Содержание:

Введение

Базы данных -- это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия. Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять. Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описанием кулинарного рецепта. Однотипные поля разных записей образуют столбец таблицы. Записи одной таблицы могут содержать ссылки на данные другой таблицы, например, в таблице со списком товаров могут храниться ссылки на справочник производителей товаров с их адресами и другими реквизитами. При этом записи, касающиеся разных товаров, могут указывать на одного и того же производителя. Такое взаимодействие таблиц называется связью. Другие модули базы данных предназначены для обработки информации, хранящейся в таблицах. С помощью запросов производится выборка данных, отвечающих определенным условиям. Формы предназначены для форматированного ввода и восприятия информации. Отчеты обеспечивают вывод (как правило, на принтер) красочно оформленного списка записей с заголовками, пунктами и подпунктами. Создав одну таблицу, мы уже получаем полноценную базу данных

1. Типы и структуры данных

Данные, хранящиеся в памяти ЭВМ представляют собой совокупность нулей и едениц (битов). Биты объединяются в последовательности: байты, слова и т.д. Каждому участку оперативной памяти, который может вместить один байт или

слово, присваивается порядковый номер (адрес). Какой смысл заключен в данных, какими символами они выражены - буквенными или цифровыми, что означает то или иное число - все это определяется программой обработки. Все данные необходимые для решения практических задач подразделяются на несколько типов, причем понятие тип связывается не только с представлением данных в адресном пространстве, но и со способом их обработки. Любые данные могут быть отнесены к одному из двух типов: основному (простому), форма представления которого определяется архитектурой ЭВМ, или сложному, конструируемому пользователем для решения конкретных задач. Данные простого типа это - символы, числа и т.п. элементы, дальнейшее дробление которых не имеет смысла. Из элементарных данных формируются структуры (сложные типы) данных. Некоторые структуры: ▪ Массив(функция с конечной областью определения) - простая совокупность элементов данных одного типа, средство оперирования группой данных одного типа. Отдельный элемент массива задается индексом. Массив может быть одномерным, двумерным и т.д. Разновидностями одномерных массивов переменной длины являются структуры типа *кольцо, стек, очередь и двухсторонняя очередь*. ▪ Запись(декартово произведение) - совокупность элементов данных разного типа. В простейшем случае запись содержит постоянное количество элементов, которые называют *полями*. Совокупность записей одинаковой структуры называется *файлом*. (Файлом называют также набор данных во внешней памяти, например, на магнитном диске). Для того, чтобы иметь возможность извлекать из файла отдельные записи, каждой записи присваивают уникальное имя или номер, которое служит ее идентификатором и располагается в отдельном поле. Этот идентификатор называют *ключом*. Такие структуры данных как массив или запись занимают в памяти ЭВМ постоянный объем, поэтому их называют статическими структурами. К статическим структурам относится также *множество*. Имеется ряд структур, которые могут изменять свою длину - так называемые *динамические структуры*. К ним относятся дерево, список, ссылка.

Важной структурой, для размещения элементов которой требуется нелинейное адресное пространство является дерево. Существует большое количество структур данных, которые могут быть представлены как деревья. Это, например, классификационные, иерархические, рекурсивные и др. структуры. информация массив реляционный сетевой Существует большое разнообразие сложных типов данных, но исследования, проведенные на большом практическом материале, показали, что среди них можно выделить несколько наиболее общих. Обобщенные структуры называют также моделями данных, т.к. они отражают представление пользователя о данных реального мира. **Любая модель данных должна**

содержать три компоненты: ***1****. структура данных* - описывает точку зрения пользователя на представление данных. ***2****. набор допустимых операций*, выполняемых на структуре данных. Модель данных предполагает, как минимум, наличие языка определения данных (ЯОД), описывающего структуру их хранения, и языка манипулирования данными (ЯМД), включающего операции извлечения и модификации данных. **3***. ограничения целостности* - механизм поддержания соответствия данных предметной области на основе формально описанных правил. **В процессе исторического развития в СУБД использовалось следующие модели данных:** · иерархическая · сетевая · реляционная В последнее время все большее значение приобретает объектно-ориентированный подход к представлению данных.

2. Виды баз данных

2.1 Иерархическая модель данных

Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй -- объекты второго уровня и т. д. Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами. Организация данных в СУБД иерархического типа определяется в терминах: *элемент, агрегат, запись (группа), групповое отношение, база данных.* · ***Атрибут*** *(элемент данных)* - наименьшая единица структуры данных. Обычно каждому элементу при описании базы данных присваивается уникальное имя. По этому имени к нему обращаются при обработке. Элемент данных также часто называют полем. · ***Запись*** - именованная совокупность атрибутов. Использование записей позволяет за одно обращение к базе получить некоторую логически связанную совокупность данных. Именно записи изменяются, добавляются и удаляются. Тип записи определяется составом ее атрибутов. Экземпляр записи - конкретная запись с конкретным значением элементов · ***Групповое отношение*** - иерархическое отношение между записями двух типов. Родительская запись

(владелец группового отношения) называется исходной записью, а дочерние записи (члены группового отношения) - подчиненными. Иерархическая база данных может хранить только такие древовидные структуры. Корневая запись каждого дерева обязательно должна содержать ключ с уникальным значением. Ключи некорневых записей должны иметь уникальное значение только в рамках группового отношения. Каждая запись идентифицируется полным сцепленным ключом, под которым понимается совокупность ключей всех записей от корневой по иерархическому пути. При графическом изображении групповые отношения изображают дугами ориентированного графа, а типы записей - вершинами (диаграмма Бахмана). Для групповых отношений в иерархической модели обеспечивается *автоматический режим включения* и *фиксированное членство*. Это означает, что для запоминания любой некорневой записи в БД должна существовать ее родительская запись. При удалении родительской записи автоматически удаляются все подчиненные.

Существуют операции над данными, определенные в иерархической модели: · **ДОБАВИТЬ** в базу данных новую запись. Для корневой записи обязательно формирование значения ключа. · **ИЗМЕНИТЬ** значение данных предварительно извлеченной записи. Ключевые данные не должны подвергаться изменениям. · **УДАЛИТЬ** некоторую запись и все подчиненные ей записи. · **ИЗВЛЕЧЬ:** о извлечь корневую запись по ключевому значению, допускается также последовательный просмотр корневых записей о извлечь следующую запись (следующая запись извлекается в порядке левостороннего обхода дерева) В операции ИЗВЛЕЧЬ допускается задание условий выборки (например, извлечь сотрудников с окладом более 1 тысячи руб.) Как видим, все операции изменения применяются только к одной "текущей" записи (которая предварительно извлечена из базы данных). Такой подход к манипулированию данных получил название "навигационного".

2.2 Сетевая модель данных

На разработку этого стандарта большое влияние оказал американский ученый Ч.Бахман. Основные принципы сетевой модели данных были разработаны в середине 60-х годов, эталонный вариант сетевой модели данных описан в отчетах рабочей группы по языкам баз данных (COnference on DAta SYstem Languages) CODASYL (1971 г.). Сетевая модель данных определяется в тех же терминах, что и иерархическая. Она состоит из множества записей, которые могут быть владельцами или членами групповых отношений. Связь между между записью-

владельцем и записью-членом также имеет вид 1:N. Основное различие этих моделей состоит в том, что в сетевой модели запись может быть членом *более чем одного* группового отношения. Согласно этой модели каждое групповое отношение именуется и проводится различие между его типом и экземпляром. Тип группового отношения задается его именем и определяет свойства общие для всех экземпляров данного типа. Экземпляр группового отношения представляется записью-владельцем и множеством (возможно пустым) подчиненных записей. При этом имеется следующее ограничение: экземпляр записи не может быть членом двух экземпляров групповых отношений одного типа (т.е. сотрудник, например, не может работать в двух отделах). **Иерархическая структура преобразовывается в сетевую следующим образом:** - древья (a) и (b), показанные на рисунке, заменяются одной сетевой структурой, в которой запись СОТРУДНИК входит в два групповых отношения; - для отображения типа M:N вводится запись СОТРУДНИК_КОНТРАКТ, которая не имеет полей и служит только для связи записей КОНТРАКТ и СОТРУДНИК. **Каждый экземпляр группового отношения характеризуется следующими признаками: · способ упорядочения подчиненных записей:** о произвольный, о хронологический /очередь/, о обратный хронологический /стек/, о сортированный. Если запись объявлена подчиненной в нескольких групповых отношениях, то в каждом из них может быть назначен свой способ упорядочивания. **· режим включения подчиненных записей:** о автоматический - невозможно занести в БД запись без того, чтобы она была сразу же закреплена за неким владельцем;

о ручной - позволяет запомнить в БД подчиненную запись и не включать ее немедленно в экземпляр группового отношения. Эта операция позже инициируется пользователем). **· режим исключения**

Принято выделять три класса членства подчиненных записей в групповых отношениях: *1. Фиксированное*. Подчиненная запись жестко связана с записью владельцем и ее можно исключить из группового отношения только удалив. При удалении записи-владельца все подчиненные записи автоматически тоже удаляются. В рассмотренном выше примере фиксированное членство предполагает групповое отношение "ЗАКЛЮЧАЕТ" между записями "КОНТРАКТ" и "ЗАКАЗЧИК", поскольку контракт не может существовать без заказчика. ***2. Обязательное***. Допускается переключение подчиненной записи на другого владельца, но невозможно ее существование без владельца. Для удаления записи-владельца необходимо, чтобы она не имела подчиненных записей с обязательным членством. Таким отношением связаны записи "СОТРУДНИК" и "ОТДЕЛ". Если отдел

расформировывается, все его сорудники должны быть либо переведены в другие отделы, либо уволены. ***3. Необязательное****.* Можно исключить запись из группового отношения, но сохранить ее в базе данных не прикрепляя к другому владельцу. При удалении записи-владельца ее подчиненные записи - необязательные члены сохраняются в базе, не участвуя более в групповом отношении такого типа. Примером такого группового отношения может служить "ВЫПОЛНЯЕТ" между "СОТРУДНИКИ" и "КОНТРАКТ", поскольку в организации могут существовать работники, чья деятельность не связана с выполненинем каких-либо договорных обязательств перед заказчиками. **Операции над данными, в сетевой модели данных: · ДОБАВИТЬ** - внести запись в БД и, в зависимости от режима включения, либо включить ее в групповое отношение, где она объявлена подчиненной, либо не включать ни в какое групповое отношение. **· ВКЛЮЧИТЬ В ГРУППОВОЕ ОТНОШЕНИЕ** - связать существующую подчиненную запись с записью-владельцем. **· ПЕРЕКЛЮЧИТЬ** - связать существующую подчиненную запись с другой записью-владельцем в том же групповом отношении. **· ОБНОВИТЬ** - изменить значение элементов предварительно извлеченной записи. **· ИЗВЛЕЧЬ** - извлечь записи последовательно по значению ключа, а также используя групповые отношения - от владельца можно перейти к записям - членам, а от подчиненной записи к владельцу набора. **· УДАЛИТЬ** - убрать из БД запись. Если эта запись является владельцем группового отношения, то анализируется класс членства подчиненных записей. Обязательные члены должны быть предварительно исключены из группового отношения, фиксированные удалены вместе с владельцем, необязательные останутся в БД. **ИСКЛЮЧИТЬ ИЗ ГРУППОВОГО ОТНОШЕНИЯ** - разорвать связь между записью-владельцем и записью-членом.

2.3 Реляционная модель данных

Реляционная модель предложена сотрудником компании IBM Е.Ф.Коддом в 1970 г. (русский перевод статьи, в которой она впервые описана опубликован в журнале "СУБД" N 1 за 1995 г.). В настоящее время эта модель является фактическим стандартом, на который ориентируются практически все современные коммерческие СУБД. В реляционной модели достигается гораздо более высокий уровень абстракции данных, чем в иерархической или сетевой. В упомянутой статье Е.Ф.Кодда утверждается, что "*реляционная модель предоставляет средства описания данных на основе только их естественной структуры, т.е. без потребности введения какой-либо дополнительной структуры*

для целей машинного представления". Другими словами, представление данных не зависит от способа их физической организации. Это обеспечивается за счет использования математической теории отношений (само название "реляционная" происходит от английского relation - "отношение"). Перейдем к рассмотрению структурной части реляционной модели данных. Прежде всего необходимо дать несколько определений: Декартово произведение: Для заданных конечных множеств (не обязательно различных) декартовым произведением называется множество произведений вида: , где Пример: *если даны два множества A (a1,a2,a3) и B (b1,b2), их декартово произведение будет иметь вид C=A*B (a1*b1, a2*b1, a3*b1, a1*b2, a2*b2, a3*b2)* · Отношение: Отношением *R*, определенным на множествах называется подмножество декартова произведения . При этом: o множества называются доменами отношения o элементы декартова произведения называются кортежами o число *n* определяет степень отношения (*n=1* - унарное, *n=2* - бинарное, ..., *n*-арное) o количество кортежей называется мощностью отношения Пример: *на множестве C из предыдущего примера могут быть определены отношения R1 (a1*b1, a3*b2) или R2 (a1*b1, a2*b1, a1*b2)* Отношения удобно представлять в виде таблиц. На рис. 2 представлена таблица (отношение степени 5), содержащая некоторые сведения о работниках гипотетического предприятия. Строки таблицы соответствуют кортежам. Каждая строка фактически представляет собой описание одного объекта реального мира (в данном случае работника), характеристики которого содержатся в столбцах. Можно провести аналогию между элементами реляционной модели данных и элементами модели "сущность-связь". Реляционные отношения соответствуют наборам сущностей, а кортежи - сущностям. Поэтому, также как и в модели "сущность-связь" столбцы в таблице, представляющей реляционное отношение, называют атрибутами. Каждый атрибут определен на домене, поэтому домен можно рассматривать как множество допустимых значений данного атрибута. Несколько атрибутов одного отношения и даже атрибуты разных отношений могут быть определены на одном и том же домене. В примере, показанном на рис.2 атрибуты "Оклад" и "Премия" определены на домене "Деньги". Поэтому, понятие домена имеет семантическую нагрузку: данные можно считать сравнимыми только тогда, когда они относятся к одному домену. Таким образом, в рассматриваемом нами примере сравнение атрибутов "Табельный номер" и "Оклад" является семантически некорректным, хотя они и содержат данные одного типа. Именнованное множество пар "имя атрибута - имя домена" называется схемой отношения. Мощность этого множества - называют степенью или "*арностью*" отношения. Набор именованных схем отношений представляет из себя схему базы

данных. Атрибут, значение которого однозначно идентифицирует кортежи, называется ключевым (или просто ключом). В нашем случае ключом является атрибут "Табельный номер", поскольку его значение уникально для каждого работника предприятия. Если кортежи идентифицируются только сцеплением значений нескольких атрибутов, то говорят, что отношение имеет составной ключ. Отношение может содержать несколько ключей. Всегда один из ключей объявляется первичным, его значения не могут обновляться. Все остальные ключи отношения называются *возможными ключами*. В отличие от иерархической и сетевой моделей данных в реляционной отсутствует понятие группового отношения. Для отражения ассоциаций между кортежами разных отношений используется дублирование их ключей. Рассмотренный ранее пример базы данных, содержащей сведения о подразделениях предприятия и работающих в них сотрудниках, применительно к реляционной модели будет иметь вид: Например, связь между отношениями ОТДЕЛ и СОТРУДНИК создается путем копирования первичного ключа "*Номер_отдела*" из первого отношения во второе. Таким образом:
▪ для того, чтобы получить список работников данного подразделения, необходимо **1.** из таблицы ОТДЕЛ установить значение атрибута "*Номер_отдела*", соответствующее данному "Наименованию_отдела" **2.** выбрать из таблицы СОТРУДНИК все записи, значение атрибута "*Номер_отдела*" которых равно полученному на предыдушем шаге.

▪ для того, чтобы узнать в каком отделе работает сотрудник, нужно выполнить обратную операцию: **1.** определяем "*Номер_отдела*" из таблицы СОТРУДНИК **2.** по полученному значению находим запись в таблице ОТДЕЛ. Атрибуты, представляющие собой копии ключей других отношений, называются внешними ключами. **Свойства отношений. 1**. Отсутствие кортежей-дубликатов. *Из этого свойства вытекает наличие у каждого кортежа первичного ключа. Для каждого отношения, по крайней мере, полный набор его атрибутов является первичным ключом. Однако, при определении первичного ключа должно соблюдаться требование "минимальности", т.е. в него не должны входить те атрибуты, которые можно отбросить без ущерба для основного свойства первичного ключа - однозначно определять кортеж.* **2.** Отсутствие упорядоченности кортежей. **3.** Отсутствие упордоченности атрибутов. Для ссылки на значение атрибута всегда используется имя атрибута. **4.** Атомарность значений атрибутов, т.е. среди значений домена не могут содержаться множества значений (отношения).

2.4 Объектно-ориентированные СУБД

Объектно-ориентированная база данных (ООБД) -- база данных, в которой данные моделируются в виде объектов, их атрибутов, методов и классов. Сразу же необходимо заметить, что общепринятого определения "объектно-ориентированной модели данных" не существует. Сейчас можно говорить лишь о неком "объектном" подходе к логическому представлению данных и о различных объектно-ориентированных способах его реализации. **Структура объектной модели описываются с помощью трех ключевых понятий**: ▪ **инкапсуляция** - каждый объект обладает некоторым внутренним состянием (хранит внутри себя запись данных), а также набором методов - процедур, с помощью которых (и *только* таким образом) можно получить доступ к данным, определяющим внутреннее состояние объекта, или изменить их. Таким образом, объекты можно рассматривать как самостоятельные сущности, отделенные от внешнего мира. ▪ **наследование** - подразумевает возможность создавать из классов объектов новые классы объекты, которые наследуют структуру и методы своих предков, добавляя к ним черты, отражающие их собственную индивидуальность. Наследование может быть простым (один предок) и множественным (несколько предков). ▪ **полиморфизм** - различные объекты могут по разному реагировать на одинаковые внешние события в зависимости от того, как реализованы их методы. **Для поддержания целостности объектно-ориентированный подход предлагает использовать следующие средства:** ▪ автоматическое поддержание отношений наследования ▪ возможность объявить некоторые поля данных и методы объекта как "скрытые", не видимые для других объектов; такие поля и методы используются только методами самого объекта ▪ создание процедур контроля целостности внутри объекта В объектно-ориентированных базах данных, в отличие от реляционных, хранятся не записи, а объекты. ОО-подход представляет более совершенные средства для отображения реального мира, чем реляционная модель: ▪ естественное представление данных. В реляционной модели все отношения принадлежат одному уровню, именно это осложняет преобразование иерархических связей модели "сущность-связь" в реляционную модель. ОО-модель можно рассматривать послойно, на разных уровнях абстракции. ▪ имеется возможность определения новых типов данных и операций с ними. **В то же время, ОО-модели присущ и ряд недостатков:**

▪ отсутствуют мощные непроцедурные средства извлечения объектов из базы. Все запросы приходится писать на процедурных языках, проблема их оптимизации возлагается на программиста. ▪ вместо чисто декларативных ограничений целостности (типа явного объявления первичных и внешних ключей реляционных таблиц с помощью ключевых слов PRIMARY KEY и REFERENCES) или

полудекларативных триггеров для обеспечения внутренней целостности приходится писать процедурный код. Очевидно, что оба эти недостатка связаны с отсутствием развитых средств манипулирования данными. Эта задача решается двумя способами - расширение ОО-языков в сторону управления данными (стандарт ODMG), либо добавление объектных свойств в реляционные СУБД (SQL-3, а также так называемые объектно-реляционных СУБД).

Заключение

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных. **Цель базы данных** -- помочь людям и организациям вести учет определенных вещей.

Список использованной литературы

1. Аткинсон М., Бансилон Ф., ДеВитт Д., Диттрих К., Майер Д., Здоник С. Манифест систем объектно-ориентированных баз данных. СУБД N 4, 2000

2.Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. - М.:"Финансы и статистика",2004.

3.Н.Вирт. Алгоритмы и структуры данных.-М.:"Мир",2007.

4. Медников А. Ю. Объектно-ориентированные базы данных сегодня или завтра? Открытые системы N 4,2009

5.Ким Вон Технология объектно-ориентированных баз данных. Открытые системы N 4, 1999

6.Сибуя М., Ямамото Т. Алгоритмы обработки данных.-М.:"Мир",2000

7. Пржиялковский В. Новые одежды знакомых СУБД: Объектная реальность, данная нам. СУБД N4, 2008

8.Кузнецов С. Д. Основы баз данных -- 2-е изд. -- М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. -- 484 с.

9.http://www.mstu.edu.ru

10.http://www.codenet.ru

11.http://citforum.ru