ЛЕКЦИЯ
«ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ БАЗ ДАННЫХ»

ВОПРОСЫ ЛЕКЦИИ

1. Введение в базы данных.
2. Системы управления базами данных.

ЛИТЕРАТУРА:

1. Введение в системы баз данных / Дейт К. Дж. – Издательство: Диалектика, 2019 г. – 1328 с.
2. СУБД для программиста. Базы данных изнутри/ Тарасов Сергей Витальевич – Издательство: Солон-пресс, 2018 г. – 320 с.
3. Базы данных. Учебник/ Шустова Лариса Ивановна, Тараканов Олег Владимирович – Издательство: ИНФРА-М, 2018 г. – 304 с.
4. Базы данных: учеб. пособие / О.Л. Голицына, Н.В. Максимов, И.И. Попов. — 4-е изд., 2019. — 400 с.

Введение

Технический прогресс не обошел стороной важную сферу технологического развития – информатику и вычислительную технику. Идет непрерывное совершенствование не только аппаратного обеспечения, но и программных средств. Чуть более двадцати лет назад работа с базами данных была уделом профессиональных программистов. Сами системы не были предназначены для простого пользователя. Основным потребителем таких систем был военно-промышленный комплекс. С появлением банков, акционерных обществ, частных компаний, рынка товаров и услуг базы данных нашли более широкое применение.

1. Введение в базы данных

1.1. Основные понятия

Э. Ф. Кодд, Британский основоположник теории реляционных баз данных, лауреат Премии Тьюринга, в своей публикации «A Relational Model of Data for Large Shared Data Banks» от 1970 года, построил свою теорию, основанную на взаимосвязи данных в таблицах. Так, по его мнению, одним из основных понятий в теории баз данных является информация. Информация – это один из важнейших ресурсов, это вся совокупность сведений об окружающем нас мире, о всевозможных протекающих в нем процессах, которые воспринимаются живыми организмами, электронными устройствами и другими информационными системами.

Данные – поддающееся многократной интерпретации представление информации в формализованном 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. Для долговременного хранения данных обычно используются базы данных.

Игнатьев М.Б. — лауреат Государственной премии РФ, доктор технических наук, а также заведующий кафедрой вычислительных систем Санкт-Петербургской академии выдвинул следующее определение баз данных: база данных (БД) – это комплекс упорядоченной, систематизированной и взаимосвязанной информации, составленной по определенным правилам, которые предполагают общие принципы описания, хранения и обработки данных.

Большинство учебников предлагает нам следующее определение. База данных (БД) это реализованная с помощью компьютера информационная структура, отображающая состояние объектов и их взаимосвязи.

Другими словами, это упорядоченное хранение каких-либо данных. То есть, информация хранится в классифицированном или сгруппированном виде. Типов систематизации, упорядочивания и хранения информации существует большое количество. Каждый из способов хранения информации отвечает каким-либо особым условиям или рассчитан на реализацию каких-либо конкретных операций.

Система управления базами данных – это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД – это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка - SQL.

SQL – язык структурированных запросов, основной задачей которого является предоставление простого способа считывания и записи информации в базу данных.

Основными составными элементами баз данных являются сущности (информационные объекты), связи между ними и их атрибуты (свойства).

Сущность – любой конкретный или абстрактный объект в рассматриваемой предметной области (объект, который мы можем отличить от другого), информацию о котором необходимо хранить в базе данных. Сущностями могут считаться студенты, клиенты, подразделения, места, самолеты, рейсы, вкус, цвет и т.д. Необходимо различать такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных личностей, предметов, событий или идей, выступающих как целое (например, студент, клиент и т.д.). Экземпляр сущности относится, например, к конкретной личности в наборе. Типом сущности может быть студент, а экземпляром – Петров, Сидоров и т. д.

Атрибут – это свойство или характеристика сущности в предметной области. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей

(например, атрибут «Цвет» может быть определен для многих сущностей: «Кошка», «Автомобиль» и т.д.). Атрибуты используются для определения того, какая информация должна быть собрана о сущности. Его наименование должно быть уникальным для конкретного типа сущности. Например, для сущности студент могут быть использованы следующие атрибуты: фамилия, имя, отчество, дата и место рождения, паспортные данные и т.д. Здесь также существует различие между типом и экземпляром. Тип атрибута «Цвет» имеет много экземпляров: «Красный», «Синий», «Белый» и т.д., однако, каждому экземпляру сущности присваивается только одно значение атрибута.

Связь – взаимосвязь между сущностями в предметной области. Связи представляют собой ассоциирование двух или более сущностей. Если бы назначением базы данных было только хранение отдельных, не связанных между собой данных, то ее структура могла бы быть очень простой. Однако одно из основных требований к организации базы данных – это обеспечение целостности данных, т.е. возможности отыскания одних сущностей по значениям других, для чего необходимо установить между ними определенные связи.

1.2. Виды баз данных

При создании базы данных, мы стремимся структурировать информацию по различным признакам для того, чтобы извлекать из нее нужные данные в удобном для пользователя сочетании. Это можно сделать только в том случае, если данные упорядочены. В зависимости от структуры различают следующие модели баз данных:

1. Иерархическую.
2. Сетевую.
3. Реляционную.
4. Объектно-ориентированную.

В иерархической БД данные представляются в виде древовидной

структуры. Подобная структура БД удобна для работы с данными, упорядоченными иерархически. При оперировании данными со сложными логическими связями иерархическая модель оказывается слишком громоздкой.

В сетевой БД данные организуются в виде графа. Недостатком сетевой структуры является жесткость структуры и сложность ее организации.

Реляционная БД получила свое название от английского термина relation (отношение). Реляционная БД представляет собой совокупность таблиц, связанных отношениями. Достоинствами реляционной модели данных являются простота, гибкость структуры. Кроме того ее удобно реализовывать на компьютере. Большинство современных БД для персональных компьютеров являются реляционными.

Объектно-ориентированные БД объединяют сетевую и реляционную модели и используются для создания крупных БД с данными сложной структуры.

Вышеперечисленные модели можно разделить на три поколения:

1. первого поколения: иерархические, сетевые;
2. второго поколения: реляционные;
3. третьего поколения: объектно-ориентированные.

1.3. Реляционная база данных

В курсе дисциплины мы разбирались в работе реляционных баз данных, поэтому заострим внимание именно на них.

Реляционная модель основана на математическом понятии отношения, физическим представлением которого является таблица. В любой реляционной БД предполагается, что пользователь воспринимает базу данных как набор таблиц. Однако следует подчеркнуть, что это восприятие относится только к логической структуре базы данных. Подобное восприятие не относится к физической структуре базы данных, которая может быть реализована с помощью различных структур.

Обрабатывая схемы БД, СУБД создает пустую базу данных требуемой структуры – хранилище, которое может быть заполнено данными о предметной области и запущено для удовлетворения информационных потребностей пользователей.

Реляционная база данных представляет собой множество взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного вида. Каждая строка таблицы содержит данные об одном объекте (например, автомобиле, компьютере, клиенте), а столбцы таблицы содержат различные характеристики этих объектов - атрибуты (например, номер двигателя, марка процессора, телефоны фирм или клиентов).

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

Для идентификации записей используется первичный ключ.Первичным ключом называется поле или набор полей, однозначно идентифицирующих запись.

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

Записи в таблице могут зависеть от одной или нескольких записей другой таблицы. Такие отношения между таблицами называются связями. Связь – ассоциирование двух или более сущностей. Сущность – это любой различимый объект, информацию о котором должна храниться в базе данных. Если бы назначением базы данных было только хранение

отдельных, не связанных между собой, то ее структура могла бы быть очень простой. Однако одним из основных требований для организации базы данных является предоставление возможности находить одни сущности по значениям других, для чего необходимо установить определенные отношения между ними. А поскольку реальные базы данных часто содержат сотни или даже тысячи объектов, теоретически между ними может быть установлено более миллиона связей.

Существует три типа связей между таблицами:

1. Один к одному — каждая запись родительской таблицы связана только с одной записью дочерней. Такая связь встречается на практике намного реже, чем отношение один ко многим и реализуется путем определения уникального внешнего ключа.

2. Один ко многим — каждая запись родительской таблицы связана с одной или несколькими записями дочерней. Например, один клиент может сделать несколько заказов, однако несколько клиентов не могут сделать один заказ. Связь один ко многим является самой распространенной для реляционных баз данных.

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

2. Системы управления базами данных

2.1. Функции систем управления базами данных

Системы управления базами данных имеют следующие основные функции:

1. Управление данными - можно установить доступ, кому можно ознакомиться с данными, изменять их или добавлять новую информацию.

2. Определение данных – есть возможность определить, какая именно информация будет храниться в базе данных, установить свойства данных, указать их тип (например, число символов или цифр), а также задать, как эти данные будут связаны между собой. В определенных случаях есть смысл задавать форматы и критерии проверки данных;

3. Обработка данных - данные могут быть обработаны многочисленными способами. У вас есть выбор в фильтровке и сортировке данных. Также вы можете объединить данные с другой связанной информацией и рассчитать итоговые значения.

Также принципиально важное свойство СУБД заключается в том, что она позволяет различать и поддерживать два независимых взгляда на БД – взгляд пользователя, воплощаемый в «логическом» представлении данных, и взгляд системы – «физическое» представление, характеризующее организацию хранимых данных. Пользователя не интересует при его работе с БД байты и биты, представляющие данные в среде хранения, их размещение в памяти, указатели, поддерживающие связи между структурными различными компонентами хранимых данных, выбранные методы доступа. В то же время эти факторы важны для выполнения функций управления данными самой СУБД.

Обеспечение логической независимости данных – одна из важнейших функций СУБД, предоставляющая определенную степень свободы вариации «логического» представления БД без необходимости соответствующей модификации «физического» представления. Благодаря этому можно адаптировать представление базы данных пользователя к его реальным

потребностям, создавать различные «логические» представления для одной и той же «физической» базы данных, что очень важно в социальной пользовательской среде.

«Физическая» независимость данных означает способность СУБД предоставлять некоторую свободу модификации способов организации БД в среде хранения, не вызывая необходимости внесения соответствующих изменений в «логическое» представление. Благодаря этому можно вносить изменения в организацию хранимых данных, настраивать систему с целью повышения ее эффективности, не затрагивая созданных прикладных программ, использующих базу данных. «Физическая» независимость данных реализована в СУБД благодаря тем же механизмам преобразования архитектуры системы, которые обеспечивают «логическую» независимость данных.

Поддержка логической целостности (непротиворечивости) базы данных – другая важная функция СУБД. В развитых системах ограничения целостности базы данных объявляются в схеме базы данных, и их проверка осуществляется при каждом обновлении объектов данных или связей между ними, являющихся аргументами таких ограничений.

СУБД обычно блокирует действия, которые нарушают целостность связей между таблицами, т.е. нарушают ссылочную целостность. Когда говорят о ссылочной целостности, имеют в виду совокупность связей между отдельными таблицами во всей БД. Нарушение хотя бы одной такой связи делает информацию в базе данных недостоверной.

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

1. при изменении поля связи в записи родительской таблицы следует синхронно изменить значения полей связи в соответствующих записях дочерней таблицы;

2. при удалении записи в родительской таблице следует удалить

соответствующие записи в дочерней таблице.

Изменения или удаления в записях дочерней таблицы при изменении записи родительской таблицы называются каскадными изменениями.

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

2.2. Виды систем управления базами данных

В настоящее время для построения информационных систем применяются различные системы управления базами данных, различающиеся как своими возможностями, так и требованиями к вычислительным ресурсам. Однако все разнообразие прикладных СУБД можно сократить до двух основных классов: персональных и многопользовательских.

К первому классу относились СУБД, ориентированные для работы на персональном компьютере. На каждом рабочем месте работает собственная копия программы - СУБД и прикладная программа, и на их выполнение могут оказывать существенное влияние характеристики компьютера этого рабочего места.

СУБД второго класса изначально создавались для выполнения на больших компьютерах и обеспечения параллельной работы многих пользователей. Такие СУБД, как правило, состоят из ядра (сервера), постоянно присутствующего в памяти, и большого количества программ-агентов (клиентов), обслуживающих запросы конечных пользователей и прикладных программ. Единая управляющая система позволяет эффективно организовать одновременный доступ к данным многих агентов, предотвращая конфликты между ними. Многопользовательские СУБД обладают также неоспоримыми преимуществами в таких аспектах, как надежность, безопасность, доступность.

Также есть и другие категории классификации СУБД.

По архитектуре организации хранения данных:

1. Локальные СУБД. Все части локальной СУБД размещаются на одном компьютере.

2. Распределенные СУБД. Части СУБД могут размещаться на двух и более компьютерах.

По способу доступа к БД:

1. Файл-серверные. В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. Ядро СУБД располагается на каждом клиентском компьютере. Доступ к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на ЦП сервера, а недостатком - высокая загрузка локальной сети. На данный момент файл-серверные СУБД считаются устаревшими.

2. Клиент-серверные. Такие СУБД состоят из клиентской части и сервера. Клиент-серверные СУБД, в отличие от файл-серверных, обеспечивают разграничение доступа между пользователями и мало загружают сеть и клиентские машины. Недостаток клиент-серверных СУБД в больших вычислительных ресурсах, потребляемых сервером.

Работа приложения сильно зависит от использования той или иной архитектуры. При работе с локальными базами данных сами базы данных расположены на том же компьютере, что и приложения, осуществляющие доступ к ним. Работа с базой данных происходит в однопользовательском режиме. Приложение ответственно за поддержание целостности базы и за выполнение запросов к базе данных.

При работе в архитектуре "файл-сервер" база данных и приложение расположены на файловом сервере сети. Возможна многопользовательская работа с одной и той же базой данных, когда каждый пользователь со своего компьютера запускает приложение, расположенное на сетевом сервере.

Тогда на компьютере пользователя запускается копия приложения. По каждому запросу к базе данных из приложения, данные из таблиц базы данных, перегоняются на компьютер пользователя, независимо от того, сколько реально нужно данных для выполнения запроса. После этого выполняется запрос.

Каждый пользователь имеет на своем компьютере локальную копию данных, время от времени обновляемых из реальной базы данных, расположенной на сетевом сервере. При этом изменения, которые каждый пользователь вносит в базу данных, могут быть до определенного момента неизвестны другим пользователям, что делает актуальной задачу систематического обновления данных на компьютере пользователя из реальной базы данных. Другой актуальной задачей является блокирование записей, которые изменяются одним из пользователей - это необходимо для того, чтобы в это время другой пользователь не внес изменений в те же данные.

В архитектуре «файл-сервер» вся тяжесть выполнения запросов к базе данных и управления целостностью базы данных ложится на приложение пользователя. База данных на сервере является пассивным источником данных.

2.3 Виды систем управления базами данных

Существует множество способов реализации систем управления базами данных (СУБД). Не обращая внимания на то, что все системы управления базами данных реализуют единственную главную задачу (т.е. позволяют пользователю создавать, редактировать и получать доступ к данным, находящихся в базах данных), процедура осуществления этой основной задачи колеблется в широких границах. Помимо этого, опции и возможности каждой СУБД могут значительно различаться. По-разному предоставляется и техническая поддержка.

В настоящее время существует несколько наиболее популярных СУБД:

MS Access, MySQL, MS SQL Server, MongoDB, Oracle Database и др.

Microsoft Office Access – реляционная система управления базами данных корпорации Microsoft. Входит в состав пакета Microsoft Office. Имеет широкий спектр функций, включая связанные запросы, связь с внешними таблицами и базами данных. Благодаря встроенному языку VBA, в самом Access можно писать приложения, работающие с базами данных.

Достоинства:

1. очень простой графический интерфейс, позволяющий не только создавать собственную базу данных, но и разрабатывать приложения, используя встроенные средства;

2. позволяет хранить все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных;

3. предлагает большое количество Мастеров, которые выполняют основную работу за пользователя при работе с данными и разработке приложений, помогают избежать рутинных действий и облегчают работу пользователю;

4. наличие встроенного языка макрокоманд.

Недостатки в новых версиях:

1. нет файла рабочих групп (mdw), что усложняет разграничение прав пользователей и построение защиты БД.

2. лента резко ограничила возможности по созданию собственного меню.

MongoDB – это документоориентированная система управления базами данных с открытым исходным кодом, не требующая описания схемы таблиц. Классифицирована как NoSQL, использует JSON-подобные документы и схему базы данных. Написана на языке C++. Система поддерживает ad-hoc-запросы, которые могут возвращать конкретные поля документов и пользовательские JavaScript-функции. Поддерживается поиск

по регулярным выражениям. Также можно настроить запрос на возвращение случайного набора результатов. Система может работать с набором реплик, то есть, содержать две или более копии данных на различных узлах. Каждый экземпляр набора реплик может в любой момент выступать в роли основной или вспомогательной реплики. Все операции чтения и записи по умолчанию осуществляются с основной репликой. Вспомогательные реплики поддерживают в актуальном состоянии копии данных. Если основная реплика дает сбой, набор реплик проводит выбор другой реплики, которая должна стать основной. Второстепенные реплики могут дополнительно являться источником для операций чтения. Система масштабируется горизонтально, используя технику сегментирования объектов баз данных — распределение их частей по различным узлам кластера. Администратор выбирает ключ сегментирования, определяющий, по какому критерию данные будут разнесены по узлам (в зависимости от значений хэша ключа сегментирования). Благодаря тому, что каждый узел кластера может принимать запросы, обеспечивается балансировка нагрузки. Систему можно использовать в качестве файлового хранилища с балансировкой нагрузки и репликацией данных (функция Grid File System; поставляется вместе с драйверами MongoDB). Предоставляются программные средства для работы с файлами и их содержимым. GridFS используется в плагинах для Nginx и lighttpd. GridFS разделяет файл на части и хранит каждую часть как отдельный документ. Может работать в соответствии с парадигмой MapReduce. Во фреймворке для агрегации есть аналог SQL-инструкции GROUP BY. Операторы агрегации могут быть связаны в конвейер подобно UNIX-конвейрам. Фреймворк так же имеет оператор $lookup для связки документов при выгрузке и статистические операции такие как среднеквадратическое отклонение.

MySQL — это свободная реляционная система управления базами данных. Разрабатывает и поддерживает MySQL корпорация Oracle, которая получила права на торговую марку вместе с поглощённой Sun Microsystems,

ранее приобретшей шведскую компанию MySQL AB. Продукт распространяется как под собственной коммерческой лицензией, так и под GNU General Public License. Кроме того, по заказу лицензионных пользователей разработчики создают функциональность. Благодаря этому заказу даже в самых ранних версиях появляется механизм репликации. MySQL это решение для средних и малых приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, а также в дистрибутив входит библиотека внутреннего сервера, которая позволяет включать MySQL в автономные программы. Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Кроме того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

Основные преимущества:

1. простота в использовании, MySQL достаточно легко инсталлируется, а наличие множества плагинов и вспомогательных приложений упрощает работу с базами данных;

2. безопасность – система изначально создана таким образом, что множество встроенных функций безопасности в ней работают по умолчанию;

3. обширный функционал – система MySQL обладает практически всем необходимым инструментарием, который может понадобиться в реализации практически любого проекта.

Недостатки:

1. недостаточная надежность – в вопросах надежности некоторых процессов по работе с данными (например, связь, транзакции, аудит) MySQL

уступает некоторым другим СУБД;

2. низкая скорость разработки – как и многим другим программным продуктам с открытым кодом, MySQL не достает некоторого технического совершенства, что порой сказывается на эффективности процессов разработки.