4. СУБД-ориентированные модели данных

4.1. Обзор СУБД-ориентированных моделей данных

Своим названием этот класс моделей данных обязан тому факту, что в отличие от семантических моделей, ориентированных на человека, модели этого класса предназначены для реального создания БД во внешней памяти компьютеров, поскольку они реализованы в виде программных продуктов – СУБД. Каждый разработчик СУБД по-своему подходил к интерпретации концепций модели, поэтому в этих системах были реализованы свои диалекты СУБД-ориентированных моделей данных. Собственно, последние можно считать своеобразным обобщением моделей данных конкретных СУБД.

Некоторые из моделей этого класса так и родились. Сначала в виде экспериментальных, а затем и коммерческих разработок появлялись системы, в которых предлагались новые концепции моделирования данных. Рано или поздно сообщество исследователей по моделям данных приходило к мысли стандартизовать схожие по основным концепциям подходы. Так появились иерархическая, сетевая, объектно-ориентированная и объектно-реляционная модели данных.

А вот реляционную модель, в отличие от остальных СУБД-ориентированных моделей данных, ждала другая судьба. Первоначально она родилась в голове у американского математика Эдгара Кодда, который увидел в популярном математическом понятии «отношение» перспективную форму данных. Вслед за ним в моделирование данных пришли многие результаты таких разделов математики, как теория множеств, теория отношений, логика высказываний, логика предикатов первого порядка. Таким образом, моделирование данных из технической инженерной дисциплины превратилось в приложение математики.

В ходе продолжительных и результативных теоретических и экспериментальных исследований родились и развивались теория реляционных БД, классическая методика проектирования реляционных схем БД, методы эффективного хранения отношений, методы оптимизации запросов к БД на спецификационных языках, методы обеспечения многопользовательского взаимодействия с БД и многое другое. И уже спустя несколько лет стали появляться сначала экспериментальные, а затем и коммерческие реляционные СУБД (РСУБД). Очень скоро они достигли такого состояния, при котором стали востребованными на рынке информационных систем и с тех пор постепенно вытеснили иерархические и сетевые СУБД, как инструменты разработки новых систем.

Следует отметить, что дореляционные системы (иерархические и сетевые СУБД) первоначально не основывались на каких-либо абстрактных моделях. В них доступ к БД производился на уровне записей. Пользователи этих систем осуществляли явную навигацию в БД, используя языки программирования, расширенные функциями СУБД. Интерактивный доступ к БД поддерживался только путем создания соответствующих прикладных программ с собственным интерфейсом. Понятие модели данных фактически вошло в обиход специалистов в области БД только вместе с реляционным подходом. Абстрактные представления ранних систем появились позже на основе анализа и выявления общих признаков у различных конкретных систем.

Иерархическая модель данных

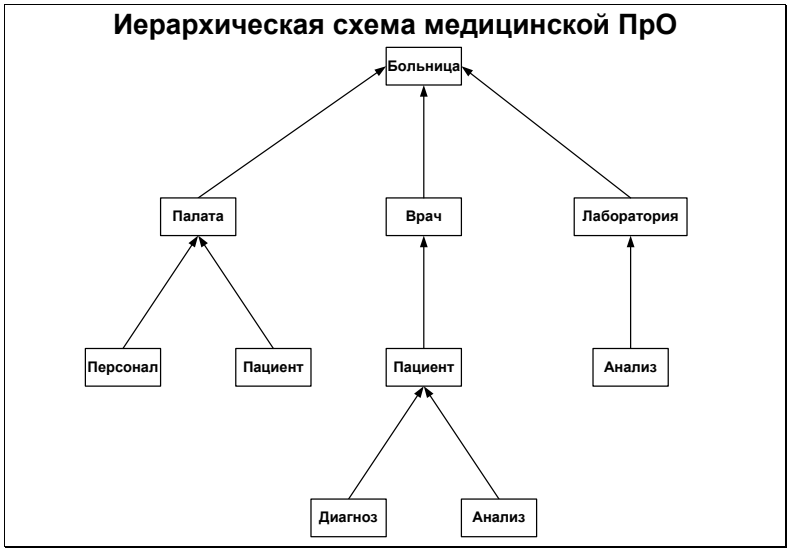

Схема БД в **иерархической модели** состоит из нескольких типов записей, один из которых определен как корневой, или входной, тип записей. Каждый тип записей может состоять из нескольких элементарных типов полей, некоторые из них могут являться ключами, однозначно идентифицирующими каждую запись. Между типами записей в иерархии могут быть определены типы связей «один ко многим» (иногда «один к одному»), где тип записей, соответствующий элементу «один» типа связей, определяется как исходный (родительский), а соответствующий элементу «много» – как порожденный (дочерний).

Тип записей может быть порожденным только в одном типе связей, это означает, что для каждого типа записей может существовать только один исходный тип записей. Однако каждый тип записей может быть исходным во многих типах связей. Корневой тип записей может быть только исходным. Обычно между двумя типами записей может быть только один тип связей.

На слайде показана иерархическая схема медицинской ПрО. Прямоугольниками представлены типы записей, дугами – типы связей. Дуги ориентированы в направлении действия функциональных отображений между дочерними и родительскими типами записей.

В силу особенностей модели (каждая дочерняя запись может иметь связь не более чем с одной родительской записью) мы вынуждены прибегнуть к дублированию типов записей и их экземпляров. Дубликаты типов записей показаны на слайде. Дубликаты записей типов *ПАЦИЕНТ* и *ЛАБОРАТОРИЯ* необходимы для представления в БД их связей с записями типов *ВРАЧ* и *БОЛЬНИЦА* соответственно, поскольку как пациенты, так и лаборатории могут быть связаны с несколькими врачами и больницами (а родитель у каждой записи может быть только один).

Иерархическая БД состоит из упорядоченного набора нескольких экземпляров одного типа дерева.

Автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: никакой потомок не может существовать без своего родителя.

Примеры типичных операторов манипулирования иерархически организованными данными:

- найти указанное дерево БД (например, дерево *больницы Святой Елены*);
- перейти от одного дерева к другому;
- перейти от одной записи к другой по иерархии (например, от больницы – к первому врачу этой больницы и наоборот);

- перейти от одной записи к другой того же типа в порядке обхода иерархии;
- вставить новую запись в указанную позицию;
- удалить текущую запись.

Типичным представителем иерархических СУБД (наиболее известным и распространенным) является Information Management System (IMS) фирмы IBM. Первая версия появилась в 1968 г. До сих пор поддерживается много баз данных этой системы.

Сетевая модель данных

Сетевой подход к организации данных является расширением иерархического. В иерархических структурах запись-потомок должна иметь в точности одного предка; в сетевой структуре данных потомок может иметь любое число предков, но в контексте разных типов связей (в пределах одного типа связей по-прежнему обязательно функциональное отображение между дочерними и родительскими записями).

В **сетевой модели** данных любая запись может быть одновременно и родительской, и дочерней, и может участвовать в образовании любого числа взаимосвязей с другими записями. Сетевая БД состоит из набора записей и набора связей между этими записями, а если говорить более точно – из набора экземпляров каждого типа из заданного в схеме БД набора типов записей и набора экземпляров каждого типа из заданного набора типов связей.

Тип связей определяется для двух типов записей: предка и потомка. Экземпляр типа связей состоит из одного экземпляра типа записей предка и упорядоченного набора экземпляров типа записей потомка. Для данного типа связей L с типом записей предка P и типом записей потомка C должны выполняться два условия:

- каждый экземпляр типа P является предком не более чем в одном экземпляре L;
- каждый экземпляр типа C является потомком не более чем в одном экземпляре L.

На формирование типов связей не накладываются особые ограничения; возможны, например, следующие ситуации.

- Тип записей потомка в одном типе связей L_1 может быть типом записей предка в другом типе связей L_2 (как в иерархии).
- Данный тип записей P может быть типом записей предка в любом числе типов связей.
- Данный тип записей P может быть типом записей потомка в любом числе типов связей.
- Может существовать любое число типов связей с одним и тем же типом записей предка и одним и тем же типом записей потомка; и если L_1 и L_2 – два типа связей с одним и тем же типом записей предка P и одним и тем же типом записей потомка C, то правила, по которым образуется родство, в разных типах связей могут различаться.
- Типы записей X и Y могут быть предком и потомком в одном типе связей и потомком и предком – в другом.

- Предок и потомок могут быть одного типа записей.

В отличие от реляционной модели, связи между записями здесь (как и в иерархической модели) реализуются с помощью указателей. Сетевую БД можно представить как граф с записями в виде узлов графа и связями в виде его ребер.

Набор операций для сетевой модели может быть таковым:

- найти конкретную запись в наборе однотипных записей (*врача Сидорова*);
- перейти от предка к первому потомку по некоторой связи (к первому *врачу больницы Святой Елены*);
- перейти к следующему потомку в некоторой связи;
- перейти от потомка к предку по некоторой связи (найти *больницу врача Сидорова*);
- создать новую запись;
- уничтожить запись;
- модифицировать запись;
- включить запись в набор связей конкретного предка;
- исключить запись из набора связей предка;
- перевести запись из набора связей одного предка в набор связей другого предка и т.д.

Типичным представителем сетевых СУБД является Integrated Database Management System (IDMS) компании Cullinet Software, Inc. Архитектура системы основана на предложениях Рабочей группы по базам данных (Data Base Task Group – DBTG) Комитета по языкам программирования (Conference on Data Systems Languages – CODASYL) – организации, ответственной за определение языка программирования Кобол. Отчет DBTG был опубликован в 1971 г., а позже появилось несколько систем, поддерживающих эту модель, среди которых была и IDMS.

Реляционная модель данных

На сегодняшний день реляционные СУБД являются доминирующим типом программного обеспечения для технологии БД. Ежегодный объем продаж в этом секторе рынка оценивается в 15-20 миллиардов долларов (или 50 миллиардов долларов вместе с инструментами разработки), причем ежегодный прирост этого объема составляет 25%. Данное программное обеспечение представляет собой второе поколение СУБД, основанное на использовании реляционной модели данных, предложенной Эдгаром Ф. Коддом (Edgar Frank Codd) в 1969 году. В **реляционной модели** все данные логически структурированы внутри отношений (таблиц). Каждое отношение имеет имя и состоит из именованных атрибутов (столбцов). Каждый кортеж (строка) содержит по одному значению каждого из атрибутов. Большое преимущество реляционной модели заключается именно в этой простоте логической структуры. Хотя, конечно же, за этой простотой скрывается серьезный теоретический фундамент, которого не было у первого поколения СУБД (т.е. у сетевых и иерархических СУБД).

Мы подробно познакомимся с этой замечательной моделью в следующем параграфе.

Объектно-ориентированная модель данных

Объектно-ориентированный подход является одним из новых подходов к созданию программного обеспечения, который считается очень перспективным для решения некоторых классических проблем разработки программного обеспечения. Базовой концепцией объектно-ориентированной технологии является то, что все программное обеспечение должно всегда, когда это возможно, создаваться на основе стандартных и повторно используемых компонентов.

Традиционно создание программного обеспечения и управление базами данных представляли собой совершенно разные дисциплины. Технология баз данных была сконцентрирована в основном на статических концепциях хранения информации о ПрО во внешней памяти компьютеров, тогда как технология создания программного обеспечения моделировала динамические аспекты приложений. С появлением следующего (третьего) поколения систем управления базами данных, а именно **объектно-ориентированных СУБД** (ООСУБД) и **объектно-реляционных СУБД** (ОРСУБД), эти две дисциплины слились воедино, что позволило параллельно моделировать данные и процессы обработки данных.

Однако это поколение СУБД вызвало горячие споры. Очевидный успех реляционных систем в течение четырех последних десятилетий позволяет сторонникам традиционных подходов считать, что реляционную модель достаточно подкрепить дополнительными (объектно-ориентированными) возможностями. Другие специалисты считают, что базовая реляционная модель неспособна адекватно обслуживать такие сложные приложения, как системы автоматизированного проектирования, системы автоматизированной разработки программного обеспечения и геоинформационные системы.

Разработка систем объектно-ориентированных баз данных началась в середине 80-х годов в связи с необходимостью удовлетворения требований приложений, отличных от тех приложений обработки данных, которые характерны для систем реляционных баз данных. Попытки использования технологий реляционных баз данных в таких сложных приложениях, как автоматизированное проектирование; автоматизированное производство; технология программирования; системы, основанные на знаниях, и мультимедийные системы, обнажили ограничения систем реляционных баз данных (РБД). Возникли потребности, которые лучшим образом удовлетворялись при применении объектно-ориентированных баз данных (ООБД).

В настоящее время на рынке представлено свыше 25 систем ООБД. Среди них – система GemStone компании Servio, ONTOS компании Ontos, ObjectStore компании Object Design. Многие из этих ООСУБД появились еще во второй половине 80-х годов, и сегодня, по прошествии двух десятилетий разработки они все еще не вступили в пору зрелости. В этом – одна из причин того, что по сей день мировой рынок реальных приложений не торопится принимать системы ООБД. Среди современных ООБД почти нет полностью оперившихся систем, сопоставимых с современными системами реляционных баз данных. Обсудим основные достижения и проблемы, связанные с нынешним состоянием ООБД.

Модель ООБД

Парадигма ООБД основывается на ряде базовых понятий, таких как объект, идентифицируемость, класс, наследование, перегрузка и отложенное связывание.

В объектно-ориентированной модели данных любая сущность реального мира представляется всего одним понятием – **объектом**. С объектом ассоциируется состояние и поведение. Состояние объекта определяется значениями его свойств – **атрибутов.** Значениями свойства могут являться примитивные значения (такие, как строки или целые числа) и непримитивные объекты. Непримитивный объект, в свою очередь, состоит из набора свойств. Следовательно, объекты можно рекурсивно определять в терминах других объектов. Поведение объекта определяется с помощью **методов**, которые оперируют состоянием объекта.

У каждого объекта имеется определяемый системой **уникальный идентификатор**. Объекты, обладающие одними и теми же свойствами и поведением, группируются в **классы**. Объект может быть экземпляром только одного класса.

Классы организуются в **иерархии классов**. Подкласс наследует свойства и методы суперкласса; кроме того, подклассы могут обладать индивидуальными свойствами и методами. В некоторых системах, например, ORION, у класса может быть более одного

суперкласса (**множественное наследование**), тогда как в других системах число суперклассов ограничено одним (**одиночное наследование**).

В большинстве ООСУБД допускается **перегрузка** унаследованных свойств и методов. Перегрузка состоит в замене домена свойств новым доменом или в замене одной реализации метода другой его реализацией.

Достоинства модели ООБД

Объектно-ориентированные базы данных позволяют представлять сложные объекты более непосредственным образом, нежели реляционные системы. Остановимся на некоторых имеющихся достижениях в области ООБД. Системы ООБД позволяют пользователям определять абстракции; облегчают проектирование некоторых связей; устраняют потребность в определяемых пользователями ключах; поддерживают новый набор предикатов сравнения; в некоторых случаях устраняют потребность в соединениях; в некоторых ситуациях обеспечивают более высокую производительность, нежели системы, основанные на реляционной модели; обеспечивают поддержку версий и длительных транзакций. Наконец, разработана объектная алгебра – хотя, возможно, пока и не столь детально, как реляционная алгебра.

Недостатки модели ООБД

Ожидалось, что объектно-ориентированные методы позволят произвести революцию в области технологии баз данных. Однако, несмотря на указанные выше достижения, ООБД так и не смогли оказать значительного влияния на положение дел в этой области. И в модели, и в технологии ООБД до сих пор сохраняются слабые места.

В объектно-ориентированных базах данных отсутствуют базовые средства, к которым пользователи систем баз данных привыкли и поэтому ожидают их видеть. Среди прочего, можно отметить: отсутствие интероперабельности между РБД и ООБД; минимальную оптимизацию запросов; отсутствие стандартной алгебры запросов; отсутствие средств обеспечения запросов; отсутствие поддержки представлений; проблемы с безопасностью; отсутствие поддержки динамических изменений определений классов; ограниченная поддержка декларативных ограничений целостности; ограниченные возможности настройки производительности; недостаточная поддержка сложных объектов; ограниченная интеграция с существующими объектно-ориентированными системами программирования; ограниченный выигрыш в производительности.

Объектно-реляционная модель данных

У большинства специалистов сложилось двойственное отношение к ОО-технологии. С одной стороны, все видят ее ценность для проектирования логики приложений. С другой стороны, многие убеждены в существовании веских причин для того, чтобы продолжать использовать нормализованные декомпозированные структуры данных, которые являются признаком качественной разработки реляционной схемы БД.

Между ООСУБД и РСУБД существует ряд фундаментальных и явно непримиримых противоречий. Основное противоречие между объектным и реляционным мирами связывается с декомпозиционной природой хранения информации в нормализованной реляционной схеме.

Для сторонника объектов декомпозиция объекта плоха по определению. Они чаще всего используют следующую аналогию: «Ставя машину в гараж, вы паркуете ее как объект, который уже собран и готов к повторному использованию. Вы же не разбираете машину на части, маркируя каждую часть и расставляя их на полках только для того, чтобы вновь собрать их на следующее утро».

Если вы (как и большинство специалистов по БД) чувствуете, что есть еще «порох в пороховницах» нормализованных реляционных представлений, и все-таки стремитесь выполнять декомпозицию своих объектов для хранения даже в мире, который все больше и больше завоевывается объектно-ориентированными методами разработки приложений,

то можете задать сторонникуобъектных технологий встречный вопрос: «Будете ли вы хранить свою одежду комплектно, сшив предметы вместе (например, прикрепив носки намертво к туфлям), или же поставите обувь в одно место, носки положите в другое, а галстуки повесите в третье?» Сторонник объектов может посмотреть на вас, как на сумасшедшего, и спросить, какое же это имеет отношение к предмету спора. (Ответ, который лучше не произносить, звучит примерно так: «Такое же самое, что и ваш пример, то есть – никакого».)

Наряду с революционным (все или ничего) подходом к ориентации на объекты, представленным сторонниками объектно-ориентированной модели данных существует другой, эволюционный подход, который предлагается в объектно-реляционной модели данных.

В ООБД возможны несколько способов выделения объектов. В результате получаются перекрывающиеся комплексные объекты, ориентированные на разные задачи, вместо одной декомпозированной схемы, пригодной для любых приложений. Не отвергая каноны реляционной модели **объектно-реляционная модель** предлагает объектные расширения как средство преобразования реляционного представления в объектно-ориентированное. Эти дополнительные возможности превратились в важный механизм построения двухсторонних мостов между объектно-ориентированными приложениями и реляционными БД.

По такому принципу развивают свои продукты главные действующие лица на рынке СУБД – компании Oracle, IBM, Microsoft, Borland, Informix.

В настоящее время объектно-реляционные СУБД (ОРСУБД) пока не могут похвастаться всеобъемлющими эффективными средствами поддержки объектов в истинно объектно-ориентированном стиле. Они вряд ли удовлетворят объектно-ориентированных пуристов, но разве это так важно? Все они построены на базе реляционной модели. Любой шаг в сторону объектной модели будет медленной метаморфозой и не помешает пользователям, которые рады остаться в реляционном мире. Если мы в лице ОРСУБД получим механизм одноуровневой виртуальной памяти с подклассами, наследованием, динамическим связыванием, полиморфизмом, способностью описывать декларативные ограничения целостности и многим другим, то будем очень заинтересованы в использовании этих возможностей в проектах.

Преимущества ОРСУБД

Основным преимуществом расширенной реляционной модели данных (каковой часто считают объектно-реляционную модель) является повторное и совместное использование компонентов. Повторное использование основано на возможности дополнения сервера ОРСУБД стандартными функциями, которые выполняются централизованно и не входят в виде исполняемого кода в состав каждого приложения. Например, в приложениях может понадобиться использовать данные пространственного типа, представляющие точки, прямые и многоугольники с соответствующими функциями, которые вычисляют расстояние между точками, расстояние между точкой и прямой, проверяют, находится ли точка в пределах многоугольника и перекрываются ли две многоугольные области, и т.д. Если реализовать эти функции на сервере, то можно избежать необходимости их определения в каждом приложении и совместно использовать данные функции всеми приложениями. Эти преимущества позволяют также повысить производительность труда как разработчиков, так и конечных пользователей.

Другое очевидное преимущество заключается в том, что расширенный реляционный подход позволяет воспользоваться обширным объемом накопленных знаний и опыта, связанных с разработкой реляционных приложений. Это очень важное достоинство, поскольку многие организации не хотели бы тратить средства на достаточно дорогостоящий переход к системе принципиально нового типа. При правильном проектировании с учетом новых функциональных возможностей подобный подход позволяет организациям воспользоваться преимуществами новых дополнений

эволюционным путем без утраты преимуществ, получаемых от использования компонентов и функций уже существующей базы данных. Это дает возможность осуществить постепенный переход к использованию ОРСУБД, начиная с разработки некоторых пробных и экспериментальных проектов. Новый стандарт SQL3 (включающий объектные расширения) разработан с учетом обратной совместимости со стандартом SQL2, поэтому такой плавный переход должны обеспечить любые ОРСУБД, совместимые с SQL3.

Недостатки ОРСУБД

Очевидными недостатками подхода с использованием ОРСУБД являются сложность и связанные с ней дополнительные расходы. Более того, некоторые сторонники реляционного подхода уверены, что простота и ясность, присущая реляционной модели, при использовании расширений подобных типов утрачивается. Другие утверждают, что расширения реляционной СУБД предназначены для незначительного количества приложений, причем в последних еще не достигнута оптимальная производительность даже в рамках современной реляционной технологии.

С другой стороны, подобные дополнения не нравятся также ревностным сторонникам объектно-ориентированного подхода. Они утверждают, что в объектно-реляционных системах искажается объектная терминология. Например, вместо объектной модели в объектно-реляционной технологии используется понятие определяемых пользователем типов данных, тогда как объектно-ориентированный подход имеет дело с абстрактными типами, иерархией классов и объектной моделью. Но компании-поставщики ОРСУБД пытаются представить объектные модели как дополнения к реляционной модели, что приводит к появлению некоторых сложностей. При этом может быть потерян смысл объектно-ориентированного подхода, что и обнаруживается в виде большого семантического разрыва между этими двумя технологиями. Объектные приложения не в такой степени ориентированы на структуру данных, как реляционные приложения. Кроме того, в объектно-ориентированных моделях и приложениях для более адекватного отражения объектов реального мира обеспечивается тесное взаимодействие инкапсулированных объектов и связей. Это приводит к созданию более широкого набора связей, чем тот, который возможно выразить с помощью средств языка SQL, а также к включению в определения объектов функциональных программ. Объекты фактически не являются очередным расширением понятия данных, поскольку представляют совершенно другую концепцию, с большим потенциалом выражения связей и правил поведения объектов реального мира.

Основными требованиями к языку баз данных являются максимальная простота использования с точки зрения пользователей, а также относительно простая структура и синтаксис команд. Исходный стандарт языка SQL, выпущенный в 1989 году, по-видимому, удовлетворял этим требованиям. Версия стандарта языка SQL, которая была выпущена в 1992 году, возросла в объеме от 120 примерно до 600 страниц, и поэтому стало труднее ответить на вопрос о том, удовлетворяет ли он этим требованиям или нет. К сожалению, объем нового стандарта SQL3 (включающий объектные расширения) стал еще более впечатляющим, и, похоже, что эти два требования организациями по стандартизации не только не удовлетворяются, но, скорее всего, даже не рассматриваются.

Война манифестов

Реляционная технология всегда подвергалась нападкам. Ее критикуют со дня ее рождения. Первые несколько лет основным возражением было то, что это была «всего лишь теория» – было «совершенно невозможно» сконструировать коммерческую систему на реляционных принципах. Затем, когда первые реляционные продукты вышли на рынок, они считались слишком медленными для построения систем OLTP. Теперь, когда реляционные продукты доказали, что они могут поддерживать тысячи пользователей

многогигабайтных баз данных, самые последние претензии состоят в том, что реляционная технология не соответствует новым требованиям приложений. В будущих приложениях требуется хранить так называемые неструктурированные данные (изображение, звук, свободный текст) и выполнять сложные операции, такие как сканирование изображений или сжатие звука. Вопрос в следующем: «Действительно ли состоятельна эта критика? Другими словами, ждет ли реляционную технологию ее Ватерлоо? Действительно ли она не соответствует требованиям новых приложений?»

Некоторые люди полагают, что нужен новый тип систем баз данных, основанный не на реляционных, а на объектно-ориентированных понятиях (таких, как наследование, классы, полиморфизм и так далее). Поэтому объектно-ориентированным СУБД в последнее время уделяется так много внимания. Однако у коммерческих ОО-систем был (и остается) такой недостаток, как отсутствие общей терминологии. Отчасти по этой причине в 1989 году был написан манифест, названный *Манифестом объектно-ориентированных систем баз данных (The Object-Oriented Database System Manifesto)*. В этом манифесте перечислялись правила, которым должна подчиняться система, чтобы она могла называться объектно-ориентированной СУБД. Он был призван внести некоторый порядок и структуру в запутанный мир ООСУБД. Поскольку у всех авторов имелся предварительный опыт работы с ОО-системами, они полагали, что в будущем базам данных требуются не реляционные, а ОО-основы.

Годом позже, в 1990 году, был опубликован второй манифест, названный *Манифестом систем баз данных третьего поколения (Third-Generation Database System Manifesto)*. В чем-то этот документ являлся реакцией на первый манифест. Он был написан другой группой экспертов по базам данных, в основном, людьми с реляционной предысторией, в том числе Майклом Стоунбрейкером (работавшим в то время в Ingres) и Джеймсом Греем (работавшим в DEC). По мнению этих авторов, не нужно было никакой новой модели – расширение реляционной модели новыми объектными понятиями и механизмами могло решить задачу. Они намеревались определить набор требуемых характеристик для всех будущих СУБД. Эти характеристики (определяемые посредством набора догматов и предложений) основывались на новых требованиях приложений и бизнеса.

Заметим, что ни в одном из первых двух манифестов не предпринималась попытка определить новую модель или новый язык; в них просто перечислялись требования, или другими словами, минимальные характеристики, которым (по мнению авторов) действительно должны были обладать СУБД.

В Третьем Манифесте (*The Third Manifesto*) другая точка зрения. В нем делается попытка дать достаточно формальное и полное определение новой версии реляционной модели. Эта новая версия, которую авторы (Хью Дарвен и Крис Дейт) называют D, по-прежнему совместима с классической реляционной моделью, первоначально определенной доктором Э. Ф. Коддом. Тем не менее, авторы полагают, что посредством более точного определения некоторых понятий, в особенности доменов, они могут показать, что реляционная модель соответствует упомянутым выше новым требованиям приложений.

Заключение

Как видим СУБД-ориентированные модели данных первого поколения практически вышли из употребления, а модели нового поколения пока находятся в стадии становления и с практической точки зрения еще не достигли достаточного уровня зрелости. Поэтому мы не можем уделить им больше внимания в нашем, ориентированном в основном на практически полезные инструменты курсе. В этой главе мы ограничимся детальным рассмотрением лишь широко распространенной и заслуженно признанной мировым сообществом реляционной модели.

Вопросы и задания к параграфу 4.1

1. Какие поколения СУБД традиционно выделяют в технологии БД? Какие модели данных относятся к каждому из этих поколений СУБД?

2. Охарактеризуйте каждую из СУБД-ориентированных моделей данных. Укажите их достоинства и недостатки.

3. Попытайтесь опровергнуть позицию автора, который отдает на сегодняшний момент пальму первенства в классе СУБД-ориентированных моделей данных реляционной модели.