**Московский государственный технический университет**

**им. Н.Э. Баумана**

Кафедра

«Системы обработки информации и управления»

(ИУ – 5)

**Пояснительная записка**

**курсовой работы на тему «Банки данных»**

**по дисциплине «Архитектура АСОИУ»**

Выполнил:

студент гр. ИУ5 - 34

Кучеренко Михаил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«30» ноября 2017 г.

Оценка «\_\_\_\_\_\_\_\_\_\_»

Проверил:

к.т.н., доцент

Шук В.П.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2017 г.

**Москва – 2017 г.**

**Тема № 101.** Банки данных

А. Локальные банки данных

Б. Распределённые банки данных

В. Банки знаний

Литература: Григорьев Ю.А. и др. Банки данных. М.: Изд-во МГТУ им. Н.Э.Баумана. 2002. – 318 с.

Шифр Российской государственной библиотеки **1 02-5/231-2;232-0**

Ключевые слова реферата: основы построения, модели данных, модель, иерархическая, сетевая, реляционная, специальная обработка, знания, представление знаний, база знаний, даталогическое проектирование, распределённая обработка, тенденции, потребности конечных пользователей, проектирование, концептуальное, логическое, проблемы.

Позиция темы в базовой морфологической модели архитектуры АСОИУ:

**Реферат**

**на тему «Банки данных»**

(тема № 101)

Данной теме посвящена работа [4]. В книге рассматриваются **основы построения** банков данных. В первой части книги - методы построения локальных банков данных, во второй части – распределенные банки данных.

После того, как началась разработка автоматизированных информационных систем (АИС), предназначенных для регистра­ции, хранения и переработки информации с целью поиска и выдачи ответов на запросы пользователей, возникла необходимость **cпециальной обработки** и создания банков данных. Банк данных – это АИС, которая включает в себя целый комплекс средств и методов, позволяющих поддерживать динамику информационной модели какой-либо конкретной предметной области для того, чтобы обеспечивать информационные запросы пользователей.

Развитие теории и практики проектирования и эксплуатации банков данных, со­провождается интенсивным развитием **моделей данных**. В [4] рассматривается, что на практике применяются следующие модели данных: иерархическая, сетевая или реляционная. Эти модели представляют типовые структуры, отличающиеся между собой. Так, например, **иерархическая – модель** древовидных структур, представляющая из себя различные классификаторы и основанная на использовании графовых и табличных форм. **Реляционная – модель** множества данных фиксированной длины. **Сетевая** - представляет собой **модель** ориентированных данных и использует графовую форму представления.

Исходными данными для **даталогического проектирования** является инфологическая модель предметной области. Конечным результатом даталогического проектирования является описание логической структуры базы данных на языке описания данных, поддерживаемом конкретной СУБД. Проектирование даталогической модели для реляционной базы данных сводится к разбитию всей информации на файлы. В каждом файле необходимо определить состав полей. Файлы в терминах реляционной модели определяются как терминалы, а поля как атрибуты.

Вторая часть работы [4] посвящена **распределенной обработке** данных. А заключается она в том, что пользователь может работать с прикладными процессами и сетевыми службами одновременно в нескольких абонентских системах, которые связаны друг с другом, и являются, потребителями и поставщиками информации. Основу системы распределенной обработки данных составляет распределенная система управления базой данных.

Системы распределенной обработки информации получают в настоящее время повсеместное развитие. **Тенденции** развития связана с информационными системами и влиянием на них бизнес-архитектуры. Сейчас используются три основные модели систем распределенной обработки информации: модель файлового сервера, модель сервера базы данных, модель сервера приложений.

Как показывает практика, потребность в распределенных базах данных возрастает с каждым днем. Прогнозируется, что скоро в различных областях будут функционировать сотни миллионов баз данных. Все они должны быть распределенными, самоуправляемыми и сбалансированными. Еще одно важное условие развития - это то, что взаимодействие между базами данных должно быть высоконадежным и полностью автоматическим.

Важнейшим этапом жизненного цикла является анализ требований разрабатываемой системы. Данный этап в дальнейшем оказывает сильное влияние на все последующие. И здесь возникает необходимость понять **потребности конечных пользователей**, задокументировать их и изложить в понятной заказчику форме. Во многих аспектах системный анализ является наиболее трудной ча­стью разработки.

Одним из этапов **проектирования** распределенных систем – является **концептуальное проектирование.** Данное проектирование требует детальной проработки, под воздействием двух факторов – цены ошибки и независимости от архитектуры. Схематически концептуальное программирование можно представить диаграммой «сущность – связь». Она представляет собой стандартный способ определения данных и отношений между ними. С помощью данной диаграммы производится документирование системы, детализация хранилищ данных, а также отношение с другими объектами.

Еще одним этапом проектирования распределенных систем является **логическое проектирование**. На этом этапе происходит отображение концептуального проекта в СУБД-ориентированную среду. По сути логическое проектирование - это процесс конструирования общей информационной модели предприятия на основе отдельных моделей данных пользователей, которая является независимой от особенностей реально используемой СУБД и других физических условий.

Важнейшей **проблемой**, которую необходимо решать при распределенной обработке данных – это уменьшение времени на обработку запроса. Основными ресурсами, из-за которых возникают перегрузки системы являются: шина или канал сети, процессор сервера и жесткий диск.

Решение проблем интеграции обработки данных привело к возникновению банков знаний. На этапе решения задач возникает необходимость преобразовать имеющуюся информацию в такую форму, которая позволит совершить машинную обработку. Для этих целей используются модели **представления знаний**. Особенностью **баз знаний** является присутствие интеллектуального интерфейса. Такой интерфейс позволяет преобразовать поставленную пользователем задачу и решить её. В настоящий момент существует три основных способа представления **знаний**: фреймовые системы, семантический сети и логическая модель.

Вопросы основных концепций локальных, распределенных банков данных, а также банков знаний будут более детально рассмотрены в пояснительной записке курсовой работы.

**Оглавление**

Стр.

1. Введение …………………………………………………………… 5
2. Локальные банки данных ……………………………….. 6
3. Распределенные банки данных ………………………. 13
4. Банки знаний …………….……………………………………… 19
5. Заключение ………………………………………………………. 24
6. Литература ……………………………………………………….. 25
   1. **Введение**

Целями выполнения индивидуального задания являются:

- углубление и расширение теоретических знаний, полученных на лекциях;

- приобретение первоначального опыта самостоятельной работы с научно-технической литературой, включая её поиск, анализ и синтез;

- практическое освоение техники грамотного изложения результатов изучения специальной литературы по заданной теме с учётом существующих норм и требований, предъявляемых к научно-техническому тексту.

* 1. **Локальные банки данных.**

Банк данных (БнД) – это АИС (автоматизированная информационная система), которая включает в себя целый комплекс средств и методов, позволяющих поддерживать динамику информационной модели какой-либо конкретной предметной области для того, чтобы обеспечивать информационные запросы пользователей. Банк данных содержит две основные части: базу данных (БД) и систему управления базой данных (СУБД), с помощью которой и осуществляется централизованное управление данными, доступ к ним, а также поддержание их в рабочем состоянии, которое соответствует состоянию данного программного обеспечения.

Услугами локального банка данных пользуется обычно очень большое количество пользователей. По этой причине в нем предусмотрено приведение запросов к единой терминологии – словарю данных. Также сформированы методы доступа к данным различных пользователей, если поступившие запросы совпали по времени и разработаны методы грамматических преобразований запросов для оптимизации их обработки.

Если рассматривать данные как один из ресурсов АС, нужно отметить, что БнД централизованно управляет данным ресурсом в интересах всей системы. Централизованное управление - главная отличительная черта банка данных. То есть можно сказать, что БнД– это информационная система, которая осуществляет централизованное управление данными в интересах пользователей АС, в состав которой она входит.

БнД, как система управления, состоит из объекта управления и управляющего органа. Объектом управления в БнД является база данных, а управляющим органом – группа специалистов, которая посредством СУБД реализует управление БД. Такая группа специалистов является АБД (администратором базы данных) и должна обладать знаниями по специфике предметной области (ПО) данной АИС, а также быть знакома с теорией систем обработки данных. Сделать БнД полностью автоматической системой для всего жизненного цикла АС пока невозможно. АС, в состав которой входит БнД, обеспечивает определенный вид человеческой деятельности и является развивающейся системой. Поэтому постоянно возникает необходимость вносить корректировки в границы предметной области, а это может быть сделано только специалистом.

Развитие теории и практики проектирования и эксплуатации банков данных, сопровождается интенсивным развитием моделей данных (МД). В [4] рассматривается, что обычно применяются следующие МД: иерархическая, сетевая или реляционная. Следует отметить, что не существует СУБД, которые поддерживали бы только одну определенную модель данных. Ка правило, языки СУБД ориентированы на определенную модель данных, но при этом предусмотрена возможность работать и с другими МД.

Иерархическая модель – модель древовидных структур, представляющая различные классификаторы и основанная на использовании графовых и табличных форм. Сетевая модель представляет собой модель ориентированных данных и использует графовую форму представления. Реляционная модель – множество данных фиксированной длины.

В работе [4] рассмотрены двухуровневый и трехуровневый подходы к построению БнД. При двухуровневом подходе пользователи составляют свои прикладные программы, используя только термины МД. СУБД получает запрос из прикладной программы (ПП), организует запрос к операционной системе на считывание из физической БД (ФБД) необходимых данных в свою буферную память. В буферной памяти СУБД оказываются записи, имеющие структуру, соответствующую схеме внутренней модели БД (ВнМД): Модель ВнМД ФБД.

После осуществляется отображение хранимых записей в записи модели, а затем затребованные записи модели передаются СУБД а рабочую область ввода-вывода ПП. Такой подход демонстрирует определенную независимость системы и технических средств. Но, данная схема предусматривает ознакомление пользователя с информацией всей БД. Такая ситуация крайне неудобна. Многие пользователи совсем не нуждаются в таком изучении и при этом еще возникает необходимость в защите данных от действий некомпетентных пользователей.

Трехуровневый подход построения БД подразумевает что СУБД реализует обмен данными между рабочей областью ввода-вывода ПП и БД. Любой запрос поступает в СУБД, а та в свою очередь обращается к операционной системе для выполнения действий, имея описание моделей и связей между ними.

ВМД КМД ВнМД ФБД, где ВМД – внешняя модель данных, КМД – концептуальная модель данных. Данный подход к построению БнД, включающий внешний, внутренний и концептуальный уровни, получил самое массовое распространение. При таком взаимодействии на внешнем уровне поддерживается лишь ограниченная предметная область, которая видима для ограниченных приложений или пользователей. На концептуальном уровне поддерживаются все предметные области. Хранимые данные выделены в отдельный внутренний уровень. Именно такая архитектура построения позволяет БнД быть способным к адаптации и изменениям в ПП или любых данных. Такие изменения могут быть внесены независимо друг от друга.

Основными этапами проектирования БД являются: инфологическое и датологическое. Датологическое проектирование в свою очередь подразделяется на: логическое и физическое. В зависимости от определенного этапа можно выделить инфологическую и датологическую КМД, а также инфологическую и датологическую ВМД.

На этапе инфологического проектирования информационной системы решаются следующие вопросы:

1. О каких явлениях или объектах реального мира необходимо накапливать информацию.
2. Какие характеристики и взаимосвязи между собой должны будут учитываться.
3. Какие понятия о явлениях, объектах или их характеристиках, вводимые в систему нуждаются в уточнении.

На инфологическом этапе проектирования БД осуществляется получение смысловых МД, которые отражают информационное состояние конкретной ПО. Сначала выделяют нужную часть ПО, отделяют её от несущественных частей (для данного случая) БнД и определяют её границы. В процессе этих действий определяются необходимые объекты и их связи, существенные для пользователей. После того как ПО изучена, производят её представление в языковой системе. Как правило, это описание на естественном языке с использованием диаграмм, рисунков или формул. Концептуальная инфологическая модель необходима для обеспечения эффективной и долговременной работы всей системы, а также для возможности замены используемой СУБД на другую.

На логическом этапе проектирования необходимо преобразовать данные, выделенные на первом этапе в форму, которая будет восприниматься СУБД. То есть нужно разработать схемы КМД и ВМД, средствами которые поддерживаются этой СУБД. На этом этапе обычно не рассматриваются вопросы доступа к данным. Однако, чтобы избежать ошибок проектирования в дальнейшем, лучше иметь представление о желаемых методах доступа.

На физическом этапе проектирования осуществляется выбор оптимальной структуры хранения данных и доступа к ним. Такой выбор опирается на имеющиеся методы и средства.

На датологическом этапе проектирования рассматриваются вопросы представления данных в памяти информационной системы. То есть исходя из имеющихся возможностей и средств представления и хранения информации, разрабатываются формы представления информации, модели и методы данного представления, а также формируются правила смысловой интерпретации данных.

Проектировщик БД, закончив первый вариант инфологического проектирования, подготавливает датологический проект для выбранного типа модели данных, а затем уже выбирает конкретную СУБД и заканчивает датологическое проектирование БД, используя уже модель данных этой конкретной СУБД. Формирование структуры данных в конкретной модели производится по определенным правилам. Каждая модель имеет свои ограничения. Поэтому основными компонентами МД являются структуры, операции и ограничения. Такие компоненты тесно связаны между собой и в различных моделях реализуются по-разному.

Когда проектировщик начинает моделирование ПО, он моделирует отдельные локальные представления, а потом их объединяет. В результате такой работы строится модель локального представления, которая зависит от масштабов ПО.

Для оптимизации проектирования желательно, чтобы в отдельной локальной модели использовалось не более шести-семи типов сущностей. Проектировщик должен принимать компромиссное решение при выборе области локального представления. Обширная область приведёт к сложности проектирования и нечеткости, а узкая область – к дроблению данных и снижению уровня интеграции.

Необходимо четко формулировать сущности, которые потребуются для описания конкретного локального представления, то есть должны быть указаны типы объектов ПО, о которых должна накапливаться информация в системе. Каждой выбранной сущности должно соответствовать четкое наименование. Лучше всего, если оно будет отражать смысловое значение вводимого понятия. Нечеткие наименования, наличие синонимов или омонимов недопустимы, так как приводят к ошибкам в проектировании. При моделировании локального представления должно быть выполнено распознавание категорий и представление каждой из них в виде самостоятельной сущности.

Такое распознавание производится с помощью концепции типа или роли. Например, содержание сущности «Учащийся вуза» может разделяться по категориям типов: «Студент», «Бакалавр», «Магистр». Данные категории могут быть обобщены в сущность «Учащийся вуза» на более позднем этапе объединения локальных сущностей. Для оптимальной работы системы общее количество сущностей в отдельном представлении должно быть ограничено.

После определения всех необходимых сущностей устанавливаются связи между ними. Выявляется какие связи важны, а какие избыточны. Для установления связей используются их характеристики. Каждый тип связи «сущность – сущность» именуется. Также устанавливаются спецификация связей «сущность – атрибут» (указывается атрибут и сущность, для описания которых он используется) и связь типа «атрибут – атрибут» (отношение между атрибутами, которые относятся к одной и той же сущности или к связи типа «сущность – сущность»).

Локальные представления в дальнейшем объединяются. Целью таких объединений являются:

1. устранение несущественных различий;
2. объединение в единое представление фрагментов о различных свойствах одного и того же объекта;
3. образование производных типов объектов, соответствующих объединению, пересечению ил разности исходных объектов;
4. образование классов или подклассов подобных объектов.

Подобные объединенные конструкции должны обеспечивать непротиворечивость представления данных. Перед объединением прежде всего решается вопрос о порядке объединения моделей локального представления. Применение подобного рода объединений позволяет обеспечить доступ пользователей к данным с использованием различных уровней абстракции. Такая практика способствует повышению гибкости системы при совместном использовании, а при формировании поиска в соответствии со смысловыми категориями, существенно уменьшается пространство, к которому обращается система.

Процесс объединения носит итеративный характер. При объединении выявляются противоречия между локальными представлениями (неполная спецификация, некорректные требования, наличие омонимов или синонимов), что позволяет устранить их.

После появления и распространения централизованных баз данных, возникла необходимость в осуществлении защиты данных, то есть недопущение специального или случайного доступа к данным, их изменение или уничтожение, а также потеря данных при сбоях программных или аппаратных средств. В большинстве случаев принято, что СУБД не позволяет обычным пользователям производить какие-либо операции над БД. АБД, в чьи полномочия входит осуществление соответствующей защиты БД, решают следующие вопросы:

1. организация систем контроля доступа к данным;
2. классификация всех данных в соответствии с их использованием;
3. определение прав доступа пользователей к отдельным группам данных и установление ограничений на определенные операции для этих пользователей;
4. тестирование всех создаваемых средств защиты данных;
5. исследование современных технологий в области защиты данных и применение их для конкретного БнД;
6. исследование всех случаев нарушения защиты данных с целью их повторного предотвращения.

Администратору БД или сотрудникам отдела необходимо досконально знать структуру, состав и прочие характеристики системы для того, чтобы оперативно реагировать и устранять неполадки и сбои в системе.

Основными методами и приемами защиты данных являются:

1. Идентификация пользователя.

Перед началом работы пользователь должен идентифицировать себя и подтвердить свою идентификацию. Данный процесс – это по сути обмен между пользователем и системой информацией, известной только им. Процесс подтверждения использует простую процедуру, но раскрыть её очень сложно. Могут быть использованы одноразовые пароли или ответы на некоторые вопросы и даже реализация каких-либо алгоритмов.

1. Управление доступом.

В системе на каждого пользователя имеется паспорт, составленный АБД и содержащий информацию о идентификационном номере, процедуре распознавания подлинности, а также перечень возможностей доступа и разрешенные операции.

1. Защита данных при статистической обработке.

Часто запросы на различную статистическую информацию (например, о возрасте сотрудников) влекут за собой обращение к анкетным данным,

которые уже являются закрытой информацией. Чтобы запрашиваемая информация все-таки предоставлялась пользователям и при этом не были использованы закрытые источники, необходимо ввести ограничения на структур запросов. Это позволит исключит возможность раскрытия данных.

1. Физическая защита.

Самый распространенный способ несанкционированного физического вмешательства – это похищение пакета данных. Защитой от подобного рода вмешательств является использование методов кодирования данных. Еще одна задача физической защиты – оградить системы от последствий аппаратных и программных сбоев.

Наряду с защитой данных стала актуальной проблема обеспечения целостности данных. Данная проблема заключается в обеспечении правильности предоставляемой пользователям информации в любой момент времени. Существует два основных вида ограничения целостности, поддерживаемые СУБД:

- Ограничения реальных значений данных, которые хранятся в базе.

Эти ограничения, как правило, требуют, чтобы значения поля

соответствовали определенному диапазону значений или выражали бы

арифметическое соотношение между значениями различных полей.

- Структурные ограничения.

Такие ограничения сверяются с помощью проверки равенства значений

соответствующих данных и задаются функциональными зависимостями.

Есть и другие ограничения целостности. Например, при переходе БД в обновленное состояние, ограничение при выполнении параллельных операций над данными. Целостность также нередко нарушается при ошибке в работе оператора, сбоях аппаратного оборудования, могут быть и ошибки в исходных данных.

Главная задача обеспечения ограничений целостности данных состоит в использовании языка манипулирования данными (ЯМД), который по сути является средством выражения данных ограничений. Декларация ограничения целостности состоит из двух частей. В первой выражается само ограничение, а во второй то, когда и при каких условиях должна проводиться сама проверка. ЯМД как раз и позволят поддерживать указанные ограничения целостности.

Еще одной большой проблемой при использовании БД являются множество параллельных запросов. Самым ярким примером можно назвать систему бронирования и продажи авиа или железнодорожных билетов. В таких системах очень часто возникают ситуации, когда несколько операторов пытаются одновременно произвести определенные действия в системе. Чтобы в таких случаях устранить нарушение ограничения целостности необходимо ввести разграничение доступа транзакций в базе (например, путем введения блокировок доступа к данным). Для того чтобы реализовать подобное блокирование БД необходимо разбить на несколько элементов, которые можно блокировать отдельно.

При этом размер элементов должен быть оптимален, так как большой размер заблокированного элемента снижает возможности параллельного использования, а малый – увеличивает затраты ресурсов системы.

Чтобы осуществлять блокировки в состав СУБД вводится функциональный модуль для управления блокировками. Данный модуль собственно и выполняет блокирование, планирует и контролирует блокировки, а также решает конфликтные ситуации.

Для блокировок применяют простую модель и модель с блокировками для чтения и записей, которой существует разграничение между доступами к элементам (доступ для чтения и записи или доступ только для чтения). При блокировании элементов древовидной структуры обычно используют одну из двух схем:

- блокировка элемента, с одновременным блокированием всех его

«потомков»;

- блокировка элемента, без блокировки его «потомков».

Часто при использовании блокировки приходится сталкиваться с проблемами тупиковых ситуаций и бесконечного ожидания. Проблема бесконечного ожидания может появиться в любой системе с параллельными запросами. Для ее решения системе блокировок необходимо регистрировать все входящие запросы и впоследствии производить блокировку по правилу очередности. Тупиковые ситуации также возникают при параллельном выполнении процессов. Существует три пути разрешения тупиковых ситуаций:

1. В систему вводится требование, чтобы все блокировки запрашивались сразу. При таком ограничении каждая из транзакций единовременно запрашивает необходимые блокировки. Такой подход позволяет СУБД управлять транзакциям без тупиковых ситуаций.
2. В системе вводится линейное упорядочение элементов и требование на составление программ, то есть все программы будут выполнять блокировки в определенном порядке. В этом случае транзакции будут запрашивать блокировки в едином строгом порядке, что и позволит избежать тупиковых ситуаций.
3. В систему не вводится никаких требований на написание программ. СУБД следит за возникновением тупиковых ситуаций и при обнаружении таковой одна из транзакций полностью аннулируется. Действия ею произведенные устраняются, а она сама переводится в режим ожидания, с последующим рестартом либо полном отменой.
4. **Распределенные банки данных.**

Опираясь на современные реалии системы распределенной обработки данных получают массовое развитие. Сильное влияние на это оказывает развитие бизнес-архитектуры.

Распределенная обработка данных состоит в том, что пользователь имеет возможность работать с прикладными процессами и сетевыми службами, расположенными в нескольких абонентских узлах.

Система распределенной обработки данных (СРОД) использует три основные модели доступа к данным:

1. модель файлового сервера;
2. модель сервера базы данных;
3. модель сервера приложений.

При использовании модели файлового сервера все приложения выполняются на рабочих станциях. Приложения состоят из модулей диалога с пользователями, бизнес-правила и ядро СУБД, которое зачастую не является ярко выраженным. На файловом сервере хранятся только файлы БД. Все данные дублируются на различных рабочих станциях. Запросы, закодированные в ПП, обрабатываются СУБД на рабочих станциях и СУБД же обеспечивает доступ к файлам БД. Все необходимые запросы передаются по сети.

Модель сервера базы данных похожа на модель файлового сервера тем, что большая часть приложений выполняется на рабочих станциях. Приложения состоят из модулей диалога с пользователя и бизнес-правил. Ядро СУБД функционирует на сервере и является общим для всех рабочих станций. Запросы пересылаются на сервер и выполняются уже там. Обратно на рабочую станцию пересылаются уже результаты запроса.

Модель сервера приложений организовывается с помощью хранимых процедур. Для этого возникает необходимость использовать языки высокого уровня, что существенно увеличивает количество задействованных ресурсов. У этого модели есть существенный недостаток: использование языков высокого уровня не позволяет произвести «тонкую» обработку данных. Поэтому для устранения данного недостатка приходится использовать дополнительные программы, которые часто называют менеджерами транзакций или мониторами транзакций.

С развитием современных технологий возникает необходимость построения сложных систем распределенной обработки. Такая возможность возникает в связи с тем, что современные СУБД становятся «открытыми системами». А такие системы обладают следующими характеристиками:

1. переносимость;
2. межоперабельность;
3. раширяемость;
4. масштабируемость;
5. интернационализация.

Переносимость – разделение программной системы и операционного окружения, позволяющее произвести перенос программы в другое операционное окружение. Переносимость обычно рассматривается в трех возможных вариациях:

* + функционирование СУБД на множестве платформ;
  + средства разработки приложений СУБД позволяют производить перенос готовых приложений с платформы на платформу без всяких измененений исходного кода;
  + коммуникационные средства обеспечивают связь между сервером и клиентами способом, который не зависит от платформ сервера или клиента и от используемого протокола.

Межоперабельность – возможность встраивания системы как элемент в сложную распределенную информационную среду. Это можно делать с помощью специальных решений, а также при помощи интерфейсов межнациональных, международных и промышленных стандартов. Такое свойство СУБД может означать следующее:

* + свойства сервера СУБД позволяют ему быть менеджером ресурсов с системе обработки распределенной информации;
  + приложения, созданные данной СУБД, имеют возможность оперировать над БД в «чужом» формате, как будто это собственные БД;
  + СУБД обладает свойствами, которые позволяют ей быть поставщиком данных для любых приложений, поддерживающих определенные стандарты обращения к БД.

Расширяемость – возможность качественного увеличения функций сервера СУБД. Данной свойство используется разработчиками в следующих целях:

* + для внедрения в сервер возможности обработки новых типов данных;
  + для программирования процедур, позволяющих расширить возможности сервера СУБД;
  + для обеспечения нового взаимодействия клиента и сервера СУБД, при этом сервер получает возможность влиять на клиента.

Масштабируемость – способность СУБД к адаптации в условиях усложнения и увеличения задач и запросов к БД. Возможности к масштабированию закладываются разработчиками заранее, если они предусмотрели такое развитие событий. Существует вертикальное и горизонтальное масштабирование. Вертикальное масштабирование подразумевает возможность замены платформы СУБД на новую, с большей мощностью и надежностью. Горизонтальное масштабирование означает возможность добавления новых компьютеров, что означает увеличение производственной мощности системы.

Интернационализация – приемы разработки, упрощающие адаптацию программного и аппаратного обеспечения к особенностям региона отличных от того, в котором разрабатывался продукт. Существует несколько аспектов СУБД, означающих возможность интернационализации:

* + Количество бит для представления символа. В профессиональных СУБД символ, как правило, составляет несколько байт. Это позволяет охватывать весь спектр символов национальных алфавитов.
  + Ресурсы, связанные с представлением национального алфавита, должны быть вынесены из ядра СУБД в отдельную файловую систему. Соответственно выбор алфавита должен осуществляться установкой соответствующих значений переменных или параметров конфигурации.
  + Процесс локализации должен состоят не только из локализации ресурсов, но также из тестирования сервера и всех средств разработки на тех платформах, где будет производиться эксплуатация системы. Локализованная версия системы должна быть сертифицирована, а её кодировки официально поддерживаться.

Проектирование систем распределенной обработки данных (СРОД) состоит из следующих этапов:

1. выявление информационных потребностей конечных пользователей;
2. концептуальное проектирование;
3. разработка архитектуры СРОД;
4. логическое проектирование;
5. отладка и тестирование прикладных программ;
6. сопровождение.

На первом этапе проектирования производится анализ ПО после чего строится функциональный граф, который связывает функции системы с входными и выходными данными. В современных компьютерах происходит разделение процессов и данных. Поэтому разработчики должны производить отделение функции от данных после первого этапа. А дальнейшая работы по данным и процессам протекает параллельно.

На втором этапе происходит создание концептуальной схемы БД, а также разработка спецификации системы (определяются входные и выходные данные, а также связи между ними).

Третий этап решает следующие задачи:

* + выбор структуры комплекса технических средств;
  + распределение задач по машинам СРОД;
  + определение состава задач общесистемных пакетов.

На четвертом этапе выполняется объединение концептуальной схемы БД и спецификаций в СУБД-ориентированную среду.

Проектирование систем распределенной обработки по такой схеме зачастую приводило к тому, что после внедрения они оказывались морально устаревшими. В середина 90-х годов была разработана новая модель проектирования систем распределенной обработки данных – спиральная. Ее основной принцип – тесная связь между бизнес и IT-архитектурой. Самой распространенной ошибкой старой схемы было выявлено на этапе определения информационных потребностей пользователей. На исправление таких ошибок приходится 82% затрат от общего объема издержек по устранению ошибок проектирования.

Проектирование систем распределенной обработки данных можно разделить на этапа: концептуальное и логическое проектирования. Концептуальный проект нуждается в тщательной проработке, так как он не зависит от архитектуры системы и цена ошибки на нем несравнимо велика.

На этапе логического проектирования происходит отображение концептуального проекта с СУБД-ориентированную среду. При разработке логической схемы (ЛС) БД решаются такие задачи:

1. Уточняются характеристики таблиц: определяются типы полей, присваиваются уникальные имена, возможности хранения пустых значений.
2. Производится оптимизация ЛС БД путем объединения нескольких таблиц в одну или разъединения одной на несколько. Основным критерием таких действий является сокращение числа проверок, и соответственно автоматизация обеспечения целостности БД.
3. Таблицы ЛС БД распределяются по машинам СРОД и определяются данные, которые будут тиражироваться.
4. Выбирается способ генерации уникальных ключей, определяются атрибуты, для которых будут формироваться вторичные индексы.
5. С помощью CASE-средств генерируется программа с описанием ЛС БД.

Проектирование больших распределенных систем имеет ряд отличий от проектирования средних и небольших систем. Сложность создания большой системы заключается в необходимости применения особых приемов разработки. Разработчиками, которые создавали большие системы были сформулированы основные принципы построения таких систем, позволяющие добиться высоких результатов:

* Присутствие в проекте хорошего архитектора проекта, умеющего выделять основные задачи.
* Строгая сегментация задачи.
* Работа команды с небольшим количеством высококвалифицированных специалистов. Слишком большое количество специалистов не идет на пользу проекту.
* Использование современных инструментов по созданию приложений. Все затраты на это, как правило оказываются оправданными и быстро окупаются.
* Применение схемы циклической разработки. При таком подходе в работе находятся постановка задачи, проектирование, программирование и тестирование. Такой подход привел к быстрой оборачиваемости информации в проекте, что позволило существенно уменьшить общее время на разработку.

Для построения систем распределенной обработки данных в настоящее время используют модели сервера приложений и сервера БД, поддерживающие архитектура клиент/сервер. Связано это с высокой скоростью работы систем при этих технологиях. Преимущества архитектуры клиент/сервер наиболее наглядны и осязаемы, если ядро СУБД работает на мощном сервере.

Проблема высокой производительности в системе распределенной обработки данных всегда актуальна. До определенного времени система справляется с нагрузками, а когда нагрузки превышают технические возможности, система начинает «зависать». Рекомендации по оптимизации производительности системы делятся на две условные части:

* Рекомендации по снижению величины потока заявок к ресурсу. Уменьшение нагрузки на процессор и внешнюю память сервера достигаются путем слабо связанных кластеров серверов, минимизированием количества объединенных таблиц. Для уменьшения нагрузки на внутреннюю память целесообразно распределять файлы и таблицы по разным дискам системы и использовать большой кэш-буфер. Уменьшить интенсивность обращения к шине или каналу связи можно при минимизировании копирования таблиц или их частей, путем использования виртуальных сетей на коммутаторах, а также при использовании новых маршрутизаторов (роутеров).
* Рекомендации по уменьшению времени обработки заявки ресурсом. Достичь этого можно, если использовать более совершенные и современные технические средства (например, высокоскоростные сети и сервера).

Перспективы развития систем распределенной обработки данных:

1. Разработка баз данных на основе многоуровневых систем (составных БД), состоящих их компонентов, взаимодействующих между собой.
2. Создание интероперабельных баз данных (составных БД с открытым интерфейсом), в которых процессор запросов может извлекать данные от поставщиков записей любого типа и соответственно может быть разработано множество типов поставщиков запросов.
3. Создание множества распределенных баз данных. Количество БД с каждым днем растет и используются они в различных областях, а это значит, что все БД должны быть распределенными, и такая распределенная инфраструктура нуждается в хорошей сбалансированности.
4. Базы данных должны быть ориентированы на реалии жизни, которая состоит из последовательных задач, выполнение которых может происходить с длительными интервалами. Соответственно базы данных должны разрабатываться с учетом возможности обработки длительных транзакций и возможности отката на несколько операций назад.
5. Использование более сложных моделей данных, которые зачастую являются более эффективными.
6. Использование языков более высокого уровня позволяет увеличить возможности системы, но должны существовать возможности не прибегать к определенным языкам, а использовать только объектную модель.
7. Оптимизация работы системы с помощью поэлементного навигационного стиля и вычислений, основанных на запросах. Ранее объектно-ориентированные БД поддерживали один стиль навигации, сетевые СУБД – другой. Реляционные БД позволяли обеспечить запросы и операции на множествах. Будущее за возможностью объединить подходы или увеличить возможности каждой из существующих БД.
8. Перемещение приложений и структур под управление менеджеров БД. Совокупная память нескольких компьютеров может быть существенно выше памяти одного сервера.
9. **Банки знаний.**

Потребность интеграции данных привела к созданию банков данных. Дальнейшее развитие этого направления потребовало решения новых задач интеграции и процессов обработки. Это привело к созданию банков знаний (БнЗ).

Задачи, решаемые в банках знаний, распределяются по данным и по их обработке. В процессе этого возникает интеллектуализация систем, которая позволяет максимально удовлетворить пользователей.

Отличительной особенностью БнЗ является наличие интеллектуального интерфейса, в состав которого входят база знаний (БЗ), программа-планировщик и процессор общения. Пользователь вводит в систему корректную задачу, исходя из возможностей своей ПО, а интеллектуальный интерфейс должен выполнить работу, ранее выполняемую программистом.

Существует несколько способов представления знаний:

* логическая модель;
* фреймовые и продукционные системы;
* семантические сети.

Семантические сети позволяют производить действия как с данными, так и со знаниями о них. Семантическая сеть не делает различий между операциями над схемой и операциями над данными. Особенностью семантической сети является целостность системы, то есть нельзя разделить БЗ и механизм вывода. Основные концепции, реализованные с помощью семантических сетей:

1. Концепция одновременного рассмотрения в модели как знака, так и типа. Знак – это конкретный экземпляр, рассматриваемого объекта. Тип – класс подобных знаков. Обобщение знаков в типы - это форма абстрагирования, для лучшего понимания сложных объектов.
2. Концепция иерархии типов. Обобщение осуществляет соотношение множества типов или знаков с одним общим типом. Существуют обобщения: «тип-тип» (обобщение); «знак-тип» (классификация); экземпляризация (процесс, обратный классификации) и специализация (процесс, обратный обобщению).
3. Концепция роли. Позволяет при помощи обобщений построить сложную систему категорий.
4. Концепция семантического расстояния. При использовании данной концепции используется мера семантической близости взаимосвязанных понятий, так как эта концепция применяется в словарных системах, где слово истолковывается другим, а то свою очередь третьим и т.д.
5. Концепция разбиения. Такая концепция осуществляет разработку механизмов ограничения доступа в сети.

Основными операциями над классами и их экземплярами в семантических сетях являются следующие:

* создание экземпляра некоторого класса;
* устранение принадлежности экземпляра к некоторому классу или полное исключение его из сети;
* установление принадлежности существующего экземпляра некоторого класса еще к одному классу;
* определение принадлежности экземпляра указанному классу;
* выборка экземпляров, принадлежащих одному классу.

Все эти операции ограничены по своим возможностям. Поэтому требуется ввод в семантическую сеть программ, которые позволяют выполнять подобные обработки. Они вводятся в сеть как классы.

Фреймы используются в семантических сетях тогда, когда возникает необходимость произвести структуризацию для упрощения процесса. Суть фрейма состоит в том, что элементы, служащие описанием объекта группируются в отдельную структурную единицу и могут быть обработаны или извлечены как единое целое. Из отдельных фреймов можно создать сложные сетевые взаимосвязанные структуры.

Фрейм – это поименованная структура, составленная из описаний – слотов. С помощью слотов можно определить основные элементы фрейма. Слот состоит из имени слота и назначения слота. Имя фрейма необходимо для определения описываемого им понятия, а имя слота – для определения понятий, которые в свою очередь описывают основное понятие.

Подход представления знаний помощью логики предикатов базируется на использовании понятий исчисления предикатов при представлении классов объектов и отношений между ними. Суть данного подхода заключается в том, что взаимосвязи между различными объектами описываются с помощью предикатов.

Системы продукций – это класс моделей, который при представлении в системе знаний основывается на использовании продукций. Продукция – это правило вида: ЕСЛИ левая – часть – правила, ТО правая – часть – правила. Подобного рода продукции находят применение в различных областях. Например, в медицине это может выглядеть так: левая часть продукции – симптом заболевания, правая – диагноз.

Особенностью данного класса моделей является то, что значения отдельных продукций могут входить в условия других. Такой подход позволяет образовывать сложные цепочки, используемые для логического вывода.

Системы продукций состоят из трех основных компонентов:

1. база данных, в которой содержится множество фактов, описывающих предметную область;
2. база правил, состоящая из правил вывода, задействованных в предметной области;
3. управляющая структура (УС), реализующая механизм логического вывода.

Управляющая структура или программа-планировщик осуществляет преобразование правил, производит соединение этих правил в цепочки для получения логического вывода. База правил и база данных образую базу знаний.

Экспертная система (ЭС) представляет из себя человеко-машинную систему. При разработке такой системы необходимо участие экспертов, конструкторов и инженеров по знаниям. Выделят основные этапы разработки ЭС:

1. Этап взаимодействия инженеров по знаниям и экспертов. На первом этапе формируется модель ПО, которая должна включать обязательные компоненты: основные понятия ПО; атрибуты их описывающие; отношения, существующие между ними, множество правил, описывающих знания данной ПО. В результате работы эксперта и инженера по знаниям должны быть выявлены те неформальные правила, которыми эксперт пользуется при работе.
2. Этап взаимодействия инженера по знаниям и конструктора. На данном этапе происходит настройка оболочки ЭС на задачи ПО, загрузка всех знаний в систему, создается макет системы и производится его тестирование.
3. Этап взаимодействия инженера по знаниям, эксперта и конструктора. На этом этапе производится промышленный образец системы.
4. Этап сопровождения и модификации ЭС.

От стадии создания до стадии изготовления промышленного образца система проходит несколько стадий:

1. Демонстрационный прототип. Система способно решить часть задач и продемонстрировать свою жизнеспособность. В базе знаний на данной стадии находится 50 – 100 правил.
2. Исследовательский прототип. Система способно решить все задачи, но пока неустойчива. В базе знаний 200 – 500 правил.
3. Действующий прототип. Система решает все конкретные задачи, но при выполнении сложных задействуется много времени и памяти. В базе знаний 500 – 1000 правил.
4. Промышленный образец. Система демонстрирует решение всех поставленных задач при минимальном времени. В базе знаний 500 – 1500 и даже до 3000 правил.

Самым сложным и проблемным этапом при проектировании экспертных систем является получение необходимых знаний для ЭС. Работа по сбору такой информации производится инженером по знаниям. Основным источником информации является эксперт. Сейчас существует ряд методов проектирования баз знаний, которые основаны на получении информации от экспертов. Основные из них приведены ниже:

* Беседа с экспертом. Здесь существуют наблюдательный и интуитивный подходы. Наблюдательный подход представляет из себя изначальное наблюдение за работой эксперта, а затем совместно вносят уточнения.

У этого подхода существует иное название – анализ протоколов. При интуитивном подходе эксперт разрабатывает модели поведения, а инженер по знаниям изучает информацию о ПО из различных источников, затем проверяет их достоверность с экспертом.

* Обсуждение задач. Инженер по знания готовит пакет задач и вопросов и затем обсуждает их с экспертом.
* Описание задачи. Эксперт сам готовит описание типичных задач ПО.
* Анализ задач. Инженер по знания готовит задачи и просит эксперта их решить, при этом выяснят какой стратегией пользуется эксперт. При обсуждении выявляются проблемные моменты.
* Оценивание системы. Эксперт осуществляет анализ всех правил, которые вводятся в систему, оценивает их, в соответствии со своими знаниями и методами решения задач.
* Проверка системы. Инженер по знаниям представляет задачи и пути их решения, выполненные самой системой и экспертом, другим экспертам для того, чтобы выявить все несоответствия и разногласия.

Все полученные знания как от эксперта, так и из различных источников требуется представить в виде концептуальной базы. Для этого полученные знания представляют в виде спецификаций:

1. Цели и задачи ЭС.
2. Спецификация по каждой задаче входных данных и результата решения, а также стратегий и гипотез решения.
3. Спецификация объектов ПО, отношений и атрибутов.
4. Спецификация причинно-следственных и родовидовых отношений, а также отношений типа «часть-целое».
5. Классифицировать по каждой задаче знания, необходимые получения и решения и обоснования такого решения.

Когда все концептуальные знания получены, производят их формализацию, в результате чего получают формальное описание БЗ, то есть на описание производится на входном языке системы. На первом этапе, который является макетом будущей системы, БЗ часто поэтому выявляются и устраняются все несоответствия.

На стадии тестирования и отладки система проверяется на работоспособность на конкретных задачах. С ростом объема базы знаний проявляются новые ошибки и несоответствия. После каждой произведенной корректировки производится новое тестирование и отладка. Оценивать работу ЭС непросто. Пользователям иногда требуются не точные, а скорее наиболее полезные решения. Поэтому целесообразно привлекать пользователей в качестве экспертов в процесс проектирования ЭС.

В процессе проектирования и создания ЭС постоянно вносятся изменения и усовершенствования. Если по какой-либо причине оказывается, что многократные изменения и различные отладки так и не привели к улучшению работы системы, то придется производить пересмотр всей структуры и состава знаний в базе, а также стратегий управления процессом решения задач.

При проектировании баз знаний, как больших программных продуктов, целесообразно использовать прием при котором происходит уменьшение сложности за счет разбиения общего цикла на несколько этапов: анализ, спецификация, макетирование, сопровождение. Чтобы сократить время создания ЭС используют инструментальные средства, позволяющие автоматизировать процесс проектирования.

Технология создания БЗ ориентирована на экспертов. Причем процесс получения знаний связан с рядом трудностей. Первая трудность определена как проблема «когнитивной защиты». Это понятие основано на теории личностных конструкторов. Чем человек более разносторонен и разнообразен (то есть выше его набор личностных конструкторов), тем выше его когнитивная сложность. Преодоление когнитивной защиты связано с выявлением личных конструкторов эксперта. Еще одна трудность, состоит в том, что многим экспертам сложно описывать свои знания. Это называется проблемой вербализации знаний.

1. **Заключение.**

Эффективность работы людей в различным областях деятельности зависит от того, насколько хорошо организован обмен информацией. Информацию и данные все чаще рассматривают, как жизненно важные ресурсы, которые должны быть организованы так, чтобы ими можно было легко пользоваться. Поэтому банки данных и знаний стали одной из основных составляющих автоматизированных систем обработки информации. Знание методов организации процессов обработки информации, реализуемых в концепции банков данных и знаний, необходимость, которая позволяет принципиально по-новому  
подойти к их реализации в автоматизированных системах.

Лекционный материал позволил изучить общую характеристику банков данных и знаний, основные понятия по этой теме. Информация, изложенная в курсовой работе, более детально описывает концепции и методы построения банков данных и знаний, возможности функционирования, особенности основных моделей данных: сетевой, иерархической и датологической, которые в лекционном материале упоминались, без детального изучения.

1. **Литература.**
2. Шук В.П. Методические указания для разработки реферата на тему курсовой работы по дисциплине “Архитектура АСОИУ”. М.: МГТУ им. Н.Э.Баумана. 2016. (электронный ресурс)
3. Шук В.П. Методическое пособие по выполнению курсовой работы по дисциплине “Архитектура АСОИУ”. М.: МГТУ им. Н.Э.Баумана. 2016. (электронный ресурс)
4. Конспект лекций по дисциплине “Архитектура АСОИУ”. М.: МГТУ им..Н.Э.Баумана. 2016. (рукопись)
5. Григорьев Ю.А. и др. Банки данных. М.: Изд-во МГТУ им. Н.Э.Баумана. 2002. – 318 с.