и, в силу этого, широко используются в учебном процессе. При освоении курса «Базы данных» можно ознакомиться только с тем разделом, который соответствует CASE-средству, которое будет реально использоваться. Если предполагается использовать CASE-систему, которая подробно не освещена в учебнике, то следует обращаться к документации по данной системе и другим доступным источникам, а по методике построения самой модели использовать общие рекомендации, изложенные в данной главе.

2.1. ОБЩИЕ СВЕДЕНИЯ О МОДЕЛИРОВАНИИ ПРЕДМЕТНОЙ ОБЛАСТИ

Уточнение понятия концептуальной модели

В базе данных отображается какая-то часть реального мира. Естественно, что полнота ее описания будет зависеть от целей создаваемой информационной системы. Как указано выше, часть реального мира, представляющая интерес для данного исследования, называется *предметной областью* (ПО). Для того чтобы база данных адекватно отражала предметную область, проектировщик должен хорошо представлять себе все нюансы, присущие ей, и уметь отобразить их в базе данных.

Предметная область должна быть предварительно описана. Для этого, в принципе, может использоваться и естественный язык, но его применение имеет много недостатков, основными из которых являются громоздкость описания и неоднозначность его трактовки. Поэтому обычно для этих целей используют искусственные формализованные (чаще всего – графические) языковые средства.

Формализованное описание предметной области будем называть ее ***концептуальной моделью (КМ).*** Предметные области могут быть различными, и для их моделирования могут потребоваться специфические средства, соответствующие особенностям этих областей. Мы в данном учебнике будем ориентироваться, в основном, на экономико-организационные системы. Хотя описываемые далее подходы к проектированию являются более универсальными и могут быть использованы и в других предметных областях.

Моделирование предметных областей выполняется с разными целями, например, для реинжиниринга бизнес-процессов, для прогнозирования развития предметной области, при проектировании баз данных и программного обеспечения и др. Используемые средства и методы моделирования при этом будут различаться. Естественно, что в данном учебном пособии мы остановимся только на тех средствах и методах моделирования, которые находят наибольшее использование при проектировании баз данных.

Как было отмечено в главе 1 данного пособия, существует большое разнообразие видов БД. Подходы к проектированию баз данных разных классов будут существенно различаться. Так как в настоящее время основную часть баз данных представляют структурированные базы данных, то основное внимание будет уделено проектированию именно таких систем.

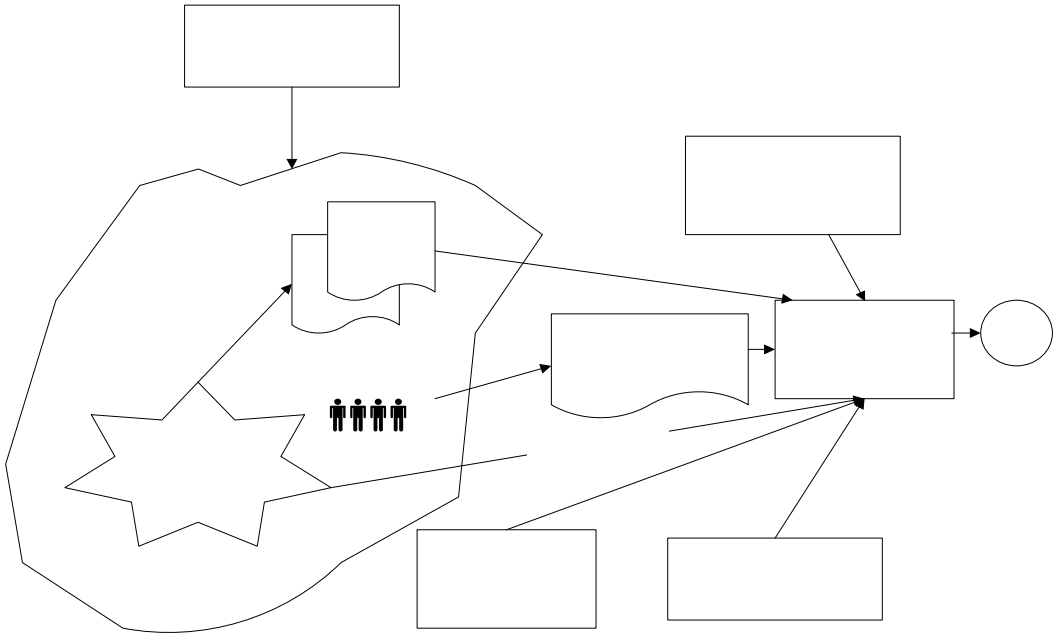


Рис. 2.1. Стадия инфологического моделирования – исходная и результатная информация

Изучение предметной области складывается из непосредственного наблюдения протекающих в ней процессов, изучения документов, циркулирующих в системе, а также интервьюирования участников этих процессов (рис. 2.1). Так как описание инфологической модели выполняется на специализированном языке, то необходимо владение этим языком. Следует обратить внимание на то, что возможности языка описания ИМЛ оказывают влияние на методику построения модели с использованием данных языковых средств. Построение концептуальной модели может выполняться как «вручную», так и с использованием автоматизированных средств проектирования. Средства автоматизации проектирования отличаются как нотациями используемых языковых средств, так и алгоритмами преобразования концептуальной модели в модели базы данных. Это в свою очередь скажется на методике построения модели в их среде.

Основные компоненты концептуальной модели

Основными компонентами концептуальной модели ПО являются (рис. 2.2):

- описание объектов ПО и связей между ними;
- описание информационных потребностей пользователей;
- описание существующей информационной системы (документы, документооборот, при наличии автоматизированной информационной системы – ее описание);
- описание алгоритмических зависимостей показателей;
- описание ограничений целостности;
- описание функциональной структуры системы, для которой создается АИС;
- требования к ИС и существующие ограничения.

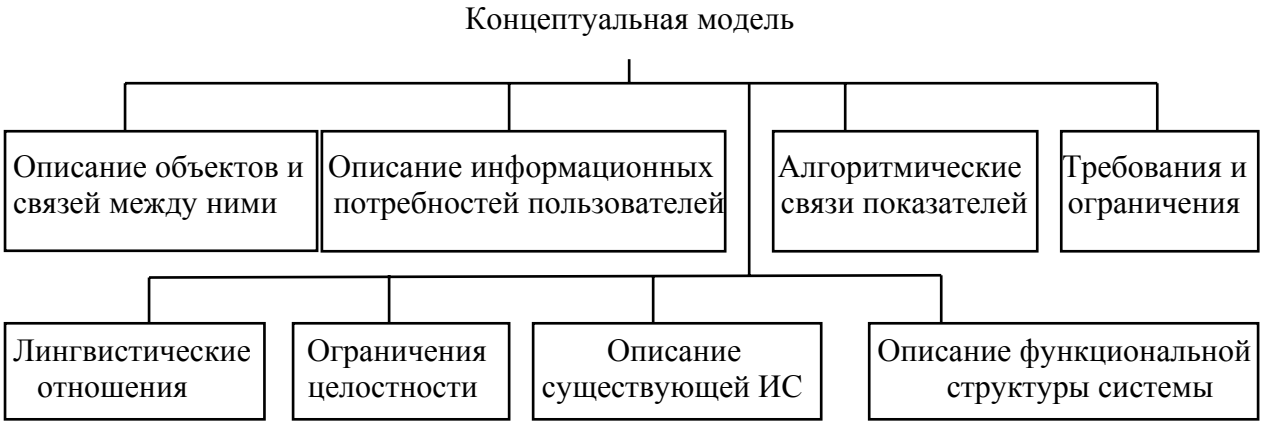


Рис. 2.2. Компоненты концептуальной модели

Далее мы более подробно остановимся на первой из перечисленных компонент, так как именно она оказывает наибольшее влияние на проектирование структуры базы данных. Чаще всего описание объектов ПО и связей между ними представляется в виде так называемых *ER-моделей* (или ER-диаграмм)[7].

Требования, предъявляемые к концептуальной модели

К концептуальной модели предъявляются следующие требования:

- адекватное отображение предметной области (язык для представления модели должен обладать достаточными выразительными возможностями для отображения явлений, имеющих место в предметной области, а сама модель должна содержать всю необходимую и достаточную информацию для дальнейшего проектирования системы);
- непротиворечивость (модель отражает взгляды и потребности всех пользователей системы, а также обычно является результатом работы многих специалистов, поэтому целостное описание ПО должно быть проверено на непротиворечивость);
- однозначная трактовка модели всеми ее пользователями (обеспечивается формализованностью языка и четким его пониманием всеми участниками процесса создания ИС);
- легкость восприятия разными категориями пользователей (обеспечивается выбором соответствующего языка моделирования);
- конечность модели (несмотря на то, что реальный мир, отображаемый в КМ, является по своей природе бесконечным, инфологическая модель является конечной, что обеспечивается четким ограничением предметной области);
- легкость модификации (в концептуальную модель по разным причинам часто приходится вводить новые объекты или модифицировать существующие; ИЛМ должна в связи с этим обладать свойством легкой расширяемости, обеспечивающим ввод новых данных без изменения раннее определенных. То же самое можно сказать и об удалении и корректировке данных);
- возможность композиции и декомпозиции модели (в связи с большой размерностью реальных инфологических моделей должна обеспечиваться возможность ее композиции и декомпозиции).

[7] ER – Entity-Relationship – Сущность-отношение.

Желательно, чтобы язык спецификации концептуальной модели был одинаково применим как при ручном, так и при автоматизированном проектировании информационных систем. Последние предъявляет к языку дополнительные требования, а именно, он должен:

- быть вычисляемым, то есть восприниматься и обрабатываться ЭВМ;
- использовать «дружелюбные» пользователю интерфейсы, в частности, графические;
- быть независимым от оборудования и других ресурсов, которые подвержены частым изменениям;
- использовать средства тестирования КМ, а также иметь аппарат для указания того, что спецификация завершена и по ней может быть выполнена генерация структур баз данных.

При автоматизированном проектировании все изменения, внесенные в КМ, должны быть автоматически отражены в связанных с модифицируемым элементом компонентах банка данных.

Желательно, чтобы КМ строили специалисты, работающие в предметной области, для которой создается АИС, а не проектировщики систем машинной обработки данных. Если в силу определенных причин это невозможно обеспечить, то необходимо, чтобы первые могли хотя бы проверить сделанное другим специалистом описание, чтобы убедиться, что специфика предметной области воспринята и отображена правильно.

Концептуальная модель является средством коммуникации разнообразных коллективов как конечных пользователей, так и разработчиков. Информация из КМ корреспондирует со словарной системой и другими компонентами банка данных.

Преимущества использования ER-моделирования

ER-модель представляет собой графическое описание предметной области в терминах «объект-свойство-связь». ER-модель является одним из элементов концептуальной модели. Использование ER-моделирования (и, особенно, в сочетании с автоматизированными средствами проектирования – CASE-средствами) дает много преимуществ**:**

- предписывая определенную методологию моделирования, делает анализ предметной области более целенаправленным и конкретным;
- является удобным средством документирования проекта;
- позволяет вести проектирование АИС без привязки к конкретной целевой СУБД и осуществлять выбор последней в любой момент времени (чем ближе к концу проектирования это будет сделано, тем точнее может быть выбор).

При использовании ER-моделирования в составе CASE-средств появляются дополнительные преимущества:

- снижаются требования к знанию деталей языков описания данных (DDL) и диалектов SQL конкретных СУБД;
- при смене используемой СУБД не надо проводить проектирование заново; следует только осуществить шаг по переводу ER-модели в целевую (если выбранная Вами целевая СУБД поддерживается данным CASE-средством, то такой переход вообще будет выполнен автоматически);
- наличие в CASE-средстве возможности «обратного проектирования» (т.е. получения ER-диаграммы по имеющимся описаниям данных) позволяет использовать существовавшие ранее наработки для «реинжиниринга» системы;
- указание связи объектов в ER-модели, и соответствующая миграция ключа при преобразовании этой модели в целевую, позволяет не только задавать контроль целостности связи при ведении БД, но и автоматически обеспечивает согласованное описание схемы (внешний ключ мигрирует в связанное отношение; при этом имя, тип и длина соответствующего атрибута повторяются в зависимой сущности);

- сокращается время проектирования системы;
- появляется возможность автоматизированного тестирования проекта на всех этапах проектирования;
- повышается качество документирования проекта;
- мощные современные CASE-средства позволяют вести коллективную разработку проекта.

2.2. ОПИСАНИЕ БАЗОВОЙ ER-МОДЕЛИ

Существует большое число нотаций (изобразительных средств) и методик построения ER-моделей. Ниже мы рассмотрим методику построения ER-модели, предлагаемую автором данного пособия. Будем называть эту методику ER-моделирования *базовой*. С некоторыми другими методиками можно познакомиться в п 2.3, а также по литературным источникам [3, 5, 7] или по документации по CASE-средствам.

Понятие «объект» и «класс объектов»

В предметной области имеется множество разнообразных объектов. Объект – понятие очень широкое. Его трудно точно определить. Обычно под объектом понимают некую сущность (реальную или абстрактную), о которой собирается какая-то информация. Объекты группируются в классы. *Классом объектов* называют совокупность объектов, обладающих одинаковым набором свойств. Например, для объектов класса «СТУДЕНТ» таким набором свойств являются: «ГОД _ РОЖДЕНИЯ», «ПОЛ» и другие.

Объекты могут быть реальными, как названный выше объект «СТУДЕНТ», и абстрактными, как, например, «ПРЕДМЕТЫ», которые изучают студенты.

ER-модель строится на уровне классов объектов, а не отдельных экземпляров объектов.

Каждому классу объектов в ER-модели присваивается уникальное имя. Именем класса объекта является грамматический оборот существительного (существительное, у которого могут быть прилагательные и предлоги). Если имя состоит из нескольких слов, то желательно, чтобы первым стояло существительное. Существительное должно употребляться в единственном, а не во множественном числе (например, «ДИСЦИПЛИНА _ ИЗУЧАЕМАЯ»). Если в предметной области традиционно используются разные имена для обозначения какого-либо класса объектов (т.е. имеет место синонимия), то все они должны быть зафиксированы при описании системы, и затем одно из них выбирается за основное, и только оно должно в дальнейшем использоваться в ER-модели. Помимо имени класса объектов в ER-модели может использоваться его короткое кодовое обозначение; для дальнейшего перехода к даталогической модели еще может указываться имя, которое будет использоваться при описании структуры базы данных.

При построении ER-модели желательно дать словесную интерпретацию каждой сущности, особенно, если возможно неоднозначное толкование понятия.

Вместо термина «объект» часто используется термин «сущность». В дальнейшем мы будем рассматривать эти термины как синонимы.

При отражении в информационной системе каждый объект (имеется в виду уже экземпляр объекта, а не весь класс) представляется своим именем, которое называет конкретный объект и отличает один объект от другого. Чтобы выполнять свою роль, имя должно быть уникальным, но, иногда, в реальной жизни это бывает не так (явление синонимии). Поэтому в концептуальной модели должны быть каким-то образом обозначены случаи, когда естественное имя объекта является не уникальным. Уникальное имя объекта будем называть *идентификатором*. Каждый объект должен иметь, по крайней мере, один идентификатор.

Каждый объект обладает определенным набором *свойств* – характеристик, описывающих состояние каждой сущности. Набор (перечень) свойств для всех объектов данного класса будет одинаковым, но конкретные значения этих характеристик и, особенно, сочетание этих значений, будут отличаться от объекта к объекту, что, собственно, и отличает один экземпляр объекта от другого.

Может случиться, что в данной предметной области свойства объекта не представляют для нас интереса. В связи с этим в ER-модели возможно наличие объектов, не имеющих свойств и представленных только своими идентификаторами.

Разновидности объектов

Различают несколько разновидностей объектов. Прежде всего, это *простые* и *сложные* объекты. Объект называется **простым**, если он рассматривается в данном исследовании как неделимый.

Сложный объект представляет собой объединение других объектов, простых или сложных, также отображаемых в информационной системе. Понятие «простой» и «сложный» объект является относительным. В одном рассмотрении объект может считаться простым, а в другом этот же объект может рассматриваться как сложный. Так, например, объект «АУДИТОРИЯ» в случае, если АИС строится только для управления учебным процессом, будет рассматриваться как простой. Если же АИС будет включать подсистемы для служб энергетика, материально-технического снабжения и др., то «АУДИТОРИЯ» будет рассматриваться как составной объект.

Выделяют несколько разновидностей сложных объектов: *составные, обобщенные и агрегированные* объекты.

Составной объект соответствует отображению отношения «целое-часть». Примерами составных объектов являются УЗЕЛ – ДЕТАЛИ, КЛАСС – УЧЕНИКИ и т.п.

Обобщенный объект отражает наличие связи «род-вид» между объектами предметной области. Например, объекты «СТУДЕНТ», «ШКОЛЬНИК», «АСПИРАНТ», «УЧАЩИЙСЯ ТЕХНИКУМА» образуют обобщенный объект «УЧАЩИЙСЯ». Объекты, составляющие обобщенный объект, называются его **категориями.**

Как «родовой» объект, так и «видовые» объекты могут обладать определенным набором свойств. Причем наблюдается так называемое наследование свойств, т.е. «видовой» объект обладает всеми теми свойствами, которыми обладает «родовой» объект, плюс свойствами, присущими только объектам этого вида.

Определение родовидовых связей означает классификацию объектов предметной области по тем или иным признакам. Естественно, что классификация может быть многоуровневой.

Агрегированные объекты соответствуют обычно какому-либо процессу, в который оказываются «вовлеченными» другие объекты. Например, агрегированный объект «ПОСТАВКА» объединяет в себе объекты «ПОСТАВЩИК», «ПОТРЕБИТЕЛЬ», а также саму поставляемую «ПРОДУКЦИЮ». Своеобразным объектом является «ДАТА _ ПОСТАВКИ». Агрегированный объект может, так же как и простой объект, иметь характеризующие его свойства. В рассматриваемом примере таким свойством может быть «РАЗМЕР _ ПОСТАВКИ». Агрегированные объекты обычно называются отглагольными существительными.

Изображение простого объекта

Для графического обозначения простого объекта будем использовать прямоугольник, ограниченный сплошной линией. Название класса объекта пишется над ним. Внутри прямоугольника записывается название атрибута, именующего объект (рис. 2.3). Если у объекта имеется несколько имен, то для каждого из них выделяется отдельный «сектор» в этом прямоугольнике.

Рис. 2.3. Изображение объекта

Если какое-либо имя объекта не является уникальным, будем использовать букву «н» рядом с таким именем (рис. 2.4).

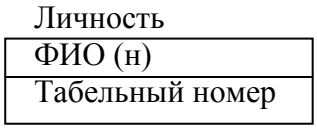


Рис. 2.4. Пример изображения объекта с несколькими идентифицирующими атрибутами

Встречаются случаи, когда идентификация одних объектов зависит от идентификации других. Например, часто для участков цехов предприятия используется не сквозная нумерация, а в пределах каждого цеха, то есть участок имеет составной идентификатор «НОМЕР _ ЦЕХА*НОМЕР _ УЧАСТКА». Назовем подобные объекты (в нашем примере это объект «УЧАСТОК») *зависимыми от идентификации сущностями*. Для отображения таких ситуаций будем перечеркивать линию, соединяющую соответствующие объекты, около конца, прилегающего к зависимому объекту (рис. 2.5, 2.6).

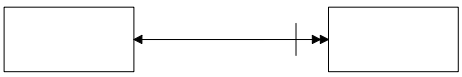


Рис. 2.5. Изображение зависимой по идентификации сущности

Если не использовать специального обозначения для указания зависимости по идентификации, то (для нашего примера) для объекта «УЧАСТОК» следует указать составной идентификатор «НОМЕР_ЦЕХА*НОМЕР_УЧАСТКА», при этом его надо указать в одном секторе прямоугольника, а не выделять несколько секторов, как в случае наличия у объекта несколько имен.

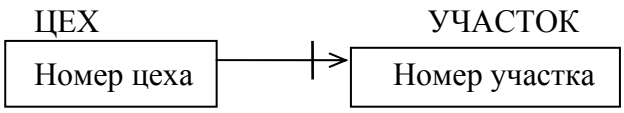


Рис. 2.6. Пример изображения зависимой по идентификации сущности

Описание свойств объекта. Разновидности свойств

Как указывалось выше, класс объектов представляет собой совокупность объектов, обладающих одинаковым набором свойств. При описании предметной области надо изобразить набор свойств, фиксируемых для объектов каждого из представленных в модели классов. Для обозначения свойств будем использовать прямоугольник, изображенный пунктирной линией.

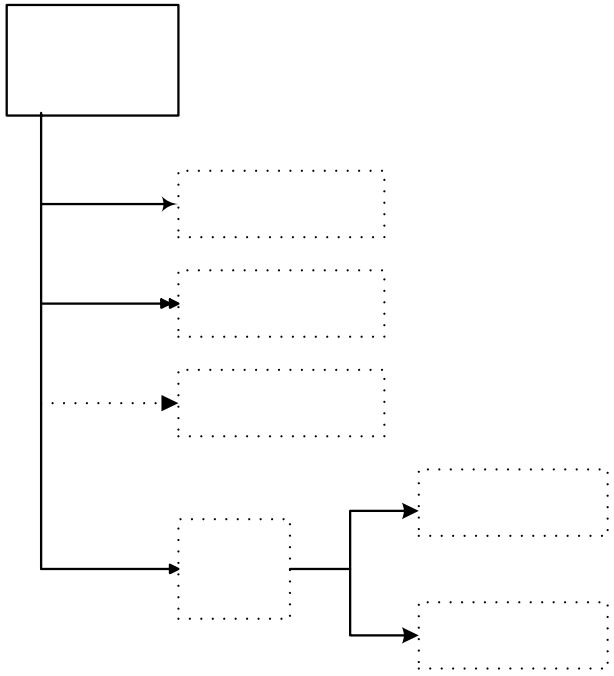


Рис. 2.7. Изображение объекта и его свойств (условные обозначения)

Связь между объектом и характеризующим его свойством изображается в виде линии, соединяющей их обозначения. Характер связи между объектом и его свойством может быть различный. Объект может обладать только одним значением какого-то свойства в каждый момент времени. Например, каждый человек может иметь только одну «ДАТУ _ РОЖДЕНИЯ» или «СТАЖ _ РАБОТЫ». Назовем такие свойства *единичными*. Для других свойств возможно существование одновременно нескольких их значений у одного и того же объекта (например, свойство «ИНОСТРАННЫЙ ЯЗЫК» у объекта «СОТРУДНИК» в случае, если «СОТРУДНИК» может владеть несколькими иностранными языками). Такое свойство будем называть *множественным*. При изображении связи между объектом и его свойствами для единичных свойств будем использовать одинарную стрелку, а для множественных свойств – двойную стрелку на конце линии, соединяющей объект с данным свойством (рис. 2.7, 2.8).

Д

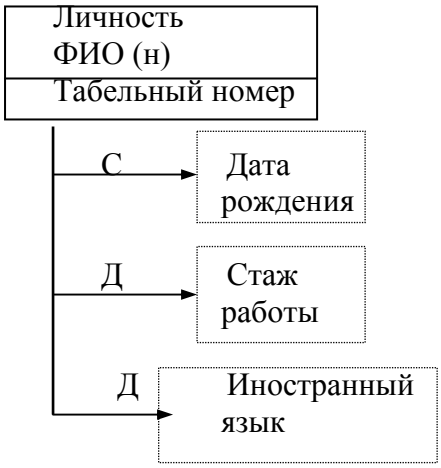


Рис. 2.8. Пример изображения единичных и множественных динамических и статических свойств

Значения некоторых свойств не может измениться с течением времени. Назовем такие свойства *статическими*, а те свойства, значения которых может изменяться со временем, будем называть *динамическими*. Для обозначения динамических свойств будем использовать букву **«Д»**, а статических – **«С»** над соответствующей линией. Так, упомянутое выше свойство ДАТА_РОЖДЕНИЯ будет являться статическим, а «СТАЖ» – динамическим.

Другой характеристикой связи между объектом и его свойством является признак того, присутствует ли это свойство у всех объектов данного класса, либо оно может отсутствовать у некоторых из объектов. Например, для отдельных служащих может иметь место свойство «УЧЕНАЯ_СТЕПЕНЬ», а другие объекты этого класса могут не обладать указанным свойством. Назовем свойства, присутствующие не у всех объектов данного класса, *условными*. При изображении связи условного свойства с объектом будем использовать пунктирную линию, а в случае, если свойство определено для всех экземпляров объектов данного класса, – сплошную (рис. 2.9).

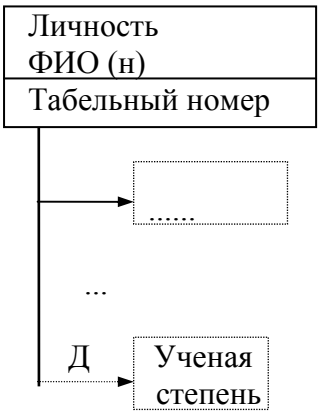


Рис. 2.9. Пример изображения условного свойства

Правильность отображения предметной области в ER-модель будет зависеть от того, какие ситуации возможны в данной предметной области, а какие – нет. Так, если в вузе сотрудник может занимать несколько должностей одновременно, например, быть одновременно ректором и заведующим кафедрой, то фрагмент ER-модели будет выглядеть так, как изображено на рис. 2.10 а), а если внутривузовское совместительство не разрешено, – то как изображено на рис. 2.10 б).

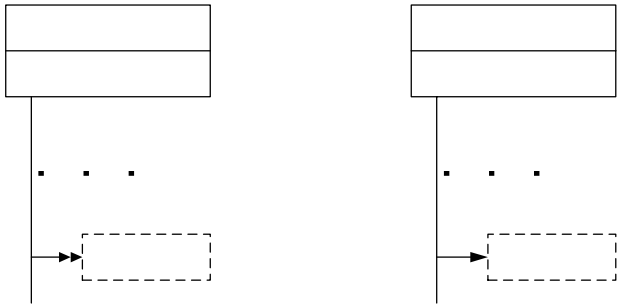


Рис. 2.10. Варианты изображения
(в зависимости от особенностей предметной области)

Иногда в ER-модели бывает полезно ввести понятие «*составного свойства*». Примером такого свойства могут быть «АДРЕС», состоящий из «ГОРОДА», «УЛИЦЫ», «ДОМА» и «КВАРТИРЫ». Будем использовать для обозначения составного свойства пунктирный квадрат, из которого исходят линии, соединяющие его с обозначениями составляющих его элементов (рис. 2.7, 2.11).

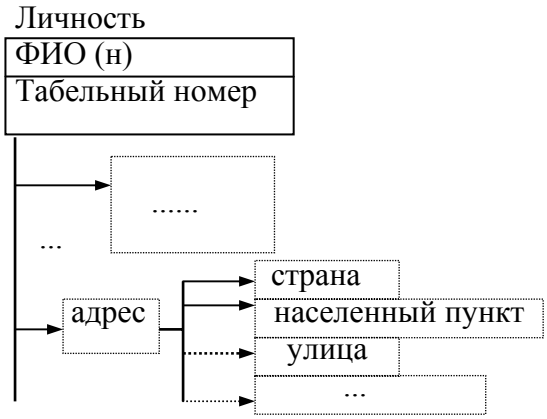


Рис. 2.11. Пример изображения составного свойства

При проектировании БД определяются тип и длина полей. Для того, чтобы иметь возможность правильно выбрать эти характеристики, необходимо иметь соответствующую информацию о типе представления атрибута в «немашинной» системе и требования / пожелания пользователей об их отображении в автоматизированной системе, может быть даже с предпочтениями. Например, предположим, что желательно было бы хранить в БД изображение. Если это невозможно в целевой СУБД, то:

- поле, соответствующее данному атрибуту, не вводить;
- связать с системой, которая может хранить рисунок;
- заменить рисунок описанием.

Например, в «Листке по учету кадров» хранится фотография. Если есть возможность ее сканирования и связи соответствующего файла с записями БД, то сделать это, если нет – то ничего, соответствующего фотографии, не хранить в ИС.

Для всех реквизитов символьного типа должна быть указана их максимальная длина (а лучше не только максимальная, но и минимальная и «средневзвешенная»).

Чтобы не загромождать ER-модель, такие характеристики рекомендуется отображать в репозитории (в каталоге реквизитов).

Таблица 2.1

Сводная таблица характеристик свойств объекта

Наименование характеристики	Что отражает	Возможные значения	Пример*
Назначение	что определяет	• идентификатор (обозначает, называет объект)	ФИО
		• качественная характеристика	пол
		• количественная характеристика	вес
		• дата / время совершения события	дата рождения
		• изображение	фотография
Изменяемость значения свойства	может ли меняться в течение жизненного цикла объекта	• динамическое	образование
		• статическое	дата рождения
Обязательность	может ли отсутствовать данное свойство у объекта	• обязательное	дата рождения
		• необязательное	ученая степень
Элементарность	возможность разбиения на составляющие элементы	• простое	пол
		• составное	адрес
Множественность	возможность наличия нескольких значений для данного свойства у одного объекта	• единичное	дата рождения
		• множественное	номера телефонов
Форма отображения в знаковой системе		• символьная	адрес
		• числовая	вес
		• дата	дата рождения
		• время	время прихода на работу
		• изображение	фотография сотрудника
		• логическое	военнообязанный (да / нет)
Способ получения		• датчики / счетчики	код продукции при автоматическом считывании штрих-кода
		• из внешней среды	количество заказанной продукции
		• производная информация	общая сумма заказа
		•	

* примеры приведены для класса объектов «ЛИЧНОСТЬ»

Понятия «объект» и «свойство» являются относительными. Что в каждой из моделей ПО следует считать самостоятельным объектом, а что – свойством другого объекта, будет зависеть от аспекта рассмотрения данной предметной области. Например, пусть строиться АИС для управления конкретным учебным заведением. Для «СОТРУДНИКОВ» и «УЧАЩИХСЯ» указывается, какое учебное заведение они закончили. Больше никакой информации об учебных заведениях не хранится; никакой специальной обработки по этому признаку не производится. В этом случае не стоит выделять отдельный объект «УЧЕБНОЕ _ ЗАВЕДЕНИЕ», а следует считать его свойством соответствующего объекта. Если же в предметной области отражается дополнительная информация об учебных заведениях, например, их адрес, тип и т.п., то «УЧЕБНОЕ _ ЗАВЕДЕНИЕ» следует рассматривать как самостоятельный объект.

В общем случае можно дать следующие рекомендации по поводу того, что следует выделять в качестве самостоятельного объекта в ER-модели. В качестве самостоятельного объекта в ER-модели следует изображать сущности:

- которые имеют более одного идентификатора;
- для которых фиксируются какие-либо их свойства;
- которые участвуют более чем в одной связи.

В экономических организационных системах большая часть информации отражается в символьном виде. При этом различают реквизиты-признаки, отражающие качественные характеристики объектов, и реквизиты-основания, отражающие количественные характеристики объектов. Некоторые характеристики могут быть получены различными способами, в том числе путем вычисления из других, хранящихся в информационной системе показателей. Такие показатели называются *производными*. При проектировании БД необходима информация о возможных способах получения каждого из реквизитов для решения вопроса о том, какая информация должна в явном виде храниться в БД, а какая может быть получена путем преобразования имеющейся информации.

Алгоритмические зависимости

В инфологической модели должны быть отражены алгоритмические зависимости между показателями. Обычно для этих целей используются графы взаимосвязи показателей, отражающие, какие показатели служат исходными для вычисления других (рис. 2.12). Расчетные формулы, а также алгоритмы вычислений также в том или ином виде должны быть представлены в ИЛМ.

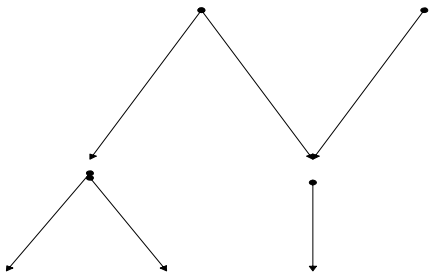


Рис. 2.12. Граф взаимосвязи показателей

Следует обратить внимание на то, что производными могут быть не только количественные, но и качественные характеристики. Например, для объекта «СТУДЕНТ» имеется характеристика «ОТЛИЧНИК». Значение этой характеристики определяется по следующему правилу: отличником считается студент, защитивший все курсовые работы / проекты на «отлично», сдавший все экзамены на «отлично», а также выполнивший в заданный срок все остальные, предусмотренные учебным планом контрольные мероприятия.

Информация об алгоритмических зависимостях не является элементом ER-модели, но является необходимым компонентом описания предметной области; эта информация используется для определения состава хранимых в базе данных показателей, определения ограничений целостности, реализации бизнес-правил.

Интегральные характеристики класса объектов

Как отмечалось выше, в ER-модели отображаются не отдельные экземпляры объектов, а классы объектов. Когда в ER-модели изображено обозначение объекта, то ясно, что речь идет о классе объектов, обладающих описанными свойствами. Поэтому в эти модели в большинстве случаев можно в явном виде не вводить еще и обозначение для класса объектов. Явное изображение класса объектов необходимо только в том случае, если в предметной области для данного класса объектов фиксируются не только характеристики, относящиеся к отдельным объектам этого класса, но и какие-то интегральные характеристики, относящиеся ко всему классу в целом. Например, если для класса объектов «СОТРУДНИК _ ПРЕДПРИЯТИЯ» фиксируется не только возраст каждого из сотрудников, но и средний возраст всех сотрудников, то в ER-модели необходимо отразить не только объект «СОТРУДНИК», но и класс объектов «СОТРУДНИКИ». Для отображения класса объектов лучше использовать какое-то специальное обозначение; в нашем случае – это прямоугольник, очерченной двойной линией (рис. 2.13).

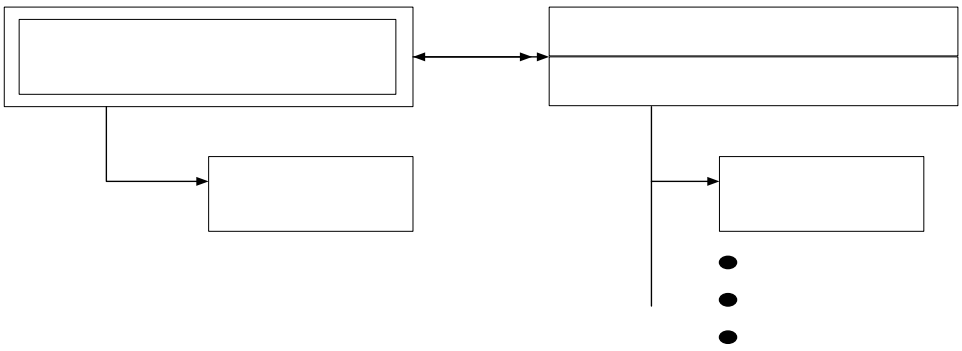


Рис. 2.13. Отображение интегральных характеристик класса объектов

Для характеристики каждого класса объектов полезно указать число объектов в классе, а также динамику его изменения. Чаще всего указывается число объектов в классе на начало периода, а также число «прибывающих» и «убывающих» объектов за фиксированный период. Для того чтобы не перегружать графическое изображение ER-модели, будем эту информацию отображать в отдельных таблицах. Например, если предметной областью является ВУЗ, имеющий стабильный ежегодный набор студентов, то количественная характеристика класса объектов «СТУДЕНТ» может иметь вид, представленный в таблице 2.2. Если число объектов в классе или динамика изменения не являются постоян-

ными, то для соответствующих показателей желательно фиксировать максимальное, минимальное и среднее число объектов в классе.

Таблица 2.2

Описание классов объектов (фрагмент)

Код класса объектов	Наименование класса	Определение	Код родительского класса	Число объектов в классе	Динамика изменения	
					+	–
1	Кадры	Все лица, которые либо работают, либо обучаются в институте	–	15000		
2	Учащиеся		1	11200		
3	Студенты		2	5000	1000	1000
4	Слушатели подготовительных курсов		2	6000	6000	6000
5	Аспиранты		2	200	80	80
6	Сотрудники		1	500		
7	Преподаватели		6	300	10	20
8	Вспомогательный персонал		6	200	60	80

Количественные характеристики классов объектов используются не только для определения объема памяти, занимаемого БД, для обоснованного принятия решений по организации данных. Знание динамики изменения объектов в классе дает информацию, необходимую для принятия решений по организации данных и технологии их обработки. Так, в приведенном выше примере число объектов в классе «СТУДЕНТ» является постоянным (5000), но ежегодно 1 / 5 часть студентов оканчивает институт и столько же новых студентов поступает. Это означает, что ежегодно добавляется 20% от общего объема данных о студентах, и такой же объем данных должен быть перенесен в архивные файлы. Без наличия информации о динамике изменения класса объектов необходимость принятия соответствующих решений была бы просто не видна.

Связи между объектами

Кроме связи между объектом и его свойствами в ER-модели фиксируются связи между объектами разных (а иногда – и одного и того же) классов. *Связь (Relationship)* – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе – нулевым) количеством экземпляров другой сущности. Обычно рассматриваются *бинарные связи*, т.е. связи между двумя классами объектов. Связи являются двунаправленными. Связи могут устанавливаться и сущностями одного класса. Например, связь «Быть Руководителем» устанавливается между разными экземплярами объектов одного класса «СОТРУДНИК».

Отображение типа связи

Различают связи типа «*один к одному*» (1 : 1), «*один ко многим*» (1 : M), «*многие к одному*» (M : 1) и «*многие ко многим*» (M : M). Иногда эти типы связей называются степенью связи (или кардинальностью).

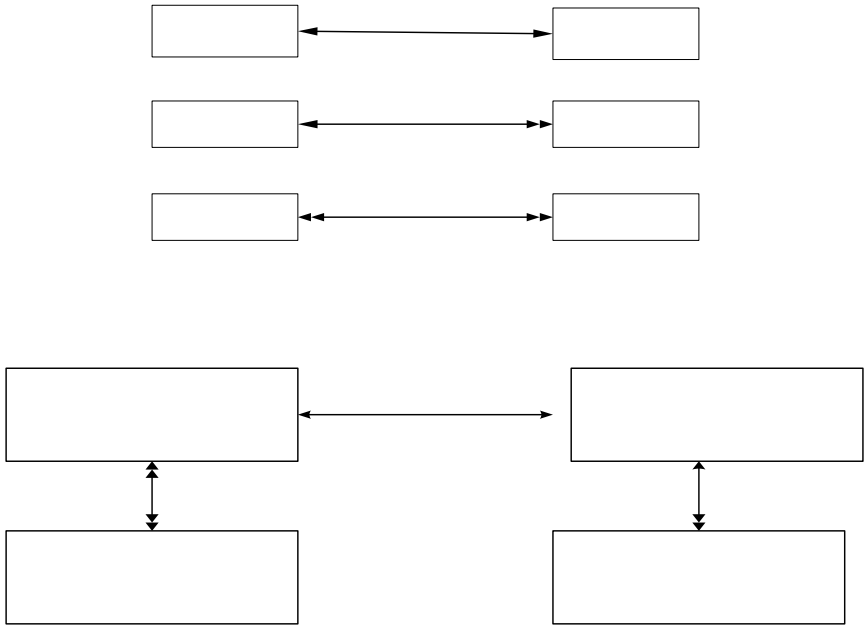


Рис. 2.14. Виды связей между объектами

Если связь «множественная», то желательно еще указать и «мощность» связи (число объектов M; это может быть минимальное, максимальное и среднее число объектов в связи).

Возможны случаи, когда между парой объектов объявлено несколько связей (рис. 2.15). В этом случае связи следует именовать (указание роли)[8].

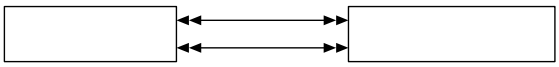


Рис. 2.15. Пример объявления двух связей между парой объектов

Отображение класса принадлежности

Кроме степени связи в ER-модели для характеристики связи между разными сущностями надо указывать так называемый «*класс принадлежности*» (обязательность связи), который показывает, может ли отсутствовать связь объекта данного класса с каким-либо

[8] Вообще будем придерживаться следующего правила: если роль связи очевидна, то ее имя можно не указывать на схеме. Но в любом случае указание этого имени (или, наоборот, его отсутствие) не является ошибкой. Указание имени связи делает модель более «информативной», легкой для понимания, но не оказывает влияния на алгоритм проектирования БД.

объектом другого класса. Класс принадлежности сущности может быть либо *обязательным*, либо *необязательным*.

Аналогично с условным свойством необязательное вхождение объекта в связь будем обозначать пунктирной линией с той стороны, объекты которой могут «не входить» в соответствующую связь. Таким образом, линия, соединяющая объекты, может быть полностью сплошной, полностью пунктирной, либо наполовину – сплошной, наполовину – пунктирной (рис. 2.16).

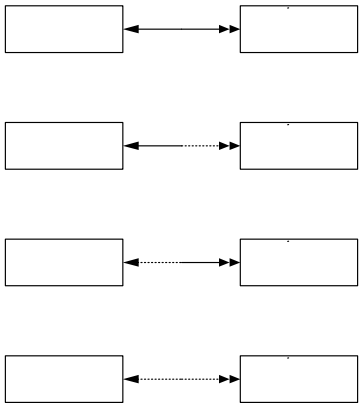


Рис. 2.16. Варианты классов членства

Поясним сказанное на конкретных примерах. Для графической иллюстрации используем так называемые диаграммы ER-экземпляров (вместо диаграмм ER-типа, которые мы использовали до сих пор).

Мы хотим отобразить в инфологической модели связь между двумя классами объектов: «ЛИЧНОСТЬ» и «ЯЗЫК _ ИНОСТРАННЫЙ».

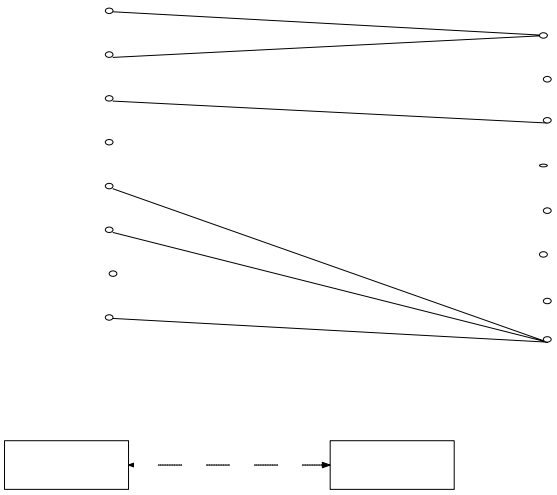


Рис. 2.17. Диаграмма ER-экземпляров (а) и ER-типов (б) (вариант 1 –необязательное членство с обоих концов связи)

Предположим, что предметной областью является завод, некоторые сотрудники которого знают какой-либо иностранный язык, но ни один из них не владеет более чем од-

ним языком. Есть некоторые сотрудники, которые не владеют ни одним языком. Естественно, что имеется много языков, которыми не владеет ни один из сотрудников, а также, что некоторые из сотрудников владеют одним и тем же иностранным языком. В этом случае диаграмма ER-экземпляров будет иметь вид, изображенный на рис. 2.17а), а диаграмма ER-типов – как на рис. 2.17б). Тип связи – 1 : M (на диаграмме это отображено со стороны объекта «ЛИЧНОСТЬ» двойной стрелкой, а со стороны объекта «ЯЗЫК _ ИНОСТРАННЫЙ» – одинарной стрелкой на линии, изображающей связь между рассматриваемыми сущностями), класс членства – необязательный с обоих концов (линия – полностью пунктирная).

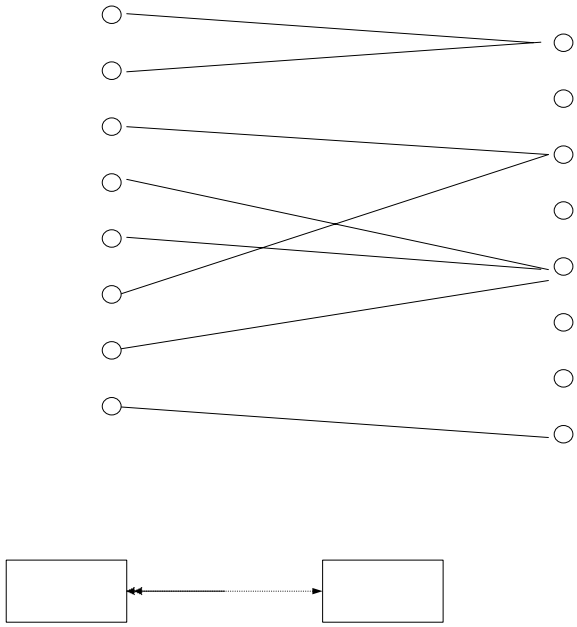


Рис. 2.18. Диаграмма ER-экземпляров (а) и ER-типов (б)
(вариант 2 – связь 1 : M и необязательное членство с одного конца связи)

Предположим далее, что предметной областью является институт, а объект «ЛИЧНОСТЬ» отображает абитуриентов, поступающих в этот институт. Каждый из абитуриентов обязательно должен владеть каким-либо иностранным языком, но никто не владеет более чем одним языком; предположение, что имеется много языков, которыми не владеет ни один из сотрудников, остается актуальным и в этой ситуации. В этом случае диаграмма ER-экземпляров будет иметь вид, изображенный на рис. 2.18 а), а диаграмма ER-типов – как на рис. 2.18 б).

Как в первом, так и во втором из рассмотренных случаев между сущностями наблюдается отношение 1 : M. Разница в рассматриваемых ситуациях заключается в том, что в первом случае класс принадлежности является необязательным для обеих сущностей, а во втором – для сущности «ЛИЧНОСТЬ» класс принадлежности является обязательным. На диаграмме это будет отображено пунктирной линией, прилегающей к объекту «ЯЗЫК _ ИНОСТРАННЫЙ», и сплошной линией, прилегающей к объекту «ЛИЧНОСТЬ».

Пусть предметная область будет та же, что и в предыдущем случае, но имеют место ситуации, когда некоторые абитуриенты знают несколько иностранных языков. В этом случае связь между объектами будет иметь тип M : M. Для такой предметной области диа-

грамма ER-экземпляров будет иметь вид, изображенный на рис. 2.19 а), а диаграмма ER-типов – как на рис. 2.19 б).

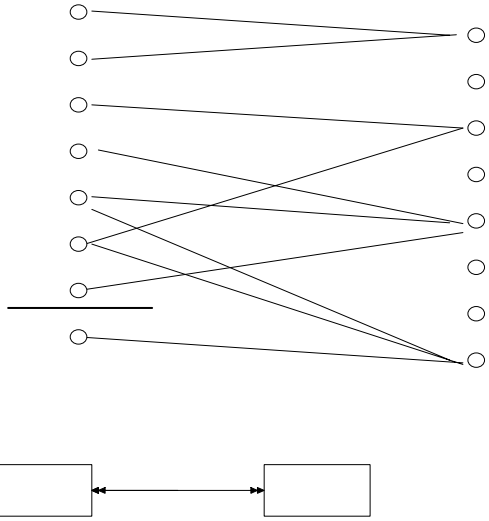


Рис. 2.19. Диаграмма ER-экземпляров (а) и ER-типов (б)
(вариант 3 – связь M : M и необязательное членство с одного конца связи)

Предположим, что предметной областью является некоторый лингвистический институт, в котором каждый из сотрудников обязательно знает несколько иностранных языков, и по каждому из известных науке языков в этом институте имеется хотя бы один специалист, владеющий им. В этом случае связь между объектами будет M : M, и класс принадлежности обоих сущностей является обязательным.

Примеры возможных ситуаций можно было бы продолжить, но суть уже ясна. Характер связи между объектами будет зависеть от особенностей предметной области. Например, если в вузе имеется экстернат и студент может обучаться по индивидуальному графику, то класс членства объекта «СТУДЕНТ» в связи с объектом «ГРУППА» будет необязательным, в противном случае он будет обязательным.

Отображение альтернативной связи

Возможны ситуации, когда в связи участвует один из нескольких возможных классов объектов. Например, если организация имеет центральный офис и филиалы, то служащий может работать либо в центре, либо в филиале, но не там и там одновременно. Назовем такие связи *альтернативными*. На схеме альтернативные связи будут объединяться скобкой (рис. 2.20).

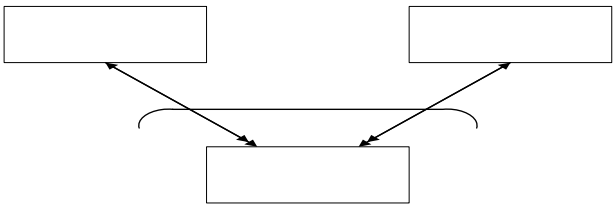


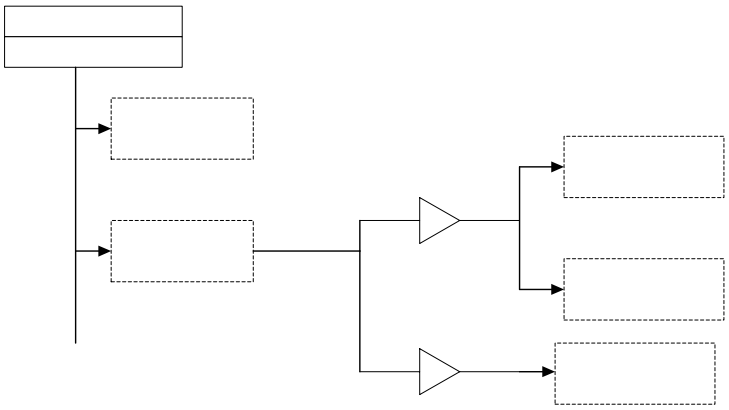
Рис. 2.20. Отображение альтернативной связи

Сложные объекты

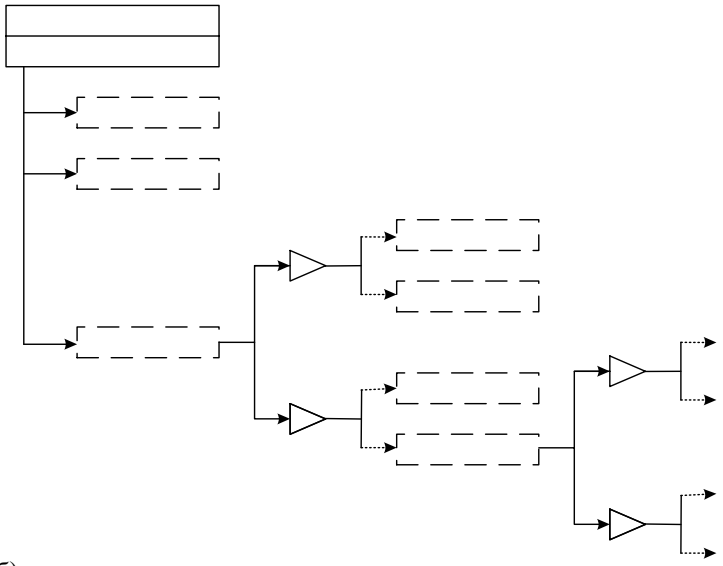
Изображение обобщенных объектов

Обобщенный объект представляет собой объект, в котором явном образом выделены подклассы. Разбиение класса на подклассы осуществляется по какому-то признаку (свойству). Свойство, по которому производится разбиение класса на подклассы, называется **дискриминатором.** Например, подклассы «ВОЕННООБЯЗАННЫЕ» и «НЕВОЕННООБЯЗАННЫЕ» выделяются в зависимости от значения свойства «отношение к воинской обязанности»; подклассы «СТУДЕНТЫ», «АСПИРАНТЫ», «ДОКТОРАНТЫ», «ДОВУЗ» выделяются в зависимости от значения свойства «вид обучения».

Для обозначения подкласса в схеме будем использовать треугольник, который связан с обозначением свойства, по которому производится разбиение на подклассы, и к которому присоединены обозначения свойств, присущих данному подклассу. Широкая часть треугольника направлена в сторону «родового» объекта, острый угол – в сторону «видового» (рис. 2.21а).



а) условные обозначения



б) пример

Рис. 2.21. Изображение обобщенного объекта

На рис. 2.21б) изображен фрагмент инфологической модели, отражающий обобщенный объект «ЛИЧНОСТЬ» для высшего учебного заведения. Для него выделено несколько категорий объектов: «ПРЕПОДАВАТЕЛЬ», «СТУДЕНТ», «АСПИРАНТ».

Естественно, что классификация может быть многоуровневой. Так, в рассматриваемом примере обобщенный объект «ЛИЧНОСТЬ» может быть разбит на два подкласса: «СОТРУДНИК» и «УЧАЩИЙСЯ». «СОТРУДНИКИ», в свою очередь, могут быть классифицированы на «ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКИЙ СОСТАВ», «АДМИНИСТРАЦИЯ» и т.д.

Кроме того, подклассы в совокупности могут составлять исходный класс (полный класс), а могут представлять лишь часть ее (неполный класс). Если при описании предметной области возникает необходимость отобразить эту информацию, то для полного класса будем изображать двойную линию, перечеркивающую линию, идущую от дискриминатора; если класс неполный, то будем изображать одинарную линию, перечеркивающую линию, идущую от дискриминатора.

Подкласс, как и класс, является совокупностью однотипных объектов. Отображать ту или иную сущность в виде отдельного класса или подкласса в составе обобщенного объекта – зависит от проектировщика. Изображение в виде обобщенного объекта является более информативным и, как следствие, дает больший выбор при принятии решений на стадии построения даталогической модели.

При использовании обобщенного объекта связи между объектами могут идти как к знаку всего обобщенного объекта, если объекты всех подклассов участвуют в данной связи, так и к знаку отдельного подкласса, если связь относится только к данному подклассу.

Обобщенный объект следует вводить в модель в том случае, когда надо подчеркнуть общность и различие категорий объектов, входящих в один класс, или в случае, если объекты разных подклассов участвуют в разных связях.

Информация о пересекающихся классах

Выделенные в предметной области классы объектов могут быть как *пересекающимися*, так и *непересекающимися*[9]. Для отображения этих сведений в инфологической модели можно использовать граф пересечений, вершины которого соответствуют классам (подклассам) объектов, а ребра связывают пару вершин лишь в том случае, если соответствующие классы объектов являются пересекающимися.

Для отображения степени пересечения можно воспользоваться взвешенным графом. При этом вес вершины будет обозначать мощность соответствующего множества объектов, а вес ребра – мощность множества, являющегося пересечением множеств, связанных этим ребром (рис 2.22).

[9] Другое название – взаимно исключающие подклассы (когда объект может входить в один из подклассов, но не в несколько одновременно).

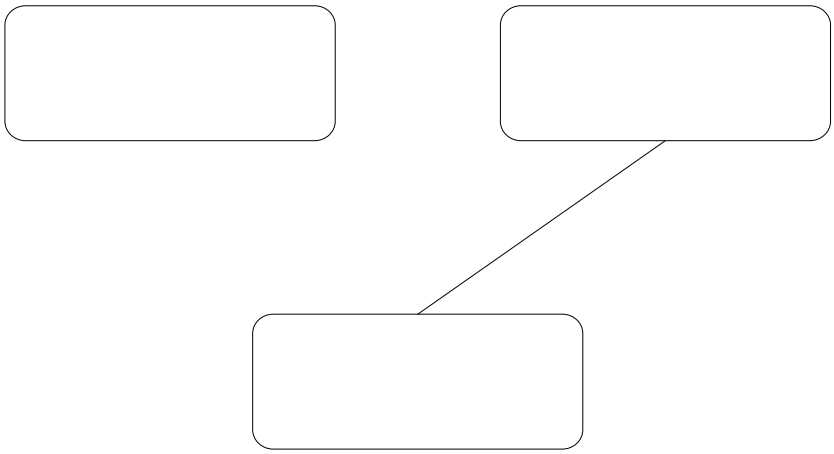


Рис. 2.22. Фрагмент графа пересечений

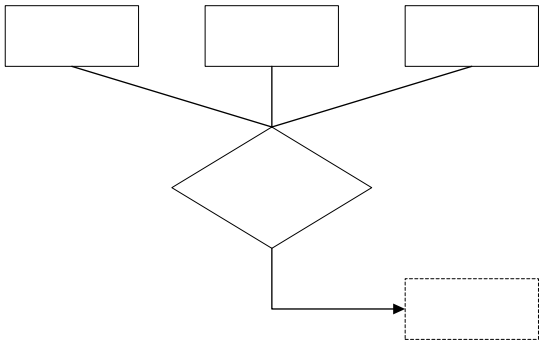
Эту же информацию можно отразить и в табличной форме (табл. 2.3)

Таблица 2.3

Класс 1	Класс 2	Размер пересечения

Изображение агрегированных объектов

Как отмечалось выше**,** агрегированные объекты соответствуют обычно какому-либо процессу, в который оказываются «вовлеченными» другие объекты. Для отображения агрегированного объекта в ER-модели будем использовать следующие условные обозначения: сам агрегированный объект будем изображать ромбом, рядом с которым указывается имя соответствующего агрегированного объекта. Этот ромб связывается с условными обозначениями тех объектов, которые образуют этот агрегированный объект. Свойства агрегированного объекта изображаются так же, как и для простого объекта (рис 2.23 а).



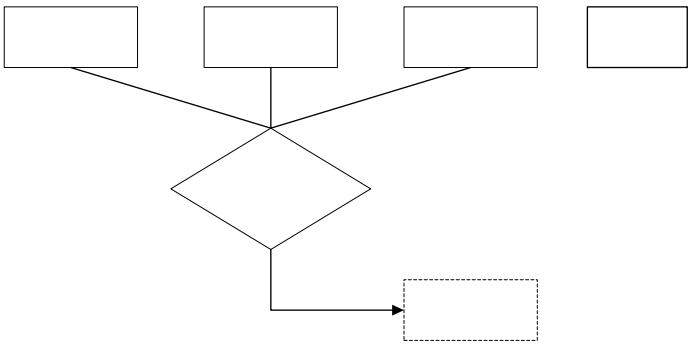


Рис. 2.23. Изображение агрегированного объекта

В качестве примера агрегированного объекта из рассматриваемой предметной области «Учебный процесс» изобразим «СДАЧУ ЭКЗАМЕНА» (рис. 2.23 б).

Изображение составных объектов

Для отображения составных объектов в ER-модели обычно не используются какие-либо специальные условные обозначения. Связь между составным объектом и составляющими его объектами отображается так же, как это было описано выше для простых объектов. Например, «ГРУППА» состоит из «СТУДЕНТОВ», и это будет отображено просто как связь 1:M между этими объектами.

Рекомендации по построению базовой ER-модели

В ER-модели должно быть отображено все, о чем идет речь в данной предметной области (во входных документах, в выходных документах и т.п.). После построения полной ER-модели необходимо определить состав хранимых показателей. Переход от ER-модели к даталогической модели должен производиться только для хранимых показателей.

При возникновении сомнений лучше принять решение о создании самостоятельного объекта, так как это в дальнейшем потребует меньших переделок модели.

Количественные характеристики всегда являются свойствами какого-либо объекта, и никогда – самостоятельными объектами.

При изображении предметной области надо стремиться отобразить информацию как можно более детально, так как это в дальнейшем даст возможность принять более обоснованные решения при проектировании структуры базы данных. Так, например, если «адрес», «ФИО» являются составными характеристиками, то желательно это отразить в ER-модели.

При решении вопроса о том, что следует отображать как обобщенный объект, приходится выбирать между двумя крайними вариантами: надо ли простой объект представить как обобщенный и надо ли два или несколько самостоятельных объектов «объединить» в обобщенный объект.

Обобщенный объект следует вводить в модель в том случае, когда надо подчеркнуть общность и различие категорий объектов, входящих в один класс, или в случае, если объекты разных подклассов участвуют в разных связях. Так, например, если для сотрудников мужского и женского пола фиксируются одни и те же свойства, эти объекты участвуют в одних и тех же связях, то не следует выделять соответствующие подклассы. Если же для студентов мужского поля фиксируются сведения о воинской обязанности, информация о том, прошли ли они срочную службу, занимаются ли они на военной кафедре и т.п., а для студенток эта информация не фиксируется, то разбивать класс объектов СТУДЕНТ на подклассы следует.

Естественно, что каждый подкласс может быть изображен в ER-модели как самостоятельный объект, а не как подкласс какого-то родового класса. Для того чтобы иметь больше информации о предметной области и, часто, сократить число элементов (свойств, связей) в ER-модели, в большинстве случаев лучше объединять подклассы в класс.

Одну и ту же ситуацию в предметной области можно представить в ER-модели разными способами. На рис. 2.24 изображены фрагменты ER-модели, отображающие факт знания сотрудником иностранных языков. Каждый из изображенных вариантов при всем их различии является правильным.

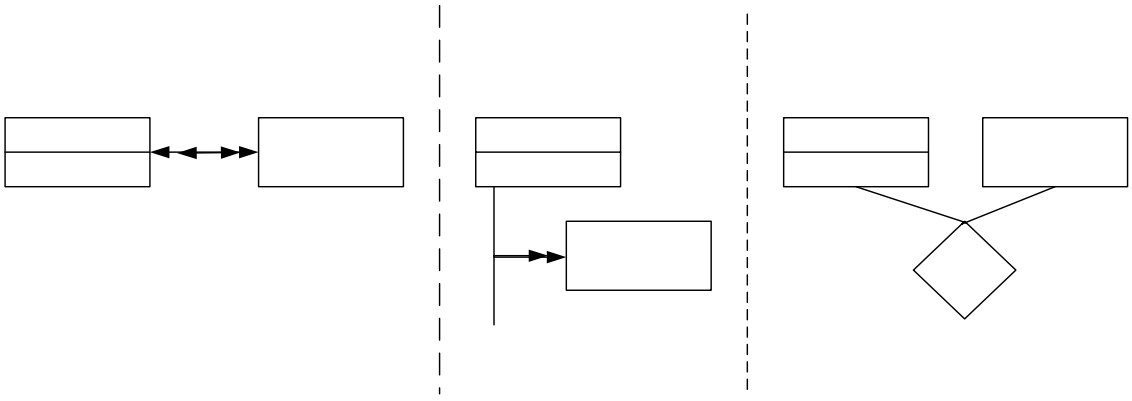


Рис. 2.24. Варианты изображения связи «СОТРУДНИК-ЯЗЫК»

В заключение данного раздела хотелось бы обратить внимание на некоторые наиболее часто допускаемые ошибки в процессе моделирования. Одной из таких ошибок является изображение «зависимых» друг от друга свойств в виде самостоятельных, не связанных друг с другом свойств. Так, на рис. 2.25 а) изображен неправильный вариант отображения информации о степени владения сотрудником тем или иным иностранным языком. Кроме правильного варианта, изображенного на рис. 2.25 б), возможен вариант, аналогичный изображенному на рис. 2.24 б), где «СТЕПЕНЬ _ ВЛАДЕНИЯ» будет свойством агрегированного объекта «ЗНАНИЕ _ ЯЗЫКА».

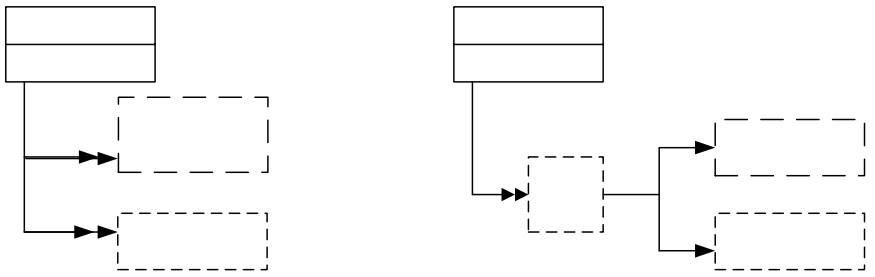


Рис. 2.25. Изображение информации о владении иностранными языками в базовой ER-модели (ошибочный и правильный варианты)

Следует напомнить, что нельзя устанавливать связь между свойством одного объекта и другим объектом или свойством. Нельзя непосредственно связывать между собой агрегированные объекты.

2.3. СРАВНЕНИЕ МЕТОДИК ПОСТРОЕНИЯ ER-МОДЕЛЕЙ

Анализ различий в нотациях ER-моделей

ER-модели очень широко используются в практике создания баз данных. Причем они применяются как при ручном, так и при автоматизированном проектировании. Чаще всего для представления ER-модели используются графические языки. Мы, в основном, будем рассматривать именно их. Изобразительные средства и методики графического представления ER-моделей, используемые в разных системах автоматизации проектирования, а также в разных литературных источниках, несколько отличаются друг от друга.

Далее мы рассмотрим особенности представления ER-моделей на примере нескольких наиболее известных систем автоматизации проектирования (CASE-системах): Prokit*WORKBENCH, Design / IDEF*, CASE ORACLE (Designer / 2000), PowerDesigner (новое название, используемое для последних версий системы S-Designor), ERWin, SILVERRUN, ERStudio и др., а также методологий, изложенных в некоторых литературных источниках.

*В системе Design / IDEF реализовано несколько разных методологий. Методология ER-моделирования носит название IDEF1X.

Можно выделить целый ряд признаков для сравнения систем моделирования предметных областей и последующего проектирования автоматизированных информационных систем. Часть из них одинаково применимы как для «ручного», так и для автоматизированного проектирования, другие признаки имеют значение только при использовании автоматизированных инструментальных средств проектирования.

Сначала рассмотрим аспекты, присущие как «ручным», так и автоматизированным способам создания ИС.

Прежде всего, остановимся на различиях в используемых изобразительных средствах. Можно выделить несколько категорий различий в изображении ER-моделей.

Несущественные различия, связанные с использованием разных условных обозначений для отображения одних и тех же элементов модели

Для обозначения простого объекта в разных системах используются прямоугольники, блоки с закругленными углами, овалы и т.д.

Наблюдаются различия в способе изображения характера связей между объектами (использование «стрелок», «лапок», «точек» и т.п. для отображения «множественного» конца связи). Такого рода различия можно продолжить. Они не накладывают никакого отпечатка на методологию построения концептуальной модели и алгоритм последующего перехода к даталогической модели.

Желательно, чтобы используемые обозначения были интуитивно понятными, излишне не загромождали модель, были просты в изображении. Часто предпочтения разработчиков в использовании тех или иных обозначений определяются просто привычкой. По возможности, следует стремиться к использованию стандартизированных или широко распространенных обозначений.

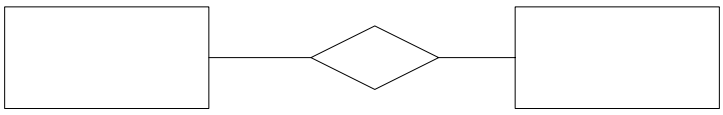


Рис. 2.26. Диаграмма Чена (условные обозначения)

В некоторых методиках при изображении связи между объектами в разъеме отображающей ее линии предлагается изображать ромб и внутри него или рядом с ним писать название связи (модель Чена) (рис. 2. 26). Другие способы изображения не требуют использования «ромбов», или требуют их изображения не всегда, а только при наличии определенных ситуаций в отображаемой предметной области. Если модель допускает связывание только пары объектов (т.е. поддерживает изображение только бинарных связей) и не допускает фиксирование дополнительных свойств для связей, то введение дополнительного обозначения только загромождает модель и не дает никаких дополнительных преимуществ.

Т.к. в разных методологиях проектирования используются разные условные обозначения для обозначения одного и того же «явления» в предметной области (т.е. фактически наблюдается синонимия в графическом языке описания предметной области), то некоторые системы автоматизации проектирования, например, ProKit*WORKBENCH, предоставляют пользователю возможность выбрать из множества допустимых обозначений те, которые ему больше нравятся или более привычны. В этой системе, к примеру, для обозначения вида связей между объектами могут использоваться условные обозначения, представленные на (рис. 2.27), а сама модель может изображаться в «стиле» Чена, Бахмана или в «реляционном» виде. Наиболее предпочтительный для пользователей системы стиль изображения должен быть выбран перед построением модели.

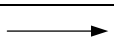
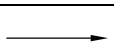
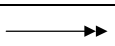
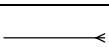
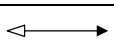
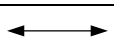
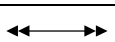
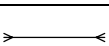
Вид отношения	Стиль отображения			
	1	2	3	4
1 : 1				
1 : M				
M : M				

Рис. 2.27. Изображение связей между объектами в системе Prokit*WORKBENCH

Рассмотрим некоторые из указанных выше различий более подробно.

Для отображения обязательности вхождения объектов в связь («класс принадлежности / членства») в разных системах моделирования используются разные условные обозначения. Так, в CASE ORACLE класс членства передается следующим образом: с той стороны связи, с которой элемент может не обязательно входить в связь, используется пунктирная линия, а там, где членство обязательное – сплошная линия. С учетом класса членства возможны типы отношений, представленные на рис. 2.28.

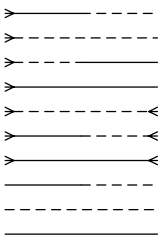


Рис. 2.28. Варианты изображения типа связей и класса членства в CASE ORACLE

В [17] для отображения класса членства используется маленький прямоугольник, который рисуется внутри блока, изображающего объект, и если класс членства объекта в связи является обязательным, то внутри этого прямоугольника ставится точка. Если класс членства необязательный, то точка ставится вне блока или вообще не ставится (рис. 2.29).

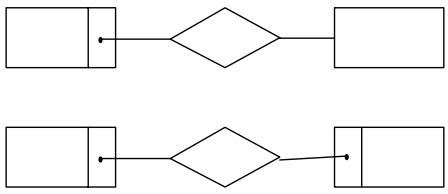


Рис. 2.29. Варианты изображения типа связей и класса членства (Г. Джексон)

Используемые в CASE ORACLE обозначения более удобны, т.к. если объект участвует в большом количестве связей, то дополнительные прямоугольники с точками становится неудобно располагать на рисунке.

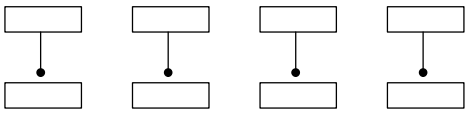
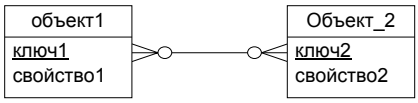


Рис. 2.30. Изображение класса членства в Design / IDEFIX

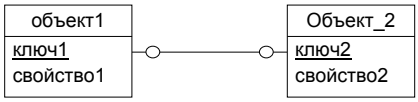
В Design / IDEF1X характер членства в связи изображается, как показано на рис. 2.30. Точка на конце линии обозначает множественную связь. Если около точки не стоит никакой буквы, то это означает нуль, один или более объектов в связи; P (positive) – один или много, Z (zero) – нуль или один, N (целое положительное число) – мощность связи в точности равна некоторому числу.

Наблюдаемые в Design / IDEF1X отличия от базовой и других рассмотренных выше моделей являются более существенными, чем просто различия в используемых условных обозначениях, т.к. это не только другой способ обозначения, но и другое «множество» возможных сочетаний «тип связи» – «класс членства». Поэтому эти различия надо отнести ко второму типу различий – существенным различиям, влияющим на процесс моделирования предметной области.

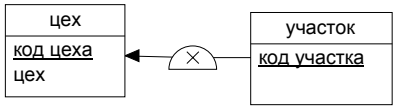
В PowerDesinger используются обозначения, представленные на рис. 2.31.



а) Связь M : M. Класс членства – обязательный с обоих концов.



б) Связь 1 : 1. Обязательный класс членства с обоих концов.



в) Изображение зависимой по идентификации сущности.

Рис. 2.31. Обозначения типов связи в PowerDesigner

В Paradigma Plus для характеристики связи используется термин Multiplicity (Множественность). Каждый конец связи имеет аннотацию, обозначающую множественность. Различают следующие разновидности множественности: «one» (один), «many» (много), «optional» (необязательный), «one or more» (один или больше), «zero or more» (нуль или больше).

На первый взгляд кажется, что подходы, используемые в PowerDesinger и Paradigma Plus, схожи между собой. Но в них есть существенное различие: В Paradigma Plus, также как и в нашей базовой модели, тип связи и кардинальность относятся к каждой стороне связи (т.е. характер связи задается в прямом и обратном направлении), в Design / IDEF – только в прямом. Поэтому в Design / IDEF нельзя выразить, например, связь M : M, в которой с одной стороны связи наблюдается обязательный класс членства, а с другой – необязательный.

Подход, принятый в базовой модели для отображения характера связи, представляется более продуктивным (экономичным и наглядным), так как для всего множества возможных сочетаний (с учетом направления связи их возможно 16) используется всего четыре условных обозначения.

Чаще всего для обозначения какого-либо явления в предметной области в конкретной методологии используется одно определенное обозначение. Но, как отмечалось выше, некоторые CASE-средства позволяют пользователю выбрать ту нотацию языка, которая является для него наиболее привычной.

Свойства объекта иногда не отображаются на той же схеме, что сами объекты и связи между ними, а перечень и описания этих свойств представляются отдельно. Часто описание свойств представляют в табличной или иной аналитической форме, а не в графическом виде.

Т.к. рассматриваемые различия не являются существенными, то легко выполнить преобразование из одной формы представления в другую, что и дают возможность автоматически делать многие CASE-средства.

Различия, приводящие к различиям в методике построения модели

Некоторые различия, также связанные со способом изображения тех или иных ситуаций, являются более существенными, приводящими к различиям и в методике построения модели, в адекватности изображения ПО и т.п. Например, в системе CASE ORACLE обобщенный объект изображается путем «вложения» блоков, обозначающих «видовые» объекты, внутрь блока, изображающего «родовой» объект. На рис. 2.32 показано изображение объекта «ЛИЧНОСТЬ», рассмотренного выше (см. рис. 2.22), в условных обозначениях, используемых в CASE ORACLE.

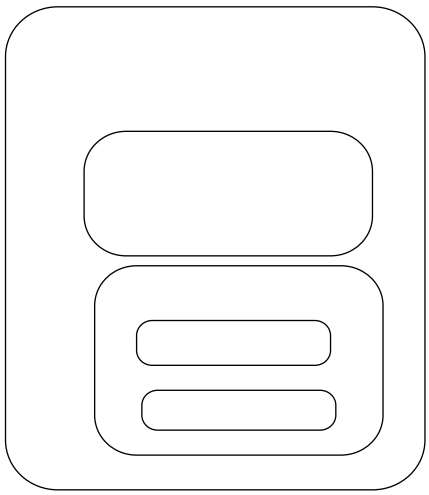


Рис. 2.32. Изображение обобщенного объекта в CASE ORACLE

Как следует из сравнения рисунков, изображение обобщенных объектов в сравниваемых методиках различаются не только по форме представления. Так, если объект классифицируется по разным признакам, то при использовании первого из рассмотренных способов изображения обобщенных объектов (см. рис. 2.22) наглядно видно, по какому признаку осуществляется классификация. Второй же способ изображения (см. рис. 2.32) не обеспечивает этого. Другими словами, предложенный в базовой модели способ изображения обобщенных объектов является семантически более содержательным, информативным, емким.

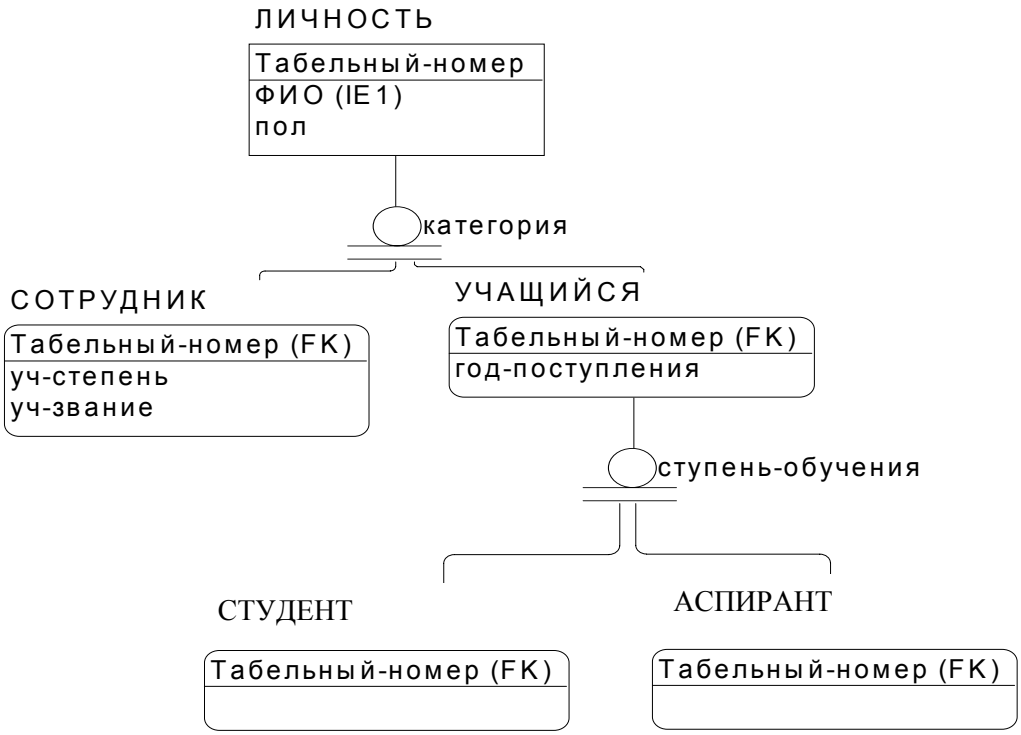


Рис. 2.33. Изображение обобщенного объекта ЛИЧНОСТЬ в IDEF1X

На рис. 2.33 изображен тот же обобщенный объект ЛИЧНОСТЬ с использованием синтаксиса IDEF1X. По своей семантике этот способ изображения ближе к предложенному нами базовому способу изображения ИЛМ. Разница заключается в том, что для сущностей-категорий и «общих» сущностей в IDEF1X используются одинаковые обозначения «сущности» (правда, в большинстве случаев «родовой» объект является независимой, а «видовой» – всегда зависимой от идентификации сущностью). Атрибут, по которому производится разбиение (дискриминатор), выносится из состава атрибутов обобщенного объекта и становиться именем, располагаемым рядом со значком дискриминатора. И хотя по форме представления изображение обобщенных объектов в нашей базовой модели и IDEF1X сильно различаются, но по «мощности» они идентичны, и различия между этими моделями можно отнести к рассмотренным выше различиям первого класса.

Предложенный нами в базовой модели способ обозначения кажется более четким: 1) он фиксирует внимание на то, что обобщенный объект представляет собой множество достаточно однородных объектов, имеющих общие свойства, а «дискриминатор» хоть и играет специфическую роль, но является одним из свойств объекта; 2) обобщенный объект изображается как единая сущность, а не совокупность множества отдельных объектов.

С точки зрения методологии способ изображения в IDEF1X фиксирует внимание на том, что видовые объекты – это самостоятельные объекты. В нашей же модели (как и в CASE ORACLE), наоборот, фиксируется, что обобщенный объект, включающий подклассы, является, тем не менее, объединяющей сущностью.

Близкий к IDEF1X, с точки зрения методологии отображения обобщенных объектов, является способ их изображения в CASE-средстве Vantage Team Builder. В нем для обозначения признака, по которому производится разбиение на подклассы, используется ромб, который соединен как с «супер-типом», так и с каждым из «подтипов». Линия, соединяющая ромб с «супер-типом», перечеркивается (рис. 2.34).

Пространственное размещение элементов ER-модели

ER-модель даже для небольшой и несложной предметной области включает в себя описание значительного числа компонентов и связей между ними. При этом встает проблема наглядности общей схемы. Эта проблема по-разному решается при ручном и автоматизированном построении инфологической модели. В автоматизированных системах чаще всего строится единое изображение ER-модели и используется прием масштабирования, когда, уменьшая или увеличивая масштаб изображения на экране, можно посмотреть как всю схему, так и отдельный ее фрагмент.

Многие CASE-средства позволяют выделять фрагменты из общей схемы и работать с ними как с самостоятельными компонентами модели, а также производить объединение отдельных фрагментов в единую схему.

Различные приемы используются и для того, чтобы уменьшить число пересечений линий на схеме. Так, в системе Prokit*WORKBENCH для этих целей допускается дублирование изображения объекта и размещение дубля рядом с тем объектом, с которым его надо связать. Для того чтобы показать, что это не новый объект, а дубликат уже изображенного в модели объекта, используется какое-либо условное обозначение, например, у соответствующих блоков, отражающих дубликат, отчеркивается уголок.

При ручном проектировании изобразить всю ER-модель в виде единой схемы обычно не представляется возможным. В этом случае можно порекомендовать следующий прием: изобразить и описать каждый объект самостоятельно, присвоить каждому объекту короткий код. Используя эти кодовые обозначения, для каждого объекта указать его связи с другими объектами в виде отдельных схем. При этом надо не упускать из вида, что, несмотря на такую дефрагментацию при изображении, модель является единой и связанной.

Многие CASE-системы позволяют выводить на экран информацию с разной степенью детальности (например, только названия сущностей, либо сущности и все их свойства, либо только ключевые атрибуты и т.п.). Такие возможности совсем не присущи ручным способам проектирования (хотя могут использоваться как методологический прием: проектирование сначала выполняется с минимальной степенью детализированности, а затем производится последовательное повышение степени детальности – обычный прием в структурном проектировании).

Отсутствующие возможности

Некоторые возможности, имеющиеся в одних системах или методиках, отсутствуют в других. В этих случаях возможны различные варианты:

а) для изображения ситуации, имеющей место в предметной области, используются возможности, предоставляемые данной методологией, но это требует применения определенных приемов, часто несколько искусственных, для их представления;
б) ситуация просто не отображается в модели.

Например, во многих системах инфологического моделирования предполагается, что свойства у объекта могут быть только единичными. В этом случае каждое множественное свойство следует представлять как самостоятельный объект и изображать связь между этим вновь введенным объектом и исходным объектом.

Так, в IDEF1X, как и в других широко известных CASE-системах, свойства объекта могут быть только единичные и всегда определенные (не условные). Если свойство может отсутствовать у каких-либо объектов, то надо выделять отдельные сущности, например, СЛУЖАЩИЙ _ ШТАТНЫЙ с атрибутом ОКЛАД, и ПОЧАСОВИК, не имеющий такого атрибута. Это приведет к необходимости выделения большого числа объектов и связей в ИЛМ, к снижению наглядности модели. Например, отдельные экземпляры объекта ЛИЧНОСТЬ могут иметь или не иметь ученое звание, ученую степень, год окончания вуза и многих других признаков. По каждому из этих признаков придется выделять подклассы (либо не фиксировать в модели, являются ли эти свойства «условными»).

Некоторые методики не вводят агрегированный объект как самостоятельную категорию. В этом случае сущность, соответствующая агрегированному объекту, изображается как простой объект; при этом пользователь должен предварительно определить идентификатор этого объекта (что является далеко не тривиальной задачей) и его свойства. Далее изображение будет зависеть от того, могут ли в используемой нотации быть изображены только бинарные связи, либо с объектом можно связать несколько разных объектов.

Кроме указанных сложностей при определении идентификатора агрегированной сущности, могут возникнуть и проблемы при переходе от ИЛМ к даталогической модели.

Если модель допускает изображение только двоичных (бинарных) связей, то проектировщик должен преобразовать n-арную связь в совокупность бинарных.

Связь типа «арк», использованная в базовой модели и имеющаяся в CASE ORACLE, отсутствует в большинстве других, рассматриваемых в данном пособии, систем. Это делает неудобным отображение тех ситуаций, где следовало бы использовать эту возможность (чаще всего при этом приходится вводить «лишнюю» сущность).

Если методика построения модели не предполагает фиксацию класса членства объекта в связи, то эта информация будет просто потеряна.

Ни в одной из известных нам систем моделирования нет понятия «составное свойство». Поэтому при моделировании предметной области проектировщик должен либо каждый из составляющих элементов составного свойства изобразить как отдельные самостоятельные свойства, либо изобразить это свойство без разделения на составляющие. И в том, и в другом случае часть информации о предметной области будет утеряна: в первом случае мы не увидим, что элементы составного свойства являются логически единым целым, а во втором – не увидим его состава.

В некоторых CASE-системах имеет место ситуация, когда какая-то конструкция допускается в системе как промежуточная. Например, в IDEF1X и CASE ORACLE связь M : M допускается как так называемое «неспецифическое» отношение. Его наличие разрешается на ранних стадиях разработки проекта, но в дальнейшем оно должно быть заменено на «специфическое» отношение (т.е. отношение типа 1 : M). Это достигается посредством введения в модель дополнительной третьей сущности и соединения с ней исходных сущностей связью типа 1 : M (другими словами одна связь M : M заменяется на две 1 : M). Это является недостатком подобных систем, так как, во-первых, не все целевые СУБД требуют такого преобразования (некоторые системы поддерживают отношение M : M в явном виде), и, во-вторых, если такое преобразование все-таки потребуется, система автоматизации проектирования вполне могла бы выполнить его автоматически на этапе даталогического проектирования. Даже если выполняется «ручное» проектирование, то указанное преобразование должно выполняться проектировщиком на стадии даталогического проектирования, а не при описании предметной области. Кроме того, при рассматриваемом преобразовании на стадии инфологического проектирования в IDEF вводится новая категория сущностей – сущности пересечения или ассоциативные сущности. Введение новых сущностей влечет за собой введение в ER-модель и дополнительных связей. Все это вместе взятое усложняет и без того не легкую задачу инфологического проектирования.

В предметной области могут быть сущности, идентификаторы которых являются зависимыми от идентификатора какого-то другого объекта. Например, если участки на предприятии нумеруются в пределах цеха, то идентификатор участка будет составным, включающим в себя код цеха и код участка. В инфологической модели можно ограничиться указанием этого составного идентификатора. Некоторые методики построения ER-моделей (например, IDEF1X, Prokit*WORKBENCH) предусматривают введение особых видов сущностей и особых видов отношений для отображения подобных ситуаций. Так, в IDEF1X сущность, для идентификации которой надо рассматривать ее отношение с другими сущностями, называется *зависимой от идентификатора сущностью*, и для ее изображения используется блок с закругленными углами (в отличие от сущности, не зависимой от идентификации, для обозначения которой используются прямоугольники). Для связи объектов, один из которых нужен для полной идентификации другого, вводится понятие *идентифицирующего отношения.* Для него также вводится свое условное обозначение. В IDEF1X для идентифицирующего отношения используется сплошная линия, а для неидентифицирующего – пунктирная.

Следует обратить внимание на то, что способ идентификации отражает не особенности предметной области, а языковые характеристики, а именно – способ именования объектов. Но лингвистические отношения тоже являются частью ИЛМ, и они могут быть отражены в ER-модели.

Если модель не использует в явном виде указание на способ идентификации объекта, то, как указывалось выше, эту ситуацию надо отразить просто путем соответствующего использования идентификаторов. Так, например, если на предприятии участки нумеруются в пределах цеха, то в качестве идентификатора объекта ЦЕХ следует использовать «КОД _ ЦЕХА*КОД _ УЧАСТКА» (рис. 2.34). Следует обратить внимание на то, что прямоугольник, соответствующий зависимой по идентификации сущности, в этом случае на сектора не делится.

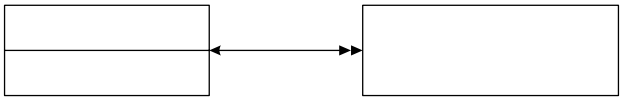


Рис. 2.34. Изображение зависимой по идентификации сущности в случае отсутствия специальных обозначений

Как отмечалось выше, ИЛМ включает в свой состав много разнообразных компонентов. Методологии моделирования и конкретные системы различаются ***полнотой и широтой охвата характеристик, отражаемых при описании предметной области***. Так, некоторые системы предусматривают описание запросов (в частности – ключей поиска), количественных характеристик классов объектов и запросов, ограничений целостности и т.д., другие – нет. Указанные описания иногда бывают объединены с ER-моделью (как, например, в Prokit*WORKBENCH) или оформляются как отдельные самостоятельные компоненты.

Различия в классификации объектов и отношений между ними

Как отмечалось выше при рассмотрении принципов инфологического моделирования, понятия «объект», «свойство», «отношение» являются относительными. Так, в предложенной нами базовой инфологической модели выделяются разные виды объектов: простые, составные, агрегированные, обобщенные. В некоторых системах моделирования такой классификации объектов нет, и вместо этого используются разновидности отношений. И тот, и другой подход имеет право на существование. Принципиальной разницы, в этих подходах нет.

Терминологические различия

Кроме различия в изображении тех или иных сущностей, в теории инфологического моделирования наблюдается расхождение в используемой терминологии. Например, в CASE ORACLE родовой объект называется супер-тип (syper-type), а видовой – под-тип (sub-type). Таких различий в терминологии можно привести много, но их фиксация не является сейчас нашей целью. Хотя обратить внимание на такие различия надо, чтобы при использовании конкретной системы не возникло ошибочное понимание происходящего. Так, в некоторых CASE-системах (например, Design / IDEF, PowerDesigner и др.) используется непривычная для теории проектирования БД трактовка понятия физического проектирования: под физическим проектированием в них понимается проектирование (описание) структуры БД для конкретной целевой СУБД.

Соглашения по именованию элементов ER-модели

При описании базовой модели мы использовали только имена классов объектов, идентификаторов и свойств объектов.

Правила задания имен сущностей различаются в разных методологиях моделирования ПО. Эти различия касаются того, что обязательно, а что необязательно именовать, где размещать имена и т.п.

Многие методики требуют обязательного именования связей. Так как связи являются двусторонними, то наименование связи будет меняться в зависимости от того, с какой стороны ее рассматривать. Поэтому часто в модели предлагается указывать оба эти названия (например, в системах CASE ORACLE, ProKit*WORKBENCH). В Design / IDEF1X обязательно надо указывать имя связи от «родителя» к «ребенку», имя связи в обратном направлении также можно указать, но это не является обязательным.

Для того чтобы было понятно, к какому из направлений связи какое название относится, принимают определенные соглашения о том, как располагать эти названия на схемах. Например, сверху линии помещать название, относящиеся к левой стороне связи, а под линией – к правой; или разделять эти имена наклонной чертой (первым помещается имя от «родителя» к «ребенку»)[10].

Наличие большого числа обозначений и подписей загромождают модель. Кроме того, само присвоение названий часто представляет некоторую трудность, что увеличивает трудоемкость инфологического моделирования. Поэтому в тех случаях, когда это не приводит к двусмысленностям и неясностям, и если это позволяет система, можно рекомендовать не использовать дополнительные особые обозначения (помимо линии, соединяющей эти объекты) и имена для связей.

[10] В некоторых автоматизированных системах наблюдается неудачная реализация рассматриваемых правил изображения. Так, например, в Design / IDEF имена связи даются в специальном окне, где сначала указывается имя родительской сущности и окно, для ввода имени связи от родительской сущности к порожденной и имя порожденной сущности. Ниже в этом же окне сущности представлены в обратном порядке, и можно указать имя связи от подчиненной сущности к родительской. Но на экране подписи выводятся справа налево (сначала имя «прямой» связи, а потом – «обратной») независимо от расположения объектов на экране. Если порожденный объект расположен правее основного, то расположение подписей на рисунке не соответствует расположению самих объектов.

Для CASE-средств важной является также связь вопросов именования элементов ER-модели и соответствующих им элементов даталогической модели. Так как ER-модель описывает предметную область и используется всеми категориями пользователей для обеспечения ее однозначного понимания, то желательно, чтобы имена давались на естественном национальном языке. Но в дальнейшем ER-модель используется для генерации описания структуры БД, а конкретные СУБД имеют разные ограничения на задание имен элементов БД. Хорошим решением этой проблемы является наличие в CASE-средстве возможности автоматически преобразовывать имена, использованные в ER-модели в соответствии с ограничениями целевой СУБД. К сожалению, далеко не все CASE-средства обладают такими возможностями. Некоторые CASE-средства позволяют при создании ER-модели каждому ее элементу давать несколько имен (одно – для использования в концептуальной модели, другое – в даталогической, или, как она называется во многих современных CASE-средствах, – физической модели и др.). Это, конечно, лучше, чем полное отсутствие возможности использовать разные имена, но не вполне соответствует сути подхода, заключающейся в том, что по исходному описанию могут генерироваться проекты для разных целевых систем, так как ограничения на допустимые имена в каждой из этих систем могут быть разными.

К сожалению, многие CASE-средства не только не обеспечивают, но даже и не контролируют правильность задания имен в полученной схеме БД. Это надо иметь в виду при построении ER-модели. При использовании некоторых CASE-средств возникают проблемы с использованием русского языка.

Назначение и состав ER-модели

Принципиально важным является решение вопроса о том, что же отражает ER-модель. Во многих методологиях и документации по конкретным CASE-средствам считается, что ER-модель является концептуальной моделью БД.

Основным отличием «базового» подхода, изложенного в учебнике, от многих методик, реализованных в современных CASE-средствах, является то, что в ней моделируется предметная область, тогда как в большинстве методологий отображается база данных (более того, именно реляционная БД). В предметной области нет понятия «ключ», «неспецифическое отношение» и т.п. (более того, эти понятия отсутствуют и в некоторых моделях данных, отличных от реляционных), которые используются во многих рассмотренных выше системах. Выбор ключа, кандидата ключа в излагаемой в пособии методике проектирования переносится в «даталогическое проектирование», а в большинстве существующих методик решается на уровне концептуального моделирования.

Кроме того, ни одна из анализируемых систем вообще не рассматривает проблему определения состава хранимых данных в БД (все, что имеется в ER-модели, «переносится» в схему БД; в лучшем случае, предоставляется возможность вручную отметить атрибуты, которые присутствуют только в концептуальной схеме и не переносятся в схему базы данных). Это служит еще одним косвенным подтверждением того, что ER-модель понимается, несмотря на декларации, все-таки как модель БД, а не модель предметной области.

Во многих методологиях проектирования ER-модель часто является фактически просто СУБД-независимым описанием логической структуры реляционной БД.

Как указывалось выше, описание запросов является необходимым элементом инфологического моделирования. Характеристика запросов существенно влияет на проектные решения при создании баз данных. В некоторых CASE-системах описание запроса частично добавляется к ER-модели.

Так, например, в Prokit*WORKBENCH вводится понятие «вход». «Вход» связывается с объектом ER-модели. Этому «входу» соответствует так называемый «Инверсный вход» (индекс). Соотносить «инверсный вход» с сущностью предметной области плохо нескольким причинам:

«инверсный вход» – это «вход» в файл; одному объекту, в принципе, может соответствовать много файлов, и наоборот;

запрос может относиться сразу к нескольким объектам (непонятно, как такого рода запросы можно отобразить в ER-модели);

запрос характеризуется не только ключом поиска, но и другими параметрами, в частности, какие атрибуты входят в ответ, как упорядочен ответ, какова частота возникновения запроса и т.п.; указание только «ключа поиска» не решает всех проблем по описанию запросов;

характер запроса влияет на проектирование БД: на разбиение информации по файлам, на выбор индексов; «входы» можно определять только после того, как определена структура БД, а не наоборот.

Дополнительные характеристики CASE-средств

ER-модель является сердцевиной систем проектирования, но, естественно, не единственной ее компонентой. Для CASE-систем[11], кроме общей характеристики используемой в них методологии ER-моделирования, существенной как для автоматизированного, так и для «ручного» проектирования, необходимо учитывать и специфические критерии, связанные с реализацией функций автоматизированного проектирования.

Ниже перечислены некоторые характеристики, которые следует учитывать при сравнении CASE-систем:

- число и перечень поддерживаемых целевых СУБД (а если не ограничиваться только рассмотрением вопросов проектирования БД, как делается в этом учебнике, то не только СУБД, но и других инструментальных средств);
- поддержка разных технологий организации БД (локальные, распределенные);
- поддержка коллективной работы при проектировании (в том числе: управление правами различных пользователей, ведение общего репозитория, возможность «консолидации» фрагментов моделей, созданных разными пользователями);
- построение концептуальной (ER-модели) по описанию структуры существующей БД – «ремоделирование» (реинжиниринг);
- поддерживаемые виды моделей (кроме концептуальной (ER-модели));
- автоматизируемые функции проектирования и степень их автоматизации;
- качество проектных решений; «жесткость» решений (возможность выбрать из нескольких альтернативных решений наиболее подходящий для данной ситуации, возможность «ручного» вмешательства в процесс проектирования на разных его этапах);
- надежность работы;
- документирование проекта;
- открытость системы (возможность «стыковки» с другими средствами автоматизации проектирования);
- удобство графического редактора;

[11] CASE-системы являются новым направлением в информационных технологиях. Так, первая версия CASE-инструментария фирмы ORACLE появилась в 1989 г.

- количественные ограничения (общее число объектов, число уровней вложенности для обобщенного объекта и др.);
- возможность автоматически оценивать объем памяти для проектируемой БД;
- возможность автоматической генерации хранимых процедур, процедур ввода данных и т.п.;
- наличие средств для моделирования хранилищ данных;
- требования к ресурсам компьютера;
- операционная среда;
- стоимость системы.

При оценке автоматизированных систем проектирования необходимо сравнивать не только язык модели и алгоритмы проектирования, но и саму реализацию: удобство интерфейса, степень автоматизации проектных решений, удобство использования и полноту справочной системы и др. характеристики именно программной реализации системы. Так, например, Help вDesign / IDEF имеет множество недостатков: помощь не контекстная; если действие по созданию какого-либо объекта не закончено (открыто какое-то окно), то помощь вообще недоступна, подписи, поясняющие назначение кнопок меню, высвечиваются только при «нажатии» на них. Если кнопка является «неактивной» (т.е. ее использование невозможно в данной ситуации), то и узнать назначение этой кнопки нельзя; далеко не все термины, сокращения, обозначения можно найти в справке; практически не представлены методические вопросы, нет рекомендаций по моделированию. В противоположность этому можно привести пример ERWin: система включает не только Help в привычном понимании, но и обширные методические материалы с большим количеством примеров.

Многие автоматизированные средства проектирования позволяют просматривать модель с разной степенью детализации: только обозначения сущностей и связей между ними или сущность + ключи, или сущность + ключи + внешние _ ключи, или сущность + все _ атрибуты. Наличие таких возможностей создает существенные удобства, особенно при создании больших и сложных моделей.

Другой существенной характеристикой CASE-средств является возможность получать различные подмодели из общей модели и, наоборот, обеспечивать интеграцию различных фрагментов в единую модель. Эти возможности могут быть полезны не только при коллективной разработке проекта несколькими разработчиками: очень удобно при обсуждении модели с конечными пользователями для каждого из них предоставлять только ту часть модели, которая имеет для него интерес; декомпозиция модели с возможностью корректной интеграции разных подсистем облегчает процесс проектирования.

Еще одним критерием для сравнения CASE-средств является степень проверки правильности построенных моделей. Следует обратить внимание, что ни одна система автоматизации проектирования, насколько бы развитой она ни была, не может гарантировать соответствия построенной концептуальной модели реалиям предметной области: это определяется только квалификацией разработчиков, их пониманием предметной области и умением адекватно отобразить ее в модели. Но наличие средств проверки моделей может помочь устранить ошибки, связанные, в основном, с невнимательностью, такие как: отсутствие идентификатора у сущности, отсутствие связи объекта с другими объектами, неправильное задание имен, отсутствие в модели информации, необходимой / полезной при дальнейшем проектировании (объемные характеристики для классов объектов и связей между ними и т.п.), противоречия в модели (особенно существенно при коллективной разработке) и другие.

Даже при использовании одной и той же методологии, как, например, в Design / IDEF и ERWin (обе используют методологию IDEF1X), способ их машинной реализации оказывается разным. Так, например, в Design / IDEF при описании атрибута указывается, что он является дискриминатором, при этом на схеме автоматически выводится знак дис-

криминатора и его имя. В ERWin не выделяется свойство, по которому производится разбиение, и «дискриминатор» автоматически не именуется. Это менее удобно. Хотя можно потом выделить соответствующую связь для редактирования и в описании супертипа указать, какой атрибут является дискриминатором. Тогда название этого атрибута повторяется рядом со знаком дискриминатора.

Анализируя CASE-средства, мы обращали внимание только на те характеристики, которые оказывают непосредственное влияние на проектирование структуры базы данных. Однако функции CASE-средств этим не ограничиваются. Многие системы позволяют задавать в модели ограничения целостности и генерируют программы (триггеры, хранимые процедуры), проверяющие эти ограничения при эксплуатации БД. Кроме того, CASE-средства могут генерировать программы ведения БД.

Многие CASE-средства позволяют экспортировать модели в другие системы и, наоборот, импортировать из других систем.

Отметим некоторые частные особенности конкретных CASE-средств.

B Design / IDEF (v. 3.5) при описании атрибута указывается, является ли он первичным ключом (Primary Key – PK), инвертированным входом (Inventory Entry – IE). Подход, при котором эти вопросы приходится решать на стадии моделирования Предметной Области, представляется не удачным, так как:

- «ключ» – понятие, относящееся к реляционной модели, а не к предметной области;
- что выбрать в качестве ключа, надо решать на стадии проектирования даталогической модели, а не инфологического моделирования, и желательно – автоматически;
- специалисту, строящему ER-модель, необходимо знать, что такое «ключ», «внешний ключ», «альтернативный ключ», уметь их определять и выбирать, что не так уж просто, особенно, если речь идет о составном ключе (как отмечалось выше, желательно активное участие специалистов предметных областей в разработке ER-модели, и требовать ото всех их знания таких специфических вопросов теории баз данных нежелательно);
- для нескольких атрибутов можно указать, что он является PK. На самом деле это означает, что каждый из таким образом обозначенных атрибутов является элементом составного ключа, а не самим ключом, что с точки зрения реляционной теории принципиально важно различать;
- термин «инвертированный вход» вообще очень не подходит для ER-модели, так как определяет способ организации хранения данных (для «инвертированных входов» строится индекс);
- система не проверяет ошибочные с точки зрения реляционной модели определения атрибутов. Так, можно объявить какой-то атрибут одновременно и первичным, и вероятным ключом, или объявить один атрибут первичным ключом (т.е. ключ простой) и его же включить в состав составного альтернативного ключа, или можно не задать длину атрибута, и при этом генерируется описание таблицы БД без указания длины поля.

Если в Design / IDEF вы изобразили объект и установили с ним связь 1 : M от другого объекта, то сразу в «подчиненный» объект мигрирует ключ из родительского объекта. Это же закладывается автоматически и в алгоритм проектирования, что далеко не всегда целесообразно (то же самое наблюдается и в ERwin и многих других системах). Недостатками такого подхода является то, что вместо инфологического моделирования на самом деле как бы сразу происходит проектирование отношения (таблицы БД), но: во-первых, связь 1 : M может быть отражена в БД разными путями (см. описание алгоритма), а, во-вторых, на уровне ER-модели происходит дублирование информации (наличие связи между сущностями отображается дважды: линией на схеме и повторением ключа «родительской» сущности).

В системе имеется целый ряд ошибок, связанных с преобразованием ER-модели в целевую схему БД. Так, например, если изобразить связь M : M (которая называется неспецифической), то схема (на SQL) сформируется, но с ошибками (ни в одину из таблиц не включается ключ связанной таблицы, не создается таблица связи, но в то же время создается внешний ключ, которому не соответствует первичный ключ). В документации по CASE-системам, в которых связь M : M является «неспецифической», сказано, что на поздних стадиях моделирования их надо преобразовать в «специфические». Но если Вы это забыли сделать, то, как сказано выше, описание на SQL сформируется и никаких сообщений об ошибках выдано не будет. Кроме того, наблюдается неполное соответствие типов данных, определяемых CASE-средством, реальному списку типов данных, поддерживаемому целевой СУБД.

Как отмечалось выше, разные целевые системы могут иметь разнообразные ограничения на допустимые имена используемых информационных единиц. В Design/IDEF имена полей в SQL-описании даются те, которыми в модели названы атрибуты (это могут быть, например, длинные русские слова или фразы с пробелами; в большинстве случаев задание таких имен недопустимо в целевых СУБД, но система не контролирует допустимость используемых имен информационных единиц).

В ERWin (в. 2.6) просматривать модель можно на разных «уровнях»: только объекты, объекты и свойства, связи с указанием кардинальности или без оной, с указанием ограничений целостности и без и др.. Для того, чтобы воспользоваться этой возможностью, надо в меню «Option» отметить позицию «Show Display menu» и в появившемся меню выбрать нужные параметры отображения.

При связывании сущностей, так же как и в Design/IDEF, ключ «родительской» сущности всегда «мигрирует» в «зависимую» сущность в качестве «неключевого» атрибута (при построении модели родительской становится та сущность, на которую «кликнули» мышью первой).

ERWin (в. 2.6) накладывает следующие ограничения на размер модели: до 500 сущностей и до 1500 атрибутов.

При изображении подтипов используется иной подход, нежели в Design/IDEF1X, а именно: изображаются сначала все объекты, как родовой, так и видовые. Потом они соединяются соответствующей связью. В отличие от Design/IDEF1X не выделяется свойство, по которому производится разбиение, и «дискриминатор» первоначально никак не именуется. Но, как указывалось выше, можно потом (щелкнув на значке дискриминатора, выйти в соответствующий редактор) указать, по какому атрибуту идет разделение, и имя этого атрибута станет именем около значка дискриминатора.

В ERWin версии 3.5 введены дополнительные возможности:

- поддержка многомерного моделирования;
- возможность оценивать объем таблиц и индексов в БД;
- словарь доменов;
- расширение опций для генерации баз данных;
- поддержка последних версий некоторых серверов баз данных;
- и некоторые другие.

Редактор Design/IDEF интуитивно более понятен и привычен (например, для того, чтобы перемещать линию, надо щелкнуть на ней мышью и перемещать метки концов или середины линии; чтобы ввести новый атрибут в сущность, достаточно двойного щелчка на этой сущности). Но более привычно не всегда означает, что это, безусловно, лучше. Так, при перетаскивании линий в ERWin нельзя «оторвать» концы линий от границы знака сущностей, которые они связывают, что скорее является достоинством, чем недостатком.

Еще одно популярное CASE-средство – ***S-Designor (новое название – PowerDesigner)***. Система включает в себя следующие модули:

ProcessAnlyst – потоки данных и бизнес-процессы; позволяет создать функциональную модель информационной системы.

DataArchitect – концептуальная и физическая модель; позволяет создавать концептуальную модель, основываясь на информации функциональной модели из ProcessAnlyst, с последующей генерацией физической модели для более чем 40 различных СУБД. Возможна обратная генерация физической модели из существующей БД.

AppModeler – генерация приложений; позволяет генерировать отдельные объекты или целые приложения для PowerBulder, Visual Basic и ряда других инструментов.

MetaWorks – поддерка коллективной работы.

В версии PowerDesigner 6.0 появился новый модуль WarehouseArchitect, предназначенный для проектирования специализированных информационных моделей для хранилищ данных, включая многомерные информационные модели. WarehouseArchitect, помимо поддержки традиционных СУБД, позволяет генерировать физические модели для специализированных серверов Sybase IQ и Red Brick.

Имеется объект «Домен» (Domain), который предназначен для определения типа данных. Посредством использования Доменов можно определить допустимые значения для каждого из элементов данных. Для Домена можно указать тип данных, длину, допустимые значения и некоторые другие свойства.

С одним Доменом можно связывать несколько элементов данных. Возможность задания Доменов, таким образом, сокращает описание концептуальной модели и позволяет стандартизировать представление однотипных данных.

Использование графических ПП для изображения ER-моделей

При изображении ER-моделей можно воспользоваться различными графическими средствами. Так, широко распространенное средство Visio Professional поддерживает множество нотаций ERD (рис. 2.35).

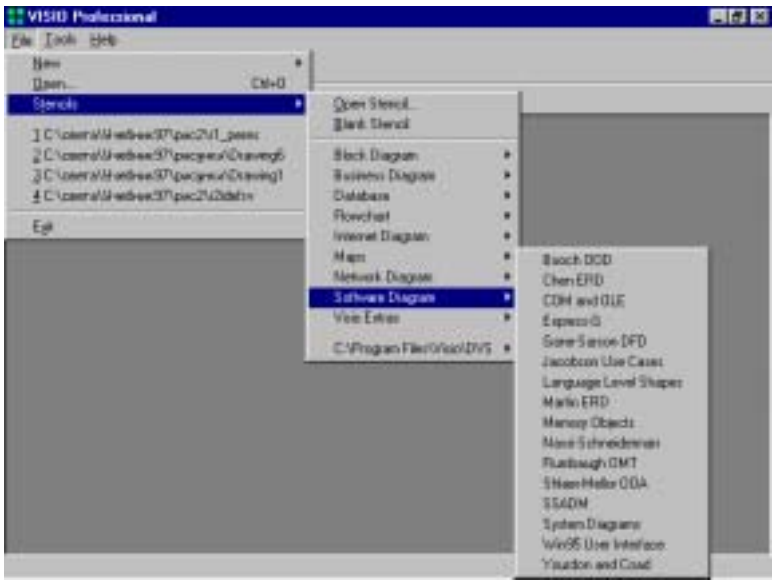


Рис. 2.35. Нотации, поддерживаемые в Visio Professional

В Microsoft Visio ER-модели можно рисовать, используя панель меню ***Stensils/Database/Entity relationship*** (рис. 2.36)**.**

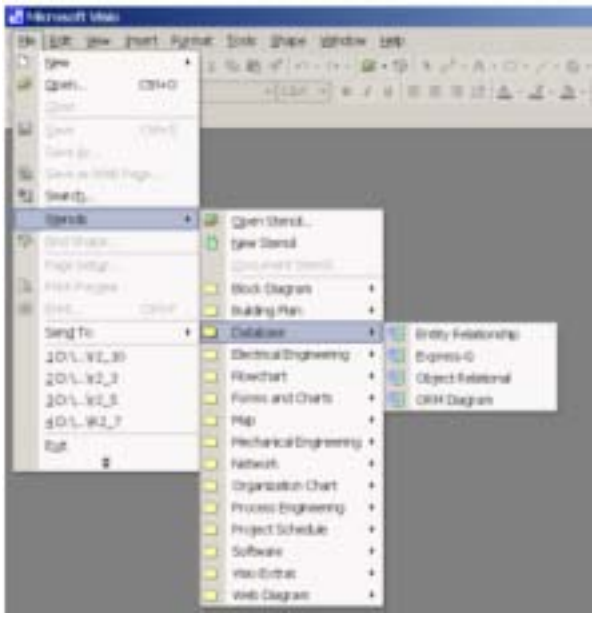


Рис. 2.36. Microsoft Visio. Меню Database

Условные обозначения, используемые при этом (рис. 2.37), соответствуют методологии IDEF1X**.**

Рис. 2.37. Условные обозначения, используемые при построении ER-диаграмм (Microsoft Visio)

Позиции меню ***Stensils/Softwere*** (рис. 2.38) в Microsoft Visio соответствуют разновидностям моделей в UML**.** Эти модели напрямую для проектирования структуры базы данных не используются.

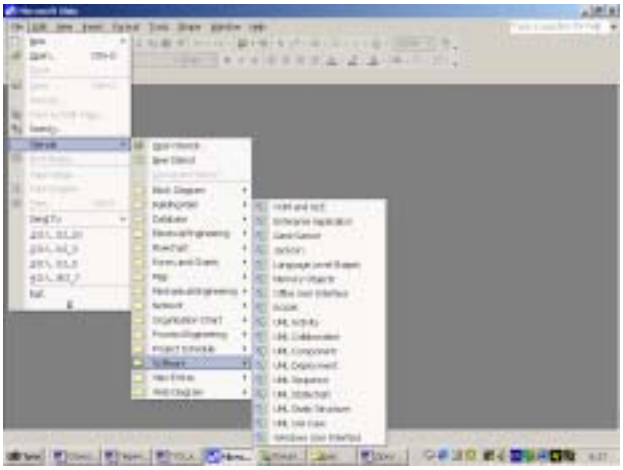


Рис. 2.38. Microsoft Visio. Меню Softwere

Особенности методологии построения ER-модели в зависимости от изобразительных средств и алгоритмов, заложенных в CASE-средствах

Несмотря на, что при использовании большинства CASE-средств на основе ER-моделей структура целевой базы данных определяется автоматически, требования к квалификации проектировщиков базы данных не только не уменьшаются, но и, напротив, увеличиваются. Специалисты должны не только в совершенстве владеть методологией моделирования предметных областей с использованием языковых средств конкретной системы автоматизации проектирования, но и понимать алгоритм проектирования, заложенный в данном CASE-средстве, а также владеть теорией проектирования баз данных. В противном случае полученное проектное решение может оказаться не просто недостаточно эффективным, но и вообще не соответствовать отображаемой предметной области и содержать ошибки.

Как мы видели, выразительные возможности языковых средств представления ER-моделей в разных CASE-системах, а также «ручных» методиках моделирования отличаются друг от друга, и иногда очень существенно. Это, безусловно, откладывает отпечаток на методику построения ER-модели в каждой конкретной среде.

Принципиально важным является решение вопроса о том, что же отражает ER-модель. Во многих методологиях и документации по конкретным CASE-средствам считается, что ER-модель является концептуальной моделью БД. Основным отличием «базового» подхода, изложенного в пособии, от многих методик, реализованных в современных CASE-средствах, является то, что в ней моделируется предметная область, тогда как в большинстве методологий отображается база данных (более того, именно реляционная БД). В предметной области нет понятия «ключ», «неспецифическое отношение» и т.п. (более того, эти понятия отсутствуют и в некоторых моделях данных, отличных от реляционных), которые вводятся во многих рассмотренных выше системах. Выбор ключа, кандидата ключа в излагаемой в пособии методике проектирования переносится в «даталогическое проектирование», а в большинстве существующих методик решается на уровне концептуального моделирования.

В ERWin вводится понятие «логический уровень» (что у нас называется ИЛМ (или концептуальная модель) и физический уровень (что соответствует ДЛМ), и считается, что одними и теми же средствами (в рамках одной модели) можно отразить и тот, и другой уровень. На самом деле это не совсем так (т.к. часть из свойств, присущих конкретной СУБД, должна быть определена все-таки не в ER-модели, а при генерации схемы).

Кроме того, ни одна из анализируемых систем вообще не рассматривает проблему определения состава хранимых в БД данных (во многих из этих систем вообще все, что имеется в ER-модели, «переносится» в схему БД). Это служит еще одним косвенным подтверждением того, что ER-модель понимается, несмотря на декларации, все-таки как модель БД, а не модель предметной области.

В предлагаемой в данном учебнике методологии проектирования в ER-модели должны быть отображены все элементы, о которых идет речь в исследуемой предметной области. В процессе проектирования БД выполняется ряд шагов, в том числе и определение состава показателей, хранимых в базе данных. В базу данных могут переноситься не все атрибуты, присутствующие в ER-модели. Так, производные показатели часто не включаются в БД.

Большинство современных CASE-систем, таких как Design / IDEF, ERWin и др., не содержат блоков, осуществляющих определение состава атрибутов, подлежащих хранению в БД (хотя даже некоторые из более ранних систем автоматизации проектирования предусматривали выполнение таких шагов). Некоторые системы, например, ERWin, хотя и не позволяют автоматически определять состав показателей, хранимых в БД, но дают возможность пометить атрибуты, которые присутствуют только в концептуальной модели.

В связи с этим ER-модель, подлежащая преобразованию в модель целевой БД, при использовании CASE-систем, в которых не только нельзя автоматически определять состав хранимых атрибутов, но и нельзя их хотя бы пометить, должна содержать только те данные, которые должны храниться в БД.

Во многих методологиях проектирования ER-модель часто является фактически просто СУБД-независимым описанием логической структуры реляционной БД.

Методология построения ER-модели зависит от изобразительных возможностей, заложенных в язык описания ER-модели, а также алгоритма перехода от ER-модели к модели базы данных в среде конкретной СУБД, заложенного в CASE-средстве. Кроме того, естественно, методология зависит от особенностей модели БД целевой СУБД. Последнее оказывает влияние не только на методологию, но и на сам язык моделирования предметной области. К сожалению, практически все современные средства ER-моделирования ориентированы на реляционные модели данных и их использование для других моделей приведет к некорректному результату.

Несмотря на то, что при использовании большинства CASE-средств на основе ER-моделей структура целевой базы данных определяется автоматически, требования к проектировщикам базы данных не только не уменьшаются, но и, напротив, возрастают: специалисты должны не только в совершенстве владеть методологией моделирования предметных областей с использованием языковых средств конкретной системы автоматизации проектирования, но и понимать алгоритм проектирования, заложенный в данном CASE-средстве, владеть теорией проектирования баз данных. В противном случае полученное проектное решение может оказаться не просто недостаточно эффективным, но и вообще не соответствовать отображаемой предметной области и содержать ошибки. Алгоритмы проектирования, заложенные в CASE-средствах, как правило, не публикуются. Поэтому, прежде чем приступить к реальному проектированию в среде конкретного CASE-средства, следует поэкспериментировать, посмотреть, как те или иные конструкции ER-модели и их сочетания преобразуются в модель БД.

Как известно, выразительные возможности языковых средств представления ER-моделей в разных CASE-системах, а также «ручных» методиках моделирования отличаются друг от друга, и иногда очень существенно. Это, безусловно, откладывает отпечаток на методику построения ER-модели в каждой конкретной среде. При построении ER-модели надо реальную предметную область отобразить с помощью тех изобразительных средств, которые предоставляет конкретное средство моделирования.

В п. 2.2 приведен язык для построения ER-модели, методика его использования, а далее (п. 3.3) будет рассмотрен алгоритм перехода от ER-модели к логической модели реляционной базы данных. Принципиальным моментом в предлагаемом подходе является то, что в процессе ER-моделирования строится именно модель предметной области, а не базы данных, а уже на основе этой модели и некоторых других компонентов инфологической модели проектируется структура базы данных.

В предлагаемом в данном пособии подходе в ER-модели должны быть отображены все элементы, о которых идет речь в исследуемой предметной области. В процессе проектирования БД выполняется ряд шагов, в том числе и определение состава показателей, хранимых в базе данных. В базу данных могут переноситься не все атрибуты, присутствующие в ER-модели. Так, производные показатели часто не включаются в БД.

Как отмечалось выше, многие современные CASE-системы, например, Design/IDEF, не содержат блоков, осуществляющих определение состава атрибутов, подлежащих хранению в БД (хотя даже некоторые из более ранних систем автоматизации проектирования предусматривали выполнение таких шагов), а также средств, позволяющих хотя бы просто указать, подлежит ли данный элемент ER-модели переносу в даталогическую модель БД. В связи с этим ER-модель, подлежащая преобразованию в модель целевой БД, при использовании подобных CASE-систем должна содержать только те данные, которые должны храниться в БД. В таких ситуациях рекомендуется строить две ER-модели: первая будет отображать предметную область в целом, безотносительно к тому, что будет храниться в базе данных, а что – нет, а вторая содержать только те элементы, которые будут храниться в БД.

Изображение ER-модели, подлежащей преобразованию в модель целевой БД, будет зависеть и от других особенностей алгоритма преобразования инфологической модели в даталогическую. В данном пособии предложен многоальтернативный алгоритм проектирования, в котором одной и той же конструкции ER-модели может соответствовать несколько проектных решений, которые принимаются проектировщиком на основе анализа многих факторов. К сожалению, далеко не все CASE-системы позволяют проектировщику выбирать проектное решение из нескольких альтернативных.

В предлагаемом в данном пособии подходе именно на алгоритм перехода от ER-модели к модели БД возлагаются задачи проектирования структуры базы данных, а ER-модель и другие компоненты инфологической модели должны содержать информацию, достаточную для обоснованного принятия проектного решения. Специалист, строящий ER-модель, в этом случае вообще может ничего не знать о модели и структуре базе данных. Именно такой подход и отвечает сущности инфологического моделирования – описание предметной области без относительно к используемым в дальнейшем СУБД. Во многих CASE-системах ER-модели ориентированы только на реляционные базы данных; в них каждому объекту ER-модели соответствует таблица базы данных. В этом случае построение ER-модели мало чем отличается от проектирования реляционной базы данных.

Заложенные в систему проектные решения оказывают влияние на методологию построения ER-модели. Так, например, если алгоритм проектирования позволяет не каждому объекту ставить в соответствие таблицу, то при построении ER-модели при решении вопроса о том, какой из сущностей предметной области ставить в соответствие отдельный объект в ER-модели, а когда отображать его в виде характеристического свойства, можно порекомендовать выделять отдельный объект, не содержащий никаких атрибутов, кроме идентификатора, если он участвует в нескольких связях. В этом случае ER-модель получается менее громоздкой, т.к. содержит меньше элементов. При использовании CASE-средств такое решение еще более удобно, чем при ручном проектировании, т.к. при этом происхо-

дит автоматическая миграция соответствующих атрибутов. Придерживаться же данной выше рекомендации при построении ER-модели при использовании тех CASE-систем, в которых каждому объекту ER-модели соответствует таблица в реляционной базе данных, нельзя. Поэтому необходимо уже на стадии ER-моделирования решить, каким сущностям ставить в соответствие таблицы, а каким – нет, и в качестве объектов ER-модели изображать только те сущности, которым будут соответствовать отдельные таблицы.

Рассмотрим еще некоторые примеры влияния языка и алгоритма проектирования на построение ER-модели. В базовой ER-модели введено понятие **«условное свойство»**, которого нет в других методологиях ER-моделирования. Оно означает, что свойство может присутствовать не у всех объектов данного класса. При изображении таких ситуаций в методологиях типа IDEF1X, где нет соответствующего понятия, возможно несколько вариантов:

1. Условное свойство изобразить как обычный атрибут.
2. Объект, обладающий условным свойством, изобразить как обобщенный объект (при этом условное свойство будет принадлежать «видовому» объекту).
3. Выделить «обладание свойством» в отдельный объект и при установлении связи этого вновь введенного объекта с исходным объектом соответствующим образом определить характеристики этой связи.

В тех CASE-системах, которые позволяют выбирать проектное решение при преобразовании ER-модели в модель целевой СУБД (как, например, в PowerDesigner), лучше использовать второй вариант. В тех системах (как, например, Design/IDEF), в которых каждому «видовому» объекту ставиться отдельная таблица, следует сначала принять решение, как вы хотите хранить информацию в БД, а только затем выбирать вариант отображения предметной области в ER-модели.

Большинство CASE-систем содержат изобразительные средства для отображения **обобщенных объектов**, хотя как используемые для этих целей условные обозначения, так и алгоритм отображения этих конструкций в даталогическую модель отличаются от системы к системе. Так, например, в методологии IDEF1X можно производить классификацию объектов по нескольким независимым признакам. В CASE ORACLE это сделать невозможно (можно изобразить только строго иерархическую, а не фасетную классификацию). Это, безусловно, скажется на подходе к моделированию предметной области: при невозможности отобразить многоаспектную классификацию в большинстве случаев придется подклассы изображать как самостоятельные объекты.

Понятие множественного свойства также отсутствует в большинстве CASE-систем. Для изображения каждого множественного свойства приходится использовать отдельный объект, зависящий по идентификации от основного объекта, обладающего этим свойством. Атрибут, соответствующий множественному свойству, должен быть отмечен как ключевой.

Многие CASE-системы (Design/IDEF, ERWin, CASE ORACLE) позволяют изображать **связь M : M** на начальных этапах построения ER-модели, но перед выполнением этапа генерации схемы БД эта связь должна быть преобразована путем введения дополнительной связующей сущности и связей 1 : M с ней. В Design/IDEF и ERWin при изображении связи M : M нельзя указать класс членства объектов в этой связи. Если класс членства является необязательным хотя бы с одной из сторон связи, то приходится либо применять искусственные приемы, чтобы адекватно отразить характер связи, либо терять часть информации о предметной области. В качестве такого искусственного приема для систем, базирующихся на методологии IDEF1X, можно предложить следующий: класс объектов, **класс членства** которого в данной связи «необязательный», «делится» только на один подкласс (естественно, что в этом случае тип подкласса будет «incomplete» – *неполный*), и уже этот «видовой» объект используется в связи.

В базовой модели введено понятие «имя объекта» – это то, что называет объект. Имя может быть как уникальным, так и не уникальным. Это фиксируется в модели. Во всех остальных проанализированных методологиях выделяется атрибут / совокупность атрибутов, соответствующих первичному ключу. Если у объекта есть несколько даже уникальных имен, то только один из них должен быть выбран в качестве идентификатора уже на стадии концептуального моделирования. Другими словами, проблема выбора ключа реляционной таблицы переносится на стадию инфологического проектирования, что не совсем оправдано.

Понятие «ключ» и «имя объекта» – это не всегда одно и то же. В качестве идентификатора может быть выбрана совокупность атрибутов, не обязательно принадлежащих имени (например, добавление даты и места рождения при идентификации личности в поисковых системах).

ER-модели служат не только непосредственной цели проектирования базы данных, но и выполняют другие, более общие функции, такие как понятное, полное, формализованное описание предметной области и использование его для обсуждения как с заказчиками / пользователями проектируемой системы, так и с разработчиками разной специализации. Поэтому можно рекомендовать при проектировании ИС с использованием таких CASE-систем, которые не позволяют определять, какие из объектов / свойств будут храниться в базе данных, строить две модели: 1) модель предметной области, содержащей все объекты, как подлежащие, так и не подлежащие хранению в БД; использующую изображение множественных связей 2) концептуальную модель БД, которая затем автоматически будет преобразована в модель целевой БД.

Недостаточные изобразительные возможности языковых средств, используемых при изображении ER-моделей, приводят, с одной стороны, к обеднению этих моделей, а с другой, как это не парадоксально звучит, усложняют процесс построения ER-модели и делают эти модели более громоздкими.

Это надо запомнить:

Для построения ER-модели, которая корректно будет преобразована в модель целевой БД, при использовании CASE-средств необходимо не только хорошо понимать отображаемую предметную область и уметь адекватно описывать ее с использованием языковых возможностей данной системы, но и хорошо представлять себе алгоритмы преобразования, заложенные в используемой системе.

2.4. ИСПОЛЬЗОВАНИЕ DESIGN/IDEF ДЛЯ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

Построение er-модели при использовании Design/IDEF

Общая характеристика

Design/IDEF является комплексной системой автоматизации проектирования ИС. Она объединяет в себе несколько методологий, каждая из которых предназначена для построения моделей определенного типа. Для построения ER-модели используется методология IDEF1X.

Нотация языка ER-моделирования, алгоритм проектирования целевой реляционной модели и, как следствие, подход к моделированию существенно отличаются от базовой модели. Изобразительные средства нотации языка ER-моделирования в Design/IDEF более бедные, чем в базовой модели. Процесс построения ER-модели в Design/IDEF практически сводится к описанию реляционной модели данных другими изобразительными средствами (каждому объекту ER-модели в Design/IDEF соответствует таблица в реляционной базе данных). В связи с этим рекомендуется сначала изучить главу 3 данного пособия, а потом, используя эти знания, строить ER-модель в нотации Design/IDEF.

Укрупненная схема работы с Design/IDEF при построении ER-модели изображена на рис. 2.39.

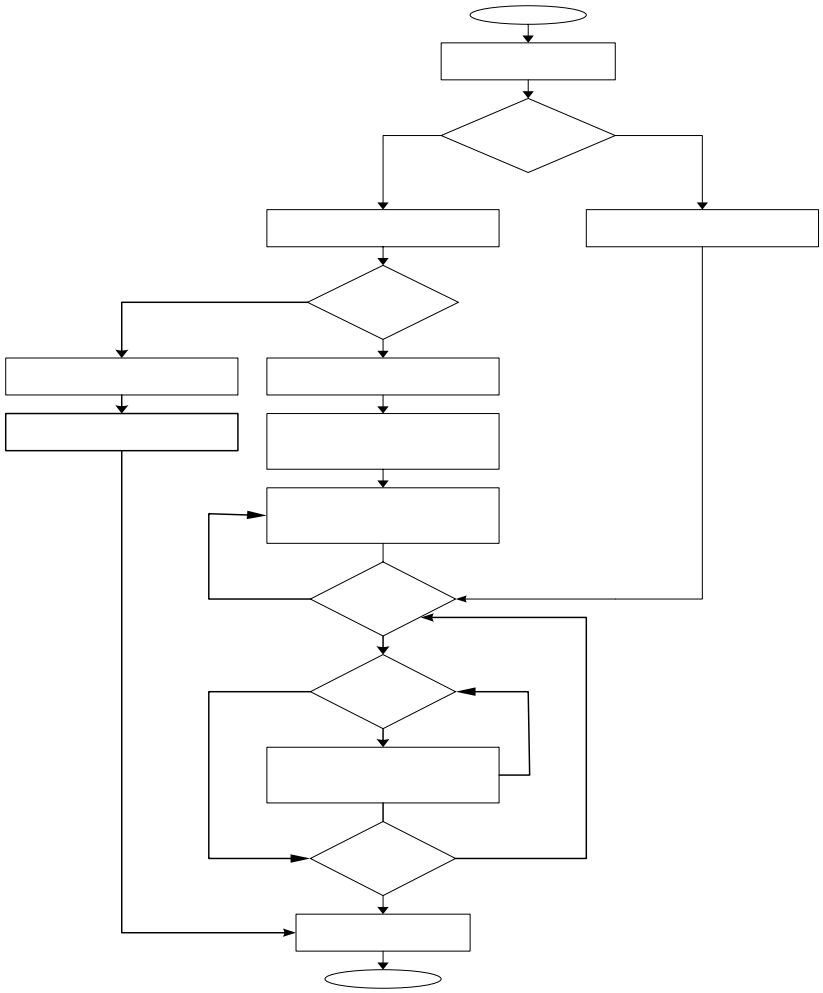


Рис. 2.39. Укрупненная схема создания ER-модели в Design/IDEF

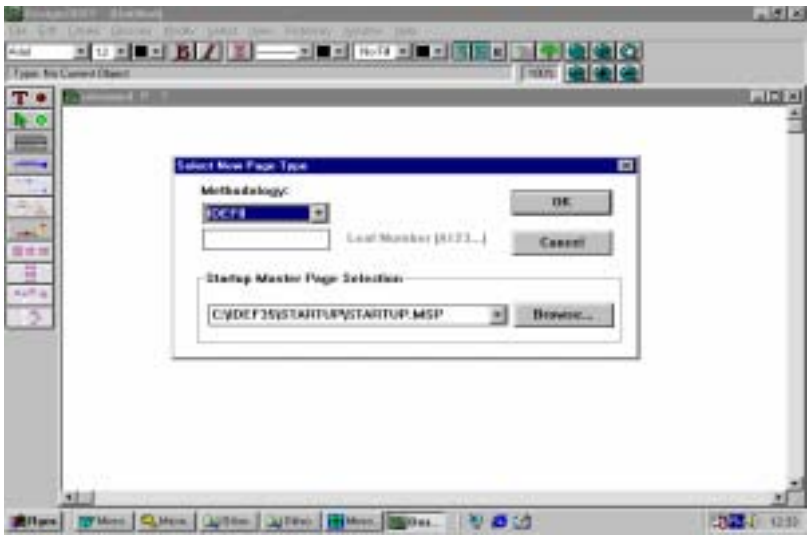


Рис. 2.40. Вид экрана после загрузки системы Design/IDEF и выбора команды **File ➢ New**

Выберите требуемую
методологию

После загрузки Design/IDEF и выбора команды ***File/New*** экран имеет вид, изображенный на рис. 2.40. В окошке **Methodology** будет высвечиваться название той методологии, с которой работали в предыдущем сеансе. Если это не методология IDEF1X, то следует нажать на кнопку в правой стороне этого окошка, из ниспадающего списка выбрать требуемую методологию (рис. 2.41).

Рис. 2.41. Выбор методологии

Начальный экран проектирования после выбора методология IDEF1X изображен на рис. 2.42.

Рис. 2.42. Начальный вид экрана проектирования после выбора методологии IDEF1X

После этого можно приступить к построению ER-модели, а можно сначала выбрать целевую СУБД – ту СУБД, для которой ведется проектирование базы данных. Для этого следует выбрать команду ***Edit/Set Options.*** В появившемся окне **IDEF Options** следует активизировать переключатель IDEF1X рис. 2.43 и в окошке **Tager Datadase** выбрать нужную СУБД из ниспадающего списка поддерживаемых системой СУБД. Выбор целевой СУБД окажет влияние на список допустимых типов данных при описании атрибутов. На методологию построения модели этот выбор влияния не оказывает. Выбор целевой СУБД

можно произвести на любом этапе проектирования. При этом типы данных будут автоматически преобразованы в соответствующие им типы в новой целевой СУБД.

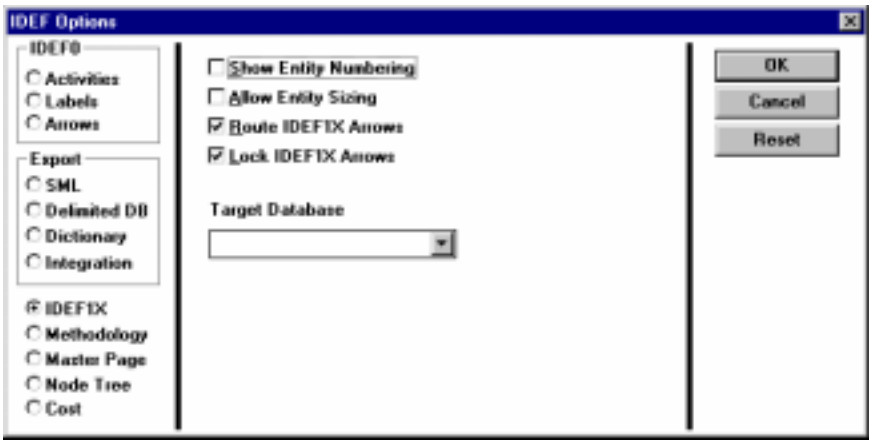


Рис. 2.43. Окно задания опций

Описание сущности

Построение новой ER-модели можно начать только с создания новой сущности. Для того чтобы построить новую сущность, надо нажать третью кнопку сверху (прямоугольник, разделенный на два сектора) на панели инструментального меню, изображенного слева на экране, (или выполнить команду **Create/Entity** или нажать **Ctrl + E**) и, позиционировав курсор с «прикрепленным» к нему квадратом в нужном месте экрана, нажать левую кнопку мыши. На экране появится окно определения сущности **(Define Entity)**, изображенное на рис. 2.44. Система автоматически присваивает каждой создаваемой сущности уникальный идентификатор (**Entity ID**). В окошке **Name** следует указать имя сущности. Кроме основного имени можно задать еще и **Aliases** (псевдоним) – имя, используемое в качестве синонима.

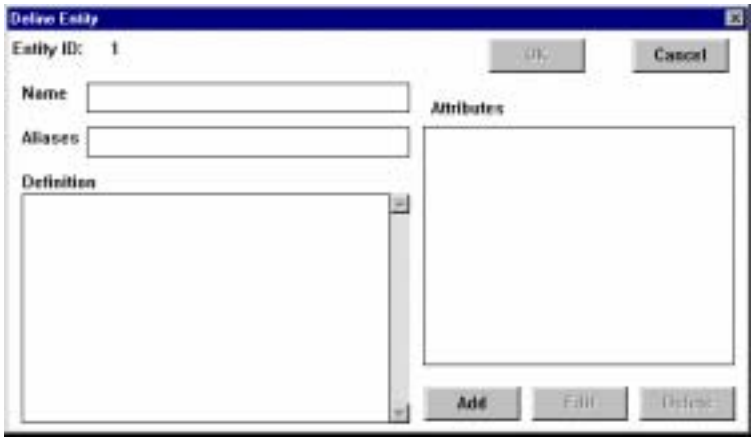


Рис. 2.44. Экран «Определение сущности»

В окошке **Definition** можно дать описание сущности. Такие описания важны для документирования модели, уточнения терминологии, используемой в проекте, но на проектирование структуры базы данных они влияния не окажут.

После этого следует перейти к описанию атрибутов сущности, для чего следует нажать кнопку **Add** в окне определения сущности. При этом на экране появится окно определения атрибута (Define Attribute), изображенное на рис. 2.45.

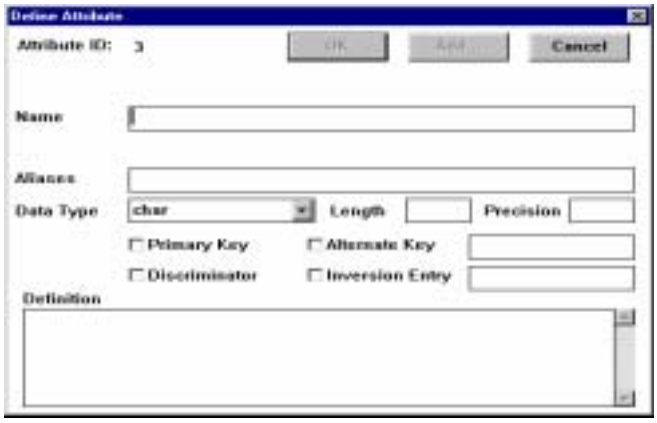


Рис. 2.45. Окно определения атрибута

В окошке **Name** следует записать наименование атрибута, в окошке **Data Type** выберите тип, присущий данному атрибуту. Список доступных типов полей зависит от выбранной целевой СУБД. Как известно, некоторые типы полей имеют фиксированную длину, и для них нет нужды указывать длину (Length); для других типов полей должна указываться длина; для числовых полей с дробной частью надо указывать точность (Precision).

В реляционной теории есть понятия «ключ» и «вероятный ключ». Эти понятия характеризируют не предметную область, а именно таблицу реляционной базы данных. При преобразовании ER-модели в схему реляционной базы данных в Design/IDEF каждой сущности ставится в соответствие таблица реляционной базы данных, и при описании сущности требуется определить, какой атрибут (или несколько атрибутов) выбран в качестве первичного ключа. Эти атрибуты должны быть помечены галочкой в переключателе **Primary Key.**

Для нескольких атрибутов можно указать, что он является PK. На самом деле это означает, что каждый из таким образом обозначенных атрибутов является элементом составного ключа, а не самостоятельным ключом, что с точки зрения реляционной теории принципиально важно различать.

Если есть несколько альтернативных (вероятных) ключей, то надо все остальные описать, как альтернативные. Вероятные ключи помечаются галочкой в переключателе **Alternate Key.** Т.к. вероятных ключей может быть несколько, то в окошке справа от свойства **Alternate Key** необходимо поставить порядковый номер. Если альтернативный ключ является составным, то все составляющие его атрибуты надо пометить одним и тем же номером в окошке рядом с переключателем Alternate Key.

При проектировании баз данных учитывается множество разных факторов, и в том числе характер запросов. Если по какому-либо полю часто осуществляется выборочный поиск, то по такому полю обычно производят индексирование. Такие проиндексированные поля иногда называют «инверсным входом». При описании сущности именно такие атрибуты следует пометить галочкой в переключателе **Inversion Entry**. Альтернативных ключей, также как и вероятных, может быть несколько; поэтому в окошке справа от свойства **Inversion Entry Inversion Entry** необходимо поставить порядковый номер.

Как известно из рассмотренного в п. 2.2 материала, объекты бывают не только простыми, но и обобщенными. То свойство, по которому классы делятся на подклассы, в Design / IDEF называются «***дискриминатором***» (Discriminator). При описании соответствующего свойства следует это отметить галочкой в переключателе **Discriminator.**

Назначение свойств **Name** и **Aliases** в окне определения атрибута аналогичны соответствующим характеристикам, задаваемым при описании самого объекта в окне определения сущности.

После описания очередного атрибута следует опять нажать кнопку **Add** и задать описание следующего атрибута. После того как определены все атрибуты описываемого объекта, нажмите **OK**.

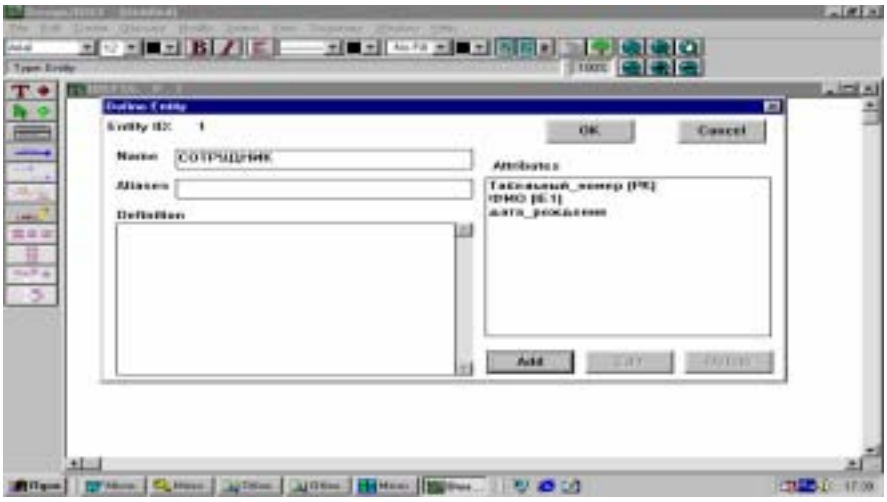


Рис. 2.46. Экран описания объекта (пример)

На рис. 2.46 приведен фрагмент описания сущности ЛИЧНОСТЬ. Если на этом завершить описание атрибутов и нажать **OK**, то графическое представление описанного объекта будет выглядеть так, как это показано на рис. 2.47.

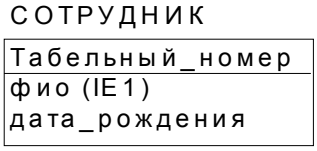


Рис. 2.47. Пример графического представления объекта (вариант 1)

В общем виде графическое представление сущности можно представить в виде, изображенном на рис. 2.48: сущность изображается в виде прямоугольника, разделенного на два сектора – в верхнем записываются ключевые атрибуты, в нижнем – все остальные. Рядом с атрибутами, являющимися альтернативными ключами, в скобках ставятся буквы (AK).

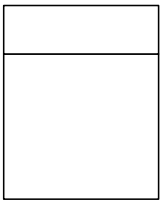


Рис. 2.48. IDEF1X ERD. Обозначение объекта

Если Вы уверены, что ФИО вместе с ДАТОЙ_РОЖДЕНИЯ является уникальной совокупностью атрибутов и их следует описать как вероятный ключ, то графическое представление такого объекта будет выглядеть так, как это представлено на рис. 2.49. Несмот-

ря на то, что в данном примере ФИО вошло в альтернативный ключ, его следует оставить и как инверсный вход, так как по ФИО часто осуществляется поиск.

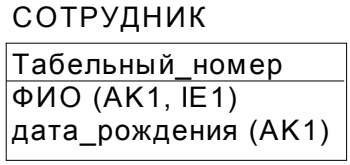


Рис. 2.49. Пример изображения объекта (вариант 2)

Предыдущий пример следует рассматривать именно как технический пример создания составного альтернативного ключа и не более того. В реальной жизни вряд ли целесообразно создавать такой альтернативный ключ. Надо понимать, что не все потенциальные альтернативные ключи реляционной таблицы следует обозначать в ER-модели. Это надо делать только тогда, когда есть необходимость контролировать уникальность альтернативного ключа.

Альтернативных ключей у одной сущности может быть несколько, как, например, на рис. 2.50. Обратите внимание на разницу в изображении сущностей на рис.2.49 и 2.50. В первом случае сущность имеет один составной альтернативный ключ, во втором – два простых.

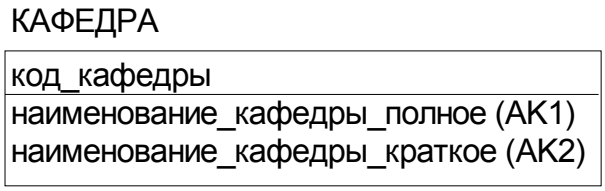


Рис. 2.50. Изображение сущности

Описание связи

После того как были определены хотя бы две сущности, можно задавать связи между ними. Для определения связи можно нажать четвертую кнопку сверху (линия с точкой на конце) на панели инструментального меню, изображенного слева на экране (или выполнить команду **Create**/**Relationship**, или нажать **Ctrl + R**). После этого курсор с «прикрепленной» к нему линией со стрелочкой надо позиционировать на значке того объекта, от которого должна начинаться связь, и, не отпуская левую клавишу мыши, протянуть связь до того объекта, на котором она должна оканчиваться.

После этого появится окно определения связи (**Define Relationship**) (рис. 2.51). В первой строке этого окна (**Entity**) указывается имя «исходного» объекта, от которого идет связь; в третьей строке с таким же названием Entity – имя объекта, к которому направлена связь. В окошке **Relationship** обязательно должно быть указано имя связи в прямом направлении (в нашем примере это связь от объекта ГРУППА к объекту СТУДЕНТ и назвать ее можно, например, «включает» или «состоит»). В окошке **Inverse** можно, но не обязательно, задать имя связи в обратном направлении (для нашего примера это может быть имя «учится»). Переключатель *Separate Lines* становится активным только в том случае, если имя обратной связи задано. Этот переключатель определяет только то, как на экране располагаются надписи прямой и обратной связи.

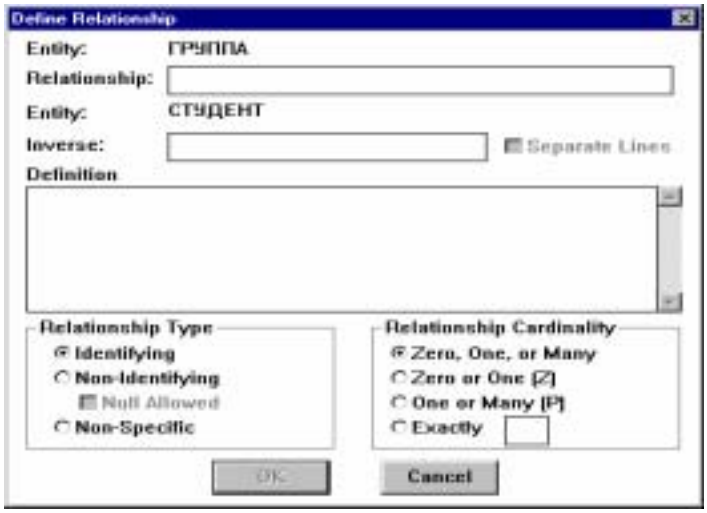


Рис. 2.51. Окно описания связи между объектами

Для каждой связи должен быть определен тип связи (**Relationship Type)** и кардинальное число (**Relationship Cardinality**).

В Design/IDEF различают следующие типы связи:

- Identifying (Идентифицирующая).
- Non-Identifying (Неидентифицирующая).
- Non-Specific (Неспецифическая).

Идентифицирующая и неидентифицирующая связи представляют собой связи 1 : M. Первая из них используется, если связь направлена к независимой по идентификации сущности (см. п.2.1.1), вторая – к зависимой. Неспецифическая связь представляет собой связь M : M между объектами. Она названа неспецифической потому, что реляционная модель не поддерживает связи M : M. В Design/IDEF неспецифические связи могут быть использованы на начальных этапах моделирования. Перед генерацией схемы базы данных ER-модель надо преобразовать и заменить неспецифические связи специфическими путем введения дополнительных связующих объектов. Если этого не сделать, то схема БД будет не адекватна предметной области.

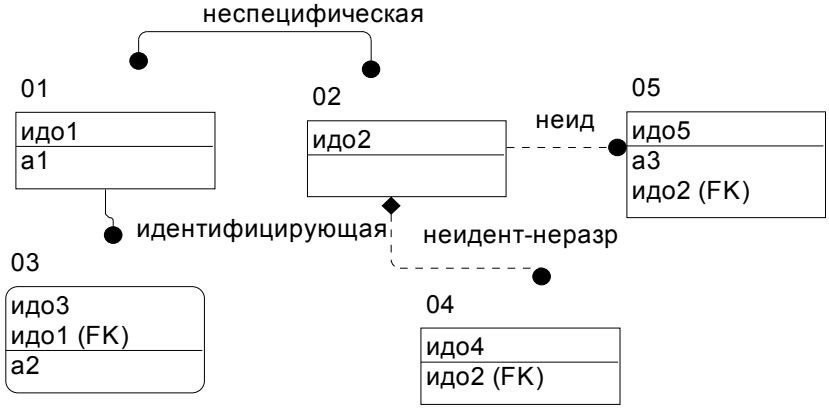


Рис. 2.52. Виды связей в Desing/DEF1X

На рис. 2.52 изображены все возможные виды связи в Design/IDEF. Идентифицирующая (сплошная линия) и неидентифицирующая (пунктирная линия) связи отличаются не только видом линии, но и тем, куда мигрирует ключ исходной сущности: в первом слу-

чае он становится частью ключа «целевой» сущности, а во втором – неключевым атрибутом (но в том и в другом случае он является внешним ключом).

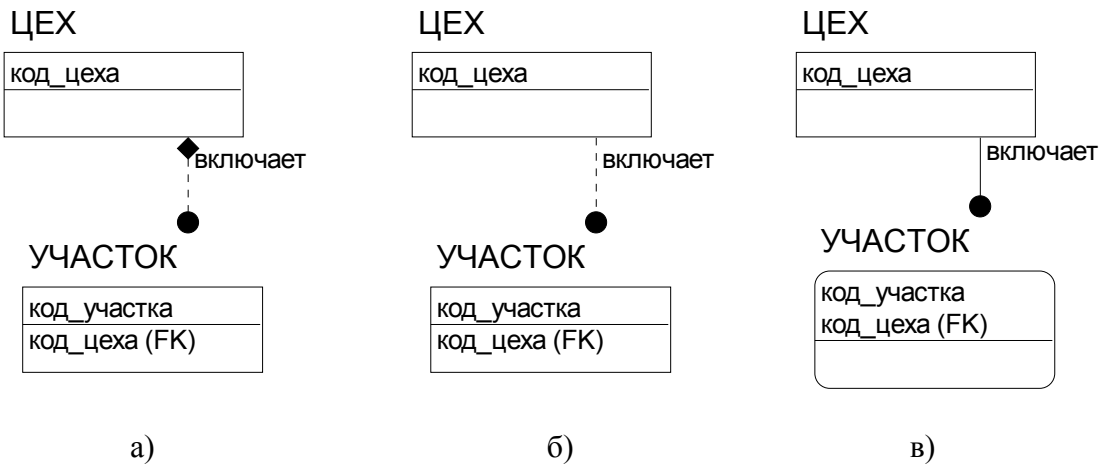


Рис. 2.53. Примеры изображения связи между объектами: а) неидентифицирующая (не разрешено «пустое» значение), б) неидентифицирующая («пустое» значение разрешено), в) идентифицирующая

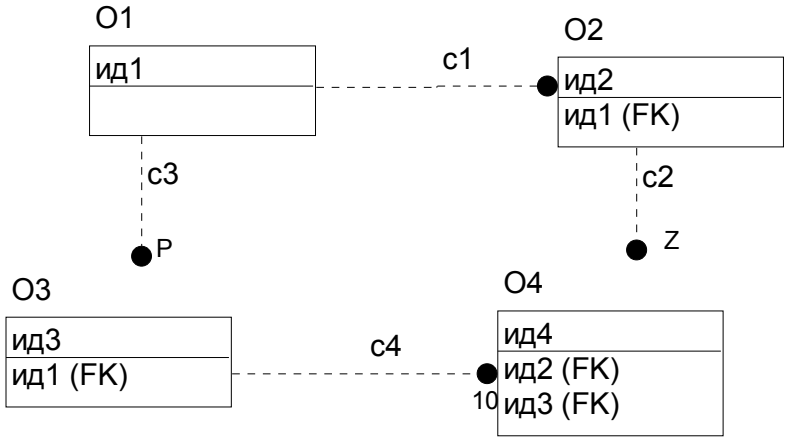
На рис. 2.53 приведены примеры использования идентифицирующей и неидетифицирующей связи. Выбор варианта будет определятся тем, как обозначаются участки в конкретной предметной области. Если выбрана неидентифицирующя связь, то возможно задание дополнительной характеристики этой связи – разрешаются ли «пустые», неопределенные значения внешнего ключа (**Null Allowed**). На рис. 2.53 изображены ситуации и с разрешенными, и с запрещенными Null-значениями. Но это сделано для того, чтобы показать, как графически различается изображение в каждой из этих ситуаций; разрешенное Null-значение означает, что возможно наличие участков, не приписанных ни к какому цеху, что в реальных предметных областях, скорее всего, недопустимо.

Если к сущности идет идентифицирующая связь, то такая сущность называется зависимой по идентификации. Зависимая по идентификации сущность, как мы видим, изображается прямоугольником с закругленными углами. Ключ зависимой по идентификации сущности чаще всего бывает составным, одним из элементов которого является ключ исходного объекта (рис. 2.53 в). Идентифицирующую связь нельзя провести, если у исходного объекта не задан ключ. Знак зависимой по идентификации сущности нельзя внести в схему непосредственно; углы закругляются «автоматически», при объявлении соответствующей связи.

Кроме указания типа связи для каждой связи задается кардинальное число (рис.2.51), указывающее количественные характеристики связи. В IDEF1X могут быть выражены следующие «мощности» связей:

- Каждый экземпляр сущности родителя может иметь нуль, один или более одного связанного с ним экземпляра сущности-потомка (*Zero, One or Many*).
- Каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка; (*Zero or One [Z]).*
- Каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка (*One or Many [P]).*
- Каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка (*Exactly*).

В последнем из рассмотренных случаев, кроме того, что в него устанавливается переключатель, в окошке следует указать число. Например, если набираются группы для обучения в компьютерных классах, и занятия не начинаются, пока не наберется 10 человек, но и большее число учащихся недопустимо, то следует выбрать вариант *Exactly* и поставить число 10.



c1 – Zero, One or Many;
c2 – Zero or One [Z];
c3 – One or Many [P]);
c4 – Exactly.

Рис. 2.54. IDEF1X. Виды «мощности» связи

На рис. 2.54 приведены графические представления всех допустимых в IDEF1X типов «мощности» связей. На рис. 2.55 приведены примеры использования разной кардинальности связи.

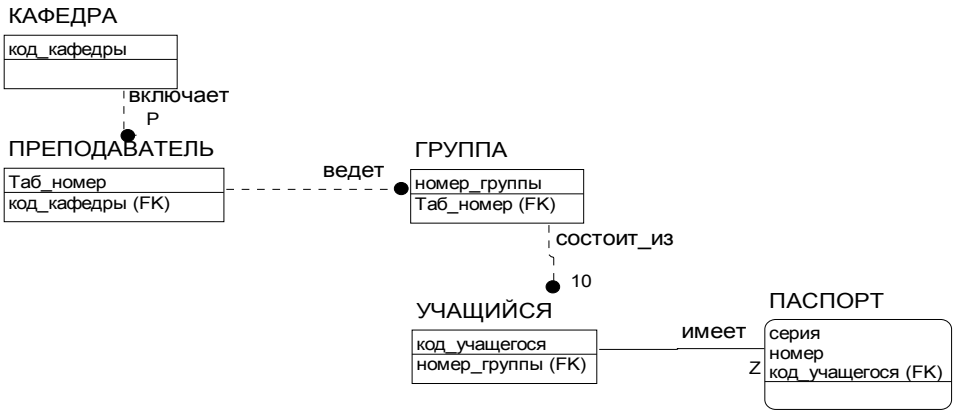


Рис. 2.55. Фрагмент ER-модели
(пример использования разной кардинальности связи)

Описание обобщенного объекта

В Design/IDEF нет специального графического значка для обозначения обобщенного объекта. Но возможность отображать обобщенные объекты есть. При изображении обобщенного объекта следует изобразить сначала «родовую» сущность. Для нее следует указать все идентификаторы изображаемого объекта, общие атрибуты, присущие всем объектам класса, а также тот атрибут, по которому класс разбивается на подклассы (такой атрибут следует обозначить как дискриминатор). На рис. 2.56 изображен вид экрана описания родовой сущности, а на рис. 2.57 – соответствующее ему графическое представление. Как видно из рис. 2.57, дискриминатор на схеме изображается окружностью с прилегающими к ней двумя параллельными линиями. При объявлении дискриминатора всегда первоначально появляется такой знак. Он означает «полное» разбиение класса на подклассы (т.е. каждый элемент класса обязательно относится к одному из выделенных подклассов). Если это не так, то надо выделить значок дискриминатора, после чего воспользоваться переключателем «полный-неполный дескриминатор» (шестая кнопка сверху в инструментальном меню или воспользоваться позицией меню ***Create/Toggle Discriminator***).

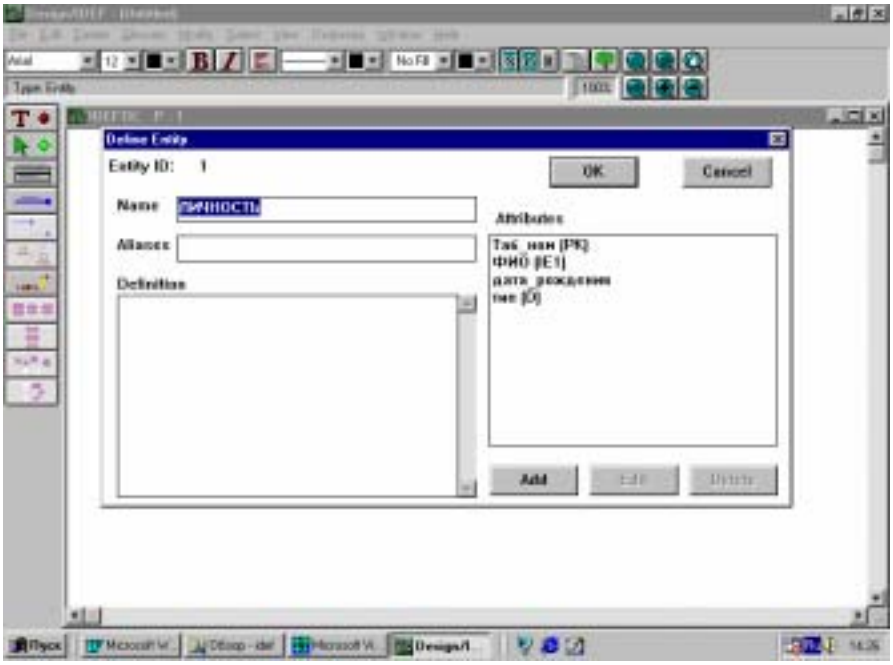


Рис.2.56. Экран описания «родового» объекта

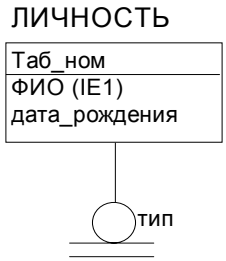


Рис. 2.57. Графическое представление «родового» объекта

Далее необходимо изобразить сущности, соответствующие «видовым» объектам. При этом ключевые поля описывать не надо. В качестве атрибутов этих сущностей надо описывать те атрибуты, которые присущи именно этому подклассу (рис. 2.58). После этого надо провести связи от значка дискриминатора к значкам видовых объектов, после чего схема примет вид, изображенный на рис. 2.59. При этом ключи «родового» объекта автоматически мигрируют в «видовые» объекты, а значки видовых объектов станут иметь закругленные углы.

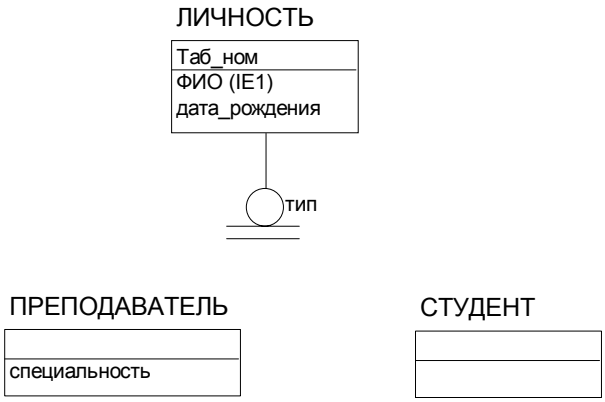


Рис. 2.58. Графическое представление «видовых» сущностей (промежуточный этап)

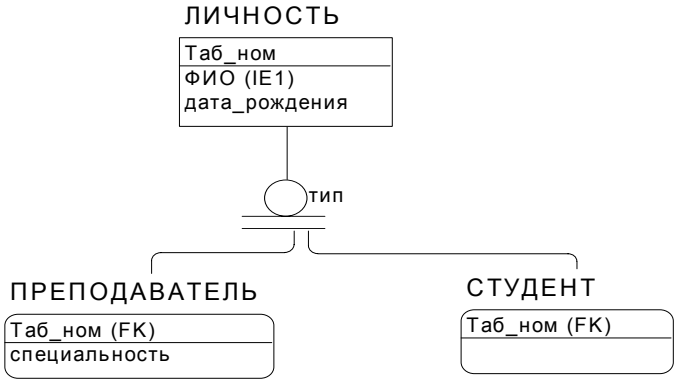
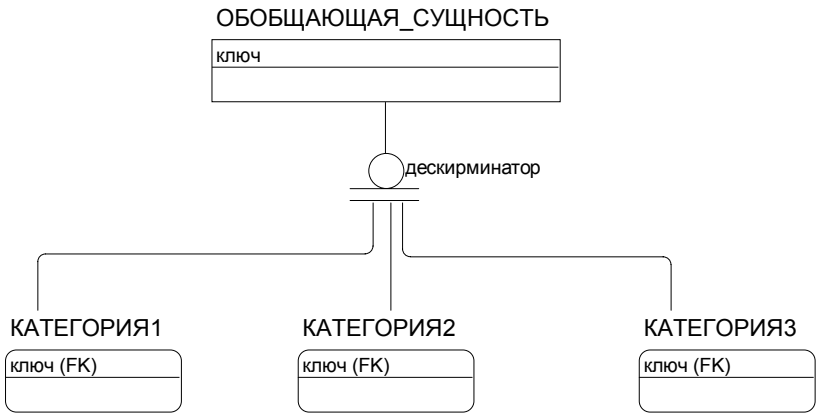


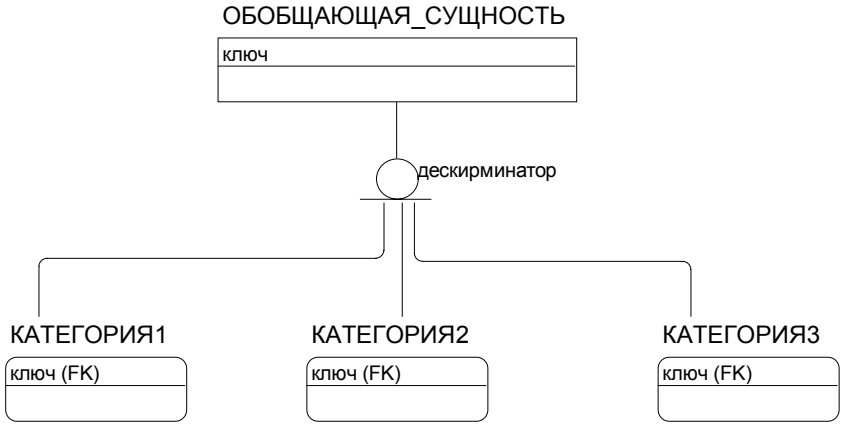
Рис. 2.59. IDEF1X ERD. Пример изображения обобщенного объекта

Изображение обобщенного объекта может выглядеть по-разному (рис. 2.60). Если дискриминатор имеет вид окружности (рис. 2.60 а), подчеркнутой двойной линией, то это означает, что подклассы в совокупности составляют полный класс. Если дискриминатор имеет вид окружности (рис. 2.60 б), подчеркнутой одинарной линией, то это означает, что подклассы в совокупности не составляют полный класс. На рис. 2.60 в представлен вариант, показывающий возможность классификации сущностей по разным не соподчиненным признакам.

а)



б)



в)

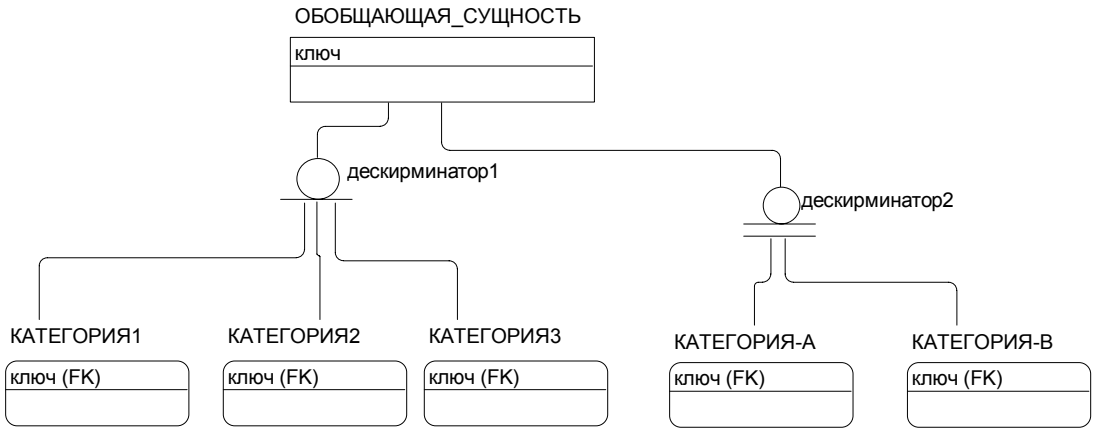


Рис. 2.60. Варианты изображения обобщенного объекта

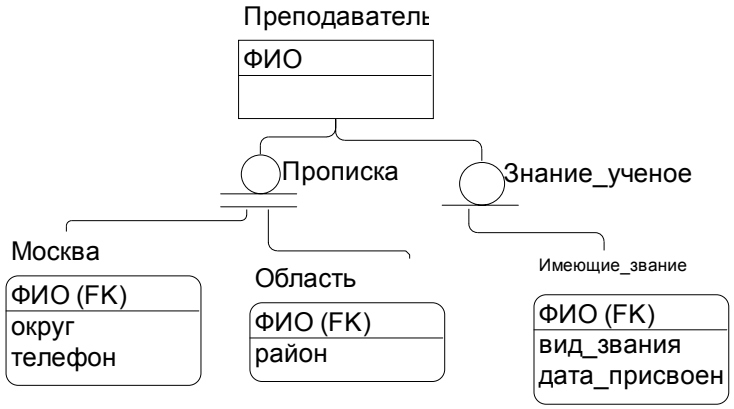


Рис. 2.61. Пример изображения об обобщенного объекта, классифицированного по нескольким признакам

На рис. 2.61 изображен пример обобщенного объекта, классифицированного по нескольким признакам.

Методология построения ER-модели при использовании Design/IDEF

Построение ER-модели является центральным моментом при проектировании автоматизированных информационных систем при использовании соответствующих технологий. Этот этап проектирования требует высокой квалификации исполнителей и оказывает существенное влияние на качество всех получаемых в результате проектных решений. Важность качественного ER-моделирования увеличивается еще и в связи с тем, что даже теоретически невозможно создать систему, позволяющую автоматически проверять правильность ER-модели с точки зрения ее адекватности отображаемой предметной области.

В п. 2.3. были изложены в общем виде рекомендации по построению ER-модели в зависимости от изобразительных средств и алгоритмов перехода от ER-модели к логической модели реляционной базы данных. В данном разделе эти рекомендации конкретизированы для системы Design/IDEF.

В Design/IDEF при преобразовании ER-модели в описание целевой базы данных каждому объекту ставится в соответствие таблица реляционной базы данных. В связи с этим, в отличие от базовой модели, в которой мы отображали все объекты предметной области, в Design/IDEF надо сначала определить не просто, что будет храниться в базе данных, а с уточнением, что будет храниться в отдельной таблице, и с учетом этого строить модель. В качестве объектов ER-модели следует изображать только те сущности, которым будут соответствовать отдельные таблицы. Другой путь – сначала построить предварительную ER-модель, а потом ее преобразовать к варианту, который будет служить исходным для генерации схемы базы данных. Все атрибуты или сущности, которые соответствуют вычисляемым значениям, которые проектировщик не хочет хранить в базе данных, должны быть устранены из «конечной» ER-модели.

Как мы видели, в Design/IDEF набор выразительных средств, предоставляемых для построения ER-модели, невелик.

Основным элементом модели является «сущность» (Entity). «Сущность» имеет имя. Для «сущностей» описываются их атрибуты. Атрибут может играть роль первичного ключа (Primary Key), альтернативного ключа (Alternate Key), дискриминатора (Discriminator) и инверсного входа (Inversion Entry), либо не играть ни одну из них. Как видно, во-первых, эти характеристики атрибутов отражают совершенно разные аспекты как предметной об-

ласти, так и организации данных; во-вторых, некоторые из них жестко привязаны к реляционной модели (понятия первичного и альтернативного ключа); в-третьих, выбор большинства из характеристик должен являться результатом проектных решений, обычно выполняемых на основе анализа разнообразных факторов; в-четвертых, ряд характеристик, которые можно отобразить в базовой ER-модели, с которой будет идти дальнейшее сравнение, в методологии IDEF1X в явном виде отобразить нельзя.

В Design/IDEF при преобразовании ER-модели в описание целевой базы данных каждому объекту ставится в соответствие таблица реляционной базы данных.

В Design/IDEF нет изобразительных средств для обозначения множественного свойства, составного свойства, нельзя показать, что свойство может присутствовать не у всех экземпляров объектов, нет понятия агрегированного объекта, нет изобразительного средства для отображения альтернативной связи («арк»). Все это надо отобразить, пользуясь имеющимися в наличии изобразительными средствами.

При построении ER-модели в Design/IDEF предлагается следовать следующим рекомендациям [С.М.1][12].

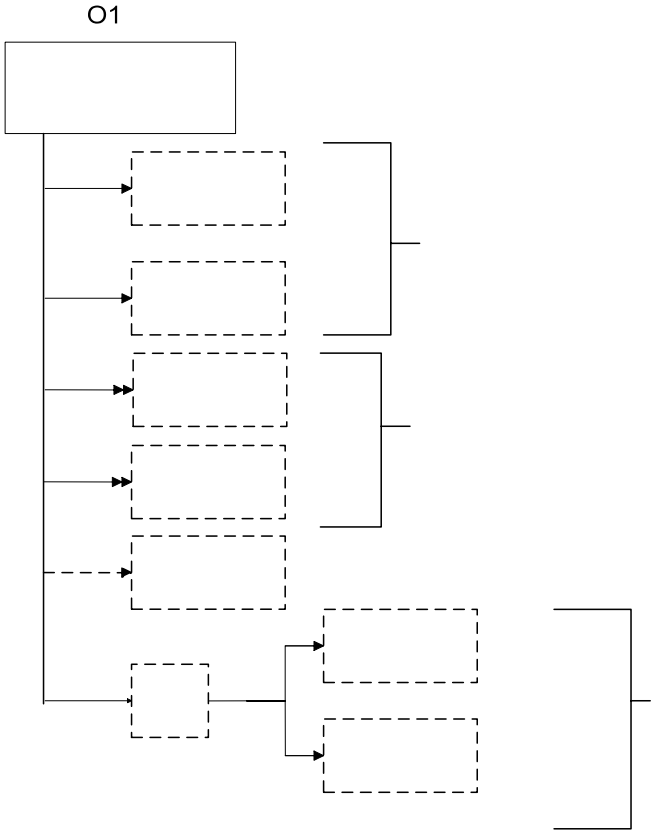
Если простой объект имеет несколько возможных идентификаторов, то из них следует выбрать тот, который будет использоваться в качестве первичного ключа таблицы, и описать его как ***Primary Key***. Рекомендации по выбору первичного ключа реляционной таблицы изложены в п. 3.3.

Для остальных возможных идентификаторов надо определить, есть ли необходимость проверять их уникальность в процессе ведения базы данных. Т.е. идентификаторы, для которых это необходимо, обозначить как ***Alternate Key.***

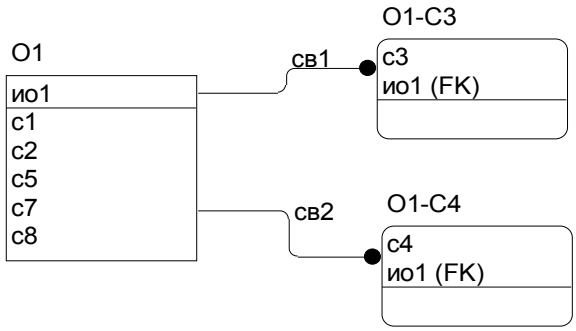
Если объект имеет неуникальное имя (например, Ф.И.О. – для сущности ЛИЧНОСТЬ), то его, как правило, следует описать как ***Inversion Entry***, так как имя обычно часто используется для поиска. В качестве инверсных входов могут быть описаны и другие атрибуты. Для того чтобы определить, для каких атрибутов следует задавать свойство ***Inversion Entry***, необходимо проанализировать характер запросов к создаваемой таблице: если по данному атрибуту ожидается частый выборочный поиск или требуется сортировка, то этот атрибут следует описать как инверсный вход. Т.к. альтернативных ключей и инверсных входов может быть несколько, то при соответствующей «пометке» атрибута надо указать порядковый номер **AK** или **IE**.

При наличии у объекта множественных свойств надо каждому из множественных свойств поставить в соответствие отдельную сущность. Название множественного свойства следует определить как ключевой атрибут. Затем надо связать идентифицирующей связью, направленной от основной сущности ко вновь созданной сущности, эти элементы модели (рис. 2.62).

[12] Так как «сущности» в методологии IDEF фактически соответствуют таблицам реляционной базы данных, то рекомендуется сначала изучить п. 3.3, тогда рекомендации, излагаемые в данном разделе, будут более понятными.



а) Простой объект. Фрагмент базовой ER-модели



б) Отображение объекта в IDEF1X

Рис. 2.62. Отображение единичных, множественных и составных свойств (переход от базовой модели к IDEF1X)

Так как в Design/IDEF нет возможности отображать составные свойства, то проектировщик должен принять решение, как это составное свойство будет храниться в базе данных. В зависимости от принятого решения надо это составное свойство описать либо как один атрибут, либо каждый из составляющих его элементов описать как отдельный атрибут. На рис. 2.62 изображено второе из названных решений. Если использовать первое решение, то вместо атрибутов С7, С8 следовало бы описать атрибут С6 (рис. 2.63).

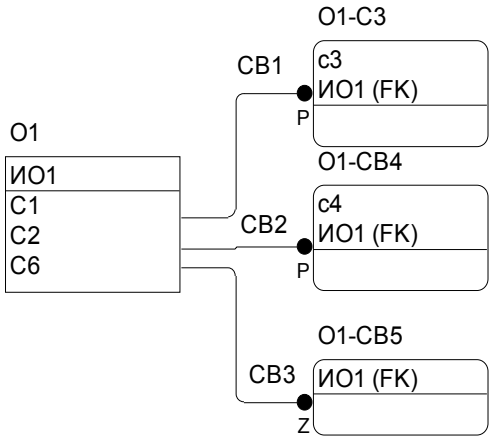
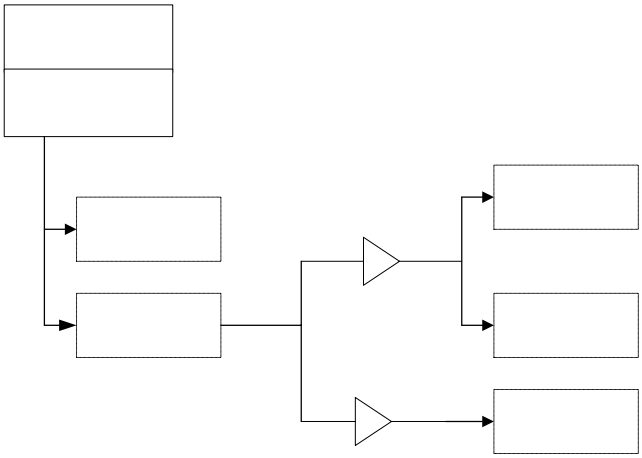


Рис. 2.63. Отображение единичных, множественных и составных свойств (переход от базовой модели к IDEF1X) – вариант 2

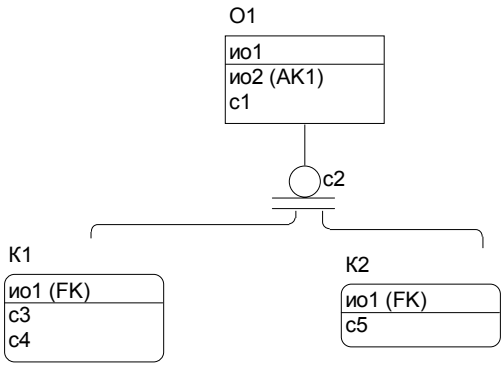
Отсутствие понятия «условное свойство» приводит к сложностям при моделировании предметной области. Возможны несколько вариантов выхода из сложившейся ситуации:

1. Никак в ER-модели не отражать, что свойство условное, и описывать его как обычный атрибут (такое решение изображено на рис. 2.62).
2. В соответствие условному свойству ставится отдельная сущность, которая идентифицирующей связью соединяется с основной сущностью. При этом тот факт, что свойство является условным, передается путем выбора соответствующего типа кардинальности связи (такое решение изображено на рис. 2.63).
3. Объект, содержащий условные свойства, отображать как обобщенный объект. Каждому условному свойству будет соответствовать категория объектов. Этот вариант использован на рис. 2.61, когда условному свойству «ученое звание» был поставлен в соответствие видовой объект «ИМЕЮЩИЕ ЗВАНИЕ».

Каждый из вариантов имеет свои недостатки. Выбор варианта будет зависеть от выбранного проектного решения по структуре БД.



а) Фрагмент базовой ER-модели

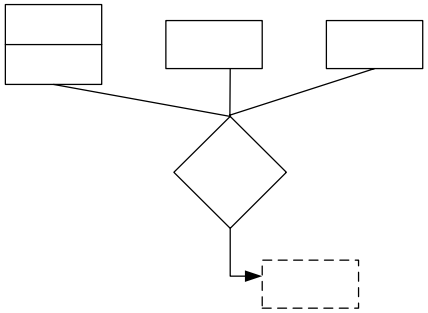


б) Отображение в IDEF1X

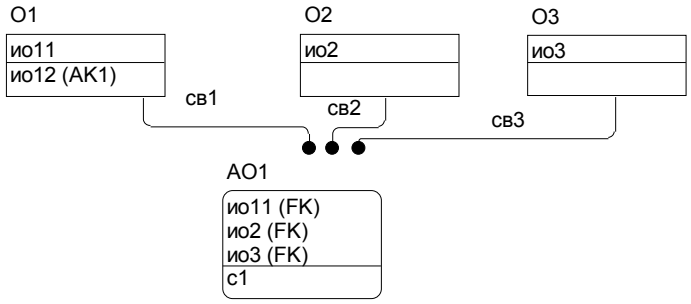
Рис. 2.64. Изображение обобщенного объекта

Для отображения обобщенного объекта в Design/DEF имеются специальные изобразительные средства. При изображении обобщенного объекта то свойство, по которому производится разбиение класса на подклассы, обозначается как «дискриминатор». Каждому дискриминатору на схеме соответствует специальный знак. После этого каждому подклассу ставится в соответствие отдельная сущность, в которой перечисляются атрибуты, присущие этому подклассу. Ключ «видовых» сущностей при описании указывать не надо. Далее от дискриминатора к «видовым» сущностям «протягивается» связь. При этом происходит автоматическая миграция ключа. Подчиненные сущности становятся, таким образом, зависимыми от идентификации сущностями (рис. 2.64).

В Design/IDEF можно отобразить многоаспектную и многоуровневую классификацию объектов.



а) Фрагмент базовой ER-модели



б) Отображение в IDEF1X

Рис. 2.65. Отображение агрегированного объекта

Для изображения агрегированных объектов в Design/IDEF не предусмотрено никаких специализированных изобразительных средств. Агрегированные объекты (процессы) в Design/IDEF следует отображать следующим образом (рис.2.65):

1) создать сущность, соответствующую данному объекту;
2) соединить ее идентифицирующими связями с сущностями, участвующими в данном процессе;
3) если для полной идентификации изображаемой сущности мигрировавших ключей оказывается недостаточно, то дополнительные атрибуты при их описании надо задать как «первичный ключ»;
4) описать остальные атрибуты.

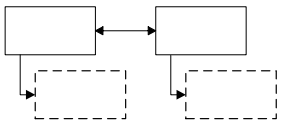
Рис. 2.66. Агрегированный объект «Успеваемость»

На рис. 2.66 изображен агрегированный объект «УСПЕВАЕМОСТЬ». «ДАТА» описана как «первичный ключ». «КОД _ СТУДЕНТА» и «КОД _ ПРЕДМЕТА» мигрируют в сущность «УСПЕВАЕМОСТЬ» при установлении соответствующих связей.

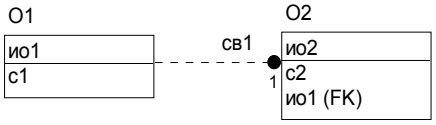
Как мы видим из выше изложенного, механизмом установления идентифицирующей связи приходится пользоваться часто, и не только в случаях, когда в предметной области реально существует такая идентификация (например, когда нумерация участков ведется внутри каждого цеха и тому подобных ситуациях), но и при «обходе» ограничений Design/IDEF, когда приходится вводить «дополнительные» объекты, или при изображении агрегированных объектов в виде обычной сущности в модели Design/IDEF.

Для определения связи в Design/IDEF надо комбинировать возможности, задаваемые в **Relationship Type** и **Relationship Cardinality.**

Идентифицирующая (Identifying) и неидентифицирующая (Non-Identifying) связь в общем случае соответствует отношению 1 : M. Если надо указать, что связь 1 : 1, то можно для **Relationship Cardinality** этой связи указать Exactly 1 (рис. 2.67).



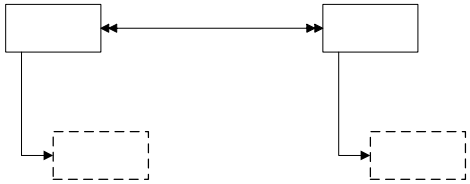
а) Фрагмент базовой ER-модели



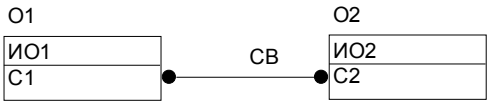
б) Отображение в IDEF1X

Рис. 2.67. Отображение связи 1 : 1

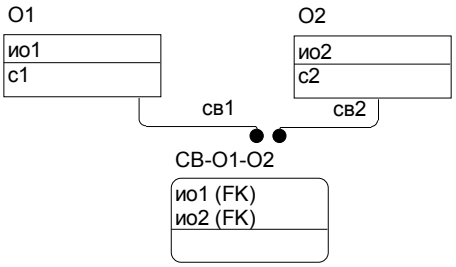
Множественная связь называется в Design/IDEF «неспецифической» (Non-Specific) (рис. 2.68). Так как алгоритм проектирования БД в Design/IDEF не обеспечивает автоматического преобразования связей M : M, то перед генерацией описания БД необходимо устранить неспецифические связи и преобразовать их в специфические. Для этого надо ввести дополнительную связующую сущность. Никаких атрибутов для нее определять не надо. После чего надо связать вновь введенную сущность с ранее существовавшими объектами идентифицирующей связью. В результате получиться схема, изображенная ни рис. 2.68 в).



а) Фрагмент базовой ER-модели



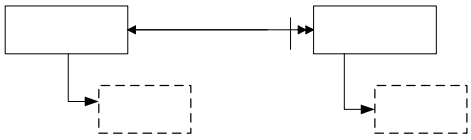
б) Отображение в IDEF1X – неспецифическая связь



в) преобразование неспецифической связи в специфические

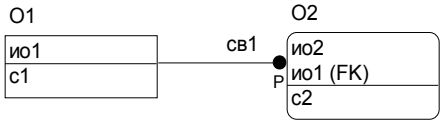
Рис. 2.68. Отображение связи M : M

Соответствие базовой модели и модели Design/IDEF при изображении идентифицирующей связи отображено на рис. 2.69, а неидентифицирующей – на рис. 2.70.



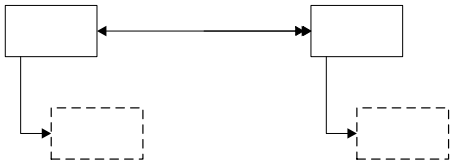
а). Фрагмен

а) Фрагмент базовой ER-модели

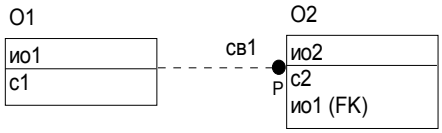


б) Отображение в IDEF1X

Рис. 2.69. Идентифицирующая связь



а) Фрагмент базовой ER-модели



б) Оображение в IDEF1X

Рис. 2.70. Отображение связи 1 : M
(обязательный класс членства с обеих сторон)

В базовой модели для характеристики связи используются три независимые характеристики: 1) тип связи («один к одному» (1 : 1), «один ко многим» (1 : M), «многие к одному» (M : 1) и «многие ко многим» (M : M)), 2) класс членства (обязательный и необязательный) и 3) зависимость по идентификации (зависимая и не зависимая по идентификации сущности). В Design/IDEF эти характеристики «смешаны» в две группы: Relationship Type и Relationship Cardinality. В связи с этим не все сочетания, которые можно передать в базовой модели, можно отобразить в Design/IDEF. На рис. 2.71 представлено соответствие конструкций базовой модели и Design/IDEF1X при изображении типа связей и класс членства. В Design/IDEF при задании «неспецифической связи» кардинальность связи указать нельзя. Поэтому в Design/IDEF нельзя отобразить ситуации, когда при связи M : M наблюдается необязательный класс членства со стороны одной из сущностей или со стороны обеих сущностей. Кроме того, в Design/IDEF нет возможности отобразить класс членства сущностей, к которым направлена связь.

а). Фрагмент

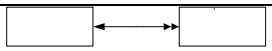
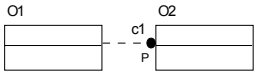
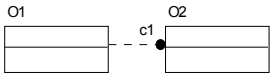
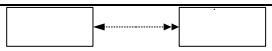
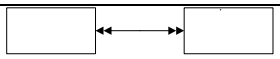
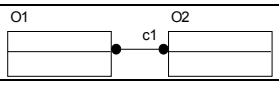
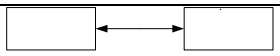
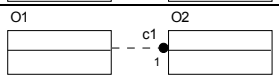
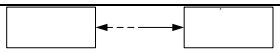
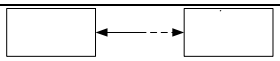
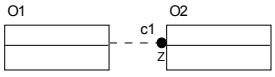
N п/п	Изображение в базовой модели	Изображение в Design/IDEF		
		Relationship Type	Relationship Cardinality	Графическое изображение
1.		Non-Identifying	One or Many [P]);	
2.		Класс членства сущности, находящейся в конце связи, указать нельзя		
3.		Non-Identifying	(Zero, One or Many);	
4.		Класс членства сущности, находящейся в конце связи, указать нельзя		
5.		Non-Specific	Нельзя указать	
6.		Non-Identifying	Exactly 1	
7.		Класс членства сущности, находящейся в конце связи, указать нельзя		
8.		Non-Identifying	Zero or One [Z]	

Рис. 2.71. Типы связей и класс членства. Соответствие базовой модели и Design/IDEF1X

На построение ER-модели оказывает влияние алгоритм проектирования базы данных. В Design/IDEF каждой сущности ER-модели соответствует таблица в реляционной базе данных. Поэтому необходимо уже на стадии ER-моделирования решить, каким сущностям ставить в соответствие таблицы, а каким – нет, и в качестве объектов ER-модели изображать только те сущности, которым будут соответствовать отдельные таблицы. Например, «дату» не надо отображать как отдельную таблицу, поэтому эту сущность следует из модели убрать. Все атрибуты или сущности, которые соответствуют вычисляемым значениям, которые проектировщик не хочет хранить в базе данных, тоже должны быть устранены из ER-модели.

В Design/IDEF можно нарисовать несколько связей между парой объектов. При этом на схеме миграция ключа происходит только один раз, а не столько раз, сколько связей объявлено между парой объектов (рис. 2.72).

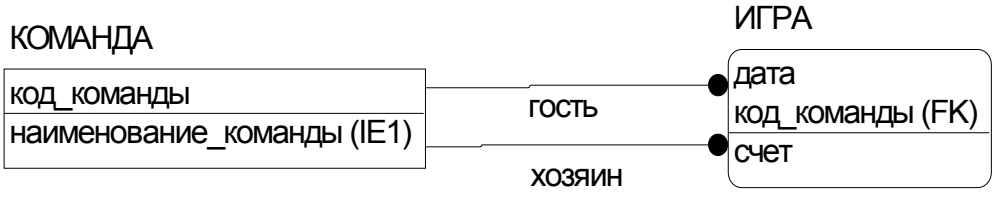


Рис. 2.72. Изображение нескольких связей между парой сущностей

На самом деле, если посмотреть на соответствующее сгенерированное системой описание объекта на SQL, то в нем присутствуют оба атрибута связи:

```
CREATE TABLE игра
        (дата DATE NOT NULL,
        счет  CHAR NULL,
        код _ команды      CHAR NOT NULL,
        код _ команды      CHAR NOT NULL);
```

Таким образом, в системе имеется некоторое несоответствие между графическим и аналитическим представлением сущности. Следует также обратить внимание на некоторые неточности в полученном описании. Так, двум полям в таблице ИГРА дано одинаковое имя «код _ команды», что недопустимо.

В связи с тем, что ER-модели играют несколько разных ролей в процессе проектирования информационных систем, среди которых важнейшей является «адекватное отображение предметной области и использование ER-модели для общения между всеми участниками (как проектировщиками, так и заказчиками) процесса создания ИС», то при применении CASE-средств, не обладающих развитым многовариантным алгоритмом проектирования, рекомендуется создавать, по меньшей мере, две модели:

- исходную, описывающую предметную область безотносительно к заложенному в CASE-системе алгоритму преобразования ER-модели в логическую модель целевой базы данных;
- ER-модель, которая будет использоваться для генерации схемы базы данных.

Так как Design/IDEF не производит преобразование имен сущностей и атрибутов в соответствии с требованиями целевой СУБД, то в модели, предназначенной для генерации схемы базы данных, имена сущностям и атрибутам следует давать такие, которые допустимы в целевой СУБД.

В связи с тем, что изобразительные средства ER-моделирования в Design/IDEF бедны, процесс моделирования предметной области не является естественным. Практически, специалист, строящий ER-модель в среде Design/IDEF, должен выполнить проектирование структуры реляционной базы данных «вручную». Подобные средства способствуют решению некоторых проблем, стоящих при создании и развитии ИС, но не облегчает задачу проектирования структуры базы данных.

Контрольные вопросы

1. Что называется предметной областью?
2. Что называется концептуальной моделью? Для каких целей она служит?
3. Перечислите основные компоненты концептуальной модели.
4. Какие требования предъявляются к концептуальной модели?
5. Какие преимущества дает использование ER-моделирования при создании БД?
6. Что называется классом объектов?
7. Какие разновидности объектов выделяются в базовой ER-модели? Какие графические обозначения используются для изображения каждого из видов объектов?
8. Приведите примеры из любых предметных областей для каждой из разновидностей объектов.
9. Какие разновидности свойств объектов выделяются в базовой ER-модели? Какие графические обозначения используются для изображения каждого из видов свойств?

10. Приведите примеры из любых предметных областей для каждой из разновидностей свойств.
11. Что называется «зависимыми от идентификации сущностями»?
12. Что следует выделять в качестве самостоятельного объекта в ER-модели?
13. Какие интегральные характеристики класса объектов обычно фиксируются при описании предметной области? Как они используются при проектировании БД?
14. В каких случаях в концептуальной модели следует в явном виде отображать класс объектов?
15. Какие разновидности связей между объектами выделяются в базовой ER-модели? Какие графические обозначения используются для изображения каждого из видов связей?
16. Приведите примеры из любых предметных областей для каждой из разновидностей связей.
17. В каком случае следует вводить в модель обобщенный объект?
18. Какую информацию о предметной области дает граф пересечений?
19. Какие CASE-средства Вы знаете?
20. Чем отличаются известные Вам методологии ER-моделирования друг от друга?
21. Какие разновидности объектов выделяются в ER-модели, построенной в нотации IDEF1X?

Глава 3. ДАТАЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

В п. 1.3 мы рассматривали классификацию баз данных по типу модели данных. Подходы к проектированию логической структуры БД существенно зависят от типа модели данных. В данной главе будут рассмотрены вопросы даталогического проектирования применительно к структурированным моделям данных.

3.1. ОБЩИЕ СВЕДЕНИЯ О ДАТАЛОГИЧЕСКОМ ПРОЕКТИРОВАНИИ

Исходные данные для даталогического проектирования

Любая СУБД оперирует с допустимыми для нее логическими единицами данных, а также допускает использование определенных правил композиции логических структур более высокого уровня из составляющих информационных единиц более низкого уровня. Кроме того, многие СУБД накладывают количественные и иные ограничения на структуру базы данных. Поэтому прежде чем приступить к построению даталогической модели, необходимо детально изучить особенности СУБД, определить факторы, влияющие на выбор проектного решения, ознакомиться с существующими методиками проектирования, а также провести анализ имеющихся средств автоматизации проектирования, возможности и целесообразности их использования.

Хотя даталогическое проектирование является проектированием логической структуры базы данных, на него оказывают влияние возможности физической организации данных, предоставляемые конкретной СУБД. Поэтому знание особенностей физической организации данных является полезным при проектировании логической структуры.

Логическая структура базы данных, а также сама заполненная данными база данных, является отображением реальной предметной области. Поэтому на выбор проектных решений самое непосредственное влияние оказывает специфика отображаемой предметной области, отраженная в инфологической модели.

Все шаги проектирования даталогической модели выполняются итеративно. Причем вероятны итерации не только внутри стадии даталогического проектирования, но и с «захватом» других стадий проектирования БД.

Результат даталогического проектирования

Конечным результатом даталогического проектирования является описание логической структуры базы данных на ЯОД. Однако если проектирование выполняется «вручную», то для большей наглядности сначала строится схематическое графическое изображение структуры базы данных. При этом должно быть обеспечено однозначное соответствие между конструкциями языка описания данных и графическими обозначениями информационных единиц и связей между ними. Графическое представление используется и при автоматизированном проектировании структуры базы данных как интерфейсное средство общения с проектировщиком и при документировании проекта.

Спроектировать логическую структуру базы данных означает определить все информационные единицы и связи между ними, задать их имена; если для информационных единиц возможно использование разных типов, то определить их тип. Следует также задать некоторые количественные характеристики, например, длину поля.

Подход к даталогическому проектированию

Каждый тип модели данных и каждая разновидность модели, поддерживаемая конкретной СУБД, имеют свои специфические особенности. Вместе с тем имеется много общего во всех структурированных моделях данных и принципах проектирования БД в их среде. Все это дает возможность использовать единый методологический подход к проектированию структуры базы данных.

В БД отражается определенная предметная область. Поэтому процесс проектирования БД предусматривает предварительную классификацию объектов предметной области, систематизированное представление информации об объектах и связях между ними.

На проектные решения оказывают влияние особенности требуемой обработки данных. Поэтому соответствующая информация должна быть определенным образом представлена и проанализирована на начальных этапах проектирования БД.

Данные о предметной области и особенностях обработки информации в ней фиксируются в инфологической модели. В инфологической модели должна быть отображена вся информация, циркулирующая в информационной системе, но это вовсе не означает, что вся она должна храниться в базе данных. В связи с этим, одним из первых шагов проектирования является определение состава БД, т.е. перечня тех показателей, которые целесообразно хранить в БД.

При проектировании логической структуры БД осуществляется преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД, и проверка адекватности полученной даталогической модели отображаемой предметной области.

Для любой предметной области существует множество вариантов проектных решений ее отображения в даталогической модели. Методика проектирования должна обеспечивать выбор наиболее подходящего проектного решения.

Минимальная логическая единица данных (несмотря на их разные названия) семантически для всех СУБД одинакова и соответствует либо идентификатору объекта, либо свойству объекта или процесса.

Связи между сущностями предметной области, отраженные в инфологической модели, могут отображаться в даталогической модели либо посредством совместного расположения соответствующих им информационных элементов, либо путем объявления связи между ними. Связь может передаваться как на внутризаписном, так и межзаписном уровне.

Не все виды связей, существующие в предметной области, могут быть непосредственно отображены в конкретной даталогической модели. Так, многие СУБД не поддерживают непосредственно отношение М : М между элементами. В этом случае в даталогическую модель вводится дополнительный вспомогательный элемент, отображающий эту связь (таким образом отношение М : М как бы разбивается на два отношения 1 : М между этим вновь введенным элементом и исходными элементами).

Следует обратить внимание на то, что отношения, имеющие место в предметной области и отражаемые в ИЛМ, могут быть переданы не только посредством структуры базы данных, но и программным путем (т.е. всегда существует альтернатива между декларативным и процедурным способом описания явления). Например, при отображении обобщенных объектов можно не выделять подклассы на уровне логической структуры базы данных. В этом случае подклассы будут выделяться программным путем при обработке хранимых данных.

Решение о том, какой из способов отображения (структурный / декларативный или программный / процедурный) следует использовать в каждом конкретном случае, будет зависеть от многих факторов, таких как стабильность отображаемой сущности, объем номенклатуры, особенности СУБД, характер обработки данных и др. Так, если в предметной

области используется классификация сотрудников по полу, в базе данных не следует создавать классификатор полов, т.к. он будет содержать всего две позиции и никогда не меняется. Как правило, не следует и выделять по этому признаку соответствующие подклассы для объектов предметной области, так как в большинстве случаев они обрабатываются совместно и, в основном, имеют одинаковый набор свойств, характеризующий их. Хотя в некоторых предметных областях деление на такие подклассы может быть целесообразным.

Если же отображаемая сущность не стабильна, то ее лучше передавать посредством данных, так как в противном случае будет часто требоваться преобразование программы, что обычно обеспечить труднее, чем изменение данных.

При отображении обобщенных объектов в БД возможны разные варианты: хранить информацию обо всем обобщенном объекте в одном файле / таблице; каждому подклассу объектов низшего уровня выделять отдельные самостоятельные файлы // таблицы. Оба эти варианта могут быть использованы в любой СУБД. В первом случае подчеркивается общность объектов разных подклассов, входящих в обобщенный объект. Во втором случае, напротив, обобщенный объект как единое целое не отображается в структуре базы данных.

Другие способы отображения связаны с явным или неявным выделением подклассов в логической структуре БД. Неявное выделение подкласса заключается в том, что в записи отводятся поля для фиксации значений свойств, общих для объектов разных подклассов, и значения признака подкласса, а вместо полей, наличие которых зависит от подкласса, используется одно поле с переменным составом, содержание которого будет зависеть от того, к какому подклассу относится описываемый объект.

Реализация принципа явного выделения подклассов в структуре БД существенно зависит от специфики СУБД.

При проектировании логической структуры БД основное значение имеет специфика отображаемой предметной области. Однако, как отмечалось выше, и характер обработки информации оказывает влияние на принимаемое проектное решение. Например, рекомендуется хранить вместе информацию, часто обрабатываемую совместно, и наоборот, разделять по разным файлам информацию, не использующуюся одновременно. Информацию, используемую часто, и информацию, частота обращения к которой мала, также следует хранить в разных файлах, причем последнюю может оказаться выгодным вынести в архивные файлы, а не поддерживать в составе БД.

Как отмечалось выше, в ИЛМ описывается не отдельный объект, а класс объектов. Но в редких случаях бывает, что класс включает в себя только один экземпляр объекта. Например, если предметной областью является какой-то конкретный институт, то класс объектов «ИСТИТУТ» будет содержать только один экземпляр объекта. Таким «вырожденным» классам объектов обычно не ставится в соответствие отдельный файл базы данных.

Определение состава базы данных

При переходе от инфологической модели к даталогической следует иметь в виду, что инфологическая модель включает в себя всю информацию о предметной области, необходимую и достаточную для проектирования БД. Это не означает, что все сущности, зафиксированные в ИЛМ, должны в явном виде отражаться в даталогической модели. Прежде чем строить даталогическую модель, необходимо решить, какая информация будет храниться в базе данных. Например, в инфологической модели должны быть отражены вычисляемые показатели, но вовсе не обязательно, что они должны храниться в базе данных.

Существуют разные подходы к определению состава показателей, которые должны храниться в базе данных. Согласно одному из них в БД должны храниться только исходные показатели, все производные показатели должны быть получены расчетным путем в момент реализации запроса (этот подход иногда называют принципом синтезирования, имея в виду возможность «синтезировать» требуемые показатели из хранимых в информационной базе данных).

Такой подход имеет очевидные достоинства: 1) простота и однозначность в принятии решения «что хранить»; 2) отсутствие неявного дублирования информации со всеми вытекающими из этого последствиями (меньше объем памяти, чем при хранении исходных, и производных показателей, – проще проблемы контроля целостности данных); 3) потенциальная возможность получить любой расчетный показатель, а не только те, которые хранятся в БД.

При принятии решения о хранении расчетных показателей принимаются во внимание несколько факторов.

Рассмотрим некоторую гипотетическую предметную область, представляющую собой учебное заведение. Учащимся этого заведения начисляется стипендия. Существует четкий алгоритм начисления стипендии. Предположим, что он выглядит следующим образом:

1. Стипендия начисляется только тем, кто успешно сдал все экзамены в сессию (т.е. сдал в срок и не получал неудовлетворительных оценок), а также учащимся, имеющим детей.
2. Известен размер обычной стипендии.
3. Тем, кто не имеет удовлетворительных оценок, назначается стипендия на 25% выше, чем обычная.
4. Тем, кто имеет только отличные оценки, стипендия увеличивается на 50% от размера обычной стипендии.

Предположим также, что размер стипендии не может изменяться в течение семестра. Таким образом, размер стипендии является вычисляемым показателем и может не храниться в базе данных, а каждый раз определяться расчетным путем. Однако в рассматриваемой ситуации его все-таки лучше хранить в БД по следующим основным причинам:

а) полученная величина многократно используется в дальнейшем;
б) алгоритм определения размера стипендии имеет достаточно сложную логику, и для выполнения расчета требуется просмотреть несколько файлов (некоторые из них являются большими по объему, что также замедляет процесс обработки);
в) размер стипендии не должен изменяться в течение семестра. Кроме того, имея реальный файл «Начисленная стипендии», легче контролировать его полноту и достоверность.

Предположим также, что к БД достаточно часто обращаются с запросом о среднем балле какого-либо студента или среднем балле по той или иной дисциплине и т.п.

При изменении любой оценки или получении новой средний балл меняется, и если Вы вдруг решите хранить и среднюю величину, и исходные оценки, то обеспечение их соответствия представляет собой некоторую проблему. При правильной организации файлов вычисление средней величины во время исполнения запроса не представляет проблемы ни с точки зрения времени обработки, ни с точки зрения задания запроса. Поэтому хранить средний балл не следует.

Введение искусственных идентификаторов

При отображении объекта в файл базы данных идентификатор объекта будет являться полем этого файла, причем в большинстве случаев – ключевым полем (т.е. полем, однозначно идентифицирующим запись). Однако в некоторых случаях появляется необходимость введения искусственных идентификаторов или, как их еще называют, кодов. Такими случаями являются следующие:

1. В предметной области может наблюдаться синонимия, т.е. может случиться, что естественный идентификатор объекта не обладает свойством уникальности. Например, среди сотрудников предприятия могут быть однофамильцы. В этом случае для обеспечения однозначной идентификации объектов предметной области в информационной системе целесообразно использовать искусственные коды.
2. Если объект участвует во многих связях / процессах, то для их отображения создается несколько файлов / таблиц, в каждом из которых повторяется идентификатор объекта. Для того чтобы не использовать во всех файлах длинный естественный идентификатор объекта, можно ввести и использовать более короткий код. Например, вместо длинных наименований продукции обычно используются их кодовые обозначения. Это не только сэкономит память, но и сократит трудоемкость ввода информации (хотя последнего можно достичь и другим путем).
3. Если естественный идентификатор может изменяться со временем (например, фамилия), то это может вызвать много проблем, если наряду с таким «динамическим» идентификатором не использовать «статический» искусственный идентификатор.

Когда присваиваются идентификаторы каким-либо объектам, то желательно, чтобы эти идентификаторы были постоянными. Например, в учебных заведениях всегда каким-либо образом обозначаются учебные группы. Для обозначения группы можно, например, использовать номер курса, на котором в настоящее время учится данная группа, и какие-то другие символы (подобно тому, как обозначаются классы в школе). Но в этом случае идентификатор группы будет каждый год меняться. А можно для обозначения группы использовать, например, две последние цифры года поступления. В этом случае идентификатор группы будет постоянным. Использование постоянных идентификаторов снимает множество проблем при функционировании информационной системы.

Встречаются и иные случаи, когда целесообразно вводить искусственные идентификаторы. Некоторые из них будут рассмотрены далее при обсуждении проблем проектирования БД.

Критерии оценки БД

Для оценки проектируемой / спроектированной БД может быть использовано множество критериев. Значимость этих критериев в свою очередь будет зависеть от большого числа разнообразных факторов. Прежде всего, эта значимость зависит от самого критерия: есть критерии, значимость которых не только никогда не понижается, а, наоборот, постоянно растет (это такие критерии, как адекватное отображение действительности, удовлетворение разнообразных потребностей пользователей и т.п.). Значимость других критериев становится менее существенной с ростом возможностей техники и программного обеспечения (например, занимаемый базой данных объем памяти).

На значимость критерия оценки влияют особенности ИС, для которой создается проект (например, для систем, работающих в реальном масштабе времени, очень важна скорость реакции системы на запросы, для банковских систем – защита информации и надежность системы).

Все критерии оценки являются взаимосвязанными и часто противоречивыми: улучшение показателей по одному из критериев может привести к ухудшению значений показателей, оценивающих модель по другим показателям. Необходима комплексная оценка проекта по всей совокупности критериев.

Критерии могут быть как количественными, та и качественными.

Перечислим основные критерии, используемые при оценке БД:

1. *Адекватность* – соответствие базы данных реальной предметной области. Естественно, что БД не должна искажать предметную область. Это является важнейшим требованием к БД, без соблюдения которого бессмысленно соблюдение всех остальных требований. Оценка адекватности является не только очень важной, но и очень трудной задачей, т.к. некоторые несоответствия очень трудно выявляются. Адекватность должна быть обеспечена как на уровне структуры БД, так и при задании ограничений целостности. Так, например, если в предметной области могут однофамильцы, а Вы зададите ФИО как ключ, то запись, касающаяся однофамильца, не сможет быть введена в базу данных.
2. *Полнота* – возможность удовлетворения существующих и новых потребностей пользователей. Использование подхода «от предметной области» к проектированию баз данных и сама идеология банков данных как интегрированного взаимоувязанного хранилища данных способствуют обеспечению этого критерия. Хранение в БД детализированных показателей также повышает возможности удовлетворения разнообразных (в том числе и нерегламентированных) потребностей пользователей. Полнота является одним из проявлений адекватности БД.
3. *Адаптируемость*:
   3.1. *Адаптируемость к изменениям в предметной области*:
      3.1.1. *Устойчивость схемы базы данных* – отсутствие необходимости в изменении структуры БД при изменении предметной области. В теории БнД широко используется понятие независимости программ от данных и данных от программ. Не менее, а даже более значимой, является проблема обеспечения независимости логической модели БД от изменений, происходящих в предметной области. Устойчивость модели является лучшим проявлением свойства адаптируемости системы. Обеспечение устойчивости модели базы данных к изменениям предметной области фактически снимает проблему независимости программ от данных, т.к. в этом случае структуры данных меняться не будут.

Поясним суть данного показателя на примере. Предположим, необходимо хранить информацию об успеваемости студентов. Ниже приведены некоторые из возможных вариантов структуры БД для хранения этой информации. Первый вариант предполагает, что создается таблица, в строках которой помещаются фамилии студентов, графами являются названия предметов, а значениями соответствующих полей будет оценка, полученная данным студентом по данному предмету. Каждому студенту соответствует одна запись в БД. Такой вариант очень похож на «Книги баллов», которые традиционно ведутся в деканатах. Для обсуждения проблемы устойчивости модели не существенно, а для оценки БД по другим критериям будет иметь значение, сколько таблиц будет создано (т.к. студенты учатся по разным учебным планам, то при создании только одной таблицы будет много пустых значений полей, содержащих оценки, а каждая запись будет содержать слишком большое число полей).

Вариант 1:

ФИО	Высш_мат.	Ин._яз.	БД	…
Иванов	5	4	5	…
Якушкина	4	5	4	….

Во втором варианте таблица содержит только три поля: «Ф.И.О.», «Предмет», «Оценка». Для каждого студента будет создано столько записей, сколько экзаменов сдал этот студент.

Вариант 2:

ФИО	Предмет	Оценка
Иванов	Высш. мат	5
Якушкина	Высш. мат	4
Иванов	Ин. яз.	4
Якушкина	Ин. яз	5
….	….	…..

Если будет выбран первый вариант, то при каждом изменении в учебном плане надо будет менять структуру базы данных. Во-втором случае никакие изменения структуры БД не потребуются. Более того, при создании БД по второму варианту вообще не надо знать действующие учебные планы. Решение будет универсальным и может использоваться в любом учебном заведении без необходимости «подстройки» на конкретное учебное заведение. Кстати, *универсальность* также может использоваться как критерий оценки БД.

Приведем другой пример, показывающий влияние выбранного типа данных на устойчивость спроектированной БД. Предположим, что в предметной области для кодирования какой-либо номенклатуры используется цифровой код, и в базе данных для соответствующего поля был выбран числовой тип данных. Если возникнет необходимость перейти на буквенную или буквенно-числовую систему кодирования, то в БД придется менять тип данных у соответствующего поля. Если бы тип данных при проектировании БД изначально был определен как текстовый, то изменения бы не потребовались.

С рассматриваемым критерием будет тесно связан критерий *затрат на поддержание системы в работоспособном состоянии.* С затратами на адаптацию структуры БД будут непосредственно связаны и *затраты на адаптацию прикладного программного обеспечения.*

3.1.2. *Простота и эффективность внесения изменений*. Речь может идти как об изменении структуры базы данных в случае возникновения такой необходимости, так и об обычной корректировке значений данных в базе данных.

3.1.3. *Простота корректировки структуры БД данных.* Например, некоторые типы полей трудно преобразовать в другие. Особенно внимательными надо быть при определении полей связей, т.к. их изменение повлечет за собой целую «цепочку» изменений.

3.1.4. *Простота и трудоемкость корректировки значений данных*. Прежде всего, следует обратить внимание на аномалии, возникающие при корректировке ненормализованных структур.

Одним из показателей простоты корректировки может быть «*необходимое число изменений, которые необходимо провести при наступлении одного события в предметной области».*

Например, предположим, что мы имеем базу данных, содержащую сведения о сотрудниках учебного заведения, и фиксируем в ней некоторые биографические и прочие справочные данные, а также информацию о том, какие учебные предметы может вести каждый преподаватель. Преподаватель может владеть несколькими предметами.

Сравним же следующие три проектных решения:

Проектное решение 1:

Ф1 (ФИО, дата_рождения, пол, ..., телефон, название_предмета)

Проектное решение 2:

Ф1 (ФИО, дата_рождения, пол, ...,)

Ф2 (ФИО, название_предмета)

Проектное решение 3:

Ф1 (код_сотрудника,ФИО, дата_рождения, пол, ...,)

Ф2 (код_сотрудника, название_предмета)

Следует сразу отметить, что варианты 1 и 2 являются неудовлетворительными и использованы только для иллюстрации недостатков, связанных с подобными решениями.

В первом случае мы имеем таблицу, находящуюся в 1НФ. Если преподаватель владеет несколькими предметами, то в таблице Ф1 ему будет соответствовать несколько строк.

Смена фамилии каким-либо сотрудником приведет в варианте 1 к корректировке стольких записей, сколько предметов ведет данный преподаватель, в варианте 2 – еще на одну запись больше, а в варианте 3 – к корректировке только одного значения. Изменение же номера телефона приведет в варианте 1 также к корректировке стольких записей, сколько предметов ведет данный преподаватель, а в вариантах 2 и 3 – к корректировке только одной записи.

Первое проектное решение – ненормализованная структура. Она плоха тем, что приводит к большому дублированию информации. Второй и третий вариант – оба представляют нормализованную структуру и отличаются тем, что в последнем варианте введен искусственный идентификатор. Именно третий вариант является наиболее предпочтительным.

Если «вдруг» (вообще-то этого лучше не делать) в базе данных в качестве поля связи определено поле, которое может менять свое значение, то следует обратить внимание на возможность автоматической корректировки связанных полей при соответствующем задании правил обеспечения ограничений целостности связи.

3.2. *Адаптация к изменениям информационных потребностей пользователей, возможность удовлетворения нерегламентированных запросов.* Например, если хранить в БД детальные данные, то любые производные данные можно получить при возникновении необходимости в них; если же хранить только какие-либо сводные данные, но не хранить исходные, то получить информацию, отличную от хранимой, в большинстве случаев нельзя.

3.3. *Адаптация к изменениям используемых программных и технических средств.* Основным способом обеспечения этого требования является соблюдение стандартов, а также, по возможности, использование при выборе проектного решения таких средств, которые являются широко распространенными, а не специфическими для конкретной системы. Так, например, использование «экзотических» типов полей, скорее всего, приведет к проблемам при переносе системы в другую среду или при обработке информации в гетерогенной среде.

Одним из проявлений рассматриваемого свойства является масштабируемость. Ведущие разработчики программных продуктов уделяют большое внимание обеспечению этих свойств.

4. *Универсальность*. Может быть обеспечена разными способами, например, реализацией возможности настройки системы на особенности предметной области, определенными приемами при проектировании структуры БД и программного обеспечения. Особое зна-

чение приобретает при создании «отчуждаемых» проектов, ориентированных на конечных пользователей, не являющихся специалистами по машинной обработке данных.

5) *Сложность структуры БД*. Речь может вестись как о сложности самой поддерживаемой в данной СУБД модели данных, так и о сложности логической структуры конкретной спроектированной БД.

Сложность модели будет определяться числом разнообразных информационных единиц, допустимых в ней, способами их соподчинения и связывания, накладываемыми ограничениями. Самыми простыми из структурированных моделей БД являются реляционные.

Естественно, что показатели сложности спроектированной БД будут зависеть от типа поддерживаемой модели БД. Сравнение по этому показателю баз данных, спроектированных в среде разных СУБД, будет иметь свою специфику. Для реляционной модели сложность будет характеризоваться числом таблиц и полей в БД, числом индексных файлов (индексов). В принципе, чем меньше сложность БД, тем лучше. Однако снижение сложности, наряду с положительными результатами, часто приводит ко многим отрицательным последствиям. Так, для реляционных систем самой «простой» БД будет одно универсальное отношение, но к каким последствиям приведет использование такого проектного решения, хорошо известно из теории нормализации.

Резюме. Критерий «сложность» никогда не рассматривается как самостоятельный.

6) *Степень дублирования данных в БД*. Различают необходимое, контролируемое и неконтролируемое дублирование. Но какими бы причинами ни было вызвано дублирование данных, оно всегда ведет к необходимости поддержки идентичности всех копий дублируемых значений, росту требуемого объема памяти, повышению трудоемкости корректировки, увеличению числа полей в БД, что увеличивает ее сложность.

7) *Сложность последующей обработки*. Оценить этот показатель достаточно трудно, так как его значение зависит как от предполагаемой обработки, так и от возможностей языка манипулирования данными конкретной СУБД. Тем не менее, для большинства СУБД справедливыми являются утверждения, что:

- легче обрабатывать один файл, чем несколько связанных файлов;
- легче объединить несколько полей, чем выделить отдельные составляющие из единого поля (например, из «Адреса» – страну, город и т.п., из «ФИО» – фамилию, имя, отчество и т.п.). Если, например, в каких-либо выходных документах надо вывести фамилию и инициалы, то в случае раздельного хранения полей это также легко сделать, например, просто изменив шаблон вывода для полей «имя» и «отчество» на (X.)). Однако хранение каждой из составляющих СЕИ в виде отдельных полей имеет и очевидные недостатки: база данных становится сложнее, затрачивается больше времени при создании файла на описании его структуры, увеличивается объем служебной информации и объем памяти, требуемой для хранения данных;
- обработка «неэлементарных» полей в реляционных системах всегда представляет сложность (это вызвано тем, что с точки зрения СУБД поле остается элементарной единицей). Например, если вы в БД «КАДРЫ» включили поле «ДЕТИ», в котором в записи о сотруднике фиксируете сведения обо всех его детях, то задать запросы типа: «Выделить всех сотрудников, имеющих больше 3-х детей» или «Определить среднее число детей у сотрудников» будет достаточно сложно. В то же время, если сведения о детях выделить в отдельную таблицу, в которой каждому ребенку будет соответствовать отдельная запись, то реализация подобных запросов не составит особого труда;

- все «групповые» операции (суммирование, подсчет, определение среднего значения и т.п.) в реляционных СУБД обычно относятся к элементам одного столбца, а не строки; это следует учитывать при проектировании структуры БД;
- для полей типа Memo число допустимых операций по их обработке сильно ограничено по сравнению с полями других типов.

Число подобных примеров можно продолжить.

Резюме. При проектировании БД необходимо знать: особенности языков манипулирования данными в целевой СУБД и особенности предполагаемой обработки данных и учитывать их при проектировании структуры БД.

8) *Объем требуемой памяти*. В связи со значительным ростом технических характеристик накопителей и снижением стоимости хранения единицы информации значимость данного фактора постоянно снижается. Исходными данными для определении требуемого объема памяти являются: число объектов отображаемой предметной области, особенности выбранной логической и физической структуры БД, особенности носителя данных. Некоторые CASE-средства включают в себя блоки оценки объемов памяти.
9) *Скорость (время) обработки информации* (время реакции на запрос). Значение данного критерия очень трудно достаточно точно оценить на стадии проектирования, т.к. на величину этого показателя влияет значительное число взаимосвязанных и взаимозависимых факторов. Если для определения требуемого объема памяти обычно используются аналитические методы, то для определения времени обработки это проблематично. Чаще всего «скоростные» характеристики определяются путем проведения специальным образом подобранных тестов. Однако факторы, влияющие на скорость обработки, известны, и их надо иметь в виду при проектировании структуры БД.

Рассматриваемый критерий особенно важен для систем, работающих в реальном масштабе времени и в интерактивном режиме.

Перечень приведенных критериев не является исчерпывающим. Кроме того, каждый из перечисленных критериев может быть разбит на множество детализирующих его показателей.

Рассмотрим некоторые примеры, иллюстрирующие оценку тех или иных проектных решений по перечисленным выше критериям. Следует обратить внимание на то, что оценка проекта по какому-либо критерию будет зависеть как от характеристики отображаемой предметной области, так и от особенностей используемой СУБД. Например, некоторые СУБД не позволяют корректировать ключевое поле. Если это не учесть и выбрать при проектировании в качестве ключа поле, которое может изменить свое значение, то в случае возникновения в предметной области такой ситуации, ее отражение в БД потребует значительных затрат, а именно потребуются перепроектирование структуры БД и довольно трудоемкая процедура «перезагрузки» данных из старой БД в новую. Если СУБД позволяет корректировать значение ключевого поля, то таких «перестроек» не потребуется (т.е. показатель оценки одного и того же проектного решения по критерию «устойчивость модели» для разных СУБД будет выглядеть по-разному.) Из сказанного не следует делать вывод, что СУБД, позволяющие корректировать ключ, лучше, чем те, которые не позволяют этого: и модель БД получается устойчивее, и проектировать легче (меньше факторов надо учитывать при проектировании). Если Вы выберете в качестве ключа поле, значение которого может изменяться, то у Вас могут возникнуть проблемы при поддержании целостности БД.

Резюме. При проектировании БД надо, с одной стороны, оценить предметную область с точки зрения ее изменчивости, а с другой стороны – проект БД с точки зрения затрат на отображение возможных изменений в предметной области в информационной системе.

Рассмотрим еще один пример, иллюстрирующий оценку проектного решения по некоторым из перечисленных выше критериев.

При ручной организации учета успеваемости в вузах обычно ведется «Книга баллов», в которой каждой группе соответствует своя страница, по строкам которой размещен список студентов, по столбцам – названия дисциплин, которые они изучают в данном семестре. На пересечении строки и столбца проставляется соответствующая отметка.

Если эту систему перенести без перепроектирования в машинную форму, то это повлечет за собой создание по меньшей мере стольких файлов / таблиц с разной структурой, сколько имеется разных учебных планов. Число полей в каждом файле будет велико. Без знания учебного плана спроектировать структуру такой БД нельзя.

Общее число файлов БД будет еще больше, т.к. для каждого семестра придется делать свой файл, т.к. в противном случае надо указывать где-то семестр, в котором студент сдавал экзамен; особенно это важно, если по одному и тому же предмету сдается несколько экзаменов.

Каждый раз, когда в учебный план вводятся новые дисциплины, придется менять структуру БД.

Обрабатывать БД с такой структуру чрезвычайно тяжело. Например, чтобы знать, какие оценки у студента X по физике, надо знать, в какой группе учится данный студент, какая / какие таблицы содержат сведения об его успеваемости, сколько оценок по физике он имеет, и как называются соответствующие поля, в которых отражены эти оценки. Многие другие запросы также сформулировать будет сложно. Т.е. такое решение плохо по всем основным показателям: плохие адаптивные свойства, велика сложность структуры и обработки. Универсальность системы равна 0.

Если предложить другой вариант, а именно, создать отношение «УСПЕВАЕМОСТЬ» с полями «КОД_СТУДЕНТА», «ПРЕДМЕТ», «СЕМЕСТР», «ОЦЕНКА», то все указанные недостатки, отмеченные для первого проектного решения, будут устранены.

Недостатком же второго варианта по сравнению с первым будет большая степень дублирования данных: в первом варианте ФИО / КОД_СТУДЕНТА фиксируется только один раз как значение поля, а названия предметов вообще фиксируются только в описании структуры, а во втором варианте они будут повторены как значения соответствующих полей при фиксации каждого факта сдачи экзамена. Но, несмотря на указные недостатки, предпочтение все равно должно быть отдано этому варианту.

Значимость критериев для каждого конкретного проекта будет зависеть от многих объективных и субъективных факторов. Прежде всего, эта значимость зависит от самого критерия: есть критерии, значимость которых не только никогда не понижается, а, наоборот, постоянно растет (это такие критерии, как адекватное отображение действительности, удовлетворение разнообразных потребностей пользователей и т.п.). Значимость других критериев становится менее существенной с ростом возможностей техники и программного обеспечения (например, занимаемый базой данных объем памяти).

На значимость критерия оценки влияют особенности ИС, для которой создается проект (например, для систем, работающих в реальном масштабе времени, очень важна скорость реакции системы на запросы, для банковских систем – защита информации и надежность системы).

Все критерии оценки являются взаимосвязанными и часто противоречивыми: улучшение показателей по одному из критериев может привести к ухудшению значений показателей, оценивающих модель по другим показателям. Необходима комплексная оценка проекта по всей совокупности критериев.

3.2. ОСОБЕННОСТИ ДАТАЛОГИЧЕСКИХ МОДЕЛЕЙ

Внутризаписная структура

В базах данных со структурированными моделями следует различать *внутризаписную* и *межзаписную* структуры. Внутризаписная структура может быть либо *линейной*, либо *иерархической*. При линейной структуре запись состоит из простых элементов (часто называемых полями), которые следуют в записи одно за другим, т.е. структура записи является нормализованной.

В случае иерархической внутризаписной структуры в состав записи могут входить не только простые, но и составные компоненты. Это могут быть векторы (когда повторяются однотипные элементы), повторяющиеся группы (когда в записи может присутствовать несколько экземпляров составных единиц информации, включающих в себя несколько разнотипных элементов), а также неповторяющиеся составные единицы информации внутри записи. Например, если мы имеем запись ЛИЧНОСТЬ, то в ее состав могут входить простые элементы, такие как ТАБЕЛЬНЫЙ _ НОМЕР, ФАМИЛИЯ и т.д., вектор ИНОСТРАННЫЙ _ ЯЗЫК (предполагается, что человек может владеть несколькими иностранными языками), повторяющаяся группа ПОСЛУЖНОЙ _ СПИСОК, включающая элементы: ДАТА _ НАЗНАЧЕНИЯ, ДАТА _ УВОЛЬНЕНИЯ, МЕСТО _ РАБОТЫ, ДОЛЖНОСТЬ, а также неповторяющаяся группа АДРЕС, состоящая из элементов ГОРОД, УЛИЦА, ДОМ, КВАРТИРА.

Иерархическая структура записи может быть как *одноуровневой*, так и *многоуровневой*. Принципиально возможны довольно сложные структуры, например, когда в состав повторяющейся группы в качестве составляющего компонента входит другая повторяющаяся группа. Однако по разным причинам (в частности, из-за сложности реализации) в конкретных СУБД имеются различные ограничения (например, повторяющаяся группа может существовать только на первом уровне иерархии, ограничивается число уровней иерархии и т.п.).

Записи могут быть с *постоянным* и *переменным* составом. Последнее обычно понимается так: если значение какого-либо компонента записи отсутствует для конкретного объекта, то и сам этот компонент в данной записи отсутствует. Например, если один из сотрудников окончил вуз, имеет ученую степень и ученое звание, то название вуза, год его окончания, ученая степень, ученое звание и даты их присвоения хранятся в записи, соответствующей данному сотруднику. Если у другого сотрудника все эти признаки отсутствуют, то в соответствующей ему записи эти поля также отсутствуют. В случае, если записи имеют постоянный состав, то все поля, описанные в структуре записи, всегда присутствуют в каждом экземпляре записи, но некоторые из этих полей могут быть «пустыми».

Другой характеристикой записи является тип ее длины. По этому признаку различают записи с *фиксированной* (постоянной), *переменной* и *неопределенной* длиной. Запись может иметь переменную длину в результате того, что переменную длину имеют ее поля, либо возможно отсутствие каких-либо полей, либо допускается разное число экземпляров для повторяющихся компонентов.

Поле является наименьшим семантически значимым элементом записи. Основными характеристиками поля является его тип и длина. Существующие СУБД различаются по набору поддерживаемых типов полей, но наблюдается тенденция к расширению этого набора. Большинство современных СУБД, кроме привычных полей символьного и числового типа, допускают использование полей типа даты, логических полей, а также полей денежного типа. Некоторые системы позволяют вводить определяемые пользователем типы полей. В последнее время все большее распространение получают мультимедийные формы представления данных.

Межзаписная структура

Традиционное деление СУБД по типу модели данных на реляционные, иерархические и сетевые основывается на характере связей между записями. При всей разнице в терминологии можно считать, что основными компонентами любой из этих моделей являются файлы, которые состоят из записей.

Иерархические модели

В классических иерархических моделях имеется один файл, который является входом в структуру (корень дерева). Остальные файлы связаны друг с другом таким образом, что каждый из них, за исключением корневой вершины, имеет ровно одну исходную вершину («родитель») и любое число подчиненных вершин («детей»). Между записью файла-«родителя» и записями порожденного файла имеется отношение 1 : M (как частный случай может быть и отношение 1 : 1).

Имеются и другие разновидности иерархических моделей, но они менее распространены.

Сетевые модели

В сетевых моделях, если на них не накладывается никаких ограничений, в принципе, любой файл может быть точкой входа в БД, каждый из файлов может быть связан с произвольным числом других файлов, и между записями связанных файлов могут быть любые отношения (1 : 1, 1 : M, M : M). Однако в реальных СУБД на модель накладываются различные ограничения. Так, имеются *сетевые* СУБД *с разнотипными файлами*. В них все файлы разделены на два типа: основные и зависимые. В таких СУБД входом в базу данных могут служить только основные файлы, а связываться между собой могут только разнотипные файлы.

Во многих сетевых СУБД не поддерживается непосредственно отношение M : M. В таких моделях каждая связь между парой файлов определяется отдельно, и для каждой из них один файл в этой паре объявляется «владельцем», а другой – «членом». Отношение между записью-«владельцем» и записями-«членами» – 1 : M.

Связи между файлами в иерархических и сетевых моделях определяются при описании структуры базы данных и физически чаще всего передаются при помощи различных указателей.

Реляционные модели

В реляционной модели используется своеобразная терминология, но это не меняет сущности модели. Часто даже в рамках одной модели в разных СУБД используется разная терминология. Так, на логическом уровне элемент чаще всего называют атрибутом; кроме того, для него используются термины «колонка», «столбец», «поле». Совокупность атрибутов образует строку (синонимичные термины – «ряд», «запись», «кортеж»). Совокупность строк образует отношение («таблицу», «файл базы данных»). Понятие базы данных как множества отношений поддерживается далеко не всеми реляционными СУБД (т.е. при создании БД описываются отдельные отношения (файлы, таблицы), а для всей базы данных как самостоятельной информационной единицы, никакого описания не предусмотрено). Хотя следует отметить, что в последнее время новые версии даже тех «настольных» СУБД, которые раньше не поддерживали понятие БД, сейчас его включают.

Связи между файлами в реляционной модели в явном виде могут не описываться. Они устанавливаются динамически в момент обработки данных по равенству значений соответствующих полей.

В сетевых и иерархических моделях структура записи, в принципе, может быть любой. Хотя многие СУБД накладывают различные ограничения на структуру записи. В реляционных моделях структура записи должна быть линейной.

Каждое отношение по определению имеет *ключ*, т.е. атрибут (простой ключ) или совокупность атрибутов (составной ключ), однозначно идентифицирующих кортеж. В некоторых случаях в отношении могут быть несколько *возможных (вероятных, альтернативных) ключей*.

К сожалению, не все реляционные СУБД поддерживают концепцию ключа, т.к. в этом случае многие проблемы (в частности, обеспечение проверки на уникальность ключа и соблюдение некоторых других ограничений целостности) возлагается на пользователя (проектировщика ИС). Вне зависимости от того, требует ли СУБД явного указания ключей при описании отношений или нет, проектировщик базы данных должен понимать, что является ключом каждого отношения.

В случае, если СУБД поддерживает концепцию ключа, при наличии нескольких вероятных ключей один из них выбирается и описывается как *первичный ключ*, остальные – как альтернативные (обычно для этих целей используется либо уникальный индекс, либо описатель UNIQUE при определении таблицы).

Атрибут или группа атрибутов, которая в рассматриваемом отношении не является ключом, а в другом отношении ключом является, называется *внешним ключом*.

Если какая-то таблица содержит внешний ключ, то она: а) логически связана с таблицей, содержащей соответствующий первичный ключ; б) эта связь имеет характер «одинко-многим» (таблица, содержащая внешний ключ, находится на стороне «много» в этой связи). В частном случае это может быть связь 1 : 1. Связь M : M в реляционной базе данных непосредственно не поддерживается.

По сути, понятия «родитель»-«ребенок» в иерархических моделях, файл-«владелец»-файл-«член» в сетевых моделях и связь «ключ»-«внешний ключ» в реляционных моделях передают одно и то же явление – наличие связи 1 : M между записями соответствующих файлов.

В реляционных СУБД часто используется понятие *«взгляд» (view).* Он представляет собой виртуальную таблицу, часто полученную в результате логического соединения нескольких связанных по значениям общих столбцов таблиц и, возможно, включающую некоторое подмножество из всей совокупности строк, отобранное по заданному условию. Это понятие расширяет традиционное для банков данных понятие «подсхема».

Понимание различий и общности моделей разных классов позволяет использовать общий подход при проектировании структуры баз данных, осуществлять преобразование одних моделей в другие, использовать средства, в частности, языковые, предназначенные для работы с одним классом моделей, при работе с другим.

Системы, построенные на основе инвертированных файлов

Существует еще одна разновидность моделей данных – *системы, построенные на основе инвертированных файлов*. С точки зрения логической структуры БД эти модели следует отнести к сетевым. Более того, это единственный вид моделей, которые в явном виде поддерживают отношение M : M между файлами БД. Особенности этих моделей заключены в способе отражения связей между информационными единицами. В системах,

построенных на основе инвертированных файлов, собственно хранимые данные и информация о связях между информационными единицами логически и физически отделены друг от друга. Основные данные в этих системах хранятся в файлах, записи которых могут иметь многоуровневую иерархическую структуру. Вся управляющая информация сосредоточена в так называемом ассоциаторе. Логическая связь между файлами устанавливается посредством компонента ассоциатора, называемого сетью связи. Отделение ассоциативной информации от собственно хранимых данных позволяет изменять связи, не изменяя при этом самих файлов.

3.3. ПРОЕКТИРОВАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

Вводные положения

Для реляционной базы данных проектирование логической структуры заключается в том, чтобы разбить всю информацию по файлам[13] (или в терминах реляционной модели – по отношениям, таблицам), а также определить состав полей (в терминах реляционной теории – атрибутов) для каждого из этих файлов. Определение ключа каждого из отношений так же является задачей логического проектирования реляционной БД.

Сейчас многие реляционные СУБД позволяют декларативно задавать связи между таблицами при описании БД, а также определять необходимость контроля и способы обеспечения целостности по связям для БД. Решение этих вопросов также следует отнести к даталогическому проектированию. Другие ограничения целостности (кроме ограничения на уникальность и ограничения по связи) в данном разделе рассматриваться не будут.

Часто при описании логической структуры реляционной БД сразу же указывается, по каким полям надо индексировать соответствующий файл, а для ключевых полей автоматически предусматривается индексация. Индексация занимает «промежуточное» положение между логической и физической структурой данных: с одной стороны она определяет способ упорядочения данных и доступ к ним, а с другой – это способ «логического упорядочения», при котором создаются вспомогательные индексные файлы, что меняет общую структуру БД. Вопросы индексирования будут частично рассмотрены в данном разделе.

Существуют разные методы проектирования логической структуры реляционных баз данных. Среди них есть и строгие математические методы, обычно базирующиеся на теории нормализации. Они имеют очень большое значение в качестве теоретической основы проектирования БД, но, в связи с вычислительной сложностью алгоритмов, практически не используются в реальном проектировании систем.

Мы рассмотрим метод проектирования, основанный на анализе ER-модели и переходе от нее к реляционным отношениям. В основу этого метода положен эмпирический подход. Предлагаемый метод является достаточно простым и наглядным, и в то же время дает хорошие результаты. Базы данных, полученные в результате применения излагаемой ниже методики проектирования, находятся в 4-ой нормальной форме. Следует отметить, что большинство имеющихся в настоящее время CASE-средств также используют аналогичный подход, но сам алгоритм проектирования обычно нигде не публикуется и может быть «воспроизведен» лишь «экспериментально», путем анализа проектных решений, полученных при преобразовании тех или иных ER-конструкций в схему целевой БД.

[13] В некоторых СУБД каждой таблице ставится в соответствие файл базы данных, в других – вся база данных хранится в одном файле. В этом параграфе термины «файл», «таблица» и «отношение» используются как синонимы и не отражают способ физического хранения данных.

Алгоритм перехода от базовой ER-модели к схеме реляционной базы данных

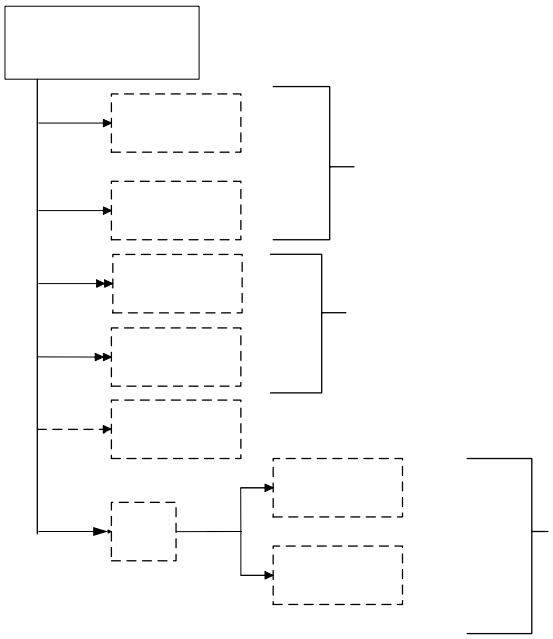
Ниже описан алгоритм перехода от базовой ER-модели к схеме реляционной базы данных. Каждый элемент ER-модели находит свое отражение в схеме базы данных. Для некоторых ситуаций в ER-модели возможно использование нескольких альтернативных решений при их отображении в модель базы данных. Выбор наиболее подходящего решения будет зависеть от разных факторов, часть из которых отображена в ER-модели, а другая часть – нет, т.е. для выбора проектного решения кроме информации непосредственно из ER-модели необходимо дополнительно использовать информацию и из других компонент концептуальной модели предметной области.

Рассмотрим рекомендации по переходу от базовой ER-модели к схеме реляционной базы данных для каждого из типов элементов ER-модели.

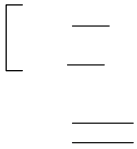
Отображение простых объектов

Определение состава полей основной таблицы. Отображение единичных свойств объекта

Для каждого простого объекта и его единичных свойств строится отношение, атрибутами которого являются идентификаторы объекта и реквизиты, соответствующие каждому из единичных свойств (рис.3.1).



а) Простой объект. Фрагмент ER-модели



б) Отображение объекта в реляционных таблицах

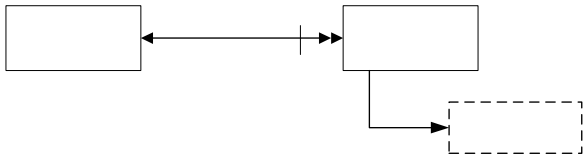
Рис. 3.1. Отображение единичных, множественных и составных свойств

Определение ключа таблицы

Любой из уникальных идентификаторов объекта является вероятным ключом полученного отношения. Если объект имеет несколько уникальных идентификаторов, необходимо один из них выбрать в качестве первичного ключа. Часто в качестве первичного ключа выбирается самый короткий из вероятных ключей. Но это не всегда должно быть обязательно так. На решение вопроса о выборе первичного ключа (кроме длины ключа) влияют следующие факторы:

1. Стабильность – может ли значение ключа изменяться. Несмотря на то, что многие современные СУБД не только не запрещают изменять значение первичного ключа, но и имеют специальные механизмы, позволяющие автоматически осуществлять изменения связанных записей (каскадное изменение), не следует злоупотреблять этой возможностью. Желательно выбирать в качестве первичного ключа атрибуты, которые не изменяются.
2. Мнемоничность – легкость запоминания. Т.к. в качестве ключа обычно используются короткие обозначения, то при прочих равных условиях следует отдавать предпочтение тем из вероятных ключей, которые легче запомнить.

Среди названных выше критериев наиболее важным является стабильность. Например, если у объекта «КАФЕДРА» есть три идентификатора: «НАИМЕНОВАНИЕ _ КАФЕДРЫ _ ПОЛНОЕ», «НАИМЕНОВАНИЕ _ КАФЕДРЫ _ КРАТКОЕ» и «КОД _ КАФЕДРЫ», то даже если «КРАТКОЕ _ НАИМЕНОВАНИЕ» короче других полей, скорее всего, в качестве первичного ключа будет выбираться «КОД» (потому, что наименования кафедр подвержены достаточно частым изменениям). Поэтому в алгоритме при выборе первичного ключа в качестве «решения по умолчанию» принимается более короткий идентификатор, но от пользователя требуется либо подтвердить это решение, либо выбрать иной первичный ключ.



а) Изображение зависимой по идентификации сущности. Фрагмент ER- модели

б) Отображение зависимой по идентификации сущности

Рис. 3.2. Отображение зависимой по идентификации сущности

Если объект является **зависимой по идентификации сущностью**, то ключ соответствующего ему отношения будет составной, включающий идентификатор этого объекта и идентификатор «вышестоящего» объекта или, как говорят, идентификатор «основного» объекта «мигрирует» в таблицу, соответствующий зависимому объекту (рис. 3.2). Если идентификаторов у «главного» объекта несколько, то выбирается один из них, а именно тот, который выбран в качестве первичного ключа «основной» сущности. Полученный таким образом составной идентификатор зависимой по идентификации сущности будет использоваться во всех тех случаях, когда надо отображать связь этого «зависимого» объекта с другими.

Обычно зависимый по идентификации объект и тот объект, от которого он зависит, связаны друг с другом отношением 1 : M, и может сложиться впечатление, что перенесе-

ние ключа просто соответствует отображению этой связи. Однако это не совсем так. В случае, если сущность независима по идентификации, то связь 1 : М можно отображать в структуре БД как путем переноса ключа связанного объекта в таблицу, соответствующую подчиненному объекту (т. е. объекту, стоящему со стороны М), так и другими способами, а ключом таблицы, соответствующей подчиненному объекту, будет являться только идентификатор самого этого объекта. В случае зависимого по идентификации объекта связь 1 : М дополнительно в схеме БД отображать не надо.

Некоторые СУБД (например, Access, Paradox и др.) позволяют автоматически генерировать поле типа «счетчик» в качестве ключа таблицы. Этот искусственный код вполне можно создавать для простых объектов, если в предметной области не предполагается использования другой системы кодирования объектов (например, для предприятий (или иных объектов хозяйственной деятельности) в БД все равно в большинстве случаев приходится хранить коды ОКПО, ОКОНХ, ИНН; в этом случае создавать дополнительный код может не иметь смысла). Созданные коды будут в дальнейшем использоваться для связи данного объекта с другими объектами в БД.

Если такой код использовать при создании таблицы, соответствующей агрегированному объекту, то он вряд ли он будет где-то использоваться в дальнейшем (хотя это утверждение нельзя рассматривать категорично; например, для агрегированного объекта «СДАЧА ЭКЗАМЕНА» каждая «попытка» может иметь дополнительную информацию (какие вопросы были заданы и т.п.), то для отображения этой информации может потребоваться отдельная таблица и код «сдачи экзамена» может оказаться полезным).

Другими словами, при создании таблицы в каждом конкретном случае надо решать вопрос, что выбрать в качестве ключа: естественный ключ, искусственный ключ, в том числе и созданный системой автоматически, а может быть, если СУБД это позволяет, отказаться от создания ключа вообще.

Отображение множественных свойств объекта

Если у объекта имеются множественные свойства, то каждому из таких свойств ставится в соответствие отдельное отношение, полями которого будут идентификатор объекта (если у объекта несколько идентификаторов, то тот, который выбран в качестве первичного ключа) и поле, соответствующее множественному полю. Ключ этого отношения будет составным, включающим оба эти атрибута (рис. 3.1).

Приведенное выше решение является универсальным. В отдельных случаях могут быть приняты и другие решения. Так, если число экземпляров множественного свойства у каждого из объектов невелико и в процессе обработки не возникает необходимости «выделять» каждое из этих значений, то можно все значения, относящиеся к одному объекту, хранить в одном поле. В этом случае отдельную таблицу для хранения множественного свойства создавать не надо.

Отображение условных свойств объекта

Если объект обладает условными свойствами, то при отображении их в реляционную модель возможны следующие варианты:

а) если многие из объектов обладают рассматриваемым свойством, то его можно хранить в базе данных также как и обычное свойство, т.е. в той же таблице, в которой бы атрибут хранился, если бы свойство было бы определенным для всех экземпляров рассматриваемой сущности;

б) если только незначительное число объектов обладает указанным свойством, то для многих записей в файле базы данных при использовании предыдущего решения значение соответствующего поля будет пустым. Для устранения этого недостатка можно выделить отдельное отношение, которое будет включать идентификатор объекта и атрибут, соответствующий рассматриваемому свойству. Это отношение будет содержать столько строк, сколько объектов имеет рассматриваемое свойство. Однако это решение в свою очередь имеет недостатки (в частности, усложнение структуры БД и сложности ее обработки) и применяется сравнительно редко.

На рис. 3.1 использован вариант «а». Если бы было выбрано второе из обсуждаемых решений, то из отношения R1 атрибут С5 следовало бы исключить и создать дополнительно новое отношение R4 (ИО1, С5).

Отображение составных свойств объекта

Если объект имеет составное свойство, то возможны два способа его отображения в БД:

1) всему составному свойству ставится в соответствие одно поле;
2) каждому из составляющих элементов составного свойства ставится в соответствие отдельное поле.

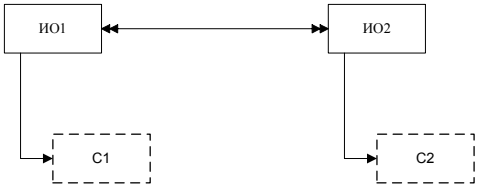
Выбор варианта будет зависеть, в основном, от характера преимущественной обработки этой информации: так как в большинстве СУБД гораздо проще при реализации запросов объединить поля, чем выделить из единого поля нужную часть, то в случае, если предполагается использование отдельных компонентов составного свойства, лучше использовать вариант 1, в противном случае – вариант 2.

Отображение связи между объектами

Связи между объектами также должны отражаться в структуре БД. Универсальным способом отображения связи между объектами является введение вспомогательного связующего файла, содержащего идентификаторы связанных объектов. Ключ этого отношения будет составным. Такое решение является практически единственно приемлемым при наличии связи М : М между объектами. Дополнительными доводами в пользу такого решения является также наличие необязательного класса членства объекта в связи. Во многих случаях можно использовать другие, более эффективные способы отображения связей в структуре БД. Выбор проектного решения, прежде всего, будет зависеть от типа связи между объектами.

Отображение связи типа М : М

Если между объектами предметной области имеется связь М : М, то для хранения такой информации потребуется три отношения: по одному для каждой сущности и одно дополнительное – для отображения связи между ними. Последнее отношение будет содержать идентификаторы связанных объектов (рис. 3.3). Ключ этого отношения будет составным.



а) Фрагмент ER-модели

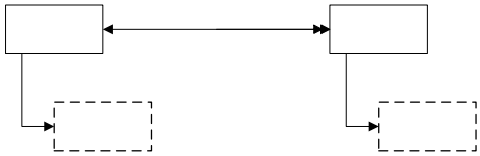
R1(ИО1,С1)
R2(ИО2, С2)
RCB(ИО2, ИО1)

б) Отображение в реляционную модель

Рис. 3.3. Отображение связи M : M

Отображение связи типа 1 : M

Если между объектами предметной области имеется связь 1 : M, то можно, также как и в случае связи M : M, использовать отдельную связующую таблицу (рис. 3.4, б – вариант 2). В отличие от связи M : M ключом связующей таблицы будет только идентификатор объекта, к которому направлен «единичный» конец связи.



а) Фрагмент ER-модели

б) Варианты отображения в реляционную модель

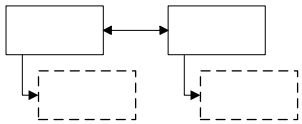
Рис. 3.4. Отображение связи 1 : M

Однако, если между объектами предметной области имеется связь 1 : M и класс принадлежности n-связной сущности является обязательным, то можно использовать только два отношения (по одному для каждой сущности) и не использовать дополнительную связующую таблицу. В отношение, соответствующее 1-связной сущности (т.е. сущности, к которой идет единичная связь), надо дополнительно добавить идентификатор связанного с ней объекта (рис. 3.4, б – вариант 1).

Если класс принадлежности n-связной сущности является необязательным**,** то появляется дополнительный довод в пользу решения о создании для отображения связи третьего отношения, которое будет содержать ключи каждой из связанных сущностей (рис. 3.4, б – вариант 2).

Отображение связи типа 1 : 1

Наличие между объектами связи типа 1 : 1 является довольно – таки редкой ситуацией в реальной жизни. В принципе, если связь между объектами 1 : 1 и класс принадлежности обеих сущностей является обязательным, то для отображения обоих объектов и связи между ними можно использовать одну таблицу (рис. 3.5, б – вариант 3). Такое решение потребует меньше всего памяти для своей реализации. Например, если имеются объекты СОТРУДНИК и ПАСПОРТ, то такое решение будет вполне приемлемым. Однако таким решением не следует злоупотреблять. Может случиться, что для каждого из объектов, находящихся в связи 1 : 1, в дальнейшем потребуется отразить какие-то свои связи, или в запросах часто требуется информация отдельно по каждому из объектов, то выбранное решение может усложнить или замедлить работу с БД.



а) Фрагмент ER-модели

б) Варианты отображения в реляционную модель

Рис. 3.5. Отображение связи 1 : 1

Если для каждого из этих объектов создаются отдельные отношения, то информацию о связях между ними можно отразить, включив в одно из отношений идентификатор связанного объекта из другого отношения. Причем, если класс принадлежности обеих сущностей является обязательным, то (если руководствоваться только типом связи) это можно сделать в любом из отношений (рис. 3.5, б – варианты 1.2).

Если класс членства одного из объектов является необязательными, то идентификатор сущности, для которой класс принадлежности является необязательным, добавляется в отношение, соответствующее тому объекту, для которого класс принадлежности – обязательный.

Если степень связи между объектами равна 1:1 и класс принадлежности каждой из них является необязательным, то, чтобы избежать наличия пустых полей, следует использовать три отношения: по одному для каждой сущности и одно – для отображения связи между ними (рис. 3.5, б – вариант 4). В приведенном решении в качестве ключа связующей таблицы обозначен ИО1. С таким же успехом мог быть выбран ИО2.

Отображение альтернативной связи

Альтернативная связь обычно используется при изображении агрегированного объекта и означает, что в действии участвует либо один объект, либо другой, но не оба вместе. Альтернативная связь трудна для ее «автоматического» преобразования в даталогическую модель. Может быть в связи с этим она отсутствует в большинстве CASE-средств.

Естественным кажется путь, при котором в таблице базы данных, соответствующей объекту, к которому идут альтернативные связи, всем этим связям будет соответствовать одно поле, в котором будет зафиксирован идентификатор связанного объекта. В экземпляре записи в этом поле будет записано значение идентификатора того объекта, который участвует в отображаемой связи в каждой конкретной ситуации. Но такое решение имеет множество недостатков, связанных с последующей обработкой таким образом спроектированных таблиц, и его, в большинстве случаев, не рекомендуется использовать.

Другой вариант решения: для отражения связи с каждым из альтернативных классов объектов использовать отдельную таблицу.

Часто объекты, объединенные альтернативной связью, по сути, являются подклассами обобщенного класса. Иногда имеет смысл по иному изобразить ER-модель, чтобы увидеть другие варианты проектных решений.

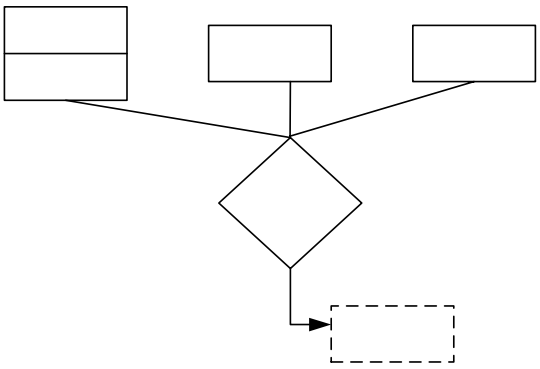
Отображение сложных объектов

Выше мы рассматривали варианты проектных решений, связанные с простыми объектами. Но в ER-модели отражаются и сложные объекты.

Отображение агрегированных объектов

Каждому агрегированному объекту, имеющему место в предметной области, в реляционной модели будет соответствовать отдельное отношение. Атрибутами этого отношения будут являться идентификаторы всех объектов, «задействованных» в данном агрегированном объекте, а также реквизиты, соответствующие свойствам этого агрегированного объекта.

Для отношений, соответствующих агрегированным объектам, ключ будет составной. В большинстве случаев им будет являться конкатенация (соединение) идентификаторов объектов, «участвующих» в этом агрегированном объекте.

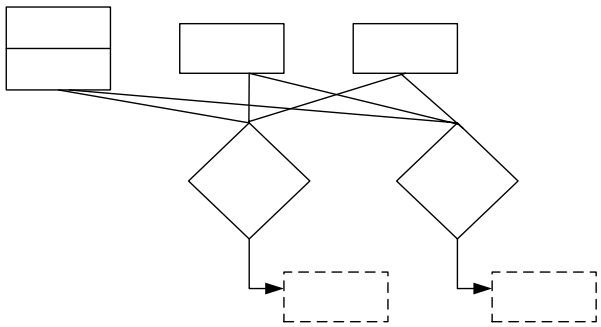


а) Фрагмент ER-модели

б) Отображение в реляционную модель

Рис. 3.6. Отображение агрегированного объекта

Объединить информацию о нескольких агрегированных объектах в одно реляционное отношение можно только в том случае, если те объекты, с которыми связан каждый из них, полностью совпадают. Это является необходимым, но не достаточным условием для такого объединения. В каждом конкретном случае возможность и необходимость такого объединения надо определять особо.



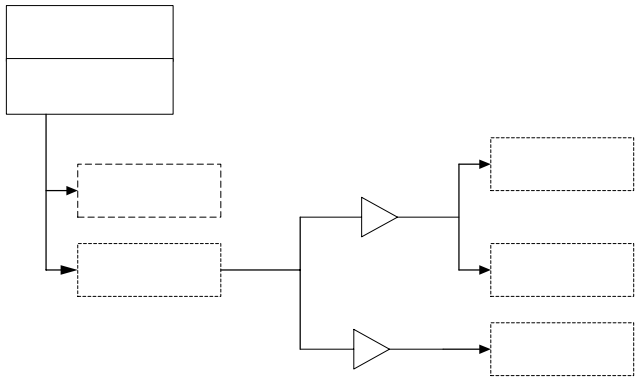
а) Фрагмент ER-модели

б) Отображение в реляционную модель

Рис. 3.7. Отображение нескольких агрегированных объектов, имеющих одинаковые связи

Отображение обобщенных объектов

При отображении обобщенных объектов могут быть приняты разные решения.



а) Фрагмент ER-модели

б) Варианты отображения в реляционную модель

Рис.3.8. Изображение обобщенного объекта

Во-первых, всему обобщенному объекту может быть поставлен в соответствие одна таблица базы данных (рис.3.8. б – вариант 1). В этом случае атрибутами этой таблицы будут идентификаторы обобщенного объекта, все единичные свойства, присущие объектам хотя бы одной категории, включая свойство, по которому производится разбиение на подклассы. Ключом таблицы будет один из идентификаторов этого объекта.

Другим «крайним» вариантом является решение, при котором каждой из категорий объектов нижнего уровня ставится в соответствие отдельное отношение (рис.3.8. б – вариант 2). В этом случае каждое отношение будет включать в себя идентификатор объекта (если идентификаторов несколько, то в каждое из отношений будут включены все они; это не приведет к дублированию информации на уровне значений), свойства, присущие родовым объектам, а также свойства, присущие данному подвиду объектов. Свойство, по которому производится разбиение класса на подклассы, в этом случае в качестве поля не включается ни в одно из отношений.

Кроме этих двух «крайних» решений возможны и комбинированные варианты. Например, можно выделить общую таблицу для отображения «родовых» свойств объектов (включающую еще и все идентификаторы объекта) и отдельные таблицы для отображения «видовых» свойств (такой алгоритм используется в системе Design/IDEF); Кроме свойств, присущих видовому объекту, в каждом из этих отношений будет повторен ключевой атрибут «основного» отношения (рис.3.8. б – вариант 3).

Другим вариантом проектного решения для отображения обобщенного объекта является использование так называемого «кодированного формата файла», при котором, так же как и варианте 1, используется одна таблица, но для всех «видовых» свойств каждого из подклассов выделяется одно поле, содержимое которого распознается по значению свойства, по которому производится разбиение класса на подклассы.

Перечень вариантов можно продолжить. Выбор конкретного решения будет зависеть от многих факторов, в том числе, от того насколько часто информация о разных категориях объектов обрабатывается совместно, как велико различие в «видовых» свойствах и некоторых других факторов.

Приведенный выше алгоритм излагался из предположения, что классификация объектов не являлась «фасетной». Если в обобщенном объекте наблюдается разбиение на подклассы по разным несоподчиненным признакам, то варианты 1 и 3 останутся верны, а вариант 2 должен быть уточнен.

Кроме того, алгоритм не учитывает, что классы могут быть пересекающимися. Для пересекающихся классов нельзя без модификации использовать вариант А (так как признак классификации у каждого из экземпляров объекта может иметь несколько значений), но может быть использован вариант В.

Кроме того, при выборе проектного решения необходимо учитывать, является ли класс полным или нет. Если класс неполный, то при выборе варианта, когда для каждого подкласса строится отдельная таблица, информация об объектах, не вошедших ни в один подкласс, может просто пропасть.

Отображение составных объектов

Как отмечалось в гл. 2, в базовой ER-модели, как и в большинстве других нотаций, нет специальных обозначений для отображения связи «целое-часть» или составного объекта.

Наличие такой связи может быть отображено как в инфологической, так и в даталогической модели по-разному. Следует отметить, что само отношение «целое-часть» может качественно различаться для разных ситуаций. Так, если речь идет о составе изделий,

то между «ИЗДЕЛИЕМ» и «ДЕТАЛЬЮ» имеется связь типа М : М, так как одна и та же деталь может входить в разные изделия и, наоборот, в изделие входят разные детали. Состав изделия обычно является сложным, и отражать его в явном виде в структуре базы данных нежелательно, а часто и просто невозможно. Кроме того, рассматриваемая связь реализована на однородном множестве объектов. В этом случае для отображения связи «целое-часть» можно воспользоваться двумя файлами базы данных. Первый из них будет хранить информацию о самих объектах, а второй – информацию о связи между ними, а также дополнительную информацию, характеризующую эту связь. Для состава изделия это могут быть поля «что входит», «куда входит» и «количество» (пис. 3.9).

Отношение «целое-часть» может отражать, например, структуру какой-то организации. В этом случае ему, скорее всего, будет соответствовать связь типа 1 : М, и для его отображения в даталогической модели можно использовать рекомендации, данные для соответствующего случая. В рассматриваемой ситуации также можно воспользоваться неявным выделением уровней, но такой прием используется при отображении организационной структуры редко.

Использование дополнительных характеристик концептуальной модели.

- Характеристики свойств динамические («Д») и статические («С») могут быть использованы при задании ограничений целостности (например, можно задать запрет на обновление для статических полей). Кроме того, эта информация может быть полезна при выборе ключа отношения, а также способов физической организации данных.
- Характеристики «число объектов», «рост числа объектов», «летучесть» могут быть использованы для управления размещением данных на носителе, выборе методов организации данных и доступа к ним, определения объемных характеристик БД (что относится к стадии физического проектирования БД). Эти характеристики также могут быть приняты во внимание при решении вопросов о целесообразности разбиения файла на несколько самостоятельных файлов.
- «Класс членства» объектов в связи оказывает влияние не только на выбор варианта построения логической структуры, но и на задание ограничений целостности.
- Информация о степени пересечения классов объектов (граф пересечений) может учитываться при выборе варианта отображения обобщенного объекта, а также при контроле целостности базы данных.

Дополнительные рекомендации по проектированию БД

Реляционная база данных, полученная в результате использования предложенного выше алгоритма проектирования, будет являться нормализованной и автоматически находится в четвертой нормальной форме.

Как известно, в принципе, вся предметная область может быть представлена в виде одного универсального отношения. Недостатки такого способа хранения известны, и нормализация отношений служит устранению этих недостатков. Нормализация выполняется путем вертикального разбиения (проекции) исходного отношения.

Однако хранение нормализованных файлов в свою очередь может привести к отрицательным последствиям, например, к замедлению выполнения некоторых операций. Поэтому в некоторых случаях сознательно идут на денормализацию отношений.

Нормализованные отношения в свою очередь могут быть подвергнуты вертикальному разбиению, например, в случае, если информация, хранящаяся в одном отношении,

редко обрабатывается совместно, или если ограничения СУБД не позволяют хранить все атрибуты в одном отношении. Вопросы вертикального разбиения нормализованных отношений выходят за рамки теории нормализации, но их часто приходится рассматривать в реальной практике проектирования ИС. Также за рамками теории нормализации остаются вопросы горизонтального разбиения отношений.

На решение вопросов о необходимости и степени вертикального и горизонтального разбиения отношений оказывает влияние много факторов, и не только напрямую связанных с оценкой структуры данных (объем занимаемой памяти, степень дублирования данных), но и с затратами времени на выполнение операций разного типа, а также корректностью получаемого при этом результата.

Горизонтальное разбиение, в принципе, может давать как непересекающиеся, так и пересекающиеся множества. При получении непересекающихся множеств степень дублирования данных не изменится, и объем памяти, занимаемый под данные, останется таким же, как и до расщепления. Однако общий объем памяти, требуемый для хранения БД, возрастет. Это произойдет за счет большего объема служебной информации (каждое отношение должно быть самостоятельно описано), а также за счет неиспользованного свободного места в конце каждого физического блока (сектора). Кроме того, возрастет сложность базы данных (одним из показателей которой является число информационных единиц в структуре БД) и сложность обработки. При пересекающихся множествах, кроме того, возрастет степень дублирования данных и возникнут проблемы с поддержанием целостности данных.

Несмотря на очевидные недостатки горизонтального разбиения, к такому приему достаточно часто прибегают в практике проектирования. Чем это бывает вызвано? Во-первых, горизонтальное разбиение может обеспечить сокращение времени обработки данных, в том числе и за счет распараллеливания выполнения операций. Так, выполнение операций селекции, проекции над горизонтальными сечениями эквивалентны выполнению этих операций над всем отношением и могут выполняться параллельно. При этом общее требуемое время выполнения запроса будет меньше, чем t / p, где t – время выполнения запроса над всем отношением, p – число доступных процессоров, так как время еще может сократиться и за счет работы с более короткими файлами.

Однако не все операции можно непосредственно выполнить над горизонтальными сечениями отношений. Рассмотрим следующий пример. Пусть в БД учебного института хранится информация о студентах. Основным потребителем этой информации являются деканаты соответствующих факультетов. Информация чаще всего обрабатывается и анализируется в пределах факультетов. В этом случае целесообразно сделать горизонтальные сечения и хранить данные в разрезе каждого факультета отдельно. Однако, если Вам, например, потребуется получить средний балл успеваемости всех студентов в последнюю сессию, то Вам либо придется объединить соответствующие файлы, либо использовать достаточно сложный алгоритм вычисления требуемого показателя.

Горизонтальное разбиение отношения может производиться по разным принципам. В качестве условия разбиения может использоваться значение какого-либо атрибута (как, например, в рассмотренном выше примере – значение атрибута ФАКУЛЬТЕТ). Довольно часто используется разбиение по временному признаку, например, данные за каждый месяц хранятся в отдельном файле. Могут использоваться и другие принципы разбиения: по достижению определенного объема файла, по активности записей и др.

Это следует запомнить:

На построение логической модели базы данных оказывает влияние множество факторов, и все они в комплексе должны быть учтены при создании даталогической модели.

Подход к проектированию логической структуры базы данных, рассмотренный для реляционной модели, может быть применен и при проектировании других структурированных баз данных. При этом в алгоритме перехода от ER-модели к модели, поддерживаемой конкретной СУБД, должны быть учтены особенности модели данных целевой СУБД.

Т.к. базовая ER-модель является объектной по своей сути, то она может быть использована и при создании объектных баз данных.

Использование CASE-средств позволяет улучшить качество создаваемых проектов, а при создании крупных корпоративных систем является практически неизбежным. Использование CASE-средств не освобождает проектировщика от понимания не только общей сущности, но и деталей даталогического проектирования.

Контрольные вопросы

1. Что называется даталогическим проектированием?
2. Какая информация является исходной для даталогического проектирования?
3. В чем заключается проектирование логической структуры базы данных для каждого из известных Вам классов СУБД или конкретных СУБД?
4. Какие критерии используются для оценки спроектированной базы данных?
5. В каких случаях и для обеспечения каких целей вводятся искусственные идентификаторы?
6. Как отображается простой объект и его единичные свойства в реляционной базе данных? В других известных Вам СУБД?
7. Как отображаются условные свойства объектов в реляционной базе данных? В других известных Вам СУБД?
8. Как отображаются множественные свойства объектов в реляционной базе данных? В других известных Вам СУБД?
9. Как отображается отношение типа 1 : 1 между объектами в реляционной базе данных? В других известных Вам СУБД? Влияет ли при этом класс принадлежности объектов на число требуемых файлов / таблиц?
10. Как отображается отношение типа 1 : M между объектами в реляционной базе данных? В других известных Вам СУБД? Влияет ли при этом класс принадлежности объектов на число требуемых файлов / таблиц?
11. Как отображается отношение типа M : M между объектами в реляционной базе данных? в других известных Вам СУБД? Влияет ли при этом класс принадлежности объектов на число требуемых файлов / таблиц?
12. Как отображается в реляционной модели составной объект? В других известных Вам СУБД?
13. Как отображается в реляционной модели обобщенный объект?
14. Как отображается в реляционной модели агрегированный объект?
15. Все ли показатели, отображенные в инфологической модели, должны включаться в базу данных?
16. Какие факторы влияют на принятие решения о том, какие показатели следует хранить в базе данных?
17. В каком случае надо производить вертикальное и горизонтальное разбиение файлов / таблиц базы данных?

4. Литература

1. Martin James. Fourth-generation languages. – Vol. 1. – New Jersy: Prentice-Hall, Inc., 1989.
2. Hansen Gary W., Hansen James V., Database Management and Design. – New Jersy: Prentice-Hall, Inc., 1992.
3. Вендров А.М. Case-технологии. Современные методы и средства проектирования информационных систем. – М.: Финансы и статистика, 1988.
4. Дейт К. Дж. Введение в системы баз данных, 6-е изд. / Пер. с англ. – СПб.: Издательский дом «Вильямс», 2000.
5. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ / Пер. с англ. – М.: Мир, 1991.
6. Диго С.М. Проектирование баз данных: учебник. – М.: Финансы и статистика, 1988.
7. Калянов Г.Н. CASE – структурный системный анализ. – М.: ЛОРИ, 1996.
8. Когаловский М.Р. Энциклопедия технологий баз данных. – М.: Финансы и статистика, 2002.
9. Мишенин А.И. Теория экономических информационных систем. – М.: Финансы и статистика, 1999.
10. «О правовой охране программ для электронных вычислительных машин и баз данных» Закон № 3523-1, 23.09.92.
11. Общеотраслевые руководящие материалы по созданию банков данных. – М.: ГКНТ, 1982.
12. Саймон А.Р. Стратегические технологии баз данных: менеджмент на 2000 год. – М.: Финансы и статистика, 1998.
13. Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования / Пер. с англ. – М.: Мир, 1999.
14. Хансен Г., Хансен Дж. Базы данных. Разработка и управление. – М.: Бином, 1999.
15. Четвериков В.Н., Ревунков Г.И., Самохвалов Э.Н. Базы и банки данных. – М.: Высшая школа, 1987.