Федеральное агентство по образованию

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизированных систем управления (АСУ)

В.Д. Сибилёв

ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

Томск 2007

Сибилёв В.Д. Проектирование баз данных: Учебное пособие /Томский государственный университет систем управления и радиоэлектроники – Томск, 2007. — 201 с.

Учебное пособие содержит материал учебных дисциплин «Базы данных» и «Проектирование баз данных», изучаемых студентами кафедры АСУ, получающими специальности 230105 и 080801. Изложены основные сведения о моделях данных концептуального и логического уровней и о методологии проектирования баз данных.

ОГЛАВЛЕНИЕ

1 ВВЕДЕНИЕ

1.1 Жизненный цикл системы баз данных

1.1.1 Структура жизненного цикла

Проектирование базы данных (БД) — это один из этапов общего процесса проектирования системы баз данных организации. Поэтому, прежде чем говорить о процедурах проектирования БД, рассмотрим жизненный цикл системы баз данных в целом.

Под *жизненным циклом (ЖЦ)* системы баз данных (СБД) понимается непрерывный процесс, который начинается в момент принятия решения о создании СБД и заканчивается в момент полного изъятия системы из эксплуатации. Этот процесс принято подразделять на ряд этапов (фаз). На каждом этапе выполняется определённый перечень работ, направленных на создание или эксплуатацию и развитие системы. Следующая схема представляет *идеальную* временну́ю последовательность этапов ЖЦ СБД (рис. 1.1).

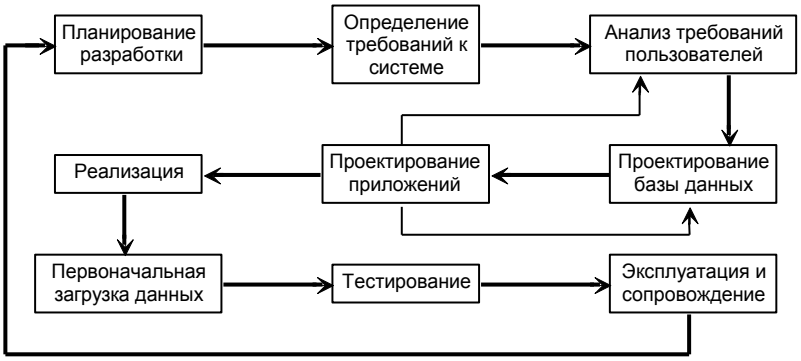


Рис. 1.1 — Жизненный цикл СБД

В идеале результаты очередного этапа должны являться исходными данными для следующего. Однако реальный жизненный цикл — это итерационный процесс. На каждом этапе могут возникнуть проблемы, которые потребуют пересмотра решений,

принятых на любом из предшествующих. Поэтому приведённую здесь схему следует рассматривать как *перечень видов работ*, а не как строгую временну́ю последовательность их исполнения.

Рассмотрим фазы ЖЦ подробнее.

1.1.2 Планирование разработки

Планирование разработки — это подготовительный этап. Здесь оцениваются

- требуемый объём работ;
- требуемые ресурсы;
- общая стоимость проекта.

Эти оценки приближённые. Они нужны руководству организации для принятия решения о создании (или радикальной переработке) системы. Обычно этот этап выполняется небольшой группой (2÷3 человека) высококвалифицированных и авторитетных специалистов. Возможно привлечение независимых экспертов.

1.1.3 Определение требований к системе

На этом этапе определяются диапазон действия системы (поддерживаемые бизнес-процессы), её границы и состав пользователей.

1.1.4 Анализ требований пользователей

Это наиболее ответственный этап ЖЦ. Недостаточно тщательно выполненный анализ приведёт к выходу за рамки бюджета проекта, превышению сроков разработки и, возможно, краху проекта. На этом этапе собирается и анализируется информация о деятельности организации, необходимая для проектирования БД и приложений. Выявляются

- функции пользователей системы,
- данные, необходимые для выполнения функций,
- бизнес-правила, ограничивающие функции.

Эта информация собирается различными способами. В частности

– путём опроса отдельных наиболее авторитетных сотрудников организации;

– посредством наблюдений за деятельностью организации;

– посредством изучения документов, в особенности тех, которые используются для сбора или представления информации;

– путём анкетирования широкого круга будущих пользователей системы.

Привлекается и опыт разработчиков в проектировании аналогичных систем.

Собранная информация по каждой области применения и каждой группе пользователей должна включать

– входную и выходную документацию организации;

– подробные сведения о выполняемых транзакциях;

– список требований пользователей с указанием приоритетов.

Получаемая аналитиками информация о деятельности организации всегда плохо структурирована. Задача аналитиков — структурировать её и определить *спецификации требований* пользователей, т.е. дать точное и однозначное описание функций пользователей и их потребностей в данных.

1.1.5 Проектирование БД

Проектирование БД — это этап создания информационного ядра системы. Основные цели этапа следующие.

- Определить данные и связи между ними, необходимые для всех основных областей применения создаваемой СБД и любых существующих групп пользователей.
- Создать модель данных, способную поддерживать выполнение любых требуемых транзакций обработки данных.
- Разработать предварительный вариант проекта СБД, структура которого позволит удовлетворить основные требования к производительности системы.

Вообще говоря, эти цели противоречивы. Задача проектировщиков — найти удачный компромисс между универсальностью и производительностью системы.

Процесс проектирования БД мы обсудим подробно в разделе 6.

1.1.6 Проектирование приложений

Проектирование приложений — это процесс создания интерфейса конечного пользователя и прикладных программ, предназначенных для обработки данных.

Реально проектирование БД и приложений выполняется параллельно. БД предназначена для поддержки приложений. Поэтому её проект должен учитывать нужды приложений, отражающие потребности локальных пользователей. Однако и приложения невозможно создавать, не зная структуры БД. Поэтому между фазами проектирования БД и приложений идёт постоянный обмен информацией, отражённый на рис. 1.1 дугами обратных связей. Проектирование приложений невозможно завершить до завершения проектирования БД.

Главная задача этой фазы — создание интерфейса конечного пользователя для всех функций системы, предусмотренных спецификациями требований пользователей.

Современные технологии разработки приложений реализуют концепцию прототипов (рис. 1.2).

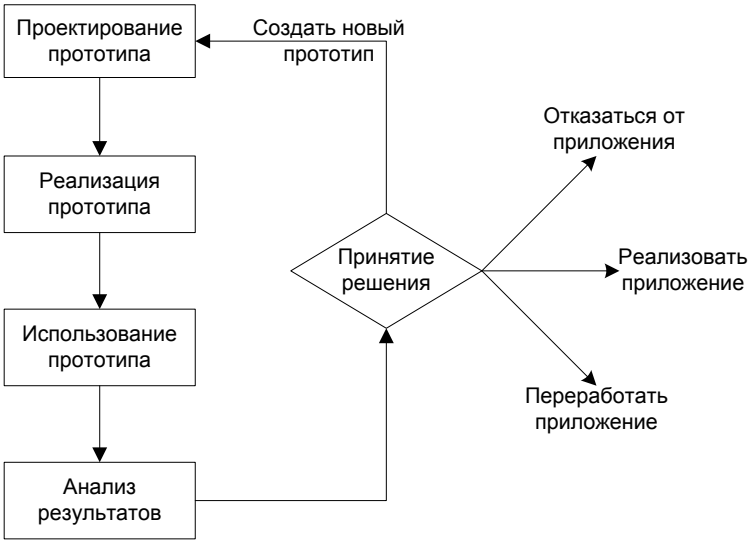


Рис. 1.2 — Процесс прототипирования

Прототип — это рабочая модель приложения, обладающая лишь частью требуемой функциональности. Он создаётся для того, чтобы пользователь мог опробовать его в работе и определить, какие из реализованных функций отвечают своему назначению, а какие — нет. С помощью прототипов разработчики уточняют требования конечных пользователей к системе. Прототипы обычно развиваются в готовые приложения.

1.1.7 Реализация

На этом этапе

– создаётся «пустая база данных»,

– определяются все специфические пользовательские представления,

– реализуются все приложения,

– реализуются все средства защиты данных и поддержания целостности.

Для реализации БД и внешних представлений создают их описание на языке определения данных (ЯОД) целевой системы управления базами данных (СУБД). Это описание используется системой для создания схем данных всех уровней, и пустых файлов физической базы данных (ФБД) на устройствах внешней памяти.

Приложения реализуются на языках программирования высокого уровня (ЯВУ). Транзакции обработки БД описываются на языке манипулирования данными (ЯМД) целевой СУБД. Они могут включаться непосредственно в текст программ, а могут быть реализованы в виде отдельного модуля. Многие современные СУБД имеют собственные среды быстрого создания приложений — языки четвёртого поколения (4GL).

Средства защиты данных и поддержания целостности частично описываются средствами ЯОД. Однако возможности СУБД в этом отношении не всегда достаточны. Поэтому часто приходится создавать специальные программы, выполняющие проверки правил целостности (триггеры БД), а также утилиты и специальные приложения, обеспечивающие безопасность данных.

1.1.8 Первоначальная загрузка

Первоначальная загрузка — это перенос *любых* существующих данных в новую БД и модификация всех существующих приложений с целью обеспечения их совместной работы с новой БД. Работы на этом этапе зависят от того, есть ли в организации компьютерная информационная система. Если новая СБД заменяет старую, то файлы старой ФБД должны быть конвертированы в новые форматы. Каждая современная СУБД имеет утилиту загрузки существующих файлов в новую БД. Этой утилите нужно задать спецификации файлов.

Адаптация старых приложений к новой БД — гораздо более сложная задача. Нередко проще разработать приложение заново, чем адаптировать его.

Если ИС организации создаётся впервые, то администратор базы данных (АБД) должен

– выделить подмножество данных организации, которые будут перенесены с бумажных носителей в БД;

– продумать процедуру ввода этих данных и создать вспомогательные приложения, предназначенные исключительно для обеспечения корректного ввода;

– обучить операторов и постоянно контролировать их работу.

1.1.9 Тестирование

Тестирование — это процесс выполнения прикладных программ с целью поиска ошибок. Тестирование *не может доказать безошибочность* программ. С помощью тестов можно лишь обнаружить часть имеющихся ошибок. Если в результате тестирования не выявлено ошибок, то либо тесты были неудачными, либо программа с большой вероятностью работает в соответствии со спецификацией. Существует множество стратегий и методик тестирования программ. Мы не будем обсуждать их здесь, поскольку это предмет технологии программирования, а не технологии БД.

1.1.10 Эксплуатация и сопровождение

Эксплуатация и сопровождение — это наблюдение за системой и поддержание её нормального функционирования по окончании развёртывания. На этом этапе осуществляется контроль производительности системы и сопровождение и модернизация приложений.

Производительность системы может падать со временем. Основные причины снижения — увеличение объёма хранимых данных и увеличение числа одновременно работающих приложений. Если среднее время реакции системы на запрос приложения становится недопустимо большим, то АБД должен произвести *дополнительную настройку* системы с целью увеличения производительности. Например, создать дополнительные индексы, объединить или расщепить отдельные таблицы, изменить структуры файлов и т.п. Для того чтобы выполнить настройку, АБД должен располагать сведениями о различных показателях функционирования системы, в частности о частотах обращений к тем или иным фрагментам БД, об эффективности системы блокировок, о выбираемых стратегиях выполнения запросов и т.п. Сбор этой информации обеспечивают специальные утилиты администрирования БД. Как правило, они входят в состав типичной СУБД. Однако если СУБД не предоставляет такие вспомогательные службы, то АБД должен создать их сам, поскольку мониторинг системы необходимо осуществлять в течение всего её жизненного цикла.

В процессе эксплуатации СБД у пользователей появляются новые потребности. Они должны отслеживаться и удовлетворяться. Новые требования могут относиться к функциональности существующих приложений, составу полей отдельных таблиц, составу таблиц БД в целом и т.п. Необходимость учёта этих требований возвращает разработчиков на начальный этап ЖЦ.

В конце концов, может наступить такой момент, когда нужно будет принять решение о проектировании новой СБД, поскольку существующая не в состоянии достичь требуемых эксплуатационных показателей.

Если новая система создана и введена в эксплуатацию, то старую не следует немедленно выводить из эксплуатации. Будет

лучше, если некоторое время они будут работать параллельно. Это обеспечит подстраховку на случай непредвиденных проблем с новой системой.

1.2 Обзор процесса проектирования БД

1.2.1 Цели проектирования

Проектирование БД — это процесс создания БД, предназначенной для поддержки функционирования организации и способствующей достижению её целей.

Этот процесс направлен на достижение двух целей.

Цель 1. Создать структуры хранения, способные обеспечить накопление данных организации и выполнение всех требуемых видов обработки данных.

Цель 2. Создать все приложения, способные обеспечить интерфейс конечных пользователей с базой данных и специфическую обработку данных в соответствии с требованиями различных конечных пользователей.

1.2.2 Фазы проектирования БД

Как вы помните, одна из основных концепций технологии БД — концепция трёхуровневого представления данных.

Описания данных на внешнем и концептуальном уровнях — внешние и концептуальная схемы (ВС и КС) — не привязаны к реализации базы данных. Они описывают логические структуры данных, соответствующие объектам ПО и их отношениям, и не содержат ссылок на программно-техническую платформу проектируемой системы. На этапе разработки ВС и КС платформа ещё не определена. В основе схем лежат *представления конечных пользователей о предметной области.*

Внешняя схема соответствует локальному представлению конечного пользователя (КП), исполняющего в организации конкретные функции, например бухгалтера группы материального учёта, инспектора отдела кадров, кладовщика и т.п.

Концептуальная схема обобщает локальные представления. Её можно понимать как описание представлений *абстрактного*

пользователя, исполняющего функции *всех* КП. Любая внешняя схема может быть выведена из концептуальной.

Внутренняя схема — это *реализация* концептуальной схемы на конкретной программно-технической платформе. Как и КС, она описывает представления обобщённого пользователя БД, но, кроме того, содержит определения физического уровня.

В соответствии со сказанным, выделяют три фазы проектирования БД (см. рис. 1.3):

- концептуальное моделирование,
- логическое моделирование,
- физическое проектирование.

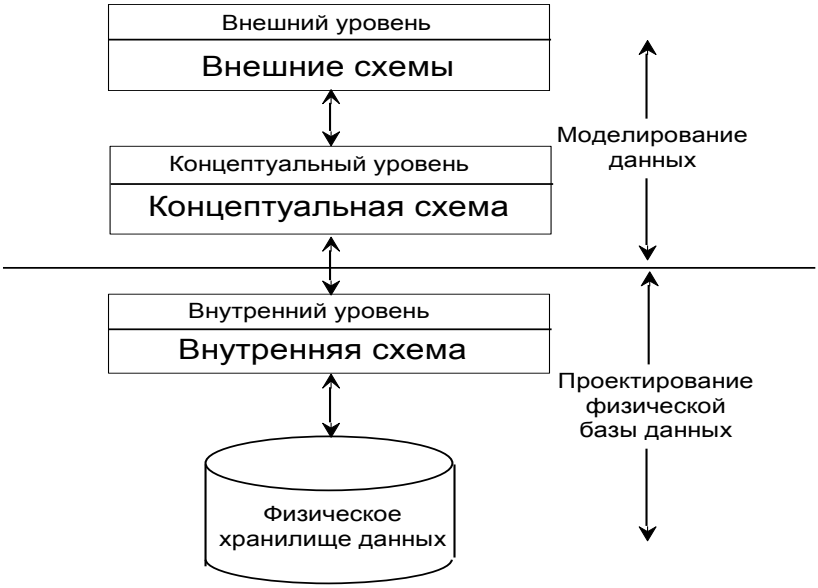


Рис. 1.3 — Фазы проектирования БД

Процесс создания внешних и концептуальной схем называют моделированием данных организации. Цель моделирования — понять, ЧТО нужно сделать.

Концептуальное моделирование не связано ни с какими соображениями реализации. Результатом этого процесса является

документированная модель представлений конечных пользователей об информационных потребностях бизнеса.

На фазе *логического моделирования* концептуальная модель преобразуется в *логическую* с учётом выбранного способа структурирования данных. Логическая модель и есть описание концептуальной схемы БД.

Процесс создания внутренней схемы называют проектированием ФБД или физическим проектированием БД. Его цель — решить, КАК сделать то, что нужно.

На фазе физического проектирования создаётся описание реализации БД на внешних запоминающих устройствах. Определяются структуры хранимых файлов и методы доступа, обеспечивающие эффективную обработку данных.

Обсудим фазы проектирования БД подробнее.

1.2.3 Концептуальное моделирование

Концептуальное моделирование — это процесс анализа информационных потребностей конечных пользователей системы. На этой фазе цель аналитика — сформировать в своей голове представления об информационных потребностях бизнеса, адекватные представлениям предполагаемых конечных пользователей системы. Эти представления формируются в процессе изучения деятельности организации. Используются следующие источники информации:

– документы организации (хозяйственные книги, счета, накладные, финансовые отчёты и т.п.),
– собственные наблюдения за работой организации,
– беседы с пользователями,
– опыт предыдущих разработок,
– собственные предположения об информационных потребностях бизнеса.

Обрабатывая полученную информацию, аналитик делает выводы о структуре и связях объектов предметной области. Эти выводы он документирует в модели данных и обсуждает с пользователем. Цель обсуждений — согласование представлений аналитика и пользователя о предметной области системы.

Основная проблема согласования в том, что пользователь и аналитик мыслят разными категориями. Пользователь знает, какие формы и отчёты и с какими данными ему нужны, но он ничего не может сказать о структурах и связях объектов, отражаемых в этих формах и отчётах. Разработчика же интересуют именно структуры и связи объектов. Он вынужден реконструировать объекты и связи из форм и отчётов.

Эта задача, как любая обратная задача, имеет множество решений. В ходе многократных обсуждений аналитик должен выбрать такое, которое адекватно модели данных, *имеющейся в голове пользователя*. В противном случае результат разработки — готовая система — вряд ли удовлетворит пользователя.

Процесс моделирования больше искусство, чем наука. Существует множество средств и способов моделирования данных. Можно изучить их все. Мы изучим некоторые. Но их использование — это искусство, требующее опыта и интуиции.

Результат концептуального моделирования — концептуальная модель данных польлзователя — однозначно описывает структуры и связи объектов предметной области и бизнес-правила организации. Это описание *не зависит* от

- типа целевой СУБД,
- набора создаваемых программ,
- используемых языков программирования,
- типа вычислительной платформы,
- а также от любых других особенностей физической реализации.

Концептуальная модель служит источником информации для фазы логического моделирования.

1.2.4 Логическое моделирование

В процессе логического моделирования концептуальная модель уточняется и преобразуется в логическую с учётом базовой модели данных целевой СУБД. Например, если целевая СУБД базируется на РМД, то структуры и связи объектов, представленные в концептуальной модели, преобразуются в систему взаимосвязанных отношений, удовлетворяющих определённым формальным требованиям.

К началу логического моделирования должен быть определён *тип* целевой СУБД (реляционная, сетевая, объектно-ориентированная…). Однако все прочие аспекты целевой СУБД (физическая организация структур хранения, способы построения индексов и т.п.) *полностью игнорируются*.

Логическая модель содержит описания структур данных в терминах выбранной модели данных, а также описания ограничений целостности. Она должна

– быть корректной,

– соответствовать требованиям пользователей, зафиксированным в концептуальной модели,

– обеспечивать поддержку всех необходимых пользователям транзакций.

Логическая модель является источником информации для фазы физического проектирования БД и для проектирования приложений.

Кроме того, она играет важную роль на этапе эксплуатации и сопровождения СБД. При правильно организованном сопровождении она составляет основную часть системного каталога и автоматически обновляется при внесении изменений в логические структуры данных. Благодаря этому АБД всегда может точно представить любые вносимые изменения и оценить их влияние на существующие приложения.

Моделирование данных — наиболее ответственный и трудоёмкий этап разработки системы. Это практически бесконечный итеративный процесс исследования информационных потребностей предприятия. Каждая итерация вносит уточнения и улучшения в модели, что, в свою очередь, может потребовать изменений в других частях проекта.

Если созданные модели неадекватно отражают представления пользователей о предметной области, то будет очень трудно или даже невозможно определить все необходимые внешние схемы, организовать поддержку целостности данных и обработку необходимых транзакций. Опыт показывает, что успешные проекты используют около 2/3 бюджета времени на моделирование данных. Именно качественные модели являются залогом их успеха.

1.2.5 Физическое проектирование

На этой фазе принимаются решения о способах реализации БД. Она может начаться только после выбора конкретной целевой СУБД. Целью физического проектирования является описание способа физической реализации логической модели.

Например, если используется реляционная СУБД, то в её среде должен быть создан набор таблиц и ограничений, представленный в логической модели. Должны быть определены конкретные структуры хранения данных (файлы) и методы доступа к данным, обеспечивающие требуемую производительность системы. Кроме того, должны быть разработаны средства защиты системы.

В идеале фазы логического моделирования и физического проектирования следует разделять, однако реально это невозможно, поскольку решения, принимаемые на фазе физического проектирования с целью повышения производительности системы, могут повлиять на структуру логической модели.

Фазы моделирования и физического проектирования БД предъявляют различающиеся требования к знаниям и профессиональным навыкам проектировщиков. Для успешного моделирования нужно обладать навыками исследовательской работы и владеть методологиями моделирования. Для успешного проектирования ФБД нужно хорошо знать возможности целевой программно-технической платформы, владеть её инструментами и иметь навыки системного программирования.

В дальнейшем мы сосредоточимся на методологиях концептуального и логического моделирования и примерах их использования.

1.3 Языковые средства моделирования

Любой проект чего бы то ни было — это *точное и однозначное описание* того, что должно быть воплощено в некотором изделии, удовлетворяющем заданным требованиям. В частности, проект базы данных есть описание сложной структуры, предназначенной для сохранения данных пользователя. Как мы видели, это описание выполняется на трёх уровнях — концепту-

альном, логическом и физическом. Ни один из них не может быть опущен.

Для реализации базы данных в определённой программно-технической среде необходимо описание структур хранимых данных в терминах этой среды — физическая схема (модель) данных.

Для того чтобы создать физическую схему и обеспечить возможность обработки хранимых данных приложениями, необходимо иметь описание логических структур данных, соответствующих объектам ПО и их отношениям — логическую модель данных.

Логическая модель представляет требования пользователя к данным в терминах логических структур определённого типа. Для того чтобы её создать, необходимо иметь точное описание требований пользователя — концептуальную модель данных.

Наконец, все три уровня описания данных пользователя необходимы для разработки эффективных приложений и для совершенствования СБД в процессе эксплуатации.

Говоря здесь о моделях данных, мы имеем в виду описания конкретных данных. Однако термин «модель данных» используется в литературе, по крайней мере, в двух смыслах.

1. Модель данных — это описание требований к структурам и связям данных, которые должны храниться в базе данных и обрабатываться приложениями.

В этом смысле мы употребляли этот термин выше. Точнее было бы говорить «модель данных пользователя», «модель требований к данным», или «модель данных предприятия».

Средствами естественного языка практически невозможно точно и однозначно описать требования к данным даже на концептуальном уровне. Поэтому для создания моделей требований к данным используют специальные языковые средства, которые также называют моделями данных. Это второй смысл термина.

2. Модель данных — это набор языковых и изобразительных стандартов, используемых для описания требований к данным.

В двух следующих разделах мы будем употреблять термин «модель данных» именно в этом смысле.

На разных фазах моделирования данных используются различные модели данных соответственно концептуального, логического и физического уровней. В настоящее время концептуальное моделирование чаще всего выполняется средствами модели «сущность-связь» (ER-модели данных), а логическое — средствами реляционной модели данных (РМД). Существуют и другие модели, но они не столь популярны по разным причинам. Например, такие модели логического уровня, как иерархическая и сетевая, устарели. Семантическая объектная модель (концептуальный уровень) пока не вошла в обиход проектировщиков, хотя имеет определённые преимущества перед ER-моделью. Универсальный язык моделирования UML, основанный на концепциях ООП, можно использовать для концептуального моделирования, но при проектировании РБД он не имеет никаких преимуществ перед ER-моделью.

Здесь мы изучим наиболее популярные модели концептуального и логического уровней — ER-модель данных и реляционную модель данных (структурную часть).

ER-модель данных — это графический язык, предназначенный для описания объектов и отношений.

Структурная часть РМД — это язык определения логических структур данных табличного типа. Существуют текстовые и графические версии реляционных ЯОД.

Владение понятийным аппаратом и знание стандартных нотаций ER-модели и РМД необходимы для освоения методологии концептуального и логического моделирования данных, излагаемой в разделах 7 и 8 настоящего пособия.

Контрольные вопросы

1. Опишите понятия «предметная область», «модель предметной области», «база данных».
2. Опишите все известные Вам значения термина «модель данных».
3. Что называется концептуальной моделью данных пользователя?
4. Что называется логической моделью данных?
5. Что называется физической моделью данных?

6. Опишите соотношения концептуальной, логической и физической моделей.

7. Перечислите основные этапы и опишите структуру жизненного цикла.

8. Какие работы выполняются на этапе планирования разработки?

9. Какие работы выполняются на этапе определения системных требований?

10. Какие работы выполняются на этапе анализа требований пользователей?

11. Какие работы выполняются на этапе проектирования БД?

12. Какие работы выполняются на этапе проектирования приложений?

13. Какие работы выполняются на этапе реализации?

14. Какие работы выполняются на этапе первоначальной загрузки?

15. Какие работы выполняются на этапе тестирования?

16. Какие работы выполняются на этапе эксплуатации и сопровождения?

17. Какова цель проектирования базы данных?

18. Перечислите основные этапы процесса проектирования базы данных.

19. Какие работы выполняются на этапе концептуального моделирования?

20. Какие работы выполняются на этапе логического моделирования?

21. Какие работы выполняются на этапе физического проектирования?

22. Сформулируйте цель проектирования логического макета базы данных и критерий её достижения.

2 МОДЕЛЬ «СУЩНОСТЬ-СВЯЗЬ»

2.1 Общее представление

Модель «сущность-связь» или ER-модель (entity-relationship model) опубликована американским исследователем в области баз данных Питером Ченом в 1976 году. С тех пор она расширялась и модифицировалась как самим Ченом, так и многими другими исследователями. В различных вариантах она вошла в состав многих CASE-средств[1] поддержки проектирования информационных систем.

По существу ER-модель данных — это графический язык для описания объектов и отношений объектов. Спецификации требований пользователя представляются в виде диаграммы, показывающей объекты ПО, их связи и свойства объектов и связей — ER-диаграммы. Существует много различных систем графических обозначений (нотаций), используемых для построения ER-диаграмм. Нет необходимости изучать их все. Усвоив основные концепции ER-модели и принципы построения диаграмм в одной системе обозначений, нетрудно разобраться в любой другой.

Какие бы нотации ни использовались, ER-диаграмма наглядно и точно отражает *представления автора о требованиях пользователя к данным*. Поэтому она является хорошим источником информации для проектировщика логической модели данных. С другой стороны, диаграммы выразительны, наглядны и легко интерпретируются конечными пользователями. Поэтому их очень удобно использовать при обсуждении требований к данным с конечными пользователями.

2.2 Понятийная основа

Базовыми понятиями ER-модели являются *сущность*, *атрибут*, *идентификатор* и *связь*.

[1] Computer Aided Software Engineering

2.2.1 Сущность

Сущность (entity) — это некоторый объект, выделяемый (*идентифицируемый*) пользователем в предметной области. Нечто, за чем пользователь хотел бы наблюдать и сохранять результаты наблюдений (данные). Например,

СТУДЕНТ Петров,
ПРЕПОДАВАТЕЛЬ Ломов,
УЧЕБНИК по БД,
АУДИТОРИЯ $238_{РК}$,
УЧЕБНОЕ ЗАНЯТИЕ лекция по ПБД для групп $441_{1,2}$ и т.п.

Из примеров видно, что сущностями могут быть люди, предметы, места, события и т.д. Обобщая, можно сказать, что сущность — это нечто, имеющее реальное (физическое) или концептуальное существование и выделяемое пользователем в окружающем мире.

Сущности одного и того же типа образуют *классы сущностей*. СТУДЕНТ — это класс сущностей, имеющих *одинаковые наборы характеристик*, значения которых представляют интерес для пользователя. Пользователь заинтересован в сведениях об *экземплярах* класса. Например, о конкретных студентах, обучающихся в настоящее время на кафедре АСУ.

В литературе часто используют термин «*сущность*» как в смысле «*класс сущностей*», так и в смысле «*экземпляр класса сущностей*». Мы будем поступать так же, когда это не будет вызывать недоразумений.

Итак, класс сущностей — это абстракция, *понятие,* выделяемое пользователем. В сознании пользователя понятию сопоставляется *символ* — имя сущности. В естественном языке это всегда имя существительное в единственном числе. Договоримся в дальнейшем записывать имена сущностей прописными буквами.

Имя сущности имеет для пользователя *единственный* вполне конкретный смысл, однако неискушённый человек не всегда может передать его с помощью других слов. Более того, различные пользователи могут наделять один и тот же символ различными смыслами.

Например, представления о СТУДЕНТе, имеющиеся у зам. декана и преподавателя, различаются.

Для зам. декана это лицо, зачисленное приказом ректора в определённую группу. Одна из обязанностей зам. декана — контролировать успеваемость этого лица на всех этапах процесса обучения. Этим определяется набор сведений о СТУДЕНТе, которыми хотел бы располагать зам. декана.

Для преподавателя СТУДЕНТ — это лицо, имеющее право посещать его занятия и обязанное в определённые сроки отчитываться о результатах изучения тех дисциплин, которые ведёт преподаватель.

2.2.2 Связь

Связь — это отношение сущностей. ER-модель различает классы и экземпляры связей.

Под *экземпляром связи* понимают ассоциацию экземпляров одного или более классов сущностей.

Примеры

ПРЕПОДАВАТЕЛЬ Грушин *преподаёт УЧЕБНую ДИСЦИПЛИНу* «Теория автостопа»	Ассоциация экземпляров двух классов сущностей ПРЕПОДАВАТЕЛЬ и ДИСЦИПЛИНА
ЧЕЛОВЕК Иван *женат на ЧЕЛОВЕКе* Марье	Ассоциация экземпляров *одного* класса сущностей *ЧЕЛОВЕК*
ТРЕНЕР Колов *проводит тренировку по СПОРТИВНой ДИСЦИПЛИНе* «волейбол» с *КОМАНДой* «Попрыгунчики»	Ассоциация экземпляров трёх классов сущностей: ТРЕНЕР, ДИСЦИПЛИНА, КОМАНДА

В естественном языке связь представляется простым предложением, передающим смысл ассоциации. Имена экземпляров классов сущностей в таких предложениях можно понимать как значения переменных, обозначенных именами классов сущностей.

Если убрать из предложений имена экземпляров, оставив имена классов, то получим обобщающие предложения:

ПРЕПОДАВАТЕЛЬ преподаёт УЧЕБНую ДИСЦИПЛИНу.

ЧЕЛОВЕК женат на ЧЕЛОВЕКе.

ТРЕНЕР проводит тренировку по СПОРТИВНой ДИСЦИПЛИНе с КОМАНДой.

Это речевые шаблоны. Подставляя в такой шаблон допустимые значения переменных, выделенных прописными буквами, можно получить любое конкретное утверждение определённого типа. Например,

Грушин преподаёт «Теорию автостопа»;

Копков преподаёт «Проектирование баз данных» и т.п.

Нетрудно догадаться, что выражают шаблоны. Они выражают осмысленные (с точки зрения пользователя) ассоциации классов сущностей. Эти ассоциации и называются классами связей.

> ***Класс связей*** — это осмысленная с точки зрения пользователя ассоциация классов сущностей. *Смысл* ассоциации передаётся глагольным оборотом, связывающим имена классов сущностей.

Экземпляр класса связей — это конкретное утверждение, подобное двум последним примерам.

В дальнейшем мы будем говорить о связях сущностей, имея в виду *классы* связей. Экземпляры (классов) связей нас будут интересовать только в примерах.

2.2.3 Атрибут

Атрибут — это характеристика сущности или связи (свойство класса), *значимая* с точки зрения пользователя. Например: Номер Студбилета, Фамилия преподавателя, Название учебника, Номер аудитории — свойства сущностей СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, УЧЕБНИК, АУДИТОРИЯ.

Атрибут имеет имя и принимает значения. Именем атрибута всегда является имя существительное, возможно, с дополнениями.

Связи, как и сущности, могут иметь атрибуты. Например, пусть имеются сущности ТОВАР и ПОСТАВЩИК и отношение ТОВАР поставлен ПОСТАВЩИКом. Факт поставки товара подтверждается документом. Документ имеет реквизиты: регистра-

ционный номер, дату поставки, количество поставленных единиц товара. Пользователь, обрабатывающий поставки, заинтересован в сохранении значений этих реквизитов. В его сознании они связаны с фактом получения товара, т.е. являются значениями атрибутов упомянутого отношения.

2.2.4 Типы атрибутов

Атрибут может быть
- простым или составным (композитным);
- однозначным или многозначным;
- первичным или производным.

Атрибут является *простым*, если его значения в представлении пользователя не имеют внутренней структуры. В противном случае он композитный.

Один и тот же атрибут (имя) в представлениях различных пользователей может быть и простым, и композитным.

Например, в ТУСУРе номер аудитории состоит из идентификатора учебного корпуса, номера этажа, на котором расположена аудитория, и номера аудитории в пределах этажа. Заместителю декана не приходится обрабатывать элементы этой структуры. В его представлении `Номер аудитории` — простой атрибут. Для работника бюро расписаний ТУСУРа, которому приходится выяснять номера компьютерных классов или аудиторий общего назначения в конкретных учебных корпусах, этот атрибут составной:

`Номер аудитории` = {`Корпус, Этаж, Номер на этаже`}

Атрибут является *многозначным*, если он может принимать более одного значения *для одного экземпляра сущности*. В противном случае он однозначный.

Один и тот же атрибут в представлениях различных пользователей может быть и однозначным, и многозначным. Например, сущность ПРЕПОДАВАТЕЛЬ имеет атрибут `Преподаваемая дисциплина`. Преподаватель может преподавать несколько дисциплин. В общем случае значением этого атрибута будет список дисциплин. Если пользователю не нужно различать элементы списка, то для него этот атрибут простой. Если

же пользователю приходится составлять списки преподавателей, преподающих конкретную дисциплину, то он воспринимает атрибут как многозначный.

> Атрибут является *производным*, если его значения могут быть определены по значениям других атрибутов. В противном случае он *первичный*.

И в этом случае атрибут может одним пользователем рассматриваться как первичный, а другим — как производный.

Примеры

1. В состав атрибутов сущности ГРУППА может входить атрибут *Численность Группы*. Его значение для каждого экземпляра ГРУППы может быть определено путём подсчёта числа экземпляров сущности СТУДЕНТ, связанных с этим экземпляром. Это производный атрибут.

2. Рассмотрим сущность ЗАКАЗ, которая связывается в голове пользователя с заявкой клиента на приобретение одного или нескольких наименований товаров. По мнению пользователя, эта сущность характеризуется атрибутами:

Заказчик = {Имя, Адрес, Телефон} — составной атрибут;

Заказанный Товар = {Наименование, Цена, Количество, Сумма} — многозначный составной атрибут;

Стоимость Заказа — производный атрибут, значение которого равно сумме значений в колонке Сумма значения многозначного атрибута Заказанный Товар, соответствующего одному экземпляру заказа. Заметим, что и значения в этой колонке также производные: Сумма = Цена * Количество.

2.2.5 Идентификаторы

Идентификаторы — это атрибуты сущностей, значения которых можно использовать для идентификации или именования экземпляров. Выделяют *уникальные* идентификаторы (потенциальные ключи) и *неуникальные*. Одно и то же значение уникального идентификатора не может встретиться у двух экземпляров сущности. Оно указывает на один и только один экземпляр.

Например, Номер студбилета — уникальный идентификатор экземпляров сущности СТУДЕНТ (в пределах ВУЗа), Номер аудитории — уникальный идентификатор экземпляров сущности АУДИТОРИЯ.

Значение *неуникального* идентификатора указывает на *множество* экземпляров (Фамилия преподавателя = Иванов указывает на всех Ивановых, преподающих в ВУЗе).

Идентификатором может быть не любой атрибут сущности. Например, Дата найма или Должность преподавателя вряд ли могут использоваться для идентификации преподавателей.

Сущность может иметь *несколько* уникальных и неуникальных идентификаторов.

Идентификатор, как уникальный, так и неуникальный, может быть простым (состоящим из одного атрибута) или составным (композитным, состоящим из нескольких атрибутов).

Заметим, что уникальность идентификатора — это свойство, определяемое природой и смыслом характеристики объекта. Атрибут нельзя *назначить* уникальным идентификатором сущности. Он либо *является* таковым, либо *не* является.

Неуникальным идентификатором можно назначить *любой* атрибут, если пользователь заинтересован в выделении множеств экземпляров сущности с одинаковыми значениями этого атрибута. Например, (СТУДЕНТ.НомерГруппы).

2.3 Обозначения для сущностей и связей

Описание представленных в голове пользователя сущностей, связей и атрибутов — это и есть концептуальная модель требований пользователя к данным. Однако средства естественного языка мало пригодны для создания такого описания, прежде всего, из-за громоздкости и малой наглядности. В текстовом описании трудно выделить все связи, в которые вступает одна и та же сущность, трудно отследить цепочки связей, которые задействованы в транзакциях, и т.п. Нужны специальные языковые средства для представления модели.

Идея П. Чена, благодаря которой его имя стало широко известным в кругах проектировщиков баз данных, состоит в том, что *сущности и связи следует представлять графически*. Тогда модель требований пользователя будет компактной и наглядной.

Договоримся обозначать класс сущностей *прямоугольником*. Внутри прямоугольника будем писать имя сущности ПРОПИСНЫМИ БУКВАМИ. Класс связей будем изображать *ромбом*. Участвующие в связи сущности будем соединять с ромбом прямыми *дугами*. Иногда глагольный оборот, передающий смысл ассоциации, будем записывать около дуги связи. На рис. 2.1 приведены примеры ER-диаграмм (уровня классов):

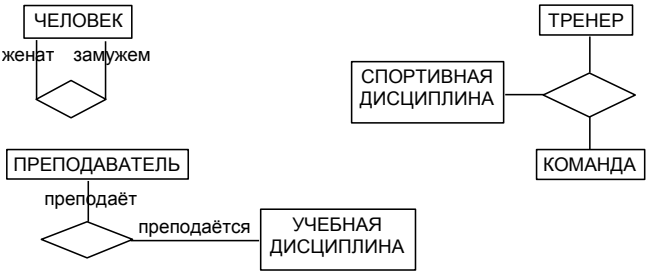


Рис. 2.1 — Примеры ER-диаграмм уровня классов

Обычно на ER-диаграммах семантически значимые имена связей не указывают, а поясняют их смысл иначе, как показано на рис. 2.2.

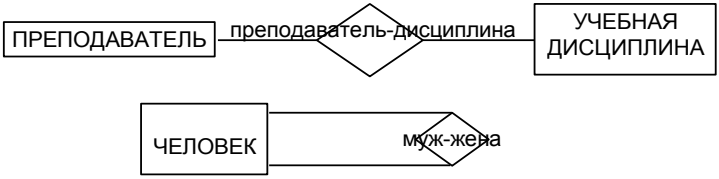


Рис. 2.2 — Типичные обозначения связей

На диаграммах используются также специальные обозначения для атрибутов, спецификаторы связей, сущностей, идентификаторов и другие символы. Мы будем вводить их по мере необходимости.

Подчеркнём, что приведённые диаграммы вовсе не утверждают, что *каждый* преподаватель ассоциируется с *каждой*

учебной дисциплиной или что *каждый* человек ассоциируется с каждым как муж и жена. Они утверждают, что существует преподаватель, ассоциированный с одной или несколькими дисциплинами, и, соответственно, дисциплина, ассоциированная с одним или несколькими преподавателями. На уровне экземпляров диаграммы выглядят так, как на рис. 2.3. Здесь точками обозначены экземпляры сущностей, а дугами — экземпляры связей.

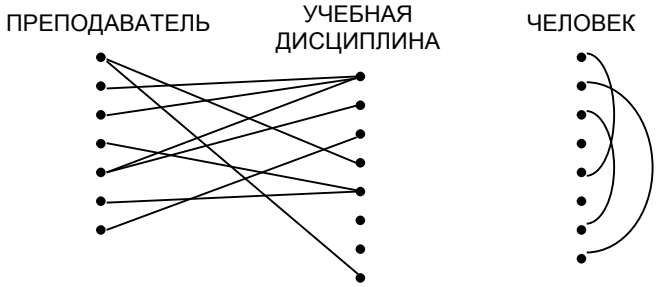


Рис. 2.3 — Диаграммы уровня экземпляров

2.4 Свойства связей

2.4.1 Степень связи

Из приведённых примеров видно, что связь может быть ассоциацией экземпляров одной сущности (класса), двух, трёх и т.д.

Число классов сущностей, участвующих в связи, называется *степенью* (*арностью*) связи.

Говорят об унарных связях (`Муж-Жена`), бинарных (`Преподаватель-Учебная Дисциплина`), тернарных (`Тренер-Команда-Спортивная Дисциплина`), тетрарных и т.д.

2.4.2 Мощность связи

Под мощностью понимается число экземпляров связи, в которые может участвовать один экземпляр сущности.

Связь степени `n` характеризуется `n` показателями мощности — по одному со стороны каждой сущности.

Можно уточнить понятие мощности так.

Пусть E1, E2,..., En — сущности, образующие n-арную связь R. Мощностью связи R со стороны сущности Ei называется число экземпляров связи, в которых *может* участвовать *один* экземпляром сущности Ei.

Значение мощности связи определяется бизнес-правилами.

Примеры

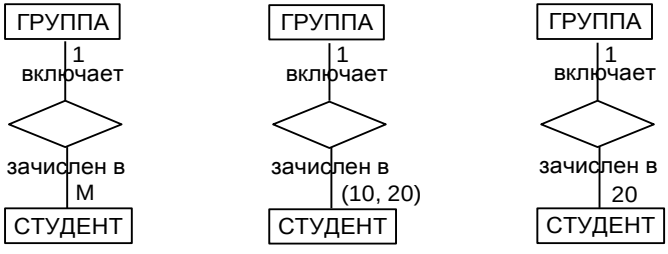
Согласно правилам Минобразования РФ *каждый* студент может быть зачислен *ровно в одну* учебную группу. Поэтому мощность связи СТУДЕНТ — ГРУППА со стороны сущности СТУДЕНТ равна единице.

Если в ВУЗе нет правила, ограничивающего численность группы, то указать точное значение мощности связи со стороны сущности ГРУППА невозможно. Можно лишь сказать, что в одну группу может быть зачислено много (более одного) студентов.

Если же по правилам ВУЗа число студентов в группе не может быть меньше 10 и больше 20, то мощность связи со стороны ГРУППЫ лежит в интервале 10 ÷ 20.

Если по правилам ВУЗа в каждой группе обучается ровно 20 студентов, то мощность связи со стороны ГРУППЫ равна 20.

Спецификации мощности связей отображаются на диаграммах так, как показано на рис. 2.4.



а *б* *в*

Рис. 2.4 — Спецификации мощности связи:
а — не ограниченная численность ГРУППЫ;
б — в ГРУППЕ от десяти до двадцати СТУДЕНТОВ;
в — в каждой ГРУППЕ ровно двадцать СТУДЕНТОВ

В реальной жизни правила, подобные двум последним, — большая редкость. Поэтому обычно на ER-диаграммах встречаются спецификации мощности, подобные приведённым на рис. 2.4, *а*.

2.4.3 Степень участия сущности в связи

Мощность характеризует максимальное число экземпляров связи с участием одного экземпляра сущности. Однако бизнес-правила могут ограничивать и минимальное. Этому ограничению в ER-модели соответствует понятие степени участия сущности в связи.

Степень участия — это характеристика обязательности/необязательности участия экземпляра сущности в связи.

Пусть E1, E2,..., En — сущности, образующие n-арную связь R. Связь R является *обязательной* со стороны сущности Ei, если *каждый* экземпляр Ei *должен* участвовать хотя бы в одном экземпляре связи. В противном случае связь является *необязательной* со стороны Ei.

Примеры

1) Согласно правилам Минобразования РФ каждый студент должен быть зачислен в какую-то группу. Вольных студентов не бывает. Поэтому связь СТУДЕНТ–ГРУППА *обязательная со стороны сущности* СТУДЕНТ. Аналогично, не бывает ГРУПП, в которые не зачислено ни одного СТУДЕНТа. Следовательно, рассматриваемая связь обязательная и со стороны сущности ГРУППА.

2) Каждый преподаватель преподаёт одну или более учебных дисциплин. Иначе, зачем его держат в ВУЗе? Однако в учебном плане специальности могут быть дисциплины, за которыми не закреплены преподаватели. Например, доцент Кошечкин, преподававший «Общую теорию заквашивания», ушёл на пенсию. Другого специалиста по заквашиванию на кафедре нет. Дисциплина «повиснет» до тех пор, пока заведующий кафедрой не найдёт такового. Следовательно, связь ПРЕПОДАВАТЕЛЬ–УЧЕБНАЯ ДИСЦИПЛИНА является обязательной со стороны сущности ПРЕПОДАВАТЕЛЬ и необязательной со стороны сущности УЧЕБНАЯ ДИСЦИПЛИНА.

3) Между сущностями СТУДЕНТ и УЧЕБНАЯ ДИСЦИПЛИНА существует связь «отчитался по», т.е. сдал экзамен или зачёт. Студент-первокурсник Колов не будет участвовать ни в одном экземпляре этой связи до первой в своей жизни зачётной недели. Дисциплина «Проектирование баз данных» — новая в учебном плане ВУЗа. Она также не будет связана ни с одним студентом до тех пор, пока первый студент не защитит курсовой проект по этой дисциплине. Связь СТУДЕНТ-УЧЕБНАЯ ДИСЦИПЛИНА необязательная со стороны обеих сущностей.

На рис. 2.5 приведены примеры спецификаций степени участия сущностей в связях.

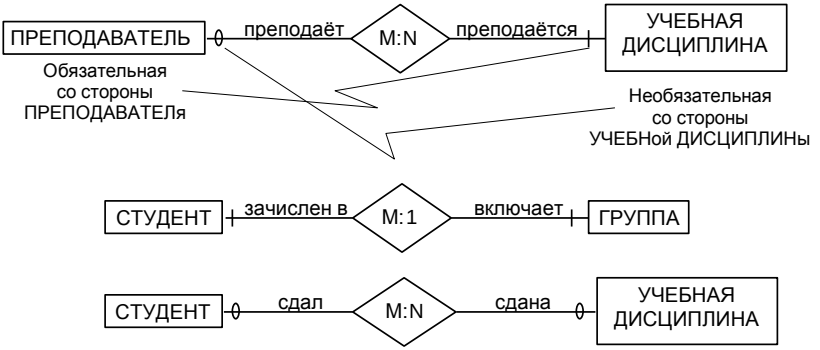


Рис. 2.5 — Спецификации степени участия

2.4.4 Типы бинарных связей

Наиболее распространёнными являются бинарные связи (связи степени 2). Они играют очень важную роль в концептуальном моделировании. Принято выделять три типа бинарных связей. Их обозначают 1:1, 1:N и M:N.

1:1 — один экземпляр E1 может участвовать не более чем в одном экземпляре связи и один экземпляр E2 может участвовать не более чем в одном экземпляре связи.

1:N — экземпляр E1 может участвовать в нескольких (N) экземплярах связи, а экземпляр E2 — не более чем в одном.

M:N — экземпляр E1 может участвовать в N экземплярах связи, а экземпляр E2 — в M экземплярах.

На уровне экземпляров эти типы связей изображены ниже (рис. 2.6).

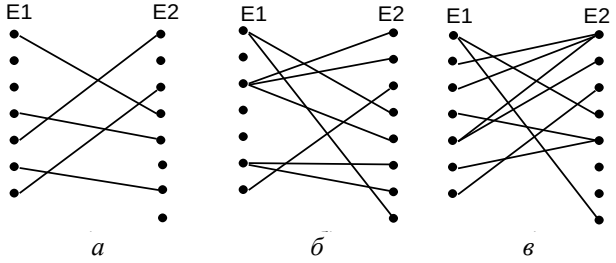


Рис. 2.6 — Типы бинарных связей:
а — связь типа 1:1; б — связь типа 1:M; б — связь типа M:N

2.5 Дополнительные элементы модели

2.5.1 Слабые сущности

Слабой сущностью в ER-модели называется сущность, экземпляры которой *не могут существовать вне связи* с экземплярами каких-то других сущностей.

Сущность, не являющаяся слабой, называется *сильной*.

Примеры.

1. Рассмотрим связь сущностей ФИЛЬМ и КОПИЯ. Здесь ФИЛЬМ — произведение киноискусства, КОПИЯ — экземпляр записи ФИЛЬМа на каком-либо носителе изображения: киноленте, магнитной ленте, компакт-диске... Сущность КОПИЯ слабая. Ни физически, ни логически никакой её экземпляр не может существовать вне связи с конкретным экземпляром сущности ФИЛЬМ.

2. Рассмотрим связь ДОМ–КВАРТИРА. ДОМ — это строение, предназначенное для проживания людей. КВАРТИРА — отдельное помещение в ДОМе, имеющее отдельный вход и некоторый набор бытовых служб. Очевидно, никакой экземпляр сущности КВАРТИРА не может существовать вне связи с каким-либо экземпляром сущности ДОМ.

3. Работники любой организации связаны отношением начальствования–подчинения. Рассмотрим связь СЛУЖАЩИЙ–

ПОДЧИНЁННЫЙ. СЛУЖАЩИЙ — физическое лицо, работающее в нашей организации по договору найма. ПОДЧИНЁННЫЙ — СЛУЖАЩИЙ, обязанный выполнять распоряжения другого СЛУЖАЩЕГО. Каждый подчинённый является служащим. Поэтому *физически* он существует вне каких бы то ни было связей. Однако логически служащий не является подчинённым, если не связан с каким-то служащим-начальником. Сущность ПОДЧИНЁННЫЙ слабая.

Таким образом, слабая сущность — это сущность, *логически зависимая* от какой-либо другой сущности (или сущностей).

Слабые сущности изображаются на ER-диаграммах прямоугольниками со скруглёнными углами (см. рис. 2.7). Связь, от которой зависит существование слабой сущности, изображается ромбом со скруглёнными углами. В некоторых системах обозначений слабые сущности и соответствующие ромбы связей рисуются двойными линиями.

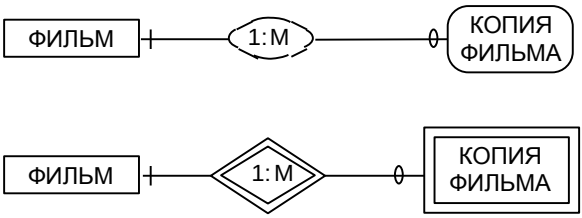


Рис. 2.7 — Варианты изображения слабых сущностей

Степень участия слабой сущности в связи *всегда обязательная*. Это следует из её определения. Однако некоторые авторы почему-то считают верным и обратное утверждение: если степень участия сущности X в связи обязательная, то X — слабая сущность. Это опровергается следующим примером.

По правилам ВУЗа *каждый* студент должен быть зачислен в какую-то группу. Минимальная кардинальность связи СТУДЕНТ-ГРУППА со стороны сущности СТУДЕНТ единица. Однако студент не зависит от группы ни физически, ни логически. Поэтому считать сущность СТУДЕНТ слабой не следует, не-

смотря на то, что деловой регламент требует её обязательного участия в связи с ГРУППой.

Слабость сущности определяется природой соответствующего объекта, а не бизнес-правилами.

Модель выделяет особую разновидность слабой сущности — *идентификационно зависимую* сущность. Это такая сущность, экземпляры которой не могут быть идентифицированы вне связи с другой сущностью. Согласно определению, идентификационно зависимой является слабая сущность, идентификатор которой содержит идентификатор другой сущности. Например, квартиру невозможно идентифицировать, не указав дом, в котором она расположена. Идентификатор квартиры композитный: {НомерДома, НомерКвартиры}. Идентификационно зависимые сущности на ER-диаграммах не выделяются.

2.5.2 Изображение атрибутов на ER-диаграммах

Некоторые системы нотаций ER-модели предусматривают обозначения для атрибутов. Атрибут изображается именованным эллипсом. Эллипс соединяется дугой с прямоугольником сущности. Контур эллипса

сплошной для простого атрибута,

штриховой — для производного и

двойной — для многозначного.

Компоненты составного атрибута обозначаются эллипсами, соединёнными дугами с эллипсом атрибута. Имена атрибутов, составляющих идентификатор сущности, подчёркиваются. Ниже на рис. 2.8 приведён пример описания сущности ЗАКАЗ на уровне атрибутов.

Рис. 2.8 — Изображение атрибутов сущности

Связь, как и сущность, может иметь свои атрибуты. Они изображаются эллипсами, соединёнными с ромбом связи.

Такое описание атрибутов чересчур громоздко. Если на диаграмме представлено несколько сущностей вместе с их атрибутами и связями, то разобраться в путанице линий весьма сложно. Гораздо удобнее представлять списки атрибутов сущностей отдельно, как показано на рис. 2.9.

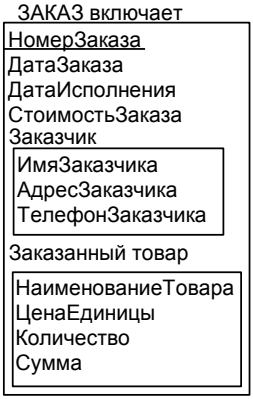


Рис. 2.9 — Список атрибутов сущности ЗАКАЗ

2.5.3 Подтипы и супертипы сущностей

Представление пользователя может содержать сущности, имеющие сходные наборы атрибутов и/или участвующие в аналогичных связях. Например, в представлении риэлтера (см. рис. 2.10) есть сущности НЕДВИЖИМОСТЬ ДЛЯ АРЕНДЫ и НЕДВИЖИМОСТЬ ДЛЯ ПРОДАЖИ, которые различаются только атрибутами Цена и Арендная ставка и участвуют в аналогичных связях.

Сущности НЕДВИЖИМОСТЬ ДЛЯ АРЕНДЫ и НЕДВИЖИМОСТЬ ДЛЯ ПРОДАЖИ можно рассматривать как подтипы некоторого супертипа — сущности НЕДВИЖИМОСТЬ.

Подтип — это подмножество экземпляров супертипа, выделенное по значению некоторого признака. Множество подтипов, соответствующее множеству допустимых значений признака, называют *кластером*. Сам признак — *дискриминатором кластера*. Дискриминатор является атрибутом супертипа.

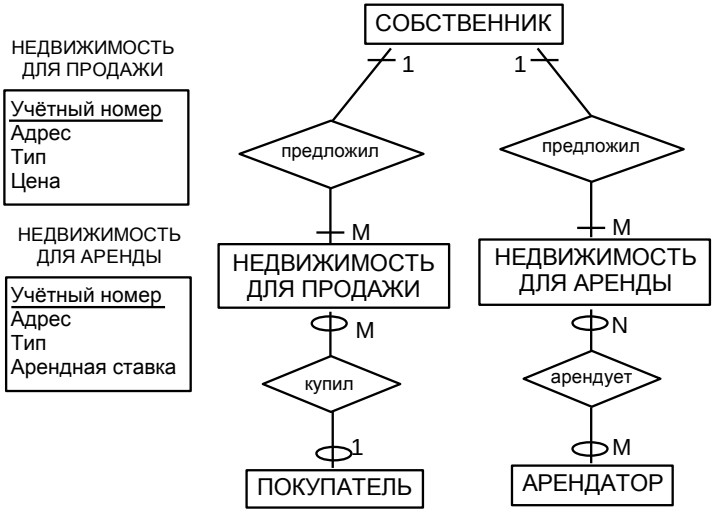


Рис. 2.10 — Сходные сущности в представлении риэлтера

В нашем примере кластер образуют подмножества объектов недвижимости, различающиеся по способу получения дохода.

Сам этот способ есть дискриминатор кластера. Допустимые значения дискриминатора — `продажа` и `аренда`.

Возможна и обратная ситуация: можно выделить подмножества экземпляров некоторой сущности, участвующие в различных связях и/или имеющие различающиеся наборы атрибутов. Например, в представлении инспектора отдела кадров ВУЗа есть сущность `СЛУЖАЩИЙ` — работник ВУЗа. Среди атрибутов этой сущности имеются такие, которые имеют определённые значения для каждого экземпляра, и такие, как `учёная степень`, `учёное звание`, имеющие смысл только для преподавателей. Кроме того, подмножество преподавателей участвует в таких связях, в которых не могут участвовать не преподаватели. Можно выделить и другие подобные подмножества служащих: учебно-вспомогательный персонал, администрация, и т.п. Сущность `СЛУЖАЩИЙ` можно рассматривать как супертип, содержащий перечисленные подтипы.

Современные модификации ER-модели предусматривают специальные обозначения для явного выделения подмножеств экземпляров сущности, имеющих различающиеся наборы атрибутов или вступающих в различные связи. Один из вариантов изображён ниже на рис. 2.11.

Подтип (категория) есть подмножество экземпляров супертипа, т.е. *каждый* экземпляр подтипа *является* экземпляром супертипа. На диаграмме это отражает символ '$\in$'. Каждый подтип наследует все атрибуты супертипа и участвует во всех его связях. Подтип может иметь собственные атрибуты и участвовать в связях, свойственных только ему.

Дуга окружности, пересекающая на диаграмме линии, связывающие подтипы с супертипом, выделяет кластер. Кластер может быть полным или неполным. Кластер *полный*, если *каждый* экземпляр супертипа является экземпляром *хотя бы одного* подтипа. На рис. 2.11, *а*) изображён полный кластер. Это показано перечёркиванием дуг, связывающих подтипы с супертипом.

Если могут существовать экземпляры супертипа, не принадлежащие ни одному подтипу, то кластер называется *неполным*. Такой кластер изображён на рис. 2.11, *б*). Большинство из $6{\cdot}10^9$ человек не относится ни к одной из указанных категорий.

Это показывают овалы около дуг, связывающих подтипы с супертипом.

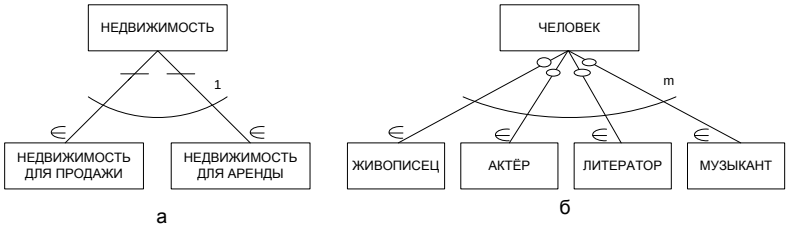


Рис. 2.11 — Обозначения для подтипов и супертипов:
а — не пересекающиеся подтипы, полный кластер;
б — пересекающиеся подтипы, неполный кластер

В ситуации, изображённой на рис. 2.11, *а*) подтипы не пересекаются. Объект не может одновременно продаваться, и сдаваться в аренду. Это показывает символ '1' около дуги кластера. В некоторых случаях экземпляры супертипа классифицируются по признаку, который может принимать несколько значений для одного экземпляра. Тогда подтипы кластера пересекаются. Такая ситуация изображена на рис. 2.11, *б*). Один и тот же человек может быть отнесён к нескольким подтипам по признаку отношения к искусству. Это показывает символ 'm' около дуги кластера.

Заметим в заключение, что сущность может быть супертипом нескольких кластеров. Любой подтип может быть супертипом. Таким образом, введённые обозначения можно использовать для описания иерархий данных.

2.6 Математическая модель сущностей и связей

ER-диаграмма описывает объекты реального мира, их свойства и отношения, выделяемые пользователем. С другой стороны, её можно рассматривать как описание структур данных, с которыми работает пользователь. Выделим минимальные структурные единицы, соответствующие базовым понятиям ER-

модели данных. Для того выясним, что представляют собой сущности, связи и атрибуты с точки зрения математики.

2.6.1 Атрибут

Атрибут — это свойство сущности. Он имеет *имя* и принимает *значения* из некоторого множества значений определённого типа данных — *доме́на.*

Значения не могут быть произвольными, т.к. имеют *смысл*. Смысл определяет условие принадлежности значения типа данных доме́ну. Формально это условие есть предикат `P(X)`, определённый на типе данных. Доме́н — это множество значений типа, на которых `P(X) = .TRUE.` Формально говоря — пара (`тип, предикат`).

> Таким образом, с точки зрения математики атрибут — это переменная, принимающая значения на домене.

2.6.2 Сущность

Сущность — это объект, представляющий интерес для пользователя БД.

Класс сущностей — потенциальное множество объектов одного типа. Класс имеет имя, идентифицирующее тип объекта, и фиксированный набор свойств — атрибутов. Формально класс сущностей — это именованный набор атрибутов.

Экземпляр сущности — это набор значений атрибутов соответствующего класса сущностей (*n-ка, корте́ж*). Математически корте́ж есть элемент декартова произведения доменов атрибутов. Не любой корте́ж имеет смысл с точки зрения пользователя. В каждом конкретном случае существует предикат, принимающий значение `.TRUE.` только на «осмысленных» корте́жах. Потенциальное множество этих «осмысленных» корте́жей и есть класс сущностей.

> Таким образом, класс сущностей есть отношение на декартовом произведении доме́нов атрибутов. Экземпляр сущности есть элемент этого отношения.

2.6.3 Связь

Связь сущностей — это осмысленная с точки зрения пользователя ассоциация сущностей.

Класс связей — ассоциация классов сущностей. Экземпляр класса связей есть ассоциация экземпляров связанных классов сущностей, в которой участвует *точно один* экземпляр *каждого* класса-участника связи. Математически это кортеж экземпляров сущностей, элемент декартова произведения связанных классов сущностей. Не любой такой кортеж имеет смысл с точки зрения пользователя. Существует предикат, принимающий значение `.TRUE.` на «осмысленных» кортежах. Следовательно, класс связей есть отношение, определённое на декартовом произведении классов сущностей.

Экземпляры сущностей имеют уникальную идентификацию. Идентификаторами экземпляров являются значения потенциальных ключей сущностей. Экземпляры сущностей в кортежах класса связей можно без потери информации заменить значениями соответствующих потенциальных ключей. В общем случае класс связей может иметь собственные атрибуты. Поэтому экземпляр класса связей можно трактовать как кортеж, составленный из значений потенциальных ключей связанных классов сущностей и собственных атрибутов связи, — элемент декартова произведения доменов этих атрибутов. Окончательно получаем следующее утверждение.

> Класс связей есть отношение на декартовом произведении доменов атрибутов потенциальных ключей связанных сущностей и доменов собственных атрибутов связи.

2.6.4 Связь как сущность

Итак, класс сущностей и класс связей с точки зрения математики объекты одной природы — *отношения*.

Можно говорить о *сущностных* отношениях, сопоставляемых классам сущностей, и об *ассоциативных* отношениях, сопоставляемых классам связей. Любому ассоциативному отношению всегда можно дать имя, выражающее смысл ассоциации. Это значит, что ассоциативное отношение можно трактовать в

модели как класс (ассоциативных) сущностей. Экземплярами этого класса являются экземпляры соответствующего класса связей. Ассоциативный класс является потомком всех классов-участников связи в бинарных связях типа 1:M.

Пример 1. ГРУППА, ПРЕПОДАВАТЕЛЬ, УЧЕБНАЯ ДИСЦИПЛИНА и АУДИТОРИЯ — сущности, состоящие в тернарной связи. Смысл связи передаётся таким предложением:

«ПРЕПОДАВАТЕЛЬ проводит занятие с ГРУППой по УЧЕБНой ДИСЦИПЛИНе в АУДИТОРИи» (см. рис. 2.12).

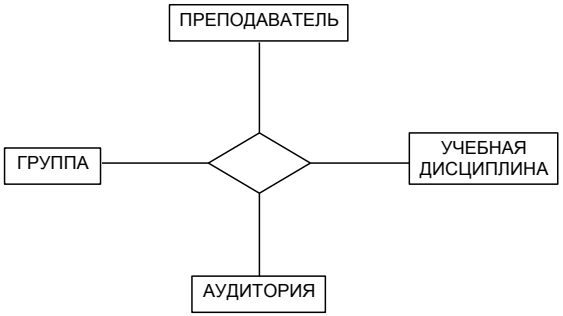


Рис. 2.12 — Ассоциация четырёх классов сущностей, уровень классов

На уровне экземпляров эта ассоциация изображена на рис. 2.13. Чёрными кружками изображены экземпляры связи. Каждый такой кружок представляет факты типа:

«Копков проводит лабораторную работу по ПБД с группой 441-1 в аудитории $438_{ФЭТ}$ в чётную среду с 15:00».

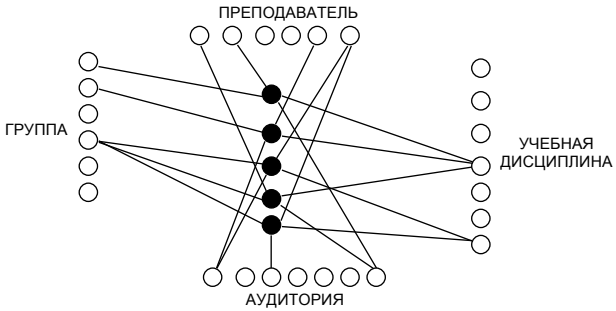


Рис. 2.13 — Ассоциация четырёх классов сущностей, уровень экземпляров

«Копков проводит лабораторную работу по ПБД с группой 441-2 в аудитории $438_{ФЭТ}$ в нечётную среду с 15:00».

«Копков проводит консультации по курсовому проектированию для группы 441-2 в аудитории $404^{б}_{РК}$ в чётный вторник с 13:15».

Эти факты можно трактовать как экземпляры ассоциативной сущности УЧЕБНОЕ ЗАНЯТИЕ — *предусмотренный расписанием сбор в назначенное время и в назначенной* АУДИТОРИИ *студентов* ГРУППы *и* ПРЕПОДАВАТЕЛЯ *для изучения* УЧЕБНой ДИСЦИПЛИНы.

Один экземпляр ПРЕПОДАВАТЕЛя, ГРУППы, АУДИТОРИи или УЧЕБНой ДИСЦИПЛИНы может быть связан с несколькими экземплярами УЧЕБНОГо ЗАНЯТИя. Но каждый экземпляр этой новой сущности связан *точно с одним* экземпляром *каждой* сущности, вступающей в тетрарную связь. Следовательно, рассматриваемая тетрарная связь эквивалентна четырём бинарным связям типа 1:M. Родителями в этих связях выступают ПРЕПОДАВАТЕЛЬ, ГРУППА, АУДИТОРИЯ и УЧЕБНАЯ ДИСЦИПЛИНА. Ассоциативная сущность УЧЕБНОЕ ЗАНЯТИЕ — потомок четырёх родителей (см. рис. 2.14).

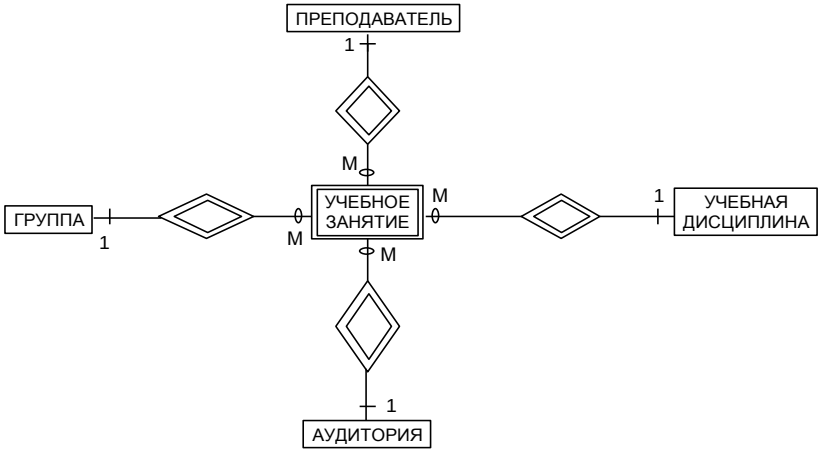


Рис. 2.14 — Ассоциативное отношение, замещающее тетрарную связь

В состав атрибутов сущности УЧЕБНОЕ ЗАНЯТИЕ входят идентификаторы экземпляров всех родителей. Это *слабая* сущность. Её участие во всех четырёх связях обязательно.

Пример 2. Ассоциации ТОВАРа и ПОСТАВЩИКа:

«ПОСТАВЩИК выполнил поставку ТОВАРа»,

можно сопоставить ассоциативное отношение (сущность) ПОСТАВКА. Экземплярами ПОСТАВКи являются экземпляры бинарной связи типа M:N — кортежи значений потенциальных ключей сущностей ТОВАР и ПОСТАВЩИК, и собственных атрибутов ПОСТАВКи, таких как Дата, Количество, Цена и т.п. Ассоциативная сущность ПОСТАВКА является потомком сущностей ПОСТАВЩИК и ПОСТАВЩИК в бинарных связях типа 1:M.

2.6.5 Редукция связей

Из выше сказанного можно сделать следующий вывод.

> Элементарной конструкцией ER-модели данных, с помощью которой можно представить любое отношение классов сущностей, является бинарная связь типа 1:M.

Любую n-арную связь или бинарную связь типа M:N можно редуцировать — преобразовать в эквивалентную (содержащую ту же информацию) совокупность бинарных связей типа 1:M. Для этого нужно заменить редуцируемую связь ассоциативной сущностью. Эта сущность должна быть потомком каждого участника редуцируемой связи в бинарной связи типа 1:M. В состав атрибутов этой сущности должны входить копии потенциальных ключей каждого участника редуцируемой связи.

Любую унарную связь также можно представить с помощью двух бинарных связей типа 1:M. В этом случае ассоциативная сущность должна содержать две копии одного и того же потенциального ключа.

2.6.6 Композитные и многозначные атрибуты

В ER-модели данных допускаются простые, композитные и многозначные атрибуты сущностей. Композитные и многозначные атрибуты принимают значения структурных типов данных.

Посмотрим, как можно представить их посредством простых однозначных атрибутов.

Представление однозначного композитного атрибута сущности посредством простых однозначных атрибутов очевидно. Он по определению есть именованный набор простых однозначных атрибутов. Достаточно разыменовать его, т.е. заменить его имя в схеме сущности подмножеством имён компонентов.

Рассмотрим теперь многозначный атрибут (простой или композитный). По определению он принимает несколько значений одного типа данных для одного экземпляра сущности. Пусть E — сущность, имеющая многозначный атрибут A, e — экземпляр сущности E, Ze = (a1, a2,…,an) — значение атрибута A для экземпляра e. Каждое значение ai можно трактовать как экземпляр некоей сущности EA, являющейся потомком сущности E в бинарной связи типа 1:M. Множество экземпляров EA, связанных с экземпляром e, и есть Ze. Сущность EA имеет однозначный атрибут AA, значениями которого являются элементы значений атрибута A и, кроме того, атрибут, являющийся копией потенциального ключа сущности E.

2.6.7 Резюме

Элементарными конструкциями ER-модели данных являются сущность, имеющая только простые однозначные атрибуты, и бинарная связь типа 1:M. Любые другие конструкции, которые могут встретиться на ER-диаграммах, можно представить с помощью элементарных конструкций без потери информации.

Контрольные вопросы

1. Для чего предназначена модель «сущность-связь».
2. Опишите понятие сущности в ER-модели.
3. Что такое класс (тип) сущностей?
4. Что такое экземпляр сущности?
5. Приведите примеры типов и экземпляров сущностей.
6. Опишите понятие атрибута в ER-модели.
7. Что такое домен атрибута?

8. Какие типы атрибутов допускаются в ER-модели?
9. Приведите примеры атрибутов.
10. Каковы критерии различения сущностей и атрибутов в практике моделирования?
11. Что называется потенциальным ключом сущности?
12. Сформулируйте математическое определение класса сущностей.
13. Опишите понятие связи в ER-модели.
14. Что такое класс (тип) связей?
15. Что такое экземпляр связи?
16. Приведите примеры типов и экземпляров связей.
17. Сформулируйте математическое определение класса связей.
18. Что такое степень (арность) связи?
19. Приведите примеры унарных, бинарных и тернарных связей.
20. Что такое мощность (кардинальность) связи?
21. Что понимается под степенью участия сущности в связи?
22. Опишите какую-либо общепринятую систему графических обозначений ER-модели.
23. Перечислите и опишите на уровне экземпляров известные Вам типы бинарных связей.
24. Опишите на уровне экземпляров преобразование бинарной связи типа `M:N` в пару бинарных связей типа `1:M` без потерь информации.
25. Опишите на уровне экземпляров преобразование n-ар-ной связи в n-ку бинарных связей типа `1:M` без потерь информации.
26. Что называется родовой сущностью (супертипом), категорией (подтипом), кластером категорий, дискриминатором кластера?

3 РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

3.1 Общая характеристика модели

Реляционная база данных (РБД) на концептуальном уровне ANSI/SPARC выглядит как совокупность взаимосвязанных *простых таблиц*. РБД управляется реляционной СУБД (РСУБД). Реляционные СУБД появились на рынке программных продуктов в конце 70-х годов XX века и быстро вытеснили господствовавшие там иерархические и сетевые СУБД. И это несмотря на то, что они существенно уступали (по крайней мере, вначале) своим конкурентам в производительности.

РСУБД и РБД базируются на реляционной модели данных (РМД). РМД предложена в 1970 году американским математиком Эдгаром Ф. Коддом. В это время фирма IBM работала над проектом, известным под названием System/R. Проект был направлен на создание СУБД нового поколения и обобщал накопленный опыт эксплуатации систем с базами данных. По замыслу разработчиков новая СУБД должна была иметь входной язык, доступный пользователю-непрограммисту. Все СУБД, представленные к тому времени на рынке программного обеспечения, имели только интерфейс прикладного программиста. Конечный пользователь обращался к СУБД *исключительно* через посредство приложения. Приложения выполняли обработку *предопределённых* запросов. Любой новый запрос требовал модификации приложения, как правило, разработки новой прикладной программы. Пользователь вынужден был обращаться к программистам с любой «мелочью». СУБД, предоставляющая конечному пользователю доступ к БД, минуя программиста, сильно укрепила бы позиции фирмы IBM на рынке. Оказалось, что РМД является хорошей основой для входного языка такой СУБД. Экспериментальная РСУБД System/R была создана в начале 70-х годов XX века. Она не избавила пользователя от программиста, но зато существенно облегчила задачи создания приложений. Многие решения, найденные в ходе разработки System/R, до сих пор служат ориентиром для проектировщиков СУБД.

Первая промышленная РСУБД DB2 также создана фирмой IBM. Она появилась на рынке в конце 70-х годов и до настоящего времени успешно эксплуатируется во многих организациях. Входной язык DB2, известный ныне под названием SQL (*Structured Query Language*), стандартизован и поддерживается всеми современными РСУБД.

Причиной популярности РСУБД среди разработчиков СБД является простота входного языка этих систем. Входной язык РСУБД содержит

— *язык определения данных (ЯОД)* — средства определения объектов РБД на логическом уровне и

— *язык манипулирования данными (ЯМД)* —непроцедурные средства описания операций *выборки/обновления данных*.

Структуры данных РБД — таблицы — просты для понимания. Каждая таблица представляет какой-то объект предметной области. Связи объектов в концептуальной схеме РБД определяются явно. Поэтому схема БД понятна конечному пользователю на интуитивном уровне. Операторы манипулирования данными (выборка/обновление) являются, по сути, описаниями свойств требуемых наборов значений данных и не содержат каких-либо указаний на то, как эти значения извлечь из БД или поместить на хранение.

Например, запрос на выборку сведений о студентах может выглядеть (в неформальной записи) так:

«Получить значения полей `Фамилия, Имя, Отчество, НомерСтудбилета` из тех строк таблицы СТУДЕНТ, в которых значение поля `НомерГруппы = 10801`».

Формальная запись этого запроса ничуть не сложнее:

```
SELECT Фамилия, Имя, Отчество, НомерСтудбилета
   FROM СТУДЕНТ
   WHERE НомерГруппы = '10801';
```

Средствами реляционного ЯМД можно сформулировать любой запрос к данным, без какого бы то ни было программирования. Для этого достаточно знать концептуальную схему БД и синтаксис деклараций ЯМД.

Опытные конечные пользователи могут взаимодействовать с реляционной системой без посредства программистов, что полностью исключено для более ранних систем.

РМД представляет собой набор понятий и языковых конструкций, предназначенных для описания структур данных, ограничений целостности данных и операций манипулирования данными *на логическом уровне.*

Модель формализует интуитивные представления о таблицах и операциях над таблицами, имеющиеся у каждого, кто когда-либо имел дело с книгами учёта, картотеками, заказами и пр.

Модель состоит из трех частей — структурной, целостностной и манипуляционной.

- *Структурная часть* ставит в соответствие интуитивному понятию таблицы математический объект — отношение и определяет набор абстрактных языковых конструкций для описания отношений.
- *Целостностная часть* содержит правила, которым должны подчиняться хранящиеся в любой реляционной БД непротиворечивые правдоподобные данные.
- *Манипуляционная часть* предоставляет набор операций над отношениями и набор специальных языковых средств для точного формулирования требований к данным, которые должны быть извлечены из БД.

В настоящем разделе изложены базовые понятия структурной и целостностной частей РМД.

3.2 Структуры

3.2.1 Основные понятия

Выделяется шесть базовых понятий РМД: *тип данных, домéн, атрибут, схема отношения, кортеж, отношение*. Первые три относятся к элементам данных, остальные — к структурам, объединяющим элементы.

Тип данных есть множество значений, не имеющих внутренней структуры. Это совпадает с понятием *простого типа данных* в программировании. Реляционные СУБД обычно поддерживают числовые, символьные, битовые и логические типы, а также специальные числовые типы, такие, как деньги, даты, время и т.п.

Домéн есть подмножество элементов типа. Формально домéн определяется как пара *(тип, предикат)*. Предикат задает условия принадлежности элемента типа домéну. На одном и том же типе можно определить произвольное число домéнов.

Например, на символьном типе данных (CHAR, TEXT и т.п.) можно определить домéны

– рабочих дней недели как множество значений

```
{'пн', 'вт', 'ср', 'чт', 'пт'};
```

– номеров телефонов как множество десятизначных последовательностей цифр;

– русских мужских имён (путём прямого перечисления)

и т.д.

Атрибут есть имя, поставленное в соответствие домéну. Если домéну поставлено в соответствие имя, то говорят, что *на домéне определен атрибут*. Атрибут *принимает значения* на домéне и наследует его свойства.

На одном и том же домéне можно определить произвольное число атрибутов.

Схема отношения. Пусть D_1, D_2, ...D_n — домéны (необязательно различные) и A_1, A_2, ..., A_n — атрибуты, определенные на соответствующих домéнах.

Определение 1. Множество R пар (домéн, атрибут):

$R = \{(D1, A1), (D2, A2), \ldots, (Dn, An)\}$

называется схемой отношения.

Интуитивно схему отношения можно понимать как заголовок таблицы. Атрибуты — это имена столбцов таблицы, домéны — множества значений, которые могут встретиться в столбцах.

Кортеж. Пусть R — схема отношения, A_i — атрибут схемы, D_i — домéн атрибута A_i, $a_i \in D_i$ — значение атрибута A_i.

Определение 2. Множество пар:

$S_R = \{(A_1, a_1), (A_2, a_2), \ldots, (A_n, a_n)\}$,

$a_i \in D_i$, $i = 1, \ldots, n$,

называется *кортежем*, соответствующим схеме R.

На интуитивном уровне кортеж представляется как строка таблицы с заданным заголовком.

Например, пусть Номера — домéн трехсимвольных строк, со-

ставленных из цифр '0', '1',...'9', Имена — домéн строк символов русского алфавита, пробелов и точек, а схема отношения СЛУЖАЩИЙ имеет вид:

{(Номера, НомерСлуж), (Имена, ИмяСлуж)}.

Кортежи этого отношения могут быть такими:

{(НомерСлуж, '345'), (ИмяСлуж, 'Иванов И.И.')},
{(НомерСлуж, '938'), (ИмяСлуж, 'Петров П.П.')}.

Отношение на интуитивном уровне можно понимать как таблицу, заголовком которой является строка атрибутов, а значимыми строками — строки их значений, однако это неточное представление.

Определение 3. Множество кортежей S_R, соответствующих одной и той же схеме R, называется *отношением*.

Отношение можно понимать как *структурный тип данных*. Тип определяется схемой отношения. Все кортежи — *значения типа* — удовлетворяют одной и той же схеме.

Можно говорить об *экземпляре* (*текущем значении*) отношения с заданной схемой. *Экземпляр (значение)* отношения — это набор кортежей с заданной схемой, существующий в некоторый фиксированный момент времени.

К сожалению, термином «отношение» обозначают как тип, так и экземпляр.

Отношение характеризуется:

– арностью (степенью) — числом пар <домен, атрибут> в схеме;

– мощностью — числом кортежей, составляющих тело отношения.

Так, приведенные выше кортежи образуют *бинарное* отношение *мощности* 2.

Степень — это характеристика типа, а мощность — характеристика экземпляра отношения. Степень отношения фиксирована. Мощность может изменяться во времени.

Отношение является *единственной* структурной единицей РМД.

Замечание. Далее для упрощения записи примеров мы будем опускать имена домéнов в описании схемы отношения и имена атрибутов в записи кортежа. Отношение и его схему будем обозначать одним и тем же символом R. Если понадобится

явно различить схему и отношение, мы сохраним это обозначение за отношением, а схему будем обозначать символом R(·).

3.2.2 Свойства отношений

Отношения РМД обладают рядом свойств, отличающих их от обычных теоретико-множественных отношений.

Атомарность значений атрибутов. Схема отношения не может включать атрибут, значения которого имеют внутреннюю структуру. Это следует из определения атрибута. Атрибут принимает значения на домéне, а домéн — подмножество простого типа данных. Таким образом, РМД не рассматривает так называемые *ненормализованные* отношения.

Уникальность атрибутов. Одноименные атрибуты недопустимы. В противном случае в схеме отношения могут оказаться дубликаты пар (*домéн, атрибут*), что противоречит определению. Кроме того, только за счёт уникальности атрибутов можно отнести значение из кортежа к определенному домéну.

Неупорядоченность атрибутов. Это свойство следует из определения схемы отношения как множества пар (*домéн, атрибут*). Разумеется, определяя схему отношения, мы выпишем эти пары в каком-то порядке. Однако этот порядок *не существенен*. Он никак не используется в операциях над отношениями и *не должен* поддерживаться реляционной СУБД. Существенна уникальность атрибутов.

Уникальность кортежей. Так как отношение есть множество кортежей, в нем не может быть дубликатов кортежей. Из этого свойства следует, что схема каждого отношения содержит некоторое подмножество атрибутов, значения которого уникально идентифицируют кортежи. Этот набор атрибутов называют *возможным ключом* отношения. Формальное определение этого понятия приведено ниже.

Неупорядоченность кортежей. Это также следствие того, что отношение — множество кортежей. Множества не упорядочены, если их упорядоченность специально не оговорена. Заметим, что хранимые в памяти системы данные так или иначе упорядочены. Однако, как и в случае атрибутов, эта упорядочен-

ность не существенна, и РСУБД *не должна* её поддерживать. Существенна уникальность кортежей. Можно найти любой кортеж отношения, указав значение его возможного ключа.

Изменяемость отношений. Тело отношения (набор кортежей) может изменяться во времени. Отдельные кортежи могут добавляться или удаляться. Могут изменяться значения атрибутов в существующих кортежах.

3.2.3 Интуитивная основа РМД

Приведённые здесь определения структурных понятий РМД — это не более чем точные формулировки интуитивных представлений о простых (плоских) таблицах.

В самом деле, создавая таблицу на бумаге, мы поименуем её столбцы различными именами, а расположим эти имена, скорее всего, в *произвольном* порядке. Вписывая в таблицу строку, мы будем следить за тем, чтобы значения в ней были расположены в том же порядке, что и имена столбцов в заголовке. При этом нам совершенно безразличен относительный порядок имён. Кроме того, мы будем следить за тем, чтобы значения в столбцах были осмысленными и подчинялись определённым ограничениям. Так, вводя значение в столбец *ДатаРождения*, мы будем следовать действующему соглашению о формате представления даты. Если дата должна представляться в виде *'дд.мм.гг'*, то мы напишем *'29.09.03'*, а не *'29/09/03'* и уж, тем более, не *'Мартобря никакого числа, день был без числа'*.

Отыскивая в таблице нужные строки, мы будем сравнивать с заданными эталонами значения столбцов в строках. Учитывать физическую упорядоченность строк в алгоритме поиска мы не будем.

Заслуга Кодда состоит в том, что он сумел отделить *существенное* в этих представлениях от *несущественного* и точно сформулировать представления о существенном в виде определений математических объектов и операций на множествах этих объектов.

3.2.4 Семантика конструкций РМД

Структурные абстракции РМД — это средства описания элементов и логических структур данных.

С формальной точки зрения атрибут может быть произвольным именем: A, B, C и т.п. В реальном проекте атрибут сопоставляется некоторому элементу данных — *реквизиту* — свойству объекта предметной области (сущности): `Скорость`, `Вес`, `ДатаРождения` — это семантически значимые имена, передающие смысл соответствующих элементов данных.

Домéн атрибута — это множество допустимых, имеющих смысл значений реквизита. Так, в БД станции наблюдения за Космосом `Скорость` имеет смысл скорости объекта в системе координат станции. Она может быть положительным или отрицательным действительным числом (тип REAL). Абсолютное значение скорости ограничено. Домéн атрибута `Скорость` есть множество значений типа REAL, удовлетворяющих этому ограничению.

В базе данных ГИБДД, хранящей сведения об авариях, атрибут `Скорость` имеет смысл скорости автомобиля относительно дорожного полотна в момент возникновения аварийной ситуации. Домéн этого атрибута — множество неотрицательных значений типа REAL.

Таким образом, тип данных и предикат, определяющие домéн в реальном проекте, должны быть выбраны в соответствии со смыслом атрибута.

Определяя домéны и атрибуты, мы тем самым задаем некоторые ограничения на операции над элементами данных, обусловленные смыслом данных. Например, `Вес` и `Скорость` нельзя сравнивать или использовать совместно в арифметических операциях, несмотря на то, что и то, и другое — числа. Эти атрибуты должны быть определены на *различных* домéнах числового типа. Эти домéны различны, даже если содержат *одни и те же* значения.

Атрибуты, определенные на общем домéне, *сравнимы*. Например, длина и ширина должны быть определены на общем домéне, т.к. их сравнения осмысленны. А атрибуты `Табель-`

ныйНомер и НомерТелефона должны быть определены на разных доме́нах. Кроме того, эти доме́ны не могут быть числовыми, несмотря на то, что их значения — последовательности цифр. Никто не складывает и не умножает табельные и телефонные номера.

Абстрактная схема отношения может быть произвольным набором атрибутов. Однако в реальном проекте БД схема имеет смысл набора свойств сущности. Например, свойства объектов СТУДЕНТ и ГРУППА могут быть представлены схемами отношений:

ГРУППА(НомерГруппы, ГодНабора, Специальность);
СТУДЕНТ(НомерСтудбилета, Фамилия, Имя, Отчество, НомерГруппы).

Кортежи отношения имеют смысл экземпляров сущностей, т.е. записей, содержащих сведения о конкретных группах, студентах и т.п. Например:

(431-1, 2001, 220400) — группа 431-1, набор 2001 года, специальность 220400 — программное обеспечение вычислительной техники и автоматизированных систем.

(74343112, Иванов Иван, Иванович, 431-1) — студент Иванов Иван Иванович, зачисленный в группу 431-1 и получивший студбилет с номером 74343112.

Отношение представляет в реальной РБД набор записей обо всех существующих в настоящий момент экземплярах соответствующего объекта ПО. Так, отношение ГРУППА содержит записи обо всех студенческих группах, созданных приказом ректора и не расформированных к настоящему моменту. Отношение СТУДЕНТ — это набор записей обо всех студентах, зачисленных в ВУЗ и не отчисленных приказом ректора.

Реляционная база данных — это набор взаимосвязанных отношений.

3.3 Реляционная целостность

3.3.1 Возможные ключи отношения

Ключ отношения — одно из важнейших понятий РМД. Именно механизмы ключей обеспечивают адресацию кортежей и под-

держание связей отношений в РБД. Существует два типа ключей: возможный и внешний. Дадим точные определения этих понятий и обсудим роль ключей в реляционной базе данных.

Мы уже упоминали, что возможный ключ отношения — это подмножество атрибутов схемы, значения которого не повторяются в различных кортежах. Дадим теперь формальное определение этого понятия.

Определение 5. Пусть `R(K, A)` — схема отношения и `K` $\subset$ `R(·)` — подмножество атрибутов схемы. Подмножество `K` называется *возможным (потенциальным) ключом* отношения, если

А) в любой момент времени в текущем значении `R` не существует двух кортежей с одинаковым значением `K`;

Б) никакое подмножество `L` $\subset$ `K` не обладает свойством А).

Свойство А) называется свойством *уникальности*, а свойство Б) — свойством *неизбыточности*. Имея это в виду, можно сказать так:

Возможный ключ отношения — это уникальное неизбыточное подмножество атрибутов его схемы.

Возможный ключ может быть *простым* — состоящим из одного атрибута или *составным* — состоящим из нескольких атрибутов. Отношение может иметь несколько возможных ключей, как простых, так и составных.

Примеры

Отношение

`СТУДЕНТ(НомерСтудбилета, Фамилия, Имя, Отчество, НомерГруппы)`

имеет простой возможный ключ {`НомерСтудбилета`}. Невозможно одновременное существование двух кортежей отношения `СТУДЕНТ` с одинаковыми значениями этого атрибута.

Заметим, что любое подмножество атрибутов схемы, включающее атрибут `НомерСтудбилета`, обладает свойством уникальности. Однако ни одно из них нельзя считать возможным ключом. Каждое избыточно.

Отношение

ПОСТАВЩИК(КодПоставщика, Наименование, ПочтовыйИндекс, Город,Улица, №Дома, Офис, Телефон, Факс, e-mail)

имеет несколько возможных ключей.

{КодПоставщика} (уникальный идентификатор поставщика в документации фирмы), {Телефон}, {Факс}, {e-mail} — это простые возможные ключи.

{Наименование, ПочтовыйИндекс} и {Наименование, Город} — два *составных* возможных ключа. Могут быть одноимённые фирмы в различных «градах и весях» нашей необъятной державы. Различно поименованные поставщики могут располагаться в одном городе или почтовом округе. Но одноимённые — никогда.

{Город, Улица, №Дома, Офис} — ещё один составной возможный ключ этого отношения. Не могут две различные фирмы занимать одно и то же помещение.

Из приведённых примеров можно сделать следующий вывод.

> Заключение о том, что некоторое подмножество атрибутов является возможным ключом отношения, можно сделать только исходя из смысла данных и правил бизнеса.

Важно заметить, что понятие возможного ключа — *логическое*. Его не следует путать с физическим понятием уникального индекса. Вовсе не обязательно должен существовать в ФБД индекс по потенциальному ключу. Какой-то специальный путь доступа в ФБД, конечно, есть, но этот путь — забота проектировщика физических структур, СУБД и ОС. Конечному пользователю о нем ничего знать не нужно.

3.3.2 Первичный и альтернативные ключи

Значения *любого* возможного ключа являются уникальными идентификаторами кортежей отношения. Если отношение имеет несколько возможных ключей, то из практических соображений выделяют какой-то один. Именно его значения считают идентификаторами кортежей. Этот выделенный возможный ключ называется *первичным ключом* отношения. Все остальные возможные ключи называют *альтернативными*.

РСУБД поддерживает единственный механизм идентификации кортежей — механизм первичного ключа. Поэтому для каждого отношения в РБД должен быть определён первичный ключ.

Замечание. РСУБД всегда создаёт собственные (системные) уникальные идентификаторы кортежей — так называемые *суррогатные* ключи. Поэтому для СУБД все кортежи различны. Однако эти ключи не видны пользователю. Если малограмотный «проектировщик» пренебрегает требованием первичного ключа, то его «база данных» будет содержать не данные, а мусор. Например, такой, как в следующей таблице.

НомерСтб	Фамилия	Имя	Отчество	НомерГр
440123	Свидригайлов	Константин	Константинович	10880
345678	Глокая	Куздра	Бокровна	18001
440123	Глокая	Куздра	Бокровна	18001
440123	Свидригайлов	Константин	Константинович	10880
	Глокая	Куздра	Бокровна	18000

Свидригайлов из первой и четвёртой строк — это одно и то же лицо? Если «да», то зачем дважды? Если «нет», то как их различить?

Глокие из второй и третьей строк — это абсолютные тёзки, зачисленные в одну группу? Такое возможно, хотя и маловероятно. Но почему одна из них имеет такой же номер студбилета, как и Свидригайлов?

А ещё одна «абсолютная тёзка», «проживающая» в последней строке, она что, не имеет студбилета? Потеряла? Или не имела, т.е. не зачислена в ВУЗ?

Если БД состоит хотя бы из десятка подобных таблиц, то человеку вполне по силам натолкать в них десятки тысяч «мусорных» строк. А вот обнаружить мусор среди хотя бы сотен «правильных» строк человеку уже не по силам.

Грамотная СУБД сообщает малограмотному проектировщику о некорректности определения отношения без первичного ключа. Она может предложить ему свои услуги в этом деле. Если «проектировщик» их примет, то получит, например, такую таблицу.

Код	НомерСтб	Фамилия	Имя	Отчество	НомерГр
1	440123	Свидригайлов	Константин	Константинович	10880
2	345678	Глокая	Куздра	Бокровна	18001
3	440123	Глокая	Куздра	Бокровна	18001
4	345678	Свидригайлов	Константин	Константинович	10880
5		Глокая	Куздра	Бокровна	18000

Здесь столбец Код добавлен в структуру таблицы системой. Фактически это суррогатный ключ, отображаемый для пользователя. Значения этого столбца отражают хронологический порядок создания строк и больше никакого смысла не имеют. Все проблемы, связанные с отсутствием семантически значимого первичного ключа, остались.

3.3.3 Связи отношений и внешние ключи

Поскольку отношения представляют в РБД сущности предметной области, связи сущностей отображаются связями отношений. РМД не содержит какой-то специальной структуры для представления связей отношений. Связь трактуется моделью как *ассоциация кортежей*. Т.е. средствами РМД можно описать только *ассоциативные* отношения сущностей.

Работая с бумажными таблицами, мы широко используем приём, который «получил прописку» в РМД как единственный механизм связывания отношений. Например, если перед нами лежит список студентов ВУЗа в виде

НомерСтб	Фамилия	Имя	Отчество	НомерГр
440123	Свидригайлов	Константин	Константинович	10880
345678	Глокая	Куздра	Бокровна	18001
440127	Глотов	Николай	Арнольдович	18001
444423	Свиристелкин	Константин	Константинович	10880
…	…	…	…	…

ГРУППА

НомерГр	Кафедра	Специальность
1088	АСУ	220400
18001	РЗИ	123456
18000	АСУ	351400

то предполагается, что где-то (возможно, также перед нами) есть аналогичный список групп, и колонка НомерГр списка студентов содержит ссылки на строки списка групп. Эта колонка «ассоциирует» СТУДЕНТов и ГРУППы. Так, Свидригайлов

и Свиристелкин *зачислены* (смысл ассоциации) в группу 10880, которую кафедра АСУ ведёт к овладению вершинами специальности 220400.

В этом и состоит суть того, что называется в РМД внешним ключом отношения. Столбец НомерГр списка групп обладает свойством потенциального ключа отношения ГРУППА. Одноимённый атрибут отношения СТУДЕНТ, значения которого суть *ссылки* на кортежи отношения ГРУППА, является *внешним ключом*. Отметим, что требование *одноимённости* первичного ключа отношения ГРУППА и соответствующего внешнего ключа *несущественно*. Существенно то, что они принимают значения из общего домéна.

Дадим теперь формальное определение внешнего ключа.

Определение 6. Пусть R_1(FK, A) — отношение и FK — некоторое подмножество атрибутов его схемы.

Подмножество FK называется *внешним ключом* (Foreign Key), если

А) существует отношение R_2(PK, B) с первичным ключом PK (Primary Key), эквивалентным FK;

Б) каждое значение FK в текущем значении R_1 *всегда совпадает* со значением PK некоторого кортежа в текущем значении R_2.

Пояснение. Множества атрибутов A и B *эквивалентны* (A = B), если их мощности равны и для каждого атрибута X ∈ A существует Y ∈ B, *определённый на том же домéне*.

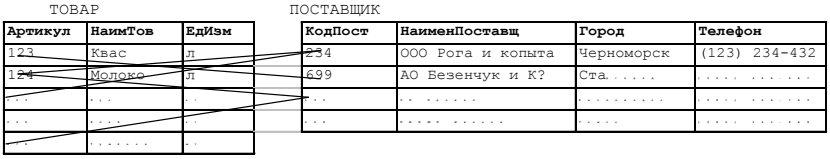
Отношение R_1 называют *потомком* или *ссылающимся* отношением, а R_2 — *родительским* или *ссылочным* отношением.

Механизм внешнего ключа поддерживает только бинарные ассоциации 1:1 и 1:M.

Бинарные связи типа M:N и ассоциации высших степеней представляются *отношениями*. Схемы этих отношений-ассоциаций содержат внешние ключи, эквивалентные первичным ключам ассоциируемых отношений.

Пример 1

Рассмотрим два отношения: ТОВАР и ПОСТАВЩИК (первичные ключи в заголовках таблиц подчёркнуты). Пусть между ними существует ассоциация типа M:N «поставлен/поставил». Один и тот же поставщик может поставить несколько видов товаров и один и тот же вид товара может поставляться различными поставщиками.



Представить эту ассоциацию средствами РМД можно *единственным* способом. Следует определить отношение, схема которого содержит атрибуты Артикул и КодПост — внешние ключи, соответствующие первичным ключам отношений ТОВАР и ПОСТАВЩИК. Реализация этого отношения приведена ниже.

ПОСТАВКА

Артикул	КодПост	ДатаПост	Объём (ед)	ЦенаЕд
124	135	23.12.03	234	12,00
223	234	23.12.03	123	6,00
678	135	12.12.03	145	30,00
124	234	12.12.03	100	15,00
123	699	12.12.03	1000	3,00

Замечание 1. У многих студентов возникает «замечательная» идея «реализации» связи M:N:

«Добавим в схему отношения ТОВАР внешний ключ КодПост, а в схему отношения ПОСТАВЩИК — внешний ключ Артикул».

Если Вам не очевидна бесперспективность этой идеи, запишите строки модифицированных так таблиц ТОВАР и ПОСТАВЩИК, сохраняющие информацию, представленную в таблице ПОСТАВКА.

Пример 2

Пусть между отношениями ПРЕПОДАВАТЕЛЬ(ТабНом,…), ГРУППА(НомГр,…) и ДИСЦИПЛИНА(Наимен,…) существует ассоциация «ПРЕПОДАВАТЕЛЬ преподаёт ДИСЦИПЛИНу для ГРУППы». Это ассоциация третьей степени. Она может быть представлена средствами РМД единственным способом — путём создания отношения со схемой {ТабНом, НомГр, Наимен}. Все атрибуты этой схемы являются внешними ключами, соответствующими первичным ключам ассоциируемых отношений.

Замечание 2. Ещё одна «замечательная» идея, которая посещает многих студентов, в условиях вышеприведённого примера выглядит так:

«Заменим ассоциацию трёх отношений ассоциациями «ПРЕПОДАВАТЕЛЬ преподаёт ДИСЦИПЛИНу», «ПРЕПОДАВАТЕЛЬ преподаёт для ГРУППы» и «ГРУППА изучает ДИСЦИПЛИНу».

По правилам высшей школы все три ассоциации имеют тип M:N. В этих условиях из тройки утверждений «Копков преподаёт БД», «Копков преподаёт для группы 10890» и «Группа 10890 изучает БД» *вовсе не следует* утверждение «Копков преподаёт БД для группы 10890».

Если Вам это не очевидно, то запишите строчки таблиц на бумаге и вообразите себя компьютером, который должен ответить на вопрос: «Для каких групп А преподаёт В?»

3.3.4 Внутренние ограничения целостности РМД

Внутренние ограничения целостности РМД — это набор средств описания делового регламента (бизнес-правил) предметной области. Мы уже упоминали об этих ограничениях, когда говорили о структурах. Резюмируем сказанное выше и обсудим роль внутренних ограничений целостности в РБД.

Принято выделять четыре вида внутренних ограничений целостности РМД: ограничения домéна, атрибута, сущности и ссылочные.

Ограничение целостности домéна — это определение множества значений простого типа данных, которые образуют этот

домéн. Домéн может быть задан либо путём прямого перечисления всех его значений, либо указанием предиката, определяющего условие принадлежности значения домéну.

С точки зрения программиста, домéн — это тип данных, определённый пользователем. Требование целостности домéна сводится к хорошо известному всем программистам правилу:

Должны быть явно определены все необходимые типы данных (домéны).

Ограничение целостности атрибута — это определение домéна, из которого выбираются значения атрибута. Требование целостности атрибута таково:

Значения атрибута должны выбираться только из его домéна.

Это требование также хорошо знакомо любому программисту в следующей формулировке: «Переменная может принимать только значения своего типа данных».

Ограничение целостности сущности — это определение первичного ключа отношения.

Требование целостности сущности:

Для каждого базового отношения должен быть определён первичный ключ.

С практической точки зрения это означает, что в базе данных не должно быть неидентифицируемых кортежей.

Ограничение ссылочной целостности — это определение внешнего ключа отношения.

Требование ссылочной целостности:

База данных не должна содержать значений внешнего ключа, не совпадающих с каким-либо значением первичного ключа в существующих кортежах родительского отношения.

Сформулированные выше требования — это *метаправила*. В совокупности они означают, что в схеме конкретной реляционной БД в соответствии со смыслом данных и бизнес-правилами должны быть определены домены всех атрибутов, первичные ключи всех базовых отношений и конкретные правила внешних ключей. Только тогда РСУБД сможет своими средствами обеспечивать согласованные обновления данных.

3.3.5 Правила внешних ключей

Правила внешних ключей определяют реакцию РСУБД на появление значений внешнего ключа, которых нет среди существующих значений родительского ключа. Такая ситуация может возникнуть при попытке

– добавления кортежа в отношение-потомок,
– удаления кортежа отношения-родителя,
– обновления значения первичного ключа в кортеже отношения-родителя.

В первом случае решение очевидно. Должно быть запрещено добавление в отношение-потомок кортежа, ссылающегося на несуществующий кортеж родителя.

Во втором случае нужно ответить на вопрос: «Какие действия должна выполнить система, если предпринимается попытка удаления кортежа родительского отношения, на который ссылается какой-то кортеж отношения-потомка?» Например, если нам нужно удалить из БД сведения о поставщике «Рога и копыта» (Остап Ибрагимович уехал в Рио-де-Жанейро, фирма закрылась), то что делать со второй и четвёртой записями о поставках (см. пример 1), которые ссылаются на фирму Великого Комбинатора?

Возможны различные варианты решений.

ОТЛОЖИТЬ удаление родительского кортежа, если существует хотя бы один кортеж потомка, содержащий ссылку на него. Это «ограниченное» (`RESTRICTED`) удаление. Если для таблицы `ПОСТАВКА` из нашего примера установлено такое правило внешнего ключа `КодПост`, то из текущего состояния БД можно удалить только запись о поставщике `ПБОЮЛ Иванов` (`КодПост = 458`). Попытка удаления любой другой записи из таблицы `ПОСТАВЩИК` будет заблокирована системой.

КАСКАДИРОВАТЬ (`CASCADE`) — распространить операцию удаления на все кортежи потомка, ссылающиеся на удаляемый родительский кортеж. Если для таблицы `ПОСТАВКА` из нашего примера установлено такое правило внешнего ключа `КодПост`, то из текущего состояния БД можно удалить *любую* запись. Запись о поставщике `ПБОЮЛ Иванов` (`КодПост = 458`)

будет удалена без «побочных эффектов». На неё нет ссылок в таблице ПОСТАВКА. Запись о поставщике ООО Рога и копыта (КодПост = 234) будет удалена *вместе со ссылающимися на неё* второй и четвёртой записями таблицы ПОСТАВКА.

Варианты действий системы при попытке обновления значения родительского ключа, на которое ссылаются значения внешнего ключа потомка, аналогичны. Например, если мы решили присвоить фирме Безенчук и К° более «приоритетный» код '001', то что делать со значением кода '699' в последней строке таблицы ПОСТАВКА? Можно отложить обновление ссылочного значения родительского или "каскадировать" обновление, т.е. распространить его на ссылающиеся значения.

Замечание. Правило ссылочной целостности *проверяется* при попытке обновления *родительского* отношения, а *задаётся* в определении свойств внешнего ключа *потомка*.

Задача 1. Пусть у отношения R_1 есть два прямых потомка — R_2 и R_3. Для R_2 определено каскадное удаление, а для R_3 — отложенное. Какие варианты действий системы при удалении кортежа отношения R_1 возможны?

Задача 2. Пусть R_2 — отношение-потомок в связи с R_1 и родитель в связи с R_3. Для R_2 определено каскадное удаление, а для R_3 — отложенное. Какие варианты действий системы при удалении кортежа отношения R_1 возможны?

Правила отложенного и каскадного удаления/обновления — это *стандартные* правила ссылочной целостности, поддерживаемые всеми современными РСУБД. Для того чтобы система применяла одно из этих правил, проектировщик БД должен установить соответствующую опцию в определении внешнего ключа. Однако эти правила не исчерпывают все варианты возможных действий системы. Например,

– может быть инициирован диалог с пользователем;

– удаляемые кортежи могут сохраняться в каком-нибудь архиве;

– значения внешнего ключа, ссылающиеся на удаляемый кортеж, могут быть заменены «значениями по умолчанию»

– и т.п.

Вариантов может быть сколько угодно. Очевидно, нельзя требовать, чтобы РСУБД поддерживала их все собственными средствами.

Одна из главных задач проектировщика СБД — реализовать процедуры поддержки правил ссылочной целостности, не поддерживаемых СУБД.

3.4 Неопределённые значения в БД и ограничения целостности данных

3.4.1 Проблема представления незнания

Изложенная выше концепция реляционной целостности данных не учитывает одной важной проблемы. Эта проблема связана с необходимостью представления *незнания* в БД. В реальной жизни, не укладывающейся ни в какие модели, нередко приходится создавать *не вполне определённые* записи об объектах. Например, абитуриент приказом ректора зачислен в состав студентов. В связи с этим мы должны занести сведения о нём в БД ВУЗа. Об абитуриенте Иванове известно всё, кроме адреса. Что занести в поле `Адрес` записи об Иванове?

- Текст *'не известен'*? Но такого значения *нет в домéне* атрибута `Адрес`. РСУБД не может его принять.
- Не регистрировать Иванова до тех пор, пока он где-то не поселится и не сообщит свой адрес? Вряд ли такое решение будет одобрено начальником студенческого отдела кадров.
- Вынудить проектировщика БД определить для атрибута `Адрес` домен, содержащий значение *'не известен'*? Но почему только для этого атрибута? Ведь неизвестными могут оказаться и значения других атрибутов этого же или другого отношения.

Ну, хорошо, потребуем, чтобы каждый домен текстового типа данных содержал строку *'не известно'*. А как быть с доменами числовых типов? Какое число считать *неизвестным*? Ведь *любое* число есть *вполне определённое* значение числового типа.

Если бы мы делали записи на бумаге, то оставили бы «дырку» на пересечении столбца `Адрес` и строки «Иванов». А что такое «дырка» в компьютерной записи? В конце концов, поле —

это последовательность битов. Каждый бит может находиться в *одном из двух* вполне определённых состояний. Какую комбинацию состояний битов следует считать «дыркой»?

Идея технического решения этой проблемы проста. Отведём первый бит каждого поля записи под маркер «*определено*»/«*не определено*». При создании новой записи СУБД устанавливает этот бит *в каждом поле* в состояние «*не определено*» (инициализирует запись). Тем самым значение, представленное прочими битами поля, объявляется *недействительным* (английское NULL ['nʌl]). Если в поле явно вводится какое-то значение, его первый бит переводится в состояние «*определено*».

Очевидно, так созданное «неопределённое значение» *не имеет типа* и может появиться (точнее — остаться) в поле любого типа. Но как с ним должен работать компьютер? Приходится определять специальные правила выполнения операций над числами, строками и т.п., учитывающие возможность появления NULL-значений[2] операндов. Заметим, что это специфическая проблема технологии баз данных.

Одна из наиболее распространённых операций над данными в РБД — *фильтрация* кортежей по какому-то условию. Например, для того чтобы из списка студентов ТУСУРа получить список группы 431-1, нужно из каждого кортежа отношения СТУДЕНТ выбрать значение атрибута НомерГруппы и сравнить его со строковой константой '431-1'. Если текущее значение номера группы совпадает с этой константой, то результат сравнения .TRUE., в противном случае — .FALSE. А если текущее значение номера группы NULL (не имеет типа!), то каким должен быть результат сравнения? По-видимому, неопределённым, .UNKNOWN. В самом деле, *не известно*, в какую группу зачислен этот студент. Может быть, в 431-1, а может быть, нет.

Результатом операции сравнения с возможным участием NULL может быть одно из *трёх* логических значений: .TRUE., .FALSE. или .UNKNOWN. Причём результатом *любой* операции сравнения, в которой участвует NULL, *всегда* будет .UNKNOWN.

---

[2] Фактически NULL является не значением, а *маркером*. Тем не менее мы будем говорить о нём как о значении.

Следовательно, все NULL-значения *различны*. А как тогда отфильтровать кортежи, содержащие NULL в некотором поле? Ведь проверять условие совпадения значения поля в кортеже со значением NULL бессмысленно. Для описания условий фильтрации кортежей приходится вводить трёхзначную логику, определять специальные правила сравнения и логические операции.

3.4.2 NULL-значения и целостность атрибута

Возможность появления неопределённых значений атрибутов в РБД разрушает концепцию реляционного домена. Теперь уже нельзя утверждать, что значение каждого атрибута *должно* выбираться из его домена. Оно *может* выбираться из домена или быть «недействительным» — NULL.

В ряде случаев допустимость NULL-значения атрибута может быть оправдана его смыслом и правилами бизнеса, но далеко не всегда. В самом деле, можно понять, почему значение атрибута ДатаОкончанияЛечения в некотором кортеже БД больницы не определено. Но почему в соседнем кортеже не определено значение атрибута ДатаПоступленияБольного, понять невозможно. Больной ещё не поступил? Тогда зачем мы создали эту запись? Больной поступил, но не известно, когда? А что за раззява его приняла? А не уволить ли её с работы? И т.п.

Для того чтобы не возникало подобных недоразумений, для каждого атрибута каждого отношения следует указать допустимость/недопустимость NULL-значений.

Требование целостности атрибута теперь должно быть сформулировано так:

> Каждое *определённое* значение атрибута должно выбираться только из его домéна.

3.4.3 Идентификация кортежей и NULL-значения

Теоретически идентификатором кортежа отношения может быть значение какого-либо (любого) потенциального ключа. Если известно значение потенциального ключа, то кортеж мож-

но найти во множестве существующих кортежей. Для этого нужно перебирать существующие кортежи и сравнивать значение ключа из текущего кортежа с заданным. Но если в состав возможного ключа входит атрибут, который может принимать NULL-значения, то этот ключ нельзя использовать для идентификации кортежей.

Например, в БД поликлиники есть отношение со схемой

{НомерПолиса, ДатаВыдачиБЛ, КодВрача, Диагноз, ДатаЗакрытияБЛ},

содержащее сведения о выдаче больничных листов. Оно имеет два возможных ключа:

K1 = {НомерПолиса, ДатаВыдачиБЛ},
K2 = {НомерПолиса, ДатаЗакрытияБЛ}.

Атрибуты НомерПолиса и ДатаВыдачиБЛ не могут принимать значения NULL, а атрибут ДатаЗакрытияБЛ может. Допустим, мы решили идентифицировать кортежи этого отношения значениями K2. В БД есть кортежи

{123456, 12.03.03, B24, ОРЗ, 19.03.03};
{123456, 02.06.03, B24, ОРЗ, NULL}.

На какой из них указывает значение k2 = {123456, NULL}? Отнюдь не на второй, поскольку результаты сравнений k2 с наборами значений {123456, 19.03.03} и {123456, NULL} будут *одинаковыми*, именно — .UNKNOWN.

Вывод: ни один компонент потенциального ключа, выбранного для идентификации кортежей, не может принимать NULL-значения.

С учётом этого требование целостности сущности следует сформулировать так:

Для каждого базового отношения должен быть определён первичный ключ, ни один компонент которого не может принимать NULL-значения.

3.4.4 Ссылочная целостность и NULL-значения

Итак, NULL-значения первичных ключей запрещены. А как быть с NULL-значениями внешних ключей? Ведь вполне воз-

можны такие ситуации, когда в поле внешнего ключа приходится «оставлять дырку».

Например, в БД хранятся сведения о сотрудниках конструкторского бюро и о проектах, которые выполняются сотрудниками. По правилам организации проект выполняется несколькими сотрудниками, но сотрудник может участвовать только в одном проекте. Тогда все сведения о сотрудниках, проектах и участии сотрудников в проектах можно представить двумя отношениями:

СОТРУДНИК (ТабельныйНомер, ФИО,..., КодПроекта);

ПРОЕКТ (КодПроекта, ...).

Здесь ПРОЕКТ.КодПроекта — первичный ключ отношения ПРОЕКТ, а СОТРУДНИК.КодПроекта — соответствующий ему внешний ключ отношения СОТРУДНИК.

Однако *не каждый* сотрудник КБ участвует в каком-то проекте. Есть ведь и вспомогательный персонал, об участии которого в конкретных проектах говорить нет смысла. Каким будет значение внешнего ключа в кортеже отношения СОТРУДНИК, представляющем *вспомогательного* сотрудника? Правильно, неопределённым, NULL. Тогда приходится требование ссылочной целостности «подправить» так:

> База данных не должна содержать определённых значений внешнего ключа, не существующих среди значений первичного ключа родительского отношения.

Вот сколько проблем связано с «пустым» с точки зрения человека, работающего с бумажной таблицей, вопросом: «Как отображать в таблице неизвестные значения столбцов в некоторых строках?»

3.5 Реляционный язык определения данных

Из сказанного выше следует, что определение схемы реляционной базы данных должно включать определения доменов, отношений, первичных и альтернативных ключей, внешних ключей и правил ссылочной целостности. Все компоненты схемы объявляются специальными предложениями языка определения данных (ЯОД). Определения объектов сохраняются в си-

стемном каталоге и используются РСУБД в процессах обработки данных.

Рассмотрим синтаксис предложений определения доменов и отношений во входном ЯОД гипотетической СУБД, поддерживающей все требования РМД.

3.5.1 Объявление домена

Определение домена может выглядеть так:

```
CREATE DOMAIN имя-домена тип [(длина)]
  {VALUES (список)} |
  {FOR ALL VALUE (предикат)};
```

Здесь *имя-домена* — уникальное имя, под которым домен будет известен системе, и на которое можно ссылаться в определениях отношений;

тип — один из поддерживаемых системой встроенных типов данных;

длина — длина поля данных в байтах;

список — список значений, разделенных запятыми;

предикат — логическое выражение, ссылающееся на переменную VALUE.

Эта декларация объявляет системе, что она должна внести в свой каталог имя нового объекта — домена, и указывает ограничения на значения, принадлежащие домену. Определение сохраняется в системном каталоге. При любой попытке обновления значения какого-либо атрибута, определенного на этом домене, будет вычислено значение предиката. Если оно окажется равным .FALSE., обновление будет отвергнуто.

Пример 1. Домен рабочих дней недели

```
CREATE DOMAIN  День CHAR (2)
  FOR ALL VALUE (VALUE = 'пн' OR
                 VALUE = 'вт' OR
                 VALUE = 'ср' OR
                 VALUE = 'чт' OR
                 VALUE = 'пт');
```

Это же можно записать короче:

```
CREATE DOMAIN день CHAR (2)
```

```
VALUES ('пн', 'вт', 'ср', 'чт', 'пт');
```

Если, скажем, в ИС банка на этом домене определен атрибут `РабочийДень`, то система сможет заблокировать выполнение банковских операций по субботам и воскресеньям.

Пример 2.

```
CREATE DOMAIN Вес INTEGER
  FOR ALL Вес (Вес ≥ 10 AND Вес ≤ 150
  AND mod(Вес, 5) = 0);
```

Каждый атрибут, определенный на этом домене, может принимать целочисленные значения в интервале [10, 150], причем только такие, которые делятся на 5 без остатка.

Если есть возможность создать домен, должна быть и возможность удалить его. Следующая команда

```
DESTROY DOMAIN имя-домена;
```

удалит из системного каталога определение домена. Она будет исполнена, если в схеме БД нет ни одного атрибута, определенного на удаляемом домене.

Если система поддерживает домены, то в ней есть возможность выполнения запросов, основанных на доменах. Например, запрос

Какие отношения в БД содержат какую-либо информацию о весе?

в терминах доменов имеет вид:

Какие отношения в БД включают атрибуты, определенные на домене `Вес`?

Это запрос к системному каталогу. В системе, не поддерживающей домены, такие сведения практически невозможно получить.

3.5.2 Объявление отношения

Можно говорить о *переменной-отношении*, заданной схемой отношения, и о *значении* отношения, существующем в БД в конкретный момент времени. Определять следует только схему, т.е. переменную. Значения отношения формируются системой в процессе выполнения запросов пользователей.

В реляционной БД всегда существует несколько основных видов отношений.

- *Именованное отношение* — это отношение, имя и определение схемы которого сохранены в системном каталоге. Именованное отношение может быть базовым или производным.
- *Производное отношение* есть отношение, определенное через именованные отношения. Производное отношение может быть именованным или неименованным.
- *Базовое отношение* — именованное отношение, не являющееся производным.

Любое производное отношение, в конце концов, выражается через базовые. Поэтому средства явного определения схемы нужны только для базовых отношений.

Предложение объявления базового отношения имеет вид:

```
CREATE BASE RELATION  имя-отношения
(список-определений-атрибутов
список-определений-возможных-ключей
список-определений-внешних-ключей);
```

Здесь *список-определений* ... — список разделенных запятыми строк определений соответствующих элементов схемы отношения.

Строка определения атрибута связывает имя атрибута с его доменом:

```
имя-атрибута DOMAIN имя-домена.
```

Указанный атрибут принимает значения на указанном домене.

Например, строка определения атрибута ВесДетали в предложении определения отношения ДЕТАЛЬ может иметь вид:

```
ВесДетали DOMAIN (Вес)
```

Предполагается, что домен Вес определен в схеме БД.

Имена атрибутов должны быть уникальны, по крайней мере, в пределах отношения. Удобно, если имя атрибута совпадает с именем домена, на котором он определен.

Строка определения возможного ключа имеет две модификации:

```
PRIMARY KEY (список-атрибутов) |
```

```
CANDIDATE KEY (список-атрибутов)
```

Здесь PRIMARY KEY и CANDIDATE KEY объявляют, соответственно, первичный и альтернативный ключи отношения;

список-атрибутов — список разделенных запятыми имен атрибутов, образующих ключ.

Например, первичный ключ отношения ПОСТАВКА будет объявлен строкой

```
PRIMARY KEY (КодПоставки)
```

Альтернативный ключ этого же отношения

```
CANDIDATE KEY (Артикул, КодПоставщика,
ДатаПоставки),
```

Строка определения внешнего ключа имеет вид:

```
FOREIGN KEY (список-атрибутов)
  REFERENCES  отношение
    ON DELETE правило-удаления
    ON UPDATE правило-обновления
```

Здесь *список-атрибутов* — список разделенных запятыми имен атрибутов (составного) внешнего ключа, эквивалентный списку атрибутов родительского ключа;

отношение — имя родительского отношения;

правило ... следует понимать как ссылку на процедуру БД, реализующую определенное проектировщиком правило внешнего ключа для операции удаления или обновления родительского ключа. Это может быть либо процедура пользователя, либо одна из стандартных процедур СУБД — CASCADE (каскадировать) или RESTRICT (отложить).

Примеры.

Будем считать, что все необходимые домены определены, и приведем предложения определения схем отношений ПОСТАВЩИК и ПОСТАВКА.

Отношение ПОСТАВЩИК не содержит ни альтернативных, ни внешних ключей и определяется предложением

```
CREATE BASE RELATION  ПОСТАВЩИК
(КодПоставщика DOMAIN КодПоставщика,
 Наименование DOMAIN НаименПост NOT NULL,
 Город DOMAIN Город,
 Телефон DOMAIN НомерТелефона,
```

```
  PRIMARY KEY (КодПоставщика));
```

Определение отношения ПОСТАВКА имеет вид

```
CREATE BASE RELATION ПОСТАВКА
(Артикул DOMAIN АртикулТовара,
 КодПоставщика DOMAIN КодПоставщика,
 Дата DOMAIN Дата,
 Объём DOMAIN ОбъёмПоставки,
 Цена DOMAIN Деньги,
PRIMARY KEY (Артикул, КодПоставщика, Дата),
FOREIGN KEY(КодПоставщика) REFERENCES ПОСТАВЩИК
  ON DELETE Restrict
  ON UPDATE Cascade,
FOREIGN KEY (Артикул) REFERENCES ТОВАР
  ON DELETE Restrict
  ON UPDATE Cascade);
```

Здесь для внешних ключей определены правила отложенного удаления и каскадного обновления. Это означает, что удаление кортежа родителя не может быть выполнено, если в отношении ПОСТАВКА существует хотя бы один кортеж, ссылающийся на удаляемое значение родительского ключа. Если же пользователь изменит значение какого-либо родительского ключа, то во всех кортежах отношения ПОСТАВКА, ссылающихся на старое значение, произойдут соответствующие изменения.

Базовое отношение может быть уничтожено командой

```
DESTROY BASE RELATION имя-отношения;
```

Команда удалит из системного каталога все сведения об указанном отношении, но будет выполнена, только если тело отношения пусто. Будут удалены также все производные отношения, в определениях которых есть ссылки на удаляемое.

Контрольные вопросы

1. Определите понятия «домен», «атрибут», «схема отношения», «кортеж», «отношение». Приведите примеры.
2. Какими свойствами обладают отношения РМД?
3. Чем отличается понятие отношения в реляционной модели данных от одноименного понятия в теории множеств?

4. Что понимается под целостностью данных?

5. Чем обусловливаются внешние ограничения целостности? Приведите примеры из Вашей курсовой работы.

6. Сформулируйте определение возможного ключа отношения.

7. Какова роль механизма возможных ключей в реляционной модели данных?

8. Что такое первичный ключ?

9. Какова роль механизма первичных ключей в реляционной модели данных?

10. Сформулируйте определение внешнего ключа отношения.

11. Какова роль механизма внешних ключей в реляционной модели данных?

12. Сформулируйте ограничение целостности домена. Как может быть реализовано это требование в реляционной базе данных? Приведите примеры из Вашей курсовой работы.

13. Сформулируйте ограничение целостности сущности. Как может быть реализовано это требование в реляционной базе данных? Приведите примеры из Вашей курсовой работы.

14. Сформулируйте ограничение ссылочной целостности. Как может быть реализовано это требование в реляционной базе данных? Приведите примеры из Вашей курсовой работы.

4 НОРМАЛЬНЫЕ ФОРМЫ ОТНОШЕНИЙ

Любая компьютерная система с базой данных должна гарантировать целостность данных. Это главное требование. Пользователь откажется от системы, как только обнаружит, что она сохраняет противоречивые данные. Поэтому главная задача проектировщика — обеспечить поддержку бизнес-правил (делового регламента).

Бизнес-правила выражают смысл данных. Они накладывают ограничения на множества значений элементов данных (атрибутов) и на свойства их отношений (связей). Многие бизнес-правила могут быть формально описаны средствами РМД как ограничения целостности атрибута, целостности сущности и ссылочной целостности. Для поддержки таких правил можно использовать механизмы ядра СУБД. Однако значительная часть бизнес-правил не выражается в терминах РМД. Для таких правил проектировщик должен обеспечить процедурную поддержку.

Соотношение системного и процедурного компонентов поддержки бизнес-правил в конкретной СБД зависит от особенностей предметной области и от искусства проектировщика. Однако в любом случае разумно стремиться возложить на СУБД как можно больше работы в этой части.

Как показывает полувековой опыт создания баз данных, наиболее трудные проблемы проектирования связаны с бизнес-правилами, ограничивающими свойства взаимосвязей атрибутов. Здесь мы посмотрим, какие проблемы возникают у пользователей, если эти правила не поддерживаются. В следующих разделах скажем, в каких случаях и как можно обеспечить их поддержку средствами ядра СУБД.

Начнём с примера, иллюстрирующего проблемы.

4.1 Пример. База данных куратора

4.1.1 Проблемы куратора

На факультете существует группа студентов, обучаемых по индивидуальным планам. Все студенты проживают в общежи-

тии. Некоторые совмещают учёбу с работой. Каждый студент в течение семестра изучает несколько учебных дисциплин. До окончания семестра студент должен отчитаться по каждой изучаемой дисциплине. Единственный предусмотренный вид отчётности — экзамен. Один из преподавателей факультета курирует группу. В его обязанности входит текущий и итоговый контроль успеваемости студентов. Для успешного выполнения обязанностей куратору нужно всегда иметь под рукой сведения о студентах и об их успеваемости. Он решил создать компьютерную ИС с базой данных. Разработку проекта поручил одному из своих студентов, любимчику Васе Дубову.

От куратора Вася получил следующие исходные данные для проектирования.

4.1.2 Техническое задание на разработку БД куратора

Цель: накопление сведений о студентах, опекаемых куратором и выборка необходимых данных по запросу куратора.

Точка зрения: куратора группы студентов, обучаемых по индивидуальным планам.

Сохраняемые данные:

`Сном` — номер студбилета студента;

`Сфам` — фамилия, имя, отчество студента;

`Кном` — номер комнаты в общежитии, в которой проживает студент;

`К_Тел` — номер телефона в комнате;

`С_Тел` — номер сотового телефона студента;

`Р_Тел` — номер рабочего телефона студента;

`Сем` — номер семестра, в котором студент сдал экзамен по дисциплине;

`Дис` — дисциплина, сданная студентом;

`Оц` — оценка, полученная студентом на экзамене по дисциплине.

Правила

1. Не существует двух студентов, имеющих студенческие билеты с одинаковыми номерами.
2. Каждый студент проживает в общежитии.
3. Студент проживает в одной комнате.

4. В одной комнате проживает не более трёх студентов.
5. В каждой комнате установлен точно один телефон.
6. Не существует двух комнат с одинаковыми номерами телефонов.
7. Студент может иметь рабочий телефон.
8. Несколько студентов могут иметь один рабочий телефон.
9. Студент может иметь один сотовый телефон.
10. Не существует двух студентов с одинаковыми номерами сотовых телефонов.
11. Студент может получить несколько оценок по дисциплине за период обучения.
12. Студент может получить только одну оценку по дисциплине в семестре.

Замечание. В задании указан ещё ряд правил, ограничивающих допустимые значения атрибутов, форматы отображения значений и т.п. Мы не приводим их здесь. В контексте проблем проектирования структуры БД они не существенны.

Некоторые типовые запросы на выборку данных

1. Вывести результаты экзаменов студента в текущем семестре.
2. Получить номер комнаты, в которой проживает студент.
3. Получить номер телефона комнаты, в которой проживает студент.
4. Получить номер сотового/рабочего телефона студента.
5. Получить списки студентов, проживающих в одной комнате.
6. Получить список студентов, проживающих в одной комнате с заданным.

4.1.3 Варианты «структуры» БД

Вариант 0.1 (имя таблицы `USPEV01`). «Дикая» структура типа 1. Создана Васей в начале работы над проектом, когда он ещё не имел никаких знаний в области проектирования БД. Первичный ключ **`Код`** создан средой разработки.

Код	Сном	Сфам	Кном	Тел	Сем	Дис	Оц
1	0221	Иванков П.С.	923	41-12-13, 80012345061	1	РЯ	5
2	0221	Иванков П.С.	923	41-12-13, 80012345061	1	Инф.	3
3	0223	Латушов К.Ж.	923	41-12-13, 41-28-89	1	Физ.	3
4	0223	Латушов К.Ж.	923	41-12-13, 41-28-89	1	БД	5
5	**0112**	**Дубов Вася**	**123**	**11-21-32**	**1**	**Физ.**	**3**
6	0221	Иванков П.С.	923	41-12-13, 80012345061	1	Ар.	4
7	0223	Латушов К.Ж.	923	41-12-13, 41-28-89	1	ОТУ	4
8	0221	Иванков П.С.	923	41-12-13, 80012345061	1	КМ	3
9	0223	Латушов К.Ж.	923	41-12-13, 41-28-89	1	ИО	5
10	**0112**	**Дубов Вася**	**123**	**11-21-32**	**1**	**Физ.**	**4**
11	0112	Дубов Вася	123	11-21-32	2	ВМ	3

Здесь нарушается правило 12. Последствия для пользователя могут быть следующие.

Запрос типа 1: Получить оценки Дубова Васи за первый семестр.

```
Select Дис, Оц
  from USPEV01
    where Сфам = 'Дубов Вася'
      and Сем = 1;
```

Результат:

Дис	Оц
Физ.	3
Физ.	4

Почему у Васи две оценки по Физ. в первом семестре?
Вторая оценка не по Физ.? Или по Физ., но в другом семестре? А может быть, она не Васина?
Не поддерживается правило 12.

Вариант 0.2 (имя таблицы USPEV02). «Дикая» структура типа 2. Создана Васей, когда он узнал, что такое первичный ключ, и понял, зачем этот ключ нужен. Первичный ключ **{Сном, Сем, Дис}** создан разработчиком. Пользователь **никогда** не сможет ввести две строки с одинаковыми значениями первичного ключа. Недоразумения, подобные этим, невозможны.

Левый столбец не нужен в базе данных, но мы его сохраним здесь в демонстрационных целях. В нём зафиксирована последовательность ввода строк.

№пп	**Сном**	Сфам	Кном	Тел	**Сем**	**Дис**	Оц
1	0221	Иванков П.С.	923	41-12-13, 80012345061	1	РЯ	5
2	0221	Иванков П.С.	923	41-12-13, 80012345061	1	Инф.	3
3	**0223**	**Латушов К.Ж.**	**923**	**41-12-13, 41-28-89**	**1**	**Физ.**	**3**
4	**0223**	**Латушов К.Ж.**	**923**	**41-12-13, 41-28-89**	**1**	**БД**	**5**
5	0112	Дубов Вася	123	11-21-32	1	Физ.	3
6	0221	Иванков П.С.	923	41-12-13, 80012345061	1	Ар.	4
7	**0223**	**Латушов К.Ж.**	**923**	**41-12-13, 41-28-89**	**1**	**ОТУ**	**4**
8	0221	Иванков П.С.	923	41-12-13, 80012345061	1	КМ	3
9	**0223**	**Латушов К.Ж.**	**923**	**41-12-13, 41-28-89**	**1**	**ИО**	**5**
10	0112	Дубов Вася	123	11-21-32	2	Физ.	4
11	0112	Дубов Вася	123	11-21-32	2	ВМ	3

Запрос типа 4. Получить номер рабочего телефона Латушова К.Ж.

```
Select Distinct Тел
 From USPEV02
  Where Сфам = 'Латушов К.Ж.';
```

Результат:

Тел
41-12-13, 41-28-89

И какой же из них рабочий?

Если у пользователя возникают подобные запросы, то с его точки зрения атрибут Тел *композитный многозначный*. Таблица USPEV02 **не является отношением РМД** и не может обрабатываться реляционной СУБД.

Вариант 1.1 (имя таблицы USPEV11). Универсальное отношение. Создано Васей, когда он понял, наконец, что реляционная СУБД не умеет обрабатывать композитные и многозначные атрибуты. Далеко не лучший вариант решения проблемы композитных многозначных атрибутов. Но Вася ещё не знает (а сам не сообразил), как нужно представлять их в структуре БД. Однако USPEV11 уже *отношение РМД*.

№пп	**Сном**	Сфам	Кном	К_Тел	С_Тел	Р_Тел	**Сем**	**Дис**	Оц
1	0221	*Иванков П.С.*	***923***	***41-12-13***	*80012345061*		1	РЯ	5
2	0221	*Иванков П.С.*	***923***	***41-12-13***	*80012345061*		1	Инф.	3
3	0223	Латушов К.Ж.	**923**	**41-12-13**		41-28-89	1	Физ.	3
4	0223	Латушов К.Ж.	**923**	**41-12-13**		41-28-89	1	БД	5
5	0112	Дубов Вася	123	11-21-32			1	Физ.	3
6	0221	*Иванков П.С.*	***923***	***41-12-13***	*80012345061*		1	Ар.	4
7	0223	Латушов К.Ж.	**923**	**41-12-13**		41-28-89	1	ОТУ	4
8	0221	*Иванков П.С.*	***923***	***41-12-13***	*80012345061*		1	КМ	3
9	0223	Латушов К.Ж.	**923**	**41-12-13**		41-28-89	1	ИО	5
10	0112	Дубов Вася	123	11-21-32			2	Физ.	4
11	0112	Дубов Вася	123	11-21-32			2	ВМ	3

Уже при первом взгляде на реализацию этого отношения бросается в глаза многократное дублирование информации. От него невозможно избавиться, просто не заполняя ячейки таблицы, например так:

№пп	**Сном**	Сфам	Кном	К_Тел	С_Тел	Р_Тел	**Сем**	**Дис**	Оц
1	0221	Иванков П.С.	923	41-12-13	80012345061		1	РЯ	5
2	0221						1	Инф.	3
3	0223	Латушов К.Ж.	923			41-28-89	1	Физ.	3
4	0223						1	БД	5
5	0112	Дубов Вася	123	11-21-32			1	Физ.	3
6	0221						1	Ар.	4
7	0223						1	ОТУ	4
8	0221						1	КМ	3
9	0223						1	ИО	5
10	0112						2	Физ.	4
11	0112						2	ВМ	3

Возникают проблемы при выборке данных. По запросу

```
Select Сем, Дис, Оц
  From USPEV11
    Where Сфам = Дубов Вася;
```

получим таблицу

Сем	**Дис**	Оц	Вася имеет ещё две оценки, но куратор их не увидит.
1	Физ.	3	

Приходится писать всё, что знаем, *в каждой* строке!

4.1.4 Проблемы обновления универсального отношения

Схема универсального отношения содержит *все* атрибуты *всех* объектов предметной области.

Аномалия вставки типа а). У куратора появился новый студент Колов Петя, который еще не сдал ни одного экзамена. Он живет вместе с Васей Дубовым. В таблицу нужно вставить строку

12	0222	Колов Петя	123	11-21-32	89012345061				

СУБД *откажется* её принять, т.к. не определены значения компонентов первичного ключа Сем и Дис.

В универсальное отношение нельзя добавить кортеж, содержащий сведения о *части* объектов ПО.

Аномалии вставки типа б, в). Петя получил первую оценку, а куратор — возможность поместить информацию о Пете в свою «базу данных». Вот её новое состояние.

№пп	**Сном**	Сфам	Кном	К_Тел	С_Тел	Р_Тел	**Сем**	**Дис**	Оц
1	0221	Иванков П.С.	923	41-12-13	80012345061		1	РЯ	5
2	0221	Иванков П.С.	923	41-12-13	80012345061		1	Инф.	3
3	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	Физ.	3
4	**0223**	Латушов К.Ж.	923	41-12-13		41-28-89	1	БД	5
5	0112	Дубов Вася	123	11-21-32			1	Физ.	3
6	0221	Иванков П.С.	923	41-12-13	80012345061		1	Ар.	4
7	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ОТУ	4
8	0221	Иванков П.С.	923	41-12-13	**80012345061**		1	КМ	3
9	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ИО	5
10	0112	Дубов Вася	123	11-21-32			2	Физ.	4
11	0112	Дубов Вася	**123**	**11-21-32**			2	ВМ	3
12	**0223**	Колов Петя	**123**	**41-21-32**	**80012345061**		1	Хим.	3

Новая строка 12 нарушает правила 1, 5 и 10, но СУБД примет её и зафиксирует новое состояние.

> В универсальное отношение можно добавить кортеж, нарушающий правила делового регламента.

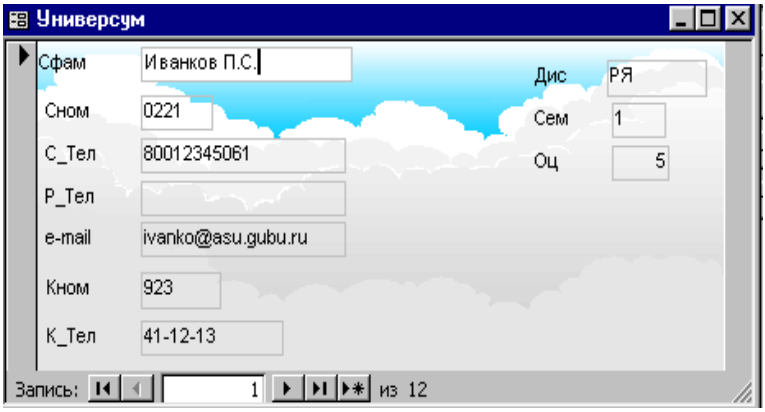
После дня первокурсника куратор нашёл в ДК «Энергетик» затоптанный студбилет, на котором виден только номер. Пытаясь найти владельца, он спросил свою «БД»:

```
Select Distinct Сфам, Кном, К_Тел, С_Тел
  From USPEV11
  Where Сном = '223';
```

И получил ответ:

Сфам	Кном	К_Тел	С_Тел	
Латушов К.Ж.	923	41-12-13		Он позвонил по первому номеру, но там никто не ответил. Позвонил по второму. Ответили: «Морг слушает…Нет, Колов пока не поступал». Позвонил по сотовому. Оказалось, что это телефон Иванкова, но он не терял свой студбилет.
Колов Петя	123	41-21-32	80012345061	

Аномалия обновления типа а). Иванков сообщил куратору, что у него больше нет мобильника (потерял, украли, нечем платить…). Куратор открыл форму обновления своей «БД», нашёл запись Иванкова (первую попавшуюся! См. ниже) и очистил поле С_Тел.



После сохранения зафиксировано следующее состояние

№пп	**Сном**	Сфам	Кном	К_Тел	С_Тел	Р_Тел	**Сем**	**Дис**	Оц
1	**0221**	**Иванков П.С.**	**923**	**41-12-13**			1	РЯ	5
2	**0221**	**Иванков П.С.**	**923**	**41-12-13**	**80012345061**		1	Ин.	3
3	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	Физ.	3
4	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	БД	5
5	0112	Дубов Вася	123	11-21-32			1	Физ.	3
6	**0221**	**Иванков П.С.**	**923**	**41-12-13**	**80012345061**		1	Ар.	4
7	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ОТУ	4
8	**0221**	**Иванков П.С.**	**923**	**41-12-13**	**80012345061**		1	КМ	3
9	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ИО	5
10	0112	Дубов Вася	123	11-21-32			2	Физ.	4
11	0112	Дубов Вася	123	11-21-32			2	ВМ	3
12	0222	Колов Петя	123	41-21-32	89012345061		1	Хим.	3

Так есть у Иванкова мобильник или нет? Во всяком случае, запрос

```
Select Distinct Сфам, С_Тел
  From USPEV11
    Where Сфам ='Иванков П.С.';
```

возвратит следующую таблицу:

Сфам	С_Тел
Иванков П.С.	
Иванков П.С.	80012345061

Куратору следовало бы очистить поле С_Тел **во всех** строках, относящихся к Иванкову.

Изменение значения атрибута A в одном кортеже универсального отношения может привести к необходимости таких же изменений в других кортежах.

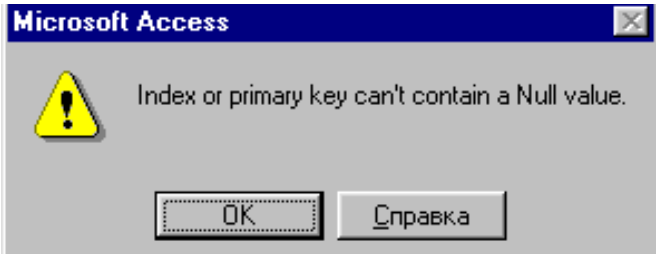
Аномалия обновления типа б). Иванков решил расстаться с Латушовым и переселиться к Дубову и Колову. Куратор добросовестно поменял во всех строках таблицы, относящихся к Иванкову, значение `Кном = 923` на `123`. Но **не изменил** значение `К_Тел` в этих строках. Зафиксировано новое состояние «БД»:

№пп	Сном	Сфам	Кном	К_Тел	С_Тел	Р_Тел	Сем	Дис	Оц
1	0221	Иванков П.С.	**123**	**41-12-13**			1	РЯ	5
2	0221	Иванков П.С.	**123**	**41-12-13**	80012345061		1	Инф.	3
3	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	Физ.	3
4	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	БД	5
5	0112	Дубов Вася	**123**	**11-21-32**			1	Физ.	3
6	0221	Иванков П.С.	**123**	**41-12-13**	80012345061		1	Ар.	4
7	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ОТУ	4
8	0221	Иванков П.С.	**123**	**41-12-13**	80012345061		1	КМ	3
9	0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ИО	5
10	0112	Дубов Вася	**123**	**11-21-32**			2	Физ.	4
11	0112	Дубов Вася	**123**	**11-21-32**			2	ВМ	3
12	0222	Колов Петя	**123**	**41-21-32**	89012345061		1	Хим.	3

Нарушено правило 5. Каков номер телефона в комнате 123?

> Изменение значения атрибута `A` в кортеже универсального отношения может привести к необходимости изменения значений других атрибутов в этом же кортеже.

Аномалия удаления. Студенты — как люди. Всякие бывают. Выяснилось, что Колов надул куратора. Никакую `Хим.` он не сдавал. Надо бы удалить значения трёх последних столбцов в последней строке. Однако любая реляционная СУБД откажется это сделать. Например, ACCESS скажет:



Что делать? Удалить оценку? Значит, Колов сдавал `Хим.`, но получил «никакую» оценку?

Удалить и `Хим.` тоже? Не получится, т.к. `Дис` — компонент первичного ключа.

Заменить значения `Сем` и `Дис` в этой строке «значениями по умолчанию»? Например, `0` и `'никакая'`. Тогда отчёт об успеваемости студентов будет содержать параграф, воспевающий замечательного студента Колова, сдавшего ещё в нулевом семестре никакую дисциплину с неизвестным результатом. Чушь!

Остаётся удалить строку целиком. Но тогда будут потеряны сведения о Колове. Они есть **только в этой** строке.

Из универсального отношения невозможно удалить сведения о части объектов ПО.

Работает ли «БД», состоящая из одного универсального отношения?

С точки зрения СУБД — да. СУБД будет обрабатывать любые таблицы, определенные синтаксически правильными предложениями ЯОД.

С точки зрения пользователя — НЕТ. В этой «БД» не поддерживаются правила делового регламента и допускается накопление противоречивых данных.

Вася тоже это понял и решил выяснить, чем обусловлены проблемы, с которыми он столкнулся, и как их решают более опытные и грамотные проектировщики. То есть Вася не «дуб». А фамилия… Ну, что же, с кем не бывает?

4.2 Бизнес-правила и функциональные зависимости атрибутов

Посмотрим, какие из перечисленных выше правил могут поддерживаться внутренними механизмами СУБД в универсальном отношении.

СУБД может поддерживать правило 12, если определён первичный ключ универсального отношения.

Правило 2 означает, что атрибут `Кном` не может принимать неопределённых значений. Любая современная СУБД имеет средство поддержки подобных ограничений. Нужно только не забыть указать это свойство в определении атрибута.

Правило 4 ограничивает число различных значений атрибута Сном, встречающихся в комбинации с одним и тем же значением Кном. Оно должно поддерживаться хранимой процедурой.

Правила 7 и 11 в специальной поддержке не нуждаются.

Правила 1, 3, 5, 6, 8, 9, 10 очень важны, но **не поддерживаются**. Проектировщик не сообщил о них системе. Как сообщить? Средствами внутренних ограничений целостности РМД при такой «структуре» БД невозможно.

Эти правила определяют свойства *связей атрибутов*, которые называются функциональными зависимостями (ФЗ). ФЗ являются *ограничениями целостности данных*, которые должны поддерживаться нашей системой.

Определение функциональной зависимости. Пусть R отношение, A и B — подмножества его атрибутов. Говорят, что A функционально определяет B или B функционально зависит от A, если и только если для любого значения отношения R два его кортежа, совпадающие по значению A, совпадают и по значению B. Обозначение ФЗ: $A \rightarrow B$.

Атрибут A называется *детерминантом* ФЗ, атрибут B — *зависимой частью*.

Определим ещё четыре важных понятия.

Взаимно однозначная ФЗ. Пусть A и B — подмножества атрибутов отношения. Говорят, что A и B связаны взаимно однозначной ФЗ $A \leftrightarrow B$, если и только если существует пара ФЗ $A \rightarrow B$ и $B \rightarrow A$.

Неприводимая ФЗ. Пусть A и B — подмножества атрибутов отношения. Говорят, что B неприводимо зависит от A, если и только если $A \rightarrow B$ и не существует такого $C \subset A$, что $C \rightarrow B$.

Транзитивная ФЗ. Пусть A, B и C — подмножества атрибутов отношения. Говорят, что C транзитивно зависит от A, если и только если существуют ФЗ $A \rightarrow B$ и $B \rightarrow C$.

Взаимная независимость атрибутов. Пусть A1, A2,..., An атрибуты. Говорят, что эти атрибуты взаимно независимы, если и только если для любого значения i атрибут Ai не зависит функционально ни от какого подмножества остальных атрибутов.

В нашем примере из правил делового регламента следуют ниже перечисленные неприводимые ФЗ:

Правило 1: **Сном → Сфам.**
Правила 1 и 3: **Сном → Ком.**
Правила 9, 10: **Сном ↔ С_Тел.**
Правила 7, 8: **Сном → Р_Тел.**
Правила 5, 6: **К_Тел → Ком.**
Правила 9, 1, 3, 5: **С_Тел → К_Тел.**
Правила 9, 1, 3: **С_Тел → Ком.**
Правила 9, 10, 7, 8: **С_Тел → Р_Тел.**
Правило 12: **{Сном, Дис, Сем} → Оц.**
Правила 12, 9, 10: **{С_Тел, Дис, Сем} → Оц.**

Возможные ключи универсального отношения: {Сном, Дис, Сем}, {С_Тел, Дис, Сем}. Заметим, что каждый потенциальный ключ функционально определяет любой атрибут отношения, не входящий в ключевую группу. Об этих ФЗ можно сообщить СУБД, объявив первичный и альтернативный ключи отношения. В этом случае СУБД будет поддерживать их, контролируя уникальность значений ключевых групп. Сообщить реляционной СУБД о наличии прочих ФЗ невозможно.

Реляционная СУБД имеет единственный механизм, который можно использовать для поддержки функциональных зависимостей атрибутов. Это механизм контроля уникальности значений. Поэтому возложить на СУБД ответственность за поддержание какой-либо ФЗ можно единственным способом. Нужно создать отношение, потенциальным ключом которого является детерминант ФЗ, а неключевым атрибутом — зависимая часть.

Отсюда следует простой вывод. В общем случае множество хранимых атрибутов должно быть структурировано, т.е., представлено в виде схем нескольких отношений так, чтобы все ФЗ, заданные бизнес-правилами, могла поддерживать реляционная СУБД. Для этого отношения должны удовлетворять определённым требованиям. Эти требования ограничивают множество допустимых ФЗ в схеме отношения. Они называются нормальными формами отношений.

4.3 Нормальные формы отношений и проблема аномалий обновления

4.3.1 Первая нормальная форма

Говорят, что отношение находится в 1НФ, если и только если каждый его атрибут определён на домене простого типа данных.

Всякое отношение РМД по определению находится в 1НФ. Таблицы `USPEV01` и `USPEV02` не находятся в 1НФ, т.е. не являются отношениями, *если пользователю нужно обрабатывать список телефонов отдельного студента.* Отношение `USPEV11` находится в 1НФ.

Потенциальным ключом любого отношения является подмножество взаимно независимых атрибутов, которое функционально определяет каждый не входящий в него атрибут (неключевой атрибут). Однако не всякая такая ФЗ является неприводимой. В нашем примере потенциальный ключ `{Сном, Дис, Сем}` функционально определяет любой из прочих атрибутов, но неприводимой является только ФЗ `{Сном, Дис, Сем}` → `Оц`. Все прочие атрибуты функционально зависят от части потенциального ключа — атрибута `Сном`. Конкретное значение этого атрибута может повторяться в кортежах отношения сколько угодно раз. И всякий раз мы должны ему сопоставлять соответствующие значения его зависимых частей. Если мы это не будем делать (оставим «дыры» в ячейках таблицы), то мы нарушим ограничения целостности.

В отношении `USPEV11` есть также ФЗ между неключевыми атрибутами (`Кном` → `К_Тел`). Она создаёт похожие проблемы.

Абстрактный пример. Пусть `R(A, B, C, D, E, F)` отношение, находящееся в 1НФ и `{A, B}` — его потенциальный ключ. В его схеме могут существовать следующие разновидности ФЗ:

`{A, B}` → `C` — неприводимая зависимость от ключа;

`A` → `D` — зависимость от части ключа;

`D` → `E` — зависимость между неключевыми атрибутами;

`F` → `B` — зависимость части ключа от неключевого атрибута.

Здесь A, B, C, D, E, F следует понимать как подмножества атрибутов.

СУБД гарантирует поддержку только тех ФЗ, детерминантом которых является потенциальный ключ отношения.

Отношение, находящееся в 1НФ, нежелательно в структуре базы данных.

4.3.2 Вторая нормальная форма

Рассмотрим проблему зависимости от части ключа. Для того чтобы от неё избавиться, нужно декомпозировать отношение, представив его в виде двух проекций — R1(A, D, E) и R2(A, B, C, F). Ключом R1 является атрибут A, ключом R2 — пара {A, B}. Атрибут A в схеме R2 является внешним ключом. Отношение R может быть восстановлено путём естественного соединения R1 и R2. В этих отношениях уже нет зависимостей от части ключа. Они находятся во второй нормальной форме (2НФ).

Говорят, что отношение находится в 2НФ, если и только если оно находится в 1НФ и не содержит ФЗ от части первичного ключа.

Вернёмся к БД куратора. Декомпозируем отношение USPEV11, представив его в виде двух проекций:

СТУДЕНТ

Сном	Сфам	Кном	К_Тел	С_Тел	Р_Тел
0221	Иванков П.С.	123	**41-12-13**	**80012345061**	
0223	Латушов К.Ж.	923	41-12-13		41-28-89
0112	Дубов Вася	123	**11-21-32**		
0222	Колов Петя	123	**41-21-32**	**80012345061**	

Отношение в 2НФ. Объявлены все ФЗ с детерминантом Сном.
Невозможны аномалии вставки типа а), б), обновления типа а).
Возможны аномалии вставки типа в), обновления типа б), удаления в отношении СТУДЕНТ. Если Латушов уйдёт, потеряем информацию о комнате 923.

УСПЕВАЕМОСТЬ

Сном	Сем	Дис	Оц
0221	1	РЯ	5
0221	1	Инф.	3
0223	1	Физ.	3
0223	1	БД	5
0112	1	Физ.	3
0221	1	Ар.	4
0223	1	ОТУ	4
0221	1	КМ	3
0223	1	ИО	5
0112	2	Физ.	4
0112	2	ВМ	3
0222	1	Хим.	3

Отношение в 2НФ.
Потомок отношения СТУДЕНТ.
Исходное отношение USPEV11 эквивалентно естественному соединению отношений СТУДЕНТ и УСПЕВАЕМОСТЬ.

Проблемы обновления отношения СТУДЕНТ — это следствия существующих, но не объявленных ФЗ с детерминантом С_тел и взаимно однозначной зависимости Кном ↔ К_Тел. Атрибут С_тел является потенциальным ключом отношения СТУДЕНТ, т.к. он функционально определяет все прочие атрибуты. Об этом необходимо сообщить СУБД, объявив его *альтернативным* ключом. На практике это сводится к созданию уникального индекса по С_тел. Никаких преобразований структуры проблемного отношения не требуется.

СТУДЕНТ

Сном	Сфам	Кном	К_Тел	*С_Тел (АК1)*	Р_Тел
0221	Иванков П.С.	123	**41-12-13**	80012345061	
0223	Латушов К.Ж.	923	41-12-13		41-28-89
0112	Дубов Вася	123	**11-21-32**		
0222	Колов Петя	123	**41-21-32**	89012345061	

Отношение в 2НФ. Проблемы обновления, обусловленные наличием ФЗ между неключевыми атрибутами (транзитивной зависимости от ключа), остаются.

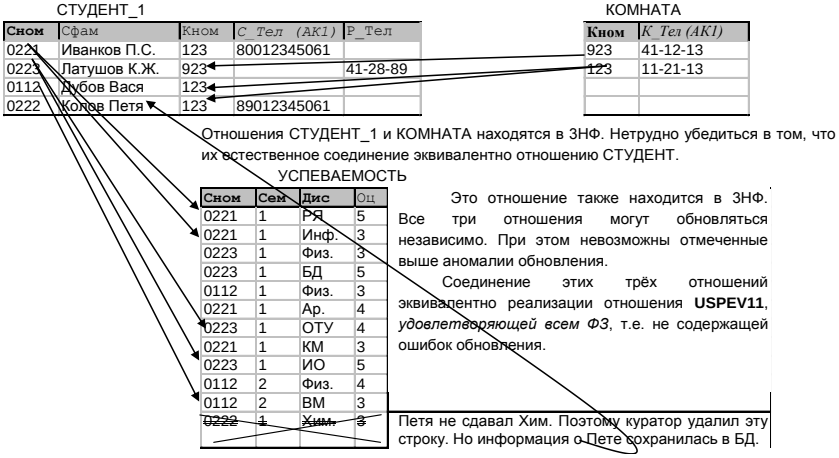
Отношение, находящееся в 2НФ, нежелательно в структуре базы данных.

4.3.3 Третья нормальная форма

Вернёмся к абстрактному примеру (см. стр. 92). Отношение R1(A, D, E) содержит ФЗ A → D и D → E. Представим его в виде двух проекций: R11(A, D) и R12(D, E). Потенциальным ключом проекции R11 является атрибут A, а проекции R12 — атрибут D. Атрибут D в схеме отношения R11 есть внешний

ключ, ссылающийся на R12. Отношение R1 может быть восстановлено путём естественного соединения R11 и R12. В этих отношениях уже нет транзитивных зависимостей. Они находятся в третьей нормальной форме (3НФ).

Определение 3НФ. Говорят, что отношение находится в 3НФ, если и только если оно находится в 2НФ и не содержит транзитивных ФЗ от первичного ключа.



Соединение СТУДЕНТ_1, КОМНАТА, УСПЕВАЕМОСТЬ

Сном	Сфам	Кном	К Тел	С Тел	Р Тел	Сем	Дис	Оц
0221	Иванков П.С.	123	11-21-32			1	РЯ	5
0221	Иванков П.С.	123	11-21-32	80012345061		1	Инф.	3
0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	Физ.	3
0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	БД	5
0112	Дубов Вася	123	11-21-32			1	Физ.	3
0221	Иванков П.С.	123	11-21-32	80012345061		1	Ар.	4
0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ОТУ	4
0221	Иванков П.С.	123	11-21-32	80012345061		1	КМ	3
0223	Латушов К.Ж.	923	41-12-13		41-28-89	1	ИО	5
0112	Дубов Вася	123	11-21-32			2	Физ.	4
0112	Дубов Вася	123	11-21-32			2	ВМ	3
0222	Колов Петя	123	11-21-32	89012345061				

Это т.н. *левое внешнее* соединение. Естественное соединение не содержало бы последней строки. Поскольку соединение — это *производное* отношение, понятие ключа для него не имеет смысла.

4.3.4 Нормальная форма Бойса-Кодда

Разобравшись с тремя нормальными формами, Вася решил, что теперь знает всё, что нужно. Он реструктурировал свою БД.

В течение некоторого времени куратор не имел никаких претензий. Система работала. Но вот ему понадобилось сохранять ещё сведения о том, какой преподаватель принял экзамен у студента. В дополнение к ранее введённым, действуют следующие правила.

13. Преподаватель может принимать экзамены только по одной дисциплине.

14. Различные преподаватели могут принимать экзамены по одной и той же дисциплине.

Замечание. Предметная область системы расширилась. Явление вполне обычное в реальной жизни. Пользователя недолго удовлетворяет хорошая система. Он не торопится от неё отказываться, но стремится расширить область её применения.

Вася смекнул, что нужно где-то сохранять значения нового атрибута Преп. Где? Естественно (подумал Вася), в отношении УСПЕВАЕМОСТЬ. Кроме того, он понял, что правила 13 и 14 задают ФЗ Преп → Дис. Это его насторожило. Он проанализировал ФЗ в расширенной схеме

УСПЕВАЕМОСТЬ_М {Сном, Сем, Дис, Оц, Преп}.

Оказалось, что

{Сном, Сем, Дис} → Оц,

{Сном, Сем, Дис} → Преп, Преп → Дис.

Других неприводимых ФЗ нет. Потенциальный ключ один — {Сном, Сем, Дис}, решил Вася. Расширенное отношение находится в 3НФ. Он реструктурировал отношение и ввёл недостающие данные. Получилось вот что.

УСПЕВАЕМОСТЬ_М

Сном	Сем	Дис	Оц	Преп
0221	1	РЯ	5	Буров
0221	1	Информ.	3	Волков
0223	1	Физ.	3	Сёмин
0223	1	БД	5	Лядов
0112	1	Физ.	3	Жадов
0221	1	Ар.	4	Суткин
0223	1	ОТУ	4	Килин
0221	1	КМ	3	Колчин
0223	1	ИО	5	Рубин
0112	2	Физ.	4	Сёмин
0112	2	ВМ	3	Зубов
0222	1	Хим.	3	Платте

Последний вывод Васи верен, *если первичным ключом объявлена тройка* {Сном, Сем, Дис}. Отношение УСПЕВАЕМОСТЬ_М действительно находится в 3НФ при этом условии. Два первых — не верны.

Из правил 12 и 13 следует ещё одна неприводимая ФЗ {Сном, Сем, Преп} → Оц. Детерминант этой ФЗ — *второй* потенциальный ключ отношения. Если в качестве первичного ключа использовать этот потенциальный, то отношение не будет находиться даже в 2НФ! Возможны аномалии обновления этого отношения, обусловленные необъявленной ФЗ Преп → Дис. Например, последняя строка могла бы выглядеть так:

0222	1	Хим.	3	Зубов	Нарушено правило 14. Что преподаёт Зубов — ВМ или Хим?

Если Колова отчислят и куратор удалит последнюю строку, то будет потеряна информация о том, что Платте преподаёт Хим.

Итак, отношение, находящееся в 3НФ, может оказаться нежелательным в структуре БД.

Дело в том, что требования 3НФ, как и двух предшествующих НФ, не учитывают того обстоятельства, что отношение может иметь несколько потенциальных ключей. Причём различные ключевые группы могут пересекаться.

Этот недостаток устранён в усиленной формулировке требования 3НФ, данной Бойсом и Коддом. Соответствующая нормальная форма отношения называется нормальной формой Бойса—Кодда (НФБК).

Определение НФБК. Отношение находится в НФБК, если и только если каждый его детерминант является возможным ключом.

Отношение УСПЕВАЕМОСТЬ_М не находится в НФБК. Декомпозируем его, выделив детерминант и зависимую часть необъявленной ФЗ в отдельное отношение.

УСПЕВАЕМОСТЬ_М1

Сном	Сем	Преп	Оц
0221	1	Буров	5
0221	1	Волков	3
0223	1	Сёмин	3
0223	1	Лядов	5
0112	1	Жадов	3
0221	1	Суткин	4
0223	1	Килин	4
0221	1	Колчин	3
0223	1	Рубин	5
0112	2	Сёмин	4
0112	2	Зубов	3
0222	1	Платте	3

ПРЕПОДАВАТЕЛЬ

Преп	Дис
Буров	РЯ
Волков	Информ.
Лядов	БД
Жадов	Физ.
Суткин	Ар.
Килин	ОТУ
Колчин	КМ
Рубин	ИО
Сёмин	Физ.
Зубов	ВМ
Плате	Хим.

Отношения УСПЕВАЕМОСТЬ_М1 и ПРЕПОДАВАТЕЛЬ находятся в НФБК.
Теперь правило 13 нарушить невозможно.
Если Колова отчислят, то информация о том, что Платте преподаёт Хим., не потеряется.
Естественное соединение этих отношений есть отношение УСПЕВАЕМОСТЬ_М.

Отношения, находящиеся в НФБК, не имеют аномалий, обусловленных необъявленными ФЗ.

4.3.5 Теорема Хеза

Итог исследований аномалий обновления, обусловленных функциональными зависимостями, подводится теоремой Хеза.

Пусть R(A, B, C) отношение, A, B, C — его атрибуты. Если R удовлетворяет функциональной зависимости $A \rightarrow B$, то оно равно соединению его проекций $R_1(A, B)$ и $R_2(A, C)$.

Под проекцией здесь понимается обычная реляционная проекция, под соединением — *естественное* соединение. Два отношения называются равными, если и только если они имеют эквивалентные схемы и эквивалентные тела.

Поясним суть теоремы Хеза.

Отношение, содержащее хотя бы одну ФЗ, всегда можно представить эквивалентной (содержащей ту же информацию) парой отношений. Схема одного из них R_1(A, B) содержит только детерминант и зависимую часть ФЗ, схема другого R_2(A, C) — детерминант ФЗ и атрибуты исходного отношения, не зависящие от него функционально.

Эта декомпозиция исходного отношения порождает ограничение ссылочной целостности. В первой проекции детерминант ФЗ является *первичным* ключом, во второй — *внешним* ключом.

Исходное отношение R(A, B, C) может быть восстановлено по его проекциям путём естественного соединения. При этом результат соединения будет содержать все кортежи исходного отношения и не будет содержать кортежей, которых не было в исходном отношении.

Наш пример подтверждает (но не доказывает!) эти заключения.

4.3.6 Независимость проекций

Однако при обновлении системы проекций могут возникнуть проблемы, связанные с возможной потерей функциональных зависимостей в результате декомпозиции.

Вернёмся к предыдущему примеру. В исходном отношении УСПЕВАЕМОСТЬ_М имеются следующие ФЗ

1. {Сном, Сем, Дис} → Оц
2. {Сном, Сем, Дис} → Преп
3. {Сном, Сем, Преп} → Оц
4. {Сном, Сем, Преп} → Дис
5. Преп → Дис

В отношениях УСПЕВАЕМОСТЬ_М1 и ПРЕПОДАВАТЕЛЬ существуют (и объявлены) зависимости 3 и 5. Зависимость 4 выводится из 5. Зависимости 1 и 2 *не выводятся* из объявленных. Они утеряны при декомпозиции.

Поэтому отношение УСПЕВАЕМОСТЬ_М1 не может обновляться независимо от отношения ПРЕПОДАВАТЕЛЬ. В самом деле, нарушает правила строка

0223	1	Жадов	4

Жадов преподаёт Физ., а Латушов (№0223) уже получил оценку по Физ. у Сёмина. Однако строка будет принята СУБД.

Для того чтобы не допустить ввода ошибочной строки, нужно выяснить, какую дисциплину преподаёт Жадов (просмотреть кортежи отношения ПРЕП), затем выяснить, какие ещё преподаватели преподают эту же дисциплину, и, наконец, проверить, нет ли у №0223 оценки полученной в этом семестре от одного из этих преподавателей.

Из теории РБД известно, что проекции отношения, находящиеся в НФБК, могут обновляться независимо, если и только если каждая ФЗ, содержащаяся в исходном отношении, является следствием определений ключей проекций.

Известно также, что любое отношение может быть приведено к совокупности отношений, находящихся не ниже, чем в 3НФ, без потерь ФЗ.

4.3.7 Многозначные зависимости и 4НФ

Не следует думать, что этим исчерпывается проблема аномалий обновления. Можно соорудить отношение, находящееся в НФБК, но имеющее аномалии обновления.

Пусть куратору нужно хранить сведения о том, какие дисциплины изучаются его студентами и какие спортивные секции посещают студенты. Каждый студент изучает много дисциплин и может посещать несколько спортивных секций.

Вася Дубов дополнил БД отношением со следующей схемой:

СТУД-ДИС-СЕКЦ

Сном	Дис	Секц

Атрибуты этого отношения *взаимно независимы*. Поэтому единственный возможный (он же первичный ключ) включает *все* его атрибуты. Отношение находится в НФБК. Вася это проверил.

Он начал заполнять новую таблицу БД собственными данными. Он изучает в настоящий момент четыре дисциплины и занимается в двух секциях.

СТУД-ДИС-СЕКЦ

Сном	Дис	Секц
0112	ВМ	Теннис
0112	Физ.	Бокс

Попытки ввести следующие строки не увенчались успехом.

0112	Информ.	
0112	РЯ	

Пришлось писать так:

Сном	Дис	Секц
0112	ВМ	Теннис
0112	Физ.	Бокс
0112	Информ.	Теннис
0112	РЯ	Бокс

Вопрос: Вася изучает ВМ и Информ. как теннисист, а Физ. и РЯ — как боксёр?

Очевидная глупость.

Если куратор захочет узнать, какие боксёры изучают ВМ, то Вася в этот список не попадёт.

Для того чтобы данные были согласованными, таблица должна содержать **все** комбинации значений атрибутов `Дис` и `Секц`, соответствующие одному значению `Сном`. Для Васи их будет восемь.

Сном	Дис	Секц
0112	ВМ	Теннис
0112	ВМ	Бокс
0112	Физ.	Теннис
0112	Физ.	Бокс
0112	Информ.	Теннис
0112	Информ.	Бокс
0112	РЯ	Теннис
0112	РЯ	Бокс

Если Латушов (`Сном = 0223`) изучает пять дисциплин и занимается в трёх секциях, то придётся вставить *пятнадцать* строк со значением `Сном = 0223`. А если он вступит ещё в одну секцию, то нужно будет добавить в таблицу *пять* строк.

Если Вася решит бросить теннис, то из таблицы придётся удалить *четыре* строки.

Суть аномалии: для того чтобы внести в БД одно простое изменение данных, требуется много модификаций таблицы СТУД-ДИС-СЕКЦ.

Причины аномалии. Фиксированному значению атрибута Сном соответствует *фиксированное множество* значений атрибута Дис и *фиксированное множество* значений атрибута Секц. Сами атрибуты Дис и Секц *независимы*.

Определение многозначной зависимости. Пусть A, B и C — произвольные подмножества атрибутов отношения R. Говорят, что A $\rightarrow\rightarrow$ B, если и только если для любой реализации R множество значений B, соответствующее заданной паре значений атрибутов A и C, зависит только от значения A и не зависит от значения C. Многозначная зависимость A $\rightarrow\rightarrow$ B имеет место, если и только если A $\rightarrow\rightarrow$ C.

Для того чтобы устранить аномалии обновления отношения СТУД-ДИС-СЕКЦ следует избавиться от многозначной зависимости. Это можно сделать, представив наши данные в двух отношениях {Сном, Дис} и {Сном, Секц}, являющихся проекциями исходного отношения {Сном, Дис, Секц}.

СТУД-ДИС

Сном	Дис
0112	ВМ
0112	Физ.
0112	Информ.
0112	РЯ

СТУД-СЕКЦ

Сном	Секц
0112	Теннис
0112	Бокс

Эти отношения находятся в четвёртой нормальной форме (4НФ). Их естественное соединение эквивалентно исходному отношению.

Определение 4НФ. Говорят, что отношение находится в 4НФ, если и только если оно находится в НФБК и не содержит многозначных зависимостей.

Из теории нормальных форм известно, что любое отношение может быть представлено в виде эквивалентной совокупности его проекций, находящихся в 4НФ.

Известно также, что отношению, находящемуся в 4НФ, не свойственны аномалии обновления, обусловленные не объявленными функциональными и многозначными зависимостями.

Заметим, что теоретики придумали ещё один вид зависимости, который может существовать в отношении 4НФ и приводить к аномалиям обновления. Сформулированы требования пятой нормальной формы (5НФ), и описан способ построения отношения в 5НФ. Однако мы не будем обсуждать эти проблемы. Васе вряд ли удастся соорудить отношение, не находящееся в 5НФ. Он всё-таки не «дуб», хоть и Дубов. Во всяком случае, мне такой Вася ещё не встретился.

4.4 Нормализация

Подведём итог сказанному на двух предыдущих лекциях и в первой части этой.

1. Использовать универсальное отношение как базу данных можно, но нежелательно.

Дело в том, что средствами ядра РСУБД могут поддерживаться только те ФЗ, детерминантами которых являются возможные ключи. Многозначные зависимости не поддерживаются ни в каком виде, т.к. РМД не допускает многозначных атрибутов.

Для того чтобы обеспечить поддержку прочих зависимостей (функциональных и многозначных), необходимо включить *в приложение* набор специальных процедур, выполняющих соответствующие проверки при обновлении данных. Если проектировщик пренебрежёт этим требованием, то система не гарантирует сохранения целостности данных при обновлении универсального отношения. В таком случае эта обязанность возлагается на пользователя. *Вряд ли это его обрадует!*

2. Вы уже поняли из предшествующего текста, что бóльшую часть проблем поддержки зависимостей атрибутов можно возложить на РСУБД. Для этого нужно декомпозировать «нежелательное» (содержащее «не поддерживаемые» зависимости) отношение, представив его в виде эквивалентной совокупности его проекций. Эта совокупность проекций должна удовлетворять двум требованиям.

— Для любой реализации исходного отношения R естественное соединение проекций должно быть эквивалентно R (декомпозиция без потерь).

— Все межатрибутные зависимости, существующие в исходном отношении, должны быть логическими следствиями определений потенциальных ключей проекций.

Если нарушается первое требование, то естественное соединение проекций содержит кортежи, которых нет в исходном отношении.

Если нарушается второе, то проекции не могут обновляться независимо.

3. То, что мы проделали с исходным универсальным отношением, называется *нормализацией*.

Нормализация — это процедура декомпозиции без потерь, применяемая к «нежелательному» отношению в ходе проектирования базы данных.

Эта процедура состоит в последовательном приведении исходного отношения R к набору мéньших отношений, эквивалентному R, но более предпочтительному. На каждом этапе процедуры создаются проекции отношений, полученных на предыдущем этапе. Для выбора проекций используются заданные ограничения — межатрибутные зависимости.

Перечислю основные правила, которые используются в процедуре нормализации. Иллюстрировать их буду на примере БД куратора.

Итак, нашему куратору необходимо сохранять значения следующих атрибутов:

{Сном, Сфам, Кном, К_Тел, С_Тел, Сем, Дис, Оц, Преп, Секц}.

Здесь и далее исключён из рассмотрения атрибут P_Тел и связанные с ним правила, т.к. они не добавляют в картину ничего существенного.

В БД куратора должны поддерживаться правила делового регламента, перечисленные в двух предыдущих лекциях. Они определяют множество неприводимых ФЗ, существующих в наборе атрибутов.

1. Сном → Сфам,
2. Сном ↔ С_Тел,
3. Сном → Кном,
4. Кном ↔ К_Тел,
5. Преп → Дис,
6. {Сном, Сем, Дис} → Оц,
7. {Сном, Сем, Преп} → Оц,
8. Сном → К_Тел,
9. С_Тел → Кном,
10. С_Тел → К_Тел,
11. {Сном, Сем, Дис} → Преп.

ФЗ 1 ÷ 7 следуют непосредственно из правил делового регламента. ФЗ 8 ÷ 11 следуют из ФЗ 1 ÷ 7.

Атрибут Секц не является частью какого-либо детерминанта неприводимой ФЗ, и сам не зависит функционально ни от какого собственного подмножества атрибутов универсального отношения. С точки зрения куратора этот атрибут *многозначный*. Множество его значений зависит только от значения Сном и не зависит от значений других подмножеств атрибутов. Следовательно, в универсальном отношении существует многозначная зависимость Сном →→ Секц.

Возможными ключами универсального отношения являются следующие четыре группы атрибутов (ключевые группы):

{Сном, Сем, Дис, Секц}; {Сном, Сем, Преп, Секц};
{С_Тел, Сем, Дис, Секц}; {С_Тел, Сем, Преп, Секц}.

Внутри каждой из этих групп существует пара многозначных зависимостей.

Сном →→ {Сем, Дис}|Секц и т.п.

Всякий атрибут, не входящий в некоторую ключевую группу, функционально зависит от неё, но ни одна из этих ФЗ не является неприводимой. Любая из групп может быть назначена первичным ключом универсального отношения. Выберем первую.

Правило 1. Отношение в 1НФ следует разбить на проекции для исключения тех ФЗ от первичного ключа, которые не являются неприводимыми. Результатом будет набор отношений, находящихся, по крайней мере, в 2НФ.

Набор атрибутов, интересующий куратора, можно рассматривать как схему универсального отношения

R(**Сном, Сем, Дис, Секц,** Сфам, Кном, К_Тел, С_Тел, Оц, Преп) **.**

На первом шаге она преобразуется так:

R1(Сном, Сем, Дис, Сфам, Кном, К_Тел, С_Тел, Оц, Преп);
R2(**Сном, Сем, Дис, Секц**) **.**

Естественное соединение этих отношений эквивалентно отношению R (проверьте это сами).

Отношение R2 не содержит ФЗ. Оно находится не только в 2НФ, но и в НФБК, и содержит пару многозначных зависимостей Сном $\rightarrow\rightarrow$ {Сем, Дис}|Секц.

Отношение R1 содержит ФЗ, не являющиеся неприводимыми, т.е. находится в 1НФ. Его следует разбить на две проекции:

R11(**Сном**, Сфам, Кном, К_Тел, С_Тел);
R12(***Сном***, ***Сем***, **Дис**, Оц, *Преп*).

Отношение R11 находится в 2НФ, а R12 — в 3НФ. Их естественное соединение эквивалентно отношению R1. Все ФЗ, которым удовлетворяет R1, сохранены (проверьте это сами).

Заметим, что отношение R12 содержит *два* потенциальных ключа. Если бы в качестве первичного мы выбрали второй, то отношение находилось бы в 1НФ, т.к. содержало бы зависимость от части первичного ключа. Тогда его следовало бы декомпозировать уже на этом этапе.

Правило 2 . Отношения в 2НФ следует разбить на проекции для исключения любых транзитивных зависимостей. Результатом будет набор отношений, находящихся, по крайней мере, в 3НФ.

В нашем примере в 3НФ не находится только отношение R11. Его следует декомпозировать, представив в виде

R111(**Сном**, Сфам, Кном, С_Тел);
R112(**Кном**, К_Тел);

Обе проекции находятся в 4НФ. Декомпозиция выполнена без потерь. Все ФЗ, которым удовлетворяет R11, следуют из определений возможных ключей проекций.

Заметим, что возможна и другая декомпозиция:

R111(**Сном**, Сфам, Кном, С_Тел); S(**Сном**, К_Тел);

однако она нежелательна, т.к. здесь не представлена (утеряна) базовая (следующая непосредственно из правил делового регламента) ФЗ Кном → К_Тел. Поэтому эти проекции нельзя обновлять независимо.

Например, если Вася Дубов переедет в комнату 923, то нужно будет выполнить транзакцию, состоящую из двух операций обновления.

— Изменить значение Кном в кортеже отношения R111, содержащем значение Сном = 0112.

— Заменить значение К_Тел в кортеже S со значением Сном = 0112 на 41-12-13.

Если Вася сообщит куратору, что у него изменился номер телефона, то придётся выяснить, с чем это связано. То ли Вася переехал, то ли в его комнате поменяли номер.

Задание. Рассмотрите все возможные сценарии обновления этих отношений и опишите соответствующие транзакции.

В любом случае эти транзакции должны поддерживать ФЗ, утерянную при декомпозиции. СУБД своими средствами это сделать не может. Значит, нужно обеспечить процедурную поддержку на уровне приложения. Либо отказаться от этой декомпозиции.

Совет. Избавляясь от транзитивных ФЗ, создавайте проекции так, чтобы в них были представлены базовые ФЗ. Тогда транзитивные будут следствиями определений первичных и внешних ключей проекций.

Правило 3. Отношения в 3НФ следует разбить на проекции для исключения любых ФЗ, детерминанты которых не являются потенциальными ключами. Результатом декомпозиции будет набор отношений, находящихся, как минимум, в НФБК.

Из полученных к настоящему моменту отношений в НФБК не находится только

R12(***Сном, Сем,*** **Дис,** Оц, *Преп*)**.**

Оно содержит ФЗ Преп → Дис. Её детерминант не является возможным ключом.

Мы уже обсуждали проблемы декомпозиции

R121(**Сном, Сем, Преп,** Оц)**;**

R122(**Преп,** Дис)**.**

Это декомпозиция без потерь, но здесь утеряна ФЗ

{Сном, Сем, Дис} → Преп.

Поэтому проекции нельзя обновлять независимо.

Другая возможная декомпозиция:

SD(**Сном, Сем, Дис,** Оц);

R122(**Преп,** Дис).

Это ещё хуже. Здесь не только утеряна та же ФЗ, но и соединение проекций не будет эквивалентно отношению R12. Убедитесь в этом сами.

Вывод. Требования декомпозиции без потерь и декомпозиции на независимо обновляемые проекции могут конфликтовать. В некоторых случаях можно выполнить декомпозицию отношения 3НФ, не находящегося в НФБК, без потерь, но полученные проекции будут зависимыми по обновлению.

Правила 1 — 3 можно сконцентрировать в одном.

Правило нормализации до НФБК. Исходное отношение следует разбить на проекции, не содержащие ФЗ, детерминанты которых не являются возможными ключами.

Правило 4. Отношение, находящееся в НФБК, следует разбить на проекции для исключения всех многозначных зависимостей. Результатом будет набор отношений в 4НФ.

В нашем примере в 4НФ не находится только отношение

R2(**Сном, Сем, Дис, Секц**).

Формально его следует разбить на две проекции

R21(**Сном, Сем, Дис**);

R22(**Сном, Секц**).

Отношение R21 является избыточным в структуре БД. Все его кортежи представлены подкортежами естественного соединения отношений R121 и R122.

Окончательно: схема БД куратора содержит отношения

R111(**Сном,** Сфам, Кном, *С_Тел*);

R112(**Кном,** *К_Тел*);

R121(**Сном, Сем, Преп,** Оц);

R122(**Преп,** Дис);

R22(**Сном, Секц**).

Из определений потенциальных ключей этих отношений следуют ФЗ 1 — 10 (см. стр. 105). Они будут поддерживаться

механизмами РСУБД, если объявлены все потенциальные ключи. ФЗ 11 утеряна. Вася должен обеспечить её поддержку на уровне приложения куратора.

Важное замечание. На практике многозначные зависимости следует исключать **до** выполнения нормализации.

Вернёмся к нашему исходному отношению

R(Сном, Сфам, Кном, К_Тел, С_Тел, Сем, Дис, Оц, Преп, Секц).

Как мы уже отметили выше, это отношение имеет четыре возможных ключа. Внутри каждой ключевой группы существует многозначная зависимость с определяющей частью Сном и зависимой частью Секц. Тройка атрибутов ключевой группы, не содержащая Секц, функционально определяет все прочие атрибуты отношения, кроме Секц. Атрибут Секц не зависит функционально от какого-либо подмножества атрибутов. Следовательно, в отношении содержится пара многозначных зависимостей

Сном $\rightarrow\!\!\!\rightarrow$ {Сфам, Кном, К_Тел, С_Тел, Сем, Дис, Оц, Преп}|Секц

Поэтому *уже на первом шаге* нормализации следовало бы выписать две проекции — R1 и R22.

Резюме. Здесь описан *аналитический* подход к проектированию нормализованной структуры РБД.

Исходными данными для проектирования являются

– набор атрибутов, значения которых должны сохраняться в БД;

– набор правил делового регламента, определяющих взаимосвязи атрибутов.

Задача: получить набор схем отношений, в котором

а) представлены все заданные атрибуты,

б) все межатрибутные зависимости, заданные правилами, следуют из определений потенциальных ключей.

Набор атрибутов рассматривается как схема отношения. Исходя из правил и смысла атрибутов, нужно найти ответы на ниже перечисленные вопросы.

- Какие функциональные зависимости существуют в отношении?
- Существуют ли атрибуты, не зависящие функционально от каких-либо подмножеств других атрибутов?
- Какие атрибуты образуют детерминанты неприводимых ФЗ?

- Какие атрибуты образуют потенциальные ключи отношения?
- Существуют ли атрибуты, функционально зависящие от части первичного ключа?
- Существуют ли функциональные зависимости между неключевыми атрибутами?
- Существуют ли детерминанты, не являющиеся потенциальными ключами?
- Содержит ли отношение многозначные зависимости?

На основании ответов на эти вопросы принимаются решения о необходимости декомпозиции и выбираются подходящие проекции.

Критерии выбора проекций:

а) декомпозиция без потерь;

б) возможность независимого обновления проекций.

Первый критерий удовлетворяется, если и только если исходное отношение содержит хотя бы одну ФЗ (теорема Хеза). На практике это требование выполняется для любой нетривиальной задачи.

Практическое правило декомпозиции без потерь: все атрибуты первичного ключа одной из проекций должны входить в состав атрибутов другой проекции.

Второй критерий удовлетворяется, если и только если все ФЗ, существующие в исходном отношении, являются логическими следствиями определений потенциальных ключей его проекций.

Любое отношение может быть нормализовано до 3НФ с сохранением ФЗ.

При нормализации до НФБК второй критерий может оказаться недостижимым. В этом случае следует выбрать одну из альтернатив:

– выполнить нормализацию и обеспечить процедурную поддержку утерянных ФЗ;

– включить в структуру БД ненормализованное отношение и обеспечить процедурную поддержку необъявленных ФЗ.

4.5 Синтез отношений

Существует и другой подход к решению той же задачи в тех же условиях. Его можно назвать *синтетическим*. В рамках этого подхода схемы отношений в 4НФ создаются не как проекции универсального отношения, а как наборы атрибутов, в которые включаются только детерминанты неприводимых ФЗ и их зависимые части. В основе подхода лежат следующие соображения.

Пусть задано множество атрибутов S и набор бизнес-правил, определяющих множество ФЗ на S. Рассмотрим бинарное отношение R(A, B), где A и B — атрибуты из S. Любое бинарное отношение находится в 4НФ. При некоторых условиях схему этого отношения можно расширить, включив в неё атрибут $C \in S$ и сохранив 4НФ. Схему полученного тернарного отношения можно попытаться дополнить ещё одним атрибутом так, чтобы новое тетрарное отношение находилось в 4НФ, и т.д. Вырисовывается идея некоторого итерационного процесса синтеза отношений в 4НФ. Остаётся определить детали. Именно:

– с чего начинать процесс, т.е. какой должна быть начальная схема отношения;

– каковы критерии допустимости расширения схемы на очередном шаге;

– когда следует прекращать процесс;

– какие атрибуты схемы синтезированного отношения следует исключить из рассмотрения в процессе синтеза очередного отношения.

4.5.1 Выбор начальной схемы

Как отмечено выше, любое бинарное отношение находится в 4НФ. Однако не любую пару атрибутов из S можно рассматривать как начальную схему в процессе синтеза. Между атрибутами существуют связи типа 1:1 (взаимно однозначная ФЗ), M:1 (однозначная ФЗ) и M:N (взаимная независимость). Рассмотрим эти варианты.

1. Если $A \leftrightarrow B$, то пара {A, B} *обязательно должна* присутствовать в схеме какого-то отношения 4НФ. Причём оба атрибута являются его потенциальными ключами. Поэтому всякую такую пару нужно считать начальной схемой.

2. Если A → B, то пара {A, B} *может* присутствовать в схеме какого-то отношения 4НФ. Если существует такое отношение, то A (детерминант ФЗ) является его потенциальным ключом. Всякую такую пару можно считать начальной схемой.

3. Если атрибуты A и B взаимно независимы, то они могут входить в схему одного отношения либо как компоненты составного потенциального ключа, либо как неключевые. Поэтому рассматривать такую пару как начальную схему можно, если она является детерминантом какой-либо неприводимой ФЗ.

4. В общем случае пусть X собственное подмножество S и атрибуты, принадлежащие X, взаимно независимы. Пусть ещё в S существует неприводимая ФЗ X → A и A не определяет функционально ни один атрибут из X. Тогда в качестве начальной схемы можно рассматривать подмножество {X, A}.

4.5.2 Расширение схемы

1. Пусть в бинарном отношении R(A, B) существует ФЗ A ↔ B. A и B его потенциальные ключи. В его схему можно добавить любой атрибут C ∈ S, зависящий функционально от A или B. Если в полученном тернарном отношении C является детерминантом, то оно имеет три потенциальных ключа. Это отношение будет находиться в 4НФ.

2. Если бинарное отношение R(A, B) имеет один потенциальный ключ, например A, то в его схему можно добавить атрибут C ∈ S только если A → C. Если существует также и обратная зависимость C → A, то ничего больше не требуется. Если обратной зависимости нет, то атрибуты C и B должны быть взаимно независимы. При этом они могут образовывать потенциальный ключ расширенного отношения. Только при этих условиях расширенное отношение будет находиться в 4НФ.

3. Пусть атрибуты бинарного отношения R(A, B) взаимно независимы. Атрибут C можно добавить в схему, не нарушив требований 4НФ, если существует неприводимая ФЗ {A, B} → C и либо C является потенциальным ключом расширенного отношения, либо ни A, ни B не зависят функционально от C.

4. В общем случае пусть R n-арное отношение, находящееся в 4НФ. Некоторый атрибут $X \in S$ можно добавить в схему R, не нарушая требований 4НФ, если выполняются следующие условия:

А) Атрибут X неприводимо зависит от каждого потенциального ключа R.

Б) Атрибут X не зависит функционально от какого-либо подмножества атрибутов R, не содержащего полного потенциального ключа.

В) В расширенной схеме отношения R не существует детерминанта неприводимой ФЗ, содержащего X и не являющегося потенциальным ключом.

4.5.3 Критерий окончания синтеза отношения

Процесс синтеза одиночного отношения R завершается тогда, когда во множестве S не останется ни одного атрибута, который можно добавить в схему R, не нарушив требования 4НФ.

4.5.4 Сокращение множества атрибутов

По окончании синтеза очередного отношения R часть атрибутов множества S можно исключить из дальнейшего рассмотрения. Какие именно?

Одним из критериев качества нормализованной структуры БД является сохранение всех ФЗ, существующих в S. Пусть синтезировано некоторое отношение R. Оно удовлетворяет всем ФЗ, следующим из определений его потенциальных ключей. Однако атрибуты R, участвующие в этих ФЗ, могут участвовать и в других ФЗ как зависимые части и/или компоненты детерминантов. Часть этих ФЗ может следовать из определений потенциальных ключей отношений, синтезированных ранее. Поэтому решить вопрос о сокращении множества рассматриваемых атрибутов на очередной итерации процедуры синтеза можно только на основании анализа ФЗ, представленных во всей совокупности синтезированных отношений.

Вывод. В процессе синтеза очередного отношения следует исключить из рассмотрения те атрибуты S, которые участвуют в

ФЗ, следующих из определений потенциальных ключей отношений, синтезированных ранее, и не участвуют ни в каких других ФЗ.

4.5.5 Пример синтеза

Рассмотрим снова множество атрибутов, интересующих куратора.

```
S = {Сном, Сфам, Кном, К_Тел, С_Тел, Сем,
     Дис, Оц, Преп, Секц}.
```

В этом множестве существуют неприводимые ФЗ, выписанные выше в п. 4.4.

1. На первом шаге нужно выявить все пары атрибутов, связанных по типу 1:1. В нашем примере их две:

{Сном, С_Тел} и {Кном, К_Тел}.

Обе пары должны рассматриваться как начальные схемы. Выберем первую.

R(Сном, С_Тел).

В схему можно добавить Сфам и один атрибут из пары (Кном, К_Тел). Второй добавлять нельзя, т.к. будет нарушено требование 3НФ. В результате получим два варианта:

R1(Сном, С_Тел, Сфам, Кном);
R2(Сном, С_Тел, Сфам, К_Тел).

Оба отношения находятся в 4НФ и имеют два потенциальных ключа — Сном и С_Тел. Формально любой из них можно назначить первичным ключом. Однако из практических соображений в качестве первичного ключа следует выбрать Сном, так как С_Тел может принимать неопределённые значения (студент может не иметь сотового телефона).

Атрибут Сфам не участвует ни в одной ФЗ, не следующей из Сном $\rightarrow$ Сфам или С_Тел $\rightarrow$ Сфам. Его можно исключить из дальнейшего рассмотрения.

Рассмотрим теперь вторую пару

R3(Кном, К_Тел).

Ни один из атрибутов сокращённого множества S не зависит функционально ни от Кном, ни от К_Тел. Эта схема не может быть расширена. Оба её атрибута не участвуют в каких-либо ФЗ, не представленных отношениями R3 и R1 или R2. Из дальнейшего рассмотрения их можно исключить.

Теперь можно принять решение о том, какой из вариантов первого отношения следует оставить в структуре БД. Формально эти варианты эквивалентны. Значения атрибутов Кном и К_Тел в кортежах этих отношений являются ссылками на кортежи отношения R3. Поэтому выбор варианта зависит от того, какой из этих атрибутов назначен первичным ключом R3. В этом случае ни один из потенциальных ключей не может принимать неопределённых значений. Решение следует принять исходя из смысла отношения R3. Его кортежи описывают комнаты, в которых проживают студенты. Значение атрибута Кном является естественным идентификатором комнаты. Его и следует считать первичным ключом. Тогда в структуре БД следует оставить отношение R1.

Теперь нужно дать синтезированным отношениям осмысленные имена и зафиксировать их схемы с указанием первичных и альтернативных ключей.

СТУДЕНТ(**Сном**, *С_Тел*, Сфам, Кном);
КОМНАТА(**Кном**, *К_Тел*).

2. На этом шаге нас интересуют атрибуты, связанные по типу М:1. На предыдущем шаге нам удалось сократить множество рассматриваемых атрибутов. Оно теперь такое:

S = {Сном, С_Тел, Сем, Дис, Оц, Преп, Секц}.

Рассмотрим пару атрибутов {Дис, Преп} как начальную схему. Единственным потенциальным ключом здесь является атрибут Преп. Ни один из атрибутов множества S не зависит от него функционально. Поэтому начальную схему нельзя расширить.

Получили отношение

ПРЕПОДАВАТЕЛЬ-ДИСЦИПЛИНА(**Преп**, Дис).

Ни один из атрибутов этого отношения нельзя исключить из множества S. Оба участвуют в ФЗ, не следующих из определений потенциальных ключей синтезированных отношений СТУДЕНТ, КОМНАТА и ПРЕПОДАВАТЕЛЬ-ДИСЦИПЛИНА.

Отношения СТУДЕНТ, КОМНАТА и ПРЕПОДАВАТЕЛЬ-ДИСЦИПЛИНА удовлетворяют всем существующим на множестве S ФЗ с простыми детерминантами.

3. Рассмотрим теперь ФЗ с составными детерминантами. Их четыре. Согласно правилам синтеза следует рассмотреть четыре начальных схемы:

R4(**Сном, Сем, Дис,** Оц);
R5(**Сном, Сем, Преп,** Оц);
R6(**С_Тел, Сем, Дис,** Оц);
R7(**С_Тел, Сем, Преп,** Оц).

Ни в одну из них нельзя добавить какой-либо атрибут из S, не нарушив требования 4НФ. На этом очередной этап синтеза закончен.

Каждое из этих отношений содержит сведения об успеваемости студентов. Поэтому в структуру БД следует включить какое-то одно.

Два последних отношения можно не рассматривать, т.к. их внешний ключ С_Тел ссылается на альтернативный, а не на первичный ключ отношения СТУДЕНТ.

Отношение R4 не следует включать в структуру БД, поскольку атрибут Дис ссылается на неключевой атрибут отношения ПРЕПОДАВАТЕЛЬ-ДИСЦИПЛИНА. Если наш куратор захочет узнать, кому Вася Дубов сдал Физ. в первом семестре, то ему придётся обратиться к отношениям R4 и ПРЕПОДАВАТЕЛЬ-ДИСЦИПЛИНА. Нужно будет выполнить их естественное соединение, отфильтровать результат по значению Сном = 0112 и спроецировать отфильтрованное множество строк на атрибут Преп. На выходе куратор с изумлением увидит полный список преподавателей, экзаменующих по Физ.

Приходится включать в БД отношение R5. Присвоим ему осмысленное имя и определим первичный ключ.

УСПЕВАЕМОСТЬ (Сном, Сем, Преп, Оц).

Однако следует иметь в виду, что в таком случае ФЗ {Сном, Сем, Дис} → Оц *не будет* поддерживаться СУБД. Об этом мы уже говорили выше.

4. Теперь все заданные ФЗ, кроме отмеченной, являются следствиями определений потенциальных ключей синтезированных отношений. Все атрибуты, кроме Секц, включены в схемы этих отношений. Куда бы пристроить его?

Ни в одно из существующих отношений нельзя. Будет нарушено требование 4НФ. По смыслу этот атрибут связан с атрибутом Сном. Конкретному значению Сном соответствует вполне определённое множество секций, которые посещает студент-владелец студбилета. В базе данных это можно отобразить единственным способом, создав «полноключевое» отношение

СТУДЕНТ-СЕКЦИЯ (**Сном, Секц**).

Процедура синтеза структуры БД куратора закончена.

4.6 Резюме

Неразумно создавать БД, состоящую из единственного универсального отношения. В такой БД неизбежно избыточное дублирование данных и аномалии обновления. В ней невозможно поддерживать целостность данных.

Проблема аномалий обновления универсального отношения порождается тем, что в нем представлены все ФЗ, обусловленные требованиями ПО, но средствами РМД можно объявить только те из них, в которых детерминантом является возможный ключ отношения.

Для того чтобы в БД не было аномалий обновления данных, следует хранить данные не в одном отношении, а в нескольких взаимосвязанных. Эти отношения должны быть проекциями универсального отношения, находящимися, по крайней мере, в 3НФ. Естественное соединение всех проекций должно восстанавливать универсальное отношение без потерь информации.

Проецирование универсального отношения должно выполняться с сохранением ФЗ. Из существующих (и объявленных!) в проекциях ФЗ должны выводиться все ФЗ, существующие в универсальном отношении.

Любое отношение 1НФ может быть приведено к системе независимо обновляемых отношений 3НФ без потерь информации.

Отношение, не находящееся в НФБК, может быть декомпозировано без потерь информации на два отношения в НФБК, но эти отношения необязательно будут независимо обновляемыми.

Процесс преобразования универсального отношения к системе отношений, не обладающей аномалиями обновления данных, называется *нормализацией*. В качестве методики проектирования логического макета БД нормализация мало пригодна, однако концепция нормализации и связанные с ней понятия ФЗ и нормальных форм определяют цели проектирования и критерии их достижения.

Цель проектирования — создание структуры данных, представляющей собой систему отношений, находящихся в НФБК.

Критерий — каждое отношение в системе должно быть таким, чтобы любой его детерминант был потенциальным ключом.

В заключение отметим, что в некоторых случаях нормализация до НФБК может оказаться недостаточной для устранения аномалий обновления. Тогда следует продолжить процесс до получения 4НФ или 5НФ. Однако при разумном использовании интуитивного (семантического) подхода к проектированию, изложенного в следующем разделе, такие ситуации возникают исключительно редко.

Контрольные вопросы

1. Что понимают под «универсальным» отношением?
2. Опишите проблемы обновления универсального отношения (аномалии обновления).
3. Дайте определение функциональной зависимости атрибутов отношения.
4. Что является основанием для утверждения о существовании функциональной зависимости атрибутов?
5. Сформулируйте определение взаимной независимости атрибутов отношения.
6. Что называется транзитивной функциональной зависимостью?
7. Что называется неприводимой функциональной зависимостью?
8. Чем обусловлены аномалии обновления универсального отношения?
9. Охарактеризуйте функциональную зависимость атрибутов как ограничение целостности данных.

10. Какие механизмы реляционной модели данных используются для объявления функциональных зависимостей в реляционной БД?

11. Сформулируйте определения первой, второй и третьей нормальных форм отношений.

12. Сформулируйте теорему Хеза и поясните её смысл.

13. Сформулируйте принцип декомпозиции отношения без потерь информации.

14. Сформулируйте практическое правило нормализации отношения до третьей нормальной формы и опишите процедуру нормализации.

15. Опишите возможные аномалии обновления отношения, находящегося в третьей нормальной форме. Объясните причины этих аномалий.

16. Сформулируйте определение нормальной формы Бойса—Кодда.

17. Какие проблемы могут возникнуть при нормализации до нормальной формы Бойса—Кодда?

18. Опишите возможные аномалии обновления отношения, находящегося в нормальной форме Бойса-Кодда. Объясните причины этих аномалий.

19. Сформулируйте определение многозначной зависимости атрибутов отношения.

20. Сформулируйте определение четвёртой нормальной формы.

21. Сформулируйте теорему Фейджина и поясните её смысл.

22. Опишите преимущества и недостатки нормализованной реляционной базы данных в различных операциях манипулирования данными.

23. Охарактеризуйте недостатки нормализации как методики проектирования реляционной базы данных.

5 РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ IDEF1X

В настоящей главе изложены основы методологии информационного моделирования IDEF1X (Integrated DEFinitions 1 eXpanded), имеющей в настоящее время статус федерального стандарта США.

5.1 Компоненты языка IDEF1X

5.1.1 Модель данных

Стандарт IDEF1X дает следующее определение модели данных:

> Модель данных — графическое и текстуальное представление, идентифицирующее потребности организации в данных для достижения ее целей, выполнения функций и определения стратегий управления. Модель данных идентифицирует сущности, домены, атрибуты и связи между данными и поддерживает единый концептуальный взгляд на данные и их взаимосвязи.

В соответствии с этим определением IDEF1X как язык описания данных содержит два важнейших компонента:

– *графический* — средства создания диаграмм, показывающих структуру и взаимосвязи данных;

– *текстовый* — правила создания и ведения текстовых документов, уточняющих и поясняющих графическую модель, — глоссариев, спецификаций, отчетов, заметок и т.п.

Мы здесь сосредоточимся, главным образом, на графическом компоненте.

5.1.2 Уровни представления диаграмм

В IDEF1X различают три уровня графического представления информации, или три уровня диаграмм.

Уровень «сущность-связь» (ER level). Это уровень наименее детального представления информации. Он используется на начальной стадии моделирования, когда еще не выяснены или не поняты до конца свойства сущностей и связей. На диаграм-

мах ER-уровня сущности и связи представлены только их именами.

Уровень ключей (Key-Based Level, KB). На этом уровне в диаграммах отражаются имена первичных и внешних ключей сущностей и спецификации связей. Диаграмма KB-уровня объявляет уникальные идентификаторы экземпляров сущностей и ограничения ссылочной целостности.

Уровень атрибутов (Fully Attributed Level, FA). Диаграмма FA-уровня показывает имена всех атрибутов сущностей и связей и полностью определяет структуру и взаимосвязи данных.

Цель моделирования — создание FA-диаграммы. Она является графическим представлением структуры реляционной базы данных с полностью определенными схемами отношений.

Синтаксис графического языка IDEF1X обеспечивает однозначное представление ограничений ссылочной целостности в диаграммах KB- и FA-уровня. Правила именования и определения сущностей, доменов и атрибутов дают возможность задать ограничения на значения в текстовых документах, сопровождающих диаграмму. В силу этого трансляция FA-диаграммы в тексты описания таблиц и триггеров ссылочной целостности на языке конкретной СУБД оказывается чисто формальной процедурой и может выполняться автоматически.

Рассмотрим теперь нотации и правила языка IDEF1X.

5.2 Сущности, атрибуты, домены

5.2.1 Обозначения

Под сущностью в IDEF1X понимается *отношение РМД*. Сущности изображаются на диаграммах именованными прямоугольниками, в которые вписываются имена атрибутов. Используемые обозначения определены на рис. 5.1.

На ER-уровне независимые и зависимые сущности не различаются, атрибуты не указываются, а имена сущностей вписываются в обозначающие их прямоугольники.

Рис. 5.1 — Обозначения сущностей и атрибутов

Специального синтаксиса доменов в графическом языке не предусмотрено. Домены определяются независимо от сущностей и диаграмм на основании требований ПО.

5.2.2 Правила именования и определения

Здесь перечислены только основные правила стандарта IDEF1X.

- Сущности, атрибуты и домены обязательно именуются. Именем может быть только имя существительное, возможно с определениями. В качестве имен допускаются аббревиатуры и акронимы.
- Имя должно быть уникальным и осмысленным. Однозначное описание смысла имени обязательно включается в глоссарий модели.

Создавая модель, проектировщик стремится сформировать ясное представление о ПО, в частности о том, какие сведения будут храниться в БД. Этого можно добиться, только сформулировав точные и однозначные определения смысла каждого имени, введенного в модель.

Например, что такое ДЕТАЛЬ? Это неделимая часть изделия или она сама может собираться из других деталей? Может ли ИЗДЕЛИЕ быть ДЕТАЛЬЮ? Наша фирма только закупает ДЕТАЛИ или может их производить?

Или что такое ФИЛЬМ? Это лента, лежащая в нашей фильмотеке, или это произведение киноискусства? Нас интересуют любые фильмы или только фильмы определенного жанра?

От ответов на подобные вопросы зависят организация данных и их взаимосвязи. Поэтому, прежде чем строить диаграммы, необходимо получить точные ответы на все подобные вопросы и зафиксировать документально определения смысла имен. Без этого модель будет неоднозначной и противоречивой.

- Имя сущности, атрибута или домена должно иметь единственный смысл, и этот смысл всегда должен выражаться этим именем. Тот же смысл не может вкладываться в другое имя, если оно не является псевдонимом или синонимом основного.

Например, атрибут Дата не может иметь смысл даты *начала ИЛИ окончания* отчетного периода. Совершенно непонятно, как интерпретировать значения этого атрибута в различных кортежах отношения.

- Сущности и атрибуты всегда именуются в единственном числе. Имя должно относиться к одному экземпляру сущности или значению атрибута.

Соблюдение этих правил обеспечивает интерпретацию диаграмм фразами естественного языка и точную передачу смысла, вложенного в имена автором модели.

5.2.3 Правила для атрибутов

Согласно стандарту атрибут есть свойство или характеристика, общая для некоторых или всех экземпляров сущности. Атрибут является конкретизацией домена в контексте сущности.

- На диаграммах KB- и FA-уровней каждая сущность имеет не менее одного атрибута. Каждый атрибут может быть собственным атрибутом сущности или присоединенным (мигрировавшим), полученным от другой сущности через связь.
- Каждый атрибут является собственным атрибутом точно одной сущности.
- Присоединенный атрибут должен быть частью первичного ключа передавшей его родительской (или родовой) сущности. Присоединенный атрибут помечается символом «(FK)», следующим за именем атрибута.

В совокупности эти требования означают, что *никакие две сущности не могут иметь одноименных атрибутов, если они не связаны каким-либо отношением*. В последнем случае одноименными могут быть атрибуты, которые являются частью первичного ключа родителя *и* частью внешнего ключа.

- Каждый экземпляр сущности должен иметь определенное (не `NULL`) значение каждого атрибута, являющегося частью первичного ключа.
- Не может быть экземпляра сущности, имеющего более чем одно значение какого-либо из атрибутов.
- Атрибуты, не являющиеся частью первичного ключа, могут иметь неопределенные значения. Для ясности такие атрибуты помечаются символом «`(O)`», следующим за именем атрибута (Optional — необязательный).
- Если атрибут является собственным атрибутом одной сущности и присоединенным атрибутом другой, то либо он имеет одинаковые имена в обеих сущностях, либо помечается именем роли как присоединенный.
- Определение атрибута должно содержать ссылку на имя домена.
- На KB-диаграммах показываются только атрибуты, входящие в состав первичных, альтернативных и внешних ключей. Атрибуты альтернативных ключей обязательно помечаются символом «`(AKn)`», где `n` — номер альтернативного ключа. Все атрибуты, входящие в состав одного и того же альтернативного ключа, помечаются одним и тем же значением `n`. На FA-диаграмме показываются все атрибуты каждой сущности.

5.3 Соединения

5.3.1 Типы соединений

Соединение — это один из двух видов связей, используемых в языке IDEF1X. Стандарт определяет соединение как ассоциацию между двумя сущностями или между экземплярами одной и той же сущности. Понятие соединения в IDEF1X совпадает с понятием бинарной связи в ER-модели. Никаких обозна-

чений для представления связей высшей арности язык IDEF1X не содержит. Это не является ограничением модели, т.к. любую n-арную связь можно представить в виде эквивалентной совокупности бинарных связей (см. п. 2.6).

Неспецифическое соединение — это бинарная связь типа M:N В такой связи нет родителя и потомка. Неспецифические соединения могут быть показаны ***только*** на диаграммах ER-уровня. На KB- и FA-диаграммах они должны быть представлены эквивалентными парами специфических соединений.

Специфическое соединение — это бинарная связь типа 1:M. В такой связи всегда можно указать родителя и потомка. Родителем называется многосвязная сущность. Мощность специфического соединения со стороны потомка 0 или 1.

В состав атрибутов потомка в специфическом соединении обязательно входят все атрибуты первичного ключа родителя. Как атрибуты потомка они называются *внешним ключом*.

Специфическое соединение называется ***идентифицирующим***, если соответствующий ему внешний ключ полностью входит в первичный ключ потомка. Это означает, что идентификация экземпляров потомка в идентифицирующем соединении возможна только через ссылку на соответствующий экземпляр родителя.

Соединение называется ***неидентифицирующим***, если соответствующий ему внешний ключ не входит в состав первичного ключа потомка или входит в него частично.

5.3.2 Синтаксис соединений

Нотации, используемые для обозначения соединений, определены на рис. 5.2.

Значение кардинальности связи от родителя указывается около точки на конце дуги. Возможные спецификации кардинальности приведены в таблице 5.1.

Кардинальность со стороны потомка необходимо указывать только для необязательного неидентифицирующего соединения. Для прочих типов соединений она всегда точно 1. Ромб на конце дуги, примыкающем к родительской сущности (см. рис. 5.2, *б*)),

означает, что с каждым экземпляром потомка в связь вступает 0 или 1 экземпляр родителя.

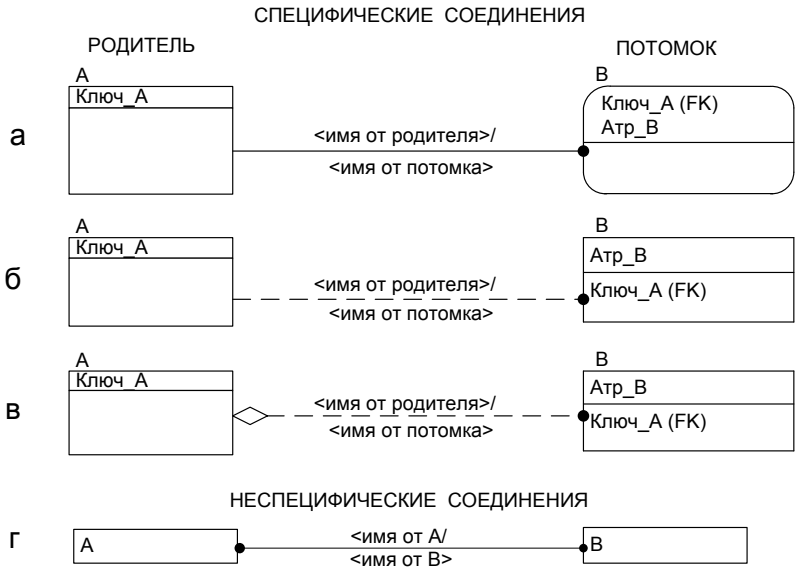


Рис. 5.2 — Обозначения соединений:
а — идентифицирующее соединение;
б — обязательное неидентифицирующее соединение;
в — необязательное неидентифицирующее соединение;
г — неспецифическое соединение

Таблица 5.1 — Спецификации кардинальности

Обозначение	Число возможных экземпляров связи
	0, 1 или более
p	1 или более
z	0 или 1
<N>	точно указанное число N
<N1> - <N2>	от N1 до N2

5.3.3 Правила именования соединений

Ниже перечислены только основные правила стандарта.

- Соединению присваивается имя, выражаемое глагольным оборотом. Имя зрительно привязывается к дуге, изображающей соединение. Имена соединений могут быть неуникальными в пределах диаграммы.
- Имя каждого соединения одной и той же пары сущностей должно быть уникальным во множестве имен связей этих сущностей.
- Имена специфических соединений выбираются так, чтобы можно было построить осмысленную фразу, составленную из имени родительской сущности, имени связи, выражения кардинальности и имени потомка (рис. 5.3).
- Связь может быть поименована «от родителя» и от «потомка» (см. рис. 5.2, *а — в*). Имя «от родителя» обязательно.
- Если связь не именуется со стороны потомка, то имя «от родителя» должно выбираться так, чтобы связь легко читалась и со стороны потомка.
- Неспецифические соединения обязательно именуются с обеих сторон.

Рис. 5.3 — Именование связей

5.4 Связи категоризации

Связью категоризации (**categorization relationship**) называется связь между родовой сущностью и категорией.

5.4.1 Синтаксис категорий

Обозначения для связей категоризации показаны на рис. 5.4, рис. 5.5.

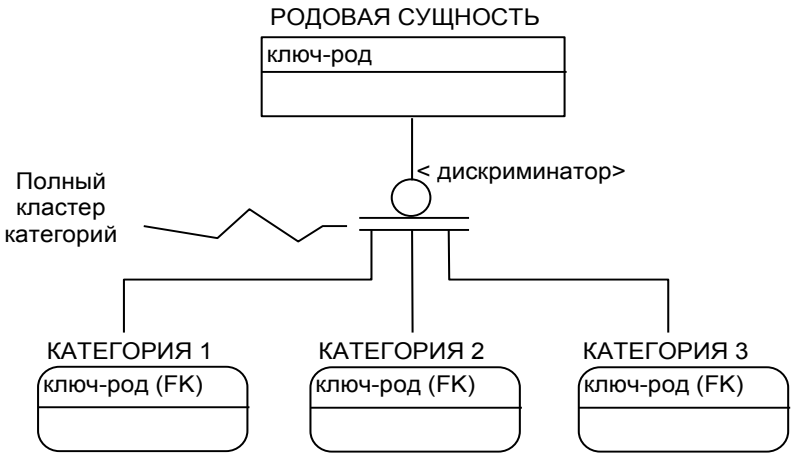


Рис 5.4 — Полный кластер категорий

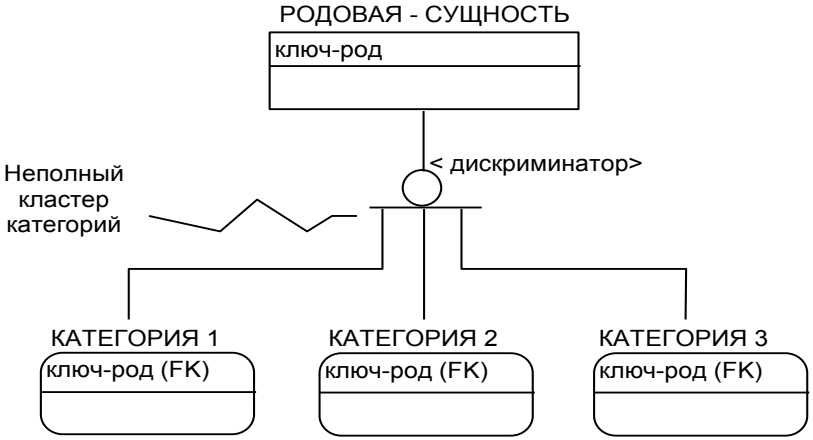


Рис 5.5 — Неполный кластер категорий

Каждая пара линий — от родовой сущности до окружности и от окружности до категории — представляет *одну* связь в кластере. Спецификации кардинальности не указываются, поскольку она всегда «ноль или один» со стороны родовой сущности и «точно один» со стороны категории. Это означает, что категории одного и того же кластера не пересекаются. Категории различных кластеров могут иметь общие экземпляры. Связи категоризации не имеют явных имен. От родовой сущности любую из них можно прочитать «может быть», а обратно — «является».

5.4.2 Правила для связей категоризации

- Категория может иметь только одну родовую сущность. Она может принадлежать только одному кластеру категорий.
- Категория в одной связи категоризации может быть родовой сущностью в другой связи категоризации.
- Сущность может быть родовой для любого числа кластеров категорий.
- Все экземпляры категории имеют одно и то же значение дискриминатора, и все экземпляры различных категорий должны иметь различные значения дискриминатора.
- Не существует двух кластеров категорий одной и той же родовой сущности, имеющих один и тот же дискриминатор.
- Дискриминатором полного кластера категорий не может быть атрибут, значения которого могут быть неопределенными.
- Первичный ключ любой категории должен совпадать с первичным ключом родовой сущности. Однако в качестве имен ключевых атрибутов категорий могут использоваться имена ролей.

5.5 Имя роли

Именование ролей — это механизм разрешения *конфликтов имен*. Они возникают, например, в унарных связях. Нередки также ситуации, когда один и тот же атрибут передается сущности через посредство нескольких различных связей.

Пример. БД должна хранить сведения о некоторых `ИЗДЕЛИях` и об `УЗЛах`, входящих в их состав. При этом сами `УЗЛы`

могут состоять из других узлов, т.е. быть составными ИЗДЕЛИЯМИ. Нужно показать, в состав каких ИЗДЕЛИЙ какие УЗЛы входят. Один и тот же УЗЕЛ может входить в состав различных ИЗДЕЛИЙ. На ER- уровне это может выглядеть так, как на рис 5.6:

Рис. 5.6 — Унарная связь

Однако на КВ-уровне придется преобразовать это неспецифическое соединение к виду, показанному на рис. 5.7.

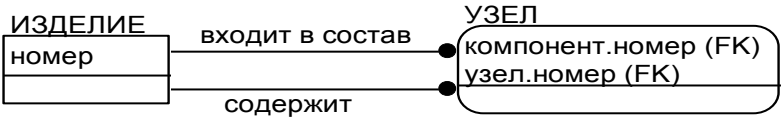


Рис. 5.6 — Имена ролей

Каждая связь передает атрибут номер сущности ИЗДЕЛИЕ в состав атрибутов сущности УЗЕЛ в качестве внешнего ключа. Внешние ключи, переданные различными связями, имеют различный смысл. В каждом экземпляре потомка значения передаваемого по разным связям атрибута номер будут различными. Поэтому он должен войти в схему потомка *дважды*, но с *различными* именами. Показанные на рисунке 5.6 префиксы имени номер называются *именами ролей*. Они явно именуют различные роли сущности ИЗДЕЛИЕ в разных связях с потомком.

Имя роли это — новое имя компонента внешнего ключа, явно указывающее на роль, исполняемую сущностью в связи.

В ситуациях, подобных рассмотренной, имена ролей необходимы для разрешения конфликта. Однако на их использование нет никаких ограничений. Имя роли может быть назначено всякий раз, когда проектировщик модели считает это нужным, например, для точной передачи смысла связи, улучшения читаемости диаграммы и т.п. На имя роли можно ссылаться в диаграмме как на атрибут.

5.6 Требования к диаграммам

5.6.1 ER-уровень

Диаграмма должна содержать сущности и связи, может показывать атрибуты и не должна показывать первичные, альтернативные или внешние ключи. На ER-уровне сущности *не различаются* как зависимые или независимые, а соединения — как идентифицирующие и неидентифицирующие. Сущности *не содержат* горизонтальных линий, отделяющих область ключей от области данных. Имена сущностей вписываются в обозначающие их прямоугольники.

Диаграмма ER-уровня может показывать категории, но указывать дискриминаторы кластеров *не обязательно*. На ER-уровне допустимы *неспецифические соединения*. Для изображения соединений можно использовать как сплошные, так и штриховые линии. *Это не специфицирует соединения*.

5.6.2 KB-уровень

Диаграммы этого уровня изображают сущности, связи, первичные, альтернативные и внешние ключи.

- Различаются зависимые и независимые сущности, а также идентифицирующие/неидентифицирующие и обязательные/необязательные соединения.
- Неспецифические соединения запрещены.
- Каждая сущность должна иметь первичный ключ и альтернативные ключи, если они существуют.
- Каждая сущность должна содержать внешний ключ для каждого соединения или связи категоризации, в которой она участвует как потомок или категория.
- Каждый кластер категорий должен иметь дискриминатор.

Диаграмма KB-уровня может также содержать неключевые атрибуты.

5.6.3 Правила для первичных и альтернативных ключей

Диаграммы КВ-уровня должны удовлетворять ряду правил для ключей.

- На диаграммах КВ- и FA-уровней каждая сущность должна иметь первичный ключ.
- Сущность может иметь несколько альтернативных ключей.
- Как первичный, так и альтернативный ключ может быть либо одиночным атрибутом, либо группой атрибутов.
- Отдельный атрибут может быть частью более чем одного ключа, первичного или альтернативного.
- Атрибуты, входящие в первичный и альтернативный ключи, могут быть как собственными атрибутами сущности, так и присоединенными через связь с другой сущностью.
- Первичный и альтернативные ключи должны содержать только те атрибуты, которые необходимы для уникальной идентификации экземпляров сущности (*неизбыточность*).
- Каждый неключевой атрибут должен неприводимо зависеть от первичного ключа, если он составной (*требование 2НФ*).
- Каждый атрибут, не являющийся частью первичного ключа или какого-либо из альтернативных, должен функционально зависеть только от первичного ключа и каждого из альтернативных (*требование НФБК*).

5.6.4 Правила для внешних ключей

- Каждая сущность, являющаяся потомком в специфической связи или категорией в связи категоризации, должна содержать множество атрибутов первичного ключа родителя — внешний ключ, реализующий связь. Конкретный атрибут может быть элементом нескольких таких множеств. Число атрибутов во внешнем ключе должно совпадать с числом атрибутов первичного ключа родительской или родовой сущности.
- Первичный ключ родовой сущности должен передаваться как первичный ключ каждой категории.
- Каждый присоединенный атрибут потомка или категории должен быть атрибутом первичного ключа связанной с ним

родительской или родовой сущности. Обратно, каждый атрибут первичного ключа родительской или родовой сущности должен быть присоединенным атрибутом связанного с нею потомка или категории.

- Каждое имя роли, назначенное присоединенному атрибуту, должно быть уникальным, и в одно и то же имя всегда должен вкладываться один и тот же смысл. Один и тот же смысл не может вкладываться в разные имена, если они не являются псевдонимами.
- Присоединенный атрибут может быть частью более чем одного внешнего ключа при условии, что он имеет одно и то же значение в этих множествах в некотором фиксированном экземпляре сущности. Такому присоединенному атрибуту может быть назначено имя роли.
- Каждый атрибут внешнего ключа должен ссылаться на один и только один атрибут первичного ключа родителя.

5.6.5 FA-уровень

Диаграмма FA-уровня должна содержать все, что содержит диаграмма КВ-уровня и, кроме того, все неключевые атрибуты. На КВ- и FA-уровнях в полной мере действуют все правила синтаксиса, изложенные выше.

Все сущности на FA-диаграмме должны находиться, по крайней мере, в 3НФ.

5.7 Дополнения к модели

Диаграммы модели сопровождаются дополнительными материалами, уточняющими и поясняющими ее смысл. Имеется два вида дополнений: *глоссарий* и *примечания*.

Глоссарий является обязательным дополнением. Он содержит описания отдельных диаграмм модели и определения сущностей, атрибутов и доменов.

Обязательные компоненты глоссария:

– имена — уникальные, осмысленные и соответствующие природе ПО наименования сущностей, атрибутов и доменов;

– определения — краткие, точные, однозначные тексты, обеспечивающие правильное понимание смысла имен, одинаковое для любого читателя.

Определение должно быть одинаково применимо в любом контексте, в котором встречается имя.

Кроме того, глоссарий может содержать *список псевдонимов*. Каждому имени могут соответствовать в различных контекстах псевдонимы. Необходимость их использования обусловлена тем, что в различных частях моделируемой ПО может использоваться разная терминология. Поэтому одни и те же вещи могут называться разными именами. Полный список псевдонимов должен сопровождать каждое имя.

Примечания — это необязательный вид дополнений. Они могут пояснять смысл диаграмм и их элементов, содержать описания путей связей, сообщать о функциях, связанных с тем или иным объектом, и т.п.

Примечания нумеруются числами натурального ряда. На диаграмме ссылка на примечание помещается около соответствующего объекта в круглых скобках.

5.8 Лексические соглашения

Для того чтобы обеспечить наглядность и читаемость диаграмм, стандарт IDEF1X рекомендует придерживаться ряда соглашений относительно построения имен и фраз на диаграммах.

5.8.1 Имена

В именах сущностей и атрибутов используются только буквы, цифры, а также знаки-разделители: «дефис», «подчерк» и «пробел». Имя должно начинаться с буквы. Части составного имени отделяются дефисом, подчерком или пробелом. Эти разделители не различаются.

Рекомендуется имена сущностей на диаграммах и в текстах глоссариев и замечаний всегда писать ПРОПИСНЫМИ буквами.

Например, глоссарий может содержать такое определение имени `ПОСТАВЩИК`: «Юридическое лицо, заключившее с нашей

фирмой договор на поставку одного или нескольких видов ДЕТАЛей для одного или нескольких видов ИЗДЕЛИЙ».

Это определение содержит сведения не только о смысле, но и о связях определяемой сущности с другими сущностями ПО. Выделение имен сущностей в тексте прописными буквами облегчает зрительное восприятие фактов связи.

Помимо вышеперечисленных разделителей в именах используются еще символы «точка» и «слэш». Точка отделяет имя роли от основного имени атрибута. Имя роли предшествует точке, основное имя следует за точкой. Ни перед, ни после точки не допускаются другие разделители. Слэш разделяет разнонаправленные имена связи. Кроме того, он разделяет имя сущности или связи и его *идентификатор*.

5.8.2 Идентификаторы

Стандарт рекомендует назначать именам сущностей и связей уникальные *идентификаторы*. За идентификатор обычно принимают *порядковый номер* сущности или связи в порядке появления объекта на диаграмме. Обычно присваиваются идентификаторы вида «E<число>» сущностям и «R<число>» — связям.

5.8.3 Метки атрибутов

Метки атрибутов размещаются вслед за именем атрибута в круглых скобках. Допускаются следующие метки:

(O) — атрибут может принимать NULL-значения;

(FK) — компонент внешнего ключа;

(AK<N>) — компонент N-го альтернативного ключа;

(<число>) — ссылка на примечание.

Контрольные вопросы

1. Дайте общую характеристику семантических методологий анализа и проектирования концептуальной модели.
2. Сформулируйте определение модели данных IDEF1X.
3. Перечислите обязательные компоненты модели требований пользователя в методологии IDEF1X?

4. Опишите содержание этапов проектирования методологии IDEF1X.

5. Перечислите определенные стандартом IDEF1X уровни модели.

6. Сформулируйте требования к уровням IDEF1X-модели.

7. Опишите сходства и различия понятия сущности в ER-модели и модели IDEF1X.

8. Какие типы сущностей IDEF1X Вы знаете?

9. Как изображаются сущности различных типов на IDEF1X-диаграммах ER-, KB- и FA-уровней?

10. Опишите сходства и различия понятия атрибута в ER-модели и модели IDEF1X.

11. Каковы правила именования сущностей и атрибутов?

12. Что такое первичный и альтернативный ключи сущности? Как они изображаются на IDEF1X-диаграммах?

13. Сформулируйте правила первичных и альтернативных ключей.

14. Укажите правила IDEF1X, обеспечивающие построение нормализованных систем отношений.

15. Какие типы связей Вы знаете?

16. Опишите сходства и различия понятия связи в ER-модели и модели IDEF1X.

17. Что такое неспецифическое соединение?

18. На каком уровне детальности модели допустимы неспецифические соединения?

19. Что такое специфическое соединение?

20. Что называется внешним ключом сущности?

21. Что такое идентифицирующее специфическое соединение?

22. Что такое неидентифицирующее специфическое соединение?

23. Что является основанием для принятия решения о типе специфического соединения?

24. Что такое необязательное неидентифицирующее специфическое соединение?

25. Может ли идентифицирующее соединение быть необязательным?

26. Как изображаются соединения различных типов на IDEF1X-диаграммах ER-, KB- и FA-уровней?

27. Каковы правила именования соединений?

28. Как отображаются кардинальности соединений на IDEF1X-диаграммах?

29. Как отображается необязательность соединения на IDEF1X-диаграммах?

30. Что называется связью категоризации, категорией, кластером категорий, дискриминатором кластера?

31. Как изображаются связи категоризации на IDEF1X-диаграммах?

32. Сформулируйте правила IDEF1X для внешних ключей.

33. Как зависит размещение внешнего ключа среди атрибутов сущности от типа соединения?

34. Что такое имя роли? В каких случаях необходимо явное именование ролей?

35. Какие метки атрибутов допустимы на IDEF1X-диаграммах?

6 МЕТОДОЛОГИЯ ПРОЕКТИРОВАНИЯ БАЗ ДАННЫХ

6.1 Понятие методологии проектирования

Методология проектирования — это структурированный подход к организации процесса проектирования, предусматривающий использование специализированных процедур, технических приёмов, инструментов и документации.

Любая методология нацелена на стандартизацию процесса проектирования. Она предусматривает разбиение всего процесса на несколько фаз. Каждая фаза состоит из нескольких этапов. На каждом этапе решается определённый набор задач. Для этого используются предлагаемые методологией технические приёмы.

Кроме того, методология включает:

- методы планирования разработки,
- методы управления проектом,
- методы оценки хода проекта,
- единый подход к анализу и моделированию всего набора требований к проектируемой системе.

Всё это в совокупности повышает управляемость проекта и вероятность его успешного завершения.

Здесь мы рассмотрим одну из нескольких существующих методологий проектирования баз данных, которая использует знакомый нам инструментарий моделирования данных.

Проектирование базы данных — это итеративный процесс. Каждая итерация выполняется для доработки проекта и вносит определённые улучшения. Число итераций не ограничено. Проектирование закончить нельзя. Его можно только прервать.

На каждой итерации процесса выполняется один и тот же набор работ. То, что будет сказано далее о фазах и этапах проектирования, следует считать идеализированным описанием одной итерации. В этом описании проектирование выглядит как строго определённая последовательность работ. Однако в реальном проекте информация, полученная на одном из поздних этапов проектирования, может потребовать пересмотра решений, принятых на ранних этапах. Поэтому излагаемую ниже методоло-

гию следует понимать не как догму, а как схему, общее руководство к действию. Творческое следование этим рекомендациям позволяет повысить эффективность работы над проектом.

6.2 Фазы проектирования базы данных

Процесс проектирования разделяется на три основные фазы: концептуальное моделирование, логическое моделирование и физическое проектирование.

Концептуальное моделирование — это процесс создания информационной модели предприятия, не зависящей от каких-либо физических условий реализации.

На этой фазе создаётся модель данных предприятия, не зависящая от следующих факторов:

– типа СУБД, которая будет использована в системе;
– состава прикладных программ, которые будут обрабатывать данные;
– языков программирования, которые будут использоваться для написания программ;
– типа операционной системы;
– конкретной вычислительной платформы;
– любых других особенностей реализации.

Такое абстрагирование от особенностей реализации даёт возможность понять и ясно сформулировать требования будущих пользователей к данным. Вовлечение каких-либо соображений реализации БД в процесс анализа требований пользователей мешает пониманию этих требований. Прежде чем думать о реализации, нужно понять, что именно должно быть реализовано.

Логическое моделирование — это процесс создания информационной модели предприятия на основе конкретной модели данных логического уровня.

На этой фазе концептуальная модель данных предприятия преобразуется в логическую модель, ориентированную на определённый тип структур данных. Здесь имеет значение тип целевой СУБД. Прочие условия реализации не принимаются во внимание.

Физическое проектирование — это процесс создания описания структур хранения данных во вторичной памяти и методов доступа к данным, обеспечивающих наиболее эффективный доступ к информации.

На этой фазе принимаются окончательные решения о способах реализации базы данных. При этом обязательно учитываются все особенности используемой СУБД и среды реализации проекта.

Между фазами логического и физического проектирования имеется обратная связь. Для обеспечения высокой производительности системы может потребоваться пересмотр логической модели данных.

6.3 Факторы успешного завершения проекта

Для успешного завершения проектирования БД необходимо придерживаться следующих рекомендаций.

- Поддерживать постоянную связь с будущими пользователями системы.
- Использовать единую методологию на всех этапах проектирования.
- Разрабатывать систему исходя из существующих характеристик данных.
- В процессе моделирования учитывать требования поддержки структурной целостности и согласованности данных.
- Дополнять процедуры методологии технологическими приёмами концептуализации, нормализации и проверки целостности транзакций.
- Максимально использовать графические языки моделирования.
- Сопровождать диаграммы моделей словарём описания данных (глоссарием).
- Пересматривать решения, принятые на ранних этапах проектирования, если это необходимо для получения оптимального результата.

6.4 Общий обзор методологии

Здесь мы перечислим все этапы процесса проектирования БД, предусматриваемые настоящей методологией, и дадим их краткий обзор. В следующих разделах каждый этап будет описан подробно.

6.4.1 Фаза концептуального моделирования

Этап 1. Создание локальных концептуальных моделей данных пользователей.

На начальном этапе проект информационной системы организации разбивается на относительно небольшие подпроекты. Для этого определяются типы конечных пользователей (КП) системы и изучаются их преставления о предметной области проекта. Локальные представления кладутся в основу соответствующих *локальных концептуальных моделей*. Проектировщик, занимающийся проблемами конкретного типа КП, действует так, как если бы он проектировал базу данных именно и только для этого КП. Проблемы других пользователей его не интересуют. Его цель — полно и точно отразить представления именно этого типа пользователей.

Ниже перечислены виды работ, которые выполняются на этом этапе.

1.1. Определение типов сущностей.
1.2. Определение типов связей.
1.3. Определение атрибутов и связывание их с типами сущностей и связей.
1.4. Определение доменов атрибутов.
1.5. Определение атрибутов, являющихся потенциальными ключами и выделение первичных ключей.
1.6. Специализация или генерализация типов сущностей (необязательно).
1.7. Создание диаграммы «сущность-связь».
1.8. Обсуждение локальной модели с заинтересованными конечными пользователями.

6.4.2 Фаза логического моделирования

Работы на этой фазе зависят от типа целевой СУБД. Точнее — от абстрактной логической модели данных, которая поддерживается СУБД. В дальнейшем изложении мы будем иметь в виду реляционную модель данных.

Этап 2. Построение и проверка локальных логических моделей данных пользователей.

Для каждого локального представления создаются спецификации отношений, представляющих типы сущностей, типы связей и атрибуты локальной концептуальной модели. Для этого из локальной концептуальной модели удаляются элементы, затрудняющие реализацию модели в среде реляционной СУБД. Однако «удаляются» не значит «выбрасываются». Проектировщик создаёт конструкции РМД, эквивалентные «нежелательным» элементам концептуальной модели.

Затем корректность локальной логической модели проверяется с помощью правил нормализации. Это делается для того, чтобы убедиться в структурной согласованности, логической целостности и минимальной избыточности модели. Далее проверяется возможность выполнения транзакций пользователя в созданной схеме локальной РБД. Все эти проверки необходимы для того, чтобы получить уверенность в корректности модели данных.

Перечислим виды работ этапа.

2.1. Очистка локальной концептуальной модели от нежелательных элементов.

2.2. Определение набора отношений на основе очищенной концептуальной модели.

2.3. Проверка соответствия логической модели требованиям нормализации.

2.4. Проверка исполнимости транзакций пользователя.

2.5. Создание диаграмм модели.

2.6. Определение требований поддержки целостности данных.

2.7. Обсуждение локальной логической модели с заинтересованными конечными пользователями.

Локальные логические модели данных нередко используются для генерации прототипов баз данных отдельных типов пользователей.

Этап 3. Создание и проверка глобальной логической модели данных предприятия.

На этом этапе локальные логические модели интегрируются в единую логическую модель данных предприятия. Создаются спецификации схемы базы данных и процедур поддержки целостности. Выполняются следующие виды работ.

3.1. Слияние локальных логических моделей в единую глобальную модель данных.
3.2. Проверка глобальной логической модели.
3.3. Проверка возможностей расширения модели в будущем.
3.4. Создание окончательного варианта диаграмм модели.
3.5. Обсуждение глобальной логической модели с пользователями.

6.4.3 Фаза физического проектирования

Результатом двух предшествующих фаз являются спецификации базы данных *логического уровня*. На фазе физического проектирования нужно принять решения о том, как *реализовать* требуемые логические структуры и ограничения целостности в среде целевой СУБД. Рассматриваемая методология группирует работы, выполняемые на этой фазе, в четыре этапа.

Этап 4. Перенос глобальной логической модели данных в среду целевой СУБД.

4.1. Проектирование основных таблиц в среде целевой СУБД.
4.2. Реализация бизнес-правил в среде целевой СУБД.

Этап 5. Проектирование физического представления базы данных.

5.1. Анализ транзакций.
5.2. Выбор файловой структуры.
5.3. Определение вторичных индексов.
5.4. Анализ необходимости введения контролируемой избыточности данных.

5.5. Определение требований к дисковой памяти.

Этап 6. Разработка механизмов защиты.

6.1. Проектирование пользовательских представлений.

6.2. Определение прав доступа.

Этап 7. Организация мониторинга и настройка функционирования системы.

Контрольные вопросы

1. Опишите понятие методологии проектирования.
2. Перечислите, и кратко опишите фазы проектирования базы данных.
3. Перечислите факторы успешного завершения проекта базы данных.
4. Перечислите виды работ, выполняемых на фазе концептуального моделирования.
5. Перечислите виды работ, выполняемых на фазе логического моделирования.
6. Перечислите виды работ, выполняемых на фазе физического проектирования.

7 МЕТОДОЛОГИЯ КОНЦЕПТУАЛЬНОГО МОДЕЛИРОВАНИЯ

7.1 Создание локальной концептуальной модели

Локальная концептуальная модель описывает представление одного типа конечного пользователя[3]. Это представление включает в себя данные, необходимые пользователю для принятия решений или выполнения некоторого задания. Обычно оно отражает некоторую функциональную область в поле деятельности предприятия: производство, маркетинг, сбыт, управление кадрами, складской учёт и т.п. Задача проектировщика на этом этапе — изучить и адекватно описать требования пользователя к данным.

Начинать изучение информационных потребностей пользователя следует с выделения функциональных областей и отдельных функций. Затем, получив общее представление о функциях, следует опросить пользователей с целью уточнения некоторых деталей и проверки адекватности ваших представлений представлениям пользователей. Далее нужно изучить деловые процедуры, существующие отчёты, формы документов, провести наблюдения за работой пользователей и т.п. Цель всех этих мероприятий — понять, *что делает* будущий пользователь, *чем* он *ограничен* в своих действиях, *какие данные и как* он использует для выполнения своих функций.

Изучая локальное представление, аналитик накапливает описания

- процессов и сценариев деятельности,
- функций пользователя,
- входных документов и сообщений,
- выходных документов и сообщений,
- делового регламента,
- транзакций пользователя.

Эти описания (спецификации на проект) могут включать текстовые фрагменты, рисунки, диаграммы, формы и примеры

[3] Словом «пользователь» может обозначаться как отдельный работник, так и группа лиц, выполняющих сходные функции.

документов и отчётов. Спецификации содержат исходные данные для концептуального моделирования. В реальном проекте процессы накопления и анализа информации о локальном представлении идут параллельно.

Создание локальной концептуальной модели сводится к описанию (специфицированию):

- типов сущностей;
- типов связей;
- атрибутов (сущностей и связей);
- доменов атрибутов;
- потенциальных ключей;
- первичных ключей.

Для того чтобы добиться успеха в этой работе, необходимо зафиксировать ***цель*** и ***точку зрения*** создаваемой модели и согласовывать с ними все решения, принимаемые в процессе моделирования.

Рассмотрим основные детали процесса моделирования.

Этап 1.1. Определение типов сущностей

Тип сущности — это объект, который интересует пользователя при выполнении им каких-то служебных функций, т.е. виден с точки зрения модели.

Известно несколько методов идентификации типов сущностей. Один из них основан на изучении спецификаций функций пользователя. Опишем основные задачи, которые нужно решить на этом шаге.

Шаг 1. Составить список имён существительных.

Сущности в естественном языке обозначаются именами существительными. Поэтому, прежде всего, следует из текста спецификаций на проект извлечь имена существительные, возможно, с прилагательными и дополнениями, уточняющими их смысл. Например:

- преподаватель,
- учебная дисциплина,
- аудитория,
- лекция,
- лабораторная работа,

– семинар,
– занятие,
– кафедра,
– фамилия,
– должность,
– номер группы,
– численность.

Эта работа должна быть проделана очень тщательно. В ходе моделирования аналитику очень часто приходится обращаться к списку имён.

Шаг 2. Выяснить смысл, вкладываемый пользователем в каждое имя.

Смысл следует зафиксировать в рабочем словаре данных. Например, так, как в таблице 7.1.

Таблица 7.1 — Фрагмент рабочего словаря

Имя	Смысл	Примеч.
Преподаватель	Сотрудник ВУЗа, проводящий учебные занятия со студентами	
Учебная дисциплина	Раздел научно-технических знаний, включённый хотя бы в один учебный план как отдельная дидактическая единица	
Занятие	Планируемая встреча преподавателя и группы студентов с целью изучения учебной дисциплины	
…	…	

В ходе работы над моделью в словаре происходят изменения: появляются новые записи, уточняются описания смысла имён, добавляются различные примечания и т.п. Однако ни в коем случае *записи словаря не следует удалять*. Часто то, что аналитик счёл несущественным на каком-то уровне понимания предметной области, оказывается важным в дальнейшем.

Шаг 3. Выбрать из списка представляющие интерес для пользователя объекты и концепции.

Например, с точки зрения работника бюро расписаний в нашем списке таковыми могут быть:

ПРЕПОДАВАТЕЛЬ — специалист, проводящий учебные занятия (объект);

АУДИТОРИЯ — помещение, предназначенное для проведения учебных занятий (объект);

УЧЕБНАЯ ДИСЦИПЛИНА — раздел научно-технических знаний, изучаемый студентами (концепция);

ЗАНЯТИЕ — запланированная встреча преподавателя и группы студентов с целью изучения учебной дисциплины (концепция).

Шаг 4. Среди оставшихся элементов списка попытаться выделить группы имён, которые могут быть характеристиками или свойствами каких-либо объектов или концепций.

Так, в приведённом выше списке это

фамилия, должность } характеристики ПРЕПОДАВАТЕЛя;

номер группы, численность } характеристики не упомянутого в списке объекта ГРУППА.

Шаг 5. Среди оставшихся элементов списка попытаться выделить группы имён, имеющих сходный смысл.

Такие группы идентифицируются именами, выражающими смысл сходства. Эти новые имена могут обозначать объекты или концепции, интересные с точки зрения пользователя.

Например, в нашем списке это лекция, лабораторная работа и семинар. Очевидно, это названия видов учебных занятий. Эту группу можно идентифицировать именем вид занятия.

Новые понятия, появляющиеся в голове аналитика в ходе анализа списка имён, обязательно и немедленно отражаются в рабочем словаре.

Шаг 6. Основываясь на результатах шагов 3 — 5, составить список «кандидатов» в сущности.

В нашем примере в этот список войдут имена ПРЕПОДАВАТЕЛЬ, АУДИТОРИЯ, УЧЕБНАЯ ДИСЦИПЛИНА, ЗАНЯТИЕ, входившие в исходный список. Кроме того, в процессе анализа появились имена — группа и вид занятия. Здравый смысл подсказывает, что они также могут оказаться сущностями.

ГРУППА — сформированное приказом ректора множество студентов одного года обучения, получающих одну специальность.

ВИД ЗАНЯТИЯ — способ изучения учебной дисциплины.

Шаг 7. Выделить в списке «кандидатов» сущности модели.

Задача этого шага — отделить сущности от атрибутов и связей. Часто объект можно вполне обоснованно отнести к любой из этих категорий.

Примеры.

Объект АУДИТОРИЯ следует трактовать как сущность, если пользователю для исполнения какой-то функции нужно знать не только номер аудитории (идентификатор), но и вместимость, тип (лекционная, компьютерный класс, лаборатория микроэлектроники…), принадлежность и т.п. Этот же объект следует рассматривать как атрибут какой-то сущности, скажем, сущности ЗАНЯТИЕ, если пользователю достаточно различать аудитории только по идентификаторам.

Объект ЗАНЯТИЕ можно трактовать как сущность, атрибутами которой являются Преподаватель, Группа, Учебная дисциплина, Аудитория. Этот же объект можно трактовать как связь (ассоциацию) четырёх сущностей.

Анализ требований пользователя — субъективный процесс. Различные аналитики могут создавать различные интерпретации одних и тех же фактов. Эти интерпретации могут быть менее или более удачными (естественными, ясными, точными). Выбор варианта интерпретации существенно зависит от двух факторов: здравого смысла аналитика (хорошо усвоенного жизненного опыта) и накопленного опыта аналитической работы. Основными ориентирами в процессе анализа являются цель модели точка зрения пользователя. Аналитик должен смотреть на предметную область его глазами и уметь описать увиденное средствами базовой модели данных. Как правило, выделить полный набор сущностей с первой попытки не удаётся.

Шаг 8. Документировать типы сущностей.

Каждой выделенной сущности нужно присвоить уникальное имя. Это должно быть имя существительное в единственном числе, возможно, в комбинации с прилагательным. Оно должно

обозначать один экземпляр сущности и в равной степени относиться к любому экземпляру. Оно должно быть понятно пользователю. Нарушение этих требований искажает смысл и затрудняет понимание модели.

Выбранное имя и описание его смысла (определение) нужно поместить в словарь данных. Если можно оценить ожидаемое число экземпляров сущности, то эту оценку также следует зафиксировать в словаре. Если сущность известна пользователю под различными именами, то все их нужно определить как синонимы основного и занести в словарь.

Этап 1.2. Определение типов связей

Тип связи — это отношение одного или более типов сущностей, представляющее интерес с точки зрения пользователя. Техника выделения типов связей подобна изложенной выше технике выделения типов сущностей. И проблемы возникают в ходе анализа подобные. Однако теперь у аналитика есть фундамент — список сущностей.

Шаг 1. Из текста спецификаций на проект, выписать простые предложения, содержащие имена сущностей.

Наиболее часто встречаются предложения, содержащие имена двух сущностей. Каждое такое предложение описывает бинарную связь. Однако нужно быть осторожным, и тщательно проверять наличие в ПО связей высших степеней. Например, обнаружив предложения:

«ГРУППА изучает УЧЕБНую ДИСЦИПЛИНу» и

«ПРЕПОДАВАТЕЛЬ преподаёт УЧЕБНую ДИСЦИПЛИНу»,

можно упустить из виду тернарную связь

«ГРУППА изучает УЧЕБНую ДИСЦИПЛИНу под руководством ПРЕПОДАВАТЕЛя».

Эта связь может оказаться более интересной с точки зрения пользователя. Она содержит информацию, которой нет в двух бинарных связях.

Шаг 2. Проверить, все ли связи, упомянутые в спецификациях и следующие из них, включены в список.

В идеале каждую пару, тройку,… сущностей следовало бы проверить на наличие между ними какой-либо связи. Однако эта

работа может оказаться чересчур трудоёмкой. Тем не менее, отказываться от этого не стоит. Число проверяемых вариантов можно сократить, исходя из точки зрения модели и соображении здравого смысла.

Например, с точки зрения работника бюро расписаний не видны связи

«ГРУППА изучает УЧЕБНую ДИСЦИПЛИНу» и

«ПРЕПОДАВАТЕЛЬ преподаёт УЧЕБНую ДИСЦИПЛИНу», но видна тернарная связь

«ГРУППА изучает УЧЕБНую ДИСЦИПЛИНу под руководством ПРЕПОДАВАТЕЛя».

А с точки зрения заместителя декана факультета, разрабатывающего рабочий план занятий на следующий учебный год, не видна эта тернарная связь, но видна связь

«ГРУППА изучает УЧЕБНую ДИСЦИПЛИНу».

Шаг 3. Определить кардинальные числа связей и ограничения на степень участия сущностей в связях.

Далеко не всегда удаётся определить точные значения кардинальных чисел или, хотя бы, интервалы значений, но к этому нужно стремиться.

Специфицирование кардинальности связей и степени участия сущностей в связях необходимо для наглядного отражения смысла связей в модели. Эти характеристики являются ограничениями целостности данных. Их можно и нужно поддерживать при обновлении БД.

Шаг 4. Документировать типы связей.

Каждому типу связи необходимо дать имя, выражающее его смысл и понятное пользователю. В словарь данных необходимо поместить развёрнутое описание связи.

Шаг 5. Создать диаграмму верхнего уровня модели.

Текстовое описание сущностей и связей (словарь данных) следует дополнить ER-диаграммой. Модель необходимо согласовать с пользователем. Опыт показывает, что диаграмма значительно облегчает этот процесс.

Этап 1.3. Определение атрибутов сущностей и связей

Шаг 1. Составить список атрибутов.

Атрибут — это свойство сущности или связи. В тексте спецификаций на проект атрибуты, как и сущности, обозначаются именами существительными. Обычно атрибутами являются реквизиты входных и выходных документов и сообщений.

Проще всего обнаружить атрибуты в процессе идентификации сущностей и связей. Выделив сущность (связь), нужно задать себе вопрос:

«Какие сведения об этом интересуют пользователя?»

Ответ следует искать в тексте спецификаций. Не стоит обращаться с таким вопросом к пользователю до того, как Вы сформируете собственное мнение. Пользователь, знающий все детали своей работы, выложит их Вам, и Вы будете вынуждены отсеивать главное от второстепенного. Это проще сделать, имея «решето» — своё понимание информационных потребностей пользователя.

Шаг 2. Классифицировать атрибуты.

Атрибут может быть простым или составным, однозначным или многозначным, первичным или производным. Задача этого шага — выявить тип каждого атрибута, определяемый набором значений этих трёх бинарных признаков.

Если значения атрибута рассматриваются пользователем как единое целое, то его следует считать простым. Если же пользователю нужен доступ к отдельным компонентам значений атрибута, то он должен быть представлен в модели как составной, т.е. именованный набор простых атрибутов. Все компоненты составного ключа должны быть идентифицированы как простые атрибуты.

Атрибут идентифицируется как многозначный, если одному экземпляру сущности соответствует в общем случае несколько его значений и пользователю необходим доступ к отдельным значениям.

Пример. Пусть пользователю нужно знать, чем увлечены студенты (помимо учёбы, разумеется). Сущность `СТУДЕНТ` содержит атрибут `Увлечения`. Рассуждая абстрактно, это многозначный атрибут. У студента Колова нет никаких увлечений, а Фолов — бас-гитарист, парапланерист, байкер, коллекционер

марок и мастер спорта по спортивной гимнастике. Кроме того, он любит вышивать гладью и плести кружева на коклюшках.

Однако чтобы сделать вывод о многозначности атрибута в *конкретном локальном представлении*, нужно ответить на вопрос:

«Нужно ли пользователю выделять экземпляры сущности по отдельным значениям этого атрибута?»

В условиях нашего примера: нужно ли пользователю отбирать и «складывать в отдельные кучки»

- парапланеристов, вышивающих гладью;
- байкеров, умеющих плести кружева;
- бас-гитаристов, увлекающихся спортивной гимнастикой;
- и т.п.?

Если ответ «нет», то атрибут следует специфицировать как однозначный.

Атрибут является производным, если его значения могут быть установлены по значениям других атрибутов. Например, значение численности группы можно определить, подсчитав число студентов, зачисленных в группу. Возраст студента можно вычислить, зная дату его рождения.

Обычно производными атрибутами оказываются реквизиты выходных документов и сообщений. Такие атрибуты вообще не отображаются в концептуальной модели. Однако иногда производный атрибут необходимо показать как атрибут какой-то сущности. В этом случае он обязательно должен быть специфицирован в словаре данных как производный, и должен быть указан способ получения его значений.

Шаг 3. Связать атрибуты с типами сущностей и связей.

Обычно атрибут идентифицируют именно как свойство конкретной сущности или связи. Однако в спецификациях на проект могут быть упомянуты реквизиты, которые не удаётся таким способом связать ни с одной сущностью/связью. Если обнаружен такой реквизит, то нужно ответить на ниже перечисленные вопросы.

Действительно ли пользователь заинтересован в сохранении и обработке значений этого реквизита?

Не является ли реквизит атрибутом какой-то известной сущности/связи?

Не пропущена ли какая-то сущность/связь, атрибутом которой может быть этот реквизит?

Если в результате этого исследования обнаружена новая сущность/связь, то её следует документировать и отобразить на диаграмме.

Иногда у аналитика создаётся впечатление, что некоторый атрибут должен быть связан с несколькими типами сущностей. Это означает, что имеет место одна из двух ситуаций.

Ситуация 1. Модель включает несколько сущностей, представляющих экземпляры одной и той же обобщённой сущности. В этом случае следует обдумать целесообразность объединения сущностей в некоторый обобщённый тип (см. Этап 1.6).

Ситуация 2. Обнаружена новая связь между типами сущностей. В этом случае нужно связать атрибут с одной сущностью — родителем в новой связи, и определить новый тип связи (см. Этап 1.2).

Шаг 4. Документирование атрибутов.

Каждому атрибуту должно быть присвоено имя. Имя должно быть понятным пользователю и уникальным в пределах сущности. О каждом атрибуте в словарь данных должны быть внесены следующие сведения:

– имя и описание смысла;
– все известные синонимы имени;
– тип данных и длина поля;
– значение по умолчанию (если необходимо);
– признак необязательности значения (допустимость неопределённых значений);
– признак композитности (является ли атрибут составным);
– перечень компонентов (для составного атрибута);
– признак многозначности;
– признак вычислимости (является ли атрибут производным);
– метод получения значений (для производного атрибута).

Этап 1.4. Определение доменов атрибутов

Домéн — это множество, элементы которого могут быть значениями одного или нескольких атрибутов. В полной модели данных каждому атрибуту должен быть сопоставлен домéн. Сведения, необходимые для определения доменов, можно получить из примеров входных и выходных документов и сообщений, а также от пользователей.

Описание домена должно содержать:

– определение множества значений;

– сведения о размере и формате поля для каждого атрибута, определённого на домене.

Пример.

Домен кодов учебных дисциплин (атрибут Код УД*).*

Значения:

строки вида «ББББЦЦ», где Б — буква, Ц — цифра;

допустимые комбинации букв:

ГСЭ, ЕНМ, ОПД, СД, ДС, ФД;

допустимые комбинации цифр:

любые в интервале от 01 до 55.

Максимальная длина поля: пять символов.

Кроме того, описание домена может включать сведения:

– о допустимых операциях со значениями;

– о сравнимых доменах.

Каждому домену присваивается уникальное имя. Имя домена и его описание помещаются в словарь данных. Одновременно в каждую запись словаря, относящуюся к атрибуту, заносится имя соответствующего домена.

В практике проектирования определяются только домены простых типов данных.

Этап 1.5. Определение потенциальных и выбор первичных ключей

Потенциальный ключ сущности — это подмножество её атрибутов, значение которого уникально идентифицирует экземпляр. Оно не может содержать собственное подмножество, обладающее таким же свойством. Определение потенциального

ключа является ограничением целостности данных. Произвольное подмножество атрибутов *нельзя назначить* потенциальным ключом[4]. Некоторое подмножество атрибутов либо обладает свойствами потенциального ключа, либо не обладает. Задача проектировщика — выяснить, обладает или нет. Выясняя это, аналитик исходит из делового регламента и руководствуется соображениями здравого смысла.

Шаг 1. Выявить стержневые сущности модели.

По отношению к каждой сущности следует задать себе вопрос:

«Зависит ли существование экземпляров этой сущности от существования экземпляров какой-либо другой сущности?»

Если ответ «НЕТ», то это стержневая сущность.

Например, пусть в модели идентифицированы сущности ТОВАР — объект купли/продажи и ПОСТАВКА — факт получения некоторой партии ТОВАРа. Существование экземпляров ТОВАРа зависит только от решения руководства организации и не зависит от существования экземпляров какой-либо другой сущности. Это стержневая сущность. Существование же экземпляров ПОСТАВКи логически зависит от существования экземпляров ТОВАРа. Нельзя поставить несуществующий товар. ПОСТАВКА — это слабая сущность.

Слабая сущность может не иметь потенциальных ключей. Стержневая сущность обязательно имеет, по крайней мере, один. Начинать выявление потенциальных ключей следует со стержневых сущностей.

Шаг 2. Выявить потенциальные ключи сущностей.

Вот самый надёжный «рецепт» отыскания потенциальных ключей.

Выбрать простой однозначный атрибут сущности, и задать себе вопрос:

«Могут ли одновременно существовать два экземпляра этой сущности с одинаковыми значениями этого атрибута?»

---

[4] Нередко приходится слышать от студента: «Я хочу, чтобы это был ключ». Или: «А это ERwin так нарисовал!». Подобные высказывания аналитику, изучающему свойства данных, не к лицу.

Если ответ «НЕТ», то этот атрибут — простой потенциальный ключ. В противном случае это не потенциальный ключ, но, возможно, компонент составного потенциального ключа.

Перебрав все одиночные атрибуты, анализ следует продолжить, перебирая пары, тройки и т.д. Обнаруженные в некоторой итерации потенциальные ключи не должны участвовать в последующих. При этом нужно иметь в виду, что *любое собственное подмножество* составного одного потенциального ключа может быть частью другого. Например, если пара атрибутов `{A, B}` — потенциальный ключ, то потенциальными ключами могут оказаться и пары `{A, C}`, `{D, B}`, и тройки `{A, D, E}`, `{B, C, E}`. Но никакая тройка, включающая пару `{A, B}`, потенциальным ключом быть не может.

Этот рецепт кажется ужасающе тупым. Опытный аналитик часто видит потенциальные ключи сущностей «невооружённым глазом». Но, во-первых, «часто» — это не «всегда», а во-вторых, даже опытный аналитик должен убедиться в том, что он увидел ВСЕ потенциальные ключи сущности. Гарантию может дать только полный перебор.

Если у какой-то сущности не удалось обнаружить ни одного потенциального ключа, то это указывает на одну из двух ситуаций.

Ситуация 1. Это слабая сущность. Решение об идентификации её экземпляров следует отложить до фазы логического моделирования.

Ситуация 2. Аналитику известны не все атрибуты сущности или бизнес-правила, или он что-то неправильно понимает. Требуется дополнительное изучение предметной области.

Шаг 3. Выбрать первичный ключ сущности.

Если сущность имеет единственный потенциальный ключ, то проблемы нет. Он и есть первичный. Если же потенциальных ключей несколько, то из них нужно выбрать один, значения которого будут реально использоваться для идентификации экземпляров сущности. Это решение уже зависит от произвола аналитика. Но произвол этот, всё же, сильно ограничен.

Идеальный первичный ключ обладает следующими свойствами:

– имеет минимальный (во множестве всех потенциальных ключей сущности) состав атрибутов;

– его значения у существующих экземпляров сущности изменяются редко;

– он ни при каких условиях не потеряет свойств потенциального ключа[5];

– если он текстовый, то имеет минимальную длину поля во множестве всех потенциальных ключей сущности;

– его значения воспринимаются пользователем именно как идентификаторы реальных объектов.

В реальной жизни наличие всех перечисленных свойств у одного потенциального ключа — большая редкость. Обычно первичный ключ выбирают по какому-то взвешенному критерию, принимая во внимание перечисленные свойства.

Выбрав первичный ключ сущности, все остальные потенциальные ключи следует пометить как альтернативные.

Шаг 4. Документировать первичные и альтернативные ключи сущностей.

В словарь данных внести записи обо всех обнаруженных ключевых группах атрибутов. Для каждой группы указать тип (первичный/альтернативный) и принадлежность.

Этап 1.6 (необязательный). Специализация/генерализация типов сущностей

Может оказаться так, что модель содержит несколько типов сущностей, имеющих сходные наборы атрибутов и/или участвующих в аналогичных связях. Или, обратная ситуация, можно выделить подмножества экземпляров сущности, участвующие в

[5] Например, в современных российских условиях ни наименование предприятия, ни его юридический адрес не могут быть идентификаторами предприятия, но пара {наименование, адрес} — может. Никто не зарегистрирует два различных предприятия с одинаковыми названиями по одному и тому же адресу. Но если завтра Госдума примет закон об уникальности наименований предприятий, то пара {наименование, адрес} потеряет свойство неизбыточности потенциального ключа.

различных связях и/или имеющие различающиеся наборы атрибутов. В таких случаях следует оценить целесообразность генерализации (обобщения сходных типов сущностей в супертип) или специализации (выделения подмножеств сущности в отдельные подтипы).

Решение о выполнении специализации/генерализации принимается с целью повышения наглядности и ясности модели. При этом обязательно учитывают особенности моделируемой предметной области.

Пример

Пусть в локальном представлении риэлтера выделены сущности СОБСТВЕННИК, НЕДВИЖИМОСТЬ ДЛЯ АРЕНДЫ, НЕДВИЖИМОСТЬ ДЛЯ ПРОДАЖИ, ПОКУПАТЕЛЬ, АРЕНДАТОР. Соответствующий фрагмент ER-диаграммы представлен на рис. 6.1.

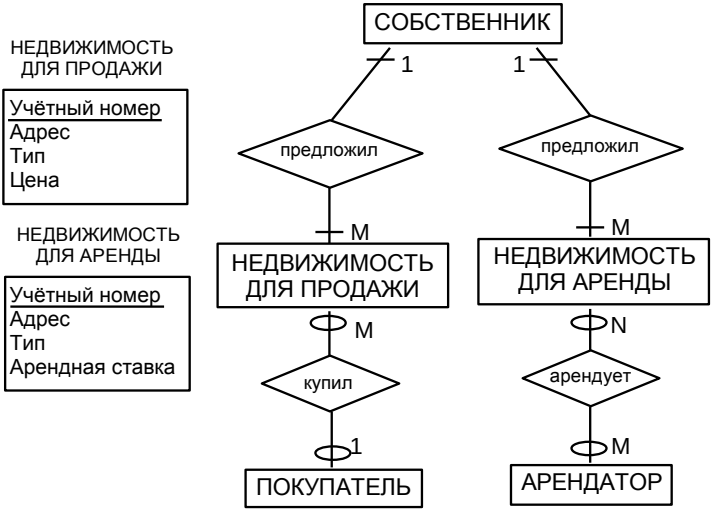


Рис. 6.1 — Фрагмент локального представления риэлтера

В этом представлении сущности НЕДВИЖИМОСТЬ ДЛЯ АРЕНДЫ и НЕДВИЖИМОСТЬ ДЛЯ ПРОДАЖИ различаются только атрибутами Цена и Арендная ставка. Каждая из них участвует в связи с сущностью СОБСТВЕННИК. Смысл обеих связей один и тот же. Можно подумать о генерализации сущностей.

Если мы примем такое решение, то рассматриваемый фрагмент будет выглядеть так, как на рис. 6.2.

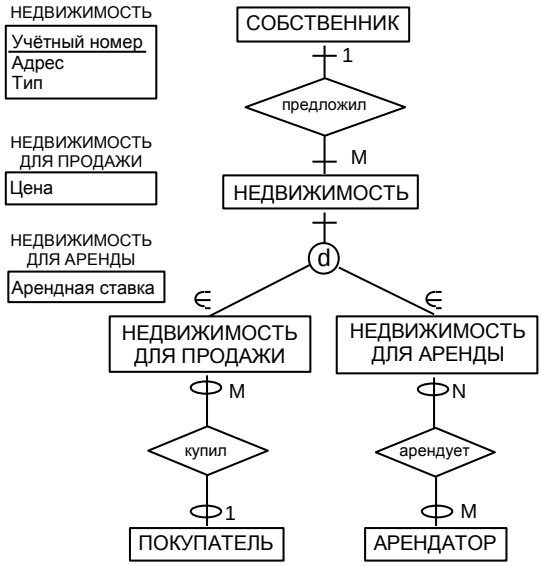


Рис. 6.2 — Генерализация сущностей НЕДВИЖИМОСТЬ ДЛЯ АРЕНДЫ *и* НЕДВИЖИМОСТЬ ДЛЯ ПРОДАЖИ

Здесь введён супертип сущности НЕДВИЖИМОСТЬ. Окружность, соединённая с ним, обозначает связь «супертип–подтип». Отрезок, который пересекает дугу, соединяющую супертип с окружностью, показывает, что каждый экземпляр супертипа принадлежит одному из подтипов. Символ `'d'` внутри окружности означает, что подтипы не пересекаются (`disjoint`). Символ `'∈'` около дуги, соединяющей подтип с окружностью, указывает на принадлежность экземпляров подтипа супертипу. С супертипом связаны только общие атрибуты обобщаемых сущностей. С каждым подтипом — только те, которые принадлежат соответствующей обобщаемой сущности. Атрибуты супертипа наследуются каждым подтипом.

Полученная в результате генерализации модель наглядно отражает правило, которое не видно в предыдущем варианте: экземпляр недвижимости не может быть одновременно объектом купли/продажи и аренды.

Исследование этого фрагмента представления риэлтера можно продолжить. Собственники, покупатели и арендаторы недвижимости являются его клиентами. Наборы атрибутов соответствующих сущностей пересекаются. Можно подумать о представлении сущностей СОБСТВЕННИК, ПОКУПАТЕЛЬ и АРЕНДАТОР в виде подтипов супертипа КЛИЕНТ. Возможный результат показан на рис. 6.3.

Здесь символ 'O' внутри значка связи «супертип–подтип» означает, что подтипы КЛИЕНТа пересекаются. Один и тот же клиент может быть покупателем, собственником и арендатором объектов недвижимости, разумеется, в различных сделках.

Если наборы атрибутов типов сущностей СОБСТВЕННИК, ПОКУПАТЕЛЬ и АРЕНДАТОР совпадают, то, возможно, нет смысла выделять подтипы КЛИЕНТа. Результат объединения показан на рис. 6.4.

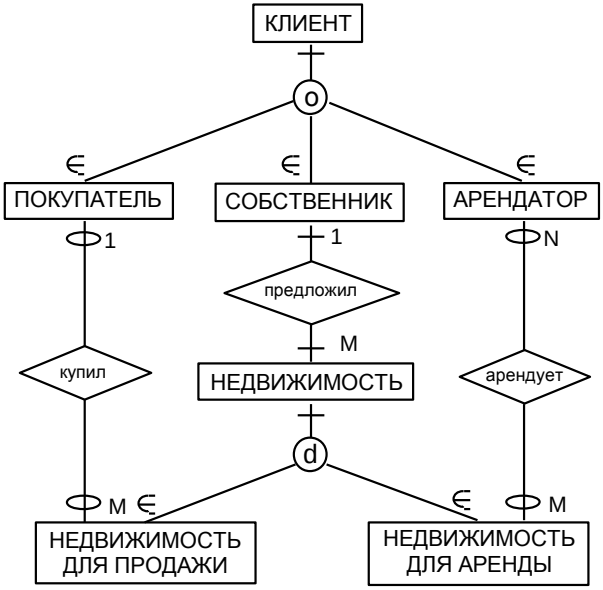


Рис. 6.3 — Генерализация сущностей СОБСТВЕННИК, ПОКУПАТЕЛЬ и АРЕНДАТОР

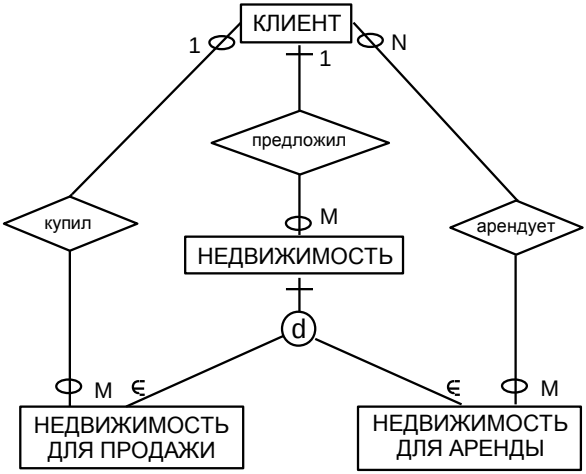


Рис. 6.4 — Объединение сущностей СОБСТВЕННИК, ПОКУПАТЕЛЬ *и* АРЕНДАТОР

Клиентами риэлтера могут быть физические лица или организации. Наборы реквизитов физического лица и организации различаются. Может быть полезна специализация типа сущности КЛИЕНТ по этому признаку. Результат специализации для варианта, приведённого на рис. 6.3, показан на рис. 6.5. Здесь тип сущности КЛИЕНТ представлен в двух «ракурсах».

В реальной задаче вариантов генерализации и специализации может быть очень много. Опытный аналитик обязательно исследует их. Даже если впоследствии все они будут отвергнуты, труды не пропадут даром. Их результатом будет лучшее понимание проблем пользователя и более высокое качество модели.

Результаты генерализации/специализации обязательно документируются. В словарь данных помещаются сведения о новых сущностях (супертипах и подтипах), описания новых связей, обновляются сведения о принадлежности атрибутов типам сущностей, сведения о первичных и альтернативных ключах.

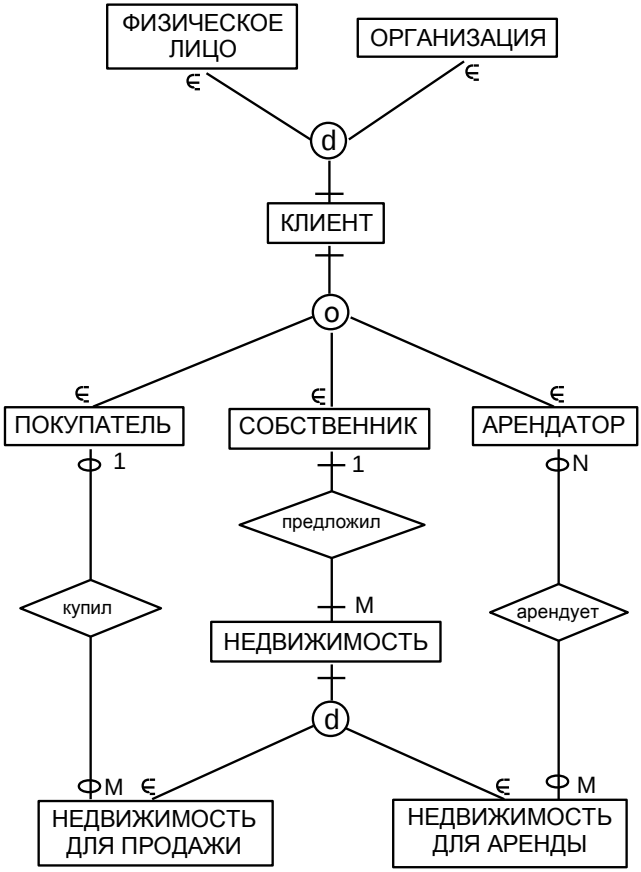


Рис. 6.5 — Специализация сущности КЛИЕНТ

Этап 1.7. Создание ER-диаграммы

Наброски и рабочие варианты диаграмм создаются с самого первого шага анализа. Они помогают аналитику определить направление дальнейших исследований, подготовиться к опросам пользователей, подобрать другие источники информации о предметной области.

Полный вариант диаграммы «сущность-связь» создаётся тогда, когда аналитик намерен обсудить модель с конечным пользователем с целью принятия решения о завершении работ

на этапе концептуального моделирования. Диаграмма должна точно и ясно отражать представления аналитика об информационных потребностях локального пользователя. Она является основным проектным документом, выносимым на защиту модели.

Этап 1.8. Обсуждение локальной концептуальной модели с конечным пользователем

Цель обсуждения: оценить степень корректности и полноты модели.

Обсуждение должно быть подготовлено. Для этого следует подобрать пакет документов, включающий ER-диаграмму и дополнительные материалы — словарь данных, список бизнес-правил, описания транзакций. Эти документы нужно передать пользователю для изучения и договориться с ним о дате и времени встречи. Помните, что обсуждение Ваших проблем — это не основная работа пользователя. И, тем более, не хобби.

Обычно в первом варианте модели обнаруживаются несоответствия. По результатам обсуждения в модель следует внести изменения, повторив шаги 1.1÷1.7, и вновь обсудить результаты с пользователем.

Этот процесс должен продолжаться до тех пор, пока пользователь не подтвердит, что модель корректно отражает его личные представления о требуемом приложении и о работе предприятия в целом. Только после этого можно переходить к следующей фазе проектирования.

Контрольные вопросы

1. Для чего предназначена локальная концептуальная модель?
2. Что является исходными данными для локальной концептуальной модели?
3. Перечислите этапы создания локальной концептуальной модели.
4. Какие виды работ выполняются на этапе определения типов сущностей?

5. Какие виды работ выполняются на этапе определения типов связей?

6. Какие виды работ выполняются на этапе определения атрибутов сущностей и связей?

7. Какие виды работ выполняются на этапе определения доменов?

8. Какие виды работ выполняются на этапе определения ключей?

9. В каких случаях может быть полезной генерализация/специализация типов сущностей?

10. Опишите завершающие действия этапа локального концептуального моделирования.

8 МЕТОДОЛОГИЯ ЛОГИЧЕСКОГО МОДЕЛИРОВАНИЯ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

Цель фазы логического моделирования — создание обобщённой модели данных организации на основе локальных концептуальных моделей и с использованием конкретного способа структурирования данных. Виды работ, выполняемых в ходе логического моделирования, зависят от типа целевой СУБД. Перечень этапов и видов работ, выполняемых на фазе логического моделирования реляционной базы данных (РБД) приведён в п. 6.4.2. В настоящем разделе эти работы описаны детально.

8.1 Построение и проверка локальной логической модели

Цель этапа — определить набор отношений для представления сущностей, связей и атрибутов локальной концептуальной модели данных пользователя.

Этап 2.1. Очистка локальной концептуальной модели от нежелательных элементов

Локальная концептуальная модель может содержать элементарные конструкции, которым в РМД соответствуют структурные единицы — отношения (см. п. 2.6). Это композитные и многозначные атрибуты, связи степени $n \neq 2$, бинарные связи типа M:N и связи с атрибутами. Все такие элементы, встречающиеся в локальной концептуальной модели, должны быть предварительно преобразованы в сущности и бинарные связи типа 1:M. Это и называется очисткой концептуальной модели.

Если заранее известно, что проектируется реляционная база данных, то очистка выполняется уже на фазе концептуального моделирования. Опытный аналитик в этом случае мыслит в категориях отношений, и стремится не допустить появления нежелательных элементов в концептуальной модели. Тем не менее, приступая к логическому моделированию, необходимо убедиться в чистоте концептуальной модели.

Шаг 1. Преобразование бинарных связей типа M:N.

Реляционная модель данных поддерживает только бинарные связи типа 1:M. Связь отношений реализуется как ссылка

из кортежа отношения–потомка на кортеж отношения–родителя. В связи типа M:N нет родителя и потомка. В терминах РМД такая связь представляется отношением (см. п.2.6). Кортежи этого отношения представляют факты соответствия экземпляров сущностей–участниц связи. В каждом конкретном случае такому отношению можно сопоставить концепцию, существующую в голове пользователя. Это означает, что в концептуальной модели бинарную связь типа M:N можно заменить сущностью. Новая сущность должна быть потомком в бинарных связях с сущностями–участницами преобразуемой связи типа M:N. Эта сущность всегда слабая. Её экземпляры не могут существовать вне связей с обеими родительскими сущностями.

Пример преобразования приведён на рис.8.1. В голове пользователя, обрабатывающего факты получения товаров от поставщиков, существует концепция ПОСТАВКА. До преобразования она скрыта в связи «выполнил/включён в», а после преобразования отражена в модели явно.

Шаг 2. Преобразование связей высших степеней.

Каждая связь степени n > 2 должна быть заменена в очищенной модели слабой сущностью. Как и в предыдущем случае, эта новая сущность является потомком в n бинарных связях типа 1:M. Родителями в этих связях выступают сущности–участницы n-арной связи.

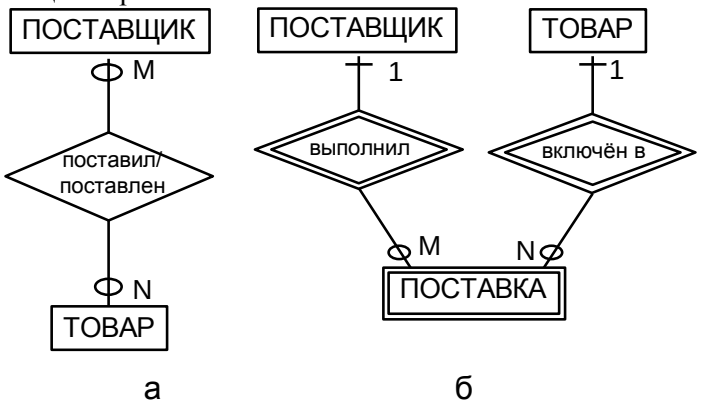


Рис. 8.1— Преобразование связи M:N:
а — исходная связь; б — результат преобразования

Пример преобразования приведён на рис. 8.2. Как и в предыдущем случае, в голове пользователя существует концепция ЗАНЯТИЕ, которая после преобразования отражена в модели явно.

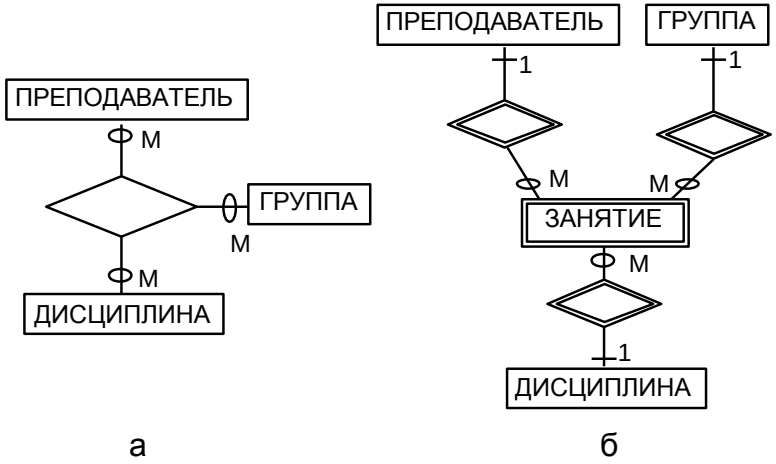


Рис. 8.2 — Преобразование n-арной связи:
а — исходная связь; б — результат преобразования

Шаг 3. Преобразование унарных связей.

Унарные связи, как и бинарные, могут иметь тип `1:1`, `1:M` или `M:N`. Каждая унарная связь типа `M:N` или `1:M` обязательно должна быть заменена слабой сущностью. Новая сущность является потомком в двух бинарных связях с одной и той же родительской сущностью.

Пример преобразования приведён на рис. 8.3. Теперь в модели явно отражена концепция КОМПОНЕНТ — часть изделия, сама являющаяся изделием.

Относительно унарной связи типа `1:1` допустимы два варианта решения: сохранить в логической модели в неизменном виде, отложив решение до этапа определения отношений, или преобразовать как унарную связь `M:N`.

Заметим, что нередко наличие унарных связей в модели указывает на желательность специализации. Следует попытаться выделить подтипы сущности и рассмотреть связи между ними.

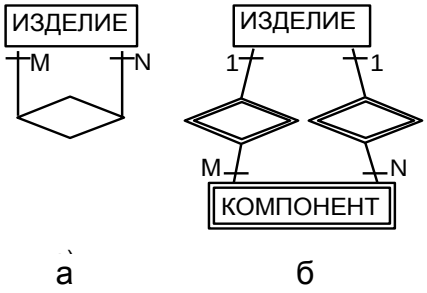


Рис. 8.3 — Преобразование унарной связи M:N:
а — исходная связь; б — результат преобразования

Например, пусть изделием с точки зрения пользователя являются:

– ДЕТАЛЬ — неделимая часть изделия,

– УЗЕЛ — отдельная часть изделия, состоящая из двух и более деталей,

– ГОТОВОЕ ИЗДЕЛИЕ — собранная из УЗЛов товарная единица, выпускаемая предприятием.

Тогда в результате специализации типа сущности ИЗДЕЛИЕ качество модели явно улучшится, а унарная связь, изображённая на рис. 8.3а), не появится (см. рис. 8.4).

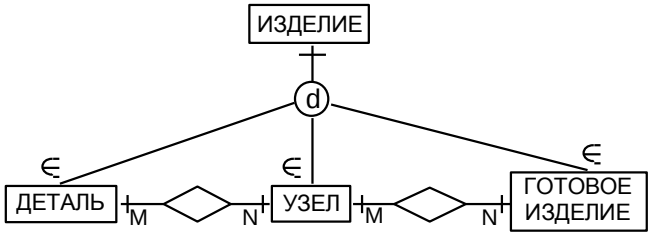


Рис. 8.4 — Специализация, устраняющая унарную связь

Шаг 4. Преобразование связей с атрибутами.

Каждая связь с атрибутами, независимо от её типа, обязательно должна быть преобразована в слабую сущность. Этой сущности назначаются все атрибуты связи. Пример преобразования показан на рис. 8.5.

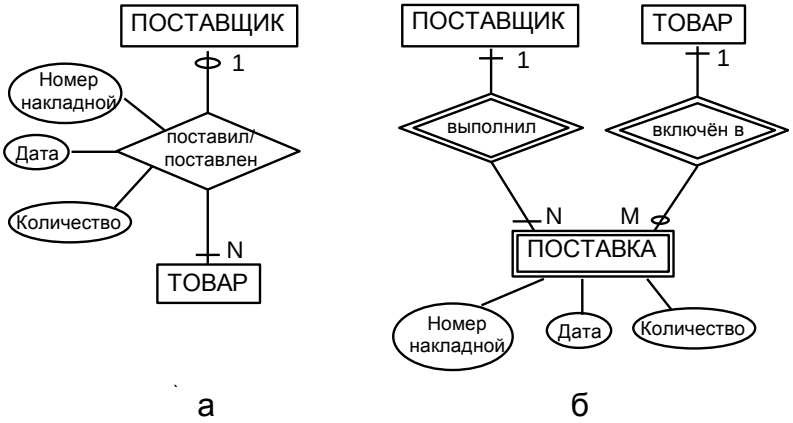


Рис. 8.5 — Преобразование связи с атирбутами:
а — исходная связь; б — результат преобразования

Здесь связь с атрибутами «поставил/поставлен» заменена слабой сущностью ПОСТАВКА и двумя бинарными связями типа 1:M. Новые связи не имеют атрибутов.

Шаг 5. Преобразование многозначных атрибутов.

Каждый многозначный атрибут (простой или композитный) должен быть заменён слабой сущностью, связанной бинарной связью типа 1:M с сущностью-владельцем многозначного атрибута. После преобразования многозначный атрибут уже не включается в состав атрибутов родительской сущности. Его значения трактуются как экземпляры потомка.

Шаг 6. Преобразование композитных атрибутов.

Каждый композитный атрибут должен быть заменён соответствующим множеством простых атрибутов.

Примеры преобразования многозначного и композитного атрибутов приведены на рис. 8.6.

Шаг 7. Проверка целесообразности связей типа 1:1.

Связь типа 1:1 может представлять действительно существующее отношение двух различных сущностей, а может быть результатом ошибки анализа.

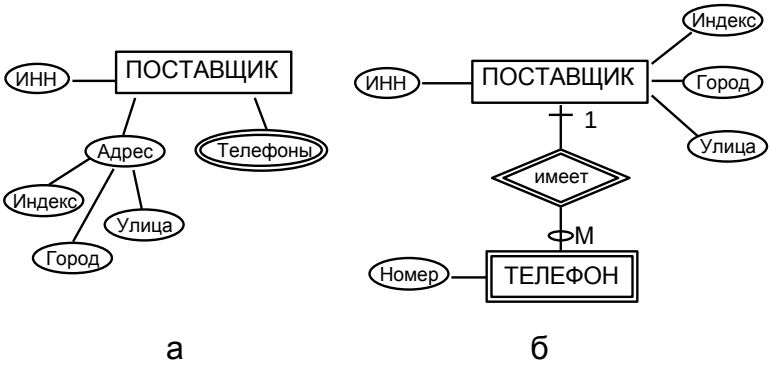


Рис. 8.6 — Преобразование многозначного и композитного атрибута: а — исходный состав атрибутов; б — результат преобразования

Может оказаться так, что за различные сущности принимаются различные аспекты одного и того же объекта предметной области. В этом случае составы атрибутов сущностей будут пересекаться, по крайней мере, в части потенциальных ключей. Например, беседуя с пользователем, и изучая документы, аналитик выделил сущности `КОНТРАГЕНТ` и `ПОСТАВЩИК`, и обнаружил, что они состоят в связи `1:1`. В дальнейшем выяснилось, что составы атрибутов и связи этих сущностей совпадают с точностью до имён. Очевидно, в этом случае слова «контрагент» и «поставщик» — синонимы. В модели следует оставить одну сущность.

В общем случае нужно исследовать степень участия сущностей в связи. Если оказывается, что участие обязательно с обеих сторон, то одну из сущностей можно удалить из модели. Принимая такое решение, нужно исследовать все связи, в которых участвуют эти сущности. Если каждой связи одной из них можно сопоставить эквивалентную по смыслу связь другой, то на самом деле эти две сущности представляют один объект (концепцию). Их нужно объединить. Процедура объединения:

1. Объединить множества атрибутов сущностей и присвоить объединению имя одной из них.

2. Имя другой сущности объявить синонимом оставленного имени.

3. Если первичные ключи сущностей различаются, то один из них объявить первичным, а другой — альтернативным ключом объединённой.

Если участие в связи одной или обеих сущностей не обязательно, то либо они действительно представляют различные объекты, либо одна из них идентифицирует набор необязательных атрибутов другой. В последнем случае лучше объединить их.

Шаг 8. Удаление избыточных связей.

Связь по своей сути — это информация о соответствии экземпляров сущностей. В идеале каждая связь концептуальной модели должна содержать уникальную информацию, т.е. такую, которую можно получить только из неё. Последний шаг процесса очистки модели от нежелательных элементов — обнаружение и удаление избыточных связей, содержащих информацию, которую можно получить через посредство других связей.

Пример.

Пусть пользователя интересуют отношения

«ПОСТАВЩИК заключил контракт на поставку ТОВАРа»

и «ПОСТАВЩИК выполнил поставку ТОВАРа».

Способы представления этих отношений в модели зависят от точки зрения пользователя и от правил организации. Рассмотрим несколько вариантов.

Вариант 1.

– Возможно заключение нескольких контрактов на поставки одного и того же вида товара.

– Возможно заключение нескольких контрактов с одним и тем же поставщиком.

– Каждый поставщик поставляет только те товары, которые включены в его контракты.

Если с точки зрения пользователя не видно, какой контракт обусловил конкретную поставку, то соответствующий фрагмент модели выглядит так, как на рис. 8.7. Нетрудно убедиться в том, что здесь нет избыточных связей. Так, из фактов «ПОСТАВЩИК X заключил КОНТРАКТ W», «ТОВАР Y включён в КОНТРАКТ W»

и «ТОВАР Y включён в ПОСТАВКУ Z» не следует утверждение «ПОСТАВЩИК X выполнил ПОСТАВКУ Z».

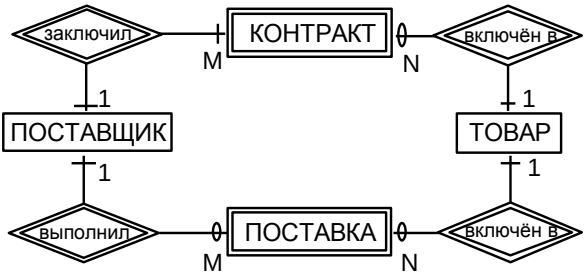


Рис. 8.7 — Вариант 1. Избыточных связей нет

Вариант 2.

– Невозможно заключение нескольких контрактов на поставки одного и того же вида товара.

– Возможно заключение нескольких контрактов с одним и тем же поставщиком.

– Каждый поставщик поставляет только те товары, которые включены в его контракты.

Изменение первого правила повлечёт только изменение спецификации мощности связи «ТОВАР включён в КОНТРАКТ» со стороны сущности ТОВАР (см. рис. 8.8).

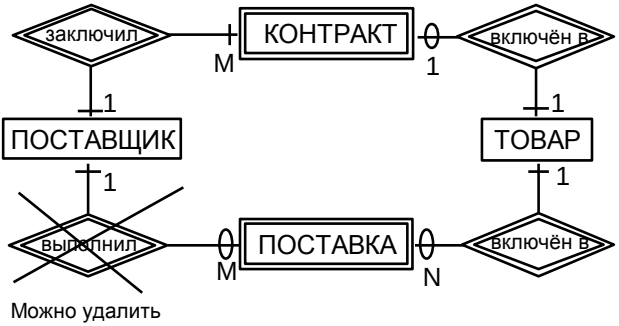


Рис. 8.8 — Вариант 2. Избыточная связь «выполнил»

Однако теперь из фактов «ПОСТАВЩИК X заключил КОНТРАКТ W», «ТОВАР Y включён в КОНТРАКТ W» и

«ТОВАР Y включён в ПОСТАВКу Z» следует утверждение «ПОСТАВЩИК X выполнил ПОСТАВКу Z».

Вариант 3.

– Возможно заключение нескольких контрактов на поставки одного и того же вида товара.

– Невозможно заключение нескольких контрактов с одним и тем же поставщиком.

В этом случае избыточной будет связь «ТОВАР включён в ПОСТАВКу».

Вариант 4.

– Невозможно заключение нескольких контрактов на поставки одного и того же вида товара.

– Невозможно заключение нескольких контрактов с одним и тем же поставщиком.

– Каждый поставщик поставляет только те товары, которые включены в его контракты.

Теперь избыточна одна (любая) из связей сущности ПОСТАВКА.

Вариант 5.

Пусть теперь пользователю важно знать, по какому контракту выполнена конкретная поставка. Фрагмент модели для первого варианта набора правил заключения контрактов показан на рис. 8.9.

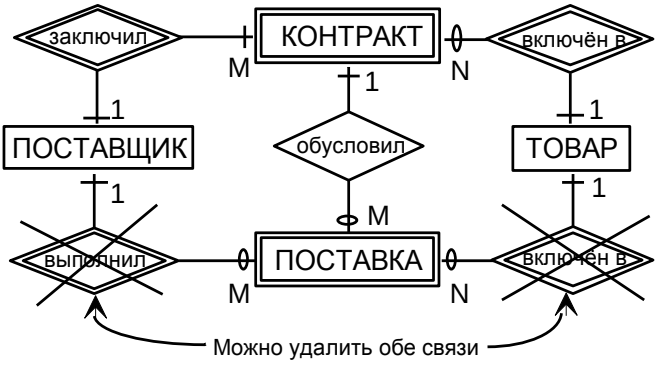


Рис. 8.9 — Вариант 5. Избыточные связи «выполнил» и «включён в»

Для любого варианта правил обе связи сущности ПОСТАВКА избыточны. Из фактов «ПОСТАВЩИК X заключил КОНТРАКТ W», «ТОВАР Y включён в КОНТРАКТ W» и «КОНТРАКТ W обусловил ПОСТАВКу Z» в любом случае следуют утверждения «ПОСТАВЩИК X выполнил ПОСТАВКу Z» и «ТОВАР Y включён в ПОСТАВКу Z».

Сохранение избыточных связей нежелательно. Их реализация в базе данных потребует создания процедур, обеспечивающих согласованность информации, содержащейся в различных связях.

Так, в последнем примере прямая информация о том, какой поставщик выполнил поставку, и какой товар в неё включён, содержится в связях «выполнил» и «включён в» сущности ПОСТАВКА. Косвенно (и независимо!) та же информация содержится и в связи «обусловил». Если прямые (избыточные) связи сохранены и реализованы в БД, то при регистрации поставки товара Y поставщиком X, выполненной по контракту Z, нужно проверить, действительно ли контракт Z заключён поставщиком X и предусматривает поставки товара Y.

Однако, обнаружив в модели избыточную связь, не следует её немедленно удалять. Прежде нужно выяснить, насколько часто пользователю требуется содержащаяся в ней информация.

Например, если пользователю часто приходится отбирать поставки конкретных поставщиков, то избыточную прямую связь «выполнил» (см. рис. 6.13) следует сохранить. В противном случае придётся сначала выбрать все контракты заданного поставщика, затем все товары, включённые в эти контракты и только после этого — все поставки, в которые включены отобранные товары.

Резюмируем сказанное об избыточных связях.

По отношению к каждой связи модели необходимо задать вопрос:

«Не представляет ли эта связь отношение, подразумеваемое другими связями?»

Если ответ «ДА», то нужно оценить желательность сохранения связи в модели по критерию эффективности обработки запросов пользователя. Если связь решено сохранить, то она должна быть помечена в словаре данных как избыточная.

Шаг 9. Документирование очищенной модели.

Результатом этапа 2.1 является упрощенная локальная концептуальная модель данных пользователя. Она адаптирована к требованиям РМД, т.е. из неё устранены все элементы, реализация которых в среде реляционной СУБД затруднительна.

Документирование очищенной модели подразумевает создание новой версии ER-диаграммы и модификацию словаря данных. В словарь должны быть добавлены описания новых сущностей с указанием соответствия элементам неадаптированной модели и описания новых связей. Все сохранённые избыточные связи должны быть помечены.

Этап 2.2. Определение набора отношений на основе очищенной концептуальной модели

На этом этапе каждой сущности очищенной локальной концептуальной модели должно быть сопоставлено отношение РМД, для каждой связи должен быть определён родитель и потомок и сопоставлен внешний ключ. Результатом этапа является набор определений схем отношений — логическая схема данных локального пользователя.

Для создания определений схем используется какой-либо текстовый или графический реляционный язык определения данных. В настоящее время имеется большой выбор как текстовых, так и графических языков. Наиболее предпочтительны графические языки, например, IDEF1X. Они обеспечивают такую же точность описания схем данных, как и текстовые, но определение схемы, выполненное средствами графического языка, гораздо более компактно и наглядно, чем текстовое. Многие из них включены в состав различных CASE-средств.

Заметим ещё, что если для описания логической схемы используется графический язык, то эту работу можно (и следует) совместить с очисткой концептуальной модели.

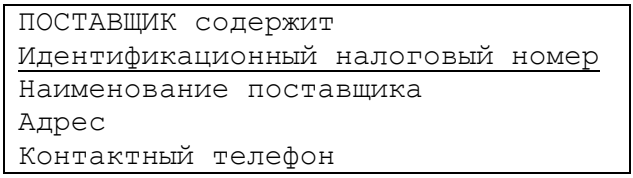
Шаг 1. Определение отношений для стержневых (сильных) сущностей.

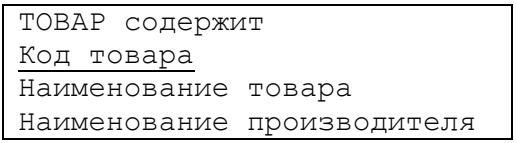
Для каждой сильной сущности очищенной концептуальной модели создаётся отношение. Ему присваивается уникальное имя. Схема отношения включает все атрибуты сущности. Имена

отношения и его атрибутов должны быть уникальны в пределах локальной модели. Они могут не совпадать с именами сущности и её атрибутов.

Первичным ключом отношения назначается первичный ключ сущности. Альтернативные ключи сущности (если они есть) помечаются как альтернативные ключи отношения.

В качестве примера рассмотрим сильные сущности ПОСТАВЩИК и ТОВАР (см. рис. 6.14).





Соответствующие отношения могут выглядеть так[6]:

```
Supplier(Inn, Name_S, Addr, Tel)
PRIMARY KEY Inn
ALTERNATE KEY Name_S;

Goods(Code_G, Name_G, Manufact)
PRIMARY KEY Code_G;
```

Шаг 2. Определение отношений для слабых сущностей.

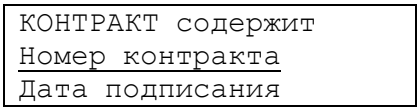
Процедура определения отношения для слабой сущности отличается от описанной выше только тем, что в его схему, кроме собственных атрибутов сущности, включается копия первичного ключа сущности-владельца. Это внешний ключ, реализующий в РМД связь слабой сущности и сущности-владельца.

[6] Здесь и далее в примерах используется формализованный язык, подобный известному языку DBDL (Data Base Definition Language). Это не означает, что именно его и нужно использовать в реальном проекте. Мы выбрали его исключительно из-за интуитивной понятности синтаксиса.

Он обязательно войдёт в состав хотя бы одного потенциального ключа отношения.

Если первичный ключ слабой сущности не был определён на этапе 1.5, то его следует определить сейчас.

Например, слабая сущность КОНТРАКТ (см. рис. 6.14) в концептуальной модели представлена так:



В логической модели ей будет соответствовать следующее отношение:

```
Contract(Contr_No, Inn, Code_G, Date)
PRIMARY KEY(Contr_No)
ALTERNATE KEY (Inn, Code_G)
FOREIGN KEY Inn REFERENCES Supplier
FOREIGN KEY Code_G REFERENCES Goods;
```

Слабая сущность может зависеть от нескольких сущностей-владельцев (в нашем примере — от двух). Соответствующее ей отношение должно содержать копии первичных ключей всех владельцев (внешние ключи). Каждый внешний ключ войдёт в состав какого-либо потенциального ключа отношения. Такие потенциальные ключи невозможно определить на этапе концептуального моделирования. Все они должны быть выявлены и описаны здесь.

Шаг 3. Реализация связей типа 1:1 и 1:M.

В реляционной модели данных такие связи реализуются единственным способом — путём помещения копии первичного ключа родительского отношения в состав атрибутов отношения-потомка. Здесь эта копия является внешним ключом. Его значение в некотором кортеже отношения-потомка — это ссылка на соответствующий родительский кортеж.

Родительской в связи 1:M является сущность, один экземпляр которой может участвовать в нескольких экземплярах связи. В ER-диаграмме родитель показан на «единичном» конце связи.

В связях типа 1:1 спецификация родителя и потомка зависит от степени участия сущностей. Пусть E1 и E2 — сущности, состоящие в связи 1:1.

Если участие сущности E2 обязательное (каждому экземпляру E2 должен соответствовать экземпляр E1), а E1 — необязательное (возможно существование экземпляра E1, которому не соответствует никакой экземпляр E2), то родителем в связи является E1.

Если участие обеих сущностей необязательное, то родителем можно выбрать любую из них.

Если участие обеих сущностей обязательное, то выбор также произволен. Однако в этом случае следует подумать об объединении сущностей. Если принято решение объединить их, то нужно создать единое отношение, содержащее все атрибуты обеих сущностей. Первичным ключом отношения назначается первичный ключ одной из них, а первичный ключ другой объявляется альтернативным.

Шаг 4. Реализация связи «супертип–подтип».

Связь «супертип–подтип» суть множество бинарных связей 1:1. Родитель в каждой связи — супертип. Специфика этого типа связи в том, что *каждый экземпляр любого подтипа является экземпляром супертипа*.

Можно указать три типовых варианта реализации таких связей.

Вариант 1. Создание отдельных отношений для супертипа и каждого из подтипов. Копии первичного ключа супертипа включаются в схему каждого отношения-подтипа как внешние ключи. Этот вариант может иметь преимущества, если подтипы участвуют в общих связях и, кроме того, каждый подтип имеет собственные, специфичные только для него, связи.

Вариант 2. Создание отдельных отношений для каждого подтипа. Копии всех атрибутов супертипа включаются в схему каждого отношения-подтипа. Отдельное отношение для супертипа не создаётся. Этот вариант хорош, если подтипы не имеют общих связей.

Вариант 3. Создание объединённого отношения для супертипа и всех подтипов. Такое решение может быть разумным, если подтипы не имеют специфичных связей.

Возможны и комбинированные варианты. Выбор подходящего зависит от специфики конкретных ограничений, накладываемых на реализуемую связь.

Шаг 5. Документирование отношений и внешних ключей.

Определения созданных отношений и их связей (внешних ключей) должны быть сохранены в рабочей документации виде текста, а лучше — в виде полноатрибутной диаграммы. Словарь данных должен быть пополнен сведениями о любых новых ключевых атрибутах, определённых на этапе 2.2.

Этап 2.3. Проверка нормализованности логической модели

В базе данных должны поддерживаться все ограничения целостности, обусловленные деловым регламентом, в том числе функциональные и многозначные зависимости атрибутов. Если зависимости не поддерживаются, то невозможно гарантировать согласованность состояния БД. Межатрибутные зависимости можно поддерживать как на уровне процедур базы данных, так и на уровне СУБД. Первую стратегию приходится использовать, если целевая СУБД не поддерживает первичные и внешние ключи. В настоящее время такие СУБД встречаются крайне редко. Любая современная реляционная СУБД может обеспечить поддержку зависимостей, если логическая схема БД удовлетворяет требованиям нормализованности.

Каждое отношение в логической модели данных пользователя должно находиться в 4НФ. При этом все межатрибутные зависимости должны быть следствиями определений потенциальных и внешних ключей.

Требования нормализованности — это ограничения модели данных, обеспечивающие исключение неконтролируемого дублирования данных и аномалий обновления, обусловливающих несогласованность данных.

Нормализованная модель гарантирует размещение данных в отношениях в соответствии с их функциональными зависимо-

стями, а также поддержку функциональных зависимостей атрибутов на уровне потенциальных и внешних ключей.

Схема каждого отношения включает детерминант ФЗ и все его зависимые части и не содержит атрибутов, не зависящих функционально от детерминанта. Если в схеме несколько детерминантов, то сказанное относится к каждому из них. В этом случае любые два детерминанта состоят во взаимно однозначной ФЗ. Это соответствует природе данных. Схема отношения, находящегося в 4НФ, всегда суть набор свойств объекта ПО, выделяемого пользователем явно или неявно.

Нормализованная модель оказывается весьма гибкой. В неё легко вносить изменения, связанные как с изменением информационных потребностей пользователя, так и с требованиями реализации физических структур и приложений.

Например, в связи с расширением функций пользователя в его поле зрения попадают новые объекты. Они как-то связаны с теми, которые уже представлены в логической модели. Чтобы обеспечить поддержку новых функций, достаточно в существующую логическую модель данных пользователя добавить новые отношения и установить их связи с уже существующими.

Другой пример. В процессе эксплуатации системы выяснилось, что запросы определённого типа обрабатываются недопустимо долго. Эффективность обработки можно увеличить, соединив несколько таблиц базы данных в одну (выполнив денормализацию). При этом придётся обеспечить процедурную поддержку функциональных зависимостей, которые в настоящее время поддерживаются ядром СУБД. Выяснить, какие это зависимости и какие именно процедуры следует создать, оценить влияние денормализации на эффективность обработки запросов других типов невозможно, не имея нормализованной модели.

Для того чтобы создать нормализованную модель, необходимо достичь глубокого понимания природы и назначения данных. Изложенные выше приёмы методологии направлены именно на это. В идеальном случае результатом этапа 2.2 должен быть набор отношений, находящихся в 4НФ. Однако идеальных аналитиков не бывает. Цель этапа 2.3 — выявить ошибки структурирования данных, допущенные на предыдущих этапах. Для достижения цели нужно, опираясь на деловой регламент, прове-

рить корректность определения схемы каждого отношения модели. Ниже описаны шаги проверки одного отношения.

Шаг 1. Проверка требования 1НФ.

Перебрать все атрибуты схемы отношения, задавая вопрос:

«Действительно ли этот атрибут в каждом кортеже отношения принимает скалярное значение?»

Если обнаружен многозначный атрибут, то для него необходимо определить собственное отношение, как для слабой сущности. Обнаруженный в схеме композитный атрибут следует заменить набором его компонентов.

Дальнейшие проверки (шаги 2 — 5) выполняются для каждого потенциального ключа отношения. В описаниях проверок будем использовать следующие обозначения:

`R(PK, A)` — проверяемое отношение;

`PK` — проверяемый потенциальный ключ;

`A` — подмножество атрибутов, не входящих в состав потенциального ключа `PK`.

Шаг 2. Проверка корректности определения потенциального ключа.

1) Для составного ключа убедиться в том, что он не содержит детерминантов, функционально определяющих его компоненты. Потенциальный ключ может состоять только из взаимно независимых атрибутов.

2) Перебрать все атрибуты подмножества `A`, задавая вопрос:

«Действительно ли этот атрибут функционально зависит от PK?»

Если хотя бы для одного атрибута ответ «НЕТ», то нужно исследовать две возможности:

– неверно определён потенциальный ключ отношения;

– в схему включён атрибут другого отношения, возможно, отсутствующего в модели.

В обоих случаях придётся вернуться к более ранним этапам анализа.

Шаг 3. Проверка неприводимости зависимости от PK.

Для каждого подмножества $S \subset PK$ убедиться в том, что не существует такого подмножества $C \subseteq A$, что $S \rightarrow C$.

Если обнаружено S → C и C ≠ A, то отношение следует декомпозировать, т.е. представить в виде двух отношений:

R1(PK, A1), R2(S, C), где A1 = A – C.

В отношении R2 подмножество S является первичным ключом. Его копия в отношении R1 (в составе PK) — внешний ключ.

Если обнаружено подмножество S ⊂ PK, функционально определяющее *каждый* атрибут из A (C = A), то исследуемый потенциальный ключ избыточен. Из него нужно удалить все атрибуты, не принадлежащие S, включив их в A.

Шаг 4. Проверка взаимной независимости неключевых атрибутов.

Для каждого подмножества D ⊂ A, не являющегося потенциальным ключом[7], убедиться в том, что не существует такого подмножества C ⊂ A, что D → C.

Если обнаружено D → C, то отношение следует декомпозировать:

R1(PK, A1), R2(D, C), где A1 = A – C.

В отношении R2 подмножество D является первичным ключом. Его копия в отношении R1 (в составе A1) — внешний ключ.

Шаг 5. Проверка независимости компонентов потенциального ключа от неключевых атрибутов.

Для каждого подмножества D ⊂ A, не являющегося потенциальным ключом, убедиться в том, что не существует такого подмножества S ⊆ PK, что D → S.

Если обнаружена такая (неприводимая) ФЗ, то подмножество D является частью потенциального ключа PK1, не выявленного ранее. Он включает все атрибуты PK, не принадлежащие S, и все атрибуты D:

PK1 = {PK – S, D}.

В этом случае можно выполнить следующую декомпозицию:

[7] Все потенциальные ключи отношения, отличные от PK, содержатся в A.

R1(PK1, A1), R2(D, S), где A1 = A – D.

В отношении R2 подмножество D является первичным ключом. Его копия в отношении R1 (в составе PK1) — внешний ключ.

В частном случае S = PK декомпозицию выполнять не нужно, т.к. подмножество D является новым потенциальным ключом.

Подчеркнём, что все описанные здесь декомпозиции выполняются *без потерь информации*. Исходное отношение R является естественным соединением отношений R1 и R2. Однако если отношение имеет несколько потенциальных ключей, то некоторые из них могут быть разрушены. Вследствие этого будут утеряны функциональные зависимости, в которых они выступают детерминантами. Эти зависимости придётся поддерживать на уровне приложений.

Например, вследствие декомпозиции с целью устранения зависимости части потенциального ключа от неключевого атрибута (см. проверку 3) будет разрушен потенциальный ключ PK и утеряны все те ФЗ, которые не следуют из объявленных в результате декомпозиции зависимостей PK1→ A1 и D→ S.

С другой стороны, если проектировщик решит отказаться от декомпозиции, то ему придётся поддерживать на уровне приложений сохранённые в отношении «запрещённые» ФЗ.

Разрушение детерминантов функциональных зависимостей при декомпозиции отношений может быть признаком наличия ошибок в логической модели. Либо концептуальная модель некорректно отображена в набор отношений, либо ошибки были допущены ещё на фазе концептуального моделирования. Однако ошибки может и не быть. В этом случае следует выполнить все необходимые декомпозиции, создав систему нормализованных до НФБК отношений, и описать в словаре данных все существующие функциональные зависимости, не следующие из определений потенциальных ключей.

Шаг 6. Проверка требования 4НФ.

Если отношение находится в НФБК и его схема содержит не менее трёх взаимно независимых (функционально) атрибутов, то в нём может существовать нетривиальная многозначная

зависимость[8]. Взаимно независимые атрибуты могут входить в схему одного отношения либо как компоненты потенциального ключа, либо как зависимые части ФЗ с общим детерминантом. В последнем случае функционально независимые атрибуты связаны логически, являются характеристиками одного объекта.

Наличие пары многозначных зависимостей `A` $\rightarrow\!\!\rightarrow$ `B|C` означает, что в отношении представлены две различных логических связи между атрибутами, не связанными функционально и не зависящими от потенциальных ключей. Следовательно, отношение, находящееся в НФБК, может содержать многозначные зависимости только внутри потенциального ключа, включающего не менее трёх атрибутов.

Вывод: проверку требования 4НФ для отношения, находящегося в НФБК, нужно выполнять, если оно содержит потенциальный ключ, составленный из трёх или более атрибутов.

Если в отношении `R`, содержащем подмножества атрибутов `A`, `B`, `C`, обнаружена пара многозначных зависимостей `A` $\rightarrow\!\!\rightarrow$ `B|C`, то его нужно представить в виде двух отношений: `R1(A, B), R2(A, C).`

Вообще говоря, возможность появления в логической модели отношения, содержащего многозначные зависимости, должна быть исключена ещё на этапе очистки концептуальной модели, при устранении многозначных атрибутов сущностей. Тем не менее, эта проверка должна быть выполнена.

Этап 2.4. Проверка исполнимости транзакций.

Логическая модель должна описывать все данные, необходимые для выполнения транзакций пользователя, и только эти данные. На этом этапе проверяется полнота и неизбыточность модели в указанном смысле.

Для того чтобы выполнить проверку, необходимо иметь описания всех транзакций. Следует попытаться выполнить каждую транзакцию вручную, используя диаграмму модели и словарь данных. На диаграмму нужно нанести карту выполнения транзакций, которая будет наглядно представлять области, задействованные в той или иной транзакции.

---

[8] Точнее говоря, связанная пара многозначных зависимостей.

Пример. Транзакции, выполняемые сотрудником, обрабатывающим поставки товаров.

Т1. Ввод сведений о поставщике товаров.

Д1. Если вводимое значение атрибута ИНН не уникально, то ввод новой записи запретить. Выдать сообщение об ошибке. Операцию завершить.

Д2. В противном случае выполнить ввод нового кортежа.

Проверить, что для каждого вводимого элемента сведений о поставщике в отношении ПОСТАВЩИК имеется соответствующий атрибут.

Т2. Удаление сведений о поставщике товаров, заданном значением ИНН.

Д1. Найти кортеж отношения ПОСТАВЩИК, содержащий заданное значение атрибута ИНН. Если такого кортежа нет, то выдать сообщение об ошибке. Операцию завершить.

Д2. В противном случае выполнить поиск кортежей отношения ПОСТАВКА, содержащих заданное значение атрибута ИНН. Если такие кортежи есть, то выдать сообщение об ошибке. Операцию завершить.

Д3. В противном случае удалить найденный кортеж отношения ПОСТАВЩИК.

Т3. Ввод сведений о поставке.

Д1. Найти в отношении ТОВАР кортеж, содержащий введённое значение атрибута Артикул. Если кортеж не найден, то выдать сообщение об ошибке. Транзакцию отменить. Иначе Д2.

Д2. Найти в отношении ПОСТАВЩИК кортеж, содержащий введённое значение атрибута ИНН. Если кортеж найден, то Д5, иначе Д3.

Д3. Выдать сообщение об ошибке. Предложить пользователю ввести сведения о новом поставщике. Если пользователь отказался, то транзакцию отменить, иначе Д4.

Д4. Выполнить транзакцию Т1. Если транзакция Т1 завершилась успешно, то Д5, иначе Д6.

Д5. Если введённый набор значений атрибутов (ИНН, Артикул, Дата поставки) уникален, то выполнить операцию добавления кортежа в отношение ПОСТАВКА. Транзакцию завершить. Иначе Д6.

Д6. Транзакцию отменить.

Проверить, что для каждого вводимого элемента сведений о поставке в отношении ПОСТАВКА имеется соответствующий атрибут.

Карта выполнения этих транзакций приведена на рис. 8.10.

Если все транзакции выполнить удалось, то модель полная. В противном случае она содержит ошибки. Возможно, утеряна какая-то сущность, связь или атрибут. Если какие-то отношения, связи или атрибуты не использовались ни в одной из транзакций, значит модель избыточна. Следует подумать о целесообразности представления этой информации.

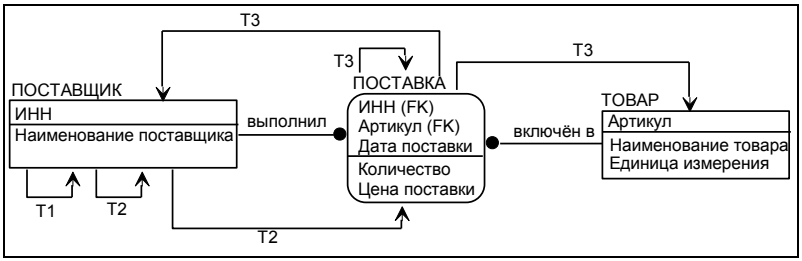


Рис. 8.10 — Карта выполнения транзакций

Этап 2.5. Создание окончательной диаграммы локального представления.

В ходе выполнения проверок требований нормализации и выполнимости транзакций принимаются окончательные решения об отношениях, атрибутах и ключевых группах. Теперь эти решения должны быть зафиксированы в диаграмме и текстовых документах модели — в словаре данных, примечаниях и т.п.

Этап 2.6. Определение ограничений целостности данных.

Ограничения целостности данных необходимы для предотвращения ввода в БД противоречивых данных. Они представ-

ляют в модели данных деловой регламент предприятия. Некоторые правила делового регламента могут быть описаны в терминах внутренних ограничений целостности РМД. Это правила, относящиеся к типам данных, допустимым значениям реквизитов, идентификаторам объектов и некоторым свойствам отношений объектов. Такие правила может поддерживать реляционная СУБД собственными средствами[9]. Другие правила должны поддерживаться специально разработанными процедурами.

Полная модель данных пользователя на логическом уровне должна включать

– ограничения целостности доменов;
– ограничения целостности атрибутов;
– спецификации допустимости/недопустимости неопределённых значений атрибутов;
– ограничения целостности сущностей;
– ограничения ссылочной целостности;
– требования делового регламента предприятия, не представленные ранее перечисленными ограничениями.

Шаг 1. Определение ограничений для доменов и атрибутов.

Эти ограничения определяются на этапе 1.4. Здесь необходимо

– убедиться в том, что каждый атрибут представленный в диаграмме, сопоставлен одному из доменов, описанных в словаре данных.
– записать определения доменов на реляционном ЯОД (см. п. 3.5.1).

Если обнаруживается атрибут, не связанный ни с каким из доменов, для него следует определить домен и внести определение в словарь данных. Кроме того, следует убедиться в том, что каждый атрибут действительно не может принимать значений, не принадлежащих его домену.

Шаг 2. Определение допустимости `NULL`-значений.

Некоторые атрибуты, кроме значений из своих доменов, могут принимать неопределённые (`NULL`) значения. Для других атрибутов это недопустимо в принципе. Таковы, например, все

[9] К сожалению, не каждая представленная на рынке СУБД поддерживает внутренние ограничения целостности РМД.

атрибуты первичных ключей отношений. Поэтому для каждого атрибута в словаре данных должно быть зафиксировано соответствующее ограничение NULL ALLOWED/NULL NOT ALLOWED. В основном эта работа выполняется на этапе 1.3. Здесь нужно убедиться в том, что для каждого атрибута действительно установлено это ограничение.

Шаг 3. Определение ограничений целостности сущности.

Ограничения целостности сущности устанавливаются на этапах 1.5 и 2.2 при определении первичных ключей. Здесь нужно проверить, что

– для каждого отношения определён первичный ключ;

– ни один атрибут первичного ключа не может принимать неопределённое значение.

Шаг 4. Определение ограничений ссылочной целостности.

Внешние ключи, реализующие связи отношений, определяются на этапе 2.2. Здесь нужно убедиться в том, что в схеме каждого отношения-потомка в какой-либо связи содержится *копия* первичного ключа родительского отношения. Каждый атрибут внешнего ключа должен быть определён на домене соответствующего компонента родительского ключа.

Требование ссылочной целостности состоит в следующем:

> В любой момент времени каждое определённое значение внешнего ключа должно принадлежать множеству значений ссылочного первичного ключа, содержащихся в существующих кортежах родительского отношения.

В модели данных должны быть определены правила поддержки этого ограничения.

Во-первых, нужно выяснить, могут ли компоненты внешнего ключа принимать неопределённые значения. Это определяется степенью участия отношения-потомка в связи. Если участие *обязательное* (каждый кортеж потомка должен ссылаться на какой-либо родительский кортеж), то каждый компонент внешнего ключа должен получить спецификацию NULL NOT ALLOWED. Если же участие потомка в связи *частичное* (могут встречаться кортежи потомка, не связанные с кортежами родительского отношения), то некоторые или все компоненты внешнего ключа должны иметь спецификации NULL ALLOWED.

Вторая группа проблем связана с операциями, требующими обновления множеств значений первичного ключа родительского отношения или внешнего ключа. Требование ссылочной целостности может быть нарушено при попытке

– добавления кортежа в отношение-потомок,

– изменения значения внешнего ключа в существующем кортеже потомка,

– удаления кортежа родительского отношения,

– изменения значения первичного ключа в существующем кортеже родителя[10].

В первом и втором случаях необходимо проверить, существует ли вводимое значение внешнего ключа во множестве значений первичного ключа родителя. Операция обновления разрешается при успешном завершении проверки. Если целевая СУБД поддерживает ссылочную целостность, то эта проверка будет выполняться автоматически. Никаких специальных процедур не требуется. Достаточно определить внешний ключ в схеме потомка.

В третьем и четвёртом случаях необходимо проверить, существуют ли значения внешнего ключа, ссылающиеся на удаляемое или изменяемое значение родительского ключа. Если ссылок нет, то операция выполняется. В противном случае СУБД должна либо отменить запрошенную операцию, либо выполнить её и какие-то компенсирующие обновления ссылающихся кортежей потомка. Возможны различные стратегии поддержки ссылочной целостности. Современные СУБД поддерживают следующий набор вариантов правил ссылочной целостности для операции удаления родительского кортежа (правила `ON DELETE`):

– `NO ACTION` — не выполнять операцию, если на удаляемое значение родительского ключа есть ссылки;

– `CASCADE` — распространить операцию удаления родительского кортежа на все кортежи потомка, ссылающиеся на удаляемый родительский кортеж;

---

[10] При попытках удаления кортежа потомка и добавления кортежа родителя ссылочная целостность не может быть нарушена, но могут нарушаться ограничения на степень участия в связи родительского отношения (см. шаг 5).

– SET DEFAULT — все ссылки на удаляемый кортеж заменить значением по умолчанию;

– SET NULL — все ссылки на удаляемый родительский кортеж заменить неопределёнными значениями;

– NO CHECK — удалить родительский кортеж, не выполняя проверки ссылочной целостности.

Подобные правила применяются и для операции обновления значения родительского ключа (правила ON UPDATE).

Выбор правила в каждом конкретном случае определяется деловым регламентом и соображениями здравого смысла. Очевидно, правила SET DEFAULT использовать нельзя, если для компонентов внешнего ключа не заданы значения по умолчанию. Правила SET NULL нельзя использовать, если участие потомка в связи обязательное. Использование правила ON DELETE CASCADE может привести к неожиданному для пользователя удалению кортежей нескольких отношений. Применение правила ON UPDATE CASCADE катастрофических последствий не обусловит, т.к. кортежи-потомки сохраняются в БД, но может не соответствовать требованиям предприятия. Правила NO ACTION могут оказаться чересчур консервативными. Наконец, правила NO CHECK означают отказ от сервиса поддержки ссылочной целостности, предоставляемого СУБД. В этом случае должны быть реализованы специальные стратегии поддержки ссылочной целостности.

Итак, на шаге 4 для каждого внешнего ключа должны быть заданы подходящие спецификации NULL ALLOWED/NULL NOT ALLOWED и правила ON UPDATE и ON DELETE. Они являются частью определения внешнего ключа в схеме отношения. Если в определении внешнего ключа использовано правило NO CHECK, то в модели должна быть описана необходимая процедура поддержки ссылочной целостности.

Замечание. До настоящего времени существуют «СУБД», не поддерживающие ссылочную целостность. Если предполагается реализация проекта в среде такой СУБД, то необходимо описать в модели все процедуры поддержки ссылочной целостности. Они должны быть реализованы на этапе физического проектирования как процедуры базы данных.

Шаг 5. Определение ограничений предприятия.

Как уже упоминалось, далеко не все правила делового регламента можно представить в модели как ограничения целостности доменов, атрибутов, сущностей и ссылок. Такие правила следует описать в разделе «Требования предприятия» словаря данных. Некоторые из них задают степень участия отношения в связи или точное значение мощности связи. Их нужно показать на диаграмме. Эти ограничения определяют способы выполнения транзакций.

Например, фрагмент диаграммы, приведённый на рис. 8.11, утверждает, что транзакция добавления кортежа в отношение `E1` вовлекает отношение `E2`. Невозможно добавить родительский кортеж, не добавив *в той же транзакции* хотя бы один связанный с ним кортеж `E2`.

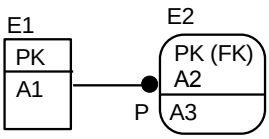


Рис. 8.11 — Обязательное участие родителя в связи

Транзакция удаления кортежа `E2` также вовлекает отношение `E1`. Удаление кортежа `E2` возможно без обновления отношения `E1`, если он не единственный потомок соответствующего кортежа `E1`. В противном случае либо операцию удаления следует запретить, либо *в той же транзакции* удалить и родительский кортеж.

Шаг 6. Документирование ограничений целостности.

Все установленные ограничения целостности должны быть описаны в диаграмме и в словаре данных. Они потребуются на фазе физического проектирования.

Этап 2.7. Обсуждение локальной логической модели с конечным пользователем.

Цель этого этапа — убедиться в том, что созданная модель точно отражает представления пользователя о предметной области приложения.

Обсуждение проводится тогда, когда аналитик принимает решение о завершении моделирования. К этому моменту должна быть полностью готова вся проектная документация.

Модель в обязательном порядке содержит два документа:

– схему данных (диаграмму), содержащую определения отношений, и

– словарь данных, содержащий описания смысла элементов и агрегатов данных, ограничения доменов, правила ссылочной целостности, описания транзакций пользователя и другие компоненты модели, не отражённые в схеме данных.

К этим документам могут прилагаться дополнительные материалы, содержащие любые сведения о работе пользователя, которые могут помочь правильно понять модель.

Модель предлагается пользователю для изучения. Согласуются условия обсуждения: дата, время, место, продолжительность.

Цель обсуждения с точки зрения пользователя — убедиться в том, что модель точно отражает *его* представления об *его* служебных функциях и способах их исполнения. Цель с точки зрения аналитика — доказать ему это.

8.2 Построение и проверка глобальной логической модели

Цель этого этапа — объединение локальных логических моделей в единую модель данных предприятия. Эта модель должна быть такой, чтобы из неё можно было вывести каждое локальное представление посредством формальных преобразований.

Основная проблема этого этапа обусловлена тем, что каждая отдельная локальная модель представляет взгляд на предприятие с точки зрения отдельного пользователя. Она описывает не данные и функции предприятия, а представления работника предприятия об этих данных и функциях. Функции работников предприятия взаимосвязаны. Поэтому данные, с которыми они работают, перекрываются. Таким образом, каждая отдельная локальная модель является *неполной*, а вся совокупность локальных моделей — *избыточной*, поскольку различные модели

описывают одни и те же данные, но с разных точек зрения. На этом этапе должно быть создано целостное представление о предметной области системы, не зависящее от конкретных пользователей, бизнес-процессов и приложений.

Глобальная модель должна быть полной и неизбыточной. Она должна отображать требования некоего абстрактного пользователя, участвующего *во всех* бизнес-процессах, исполняющего обязанности *всех* конечных пользователей проектируемой системы.

Этап 3.1. Слияние локальных моделей в глобальную логическую модель.

Слияние локальных моделей — это процесс их сравнительного анализа с целью устранения любых противоречий и последующего объединения в целостную модель. Это относительно простая задача, если система содержит два-три локальных представления. Если же их хотя бы пять, то требуется систематический подход. Необходимое условие успеха — корректность и полнота каждой локальной модели.

Шаг 1. Анализ имён сущностей и их потенциальных ключей.

Цель — обнаружить синонимы и омонимы в различных локальных моделях. Для этого нужно сравнить имена и описания сущностей. Могут обнаружиться конфликты двух типов:

– обнаружены одноимённые сущности, описания которых различаются по смыслу;

– обнаружены близкие по смыслу описания различных имён.

Для разрешения конфликта нужно сравнить составы атрибутов конфликтных сущностей, в частности, потенциальные ключи.

Шаг 2. Анализ имён связей.

На этом шаге нужно выявить связи, имеющие различные имена, но одинаковый смысл. Участники таких связей являются полными аналогами.

С другой стороны, между парами сущностей–аналогов могут быть обнаружены связи, имеющие различный смысл, но совпадающие имена.

Шаг 3. Слияние общих сущностей.

Ситуация 1. В нескольких локальных моделях обнаружены сущности с одинаковыми именами и первичными ключами. Эти сущности представляют один объект ПО, рассмотренный с разных сторон. В глобальной модели они должны быть слиты в одну сущность. В состав её атрибутов включаются все атрибуты сливаемых сущностей. Дубликаты исключаются.

Ситуация 2. В нескольких локальных моделях обнаружены сущности с одинаковыми именами и различными первичными ключами. Если имена имеют одинаковый смысл, то эти сущности должны иметь совпадающие потенциальные ключи. В глобальной модели их необходимо слить. Один из потенциальных ключей объединённой сущности следует назначить первичным, а остальные — альтернативными.

Ситуация 3. В нескольких локальных моделях обнаружены сущности с близкими по смыслу, но не совпадающими именами. Если они имеют сходный состав атрибутов и совпадающие потенциальные ключи или участвуют в аналогичных связях, то их следует слить.

При слиянии возможны конфликты.

А) В одной из моделей атрибут (например, Адрес) представлен как простой, а в другой — как составной. Способ представления этого атрибута в глобальной модели следует согласовать с пользователями конфликтующих представлений. Если одному из них действительно необходимо обрабатывать элементы структуры, то конфликтный атрибут нужно рассматривать как составной.

Б) В одной из локальных моделей атрибут описан как производный, а в другой — как первичный. В глобальной модели его следует считать производным.

В) Одноимённые атрибуты нескольких локальных моделей определены на различных доменах. В этом случае нужно уточнить смысл имени в различных представлениях. Если он совпадает, то следует согласовать определения доменов в различных представлениях. Атрибут следует представить в объединённой сущности в единственном экземпляре, определив его на согласованном домене. Если же одноимённые атрибуты имеют раз-

личающийся смысл, то необходимо уточнить их имена и под новыми именами включить в состав объединённой сущности.

Шаг 4. Включение уникальных сущностей.

Все сущности всех локальных моделей, не участвовавшие в слияниях, должны быть включены в глобальную модель без каких-либо изменений.

Шаг 5. Слияние общих связей.

Сливаться могут только связи, в которых участвуют сливаемые сущности. На этом шаге нужно проанализировать имена и смысл каждой такой связи из всех представлений. Сливаются связи, имеющие одинаковый смысл. Если они имеют различающиеся имена, то из них нужно выбрать наиболее информативное. Перед слиянием связей необходимо согласовать все различия в описаниях их свойств — степени участия и кардинальности.

Шаг 6. Включение уникальных связей.

Все связи, не участвовавшие в слияниях, включаются в глобальную модель без изменений.

Шаг 7. Выявление пропущенных сущностей и связей.

Это одна из труднейших задач. В ряде случаев вся совокупность локальных моделей может содержать не все сущности и связи, которые должны быть в глобальной модели.

Можно минимизировать вероятность возникновения такой ситуации. Для этого, беседуя с пользователем, нужно просить его изложить своё мнение об объектах и связях, которые могут существовать в других представлениях. Разумеется, не стоит задавать пользователю такие вопросы «в лоб». Он не мыслит в категориях объектов и связей. Но он имеет представления о работе и функциях других пользователей, и может рассказать об этом.

Выполняя анализ атрибутов сущностей локального представления, нужно пытаться выделить ссылки на сущности из других локальных представлений — внешние ключи. При этом следует иметь в виду, что ссылка может вести не на первичный ключ, и даже не на альтернативный, но на неключевой атрибут.

По-видимому, самый лучший подход к решению задачи обнаружения пропущенных сущностей и связей — это анализ функциональных зависимостей на полном множестве атрибутов

глобальной модели. Цель анализа — убедиться в том, что все существующие ФЗ представлены в глобальной модели, т.е. следуют из определений потенциальных ключей отношений. Не представленная ФЗ может указывать на пропущенную связь или сущность. Следует иметь в виду, что анализ ФЗ — работа сложная и трудоёмкая, однако отказываться от неё только на этом основании не следует.

Шаг 8. Проверка корректности внешних ключей.

В ходе слияния сущностей и связей могли измениться первичные ключи отношений. На этом шаге следует проверить, все ли внешние ключи соответствуют первичным ключам родителей.

Шаг 9. Проверка ограничений целостности.

На этом шаге нужно убедиться в том, что установленные для глобальной модели ограничения целостности не противоречат ограничениям локальных моделей. Все возникающие конфликты должны быть устранены. Крайне желательно участие пользователей в этой работе.

Шаг 10. Создание предварительной диаграммы глобальной модели.

Предварительный вариант диаграммы глобальной модели создаётся на основе диаграмм локальных моделей с учётом слияний сущностей и связей.

Шаг 11. Обновление документации.

Модель состоит из множества мелких деталей, содержащихся в документации. Удержать их все в голове человек не в состоянии. Модель изменяется в процессе анализа. Любые изменения, вносимые в процессе создания глобальной модели, должны *немедленно* отображаться в словаре данных и других документах. Если аналитик пренебрегает этим требованием, то некорректность модели гарантирована. Гарантирован также крах проекта.

Этап 3.2. Проверка глобальной логической модели

На этом этапе выполняются те же работы, что и на этапах 2.3, 2.4, при проверках локальных логических моделей.

Этап 3.3. Проверка возможности расширения модели

Информационные потребности предприятия с течением времени изменяются. Поскольку СБД создаётся не на один день, она должна адаптироваться к этим изменениям. Поскольку глобальная логическая модель данных лежит в основе СБД, адаптивность системы напрямую зависит от возможности модификации модели без коренной переработки существующих приложений. Никаких конкретных рекомендаций и процедур, гарантирующих построение расширяемых моделей, не существует. Можно лишь дать следующий совет. Выполняя анализ информационных потребностей пользователей, пытайтесь оценить вероятность и направления изменений их требований. Создавая локальную модель, исследуйте возможность отражения этих изменений в модели без коренной перестройки. Наконец, завершая создание глобальной модели, убедитесь в том, что в неё можно будет внести ожидаемые изменения, если они действительно произойдут.

Этап 3.4. Создание окончательной диаграммы глобальной модели

На этом этапе выполняется подготовка глобальной модели к публикации. Создаются окончательные версии диаграммы, словаря данных и вспомогательных документов, отражающие все изменения, внесённые в ходе проверок.

Этап 3.5. Обсуждение глобальной модели с пользователями

Цель обсуждения — убедиться в том, что модель адекватно отображает информационные потребности предприятия.

Полностью готовая модель (диаграмма и текстовые документы) рассылается пользователям для анализа. Пользователи должны убедиться в том, что она точно отражает структуру и функции предприятия. В назначенное время проводится конференция пользователей. Если в ходе обсуждений выявились проблемы, требующие корректировки модели, эта корректировка должна быть выполнена.

Контрольные вопросы

1. Перечислите этапы создания логической модели данных предприятия.
2. Что является исходными данными для логической модели?
3. Перечислите этапы создания локальной логической модели.
4. Какие элементы ER-модели данных являются нежелательными при её отображении на реляционную модель?
5. Какие виды работ выполняются на этапе очистки локальной концептуальной модели?
6. Как устраняются композитные и многозначные атрибуты?
7. Как устраняются n-арные связи, бинарные связи типа M:N и унарные связи?
8. Какие виды работ выполняются на этапе определения набора отношений?
9. Какие способы описания отношений Вы знаете?
10. Какие типы связей и как могут быть реализованы в реляционной модели данных?
11. Сформулируйте требования нормализованности логической модели данных и объясните преимущества нормализованной модели.
12. Перечислите и опишите действия, выполняемые на этапе проверки нормализованности модели.
13. Какие проблемы могут возникать в отношении, находящемся в 3НФ, и какие способы их решения Вы знаете?
14. В каких ситуациях имеет смысл проверка отношения, находящегося в НФБК на соответствие требованию 4НФ?
15. Опишите известные Вам способы проверки исполнимости транзакций.
16. Перечислите типы внутренних ограничений целостности РМД.
17. Опишите процедуры определения ограничений для доменов и атрибутов.
18. Объясните смысл ограничения допустимости NULL - значений.

19. Сформулируйте ограничение целостности сущности и объясните его практическое значение.

20. Сформулируйте ограничение ссылочной целостности и объясните его практическое значение.

21. Опишите известные Вам стратегии поддержки ссылочной целостности.

22. Какие типы требований предприятия накладывают ограничения на способы выполнения транзакций? Как эти требования представляются на диаграмме модели?

23. Какие требования предъявляются к основным документам модели?

24. Для чего проводится и как организуется обсуждение модели с конечным пользователем?

25. Что является исходными данными для этапа создания глобальной логической модели данных предприятия?

26. Перечислите виды работ, выполняемых на этом этапе.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. — Изд. 3. — СПб.: КОРОНА принт, 2003. — 672 с.

2. Дейт К. Дж. Введение в системы баз данных. — Изд. 7. — М.-СПб.-Киев: Вильямс, 2001. — 1072 с.

3. Крёнке Д. Теория и практика построения баз данных. Изд. 8 — М.: Бином, 2003. — 800 с.

4. Конноли Т., Бегг К., Страчан А. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. — М.-СПб.-Киев: Вильямс, 2001. — 1112 с.

5. Хансен Г., Хансен Дж. Базы данных. Разработка и управление. — М.-СПб.-Киев: Вильямс, 2001. — М.: Бином, 1999. — 700 с.