

8. Модели архитектуры «клиент-сервер»

Современная СУБД должна удовлетворять ряду требований, важнейшее из которых — высокопроизводительный интеллектуальный сервер базы данных. Процесс технического совершенствования сервера БД пока остается невидимым для большинства пользователей современных СУБД. Поэтому при выборе той или иной системы они, как правило, не учитывают ни технический уровень решений, заложенных в механизм его функционирования, ни влияние этих решений на общую производительность СУБД. Между тем ее качество определяется отнюдь не богатством интерфейсов с пользователем, не разнообразием средств поддержки разработок, а в первую очередь зависит от особенностей архитектуры сервера базы данных.

«Клиент-сервер» — это модель взаимодействия компьютеров в сети. Как правило, компьютеры не являются равноправными. Каждый из них имеет свое, отличное от других, назначение, играет свою роль. Некоторые компьютеры в сети владеют и распоряжаются информационно-вычислительными ресурсами, такими как процессоры, файловая система, почтовая служба, служба печати, база данных. Другие же компьютеры имеют возможность обращаться к этим службам, пользуясь услугами первых. Компьютер, управляющий тем или иным ресурсом, принято называть *сервером* этого ресурса, а компьютер, желающий им воспользоваться — *клиентом*. Конкретный сервер определяется видом ресурса, которым он владеет. Так, если ресурсом являются базы данных, то речь идет о *сервере баз данных*, назначение которого — обслуживать запросы клиентов, связанные с обработкой данных; если ресурс — это *файловая система*, то говорят о *файловом сервере*, или *файл-сервере*, и т. д.

В сети один и тот же компьютер может выполнять роль как клиента, так и сервера. Например, в информационной системе, включающей персональные компьютеры, большую ЭВМ и мини-компьютер под управлением UNIX, последний может выступать как в качестве сервера базы данных, обслуживая запросы от клиентов — персональных компьютеров, так и в качестве клиента, направляя запросы большой ЭВМ.

Этот же принцип распространяется и на взаимодействие программ. Если одна из них выполняет некоторые функции, предоставляя другим соответствующий набор услуг, то такая программа выступает в качестве сервера. Программы, которые пользуются этими услугами, принято называть клиентами. Так, ядро реляционной SQL-ориентированной СУБД часто называют сервером базы данных, или SQL-сервером, а программу, обращающуюся к нему за услугами по обработке данных — SQL-клиентом.

Как поддерживающая интерфейс с пользователем программа, СУБД, в общем случае, реализует следующие основные функции:

- *управление* данными, находящимися в базе;
- *обработка* информации с помощью прикладных программ;
- *представление* информации в удобном для пользователя виде.

Если система размещается на одной ЭВМ, то все функции собраны в одной программе. При размещении СУБД в сети возможны различные варианты распределения функций по узлам. В зависимости от числа узлов сети, между которыми выполняется распределение функций СУБД, можно выделить двухзвенные модели, трехзвенные модели и т.д. Место разрыва функций соединяется коммуникационными функциями (средой передачи информации в сети).

§8.1 Двухзвенные модели распределения функций

Двухзвенные модели соответствуют распределению функций СУБД между двумя узлами сети. Компьютер (узел сети), на котором обязательно присутствует функция управления данными, назовем компьютером-сервером. Компьютер, близкий к пользователю и обязательно занимающийся вопросами представления информации, назовем компьютером-клиентом.

Наиболее типичными вариантами (рисунок 1) разделения функций между компьютером-сервером и компьютером-клиентом являются следующие:

- распределенное представление;
- удаленное представление;
- удаленный доступ к данным;
- распределенная БД.

Перечисленные способы распределения функций в системах с архитектурой клиент-сервер иллюстрируют различные варианты: от мощного сервера, когда практически вся работа производится на нем, до мощного клиента, когда большая часть функций выполняется на рабочей станции, а сервер обрабатывает поступающие к нему по сети SQL-вызовы.

В моделях удаленного доступа к данным и удаленного представления производится строгое распределение функций между компьютером-клиентом и компьютером-сервером. В других моделях имеет место выполнение одной из двух функций одновременно на двух компьютерах: управления данными (модель распределенной БД) или представления информации (модель распределенного представления).

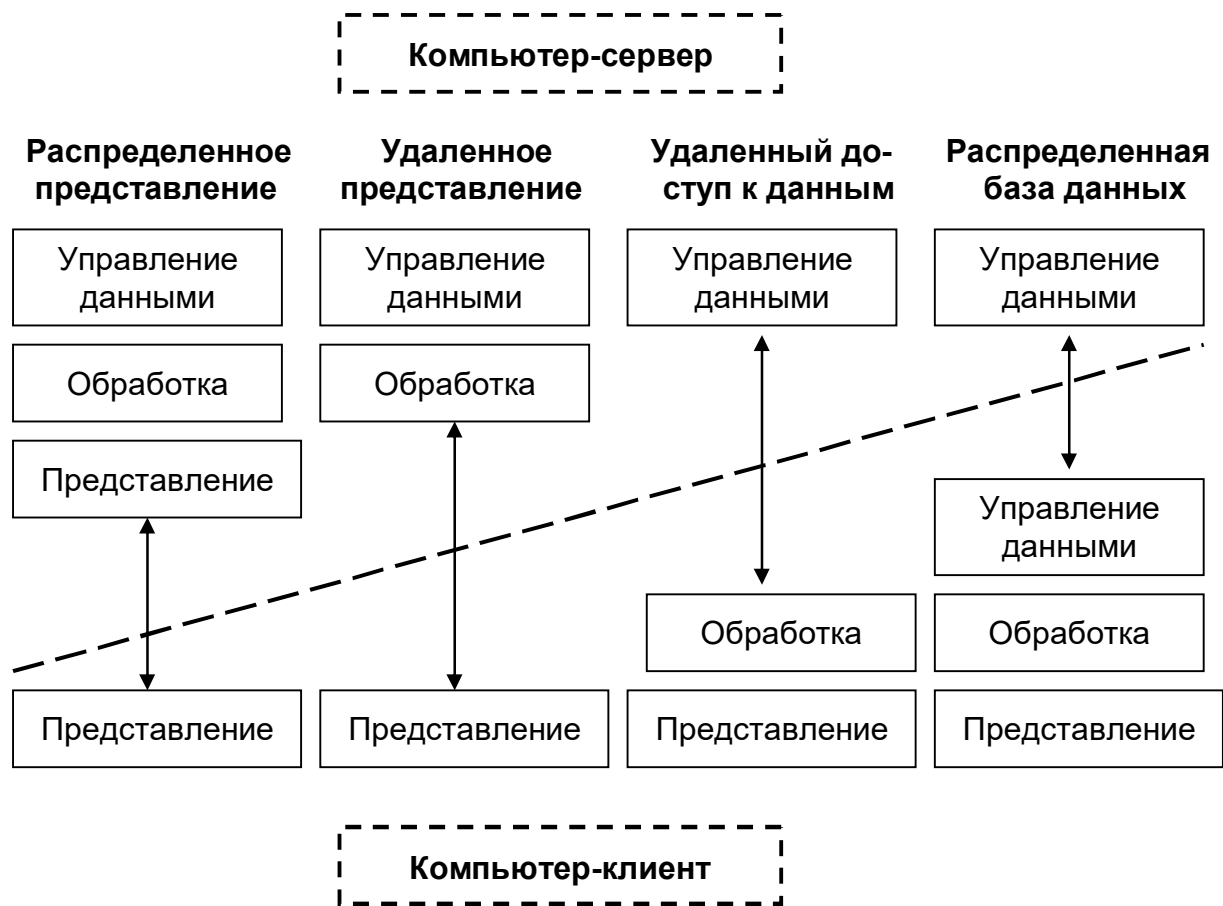


Рисунок 1 — Спектр моделей архитектуры клиент-сервер

Рассмотрим сначала модели *удаленного доступа к данным* и *удаленного представления (сервера БД)* как наиболее широко распространенные.

В модели *удаленного доступа к данным* (Remote Data Access — **RDA**) программы, реализующие функции представления информации и логику прикладной обработки, совмещены и выполняются на компьютере-клиенте. Обращение за сервисом управления данными происходит через среду передачи с помощью операторов языка SQL или вызовом функций специальной библиотеки **API** (*Application Programming Interface* — интерфейса прикладного программирования) как показано на рисунке 2.

Основное достоинство RDA-модели состоит в большом обилии готовых СУБД, имеющих SQL-интерфейсы, и существующих инструментальных средств, обеспечивающих быстрое создание программ клиентской части. Средства разработки чаще всего поддерживают графический интерфейс пользователя в MS Windows, стандарт интерфейса ODBC и средства автоматической генерации кода. Подавляющее большинство средств разработки использует языки четвертого поколения.

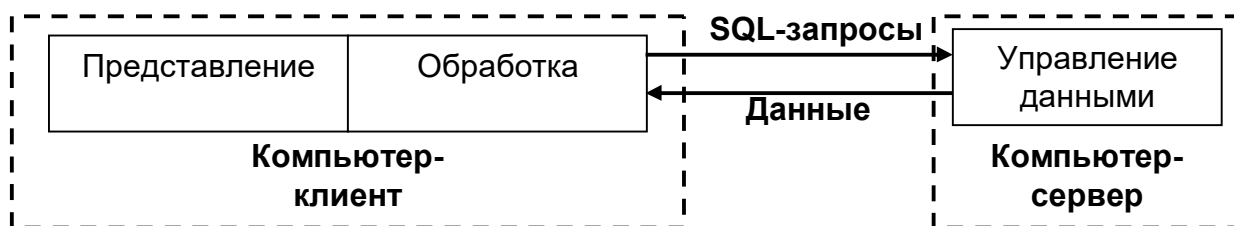


Рисунок 2 — Модель удаленного доступа к данным

Недостатками RDA-модели являются, во-первых, довольно высокая загрузка системы передачи данных вследствие того, что вся логика сосредоточена в приложении, а обрабатываемые данные расположены на удаленном узле. Как увидим далее, во время работы приложений обычно по сети передаются целые БД.

Во-вторых, системы, построенные на основе модели RDA, неудобны с точки зрения разработки, модификации и сопровождения. Основная причина состоит в том, что в получаемых приложениях прикладные функции и функции представления тесно взаимосвязаны. Поэтому даже при незначительном изменении функций системы требуется переделка всей прикладной ее части, усложняющая разработку и модификацию системы.

Модель *сервера БД* (DataBase Server — *DBS*) отличается от предыдущей модели тем, что функции компьютера-клиента ограничиваются функциями представления информации, в то время как прикладные функции обеспечиваются приложением, находящимся на компьютере-сервере (рисунок 3).

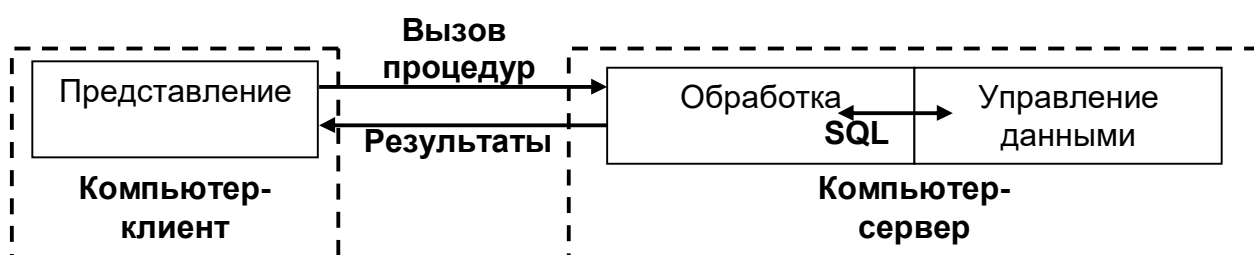


Рисунок 3 — Модель сервера БД

Перед сервером БД стоят следующие задачи:

1. Необходимо, чтобы БД в каждый момент времени отражала текущее состояние предметной области, которое определяется не только собственными данными, но и связями между объектами данных. То есть

данные, которые хранятся в БД, в каждый момент времени должны быть непротиворечивыми.

2. БД должна отражать некоторые правила предметной области, законы, по которым она функционирует. Например, завод может нормально функционировать только в том случае, если на складе имеется некоторый достаточный запас (страховой запас) деталей определенной номенклатуры.
3. Необходим постоянный контроль за состоянием БД, отслеживание всех изменений и адекватная реакция на них: например, при достижении некоторым измеряемым параметром критического значения должно произойти отключение определенной аппаратуры, при уменьшении товарного запаса ниже допустимой нормы должна быть сформирована заявка конкретному поставщику на соответствующий товар.
4. Необходимо, чтобы возникновение некоторой ситуации в БД четко и оперативно влияло на ход выполнения прикладной задачи.
5. Одна из важных проблем СУБД — контроль типов данных. В настоящий момент СУБД может проверять только стандартно-допустимые типы данных, т.е. такие, которые определены в DDL (*Data Definition Language*) — языке описания данных, являющемся частью SQL. Однако в реальных предметных областях могут потребоваться нестандартные типы данных (плоскостные и пространственные координаты, единицы различных метрик, пятидневная неделя, т.е. рабочая неделя, в которой сразу после пятницы следует понедельник, дроби, не говоря уже о графических изображениях).

Эта модель является более технологичной чем RDA-модель и применяется в таких СУБД, как Ingress, Sybase и Oracle. При этом приложения реализуются в виде *хранимых процедур*.

Клиентское приложение обращается к серверу с командой запуска хранимой процедуры, а сервер выполняет эту процедуру и регистрирует все изменения в БД, которые в ней предусмотрены. Сервер возвращает клиенту данные, отвечающие запросу, при этом трафик обмена информацией между клиентом и сервером резко уменьшается.

Централизованный контроль в модели сервера баз данных выполняется с использованием механизма *триггеров*. Триггеры являются частью БД и представляют собой некоторый тумблер, который срабатывает при возникновении определенного события в БД. СУБД проводит мониторинг всех событий, которые вызывают созданные и описанные триггеры в БД, и при возникновении соответствующего события сервер запускает соответствующий триггер. Триггеры могут вызывать хранимые процедуры.

Механизм использования триггеров предполагает, что при срабатывании одного триггера могут возникнуть события, которые вызовут срабатывание

других триггеров. Этот мощный инструмент требует тонкого и согласованного применения, чтобы не получился бесконечный цикл срабатывания триггеров.

В данной модели сервер является активным, потому что не только клиент, но и сам сервер, используя механизм триггеров, может быть инициатором обработки данных в БД.

И хранимые процедуры, и триггеры хранятся в словаре БД, они могут быть использованы несколькими клиентами, что существенно уменьшает дублирование алгоритмов обработки данных в разных клиентских приложениях.

Достоинствами модели DBS являются: возможность хорошего централизованного администрирования приложений на этапах разработки, сопровождения и модификации, а также эффективное использование вычислительных и коммуникационных ресурсов. Последнее достигается за счет того, что выполнение программ в режиме коллективного пользования требует существенно меньших затрат на пересылку данных в сети.

Один из недостатков модели DBS связан с ограничениями средств разработки хранимых процедур. Основное ограничение — сильная привязка операторов хранимых процедур к конкретной СУБД. Язык написания хранимых процедур, по сути, является процедурным расширением языка SQL, и не может соперничать по выразительным средствам и функциональным возможностям с традиционными языками третьего поколения, такими, как Си и Паскаль. Кроме того, в большинстве СУБД нет удовлетворительных средств отладки и тестирования хранимых процедур, что делает их механизм опасным инструментом — неотлаженные программы могут приводить к некоррекностям БД, зависаниям серверных и клиентских программ во время работы системы и т. п.

Другим недостатком DBS-модели является низкая эффективность использования вычислительных ресурсов ЭВМ, поскольку не удастся организовать управление входным потоком запросов к программам компьютера-сервера, а также обеспечить перемещение процедур на другие компьютеры-серверы.

В модели ***распределенного представления*** имеется мощный компьютер-сервер, а клиентская часть системы практически вырождена. Функцией клиентской части является просто отображение информации на экране монитора и связь с основным компьютером через локальную сеть.

СУБД подобного рода могут иметь место в сетях, поддерживающих работу так называемых Х-терминалов. В них основной компьютер (хост-машина) должен иметь достаточную мощность, чтобы обслуживать несколько Х-терминалов. Х-терминал тоже должен обладать достаточно быстрым процессором и иметь

достаточный объем оперативной памяти (дисковые накопители отсутствуют). Все программное обеспечение находится на хост-машине. Программное обеспечение X-терминала, выполняющее функции управления представлением и сетевые функции, загружается по сети с сервера при включении X-терминала.

Модель распределенного представления имели СУБД ранних поколений, которые работали на малых, средних и больших ЭВМ. В роли X-терминалов выступали дисплейные станции и абонентские пункты (локальные и удаленные). В этом случае основную часть функций представления информации реализовывали сами СУБД, а окончательное построение изображений на терминалах пользователя выполнялось на конечных устройствах.

По модели распределенного представления построены системы обслуживания пользователей БД в гетерогенной (неоднородной) среде. Серверная часть таких систем обычно обеспечивает некоторый унифицированный интерфейс, а клиентские части реализуют функции учета специфики конечного оборудования или преобразования одного формата представления информации в другой.

Модель распределенного представления реализует централизованную схему управления вычислительными ресурсами. Отсюда следуют ее основные достоинства — простота обслуживания и управления доступом к системе и относительная дешевизна (ввиду невысокой стоимости конечных терминалов). Недостатками модели являются уязвимость системы при невысокой надежности центрального узла, а также высокие требования к серверу по производительности при большом числе клиентов.

Модель *распределенной БД* предполагает использование мощного компьютера-клиента, причем данные хранятся на компьютере-клиенте и на компьютере-сервере. Взаимосвязь обеих баз данных может быть двух разновидностей: а) в локальной и удаленной базах хранятся отдельные части единой БД; б) локальная и удаленная БД являются синхронизируемыми друг с другом копиями.

Достоинством модели распределенной БД является гибкость создаваемых на ее основе ИС, позволяющих компьютеру-клиенту обрабатывать локальные и удаленные БД. При наличии механизмов координации соответствия копий система в целом, кроме того, обладает высокой живучестью, так как разрыв соединения клиента и сервера не приводит к краху системы, а ее работа может быть восстановлена с возобновлением соединения. К недостатку модели можно отнести высокие затраты при выполнении большого числа одинаковых приложений на компьютерах-клиентах.

§8.2 Трехзвенная модель распределения функций

Трехзвенная модель распределения функций представляет собой типовой вариант, при котором каждая из трех функций приложения реализуется на

отдельном компьютере. Варианты распределения функций приложения на большее число компьютеров могут также иметь место.

Рассматриваемая нами модель имеет название *модель сервера приложений*, или *AS-модель* (Application Server) и показана на рисунке 4.

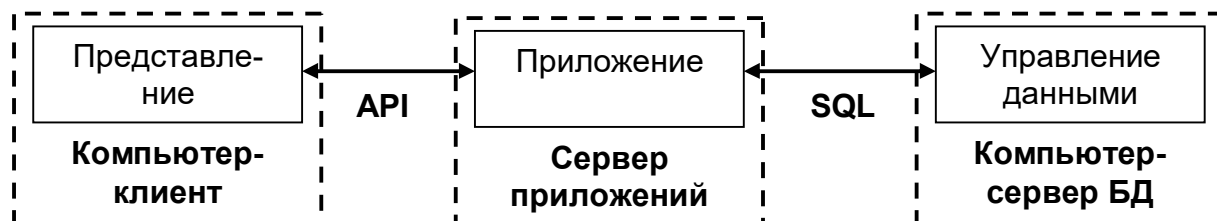


Рисунок 4 — Трехзвенная модель сервера приложений

Согласно трехзвенной AS-модели, отвечающий за организацию диалога с конечным пользователем процесс, как обычно, реализует функции представления информации и взаимодействует с компонентом приложения так же, как в модели DBS. Компонент приложения, располагаясь на отдельном компьютере, в свою очередь, связан с компонентом управления данными подобно модели RDA.

Центральным звеном AS-модели является сервер приложений. На сервере приложений реализуется несколько прикладных функций, каждая из которых оформлена как служба предоставления услуг всем требующим этого программам. Серверов приложений может быть несколько, причем каждый из них предоставляет свой вид сервиса. Любая программа, запрашивающая услугу у сервера приложений, является для него клиентом. Поступающие от клиентов к серверам запросы помещаются в очередь, из которой выбираются в соответствии с некоторой дисциплиной, например, по приоритетам.

Компонент, реализующий функции представления и являющийся клиентом для сервера приложений, в этой модели трактуется более широко, чем обычно. Он может служить для организации интерфейса с конечным пользователем, обеспечивать прием данных от устройств, например, датчиков, или быть произвольной программой.

Достоинством AS-модели является гибкость и универсальность вследствие разделения функций приложения на три независимые составляющие. Во многих случаях эта модель оказывается более эффективной по сравнению с двухзвенными. Основным недостаток модели — более высокие затраты ресурсов компьютеров на обмен информацией между компонентами приложения по сравнению с двухзвенными моделями.

§4.3 Стандарт ODBC

При построении информационных систем типа клиент-сервер возникает проблема доступа со стороны СУБД или приложений, разработанных в одной

среде, к данным, порожденным другой СУБД. В среде Windows эта проблема решается с помощью стандартного интерфейса **ODBC** (*Open Database Connectivity* — совместимость открытых баз данных) фирмы Microsoft. Основное его назначение заключается в обеспечении унифицированного доступа к локальным и удаленным БД различных производителей.

Интерфейс ODBC обеспечивает взаимную совместимость серверных и клиентских компонентов доступа к данным. Для реализации унифицированного доступа к различным СУБД, было введено понятие **драйвера** ODBC (представляющего собой динамически загружаемую библиотеку).

ODBC-архитектура содержит четыре компонента:

- приложение;
- менеджер драйверов;
- драйверы;
- источники данных.

Роли среди них распределены следующим образом. *Приложение* вызывает функции ODBC для выполнения SQL-инструкций, получает и интерпретирует результаты; *менеджер драйверов* загружает ODBC-драйверы, когда этого требует приложение; *ODBC-драйверы* обрабатывают вызовы функций ODBC, передают операторы SQL СУБД и возвращают результат в приложение; *источник данных (data source)* — объект, скрывающий СУБД, детали сетевого интерфейса, расположение и полное имя базы данных и т.д.

Действия, выполняемые приложением, использующем интерфейс ODBC, сводятся к следующему: для начала сеанса работы с базой данных приложение должно подключиться к источнику данных, ее скрывающему; затем приложение обращается к базе данных, посылая SQL-инструкции, запрашивает результаты, отслеживает и реагирует на ошибки и т.д., то есть имеет место стандартная схема взаимодействия приложения и сервера БД, характерная для RDA-модели. Важно, что стандарт ODBC включает функции управления транзакциями (начало, фиксация, откат транзакции). Завершив сеанс работы, приложение должно отключиться от источника данных.