5. Семантическая методика проектирования реляционных схем БД

В предыдущей главе мы познакомились с классической методикой проектирования реляционных БД. Трудно не согласиться с мнением многих авторов, утверждающих, что практическое ее использование осложняется такими факторами, как:

- нетрадиционный для людей способ восприятия и формализации ПрО;
- практическая неприменимость для анализа сложных ПрО;
- неоднозначность решения проблемы проектирования, граничащая с прямым перебором многочисленных вариантов схемы в поисках наиболее подходящего.

Вскоре после появления реляционной модели исследователи по моделям данных озаботились поиском более приемлемых способов решения проблемы проектирования. Именно в ходе этих работ рождались первые семантические модели, обеспечивающие высокоуровневое формальное представление ПрО. Они должны были устранить первые два недостатка классической методики – неестественность представлений для человека и неприменимость для анализа сложных ПрО.

Но, поскольку коммерческие СУБД не собираются поддерживать напрямую семантические модели данных, необходима трансляция схем БД с языка этих моделей на язык коммерческих СУБД и в первую очередь – реляционных. Примерно так, практически с первых публикаций по ER-модели стала вырисовываться основная схема семантической методики проектирования реляционных схем БД:

- проектирование семантической схемы ПрО с использованием той или иной модели;
- перевод схемы в реляционную модель с применением подходящего набора правил трансформации и получение множества предварительных отношений;
- проверка полученных отношений на удовлетворение требований нормальных форм и дальнейшая нормализация методом декомпозиции.

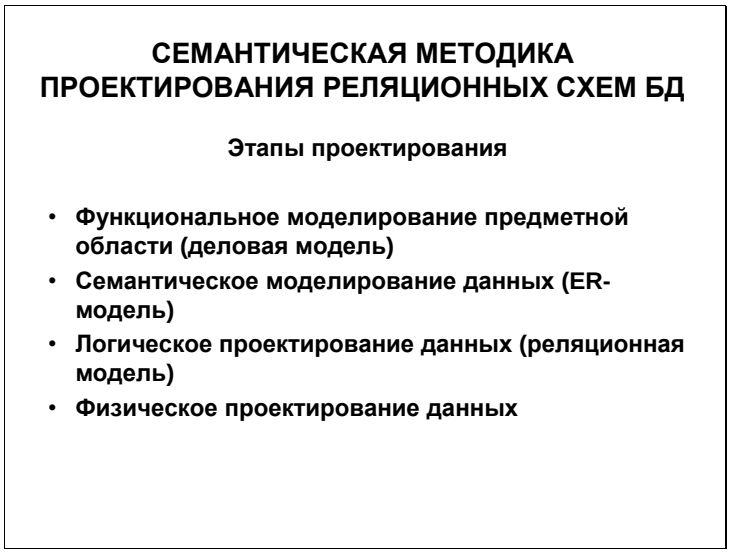
Все семантические методики действуют по одной и той же схеме и отличаются лишь используемыми моделями и правилами трансляции схем.

В этой главе вам сначала будет предложена достаточно полная семантическая методика, использующая ER-модель Чена и набор простейших правил трансформации. После этого мы рассмотрим другой набор более детальных правил, позволяющий в ряде случаев получить более эффективную схему БД. Далее мы обсудим некоторые идеи, которые будут полезны, если вы использовали расширенную ER-модель и определили в семантической схеме специализации и категоризации. Законченной методики их преобразования в реляционную схему пока нет (может быть, ее никогда и не будет), но кое-какой информацией с будущими проектировщиками мы поделимся.

Последней будет представлена развернутая методика трансформации схемы БД из ERM-модели в реляционную модель.

Наряду с достижением главной цели нашего курса – освоением знаний, необходимых проектировщику схем БД для коммерческих СУБД, по ходу обсуждения методики мы затронем также некоторые сопутствующие этому проектированию вопросы, а именно:

- функциональное моделирование ПрО, предшествующее процессу информационного моделирования;
- повышение эффективности полученной логической схемы БД за счет внесения управляемой избыточности (денормализации);
- повышение эффективности схемы БД за счет физического проектирования.



Если рассматривать задачу проектирования схемы БД для конкретной СУБД максимально широко, начиная с первоначальной постановки задачи и заканчивая полностью готовой схемой, можно выделить в ней четыре основных этапа:

- функциональное моделирование бизнес-процессов предприятия, для информационного обеспечения которых создается БД;
- семантическое моделирование данных в рамках той или иной семантической модели;
- получение логической схемы БД в понятиях логической модели выбранной СУБД;
- настройка БД на языке физической модели СУБД.

Далее в этой главе мы покажем, как в целом осуществлять эти этапы семантической методики на примере конкретных моделей (они обозначены на слайде) для медицинской ПрО, а также обсудим особенности применения других, альтернативных им моделей.

Следует отметить, что указанные этапы чаще всего не удается выполнить безупречно за один проход. Процесс проектирования, как правило, итеративен. Приходится неоднократно возвращаться к тому или иному предыдущему этапу, чтобы устранить проблемы, замеченные на каком-то из последующих этапов.

Очень часто проектировщики схем БД выполняют первые два этапа методики только на первой итерации, не утруждая себя внесением изменений в артефакты (аналитические и проектные документы) этих этапов при последующих итерациях. Логика их на первый взгляд убедительна – ведь окончательная схема передается СУБД в виде, определяемом только на последних двух этапах. Зачем же тогда выполнять лишнюю работу, особенно, когда каждая минута на счету.

В условиях несложных проектов это еще можно оправдать. Хотя и в этом случае теряется целостность восприятия проектных документов. Ведь на первых двух этапах создаются высокоуровневые представления об информационной системе, которые кроме собственно разработчиков нужны и представителям заказчика для контроля над проектом.

К тому же эти представления за счет своей наглядности и простоты позволяют быстро познакомиться или вспомнить ключевые моменты схемы. Особенно это актуально в сложных проектах, в которых участвуют несколько разработчиков, специализирующихся на разных этапах или подсистемах. В таком случае даже проблема смены исполнителей работ решается гораздо проще и быстрее.

И, наконец, практика внесения изменений в как можно более ранние артефакты с перепроектированием последующих абсолютно необходима в условиях использования

интегрированных CASE-инструментов, обеспечивающих все этапы разработки. Этому способствуют имеющиеся в таких системах автоматические преобразователи представлений и генераторы логической и физической схем БД и кода приложения.

5.1. Функциональное моделирование предметной области

На этапе анализа потребностей пользователей предусматривается серия макроанализов функционирования организации. Цели этого этапа состоят в следующем.

1. Изучение области деятельности моделируемой организации.
2. Определение информационных потребностей организации; при этом единственное существенное ограничение, которое должно учитываться – адекватность процессам функционирования организации.
3. Обеспечение представления указанных потребностей с помощью техники формального моделирования.

Изучение области (областей) деятельности организации сводится к определению целей организации и стратегии их достижения. Для определения информационных потребностей производится сбор метаданных и формирование метаописаний процессов функционирования организаций. Собранные метаданные будут основным исходным материалом для последующих этапов проектирования схемы. И, наконец, информационные потребности представляются средствами формальной модели.

Потребности анализируются посредством изучения документов, а также с помощью серии интервью. Этот процесс является итерационным. Как правило, необходимо несколько интервью по поводу каждой области деятельности организации. Они позволят уточнить информационные потребности и устранить кажущиеся противоречия в требованиях. Как правило, интервью необходимы также для получения одобрения результатов анализа пользователями.

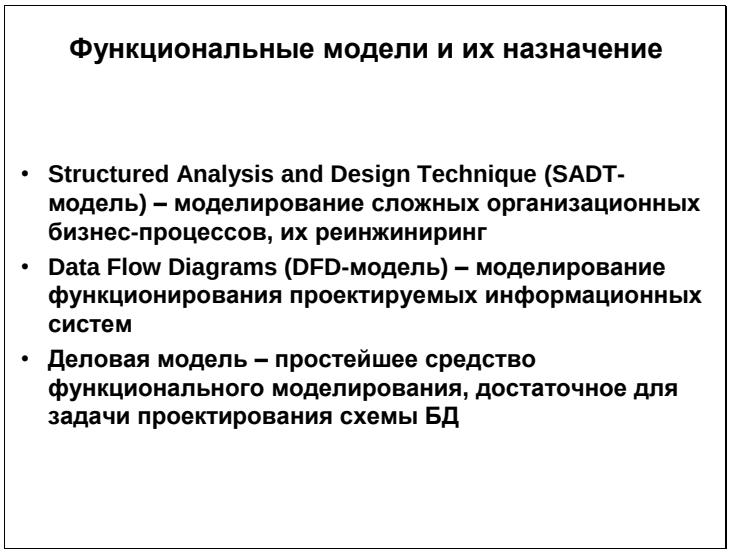
Информационные потребности определяются для всех уровней руководства организации. В большинстве организаций можно выделить три таких уровня руководства: высшее звено, среднее звено, оперативное звено. Тип собираемых метаданных и техника их сбора для всех этих уровней различны. Дабы уяснить содержимое документов и определить информационные потребности, не выраженные в форме документов, на каждом уровне пользуются интервью.

Высшее звено предоставляет информацию о целях и задачах организации, стратегии их достижения, методах управления реализацией стратегических планов, о возможных изменениях в функционировании организации, текущих и будущих изменениях в тактике. Среднее звено предоставляет информацию, позволяющую уточнить направление анализа, детализировать представление о политике, проводимой организацией, и ограничениях, которые она должна соблюдать, о различиях в производственных и управленческих операциях, получить сведения относительно требуемых величин времени реакции, надежности, целостности и конфиденциальности данных. И, наконец, оперативное звено может детально информировать о данных (имена, форматы, число экземпляров, ограничения целостности и т.п.). От руководства этого уровня исходят также сведения об использовании данных, их объеме, частоте реализации транзакций и их приоритете, последовательности выполнений транзакции и о требованиях к производительности.

Взаимодействие лица, выполняющего анализ, с пользователем и, в частности, установление обратной связи с ним, существенно влияет на успех этапа анализа потребностей. Необходимо тщательно выбрать модель представления информационных потребностей и информационных потоков, позволяющую пользователям понять и прокомментировать полученные спецификации. Моделями информационных потоков могут служить блок-схемы. Информационные потребности как основной объект нашего интереса могут быть представлены средствами любой из моделей, охарактеризованных в предыдущих главах; тем не менее, в зависимости от круга пользователей одним моделям может быть отдано предпочтение перед другими.

Непосредственно сформировать полностью формальное представление информационных потребностей довольно трудно. Организация обратной связи с

пользователем предполагает, что описание информационных потребностей будет понятно даже неискушенному пользователю. Следовательно, оно не может быть сделано в терминах очень сложной модели. При выявлении информационных потребностей целесообразно постоянно иметь в виду семантическую модель данных, что позволит сфокусировать внимание на специальных вопросах и, кроме того, облегчит последующее формальное описание потребностей. В данном случае не обязательно, чтобы модель на этом этапе соответствовала целевой модели данных (модели, которую поддерживает выбранная СУБД). Последующие шаги проектирования позволят преобразовать описание потребностей в целевую схему.



Создание любой программы начинается с выяснения функциональных требований – формулировки тех функций, которые должна выполнять система для пользователей. А поскольку, как мы выяснили, любая информационная технология лишь является орудием, способствующим реализации бизнес-процессов предприятия, начинать функциональное моделирование следует с выделения организационных бизнес-процессов и выяснения их структуры. Для этих целей наилучшим образом подходит метод SADT. В простейших случаях достаточно использования деловой модели.

Деловая модель является самым простым средством функционального моделирования. Она применяется в том случае, когда нет необходимости в построении детальной модели решаемых задач. Это в частности происходит, когда главной целью проектировщика является построение информационной модели. Тогда деловая модель предоставляет возможность разбить общую задачу на подзадачи по принципу «разделяй и властвуй».

Деловая модель дает простейшее описание функционирования организации и ее потребностей. Аспекты «как» и «что» можно представить в терминах **функций** (бизнес-процессов) организации и **классов данных**, которые используются для обеспечения выполнения этих функций. Соответствие функций и классов данных легко показать с помощью матрицы.

В этой модели ограничиваются рассмотрением только списка основных бизнес-процессов без их декомпозиции.

Вторым аспектом модели являются классы данных, представляющие семантически единые информационные объекты, либо возникающие или меняющиеся в процессе выполнения функции, либо необходимые ей для работы. Классы данных являются высокоуровневыми прообразами будущих множеств сущностей (ER-модель) и отношений (реляционная модель).

Сама деловая модель представляет собой матрицу, строки которой соответствуют функциям, а столбцы – классам данных. На пересечении столбца и строки ставится метка, означающая, что этот класс данных участвует в выполнении соответствующей функции. Позже, при информационном семантическом моделировании функции будут определять независимые этапы моделирования, а классы данных, упомянутые в строке, – границы рассмотрения на этих этапах. На последнем этапе семантического моделирования независимые информационные модели для каждой отдельной функции будут интегрированы в общую модель всей системы.

Деловая модель для процесса лечения пациента

КЛАССЫ ДАННЫХ / ФУНКЦИИ	ПАЦИЕНТ	БОЛЬНИЦА	ВРАЧ	ЛАБОРАТОРИЯ	АНАЛИЗ	ДИАГНОЗ	ПАЛАТА	ПЕРСОНАЛ
Провести первичный осмотр	X	X	X					
Обследовать больного	X		X	X	X			
Поставить диагноз	X		X			X		
Провести лечение в стационаре	X	X	X			X	X	X

Бизнес-процесс (или функция организации) – это совокупность различных видов деятельности компании, в рамках которой «на входе» используются один или более видов ресурсов, и в результате этой деятельности «на выходе» создается продукт, представляющий ценность для потребителя. Функции организации выявляются:

1) анализом формулировок целей и задач организации;
2) анализом рабочих программ организации;
3) идентификацией продукции или услуг, обеспечиваемых организацией, и определением функций производства продукции или оказания услуг.

Класс данных организации – это совокупность данных (атрибутов), необходимых для выполнения функции или вырабатывающихся в результате ее выполнения. Классы данных идентифицируются на основе анализа функций.

После определения функций и классов данных производится проверка непротиворечивости, неизбыточности и понятности определений.

Деловая модель для процесса лечения пациента представлена на слайде.

Возможно, кто-то готов поспорить с автором по поводу тех или иных проектных решений, их соответствия личным представлениям читателя об этой ПрО. Это совершенно естественно.

Любой фрагмент мира (даже на первый взгляд весьма небольшой) бесконечно сложен. И поэтому, чтобы всунуть его описание в «прокрустово ложе» всегда конечной схемы БД необходимо придерживаться одного очень важного правила – все проектные решения должны быть адекватны тем задачам, которые будут решаться с помощью будущей информационной системы. Это значит, что БД должна содержать всю информацию, существенную для решения этих задач, и только ее. От всего второстепенного и неважного надо стараться избавиться.

Так вот, в нашем случае демонстрируемый пример является всего-навсего учебным, призванным показать применение методики, моделей данных и, естественно, не претендует на всесторонний анализ медицинской ПрО. К тому же, хочется, чтобы в ходе проектирования были получены те самые схемы данных, которые ранее приводились при рассмотрении соответствующих моделей.

Какие же рассуждения привели нас к представленной деловой модели процесса лечения пациента?

Прежде всего, зададимся вопросом, для кого существует система здравоохранения. Хочется верить, что для пациентов. Она должна помогать им как можно дольше оставаться здоровыми (что вроде бы вытекает из самого названия этой системы). Теперь

представим себя пациентами и подумаем о тех случаях, когда нам приходится обращаться к услугам этой системы.

Во-первых, это консультации с врачами. Когда мы обнаруживаем какие-то проблемы со здоровьем, мы обращаемся именно к этому бизнес-процессу, как бы запуская новый его экземпляр. В лучшем для нас случае (проблемы оказались несущественными) на этом все в этот раз и заканчивается.

В худшем случае для выяснения ситуации может понадобиться второй выделенный нами бизнес-процесс обследования больного. Для этого врач направляет пациента на необходимые анализы и получает их результаты. Возможно, анализы покажут, что ничего страшного с организмом пациента не происходит, и опять необходимость в дальнейших посещениях больницы временно отпадет.

Если же проблемы со здоровьем не мнимые, осуществляется третий бизнес-процесс постановки диагноза.

И, наконец, болезнь может оказаться настолько серьезной, что потребуется лечение в стационаре, которое соответствует нашему четвертому бизнес-процессу.

Как вы понимаете, в каждом конкретном случае могут понадобиться экземпляры различных бизнес-процессов, причем в разном количестве (например, больному понадобится сделать несколько анализов). Не исключены ситуации, неучтенные в нашем анализе бизнес-процессов (например, обращение в скорую помощь или амбулаторное лечение). Но, наверное, трудно не согласиться с тем, что основные бизнес процессы, необходимые в большинстве случаев, мы выделили. Будем считать, что именно их информационная поддержка интересует нашего вымышленного заказчика.

Теперь в ходе детального анализа каждой функции в отдельности необходимо выяснить ее информационные потребности. То есть ответить на вопрос, информация о каких типах объектов понадобится для выполнения этой функции или будет порождена ею.

Очевидно, что во всех функциях главными действующими лицами являются *ПАЦИЕНТ* и *ВРАЧ*. Пожалуй, само лечебное учреждение – *БОЛЬНИЦА*, играет особую роль лишь в первой и последней функции. А учреждения типа *ЛАБОРАТОРИЯ* понадобится рассмотреть в бизнес-процессе обследования больного. При этом материал, взятый у пациента на анализ, и результаты его исследования будет олицетворять класс данных *АНАЛИЗ*. Констатацию наличия у пациента некоторого заболевания назовем *ДИАГНОЗОМ.*

Вызывает сомнение, необходим ли крестик на пересечении строки *Поставить диагноз* и столбца *АНАЛИЗ*. Если бы мы сочли, что результаты определенных анализов оказывают существенную роль при постановке некоторых диагнозов, этот крестик был бы необходим. Мы же приняли противоположное решение, посчитав, что, если даже такое влияние и имеет место, оно не столь существенно для этого бизнес-процесса. К тому же это решение упрощает схему, что для учебного примера вполне уместно.

Ну и, наконец, для последнего бизнес-процесса кроме ранее указанных будем считать существенными классы данных *ДИАГНОЗ* (болезни, которыми страдают пациенты стационара), *ПАЛАТА* (в них лежат пациенты стационара) и *ПЕРСОНАЛ* (медработники среднего звена, обслуживающие пациентов стационара).

Вот так выглядит функциональное моделирование в простейшем случае.

Вопросы и задания к параграфу 5.1

1. Каковы основные недостатки классической методики проектирования реляционных БД?
2. Как выглядит основная схема любой семантической методики проектирования БД?
3. Укажите этапы семантической методики проектирования БД?

4. Какие доводы можно привести в пользу необходимости внесения изменений синхронно в артефакты всех этапов проектирования, включая самые ранние?

5. Каковы цели этапа анализа потребностей задач ПрО? Каким образом они достигаются?

6. Для чего предназначены функциональные модели ПрО?

7. В каких понятиях описывается функционирование организации в деловой модели? Что стоит за этими понятиями?

8. Каковы основные принципы построения деловой модели?

9. Как в дальнейшем будет использоваться деловая модель ПрО на последующих этапах семантической методики?

5.2. Семантическое моделирование данных

Некоторые авторы называют этап семантического моделирования описанием предметной области, противопоставляя его, таким образом, следующему этапу – описанию базы данных.

В нашем примере, поясняющем методику проектирования, мы применяем для описания ПрО ER-модель (хотя можно использовать и другую семантическую модель). Это описание главным образом обеспечивает взаимодействие между пользователями и проектировщиками схемы. Описание должно быть достаточно неформальным, чтобы его могли понимать непрограммисты. Использование целевой модели данных (в нашем случае – реляционной) при описании предметной области не соответствует этой цели.

Мы приведем методику ER-моделирования, основанную на использовании деловой модели. Ясно, что, если бы мы выбрали другую функциональную или семантическую модель, методика была бы другой. Мы ранее рассмотрели методику ERM-моделирования, которая при определенных условиях также применима на этом этапе.

Еще раз отметим, что на этом первом этапе формализации информации о ПрО схема данных должна в идеале вобрать в себя определения всех основных понятий ПрО и всех закономерностей взаимоотношений между ними. С тем, чтобы все последующие межмодельные преобразования схемы носили чисто синтаксический характер и не требовали повторного анализа семантики ПрО.

Исходя из этой цели, следует для этапа семантического моделирования выбирать наиболее мощную в выразительном плане модель данных, способную представить все особенности информации о вашей ПрО.

Ну а теперь приступим к рассмотрению методики ER-моделирования, основанной на одном из краеугольных принципов структурного подхода к анализу и проектированию – сведение глобальной задачи к подзадачам, решение этих подзадач и синтез решения глобальной задачи из решений подзадач. Часто его кратко называют принципом «разделяй и властвуй». Кстати, как вы сейчас увидите, основа для разбиения на подзадачи заложена уже в деловой модели.

Семантическое моделирование данных (ER-модель)

Этапы построения ER-схемы предметной области

1. **Построение подсхем для каждой функции в отдельности:**
   - **1. А. Определение множеств сущностей**
   - **1. Б. Определение множеств связей**
   - **1. В. Определение ограничений целостности**
2. **Интеграция подсхем в общую ER-схему предметной области**

В целом методика семантического моделирования с использованием ER-модели выглядит следующим образом.

На первой фазе этапа описания ПрО идентифицируются все множества сущностей, выделяемые для каждой представляющей интерес области деятельности (функции), множества связей между ними, а также ограничения целостности. В результате формируется ряд «взглядов» на схему (подсхем) в зависимости от сфер деятельности. «Взгляды» затем интегрируются в описание ПрО, соответствующее схеме в целом.

Бизнес-процессы или функции, выделенные ранее в деловой модели, задают как бы новые, более простые ПрО. Методика предлагает сначала построить ER-схемы для них, а после объединить эти подсхемы в одну общую ER-схему. Причем метки в деловой модели определяют информационные потребности каждой функции.

Подход «разделяй и властвуй» применим в том случае, когда суммарная трудоемкость решения подзадач и синтеза решения глобальной задачи меньше трудоемкости решения изначальной задачи «в лоб». Наша ситуация (ER-моделирование ПрО) в полной мере удовлетворяет этому требованию, и предлагаемая методика существенно упрощает задачу проектирования. Особенно она актуальна для сложных ПрО с большим количеством бизнес-процессов, взаимосвязанных по данным.

Далее мы по очереди обсудим каждый этап ER-моделирования и осуществим соответствующие действия на примере медицинской ПрО.

Семантическое моделирование данных

Этап 1. Построение подсхем для каждой функции в отдельности

1.А. Определение множеств сущностей

1) **Каким множествам сущностей соответствует каждый класс данных?**
2) **Каковы семантика и имя каждого множества сущностей?**
3) **Какие атрибуты этих множеств сущностей представляют интерес с точки зрения функции?**
4) **Каковы их семантика и имена?**

Основа для определения множеств сущностей – сведения о классах данных, выделенных на этапе анализа потребностей. Может оказаться, что при построении деловой модели использовался сложный класс данных, который при детальном рассмотрении оказался агрегатом нескольких взаимосвязанных множеств сущностей. Выделение этих «атомарных» типов объектов – одна из важнейших задач этого этапа информационного моделирования. Необходимую дополнительную информацию получают интервьюированием пользователей.

В процессе определения множеств сущностей необходимо получить ответы на следующие вопросы.

1. Каким множествам сущностей соответствует каждый класс данных?
2. Каково значение (семантика) каждого множества сущностей?
3. Каково имя каждого множества сущностей?
4. Какие атрибуты каждого множества сущностей представляют интерес?
5. Каково значение (семантика) каждого атрибута?
6. Каково имя каждого атрибута?

Для каждого выделенного множества создается его описание с указанием ассоциированного класса данных. Описание именует множество сущностей, определяет, что представляют собой составляющие его сущности, перечисляет все его атрибуты. В случае необходимости в описание множества сущностей включают все его подтипы.

1.А. Определение множеств сущностей (пример)

Провести осмотр
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Адрес, Дата рождения (Д/Р), Пол, Номер медицинского полиса (НМП))
БОЛЬНИЦА (Название, Адрес, Телефон)
ВРАЧ (Фамилия, Специальность)
Обследовать больного
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст)
ВРАЧ (Фамилия, Специальность)
ЛАБОРАТОРИЯ (Название, Адрес, Телефон)
АНАЛИЗ (Тип анализа (Т/А), Назначенная дата (Н/Д), Назначенное время (Н/В), Номер направления (Н/Н), Состояние)
Поставить диагноз
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст)
ВРАЧ (Фамилия, Специальность)
ДИАГНОЗ (Тип диагноза (Т/Д), Осложнения)
Провести лечение
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст, Пол)
БОЛЬНИЦА (Название, Адрес, Телефон, Число коек (Ч/К))
ВРАЧ (Фамилия, Специальность)
ДИАГНОЗ (Тип диагноза (Т/Д), Осложнения, Предупреждающая информация)
ПАЛАТА (Номер палаты (Н/П), Название, Число коек (Ч/К))
ПЕРСОНАЛ (Фамилия, Должность, Смена, Зарплата (З/П))

На слайде приведены описания множеств сущностей для каждой функции в отдельности. Описание семантики множеств сущностей и атрибутов мы не приводим, надеясь, что читателю она давно известна.

Обратим ваше внимание на два момента.

1. Опытный проектировщик отождествляет понятия «класс данных» и «множество сущностей», выполняя тем самым декомпозицию сложных классов данных еще на этапе построения деловой модели.

Для новичков поясним сказанное на примере класса данных *ГАРАЖ*. Предположим, что нам заказали информационное обеспечение бизнес-процесса *Оказать пациенту услугу на дому*. В деловой модели для этой функции кроме всего прочего был указан класс данных *ГАРАЖ*. Соответствующие ему объекты обеспечивают доставку специалистов и оборудования к пациентам.

На этапе построения ER-схемы для этой функции детальный анализ указанного класса со всей очевидностью потребовал выделения в нем таких множеств сущностей, как *АВТОМОБИЛЬ*, *МЕСТО СТОЯНКИ*, *ВОДИТЕЛЬ*, *ДИСПЕТЧЕР*. Само множество сущностей *ГАРАЖ* могло при этом и не потребоваться.

2. Каждая функция рассматривается изолированно так, как будто других функций не существует. Отсюда вытекает, что одни и те же множества сущностей и их атрибуты могут встречаться в различных функциях, но состав атрибутов одного и того же множества сущностей может отличаться.

Какая информация о пациентах нам потребуется при его обращении за консультацией? В регистратуре его обязательно спросят о регистрационном номере (или номере карточки) и фамилии. Чтобы правильно определить участкового терапевта, ему надо знать адрес местожительства. Дата рождения и пол пациента помогут определиться с подходящими специалистами (детям нужен педиатр, а женщинам – гинеколог). И, наконец, без медицинского полиса с ним вообще не будут разговаривать.

А вот для выдачи пациенту направления на анализ от него потребуется указать только регистрационный номер, фамилию и возраст. Причем именно возраст, а не дату рождения, как будто до сих пор для сотрудников лабораторий вычисление возраста по дате рождения – неразрешимая проблема. Возраст же пациента им необходим, чтобы правильно определить отклонения от нормы тех показателей, которые с возрастом могут изменяться.

Семантическое моделирование данных

Этап 1. Построение подсхем для каждой функции в отдельности (продолжение)

1.Б. Определение множеств связей

1) **Какие множества связей между множествами сущностей ассоциируются с функцией?**
2) **Каковы семантика, имя и степень каждого множества связей?**
3) **Существуют ли у множеств связей собственные атрибуты, представляющие интерес с точки зрения функции?**
4) **Если да, то каковы их семантика и имена?**

Далее, на основе сведений о функциях и множествах сущностей, используемых в функциях, определяются множества связей. В ходе этого процесса необходимо получить ответы на следующие вопросы.

1. Какие взаимосвязи (множества связей) между множествами сущностей ассоциируются с каждой функцией?
2. Каково значение (семантика) каждого множества связей?
3. Каково имя каждого множества связей?
4. Какова степень каждого множества связей?
5. Существуют ли у множеств связей собственные атрибуты, представляющие интерес с точки зрения функции?
6. Если да, то каковы их семантика и имена?

Описание каждого множества связей содержит его имя, определение, а также ссылки на связываемые им множества сущностей и их роли.

Удобно и наглядно определять множества связей в виде ER-диаграмм для отдельных функций. Сначала в них указываются множества сущностей, выделенные для функций. Они представляются изолированными прямоугольными вершинами. А затем в ER-диаграмму добавляются множества связей в виде вершин-ромбов, которые связываются с соответствующими прямоугольниками ребрами ролей.

Если у множеств связей имеются собственные атрибуты, необходимо указать их семантику и имена в виде текстовых описаний.

Обратите внимание на то, что как и в случае определения множеств сущностей выделение множеств связей осуществляется для каждой функции по отдельности так, как будто бы других функций не существует. Благодаря этому в каждой из ER-диаграмм, в частности, появилось свое специфическое для функции множество связей между *ВРАЧОМ* и *ПАЦИЕНТОМ*: *Консультация*, *Направление на анализ*, *Постановка диагноза* и *Лечащий врач*. Более того, два из них бинарны, а два – тернарны.

И еще, обратите внимание на то, что для разных функций можно применить различные варианты проектных решений одних и тех же явлений ПрО. Так в бизнес-процессе постановки диагноза нам требуется давать ответы на вопросы:

- Кто поставил пациенту конкретный диагноз?
- Какие диагнозы поставил врач пациенту?

По этой причине в ER-диаграмме этой функции определено тернарное множество связей, обеспечивающее необходимую информативность.

В последнем бизнес-процессе лечащего врача интересует лишь картина заболеваний пациента, и ему не важно, кто их у него определил. В таком случае достаточно бинарного множества связей между *ПАЦИЕНТОМ* и *ДИАГНОЗОМ*.

На этом этапе как раз актуальны наши рассуждения по поводу разрешения проблемы триализма представлений для явлений ПрО, а также правила верного определения степени множества связей.

Не забудем отметить, что у множества связей *РАЗМЕЩЕНИЕ* есть свой атрибут *Номер койки*. Это можно выразить в текстовом виде следующим образом:

РАЗМЕЩЕНИЕ (*Номер койки*).

Семантическое моделирование данных

Этап 1. Построение подсхем для каждой функции в отдельности (продолжение)

1.В. Определение ограничений целостности

1) **Каковы области значений каждого атрибута? Есть ли среди них многозначные атрибуты?**
2) **Каковы ключи каждого множества сущностей и множества связей? Как можно еще идентифицировать сущности и связи каждого типа?**
3) **Какие типы отображений соответствуют каждому множеству связей с точки зрения функции?**
4) **Каковы другие ограничения целостности, которые напрямую не отражаются в ER-модели? Сформулировать их на естественном языке или языке логики предикатов первого порядка.**

И, наконец, в заключение первого этапа построения описания ПрО определяются ограничения, накладываемые на атрибуты, множества сущностей и множества связей.

При определении ограничений необходимо получить ответ на такие, например, вопросы.

1. Какова область значений каждого атрибута? Есть ли среди них многозначные?
2. Каковы известные функциональные зависимости между атрибутами каждого множества сущностей и множества связей?
3. Каковы ключи (если таковые существуют) каждого множества сущностей? Как можно еще идентифицировать сущности и связи каждого типа?
4. Какой тип отношений соответствует каждому бинарному множеству связей?
5. Какие типы отображений соответствуют каждому множеству связей?
6. Какие ограничения, выражаемые в логике предикатов первого порядка, накладываются на данные?

На самом деле приведенный список не является исчерпывающим. Здесь упомянуты лишь наиболее существенные и часто встречающиеся ограничения целостности.

В последнем вопросе предложена идеальная и универсальная форма выражения ограничений целостности, для которых ER-модель не предлагает подходящих декларативных средств. Для проектировщиков, не владеющих логикой предикатов первого порядка в достаточной степени, допустима форма предложений естественного языка.

В любом случае надо постараться представить так или иначе все закономерности, выявленные в ПрО на этапе анализа. Причем требуется максимально использовать багаж выразительных возможностей ER-модели, которую проектировщик должен знать досконально.

1.В. Определение ограничений целостности (пример)

Провести осмотр
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Адрес, Дата рождения (Д/Р), Пол, Номер медицинского полиса (НМП))
БОЛЬНИЦА (Название, Адрес, Телефон(М))
ВРАЧ (Фамилия, Специальность)
Обследовать больного
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст)
ВРАЧ (Фамилия, Специальность)
ЛАБОРАТОРИЯ (Название, Адрес, Телефон(М))
АНАЛИЗ (Тип анализа (Т/А), Назначенная дата (Н/Д), Назначенное время (Н/В), Номер направления (Н/Н), Состояние)
Поставить диагноз
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст)
ВРАЧ (Фамилия, Специальность)
ДИАГНОЗ (Тип диагноза (Т/Д), Осложнения)
Провести лечение
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Возраст, Пол)
БОЛЬНИЦА (Название, Адрес, Телефон(М), Число коек (Ч/К))
ВРАЧ (Фамилия, Специальность)
ДИАГНОЗ (Тип диагноза (Т/Д), Осложнения, Предупреждающая информация)
ПАЛАТА (Номер палаты (Н/П), Название, Число коек (Ч/К))
ПЕРСОНАЛ (Фамилия, Должность, Смена, Зарплата (З/П))

Определение ограничений целостности, которые отражают специфику ПрО, являются непротиворечивыми и могут быть удовлетворены, – это предельно трудная задача. Дело в том, что некоторые виды ограничений весьма сложны для понимания, что влечет за собой ошибки. Непросто и показать, что множество ограничений непротиворечиво. Здесь легко вторгнуться в область сложных проблем доказательства теорем. И, наконец, сформированное множество ограничений может оказаться настолько жестким, что БД, которая, возможно, уже существует, не удовлетворит ему. И это неудивительно, так как, к сожалению, большинство реальных БД сильно «замусорены». Иными словами, они содержат данные, не подчиняющиеся формальным ограничениям, выводимым на основе анализа потребностей, что, однако, зачастую не рассматривается как дефект БД.

Тем не менее, продемонстрируем, как реализовать этот этап семантического моделирования на примере медицинской ПрО. Часть ОЦ можно выразить в символьной форме. К ним относятся:

- ключевые атрибуты множеств сущностей и связей (они подчеркиваются в списках атрибутов);
- многозначные атрибуты (им в списках атрибутов сопоставляется описатель «(*M*)»);
- указание множества значений или типа атрибутов в виде описателя атрибута (возможно с заданием максимальной длины значений);
- другие символьные ОЦ на значения атрибутов, которые задаются в виде выражений логики предикатов первого порядка, как мы это делали в параграфе 3.2.

В нашем примере единственный ключевой атрибут оказался у множества сущностей *ПАЦИЕНТ*, им является *Регистрационный номер*. Другой, подчеркнутый на слайде атрибут *Номер палаты* в отдельности ключевым не является, но вместе со связями типа *БОЛЬНИЧНАЯ ПАЛАТА* его значения обеспечивают идентификацию сущностей типа *ПАЛАТА* (ID-зависимость).

Многозначными в нашей ПрО являются атрибуты *Телефон* множеств сущностей *БОЛЬНИЦА* и *ЛАБОРАТОРИЯ*. На слайде для них указан соответствующий описатель. Другие описатели атрибутов на слайде для краткости опущены, но мы думаем, что с ними у читателей проблем не будет. Кстати, поскольку в ER-модели не предусмотрены стандартные множества значений и типы данных, рекомендуем использовать типы данных целевой СУБД, с моделью которой к этому моменту надо уже познакомиться.

В качестве примера символьных ОЦ, не имеющих специальных синтаксических конструкций в ER-модели и поэтому записываемых на естественном языке или языке логики предикатов первого порядка, приведем следующие:

- Значение атрибута *Назначенное время* множества сущностей *АНАЛИЗ* меньше 24 или

  $\forall e \in АНАЛИЗ(Назначенное\ время(e) < 24)$.

- Атрибут *Пол* множества сущностей *ПАЦИЕНТ* может принимать только два допустимых значения «*М*» и «*Ж*» или

  $\forall e \in ПАЦИЕНТ(Пол(e) \in \{"М","Ж"\})$.

В ER-диаграммах функций следует представить те ОЦ, для которых предусмотрена наглядная графическая нотация. Прежде всего, это:

- пометки на ребрах ролей (для бинарных множеств связей они определяют тип отношения, для множеств связей степени, большей 2, – максимальные кардинальные числа отображений, определяющих роли);
- атрибутика зависимостей существования (E-зависимостей) и зависимостей по идентификации (ID-зависимостей) – двойная граница зависимого множества сущностей, стрелка на ребре, ведущем к нему, и соответствующая пометка в ромбе.

Наряду с этими обозначениями, можно использовать для тех же целей графическую нотацию ERM-модели, в которой на каждом ребре роли указываются по две пометки в виде минимального и максимального кардинальных чисел отображений, определяющих роль и определяемых ролью.

На этом завершается первый этап семантического моделирования, в ходе которого для каждой функции организации построены отдельные ER-схемы, включающие:

- списки множеств сущностей и связей с их атрибутами;
- ER-диаграммы;
- текстовые описания элементов схемы и ОЦ (для последних можно использовать язык логики предикатов первого порядка).

Семантическое моделирование данных

Этап 2. Интеграция подсхем в общую ER-схему предметной области

1) **Интеграция множеств сущностей методом их семантического объединения**
2) **Интеграция атрибутов каждого множества сущностей методом их семантического объединения**
3) **Интеграция ограничений целостности, ассоциированных с каждым множеством сущностей методом их объединения**
4) **Интеграция множеств связей методом их семантического объединения; возможна генерализация множеств связей**
5) **Интеграция атрибутов каждого множества связей методом их семантического объединения**
6) **Повторная тщательная проверка типов отображений, соответствующих каждому множеству связей**
7) **Интеграция других ограничений целостности, которые напрямую не отражаются в ER-модели, методом их объединения; сформулировать их на естественном языке или языке логики предикатов первого порядка.**

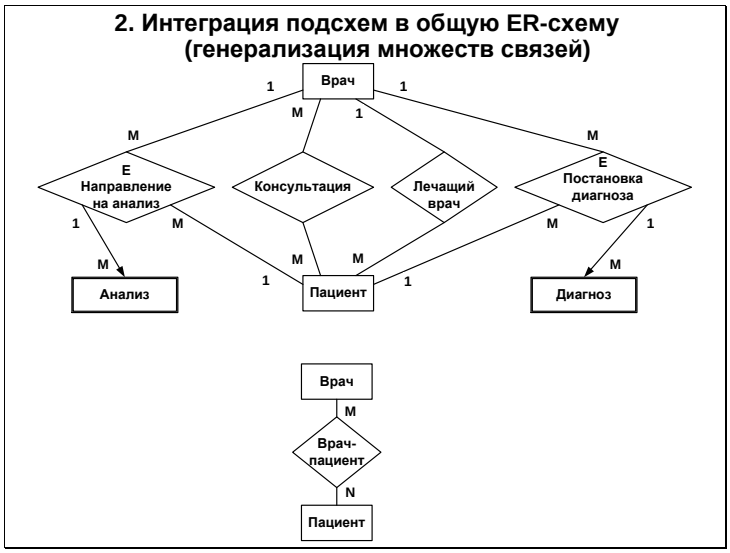
На заключительном этапе синтеза описания ПрО интегрируются отдельные «взгляды», характерные для конкретных областей деятельности организации. При этом необходимо разрешить возможные конфликты именования, ликвидировать избыточность и неоднозначность.

Основной операцией этапа интеграции является объединение в теоретико-множественном смысле, в результате которого устраняются элементы-дубликаты. Правда, следует различать обычное и семантическое объединение. В первом случае объединяются множества знаков, которыми представлены элементы подсхем, а во втором – множества значений этих знаков или их смыслов. Поясним сказанное на примере.

Предположим, что в одной функции мы определили множество сущностей с именем *ПАЦИЕНТ*, а в другой функции для того же по смыслу множества сущностей использовалось имя *БОЛЬНОЙ*. Для устранения синонимии и дубликатов структур мы применяем при интеграции множеств сущностей этих функций семантическое объединение, при котором множества сущностей объединяются по смыслу, и в результирующей схеме останется одно из этих двух множеств сущностей. Следует только выбрать для него одно из ранее использовавшихся имен.

Этот способ интеграции используется для множеств сущностей, множеств связей и атрибутов. А вот для ОЦ подойдет и обычное объединение. Все ОЦ, построенные для подсхем, переносятся в результирующую схему. Правда и в этом случае повторным анализом полученной схемы необходимо устранить противоречия и избыточность.

Напомним, что главный критерий, которому должна соответствовать любая схема БД, – это ее удовлетворяемость или непротиворечивость. Иначе схема вообще не работоспособна. Ну, а избыточность ОЦ – верный признак недостаточной эффективности схемы.



Как отмечалось ранее, при интеграции множеств связей возможна их генерализация в итоговой схеме, т.е. замена нескольких специализированных множеств связей одним обобщенным множеством связей. На слайде первым приведен фрагмент ER-диаграммы для медицинской ПрО, полученной семантическим объединением ER-диаграмм функций (верхний граф), и фрагмент ER-диаграммы, образованный из первого фрагмента методом генерализации множеств связей (нижний граф).

Вы уже, наверное, понимаете, что подобное преобразование схемы влечет за собой, как правило, потерю информативности. Если в первом случае схема позволяет отвечать на вопрос, по какому конкретному поводу возникла связь врача с пациентом, то во втором случае мы сможем получить информацию лишь о том, что хотя бы одна их встреча состоялась. К тому же, предшествующая генерализации замена тернарных множеств связей *Направление на анализ* и *Постановка диагноза* на два бинарных множества связей, одно из которых – *ВРАЧ-ПАЦИЕНТ*, а второе – *Анализ пациента* и *Диагноз пациента* соответственно, еще больше понижает информативность схемы. Теперь мы не сможем определить, кто из врачей направил пациента на тот или иной анализ или кто определил у пациента конкретное заболевание.

Однако, если повышенная информативность ни к чему при выполнении бизнес-процессов организации, подобные преобразования вполне уместны, и они приводят к более простой, а значит и эффективной схеме БД.

Следует лишь отметить, что после генерализации необходимо заново проанализировать характеристики отображений для результирующего множества связей. Так, например, в ходе генерализации (объединения) функционального и нефункционального отображений образуется нефункциональное отображение.

Рассмотрим исходную схему, в которой между множествами сущностей *ВРАЧ* и *ПАЦИЕНТ* есть лишь множества связей *Консультация* и *Лечащий врач*. Они определяют в частности отображения

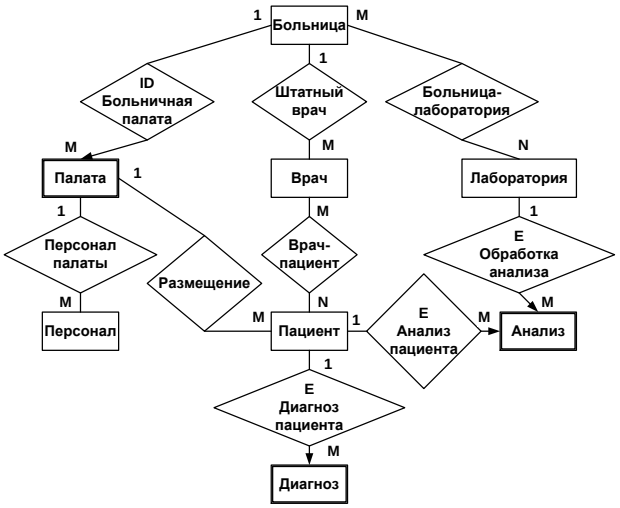
Консультирующий врач пациента: *ПАЦИЕНТ -> ВРАЧ* и

Лечащий врач пациента: *ПАЦИЕНТ -> ВРАЧ*.

В ранее высказывавшихся предположениях первое отображение нефункционально, а второе – функционально.

В итоговой схеме мы ограничились одним обобщенным множеством связей *ВРАЧ-ПАЦИЕНТ*. Его отображение *ПАЦИЕНТ -> ВРАЧ* суть объединение ранее упомянутых отображений. Так вот оно нефункционально.

2. Интеграция подсхем в общую ER-схему (пример)



2. Интеграция подсхем в общую ER-схему (пример)

БОЛЬНИЦА (Название, Адрес, Телефон(М), Число коек (Ч/К))
ПАЛАТА (Номер палаты (Н/П), Название, Число коек (Ч/К))
ПЕРСОНАЛ (Фамилия, Должность, Смена, Зарплата (З/П))
ВРАЧ (Фамилия, Специальность)
ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Адрес, Дата рождения (Д/Р), Пол, Номер медицинского полиса (НМП), Возраст(вирт.))
ДИАГНОЗ (Тип диагноза (Т/Д), Осложнения, Предупреждающая информация)
ЛАБОРАТОРИЯ (Название, Адрес, Телефон(М))
АНАЛИЗ (Тип анализа (Т/А), Назначенная дата (Н/Д), Назначенное время (Н/В), Номер направления (Н/Н), Состояние)
РАЗМЕЩЕНИЕ (Номер койки (Н/К))

В результате семантического моделирования получилась ER-схема ПрО, представленная на слайдах. Первый слайд демонстрирует ER-диаграмму в нотации Чена, на втором приведены списки атрибутов множеств сущностей и связей.

Обратите внимание, что во множестве сущностей *ПАЦИЕНТ* сохранен как атрибут *Дата рождения*, так и атрибут *Возраст*. Правда последний снабжен описателем «*вирт.*». Это говорит о том, что он является виртуальным, а не базовым. Это означает, что в хранимой таблице *ПАЦИЕНТ* он будет отсутствовать, а необходимые значения этого атрибута будут вычисляться в представлениях (view) на основе значений хранимого атрибута *Дата рождения*.

Помимо формального представления, описание ПрО сопровождается формированием некоторого документа, содержащего изложенные в свободной форме обобщенные сведения, полученные в результате интервьюирования. Документы направляются для одобрения в соответствующие подразделения организации. Прежде чем все подразделения подтвердят, что описание ПрО точно отражает информационные потребности каждой области деятельности организации, может оказаться необходимым провести несколько итераций семантического моделирования.

На конечной фазе этапа описания ПрО определяются требования к обработке транзакций. Рассматриваются все текущие и перспективные типы транзакций. Для каждой транзакции указываются вид (выборка, обновление), частота реализации, внешний источник, получатель результатов, части схемы, затрагиваемые транзакцией.

Определяя требования к реализации транзакций, целесообразно задаваться такими, например, вопросами.

1. Какие транзакции необходимо реализовать для каждой области деятельности организации?
2. Какие множества сущностей, атрибуты и множества связей затрагиваются каждой транзакцией?
3. Как в общих чертах охарактеризовать каждую транзакцию в терминах описания ПрО?
4. Какого рода доступ к данным (выборка, обновление) связан с каждой транзакцией?
5. Каков режим выполнения транзакций (пакетный, оперативный)?
6. Какова частота реализации каждой транзакции (ежедневно, еженедельно)?
7. Каков приоритет обработки, присваиваемый каждой транзакции?
8. Каковы требования к параллельной обработке транзакций?
9. Какие отчеты необходимы?
10. Каков формат каждого отчета?
11. Каковы приемлемые временные рамки получения каждого отчета?
12. Какие требования к безопасности данных наиболее важны?
13. Какие части БД наиболее существенны с точки зрения обеспечения функционирования БД?

Вопросы и задания к параграфу 5.2

1. Какова главная стратегия процесса семантического моделирования с использованием деловой модели, как исходного артефакта, и ER-модели, как целевого формализма для представления схемы БД?
2. Какие этапы выделяются в этом процессе? Какие задачи решаются в ходе этих этапов?
3. На какие вопросы необходимо дать ответ при определении множеств сущностей?
4. На какие вопросы необходимо дать ответ при определении множеств связей?
5. На какие вопросы необходимо дать ответ при определении ограничений целостности?
6. По каким правилам осуществляется интеграция подсхем в общую ER-схему ПрО?
7. Что представляет собой генерализация множеств связей?

5.3. Логическое проектирование данных

За этапом синтеза описания ПрО следует этап преобразования этого описания в описание БД или логического проектирования данных. Под описанием БД будем понимать описание схемы, соответствующее модели данных, поддерживаемой целевой СУБД, точнее, ее логической модели. Напомним, что любая СУБД на самом деле поддерживает не одну, а две модели данных – логическую (предназначенную для взаимодействия с пользователями) и физическую (используемую для представления данных в оперативной памяти и на диске).

Так вот задачей логического проектирования является подготовка схемы БД в терминах и на языке логической модели СУБД. В нашем примере речь будет идти об обобщенной СУБД-ориентированной реляционной модели данных. Таким образом, цель этого этапа в нашем случае – получение реляционной схемы, как совокупности схем отношений.

Дальнейшая судьба этой схемы существенным образом зависит от предлагаемых СУБД интерфейсов для передачи схемы БД. В большинстве современных систем это язык SQL, а, значит, схема представляется в виде скриптов (англ. script), состоящих из команд языка определения данных (DDL-команд) этого языка. Но встречаются и системы, для ввода схемы БД в которые предлагается диалоговый интерфейс, как правило, относительно простой и удобный.

И команды SQL, и диалоговые окна формирования схемы БД могут наряду с элементами логической схемы, содержать и параметры настройки физической модели СУБД. О них более детально пойдет речь на последнем этапе методики проектирования схемы БД. Чтобы четко отличать одно от другого, надо помнить, что логическая схема представляет семантику ПрО (понятия и их взаимосвязи), а физический уровень определяет параметры элементов схемы БД, связанные с эффективностью их хранения и методов доступа к ним.

Логическое проектирование данных (реляционная модель)

Этапы логического проектирования данных

1. **Трансформация ER-схемы (ERM-схемы) в реляционную схему с помощью соответствующих правил.**
2. **Проверка отношений полученной реляционной схемы на выполнение условий требуемых нормальных форм и их дальнейшая нормализация (см. п. 4.2.5.5).**
3. **В необходимых случаях (неэффективность выполнения запросов к БД) введение контролируемой избыточности данных (денормализация схемы), которой сопутствуют меры, исключающие возникновение аномалий.**

В общем случае логическое проектирование данных для реляционной модели состоит из трех этапов, первый из которых необходимо выполнить при любом таком проектировании с использованием семантической методики, два последующих этапа следует иметь в виду, но потребность в них будет возникать отнюдь не всегда.

Собственно логическая схема реляционной БД создается именно на первом этапе с помощью правил преобразования семантической схемы (в нашем примере – ER-схемы) в реляционную схему. В большинстве случаев эта схема будет вполне работоспособной и не потребует дальнейших улучшений. Однако возможны ситуации, при которых еще можно будет улучшить ее качество и/или эффективность. Для этого необходимо выполнить какой-либо из оставшихся этапов или оба эти этапы.

Второй этап логического проектирования позволит устранить еще оставшиеся в реляционной схеме аномалии вставки, обновления и удаления. Как вы понимаете, это осуществляется приведением отношений схемы в надлежащие нормальные формы (п. 4.2.5). Опытный проектировщик, знающий требования нормальных форм (НФБК, 4НФ, 5НФ) еще на этапе семантического моделирования учитывает их при выделении множеств сущностей и множеств связей. Поэтому у него реляционная схема, полученная на первом этапе логического проектирования, редко когда не находится в нужной нормальной форме.

Чего не скажешь в случае, когда за дело берется новичок. Основной причиной «ненормальности» схемы является недостаточная продуманность набора множеств сущностей и множеств связей. Ненормальные отношения будут образовываться для множеств сущностей и множеств связей, представляющих не элементарные типы объектов и связей, а агрегаты таких типов.

Одним из выходов из этой ситуации является выполнение второго этапа логического проектирования. Он заключается в выполнении алгоритма декомпозиции (п. 4.2.5.5) для каждого из предварительных отношений, полученных на первом этапе логического проектирования. Другим способом повышения качества реляционной схемы может явиться возврат к этапу семантического моделирования и пересмотр набора множеств сущностей и множеств связей ER-схемы. После этого придется повторить первый этап логического проектирования.

Первые два этапа, в конечном счете, должны привести к идеальной реляционной схеме в смысле нормализованности ее отношений (об обязательном удовлетворении потребностей задач ПрО мы, естественно, умалчиваем как о само собой разумеющемся требовании). Такой по-хорошему она и должна быть передана СУБД. В этом случае гарантированы идеальные эксплуатационные качества БД.

Все бы хорошо, но при использовании нормализованной схемы можно столкнуться с неудовлетворительным временем отклика системы на некоторые запросы и отчеты. Причина этого в том, что в угоду нормализации данные, необходимые для этих запросов и отчетов, разбросаны по многочисленным отношениям. И для того, чтобы собрать их вместе, требуется выполнить серию самых дорогостоящих операций над данными – соединений.

Часто сформулированную проблему можно преодолеть с использованием средств физического моделирования (индексов, кластеров и т.д.). О них мы расскажем позже. Другим, часто более эффективным методом является введение в логическую схему тех или иных избыточных структур (транзитивных внешних ключей, дубликатов атрибутов, агрегированных атрибутов и таблиц). Поскольку все они приводят к нарушению требований нормальных форм отношений, иногда этот процесс называют **денормализацией** схемы.

Самое главное, чтобы при этом в схему не вернулись те аномалии, от которых мы старательно избавлялись, то есть внесенная избыточность должна быть контролируемой. Главный принцип при этом следующий. Ввод, модификация и удаление, осуществляемые пользователем, должны выполняться на нормализованных структурах, структуры-дубликаты модифицируются исключительно специальными программами, исключающими некорректность данных. Избыточные данные должны предъявляться пользователям только в режиме чтения.

Поскольку денормализованные структуры являются частью реляционной схемы, их создание осуществляется на этапе логического проектирования данных.

Логическое проектирование данных
Этап 1. Трансформация ER-схемы в реляционную схему с помощью соответствующих правил

Успех семантической методики проектирования в основном определяется двумя факторами:

- мощностью применяемой семантической модели данных;
- способностью используемого набора правил преобразования семантической схемы в СУБД-ориентированную схему порождать в каждом конкретном случае идеальные логические схемы.

По поводу первого аспекта мы уже много говорили. Действительно, способность ее семантической модели формально описать любую, сколь угодно сложную ПрО – одно из главных достоинств семантической методики. Остается лишь полно и эффективно перенести отраженную в семантической схеме семантику ПрО в СУБД-ориентированную схему. И тут вступает в силу второй фактор в лице надежного, детально проработанного набора правил этой трансформации.

Надежность правил означает, что за предлагаемыми ими преобразованиями скрывается убежденность (в идеале – математически доказанная) в их безупречности и эффективности. Как правило, этому качеству набора правил сопутствует детальность учета многочисленных факторов семантической схемы. Идеальной логической схемы позволяет достигнуть проектирование, при котором ни один из элементов семантической схемы не остается неучтенным. Любое упрощенное рассмотрение способно привести в отдельных случаях к неэффективным схемам БД.

Итак, сочетание выразительной семантической модели данных и мощного набора правил ее преобразования в логические модели СУБД в состоянии обеспечить идеальную семантическую методику, с использованием которой можно будет получать безупречные схемы БД. Но это пока лишь перспектива, в направлении которой устремлены научные исследования автора. Их первые результаты будут представлены в заключительной части повествования о логическом проектировании данных в виде оригинальной методики преобразования «ERM-схема – реляционная схема».

До этого нам предстоит рассмотреть некоторые, известные по многочисленным публикациям в основном зарубежных авторов способы реализации этого этапа логического проектирования для ER-модели Чена.

Поскольку, как вы уже знаете, ничего в этой книге не приводится просто так, несколько слов по поводу последнего слайда. На нем в стилизованном виде изображено автоматическое устройство, преобразующее семантические схемы ПрО в логические схемы БД. Действительно, наивысшей формой семантической методики, доведенной до удобного практического использования, является ее реализация в виде CASE-средства – программной системы, автоматизирующей труд проектировщика схем БД. В них, наряду с

другими компонентами, присутствуют преобразователи проекта БД – Database Design Transformer (DDT), автоматизирующие этап межмодельного преобразования. Эти преобразователи – не что иное, как программное воплощение некоторого набора правил трансформации.

Следует отметить, что, к великому сожалению, в широко распространенных CASE-инструментах пока реализованы лишь самые слабые семантические модели (максимум – ER-модель Баркера) с часто тривиальными правилами преобразования. Это приводит к тому, что иногда задумываешься, а стоит ли «городить огород», применяя эти средства, может спроектировать схему по «бумажной» технологии, но с использованием более выразительной модели и адекватными правилами преобразования.

В силу примитивности CASE-преобразователей мы как раз рассмотрим более изощренные правила трансформации схем, хотя и существуют они пока лишь на бумаге.

Простейшие правила перехода от ER-схемы к реляционной схеме БД

1. **Каждое множество сущностей представляется самостоятельным отношением, однозначные атрибуты множества сущностей становятся атрибутами отношения, ключи множества сущностей являются возможными ключами отношения; при необходимости в качестве первичного ключа отношения используется суррогатный атрибут.**
2. **Бинарные множества связей типа 1:M без атрибутов представляются дублированием первичного ключа 1-отношения в M-отношение.**
3. **Бинарные множества связей типа M:N без атрибутов представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей.**
4. **Множества связей с атрибутами представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.**

Простейшие правила перехода от ER-схемы к реляционной схеме БД (продолжение)

5. **Множества связей степени больше 2-х представляются самостоятельными отношениями, куда дублируются первичные ключи отношений, построенных для множеств сущностей. Однозначные атрибуты множества связей становятся атрибутами этого отношения.**
6. **Каждый многозначный атрибут множества сущностей представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для множества сущностей; второй атрибут этого отношения предназначен собственно для значения.**
7. **Каждый многозначный атрибут множества связей представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для множества связей; второй атрибут этого отношения предназначен собственно для значения.**

Реляционная схема медицинской БД

БОЛЬНИЦА (Код больницы (К/Б), Название, Адрес, Число коек (Ч/К))

ПАЛАТА (Код больницы (К/Б), Номер палаты (Н/П), Название, Число коек (Ч/К))

ПЕРСОНАЛ (Код больницы (К/Б), Номер палаты (Н/П), Фамилия, Должность, Смена, Зарплата (З/П))

ВРАЧ (Код врача (К/В), Код больницы (К/Б), Фамилия, Специальность)

ПАЦИЕНТ (Регистрационный номер (Р/Н), Фамилия, Адрес, Дата рождения (Д/Р), Пол, Номер медицинского полиса (НМП))

ДИАГНОЗ (Регистрационный номер (Р/Н), Тип диагноза (Т/Д), Осложнения, Предупреждающая информация)

ЛАБОРАТОРИЯ (Код лаборатории (К/Л), Название, Адрес)

АНАЛИЗ (Регистрационный номер (Р/Н), Код Лаборатории (К/Л), Тип анализа (Т/А), Назначенная дата (Н/Д), Назначенное время (Н/В), Номер направления (Н/Н), Состояние)

БОЛЬНИЦА-ЛАБОРАТОРИЯ (Код больницы (К/Б), Код Лаборатории (К/Л))

ВРАЧ-ПАЦИЕНТ (Код врача (К/В), Регистрационный номер (Р/Н))

РАЗМЕЩЕНИЕ (Код больницы (К/Б), Номер палаты (Н/П), Регистрационный номер (Р/Н), Номер койки (Н/К))

ТЕЛЕФОН БОЛЬНИЦЫ (Код больницы (К/Б), Телефон)

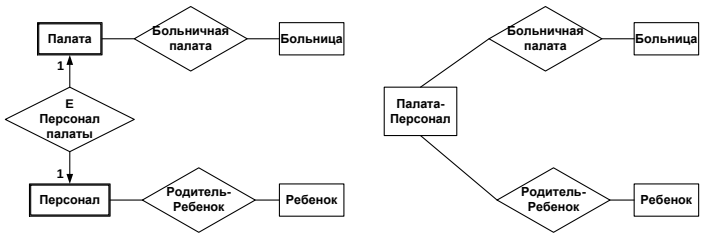
ТЕЛЕФОН ЛАБОРАТОРИИ (Код лаборатории (К/Л), Телефон)

На слайдах приведены уже знакомые вам правила межмодельного преобразования схем, рассмотренные нами в п. 4.2.1 и приведенные здесь для целостности семантической

методики. Отличительной особенностью этих правил является абсолютное игнорирование такой характеристики отображений, как минимальное кардинальное число и, как следствие, E-зависимостей и ID-зависимостей. Подобный подход конечно проще и может быть оправдан тем, что дополнительный анализ не всегда дает ощутимый результат. Так, для ER-схемы нашего примера использование более детального набора правил даст с точки зрения порождаемых структур, скорее всего, тот же результат. Правда, при этом будут порождены важные ограничения целостности, которые абсолютно игнорируются простейшими правилами.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД

1. **Каждое бинарное множество связей (МСв) типа $R(S1(1,1):S2(1,1))$ и множества сущностей (МСу) S1 и S2, в нем участвующие, заменяются в ER-схеме одним агрегированным МСу, с которым соединяются все другие МСв, имевшиеся у двух исходных МСу. Атрибуты нового МСу представляют собой объединение атрибутов обоих исходных МСу и МСв. Ключами полученного МСу являются ключи исходных МСу и МСв.**



Как уже отмечалось, чем проще набор правил (в нем с меньшей детальностью анализируются свойства семантической схемы), тем менее эффективная логическая схема БД может быть в общем случае построена с его помощью. Мы, например, вообще не рассматриваем такой тривиальный набор, в котором есть следующее правило: «для каждого множества сущностей или множества связей, независимо от его свойств, необходимо построить самостоятельное отношение». Понятно, какое количество лишних, абсолютно избыточных отношений может быть получено по этому правилу. Хотя, по большому счету, и такая БД будет работать.

Однако мы пойдем другим путем – в сторону повышения способности правил преобразования улучшать качество схемы БД. Вашему вниманию предлагается усовершенствованный набор правил, отличающийся от предыдущего анализом таких особенностей семантической схемы, как:

- бинарных множеств связей типа 1:1;
- значений минимальных кардинальных чисел отображений, определяемых бинарными множествами связей;
- специализаций и категоризаций.

Вы уже привыкли к тому, что правила межмодельного перехода строятся по шаблону «*ЕСЛИ в семантической схеме встречается определенная конструкция, ТО в логической схеме создается подходящая ей конструкция*». Каждое правило применяется до тех пор, пока не будут рассмотрены все варианты, обеспечивающие истинность его посылки. После того, как набор правил будет исчерпан, первый этап логического проектирования завершен. По такому принципу работают все трансляторы схем БД. Не является исключением и предлагаемый далее набор правил.

Правда, его первое правило, приведенное на слайде, все же отличается от других, поскольку по нему не строятся конструкции логической схемы, а осуществляются перестроения ER-схемы. Вместо трех вершин (двух множеств сущностей и одного бинарного множества связей) в ER-диаграмме остается одна вершина, олицетворяющая агрегированное множество сущностей.

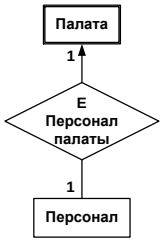
Чтобы показать пример применения этого правила, нам потребовалось изменить семантику множества связей *ПЕРСОНАЛ ПАЛАТЫ*. Здесь предполагается, что каждая палата обязательно обслуживается одним и только одним служащим, а каждый из служащих обязательно работает в одной и только одной палате. Другими словами их связывает взаимнооднозначное соответствие.

Могут, конечно, возникнуть проблемы с именованием вновь образованного агрегата. Но, с другой стороны, целесообразность применения этого правила не вызывает

никаких сомнений, поскольку, благодаря ему, будет получена явно более эффективная схема.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

2. **Если МСу не вступает в связи типа $R(S1(1,1):S2(1,1))$, оно образует одно отношение, построенное на однозначных атрибутах этого МСу. Возможными ключами этого отношения являются ключи МСу.**
3. **Бинарное МСв типа $R(S1(1,1):S2(0,1))$ представляется дублированием первичного ключа отношения, построенного для МСу S1 (множества регулярных сущностей), в отношение, построенное для МСу S2 (множества слабых сущностей). Этот внешний ключ является возможным ключом второго отношения и имеет описатель NOT NULL.**



Палата_ID	…	Персонал_ID

Персонал_ID	…

Второе правило практически без изменений перешло в этот набор из ранее рассмотренного. В соответствии с ним для каждого из оставшихся после применения первого правила множеств сущностей будет образовано свое отношение реляционной схемы.

Два следующих правила, так же как и первое, касаются бинарных множеств связей типа 1:1. Но, если первое правило относилось к случаю, когда оба определяемые им функциональные отображения были полными, то в третьем правиле одно из них полное, второе – частичное, а в четвертом – частичны оба.

Третье правило предлагает представлять множество связей дублированием первичного ключа одного отношения в другое. Обратите внимание, что образованный при этом внешний ключ является, с одной стороны, возможным ключом отношения, а, с другой стороны, является обязательным атрибутом. Таким образом в реляционной модели представлены, во-первых, функциональность отображения *ПЕРСОНАЛ -> ПАЛАТА*, а, во-вторых, полнота отображения *ПАЛАТА -> ПЕРСОНАЛ*. Функциональность последнего отображения обеспечивается тем, что атрибут, однозначно определяющий служащего (*Персонал_ID*), добавлен в отношение *ПАЛАТА*, а, значит, в силу требования первой нормальной формы каждой палате, представленной в этом отношении одним кортежем, в принципе нельзя сопоставить нескольких служащих.

Еще раз обратим ваше внимание на то, что полнота отображения и зависимость существования (Е-зависимость) являются следствиями друг друга и фактически означают одно и то же.

Имеет смысл проанализировать примеры, приведенные на двух последних слайдах, чтобы почувствовать разницу образованных реляционных схем. В первом случае (два полных функциональных отображения) последовательное применение первого и второго правил набора построит одно отношение, каждый кортеж которого описывает и палату, и служащего, и их связь между собой. Причем в силу взаимной слабости сущностей все компоненты кортежа будут всегда заполнены.

Если для примера из третьего правила применить то же единственное отношение, что и в предыдущем случае, то для кортежей, описывающих служащих, не имеющих связь с конкретной палатой, часть этих кортежей для значений атрибутов множества сущностей *ПАЛАТА* будет пустой.

На лицо какая-то аномалия. В первом случае все ясно, потому что при отмеченных особенностях ПрО число палат равно числу служащих и равно числу их связей между собой. При возникновении в БД одного из компонентов обязательно должны быть определены и два других. Во втором случае (в предположениях третьего правила) кроме

кортежей, имеющих также все три компонента, законными являются и такие кортежи, которые описывают только служащего.

Представьте ситуацию, при которой служащий выполнял в больнице общие функции и не был привязан к палате. Затем его перевели на обслуживание палаты взамен уволенного сотрудника. В соответствующем кортеже палаты надо произвести модификацию атрибутов, описывающих нового служащего, и удалить старый его кортеж. Возможны также симметричные действия. В любом случае определенные практические неудобства ощущаются.

Чтобы снять эту аномалию, усовершенствованный набор правил предлагает строить два отношения – одно для служащих (независимо от наличия или отсутствия связи с палатой), второе для палат и их связей со служащими.

В такой схеме вышеуказанная задача сводится к изменению значения атрибута *Персонал_ID* отношения *ПАЛАТА*, определяющего связь палаты с персоналом, и удалению кортежа с данными о бывшем служащем из отношения *ПЕРСОНАЛ*. Согласитесь, эти действия более соответствуют реальным событиям в ПрО.

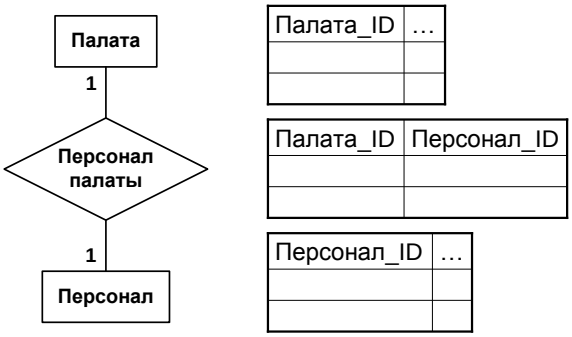
Следует, правда, отметить, что третье правило, как и многие другие правила этого набора, приводит к появлению дополнительных отношений, которые на первый взгляд являются лишними. С практической точки зрения это чревато дополнительными операциями соединения. Действительно, если в ситуации третьего правила создать одно, а не два отношения, для получения информации о том, кто обслуживает палаты, нет потребности соединять два отношения.

В конечном счете, выбор, применять или не применять правила нового набора, ложится на проектировщика. В любом случае, знать о них надо и в каждой конкретной ситуации принимать обоснованное решение. Одним из факторов при этом является степень неполноты отображения *ПЕРСОНАЛ -> ПАЛАТА*. Если отсутствие связи с палатой – стабильное свойство небольшого числа служащих, предпочтительнее одно отношение. В противном случае лучше использовать два отношения.

Подобные рассуждения уместны для многих правил нового набора, касающихся представления бинарных множеств связей. В дальнейшем для краткости мы не будем их повторять, но это вовсе не означает, что их не надо иметь в виду.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

4. **Бинарное МСв типа $R(S1(0,1):S2(0,1))$ представляется самостоятельным отношением, куда дублируются первичные ключи отношений, построенных для МСу S1 и S2. Каждый из этих внешних ключей является возможным ключом нового отношения и имеет описатель NOT NULL.**



Четвертое правило применяется, когда бинарное множество связей типа 1:1 определяет два частичных функциональных отображения. В этом случае логически безупречной будет схема, в которой это множество связей представляется самостоятельным отношением с внешними ключами, идентифицирующими сущности обоих типов.

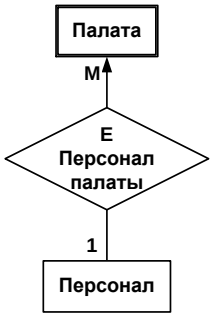
Если попытаться представить эту ситуацию реляционной схемой с одним (правила 1 и 2) или двумя (правило 3) отношениями, мы опять столкнемся с аномалиями. Рассмотрим первый вариант – ограничимся одним отношением. Часть его кортежей будет определять необслуживаемые палаты, часть – служащих, не привязанных к палатам, и только часть – и палату, и служащего, и их соответствие друг другу. Представим ситуацию, когда ранее не обслуживавшаяся палата и не работавший в палатах служащий связываются между собой. Раньше они представлялись двумя различными кортежами, теперь их надо слить в один кортеж. Необходимо либо создать новый кортеж, удалив два старых, либо выполнить модификацию одного из старых кортежей и удалить другой. Ситуация явно аномальная. Вариант с двумя отношениями тоже небезупречен.

Все проблемы практического использования БД снимает вариант схемы, предлагаемый четвертым правилом. Одно отношение описывает только палаты, другое – только служащих, а третье – их связи друг с другом. Для только что описанной ситуации необходимо просто создать кортеж в отношении, построенном для множества связей. Что может быть проще и естественней. Более того, соответствующие ограничения внешних ключей позаботятся об автоматическом удалении кортежей-связей при удалении кортежей-родителей.

Правда, следует заметить, в ситуации четвертого правила надо быть еще более осмотрительным, поскольку в этом случае для восстановления полной информации о связях палат с персоналом потребуется уже две операции соединения отношений.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

5. Бинарное МСв типа $R(S1(1,1):S2(\_,\infty))$ (_ означает, что минимальное кардинальное число отображения не имеет значения) представляется дублированием первичного ключа отношения, построенного для МСу S1 , в отношение, построенное для МСу S2. Этот внешний ключ имеет описатель NOT NULL.



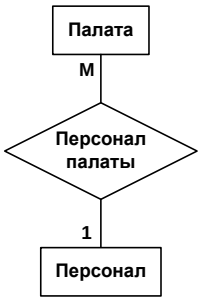
Палата_ID	…	Персонал_ID

Персонал_ID	…

Следующие два правила относятся к бинарным множествам связей типа 1:*M*. Пятое правило регламентирует ситуацию, когда функциональное отображение является к тому же полным. В этом случае без всяких сомнений надо представлять такое множество связей с помощью внешнего ключа. Поскольку отображение полно, он к тому же будет обязательным атрибутом.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

6. Бинарное МСв типа $R(S1(0,1):S2(\_,\infty))$ представляется самостоятельным отношением, куда дублируются первичные ключи отношений, построенных для МСу S1 и S2. Внешний ключ, являющийся дубликатом первичного ключа второго отношения (для S2), представляет собой возможный ключ нового отношения. Оба атрибута этого отношения имеют описатель NOT NULL.



Палата_ID	…

Палата_ID	Персонал_ID

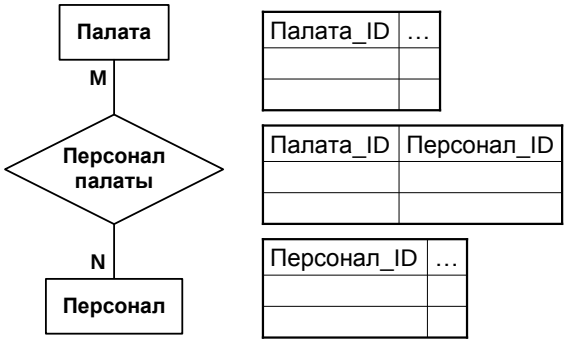
Персонал_ID	…

Если функциональное отображение частично, усовершенствованный набор правил предлагает построить для множества связей типа 1:*M* самостоятельное отношение. Обратите внимание на то, что один из внешних ключей этого отношения является его возможным ключом, а второй – нет.

Целесообразность применения этого правила также следует внимательно проанализировать. Возможно, лучше следует воспользоваться пятым правилом и в этом случае.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

7. **Бинарное МСв типа M:N представляется самостоятельным отношением, куда дублируются первичные ключи отношений, построенных для МСу S1 и S2. Возможным ключом нового отношения является группа, состоящая из обоих внешних ключей, и оба они имеют описатель NOT NULL.**



Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

8. **МСв степени больше двух представляется самостоятельным отношением, куда дублируются первичные ключи отношений, построенных для МСу. Анализ ограничений целостности в этом случае требует более детального рассмотрения.**
9. **Если МСв имеет свои собственные однозначные атрибуты, они добавляются в то отношение, куда осуществляется дублирование первичного(-ых) ключа(-ей) для представления этого множества связей.**
10. **Каждый многозначный атрибут МСу представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для МСу; второй атрибут этого отношения предназначен собственно для значения.**

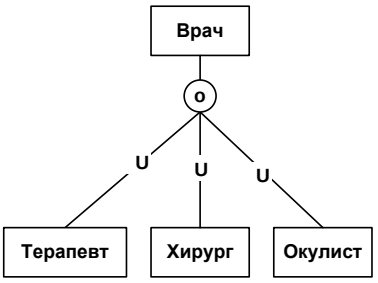
Что касается случаев бинарного множества связей типа *M*:*N* и множества связей степени больше двух, в которых применяются правила 7 и 8, то здесь все как и прежде, никаких новых идей.

Девятое правило определяет действия в случае, когда множество связей имеет свои собственные однозначные атрибуты. Понимать его надо следующим образом. Во всех предыдущих правилах использовались два способа представления множеств связей – самостоятельным отношением и через внешний ключ. И в том, и в другом случае создавались дубликаты первичных ключей множеств сущностей – внешние ключи, только в первом – в новом отношении для множества связей, а во втором – в ранее определенном отношении для множества сущностей. Именно в этих отношениях и следует разместить однозначные атрибуты множеств связей.

Десятое правило, касающееся многозначных атрибутов множеств сущностей, также осталось без изменений.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

11. **Каждый многозначный атрибут МСв представляется отдельным отношением, куда дублируется первичный ключ отношения, построенного для МСв; второй атрибут этого отношения предназначен собственно для значения.**
12. **Каждая ветвь специализаций и категоризаций трактуется как обычное бинарное МСв типа 1:1, которое преобразуется согласно указанным ранее правилам в зависимости от минимальных кардинальных чисел отображений.**



Врач_ID	…

Терапевт_ID	Врач_ID	…

Хирург_ID	Врач_ID	…

Окулист_ID	Врач_ID	…

Одиннадцатое правило, хотя и сохранилось неизменным, требует небольшого пояснения. Под отношениями, построенными для множеств связей, понимаются все те же два вида отношений, которые фигурировали в комментариях к девятому правилу.

Двенадцатое правило, хоть и относится к EER-модели (расширенной ER-модели), тем не менее, присутствует в усовершенствованном наборе правил для ER-модели. Оно предлагает довольно примитивный, но простой и универсальный способ поведения проектировщика в случае наличия в семантической схеме специализаций или категоризаций.

Под ветвью специализации понимается связь между родительским множеством сущностей и одним из дочерних множеств сущностей, а под ветвью категоризации – связь между категорией и одним из суперклассов. Правило предлагает заменять каждую такую ветвь бинарным множеством связей типа 1:1 между теми же множествами сущностей и далее действовать по ранее обсуждавшимся правилам.

Что касается специализаций, то всегда отображение из подкласса в суперкласс будет полным, а обратное ему отображение будет, скорее всего, частичным. У категоризаций отображение из категории в суперкласс всегда частично, обратное отображение зависит от вида категоризации: для полных – оно полно, для частичных – оно частично.

На слайде приведена частичная пересекающаяся специализация врачей. В соответствии с предложенным правилом имеем три бинарных множества связей типа 1:1, определяющих одно полное и одно частичное отображения. Эта ситуация регламентируется третьим правилом, по которому и получены изображенные на слайде отношения.

Усовершенствованные правила перехода от ER-схемы к реляционной схеме БД (продолжение)

13. **Перенос в реляционную схему ограничений целостности на значения атрибутов, как правило, весьма прост и определяется выразительными возможностями конкретного языка определения данных.**

Последнее правило дополняет картину преобразований «ER-схема – реляционная схема» и, не детализируя подробности, напоминает не забыть о других ограничениях целостности, в частности, об ограничениях на значения атрибутов.

Как вы знаете, расширенная ER-модель (EER-модель) отличается от классической ER-модели Чена лишь дополнительными понятиями «специализация» и «категоризация». Эти конструкции позволяют включить в семантическую схему иерархии обобщения множеств сущностей, часто являющиеся важным звеном семантики ПрО, которое должно найти свое отражение в схеме БД.

Исследователи моделей данных давно пытаются найти адекватные представления специализаций и категоризаций в реляционной схеме. Одно из них мы изложили, повествуя об усовершенствованном наборе правил трансформации. В этой части пособия мы расскажем о некоторых других методах.

Но, к великому сожалению, мы не сможем предложить вам детерминированный набор правил трансформации специализаций и категоризаций в реляционную модель по той простой причине, что его пока нет или, по крайней мере, автору он не известен. Это «белое пятно» семантической методики проектирования еще ждет своего Эйнштейна. Тем не менее, мы попытаемся снабдить проектировщика кое-какой информацией, которая поможет ему решать эту задачу в конкретных ситуациях, продиктованных особенностями ПрО.

Наряду с различными методами представления специализаций и категоризаций в реляционной модели, мы обсудим критерии, которые будут направлять процесс преобразования. Ведь в зависимости от преследуемой проектировщиком цели одни и те же методы могут оказаться и плохими, и хорошими.

Но мало определить, каких качеств схемы мы пытаемся достигнуть в ходе преобразования, надо проанализировать и взвешенно учесть многочисленные факторы, влияющие на выбор метода. Как раз одна из самых сложных проблем, связанных с синтезом набора правил преобразования специализаций и категоризаций, касается адекватного учета этих многочисленных взаимосвязанных факторов.

Что-то подсказывает автору, что решение этой проблемы неподвластно детерминированному алгоритму с точными числовыми оценками количественных характеристик, дающему однозначное решение по выбору метода. Скорее, можно надеяться, что адекватный учет всех факторов обеспечит экспертная система, построенная на нечетких правилах. В результате диалога с проектировщиком, в ходе которого последний сообщит системе все интересующие ее особенности семантики ПрО, будет получен нечеткий рекомендательный совет. В нем будет предложено, как правило, несколько методов, возможно ранжированных по степени предпочтения. То есть окончательное решение все-таки останется за человеком.

Мы постараемся дать вам развернутую картину факторов, влияющих на выбор метода, с тем, чтобы, когда появится такая экспертная система, вы понимали, о чем она вас спрашивает. Ну а пока такой системы нет, воспользуйтесь собственной головой.

Рассказ о методах, используемых для представления специализаций и категоризаций в реляционной модели, будет сопровождаться их демонстрацией на примере специализации врачей и категоризации субъектов права, о которых вам напоминает слайд.

Трансформация схемы БД из EER-модели в реляционную модель

Методы представления специализаций в реляционной схеме (*n* – количество подклассов).

Метод 1: *1 отношение.* Создается одно отношение, которое включает однозначные атрибуты суперкласса и всех подклассов, т.е. каждый объект в рамках специализации описывается кортежем только одного отношения.

ВРАЧ (Врач_ID,…, Терапевт_ID,…, Хирург_ID,…, Окулист_ID,…)

Метод 2: *n+1 отношение.* Для суперкласса и каждого из подклассов создается по одному отдельному отношению. Один объект может описываться кортежами нескольких отношений: от 1 до *n*+1.

1 ВРАЧ (Врач_ID (PK),…)
ТЕРАПЕВТ (Терапевт_ID (PK),…, Врач_ID (FK))
ХИРУРГ (Хирург_ID (PK),…, Врач_ID (FK))
ОКУЛИСТ (Окулист_ID (PK),…, Врач_ID (FK))

2 ВРАЧ (Врач_ID (PK),…, Терапевт_ID (FK), Хирург_ID (FK), Окулист_ID (FK))
ТЕРАПЕВТ (Терапевт_ID (PK),…)
ХИРУРГ (Хирург_ID (PK),…)
ОКУЛИСТ (Окулист_ID (PK),…)

3 ВРАЧ (Врач_ID (PK),…)
ТЕРАПЕВТ (Терапевт_ID (PK, FK),…)
ХИРУРГ (Хирург_ID (PK, FK),…)
ОКУЛИСТ (Окулист_ID (PK, FK),…)

Для специализаций известно 6 вариантов их представления в реляционной модели. Далее, *n* – количество подклассов.

Метод 1: *1 отношение.* Создается одно отношение, которое включает однозначные атрибуты суперкласса и всех подклассов, т.е. каждый объект в рамках специализации описывается кортежем только одного отношения.

Пожалуй, это самый простой в реализации метод, дающий к тому же максимальную скорость выборки. Но в информатике, как правило, выигрывая в скорости, проигрываешь в объеме памяти. Так вот первый метод – один из самых неэффективных по памяти. Особенно плохие результаты он дает в случае частичных непересекающихся специализаций с большим числом подклассов. При этом отношение, построенное для таких специализаций, представляет собой «сильно разреженную матрицу». Лучше всего этот метод подходит для полных пересекающихся специализаций, в которых каждый член суперкласса одновременно присутствует в большинстве подклассов. В таком случае все кортежи этого единственного отношения будут максимально заполнены значениями атрибутов.

Метод 2: *n+1 отношение.* Для суперкласса и каждого из подклассов создается по одному отдельному отношению. Один объект может описываться кортежами нескольких отношений: от 1 до *n*+1.

Этот метод обеспечивает тот же результат, что и правило усовершенствованного набора. Этого мы достигаем, если создаем внешние ключи в отношениях, построенных для подклассов. Но это лишь один вариант определения связей кортежей подклассов с кортежами суперкласса. Второй вариант предлагает обратный способ – в отношении суперкласса создается *n* внешних ключей, каждый для связи со своим подклассом. И, наконец, есть третий вариант, отличающийся от первого тем, что внешние ключи подклассов одновременно используются в качестве первичных ключей, а свои собственные суррогатные ключи для этих отношений не создаются. Пусть читателя не вводят в заблуждение имена внешних ключей. Это всего лишь дань традиции, согласно которой имя искусственного первичного ключа включает имя отношения.

Понятно, что с точки зрения расходов памяти второй метод выигрывает по сравнению с первым. Но, опять-таки в силу действия все того же закона информатики, он в большинстве случаев проигрывает ему по скорости, ведь для полного представления об объекте необходимо выполнять операции соединения. Причем тем больше, чем выше степень пересечения подклассов. Отсюда ясно, что этот метод больше подходит для частичных непересекающихся специализаций.

Трансформация схемы БД из EER-модели в реляционную модель

Метод 3: *n отношений.* Создается *n* отношений – по одному для каждого подкласса, однозначные атрибуты суперкласса включаются в каждое из этих отношений. Один объект может быть представлен кортежами нескольких отношений: от 1 до *n*.

ТЕРАПЕВТ (Врач_ID,…, Терапевт_ID,…)
ХИРУРГ (Врач_ID,…, Хирург_ID,…)
ОКУЛИСТ (Врач_ID,…, Окулист_ID,…)

Метод 4 (гибрид 2 и 3 методов): *n+1 отношение.* Для суперкласса и каждого из подклассов создается по одному отдельному отношению. Кроме этого, однозначные атрибуты суперкласса добавляются в каждое отношение, построенное для подклассов. Один объект может быть представлен кортежами нескольких отношений: от 1 до *n*.

ВРАЧ (Врач_ID,…)
ТЕРАПЕВТ (Врач_ID,…, Терапевт_ID,…)
ХИРУРГ (Врач_ID,…, Хирург_ID,…)
ОКУЛИСТ (Врач_ID,…, Окулист_ID,…)

Метод 3: *n отношений.* Создается *n* отношений – по одному для каждого подкласса, однозначные атрибуты суперкласса включаются в каждое из этих отношений. Один объект может быть представлен кортежами нескольких отношений: от 1 до *n*.

Этот метод практически идеален для полных непересекающихся специализаций. Во-первых, поскольку все объекты суперкласса будут одновременно членами какого-то подкласса, у нас не будет потребности создавать кортежи только для суперкласса. Во-вторых, в этом случае не возникает дублирования значений атрибутов суперкласса, так как любой объект будет представлен всегда только одним кортежем какого-то из отношений. И, в-третьих, для получения полной информации об объекте необходимо обратиться только к этому единственному кортежу и никаких операций соединения делать не придется.

Однако, этот метод абсолютно неприменим для частичных специализаций (негде сохранить информацию об объектах, являющихся членами только суперкласса) и весьма неэффективен и чреват аномалиями для специализаций с высокой степенью пересечения подклассов (потребность в операциях соединения и дублирование значений атрибутов суперкласса).

Что касается ключей отношений, то внешних, как вы понимаете, в этом случае нет, а вот с первичными ключами можно поступать по-разному. Можно оставить оба ключевых атрибута (один из суперкласса, другой из подкласса), объявив один из них первичным, второй – возможным, а можно какой-то из них не включать в отношение. Выбор одного из этих решений отнюдь нетривиален и зависит от многочисленных факторов, о которых речь пойдет далее.

Метод 4 (гибрид 2 и 3 методов): *n+1 отношение.* Для суперкласса и каждого из подклассов создается по одному отдельному отношению. Кроме этого, однозначные атрибуты суперкласса добавляются в каждое отношение, построенное для подклассов. Один объект может быть представлен кортежами нескольких отношений: от 1 до *n*.

Этот метод, с одной стороны, может объединить достоинства второго и третьего методов, но, с другой стороны, может усилить и их недостатки. Поэтому применять его следует, пожалуй, лишь в случае непересекающихся специализаций. Только в этом случае каждый объект будет представлен кортежем одного отношения. Если при этом специализация полна, это будет кортеж отношения для подкласса, если она частична, то это будет либо кортеж отношения для подкласса, либо кортеж отношения для суперкласса. Для пересекающихся специализаций этот метод даст неэффективное, аномальное решение. По поводу ключей отношений, полученных четвертым методом, можно сделать то же замечание, что и в комментарии к третьему методу.

Трансформация схемы БД из EER-модели в реляционную модель

Метод 5: *n+2 отношения.* Помимо *n* отношений для подклассов и одного для суперкласса создается связующее отношение, которое имеет следующие атрибуты: имя родительского отношения; первичный ключ объекта в родительском отношении; имя дочернего отношения; первичный ключ объекта в дочернем отношении. Таким образом, один объект может описываться кортежами нескольких отношений: от 1 до *n*+2.

ВРАЧ (Врач_ID (PK),…)
ТЕРАПЕВТ (Терапевт_ID (PK),…)
ХИРУРГ (Хирург_ID (PK),…)
ОКУЛИСТ (Окулист_ID (PK),…)
РОД_РЕБ (Имя_Род_Отн, Род_Отн_ID, Имя_Реб_Отн, Реб_Отн_ID)

Описанные ранее методы, хоть и определялись на примере одной специализации, могут применяться также и для многоуровневых иерархий специализаций. Причем для разных специализаций такой иерархии можно применить как один и тот же метод, так и разные методы. Два предлагаемых далее метода представляют собой специализированные способы определения именно иерархий специализаций, их применение для изолированных специализаций, как правило, неэффективно.

Метод 5: *n+2 отношения.* Помимо *n* отношений для подклассов и одного для суперкласса создается связующее отношение, которое имеет следующие атрибуты: имя родительского отношения; первичный ключ объекта в родительском отношении; имя дочернего отношения; первичный ключ объекта в дочернем отношении. Таким образом, один объект может описываться кортежами нескольких отношений: от 1 до *n*+2. Причем в связующем отношении к этому объекту может относиться несколько кортежей.

Мы определили пятый метод также на примере двухуровневой специализации, но это определение легко обобщается на случай многоуровневой специализации. Так, если иерархия включает *N* множеств сущностей, то для каждого из них создается самостоятельное отношение, для идентификации кортежей которого используется однотипный суррогатный первичный ключ. Для представления информации о том, что кортежи различных отношений описывают один и тот же объект ПрО, создается одно дополнительное связующее отношение с указанной на слайде структурой.

Положительным качеством этого метода является то, что он без каких-либо изменений позволяет представлять не только иерархии специализаций, но и более сложные по структуре графы обобщения. Такие взаимосвязи между множествами сущностей возникают, когда один или несколько подклассов имеет связи с разными суперклассами в рамках различных специализаций. В объектно-ориентированной парадигме этой ситуации соответствует термин «множественное наследование».

К недостаткам метода можно отнести то, что в связующем отношении невозможно определить декларативные ограничения внешних ключей и уникальности образов-родителей и образов-детей. Впрочем, и ограничения на специализации тоже придется программировать.

Трансформация схемы БД из EER-модели в реляционную модель

Метод 6: *2^n отношений.* Для любого возможного поддерева в иерархии специализаций, включающего корневой суперкласс, создается одно отношение, которое содержит в качестве атрибутов все однозначные атрибуты множеств сущностей, принадлежащих поддереву. Таким образом, каждый объект описывается кортежем одного и только одного отношения.

ВРАЧ (Врач_ID,…)
ТЕРАПЕВТ (Врач_ID,…, Терапевт_ID,…)
ХИРУРГ (Врач_ID,…, Хирург_ID,…)
ОКУЛИСТ (Врач_ID,…, Окулист_ID,…)
ТЕРАПЕВТ-ХИРУРГ (Врач_ID,…, Терапевт_ID,…, Хирург_ID,…)
ТЕРАПЕВТ-ОКУЛИСТ (Врач_ID,…, Терапевт_ID,…, Окулист_ID,…)
ХИРУРГ-ОКУЛИСТ (Врач_ID,…, Хирург_ID,…, Окулист_ID,…)
ТЕРАПЕВТ-ХИРУРГ-ОКУЛИСТ (Врач_ID,…, Терапевт_ID,…, Хирург_ID,…, Окулист_ID,…)

Метод 6: *2^n отношений.* Для любого возможного поддерева в иерархии специализаций, включающего корневой суперкласс, создается одно отношение, которое содержит в качестве атрибутов все однозначные атрибуты множеств сущностей, принадлежащих поддереву. В конечном счете, каждый объект ПрО относится к одному из этих 2^n классов, и, значит, его кортеж надо создавать в соответствующем отношении. Таким образом, каждый объект в рамках рассматриваемого метода описывается кортежем одного и только одного отношения.

Скорее всего, предлагаемый последним метод, будучи реализован в чистом виде, даст в большинстве случаев не очень удовлетворительный результат с практической точки зрения. Но вот применение его в некоторых экзотических случаях может дать положительный эффект.

Представьте себе, что в вашей семантической схеме имеется многоуровневая иерархия специализаций, а, может быть, и еще более сложная конструкция обобщения. Понятно, что в последнем случае количество различных классов объектов, которые можно выделить в зависимости от принадлежности объектов к тому или иному множеству сущностей, становится гигантским. Но, предположим, что среди этих классов есть такой, которому принадлежит львиная доля объектов суперкласса.

В такой ситуации имеет смысл для этого класса выделить в графе обобщения соответствующий подграф и в соответствии с шестым методом в виде исключения создать для него в реляционной схеме самостоятельное отношение. Для немногочисленных объектов других классов можно применить какие-нибудь другие методы.

Трансформация схемы БД из EER-модели в реляционную модель

Методы представления категоризаций в реляционной схеме (*n* – количество суперклассов).

Метод 1: *n отношений.* Данный метод предполагает, что все однозначные атрибуты категории становятся атрибутами в отношениях, представляющих суперклассы. Отношение для категории не создается.

ФЛ (ФЛ_ID,..., СП_ID,...)
ЮЛ (ЮЛ_ID,..., СП_ID,...)
АТО (АТО_ID,..., СП_ID,...)

Метод 2: *n+1 отношение.* Для подкласса (категории) и каждого из суперклассов создается отдельное отношение. Варианты способов организации связей между кортежами отношений суперклассов и кортежами отношения категории аналогичны предлагавшимся для соответствующего метода специализаций.

СП (СП_ID, ...)
ФЛ (ФЛ_ID,..., СП_ID)
ЮЛ (ЮЛ_ID,..., СП_ID)
АТО (АТО_ID,..., СП_ID)

Метод 3: *n+2 отношения.* Аналогичен методу 5 для специализаций.

Отнюдь не все методы, предложенные для специализаций, подходят для представления категоризаций. Сказывается то, что, во-первых, не все объекты, представленные в суперклассах, обязаны быть и в подклассе, а, во-вторых, суперклассы в принципе не пересекаются. К тому же, неуместны иерархии категоризаций.

Можно предложить следующие методы представления категоризаций в реляционной схеме (*n* – количество суперклассов).

Метод 1: *n отношений.* Данный метод предполагает, что все однозначные атрибуты категории становятся атрибутами в отношениях, представляющих суперклассы. Отношение для категории не создается.

Этот метод больше подходит для полных категоризаций, поскольку в этом случае все атрибуты всех отношений с гарантией будут заполнены определенными значениями во всех кортежах.

Метод 2: *n+1 отношение.* Для подкласса (категории) и каждого из суперклассов создается отдельное отношение. Варианты способов организации связей между кортежами отношений суперклассов и кортежами отношения категории аналогичны предлагавшимся для соответствующего метода специализаций.

Второй метод, наоборот, даст более подходящее решение для частичных категоризаций, особенно если степень участия объектов суперклассов в категории невелика.

Метод 3: *n+2 отношения.* Аналогичен методу 5 для специализаций.

Третий метод, пожалуй, можно применить лишь в случае, когда члены суперклассов и/или подкласса одновременно участвуют в специализациях. В такой ситуации связующее отношение можно с успехом использовать одновременно для представления связей как специализаций, так и категоризации.

Трансформация схемы БД из EER-модели в реляционную модель

Предлагается при выборе метода представления специализаций и категоризаций руководствоваться совокупностью следующих **критериев**:

1. Простота реализации. Помимо собственно времени, затрачиваемого на реализацию метода, критерий может включать и специфические требования, такие как учет потенциальной необходимости вносить изменения в реляционную схему (в частности, возможность появления дополнительных подклассов).

2. Эффективность хранения. Имеется возможность за счет выбора метода снизить влияние некоторых факторов, отрицательно сказывающихся на эффективности хранения:

а) дублирование значений атрибутов:

– дублирование значений ключей;

– дублирование значений остальных атрибутов;

б) представление отсутствующих значений (NULL).

3. Эффективность обработки:

а) обновление;

б) проверка ограничений целостности (совокупность ограничений целостности, заданных как декларативно, так и процедурно, неизменна, однако, сложность проверки одного и того же ограничения будет различной в зависимости от выбранного метода);

в) выборка.

Как уже отмечалось, не существует идеального метода представления специализаций и категоризаций в реляционной модели, подходящего для любой ПрО. Более того, в зависимости от требуемых качеств схемы БД один и тот же метод может быть как хорош, так и плох. Пора поговорить об этих качествах схемы и иметь их в виду, решая задачу выбора метода.

Предлагается при выборе метода представления специализаций и категоризаций руководствоваться совокупностью следующих **критериев.**

1. Простота реализации. Помимо собственно времени, затрачиваемого на реализацию метода, критерий может включать и специфические требования, такие как учет потенциальной необходимости вносить изменения в реляционную схему (в частности, возможность появления дополнительных подклассов).

Другим примером специфических требований является желание упростить реализацию изменений в отношении принадлежности объекта к тем или иным подклассам специализации. Речь идет, например, о ситуации, когда врач получает или лишается лицензии по специальности. Одни методы могут упростить удовлетворение этих требований, в то время как другие – усложнить.

2. Эффективность хранения. Имеется возможность за счет выбора метода снизить влияние некоторых факторов, отрицательно сказывающихся на эффективности хранения:

а) дублирование значений атрибутов:

- дублирование значений ключей;

- дублирование значений остальных атрибутов;

б) представление отсутствующих значений (*NULL*).

Если у вас на счету каждый байт внешней памяти, этот критерий является для вас основным. Особенно это касается СУБД, не обеспечивающих эффективные методы хранения данных на диске, в частности, неопределенных значений.

3. Эффективность обработки:

а) обновления;

б) проверки ограничений целостности (совокупность ограничений целостности, заданных как декларативно, так и процедурно, неизменна, однако, сложность проверки одного и того же ограничения будет различной в зависимости от выбранного метода);

в) выборки.

Последний критерий предлагает оценивать методы с точки зрения эффективности процедур обработки данных. При этом следует проанализировать:

- какие команды будут выполняться при изменениях объектов ПрО, насколько они трудоемки;

- будут ли ограничения целостности, связанные со специализациями и категоризациями, представлены декларативно или потребуется написание специальных программ, какова их трудоемкость;

- обеспечит ли метод представления специализаций и категоризаций требуемую эффективность выборки.

Трансформация схемы БД из EER-модели в реляционную модель

Для специализаций необходимо учесть следующие **факторы**:

1. Количество атрибутов и множеств связей, ассоциированных с суперклассом и подклассами.

2. Свойства специализации (полное или частичное участие членов суперкласса в подклассах, пересечение или непересечение подклассов).

3. Степень пересечения подклассов в случае пересекающейся специализации.

4. Степень неполноты в случае специализации с частичным участием.

5. Количество подклассов специализации, их динамизм.

При выборе метода для специализаций необходимо учесть следующие **факторы**:

1. Количество атрибутов и множеств связей, ассоциированных с суперклассом и подклассами.

Понятно, что в зависимости от наличия у суперкласса и подклассов собственных атрибутов и множеств связей, а также от их количества стоит задуматься о необходимости создания для них самостоятельных отношений. С другой стороны, может, следует предпочесть другие методы, при которых они не образуются.

2. Свойства специализации (полное или частичное участие членов суперкласса в подклассах, пересечение или непересечение подклассов).

3. Степень пересечения подклассов в случае пересекающейся специализации.

4. Степень неполноты в случае специализации с частичным участием.

Мы уже характеризовали методы с точки зрения их преимущественного использования для тех или иных видов специализаций (полных/частичных, пересекающихся/непересекающихся). Как вы помните, важными особенностями при этом являлись степень пересечения подклассов и степень неполноты специализации. На это следует обратить особое внимание, так как некоторые методы категорически неприемлемы в некоторых случаях.

5. Количество подклассов специализации, их динамизм.

Огромную роль играет при выборе метода количество подклассов. Одно дело, когда их 2-3, и другое дело, когда их десятки. Динамизм подклассов характеризуется как интенсионально (подклассы могут возникать и исчезать), так и экстенсионально (объекты в течение жизни могут менять свою принадлежность к подклассам). В каждом случае следует внимательно проанализировать потребности ПрО, касающиеся этого фактора, и обоснованно выбрать адекватный метод.

Трансформация схемы БД из EER-модели в реляционную модель

6. Наличие множеств связей или многозначных атрибутов, для представления которых правила преобразования в реляционную схему создают для некоторого отношения внешние ключи, ссылающиеся на ключ суперкласса, но не входящие в состав первичного ключа этого отношения.

7. Выполнение условия, аналогичного условию пункта 6, для одного или нескольких подклассов.

Факторы 6-9 связаны с необходимостью дублировать значения первичных ключей множеств сущностей, составляющих специализацию, в другие отношения.

6. Наличие множеств связей или многозначных атрибутов, для представления которых правила преобразования в реляционную схему создают для некоторого отношения внешние ключи, ссылающиеся на ключ суперкласса, но не входящие в состав первичного ключа этого отношения.

Предположим, что в ER-схеме наряду со специализацией врачей имеется множество связей *ВРАЧ-ПАЦИЕНТ* типа *M*:*N* между множествами сущностей *ВРАЧ* и *ПАЦИЕНТ*. Посмотрим, какие имеются варианты его реализации, если для представления специализации выбрать 3-й метод (при этом отношение для суперкласса не создается). На какой первичный ключ должен ссылаться внешний ключ *Врач_ID* отношения *ВРАЧ-ПАЦИЕНТ*?

Ни один из ключей отношений *ТЕРАПЕВТ, ХИРУРГ* и *ОКУЛИСТ* не обеспечивает полного набора допустимых значений этого внешнего ключа; он образуется лишь их объединением. Очевидно, что в этом случае декларативно задать ограничение ссылочной целостности мы не можем.

Есть, конечно, вариант с созданием в отношении *ВРАЧ-ПАЦИЕНТ* трех внешних ключей *Терапевт_ID, Хирург_ID, Окулист_ID,* ссылающихся на соответствующие первичные ключи этих трех отношений. Но это тоже не очень подходящий выход.

Очевидно, что в этом случае лучше избегать использования методов, не образующих отношение для суперкласса.

7. Выполнение условия, аналогичного условию пункта 6, для одного или нескольких подклассов.

Ситуация похожа на рассмотренную ранее, только в этом случае дублировать необходимо значения первичного ключа подкласса. Здесь проблемным может оказаться первый метод, вообще не создающий отношения для подклассов. В отношении, построенном для суперкласса, первичным определяется именно его ключ. Более того, несколько первичных ключей в отношении быть не может. Многие СУБД, правда, позволяют внешним ключам ссылаться и на возможные ключи (атрибуты с описателем *UNIQUE*) отношений. Но как-то они отнесутся к тому, что эти атрибуты могут принимать неопределенные значения. А атрибуты *Терапевт_ID, Хирург_ID, Окулист_ID* являются как раз такими.

Наверное, в этом случае тоже лучше использовать другой метод.

Трансформация схемы БД из EER-модели в реляционную модель

8. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ суперкласса, а также входят в состав первичного ключа данного отношения.

9. Выполнение условия, аналогичного указанному в пункте 8, для одного или нескольких подклассов.

10. Использование несуррогатных (естественных) ключей для суперкласса или подклассов.

8. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ суперкласса, а также входят в состав первичного ключа данного отношения.

Если в ситуации, рассматривавшейся в комментарии к 6-му фактору, нам удалось все-таки объявить три внешних ключа, то вот определить с их помощью первичный ключ отношения *ВРАЧ-ПАЦИЕНТ* нам не удастся. Все атрибуты первичного ключа должны быть *NOT NULL*. А атрибуты *Терапевт_ID, Хирург_ID, Окулист_ID* этого отношения таковыми не являются.

9. Выполнение условия, аналогичного указанному в пункте 8, для одного или нескольких подклассов.

Продолжая рассмотрение ситуации, упомянутой в комментарии к 7-му фактору, отметим лишь возможные проблемы, которые касаются того факта, что внешние ключи, ссылающиеся на неопределенные (*NULL*) возможные ключи *Терапевт_ID*, *Хирург_ID*, *Окулист_ID*, необходимо в этом случае к тому же объявлять как определенные (*NOT NULL*). Это неотъемлемое свойство атрибутов, входящих в состав первичного ключа.

Чтобы избежать таких проблем, лучше использовать в этой ситуации метод, в котором создаются отношения для подклассов.

10. Использование несуррогатных (естественных) ключей для суперкласса или подклассов.

Надеюсь, вы помните наши рассуждения по поводу потенциальной опасности использования естественных атрибутов в качестве первичных ключей отношений. Так вот для представления специализаций эта опасность становится реальной в случае использования некоторых методов. А пятый метод в этой ситуации вообще неприменим, поскольку естественные ключи практически никогда не являются однотипными.

Отметим, что сочетание 10-го фактора с факторами 6-9 создает совершенно непреодолимые проблемы.

Трансформация схемы БД из EER-модели в реляционную модель

Для категоризаций необходимо учесть следующие **факторы**:
1. Количество атрибутов и множеств связей, ассоциированных с категорией и суперклассами.
2. Свойство полноты/неполноты участия для категоризации.
3. Степень неполноты в случае категоризации с неполным участием.
4. Количество суперклассов, их динамизм.
5. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ категории, но не входят в состав первичного ключа данного отношения.
6. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ категории, а также входят в состав первичного ключа данного отношения.
7. Использование естественных ключей для суперклассов и подкласса.

Отличительные особенности категоризаций и предлагаемых для них методов представления в реляционной модели накладывают свой отпечаток на набор факторов, которые надо учесть при выборе метода. Хотя большая часть этих факторов схожа с факторами для специализаций. Поэтому мы позволим себе подробно их не пояснять, надеясь на то, что аналогичные ранее высказанным рассуждения читатель проведет самостоятельно.

При выборе метода для категоризаций необходимо учесть следующие **факторы**:

1. Количество атрибутов и множеств связей, ассоциированных с категорией и суперклассами.

2. Свойство полноты/неполноты участия для категоризации.

3. Степень неполноты в случае категоризации с неполным участием.

4. Количество суперклассов, их динамизм.

5. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ категории, но не входят в состав первичного ключа данного отношения.

6. Наличие множеств связей или многозначных атрибутов, порождающих в процессе трансформации в реляционную схему отношения, внешние ключи которых ссылаются на ключ категории, а также входят в состав первичного ключа данного отношения.

7. Использование естественных ключей для суперклассов и подкласса.

Поскольку методы, предлагаемые для категоризаций, всегда предусматривают создание отношений для суперклассов, рассматривать факторы, аналогичные 7-му и 9-му факторам для специализаций, не имеет смысла.

Методика трансформации схемы базы данных из ERM-модели в реляционную модель

Критерии качества проекта БД

1. **Информационная полнота**: схема БД удовлетворяет этому критерию, если она полностью обеспечивает информационные потребности всех будущих пользователей системы.
2. **Информационная корректность**: схема БД удовлетворяет этому критерию, если в ней нашли отражение все выявленные на этапе анализа ПрО типичные свойства информационных объектов – ограничения целостности.
3. **Информационная неизбыточность**: схема БД удовлетворяет этому критерию, если в ней отсутствуют избыточные структуры данных и ограничения целостности. Избыточными будут структуры, предназначенные для представления высказываний, которые могут быть получены из данных другой более оптимальной структуры с помощью операций манипулирования данными. Ограничение целостности избыточно, если оно является следствием других ограничений целостности.

Применение семантической методики проектирования схем БД с использованием ERM-модели в качестве семантической модели предполагает не только оригинальную методику семантического моделирования (п. 3.3.6), но также не менее оригинальную методику преобразования ERM-схемы в реляционную схему. Если классическая методика проектирования реляционной схемы и методика преобразования ER-схемы в реляционную схему обеспечивают результат в ходе выполнения единого алгоритма, и, непонятно, на каком шаге этого алгоритма мы обеспечиваем те или иные качества схемы БД, то предлагаемая далее методика совершенно четко фиксирует цель каждого этапа и те действия, которые приводят к этой цели наилучшим образом.

Напомним, что без определения схемы БД не может идти речь об информационной системе, созданной по технологии БД. Этот элемент системы является ее основой, по сути, единственным обязательным проектным артефактом (за исключением естественно самой СУБД, операционной системы и компьютера, на котором все это функционирует).

Неадекватная схема часто приводит к неудовлетворенности конечных пользователей, дополнительным проблемам администратора БД и разработчиков прикладного программного обеспечения. Естественно возникает вопрос о критериях, удовлетворение которых способствует повышенному качеству этого столь важного элемента системы.

Основными критериями качества проекта схемы БД в порядке убывания важности (каждый последующий критерий рассматривается при максимально полном удовлетворении всех предыдущих) являются следующие.

Информационная полнота: схема БД удовлетворяет этому критерию, если она полностью обеспечивает информационные потребности всех будущих пользователей информационной системы. В контексте методики ERM-моделирования это означает, что она в состоянии воспринять все возможные входные единичные высказывания и сохранить их таким образом, чтобы быть в состоянии продуцировать на их основе все требуемые производные высказывания.

Информационная корректность: схема БД удовлетворяет этому критерию, если в ней нашли отражение все выявленные на этапе анализа ПрО типичные свойства информационных объектов. Речь идет, прежде всего, об ограничениях целостности, проверка которых при всех модификациях БД будет способствовать тому, что БД всегда будет находиться в максимально непротиворечивом состоянии, и, следовательно, ответы системы на информационные запросы пользователей будут более достоверными. При этом следует отдавать себе отчет в том, что полностью возложить задачу актуальности БД на плечи информационной системы невозможно. Она может лишь обеспечить проверку

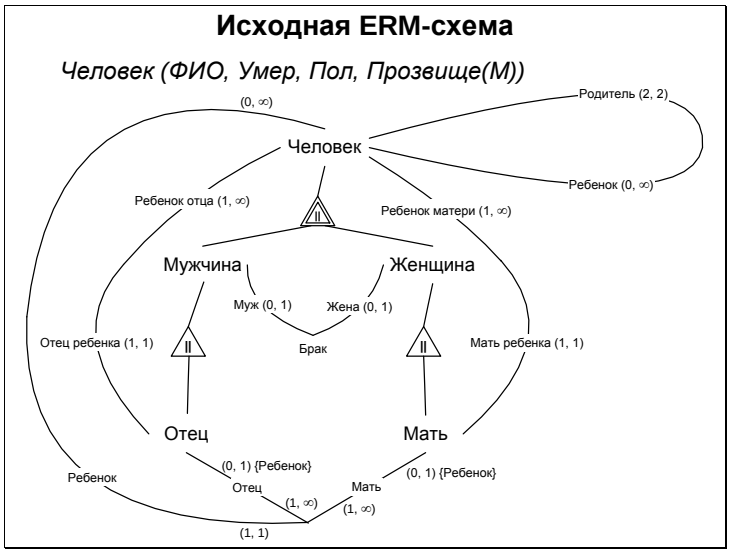
декларированных в схеме БД ограничений целостности. Ясно, что чем больше таких условий удалось сформулировать в ERM-схеме и перенести без потерь в реляционную схему, тем больше степень достоверности откликов системы.

Информационная неизбыточность: схема БД удовлетворяет этому критерию, если в ней отсутствуют избыточные структуры данных и ограничения целостности. Избыточными будут структуры, предназначенные для представления высказываний, которые могут быть получены из данных другой более оптимальной структуры с помощью операций манипулирования данными. Ограничение целостности избыточно, если оно является следствием других ограничений целостности.

В тех случаях, когда модель данных (в частности – реляционная модель) вынуждает прибегать к дублированию данных (необходимое дублирование) для представления связей между элементами БД, этот критерий требует, чтобы такое дублирование было минимальным. Отметим, что следование этому последнему критерию не должно идти в ущерб ранее упомянутым критериям полноты и корректности.

Удовлетворение всех указанных критериев приведет к «хорошему» совокупному качеству схемы БД: первый критерий гарантирует полное удовлетворение потребностей пользователей, второй – высокую степень достоверности информации, третий – эффективность работы системы.

Вся методика трансформации ERM-схемы в реляционную схему представляет собой три группы шагов, последовательное выполнение которых обеспечит результат с наилучшим качеством. Первая группа шагов построит хоть и в большинстве своем избыточные, но удовлетворяющие всем потребностям пользователей структуры данных. Вторая группа шагов обеспечит для этих структур максимально полный перенос ограничений целостности ERM-схемы. И, наконец, третья группа шагов позаботится об исключении избыточных структур и ограничений целостности.



Для демонстрации шагов методики мы воспользуемся очень показательной ПрО родственных отношений, которую мы использовали и для методики семантического ERM-моделирования. Позволим напомнить вам результирующую ERM-схему, полученную в ходе выполнения этой методики. Понятно, что именно она будет для предлагаемой методики трансформации исходным артефактом.

Основными формами представления ERM-схемы являются граф классов и списки характеристик классов. На слайде представлен граф классов ПрО ближайших родственников. Единственным классом, имеющим естественные атрибуты, является множество сущностей *Человек*. Один из них – *Прозвище* – многозначный.

Читателю предлагается прикинуть, сколько отношений понадобится для представления в БД всей информации об этой ПрО. Впоследствии у него будет возможность сравнить свою оценку с результатом, полученным в ходе выполнения нашей методики.

Методика трансформации «ERM-модель – реляционная модель» (шаг 1)

Каждому множеству сущностей соответствует свое отношение, включающее следующие атрибуты:

- суррогатный первичный ключ;
- однозначные атрибуты этого множества сущностей.

Человек (*Человек_ID*, *ФИО*, *Умер*, *Пол*);
Мужчина (*Мужчина_ID*);
Женщина (*Женщина_ID*);
Отец (*Отец_ID*);
Мать (*Мать_ID*).

Информационная полнота БД обеспечивается в первую очередь полнотой набора типичных для ПрО единичных высказываний, определенных на основании анализа информационных потребностей будущих пользователей системы. Этот набор высказываний послужил отправной точкой методики построения ERM-схемы, воплотившей в своих структурах и ограничениях целостности требуемую семантику ПрО.

Теперь эти формальные ERM-описания требуется реализовать выразительными средствами реляционной модели. Но прежде напомним ее основные концепции.

Основной и, по сути, единственной формой хранения данных в ней является отношение (в просторечии – таблица). Классическое определение этой формы мы уже приводили в параграфе 2.2. Интенсионал отношения наряду с его именем составляет множество пар вида $A_i : T_i$, где A_i – имя *i*-го атрибута отношения, а T_i – домен или множество допустимых значений этого атрибута. Экстенсионал отношения представляет собой множество кортежей отношения, структура которых соответствует интенсионалу (или схеме) отношения. Таким образом, кортеж отношения – это множество пар вида $A_i : v_i$, где v_i – значение из домена T_i.

С точки зрения семантики кортеж является знаком, представляющим конкретную сущность или конкретную связь между сущностями. Все отношение в целом служит для хранения информации о сущностях или связях одного типа, составляющих объем одного понятия об индивидах или их *n*-ках.

Для уникальной идентификации кортежа отношения (а вместе с ним и объекта или связи, им представленных) служат первичный и возможные ключи отношения. Таковые могут составлять как естественные атрибуты (возможные ключи класса, определенные на шаге 6 методики ERM-моделирования), так и искусственные (внутрисистемные, суррогатные) атрибуты, пользователям невидимые и недоступные. В силу того, что в реляционной модели связи между кортежами отношений (а стало быть, и между объектами ПрО) представляются совпадением значений атрибутов этих отношений, первичные ключи помимо идентификации кортежа (и благодаря ей) выполняют роль этих связующих атрибутов.

Именно памятуя об этой второй роли первичных ключей, следует в качестве таковых вводить в каждое отношение простые суррогатные атрибуты. Многие реляционные СУБД предусматривают для них особый тип атрибутов (счетчики или последовательности), генерация уникальных значений которых – исключительная прерогатива системы. Значения суррогатных ключей – своеобразные единичные неописательные имена объектов (сущностей или связей). В таком случае кортеж отношения можно рассматривать как единичное описательное имя объекта.

Точно так же как мы использовали имена объектов в высказываниях об отношениях, в кортежах-связях обязательно хранятся значения суррогатных ключей кортежей-сущностей, составляющих эту связь. Такие связующие атрибуты носят название внешних ключей и вместе с первичными или возможными ключами, определяющими область допустимых значений внешних ключей, составляют ограничения ссылочной целостности.

Кроме первичных и внешних ключей множество атрибутов отношения составляют атрибуты–характеристики, соответствующие отображениям-характеристикам, прообразами которых являются объекты, представленные кортежами отношения. Списки таких характеристик определялись на шаге 4 методики ERM-моделирования. Поскольку все отношения реляционной модели по определению должны находиться в первой нормальной форме, для представления многозначных характеристик классов (шаг 6 методики ERM-моделирования) необходимо создавать дополнительные бинарные отношения с внешним ключом для идентификации объекта – обладателя значения многозначной характеристики – второго атрибута этого отношения.

Представление объекта как кортежа отношения соответствует в логике рассмотрению предмета как системы признаков, свойственных ему как элементу объема некоторого понятия или что то же – элементу класса, экстенсиональной формой которого является отношение. В некоторых случаях в БД необходимо восприятие объекта как некой субстанции, единой, несмотря на многочисленные ее представления как элемента различных понятий. Другими словами, часто нужно констатировать тот факт, что кортежи различных отношений соответствуют в ПрО одному и тому же объекту.

В таком случае можно воспользоваться дублированием в эти отношения одних и тех же естественных атрибутов – ключей объекта. Однако надежнее для этих целей также воспользоваться суррогатными ключами.

Если в ERM-схеме БД имеются непересекающиеся по классам иерархии специализаций (у них нет общих понятий-классов), для уникальной идентификации объектов в каждой иерархии достаточно одного общего суррогатного ключа. В корневом классе он объявляется первичным, в каждом подклассе создается его дубликат – внешний ключ, который в свою очередь объявляется первичным, и на него ссылаются внешние ключи его подклассов и т.д.

Если же в ERM-схеме наблюдается так называемое множественное наследование (подкласс имеет несколько суперклассов), для представления связей «суперобъект-подобъект» необходимо специальное отношение следующей структуры:

<имя суперкласса><значение ключа объекта в отношении суперкласса>

<имя подкласса><значение ключа этого же объекта в отношении подкласса>.

В этом случае нет необходимости в использовании одного и того же значения суррогатного ключа объекта во всех отношениях иерархии классов – первичные ключи независимы.

Как видим, предлагаемая методика для простоты изложения фиксирует определенные методы представления специализаций, о категоризациях же вообще умалчивает. На самом деле проектировщик может воспользоваться любым из предложенных ранее методов, если считает, что в его случае он больше подходит и порождает более эффективные структуры данных.

Таким образом, полный перенос структур данных из ERM-модели в реляционную модель осуществляется по следующим правилам.

Шаг 1. Каждому множеству сущностей соответствует свое отношение, включающее следующие атрибуты:

- суррогатный первичный ключ;
- однозначные атрибуты этого множества сущностей.

Имя отношения совпадает с именем множества сущностей. Имя первичного ключа образуется добавлением к имени отношения суффикса «*_ID*». Имена атрибутов-характеристик совпадают с именами соответствующих отображений-характеристик.

Для примера с родственными отношениями имеем следующие отношения для множеств сущностей:

Человек (Человек_ID, ФИО, Умер, Пол);
Мужчина (Мужчина_ID);
Женщина (Женщина_ID);
Отец (Отец_ID);
Мать (Мать_ID).

Методика трансформации «ERM-модель – реляционная модель» (шаг 2)

Для каждого множества связей степени *n* (представленного в графе классов ребром с *n* концами) строим отношение реляционной модели с *n* атрибутами, каждый из которых представляет собой внешний ключ – дубликат первичного ключа отношения, построенного для соответствующего множества сущностей.

Однозначные атрибуты этого множества связей становятся атрибутами отношения, построенного для его представления.

Если множество связей имеет многозначные атрибуты или участвует в специализации множеств связей, в отношении, построенном для него, понадобится выделить первичный ключ.

Родитель-Ребенок (*Родитель_Человек_ID*, *Ребенок_Человек_ID*);
Брак (*Мужчина_ID*, *Женщина_ID*);
Отец-Ребенок (*Отец_ID*, *Человек_ID*);
Мать-Ребенок (*Мать_ID*, *Человек_ID*);
Отец-Мать-Ребенок (*Отец_ID*, *Мать_ID*, *Человек_ID*).

Шаг 2. Для каждого множества связей степени *n* (представленного в графе классов ребром с *n* концами) строим отношение реляционной модели с *n* атрибутами, каждый из которых представляет собой внешний ключ – дубликат первичного ключа отношения, построенного для соответствующего множества сущностей. Именем каждого такого отношения становится соответствующий предикатор. Имена внешних ключей совпадают с именами первичных ключей множеств сущностей. В том случае, когда одно и то же множество сущностей играет сразу несколько ролей в множестве связей, имя роли становится дополнительным префиксом имени внешнего ключа (например, *Родитель_Человек_ID* и *Ребенок_Человек_ID*).

Однозначные атрибуты этого множества связей становятся атрибутами отношения, построенного для его представления.

Если множество связей имеет многозначные атрибуты или участвует в специализации множеств связей, в отношении, построенном для него, понадобится выделить первичный ключ. Таковым может являться группа атрибутов этого отношения, состоящая из внешних ключей и однозначных атрибутов множества связей. Однако лучше создать дополнительный суррогатный первичный ключ.

Применяя данное правило для ПрО родственных связей, получаем следующие отношения для множеств связей:

Родитель-Ребенок (*Родитель_Человек_ID, Ребенок_Человек_ID*);
Брак (*Мужчина_ID, Женщина_ID*);
Отец-Ребенок (*Отец_ID, Человек_ID*);
Мать-Ребенок (*Мать_ID, Человек_ID*);
Отец-Мать-Ребенок (*Отец_ID, Мать_ID, Человек_ID*).

Методика трансформации «ERM-модель – реляционная модель» (шаг 3)

Каждой многозначной характеристике каждого класса (множества сущностей или множества связей) соответствует свое бинарное отношение с атрибутами:

- внешний ключ, представляющий собой дубликат первичного ключа класса;
- атрибут для значения характеристики.

Прозвище человека (*Человек_ID*, *Прозвище*)

Шаг 3. Каждой многозначной характеристике каждого класса (множества сущностей или множества связей) соответствует свое бинарное отношение с атрибутами:

- внешний ключ, представляющий собой дубликат первичного ключа класса;
- атрибут для значения характеристики.

Имя такого отношения составляется из имени характеристики и имени класса в родительном падеже (например, *Прозвище человека*). Имя внешнего ключа совпадает с именем первичного ключа класса. Имя второго атрибута – имя соответствующего многозначного отображения-характеристики.

Для единственной многозначной характеристики множества сущностей *Человек* строим отношение *Прозвище человека (Человек_ID, Прозвище)*.

Методика трансформации «ERM-модель – реляционная модель» (шаг 4)

Если группа классов (множеств сущностей или множеств связей) образует независимую иерархию специализаций, для идентификации объектов всех классов этой группы используется один суррогатный ключ. Он определяется сначала на уровне корневого класса в качестве первичного ключа его отношения. В классах, являющихся ближайшими потомками корневого класса, создаются соответствующие ему внешние ключи, выступающие в то же время в качестве первичных ключей. На них в свою очередь ссылаются внешние ключи потомков 2-го рода и т.д.

Шаг 4. Если группа классов (множеств сущностей или множеств связей) образует независимую иерархию специализаций, для идентификации объектов всех классов этой группы используется один суррогатный ключ. Он определяется сначала на уровне корневого класса в качестве первичного ключа его отношения. В классах, являющихся ближайшими потомками корневого класса, создаются соответствующие ему внешние ключи, выступающие в то же время в качестве первичных ключей. На них в свою очередь ссылаются внешние ключи потомков 2-го рода и т.д. Самостоятельные суррогатные первичные ключи классов-потомков не создаются. Имена ключей образуются по общему для первичных ключей принципу – <*имя отношения*>*_ID*.

Пример

В ПрО родственных отношений имеется одна иерархия множеств сущностей, состоящая из трех специализаций.

Применение этого правила не изменяет структуры спроектированных ранее отношений с точки зрения состава атрибутов.

Следует отметить лишь, что:
- областью значений внешнего ключа *Мужчина_ID* является первичный ключ отношения *Человек – Человек_ID*;
- областью значений внешнего ключа *Женщина_ID* является первичный ключ отношения *Человек – Человек_ID*;
- областью значений внешнего ключа *Отец_ID* является первичный ключ отношения *Мужчина – Мужчина_ID*;
- областью значений внешнего ключа *Мать_ID* является первичный ключ отношения *Женщина – Женщина_ID*.

В ПрО родственных отношений имеется одна иерархия множеств сущностей, состоящая из трех специализаций. Применение этого правила не изменяет структуры спроектированных ранее отношений с точки зрения состава атрибутов.

Следует отметить лишь, что:
- областью значений внешнего ключа *Мужчина_ID* является первичный ключ отношения *Человек – Человек_ID*;
- областью значений внешнего ключа *Женщина_ID* является первичный ключ отношения *Человек – Человек_ID*;
- областью значений внешнего ключа *Отец_ID* является первичный ключ отношения *Мужчина – Мужчина_ID*;
- областью значений внешнего ключа *Мать_ID* является первичный ключ отношения *Женщина – Женщина_ID*.

Методика трансформации «ERM-модель – реляционная модель» (шаг 5)

Для представления фактов множественного наследования (класс принадлежит одновременно специализациям различных родительских классов) создается одно дополнительное отношение с четырьмя атрибутами:

- имя отношения суперкласса (*SUPERCLASS*);
- значение первичного ключа объекта в отношении суперкласса (*SUPER_ID*);
- имя отношения подкласса (*SUBCLASS*);
- значение первичного ключа объекта в отношении подкласса (*SUB_ID*).

Шаг 5. Для представления фактов множественного наследования (класс принадлежит одновременно специализациям различных родительских классов) создается одно дополнительное отношение с четырьмя атрибутами:

- имя отношения суперкласса (*SUPERCLASS*);
- значение первичного ключа объекта в отношении суперкласса (*SUPER_ID*);
- имя отношения подкласса (*SUBCLASS*);
- значение первичного ключа объекта в отношении подкласса (*SUB_ID*).

Методика трансформации «ERM-модель – реляционная модель» (шаг 6)

Ограничения целостности на области значений отображений-характеристик. В реляционной модели их иногда называют ограничениями целостности на значения атрибутов. К ним относятся:

- обязательное указание типа данных для каждого атрибута отношения (с максимальной длиной – для строк и разрядностью – для чисел);
- конструкция *CHECK*.

ФИО CHARACTER(50);
Умер CHARACTER(6);
Пол CHARACTER(7);
Прозвище CHARACTER(20);
CHECK (Умер IN {'Истина', 'Ложь'});
CHECK (Пол IN {'Мужской', 'Женский'}).

После выполнения действий, предписанных первой группой шагов, структурный компонент реляционной схемы БД получит свое первое воплощение в виде отношений, сформированных для каждого множества сущностей, множества связей и многозначной характеристики, а также, возможно, отношения для представления множественного наследования. Этот набор отношений назовем предварительным, поскольку он хоть и полон, но, почти наверняка, избыточен. Исключение этой избыточности – основная задача последней, третьей группы шагов.

Задачей же второй группы шагов методики трансформации «ERM-модель – реляционная модель» является отражение средствами реляционной модели ограничений целостности ПрО, представленных в ERM-схеме. Если концепции структур данных реляционной модели одинаково представлены в различных СУБД, то средства определения ограничений целостности в них могут быть весьма разнообразными. Для конкретизации нашего рассмотрения имеет смысл воспользоваться стандартом языка SQL, поддерживаемым большинством коммерческих реляционных СУБД.

В командах определения схемы отношения SQL (*CREATE TABLE*, *ALTER TABLE*) предусмотрены следующие конструкции для задания ограничений целостности:

- тип данных атрибута;
- описатели атрибута *NULL* и *NOT NULL*, позволяющие определить соответственно, может или нет атрибут иметь неопределенное значение в кортежах отношения;
- описатель атрибута или группы атрибутов *PRIMARY KEY*, определяющий первичный ключ отношения;
- описатель атрибута или группы атрибутов *UNIQUE*, определяющий возможный ключ отношения;
- конструкция *FOREIGN KEY*, определяющая внешний ключ отношения;
- конструкция *CHECK*, определяющая условие на значения атрибута (-ов), которому должны удовлетворять все кортежи отношения.

Наряду с этими декларативными средствами, которые следует применять в первую очередь, в SQL предусмотрена возможность определять проверку ограничений целостности и выполнение связанных с этим действий более сложного характера с помощью процедурного кода, автоматическое выполнение которого осуществляется механизмом триггеров обновления.

Теперь рассмотрим, как декларативными средствами реляционной модели можно выразить ограничения целостности, построенные в ходе выполнения шагов 5-9 методики ERM-моделирования.

Ограничения целостности на области значений отображений-характеристик (шаг 6). В реляционной модели их иногда называют ограничениями целостности на значения атрибутов. К ним относятся:

- обязательное указание типа данных для каждого атрибута отношения (с максимальной длиной – для строк и разрядностью – для чисел);
- конструкция *CHECK*.

Определенные на шаге 5 методики ERM-моделирования ограничения целостности представляются в реляционной модели следующими конструкциями:

ФИО CHARACTER(50);
Умер CHARACTER(6);
Пол CHARACTER(7);
Прозвище CHARACTER(20);
CHECK (Умер IN {‘Истина’, ’Ложь’});
CHECK (Пол IN {‘Мужской’, ’Женский’}).

Методика трансформации «ERM-модель – реляционная модель» (шаг 7)

Ограничения целостности на отображения-характеристики и обратные им отображения. В реляционной модели их иногда называют ограничениями уникальности и определенности.

Если отображение-характеристика – полное функциональное (МинКЧ = 1, МаксКЧ = 1), соответствующий ей атрибут принадлежит отношению класса и снабжается описателем *NOT NULL* (в случае частичного функционального отображения (МинКЧ = 0, МаксКЧ = 1) – описателем *NULL*).

Если отображение-характеристика многозначно (МаксКЧ ≠ 1), соответствующий ей атрибут расположен в специальном дополнительном отношении. Оба атрибута этого отношения всегда объявляются *NOT NULL*.

Если отображение, обратное однозначному отображению-характеристике, функционально, соответствующий атрибут отношения класса объявляется возможным ключом (*UNIQUE*).

Если отображение, обратное многозначному отображению-характеристике, функционально, первичным ключом (*PRIMARY KEY*) дополнительного отношения объявляется атрибут со значениями характеристики, в противном случае – оба атрибута этого отношения.

Ограничения целостности на отображения-характеристики и обратные им отображения (шаг 7). В реляционной модели их иногда называют ограничениями уникальности и определенности. Они связаны, прежде всего, с определением первичных и возможных ключей отношений. Также в эту группу попадают ограничения целостности на обязательность определенных значений атрибутов.

Если отображение-характеристика – полное функциональное (МинКЧ = 1, МаксКЧ = 1), соответствующий ей атрибут принадлежит отношению класса и снабжается описателем *NOT NULL* (в случае частичного функционального отображения (МинКЧ = 0, МаксКЧ = 1) – описателем *NULL*).

Если отображение-характеристика многозначно (МаксКЧ ≠ 1), соответствующий ей атрибут расположен в специальном дополнительном отношении. Оба атрибута этого отношения всегда объявляются *NOT NULL*. Факт полного (МинКЧ = 1) определения многозначного отображения-характеристики выразить декларативными средствами реляционной модели невозможно.

Если отображение, обратное однозначному отображению-характеристике, функционально, соответствующий атрибут отношения класса объявляется возможным ключом (*UNIQUE*). Возможным ключом отношения может являться не один атрибут, а неизбыточная группа естественных атрибутов класса. В этом случае отображение из декартова произведения областей значений этих атрибутов функционально, а любые отображения из декартовых произведений меньшей степени – не функциональны.

Если отображение, обратное многозначному отображению-характеристике, функционально, первичным ключом (*PRIMARY KEY*) дополнительного отношения объявляется атрибут со значениями характеристики, в противном случае – оба атрибута этого отношения.

Для данной группы ограничений целостности нашего примера имеем.

Отношение *Человек*:
Человек_ID *NOT NULL*,
ФИО *NOT NULL*,
Умер *NOT NULL*,
Пол *NOT NULL*,
PRIMARY KEY (Человек_ID).

Отношение *Мужчина*:
Мужчина_ID *NOT NULL*,
PRIMARY KEY (Мужчина_ID).

Отношение *Женщина*:

Женщина_ID NOT NULL,
PRIMARY KEY (Женщина_ID).
Отношение *Отец*:
Отец_ID NOT NULL,
PRIMARY KEY (Отец_ID).
Отношение *Мать*:
Мать_ID NOT NULL,
PRIMARY KEY (Мать_ID).
Отношение *Прозвище человека*:
Человек_ID NOT NULL,
Прозвище NOT NULL,
PRIMARY KEY (Человек_ID, Прозвище).

Методика трансформации «ERM-модель – реляционная модель» (шаг 8)

Ограничения целостности на отображения между множествами сущностей. Эти ограничения целостности воплощаются в реляционной модели в виде так называемых ограничений ссылочной целостности, связанных с определением внешних ключей отношений, образованных из множеств связей.

Каждое определение внешнего ключа неявно подразумевает функциональность отображения между отношением, в котором он определен, и отношением, первичный или возможный ключ которого задает область допустимых значений внешнего ключа. Если это отображение к тому же полностью определено, внешний ключ снабжается описателем *NOT NULL*.

Если отображение, определяемое ролью (группой ролей), функционально, внешний ключ (группа внешних ключей), соответствующий этой роли (группе ролей), является возможным ключом (*UNIQUE*) отношения, которое задает экстенсионал этого отображения.

Ограничения целостности на отображения между множествами сущностей (шаг 8). Эти ограничения целостности воплощаются в реляционной модели в виде так называемых ограничений ссылочной целостности, связанных с определением внешних ключей отношений, образованных из множеств связей. Порождение атрибутов – внешних ключей в отношениях – задача структурного этапа проектирования реляционной схемы (шаг 2 настоящей методики).

Каждое определение внешнего ключа неявно подразумевает функциональность отображения между отношением, в котором он определен, и отношением, первичный или возможный ключ которого задает область допустимых значений внешнего ключа. Если это отображение к тому же полностью определено, внешний ключ снабжается описателем *NOT NULL*.

Если отображение, определяемое ролью (группой ролей), функционально, внешний ключ (группа внешних ключей), соответствующий этой роли (группе ролей), является возможным ключом (*UNIQUE*) отношения, которое задает экстенсионал этого отображения.

Отношения, представляющие в реляционной модели множества связей, будут объектом дальнейшего обсуждения. Там же мы продолжим разговор об ограничениях целостности, свойственных этим отношениям.

Пока же для нашего примера имеем следующие ограничения целостности этой группы.

Отношение *Родитель-Ребенок*:
Родитель_Человек_ID NOT NULL,
Ребенок_Человек_ID NOT NULL,
UNIQUE (Родитель_Человек_ID, Ребенок_Человек_ID),
FOREIGN KEY (Родитель_Человек_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Ребенок_Человек_ID) REFERENCES Человек (Человек_ID).

Отношение *Брак*:
Мужчина_ID NOT NULL,
Женщина_ID NOT NULL,
UNIQUE (Мужчина_ID),
UNIQUE (Женщина_ID),
FOREIGN KEY (Мужчина_ID) REFERENCES Мужчина (Мужчина_ID),
FOREIGN KEY (Женщина_ID) REFERENCES Женщина (Женщина_ID).

Отношение *Отец-Ребенок*:
Отец_ID NOT NULL,

Человек_ID NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Отец_ID) REFERENCES Отец (Отец_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Отношение *Мать-Ребенок*:

Мать_ID NOT NULL,
Человек_ID NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Мать_ID) REFERENCES Мать (Мать_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Отношение *Отец-Мать-Ребенок*:

Отец_ID NOT NULL,
Мать_ID NOT NULL,
Человек_ID NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Отец_ID) REFERENCES Отец (Отец_ID),
FOREIGN KEY (Мать_ID) REFERENCES Мать (Мать_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Методика трансформации «ERM-модель – реляционная модель» (шаг 9)

Ограничения целостности на специализации. Возможности представления ограничений целостности этого типа в реляционной модели весьма ограничены.

В случае независимых иерархий специализаций полная функциональность отображения «подкласс-суперкласс» представляется ограничением ссылочной целостности и описателем *NOT NULL* для внешнего ключа отношения, представляющего подкласс.

Функциональность обратного отображения гарантируется объявлением одновременно этого внешнего ключа первичным ключом отношения для подкласса.

В случае же реализации множественных наследований одним универсальным дополнительным отношением невозможно определить вообще никаких ограничений целостности, даже ссылочных.

Возможности переноса в реляционную модель ограничений целостности на специализации и категоризации существенным образом зависят от метода, выбранного для их представления. Мы подробно разберем ситуацию для наиболее часто применяемого для специализаций 2-го метода. Причем связи между кортежами суперкласса и кортежами подклассов определяются внешними ключами в подклассах, ссылающимися на первичный ключ суперкласса. Для других методов представления специализаций и категоризаций читателю предлагается провести самостоятельный анализ возможностей переноса в реляционную модель ограничений целостности на эти конструкции.

Ограничения целостности на специализации (шаг 9). Возможности представления ограничений целостности этого типа в реляционной модели весьма ограничены.

Прежде чем их рассмотреть напомним, как они превращаются в ограничения целостности на отображения. В качестве одного из множеств, участвующих в этих отображениях, естественно взять суперкласс. Тогда вторым множеством будет объединение множеств подклассов специализации. Причем элементами всех упомянутых классов являются не сами объекты, а их абстракции в контексте соответствующих понятий. В таком случае один и тот же объект как элемент суперкласса может быть представлен в объединении подклассов несколькими элементами разных понятий.

Тогда рассмотренные ранее ограничения целостности на специализации задаются следующими ограничениями на отображения между суперклассом и объединением подклассов:

- полная, пересекающаяся специализация –
  $\cup$ПК -> СК (1, 1), СК -> $\cup$ПК (1, ∞),
  где $\cup$ПК – объединение подклассов, а СК - суперкласс;
- полная, непересекающаяся специализация –
  $\cup$ПК -> СК (1, 1), СК -> $\cup$ПК (1, 1);
- частичная, пересекающаяся специализация –
  $\cup$ПК -> СК (1, 1), СК -> $\cup$ПК (0, ∞);
- частичная, непересекающаяся специализация –
  $\cup$ПК -> СК (1, 1), СК -> $\cup$ПК (0, 1).

Чаще всего каждый класс представлен самостоятельным отношением, причем внешние ключи подклассов ссылаются на первичный ключ суперкласса. Это вынуждает переходить от рассмотрения вышеупомянутых отображений к независимым отображениям ПК -> СК и СК -> ПК.

В случае независимых иерархий специализаций полная функциональность отображения «подкласс-суперкласс» представляется ограничением ссылочной целостности и описателем *NOT NULL* для внешнего ключа отношения, представляющего подкласс.

Функциональность обратного отображения гарантируется объявлением одновременно этого внешнего ключа первичным ключом отношения для подкласса. Тот факт, что это отображение полностью или не полностью определено, выразить декларативными средствами реляционной модели не удается.

Что касается отображения ПК -> СК, то из полной функциональности всех таких отображений одной специализации следует полная функциональность отображения ∪ПК -> СК. Чего, к сожалению, нельзя сказать об обратных им отображениях. Это означает, что функциональность отображения СК -> ∪ПК не может быть выражена средствами реляционной модели. То же касается и его полной определенности.

В случае же реализации множественных наследований одним универсальным дополнительным отношением невозможно определить вообще никаких ограничений целостности, даже ссылочных.

В нашем примере с родственными связями отношениям, определенным для множеств сущностей, добавятся следующие ограничения ссылочной целостности.

Отношение *Мужчина*:
FOREIGN KEY (Мужчина_ID) REFERENCES Человек (Человек_ID).
Отношение *Женщина*:
FOREIGN KEY (Женщина_ID) REFERENCES Человек (Человек_ID).
Отношение *Отец*:
FOREIGN KEY (Отец_ID) REFERENCES Мужчина (Мужчина_ID).
Отношение *Мать*:
FOREIGN KEY (Мать_ID) REFERENCES Женщина (Женщина_ID).

К великому сожалению такой важный класс ERM-ограничений как **ограничения целостности на взаимоотношения отображений типа следствие и эквивалентность** вообще не удается явно выразить декларативными средствами реляционной модели. В силу этого они на практике часто вообще не анализируются, в лучшем же случае некоторые из них находят свое воплощение в виде триггеров, то есть реализуются процедурно.

Что же из ERM-ограничений удается реализовать декларативными средствами определения ограничений реляционной модели? Итоги проведенного выше анализа приведены в таблицах.

Класс ERM-ограничений	ERM-ограничение	Ограничение реляционной модели
Ограничения целостности на области значений отображений-характеристик	**Синтаксис значений**	**Тип данных атрибута**
	Логическое выражение	**Конструкция CHECK**
Ограничения целостности на отображения-характеристики и обратные им отображения	**Отображение-характеристика:** **МаксКЧ = 1** **МинКЧ = 0** **МинКЧ = 1** **МаксКЧ = ∞** **МинКЧ = 1**	 **Атрибут в отношении-классе** **NULL** **NOT NULL** **Атрибут в доп. отношении – NOT NULL** ---
	Обратное отображение с простой или сложной ООО: **при однозначной характеристике** **МаксКЧ = 1** **МинКЧ = 0** **МинКЧ = 1** **МаксКЧ = ∞** **МинКЧ = 1** **при многозначной характеристике** **МаксКЧ = 1** **МинКЧ = 0** **МинКЧ = 1** **МаксКЧ = ∞** **МинКЧ = 1**	 **Атрибут отношения класса UNIQUE** --- --- **Атрибут отношения класса не UNIQUE** --- **Атрибут доп. отношения PRIMARY KEY** --- --- **Оба атрибута доп. отношения составляют PRIMARY KEY** ---

Ограничения целостности на отображения между множествами сущностей	**Отображение, определяемое ролью или группой ролей:** **МаксКЧ = 1** **МинКЧ = 0** **МинКЧ = 1** **МаксКЧ = ∞** **МинКЧ = 1**	 **Соответствующий внешний ключ или группа внешних ключей объявляется UNIQUE** --- --- **Соответствующий внешний ключ не объявляется UNIQUE** ---
Ограничения целостности на специализации	**Специализация полная, пересекающаяся ∪ПК -> СК (1, 1), СК -> ∪ПК (1, ∞)**	**Декларативными средствами реляционной модели удается представить лишь полную функциональную определенность отображений ПК -> СК, а вместе с ними – и отображения ∪ПК -> СК.**
	Специализация полная, непересекающаяся ∪ПК -> СК (1, 1), СК -> ∪ПК (1, 1)	
	Специализация частичная, пересекающаяся ∪ПК -> СК (1, 1), СК -> ∪ПК (0, ∞)	
	Специализация частичная, непересекающаяся ∪ПК -> СК (1, 1), СК -> ∪ПК (0, 1)	
Ограничения целостности на взаимоотношения отображений	**Следствие** **Эквивалентность**	--- ---

В результате выполнения указанных действий для нашего примера имеем следующую предварительную реляционную схему, в которой максимально полно представлены декларативные ограничения целостности.

Пример

Человек (Человек_ID, ФИО, Умер, Пол):
Человек_ID NUMBER(6,0) NOT NULL,
ФИО CHARACTER(50) NOT NULL,
Умер CHARACTER(6) NOT NULL,
Пол CHARACTER(7) NOT NULL,
PRIMARY KEY (Человек_ID),
CHECK (Умер IN {'Истина', 'Ложь'}),
CHECK (Пол IN {'Мужской', 'Женский'}).
Прозвище человека (Человек_ID, Прозвище):
Человек_ID NUMBER(6,0) NOT NULL,
Прозвище CHARACTER(20) NOT NULL,
PRIMARY KEY (Человек_ID, Прозвище).
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).
Мужчина (Мужчина_ID):
Мужчина_ID NUMBER(6,0) NOT NULL,
PRIMARY KEY (Мужчина_ID),
FOREIGN KEY (Мужчина_ID) REFERENCES Человек (Человек_ID).
Женщина (Женщина_ID):
Женщина_ID NUMBER(6,0) NOT NULL,
PRIMARY KEY (Женщина_ID),
FOREIGN KEY (Женщина_ID) REFERENCES Человек (Человек_ID).

Пример (продолжение)

Отец (Отец_ID):
Отец_ID NUMBER(6,0) NOT NULL,
PRIMARY KEY (Отец_ID),
FOREIGN KEY (Отец_ID) REFERENCES Мужчина (Мужчина_ID).
Мать (Мать_ID):
Мать_ID NUMBER(6,0) NOT NULL,
PRIMARY KEY (Мать_ID),
FOREIGN KEY (Мать_ID) REFERENCES Женщина (Женщина_ID).
Родитель-Ребенок (Родитель_Человек_ID, Ребенок_Человек_ID):
Родитель_Человек_ID NUMBER(6,0) NOT NULL,
Ребенок_Человек_ID NUMBER(6,0) NOT NULL,
UNIQUE (Родитель_Человек_ID, Ребенок_Человек_ID),
FOREIGN KEY (Родитель_Человек_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Ребенок_Человек_ID) REFERENCES Человек (Человек_ID).

Пример (продолжение)

Брак (Мужчина_ID, Женщина_ID):
Мужчина_ID NUMBER(6,0) NOT NULL,
Женщина_ID NUMBER(6,0) NOT NULL,
UNIQUE (Мужчина_ID),
UNIQUE (Женщина_ID),
FOREIGN KEY (Мужчина_ID) REFERENCES Мужчина (Мужчина_ID),
FOREIGN KEY (Женщина_ID) REFERENCES Женщина (Женщина_ID).

Отец-Ребенок (Отец_ID, Человек_ID):
Отец_ID NUMBER(6,0) NOT NULL,
Человек_ID NUMBER(6,0) NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Отец_ID) REFERENCES Отец (Отец_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Мать-Ребенок (Мать_ID, Человек_ID):
Мать_ID NUMBER(6,0) NOT NULL,
Человек_ID NUMBER(6,0) NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Мать_ID) REFERENCES Мать (Мать_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Пример (продолжение)

Отец-Мать-Ребенок (Отец_ID, Мать_ID, Человек_ID):
Отец_ID NUMBER(6,0) NOT NULL,
Мать_ID NUMBER(6,0) NOT NULL,
Человек_ID NUMBER(6,0) NOT NULL,
UNIQUE (Человек_ID),
FOREIGN KEY (Отец_ID) REFERENCES Отец (Отец_ID),
FOREIGN KEY (Мать_ID) REFERENCES Мать (Мать_ID),
FOREIGN KEY (Человек_ID) REFERENCES Человек (Человек_ID).

Предварительная реляционная схема БД (структурный компонент)

Человек (Человек_ID, ФИО, Умер, Пол);
Прозвище человека (Человек_ID, Прозвище);
Мужчина (Мужчина_ID);
Женщина (Женщина_ID);
Отец (Отец_ID);
Мать (Мать_ID);
Родитель-Ребенок (Родитель_Человек_ID, Ребенок_Человек_ID);
Брак (Мужчина_ID, Женщина_ID);
Отец-Ребенок (Отец_ID, Человек_ID);
Мать-Ребенок (Мать_ID, Человек_ID);
Отец-Мать-Ребенок (Отец_ID, Мать_ID, Человек_ID).

Представляет интерес вопрос качества полученных отношений с точки зрения теории проектирования реляционных БД.

Отношения, образованные из множеств сущностей, находятся в 5НФ (пятая нормальная форма – наивысшая форма нормальности отношений). Они включают суррогатный атрибут и однозначные характеристики множеств сущностей. Первый из них либо введен в этом отношении, либо унаследован через механизм ссылочной целостности от родового класса. В любом случае он является первичным ключом и функционально определяет все остальные атрибуты (многозначные характеристики множеств сущностей в таких отношениях отсутствуют). Отдельный характеристический атрибут (или группа атрибутов) может являться возможным ключом. В таком случае он также функционально определяет все остальные атрибуты. Других зависимостей между атрибутами этих отношений не существует.

Дополнительные отношения для многозначных характеристик находятся в 5НФ. Первичный ключ этих отношений образуется либо из обоих атрибутов, если отображение, обратное многозначному отображению-характеристике, нефункционально, либо из одного атрибута со значениями характеристики в противном случае. Других зависимостей между атрибутами этих отношений не существует.

Дополнительное отношение для представления множественного наследования находится в 5НФ. Первичный ключ этого отношения представляет собой группу из всех четырех атрибутов отношения. Других зависимостей между атрибутами этих отношений не существует.

Единственными «проблемными» отношениями являются отношения, образованные из множеств связей. Об улучшении качества этих отношений будет идти речь в дальнейшем, поскольку прежде чем совершенствовать отношение, следует посмотреть, а нужно ли оно вообще.

Методика трансформации «ERM-модель – реляционная модель» (шаг 10)

Полностью избыточные отношения для множеств связей, удаление которых не приводит к изменению остальных отношений. Отношение, построенное для множеств связей степени большей двух, избыточно, если для него и соответствующих бинарных отношений выполняются условия эквивалентности схем. Эти условия обеспечивают теоремы ТСЗО .

В частности, если в отношении есть функциональное отображение, определяемое ролью, и его проекции эквивалентны соответствующим родственным отображениям отношений, построенных для бинарных множеств связей, можно смело избавляться от первого отношения.

В нашем примере таковым является отношение *Отец-Мать-Ребенок*.

Отображение *Родители*, определяемое ролью *Ребенок*, – функционально. Для его проекций справедливо *Родители [Отец]* <-> *Отец ребенка* и *Родители [Мать]* <-> *Мать ребенка*.

Исключение этого отношения из схемы не приведет к потере функциональной зависимости *Ребенок -> Отец, Мать* поскольку она является следствием функциональных зависимостей *Ребенок -> Отец* и *Ребенок -> Мать*.

К этому моменту создана вполне работоспособная реляционная схема, удовлетворяющая первым двум критериям полноты и корректности. Ее главный недостаток – излишнее дублирование данных и ограничений целостности.

К сожалению, обилие факторов (в том числе и особенности физической модели СУБД), влияющих на эффективность реляционной схемы, не позволяет пока сформулировать методологию ее оптимизации, как набор четких однозначных правил, гарантировано приводящих к получению самой оптимальной из всех возможных схем.

В дальнейшем будет представлен набор правил и рекомендаций, творческое применение которых поможет достичь желаемого результата. Все-таки тезис «проектирование БД – это скорее искусство, чем наука», по-видимому, будет актуален всегда. Данная книга позволяет лишь сместить границу между искусством и наукой в сторону искусства, расширяя тем самым сферу науки.

Как отмечалось ранее к этому моменту методики единственными «проблемными» отношениями остались отношения, построенные для множеств связей, представляющих отображения между множествами сущностей. Отношения, образованные из множеств сущностей, хоть и находятся в 5НФ, тоже могут оказаться избыточными. Именно о таких отношениях в основном и пойдет речь в дальнейшем. Правда это не означает, что изменениям не будут подвергаться другие отношения схемы. Но эти изменения не будут нарушать их пятой нормальной формы.

Полностью избыточные отношения для множеств связей, удаление которых не приводит к изменению остальных отношений (шаг 10). Отношение, построенное для множеств связей степени большей двух, избыточно, если для него и соответствующих бинарных отношений выполняются условия эквивалентности схем. Эти условия обеспечивают теоремы ТСЗО (теории семантически значимых отображений).

К великому сожалению в настоящий момент только закладываются основы ТСЗО, и большая часть теорем пока существует в виде недоказанных гипотез. А многие закономерности теории еще вообще не выявлены. После должного развития ТСЗО этот шаг методики будет обеспечивать самый существенный вклад в получение действительно эффективных реляционных схем. Хотя некоторые результаты в этом направлении уже достигнуты.

В частности, если в отношении есть функциональное отображение, определяемое ролью, и его проекции эквивалентны соответствующим родственным отображениям отношений, построенных для бинарных множеств связей, можно смело избавляться от первого отношения.

В нашем примере таковым является отношение *Отец-Мать-Ребенок*. Отображение *Родители*, определяемое ролью *Ребенок*, – функционально. Для его проекций справедливо *Родители [Отец] <-> Отец ребенка* и *Родители [Мать] <-> Мать ребенка*. Исключение этого отношения из схемы не приведет к потере функциональной зависимости *Ребенок -> Отец, Мать* поскольку она является следствием функциональных зависимостей *Ребенок -> Отец* и *Ребенок -> Мать*.

Методика трансформации «ERM-модель – реляционная модель» (шаг 10) (продолжение)

Полностью избыточными являются также отношения, построенные для множеств связей, все отображения которых эквивалентны отображениям (возможно полученным с помощью операций над отображениями), определяемым другими отношениями проекта.

Подобная ситуация наблюдается с отношением *Родитель-Ребенок*. Оба определяемые им отображения *Родитель* и *Ребенок* таковы, что

Отец ребенка ∪ *Мать ребенка* <-> *Родитель* и

Ребенок отца ∪ *Ребенок матери* <-> *Ребенок*.

В переводе на «реляционный» язык это означает, что отношение *Родитель-Ребенок* может быть получено объединением отношений *Отец-Ребенок* и *Мать-Ребенок*. В таком случае отношение *Родитель-Ребенок* также избыточно.

Полностью избыточными являются также отношения, построенные для множеств связей, все отображения которых эквивалентны отображениям (возможно полученным с помощью операций над отображениями), определяемым другими отношениями проекта.

Подобная ситуация наблюдается с отношением *Родитель-Ребенок*. Оба определяемые им отображения *Родитель* и *Ребенок* таковы, что

Отец ребенка ∪ *Мать ребенка* <-> *Родитель* и

Ребенок отца ∪ *Ребенок матери* <-> *Ребенок*.

В переводе на «реляционный» язык это означает, что отношение *Родитель-Ребенок* может быть получено объединением отношений *Отец-Ребенок* и *Мать-Ребенок*. В таком случае отношение *Родитель-Ребенок* также избыточно.

Методика трансформации «ERM-модель – реляционная модель» (шаг 11)

Избыточные отношения для множеств связей, удаление которых приводит к изменению других отношений. Если хотя бы одно из определяемых бинарным отношением отображений функционально, это отношение может быть исключено с одновременным добавлением дополнительного внешнего ключа в отношение для множества сущностей, являющегося множеством прообразов функционального отображения. Областью значений этого внешнего ключа является первичный ключ отношения для другого множества сущностей.

Это правило следует безусловно применять в случае полной функциональности отображения. При этом добавленный внешний ключ имеет описатель *NOT NULL*.

В случае частичной функциональности применение этого правила оправдано, если справедливо хотя бы одно из следующих условий:

- СУБД эффективно хранит пропущенные значения на диске;
- вероятность наличия у объектов-прообразов соответствующих образов близка к единице;
- операция соединения с участием рассматриваемого отношения часто выполняется и требует немалых затрат памяти и времени.

Если все-таки принимается решение об исключении отношения, добавленный внешний ключ в таком случае имеет описатель *NULL*.

Избыточные отношения для множеств связей, удаление которых приводит к изменению других отношений (шаг 11). Если хотя бы одно из определяемых бинарным отношением отображений функционально, это отношение может быть исключено с одновременным добавлением дополнительного внешнего ключа в отношение для множества сущностей, являющегося множеством прообразов функционального отображения. Областью значений этого внешнего ключа является первичный ключ отношения для другого множества сущностей.

Это правило следует безусловно применять в случае полной функциональности отображения. При этом добавленный внешний ключ имеет описатель *NOT NULL*.

В случае частичной функциональности применение этого правила оправдано, если справедливо хотя бы одно из следующих условий:

- СУБД эффективно хранит пропущенные значения на диске;
- вероятность наличия у объектов-прообразов соответствующих образов близка к единице;
- операция соединения с участием рассматриваемого отношения часто выполняется и требует немалых затрат памяти и времени.

Если все-таки принимается решение об исключении отношения, добавленный внешний ключ в таком случае имеет описатель *NULL*.

Следует отметить, что добавление дополнительного атрибута в отношение для множества сущностей не нарушает его пятой нормальной формы, поскольку этот атрибут функционально зависит от первичного ключа этого отношения.

В том случае если исключенное отношение являлось отношением типа «один-к-одному», т.е. определяло пару функциональных отображений, дополнительный атрибут является возможным ключом отношения и снабжается описателем *UNIQUE*. При этом, если одно отображение полно функционально, а второе – частично, дублирование атрибута следует производить в соответствии с полным функциональным отображением.

Пример

В нашем примере применение этого правила к отношениям *Отец-Ребенок* и *Мать-Ребенок* приведет к тому, что эти отношения будут исключены из схемы, а в отношение *Человек* будут добавлены два дополнительных атрибута и ограничения внешних ключей:

```
Человек (Человек_ID, Умер, Пол, Отец_ID, Мать_ID):
Человек_ID      NUMBER(6,0)       NOT NULL,
ФИО             CHARACTER(50)     NOT NULL,
Умер            CHARACTER(6)      NOT NULL,
Пол             CHARACTER(7)      NOT NULL,
Отец_ID         NUMBER(6,0)       NOT NULL,
Мать_ID         NUMBER(6,0)       NOT NULL,
PRIMARY KEY (Человек_ID),
CHECK (Умер IN {'Истина', 'Ложь'}),
CHECK (Пол IN {'Мужской', 'Женский'}),
FOREIGN KEY (Отец_ID) REFERENCES Отец (Отец_ID),
FOREIGN KEY (Мать_ID) REFERENCES Мать (Мать_ID).
```

Описатель *NOT NULL* у обоих дополнительных атрибутов отражает полную функциональность отображений *Отец ребенка* и *Мать ребенка*.

Пример

Как правило, в случае отношения типа 1:1 осуществляется дублирование только одного атрибута. В нашем случае при исключении отношения *Брак* мы осуществим перекрестное дублирование: *Мужчина_ID* в отношение *Женщина* и *Женщина_ID* в отношение *Мужчина*.

```
Мужчина (Мужчина_ID, Женщина_ID);
Мужчина_ID      NUMBER(6,0)       NOT NULL,
Женщина_ID      NUMBER(6,0)       NULL,
PRIMARY KEY (Мужчина_ID),
UNIQUE (Женщина_ID),
FOREIGN KEY (Мужчина_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Женщина_ID) REFERENCES Женщина (Женщина_ID).

Женщина (Женщина_ID, Мужчина_ID);
Женщина_ID      NUMBER(6,0)       NOT NULL,
Мужчина_ID      NUMBER(6,0)       NULL,
PRIMARY KEY (Женщина_ID),
UNIQUE (Мужчина_ID),
FOREIGN KEY (Женщина_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Мужчина_ID) REFERENCES Мужчина (Мужчина_ID).
```

Как правило, в случае отношения типа 1:1 осуществляется дублирование только одного атрибута. В нашем случае при исключении отношения *Брак* мы осуществим перекрестное дублирование: *Мужчина_ID* в отношение *Женщина* и *Женщина_ID* в отношение *Мужчина*.

Причина, по которой мы проделали эти, казалось бы, нелогичные действия, будет ясна из последующего изложения. Обратите внимание, что описатели *NOT NULL* атрибутов отношения *Брак* заменились на описатели *NULL* у внешних ключей. Таким образом, после преобразования отношений нашла отражение частичная функциональность отображений отношения *Брак*.

Как видим, иногда последующие шаги методики улучшают результат с точки зрения сформулированных ранее критериев (в данном случае – критерия корректности). Это не может не радовать. Хуже, когда происходит наоборот.

Полностью избыточные отношения для множеств сущностей, удаление которых не приводит к изменению остальных отношений (шаг 12). Если отношение, построенное для множества сущностей, можно получить в результате выполнения операций над другими отношениями, и это множество сущностей не участвует ни в каких отображениях, его можно удалить из схемы. Другими словами, в таком отношении нет атрибутов кроме первичного суррогатного ключа, и последний не участвует в определениях внешних ключей. К тому же, текущее множество значений этого ключа может быть автоматически получено по запросу с участием других отношений.

Логической основой таких запросов являются формальные определения понятий ПрО, полученные на шаге 1 методики ERM-моделирования.

В схеме нашего примера подобных отношений нет.

Методика трансформации «ERM-модель – реляционная модель» (шаг 13)

Избыточные отношения для множеств сущностей, удаление которых приводит к изменению других отношений. Если отношение, построенное для множества сущностей, можно получить в результате выполнения операций над другими отношениями, и это множество сущностей участвует в отображениях между множествами сущностей, его можно удалить из схемы, переопределив внешние ключи, ссылающиеся на первичный ключ этого отношения. Вместо него необходимо указать первичный ключ отношения, построенного для множества сущностей, которое является суперклассом по отношению к исходному.

Если кандидатом на исключение является отношение, множество сущностей которого имеет однозначные характеристики, решение во многом зависит от эффективности представления в СУБД пропущенных значений. Если вас устраивает, что в кортежах отношения суперкласса будут пропущенные значения в этих характеристиках, перенесите их в отношение суперкласса. С многозначными характеристиками проблем не будет. Надо лишь поменять ссылки на первичные ключи во внешних ключах соответствующих отношений для многозначных характеристик.

Избыточные отношения для множеств сущностей, удаление которых приводит к изменению других отношений (шаг 13). Если отношение, построенное для множества сущностей, можно получить в результате выполнения операций над другими отношениями, и это множество сущностей участвует в отображениях между множествами сущностей, его можно удалить из схемы, переопределив внешние ключи, ссылающиеся на первичный ключ этого отношения. Вместо него необходимо указать первичный ключ отношения, построенного для множества сущностей, которое является суперклассом по отношению к исходному.

Если кандидатом на исключение является отношение, множество сущностей которого имеет однозначные характеристики, решение во многом зависит от эффективности представления в СУБД пропущенных значений. Если вас устраивает, что в кортежах отношения суперкласса будут пропущенные значения в этих характеристиках, перенесите их в отношение суперкласса. С многозначными характеристиками проблем не будет. Надо лишь поменять ссылки на первичные ключи во внешних ключах соответствующих отношений для многозначных характеристик.

По-прежнему, логической основой операций восстановления удаленных избыточных отношений из оставшихся отношений являются формальные определения понятий ПрО, полученные на шаге 1 методики ERM-моделирования. Рассматривать отношения для множеств сущностей на предмет избыточности надо, начиная с нижних уровней иерархии специализаций.

Пример

В нашем случае с родственными связями на нижнем уровне иерархии специализаций имеются отношения для множеств сущностей *Отец* и *Мать*. Приведем соответствующие определения понятий:

x Отец(*x*) $=_{Df}$ *x* (*x есть* а *Мужчина*(а) & ∃*y Родитель-Ребенок* (*x*, *y*))
x Мать(*x*) $=_{Df}$ *x* (*x есть* а *Женщина*(а) & ∃*y Родитель-Ребенок* (*x*, *y*)).

Экстенсионалы этих отношений можно получить из других отношений схемы, выполнив следующие запросы SQL:

SELECT Мужчина.Мужчина_ID FROM Мужчина, Человек
WHERE Мужчина.Мужчина_ID = Человек.Отец_ID и
SELECT Женщина.Женщина_ID FROM Женщина, Человек
WHERE Женщина.Женщина_ID = Человек.Мать_ID

Удаляем отношения для множеств сущностей *Отец* и *Мать*, заменив определения внешних ключей отношения *Человек* на следующие:

FOREIGN KEY (Отец_ID) REFERENCES Мужчина (Мужчина_ID),
FOREIGN KEY (Мать_ID) REFERENCES Женщина (Женщина_ID).

Пример (продолжение)

В более сложном случае отношение для множества сущностей может иметь дополнительные атрибуты – внешние ключи, представляющие функциональные отображения, в которых рассматриваемое множество сущностей играет роль множества прообразов. Такое отношение можно исключить из схемы, только в случае, когда эти внешние ключи удастся переместить в отношение-суперкласс, не нарушив критериев полноты и корректности. К тому же следует оценить такие изменения с точки зрения повышения эффективности с учетом уже упоминавшихся особенностей физической модели СУБД.

Пример (продолжение)

К настоящему моменту в нашем примере на нижнем уровне иерархии остались множества сущностей *Мужчина* и *Женщина*, соответствующие понятиям:
x Мужчина(*x*) $=_{Df}$ *x* (*x есть* а *Человек*(а) & *x = Человек пола* (*Мужской*)) и
x Женщина(*x*) $=_{Df}$ *x* (*x есть* а *Человек*(а) & *x = Человек пола* (*Женский*)).

Экстенсионалы их отношений можно получить из отношения *Человек*, выполнив следующие запросы SQL:
SELECT Человек.Человек_ID FROM Человек
WHERE Человек.Пол = 'Мужской' и
SELECT Человек.Человек_ID FROM Человек
WHERE Человек.Пол = 'Женский'

Обобщая понятия *Муж* и *Жена*, можно прийти к понятию *Супруг*, объем которого является объединением объемов исходных понятий. Поскольку специализация множества сущностей *Человек* на множества сущностей *Мужчина* и *Женщина* – полная и непересекающаяся, можно из внешних ключей отношений *Мужчина* и *Женщина* образовать один внешний ключ отношения *Человек – Супруг_ID*. Областью определения значений этого ключа является первичный ключ отношения *Человек – Человек_ID*.

Методика трансформации «ERM-модель – реляционная модель» (результат)

Человек (Человек_ID, Умер, Пол, Отец_ID, Мать_ID, Супруг_ID):

Человек_ID	*NUMBER(6,0)*	*NOT NULL,*
ФИО	*CHARACTER(50)*	*NOT NULL,*
Умер	*CHARACTER(6)*	*NOT NULL,*
Пол	*CHARACTER(7)*	*NOT NULL,*
Отец_ID	*NUMBER(6,0)*	*NOT NULL,*
Мать_ID	*NUMBER(6,0)*	*NOT NULL,*
Супруг_ID	*NUMBER(6,0)*	*NULL,*

PRIMARY KEY (Человек_ID),
UNIQUE (Супруг_ID),
CHECK (Умер IN {'Истина', 'Ложь'}),
CHECK (Пол IN {'Мужской', 'Женский'}),
FOREIGN KEY (Отец_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Мать_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Супруг_ID) REFERENCES Человек (Человек_ID).

Прозвище человека (Человек_ID, Прозвище):

Человек_ID	*NUMBER(6,0)*	*NOT NULL,*
Прозвище	*CHARACTER(20)*	*NOT NULL,*

PRIMARY KEY (Человек_ID, Прозвище).

В результате исключения отношений *Мужчина* и *Женщина* определение отношения *Человек* примет следующий вид:

Человек (Человек_ID, ФИО, Умер, Пол, Отец_ID, Мать_ID, Супруг_ID):

Человек_ID	*NUMBER(6,0)*	*NOT NULL,*
ФИО	*CHARACTER(50)*	*NOT NULL,*
Умер	*CHARACTER(6)*	*NOT NULL,*
Пол	*CHARACTER(7)*	*NOT NULL,*
Отец_ID	*NUMBER(6,0)*	*NOT NULL,*
Мать_ID	*NUMBER(6,0)*	*NOT NULL,*
Супруг_ID	*NUMBER(6,0)*	*NULL,*

PRIMARY KEY (Человек_ID),
UNIQUE (Супруг_ID),
CHECK (Умер IN {'Истина', 'Ложь'}),
CHECK (Пол IN {'Мужской', 'Женский'}),
FOREIGN KEY (Отец_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Мать_ID) REFERENCES Человек (Человек_ID),
FOREIGN KEY (Супруг_ID) REFERENCES Человек (Человек_ID).

Наряду с этим отношением в схеме нашего примера осталось только отношение *Прозвище человека*:

Прозвище человека (Человек_ID, Прозвище):

Человек_ID	*NUMBER(6,0)*	*NOT NULL,*
Прозвище	*CHARACTER(20)*	*NOT NULL,*

PRIMARY KEY (Человек_ID, Прозвище).

Методика трансформации «ERM-модель – реляционная модель» (заключительные проверки)

На завершающей фазе методики следует проверить оставшиеся отношения на удовлетворение критериев полноты и корректности – все ли формы высказываний и ограничения целостности поддерживаются схемой.

На завершающей фазе методики следует еще раз тщательно проверить оставшиеся отношения на удовлетворение критериев полноты и корректности – все ли формы высказываний и ограничения целостности по-прежнему поддерживаются схемой.

В нашем случае созданная реляционная схема наилучшим образом соответствует всем сформулированным критериям.

В заключение описания методики проектирования реляционных схем БД с использованием ERM-модели следует отметить, что детальность ее изложения и кажущаяся на первый взгляд избыточность некоторых действий, результаты которых исключаются на следующих этапах, не должны пугать и вводить в искушение отойти от предлагаемого подхода. Особенно на первых порах ее применения следует максимально придерживаться предлагаемого подхода. Позже, когда проектирование БД дойдет до автоматизма, и действия будут осуществляться, как правило, интуитивно, вы просто не будете замечать, что делаете что-то лишнее.

Логическое проектирование данных
Этап 3. Введение контролируемой избыточности данных

1. **Объединение таблиц со связями типа «один к одному» (1:1).**
2. **Дублирование неключевых атрибутов родительской таблицы в связях «один ко многим» (1:M).**
3. **Создание агрегирующих столбцов родительской таблицы в связях «один ко многим» (1:M).**
4. **Дублирование атрибутов внешних ключей в иерархии связей «один ко многим» (1:M).**
5. **Дублирование атрибутов в связях «многие ко многим» (M:N).**
6. **Введение повторяющихся групп полей.**
7. **Объединение справочных таблиц с базовыми таблицами.**
8. **Создание таблиц из данных, содержащихся в других таблицах.**

Последний, третий этап логического проектирования данных может сопутствовать любой методике проектирования реляционной схемы БД. Мы рассмотрим его в контексте именно семантической методики, поскольку считаем ее наиболее подходящей для практического использования, и поэтому хотим обеспечить ее полноту.

Целью этого этапа является определение необходимости ввода контролируемой избыточности за счет ослабления условий нормализации для повышения производительности системы.

Результатом нормализации является логическая схема базы данных – структурно цельная система с минимальной избыточностью. Но в некоторых случаях оказывается, что нормализованная схема не обеспечивает максимальной производительности при обработке данных. Следовательно, при некоторых обстоятельствах может оказаться необходимым ради повышения производительности пожертвовать частью той выгоды, которую обеспечивает схема с полной нормализацией.

К денормализации следует прибегать лишь тогда, когда нормализованная база не удовлетворяет требованиям, предъявляемым к производительности системы. Мы не призываем к полному отказу от нормализации в логической схеме базы данных: нормализация позволяет однозначно зафиксировать место каждого атрибута в реляционной базе, а это может оказаться решающим фактором при создании эффективно работающей системы. Кроме того, необходимо учесть и следующие особенности:

- денормализация может усложнить физическую реализацию системы;
- денормализация часто приводит к снижению гибкости;
- денормализация может ускорить чтение данных, но при этом замедлить обновление записей.

Формально **денормализацию** можно определить как модификацию реляционной схемы, при которой степень нормализации модифицированного отношения (отношений) становится ниже, чем степень нормализации, по меньшей мере, одного из исходных отношений. Термин «денормализация» будет также применяться в случае, когда два отношения объединяются в одно, и полученное отношение остается нормализованным, однако содержит больше пустых значений, чем исходные отношения. Некоторые авторы определяют денормализацию как модификацию реляционной схемы с учетом требований эксплуатации.

Если говорить конкретнее, на этом этапе рассматривается возможность дублирования отдельных атрибутов или объединения нескольких отношений в одну таблицу с целью сокращения числа запросов на соединение отношений.

При формулировке цели этого этапа логического проектирования мы упомянули одно важное слово – «контролируемая». Оно очень существенно при проведении денормализации реляционной схемы. Основной задачей процесса нормализации являлось исключение аномалий при работе с данными. В ходе этого процесса мы добиваемся идеальной в этом смысле схемы БД.

Так вот, чтобы не нарушить этого свойства схемы, процесс денормализации должен проводиться по определенным правилам. Во-первых, создаваемые в результате денормализации избыточные структуры (столбцы и таблицы) должны добавляться в схему без удаления уже созданных при нормализации структур. Во-вторых, клиентские приложения, осуществляющие изменения данных, должны воздействовать напрямую только на нормализованные структуры. В-третьих, данные избыточных структур должны модифицироваться исключительно специализированным программным кодом, реализованным либо в виде триггеров обновления нормализованных структур, либо в виде пакетных программ, периодически запускаемых в моменты низкой загрузки системы БД.

Такой подход идеально разрешает ситуацию. С одной стороны, он обеспечивает необходимую эффективность критических запросов, с другой стороны – не возвращает в схему аномалии изменения данных.

К сожалению, нельзя сформулировать общие правила определения того, когда действительно требуется денормализация отношений. Мы рассмотрим возможность применения денормализации в ситуациях, когда требуется ускорить выполнение часто повторяющихся или важных транзакций.

Объединение таблиц со связями типа «один к одному» (1:1)

Таблицы, между которыми установлены связи 1:1, иногда целесообразно объединить в одно отношение. Этот прием имеет смысл применять в том случае, если наиболее часто генерируются запросы, обращающиеся сразу к нескольким объединяемым таблицам, и редко – запросы к отдельным таблицам.

Дублирование неключевых атрибутов родительской таблицы в связях «один ко многим» (1:*M*)

На этапе денормализации основной целью является уменьшение количества соединений (или полное их устранение) в запросах, которые выполняются особенно часто или являются важными. Этому может способствовать повторение одного или нескольких неключевых атрибутов родительского отношения в дочернем отношении в связи 1:*M*.

Следует учитывать, что такие изменения могут не только принести выгоду, но и вызвать ряд проблем. Например, если продублированные данные изменятся в родительском отношении, необходимо будет обновить соответствующее поле в дочернем отношении. Кроме того, связь 1:*M* допускает многократное повторение каждого элемента данных в продублированном поле дочернего отношения, и тогда эти копии приходится постоянно синхронизировать. Отсюда вытекает проблема потери времени на синхронизацию данных в продублированном поле – синхронизация потребуется при каждой вставке, изменении или удалении строки.

Еще одна проблема – дополнительный расход памяти для хранения дублированных данных. Учитывая нынешнюю относительно невысокую стоимость хранения данных во внешней памяти, эта проблема не выглядит слишком серьезной.

Создание агрегирующих столбцов родительской таблицы в связях «один ко многим» (1:*M*)

Если в критических запросах часто возникает потребность в получении для родительского объекта агрегированного значения (суммы, среднего и т.д.) характеристик дочерних объектов, имеет смысл добавить соответствующий атрибут (или несколько атрибутов) в родительскую таблицу и позаботиться о поддержании его значения в актуальном состоянии. Такой подход позволит полностью избежать соединений этих таблиц в запросах.

Дублирование атрибутов внешних ключей в иерархии связей «один ко многим» (1:*M*)

Речь идет о ситуации, когда в схеме БД имеется иерархическая структура данных, образованная несколькими таблицами, которые имеют связи 1:*M* между собой. Корнем дерева является некоторая таблица. На ее первичный ключ ссылается внешний ключ другой таблицы. Первичный ключ второй таблицы в свою очередь является областью допустимых значений внешнего ключа третьей таблицы и т.д. Если в запросе требуется получить информацию из корневой и листовой таблиц, необходимо выполнить *n*-1 соединение, где *n* – высота дерева.

С помощью дублирования значений первичного ключа таблицы-корня в таблице-листе и образования таким образом дополнительного внешнего ключа можно свести реализацию упомянутого запроса к одному соединению. Вновь созданный внешний ключ позволяет явно представлять в БД так называемые транзитивные связи.

Дублирование атрибутов в связях «многие ко многим» (*M*:*N*)

Проектируя логическую схему базы данных, мы ввели промежуточное отношение, реализующее связь *M*:*N*. Теперь для доступа к данным по связи *M*:*N* нам придется соединить три таблицы: две исходные и одну новую, реализующую связь между двумя исходными. Но в некоторых случаях удается сократить количество таблиц, подлежащих соединению, продублировав атрибуты одной из исходных таблиц (или обеих) в промежуточной таблице-связи.

Введение повторяющихся групп полей

Мы убрали повторяющиеся группы полей из логической схемы данных, когда приводили объекты базы к первой нормальной форме. Затем мы выделили повторяющиеся группы в отдельную таблицу, сформировав при этом связь 1:*M* с исходным (родительским) отношением. Но иногда для повышения общей производительности системы имеет смысл снова ввести повторяющиеся группы за счет добавления соответствующих атрибутов в родительское отношение.

Например, в БД детского сада часто необходимо получать фамилии, имена и отчества обоих родителей ребенка. Наряду с выделением отдельного отношения *РОДИТЕЛЬ*, можно в отношение *РЕБЕНОК* добавить избыточные атрибуты *Фамилия отца*, *Имя отца*, *Отчество отца*, *Фамилия матери*, *Имя матери*, *Отчество матери*.

Такой тип денормализации данных следует применять лишь при следующих условиях:

- известно максимальное количество элементов в повторяющейся группе;
- число элементов в повторяющейся группе постоянно и долгое время не будет меняться;
- число элементов в повторяющейся группе не слишком велико, обычно не больше 10.

Объединение справочных таблиц с базовыми таблицами

Справочные таблицы (англ. lookup tables, reference tables), или списки для выбора (англ. pick lists), представляют собой особый случай связи 1:*M*. Как правило, справочная таблица включает поле с кодом и поле с описанием.

Использование справочных таблиц дает следующие преимущества:

- уменьшение размеров дочернего отношения: тип данных, используемый для кода, компактнее типа для описания;
- если потребуется изменить описание объекта, достаточно будет внести изменение в одну из записей справочной таблицы и не изменять множество записей в дочернем отношении;
- справочные таблицы используют для эффективного контроля правильности данных, вводимых пользователем.

Но если на справочные таблицы ссылаются часто выполняющиеся или важные запросы, а изменения в описании объекта весьма маловероятны, следует подумать о

дублировании атрибута описания в дочернем отношении. Первоначальная справочная таблица не становится лишней, поскольку ее можно использовать для контроля правильности вводимых пользователем данных. Однако, повторив описание объекта в дочернем отношении, можно избавиться от необходимости формировать соединение дочернего отношения со справочной таблицей.

Создание таблиц из данных, содержащихся в других таблицах

Бывают случаи, когда отчеты необходимо формировать в самое напряженное (пиковое) время, а для их подготовки требуется доступ к данным, хранимым в разных таблицах, и соединение нескольких отношений. Однако данные, включаемые в отчет, во многих случаях относительно неизменны или не должны обязательно отражать самые актуальные данные (т.е. отчет считается приемлемым, если данные получены несколько часов тому назад). В таких случаях можно создать единую, в значительной степени денормализованную таблицу, содержащую те поля из разных базовых отношений, на которые ссылается отчет; пользователь будет использовать эту таблицу вместо того, чтобы непосредственно обращаться к базовым отношениям. Такие таблицы чаще всего создаются и заполняются в пакетном режиме в ночное время, когда система загружена незначительно.

Вопросы и задания к параграфу 5.3

1. Какая задача решается на этапе логического проектирования данных?
2. Какие действия предусмотрены на этапе логического проектирования данных для реляционной модели? В каких случаях они выполняются?
3. Какие факторы в основном влияют на успех применения семантической методики?
4. Вспомните простейшие правила перехода от ER-схемы к реляционной схеме БД.
5. За счет чего повышается качество схемы при использовании усовершенствованных правила перехода от ER-схемы к реляционной схеме БД?
6. Какие решения предлагают усовершенствованные правила для множеств связей типа 1:1? Какими рассуждениями следует сопровождать применение этих правил?
7. Какие решения предлагают усовершенствованные правила для множеств связей типа 1:*M*? Какими рассуждениями следует сопровождать применение этих правил?
8. Как изменились решения в случае однозначных атрибутов множеств связей?
9. Какими методами могут быть представлены в реляционной модели специализации и категоризации?
10. Какие критерии могут направлять процесс выбора метода представления специализаций и категоризаций?
11. Какие факторы необходимо учесть при выборе метода?
12. В чем особенность семантической методики, использующей ERM-модель?
13. Укажите и охарактеризуйте критерии качества проекта схемы БД.
14. Как методика трансформации ERM-схем в реляционные схемы обеспечивает удовлетворение этих критериев?
15. Опишите шаги этой методики.
16. Какой принцип используется для денормализации отношений на завершающем этапе логического проектирования?
17. Опишите типичные случаи денормализации.

5.4. Физическое проектирование данных

Физическое проектирование данных

1. **Анализ транзакций.**
2. **Выбор файловой структуры.**
3. **Определение индексов.**
4. **Определение требований к дисковой памяти.**

В идеале структурная часть схемы должна отражать только логические свойства БД. На практике структуры оказывают значительное влияние на производительность системы. Это справедливо, по крайней мере, для существующих коммерческих систем. Будем надеяться, что системы будущего приблизятся к идеалу, при котором заботой проектировщика является определение лишь логической схемы и требований транзакций, а выбор эффективной физической схемы осуществляется системой автоматически. Но и в этом случае вопросы производительности не потеряют своей актуальности. Следовательно, как современная, так и перспективная архитектура предопределяют необходимость анализа схемы с точки зрения ее влияния на показатели производительности. Поскольку в данной книге основной акцент сделан на логических свойствах данных, мы рассмотрим физическое проектирование только вкратце и только в связи с проектированием схемы.

Обеспечение производительности БД – весьма сложная проблема. Обычно альтернатив выбора параметров физической организации достаточно много, причем, к сожалению, эти параметры не являются независимыми.

Целью этапа физического проектирования данных является определение оптимальной файловой организации для хранения базовых отношений, а также индексов, необходимых для достижения приемлемой производительности.

Существует несколько показателей, которые могут быть использованы для оценки достигнутой эффективности.

- Производительность выполнения транзакций. Этот показатель представляет собой количество транзакций, которые могут быть обработаны за заданный интервал времени. В некоторых системах, например в службах резервирования авиабилетов, обеспечение высокой производительности выполнения транзакций является решающим фактором успешной эксплуатации всей системы.
- Время ответа. Этот показатель характеризует временной промежуток, необходимый для выполнения одной транзакции. С точки зрения пользователя желательно сделать время ответа системы минимальным. Однако существуют некоторые факторы, которые оказывают влияние на быстродействие системы, но не могут контролироваться разработчиками, например, уровень загрузки системы или время, затрачиваемое на передачу данных.

- Дисковая память. Этот показатель представляет собой объем дискового пространства, необходимого для размещения файлов базы данных. Разработчик должен стремиться минимизировать объем используемой дисковой памяти.

Однако ни один из этих факторов не является самодостаточным. Как правило, разработчик вынужден искать компромисс между этими показателями для достижения приемлемого баланса. Например, увеличение объема хранимых данных может вызвать увеличение времени ответа системы или уменьшение производительности выполнения транзакций.

Исходный вариант физического проекта базы данных не следует рассматривать как нечто неизменное, а нужно применять как средство оценки возможного уровня производительности системы. После реализации исходного варианта проекта необходимо вести наблюдение за показателями работы системы и в соответствии с полученными результатами выполнять ее настройку с целью улучшения показателей работы и учета изменяющихся требований пользователей. Многие типы СУБД предоставляют в распоряжение администратора базы данных (англ. Data Base Administrator – DBA) комплект утилит, предназначенный для текущего контроля над функционированием системы и ее настройки.

Диапазон выбора возможных типов организации файлов зависит от целевой СУБД, поскольку различные системы поддерживают разные наборы допустимых структур хранения информации. Очень важно, чтобы разработчик физического проекта базы данных имел полное представление обо всех типах структур хранения данных, поддерживаемых целевой СУБД, а также обо всех особенностях использования этих структур в системе.

Приняв во внимание все вышесказанное, приступим к обсуждению тех действий, которые должны быть выполнены на четвертом этапе разработки схемы базы данных.

Физическое проектирование данных. Анализ транзакций

Цель. Определение характеристик транзакций, которые будут выполняться в проектируемой базе данных, и выделение наиболее важных из этих транзакций.

• Отношения и атрибуты, к которым осуществляется доступ в процессе выполнения транзакции, а также тип доступа.

• Атрибуты, которые используются в предикатах.

• Атрибуты, участвующие в соединении двух или нескольких отношений, атрибуты, применяемые для упорядочения результатов, атрибуты, применяемые для группирования данных.

• Ожидаемая частота выполнения транзакции.

• Установленные показатели производительности для транзакции.

Анализ транзакций

Цель. Определение характеристик транзакций, которые будут выполняться в проектируемой базе данных, и выделение наиболее важных из этих транзакций.

Для того чтобы разрабатываемый физический проект базы данных обладал требуемым уровнем эффективности, необходимо получить максимум сведений о тех транзакциях и запросах, которые будут выполняться в базе данных. Нам потребуются как качественные, так и количественные характеристики. Для успешного планирования каждой транзакции необходимо знать следующее:

- транзакции, выполняемые наиболее часто и оказывающие существенное влияние на производительность;
- транзакции, наиболее важные для работы организации;
- периоды времени на протяжении суток/недель, в которые нагрузка базы данных возрастает до максимума (называемые периодами пиковой нагрузки).

После определения критических транзакций подробно анализируется каждая из них. Для каждой транзакции необходимое выяснить следующее.

- Отношения и атрибуты, к которым осуществляется доступ в процессе выполнения транзакции, а также тип доступа. Последнее означает определение того, выполняется ли в этой транзакции вставка, обновление, удаление или выборка данных (транзакции последнего типа называются также запросами). При изучении транзакции обновления необходимо определить, какие атрибуты обновляются в данной транзакции.
- Атрибуты, которые используются в предикатах (в языке SQL предикатами являются условия, указанные в конструкции *WHERE*). Проверка того, предусматривают ли эти предикаты следующее:
  - сопоставление с шаблоном, например, *name LIKE ' %Smith% '*;
  - поиск в диапазоне, например, *salary BETWEEN 10000 AND 20000*;
  - выборка по точному значению ключа, например, *salary = 30000*.

  Эта задача должна быть выполнена не только по отношению к запросам, но и к транзакциям обновления и удаления, поскольку в них также может быть ограничено количество строк, обновляемых или удаляемых в некотором отношении. Следует учитывать, что такие атрибуты могут потребовать создания вспомогательных структур доступа.
- Атрибуты, участвующие в соединении двух или нескольких отношений, атрибуты, применяемые для упорядочения результатов запросов, атрибуты,

применяемые для группирования данных в запросах. Эти атрибуты также могут потребовать применения вспомогательных структур доступа.

- Ожидаемая частота выполнения транзакции; например, может быть установлено, что транзакция выполняется примерно 50 раз в сутки.
- Установленные показатели производительности для транзакции; например, требование, чтобы транзакция выполнялась в течение 1 секунды.

Эта информация позволяет определить отношения, требующие тщательного изучения для решения вопроса об использовании вспомогательных структур доступа. Наибольший приоритет при определении вспомогательных структур доступа должны иметь атрибуты, применяемые в часто выполняемых или важных транзакциях.

Физическое проектирование данных. Выбор файловой структуры

Цель. Определение наиболее эффективного файлового представления для каждого из базовых отношений.

Варианты представления базовых отношений в СУБД Oracle:

- **Обычная таблица.**
- **Секционированная таблица.**
- **Индекс-таблица.**
- **Кластеризованная таблица.**

Выбор файловой структуры

Цель. Определение наиболее эффективного файлового представления для каждого из базовых отношений.

Одной из главных задач физического проектирования базы данных является обеспечение эффективного хранения данных. Например, если выборка строк с данными о сотрудниках компании должна выполняться по именам в алфавитном порядке, то наиболее подходящей структурой для хранения этих данных является файл, отсортированный по именам сотрудников. А если должна выполняться выборка данных обо всех сотрудниках, зарплата которых находится в определенном диапазоне, файл, отсортированный по именам сотрудников, не подходит для выполнения такой задачи. Положение еще более усложняется в связи с тем, что некоторые способы организации файлов хорошо подходят для массовой загрузки данных в базу данных, но их применение в дальнейшем становится неэффективным. Иными словами, задача состоит в использовании эффективной структуры хранения данных на внешнем устройстве, которая обеспечивает быструю загрузку базы данных, а затем ее преобразование в структуру, более подходящую для повседневной эксплуатации.

Итак, цель настоящего этапа состоит в определении оптимальной файловой организации для каждого отношения, если целевая СУБД это позволяет. Во многих случаях реляционная СУБД может предоставлять лишь ограниченный выбор или даже вообще не позволять выбрать определенную файловую организацию, но в некоторых СУБД на организацию размещения данных на внешних устройствах можно в определенной степени повлиять.

Рассмотрим варианты представления базовых отношений в СУБД Oracle.

Обычная таблица.

Обычная таблица – это наиболее часто используемая форма хранения данных (по умолчанию). Администратор БД может оказать очень незначительное влияние на расположение строк некластеризованной таблицы. Строки могут храниться в любом порядке, в зависимости от операций, выполняемых с данной таблицей.

Секционированная таблица.

Секционированная таблица используется для создания масштабируемых приложений. Она обладает следующими особенностями.

- Секционированная таблица состоит из одной или более секций, в каждой из которых строки хранятся согласно определенному диапазону ключевых значений, хеш-функции или с учетом того и другого.
- Каждая секция секционированной таблицы – это сегмент, который может располагаться в отдельном табличном пространстве (ТП).

- Секции особенно практичны для больших таблиц, управление которыми может осуществляться при помощи нескольких параллельных процессов.
- Для сопровождения секций такой таблицы имеется особый набор команд.

Индекс-таблица.

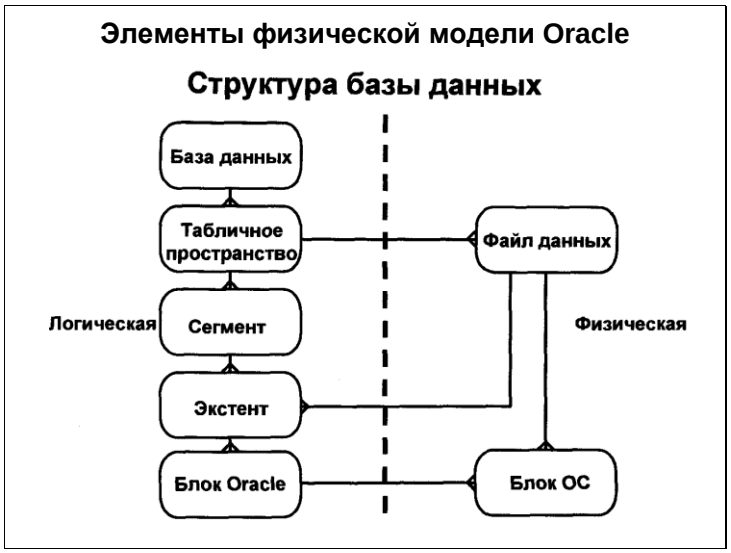
Индекс-таблица аналогична таблице с индексом по первичному ключу, составленному по одному или нескольким столбцам. Отличие заключается в том, что в индекс-таблице все данные хранятся непосредственно в структуре индексного дерева, поскольку листовые блоки в индексном дереве содержат не значения *ROWID* (физический адрес строки), а неключевые столбцы. Т.о. использование индекс-таблиц исключает потребность в двух отдельных сегментах – для хранения таблиц и индексов.

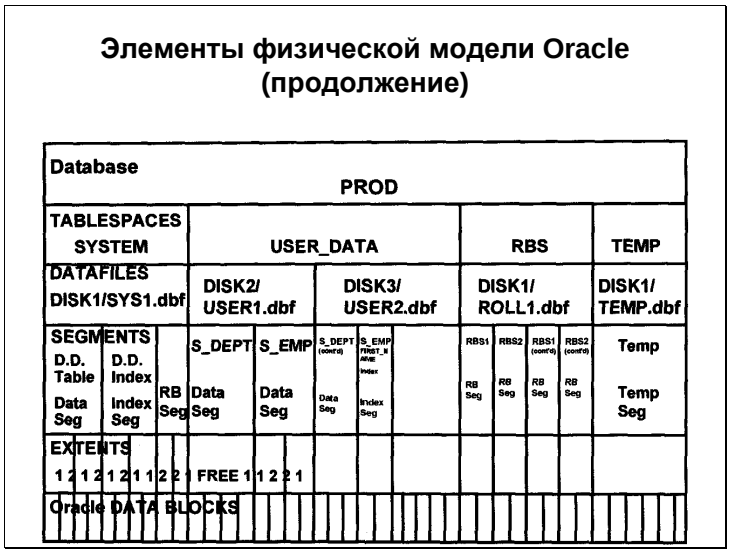
Индекс-таблицы предоставляют быстрый доступ к данным либо через первичный ключ, либо через комбинацию столбцов, составляющую левую часть первичного ключа, для выполнения запросов с условием точного соответствия или поиска в интервале значений.

Кластеризованная таблица.

Кластер может использоваться для хранения связанных наборов строк группы таблиц в одном блоке БД. Кластеризованные таблицы предоставляют оптимальный способ хранения данных нескольких таблиц, которые обычно используются вместе. Кластер характеризуется следующими особенностями.

- Имеется кластерный ключ, используемый для идентификации строк, которые должны храниться вместе.
- Кластерный ключ может состоять из одного или нескольких столбцов.
- Таблицы в кластере имеют столбцы, соответствующие кластерному ключу.
- Кластеризация – это прозрачный механизм использования таблиц в приложении. Работа с данными в кластеризованных таблицах выполняется так же, как и в обычных таблицах.
- Обновление одного из столбцов кластерного ключа может привести к физическому перемещению строки.
- Кластерный ключ не зависит от первичного ключа. Набор столбцов первичного ключа кластеризованных таблиц может совпадать с набором столбцов кластерного ключа или отличаться от него.





В ранее приведенном тексте использовались такие важные понятия, как табличное пространство, сегмент, блок данных. Для их правильного понимания рассмотрим более подробно элементы физической модели СУБД Oracle.

Архитектура БД Oracle состоит из логических и физических структур, которые образуют БД. Физические структуры включают управляющие файлы, оперативные журнальные файлы и файлы БД. Логические структуры включают табличные пространства, сегменты, экстенты и блоки данных.

Табличные пространства.

Информация в БД Oracle хранится в табличных пространствах (ТП). БД Oracle может быть разделена на логические области, называемые табличными пространствами. Каждое ТП в определенный момент времени принадлежит только одной БД. Каждое ТП состоит из одного или более файлов ОС, которые называются файлами данных. ТП может состоять из одного или более сегментов. ТП могут быть переведены в оперативный режим во время работы БД. ТП могут быть переведены в автономный режим, в то время как БД останется запущенной. ТП могут принимать статус «только чтение» и «чтение-запись».

База данных состоит по крайней мере из одного ТП – *SYSTEM*. Дополнительные ТП БД создаются для повышения ее управляемости и упрощения эксплуатации. Сервер Oracle, по существу, различает два вида ТП: *SYSTEM* и все остальные.

ТП *SYSTEM*:

- создается во время создания БД,
- необходимо для функционирования всей БД,
- содержит словарь данных, включая определения хранимых программных единиц,
- содержит сегмент отката *SYSTEM,*
- не должно содержать пользовательские данные, хотя это возможно.

ТП, отличные от *SYSTEM*:

- обеспечивают большую гибкость для администрирования БД,
- обеспечивают раздельное хранение сегментов отката, временных сегментов, данных и индексов приложений,
- обеспечивают раздельное хранение данных в соответствие с требованиями резервирования,
- обеспечивают раздельное хранение динамических и статических данных,
- обеспечивают контроль выделения пространства под объекты пользователя.

Файлы данных.

Каждое ТП БД Oracle состоит из одного или более файлов, называемых файлами данных (ФД). Это физические структуры, соответствующие ОС, на которой работает сервер Oracle. ФД может принадлежать только одному ТП. Сервер Oracle создает ФД для какого-либо ТП, отводя для него указанное количество дискового пространства с небольшим резервом. Администратор БД может изменять размеры ФД после его создания или организовать его таким образом, что он будет динамически расширяться по мере того, как растут объекты ТП.

Сегменты.

Сегмент – это пространство, выделенное под определенную логическую структуру хранения в ТП. Например, пространство, отведенное для хранения таблицы, образует сегмент. ТП может состоять из одного или более сегментов. Сегмент не может располагаться в нескольких ТП; однако, сегмент данных может охватывать несколько файлов, принадлежащих одному и тому же ТП. Каждый сегмент состоит из одного или более экстентов.

Экстенты.

Экстент – это набор нескольких смежных блоков Oracle. Пространство под сегменты выделяется экстентами. Один или более экстентов образуют сегмент. Когда сегменты создаются, они содержат по крайней мере один экстент. По мере роста сегмента к нему добавляются новые экстенты. АБД может вручную добавить экстенты сегменту. Экстент должен существовать в рамках одного файла данных, но необязательно охватывать его целиком.

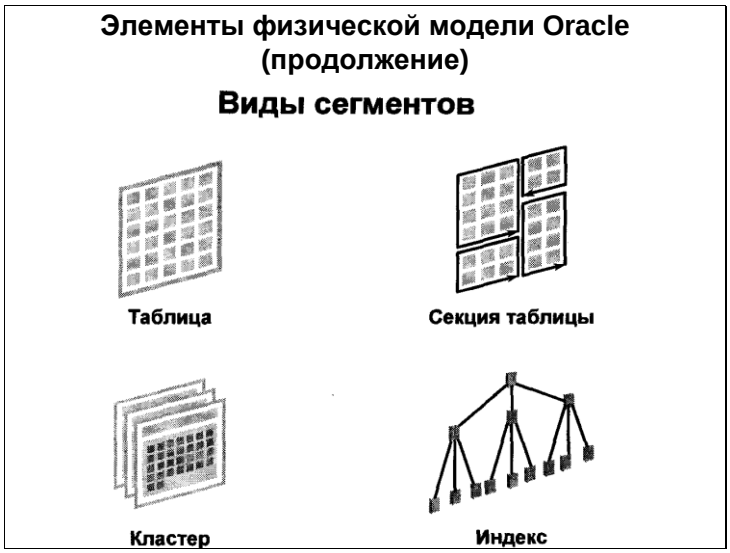
Блоки данных.

Сервер Oracle управляет пространством хранения в ФД с помощью структурных единиц, называемых блоками данных или блоками. На конечном уровне детализации информация в БД Oracle хранится в блоках. Блок данных – это самая мелкая единица хранения, которую сервер Oracle может выделять, читать и писать. Один блок данных соответствует одному или более физическим блокам ОС, отведенным для него в существующем ФД. Размер блока указывается для каждой БД во время ее создания при помощи параметра инициализации *DB_BLOCK_SIZE*. Размер блока должен быть кратен размеру блока ОС, чтобы не было излишнего ввода-вывода. Максимальный размер блока БД зависит от ОС.

Блоки данных Oracle имеют следующую структуру.

- Заголовок блока (адрес блока данных, список указателей таблиц, список указателей строк и слоты транзакций, которые используются транзакциями, вносящими изменения в строки этого блока). Заголовки блоков растут сверху вниз.

- Пространство данных (данные столбцов хранящихся в этом блоке строк). Заполняется снизу вверх.
- Свободное пространство (располагается посередине между ними, что предоставляет возможность роста и заголовку, и данным строк.) Свободное пространство блока изначально непрерывно. Однако удаления и обновления могут фрагментировать свободное пространство блока. Сервер Oracle объединяет свободные пространства блока при необходимости.



Сегменты – это объекты, использующие пространство БД. В СУБД Oracle имеются следующие виды сегментов.

Таблица.

Таблица, также называемая некластеризованной и несекционированной таблицей, является наиболее часто используемым способом хранения информации в БД. Данные в таблице хранятся неупорядочено, и администратор БД практически не может контролировать распределение строк по блокам таблицы. Все данные несекционированной таблицы должны храниться в одном ТП.

Секция таблицы.

Наиболее важными свойствами таблицы БД, информация которой интенсивно используется несколькими процессами, являются ее доступность и масштабируемость. В таких случаях данные таблицы могут храниться в нескольких секциях, которые расположены в различных ТП. В настоящий момент сервер Oracle обеспечивает секционирование по диапазону ключевых значений и хеш-секционирование. Каждой секции секционированной таблицы соответствует свой сегмент, для управления которым могут быть указаны особые параметры хранения.

Кластер.

Строки в кластере хранятся согласно ключевым значениям столбцов. Кластер может содержать одну или более таблиц и является одной из разновидностей сегментов данных. Таблицы кластера принадлежат одному и тому же сегменту и имеют одинаковые параметры хранения.

Индекс.

Все записи индекса хранятся в одном индексном сегменте. Если таблица имеет три индекса, то, соответственно, используются три индексных сегмента. Назначением этого сегмента является возможность определения местоположения строк согласно указанному ключевому значению.

Индекс-таблица.

В индекс-таблицах данные хранятся согласно индексу, основанному на ключевом значении. В случае использования индекс-таблицы чтение строк данных из таблицы не требуется, так как все данные могут быть извлечены непосредственно из дерева индекса.

Индексная секция.

Индекс может быть секционирован и распределен по нескольким ТП. В этом случае каждая секция индекса соответствует какому-то сегменту и не может охватывать более одного ТП. Основной целью использования секционированного индекса является необходимость минимизации конкуренции при помощи распределения ввода-вывода индекса.

Сегмент отката (отмены).

Сегмент отката (отмены) используется транзакциями, вносящими изменения в БД. Перед изменением блоков данных или индексов их исходное значение помещается в сегмент отката. Это предоставляет пользователю возможность отменять внесенные изменения.

Временный сегмент.

Для выполнения сортировки данных в таких командах, как *CREATE INDEX*, *SELECT DISTINCT, SELECT ... GROUP BY,* Oracle старается максимально использовать оперативную память. Иногда приходится записывать на диск промежуточные результаты тех сортировок, для которых требуется много пространства, например, при создании индексов для больших таблиц. В этом случае создаются временные сегменты.

Сегмент LOB.

Для хранения больших объектов (англ. Large Object – LOB), таких как текстовые документы, изображения или видеоданные, могут использоваться один или более столбцов таблицы. Если этот столбец особенно велик, сервер Oracle хранит его значения в отдельном сегменте, называемом сегментом LOB. В таблице содержится лишь указатель на местоположение соответствующих данных LOB.

Индекс LOB.

Индексный сегмент LOB неявно организуется при создании LOB. Администратор БД может указать параметры хранения индекса LOB. Индекс LOB используется для поиска определенных значений столбца LOB.

Физическое проектирование данных. Определение индексов

Цель. Определение того, будет ли добавление индексов способствовать повышению производительности системы.

1. Не создавать индекс на небольших отношениях.

2. Следует создавать индекс на первичном ключе отношения

3. Ввести дополнительный индекс на внешнем ключе.

4. Ввести дополнительные индексы на всех атрибутах, которые часто применяются в качестве дополнительного поискового ключа.

5. Ввести дополнительные индексы на атрибутах, которые часто применяются в конструкциях *WHERE*, *ORDER BY*, *GROUP BY*;

6. Не индексировать атрибут или отношение, которые часто обновляются.

7. Не индексировать атрибут, если в запросах с использованием этого атрибута обычно происходит выборка значительной части (например, 25%) кортежей в отношении.

8. Не индексировать атрибуты, которые состоят из длинных символьных строк.

Определение индексов

Цель. Определение того, будет ли добавление индексов способствовать повышению производительности системы.

Один из вариантов выбора подходящей файловой структуры для отношения состоит в том, что строки остаются неупорядоченными и создается любое необходимое количество дополнительных индексов. Еще один вариант предусматривает упорядочение строк в отношении с использованием первичного или кластеризующего индекса.

Дополнительные индексы предоставляют возможность определять дополнительные поисковые ключи для базового отношения, которые могут применяться для повышения эффективности выборки данных.

Сопровождение и применение дополнительных индексов приводит к увеличению издержек, поэтому необходимо определить, оправдывают ли они повышение производительности при выборке данных, достигнутое благодаря их использованию. Основные издержки, связанные с применением дополнительных индексов, перечислены ниже.

- Ввод индексной записи в каждый дополнительный индекс при вставке строки в отношение.
- Обновление дополнительного индекса при удалении или обновлении соответствующей строки в отношении.
- Увеличение потребности в дисковом пространстве в связи с необходимостью хранения дополнительного индекса.
- Возможное снижение производительности процесса оптимизации запросов, поскольку оптимизатор запросов должен учесть наличие всех дополнительных индексов и только после этого выбрать оптимальную стратегию выполнения запросов.

Один из способов определения необходимого количества дополнительных индексов состоит в подготовке списка требований к атрибутам, которые рассматриваются для определения необходимости применения их для индексации, а затем изучении затрат на сопровождение каждого из этих индексов. Ниже приведены рекомендации по подготовке такого списка требований.

1. Не создавать индекс на небольших отношениях. Может оказаться более эффективным поиск в отношении, данные которого хранятся в буфере оперативной памяти, чем хранить дополнительную индексную структуру.

2. Как правило, следует создавать индекс на первичном ключе отношения, если он не применяется в качестве ключа файловой структуры. Хотя в стандарте SQL предусмотрена конструкция, позволяющая задать спецификацию первичного ключа,

следует отметить, что применение этой конструкции не всегда гарантирует создание индекса на первичном ключе.

3. Ввести дополнительный индекс на внешнем ключе, если с его помощью часто происходит доступ к отношению. Но следует учитывать, что в некоторых СУБД индексы на внешних ключах создаются автоматически.

4. Ввести дополнительные индексы на всех атрибутах, которые часто применяются в качестве дополнительного поискового ключа.

5. Ввести дополнительные индексы на атрибутах, которые часто применяются в конструкциях *WHERE, ORDER BY, GROUP BY.*

6. Не индексировать атрибут или отношение, которые часто обновляются.

7. Не индексировать атрибут, если в запросах с использованием этого атрибута обычно происходит выборка значительной части (например, 25%) кортежей в отношении. В таком случае может оказаться более эффективным поиск прямым перебором кортежей отношения, чем поиск с использованием индекса.

8. Не индексировать атрибуты, которые состоят из длинных символьных строк.

Целесообразно по возможности провести эксперименты для определения того, способствует ли создание индекса повышению производительности, почти не влияет на производительность или приводит к ее снижению. Если обнаруживается снижение производительности, этот индекс, безусловно, должен быть удален из списка требований. А если в результате ввода дополнительного индекса наблюдается лишь незначительное повышение производительности, может потребоваться дальнейшее исследование для определения того, при каких обстоятельствах этот индекс может оказаться полезным, и так ли часто возникают эти обстоятельства, чтобы создание индекса действительно было оправданным.

Если в отношении с одним или несколькими индексами происходит вставка большого количества строк, может оказаться более эффективным решение вначале удалить индексы, выполнить вставку, а затем снова создать индексы. В качестве эмпирического правила можно указать, что если в результате вставки общий объем данных в отношении увеличивается по меньшей мере на 10%, целесообразно удалить на время индексы этого отношения.

Физическое проектирование данных. Определение требований к дисковой памяти

Цель. Определить объем дискового пространства, который требуется для базы данных.

***CREATE TABLE* [<*имя схемы/владельца*>.]**
<*имя таблицы*>
(<*имя столбца*> <*тип данных столбца*>
[,<*имя столбца*> <*тип данных столбца*>]...)
[*TABLESPACE* <*имя ТП*>]
[*PCTFREE* <*целое*>]
[*PCTUSED* <*целое*>]
[*INITRANS* <*целое*>]
[*MAXTRANS* <*целое*>]
[<*фраза STORAGE*>]
[*LOGGING* | *NOLOGGING*]

Определение требований к дисковой памяти

Цель. Определить объем дискового пространства, который требуется для базы данных.

Во многих проектах выдвигается требование, чтобы физическая реализация базы данных могла быть осуществлена на основе существующей конфигурации аппаратных средств. Но даже при отсутствии такого требования проектировщик обязан оценить объем дискового пространства, который требуется для хранения файлов базы данных, хотя бы на тот случай, что в связи с этим потребуется приобретение новых аппаратных средств. Цель данного этапа состоит в оценке объема дискового пространства, необходимого для поддержки реализации базы данных во внешней памяти. Как и на предыдущих этапах, проведение оценки потребности в дисковом пространстве в значительной степени зависит от целевой СУБД и от аппаратных средств, которые применяются для поддержки функционирования базы данных. Как правило, такая оценка основана на информации о среднем размере каждой строки и количестве строк в отношении. Начальная оценка объема дискового пространства может быть определена, исходя из текущего (начального) состояния БД, но может также оказаться целесообразным проведение анализа интенсивности роста размеров отношения и корректировка итоговых сведений об объеме диска с учетом полученного коэффициента роста для определения возможных размеров базы данных в будущем.

Далее мы рассмотрим лишь некоторые параметры, определяющие расход дискового пространства. Все они указываются при создании таблицы.

CREATE TABLE [<*имя схемы/владельца*>.] <*имя таблицы*>

(<*имя столбца*> <*тип данных столбца*>

[,<*имя столбца*> <*тип данных столбца*>]...)

[*TABLESPACE* <*имя ТП*>] – ТП, в котором будет создана таблица;

[*PCTFREE* <*целое*>] – объем пространства, резервируемого в каждом блоке (процентное отношение от пространства данных, т.е. общего размера блока минус заголовок) для потенциального роста длины строк;

[*PCTUSED* <*целое*>] – нижний предел используемости пространства блока (после заполнения до *PCTFREE*), ниже которого необходимо опуститься, чтобы блок вновь стал доступным для вставки;

[*INITRANS* <*целое*>] – начальное количество транзакций, которые могут одновременно обращаться к строкам блока данных (по умолчанию – 1);

[*MAXTRANS* <*целое*>] – максимальное количество транзакций, которые могут одновременно обращаться к строкам блока данных (по умолчанию – 255);

[<*фраза STORAGE*>] – фраза, определяющая правила выделения экстентов таблице;

[*LOGGING* | *NOLOGGING*] – ключевые слова, определяющие будет или нет сохраняться в журнальном файле информация о создании и последующих операциях изменения таблицы.

Большое значение параметра *PCTFREE* резервирует больший объем пространства блока БД для обновлений. Для этого параметра должно быть установлено большое значение, если таблица содержит столбцы, которые сначала имеют пустое значение, но могут быть изменены, и столбцы, размер которых может возрасти в результате обновлений. Если указать большое значение, то блоки будут менее плотными, каждый блок сможет разместить меньшее количество строк. Оценку значения параметра можно получить следующим образом:

*PCTFREE = (СРЕДНИЙ РАЗМЕР СТРОКИ – ИСХОДНЫЙ РАЗМЕР СТРОКИ) * 100 / СРЕДНИЙ РАЗМЕР СТРОКИ.*

Параметр *PCTUSED* используется для того, чтобы обеспечить возврат блока в список свободных только в том случае, если в нем будет достаточно свободного пространства для вставки строки среднего размера. Если какой-либо блок из списка свободных блоков не имеет достаточно свободного пространства для вставки строки, сервер Oracle проверяет наличие такового в следующем блоке списка. Этот последовательный поиск продолжается до тех пор, пока не будет найден блок, содержащий нужный объем свободного пространства или не будет достигнут конец списка. Использование параметра сокращает время поиска в списке свободных блоков, так как увеличивается вероятность раннего обнаружения блока, содержащего требуемый объем свободного пространства. Оценку значения параметра можно получить следующим образом:

*PCTUSED = 100 – PCTFREE – (СРЕДНИЙ РАЗМЕР СТРОКИ * 100 / ОБЪЕМ ПРОСТРАНСТВА ДАННЫХ)*

Строка, размер которой вырос в результате обновления, может не поместиться целиком в блоке, для которого было установлено небольшое значение параметра *PCTFREE*. В таком случае сервер Oracle переместит всю строку в новый блок, а в исходный блок поместит указатель на новое местоположение строки. Этот процесс называется миграцией строк. Производительность ввода-вывода относительно этого блока понижается, так как сервер должен просмотреть два блока данных для извлечения мигрировавшей строки.

Если строка слишком велика для того, чтобы можно было разместить ее в каком-либо блоке, выполняется сцепление строк. Это может произойти в том случае, если строка содержит очень длинные столбцы. Сервер разделяет строки на меньшие порции данных, называемые отрезками строки. Вместе с каждым отрезком строки в блоке хранится указатель, необходимый для извлечения и составления всей строки. Сцепление строк можно минимизировать, используя блоки большего размера или разделив исходную таблицу на несколько других, содержащих меньшее число столбцов.

Приведенные сведения составляют лишь малую толику тех знаний, которыми должен обладать администратор БД Oracle, чтобы помочь разработчикам в физическом проектировании данных. Но даже эта информация говорит о том, что настройка БД на физическом уровне – дело очень сложное. Самое главное, что эта работа (в отличие от проектирования логической схемы данных приложения) не разовая, она должна продолжаться все время, пока БД будет эксплуатироваться. И это одна из самых важных задач АБД. Но подробный рассказ об этом выходит за рамки данной книги.

Вопросы и задания к параграфу 5.4

1. Какие действия осуществляются на этапе физического проектирования данных?

2. Какие виды сегментов предоставляет СУБД Oracle для хранения таблиц?
3. Каких правил следует придерживаться при построении индексов?
4. Какими параметрами команды CREATE TABLE определяются требования к дисковой памяти?