PROАГРО ЛЕКТОРИЙ

Обеспечение целостности баз данных

Быстренина Ирина Евгеньевна,
кандидат педагогических наук, доцент
кафедры прикладной информатики
РГАУ-МСХА имени К. А. Тимирязева

Архитектуры баз данных

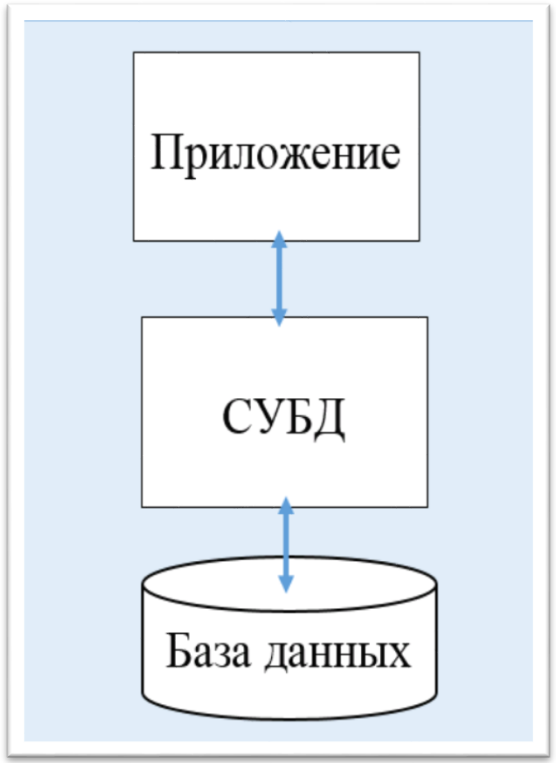
СУБД и прикладная программа (приложение) располагаются на одном компьютере. Для такого способа организации не требуется поддержки сети и все сводится к автономной работе.

Работа построена следующим образом:

- База данных в виде набора файлов находится на жестком диске компьютера;
- На том же компьютере установлены СУБД и приложение для работы с БД;
- Пользователь запускает приложение и через пользовательский интерфейс приложения инициирует обращение к БД на выборку/обновление информации;
- Все обращения к БД идут через СУБД, которая аккумулирует в себе все сведения о физической структуре БД;
- СУБД инициирует обращения к данным, обеспечивая выполнение запросов пользователя;
- Результат СУБД возвращает в приложение;
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

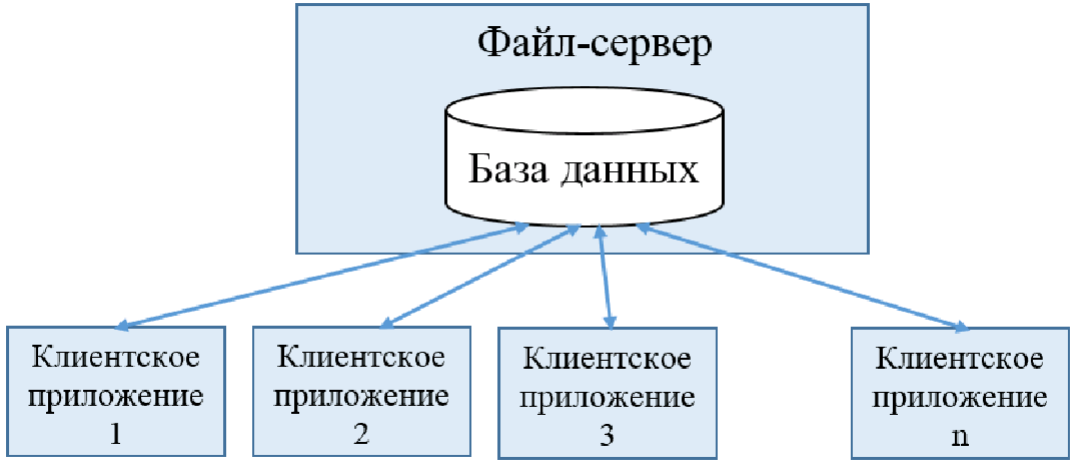
Таким образом, в этой модели реализуется однопользовательский режим работы.

Автономная архитектура



Архитектура «файл-сервер»

В архитектуре «файл-сервер» базы данных хранятся на сервере, клиент обращается к серверу с файловыми командами, а механизм управления всеми информационными ресурсами находится на компьютере клиента.
Файл-серверные базы данных могут быть доступны многим клиентам через сеть. Сама база данных хранится на сетевом «файл-сервере» в единственном экземпляре. Для каждого клиента во время работы создается локальная копия данных, с которой он манипулирует.



Недостатки архитектуры «файл-сервер»

- Непроизводительная загрузка сети (при каждом запросе клиента данные в его локальной копии полностью обновляются из базы данных на сервере, даже если запрос относится всего к одной записи, обновляются все записи базы данных);
- Забота о целостности данных при такой организации работы возлагается на клиентские приложения, что приводит к их усложнению и, если они недостаточно тщательно продуманы, в базу данных легко занести ошибки, которые могут отразиться на всех пользователях;
- Изменения, сделанные в базе данных одним пользователем, не видны другим пользователям, пока один пользователь редактирует какую-либо запись, она заблокирована для других клиентов;
- Управление базой данных осуществляется с разных компьютеров, поэтому в значительной степени затруднена организация контроля доступа, соблюдения конфиденциальности, что также усложняет поддержку целостности базы данных.

Архитектура «файл-сервер»

Несмотря на перечисленные недостатки файл-серверная архитектура проста и доступна.

Часто файл-серверные СУБД называют настольными. Настольные СУБД с успехом применяются для сравнительно небольших задач:

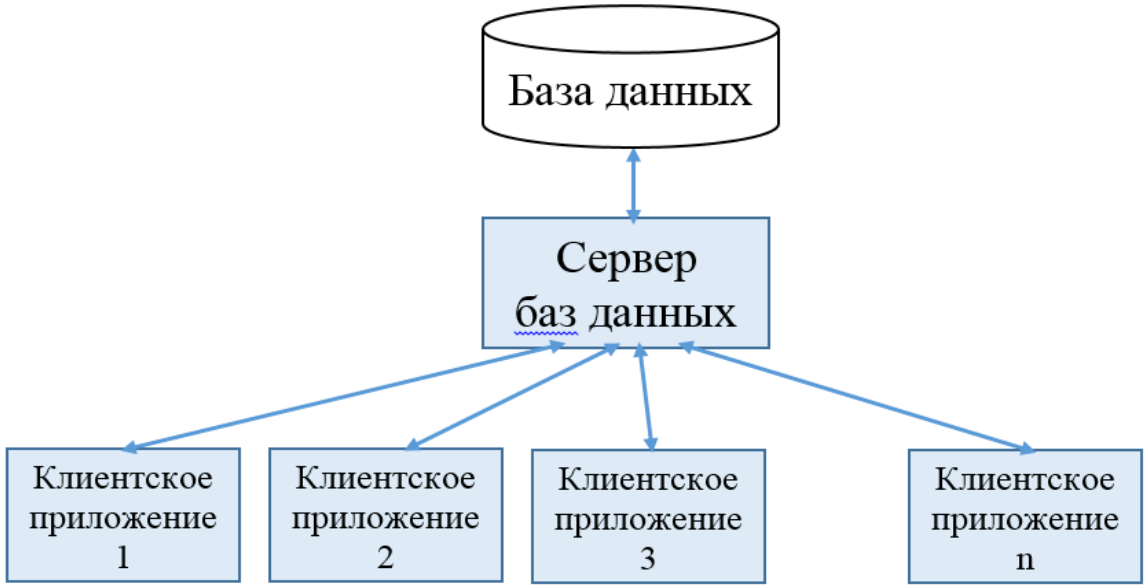
- небольшой объем обрабатываемых данных,
- малое количество пользователей (1 – 10).

Для крупных организаций использование файл-серверных технологий является неприемлемым.

Двухуровневая архитектура «клиент-сервер» (толстый клиент)

Особенностью архитектуры клиент-сервер является наличие выделенных серверов баз данных, принимающих запросы на языке структурированных запросов (Structured Query Language, SQL) и выполняющих поиск, сортировку и агрегирование информации.

В общем случае предполагается, что клиент и сервер функционируют на разных компьютерах.



Двухуровневая архитектура «клиент-сервер» (толстый клиент)

Разделение процесса выполнения запроса на клиентскую и серверную части позволяет выполнить следующее:

- Различным прикладным (клиентским) программам одновременно использовать общую базу данных;
- Централизовать функции управления;
- Защита информации;
- Обеспечение целостности данных;
- Управление совместным использованием ресурсов;
- Обеспечивать параллельную обработку запроса;
- Высвобождать ресурсы компьютеров-клиентов;
- Повышать эффективность управления данными за счет использования специализированных компьютеров — серверов баз данных.

Двухуровневая архитектура «клиент-сервер» (толстый клиент)

Технология работы двухуровневой архитектуры «клиент-сервер»

База данных в виде набора файлов находится па жестком диске специально выделенного компьютера (сервера сети). СУБД располагается также на сервере сети.

Существует локальная сеть, состоящая из клиентских компьютеров, на каждом из которых установлено клиентское приложение для работы с БД.

Пользователь, используя пользовательский интерфейс приложения инициирует обращение к СУБД, расположенной на сервере, с помощью специального языка запросов SQL (по сети от клиента к серверу передается лишь текст запроса).

СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере и инициирует обращения к данным, находящимся на сервере, в результате которых на сервере осуществляется вся обработка данных и лишь результат выполнения запроса копируется на клиентский компьютер. Таким образом, СУБД возвращает результат в приложение.

Используя пользовательский интерфейс, приложение отображает результат выполнения запросов.

Двухуровневая архитектура «клиент-сервер» (толстый клиент)

Функции приложения-клиента:

- посылка запросов серверу;
- интерпретация результатов запросов, полученных от сервера;
- представление результатов пользователю в некоторой форме (интерфейс пользователя).

Функции серверной части:

- прием запросов от приложений-клиентов;
- интерпретация запросов;
- оптимизация и выполнение запросов к БД;
- отправка результатов приложению-клиенту;
- обеспечение системы безопасности и разграничение доступа;
- управление целостностью БД;
- реализация стабильности многопользовательского режима работы.

Двухуровневая архитектура «клиент-сервер» (толстый клиент)

В архитектуре «клиент-сервер» работают так называемые «промышленные» СУБД – СУБД, которые могут обеспечить работу информационных систем масштаба среднего и крупного предприятия, организации, банка (MS SQL Server, Oracle, Gupta, Informix, Sybase, DB2, InterBase и ряд других).

Достоинства архитектуры «клиент-сервер» по сравнению с архитектурой «файл-сервер»

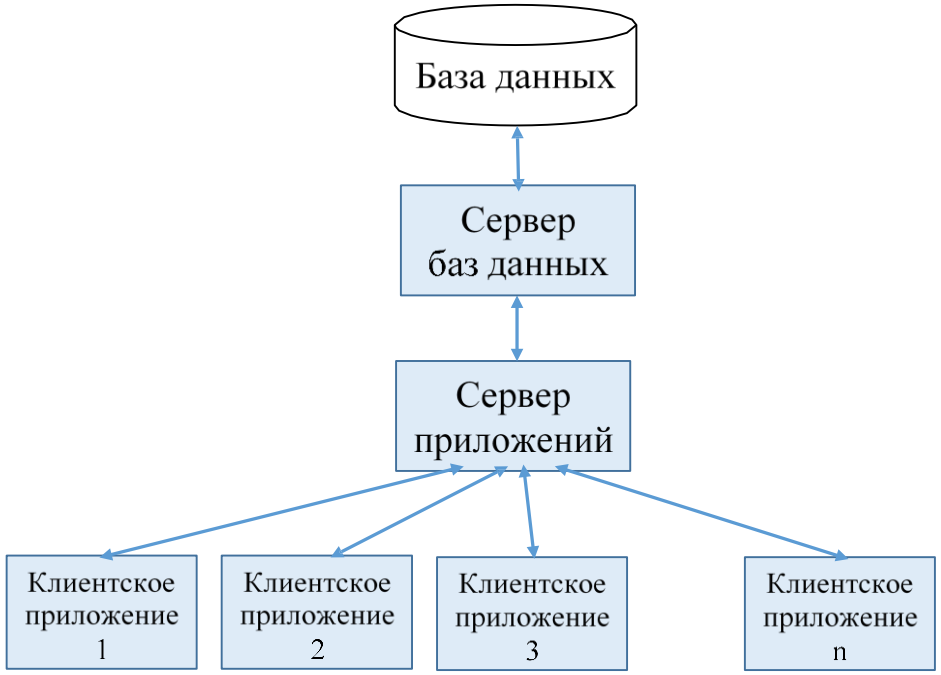
- Обеспечивается более широкий доступ к существующим базам данных;
- Повышается общая производительность системы (клиенты и сервер находятся на разных компьютерах, их процессоры способны выполнять приложения параллельно);
- Снижается стоимость аппаратного обеспечения (мощный компьютер с большим устройством хранения нужен только серверу для хранения и управления базой данных, нет необходимости в мощных рабочих станциях);
- Снижается нагрузка на сеть (приложения выполняют часть операций на клиентских компьютерах и посылают через сеть только запросы к базам данных);
- Повышается уровень непротиворечивости данных (сервер может самостоятельно управлять проверкой целостности данных, каждому приложению не придется выполнять собственную проверку).

Недостатки архитектуры «клиент-сервер»

- Высокие финансовые затраты на аппаратное и программное обеспечение;
- Трудности со своевременным обновлением клиентских приложений на всех компьютерах-клиентах, расположенных в разных местах.

Трехуровневая архитектура «клиент-сервер»

- Нижний уровень – компьютеры пользователей, на которых расположены приложения клиентов, обеспечивающие программный интерфейс для вызова приложения на среднем уровне.
- Средний уровень – сервер приложений обеспечивает обмен данными между пользователями и распределенными базами данных. Сервер приложений размещается в узле сети доступно всем клиентам.
- Верхний уровень – удаленный специализированный сервер базы данных, принимающий информацию от сервера приложений. Сервер без данных выделен для услуг обработки данных и файловых операций.



Трехуровневая архитектура «клиент-сервер»

Технология работы

- База данных в виде набора файлов находится на жестком диске специально выделенного компьютера (сервера сети);
- СУБД располагается также на сервере сети;
- На специально выделенном сервере приложений располагается программное обеспечение (бизнес-логика);
- На каждом из множества клиентских компьютеров установлен «тонкий» клиент - клиентское приложение, реализующее интерфейс пользователя. Используя предоставляемый приложением пользовательский интерфейс, он инициирует обращение к программному обеспечению, расположенному на сервере приложений;
- Сервер приложений анализирует требования пользователя и формирует запросы к БД. Для общения используется специальный язык запросов SOL, т.е. по сети от сервера приложений к серверу БД передается лишь текст запроса;
- СУБД инкапсулирует внутри себя все сведения о физической структуре БД, расположенной на сервере;
- СУБД инициирует обращения к данным, находящимся на сервере, в результате которых результат выполнения запроса копируется на сервер приложений;
- Сервер приложений возвращает результат в клиентское приложение (пользователю);
- Приложение, используя пользовательский интерфейс, отображает результат выполнения запросов.

Трехуровневая архитектура «клиент-сервер»

Достоинства

- Разгрузка сервера баз данных от выполнения части операций, перенесенных на сервер приложений;
- Уменьшение размера клиентских приложений за счет разгрузки их от лишнего кода;
- Единое поведение всех клиентов;
- Упрощение настройки клиентов – при изменении общего кода сервера приложений автоматически изменяется поведение приложений клиентов.

Объекты базы данных

Объекты базы данных содержат всю информацию о ее структуре и данных.

Таблица (table) — основной объект любой реляционной базы данных.

В таблицах хранятся все данные и метаданные базы данных.

Таблица — это плоская двумерная структура, содержащая произвольное количество строк. Часто строку называют записью (record). Таблица может не содержать и ни одной строки, т. е. может быть пустой. Все строки одной таблицы имеют одинаковую структуру. Они состоят из столбцов (полей). Таблица в реляционной базе данных может содержать не менее одного столбца. Каждый столбец имеет конкретный тип данных.

Объекты базы данных

Домен (domain) — объект базы данных, описывающий некоторые характеристики столбца. На домен можно ссылаться при описании столбцов создаваемой таблицы или при изменении характеристик столбцов существующей таблицы. В этом случае все характеристики домена копируются в столбец.

Объекты базы данных

Индекс (index) — объект базы данных, предназначенный для ускорения выборки данных из таблицы. Каждый индекс создается для одной конкретной таблицы. Индекс представляет собой множество упорядоченных строк, каждая из которых содержит значение полей, входящих в состав индекса, и указатель на строку таблицы, содержащую соответствующие значения этих полей. Существуют средства для активации/деактивации индексов, улучшения их характеристик. Для первичных, уникальных и внешних ключей система автоматически создает индексы.

Объекты базы данных

Генератор (generator) —простой объект реляционной базы данных, предназначенный для получения уникального числового значения, используемого, как правило, для формирования значения искусственного первичного ключа или иногда уникального ключа.

Хранимая процедура (stored procedure) — программа, написанная на процедурном расширении языка SOL, позволяющая выполнять различные действия с данными в базе данных. К хранимым процедурам могут обращаться хранимые процедуры этой базы данных, пользовательские (клиентские) программы. Хранимая процедура выполняется на стороне сервера, что во многих случаях может резко сократить сетевой трафик и заметно увеличить скорость решения задач предметной области.

Объекты базы данных

Триггер (trigger), как и хранимая процедура, является программой, написанной па процедурном расширении языка SOL, хранящейся в области метаданных и выполняемой на сервере. Однако обращение напрямую к триггеру невозможно. Он автоматически вызывается при наступлении одной из фаз события, связанного с изменением данных в таблицах, или события, связанного с подключением к базе данных и с работой с транзакциями. **События таблицы** — добавление данных, изменение строки таблицы, удаление строки. **Фазами** являются «до» (before) выполнения действия и «после» (after) выполнения действия.

Объекты базы данных

Пользовательские исключения (exception) — объект реляционной базы данных, который позволяет создавать, а затем выдавать сообщения пользователю при появлении некоторой ситуации в процессе работы программ с базой данных. Эти исключения могут использоваться только в хранимых процедурах и триггерах.

События базы данных (event) — существует возможность из хранимых процедур и триггеров передавать некоторые сообщения всем клиентским приложениям, работающим с конкретной базой данных. Это средство позволяет во многих случаях осуществлять синхронизацию отображения данных либо выполнять более сложные действия по взаимодействию клиентских приложений при их совместной одновременной работе с базой данных.

Объекты базы данных

Представление (view; обзор, просмотр) — результат выборки данных из одной или более таблиц базы данных на основании конкретных критериев. Основа представления — оператор SELECT любой сложности. Представление — удобное средство для получения данных в случае сложной выборки данных. Кроме того, представления позволяют скрыть от пользователя некоторые столбцы таблиц, значения которых не предназначены для просмотра этим пользователем.

Функции, определенные пользователем (User Defined Functions, UDF) — функции, написанные на любом языке программирования и хранящиеся вне базы данных, но описанные в этой базе. Могут использоваться для расширения возможностей языка SQL и соответствующих языков программирования. Часто позволяют описывать довольно сложные действия с данными из базы данных.

Транзакции

Транзакция — это неделимая, с точки зрения воздействия на данные, последовательность операций манипулирования данными. Для пользователя транзакция выполняется по принципу «все или ничего». Либо транзакция выполняется целиком и переводит базу данных из одного целостного (непротиворечивого) состояния в другое целостное состояние, либо, если по каким-либо причинам, одно из действий транзакции невыполнимо или произошло какое-либо нарушение работы системы, база данных возвращается в исходное состояние, которое было до начала транзакции, т. е. происходит откат транзакции.

Понятие транзакции имеет непосредственную связь с понятием целостности БД – верной структуре и непротиворечивости содержимого БД.

Транзакции

БД должна обеспечивать корректную параллельную работу всех пользователей над одними и теми же данными.

Для согласованного выполнения параллельных транзакций требуется выработать процедуру, которая должна удовлетворять следующим правилам:

- В ходе выполнения транзакции пользователь видит только согласованные данные. Пользователь не должен видеть несогласованных промежуточных данных;
- Когда в БД две транзакции выполняются параллельно, СУБД гарантированно поддерживает принцип независимого выполнения транзакций, который гласит, что результаты выполнения транзакций будут такими же, как если бы вначале выполнялась транзакция 1, а потом транзакция 2, или наоборот, сначала транзакция 2, а потом транзакция 1.

Транзакции

Уровень изолированности транзакций — значение, определяющее уровень, при котором в транзакции допускаются несогласованные данные, т.е. степень изолированности одной транзакции от другой. Более высокий уровень изолированности повышает точность данных, но при этом может снижаться количество параллельно выполняемых транзакций. С другой стороны, более низкий уровень изолированности позволяет выполнять больше параллельных транзакций, но снижает точность данных.

Транзакции

Уровни изолированности транзакций

- Самый высокий уровень изолированности соответствует протоколу сериализации транзакций. Этот уровень, называемый SERIALIZABLE (упорядочиваемость), обеспечивает полную изоляцию транзакций и полную корректную обработку параллельных транзакций.
- Уровень подтвержденного чтения REPEATABLE READ. На этом уровне транзакция не имеет доступа к промежуточным или окончательным результатам других транзакций, поэтому такие проблемы, как пропавшие обновления, промежуточные или несогласованные данные, возникнуть не могут. Однако во время выполнения своей транзакции вы можете увидеть строку, добавленную в базу данных другой транзакцией. Поэтому один и тот же запрос, выполненный в течение одной транзакции, может дать разные результаты, т.е. проблема строк - призраков остается.

Транзакции

Уровни изолированности транзакций

- Уровень READ COMMITED (чтение фиксированных данных). На этом уровне изолированности транзакция не имеет доступа к промежуточным результатам других транзакций, поэтому проблемы пропавших обновлений и промежуточных данных возникнуть не могут. При этом уровне изолированности транзакция не может обновлять строку, уже обновленную другой транзакцией. При попытке выполнить подобное обновление транзакция будет отменена автоматически во избежание возникновения проблемы пропавшего обновления.

Транзакции

Уровни изолированности транзакций

- Самый низкий уровень изолированности – уровень неподтвержденного, или «грязного» – READ UNCOMMITED. При этом уровне изолированности текущая транзакция видит промежуточные и несогласованные данные, ей также доступны строки-призраки. Если несколько транзакций одновременно пытались изменять одну и ту же строку, то в окончательном варианте строка будет иметь значение, определенное последней успешно выполненной транзакцией. Однако даже при этом уровне изолированности СУБД предотвращает пропавшие обновления.

Транзакции

Для того чтобы обеспечить изолированность транзакций, в СУБД должны использоваться методы регулирования их совместного выполнения.

Способ выполнения набора транзакций, в котором операции разных транзакций чередуются или выполняются параллельно, называется сериальным, если результат совместного выполнения транзакций эквивалентен результату некоторого последовательного выполнения этих же транзакций.

Сериализация транзакций — механизм их выполнения по некоторому сериальному плану. Обеспечение такого механизма является основной функцией компонента СУБД, ответственного за управление транзакциями. Система, в которой поддерживается сериализация транзакций, обеспечивает реальную изолированность пользователей.

PROАГРО
ЛЕКТОРИЙ

Спасибо за внимание!