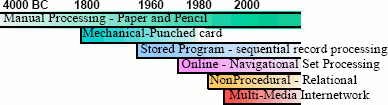
**Управление данными: Прошлое, Настоящее и Будущее**

*Джим Грей*, *Системы Управления Базами Данных # 3/1998, Новая редакция: Сергей Кузнецов, 2009 г.*

*Оригинал: Jim Gray. Data Management: Past, Present, and Future. IEEE Computer 29(10): 38-46 (1996), Vol. 29, # 10, October 1996..*

Быстрое развитие технологий хранения данных, коммуникаций и обработки позволяет переместить всю информацию в киберпространство. Программное обеспечение для определения, поиска и визуализации оперативно доступной информации является ключом к созданию такой информации и доступу к ней. Начало систем управления данными связано с решением традиционных задач автоматизации – учета транзакций в бизнесе, науке и коммерции. Эти данные состояли главным образом из чисел и символьных строк. Сегодня эти системы обеспечивают инфраструктуру для большей части общества, предоставляя возможность быстрого, надежного, безопасного и автоматического доступа к данным, распределенным по всему миру. Следующие шаги состоят в автоматизации доступа к более богатым формам данных: изображениям, аудио- и видеоданным, картам и т.д. Другой важной задачей является автоматическое обобщение и абстрагирование данных в соответствии с запросами пользователей. Мультимедийные базы данных и средства доступа к ним будут краеугольным камнем в нашем движении к киберпространству.

  
Рис. 1. Шесть поколений управления данными, начиная от ручных методов, через несколько стадий автоматизации управления данными.

Системы управления данными обычно хранят громадные объемы данных, представляющих исторические записи организации. Размеры этих баз данных бурно растут. Системы постоянно изменяются. Действительно, б*о*льшая часть крупных систем баз данных была разработана несколько десятков лет тому назад и развивалась вместе с развитием технологии. Взгляд в историю помогает понять текущие системы.

В управлении данными имелось шесть разных фаз. Вначале данные обрабатывались вручную. На следующем шаге использовались оборудование с перфокартами и электромеханические машины для сортировки и табулирования миллионов записей. На третьей фазе данные хранились на магнитных лентах, и сохраняемые программы выполняли пакетную обработку последовательных файлов. На четвертой фазе было введено понятия схемы базы данных и оперативного навигационного доступа к данным. На пятой фазе был обеспечен автоматический доступ к реляционным базам данным и была внедрена распределенная и клиент-серверная обработка. Теперь мы находимся в начале шестого поколения систем, которые хранят более богатые типы данных, в особенности, документы, изображения, аудио- и видеоданные. Эти системы шестого поколения представляют собой базовые средства хранения для появляющихся приложений Internet и intranet.

**2.0 Нулевое поколение: менеджеры записей (4000 г. до н. э. – 1900)**

В первых известных письменных свидетельствах описывается учет царской казны и налогов в Шумере. Поддержка записей имеет долгую историю. В следующие шесть тысяч лет наблюдается эволюция от глиняных таблиц к папирусу, затем к пергаменту и, наконец, к бумаге. Имелось много новшеств в представлении данных: фонетические алфавиты, сочинения, книги, библиотеки, бумажные и печатные издания. Это были большие достижения, но обработка информации в эту эпоху производилась вручную.

**2.1. Первое поколение: менеджеры записей (1900-1955)**

Впервые автоматизированная обработка информации появилась приблизительно в 1800 году, когда Джеквард Лум (Jacquard Loom) начал производить раскрой ткани по образцам, представленным перфокартами. В 1890 г. Холлерит (Herman Hollerith) использовал технологию перфокарт для выполнения переписи населения Соединенных Штатов. Его система содержала запись для каждой семьи. Каждая запись данных представлялась в виде двоичных структур на перфокарте. Бизнес Холлерита в конце концов привел к возникновению International Business Machines. Эта небольшая компания IBM процветала в период от 1915 до 1960 года как поставщик оборудования регистрации данных для бизнеса и правительственных организаций.

К 1955 году у многих компаний имелись целые этажи, предназначенные для хранения перфокарт, на других этажах размещались шеренги перфораторов, сортировщиков и табуляторов. Было ясно, что наступает время, когда новая технология вытеснит перфокарты и электромеханические компьютеры.

**2.2. Второе поколение: программируемое оборудование обработки записей (1955-1970)**

Электронные компьютеры с хранимыми программами были разработаны в 1940-х и начале 1950-х годов для выполнения научных и численных вычислений. Примерно в то же время компания Univac разработала аппаратуру магнитных лент, каждая из которых могла хранить столько информации, сколько десять тысяч перфокарт.

Ключевым компонентом этой новой технологии было программное обеспечение. Было гораздо проще сортировать, анализировать и обрабатывать данные с применением таких языков, как COBOL и RPG. Начали появляться стандартные пакеты для таких общеупотребительных бизнес-приложений как общая бухгалтерия, расчет заработной платы и пр.

Программное обеспечение этого времени поддерживало модель **обработки записей на основе файлов**. Типичные программы последовательно читали несколько входных файлов и производили на выходе новые файлы. Для облегчения определения этих ориентированных на записи последовательных задач были созданы COBOL и несколько других языков программирования. Операционные системы обеспечивали абстракцию файла для хранения этих записей, язык управления заданиями для выполнения заданий и планировщик заданий для управления потоком работ.

Системы **пакетной обработки транзакций** сохраняли транзакции на картах или лентах и собирали их в пакеты для последующей обработки. Пакетная обработка позволяла очень эффективно использовать компьютеры, но обладала двумя серьезными ограничениями. Если в транзакции имелась ошибка, она не распознавалась до вечерней обработки основного файла, и могло потребоваться несколько дней для исправления транзакции. Более важным является то, что бизнес не знал текущего состояния базы данных – поскольку транзакции реально не обрабатывались до следующего утра. Для решения этих двух проблем требовался следующий шаг эволюции – оперативные системы. Этот шаг также существенно облегчил написание приложений.

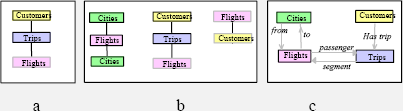
**2.3. Третье поколение: оперативные сетевые базы данных (1965-1980)**

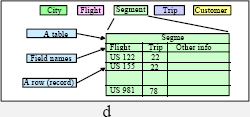
Для таких приложений, как ведение операций на фондовой бирже или резервирование билетов, требуется знание текущей информации. С конца 1950-х лидирующие компании из нескольких областей индустрии начали вводить в использование системы баз данных с оперативными транзакциями. Оперативный доступ к данным обеспечивался несколькими ключевыми технологиями. Аппаратура для подключения к компьютеру интерактивных компьютерных терминалов прошла путь развития от телетайпов к простым алфавитно-цифровым дисплеям и, наконец, к сегодняшним интеллектуальным терминалам, основанным на технологии персональных компьютеров. **Мониторы телеобработки** представляли собой специализированное программное обеспечение для мультиплексирования тысяч терминалов со скромными серверными компьютерами того времени. Эти мониторы собирали сообщения-запросы, поступающие с терминалов, быстро назначали программы сервера для обработки каждого сообщения и затем направляли ответ на соответствующий терминал. Оперативная обработка транзакций дополняла возможности пакетной обработки транзакций, за которой оставались задачи фонового формирования отчетов.

Оперативные базы данных хранились на магнитных дисках или барабанах, которые обеспечивали доступ к любому элементу данных за доли секунды. Эти устройства и программное обеспечение управления данными давали возможность программам считывать несколько записей, изменять их и затем возвращать новые значения оперативному пользователю. В начале системы обеспечивали простой поиск данных: либо прямой поиск по номеру записи, либо ассоциативный поиск по ключу.

Простые индексно-последовательные организации записей быстро развились к более мощной **модели записей, ориентированной на наборы**.

На рис. 2.a показаны некоторые типы записей простой системы резервирования авиационных билетов и их связи. Каждому городу соответствует набор отбывающих рейсов. Каждому заказчику соответствует набор путешествий, а каждое путешествие состоит из набора рейсов. Кроме того, каждому рейсу соответствует набор пассажиров. Как показано на рис. 2.b, эта информация может быть представлена в виде трех иерархий наборов. Каждая из трех иерархий отвечает на отдельный вопрос. Первый – это планирование рейсов в городе. Вторая иерархия дает представление о рейсах заказчика. Третья иерархия говорит, к какому рейсу относится каждый заказчик. Приложение резервирования билетов нуждается во всех трех этих представлениях данных.



  
Рис. 2. Эволюция моделей данных.  
(a) Чистая иерархическая модель с записями, сгруппированными под другими записями.   
(b) По мере развития приложения разным пользователям желательно иметь разные представления данных, выражаемые в виде разных иерархий.  
(c) Диаграмма Бахмана, показывающая типы записей и связи между типами записей.   
(d) Та же информация, представленная в реляционной модели, где все данные и все связи явно представляются как записи. Отношения показаны вверху рисунка. Некоторые детали отношения Segment показаны в нижнем правом углу; это отношение содержит запись для каждого рейса (сегмента) маршрута пассажира.

Иерархическое представление на рис. 2.b обладает существенным недостатком. Избыточное хранение данных не только дорого стоит, но также и порождает проблемы обновлений: при создании рейса или изменении его информации необходимо обновить данные во всех трех местах (всех трех иерархиях). Для решения этих проблем информацию можно представлять в **сетевой модели данных**, что показано на рис. 2.c. На рис. 2.c изображена одна база данных, в которой каждая запись хранится в одном экземпляре и связывается с набором других записей посредством связи. Например, все рейсы, используемые в путешествии конкретного заказчика, связываются с этим путешествием. Программа может попросить систему баз данных перебрать эти рейсы. При потребности между записями могут создаваться новые связи. Рис. 2.c называется по-разному – диаграммой Бахмана или диаграммой «сущность-связь» . Реляционная диаграмма с рис. 2 (рис. 2.d) описывается в следующем разделе.

Управление ассоциативным доступом и обработкой, ориентированной на наборы, было настолько распространено, что в сообществе COBOL выделилась группа Data Base Task Group (DBTG) для определения стандартного способа спецификации таких данных и доступа к ним. Чарльз Бахман (Charles Bachman) в GE (General Electric) построил прототип системы навигации по данным. За руководство работы DBTG, разработавшей стандартный язык определения данных и манипулирования данными, Бахман получил Тьюринговскую премию. В своей Тьюринговской лекции он описал эволюцию моделей плоских файлов к новому миру, где программы могут осуществлять навигацию между записями, следуя связям между записями. Модель Бахмана напоминает навигационную модель страниц и ссылок сегодняшнего Internet.

В сообществе баз данных COBOL кристаллизовалась концепция **схем** и независимости данных. Они поняли потребность в сокрытии физических деталей расположения записей. Программы должны были видеть только логическую организацию записей и связей, так что программы продолжали работать при изменении и развитии способов хранения данных. Записи, поля и связи, не используемые программой, следовало сокрыть – как по соображениям безопасности, так и для того, чтобы изолировать программу от неизбежных изменений схемы базы данных.

В этих ранних базах данных поддерживались три вида схем данных: 1) **логическая схема**, которая определяет глобальный логический проект записей базы данных и связей между записями; 2) **физическая схема**, описывающая физическое размещение записей базы данных на устройствах памяти и в файлах, а также индексы, нужные для поддержки логических связей; 3) предоставляемая каждому приложению **подсхема**, раскрывающая только часть логической схемы, которую использует программа. Механизм логических, физических и подсхем обеспечивал **независимость данных**. И на самом деле, многие программы, написанные в ту эпоху, все еще работают сегодня с использованием той же самой подсхемы, с которой все начиналось, хотя логическая и физическая схемы абсолютно изменились.

Эти оперативные системы должны были решить проблему одновременного выполнения многих транзакций над базой данных, совместно используемой многими терминальными пользователями. До этого подход, основанный на однопрограммном режиме и периодическом изменении основного файла, устранял проблемы параллельного доступа и восстановления. Ранние оперативные системы проложили путь понятию **транзакций**, блокировавших только те записи, к которым производился доступ. Блокировки позволяют конкурирующим транзакциям получать доступ к разным записям. Системы также поддерживали журнал записей, изменявшихся каждой транзакцией. При сбое транзакции журнал использовался для устранения в базе данных воздействий этой транзакции. Журнал транзакций использовался также для восстановления базы данных в случае аварии носителя. При крахе системы для воссоздания текущего состояния базы данных журнал применялся к ее архивной копии.

К 1980 году сетевые (и иерархические) модели данных, ориентированные на наборы записей, стали очень популярны

**2.4. Четвертое поколение: реляционные базы данных и архитектура клиент-сервер (1980-1995)**

Несмотря на успех сетевой модели данных, многие разработчики программного обеспечения чувствовали, что навигационный программный интерфейс был интерфейсом слишком низкого уровня. Было трудно проектировать и программировать эти базы данных. В статье Э.Ф. (Теда) Кодда (E.F. Codd) 1970 года была обрисована реляционная модель, которая казалась альтернативой низкоуровневому навигационному интерфейсу.

Идея **реляционной модели** состоит в том, чтобы единообразно представлять и сущности, и связи. Реляционная модель данных обладала унифицированным языком для определения данных, навигации по данным и манипулирования данными, а не отдельными языками для каждой из этих задач. Еще более важно то, что реляционная алгебра имеет дело со множествами записей (отношениями) как единым целым, применяя операции к множествам записей целиком и производя множества записей в результате. Реляционные модель данных и операции дают возможность получения более коротких и более простых программ для решения задач управления записями.

«Найти номера рейсов до Сан-Франциско, которые являются сегментами путешествия любого заказчика с именем Джонс. Для поиска этих рейсов использовать таблицы City, Flight, Segment, Trip и Customer». Эта программа может показаться сложной, но она значительно проще соответствующей навигационной программы.

Получая такой непроцедурный запрос, реляционная система баз данных автоматически находит лучший способ подбора записей в таблицах City, Flight, Segment, Trip и Customer. Запрос не зависит от того, какие определены связи. Он будет продолжать работать даже после логической реорганизации базы данных. Следовательно, запрос обладает гораздо лучшей независимостью от данных, чем навигационный запрос, основанный на сетевой модели данных.

Вдохновляемые идеями Кодда в течение 1970-х исследователи из академии и индустрии экспериментировали с этим новым подходом к структуризации баз данных и обеспечения доступа к ним, обещавшим более простое моделирование данных и прикладное программирование. Многие реляционные прототипы, разработанные в течение этого периода, сошлись на общей модели и языке. Исследования в IBM Research, возглавлявшиеся Тедом Коддом, Реймондом Бойсом (Raymond Boyce) и Доном Чемберлином (Don Chamberlin), и работа в Калифорнийском университете г. Беркли, которой руководил Майкл Стоунбрейкер (Michael Stonebraker), породили язык, названный SQL. Этот язык был впервые стандартизован в 1985 году. Впоследствии были произведены два основных расширения стандарта. Сегодня почти все системы баз данных обеспечивают интерфейс SQL. Кроме того, во всех системах поддерживаются собственные расширения, выходящие за рамки этого стандарта.

Реляционная модель обладает некоторыми неожиданными преимуществами, кроме повышения продуктивности программистов и простоты использования. Реляционная модель оказалась хорошо пригодной к использованию в архитектуре **клиент-сервер**, к параллельной обработке и графическим пользовательским интерфейсам. Приложение клиент-сервер разбивается на две части. **Клиентская** часть отвечает за поддержку ввода и представление выводных данных для пользователя или клиентского устройства. **Сервер** отвечает за хранение базы данных, обработку клиентских запросов к базе данных, возврат клиенту общего ответа. Реляционный интерфейс особенно удобен для использования в архитектуре клиент-сервер, поскольку приводит к обмену высокоуровневыми запросами и ответами. Высокоуровневый интерфейс SQL минимизирует коммуникации между клиентом и сервером. Сегодня многие клиент-серверные средства строятся на основе протокола Open Database Connectivity (ODBC), который обеспечивает для клиента стандартный механизм запросов высокого уровня к серверу. Парадигма клиент-сервер продолжает развиваться.

**Параллельная обработка баз данных** была вторым неожиданным преимуществом реляционной модели. Отношения являются однородными множествами записей. Реляционная модель включает набор операций, замкнутых по композиции: каждая операция получает отношения на входе и производит отношение как результат. Поэтому реляционные операции естественным образом предоставляют возможности конвейерного параллелизма путем направления вывода одной операции на вход следующей.

Сегодня распространенной практикой реляционных систем является стократное увеличение скорости за счет параллелизма. Задачи интеллектуального анализа данных (data mining), для решения которых могли бы потребоваться недели или месяцы поиска в многотерабайтных базах данных, при использовании параллелизма выполняются за часы. Этот параллелизм полностью автоматизирован. Проектировщики только поставляют данные системе баз данных, и сама система разделяет и индексирует данные. Пользователи предоставляют системе запросы (на основе ODBC), и система автоматически обеспечивает параллельный план выполнения запроса и выполняет его.

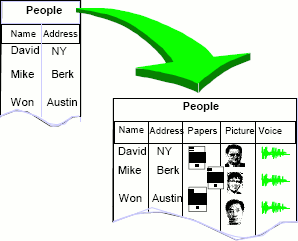
Реляционные данные также хорошо приспособлены к **графическим пользовательским интерфейсам**(GUI). Очень легко представлять отношение как множество записей – к отношениям пригодна метафора электронных таблиц. Пользователи могут просто создавать отношения в виде электронных таблиц и визуально манипулировать ими. На самом деле имеется много инструментальных средств, позволяющих перемещать данные между документами, электронными таблицами и базами данных. Явно и единообразно представленные данные, связи и метаданные делают это возможным.

Комбинация GUI и реляционных систем подходит ближе всего в достижению цели автоматического программирования. GUI позволяют просто конструировать сложные запросы. Для заданного непроцедурного запроса реляционные системы находят наиболее эффективные способы его выполнения.

К 1980 году Oracle, Informix и Ingress принесли на рынок свои системы управления реляционными базами данных. Через несколько лет на рынке появились продукты IBM и Sybase. К 1990 году реляционные системы стали более популярными, чем предшествовавшие им ориентированные на наборы записей навигационные системы. Между тем файловые системы и системы, ориентированные на наборы, оставались рабочими лошадками многих корпораций. С годами эти корпорации построили громадные приложения и не могли легко перейти к использованию реляционных систем. Реляционные системы скорее стали ключевым средством для новых клиент-серверных приложений.

**2.5. Пятое поколение: мультимедийные базы данных (1995-...)**

Реляционные системы внесли много усовершенствований в удобство использования, графические интерфейсы, клиент-серверные приложения, распределенные базы данных, параллельный поиск данных и интеллектуальный анализ данных. Тем не менее, около 1985 года исследовательское сообщество начало рассматривать вопросы, выходящие за рамки реляционной модели. Традиционно существовало четкое разделение программ и данных. Этот подход хорошо работал, пока речь шла только о таких данных, как числа, символы, массивы, списки или множества записей. По мере появления новых приложений разделение программ и данных стало проблематичным. Приложениям требовалось дать данным поведение. Например, если данные представляли сложный объект, то методы поиска, сравнения и манипулирования данными становились специфичными для типов данных документов, изображений, звука или карта.

  
Рис. 3. Переход от традиционных баз данных, хранящих числа и символы, к объектно-реляционным базам данных, где каждая запись может содержать данные со сложным поведением. Это поведение может быть инкапсулировано в библиотеках классов, которые поддерживают новые типы. В этой модели система баз данных хранит и выбирает данные и обеспечивает связи между элементами данных, а библиотеки классов обеспечивают поведение этих элементов.

Традиционный подход заключался во встраивании типов данных прямо в систему баз данных. В языке SQL были добавлены новые типы данных для времени, временных интервалов и символьных строк с двухбайтовым представлением символов. Каждое из этих расширений было значительным достижением. Когда они были сделаны, результаты удовлетворили не всех. Например, в SQL тип данных «время» не позволяет представлять даты до Рождества Христова, а в символьных строках нельзя применять кодировку Unicode (универсальный набор символов почти для всех языков). Пользователи, желающие использовать Unicode или даты до нашей эры, должны определять свои собственные типы данных. Эти простые примеры, а также многие другие убедили сообщество баз данных в том, что системы баз данных должны разрешать прикладным специалистам реализовывать типы данных для своих прикладных областей. Географам следует иметь возможность реализации карт, специалистам в области текстов имеет смысл реализовывать индексацию и выборку текстов, специалистам по изображениям стоило бы реализовать библиотеки типов для работы с изображениями. Конкретным примером может служить распространенный объектный тип временных рядов. Вместо встраивания этого объекта в систему баз данных рекомендуется реализация соответствующего типа в виде библиотеки классов с методами для создания, обновления и удаления временных рядов. Дополнительные методы суммируют отклонения и интерполируют точки в рядах, сравнивают, комбинируют и вычитают временные ряды. После построения такой библиотеки классов она может быть «вставлена» в любую систему баз данных. Система баз данных будет хранить объекты этого типа и управлять данными (безопасность, параллельный доступ, восстановление и индексирование), но содержимым и поведением объектов временных рядов будет заведовать тип данных.

Люди из сообщества объектно-ориентированного программирования четко распознали проблему: для разработки типов данных требуется хорошая модель данных и унификация процедур и данных. Действительно, программы инкапсулируют данные и обеспечивают все методы для манипулирования данными. Исследователи, начинающие и установившиеся поставщики реляционных баз данных долго и упорно работали, начиная с 1985 года, чтобы либо заменить реляционную модель, либо объединить объектно-ориентированные и реляционные системы. В конце 1980-х на рынке появилось более десяти продуктов объектно-ориентированных баз данных, но заказчики не спешили принять эти системы. Тем временем традиционные поставщики старались расширить язык SQL, чтобы включить в него концепции объектно-ориентированного подхода, сохраняя преимущества реляционной модели.

Быстрое развитие Internet усиливает эти дебаты. Клиенты и серверы Internet строятся с использованием апплетов и «хелперов», которые сохраняют, обрабатывают и отображают данные того или иного типа. Пользователи вставляют эти апплеты в браузер или сервер. Общераспространенные апплеты управляют звуком, изображениями, текстами, видео, электронными таблицами, графами. Для каждого из ассоциированных с этими апплетами типов данных имеется библиотека классов. Настольные компьютеры и Web-браузеры являются распространенными источниками и приемниками большей части данных. Поэтому типы и объектные модели, используемые в настольных компьютерах, будут диктовать, какие библиотеки классов должны поддерживаться на серверах баз данных.

Подводя итог, заметим, что базы данных призваны хранить больше, чем только числа и текстовые строки. Они используются для хранения многих видов объектов, которые мы видим в World Wide Web, и связей между этими объектами. Различие между базой данных и остальной частью Web становится неясным.

Унификация процедур и данных расширяет традиционную вычислительную модель клиент-сервер в двух интересных направлениях: 1) активные базы данных и 2) потоки работ (workflow). Активные базы данных самостоятельно выполняют задачи при изменении базы данных. Идея состоит в том, что определенная пользователем процедура-триггер срабатывает, когда состояние базы данных переводит условие триггера в true. Используя язык процедур базы данных, проектировщики базы данных могут определять предусловия и триггерные процедуры. Например, если в базе данных товаров определен триггер повторного заказа, то база данных будет вызывать процедуру повторного заказа данного товара, когда его количество достигнет нижнего порога. Триггеры упрощают приложения, позволяя перенести логику из в приложений к данным. Механизм триггеров является мощным способом построения активных баз данных, являющихся самоуправляемыми.

Потоки работ обобщают типичную модель вычислений запрос-ответ. Поток задач – это сценарий задач, которые должны быть выполнены. Например, простое покупательское соглашение состоит из шестишагового потока работ для 1) запроса покупателя, 2) предложения цены, 3) подтверждения, 4) поставки, 5) выписки счета, 6) оплаты. Системы для составления сценариев, выполнения и управления потоками работ становятся общераспространенными.

Чтобы приблизиться к современному состоянию технологии управления данными, имеет смысл описать два крупных проекта управления данными, в которых используются предельные возможности сегодняшней технологии. Система Earth Observation System/Data Information System (EOS/DIS) разрабатывается агентством NASA и его подрядчиками для хранения всех спутниковых данных, которые начнут поступать со спутников серии «Миссия к планете Земля» в 1997 году. Объем базы данных, включающей данные от удаленных сенсорных датчиков, будет расти на 5 Тбайт в день (терабай – это миллион гигабайт). К 2007 году размер базы данных вырастит до 15 петабайт. Это в тысячу раз больше объема самых больших сегодняшних оперативных баз данных. NASA желает, чтобы эта база данных была доступна каждому в любом месте в любое время. Любой человек сможет производить поиск, анализ и визуализацию данных из этой базы данных. Для построения EOS/DIS потребуются наиболее развитые методы хранения, управления, поиска и визуализации данных. Большая часть данных будет обладать пространственными и временными характеристиками, так что для системы потребуются существенное развитие технологии хранения данных этих типов, а также библиотеки классов для различных научных наборов данных. Например, для этого приложения потребуется библиотека для определения снежного покрова, каталога растительных форм, анализа облачности и других физических свойств образов LandSat. Эта библиотека классов должна легко подключаться к менеджеру данных EOS/DIS.

Другим впечатляющим примером базы данных является возникающая всемирная библиотека. Многие ведомственные библиотеки открывают доступ к своим хранилищам в режиме online. Новая научная литература публикуется в режиме online. Такой вид публикации поднимает трудные социальные вопросы по поводу авторских прав и интеллектуальной собственности и заставляет решать глубокие технические проблемы. Пугают размеры и многообразие информации. Информация появляется на многих языках, во многих форматах данных и в громадных объемах. При применении традиционных подходов к организации такой информации (автор, тема, название) не используются мощности компьютеров для поиска информации по содержимому, для связывания документов и для группирования сходных документов. Обнаружение информации, нахождение требуемой информации в море документов, карт, фотографий, звука и видео представляет собой захватывающую и трудную проблему.

**3. Размышления и предсказания**

Достижения в области компьютерной аппаратуры сделали возможным переход от ручной бумажной обработки к современным средствам информационного поиска. Ожидается, что этот прогресс в мире аппаратуры будет продолжаться в будущем в течение многих лет.

Программное обеспечение управления данными развивалось параллельно с развитием аппаратных средств. Системы, ориентированные на записи и наборы, открыли дорогу реляционным системам, которые теперь эволюционизируют к объектно-реляционным системам. Эти нововведения представляют один из лучших примеров преобразования в продукты исследовательских прототипов. Реляционная модель, параллельные системы баз данных, активные базы данных и объектно-реляционные базы данных вышли в свет из академических и промышленных исследовательских лабораторий. Развитие технологии баз данных является хрестоматийным примером успешного сотрудничества академии и индустрии.

Недорогая аппаратура и простое для использования программное обеспечение сделали компьютеры доступными почти для каждого. Теперь просто и недорого создать Web-сервер или базу данных. Это делают миллионы людей. Эти пользователи ожидают, что компьютеры будут в состоянии сами себя автоматически проектировать и управлять. Эти пользователи не хотят быть операторами компьютеров. Они ожидают возможности добавления новых приложений почти без усилий: менталитет plug-and-play. Это видение распространяется от настольных компьютеров до наиболее мощных серверов. Пользователи ожидают появления автоматизированного управления с интуитивно понятными графическими интерфейсами для решения задач администрирования, обработки и проектирования. После построения и ввода в действие базы данных пользователи ожидают возможности применения простых и мощных средств для просмотра, поиска, анализа и визуализации данных. Эти требования выходят за пределы сегодняшних технологий.

Остаются нерешенными многие задачи управления данными, как технические, так и социальные. Электронный обмен данными и программное обеспечение добывания данных делают сравнительно простым для крупной организации отслеживание всех ваших финансовых транзакций. Но при выполнении этого кто-то сможет получить очень детальный профиль ваших интересов, путешествий и финансов. Не является ли это вторжением в частную жизнь? Действительно, это можно проделать почти для каждого в нашем развивающемся мире. Каковы последствия всего этого? Каковы должны быть правила защиты частной жизни и безопасности по отношению к оперативно доступным медицинским записям? Кому следует позволить просматривать ваши записи? Как будет работать охрана авторских прав, когда любой человек в любом месте имеет доступ к электронной копии документа? Киберпространство пересекает национальные границы. Каковы права и обязанности людей, действующих в киберпространстве?

Возможно, что и наши внуки через 50 лет все еще будут бороться с этими социальными проблемами. Технические задачи более легко поддаются решению. В сообществе баз данных имеется согласие по поводу основных задач и состава необходимых исследовательских работ. Каждые пять лет сообщество баз данных производит самооценку, в которой очерчивается класс первоочередных задач. В последней такой самооценке, называемой отчетом Lagunita II [8], выделяются следующие задачи:

* Определение моделей данных для новых типов (например, пространственных, темпоральных, графических) и их интеграция с традиционными системами баз данных.
* Масштабирование баз данных по размеру (до петабайт), пространственному размещению (распределенные) и многообразию (неоднородные).
* Автоматическое обнаружение тенденций данных, их структур и аномалий (интеллектуальный анализ данных).
* Интеграция (комбинирование) данных из нескольких источников.
* Создание сценариев и управление потоком работ (процессом) и данными в организациях.
* Автоматизация проектирования и администрирования баз данных.

Это трудные проблемы. Их решение сделают доступными для частных лиц и организаций новые компьютерные приложения. Эти системы позволят нам иметь доступ ко всей информации и анализировать ее из любого места в любое время. Этот облегченный доступ к информации изменит способы ведения научной деятельности, способы управления бизнесом, способы обучения и развлечения. Он и обогатит и усилит нас и грядущие поколения.

Возможно, наиболее сложной проблемой является понимание данных. Не так уж трудно обеспечить оперативный доступ к большей части данных – поскольку хранение данных в компьютерах стоит недорого и поскольку хранить данные в компьютерах удобно. Реальной проблемой, встающей перед нами, является такая организация этих огромных архивов данных, которая позволила бы людям легко находить требующуюся им информацию. Нахождение в большой базе данных структур, тенденций, аномалий и релевантной информации является одной из новых, наиболее впечатляющих областей управления данными. В действительности, я надеюсь, что компьютеры смогут уплотнять и обобщать для нас информацию, позволяя избежать тяжелого поиска самородков среди не относящихся к делу данных. Решение этой проблемы потребует вклада со стороны многих дисциплин.

**NoSQL** ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *not only SQL*, *не только SQL*), в [информатике](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) — термин, обозначающий ряд подходов, направленных на реализацию хранилищ [баз данных](https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), имеющих существенные отличия от моделей, используемых в традиционных [реляционных СУБД](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94) с доступом к данным средствами языка[SQL](https://ru.wikipedia.org/wiki/SQL).

Сама по себе идея нереляционных баз данных не нова, а использование нереляционных хранилищ началось ещё во времена первых компьютеров. Нереляционные базы данных процветали во времена [мэйнфреймов](https://ru.wikipedia.org/wiki/%D0%9C%D1%8D%D0%B9%D0%BD%D1%84%D1%80%D0%B5%D0%B9%D0%BC" \o "Мэйнфрейм), а позднее, во времена доминирования реляционных СУБД, нашли применение в специализированных хранилищах, например, иерархических [службах каталогов](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_%D0%BA%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%D0%BE%D0%B2). Появление же нереляционных СУБД нового поколения произошло из-за необходимости создания параллельных распределённых систем для высокомасштабируемых интернет-приложений, таких как [поисковые системы](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B8%D1%81%D0%BA%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0)[[2]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Tiwari.E2.80.942011.E2.80.94Chapter_1:_NoSQL:_What_It_Is_and_Why_You_Need_it_.3E_Definition_and_Introduction.E2.80.94-2).

В начале 2000-х годов [Google](https://ru.wikipedia.org/wiki/Google_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D1%8F)" \o "Google (компания)) построил свою высокомасштабируемую поисковую систему и приложения: [GMail](https://ru.wikipedia.org/wiki/GMail" \o "GMail), [Google Maps](https://ru.wikipedia.org/wiki/Google_Maps" \o "Google Maps), [Google Earth](https://ru.wikipedia.org/wiki/Google_Earth" \o "Google Earth) и т. п., решая проблемы масштабируемости и параллельной обработки больших объёмов данных. В результате была создана [распределённая файловая система](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D0%B0%D1%8F_%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0) и распределённая система координации, хранилище семейств колонок ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *column family store*), [среда выполнения](https://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F), основанная на алгоритме [MapReduce](https://ru.wikipedia.org/wiki/MapReduce" \o "MapReduce). Публикация компанией Google описаний этих технологий привела к всплеску интереса среди разработчиков [открытого программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), в результате чего был создан [Hadoop](https://ru.wikipedia.org/wiki/Hadoop" \o "Hadoop) и запущены связанные с ним проекты, призванные создать подобные Google технологии. Через год, в 2007 году, примеру Google последовал [Amazon.com](https://ru.wikipedia.org/wiki/Amazon.com), опубликовав статьи о высокодоступной базе данных [Amazon DynamoDB](https://ru.wikipedia.org/wiki/Amazon_DynamoDB" \o "Amazon DynamoDB)[[3]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Tiwari.E2.80.942011.E2.80.94.E2.80.944-6-3).

Поддержка гигантов индустрии менее чем за пять лет привела к широкому распространению технологий NoSQL (и подобных) для управления «большими данными», а к делу присоединились другие большие и маленькие компании, такие как: [IBM](https://ru.wikipedia.org/wiki/IBM), [Facebook](https://ru.wikipedia.org/wiki/Facebook" \o "Facebook), [Netflix](https://ru.wikipedia.org/wiki/Netflix" \o "Netflix), [EBay](https://ru.wikipedia.org/wiki/EBay" \o "EBay), [Hulu](https://ru.wikipedia.org/wiki/Hulu" \o "Hulu), [Yahoo!](https://ru.wikipedia.org/wiki/Yahoo!" \o "Yahoo!), со своими [проприетарными](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%BF%D1%80%D0%B8%D0%B5%D1%82%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5" \o "Проприетарное программное обеспечение) и открытыми решениями[[3]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Tiwari.E2.80.942011.E2.80.94.E2.80.944-6-3).

Традиционные СУБД ориентируются на требования [ACID](https://ru.wikipedia.org/wiki/ACID) к транзакционной системе: атомарность, согласованность, изолированность ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *isolation*), надёжность ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *durability*), тогда как в NoSQL вместо ACID может рассматриваться набор свойств BASE[[1]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94What_NoSQL_is_and_what_it_is_not.E2.80.94-1):

* базовая доступность ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *basic availability*) — каждый запрос гарантированно завершается (успешно или безуспешно).
* гибкое состояние ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *soft state*) — состояние системы может изменяться со временем, даже без ввода новых данных, для достижения согласования данных.
* согласованность в конечном счёте ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *eventual consistency*) — данные могут быть некоторое время рассогласованы, но приходят к согласованию через некоторое время.

Термин «BASE» был предложен Эриком Брюером, автором [теоремы CAP](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_CAP), согласно которой в распределённых вычислениях можно обеспечить только два из трёх свойств: согласованность данных, доступность или устойчивость к разделению[[1]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94What_NoSQL_is_and_what_it_is_not.E2.80.94-1).

Разумеется, системы на основе BASE не могут использоваться в любых приложениях: для функционирования биржевых и банковских систем использование транзакций является необходимостью. В то же время, свойства ACID, какими бы желанными они ни были, практически невозможно обеспечить в системах с многомиллионной веб-аудиторией, вроде [amazon.com](https://ru.wikipedia.org/wiki/Amazon.com)[[1]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94What_NoSQL_is_and_what_it_is_not.E2.80.94-1). Таким образом, проектировщики NoSQL-систем жертвуют согласованностью данных ради достижения двух других свойств из теоремы CAP[[4]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-4). Некоторые СУБД, например, [Riak](https://ru.wikipedia.org/wiki/Riak" \o "Riak), позволяют настраивать требуемые характеристики доступности-согласованности даже для отдельных запросов путём задания количества узлов, необходимых для подтверждения успеха транзакции.[[5]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-5)

Решения NoSQL отличаются не только проектированием с учётом масштабирования. Другими характерными чертами NoSQL-решений являются[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6)[[7]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94Why_NoSQL.3F.E2.80.94-7):

* Применение различных типов хранилищ[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6).
* Возможность разработки базы данных без задания [схемы](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6)[[7]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94Why_NoSQL.3F.E2.80.94-7).
* Использование [многопроцессорности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C)[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6).
* Линейная масштабируемость (добавление процессоров увеличивает производительность)[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6).
* Инновационность: «не только SQL» открывает много возможностей для хранения и обработки данных[[6]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-McCreary.2C_Kelly.E2.80.942013.E2.80.941.1._What_is_NoSQL.3F.E2.80.94-6).
* Сокращение времени разработки[[7]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94Why_NoSQL.3F.E2.80.94-7)
* Скорость: даже при небольшом количестве данных конечные пользователи могут оценить снижение времени отклика системы с сотен миллисекунд до миллисекунд [[7]](https://ru.wikipedia.org/wiki/NoSQL#cite_note-Vaish.E2.80.942013.E2.80.94Why_NoSQL.3F.E2.80.94-7)

Описание [схемы данных](https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) в случае использования NoSQL-решений может осуществляться через использование различных структур данных: [хеш-таблиц](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88-%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0), [деревьев](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_(%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)) и других.

В зависимости от [модели данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) и подходов к [распределённости](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D1%91%D0%BD%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85" \o "Распределённая база данных) и [репликации](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BF%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) можно выделить четыре типа хранилищ: «ключ-значение» (key-value store), документно-ориентированные (document store), хранилища семейств колонок (column database), графовые базы данных (graph database).