КОНСПЕКТ ЗАНЯТИЯ № 2

2. ВВЕДЕНИЕ В АРХИТЕКТУРУ СИСТЕМ БАЗ ДАННЫХ

Под *архитектурой системы БД* понимается совокупность ее функциональных компонентов, а также средств обеспечения их взаимодействия друг с другом, с пользователями и с системным персоналом.

Представленный далее материал обобщенно описывает архитектуру системы БД. Это вовсе не означает, что данное описание подходит для каждой системы баз данных. Например, «малые» системы, возможно, не будут поддерживать всех аспектов архитектуры.

Ниже в упрощенном виде рассматривается архитектура, предложенная подкомитетом SPARC (*англ.* Standards Planning and Requirements Committee, комитет по планированию стандартов) американского национального института стандартов ANSI, так называемая архитектура ANSI/SPARC, впервые представленная в 1975 году.

2.1. ТРЕХУРОВНЕВАЯ АРХИТЕКТУРА СИСТЕМ БАЗ ДАННЫХ ANSI/SPARC

Архитектура систем баз данных ANSI/SPARC описывает логическую организацию системы с точки зрения представления данных пользователям и включает три уровня: внутренний, концептуальный и внешний. Каждый из них состоит из одного или нескольких представлений (рис. 2.1). Уровни определяются следующим образом:

- *внутренний уровень* – уровень, наиболее близкий к физическому хранению;
- *внешний уровень* наиболее близок к пользователям, он связан со способами представления данных для отдельных пользователей;

- *концептуальный уровень* – это промежуточный уровень между двумя первыми, на котором представлены все имеющиеся данные, но в более абстрактном по сравнению с внутренним уровнем виде.

Рис. 2.1. Схематичное представление архитектуры ANSI/SPARC

Прежде чем более подробно рассматривать архитектуру БД введем несколько определений [3]. *Хранимое поле* – наименьшая единица хранимых данных. БД содержит экземпляры каждого из нескольких типов хранимых полей. *Хранимая запись* – набор связанных хранимых полей. *Хранимый файл* – набор всех экземпляров хранимых записей одного типа.

Внешний уровень

Внешний уровень – это индивидуальный уровень пользователя. Пользователи могут относиться к различным группам: прикладные программисты, конечные пользователи, администраторы (они работают также с внутренним и концептуальным уровнем). У каждого пользователя есть свой язык общения с СУБД. У конечного пользователя, это может быть язык запросов или специальный язык, основанный на формах и меню. Для прикладных программистов им

может быть один из языков высокого уровня. Все эти языки включают подъязык данных, то есть подмножество операторов базового языка, связанное только с объектами и операциями БД. Наиболее распространенный подобный язык – SQL, он поддерживается большинством систем реляционного типа.

Любой язык данных является комбинацией, по крайней мере, двух подчиненных языков – *языка определения данных* (*англ.* data definition language, DDL), который поддерживает определение и объявление объектов базы данных, и *языка обработки данных* (*англ.* data manipulation language, DML), который поддерживает операции с объектами БД, их обработку.

Вернемся к рассмотрению трехуровневой архитектуры систем БД. Отдельного пользователя интересует только некоторая часть всей БД. Кроме того, пользовательское представление этих данных может существенно отличаться от того, как они хранятся. В соответствии с терминологией ANSI/SPARC представление отдельного пользователя называется внешним представлением. *Внешнее представление* – это содержимое базы данных, каким его видит определенный конечный пользователь или группа пользователей. Можно сказать, что для пользователя его внешнее представление и есть база данных.

Внешних представлений обычно бывает несколько. Например, пользователь из отдела кадров может рассматривать базу данных как набор записей об отделах и служащих. Он может ничего не знать про записи о деталях и поставщиках, с которыми работают пользователи в отделе обеспечения.

В общем случае, внешнее представление состоит из множества экземпляров *внешних записей*, которые могут не совпадать с хранимыми записями. В системах отличных от БД, логическая (внешняя) запись обычно совпадает с хранимой.

Концептуальный уровень

Концептуальный уровень состоит из одного представления. *Концептуальное представление* – это представление всей информации базы данных в несколько более абстрактной форме (как и в случае

внешнего представления) по сравнению с физическим способом хранения данных.

Концептуальное представление состоит из множества экземпляров каждого типа концептуальной записи. Концептуальная запись не обязательно должна совпадать с внешней записью, с одной стороны, и с хранимой записью – с другой.

Концептуальное представление – это представление всего содержимого базы данных, а концептуальная схема – это определение такого представления. Определения в концептуальной схеме могут включать определения многих дополнительных средств, таких как средства безопасности или правила для обеспечения целостности. Администратор данных, определяя какие характеристики предметной области требуется сохранять в системе, фактически определяет основу концептуальной схемы.

Внутренний уровень

Также как и концептуальный, внутренний уровень состоит только из одного представления. *Внутреннее представление* описывает все подробности, связанные с хранением данных в базе. Оно состоит из экземпляров каждого типа внутренней записи. Термин «внутренняя запись» принадлежит терминологии ANSI/SPARC и фактически соответствует хранимой записи. Внутреннее представление так же, как внешнее и концептуальное, не связано с аппаратным уровнем, и не включает подробностей, связанных с размещением данных на дисках, таких как номера секторов и т. п.

Внутреннее представление описывается с помощью внутренней схемы, которая определяет типы хранимых записей, индексы (служебные структуры, упрощающие поиск данных), способы представления хранимых полей, физическую последовательность хранимых записей и т. д.

Отображения

В представленной архитектуре присутствует два уровня отображения. Отображение концептуального уровня на внутренний определяет соответствие между концептуальным представлением и хранимой базой данных. При изменении структур хранения, изменяется

и отображение «концептуальный – внутренний» таким образом, чтобы концептуальная схема осталась неизменной. Это обеспечивает так называемую *физическую независимость данных*.

Отображение внешнего уровня на концептуальный определяет соответствие между внешними представлениями и концептуальным. Например, несколько концептуальных полей «индекс», «город», «улица», «дом» для пользователя могут быть объединены в одно внешнее поле «адрес». Появляется возможность менять отдельные внешние представления, дополнительно изменяя только отображения, и не затрагивая остальные уровни системы. Отделение внешнего уровня от концептуального обеспечивает *логическую независимость данных* [5].

Определения представлений каждого из уровней и отображений, СУБД должна хранить вместе с прочей метаинформацией и использовать их при обработке запросов.

Архитектура ANSI/SPARC имеет важное теоретическое значение, определяя пути обеспечения логической и физической независимости данных. Но два уровня отображения приводят к дополнительным накладным расходам при обработке запросов пользователя, поэтому разработчики СУБД, стараясь увеличить быстродействие систем, обычно отходят от строгой реализации этой архитектуры.

2.2. АРХИТЕКТУРА МНОГОПОЛЬЗОВАТЕЛЬСКИХ СИСТЕМ БАЗ ДАННЫХ

Рассмотрим аспекты архитектуры систем БД, связанные с обеспечением совместной работы пользователей. В несколько абстрактной форме можно представить многопользовательскую систему БД совокупностью двух компонент – создающего запросы клиента, и сервера, который выполняет приходящие запросы. В частном случае, и клиентское, и серверное приложения могут выполняться на одном и том же компьютере. Но они могут находиться и на разных компьютерах, связанных телекоммуникационной сетью. В зависимости от разделения функций между клиентом и сервером выделяются несколько типов архитектур систем БД.

Файл-серверная архитектура

Некоторые СУБД могут работать только в режиме «файл-сервер». Это означает, что СУБД находится на клиентской рабочей станции и даже может быть скомпонована с прикладным ПО. Для осуществления совместного доступа к данным, файлы БД в этом случае размещаются на файловом сервере (рис. 2.2).

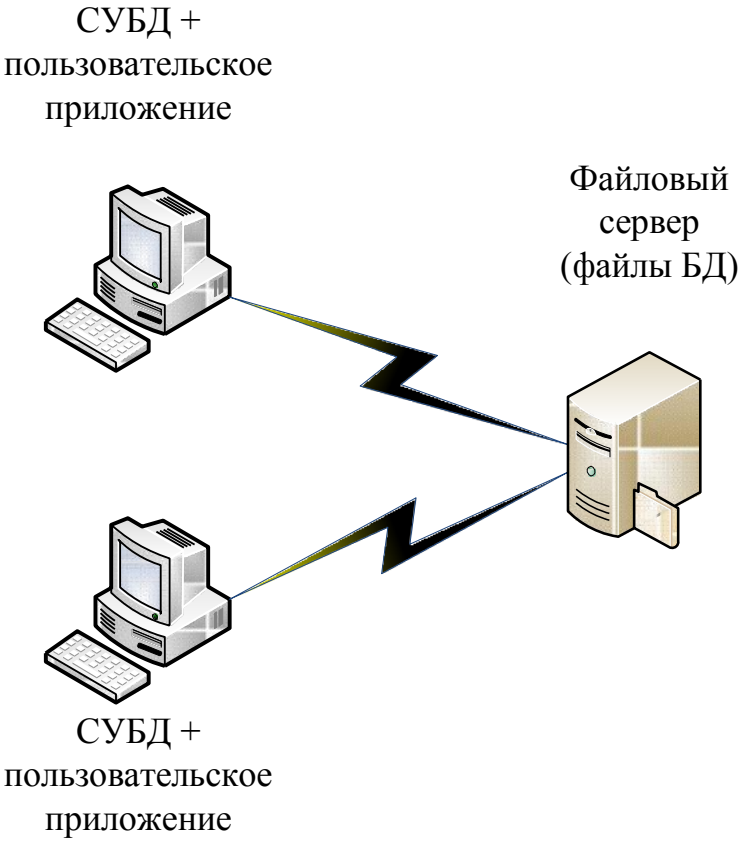

Рис. 2.2. Файл-серверная архитектура

Если работа с данными ведется на нескольких компьютерах, то на каждом из них будет запущен экземпляр СУБД. Для обработки задания пользователя, с сервера запрашиваются файлы с данными, а их обработка производится локально. В результате, в таких системах приходится передавать по сети много данных для того, что бы на стороне клиента среди них найти те, что удовлетворяют запросу.

Основным достоинством подобной архитектуры является простота: для того чтобы из локального приложения сделать многопользовательское сетевое, надо просто переместить файлы БД на файловый сервер. При этом СУБД должна обеспечить минимально

необходимый набор функциональности в области совместного использования файлов БД: блокировка изменяемых записей и т. п.

Недостатков у такой архитектуры существенно больше. Во-первых, это неэффективная загрузка сети передачи данных: по сети передается во много раз больше данных, чем это необходимо приложению. Например, если в таблице 100 000 записей, из которых нам нужно 10, в неудачном случае может потребоваться передача всей таблицы на клиентский компьютер для последующей обработки. Помимо загрузки сети, это существенно увеличит время выполнения операции, по сравнению с обработкой базы, хранящейся локально.

Во-вторых, могут возникать существенные проблемы при синхронизации совместной работы пользователей. Для поддержания информации об изменяемых в данный момент записях, файл-серверные СУБД могут создавать на сервере так называемые файлы блокировок или другие подобные структуры. Если один из клиентов заблокировал часть записей (СУБД внесла информацию в файл блокировок, что эти записи будут изменяться и их нельзя использовать другим клиентам), после чего потерял связь с сервером, экземпляры СУБД на других клиентах не смогут использовать соответствующие фрагменты БД, пока эту проблему не решит администратор, откорректировав файл блокировок.

В-третьих, файл-серверная архитектура предъявляет дополнительные требования к производительности клиентских рабочих мест: вся обработка данных производится на клиенте.

Примером многопользовательской системы с файл-серверной архитектурой, является совместное использование базы данных Microsoft Access в локальной сети. В таком режиме с одной базой могут нормально работать несколько пользователей: в зависимости от размеров базы, характеристик сети и интенсивности работы с данными, это может быть до 5-10 одновременных сеансов.

Двухзвенная архитектура «клиент-сервер»

Данная архитектура предполагает, что СУБД находится на сервере и только она имеет доступ к файлам баз данных (рис. 2.3).

На клиентских компьютерах работают пользовательские приложения и клиентские компоненты СУБД, осуществляющие взаимодействие с сервером. От клиента на сервер приходят запросы, которые обрабатываются СУБД и результат отправляется на клиентский компьютер. По сравнению с файл-серверной архитектурой, в этом случае минимизируется сетевой трафик: по сети передаются только затребованные данные.

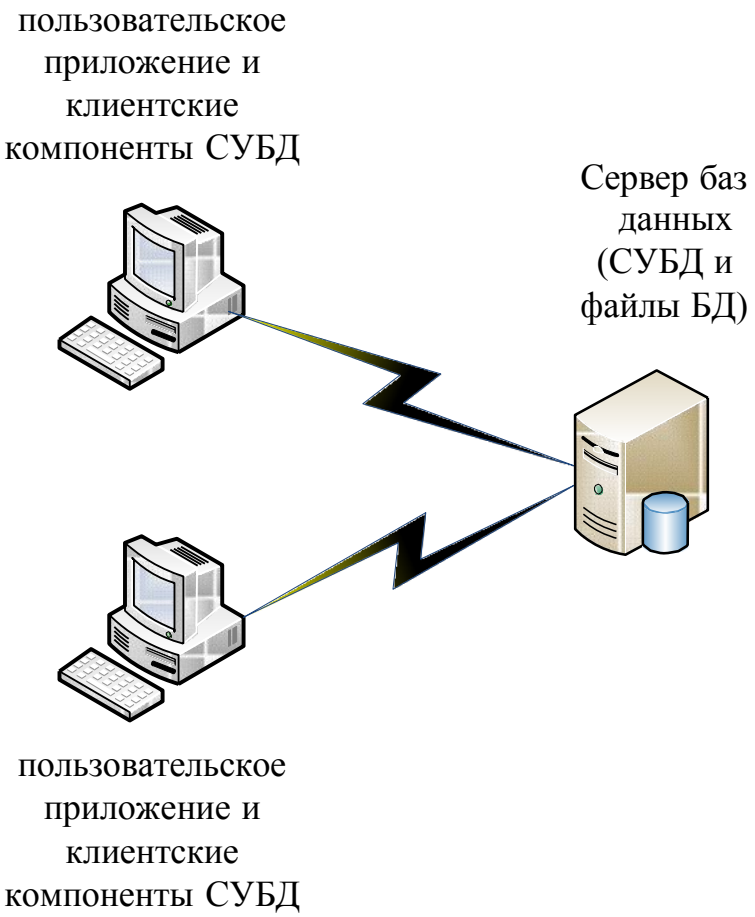

Рис. 2.3. Двухзвенная архитектура «клиент-сервер»

Централизованная работа с файлами баз данных позволяет более эффективно решать вопросы, связанные с совместным доступом к данным и с обеспечением безопасности. В связи с тем, что СУБД работает на сервере, снижаются и требования к оборудованию на стороне клиента. Прикладная программа может быть написана на любом языке, который поддерживает взаимодействие с используемой СУБД через имеющиеся клиентские компоненты: могут использоваться такие технологии, как ODBC, ADO или ADO.Net, JDBC и другие.

Двухзвенная архитектура широко используется для создания информационных систем, но и у нее есть ряд недостатков. Во-первых, это «логика» приложения, вынесенная на строну клиента. При большом количестве и географической удаленности клиентских рабочих мест возникают проблемы с обновлением и устранением ошибок. Вторая проблема – каждый клиент независимо от других может в произвольный момент открыть соединение с СУБД. Сервер поддерживает открытые соединения со всеми активными клиентами, даже если никакой работы нет. При большом числе клиентов это может негативно влиять на производительность сервера баз данных.

Трехзвенная архитектура

Исправить недостатки двухзвенной архитектуры позволяет трехзвенная, также называемая трехуровневой (рис. 2.4.). Данная архитектура предполагает наличие дополнительного сервера приложений, который проводит предварительную обработку запросов клиентов, формирует запросы к серверу баз данных и обрабатывает полученные результаты перед отправкой их клиенту.

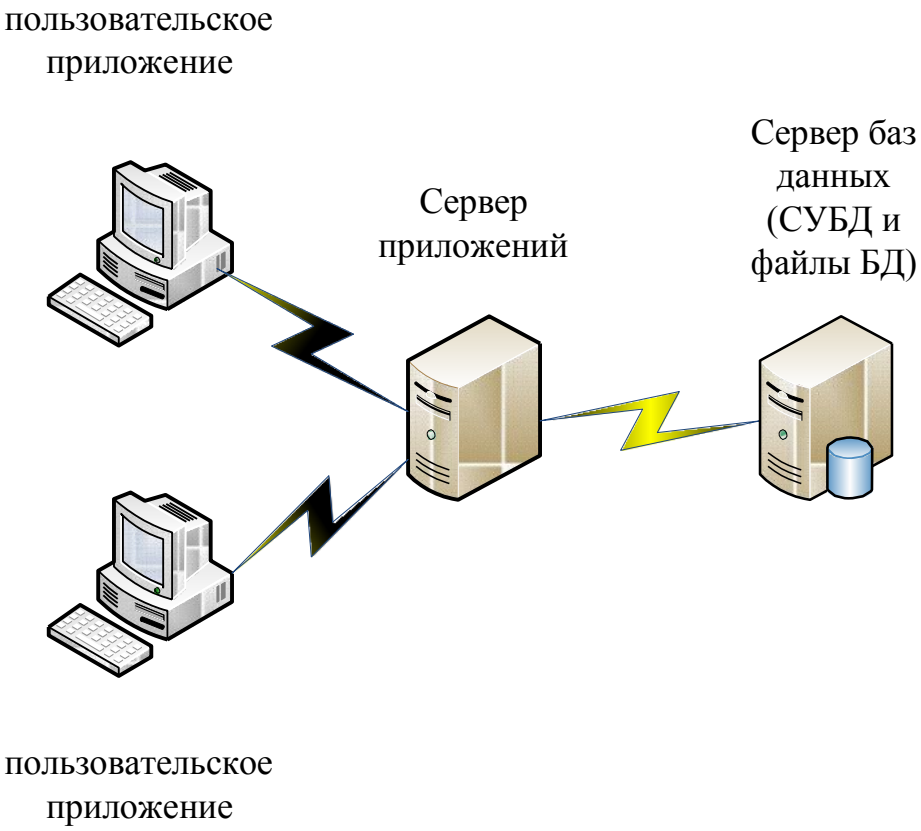

Рис. 2.4. Трехзвенная архитектура

В трехзвенной архитектуре большая часть логики приложения перенесена с клиента на сервер и задачи клиентского приложения сводятся, в основном, к реализации пользовательского интерфейса и представлению результатов. Появляется возможность централизованно вносить изменения в алгоритмы бизнес-логики, реализованные в ПО сервера приложений. Также в этой архитектуре только сервер приложений может подключаться к серверу баз данных. Это снимает проблему поддержания неиспользуемых соединений и более предпочтительно с точки зрения безопасности. Недостатком многоуровневой архитектуры клиент-сервер является сложность разработки подобных решений.

Архитектура Интернет/интранет решений

Заканчивая раздел, необходимо упомянуть об архитектуре Интернет/интранет приложений (рис.2.5).

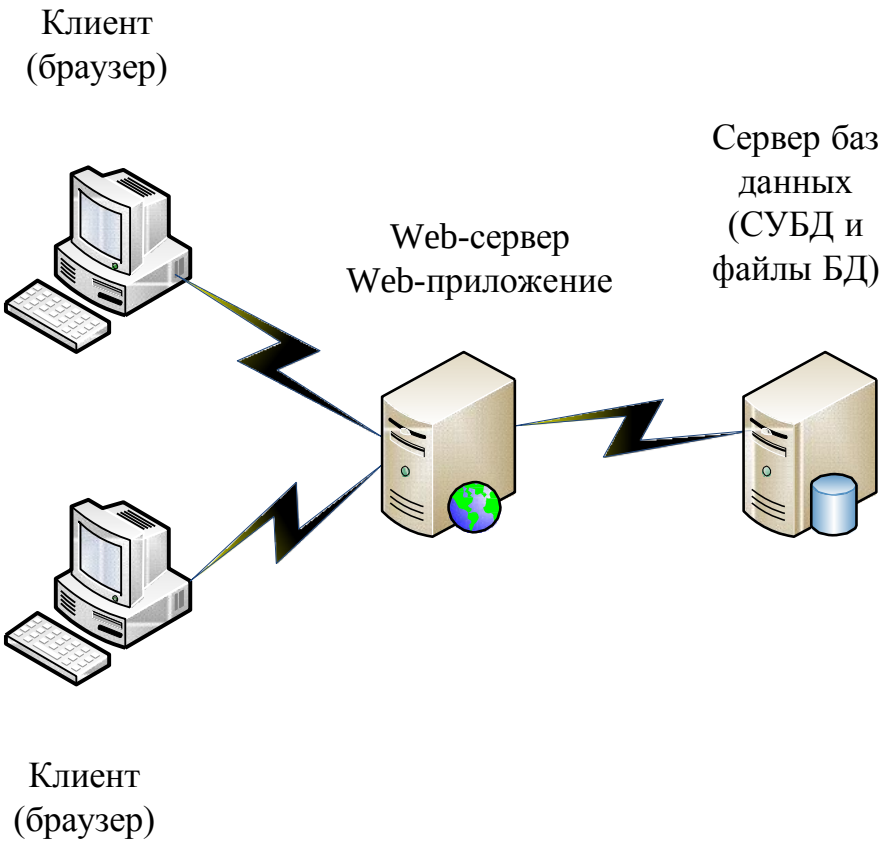

Рис. 2.5. Архитектура Интернет/интранет решений

В этом случае обязательным компонентом серверной части является web-сервер, на котором выполняется web-приложение, обращающееся к СУБД для доступа к базам данных. В роли универсального клиентского приложения выступает интернет-браузер на устройстве клиента (в данном случае, это может быть персональный компьютер, планшетный компьютер или какое-то мобильное устройство).

В небольших решениях, web-сервер может находиться на одном физическом компьютере с СУБД. В крупных системах, испытывающих большие нагрузки, может задействоваться множество web-серверов и серверов баз данных.

Достоинства и недостатки подобной архитектуры связаны с использованием браузера в качестве универсального клиента. С одной стороны, отпадает необходимость в разработке, распространении и поддержке клиентского приложения. Появляется универсальность и частичная независимость от клиентской платформы. С другой стороны, в случае ограниченного числа пользователей приложения, с помощью специально разработанного клиентского ПО можно добиться большей функциональности, по сравнению с использованием web-технологий.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

3. Дейт К. Дж. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб.: Издательский дом «Вильямс», 1999. – 848 с.

5. Кириллов В.В. Введение в реляционные базы данных. / В.В. Кириллов, Г.Ю. Громов – СПб.: БХВ-Петербург, 2009. – 464 с.