ДМИТРИЙ ОСИПОВ

bhv®

Базы данных и Delphi

Теория и практика

+ ПРОБНЫЕ ВЕРСИИ ПО

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

- ВВЕДЕНИЕ В ТЕОРИЮ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ
- ПРОЕКТИРОВАНИЕ ЛОКАЛЬНЫХ И КЛИЕНТ-СЕРВЕРНЫХ БАЗ ДАННЫХ
- СТРУКТУРИРОВАННЫЙ ЯЗЫК ЗАПРОСОВ SQL
- РАСШИРЯЕМЫЙ ЯЗЫК РАЗМЕТКИ XML
- ТЕХНОЛОГИИ DBEXPRESS, INTERBASE EXPRESS И ADO
- МНОГОУРОВНЕВЫЕ БАЗЫ ДАННЫХ НА ОСНОВЕ DATASNAP
- НЕСТАНДАРТНЫЕ ПРИЕМЫ РАЗРАБОТКИ ПРИЛОЖЕНИЙ БАЗ ДАННЫХ

Дмитрий Осипов

Базы данных и Delphi

Теория и практика

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.06
ББК 32.973.26-018.2
О-74

Осипов Д. Л.

О-74 Базы данных и Delphi. Теория и практика. — СПб.: БХВ-Петербург, 2011. — 752 с.: ил. + DVD — (Профессиональное программирование)

ISBN 978-5-9775-0659-5

Книга основана на материалах лекций и практических занятий, разработанных автором, и объединяет теоретические основы и практические аспекты разработки реляционных баз данных. В первой части рассмотрена концепция реляционных баз данных: реляционная модель данных, жизненный цикл информационной системы, концептуальное и логическое моделирование БД, нормализация отношений, обеспечение многопользовательского доступа к данным, вопросы обеспечения безопасности БД, языки SQL и XML и др. Во второй части описаны возможности современных версий Delphi в области разработки приложений баз данных: подробное описание технологий dbExpress, Interbase Express и ADO, особенности использования компонентов управления данными визуальной библиотеки Delphi, механизм разработки многоуровневых приложений на основе технологии DataSnap, порядок создания отчетов для печати и многое другое. На DVD размещены дополнительные главы, а также материалы и пробные версии ПО компании Embarcadero, включая Delphi XE.

Для студентов и программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Леонид Кочин*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.01.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 60,63.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию № 77.99.60.953.Д.005770.05.09
от 26.05.2009 г. выдано Федеральной службой по надзору
в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0659-5

Оглавление

МАТЕРИАЛ НА ДИСКЕ

Генератор отчетов своими руками

Диаграммы и графики

ОТЧЕТЫ В MICROSOFT OFFICE

Введение

Интересно, что послужило первопричиной столь страстного желания человека систематизировать большое количество информации? Быть может, толчком стала необходимость строгого учета сбитых птеродактилей или поверженных динозавров? Почему бы и нет? Коротая время между схватками с саблезубыми тиграми и себе подобными, первобытный человек кроме чувства голода, возможно, испытывал еще одно — тягу к систематизации и учету. Куда деваться, когда первобытная супруга постоянно твердила на ухо: "А у Петровых из 35-й пещеры на ужин филе из звероящера". Не верите? Доказательством тому служат многочисленные наскальные рисунки, до сих пор находимые на стоянках наших пращуров. Несчастные мужчины еще не знали систем счисления, посему учет им приходилось вести в графической форме. Поймал динозавра — нарисовал, поймал второго — нарисовал еще одну картинку. Не поймал ни одного — нарисовал трех… В итоге по наскальным пиктограммам можно выяснить, сколько было съедено антилоп, бизонов и упомянутых динозавров за отчетный период.

Минуло несколько тысячелетий. Мамонты в страхе попасть на ужин к нашим предкам бежали на север, а человек перебрался из уютных, хорошо проветриваемых пещер в каменные дома древнего Вавилона. Первая же попытка вести учет по старинке столкнулась с противодействием лучшей половины человечества. Только представьте себе, как по улицам ночного Вавилона жены гоняли скалками первых разработчиков БД, попытавшихся по "пещерной" привычке начертать полугодовой отчет (в виде стада овечек) на недавно побеленной стене. Загнанным в угол мужчинам не оставалось больше ничего, кроме как вновь двинуть вперед научный прогресс. В результате была изобретена первая система счисления, а записи отныне стали наносить на каменные скрижали. Сами понимаете, что камни ворочать довольно тяжело, зато наверняка все программисты тех времен были в прекрасной физической форме. Представьте себе вавилонскую картотеку на пару-другую тысяч каменных плит, среди которых вы второй месяц ищете приходный ордер от 2000 года до рождества Христова! Хочешь, не хочешь, а мускулы станут как сталь. Впрочем, ни стали, ни, тем более физической культуры, в то время еще не знали, но прототип первой базы данных уже сформировали. Таким образом, предлагаю считать, что профессия проектировщика баз данных входит в десятку древнейших профессий… Ну да ладно, мы немного отвлеклись.

Практически рядом с древним Вавилоном (как в пространстве, так и во времени) над вопросами учета и контроля бились египтяне. Их система хранения данных базировалась на папирусах и росписях пирамид. Такого размаха не достигала еще ни одна из цивилизаций. Древнеегипетских наработок хватило вплоть до нашего времени. И по сей день, археологи пытаются восстановить целостность данных, разыскивая утраченные богатства и дешифруя иероглифы.

Небывалых успехов достигли математики из далекой Индии, примерно два тысячелетия назад они придумали цифру ноль. Ноль — не только число в современной системе счисления, но и предвестник краеугольного для БД понятия — пустоты NULL. Без NULL невозможно корректно представить данные ни в одной из современных систем управления базами данных. Я докажу вам это на страницах книги, которую вы держите в руках.

Мощнейшие цивилизации Европы и Азии несколько тысячелетий развивали системы хранения и представления данных. До сих пор ходят легенды об одном из самых великих хранилищ древности — Александрийской библиотеке. Несколько веков ее полки пополнялись новой информацией, пока бóльшая часть книг не погибла в пожаре войны между Клеопатрой и ее братом Птолемеем XIII. Если бы тогда существовало понятие создания резервной копии данных! Человечество бы не утратило бесценные знания.

Годы сменялись десятилетиями, десятилетия складывались в века. Человечество изобрело печатный станок, покорило просторы океана, научилось парить в небесах, побывало на Луне, и лишь потом открыло величайшую тайну — научилось хранить информацию в базах данных.

На сегодняшний день не осталось ни одной области знаний, где не применялись бы компьютерные базы данных (БД). Наука, образование, экономика, электронная коммерция, медицина, статистика, военное дело, — список можно продолжать до бесконечности. Базы данных сопровождают человека на протяжении всей жизни. Появление на свет ребенка, учеба в школе и вузе, получение паспорта или водительских прав, посещение врача, покупки в магазинах, поиск книги в библиотеке, открытие счета в банке, в современном информационном обществе ни одно из этих и многих других событий не обходится без появления очередной электронной пометки в памяти многочисленных компьютеров. Одним словом, базы данных так глубоко вошли в нашу повседневную жизнь, что мы зачастую воспринимаем их как что-то само собой разумеющееся.

В перечне современного программного обеспечения базы данных входят в десятку самых востребованных программных продуктов и служат источником неплохого заработка для профессиональных программистов. Судите сами, без хранения и учета данных сегодня обойтись весьма сложно. Многочисленные магазины, склады, страховые агентства, отделы кадров, бухгалтерии, учебные заведения и много других предприятий и организаций остро нуждаются в разработанных специально для них БД. И спрос все еще превышает предложение.

Создать эффективную базу данных весьма непросто, даже если она предназначена для обслуживания незначительных объемов данных и подлежит эксплуатации на домашнем компьютере. Сложность проекта возрастает на порядок, когда возникает задача разработать жизнеспособный коммерческий продукт, с которым будут одновременно работать десятки пользователей. Именно поэтому главная задача этой книги — вооружить читателя знаниями о порядке проектирования реляционных баз данных и разработке приложений БД на одном из самых совершенных и успешных языков программирования — Delphi.

Столь популярный сегодня язык программирования своим рождением обязан профессору Цюрихской высшей технической школы, блестящему ученому, обладателю премии Тюринга — Никлаусу Вирту (*Niklaus Wirth*). Язык изначально предназначался для обучения студентов основам структурного программирования. Уже с самых первых дней своего существования язык Pascal (именно так назывался в те времена предшественник Delphi) был обречен на широчайшую популярность благодаря ряду своих неоспоримых достоинств: гибкости и надежности; простоте и наглядности конструкций; контролю правильности исходного кода на этапе компиляции; возможности построения новых типов данных и многим другим качествам, с которыми мы далее познакомимся.

Книга состоит из двух частей. В первой части излагаются теоретические основы современной реляционной модели данных. Здесь вы узнаете:

- историю появления реляционной модели данных;
- функциональные обязанности и архитектуру систем управления базами данных (СУБД);
- особенности организации доступа к БД;
- основы реляционной модели данных;
- особенности концептуального моделирования БД;
- порядок нормализации реляционных таблиц;
- правила индексирования данных;
- обеспечение многопользовательского доступа к данным и правильную организацию транзакций;
- особенности всех этапов проектирования БД;
- вопросы обеспечения безопасности данных;
- структурированный язык запросов SQL;
- расширяемый язык разметки XML.

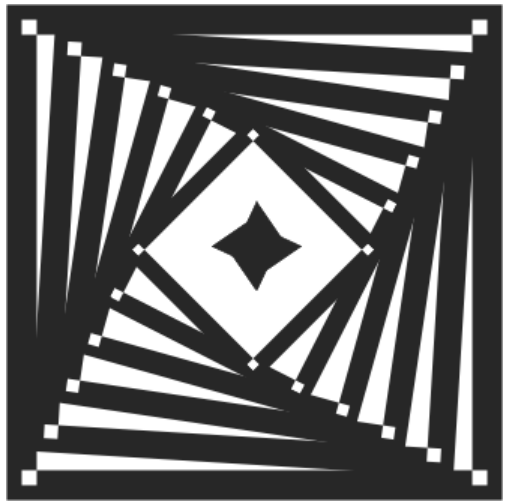
Вторая часть книги посвящена вопросам проектирования приложений баз данных на языке программирования Delphi. Здесь вы найдете исчерпывающую информацию о:

- концепции построения приложения БД Delphi;
- универсальном наборе данных `TDataSet` и полях набора данных;
- технологиях доступа к данным ADO, InterBase и dbExpress;
- порядке построения настольных и клиент-серверных БД;

- применении технологии DataSnap в проектах БД;
- организации взаимодействия с приложениями Microsoft Office.

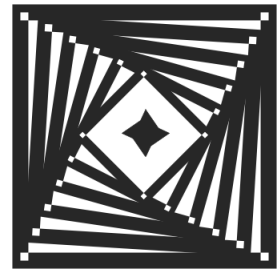
На прилагаемом к книге DVD находятся дополнительные главы, в которых рассмотрена подготовка отчетов в Microsoft Office, создание собственного генератора отчетов, а также построение диаграмм и графиков.

Так как основная направленность этой книги — практическая разработка БД, то читатель, имеющий хотя бы небольшой опыт программирования в Delphi, найдет здесь все необходимое, чтобы самостоятельно спроектировать реляционную базу данных и разработать клиентское приложение для совместной работы с БД. Самое главное, что получит читатель после изучения предложенного материала, — владение методологией работы с любой БД. Книга не ограничивает читателя в выборе целевой СУБД, поэтому вашу базу данных можно развернуть как на основе простейших настольных систем (например, Microsoft Access), так и на фундаменте профессиональных многопользовательских клиент-серверных систем (таких как InterBase, FireBird, Blackfish SQL, Microsoft SQL Server, Oracle, MySQL и т. п.).



ЧАСТЬ I

Введение в реляционные базы данных



Глава 1

Системы, основанные на файлах

Разговор о первых прототипах современных баз данных — системах, основанных на файлах, начнем с небольшого, но весьма поучительного экскурса в историю. В пятидесятых годах XX века вокруг электронно-вычислительных машин постоянно "вился рой" инженеров, математиков и программистов. Инженеры меняли радиодетали, математики выдумывали формулы, а программисты замыкали круг — закладывали в ЭВМ программы, которые жгли радиодетали. Огромное количество обслуживающего персонала, "роящегося" вокруг первых вычислительных устройств, в некоторой степени роднило обслуживание ЭВМ с возведением Вавилонской башни. Процесс разработки и загрузки прикладного программного обеспечения тех времен был достоин кисти Сальвадора Дали. Наивный заказчик расчетов приходил к программисту-алгоритмисту и просил, чтобы машина ответила, сколько на ее взгляд будет равняться 2+2. Алгоритмист в глубокой задумчивости рисовал алгоритм и передавал его программисту, перекладывающему алгоритм на низкоуровневый язык машинных команд, чтобы ЭВМ смогла "усвоить" программу, которую техники набивали на перфоленту или на перфокарты. Данные с перфокарт полдня загружались в память лампового монстра, затем он пару-другую раз зависал, шипел, кряхтел и, наконец, к всеобщей радости выдавал распечатку. К этому моменту времени результат уже никого не интересовал, т. к. заказчик все успел посчитать в уме…

К шестидесятым годам XX века картина приобрела больше мажорных нот. ЭВМ подверглись усовершенствованию настолько, что уже были способны не только воодушевлять писателей-фантастов и согревать воздух, но и выполнять некоторые полезные операции, в первую очередь связанные с утомительными математическими расчетами. Повысилась не только производительность процессоров, но и на качественно другой уровень перешли периферийные устройства. Теперь почти отпала необходимость вставки между пользователем и машиной гильдии разношерстных посредников. Пользователь просто садился за терминал и получал возможность прямого общения с ЭВМ. Заказчики — люди разные, и далеко не всех интересовали расчеты траекторий баллистических ракет. Многие искали применения вычислительных машин в другой не менее важной области — хранении и обработке больших массивов данных. Судите сами: XX век — век информации; в офисах корпораций, в правительственных учреждения, в архивах и библиотеках

хранились миллионы тонн бумаги с данными о чем угодно, начиная с роста поголовья пингвинов в Антарктике и заканчивая налоговыми декларациями от горячо любимых сограждан. Все это необходимо не просто хранить, а еще и максимально быстро обрабатывать с возможностью получения аналитических выводов, графиков и статистики. До сих пор все попытки систематизации колоссальных объемов данных, представленных на бумажных носителях, были в некотором родстве с сизифовым трудом. И вдруг, о чудо, наконец, у человечества появился шанс вкатить камень на вершину горы!

В обязанности вычислительных машин, кроме проведения расчетов, вменили еще один элемент — хранение и обработку больших объемов данных. Только бесконечно наивный пользователь (метко называемый в народе "чайником") может предположить, что ЭВМ сразу же подставила человечеству свое плечо и в мановение ока все бумажные архивы превратились во всех отношениях в совершенные базы данных. Не тут-то было! Вычислительные машины — лишь инструмент, который работает ровно так, как его научат программисты. Поэтому примерно на стыке 50-х и 60-х годов XX века перед программистами была поставлена очередная задача — научить машины обслуживать большие объемы данных.

Мы с вами все учились сами, а некоторые из нас даже пытались учить других. Любой участник образовательного процесса, находящийся в здравом уме и доброй памяти, понимает, что научить можно только тому, что в совершенстве знаешь сам. А теперь ответьте на вопрос: "Что в начале шестидесятых программисты знали о хранении больших банков данных?" Прав окажется тот, кто скажет — практически ничего! Не удивительно, что на первых этапах становления такой области знаний, как базы данных, все пошли по особому пути, который получил название *системы, основанные на файлах* (file-based system) или просто *системы файлов*. К чему столько скепсиса? Поймете к концу главы, а пока рассмотрим историю болезни прототипов современных БД — файловых систем.

Особенность человеческого образа мышления заключается в том, что при поиске решений новых проблем в первую очередь мы пытаемся применить подходы, которые были придуманы ранее. Примеров в истории предостаточно. Вспоминайте сами, разработчики первых летательных аппаратов весьма настойчиво пытались обучить их махать крылами. Эффективность подобного решения они обычно проверяли сами, поэтому долго не жили. В большинстве своем программисты также народ прямолинейный. Разработчики прообразов баз данных при поиске решения посмотрели вокруг и увидели, что везде, где есть бумажные архивы, данные хранятся в картотеках. Возьмем обычную городскую библиотеку тех лет. Здесь каждой книге соответствует отдельная бумажная карточка, в которой отражены данные об авторе, названии книги, годе издания, месте хранения и т. д. Карточки систематизированы по областям знаний и упорядочены по алфавиту. Любой грамотный посетитель, придя в библиотеку, достаточно быстро (за пару-тройку часов перелопатив с тысячу карточек) выяснял, что интересующей его книги там нет, и с гордо поднятой головой уходил восвояси... Плохо это или хорошо, но на тот момент времени другого, более рационального решения проблемы архивного дела не существовало. Поэтому воодушевленные программисты, не теряя ни одной секун-

ды на "лишние" размышления, самоотверженно приступили к обучению самолетов махать крыльями — взялись за "лобовое" проектирование электронных аналогов бумажных картотек.

Замечание

Почему системам, основанным на файлах, мы уделяем столько времени и не переходим сразу к изучению баз данных? По двум причинам. Во-первых, понимание сути недостатков, присущих файловым системам, позволяет избежать их в дальнейшем. Во-вторых, даже сегодня в спектре программного обеспечения есть место для приложений, построенных на основе файлов.

Шутки шутками, а пионером в области разработки файловых систем стала фирма IBM, разработавшая операционную систему OS/360. В те годы это был мощный эволюционный скачок, т. к. прежде в качестве устройств внешней памяти в основном использовали стримеры — накопители на магнитной ленте, а машины 360-й серии были снабжены новейшими контроллерами жестких дисков. С тех пор в тезаурус пользователей и программистов вошло новое понятие — файл.

Принцип построения систем, основанных на файлах

Рассмотрим в общих чертах идею построения систем, основанных на файлах. Допустим, что мы планируем создать программку, хранящую сведения об именах и днях рождений наших сотрудников. Получивший столь ответственное задание программист поступает следующим образом.

Во-первых, он готовит структуру, подходящую для хранения указанных данных. Для этого программист просматривает список персонала и выясняет максимальное число символов в фамилии, имени и отчестве (допустим, это значения: 20, 15 и 15 байтов). Затем программист выделяет какое-то число байтов для хранения даты (пусть это будет 8 байтов). Узнав все необходимые размерности, разработчик описывает структуру непосредственно в коде программы (рис. 1.1).

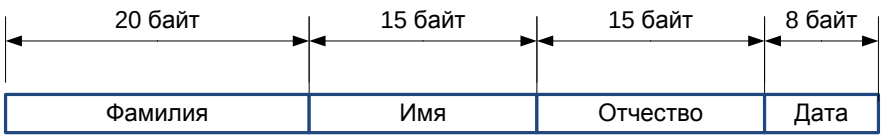


Рис. 1.1. Структура для хранения одной записи в файле

Во-вторых, программист разрабатывает процедуры, осуществляющие основные операции по работе с файлом. Как минимум это добавление новой записи, редактирование, удаление и чтение записи. Ни одну из этих операций невозможно осуществить без знания размерности и состава полей исходной структуры. При вставке

новой строки в файл к нему следует добавить 20+15+15+8=58 байтов, причем процедура добавления должна знать, что фамилия начинается с 1-го байта, имя с 21-го и т. д. При просмотре файла необходимо осуществлять последовательные операции чтения порциями по 58 байтов, опять же процедура чтения должна иметь информацию, сколько байт отводится тому или иному полю. Как вы догадываетесь, операции редактирования и удаления не являются исключением из правил и также нуждаются в знаниях об исходной структуре. К счастью, эти сведения искать не нужно — все данные о составе полей нашей программе хорошо известны, ведь описание структуры спрятано внутри нее.

Недостатки систем, основанных на файлах

Поначалу у разработчиков систем, основанных на файлах, дела шли весьма неплохо. Пользователям очень нравилось, что работа с электронными картотеками была схожа с работой с бумажными архивами. В свою очередь, программисты были довольны, что они сравнительно легко зарабатывают себе на жизнь.

Идиллия продолжалась недолго. Очень скоро на, казалось, безоблачном горизонте забрезжили грозовые тучи — стали проявляться отрицательные стороны "лобового" подхода разработчиков первых прототипов баз данных. Из всего сонма недостатков особо выделяются пять проблем (рис. 1.2).

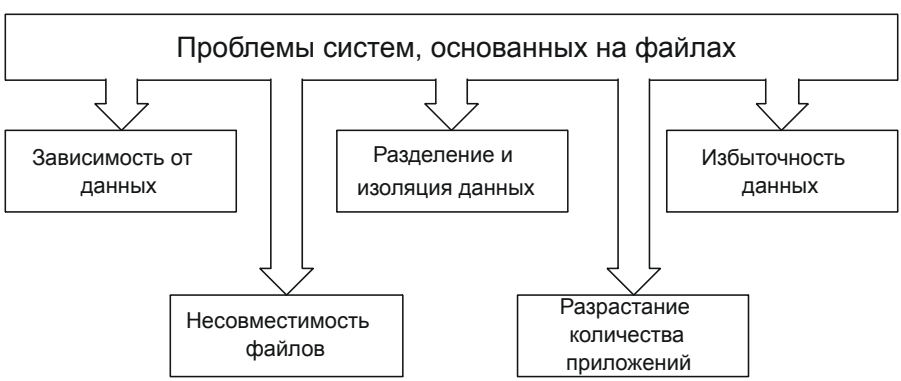


Рис. 1.2. Проблемы систем, основанных на файлах

- *Зависимость от данных*. Уже первая проблема, с которой столкнулись разработчики файловых систем, не предвещала ничего хорошего. Допустим, что нам требуется осуществить элементарную операцию — увеличить на один символ размер поля, отвечающего за хранение фамилии. Оказалось, что даже незначительное усовершенствование структуры влечет за собой ком дополнительных трудностей. Судите сами, изменение размера или состава полей записи типизи-

рованного файла вынуждает нас не только перекомпилировать исполняемый файл, но и написать одноразовую программу-конвертор, которая должна преобразовать старые данные к новому формату. То же самое произойдет не только при изменении размерностей, но и в случае, если мы попытаемся добавить новые или удалить лишние поля с данными.

Таким образом, первой болезнью, поразившей первые прототипы баз данных, стала зависимость системы, основанной на файлах от данных.

- *Разделение и изоляция данных*. Файловые системы объединяют в себе десятки отдельных файлов с данными. В одном файле хранится информация о сотрудниках фирмы, в другом — сведения о клиентах, в третьем — перечень предоставляемых услуг, в четвертом — список заказов и т. д. Для извлечения логически связанных данных (допустим о клиентах и их заказах) программисту приходилось писать сложные алгоритмы синхронного чтения из двух файлов. С увеличением числа файлов, вовлекаемых в итоговый отчет, сложность задачи возрастала в арифметической прогрессии. Задача извлечения взаимоувязанных данных из десятка файлов могла стать непосильной не только для программиста, но и для вычислительных машин тех времен. Масла в огонь дополнительно подливал тот факт, что данные могли быть разделены между отделами и службами предприятия — в отделе кадров находились данные о сотрудниках, в отделе продаж — о заказах и т. п. В 1960-х годах удельный вес предприятий, чьи машины были объединены локальными вычислительными сетями, стремился к нулю, поэтому данные были еще и изолированы друг от друга. А теперь представьте себе программиста тех времен, мечущегося по организации, в надежде объединить данные в единое целое…
- *Избыточность (дублирование) данных*. С усовершенствованием микропроцессорной техники многие руководители предприятий стали отказываться от покупок больших ЭВМ и начали отдавать свои предпочтения более дешевым мини-ЭВМ, расставляя их по отделам и службам своих организаций. Подобное, во многом правильное решение имело и свои отрицательные стороны, одна из них — вынужденный отказ от централизованного хранения данных. Децентрализованное хранение данных систем, основанных на файлах, на нескольких машинах приводила к тому, что одна и та же информация повторялась на магнитных носителях многих мини-ЭВМ, разбросанных по учреждению. В этом случае страшна не столько избыточность данных, сколько нарушение непротиворечивости данных предприятия — на всех машинах должны храниться идентичные копии данных.

  Избыточность данных порождает целый букет проблем и проблемок.
- *Противоречивость данных*. На одной из рабочих станций предприятия хранится устаревший номер телефона вашего контрагента. На второй этого номера вовсе нет. На третьем компьютере, за счет ошибки оператора, там находится телефон его бабушки. В результате ни один из звонков не достигает цели. Как следствие, фирма терпит убытки.

- *Аномалии данных*. Некорректные данные влекут за собой шлейф дополнительных неприятностей: аномалий добавления, редактирования и удаления записей. Какая из аномалий способна принести больше печали в наш офис, судите сами. Допустим, у нашей фирмы появился новый, не жалеющий денег, оптовый покупатель. Данные нового клиента следует ввести сразу в несколько систем файлов (отел сбыта, личная картотека главного менеджера, бухгалтерия и т. д.). Если все сделано безошибочно, то порядок обеспечен… Но если таких покупателей несколько, то я могу гарантировать, что где-нибудь, кто-нибудь спутает пару цифр в номерах счетов (только вслушайтесь в названия "БИК", "ИНН", "ОКПО", услышав эти аббревиатуры, не грех и ошибиться). В результате платеж в пару миллионов уходит на чужой расчетный счет. После долгого судебного разбирательства и уплаты неустоек вы, наконец, выясняете, в чем причина сбоя, но к этому времени вам уже все равно... С редактированием данных в избыточных системах дела также обстоят далеко не лучшим образом. В идеале следует нанять отдельного сотрудника, задачей которого станет регулярная "пробежка" по всем структурным подразделениям компании, чтобы исправить почтовый адрес (номер телефона, дату рождения, номер счета или что-нибудь в этом духе) главного спонсора фирмы. В результате поздравительная открытка, отправленная в канун очередного юбилея, не попадет к адресату, а в отместку "благодарный" юбиляр не перечислит вашей компании давно обещанные (и так необходимые сейчас) финансовые вливания. Впрочем, считайте, что вам крупно повезло, ведь уязвленное самолюбие может привести и к более серьезным последствиям… Аномалия удаления в состоянии принести не меньшие неприятности. Как вы думаете, как скажется на финансовом состоянии фирмы тот факт, что она станет выплачивать ежегодные премии давно уволенному менеджеру? Это печальное событие произойдет только потому, что в бухгалтерии забудут вычеркнуть всего одну строку с данными.

  В "доисторических" системах файлов избыточность данных вынужденно присутствовала и в рамках одного-единственного проекта. Допустим, что от нас потребуют дополнить программу "Дни рождений сотрудников" еще одним информационным полем — местом работы. В результате в файле появятся многократно дублирующиеся данные, например Петров — Бухгалтерия, Иванов — Бухгалтерия и т. д. Исследования файловых систем тех времен показали, что до 60% хранящихся в них данных избыточны, в этом убедилась компания North American Rockwell, участвовавшая в 1960-х годах в программе высадки астронавтов на Луну. Учитывая астрономическую стоимость первых жестких дисков тех времен, это были весьма непродуктивные расходы.
- *Несовместимость файлов*. Структура файлов с данными определялась не только разработчиками программного обеспечения, но и языками программирования, применяемыми в тех или иных организациях. Построение файла, описанного на языке Algol, могло принципиально отличаться от структур, генерируемых программами на PL/1, ADA или каким-нибудь еще средством разработки приложений тех времен. Более того, дополнительные ограничения

вносились из-за особенностей архитектурных решений тех или иных ЭВМ. Помножьте это на специфичные черты различных операционных систем. В результате файлы, выходящие из-под "пера" программистов, зачастую становились несовместимыми, хотя и содержали практически одно и то же описание данных.

- *Разрастание количества приложений*. Сами по себе данные не представляют никакого интереса. Представьте, что у вас имеется файл с несортированными телефонными номерами жителей миллионного города — это хорошая новость. А теперь плохая — у вас нет средств упорядочивания и поиска данных. В результате цена таким данным — ломаный грош, ну-ка найдите номер телефона гражданина Иванова, затерявшегося где-то среди сотен тысяч других номеров… Данные нужно не только хранить, но и уметь представлять пользователю в удобном формате. А пожеланий у пользователей ни счесть, одним требуется, чтобы списки заказов упорядочивались по алфавиту, другим по дате заказа, третьим хотелось бы, чтобы имелась возможность сортировки записей по денежной сумме. Старуха, затребовавшая у Золотой рыбки перечень услуг (начиная от тривиального корыта и заканчивая царским троном), — просто ангел в сравнении с запросами к данным у биржевого аналитика, главного бухгалтера завода или заведующего гипермаркетом. Идеи и пожелания пользователей сыплются на программиста как из рога изобилия и никто кроме него не ведает, что каждый новый запрос приводит к цепной реакции — бесконечной переработке исходного приложения. В конце концов, головная программа обрастает скопищем утилит и "утилиток", что, в свою очередь, необратимо ведет к хаосу.

Пути устранения недостатков систем, основанных на файлах

Сегодня системы, основанные на файлах, практически не используются, исключение составляют небольшие по числу записей хранилища, состоящие из одного-двух файлов данных. Чтобы не повторять прошлые ошибки, проектировщики БД сделали нужные выводы:

Во-первых, разработчики в принципе отказались от хранения физической структуры данных в коде приложений. Вместо этого описание данных стали выносить в отдельное хранилище, называемое *системным каталогом* (system catalog). Таким образом, во всех современных БД помимо собственно хранимых в них данных еще имеются метаданные (данные о данных). Если внешняя программа обладает возможностью чтения метаданных, то она без труда сможет получить доступ к хранимой в БД информации.

Во-вторых, стали предпринимать активные попытки стандартизировать способы описания и хранения данных. Наличие стандарта, единого для всех разработчиков, значительно упростило доступ к данным.