# Оглавление

# Введение

Вряд ли сегодня кому-то удастся назвать современную область знаний, в которой не нашли бы применения компьютерные базы данных (БД). Наука, образование, экономика, электронная коммерция, медицина, статистика, военное дело – список можно продолжать очень долго. 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.

Базы данных входят в десятку самых востребованных программных продуктов и служат источником неплохого заработка для профессиональных разработчиков. Судите сами, без хранения и учета данных сегодня обойтись весьма сложно. Многочисленные магазины, склады, страховые агентства, отделы кадров, бухгалтерии, учебные заведения и множество 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 смогут одновременно работать десятки пользователей. Именно поэтому главная задача книги – вооружить читателя знаниями о технологии проектирования баз данных. Здесь вы подчерпнете всю необходимую информацию о:

- задачах, решаемых с помощью БД;
- архитектуре систем управления базами данных (СУБД);
- реляционной, объектно-ориентированной, документ-ориентированной (и ряде других) моделях данных;
- жизненном цикле проектов БД, этапах концептуального, логического и физического проектирования БД;
- особенностях физического хранения и правилах индексирования данных;

- многопользовательском доступе к данным и управлении транзакциями;
- особенностях организации доступа к БД и обеспечении безопасности данных;
- структурированном языке запросов SQL;
- применении в SQL регулярных выражений;
- расширяемом языке разметки XML;
- многомерном анализе данных;
- централизованных и распределенных БД;
- правилах разработки клиентских приложений БД;
- особенностях документирования проектов БД.

Самое главное, что получит читатель после изучения предложенного материала, – владение методологией работы с любой БД. Книга не ограничивает читателя в вопросе выбора целевой СУБД, поэтому ваша база данных может быть развернута как на основе простейших настольных систем, так и на фундаменте профессиональных многопользовательских клиент-серверных программных решений.

# Соглашения

Для акцентирования внимания читателя на ключевых частях излагаемого материала такой текст отмечен особым форматированием:

> **Замечание**
> Таким способом в тексте книги выделен материал, который вы должны принять к сведению. Обычно это определения, комментарии или замечания.

Кроме того:

- впервые встречающиеся термины и определения выделены **полужирным шрифтом**;
- впервые встречающиеся аббревиатуры комментируются;
- код примеров, синтаксические конструкции даны моноширинным шрифтом;
- помимо содержания, книга включает подробный предметный указатель, позволяющий найти в тексте интересующую читателя информацию.

# Глава 1

# Эволюция баз данных

Очень сложно привести пример какой-либо области науки и техники, сделавшей за последние полстолетия столь же значимый рывок в своем развитии, как современная микроэлектроника и идущие с ней рука об руку информационные технологии (ИТ). Более того, ИТ развиваются столь динамично, что даже специалистам в этой области приходится едва ли не каждые 5–6 лет кардинально переучиваться, дабы успеть попасть в последний вагон улетающего вдаль с космической скоростью экспресса технологий разработки программного обеспечения.

Как человек сумел за столь короткий срок совершить столь большие шаги в области ИТ? Чтобы ответить на этот вопрос, стоит разобраться с тем, что происходило в те «старозаветные» времена, в которых компьютер упоминался лишь в произведениях писателей-фантастов. Для этого нам предстоит отмотать временную ленту всего на несколько десятилетий назад и попасть в эпоху, когда на смену механическому арифмометру стали приходить первые вычислительные машины. Именно этот момент времени мы и станем считать началом всех начал для современных информационных технологий в целом и зарождающихся баз данных (БД) в частности.

Для полного погружения в эпоху нарисуем картину научно-вычислительной лаборатории 50-х годов XX века. Посреди огромного зала всеми цветами радуги переливается огромный ламповый монстр – электронно-вычислительная машина. А вокруг нее вьется рой инженеров, математиков и программистов. Инженеры меняют радиодетали, математики выдумывают формулы, а программисты замыкают круг – на основе предложенных математиками формул закладывают в ЭВМ программы, чтобы последние жгли электронные лампы. Огромное количество обслуживающего персонала, трудящегося во благо научно-технического прогресса, в некоторой степени роднило обслуживание ЭВМ с ходом

возведения Вавилонской башни (как с точки зрения процесса, так и по результату).

Технология разработки прикладного программного обеспечения тех времен был достойна кисти Сальвадора Дали. Наивный заказчик расчетов приходил к математику и просил, чтобы машина ответила – сколько, на ее взгляд, будет равняться 2 + 2. Математик-алгоритмист в глубокой задумчивости рисовал алгоритм и передавал его программисту, перекладывающему алгоритм на низкоуровневый язык машинных команд. Для того чтобы ЭВМ смогла усвоить программу, последняя набивалась техниками на перфоленту или на перфокарты. Данные полдня загружались в память машины, затем она пару-другую раз зависала, шипела, кряхтела и, наконец, к всеобщей радости, выдавала распечатку. К этому времени результат уже никого не интересовал… Но согласитесь – сколь увлекателен сам процесс!

К 60-м годам прошлого века ЭВМ подверглись усовершенствованию настолько, что уже были способны не только воодушевлять писателей-фантастов и согревать воздух, но и производить некоторые полезные операции, в первую очередь связанные с утомительными математическими расчетами. Вас интересует таблица синусов с точностью до 18 знаков после запятой? Теперь уже не надо напрягать свой мозг – обращаемся за помощью в научно-вычислительную лабораторию, и всего лишь через час распечатка у вас в кармане. И не важно, что вместо синусов вам посчитали косинусы: главное – никаких умственных затрат, да и точность соблюдена!

Повысилась не только производительность процессоров, но и на качественно другой уровень перешли периферийные устройства. Теперь почти пропала необходимость вставки между пользователем и машиной гильдии разношерстных посредников. Человек, имеющий некоторое представление о способах общения с машиной, просто садился за терминал и получал возможность самостоятельно, без какой-либо посторонней помощи вгонять ЭВМ в ступор. Пользователи – люди разные, и далеко не всех интересовали расчеты траекторий баллистических ракет. Огромный пласт людей искал применения для машин в другой не менее важной области – области хранения и обработки больших массивов данных. Судите сами: XX век – век информации, в офисах корпораций, в правительственных учреждениях, 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 обработать с возможно-

стью получения аналитических выводов, графиков и статистики. До сих пор все эти 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!

Именно с этого момента в сферу обязанностей вычислительных машин, кроме проведения расчетов, вменили еще одну задачу – хранение и обработку больших объемов данных. Тем более что на смену перфолентам и стримерам с магнитной пленкой стали приходить жесткие диски.

> **Определение**
> Данные (data) – информация, представленная в формализованном виде, пригодном для передачи, интерпретации или обработки с участием человека или автоматическими средствами.

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 в во всех отношениях совершенные базы данных. Не тут-то было! Вычислительные машины – это 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 задача научить машины обслуживать большие объемы данных.

# Электронные картотеки

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 60-х программисты знали о хранении больших данных?» Прав окажется тот, кто скажет – практически ничего! Так что нет ничего удивительного в том, что на первых этапах становления такой области знаний, как базы данных, все пошли по особому пути, который получил название **систем, основанных на файлах** (file-based system), или просто **системы файлов**. К чему столько скепсиса? Поймете через пару страниц, а пока рассмотрим историю болезни прототипов современных БД – систем файлов.

Особенность человеческого образа мышления заключается в том, что при поиске решений сложных проблем в первую очередь мы пыта-

емся применить уже известные методики. Примеров такого подхода в истории предостаточно. Например, разработчики первых летательных аппаратов весьма настойчиво пытались обучить их махать крыльями – ведь именно так делали птицы. Эффективность подобного решения новаторы обычно проверяли сами, поэтому долго не жили. В большинстве своем программисты – также народ прямолинейный. Создатели прообразов баз данных при поиске решения огляделись 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, 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 секунды 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.

## Принцип построения систем файлов

Рассмотрим в общих чертах идею построения систем, основанных на файлах. Допустим, что мы планируем создать программу, хранящую сведения о сотрудниках предприятия. Получивший столь ответственное задание программист поступает следующим образом.

Во-первых, он готовит структуру, подходящую для хранения данных. Для этого программист просматривает список персонала и выясняет максимальное число символов в фамилии, имени и отчестве (допустим, это значения: 20–15–15 байт). Затем программист выделяет ка-

кое-то число байтов для хранения названия должности, даты рождения и остальных подлежащих учету элементов структуры. Узнав все необходимые размерности, разработчик описывает структуру непосредственно в коде программы (рис. 1.1).

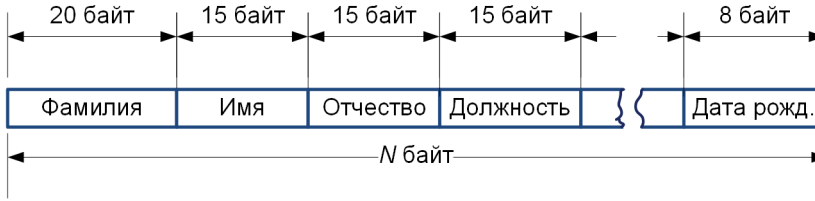


**Рис. 1.1.** Структура для хранения одной записи в файле

Во-вторых, программистом разрабатываются процедуры, осуществляющие основные операции по работе с файлом. Как минимум это добавление новой записи, редактирование, удаление и просмотр 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 *N* байт, 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, пропорциональными размеру отдельной записи *N* байт; опять же, процедура чтения должна обладать информацией – сколько байт отводится тому или иному полю. В свою очередь, операции редактирования и удаления также не являются исключением из правил и нуждаются в знаниях об исходной структуре. К счастью, эти сведения искать не стоит – все данные о составе полей нашей программе хорошо известны, ведь определение структуры спрятано внутри нее.

## Недостатки систем файлов

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, программистам нравилось то, что они сравнительно легко зарабатывают себе на жизнь.

Идиллия продолжалась недолго. Очень скоро, казалось на безоблачном горизонте, забрезжили грозовые тучи – стали проявляться отрицательные стороны лобового подхода разработчиков первых прототипов баз данных. Из всего сонма недостатков особо выделяются пять проблем [25]:

1) зависимость от данных;
2) разделение и изоляция данных;
3) избыточность данных;
4) несовместимость файлов;
5) разрастание количества приложений.

**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 структуры вынуждает нас не только переписать исполняемый файл, но и сочинить одноразовую программу-конвертор, которая должна преобразовать старые данные к новому формату.

**Разделение и изоляция данных**. Системы, основанные на файлах, объединяют в себе десятки отдельных файлов с данными. В одном файле хранятся данные 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 выдумывать замысловатые алгоритмы синхронного чтения из двух файлов. С увеличением количества файлов, вовлекаемых в итоговый отчет, сложность возрастает в арифметической прогрессии. Задача извлечения взаимоувязанных данных из десятка файлов могла стать непосильной не только для программиста, но и для вычислительных машин тех времен. Масло в огонь подливало еще то, 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, посему данные были еще и изолированы друг от друга. А теперь представьте себе программиста тех времен, мечущегося по организации в надежде объединить разделенные данные в единое целое…

**Избыточность (дублирование) данных**. С появлением первой 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. Система децентрализованного хранения данных систем, основанных на файлах на нескольких машинах, приводила к тому, что одни и те же сведения повторялись на магнитных носителях многих мини-ЭВМ, разбросанных по учреждению. В этом случае даже был не столь страшен факт избыточности данных, сколь высока вероятность нарушения непротиворечивости данных предприятия – на всех машинах должны храниться идентичные копии данных, но ведь любое изменение данных на одной из ЭВМ никак не отражалось на остальных станциях до тех пор, пока данные не синхронизировали вручную.

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 многократные дубликаты данных, например Петров – Бухгалтерия, Иванов – Бухгалтерия и т. д. Исследования файловых систем тех времен показали, что до 60 % хранящейся в них данных были избыточны. Учитывая астрономическую стоимость жестких дисков, это весьма непродуктивные расходы.

В свою очередь, избыточность данных порождала целый букет проблем и проблемок. Одна из них – **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**. Избыточные и противоречивые данные влекут за собой шлейф дополнительных неприятностей в лице: аномалий добавления новой записи, аномалий редактирования и аномалий удаления. 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 (отдел сбыта, личная картотека главного менеджера, бухгалтерия и т. д.). Если все сделано безошибочно, то все в порядке… но если таких покупателей несколько, то можно гарантировать, что где-нибудь кто-нибудь спутает пару цифр в номерах счетов. В результате платеж 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 можно нанять отдельного сотрудника, задачей которого станет регулярная пробежка по всем структурным подразделениям компании с целью 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 влияния. 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.

**Несовместимость файлов**. Структура файлов с данными определялась не только разработчиками программного обеспечения, но и языками программирования, состоящими на вооружении в тех или иных организациях. Построение файла, описанного на языке Algol, могло принципиально отличаться от структур, генерируемых средствами PL/1, ADA или какого-нибудь еще средства разработки приложений тех времен. Более того, дополнительные ограничения вносились из-за особенностей архитектурных решений, принятых в основу построения тех или иных ЭВМ. Помножьте это на специфичные черты различных операционных систем. Как следствие выходящие из-под «пера» программистов файлы зачастую становились несовместимыми, хотя и содержали практически идентичное 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 надо не только хранить, но и уметь представлять пользователю в удобном формате. А пожеланий у пользователей не 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 от 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 рано или поздно (а главное – необратимо) приведет проект к коллапсу.

## Пути устранения недостатков систем файлов

Сегодня системы, основанные на файлах, практически не используются, исключение составляют состоящие из одного-двух файлов данных небольшие по числу записей хранилища. Дабы не повторить ошибки программистов тех времен, проектировщики БД сделали нужные выводы.

Во-первых, разработчики в принципе отказались от хранения физической структуры данных в коде приложений. Вместо этого описание данных стало выноситься 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 предприниматься активные попытки стандартизировать способы описания и хранения данных. Наличие единого для всех разработчиков стандарта значительно упростило доступ к данным.

В-третьих, возникла необходимость создания единого универсального языка, позволяющего производить с данными наиболее важные операции: вставки, редактирования, удаления и просмотра.

В результате на смену морально устаревшим системам файлов пришли свободные от недостатков своих предшественников базы данных.

Как видите, в составе баз данных основную роль играют структуры и описывающие эти структуры метаданные, кроме того, в БД задействуются индексы, представления, хранимые процедуры, триггеры и т. п.