Лекция 3

ТЕМА: СУБД.

Опр: СУБД представляет собой совокупность языковых и программных средств, предназначенных для создания, ведения и использования БД.

По характеру использования СУБД разделяют на **персональные и многопользовательские**. **Персональные** СУБД обеспечивают возможность создания локальных БД, работающих на одном компьютере. К персональным СУБД относят Paradox, dBase, FoxPro, Access и др. **Многопользовательские** СУБД позволяют создавать информационные системы, функционирующие в архитектуре «клиент-сервер». К многопользовательским СУБД относятся Oracle, Informix, SyBase, Microsoft SQL Server, InterBase и др.

3 уровня представления БД.

Взаимодействие пользователя с БД осуществляется с помощью СУБД, представляющей пользовательский интерфейс. Это взаимодействие осуществляется подобно тому, как общение с компьютером происходит с помощью ОС.

С логической точки зрения выделяют 3 разных уровня СУБД. Самый верхний - **внешний**, затем **концептуальный** и **внутренний**.

1). **Внешний уровень** имеет дело непосредственно с представлением пользователя о структуре БД, причем такое представление для разных пользователей может быть различным. В частности, пользователи, программирующие на каком-либо языке программирования, например, Delphi, имеют дело с интересующей их структурой файлов или таблиц, ничего не зная о дополнительных структурных элементах этих объектов, которые, возможно, нужны для других пользователей.

2). **Концептуальный уровень** - на нем описывается логическая структура всей БД в целом. Именно с концептуального уровня описания осуществляется проекция на внешний уровень.

3). **Внутренний уровень** — это уровень логического описания способа хранения БД на физических носителях, а также способов доступа к этой БД. Заметим, что внутренний уровень на самом деле может быть машинно-независимым, хотя и имеет дело с физическим представлением. В частности, на этом уровне периферийная память может пониматься просто как однородный массив элементов (байтов), без выделения конкретной физической структуры носителя (например, для диска - цилиндры, дорожки и т. д.). На этом уровне описываются всевозможные индексные файлы, ускоряющие доступ к элементам БД. Именно на этом уровне описывается привязка концептуального уровня к конкретным физическим носителям.

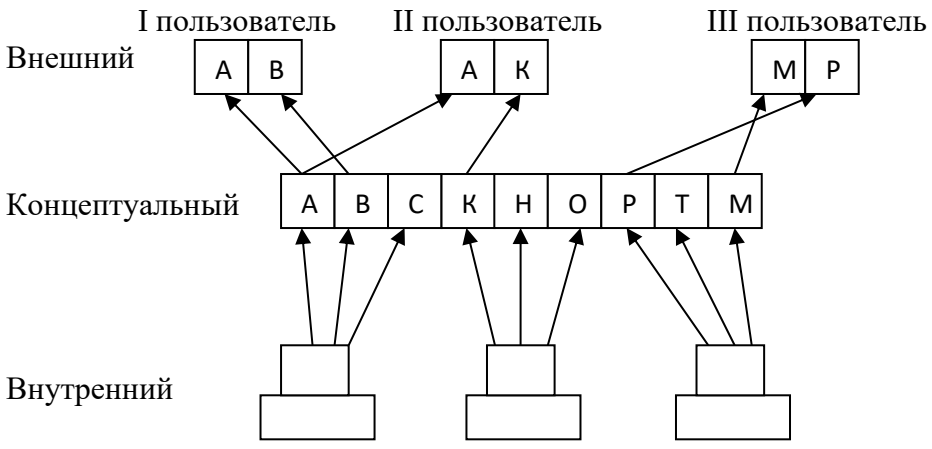

Схема представления базы данных.

Если меняется оборудование, на котором функционировала прежняя СУБД, то можно безболезненно выполнить переход на новое оборудование. Для этого понадобится переделать только внутренний уровень, т. осуществить привязку концептуального уровня к новому оборудованию. Пользователь при этом никак не пострадает, его программы не надо будет переделывать.

Концептуальный и внутренний уровни для пользователя БД не являются непосредственно доступными. Они создаются и поддерживаются специальным лицом администратором БД. Администратор БД знает досконально всю структуру БД и обеспечивает конечных пользователей той и только той информацией, которая им необходима для решения их специфических задач.

Языки СУБД.

Для определения и использования БД необходимо в распоряжении иметь некоторый язык, позволяющий, например, пользователю определить логическую структуру интересующих его записей, а также задать операции поиска, добавления, обновления и т. д. Поскольку рассмотренная модель трехуровневая, то на самом деле может существовать отдельный язык для обработки каждого уровня. Причем с внешним уровнем должны иметь дело языки, непосредственно доступные конечному пользователю. Такие пользовательские языки делятся на 2 класса:

1). Интерактивные (диалоговые) языки.

2). Языки программирования.

Интерактивный язык — это язык, позволяющий конечному пользователю непосредственно взаимодействовать с БД без запуска каких-либо программ (аналогом таких языков может быть командный язык DOS для ОС). На таком интерактивном языке пользователь может формировать свой запрос, например на поиск информации, и получать ответ непосредственно на экран дисплея.

Вторая категория языков используется прикладными программистами, а не конечными пользователями.

Существуют языки, специально созданные для этих целей (FoxPro). Однако прослеживается тенденция встраивать такие языки в существующие популярные универсальные языки программирования, расширяя последние. В этом случае расширения, позволяющие работать с БД, называются подъязыками базовых языков. В последние годы в качестве такого расширения наиболее часто используется SQL. В частности конструкции SQL доступны в таких языках как Си и Delphi.

В состав языковых средств современных СУБД входят:

1. Язык описания данных, предназначенный для описания логической структуры данных.

2. Язык манипулирования данными, обеспечивающий выполнение основных операций над данными - ввод, модификацию и выборку.

3. Язык структурированных запросов (SQL - Structured Query Language), обеспечивающий управление структурой БД и манипулирование данными, а также являющийся стандартным средством доступа к удаленным БД.

4. Язык запросов по образцу (QBE - Query By Example), обеспечивающий визуальное конструирование запросов к БД.

Архитектура «клиент-сервер»

Применительно к системам баз данных архитектура "клиент-сервер" интересна и актуальна главным образом потому, что обеспечивает простое и относительно дешевое решение проблемы коллективного доступа к базам данных в локальной сети.

Рабочая станция предназначена для непосредственной работы пользователя или категории пользователей и обладает ресурсами, соответствующими локальным потребностям данного пользователя. Специфическими особенностями рабочей станции могут быть объем оперативной памяти, наличие и объем дисковой памяти, характеристики

процессора и монитора. При необходимости можно использовать ресурсы и/или услуги, предоставляемые сервером.

Сервер локальной сети предоставляет ресурсы (услуги) рабочим станциям и/или другим серверам.

Принято называть клиентом локальной сети, компонент локальной сети запрашивающий услуги у некоторого сервера и сервером - компонент локальной сети, оказывающий услуги некоторым клиентам.

Понятно, что в общем случае, чтобы прикладная программа, выполняющаяся на рабочей станции, могла запросить услугу у некоторого сервера, как минимум требуется некоторый интерфейсный программный слой, поддерживающий такого рода взаимодействие. Из этого, собственно, и вытекают основные принципы системной архитектуры "клиент-сервер".

Система разбивается на две части, которые могут выполняться в разных узлах сети, - клиентскую и серверную части. Прикладная программа или конечный пользователь взаимодействуют с клиентской частью системы, которая в простейшем случае обеспечивает просто надсетевой интерфейс. Клиентская часть системы при потребности обращается по сети к серверной части.

Интерфейс серверной части определен и фиксирован. Поэтому возможно создание новых клиентских частей существующей системы.

Данные (файлы) удаленной базы данных находятся на удаленном компьютере.

Программа работы с удаленной базой данных состоит из двух частей: клиентской и серверной. Клиентская часть программы, работающая на компьютере пользователя, обеспечивает взаимодействие с серверной программой: посредством запросов, передаваемых на удаленный компьютер, предоставляет доступ к данным.

Сервер - это программа, реализующая функции собственно СУБД: определение данных, запись - чтение данных, поддержка схем внешнего, концептуального и внутреннего уровней, диспетчеризации и оптимизации выполнения запросов, защита данных.

Клиент - это различные программы, написанные как пользователями, так и поставщиками СУБД, внешние или «встроенные» по отношению к СУБД. Программа-клиент организована в виде приложения, работающего «поверх» СУБД, и обращающегося для выполнения операций над данными к компонентам СУБД через интерфейс внешнего уровня.

Серверная часть программы, работающая на удаленном компьютере, принимает запросы, выполняет их и пересылает данные клиентской программе. Запросы представляют собой команды, представленные на языке SQL (Structured Query Language) — языке структурированных запросов.

Программа, работающая на удаленном сервере, проектируется таким образом, чтобы обеспечить одновременный доступ к информации нескольким пользователям. При этом для обеспечения доступа к данным вместо механизма блокировки файлов используют механизм транзакций.

Базовые архитектуры распределенной обработки.
Архитектура «выделенный сервер базы данных»

В данной архитектуре средства управления базой данных и база данных размещены на машине-сервере.

Запросы клиента обрабатываются СУБД на машине-сервере. Сервер базы данных осуществляет поиск записей и анализирует их. Записи, удовлетворяющие условиям, могут накапливаться на сервере и после того, как запрос будет целиком обработан, пользователю на клиентскую машину передаются все логические записи (запрашиваемые элементы данных), удовлетворяющие поисковым условиям.

Рис. 1 «Архитектура с выделенным сервером базы данных»

Достоинства:

- Возможность обслуживания запросов нескольких клиентов;
- Снижение нагрузки на сеть и машины сервера и клиентов;
- Защита данных осуществляется средствами СУБД, что позволяет блокировать не разрешенные пользователю действия;
- Сервер реализует управление транзакциями и может блокировать попытки одновременного изменения одних и тех же записей.

Недостатки:

- Бизнес-логика функциональной обработки и представление данных могут быть одинаковыми для нескольких клиентских приложений, и это увеличит совокупные потребности в ресурсах при использовании вследствие повторения части кода программ и запросов;
- Низкий уровень управления непротиворечивостью информации, так как бизнес-правила функциональной обработки, сосредоточенные на клиентской части, могут быть противоречивыми.

Архитектура «активный сервер базы данных»

Для того чтобы устранить недостатки, свойственные архитектуре сервера базы данных, необходимо, чтобы непротиворечивость бизнес-логики и изменения базы данных контролировались на стороне сервера. Причем некоторые заранее специфицированные состояния могли бы изменять последовательность взаимодействия приложения с базой данных.

В данной архитектуре функции бизнес-логики разделяются между клиентской и серверной частями. Общие или критически значимые функции оформляются в виде хранимых процедур, включаемых в состав базы данных. Кроме этого вводится механизм отслеживания событий БД - триггеров, также включаемых в состав базы. При возникновении соответствующего события (обычно изменения данных), СУБД вызывает для выполнения хранимую процедуру, связанную с триггером, что позволяет эффективно контролировать изменение базы данных.

Хранимые процедуры и триггеры могут быть использованы любыми клиентскими приложениями, работающими с базой данных. Это снижает дублирование программных кодов и исключает необходимость компиляции каждого запроса.

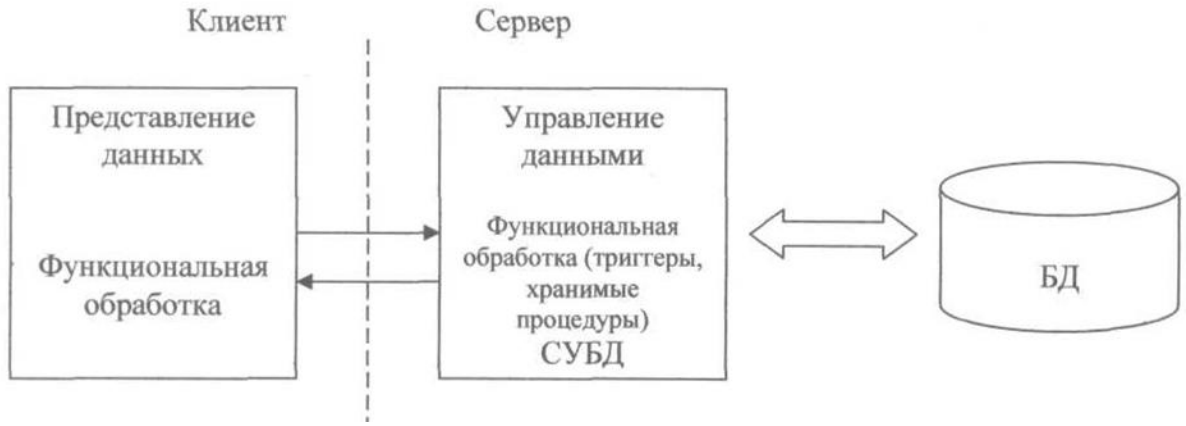

Рис. 2 Архитектура «активный сервер базы данных»

Недостатком такой архитектуры становиться существенно возрастающая загрузка сервера за счет необходимости отслеживания событий и выполнения части бизнес-правил.

Такую архитектуру организации взаимодействия (а так же рассматриваемый далее сервер приложений) иногда называют моделью с «тонким клиентом», в отличии от предыдущих архитектур, называемых моделью с «толстым клиентом», где на стороне клиента выполняется большинство функций.

Архитектура «сервер приложений»

Рассмотренные выше архитектуры являются двухзвенными: здесь все функции доступа и обработки распределены между программой клиента и сервером БД.

В данной архитектуре снижение уровня требований к ресурсам клиента достигается за счет введения промежуточного звена - сервера приложений, на который переноситься значительная часть программных компонентов управления данными и большая часть бизнес-логики. При этом серверы баз данных обеспечивают исключительно функции СУБД по ведению и обслуживанию базы данных.

Рис. 3 Архитектура сервера приложений

К другим (организационно-технологическим) достоинствам трехзвенной архитектуры можно отнести:

• Центральное ведение бизнес-логики, и в случае внесения изменения отсутствие необходимости их тиражирования в клиентских приложениях;

• Отсутствие необходимости устанавливать на клиентских машинах компоненту программного обеспечения управления доступом к данным;

• Возможность отложенного обновления БД в случае изменения данных, запрошенных с сервера, в автономном режиме. Данные будут обновлены в базе после следующего соединения клиентской программы с сервером приложений.