1. Основные виды информации

Информация -любые данные или сведения, которые кого-либо интересуют, например, сообщение о каких-либо событиях, о чьей-либо деятельности и т. п.

Виды информации:

1) Графическая или изобразительная — первый вид, для которого был реализован способ хранения информации об окружающем мире в виде наскальных рисунков, а позднее в виде картин, фотографий, схем, чертежей на бумаге, холсте, мраморе и др. материалах, изображающих картины реального мира;
2) Звуковая — мир вокруг нас полон звуков и задача их хранения и тиражирования была решена с изобретение звукозаписывающих устройств в 1877 г. ее разновидностью является музыкальная информация — для этого вида был изобретен способ кодирования с использованием специальных символов, что делает возможным хранение ее аналогично графической информации;
3) Текстовая — способ кодирования речи человека специальными символами — буквами, причем разные народы имеют разные языки и используют различные наборы букв для отображения речи; особенно большое значение этот способ приобрел после изобретения бумаги и книгопечатания;
4) Числовая — количественная мера объектов и их свойств в окружающем мире; особенно большое значение приобрела с развитием торговли, экономики и денежного обмена; аналогично текстовой информации для ее отображения используется метод кодирования специальными символами — цифрами, причем системы кодирования (счисления) могут быть разными;
5) Видеоинформация — способ сохранения «живых» картин окружающего мира, появившийся с изобретением кино.

2. Определение, состав БД

База данных– поименованная совокупность взаимосвязанных данных, управляемых специальной системой, называемой системой управления базой данных (СУБД).

СУБД– совокупность специальных языковых и программных средств, облегчающих пользователям выполнение всех операций, связанных с организацией хранения данных, их корректировкой и доступом к ним.

3. Уровни моделей представления баз данных

Аналитический уровень представления модели данных дает четкое понимание рассматриваемых в предметной области данных и объектов, которые ими описываются. Основу данного уровня представления модели данных составляют объекты, их атрибутивный состав и связи между ними. Аналитическое представление модели данных формируется при выполнении анализа предметной области и является базовой моделью для дальнейшего проектирования и реализации базы данных.

Концептуальный уровень представления модели базы данных отражает абстрактное представление данных предметной области в структурах, отражающих атрибутивный состав каждого элемента (объекта) модели и особенности связей между ними.

Логический уровень представления модели базы данных является расширением концептуальной модели и дополняется комплексом дополнительных элементов, а именно: типы данных, ограничения, умолчания и т.д. Логический уровень является более полным представлением модели базы данных, но сравнению с концептуальным представлением и формируется с целью перевести модели из плоскости предметной области в плоскость будущей базы данных.

Даталогический уровень представления модели базы данных отражает особенности реализации структуры базы данных, учитывая требования выбранной системы управления базами данных (СУБД). Дополнительно к представлению структуры базы данных на даталогическом уровне указываются правила ограничений ссылочной целостности данных, определяющие возможные действия с данными при выполнении отдельных операций (добавление, изменение, удаление). Переходя с логического на даталогический уровень, разработчик базы данных меняет используемую терминологию, а именно: от термина "Сущность" выполняется переход к термину "Таблица", от термина "Атрибут" — к термину "Поле" ("Колонка") и т.д. Этот переход обусловлен необходимостью рассматривать модель в терминах СУБД, которая работает на физическом уровне.

Физический уровень модели базы данных является расширением даталогического уровня, представляя описание правил обработки данных с учетом особенностей выбранной СУБД. В зависимости от используемого инструмента моделирования базы данных степень углубления на физический уровень моделирования может быть различной. В общем виде физический уровень моделирования предполагает описание структур данных, но используемым в базе данных представлениям, шаблоны программных кодов для хранимых процедур и триггеров. Фактически физический уровень представляется программным уровнем представления базы данных.

Внутренний уровень представляется реализацией базы данных в СУБД и формируется на основе физической модели базы данных. Реализация всех элементов базы данных на внутреннем уровне позволяет разработчику выполнить не только структурирование и описание правил работы базы данных, но и внести основные базовые сведения в базу данных, настраивая ее на начальную работу пользователя.

Внешний уровень представляет реализацию базы данных с учетом особенностей доступности отдельных объектов и сведений конкретным пользователям и их группам. Обычно это реализуется через организацию ролей базы данных, где определяются возможности выполнения операций (добавление, изменение, удаление, выборка), получения отдельных данных по представлениям, доступности взаимодействия с таблицами и полями (колонками) и т.д.

4. Классификация БД

По характеру хранимой информации:

1) Документальные - единицей хранения является какой-либо документ (например, текст закона или статьи), и пользователю в ответ на его запрос выдается либо ссылка на документ, либо сам документ, в котором он может найти интересующую его информацию.
2) Фактографические - хранится информация об интересующих пользователя объектах предметной области в виде «фактов» (например, биографические данные о сотрудниках, данные о выпуске продукции производителями и т.п.). В ответ на запрос пользователя выдается требуемая информация об интересующем его объекте (объектах) или сообщение о том, что искомая информация отсутствует в БД.

По структуре:

1) Иерархические - модель данных представляет собой древовидную (иерархическую) структуру;
   Особенности:
   a) имеется один главный объект и остальные - подчиненные объекты, находящиеся на разных уровнях иерархии;
   b) каждый объект-потомок связан только с одним объектом-предком вышележащего уровня иерархии;
   c) связи между объектами одного уровня не допускаются;
   d) между объектами двух уровней могут поддерживаться только связи «один ко многим»
2) Сетевые - являющиеся обобщением иерархической за счет допущения объектов, имеющих более одного предка. На связи между объектами в сетевых моделях данных не накладывается никаких ограничений.
3) Реляционные - все данные организованы в виде таблиц, между которыми установлены отношения
   Поле - столбец таблицы, заголовок которого определяет имя поля;
   Запись - это строка в таблице, содержащая все атрибуты, относящиеся к объекту.
   Особенности:
   a) каждый столбец таблицы содержит данные одного типа;
   b) каждый столбец имеет уникальное имя;
   c) в таблице нет одинаковых строк.

5. Понятия схемы, подсхемы и схемы хранения

Схема базы данных - это описание базы данных в терминах конкретной модели данных. (Описание концептуального и соответствующего ему физического представления) – структура.

Подсхема базы данных - это описание структуры данных прикладного программиста.

6. Основные понятия документальных БД

Фактографические системы хранят хорошо структурированные сведения (факты). Соответственно и запросы к ним носят более четкий (определенный) характер. Например, запрос к БД, содержащей сведения о сотрудниках предприятия, может быть таким: найти должность, оклад и телефон сотрудника Иванова.

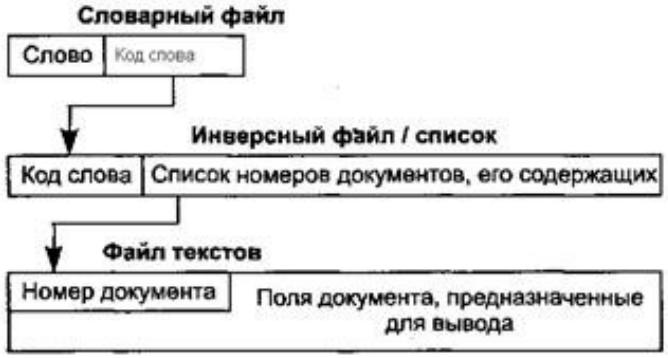
Документальные системы хранят не факты, а документы, содержащие эти факты. Соответственно наш запрос о сотруднике может выглядеть следующим образом: найти документы, содержащие сведения о должности, окладе и телефоне сотрудника Иванова.

Иными словами, запись документальной базы данных — это документ (обычно большого размера), который задается как набор в общем случае необязательных полей (например, аннотаций, глав, разделов, подразделов и т.д.), для каждого из которых оп-ределены имя и тип.

Поисковый критерий (критерий поиска документов) может включать в себя разные слова, причем пользователь может потребовать, чтобы заданное слово встречалось в названии документа, аннотации, введении или в каком-то конкретном параграфе.

Документальная БД включает в себя как минимум три области хранения данных, представляемые из-за своего большого размера, как правило, в виде файлов операционной системы (в действительности их всегда больше):

- файл словаря, устанавливающий соответствие между словом, встречающимся в БД, и его кодом;
- инверсный (инвертированный, обратный) список, содержащий для каждого слова БД список документов, его содержащих, используется при текстовом поиске;
- текстовый файл, содержащий собственно документы, используется при выдаче (просмотре) документов.



7. Основные понятия гипертекстовых БД

Гипертекстовая база данных - это набор текстовых файлов, написанных на языке HTML, который определяет форму представления информации (разметка) и структуру связей этих файлов (гипертекстовые ссылки).

Гипертекстовая база данных – текстовая база данных, записи которой содержат связи с другими записями, позволяющими компоновать ансамбли записей на основе их логической связанности.

Гипертекстовая база данных состоит из двух типов объектов, представляющих информацию: узлов и дуг устанавливающих явные смысловые и структурные связи между ними.

Использование термина «информационная единица» подчеркивает возможность предельно широкой интерпретации материала, составляющего содержание узла.

Основная идея гипертекстовых систем заключается в концепции автоматически поддерживаемых связей между различными объектами информации (информационные единицы). Поддержка таких связей позволяет организовывать «нелинейные» информационные структуры. Существует много

различных определений гипертекстовой системы, отражающих те или иные аспекты последней. В качестве примера можно привести определение, данное гипертекстовой системе В.Л. Эпштейном:

«…гипертекстовой системой называется информационная система, способная хранить информацию в виде электронного текста, позволяющая устанавливать электронные связи между любыми «информационными единицами», хранящимися в ее памяти, и вызывать их на экран монитора «простым нажатием кнопки».

8. Основные понятия мультимедийных БД

Мультимедийные базы данных

- Запросы по содержимому, обновления
- Управление параллельным выполнением операций, восстановление
- Используемое программное обеспечение: объектно-ориентированные или объектно-реляционные СУБД
- Пример: музей Эрмитаж – поиск QBIC по цвету и композиции (http://www.hermitagemuseum.org/fcgi-

Мультимедийные типы данных

- Текст
- Графика
- Звук
- Видео

Текст

- Присутствует в большинстве мультимедийных приложений; дополняет/поясняет нетекстовые форматы данных
- Визуальное разнообразие достигается шрифтами
- Самый компактный (при хранении) тип данных

Аудио

- Все более популярный тип данных
- Множество форматов (wav, cd, mp3, au, aiff, qt, ra, wma, …)
- Оцифрованное аудио имеет относительно большой размер (одна секунда занимает десятки килобайт)
- Используется сжатие (коэффициент сжатия mp3 - 12:1)
- Более компактное представление аудио-данных: синтезируемая музыка в формате MIDI, MPEG-4SA(Structured Audio)

Статичные растровые изображения

- Черно-белые/градации серого/цветные
- Одностраничное изображение в хорошем разрешении занимает несколько мегабайт
- Множество графических форматов (bmp, gif, tiff, jpeg, pcx, png, …)
- Формат JPEG (текущая версия - JPEG-2000): коэффициент сжатия обычно больше 10

Цифровое видео

- Последовательность кадров (фреймов) (= статичных изображений)
- Требует много дискового пространства
- Коэффициент сжатия более высокий, чем у статичных изображений (мало отличий между последовательными фреймами)
- Скорость компрессии/декомпрессии и передачи должна быть не менее 20-30 фреймов в секунду
- Анимационное видео более компактно (синтезированные изображения, использование стандартных шаблонов)
- MPEG-4: объектно-базирующееся представление, специальные методы

Возможные запросы

а) Текстовый запрос: *найти все документы (из милицейских архивов, архивов газет, заявлений свидетелей, банковских транзакций), в которых подозреваемое лицо/компания косвенно или прямо совершила какие-либо операции с компанией АБВ*. Документы должны индексироваться на основе семантического значения ключевых слов.

б) Запрос по изображению: *по имеющейся фотографии человека найти другие фотографии с этим же человеком*.

в) Аудио-запрос: *определить говорящего по записи*. Потребуются специальные методы обработки звука; основная идея: характеристический вектор (feature vector) – характеристика речи

Сферы применения

а) **Мультимедийные образовательные сервисы:**

- Удаленное обучение
- Учебные материалы
- Архивы аудио-/видеоматериалов (для образования)
- Возможность предварительного просмотра

б) **Видео по требованию:**

- Выбор видеоматериала (фильма, …), возможно с помощью запросов
- Возможность предварительного просмотра; перемотка вперед/назад
- Высокая пропускная способность
- Простой способ оплаты
- В ближайшем будущем, но еще не сегодня

Упрощенное представление о мультимедийных бд

Популярные, но упрощенные представления о мультимедийной базе данных:

а) CD-ROM содержащий мультимедийные данные
б) Мультимедийная файловая система
в) Видео/аудио по запросу: быстродействующие параллельные диски и высокоскоростная сеть
г) Системы обработки документов и изображений: сканирование, хранение, индексирование, и извлечение больших объемов печатных документов
д) Реляционная бд + поддержка больших двоичных объектов (BLOB): фрагментарная (кусочная) обработка массивных двоичных объектов, пользовательские функции
е) Объектно-реляционные бд + поддержка массивных двоичных объектов: поведение (характеристики) мультимедийных объектов могут быть реализованы в СУБД; такие системы поддерживают ряд мультимедийных типов данных
ж) Объектно-ориентированные бд + поддержка массивных двоичных объектов: аналогично е), но более явное представление сложных мультимедийных объектов; хорошо подходит для систем автоматизированного проектирования и производства (CAD/CAM)

9. Иерархические модели данных

В основе иерархической модели данных лежит древовидная структура. Иерархическая модель представляет собой упорядоченный набор деревьев, точнее, набор экземпляров одного типа дерева.

Основными информационными единицами в иерархической модели являются сегмент и поле. Поле - наименьшая неделимая единица данных, доступная пользователю. Для сегмента определяются тип сегмента и экземпляр сегмента. Тип сегмента — это поименованная совокупность входящих в него типов полей. Экземпляр сегмента образуется из конкретных значений полей данных.

Между сегментами существуют связи типа «предок-потомок». Дерево начинается с корневого сегмента. Каждый сегмент имеет не более одного предка, произвольное количество потомков и, по крайней мере, одно поле.

В иерархической модели автоматически поддерживается целостность ссылок между предками и потомками. Основное правило: *никакой потомок не может существовать без своего предка.*

Иерархическая модель довольно удобна для представления предметных областей, так как иерархические отношения довольно часто встречаются между сущностями реального мира. Иерархическая модель позволяет реализовать связи типа «один-к-одному» и «один-ко-многим».

Достоинства иерархической модели: простота и быстродействие.

Недостатки иерархической модели:

- - необходимость дублирования деревьев для реализации связей типа «многие-ко-многим» и, следовательно, увеличение затрат на поиск;
- - невозможность существования потомка без предка (ввод пустых сегментов-предков, удаление предка влечет за собой удаление всех его потомков);

10. Сетевые модели данных

Сетевая база данных – это база данных, в которой одна запись может участвовать в нескольких отношениях предок-потомок

Физически данная модель также реализуется за счет хранящихся внутри самой записи указателей на другие записи, только, в отличие от иерархической модели, число этих указателей может быть произвольным.

И иерархическая и сетевая модели достаточно просты, однако они имеют общий недостаток: для того, чтобы получить ответ даже на простой вопрос, программист должен был написать программу, которая просматривала базу данных, двигаясь по указателям от одной записи к другой. Написание программы занимало некоторое время, и часто к тому моменту, когда такая программа была написана, необходимость в получении данных уже не требовалась. Поэтому в середине 80-х годов 20 века произошел практически повсеместный переход к реляционным базам данных.

11. СУБД, построение по предложениям CODASYL

Система управления базами данных (СУБД) – это система программного обеспечения, позволяющая обрабатывать обращения к базе данных, поступающие от прикладных программ конечных пользователей. Иными словами, СУБД является интерфейсом между базой данных и прикладными задачами.

Системы управления базами данных позволяют объединять большие объемы информации и обрабатывать их, сортировать, делать выборки по определённым критериям и т.п.

Основные функции СУБД – это:

– определение данных;
– обработка данных;
– управление данными.

Современные СУБД дают возможность включать в них не только текстовую и графическую информацию, но и звуковые фрагменты и даже видеоклипы.

Простота использования СУБД позволяет создавать новые базы данных, не прибегая к программированию, а пользуясь только встроенными функциями.

СУБД обеспечивают правильность, полноту и непротиворечивость данных, а также удобный доступ к ним.

Для менее сложных применений вместо СУБД используются информационно-поисковые системы (ИПС), которые выполняют следующие функции:

– хранение большого объема информации;
– быстрый поиск требуемой информации;
– добавление, удаление и изменение хранимой информации;
– вывод ее в удобном для человека виде.

Выделяют следующие виды СУБД :

* полнофункциональные СУБД;
* серверы БД;
* средства разработки программ работы с БД.

Полнофункциональные СУБД представляют собой традиционные СУБД. К ним относятся dBaseIV, Microsoft Access, Microsoft FoxPro и др.

Серверы БД предназначены для организации центров обработки данных в сетях ЭВМ.

Серверы БД обеспечивают обработку запросов клиентских программ обычно с помощью операторов SQL. Примерами серверов БД являются: Microsoft SQL Server, InterBase и др.

В роли *клиентских программ* в общем случае могут использоваться СУБД, электронные таблицы, текстовые процессоры, программы электронной почты и др.

Средства разработки программ работы с БД могут использоваться для создания следующих программ:

* клиентских программ;
* серверов БД и их отдельных компонентов;
* пользовательских приложений.

СУБД, реализованная по принципам КОДАСИЛ, накладывает меньше ограничений, чем система с разнотипными файлами. Это облегчает отображение предметной области в

даталогическую модель. Данные системы возлагают все функции по формированию и поддержке наборов на СУБД. Языки программирования с произвольными наборами (типами) данных побуждают сосредотачивать все функции работы с наборами данных в самом типе набора.

Выполнение этих функций необходимо при реализации следующих операций:

1) Добавление или исключение записи из БД.
2) Добавление или исключение записи из набора.
3) Модификация записи таким образом, что изменяется её логическая позиция внутри набора.
4) Модификация записи таким образом, что изменяется набор, в котором она участвует.

Наиболее известная реализация КОДАСИЛ – СУБД сеть, компас.

12. Реляционная модель

Реляционная модель базируется на теоретико-множественном понятии отношения. В математических дисциплинах существует понятие ''***отношение***'' (relation), физическим представлением которого является ***таблица***. Отсюда и произошло название модели – ***реляционная***.

.

Реляционная модель – модель представления данных предметной области, построенная на взаимосвязи отношении. Согласно К. Дж. Дейту реляционная модель данных описывает три аспекта: структурный, целостный манипуляционный:

- Структурный — данные в модели представляют собой набор отношений.
- Целостный — отношения отвечают определенным условиям целостности. (декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных).
- Манипуляционный (обработки) — модель поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

В *реляционной базе данных* вся информации представляется в виде таблиц, и любые операции над данными – это операции над таблицами. Таблицы строят из строк и столбцов. *Строки* – это записи, а *столбцы* представляют собой структуру записи (каждый столбец имеет определенный тип данных и длину данных). Строки в таблице не упорядочены – не существует первой или десятой строки. Однако поскольку на строки необходимо как-то ссылаться, то вводится понятие «первичный ключ».

*Реляционная модель является удобной и наиболее привычной формой представления данных в виде **таблицы (отношения**). Каждое отношение имеет **имя** и состоит из поименованных **атрибутов** (столбцов) данных. Одним из базовых преимуществ реляционной модели является ее <u>однородность</u>. Все данные хранятся в таблицах, в которых каждая строка имеет один и тот же формат. Каждая строка в таблице представляет некоторый объект реального мира или соотношение между объектами.*

Основными понятиями, с помощью которых определяется реляционная модель, являются следующие:

*1. **реляционная БД** – набор нормализованных отношений;*

*2. **отношение** – файл, плоская таблица, состоящая из столбцов и строк; таблица, в которой каждое поле является атомарным;*

*3. **домен** – совокупность допустимых значений, из которой берется значение соответствующего атрибута определенного отношения. С точки зрения программирования, **домен** - это тип данных;*

*4. **универсум** – совокупность значений всех полей или совокупность доменов;*

*5. **кортеж** – запись, строка таблицы;*

*6. **кардинальность -** количество строк в таблице;*

*7. **атрибуты –**поименованныеполя, столбцы таблицы;*

*8. **степень отношения** - количество полей (столбцов);*

*9. **схема отношения** – упорядоченный список имен атрибутов;*

*10. **схема реляционной БД** – совокупность схем отношений;*

*11. **первичный ключ** – уникальный идентификатор с неповторяющимися записями – столбец или некоторое подмножество столбцов, которые единственным образом определяют строки.*

*12. **внешний ключ -** это столбец или подмножество столбцов одной таблицы, которые могут служить в качестве первичного ключа для другой таблицы. Внешний ключ таблицы является ссылкой на первичный ключ другой таблицы. Поскольку целью построения БД является хранение вс ех данных, по возможности, в одном экземпляре, то если некий атрибут присутствует в нескольких отношениях, то его наличие обычно отражает определ енную связь между строками этих отношений.*

*Первичный ключ, который включает более одного столбца, принято называть **множественным**, или **комбинированным**, или **составным**, или **суперключом**.*

***Правило целостности объектов** утверждает, что первичный ключ не должна быть полностью или частично пустым.*

Каждая ***реляционнаятаблица*** обладает следующими *свойствами*:

- - имеет имя, которое отличается от имен вс ех других таблиц;
- - данные в ячейках таблицы должны быть структурно неделимыми. Недопустимо, чтобы в ячейке таблицы содержалось более одной порции информации. К примеру, номер и серия паспорта должны располагаться в разных столбцах таблицы;
- - все столбцы в таблице однородные, т.е. вс е элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- - каждый столбец имеет уникальное имя;
- - одинаковые строки в таблице отсутствуют;
- - порядок следования строк и столбцов должна быть произвольным, независимо от их переупорядочивания отношение будет оставаться одним и тем же, а потому иметь тот же смысл.

Цели создания реляционной модели данных:

1. Обеспечение более высокой степени независимости от данных.

2. Создание прочного фундамента для решения проблем непротиворечивости и избыточности данных.

3. Расширение языков управления данными за счёт включения операций над множествами

13. Понятия, структура, преимущества и недостатки реляционной модели данных (в 12)
14. Схема и свойства отношений (в 12)
15. Реляционная алгебра и исчисления отношений

Основная идея реляционной алгебры состоит в том, что коль скоро отношения являются множествами, то средства манипулирования отношениями могут базироваться на традиционных теоретико-множественных операциях, дополненных некоторыми специальными операциями, специфичными для баз данных.

Набор основных алгебраических операций состоит из восьми операций, которые делятся на два класса - теоретико-множественные операции и специальные реляционные операции. В состав теоретико-множественных операций входят операции:

- объединения отношений;
- пересечения отношений;
- взятия разности отношений;
- прямого произведения отношений.

Специальные реляционные операции включают:

- - ограничение отношения;
- - проекцию отношения;
- - соединение отношений;
- - деление отношений.

Кроме того, в состав алгебры включается операция присваивания, позволяющая сохранить в базе данных результаты вычисления алгебраических выражений, и операция переименования атрибутов, дающая возможность корректно сформировать заголовок (схему) результирующего отношения.

Каждое отношение характеризуется схемой (или заголовком) и набором кортежей (или телом). Поэтому, если действительно желать иметь алгебру, операции которой замкнуты относительно понятия отношения, то каждая операция должна производить отношение в полном смысле, т.е. оно должно обладать и телом, и заголовком. Только в этом случае будет действительно возможно строить вложенные выражения.

Реляционное исчисление

Обычно говорят, что алгебраическая формулировка является процедурной, т.е. задающей правила выполнения запроса, а логическая - описательной (или декларативной), поскольку она всего лишь описывает свойства желаемого результата

Предположим, что мы работаем с базой данных, обладающей схемой *СОТРУДНИКИ (СОТР_НОМ, СОТР_ИМЯ, СОТР_ЗАРП, ОТД_НОМ) и ОТДЕЛЫ (ОТД_НОМ, ОТД_КОЛ, ОТД_НАЧ),* и хотим узнать имена и номера сотрудников, являющихся начальниками отделов с количеством сотрудников больше 50.

Если бы для формулировки такого запроса использовалась реляционная алгебра, то мы получили бы алгебраическое выражение, которое читалось бы, например, следующим образом:

- выполнить соединение отношений СОТРУДНИКИ и ОТДЕЛЫ по условию СОТР_НОМ = ОТД_НАЧ;
- ограничить полученное отношение по условию ОТД_КОЛ > 50;
- спроецировать результат предыдущей операции на атрибут СОТР_ИМЯ, СОТР_НОМ.

Если же сформулировать тот же запрос с использованием реляционного исчисления, которому посвящается этот раздел, то мы получили бы формулу, которую можно было бы прочитать, например, следующим образом: Выдать СОТР_ИМЯ и СОТР_НОМ для сотрудников таких, что существует отдел с таким же значением ОТД_НАЧ и значением ОТД_КОЛ большим 50. В этом случае система должна сама решить, какие операции и в каком порядке нужно выполнить над отношениями СОТРУДНИКИ и ОТДЕЛЫ.

16. Теория нормальных форм

Если даны два атрибута *X* и *Y* некоторого отношения, то говорят, что *Y* функционально зависит от *X*, если в любой момент времени каждому значению *X* соответствует ровно одно значение *Y*. Функциональная зависимость обозначается *X -> Y*. Отметим, что *X* и *Y* могут представлять собой не только единичные атрибуты, но и группы, составленные из нескольких атрибутов одного отношения.

Можно сказать, что функциональные зависимости представляют собой связи типа "один ко многим", существующие внутри отношения.

Некоторые функциональные зависимости могут быть нежелательны.

Избыточная функциональная зависимость - зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей, имеющихся в базе данных.

Корректной считается такая схема базы данных, в которой отсутствуют избыточные функциональные зависимости. В противном случае приходится прибегать к процедуре декомпозиции (разложения) имеющегося множества отношений. При этом порождаемое множество содержит большее число отношений, которые являются проекциями отношений исходного множества. (Операция проекции описана в разделе, посвященном реляционной алгебре).

Обратимый пошаговый процесс замены данной совокупности отношений другой схемой с устранением избыточных функциональных зависимостей называется **нормализацией**.

Условие обратимости требует, чтобы декомпозиция сохраняла эквивалентность схем при замене одной схемы на другую, т.е. в результирующих отношениях:

- не должны появляться ранее отсутствовавшие кортежи;
- на отношениях новой схемы должно выполняться исходное множество функциональных зависимостей.

17. Функциональные и транзитивные зависимости

Функциональная зависимость. В отношении R атрибут Y функционально зависит от атрибута X — если каждому значению X соответствует в точности одно значение Y. Обозначается y:x→y (x функционально определяет y)

Полная функциональная зависимость. Функциональная зависимость y:x→y называется полной, если атрибут Y не зависит функционально от любого точного подмножества X

Транзитивная функциональная зависимость. Функциональная зависимость y:x→y называется транзитивной, если существует такой атрибут Z, что имеются функциональные зависимости x →z и z→y (обратная зависимость отсутствует).

18. 1-5 нормальные формы

Первая нормальная форма (1NF)

Таблица находится в первой нормальной форме, если каждый её атрибут атомарен. Под выражением «атрибут атомарен» понимается, что атрибут может содержать только одно значение. Таким образом, не существует 1NF таблицы, в полях которых могут храниться списки значений. Для приведения таблицы к 1NF обычно требуется разбить таблицу на несколько отдельных таблиц. Замечание: в реляционной модели отношение всегда находится в 1 (или более высокой) нормальной форме в том смысле, что иные отношения не рассматриваются в реляционной модели. То есть само определение понятия отношение заведомо подразумевает наличие 1NF.

Вторая нормальная форма (2NF)

Таблица находится во второй нормальной форме, если она находится в первой нормальной форме, и при этом любой её атрибут, не входящий в состав первичного ключа, функционально полно зависит от первичного ключа. Функционально полная зависимость означает, что атрибут функционально зависит от всего первичного составного ключа, но при этом не находится в функциональной зависимости от какой-либо из входящих в него атрибутов (частей). Или другими словами: в 2NF нет неключевых атрибутов, зависящих от части составного ключа (+ выполняются условия 1NF).

Третья нормальная форма (3NF)

Таблица находится в третьей нормальной форме (3NF), если она находится во второй нормальной форме 2NF и при этом любой ее неключевой атрибут зависит только от первичного ключа (Primary key, PK) (иначе говоря, один факт хранится в одном месте).

Таким образом, отношение находится в 3NF тогда и только тогда, когда оно находится во 2NF и отсутствуют транзитивные зависимости неключевых атрибутов от ключевых. Транзитивной зависимостью неключевых атрибутов от ключевых называется следующая: A → B и B → C, где A — набор ключевых атрибутов (ключ), B и C — различные множества неключевых атрибутов.

При решении практических задач в большинстве случаев третья нормальная форма является достаточной. Процесс проектирования реляционной базы данных, как правило, заканчивается приведением к 3NF.

Четвёртая нормальная форма (4NF)
Таблица находится в 4NF, если она находится в BCNF и не содержит нетривиальных многозначных зависимостей. Многозначная зависимость не является функциональной, она существует в том случае, когда из факта, что в таблице содержится некоторая строка X, следует, что в таблице обязательно существует некоторая определённая строка Y. То есть, таблица находится в 4NF, если все ее многозначные зависимости являются функциональными.
Пятая нормальная форма (5NF)
Таблица находится в 5NF, если она находится в 4NF и любая многозначная зависимость соединения в ней является тривиальной. Пятая нормальная форма в большей степени является теоретическим исследованием и практически не применяется при реальном проектировании баз данных. Это связано со сложностью определения самого наличия зависимостей «проекции — соединения», поскольку утверждение о наличии такой зависимости должно быть сделано для всех возможных состояний БД.

19. Нормальная форма Бойса-Кодда

Нормальная форма Бойса — Кодда (BCNF)

Это модификация третьей нормальной формы (в некоторых источниках именно 3NF называется формой Бойса — Кодда).

Таблица находится в BCNF, если она находится в 3NF, и при этом отсутствуют функциональные зависимости атрибутов первичного ключа от неключевых атрибутов. Таблица может находиться в 3NF, но не в BCNF, только в одном случае: если она имеет, помимо первичного ключа, ещё по крайней мере один возможный ключ. Все зависимые от первичного ключа атрибуты должны быть потенциальными ключами отношения. Если это условие не выполняется, для них создаётся отдельное отношение. Чтобы сущность соответствовала BCNF, она должна находиться в третьей нормальной форме. Любая сущность с единственным возможным ключом, соответствующая требованиям третьей нормальной формы, автоматически находится в BCNF.

20. Язык запросов SQL. Оператор создания, вставки, удаления и модификации.

Каждое предложение SQL — это либо *запрос* данных из базы, либо обращение к базе данных, которое приводит к изменению данных в базе. Различают следующие типы запросов, в соответствии с изменениями, происходящими в базе данных в результате их выполнения:

- запросы на создание или изменение в базе данных новых или существующих объектов (при этом в запросе описывается тип и структура создаваемого или изменяемого объекта);
- запросы на получение данных;
- запросы на добавление новых данных (записей);
- запросы на удаление данных;
- обращения к СУБД.

В основном, все выражения SQL имеют следующий синтаксис:

```
ОПЕРАТОР аргументы ИНСТРУКЦИЯ аргументы;
```

запросы делятся на:

- запросы, оперирующие самими таблицами (создание и изменение таблиц);
- запросы, оперирующие с отдельными записями (или строками таблиц) или наборами записей.

Самый главный вид запроса — это запрос, возвращающий (пользователю) некоторый набор строк, с которым можно осуществить одну из трех операций:

- просмотреть полученный набор;
- изменить все записи набора;
- удалить все записи набора.

CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.),
INSERT добавляет новые данные,
UPDATE изменяет существующие данные,
DELETE удаляет данные; ALTER изменяет объект,
DROP удаляет объект;

21. Язык запросов SQL. Оператор выборки

Общее название `SELECT`

Выборка всех данных

```
SELECT * FROM table_name;
```

Выборка колонок

```
SELECT column_1, column_2 FROM table_name;
```

Выборка по условию

```
SELECT ... FROM table_name WHERE condition;
```

Выборка только уникальных записей

```
SELECT DISTINCT ... FROM table_name ...;
```

В SQL, оператор `SELECT` состоит из *блоков*, определяющих детали выражения.

В MySQL единственным обязательным блоком является первый: `SELECT`.

Блоки оператора `SELECT`:

- `SELECT` – указывает, что должно быть выбрано. Отчасти соответствует операциям проекции и переименования реляционной алгебры
- `FROM` – указывает, откуда должны быть выбраны данные. Соответствует аргументу операции реляционной алгебры.
- `WHERE` – указывает условие, которому выбранные данные должны удовлетворять. Соответствует операции выборки реляционной алгебры
- `GROUP BY` – используется для группировки результатов по одинаковым значениям столбцов. Часто используется со статистическими запросами.
- `HAVING` – аналогичен `WHERE`, но используется со статистическими запросами (`WHERE` с ними не работает)
- `ORDER BY` – сортировка выдачи.

22. Основные понятия модели сущность-связь

Сущность (entity) - это объект, который должна быть идентифицирован неким способом, отличающим его от других объектов. Примеры: конкретный человек, предприятие, событие и т.д.
Набор сущностей (entity set) − множество сущностей одного типа (обладающих одинаковыми свойствами). Примеры: все люди, предприятия, праздники и т.д. Наборы сущностей необязательно

должны быть непересекающимися. К примеру, сущность, принадлежащая к набору *Мужчины*, также принадлежит набору *Люди*.

Связь(relationship) - это ассоциация, установленная между несколькими сущностями. К примеру:

· Поскольку каждый сотрудник работает в каком-либо отделе, между сущностями *Сотрудник* и *Отдел* существует связь "работает в …" или *Отдел−Работник*.

Связь также может иметь атрибуты.

Атрибут (Attribute) — любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности.

Рис. 2.24. Сущность с атрибутами

Уникальным идентификатором называется неизбыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. Неизбыточность заключается в том, что удаление любого атрибута из уникального идентификатора нарушает его уникальность.

Степенью связиназывается количество сущностей, участвующих в связи.

Мощностью связиназывается максимальное число экземпляров сущности, которое может быть связано с одним экземпляром данной сущности.

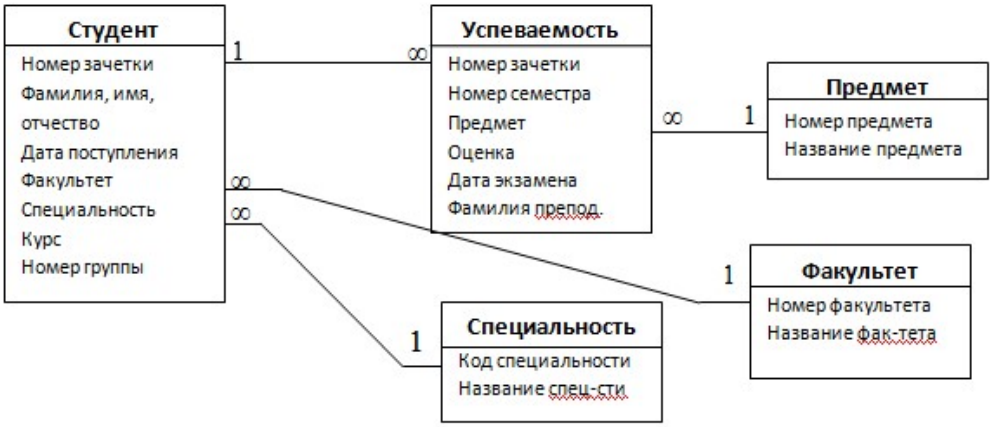
Связь может иметь один из следующих трех типов: 1к1, 1к многим, многие ко многим.

23. Правила построения ER-диаграмм

Класс сущностей представляется в виде четырехугольника. В четырехугольнике записано уникальное имя класса сущности (прописными буквами) и имена атрибутов.

Связи между сущностями обозначаются стрелками

На диаграммах атрибуты, входящие в первичный ключ, подчеркиваются



24. Понятия транзакции и целостности БД

Под транзакцией понимается неделимая с точки зрения воздействия на БД последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации) такая, что либо результаты всех операторов, входящих в транзакцию, отображаются в БД, либо воздействие всех этих операторов полностью отсутствует. Основной смысл транзакции - "Все или ничего": при завершении транзакции оператором COMMIT результаты гарантированно фиксируются во

внешней памяти (смысл термина commit - "зафиксировать" результаты транзакции); при завершении транзакции оператором ROLLBACK результаты гарантированно отсутствуют во внешней памяти (смысл термина rollback - ликвидировать результаты транзакции).

Поэтому для поддержания подобных ограничений целостности (банковский счет) допускается их нарушение внутри транзакции с тем условием, чтобы к моменту завершения транзакции условия целостности были соблюдены. В системах с развитыми средствами ограничения и контроля целостности каждая транзакция начинается при целостном состоянии БД и должна оставить это состояние целостными после своего завершения. Несоблюдение этого условия приводит к тому, что вместо фиксации результатов транзакции происходит ее откат

Целостность базы данных - соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам

25. Свойства АСИД

Транзакция обладает четырьмя важными свойствами, известными как ***свойства АСИД***:

- ***(А) Атомарность***. Транзакция выполняется как атомарная операция - либо выполняется вся транзакция целиком, либо она целиком не выполняется.
- ***(С) Согласованность***. Транзакция переводит базу данных из одного согласованного (целостного) состояния в другое согласованное (целостное) состояние. Внутри транзакции согласованность базы данных может нарушаться.
- ***(И) Изоляция***. Транзакции разных пользователей не должны мешать друг другу (например, как если бы они выполнялись строго по очереди).
- ***(Д) Долговечность***. Если транзакция выполнена, то результаты ее работы должны сохраниться в базе данных, даже если в следующий момент произойдет сбой системы.

26. Классификация ограничений целостности

Классификация ограничений целостности по времени проверки

1. Немедленно проверяемые ограничения. 2. Ограничения с отложенной проверкой.

Немедленно проверяемые ограничения проверяются непосредственно в момент выполнения операции, могущей нарушить ограничение. Например, проверка уникальности потенциального ключа проверяется в момент вставки записи в таблицу. Если ограничение нарушается, то такая операция отвергается. Транзакция, внутри которой произошло нарушение немедленно проверяемого утверждения целостности, обычно откатывается.

Ограничения **с** *отложенной проверкой* проверяется в момент фиксации транзакции оператором COMMIT WORK. Внутри транзакции ограничение может не выполняться. Если в момент фиксации транзакции обнаруживается нарушение ограничения с отложенной проверкой, то транзакция откатывается.

Классификация ограничений целостности по области действия

1. Ограничения домена 2. Ограничения атрибута 3. Ограничения кортежа 4. Ограничения отношения 5. Ограничения базы данных 6. Ограничения домена.

Ограничения целостности **домена** представляют собой ограничения, накладываемые только на допустимые значения домена. Ограничения домена сами по себе не проверяются. Если на каком-либо домене основан атрибут, то ограничение соответствующего домена становится ограничением этого атрибута.

Ограничение целостности **атрибута** представляют собой ограничения, накладываемые на допустимые значения атрибута вследствие того, что атрибут основан на каком-либо домене. Ограничение атрибута в точности совпадают с ограничениями соответствующего домена. Отличие ограничений атрибута от ограничений домена в том, что ограничения атрибута проверяются. Если логика предметной области такова, что на значения атрибута необходимо наложить дополнительные ограничения, помимо ограничений домена, то такие ограничения переходят в следующую категорию. Ограничение атрибута является немедленно проверяемым ограничением. Действительно, ограничение атрибута не зависит ни от каких других объектов базы данных, кроме

домена, на котором основан атрибут. Поэтому никакие изменения в других объектах не могут повлиять на истинность ограничения.

Ограничения целостности **кортежа** представляют собой ограничения, накладываемые на допустимые значения отдельного кортежа отношения, и не являющиеся ограничением целостности атрибута. Требование, что ограничение относится к отдельному кортежу отношения, означает, что для его проверки не требуется никакой информации о других кортежах отношения. Приведенное ограничение кортежа, по сути, является дополнительным ограничением на значения одного атрибута. В этом случае допустимы два решения. Можно объявить новый домен "Возраст сотрудника спецподразделения" и тогда ограничение кортежа становится ограничением домена и атрибута, либо рассматривать это ограничение именно как ограничение кортежа. Оба решения имеют свои положительные и отрицательные стороны.

27. Структура двухзвенной архитектуры клиент-сервер

Двухзвенная архитектура "клиент-сервер"

В архитектуре "клиент-сервер" программное обеспечение разделено на две части -клиентскую часть и серверную часть. Задача клиентской-части (программы-клиента) состоит во взаимодействии с пользователем, передаче пользовательского запроса серверу, получение запроса от серверной части (программы-сервера) и представление его в удобном для пользователя виде. Программа-сервер же обрабатывает запросы клиента и выдает ответы. Классические примеры: Web-технологии (клиент-браузер, сервер-Web-сервер), работа с распределенными СУБД (клиент - специальная программа, сервер - сервер базы данных).

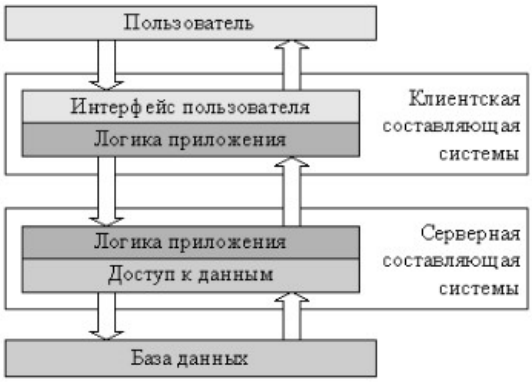


Рис. 2.3. Двухзвенная архитектура

28. Структура трехзвенной архитектуры клиент сервер

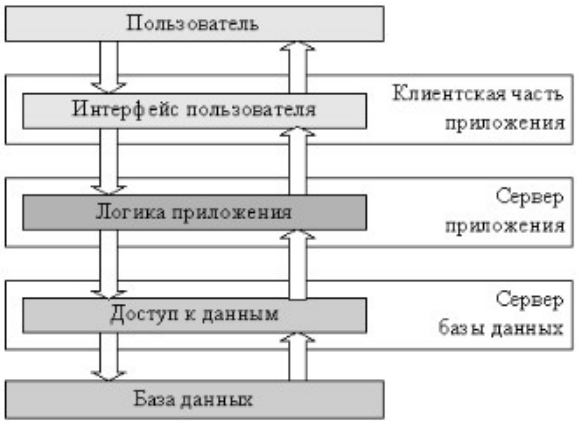


Рис. 2.4. Трехзвенная архитектура

29. Распределенные БД

Рассредоточенная или, если правильно называть, распределенная база данных – это такая БД, в состав которой входит несколько компьютеров, связанных сетью, на каждом из которых работает локальная база данных. Совокупность всех этих программно-аппаратных средств создает общую БД. Распределенные базы данных извне выглядят как обычные локальные базы, их аппаратная разнесенность не заметна пользователям. Распределенная система управления контролирует все узлы БД и обеспечивает связность данных.

Рассмотрим основные принципы распределенной обработки.

1. Локальная независимость. Узлы в распределённой системе должны быть независимы, или автономны. Локальная независимость означает, что все операции на узле контролируются этим узлом.

2. Отсутствие опоры на центральный узел. Локальная независимость предполагает, что все узлы в распределённой системе должны рассматриваться как равные. Поэтому не должно быть никаких обращений к «центральному» или «главному» узлу с целью получения некоторого централизованного сервиса.

3. Непрерывное функционирование. Распределённые системы должны предоставлять более высокую степень надёжности и доступности.

4. Независимость от расположения. Пользователи не должны знать, где именно данные хранятся физически и должны поступать так, как если бы все данные хранились на их собственном локальном узле.

5. Независимость от фрагментации (дробления данных на множество мелких разрозненных фрагментов). Система поддерживает независимость от фрагментации, если данная переменная-отношение может быть разделена на части или фрагменты при организации её физического хранения. В этом случае данные могут храниться в том месте, где они чаще всего используются, что позволяет достичь локализации большинства операций и уменьшения сетевого трафика.

6. Независимость от репликации. Система поддерживает репликацию данных, если данная хранимая переменная-отношение — или в общем случае данный фрагмент данной хранимой переменной-отношения — может быть представлена несколькими отдельными копиями или репликами, которые хранятся на нескольких отдельных узлах.

7. Обработка распределённых запросов. Суть в том, что для запроса может потребоваться обращение к нескольким узлам. В такой системе может быть много возможных способов пересылки данных, позволяющих выполнить рассматриваемый запрос.

8. Управление распределёнными транзакциями (последовательность операций, представляющая собой логическую единицу работы с данными). Существует 2 главных аспекта управления транзакциями: управление восстановлением и управление параллельностью обработки. Что касается управления восстановлением, то чтобы обеспечить атомарность транзакции в распределённой среде, система должна гарантировать, что все множество относящихся к данной транзакции агентов (агент — процесс, который выполняется для данной транзакции на отдельном узле) или зафиксировало свои результаты, или выполнило откат. Что касается управления параллельностью, то оно в большинстве распределённых систем базируется на механизме блокирования, точно так, как и в нераспределённых системах.

9. Аппаратная независимость. Желательно иметь возможность запускать одну и ту же СУБД на различных аппаратных платформах и, более того, добиться, чтобы различные машины участвовали в работе распределённой системы как равноправные партнёры.

10. Независимость от операционной системы. Возможность функционирования СУБД под различными операционными системами.

11. Независимость от сети. Возможность поддерживать много принципиально различных узлов, отличающихся оборудованием и операционными системами, а также ряд типов различных коммуникационных сетей.

12. Независимость от типа СУБД. Необходимо, чтобы экземпляры СУБД на различных узлах все вместе поддерживали один и тот же интерфейс, и совсем необязательно, чтобы это были копии одной и той же версии СУБД.