1. Введение в базы данных
   1. Основные понятия и определения

Стержневые идеи современных информационных технологий базируются на концепции *баз Данных.*

Согласно этой концепции, основой информационных технологий являются *Данные,* которые должны быть организованы в базы данных в целях адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей.

Одним из важнейших понятий в теории баз данных является понятие *информации.* Под *информацией* понимаются любые сведения о каком-либо событии, процессе, объекте.

*Данные —* это информация, представленная в определенном виде, позволяющем автоматизировать ее сбор, хранение и дальнейшую обработку человеком или информационным средством. Для компьютерных технологий данные — это информация в дискретном, фиксированном виде, удобная для хранения, обработки на ЭВМ, а также для передачи по каналам связи.

*База Данных* (БД) — именованная совокупность данных,

отражающая состояние объектов и их отношений в рассматриваемой предметной области, или иначе БД — это совокупность взаимосвязанных данных при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений в определенной предметной области. БД состоит из множества связанных файлов.

*Система управления базами Данных* (СУБД) — совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями.

*Автоматизированная информационная система* (АИС) — это система, реализующая автоматизированный сбор, обработку, манипулирование данными, функционирующая на основе ЭВМ и других технических средств и включающая соответствующее программное обеспечение (ПО) и персонал. В дальнейшем в этом качестве будет использоваться термин *информационная система* (ИС), который подразумевает понятие автоматизированная.

Каждая ИС в зависимости от ее назначения имеет дело с той или иной частью реального мира, которую принято называть *предметной областью* (ПрО) *системы.* Выявление ПрО — это необходимый начальный этап разработки любой ИС. Именно на этом этапе определяются информационные потребности всей совокупности пользователей будущей системы, которые, в свою очередь, предопределяют содержание ее базы данных.

*Банк Данных* (БнД) является разновидностью ИС. БнД — это система специальным образом организованных данных: баз данных, программных, технических, языковых, организационно-методических средств, предназначенных для обеспечения централизованного накопления и коллективного многоцелевого использования данных.

Под *задачами обработки данных* обычно понимается специальный класс решаемых на ЭВМ задач, связанных с видом, хранением, сортировкой, отбором по заданному условию и группировкой записей однородной структуры.

Отдельные программы или комплекс программ, реализующие автоматизацию решения прикладных задач обработки данных, называются *приложениями.* Приложения, созданные средствами СУБД, относят к *приложениям СУБД.* Приложения, созданные вне среды СУБД с помощью систем программирования, использующих средства доступа к БД, к примеру, Delphi или Visual Studio, называют *внешними приложениями.*

* 1. Современное состояние технологий баз данных

Кратко сформулируем основные современные принципы организации баз данных.

* Значительная часть современных СУБД способна работать на компьютерах различной архитектуры под управлением разных операционных систем.
* Подавляющее большинство современных СУБД обеспечивают поддержку полной реляционной модели данных, обеспечивая целостность категорий и целостность на уровне ссылок.
* Современные СУБД для определения данных и манипуляции ими опираются на принятые стандарты в области языков, а при обмене данными между различными СУБД базируются на существующих технологиях по обмену информацией.
* Многие существующие СУБД относятся к так называемым сетевым

СУБД, которые предназначены для поддержки

многопользовательского режима работы с базой данных и поддержки возможности децентрализованного хранения данных.

* Такие СУБД имеют развитые средства администрирования баз данных и средства защиты хранимой в них информации.
* Подобные СУБД имеют средства подключения клиентских приложений.
* Современные СУБД характеризуются опытами применения концепции фундаментальной идеи объектно-ориентированного подхода, способствующей повышению уровня абстракции баз данных, являющейся перспективным этапом на пути развития технологий баз данных.

Информационные системы, созданные средствами технологии баз данных, иногда принято называть *банками Данных (БнД).*

БнД включает в себя:

* технические средства;
* одну или несколько БД;

- СУБД;

* словарь или каталог данных;
* администратора;
* вычислительную систему;
* обслуживающий персонал.

Схематично это выглядит так, как показано на рис. 1.1.

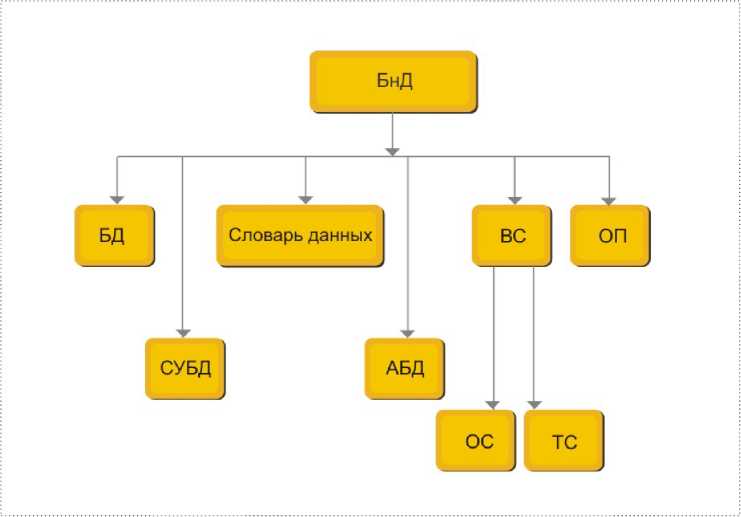


Рис. 1.1. Банк данных

Дадим краткие определения новым составляющим этой схемы.

*Словарь или каталог Данных* служит для централизованного накопления и описания ресурса данных. Он содержит описание ПрО, сведения о структуре БД, о связях между элементами БД. Словарь данных можно рассматривать как часть самой базы данных.

*Администратор БД (АБД) —* человек или группа лиц, которые принимают решения. Основные функции АБД:

* участие в разработке БД;
* контроль правильности функционирования БД.

*Вычислительная система (ВС) —* включает программные (ПС) и аппаратные средства (ТС).

*Обслуживающий персонал (ОП) —* это лица, прямыми

обязанностями которых является создание и поддержание корректного функционирования банка данных. Они ответственны за работу БнД и прикладного программного обеспечения. К обслуживающему персоналу относятся: разработчики и администраторы базы данных, аналитики, программисты.

* 1. Базы данных

БД, как правило, создается как общий ресурс всего предприятия, где данные являются *интегрированными и общими.* Под понятием *интегрированные* данные подразумевается возможность представить базу данных как объединение нескольких отдельных файлов данных. Под понятием *общие* данные подразумевается возможность использования отдельных областей данных в БД несколькими различными пользователями для разных целей.

В базе данных информация должна быть организована так, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, так как чрезмерная избыточность данных влечет за собой ряд негативных последствий. Главные из них:

* увеличение объема информации, а значит, потребность в дополнительных ресурсах для хранения и обработки дополнительных объемов данных;
* появление ошибок при вводе дублирующей информации, нарушающих целостность базы данных и создающих противоречивые данные.

БД содержит не только данные, всесторонне характеризующие деятельность самой организации, фирмы, процесса или другой предметной области, но и описания этих данных. Информацию о данных принято называть *"метаданными",* т. е. "данными о данных". В совокупности описания всех данных образуют *словарь Данных.*

В БД должны храниться данные, логически связанные между собой. Для того чтобы данные можно было связать между собой, и связать так, чтобы эти связи соответствовали реально существующим в данной предметной области, последнюю подвергают детальному анализу, выделяя сущности или объекты. Сущность или объект — это то, о чем необходимо хранить информацию. Сущности имеют некоторые характеристики, называемые атрибутами, которые тоже необходимо сохранять в БД. Атрибуты по своей внутренней структуре могут быть простыми, а могут быть сложными. Простые атрибуты могут быть представлены простыми типами данных. Различного рода графические изображения, являющиеся атрибутами сущностей, — это пример сложного атрибута. Определив сущности и их атрибуты, необходимо перейти к выявлению связей, которые могут существовать между некоторыми сущностями. Связь — это то, что объединяет две или более сущностей. Связи между сущностями также являются частью данных, и они также должны храниться в базе данных.

Если все это: сущности, атрибуты сущностей и связи между сущностями определено, то схема базы данных может выглядеть примерно так, как представлено на рис. 1.2. На нем показан пример схемы базы данных, которую можно назвать ФАКУЛЬТЕТ.

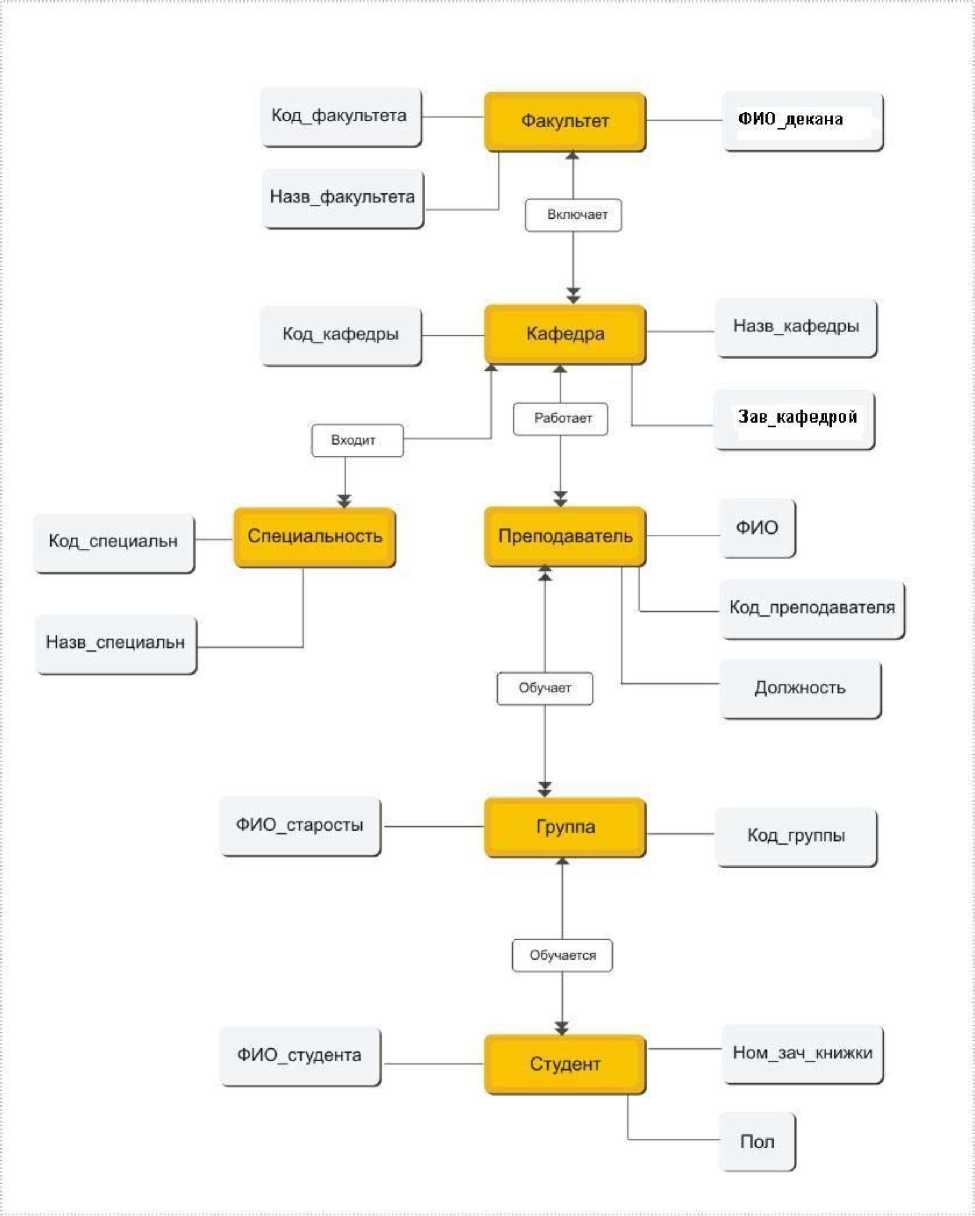


Рис. 1.2. Пример ER-диаграммы базы данных ФАКУЛЬТЕТ

Схема, которая называется ER-диаграммой (Entity-Relationship), состоит из следующих компонентов:

* шести сущностей, которые изображены прямоугольниками, каждый из которых имеет свои атрибуты, помещенные в овалы, а в нижеприведенном списке они перечислены в скобках рядом с именем сущностей:

**ФАКУЛЬТЕТ: (Код\_факультета, Назв\_факультета, ФИО\_декана); КАФЕДРА: (Код\_кафедры, Назв\_кафедры, Зав\_кафедрой); СПЕЦИАЛЬНОСТЬ: (Код\_специальности, Назв\_специальности); ПРЕПОДАВАТЕЛЬ: (Код\_преподавателя, ФИО, Должность); ГРУППА: (Код\_группы, ФИО\_старосты);**

**СТУДЕНТ: (Ном\_зач\_книжки, ФИО, Пол);**

* пяти связей, которые обозначены стрелками и связывают те сущности, на которые они направлены:

связь **ВКЛЮЧАЕТ** показывает, что на факультете несколько кафедр;

связь **ВХОДИТ** изображает, что одна и та же кафедра готовит специалистов по нескольким специальностям;

связь **РАБОТАЕТ** определяет то, что на кафедре работает ряд преподавателей;

связь **ОБУЧАЕТ** с двойными стрелками в обоих направлениях поясняет тот факт, что один и тот же преподаватель преподает в разных группах, а одна и та же группа занимается с разными преподавателями;

связь **ОБУЧАЕТСЯ** определяет, что каждая группа включает в себя ряд студентов.

Из представленной диаграммы понятно, что данные обладают определенной структурой. Для выявления этой структуры база данных должна пройти процесс проектирования.

Проектируемая БД должна обладать определенными свойствами. Назовем основные свойства БД.

*Целостность.* В каждый момент времени существования БД сведения, содержащиеся в ней, должны быть непротиворечивы. Целостность БД достигается вследствие введения ограничений целостности, в частности, к ним относятся ограничения, связанные с нормализацией БД.

*Восстанавливаемость.* Данное свойство предполагает возможность восстановления БД после сбоя системы или отдельных видов порчи системы.

*Безопасность.* Безопасность БД предполагает защиту данных от преднамеренного и непреднамеренного доступа, модификации или разрушения. Применяется запрещение несанкционированного доступа, защита от копирования и криптографическая защита.

*Эффективность.* Свойство эффективности обычно понимается как:

* минимальное время реакции на запрос пользователя;
* минимальные потребности в памяти;
* сочетание этих параметров.
  1. Системы управления базами данных

Итак, как уже не один раз упоминалось, СУБД — это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ.

По степени универсальности различаются два класса СУБД — системы общего назначения и специализированные системы.

*СУБД общего назначения* не ориентированы на какую-либо конкретную предметную область или на информационные потребности конкретной группы пользователей. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторой модели ЭВМ в определенной операционной обстановке. СУБД общего назначения обладает средствами настройки на работу с конкретной БД в условиях конкретного применения.

В некоторых ситуациях СУБД общего назначения не позволяют добиться требуемых проектных и эксплуатационных характеристик (производительность, занимаемый объем памяти и прочее). Тем не менее создание *специализированных СУБД* весьма трудоемкий процесс и для того, чтобы его реализовать, нужны очень веские основания.

В процессе реализации своих функций СУБД постоянно взаимодействует с базой данных и с другими прикладными программными продуктами пользователя, предназначенными для работы с данной БД и называемыми приложениями.

Для того чтобы СУБД успешно справлялась со своими задачами, она должна обладать определенными возможностями.

Можно дать следующую обобщенную характеристику возможностям современных СУБД.

1. СУБД включает язык определения данных, с помощью которого можно определить базу данных, ее структуру, типы данных, а также средства задания ограничений для хранимой информации. В многопользовательском варианте СУБД этот язык позволяет формировать представления как некоторое подмножество базы данных, с поддержкой которых пользователь может создавать свой взгляд на хранимые данные, обеспечивать дополнительный уровень безопасности данных и многое другое.
2. СУБД позволяет вставлять, удалять, обновлять и извлекать информацию из базы данных посредством языка управления данными.
3. Большинство СУБД могут работать на компьютерах с разной архитектурой и под разными операционными системами, причем на работу пользователя при доступе к данным практически тип платформы влияния не оказывает.
4. Многопользовательские СУБД имеют достаточно развитые средства администрирования БД.
5. СУБД предоставляет контролируемый доступ к базе данных с помощью:

* системы обеспечения безопасности, предотвращающей несанкционированный доступ к информации базы данных;
* системы поддержки целостности базы данных,

обеспечивающей непротиворечивое состояние хранимых данных;

* системы управления параллельной работой приложений, контролирующей процессы их совместного доступа к базе данных;
* системы восстановления, позволяющей восстановить базу данных до предыдущего непротиворечивого состояния, нарушенного в результате аппаратного или программного обеспечения.

1. Архитектура СУБД
   1. Трехуровневая архитектура базы данных

Одним из важнейших аспектов развития СУБД является идея отделения логической структуры БД и манипуляций данными, необходимыми пользователям, от физического представления, требуемого компьютерным оборудованием.

Одна и та же БД в зависимости от точки зрения может иметь различные уровни описания. По числу уровней описания данных, поддерживаемых СУБД, различают одно-, двух- и трехуровневые системы. В настоящее время чаще всего поддерживается трехуровневая архитектура описания БД (рис. 2.1), с тремя уровнями абстракции, на которых можно рассматривать базу данных.

Такая архитектура включает:

* внешний уровень, на котором пользователи воспринимают данные, где отдельные группы пользователей имеют свое представление (ПП) на базу данных;
* внутренний уровень, на котором СУБД и операционная система воспринимают данные;
* концептуальный уровень представления данных, предназначенный для отображения внешнего уровня на внутренний уровень, а также для обеспечения необходимой их независимости друг от друга; он связан с обобщенным представлением пользователей.

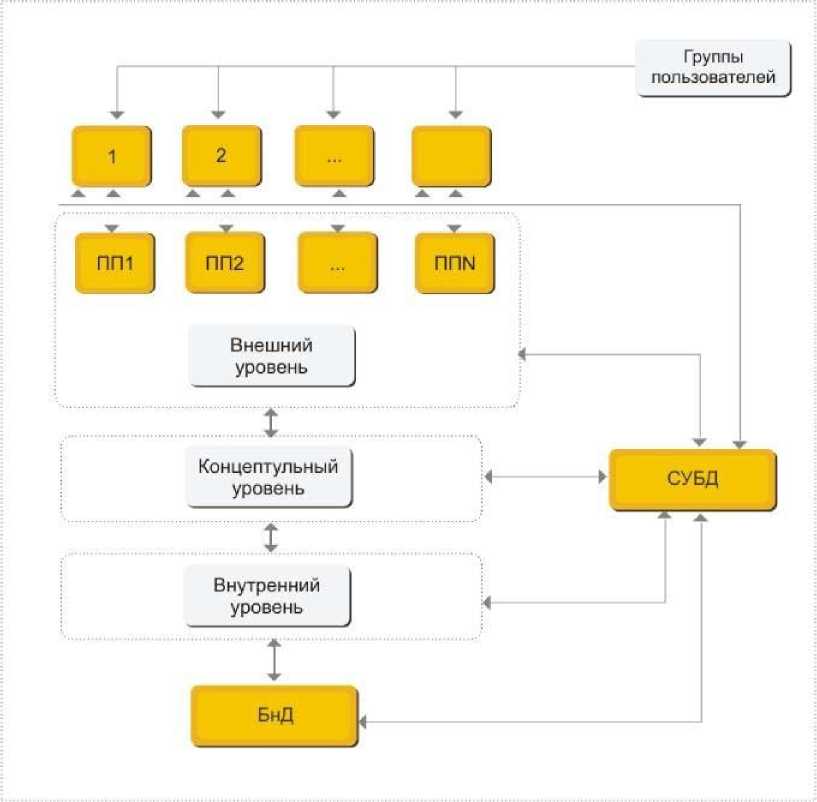


Рис. 2.1. Трехуровневая архитектура СУБД

Описание структуры данных на любом уровне называется *схемой.* Существует три различных типа схем базы данных, которые определяются в соответствии с уровнями абстракции трехуровневой архитектуры. На самом высоком уровне имеется несколько внешних схем или подсхем, которые соответствуют разным представлениям данных. На концептуальном уровне описание базы данных называют *концептуальной схемой,* а на самом низком уровне абстракции — *внутренней схемой.*

Основным назначением трехуровневой архитектуры является обеспечение независимости от данных. Суть этой независимости заключается в том, что изменения на нижних уровнях никак не влияют на верхние уровни. Различают два типа независимости от данных: логическую и физическую.

*Логическая независимость от Данных* означает полную защищенность внешних схем от изменений, вносимых в концептуальную схему. Такие изменения концептуальной схемы, как добавление или удаление новых сущностей, атрибутов или связей, должны осуществляться без необходимости внесения изменений в уже существующие внешние схемы для других групп пользователей.

*Физическая независимость от Данных* означает защищенность концептуальной схемы от изменений, вносимых во внутреннюю схему. Такие изменения внутренней схемы, как использование различных файловых систем или структур хранения, разных устройств хранения, модификация индексов или хеширование, должны осуществляться без необходимости внесения изменений в концептуальную или внешнюю схемы.

Далее рассмотрим каждый из трех названных уровней.

*Внешний уровень —* это пользовательский уровень. Пользователем может быть программист, или конечный пользователь, или администратор базы данных. Представление базы данных с точки зрения пользователей называется *внешним представлением.* Каждая группа пользователей выделяет в моделируемой предметной области, общей для всей организации, те сущности, атрибуты и связи, которые ей интересны. Эти частичные или переопределенные описания БД для отдельных групп пользователей или ориентированные на отдельные аспекты предметной области называют *подсхемой.*

*Концептуальный уровень* является промежуточным уровнем в трехуровневой архитектуре и обеспечивает представление всей информации базы данных в абстрактной форме. Описание базы данных на этом уровне называется концептуальной схемой, которая является результатом концептуального проектирования.

Концептуальное проектирование базы данных включает анализ информационных потребностей пользователей и определение нужных им элементов данных. Таким образом, *концептуальная схема —* это единое логическое описание всех элементов данных и отношений между ними, логическая структура всей базы данных. Для каждой базы данных имеется только одна концептуальная схема.

Концептуальная схема должна содержать:

* сущности и их атрибуты;
* связи между сущностями;
* ограничения, накладываемые на данные;
* семантическую информацию о данных;
* обеспечение безопасности и поддержки целостности данных.

*Внутренний уровень* является третьим уровнем архитектуры БД. Внутреннее представление не связано с физическим уровнем, так как физический уровень хранения информации обладает значительной индивидуальностью для каждой системы.

На нижнем уровне находится внутренняя схема, которая является полным описанием внутренней модели данных. Для каждой базы данных существует только одна внутренняя схема.

Внутренняя схема описывает физическую реализацию базы данных и предназначена для достижения оптимальной производительности и обеспечения экономного использования дискового пространства. На внутреннем уровне хранится следующая информация:

* распределение дискового пространства для хранения данных и индексов;
* описание подробностей сохранения записей (с указанием реальных размеров сохраняемых элементов данных);
* сведения о размещении записей;
* сведения о сжатии данных и выбранных методах их шифрования.

СУБД отвечает за установление соответствия между всеми тремя типами схем разных уровней, а также за проверку их непротиворечивости.

Ниже внутреннего уровня находится *физический уровень,* который контролируется операционной системой, но под руководством СУБД. Физический уровень учитывает, каким образом данные будут представлены в машине. Он обеспечивает физический взгляд на базу данных: дисководы, физические адреса, индексы, указатели и т. д. За этот уровень отвечают проектировщики физической базы данных, которые работают только с известными операционной системе элементами. Область их интересов: указатели, реализация последовательного распределения, способы хранения полей внутренних записей на диске. Однако функции СУБД и операционной системы на физическом уровне не вполне четко разделены и могут варьироваться от системы к системе.

* 1. Функции СУБД

**Управление данными во внешней памяти**

Данная функция предоставляет пользователям возможности выполнения самых основных операций, которые осуществляются с данными, — это сохранение, извлечение и обновление информации. Она включает в себя обеспечение необходимых структур внешней памяти как для хранения данных, непосредственно входящих в БД, так и для служебных целей, например для ускорения доступа к данным.

**Управление транзакциями**

*Транзакция —* это последовательность операций над БД, рассматриваемых СУБД как единое целое. Транзакция представляет собой набор действий, выполняемых с целью доступа или изменения содержимого базы данных. Примерами простых транзакций может служить добавление, обновление или удаление в базе данных сведений о некоем объекте. Сложная же транзакция образуется в том случае, когда в базу данных требуется внести сразу несколько изменений. Инициализация транзакции может быть вызвана отдельным пользователем или прикладной программой.

**Восстановление базы данных**

Одним из основных требований к СУБД является надежность хранения данных во внешней памяти. Под надежностью хранения понимается то, что СУБД должна быть в состоянии восстановить последнее согласованное состояние БД после любого аппаратного или программного сбоя. Обычно рассматриваются два возможных вида аппаратных сбоев:

* мягкие сбои, которые можно трактовать как внезапную остановку работы компьютера (например, аварийное выключение питания);
* жесткие сбои, характеризуемые потерей информации на носителях внешней памяти.

Поддержание надежности хранения данных в БД требует избыточности хранения данных, причем та часть данных, которая используется для восстановления, должна храниться особо надежно. Наиболее распространенным методом поддержания такой избыточной информации является ведение журнала изменений БД.

**Поддержка языков БД**

Для работы с базами данных используются специальные языки, называемые языками баз данных.

В современных СУБД обычно поддерживается единый интегрированный язык, содержащий все необходимые средства для работы с БД, начиная от ее создания, и обеспечивающий базовый пользовательский интерфейс с базами данных. Стандартным языком наиболее распространенных в настоящее время реляционных СУБД является язык SQL (Structured Query Language — язык структурированных запросов). Язык SQL позволяет определять схему реляционной БД и манипулировать данными.

**Словарь данных**

Одной из основополагающих идей рассмотренной выше трехуровневой архитектуры является наличие интегрированного системного каталога с данными о схемах, пользователях, приложениях и т. д. Системный каталог, который еще называют словарем данных, является, таким образом, хранилищем информации, описывающей данные в базе данных. Предполагается, что каталог доступен как пользователям, так и функциям СУБД. Обычно в словаре данных: содержится следующая информация:

* имена, типы и размеры элементов данных;
* имена связей;
* накладываемые на данные ограничения поддержки целостности;
* имена пользователей, которым предоставлено право доступа к данным;
* внешняя, концептуальная и внутренняя схемы и отображения между ними;
* статистические данные, например частота транзакций и счетчики обращений к объектам базы данных.

**Управление параллельным доступом**

Одна из основных целей создания и использования СУБД заключается в том, чтобы множество пользователей могло осуществлять параллельный доступ к совместно обрабатываемым данным. Параллельный доступ сравнительно просто организовать, если все пользователи выполняют только чтение данных, поскольку в этом случае они не могут помешать друг другу. Однако когда два или больше пользователей одновременно получают доступ к базе данных, конфликт с нежелательными последствиями легко может возникнуть, например, если хотя бы один из них попытается обновить данные.

СУБД должна гарантировать, что при одновременном доступе к базе данных многих пользователей подобных конфликтов не произойдет.

**Управление буферами оперативной памяти**

СУБД обычно работают с БД значительного размера. Понятно, что если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти. Практически единственным способом реального увеличения этой скорости является буферизация данных в оперативной памяти. В развитых СУБД поддерживается собственный набор буферов оперативной памяти с собственной дисциплиной замены буферов.

**Контроль доступа к данным**

СУБД должна иметь механизм, гарантирующий возможность доступа к базе данных только санкционированных пользователей и защищающий ее от любого несанкционированного доступа.

В современных СУБД поддерживается один из двух широко распространенных подходов к вопросу обеспечения безопасности данных: избирательный подход или обязательный подход.

В большинстве современных систем предусматривается избирательный подход, при котором некий пользователь обладает различными правами при работе с разными объектами. Значительно реже применяется альтернативный, обязательный подход, где каждому объекту данных присваивается некоторый классификационный уровень, а каждый пользователь обладает некоторым уровнем допуска.

**Поддержка целостности данных**

Термин *целостность* используется для описания корректности и непротиворечивости хранимых в БД данных. Реализация поддержки целостности данных предполагает, что СУБД должна содержать сведения о тех правилах, которые нельзя нарушать при работе с данными, и обладать инструментами контроля за тем, чтобы данные и их изменения соответствовали заданным правилам.

* 1. Языки баз данных

В СУБД поддерживается несколько специализированных по своим функциям подъязыков. Их можно разбить на две категории:

* язык определения данных БД — ЯОД {DDL — Data Definition Language);
* язык манипулирования данными— ЯМД (DML — Data Manipulation , Language).

1. Язык определения данных

*Язык определения Данных —* описательный язык, с помощью которого описывается предметная область: именуются объекты, определяются их свойства и связи между объектами. Он используется главным образом для определения логической структуры БД.

Схема базы данных, выраженная в терминах специального языка определения данных, состоит из набора определений. Язык ЯОД используется как для определения новой схемы, так и для модификации уже существующей.

Результатом компиляции ЯОД — операторов является набор таблиц, хранимый в системном каталоге, в котором содержатся метаданные — т. е. данные, которые включают определения записей, элементов данных, а также другие объекты, представляющие интерес для пользователей или необходимые для работы СУБД. Перед доступом к реальным данным СУБД обычно обращается к системному каталогу.

1. Языки манипулирования данными

*Язык манипулирования Данными* содержит набор операторов манипулирования данными, т. е. операторов, позволяющих заносить данные в БД, удалять, модифицировать или выбирать существующие данные.

Множество операций над данными можно классифицировать следующим образом:.

1. операции селекции;
2. действия над данными:

■ включение — ввод экземпляра записи в БД с установкой его связей;

* удаление — исключение экземпляра записи из БД с установкой новых связей;
* модификация — изменение содержимого экземпляра записи и коррекция связей при необходимости.

Языки манипулирования данными делятся на два типа. Это разделение обусловлено коренным различием в подходах к работе с данными, а следовательно, различием в базовых конструкциях в работе с данными.

Первый тип — это процедурный ЯМД.

Второй тип — это декларативный (непроцедурный) ЯМД.

К процедурным языкам манипулирования данными относятся и языки, поддерживающие операции реляционной алгебры, которую основоположник теории реляционных баз данных Э. Ф. Кодд ввел для управления реляционной базой данных. Реляционная алгебра — это процедурный язык обработки реляционных таблиц, где в качестве операндов выступают таблицы в целом.

Декларативные языки предоставляют пользователю средства, позволяющие указать лишь то, какие данные требуются. Решение вопроса о том, как их следует извлекать, берет на себя процессор данного языка, работающий с целыми наборами записей.

Реляционные СУБД обычно включают поддержку непроцедурных языков манипулирования данными — чаще всего это бывает язык структурированных запросов SQL или язык запросов по образцу QBE.

В настоящее время нормой является поддержка декларативного языка SQL, в основе которого лежит реляционное исчисление, также введенное Э Коддом. Этот язык стал стандартом для языков реляционных баз данных, что позволяет использовать один и тот же синтаксис и структуру команд при переходе от одной СУБД к другой

Следует отметить, что язык SQL имеет сразу два компонента: язык DDL (ЯОД) для описания структуры базы данных, и язык DML (ЯМД) для выборки и обновления данных.

Другим широко используемым языком обработки данных является язык QBE, который заслужил репутацию одного из самых простых способов извлечения информации из базы данных. Особенно это ценно для пользователей, не являющихся профессионалами в этой области Язык предоставляет графические средства создания запросов на выборку данных с использованием шаблонов Ответ на запрос также представляет собой графическую информацию

Часть непроцедурного языка ЯМД, которая отвечает за извлечение данных, называется *языком запросов* Язык запросов можно определить как высокоуровневый узкоспециализированный язык, предназначенный для удовлетворения различных требований по выборке информации из базы данных.

* 1. Архитектура многопользовательских СУБД
     1. Модели двухуровневой технологии "клиент — сервер"

Систему баз данных можно рассматривать как систему, где осуществлено распределение процесса выполнения по принципу взаимодействия двух программных процессов, один из которых в этой модели называется "клиентом", а другой, обслуживающий клиента, — *сервером* (машина, хранящая базы данных). Клиентский процесс запрашивает некоторые услуги, а серверный процесс обеспечивает их выполнение. При этом предполагается, что один серверный процесс может обслужить множество клиентских процессов (рис. 2.2).



Клиент 1 4

>

Клиент 3

>

Клиент 2

Рис. 2.2. Структура системы БД с выделением клиентов и сервера

*Сервер* в простейшем случае — это собственно СУБД. Он поддерживает все основные функции СУБД и предоставляет полную поддержку на внешнем, концептуальном и внутреннем уровнях.

*Клиенты —* это различные приложения, которые выполняются над СУБД.

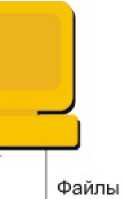
Обычно в приложении выделяются следующие группы функций:

* функции ввода и отображения данных;
* прикладные функции, определяющие основные алгоритмы решения задач приложения;
* функции обработки данных внутри приложения,
* функции управления информационными ресурсами;
* служебные функции, играющие роль связок между функциями первых четырех групп.

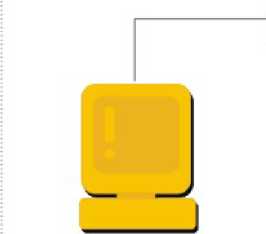
Если все пять компонентов приложения распределяются только между двумя процессами, которые выполняются на двух платформах: наклиенте и на сервере, то такая модель называется *двухуровневой.* Она имеет несколько основных разновидностей. Рассмотрим их.

**Файловый сервер**

Модель файлового сервера называется моделью *удаленного управления Данными.* Данная модель предполагает следующее распределение функций - на клиенте располагаются почти все части приложения: презентационная часть приложения, прикладные функции, а также функции управления информационными ресурсами. Файловый сервер содержит файлы, необходимые для работы приложений и самой СУБД и поддерживает доступ к файлам (рис. 2.3).

Файл-сервер

База данных

- <

| Локальная |
| --- |
| сеть |
| .J |

Команды

Клиент 1

Клиент 3

Клиент 2

Рис. 2.3. Модель файлового сервера

Поскольку передача файлов представляет собой длительную процедуру, такой подход характеризуется значительным сетевым трафиком, что может привести к снижению производительности всей системы в целом.

Помимо этого недостатка использование файлового сервера несет еще и другие:

* на каждой рабочей станции должна находиться полная копия СУБД;
* управление параллельностью, восстановлением и целостностью усложняется, поскольку доступ к одним и тем же файлам могут осуществлять сразу несколько экземпляров СУБД;
* узкий спектр операций манипулирования данными, который определяется только файловыми командами;

■ защита данных осуществляется только на уровне файловой системы.

Основное достоинство этой модели, заключается в том, что в ней уже осуществлено разделение монопольного приложения на два взаимодействующих процесса. При этом сервер может обслуживать множество клиентов, обращающихся к нему с запросами.

**Модель удаленного доступа к данным**

В модели удаленного доступа база данных также хранится на сервере. На сервере же находится и ядро СУБД. На клиенте располагаются части приложения, поддерживающие функции ввода и отображения данных и прикладные функции.

Клиент обращается к серверу с запросами на языке SQL. Структура модели удаленного доступа приведена на рис. 2.4.

Сервер с СУБД

Результаты запроса

База данных

<

|  |  |
| --- | --- |
| сеть |  |
|  | |

SQL-запрос

Локальная

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | J | . \_> |  |
| Клиент 1 | Клиент 2 | Клиент 3 | |

Рис. 2.4. Модель удаленного доступа

Сервер принимает и обрабатывает запросы со стороны клиентов, проверяет полномочия пользователей, гарантирует соблюдение ограничений целостности, выполняет обновление данных, выполняет запросы и возвращает результаты клиенту, поддерживает системный каталог, обеспечивает параллельный доступ к базе данных и ее восстановление. К тому же резко уменьшается загрузка сети, так как по ней от клиентов *к* серверу передаются не файловые команды, а запросы на SQL, и их объем существенно меньше. В ответ на запросы клиент получает только данные, соответствующие запросу, а не блоки файлов, как в модели файлового сервера.

Тем не менее, данная технология обладает и рядом недостатков:

* запросы на языке SQL при интенсивной работе клиентских приложений могут существенно загрузить сеть;
* презентационные и прикладные функции приложения должны быть повторены для каждого клиентского приложения;
* сервер в этой модели играет пассивную роль, поэтому функции управления информационными ресурсами должны выполняться на клиенте.

**Модель сервера баз данных**

Технологию "клиент — сервер" поддерживают большинство современных СУБД: Informix, Ingres, Sybase, Oracle, MS SQL Server. В основу данной модели добавлен механизм хранимых процедур и механизм триггеров.

Механизм *хранимых процедур* позволяет создавать подпрограммы, работающие на сервере и управляющие его процессами.

Таким образом, размещение на сервере хранимых процедур означает, что прикладные функции приложения разделены между клиентом и сервером. Трафик обмена информацией между клиентом и сервером резко уменьшается.

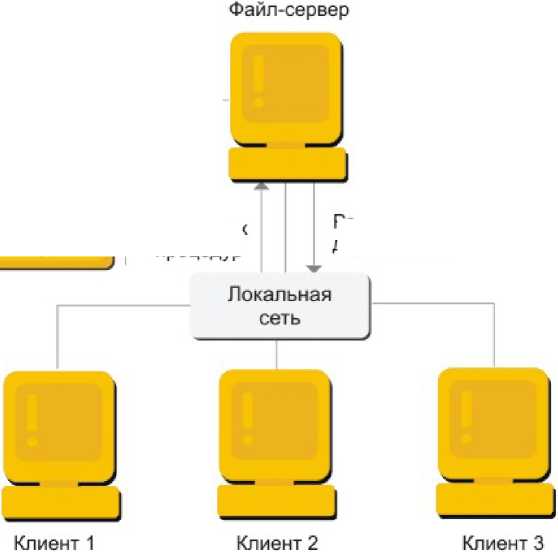
Централизованный контроль целостности базы данных в модели сервера баз данных выполняется с использованием механизма триггеров. Триггеры также являются частью БД.

*Триггер —* это особый тип хранимой процедуры, реагирующий на возникновение определенного события в БД. Он активизируется при попытке изменения данных — при операциях добавления, обновления и удаления. Триггеры определяются для конкретных таблиц БД.

Внедрение триггеров незначительно влияет на производительность сервера и часто используется для усиления приложений, выполняющих многокаскадные операции в БД.

В данной модели (рис. 2.5) сервер является активным, потому что не только клиент, но и сам сервер, используя механизм триггеров, может быть инициатором обработки данных в БД. Поскольку функции клиента облегчены переносом части прикладных функций на сервер, он в этом случае называется "тонким".

При всех положительных качествах данной модели у нее все же есть один недостаток — очень большая загрузка сервера.



Триггеры

Рис. 2.5. Модель сервера БД

Результаты для вывода

Вызов хранимых процедур

>

Хранимые процедуры

База данных

j

* + 1. Сервер приложений. Трехуровневая модель

Эта модель является расширением двухуровневой модели и в ней вводится дополнительный промежуточный уровень между клиентом и сервером. Архитектура трехуровневой модели приведена на рис. 2.6.

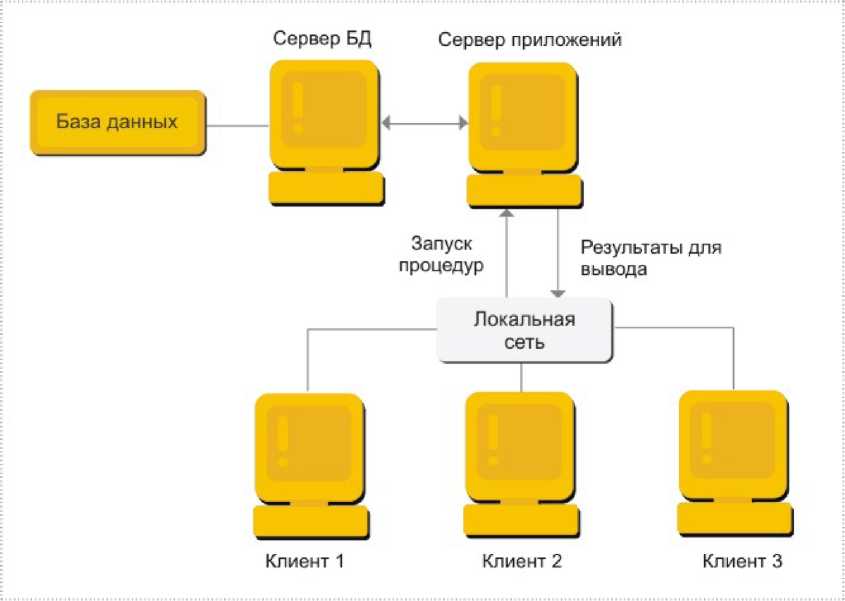


Рис. 2.6. Архитектура трехуровневой модели

Такая архитектура предполагает, что на *клиенте* располагаются: функции ввода и отображения данных, включая графический пользовательский интерфейс, локальные редакторы, коммуникационные функции, которые обеспечивают доступ клиенту в локальную или глобальную сеть.

*Серверы баз Данных* в этой модели занимаются исключительно функциями управления информационными ресурсами БД: обеспечивают функции создания и ведения БД, поддерживают целостность БД, осуществляют функции создания резервных копий БД и восстановления БД после сбоев, управления выполнением транзакций и так далее.

*Промежуточному уровню,* который может содержать один или несколько серверов приложений, выделяются общие не загружаемые функции для клиентов: наиболее общие прикладные функции клиента, функции, поддерживающие сетевую доменную операционную среду, каталоги с данными, функции, обеспечивающие обмен сообщениями и поддержку запросов.

Преимущества трехуровневой модели наиболее заметны в тех случаях, когда клиенты выполняют сложные аналитические расчеты над базой данных.