Информационные системы

Основной целью создания ИС является удовлетворение информационных потребностей пользователей путем предоставления необходимой им информации на основе хранимых данных. Потребность в информации как таковой не исчерпывает понятия информационных потребностей. Обычно в понятие информационных потребностей включают определенные требования к качеству информационного обслуживания и поведению системы в целом (производительность, актуальность и надежность данных, ориентация на пользователя и ряд других).

Под информационной системой понимается организационная совокупность технических и обеспечивающих средств, технологических процессов и кадров, реализующих функции сбора, обработки, хранения, поиска, выдачи и передачи информации.

Необходимость повышения производительности труда в сфере информационной деятельности приводит к тому, что в качестве внешних средств хранения и быстрого доступа к информации чаще всего используются средства вычислительной техники (цифровой и аналоговой) на основе компьютеров. Современные ИС - сложные комплексы аппаратных и программных средств, технологии и персонала, которые еще называют автоматизированными *информационными системами*. Структурно ИС включают в себя аппаратное (hardware), программное (software), коммуникационное (netware), промежуточного слоя (*middleware*), лингвистическое и *организационно-технологическое обеспечение*.

Аппаратное обеспечение ИС включает в себя широкий набор средств вычислительной техники, средства передачи данных, а также целый ряд специальных технических устройств (устройства графического отображения *информации*, аудио- и видеоустройства, средства речевого ввода и т.д.). *Аппаратное обеспечение* является основой любой ИС.

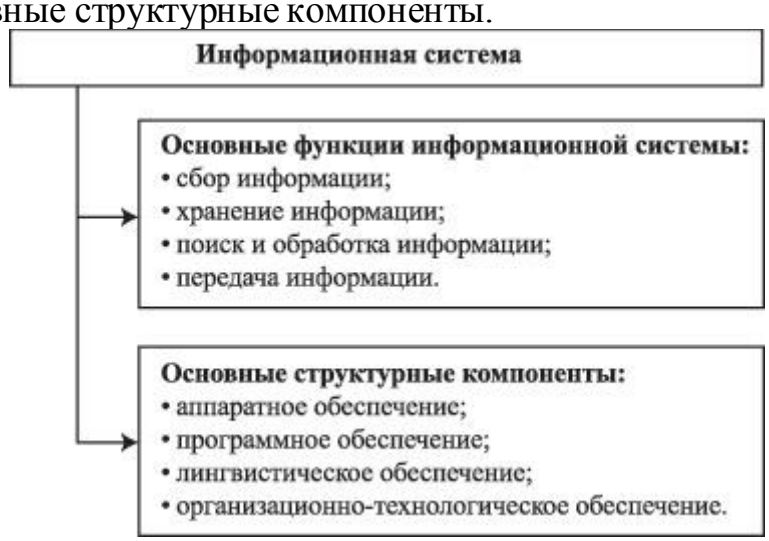
Коммуникационное (сетевое) обеспечение включает в себя комплекс аппаратных сетевых коммуникаций и программных средств поддержки коммуникаций в ИС. Оно имеет существенное значение при создании распределенных ИС и ИС на основе Интернета. При создании распределенных ИС огромную роль также играет *программное обеспечение* промежуточного слоя, состоящее из набора программных средств (служб и сервисов), которые управляют взаимодействием распределенных объектов в системе.

Программное обеспечение ИС обеспечивает реализацию функций ввода данных, их размещения на машиночитаемых носителях, модификации данных, доступ к данным, поддержку функционирования оборудования. *Программное обеспечение* можно разделить на системное (которое венчает процесс выбора аппаратно-программного решения, или платформы, как говорят в настоящее время) и пользовательское (которое применяется для решения задач удовлетворения потребностей пользователя в компьютерной среде, а именно, реализует бизнес-логику).

Лингвистическое обеспечение ИС предназначено для решения задач формализации смыслового содержания полнотекстовой и специальной информации для создания поискового образа данных (профиля). В классическом смысле обычно оно включает процедуры индексирования текстов, их классификацию и тематическую рубрикацию. Зачастую ИС, содержащие сложно-структурированную информацию, включают в себя *тезаурусы* терминов и понятий (средства поддержки метаданных). Сюда можно отнести и создание процессоров специализированных *формальных языков* конечных пользователей, например языков для манипулирования бухгалтерской информацией и т.д. Чаще всего работам по разработке *лингвистического обеспечения* не придается должного значения. Подобные упущения чаще всего ведут к неприятию пользователями самой системы и, как следствие, к ее закономерной гибели. Это относится в первую очередь к узко специализированным ИС.

По мере возрастания сложности и масштабов ИС важную роль начинает играть *организационно-технологическое обеспечение*, которое соединяет разнородные компоненты (аппаратуру, программы и персонал) в единую систему и обеспечивает процедуры ее управления и функционирования. Недооценка этой составляющей ИС чаще всего приводит к срыву сроков внедрения системы и вывода ее на производственные мощности.

На рисунке просуммированы в общих чертах функции ИС через ее основные структурные компоненты.



Классификация информационных систем.

Информационные системы классифицируются по разным признакам. Наиболее часто используемые способы классификации по масштабу, по сфере применения, способу организации.

По масштабу информационные системы подразделяют на следующие группы

Одиночные информационные системы реализуются на автономном персональном компьютере (сеть не используется). Такая система может содержать несколько простых приложений, связанных общим информационным фондом, и рассчитана на работу одного пользователя или группы пользователей, разделяющих во времени одно рабочее место. Подобные приложения создаются с помощью так называемых *настольных*, или *локальных* систем управления базами данных (СУБД). Среди локальных СУБД наиболее известными являются Clarion, Clipper, d Base, Microsoft Access и др.

Групповые информационные системы ориентированы на коллективное использование информации членами рабочей группы и чаще всего строятся на базе локальной компьютерной сети. При разработке таких приложений используются серверы баз данных (называемые также SQL - серверами) для рабочих групп. Существует довольно большое количество различных SQL – серверов как коммерческих, так и свободно распространяемых. Среди них наиболее известны такие серверы баз данных, как Oracle, Microsoft SQL Server, Inter Base, Sybase и др.

Корпоративные информационные системы являются развитием систем рабочих групп, они ориентированы на крупные компании и могут поддерживать территориально разнесенные узлы или сети. В основном они имеют иерархическую структуру из нескольких уровней. Для таких систем характерна архитектура клиент-сервер со специализацией серверов или же многоуровневая архитектура. При разработке таких систем могут использоваться те же серверы баз данных, что и при разработке групповых информационных систем. В крупных информационных системах наибольшее распространение получили серверы Oracle, DB2 и Microsoft Server.

Для групповых и корпоративных систем существенно повышаются требования к надежности функционирования и сохранности данных. Эти свойства обеспечиваются поддержкой целостности данных, ссылок и транзакций в серверах баз данных.

Классификация по сфере применения.

По сфере применения информационные системы подразделяются на четыре группы.

· системы обработки транзакций;

· системы поддержки принятия решений;

· информационно-справочные системы;

· офисные информационные системы.

Системы обработки транзакций, в свою очередь, по оперативности обработки данных разделяются на пакетные информационные системы и оперативные информационные системы. В информационных системах организационного управления преобладает режим оперативной обработки транзакций (On line Transaction Processing, OLTP) для отражения актуального состояния предметной области в любой момент времени, а пакетная обработка занимает весьма ограниченную часть.

Для систем OLTP характерен регулярный (возможно интенсивный) поток довольно простых транзакций, играющих роль заказов, платежей, запросов и т.п. Важными требованиями для них являются:

· высокая производительность обработки транзакций;

· гарантированная доставка информации при удаленном доступе к БД по телекоммуникациям.

Системы поддержки принятия решений (Decision Support System, DSS) представляют собой другой тип информационных систем, в которых с помощью довольно сложных запросов производится отбор и анализ данных в различных разрезах: временных, географических, по другим показателям.

Обширный класс *информационно-справочных систем* основан на гипертекстовых документах и мультимедиа. Наибольшее развитие такие информационные системы получили в Интернете.

Класс офисных информационных систем нацелен на перевод бумажных документов в электронный вид, автоматизацию делопроизводства и управление документооборотом.

Классификация по способу организации.

По способу организации групповые и корпоративные информационные системы подразделяются на следующие классы:

· системы на основе архитектуры файл-сервер;
· системы на основе архитектуры клиент-сервер;
· системы на основе многоуровневой архитектуры;
· системы на основе Интернет/ интранет – технологий.

В зависимости от класса задач используются различные прикладные информационные системы (ПИС). ПИС – производство профессиональной информации, связанной с определенной профессиональной деятельностью. Она обеспечивает сбор, хранение, обработку, поиск и выдачу информации необходимой в процессе принятия решений задач из любой предметной области. Задача ПИС помочь в анализе проблем и создание новых продуктов в интересах достижения поставленной цели. Компьютеры, оснащенные специализированными программными средствами, являются технической базой и инструментом для любой ПИС. В крупных организациях наряду с ПК в состав технической базы ПИС могут входить серверы, супер-ЭВМ, компьютерные системы.

Перечень прикладных информационных систем и их базовые определения.

Рассмотрим перечень основных прикладных информационных систем.

1. Системы автоматизированного проектирования (САПР).
2. Системы мультимедийной обработки.
3. Экспертные системы.
4. Системы научных исследований и экспериментов.
5. Корпоративные системы.
6. Системы обработки сигналов и изображений.
7. Системы реального времени.

8. Системы обучения.
9. Информационно-справочные системы.
10. Системы управления базами данных (СУБД).
11. Медицинские информационные системы.

Рассмотрим назначение некоторых из них.

Среди прикладных информационных технологий автоматизация проектирования занимает особое место. Использование системы автоматизированного проектирования (САПР) позволяет снизить стоимость и сократить сроки проектирования при улучшение качества проектных решений. Предприятия ведущие разработки без САПР оказываются неконкурентоспособными вследствие как больших материальных и временных затрат на проектирование, так и невысокого качества проектов. САПР содержит техническое, математическое, методическое и программное обеспечение. Существуют и информационные поддержки типа CASE и CALS – технологий (CASE – Computer Aided System Engineering; CALS – Continuousr Acquisition and Life Cycle Support).

Экспертные системы(ЭС), или «системы, основанные на знаниях», представляют собой программное обеспечение, анализирующее некоторую информацию на основе специальных механизмов представления знаний о предметной области и логического вывода. Экспертные системы разрабатываются для широкого спектра проблем диагностики, проектирования, планирования, управления, пронозирования и др. Использование ЭС во многом объясняется их способностью воспринимать знания специалистов в определенной предметной области, обеспечивать доступ и манипулирование ими, а также выдавать рекомендации при решении практических задач на уровне высококвалифицированного эксперта. В ЭС принять выделять четыре существенные компоненты: базу знаний, машину логического вывода, модуль извлечения знаний и систему объяснения.

Системы мультимедийной обработки. Здесь под мультимедиа понимают совокупность визуальных и аудиоэффектов воспроизводимых с помощью компьютера и управляемых интерактивными программами. Средства мультимедиа – это комплекс средств, позволяющих человеку общаться с компьютером, используя самые разные естественные для себя среды: звук, видео, графику, тексты, анимацию и т.д.

Средствами мультимедиа являются:

· системы речевого ввода и вывода информации (системы распознавания речи и системы синтеза речи);

· компьютерные средства обеспечения звуковых технологий (звуковые карты и акустические системы);

· компьютерные средства обеспечения видеотехнологий.

Корпоративной информационной системой будем называть совокупность специализированного программного обеспечения и

вычислительной аппаратной платформы, на которой установлено и настроено программное обеспечение.

Корпоративные информационные системы – это интегрированные системы управления территориально распределенной корпорацией, основанные на углубленном анализе данных, широком использовании систем информационной поддержки принятия решений, электронном документообороте и делопроизводстве. Неотъемлемой частью корпоративных информационных систем являются корпоративные сети. Известны корпоративные сети на основе ОС Windows Server 2000, на основе СОС Novell NetWare 5.1 и др.

В системах реального времени обработка информации происходит в реальном масштабе времени (РМВ). Режим, при котором организация обработки данных подчиняется темпу процессов вне систем обработки данных, называется обработкой в реальном масштабе времени. В системах управления реальными объектами, построенных на основе компьютеров, процесс управления сводится к решению фиксированного набора задач. К системам РМВ могут быть отнесены и высокопроизводительные КС.

Система управления базой данных – это специальный пакет программ, посредством которого реализуется централизованное управление базой данных и обеспечивается доступ к данным СУБД позволяет структурировать, систематизировать и организовывать данные для компьютерного хранения и обработки. Именно системы управления базами данных являются основной практически любой информационной системы. База данных – это датологическое представление информационной модели предметной области. Созданы специальные языки СУБД. Например, для управления реляционными базами данных может быть использован язык SQL (Structured Query Language) – структурированный язык запросов.

Медицинские информационные системы – разработка технологии для организационного управления и обработки данных медицинского характера, разработка медицинских приборов и систем.

В настоящее время успешное функционирование различных фирм, организаций и предприятий просто не возможно без развитой информационной системы, которая позволяет автоматизировать сбор и обработку данных. Обычно для хранения и доступа к данным, содержащим сведения о некоторой предметной области, создается база данных.

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

Под предметной областью принято понимать некоторую область человеческой деятельности или область реального мира, подлежащих изучению для организации управления и автоматизации, например, предприятие, вуз и.т.д.

Системы БД существуют и на малых, менее мощных компьютерах, и на больших, более мощных. На больших применяют в основном многопользовательские системы, на малых – однопользовательские.

Однопользовательская система (single-user system) – это система, в которой в одно и то же время к БД может получить доступ не более одного пользователя.

Многопользовательская система (multi-user system) - это система, в которой в одно и то же время к БД может получить доступ несколько пользователей.

Основная задача большинства многопользовательских систем – позволить каждому отдельному пользователю работать с системой как с однопользовательской.

Различия однопользовательской и многопользовательской систем – в их внутренней структуре, конечному пользователю они практически не видны.

Система баз данных содержит четыре основных элемента: ***данные, аппаратное обеспечение, программное обеспечение*** и ***пользователи***.

Данные в БД являются *интегрированными* и *общими*.

Интегрированные – значит, данные можно представить как объединение нескольких, возможно перекрывающихся, отдельных файлов данных. (Например, имеется файл, содержащий данные о студентах – фамилию, имя, отчество, дату рождения, адрес и т.д., а другой – о спортивной секции. Необходимые данные о студентах, посещающих секцию, можно получить путем обращения к первому файлу.)

Общие – значит, отдельные области данных могут использовать различные пользователи, т.е. каждый из этих пользователей может иметь доступ к одной и той же области данных, даже одновременно. (Например, одни и те же данные БД о студентах может одновременно использовать студенческий отдел кадров и деканат.)

К **аппаратному обеспечению** относятся:

- Накопители для хранения информации вместе с подсоединенными устройствами ввода-вывода, каналами ввода-вывода и т.д.
- Процессор (или процессоры) вместе с основной памятью, которая используется для поддержки работы программного обеспечения системы.

Между собственно данными и пользователями располагается уровень **программного обеспечения.** Ядром его является система управления базами данных (database management system – DBMS), или диспетчер БД (database manager).

Система управления базами данных (СУБД) - это комплекс программных и языковых средств, необходимых для создания БД,

поддержания их в актуальном состоянии и организации поиска в них необходимой информации.

Основная функция СУБД – это предоставление пользователю БД возможности работы с ней, не вникая в детали на уровне аппаратного обеспечения. Т.е. все запросы пользователя к БД, добавление и удаление данных, выборки, обновление данных – все это обеспечивает СУБД.

Иными словами, СУБД поддерживает пользовательские операции высокого уровня. Сюда включены и операции, которые можно выполнить с помощью языка SQL.

SQL - это специальные язык БД. Сейчас он поддерживается большинством СУБД, он является официальным стандартом языка для работы с реляционными системами. Название SQL вначале было аббревиатурой от Structured Query Language (язык структурированных запросов), сейчас название языка уже не считается аббревиатурой, т.к. функции его расширились и не ограничиваются только созданием запросов.

СУБД – это не единственный программный компонент системы, хотя и наиболее важный. Среди других – утилиты, средства разработки приложений, средства проектирования, генераторы отчетов и т.д.

Пользователей СБД можно разделить на три группы:

- ***Прикладные программисты***. Отвечают за написание прикладных программ, использующих БД. Для этих целей применимы различные языки программирования. Прикладные программы выполняют над данными стандартные операции – выборку, вставку, удаление, обновление – через соответствующий запрос к СУБД. Такие программы бывают простыми – *пакетной обработки*, или *оперативными приложениями* – для поддержки работы конечного пользователя.
- ***Конечные пользователи***. Работают с системами БД непосредственно через рабочую станцию или терминал. Конечный пользователь может получить доступ к БД, используя оперативное приложение или интегрированный интерфейс самой СУБД (такой интерфейс тоже является оперативным приложением, но встроенным). В большинстве систем есть хотя бы одно такое встроенное приложение – *процессор языка запросов* (или *командный интерфейс*). Язык SQL – пример языка запросов для БД. Кроме языка запросов в современных СУБД, как правило, есть *интерфейсы, основанные на меню и формах* – для непрофессиональных пользователей. Понятно, что командный интерфейс более гибок, содержит больше возможностей.
- ***Администраторы БД***. Отвечают за создание БД, технический контроль, обеспечение быстродействия системы, ее техническое обслуживание.

СУБД имеют свою **архитектуру**. В процессе разработки и совершенствования СУБД предлагались различные архитектуры, но самой удачной оказалась трехуровневая архитектура, предложенная исследовательской группой ANSI/SPARC американского комитета по

стандартизации ANSI (American National Standards Institute). Упрощенная схема архитектуры СУБД приведена на рис. 1.

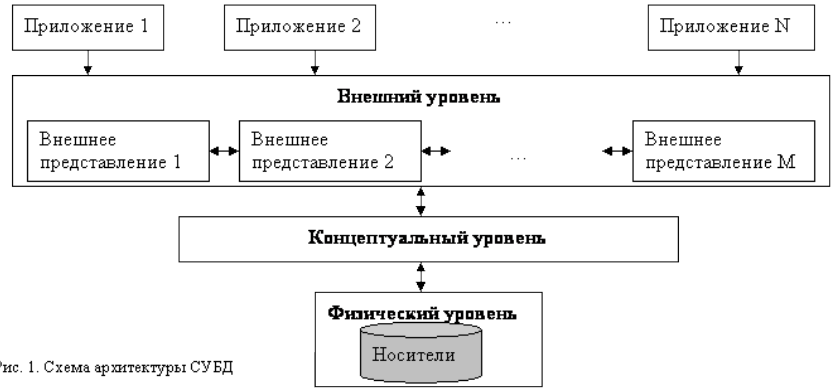


Рис. 1. Схема архитектуры СУБД

Внешний уровень – это уровень пользователя. По сути, это совокупность внешних представлений данных, которые обрабатывают приложения и какими их видит пользователь на экране. Это может быть таблица с отсортированными данными, с примененным фильтром, форма, отчет, результат запроса. Внешние представления взаимосвязаны, т.е. из одного внешнего представления можно получить другое.

Концептуальный уровень – центральный. Здесь БД представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями. Т.е. это обобщенная *модель* предметной области, для которой созданы БД. Можно сказать, что концептуальный уровень формируется при создании таблиц (определение их полей, типов, свойств), связей, а так же при заполнении таблиц.

Физический уровень – собственно данные, расположенные на внешних носителях.

Классификация моделей данных

Ядром любой БД является модель данных.

Модель данных – это совокупность структур данных и операций их обработки.

Т.к. СУБД имеет 3-х уровневую архитектуру, то понятие модели данных связано с каждым уровнем.

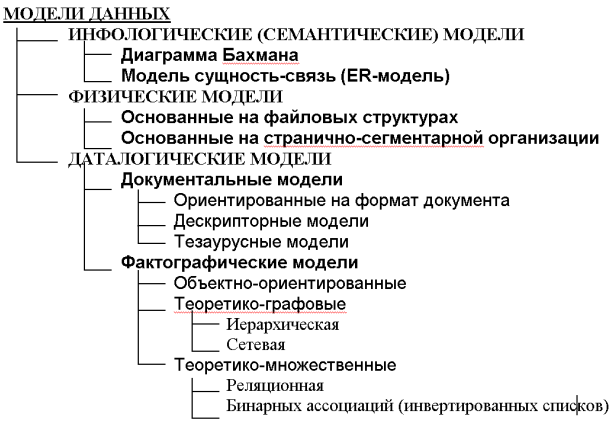
Физическая модель данных связана с организацией внешней памяти и структур хранения, используемых в данной операционной среде.

На *концептуальном* уровне модели данных наиболее важны для разработчиков БД, т.к. именно ими определяется тип СУБД.

Для *внешнего* уровня отдельных моделей данных нет, они лишь являются подсхемами концептуальных моделей данных.

Кроме моделей данных, соответствующих трем уровням архитектуры СУБД, существуют *предшествующие* им, не связанные с компьютерной реализацией. Они служат переходным звеном от реального мира к БД. Это класс инфологических (семантических) моделей.

Общая классификация моделей данных приведена на рисунке



Инфологические (семантические) модели данных используются на ранних стадиях проектирования БД.

Даталогические модели данных уже поддерживаются конкретной СУБД.

Физические модели данных связаны с организацией данных на носителях.

Документальные модели данных соответствуют слабоструктурированной информации, ориентированной на свободные форматы документов на естественном языке.

Модели данных, ***ориентированные на формат документа,*** связаны со стандартным общим языком разметки SGML (Standart Generaliset Markup Language), а также HTML, предназначенным для управления процессом вывода содержимого документа на экран.

Дескрипторные модели данных – самые простые, широко использовались раньше. В них каждому документу соответствует дескриптор – описатель, который имеет жёсткую структуру и описывает документ в соответствии с заранее определенными характеристиками.

Тезаурусные модели данных основаны на принципе организации словарей. Содержат языковые конструкции и принципы их взаимодействия в заданной грамматике. Эти модели используются, например, в системах-переводчиках.

Объектно-ориентированная модель перекликается с семантическими моделями данных. Принципы похожи на принципы объектно-ориентированных языков программирования. Структура таких моделей графически представима в виде дерева, узлами которого являются объекты. Свойства объектов описываются типом.

Объекты ***иерархической*** модели данных связанны иерархическими отношениями и образуют ориентированный граф. Основные понятия иерархических структур: уровень, узел (совокупность свойств данных, описывающих объект), связь.

В ***сетевой*** модели данных при тех же основных понятиях (уровень, узел, связь) каждый элемент может быть связан с любым другим элементом.

В ***реляционной*** модели данных данные представлены только в виде таблиц.

Базы данных можно разделить на базы данных *первого поколения*: иерархические, сетевые; *второго поколения*: реляционные; *третьего поколения*: объектно-ориентированные, обектно-реляционные.

Программы, с помощью которых пользователи работают с базой данных, называются *приложениями*. В общем случае с одной базой данных могут работать множество различных приложений. Например, если база данных моделирует некоторое предприятие, то для работы с ней может быть создано приложение, которое обслуживает подсистему учета кадров, другое приложение может быть посвящено работе подсистемы расчета заработной платы сотрудников, третье приложение работает как подсистемы складского учета, четвертое приложение посвящено планированию производственного процесса. При рассмотрении приложений, работающих с одной базой данных, предполагается, что они могут работать параллельно и независимо друг от друга, и именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, то учитывало все изменения в базе данных, вносимые другими приложениями.

Для поиска информации в базах данных используется информационно-поисковая система. Информационно-поисковая система опирается на базу данных, в которой осуществляется поиск нужных документов по заявкам пользователей.

Различают ***фактографические*** автоматизированные информационные системы (АИС), у которых базы данных составляются из форматированных (формализованных) записей, и ***документальные*** АИС, записями которых могут служить различные неформализованные документы (статьи, письма и т.п.). В фактографических АИС примером форматированных записей могут служить, скажем, записи об операциях по приему и выдаче денег в сберкассе; запись имеет четыре основных атрибута: дата, характер операции (принято, выдано), сумма, остаток вклада.

В качестве форматированной записи может рассматриваться кадровая анкета (личный листок по учету кадров). Правда, такие ее разделы, как «прежняя работа», «поездки за границу» и др. в обычной анкете не до конца формализованы и имеют переменную длину, поэтому при автоматизации этой задачи необходимы некоторые поправки. Обычно бывает целесообразно фиксировать максимальное количество позиций в каждом разделе и тем самым выравнивать длину записей (у многих записей при этом могут возникать позиции с пустым заполнением).

Основной задачей, решаемой в документальных АИС, является поиск документов по их содержанию. Документальная система по заданию пользователя выдает необходимые ему документы (книги, статьи, законы, патенты, отчеты и т.д.). В задании могут указываться сведения об искомых документах: автор, наименование, время издания, издательство и т.д.

Реляционные базы данных. Ограничения целостности

Американский математик Э.Ф.Кодд (E.F.Codd) в 1970 впервые сформулировал основные понятия и ограничения реляционной модели. Цели создания реляционной модели формулировались следующим образом:

- обеспечение более высокой степени независимости от данных. Прикладные программы не должны зависеть от изменений внутреннего представления данных, в частности от изменений организации файлов, переупорядочивания записей и путей доступа;
- создание прочного фундамента для решения семантических вопросов, а также проблем непротиворечивости и избыточности данных. В частности, в статье Кодда вводится понятие нормализованных отношений, т.е. отношений без повторяющихся групп;
- расширение языков управления данными за счет включения операций над множествами.

В статье, опубликованной в 1985 году, Э. Кодд сформулировал двенадцать правил, которым должна соответствовать настоящая реляционная БД. Они являются полуофициальным определением понятия **реляционная база данных.**

1. **Правило информации.** Вся информация в базе данных должна быть предоставлена исключительно на логическом уровне и только одним способом - в виде значений, содержащихся в таблицах.

2. **Правило гарантированного доступа.** Логический доступ ко всем и каждому элементу данных (атомарному значению) в реляционной базе данных должен обеспечиваться путём использования комбинации имени таблицы, первичного ключа и имени столбца. Правило указывает на роль первичных ключей при поиске информации в базе данных. Имя таблицы позволяет найти требуемую таблицу, имя столбца позволяет найти требуемый столбец, а первичный ключ позволяет найти строку, содержащую искомый элемент данных.

3. **Правило поддержки недействительных значений.** В настоящей реляционной базе данных должна быть реализована поддержка недействительных значений, которые отличаются от строки символов нулевой длины, строки пробельных символов, и от нуля или любого другого числа и используются для представления отсутствующих данных независимо от типа этих данных. Правило требует, чтобы отсутствующие данные можно было представить с помощью недействительных значений (NULL).

4. **Правило динамического каталога, основанного на реляционной модели.** Описание базы данных на логическом уровне должно быть представлено в том же виде, что и основные данные, чтобы пользователи, обладающие соответствующими правами, могли работать с ним с помощью того же реляционного языка, который они применяют для работы с основными данными. Правило гласит, что реляционная база данных должна

сама себя описывать. Другими словами, база данных должна содержать набор *системных таблиц,* описывающих структуру самой базы данных.

5. **Правило исчерпывающего подъязыка данных.** Реляционная система может поддерживать различные языки и режимы взаимодействия с пользователем (например, режим вопросов и ответов). Однако должен существовать по крайней мере один язык, операторы которого можно представить в виде строк символов в соответствии с некоторым четко определенным синтаксисом и который в полной мере поддерживает следующие элементы:

· определение данных;
· определение представлений;
· обработку данных (интерактивную и программную);
· условия целостности;
· идентификация прав доступа;
· границы транзакций (начало, завершение и отмена).

Правило требует, чтобы СУБД использовала язык реляционной базы данных, например SQL, хотя явно SQL в правиле не упомянут. Такой язык должен поддерживать все основные функции СУБД — создание базы данных, чтение и ввод данных, реализацию защиты базы данных и т.д.

6. **Правило обновления представлений.** Все представления, которые теоретически можно обновить, должны быть доступны для обновления. Правило касается *представлений,* которые являются виртуальными таблицами, позволяющими показывать различным пользователям различные фрагменты структуры базы данных. Это одно из правил, которые сложнее всего реализовать на практике.

7. **Правило добавления, обновления и удаления.** Возможность работать с отношением как с одним операндом должна существовать не только при чтении данных, но и при добавлении, обновлении и удалении данных. Правило акцентирует внимание на том, что базы данных по своей природе ориентированы на множества. Оно требует, чтобы операции добавления, удаления и обновления можно было выполнять над множествами строк. Это правило предназначено для того, чтобы запретить реализации, в которых поддерживаются только операции над одной строкой.

8. **Правило независимости физических данных.** Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при любых изменениях способов хранения данных или методов доступа к ним.

9. **Правило независимости логических данных.** Прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться нетронутыми при внесении в базовые таблицы любых изменений, которые теоретически позволяют сохранить нетронутыми содержащиеся в этих таблицах данные. Правила 8 и 9 означают отделение пользователя и прикладной программы от низкоуровневой реализации базы данных. Они утверждают, что конкретные способы реализации хранения или доступа,

используемые в СУБД, и даже изменения структуры таблиц базы данных не должны влиять на возможность пользователя работать с данными.

10. **Правило независимости условий целостности.** Должна существовать возможность определять условия целостности, специфические для конкретной реляционной базы данных, на подъязыке реляционной базы данных и хранить их в каталоге, а не в прикладной программе. Правило гласит, что язык базы данных должен поддерживать ограничительные условия, налагаемые на вводимые данные и действия, которые могут быть выполнены над данными.

11. **Правило независимости распространения.** Реляционная СУБД не должна зависеть от потребностей конкретного клиента. Правило гласит, что язык базы данных должен обеспечивать возможность работы с распределенными данными, расположенными на других компьютерных системах.

12. **Правило единственности.** Если в реляционной системе есть низкоуровневой язык (обрабатывающий одну запись за один раз), то должна отсутствовать возможность использования его для того, чтобы обойти правила и условия целостности, выраженные на реляционном языке высокого уровня (обрабатывающем несколько записей за один раз). Правило предотвращает использование других возможностей для работы с базой данных, помимо языка базы данных, поскольку это может нарушить ее целостность.

Коммерческие системы на основе реляционной модели данных начали появляться в конце 70-х – начале 80-х годов. Благодаря популярности реляционной модели многие нереляционные системы теперь обеспечиваются реляционным пользовательским интерфейсом, независимо от используемой базовой модели.

Кроме того, позже были предложены некоторые расширения реляционной модели данных, предназначенные для наиболее полного и точного выражения смысла данных, для поддержки объектно-ориентированных, а также для поддержки дедуктивных возможностей.

Реляционная модель основана на математическом понятии отношения**,** физическим представлением которого является таблица**.** Дело в том, что Кодд, будучи опытным математиком, широко использовал математическую терминологию, особенно из теории множеств и логики предикатов.

Отношение – это плоская таблица, состоящая из столбцов и строк.

В любой реляционной СУБД предполагается, что пользователь воспринимает базу данных как набор таблиц. Однако следует подчеркнуть, что это восприятие относится только к логической структуре базы данных, т.е. ко внешнему и концептуальному уровням. Подобное восприятие не относится к физической структуре базы данных, которая может быть реализована с помощью различных структур.

Атрибут - это поименованный столбец отношения.

В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение обычно имеет вид двумерной таблицы, в которой строки соответствуют отдельным записям, а столбцы - атрибутам. При этом атрибуты могут располагаться в любом порядке, независимо от их переупорядочивания, отношение будет оставаться одним и тем же, а потому иметь тот же смысл.

Элементами отношения являются кортежи, или строки, таблицы. **Кортеж** – это строка отношения. Кортежи могут располагаться в любом порядке, при этом отношение будет оставаться тем же самым, а значит, и иметь тот же смысл.

Степень отношения определяется количеством атрибутов, которое оно содержит.

Отношение только с одним атрибутом имеет степень 1 и называется *унарным* (unary) отношением (или 1-арным кортежем). Отношение с двумя атрибутами называется *бинарным* (binary), отношение с тремя атрибутами – *тернарным*(ternary), а для отношений с большим количеством атрибутов используется термин *n-арный*(*n*-ary). Определение степени отношения является частью заголовка отношения.

Количество содержащихся в отношении кортежей называется кардинальностью отношения. Эта характеристика меняется при каждом добавлении или удалении кортежей. Кардинальность является свойством тела отношения и определяется текущим состоянием отношения в произвольно взятый момент.

Альтернативная терминология. Терминология, используемая в реляционной модели, порой может привести к путанице, поскольку помимо предложенных терминов существует еще один. Отношение в нем называется таблицей, кортежи – записями (records), а атрибуты – полями (fields). Эта терминология основана на том факте, что физически СУБД может хранить каждое отношение в отдельном файле. В таблице показаны соответствия, существующие между упомянутыми выше группами терминов.

Вариант1	Вариант2
Отношение	Таблица
Кортеж	Запись
Атрибут	Поле

Фундаментальные свойства отношений (таблиц)

Отношение обладает следующими характеристиками:

- оно имеет имя, которое отличается от имен всех других отношений;
- каждая ячейка отношения содержит только атомарное (неделимое) значение;

- каждый атрибут имеет уникальное имя;
- значения атрибута берутся из одного и того же домена;
- порядок следования атрибутов не имеет никакого значения;
- каждый кортеж является уникальным, т.е. дубликатов кортежей быть не может;
- теоретически порядок следования кортежей в отношении не имеет никакого значения. (Однако практически этот порядок может существенно повлиять на эффективность доступа к ним.)

Поскольку каждая ячейка должна содержать только одно значение, то не допускается хранение в одной и той же ячейке двух номеров телефона одного и того же отделения компании. Иначе говоря, отношения не могут содержать повторяющихся групп. Об отношении, которое обладает таким свойством, говорят, что оно нормализовано, или находится в первой нормальной форме.

Имена столбцов, указанные в их верхней строке, соответствуют именам атрибутов отношения.

Большая часть свойств отношений происходит от свойств математических отношений реляционной алгебры.

Как уже говорилось, наиболее популярны реляционные модели данных. В соответствии с реляционной моделью данных данные представляются в виде совокупности таблиц, над которыми могут выполняться операции, формулируемые в терминах реляционной алгебры или реляционного исчисления.

В отличие от иерархических и сетевых моделей данных в реляционной модели операции над объектами имеют теоретико-множественный характер. Это дает возможность пользователям формулировать их запросы более компактно, в терминах более крупных агрегатов данных.

Рассмотрим терминологию, используемую при работе с реляционными базами данных.

Первичный ключ.

Первичным ключом называется поле или набор полей, однозначно идентифицирующих запись.

Нередко возможны несколько вариантов выбора первичного ключа. Например, в небольшой организации первичными ключами сущности "сотрудник" могут быть как табельный номер, так и комбинация фамилии, имени и отчества (при уверенности, что в организации нет полных тезок), либо номер и серия паспорта (если паспорта есть у всех сотрудников). В таких случаях при выборе первичного ключа предпочтение отдается наиболее простым ключам (в данном примере - табельному номеру). Другие кандидаты на роль первичного ключа называются альтернативными ключами.

Требования, предъявляемые к первичному ключу:

- уникальность – то есть в таблице не должно существовать двух или более записей с одинаковым значением первичного ключа;
- первичный ключ не должен содержать пустых значений.

При выборе первичного ключа рекомендуется выбирать атрибут, значение которого не меняется в течение всего времени существования экземпляра (в этом случае табельный номер предпочтительнее фамилии, так как ее можно сменить, вступив в брак).

По полям, которые часто используются при поиске и сортировке данных устанавливаются ***вторичные ключи***: они помогут системе значительно быстрее найти нужные данные. В отличие от первичных ключей поля для индексов (вторичные ключи) могут содержать неуникальные значения.

Первичные ключи используются для установления связей между таблицами в реляционной БД. В этом случае первичному ключу одной таблицы (родительской) соответствует ***внешний ключ*** другой таблицы (дочерней). Внешний ключ содержит значения связанного с ним поля, являющегося первичным ключом. Значения во внешнем ключе могут быть неуникальными, но не должны быть пустыми. Первичный и внешний ключи должны быть одинакового типа.

Связи между таблицами.

Записи в таблице могут зависеть от одной или нескольких записей другой таблицы. Такие отношения между таблицами называются *связями*. Связь определяется следующим образом: поле или несколько полей одной таблицы, называемое *внешним ключом,* ссылается на первичный ключ другой таблицы. Рассмотрим пример. Так как каждый заказ должен исходить от определенного клиента, каждая запись таблицы *Orders*(заказы) должна ссылаться на соответствующую запись таблицы *Customers* (клиенты). Это и есть связь между таблицами *Orders* и *Customers*. В таблице *Orders* должно быть поле, где хранятся ссылки на те или иные записи таблицы *Customers*.

Существует три типа связей между таблицами.

Один к одному — каждая запись родительской таблицы связана только с одной записью дочерней. Такая связь встречается на практике намного реже, чем отношение *один ко многим* и реализуется путем определения уникального внешнего ключа. Связь *один к одному* используют, если не хотят, чтобы таблица «распухала» от большого числа полей. Базы данных, в состав которых входят таблицы с такой связью не могут считаться полностью нормализованными.

Один ко многим — каждая запись родительской таблицы связана с одной или несколькими записями дочерней. Например, один клиент может

сделать несколько заказов, однако несколько клиентов не могут сделать один заказ. Связь о*дин ко многим* является самой распространенной для реляционных баз данных.

Многие ко многим — несколько записей одной таблицы связаны с несколькими записями другой. Например, один автор может написать несколько книг и несколько авторов — одну книгу. В случае такой связи в общем случае невозможно определить, какая запись одной таблицы соответствует выбранной записи другой таблицы, что делает неосуществимой физическую (на уровне индексов и триггеров) реализацию такой связи между соответствующими таблицами. Поэтому перед переходом к физической модели все связи "многие ко многим" должны быть переопределены (некоторые CASE-средства, если таковые используются при проектировании данных, делают это автоматически).Подобная связь между двумя таблицами реализуется путем создания третьей таблицы и реализации связи типа «один ко многим» каждой из имеющихся таблиц с промежуточной таблицей.

Для рассмотрения ссылочной целостности возьмем в качестве примера наиболее часто встречающуюся в базах данных связь один-ко-многим – см таблицы 2 и 3. Как можно заметить, дочерняя и родительская таблицы связаны между собой по общему полю «Товар». Назовем это поле полем связи.

Таблица 2. Таблица «Товары»

Товар	Ед изм	Цена
Сахар	кг	18
Макароны	кг	18
Куры	кг	90
Фанта	бут	20

Таблица 3. Таблица «Отпуск товаров»

Товар	Дата	Количество
Сахар	10.12.07.	100
Сахар	12.12.07.	200
Сахар	14.12.07	50
Макароны	10.12.07	1000
Макароны	12.12.07	500
Фанта	07.12.07	2000
Фанта	05.12.07	3000

Возможны два вида изменений, которые приведут к утере связей между записями в родительской и дочерней таблицах:

- изменение значения поля связи в записи родительской таблицы без изменения значений полей связи в соответствующих записях дочерней таблицы;
- изменение значения поля связи в одной из записей дочерней таблицы без соответствующего изменения значения полей связи в родительской и дочерней таблицах.

Рассмотрим первый случай. Если изменить значения поля «Товар» с «Сахар» на «Рафинад» в таблице «Товары», а в таблице «Отпуск товаров» значение поля связи «Сахар» оставить прежним. В результате получим:

- в дочерней таблице «Отпуск товаров» для товара «Рафинад» (таблица «Товары») нет сведений о его отпуске со склада;
- некоторые записи таблицы «Отпуск товаров» содержат сведения об отпуске товара «Сахар», о котором нет информации в таблице «Товары».

Рассмотрим второй случай. Пусть в одной из записей таблицы «Отпуск товаров» значение поля связи «Сахар» изменилось на «Рафинад». В результате:

- в дочерней таблице «Отпуск товаров» недостоверны сведения об отпуске со склада товара «Сахар» (таблица «Товары»);
- одна из записей таблицы «Отпуск товаров» содержит данные об отпуске товара «Рафинад», сведения о котором отсутствуют в таблице «Товары».

И в первом, и втором случае мы наблюдаем нарушение целостности базы данных; это означает, что хранящаяся в ней информация становится недостоверной.

СУБД обычно блокирует действия, которые нарушают целостность связей между таблицами, т.е. нарушают ссылочную целостность.

Когда говорят о ссылочной целостности, имеют в виду совокупность связей между отдельными таблицами во всей БД.

Нарушение хотя бы одной такой связи делает информацию в БД недостоверной.

Чтобы предотвратить потерю ссылочной целостности, используется *механизм каскадных изменений*. Он состоит в обеспечении следующих действий:

- при изменении поля связи в записи родительской таблицы следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;
- при удалении записи в родительской таблице следует удалить соответствующие записи в дочерней таблице.

Изменения или удаления в записях дочерней таблицы при одновременном изменении (удалении) записи родительской таблицы называются каскадными изменениями и каскадными удалениями.

Существует другая разновидность каскадного удаления: при удалении родительской записи в записях дочерних таблиц значения полей связи обнуляются. Эта разновидность применяется редко, т.к. дочерние таблицы в этом случае будут содержать избыточные данные, например, сведения о товаре, которого нет на складе.

Обычно для реализации ссылочной целостности в дочерней таблице создают внешний ключ, в который входят поля связи дочерней таблицы. Этот ключ для дочерней таблицы является первичным и поэтому по составу полей должен совпадать с, первичным ключом родительской таблицы или реже - с частью первичного ключа.