I. ТЕХНОЛОГИИ ОБРАБОТКИ ДАННЫХ

Революция 2005 года в реляционных базах данных

В. Л. Арлазаров, Н. Е. Емельянов

Институт системного анализа Российской академии наук,
Россия, 117312 Москва, пр. 60-летия Октября, 9

Рассматриваются достоинства и недостатки реляционных баз данных (РБД), обоснованно появление нового типа данных — XML data type, зачеркнувшего главный принцип РБД — атомарность данных. Обсуждаются вслед за Коддом альтернативные пути развития РБД.

Что же произошло в 2005 году? В СУБД MS SQL Server-2005 появился тип данных «XML data type», т. е. фирма Майкрософт разрешила хранить в одной ячейке таблицы сколь угодно сложные структуры данных, вплоть до всей базы данных (БД). Чтобы охарактеризовать революционность этого события, вспомним кратко историю развития реляционных СУБД (РБД).

Кодд опубликовал первую работу по РБД в 1970 году, а в 1981 году появилась первая промышленная реализация — SQL/DS. В 2006 году во всем мире отпраздновали 25-летие РБД. Математик Кодд ввел строгие постулаты, определяющие суть реляционных БД. Первый постулат РБД: БД — совокупность изменяющихся во времени отношений (relation) — таблиц, все ячейки которых атомарны (неделимы). В течение более 30 лет изучаются нормальные формы РБД, предназначенные для эффективного описания информационных объектов. Первая нормальная форма (НФ) требует выполнения названного постулата, все следующие НФ от 2-й до 19-й требуют выполнения свойств всех предыдущих нормальных форм и каких-то дополнительных свойств. То есть все РБД должны выполнять

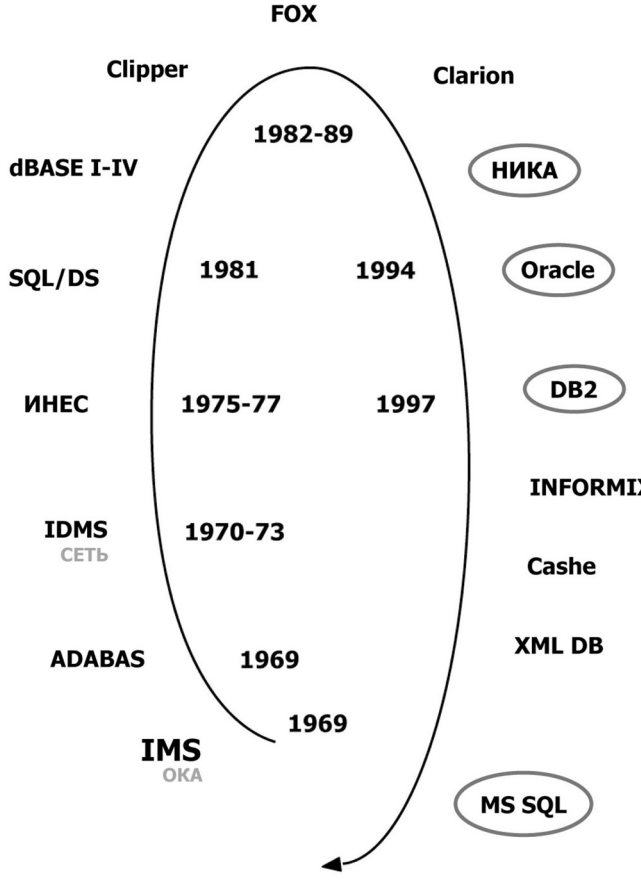

Рис. 1. СУБД и годы их создания

первый постулат — атомарности данных во всех ячейках всех отношений. Этому учили и учат все профессора по тысячам учебников во всех университетах мира.

Таким образом, Майкрософт, можно сказать, перечеркнула, не дожидаясь двадцатипятилетнего юбилея, основу РБД. Тут же, вслед за Майкрософт, все разработчики РСУБД, включая ORACLE и IBM с DB2, боясь отстать в конкурентной борьбе, тоже поспешили ввести такие спецификации [1].

Все развивается по спирали. В информационной системе, созданной по проекту РФФИ в 1996–1998 годах, у нас фиксировано более 1000 СУБД [2]. На рис. 1 перечислены некоторые наиболее значительные продукты с годами их появления. Овалами отмечены те из них, которые дожили до наших дней.

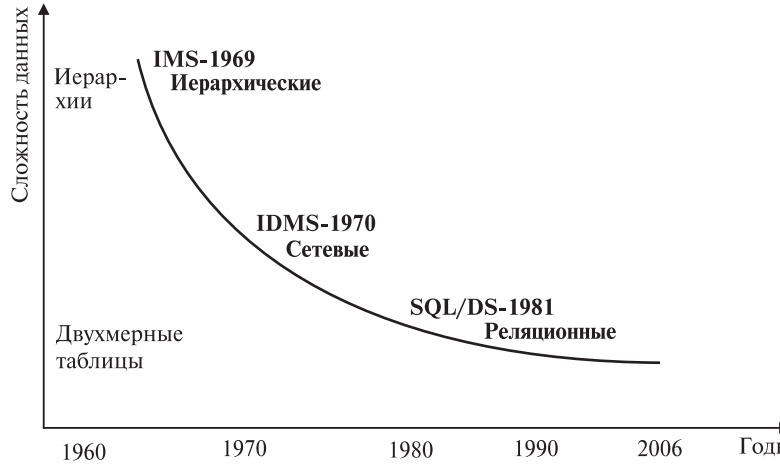

Рис. 2. Упрощения структур данных в 1970–2005 годах

Первые промышленные СУБД — IMS, поддерживающая иерархическую модель данных, и IDMS, основанная на сетевой модели, — имели очень низкоуровневые интерфейсы как при описании данных, так и при манипулировании. Программирование в среде этих СУБД было чрезвычайно утомительным. Поэтому появление реляционной модели, изначально предполагавшей достаточно высокий уровень абстрагирования от физической организации данных и дающей хорошую основу для построения языка манипулирования высокого уровня, было мощным толчком к построению СУБД следующего поколения. Появление стандарта SQL, базирующегося на реляционной модели, закрепило ее ведущее положение.

Таким образом, можно сказать, что первые полтора десятилетия развития СУБД прошли под знаком упрощения модели данных и повышения уровня языка манипулирования. На рис. 2 приведен условный график, который показывает по годам упрощение структур данных.

Достоинства реляционной модели, состоящие в простоте ее элементарных структур — таблиц (отношений), и возможность вследствие этого построить математически прозрачный метод манипулирования — реляционную алгебру, нашедший свое воплощение в языке SQL, многократно описаны. Менее известны хронические дефекты этой модели. Главный из них состоит в том, что основные элементы структур — строки (кортежи) — независимы и должны иметь уникальный идентификатор — первичный ключ. Именно по этому ключу данный элемент может быть связан с другими элементами базы данных. Обычно учебники по РБД приводят примеры, когда этот ключ содержится среди содержательных элементов строки

(уникальное название, номер паспорта и т. п.). Разумеется, это далеко не всегда так. Приведем несколько простых примеров.

Пусть имеются комнаты, а в них стоят столы, каждый из которых обладает своими размерами. Если столы имеют идентификаторы (к примеру, инвентарные номера, как во многих учреждениях), то мы легко построим базу данных из одного отношения, в которой перенос стола в другую комнату отобразится простейшей операцией замены номера комнаты в строке. А если инвентарного номера нет?

Поскольку в комнате может стоять несколько одинаковых столов, нам придется ввести в строку новый элемент-счетчик «количество столов данного типа», и перенос стола из комнаты в комнату отобразиться в целую программу, примерно такую:

1. Найти в данной комнате стол данного типа (кортеж).
2. Если счетчик столов в кортеже больше 1, уменьшить его на 1.
3. Если счетчик столов равен единице, вычеркнуть кортеж.
4. Найти во второй комнате стол данного типа.
5. Если он есть, добавить к соответствующему счетчику 1.
6. Если такого стола нет, добавить новый кортеж.

Нравится?

Другой пример. Мы хотим построить генеалогическое древо, отражающее связи между отцами и детьми. Так как любые характеристики имеющихся в базе людей могут повторяться, то для описания связи нам обязательно придется придумать и вводить уникальный идентификатор каждого человека. Но даже в этом случае простейший запрос «выдать всех предков данного персонажа» выльется в циклический поиск по идентификаторам.

Рассмотрим еще один близкий по смыслу пример. Пусть есть несколько объектов, структура информации о которых очень близка, но имеются некоторые отличия (республики и области, самостоятельные отделы, управления и службы и т. п.). В реальности подавляющее число действий с этими объектами в БД одинаковы. Спрашивается, как построить концептуальную схему? Можно выбрать один из нескольких путей. Мы можем построить отдельное отношение для каждого типа объектов. Но тогда мы вынуждены будем обеспечивать совместную отработку их на уровне операторов во время исполнения.

Мы можем построить одно отношение, включив в него все атрибуты всех объектов. Однако создание отношений, большая часть атрибутов которых всегда пуста, считается плохим стилем, и не зря. Наконец, мы можем «отселить» различающиеся атрибуты в отдельные отношения, снабдив их ключами основного отношения для установления связей. Странноватое решение, но почему бы нет?

Во всех рассмотренных примерах проблемы присутствуют именно в реляционных структурах данных. В любой объектной (иерархической, сетевой) базе данных никаких вопросов даже не возникнет. Данные там определяются местоположением в общей структуре и ни в какой дополнительной идентификации не нуждаются.

Конечно, ничего смертельного для реляционных СУБД в приведенных примерах нет. Первый из них легко покрывается введением скрытых от пользователя идентификаторов, имеющихся во многих системах, последний — механизмом внешних схем представлений, играющим вообще важную роль в современных системах баз данных. Пожалуй, лишь второй по-настоящему неприятен.

Мы привели эти примеры только для того, чтобы показать, что простота реляционных структур дается вовсе не бесплатно и приводит к появлению абсолютно бессодержательных атрибутов, а то и целых отношений, делающих концептуальную схему еще более непрозрачной, нуждающейся в поддержке и усложняющих манипулирование.

Однако настоящий бич реляционных баз данных — чрезвычайная сложность связей между отношениями, вытекающая непосредственно из примитивности основных структур. Даже простейшая прикладная база данных содержит 15–20 таблиц — отношений. Если же система более серьезная, число отношений резко возрастает, их становится много десятков. Это еще терпимо. Однако с ростом скоростей и объемов внешней памяти компьютеров растет и сложность решаемых задач. Теперь концептуальная схема интегрированной информационной системы содержит сотни, а то и тысячи таблиц.

Уложенное на бумагу описание данных такой системы занимает лист формата A1, причем каждое отношение на нем едва можно разглядеть. Уследить за стрелками, обозначающими связи между отношениями, практически невозможно. Появились мощнейшие средства проектирования баз данных, позволяющие автоматически формировать ее схему на основе описания бизнес-процессов и содержащейся в них информации. Но и это не помогает. Многие создатели крупных интегрированных систем (например, ERP) дают пользователям доступ только к части данных, имеющихся в системе, подрывая один из основных лозунгов отцов-основателей реляционных баз данных: пользователь может сам, без помощи программистов манипулировать своими данными.

Давайте проведем аналогию с программированием. Можете ли вы себе представить, чтобы в программе, написанной на любом из современных языков, описания всех переменных, функций и объектов были вынесены на верхний уровень и все связи между ними нарисованы в виде стрелок? Это будет безграмотность и нонсенс. Человечество придумало модульность, локализацию областей действий переменных и описаний, а

затем и объектное программирование ровно для того, чтобы каждый информационный объект, будь то элементарное данное, структура или массив могли (и должны были) существовать в некоторой вложенной друг в друга системе объектов, а не в абстрактной свалке себе подобных.

Между тем в РБД не существует никаких локализаций, и мы вынуждены рассматривать все таблицы как потенциально связанные между собой, что чрезвычайно сложно, несмотря ни на какие внешние схемы. Поэтому уже начиная со второй половины 1980-х годов делались попытки построить на основе реляционной модели «объектные» базы данных. Появился даже термин «реляционно-объектные СУБД». Хотя таких попыток за последние 20 лет сделано немало и некоторые из них очень интересны, большого успеха они не принесли.

Новый мощный удар по пуристам реляционных БД был нанесен в конце 1990-х годов быстрым развитием компьютерных сетей и появлением языка XML. Компьютерные сети требуют развитого аппарата обмена данными и, соответственно, форматов обмена. Чем же должны обмениваться компьютеры? Таблицами? Очевидно, нет. Ясно, что обмен должен осуществляться некими осмысленными единицами — документами. На роль языка описания содержания документов и претендовал XML, скоро ставший стандартом.

Довольно быстро стало понятно, что XML может использоваться не только для связи между машинами, но и между программами внутри одной машины. В его формат удобно переводить данные при вводе, выводе, в системах документооборота и т. п. Появились XML-хранилища, парсеры и другие средства обработки. И вот закономерный итог.

Рассмотрим основные положения сравнения реляционной модели данных и XML, проведенное сотрудниками MicroSoft [3]:

- Если данные хорошо структурированы по известной схеме, то РБД хорошо работает.
- Если же данные полу(слабо)структурированы, или вовсе не структурированы, или структура не известна, кроме того, если вы добиваетесь мобильности системы (или данных), то использование XML-модели — хороший выбор.
- XML хорош также, если**:**
  - данные разрежены;
  - структура данных может существенно изменяться;
  - возможна рекурсия в описании данных;
  - порядок принципиален для данных;
  - вы хотите запрашивать или обновлять данные на основе их структуры.
- Если нет ни одного из этих условий, то можно (но не обязательно) использовать РБД.

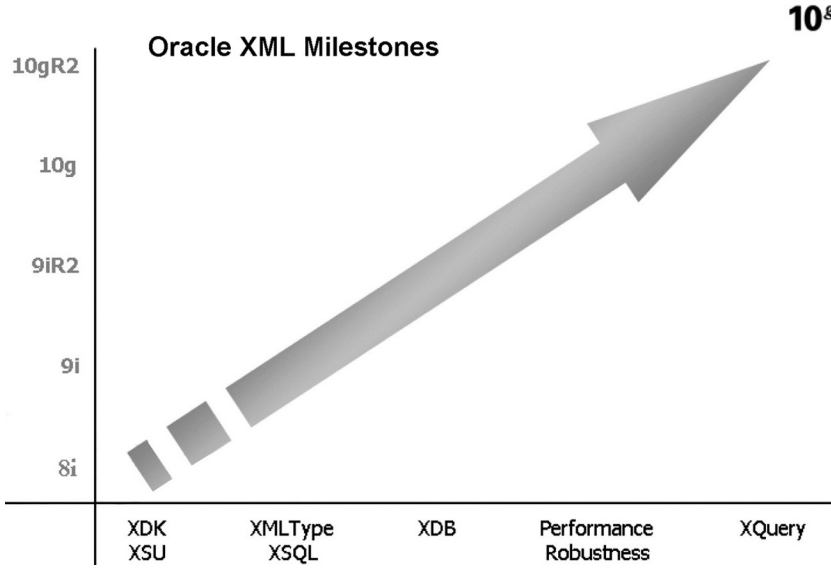

Рис. 3. Планы ORACLE по наращиванию сложности структур данных начиная с 2005 года. По вертикальной оси — версии ORACLE, последняя — 2009 год. По горизонтальной оси — возможности работы со сложноструктурированными данными

Такие выводы потребовали революционной модернизации, в частности введения нового типа данных XML data type. На рис. 3 представлены планы ORACLE [4] по наращиванию сложности структур данных начиная с 2005 года.

Если у этого графика поменять оси, то можно увидеть, как наращивалась сложность структур данных от версии к версии СУБД ORACLE в 2005–2009 годах. Совместив тенденции на рис. 2 и 3, получаем график упрощения структур данных начиная с 1980 года с последующим быстрым всплеском сложности структур в 2005–2009 годах (рис. 4).

Надо сказать, что СУБД, в которых структуры данных ориентированы на непосредственное описание объекта, а не на упрощение самих себя, активно развивались все годы. Достаточно вспомнить одну из крупнейших софтверных разработок советского времени СУБД ИНЕС, которая в 1980-х годах функционировала на каждой третьей машине ЕС ЭВМ. Из более поздних работ отметим СУБД Cache с ярко выраженной объектной ориентацией и отечественную СУБД НИКА, позволяющую эффективно работать с данными любой сложности.

В СУБД ИНЕС [5] и НИКА [6] использовался для этих целей разработанный в конце 1970-х годов язык DCM (от слова document), опередив-

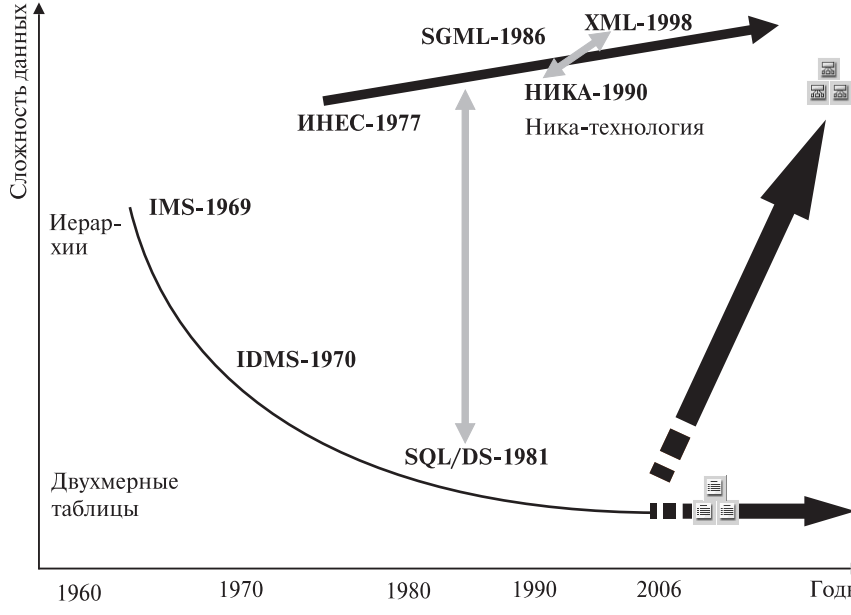

Рис. 4. Наращивание возможности работы со сложными структурами данных ORACLE (стрелка вверх), нарушило установку работать только с атомарными данными (горизонтальная стрелка)

ший SGML и его подмножества HTML и XML, имеющий много общего с ними, см. рис. 4.

СУБД НИКА интересна тем, что ее структура данных практически совпадает со структурой данных XML. Ее отличает оригинальное построение поисковых индексов, а главное — сквозная система работы с описанием объекта при помощи форм, служащих генерации одновременно схемы базы данных, программ ввода данных, запроса и вывода. В то же время СУБД НИКА обеспечивает быстрый доступ как ко всему XML-документу, так и ко всем его подобъектам, вплоть до отдельных реквизитов. Тем самым не нужно дублировать информацию и выигрывается объем памяти, не проигрывая во времени чтения и записи. И хотя в СУБД НИКА, а лучше сказать, в НИКА-технологии [7] многого не хватает, чтобы ее можно было рассматривать как современную полнофункциональную СУБД, она является прекрасным макетом, или «промышленным прототипом», СУБД, непосредственно ориентированной на XML [8].

Конечно, невозможно игнорировать миллионы программистов и пользователей, освоивших SQL, и реальные разработки, имеющиеся в сотнях тысяч организаций на базе, прежде всего, таких систем, как ORACLE

и MS SQL. Однако введение в реляционную СУБД нового типа данных «XML data type» — не лучшее решение проблем работы со сложно структурированными документами. Параллельное согласованное ведение больших объемов данных вызывает много дополнительных осложнений и должно быть изжито.

Рассмотрим аргументы, которые представил сам Кодд в статье про «Великое сражение», рассматривая возможные альтернативы будущего систем баз данных. В изложении Дейта это выглядит так [9].

1. Если мы добавляем операции высокого уровня (соединение и т. д.), то получаем автоматическую навигацию, значит, даже непрограммисты смогут достичь своих целей без чьей-либо помощи.
2. Наоборот, если мы не добавляем такие операции, а добавляем новые структуры данных, такие как связи, то непосредственная навигация становится необходимой, значит, для достижения целей конечных пользователей необходимы программисты.
3. Если мы добавляем и операции, и структуры, то получаем ненужную сложность, значит, программистам и администратору базы данных придется принимать намного больше решений (и, следует добавить, при отсутствии каких-либо хороших руководств о том, как принимать такие решения).

С 2005 года развитие РСУБД пошло по третьему пути, самому абсурдному, с точки зрения Кодда. В учебниках по РСУБД часто говорится, что все документы, с которыми работают люди, — плоские таблицы, и поэтому реляционные отношения являются идеальной моделью данных. Что документы плоские, это, безусловно, верно, так как они представляются на листе бумаги. Но в то же время надо понимать, что нет ни одного реального документа, равного реляционному отношению, эта абстракция в своем чистом виде на практике никогда не встречается. Заголовки таблиц, итоги, промежуточные заголовки, повторяющиеся строки и многое другое не укладываются в реляционные отношения.

Справедливо другое утверждение: любой встречающийся на практике документ можно записать на языке XML. Правильной представляется ориентация не на записи в реляционных двухмерных таблицах, а на деревья в XML-документах и операции высокого уровня (соединение и т. д. с операндами деревьями), такие чтобы и непрограммисты смогли достичь своих целей, как правило, без непосредственной навигации. Должна появиться новая модель данных, объединяющая достоинства реляционной БД и XML DB.

К трем путям развития систем БД, рассмотренным Коддом, можно добавить четвертый: если мы добавляем, сохраняя «спартанскую простоту», новые структуры данных (XML-документы) и одновременно вводим

операции высокого уровня, которые обеспечивают навигацию и автоматическую обработку таких данных (практически всех входных и выходных документов), то разнообразные пользователи, от новичков до самых опытных, могут взаимодействовать с данными на основе общей модели, и даже непрограммисты могут достичь своих целей без чьей-либо помощи.

Литература

1. [Электронные ресурсы]:
   http://www.citforum.ru/database/oracle/xml-oracle;
   http://www.rdtex.ru/win/courses.overview?ccode = O10gXSPL;
   http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic = /com.ibm.db2.udb.apdv.embed.doc/doc/c0023366.htm;
   http://www.vldb2005.org/program/paper/thu/p1164-nicola.pdf и многие др.
2. Проект РФФИ 96–07–89394. См.: http://sr.isa.ru
3. XML Best Practices. MS, 2006. msdn2.microsoft.com
4. http://www.slideshare.net/MGralike/ogh-ace-case-part-1-and-2-oracle-xml-database-marco-gralike
5. *Емельянов Н.* Введение в СУБД ИНЕС. М.: Наука, 1988. С. 256
6. *Годунов А.*, *Емельянов Н.*, *Космынин А.*, *Солдатов В.* Система НИКА // В кн.: Системы управления базами данных и знаний / М.: Финансы и статистика, 1991. С. 209–248
7. *Богданов А. С.*, *Емельянов Н. Е.*, *Ерохин В. И.*, *Скорняков В. А.*, *Романов Б. Л.* НИКА-технология построения информационных систем // Организационное управление и искусственный интеллект: Сб. трудов ИСА РАН / Под ред. чл.-корр. РАН В. Л. Арлазарова и д. т. н. проф. Н. Е. Емельянова. М.: URSS, 2003. С. 52–67.
8. См., например: *Акимова Г. П.*, *Богданов Д. С.*, *Босов А. В.*, *Даниленко А. Ю.*, *Ерохин В. И.*, *Корольков Г. В.* Реализация защищенного хранилища данных и электронного документооборота при интегрированной аналитической обработке разнородной информации // Системы высокой доступности. Радиотехника. 2007. Т. 3. № 4. С. 33–42.
9. *Date C. J.* The relational model will stand the test of time // Intelligent Enterprise. June 1, 1999. Vol. 2. № 8 (www.intelligententerprise.com/992206/online1.shtml — текст на английском, www.citforum.ru/database/digest/codd_3.shtml — перевод на русский).
10. *Codd E. F.* Recent Investigations into Relational Data Base Systems. IBM Research Report RJ1385. 1974. Republished in Proc. 1974 Congress (Stockholm, 1974). New York, N. Y.: North-Holland, 1974.