БАЗЫ ДАННЫХ
часть II

Распределенные и
параллельные системы
управления базами данных

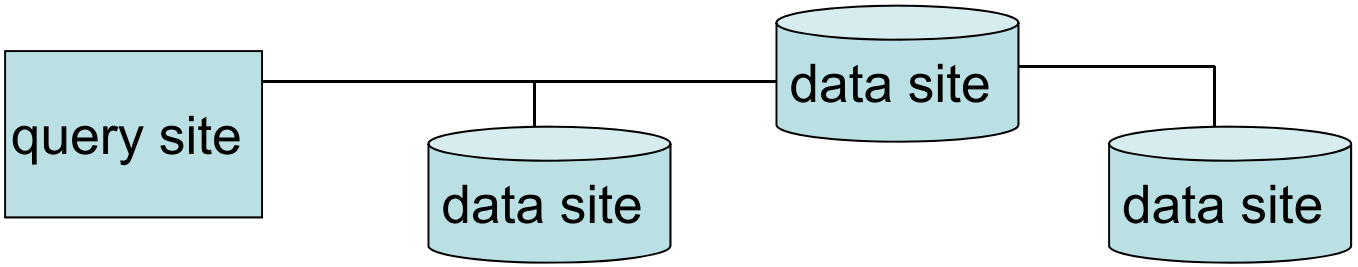
Распределенные и параллельные СУБД

Распределенная база данных (DDB - distributed database) - это совокупность множества взаимосвязанных баз данных, распределенных в компьютерной сети.

Система управления распределенной базой данных определяется как программная система, которая позволяет управлять базой данных таким образом, чтобы ее распределенность была прозрачна для пользователей

Распределенные и параллельные СУБД

Важнейший отличительный признак - слабосвязанный характер среды, где каждый узел имеет собственную операционную систему и функционирует независимо.



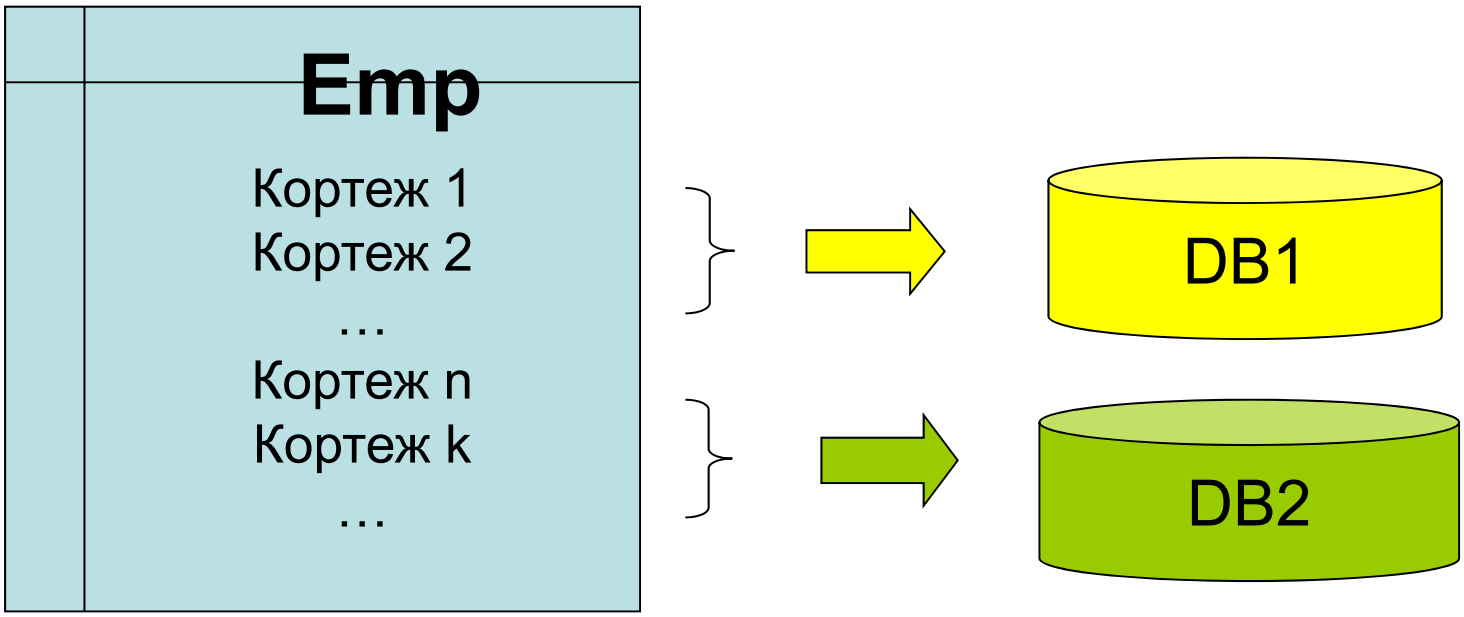
Распределенные и параллельные СУБД

База данных физически распределяется по узлам данных при помощи **фрагментации** и **репликации**, или **тиражирования** данных.

Отношения, принадлежащие реляционной базе данных, могут быть фрагментированы на *горизонтальные* или *вертикальные* разделы.

Распределенные и параллельные СУБД

Горизонтальная фрагментация реализуется при помощи операции селекции, которая направляет каждый кортеж отношения в один из разделов, руководствуясь предикатом фрагментации.



Распределенные и параллельные СУБД

При *вертикальной фрагментации* отношение делится на разделы при помощи операции проекции.

Распределенные и параллельные СУБД

Параллельную СУБД можно определить как реализацию СУБД для многопроцессорного компьютера.

Решение заключается в применении широкомасштабного параллелизма, чтобы усилить мощность отдельных компонентов путем их интеграции в целостную систему на основе соответствующего программного обеспечения параллельных баз данных.

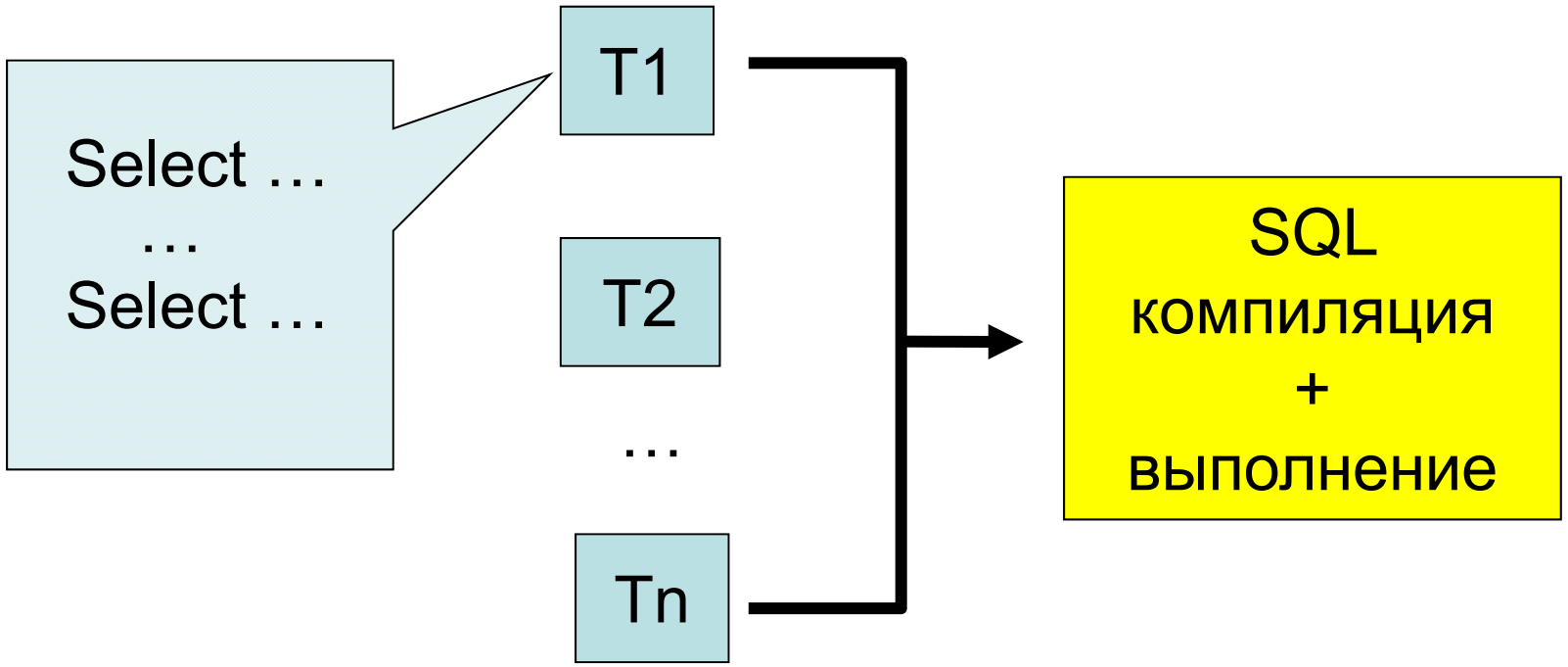
Распределенные и параллельные СУБД

В программном обеспечении базы данных могут быть предусмотрены три вида параллелизма, присущие приложениям интенсивной обработки данных:

- **межзапросный параллелизм**
- **внутризапросный параллелизм**
- **внутриоперационный параллелизм**

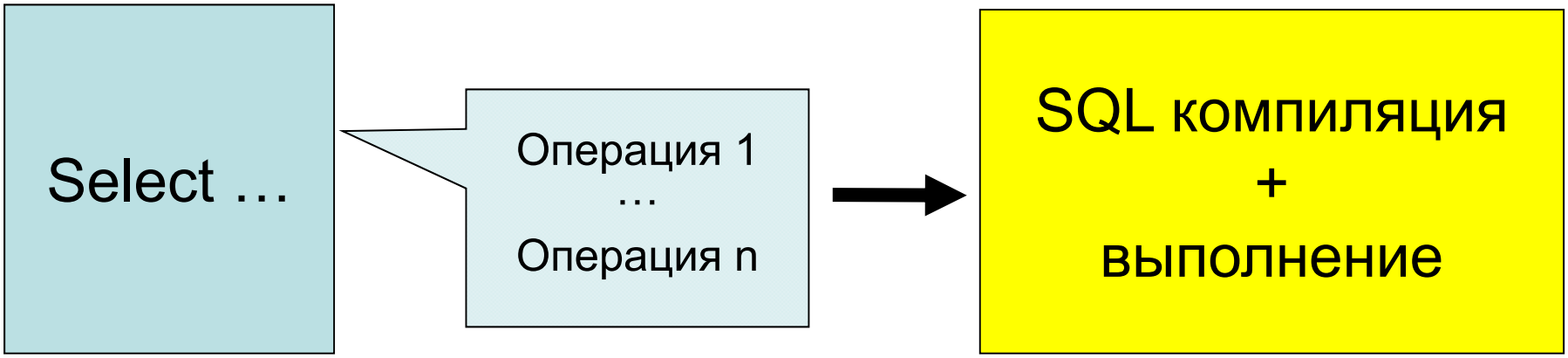
Распределенные и параллельные СУБД

Межзапросный параллелизм предполагает одновременное выполнение множества запросов, относящихся к разным транзакциям.



Распределенные и параллельные СУБД

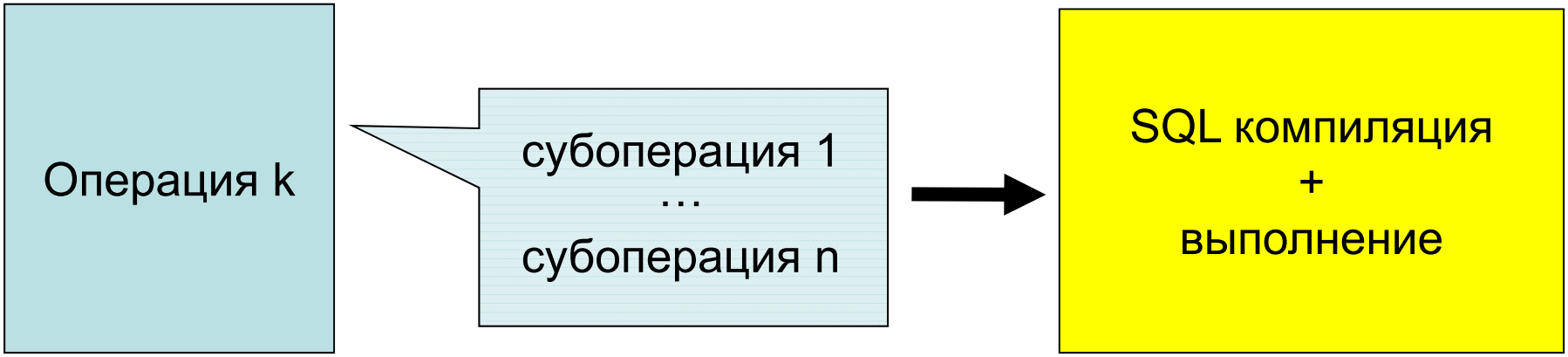
Под **внутризапросным параллелизмом** понимается одновременное выполнение сразу нескольких операций (например операций выборки), относящихся к одному и тому же запросу.



И внутризапросный, и межзапросный параллелизм реализуется на основе *разбиения данных* по типу горизонтальной фрагментации.

Распределенные и параллельные СУБД

Понятие **внутриоперационного параллелизма** означает параллельное выполнение одной операции в виде множества субопераций с применением, в дополнение к фрагментации данных, также *фрагментации функций*.



Распределенные и параллельные СУБД

Идентифицирующие характеристики параллельных и распределенных СУБД:

1. Распределенная/параллельная база данных - это именно база данных, а не "коллекция" файлов, индивидуально хранимых на разных узлах сети.
2. Система обладает полной функциональностью СУБД. Она не сводится по своим возможностям ни к распределенным файловым системам, ни к системам обработки транзакций.
3. Распределение (включая фрагментацию и репликацию) данных по множеству узлов невидимо для пользователей (***прозрачность***).

Распределенные и параллельные СУБД

Технология распределенных/параллельных баз данных распространяет основополагающую для управления базами данных концепцию *независимости данных* на среду, где данные распределены и тиражированы по множеству компьютеров, связанных сетью.

Это обеспечивается за счет нескольких видов прозрачности: *прозрачность сети* (следовательно, *прозрачность распределения*), *прозрачность репликации* и *прозрачность фрагментации*.

Распределенные и параллельные СУБД

Прозрачность доступа означает, что пользователи имеют дело с единым логическим образом базы данных и осуществляют доступ к распределенным данным точно так же, как если бы они хранились централизованно.

В идеале полная прозрачность подразумевает наличие языка запросов к распределенной СУБД, не отличающегося от языка для централизованной СУБД.

Распределенные и параллельные СУБД

- Высокая производительность - одна из важнейших целей, на достижение которой направлены технологии параллельных СУБД. Как правило, она обеспечивается за счет сочетания нескольких взаимно дополняющих решений, таких как применение операционных систем, ориентированных на поддержку баз данных, параллелизм, оптимизация, баланс загрузки.

- Наличие операционной системы, "осведомленной" о специфических потребностях баз данных, упрощает реализацию функций баз данных нижнего уровня и способствует снижению их стоимости.

Распределенные и параллельные СУБД

- Затраты на передачу сообщения могут быть значительно снижены за счет применения специализированного коммуникационного протокола.

- Механизмы распараллеливания способствуют повышению общей пропускной способности системы (межзапросный параллелизм), снижению времени отклика для отдельных транзакций (внутризапросный и внутриоперационный параллелизм).

Распределенные и параллельные СУБД

- Технологии распределенных и параллельных СУБД направлены также на повышение надежности, поскольку благодаря репликациям данных исключаются одиночные точки отказа.
- Отказ одного узла или сбой на линии связи не приводит к выходу из строя всей системы. Даже если часть данных становится недоступной, при правильной организации системы пользователи могут иметь доступ к остальной части информации.

Распределенные и параллельные СУБД

В идеале параллельная (и, в меньшей степени, распределенная) СУБД обладает свойством **линейной расширяемости** и **линейного ускорения**.

Распределенные и параллельные СУБД

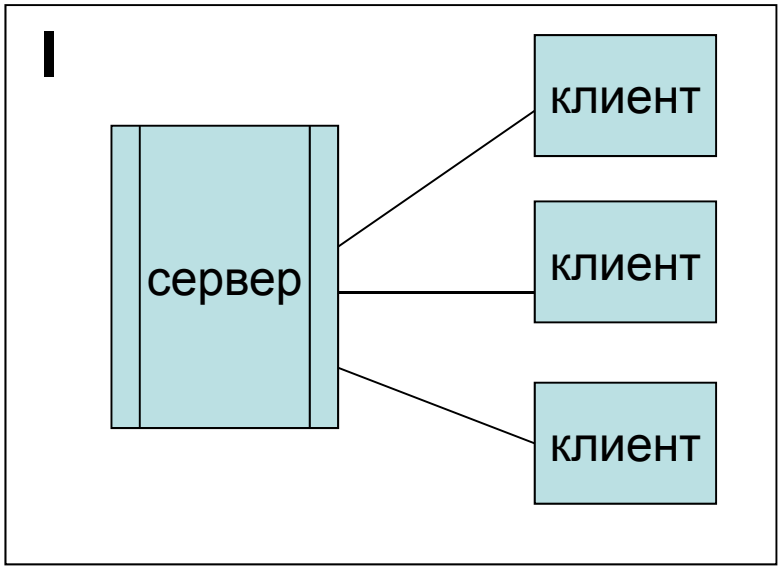
Под **линейной расширяемостью** понимается сохранение того же уровня производительности при увеличении размера базы данных и одновременном пропорциональном увеличении процессорной мощности и объема памяти.

Распределенные и параллельные СУБД

Линейное ускорение означает, что с наращиванием процессорной мощности и объема памяти при сохранении прежнего размера базы данных пропорционально возрастает производительность.

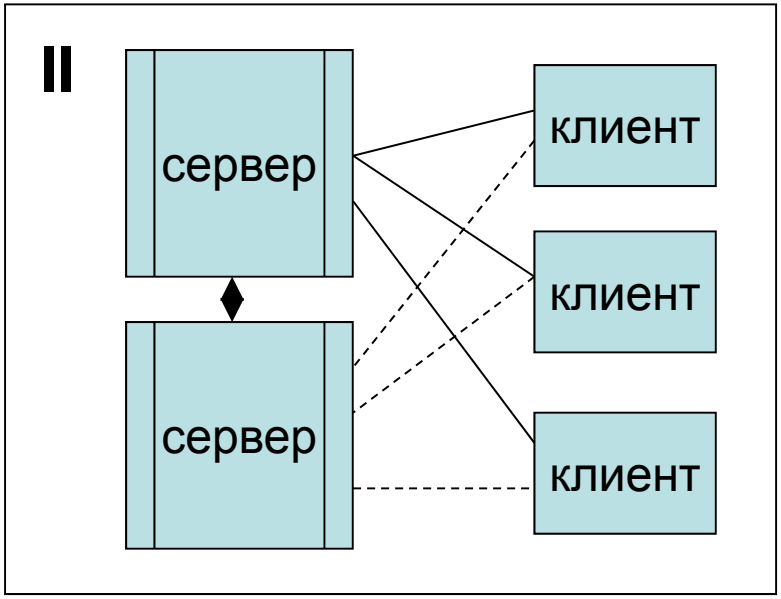
Распределенные и параллельные СУБД

Наиболее популярна в настоящее время **архитектура клиент-сервер**, когда множество машин-клиентов осуществляют доступ к одному серверу баз данных.



Распределенные и параллельные СУБД

Архитектура типа *много-клиентов/много-серверов,* когда база данных размещена на множестве серверов, которым, для того чтобы вычислить результат пользовательского запроса или выполнить транзакцию, необходимо взаимодействовать друг с другом.

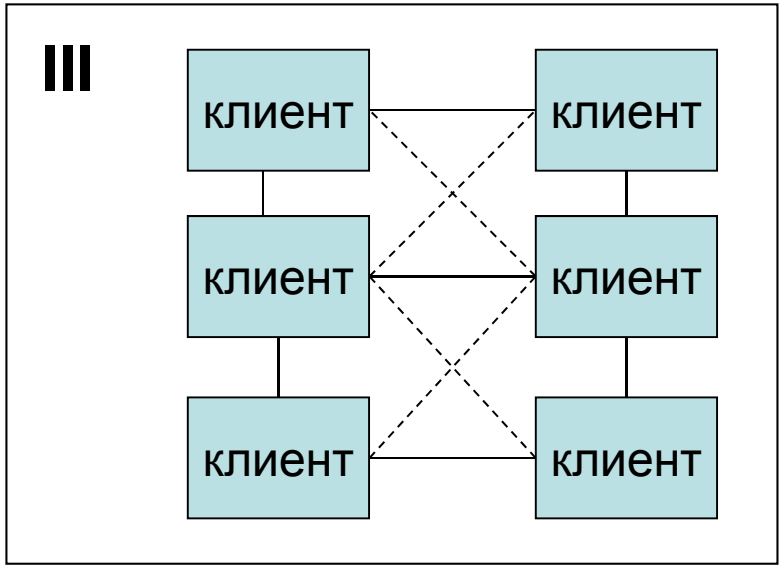


Каждая клиентская машина имеет свой "домашний" сервер; ему она направляет пользовательские запросы. Взаимодействие серверов друг с другом прозрачно для пользователей.

Распределенные и параллельные СУБД

В истинно распределенной СУБД клиентские и серверные машины не различаются. В идеале каждый узел может выступать и как клиент, и как сервер.

Такие архитектуры, тип которых определяют как *равный-к-равному,* требуют сложных протоколов управления данными, распределенными по множеству узлов.



Распределенные и параллельные СУБД

Архитектуры параллельных систем варьируются между двумя крайними точками, называемыми **архитектура без разделяемых ресурсов** и **архитектура с разделяемой памятью.** Промежуточную позицию занимает архитектура с разделяемыми дисками.

Распределенные и параллельные СУБД

В случае **неразделения ресурсов** каждый процессор имеет эксклюзивный доступ к собственной оперативной памяти и к набору дисков.

Архитектуры без разделяемых ресурсов обладают тремя важнейшими преимуществами: низкие затраты, расширяемость, высокая доступность

Наиболее существенные проблемы - сложность реализации и (потенциальные) трудности соблюдения баланса загрузки.

Распределенные и параллельные СУБД

Подход, основанный на разделении памяти, заключается в том, что каждый процессор посредством быстрых линий связи (высокоскоростных шин или кросс-панельных коммутаторов) соединен со всеми модулями памяти и дисковыми устройствами.

Сильные стороны - простота и хороший баланс загрузки.

Наиболее существенные проблемы - стоимость, ограниченная расширяемость, невысокая надежность.

Распределенные и параллельные СУБД

В системах с разделяемыми дисками каждый процессор имеет доступ к любому дисковому устройству посредством специальных соединений и эксклюзивный доступ к своей собственной оперативной памяти.

Преимущества: низкие затраты, расширяемость, хороший баланс загрузки, высокая доступность, простота миграции с однопроцессорных систем.

Трудности: сложность системы, потенциальные проблемы производительности.

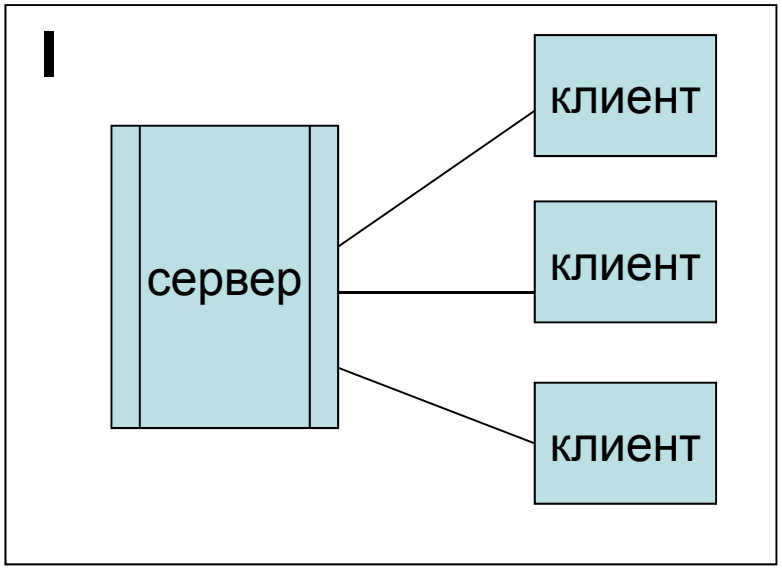
БАЗЫ ДАННЫХ
часть II

Распределенные и
параллельные системы
управления базами данных

Возможные архитектуры баз данных

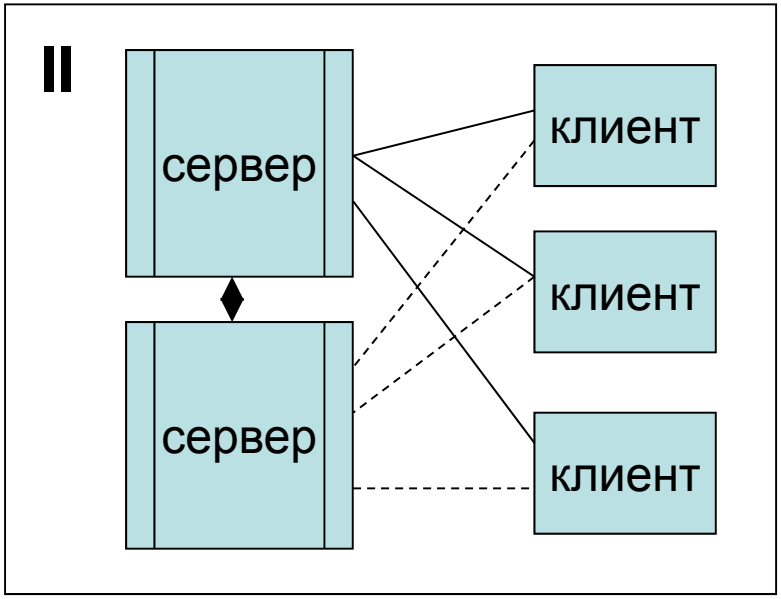
Распределенные и параллельные СУБД

Наиболее популярна в настоящее время **архитектура клиент-сервер**, когда множество машин-клиентов осуществляют доступ к одному серверу баз данных.



Распределенные и параллельные СУБД

Архитектура типа *много-клиентов/много-серверов,* когда база данных размещена на множестве серверов, которым, для того чтобы вычислить результат пользовательского запроса или выполнить транзакцию, необходимо взаимодействовать друг с другом.

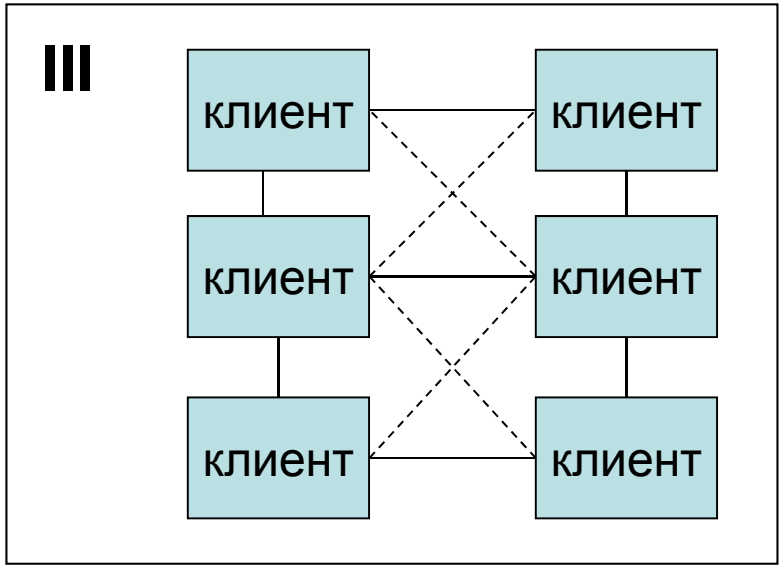


Каждая клиентская машина имеет свой "домашний" сервер; ему она направляет пользовательские запросы. Взаимодействие серверов друг с другом прозрачно для пользователей.

Распределенные и параллельные СУБД

В истинно распределенной СУБД клиентские и серверные машины не различаются. В идеале каждый узел может выступать и как клиент, и как сервер.

Такие архитектуры, тип которых определяют как *равный-к-равному,* требуют сложных протоколов управления данными, распределенными по множеству узлов.



Распределенные и параллельные СУБД

Архитектуры параллельных систем варьируются между двумя крайними точками, называемыми **архитектура без разделяемых ресурсов** и **архитектура с разделяемой памятью.** Промежуточную позицию занимает архитектура с разделяемыми дисками.

Распределенные и параллельные СУБД

В случае **неразделения ресурсов** каждый процессор имеет эксклюзивный доступ к собственной оперативной памяти и к набору дисков.

Архитектуры без разделяемых ресурсов обладают тремя важнейшими преимуществами: низкие затраты, расширяемость, высокая доступность

Наиболее существенные проблемы - сложность реализации и (потенциальные) трудности соблюдения баланса загрузки.

Распределенные и параллельные СУБД

Подход, основанный на разделении памяти, заключается в том, что каждый процессор посредством быстрых линий связи (высокоскоростных шин или кросс-панельных коммутаторов) соединен со всеми модулями памяти и дисковыми устройствами.

Сильные стороны - простота и хороший баланс загрузки.

Наиболее существенные проблемы - стоимость, ограниченная расширяемость, невысокая надежность.

Распределенные и параллельные СУБД

В системах с разделяемыми дисками каждый процессор имеет доступ к любому дисковому устройству посредством специальных соединений и эксклюзивный доступ к своей собственной оперативной памяти.

Преимущества: низкие затраты, расширяемость, хороший баланс загрузки, высокая доступность, простота миграции с однопроцессорных систем.

Трудности: сложность системы, потенциальные проблемы производительности.

Обработка и оптимизация запросов

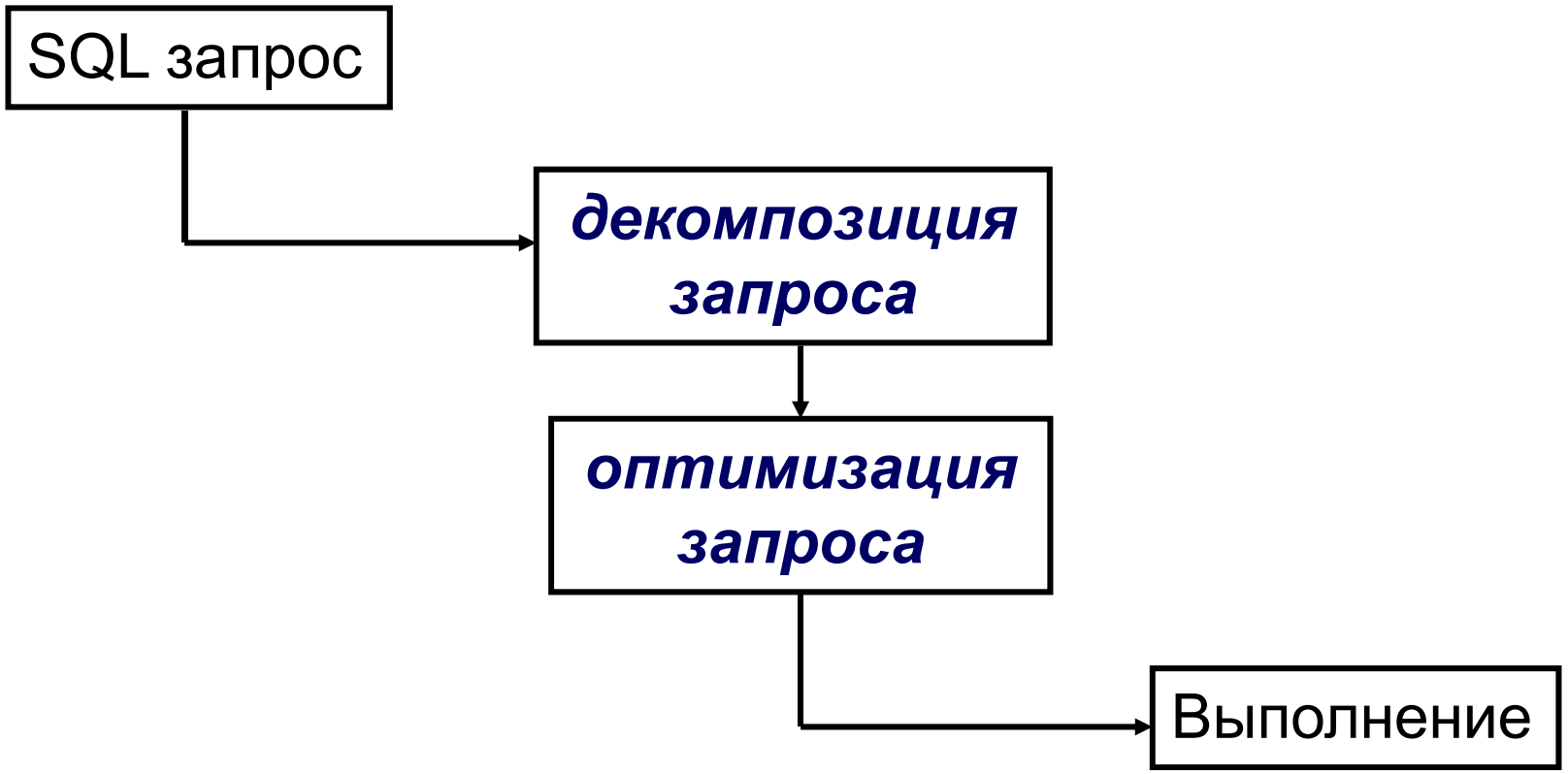
Распределенные и параллельные СУБД

Обработка запроса - это процесс трансляции декларативного определения запроса в операции манипулирования данными низкого уровня. Стандартным языком запросов, поддерживаемым современными СУБД, является SQL.

Оптимизация запроса - это процедура выбора "наилучшей" стратегии для реализации запроса из множества альтернатив.

Распределенные и параллельные СУБД

Для централизованной СУБД весь процесс состоит обычно из двух шагов:



Распределенные и параллельные СУБД

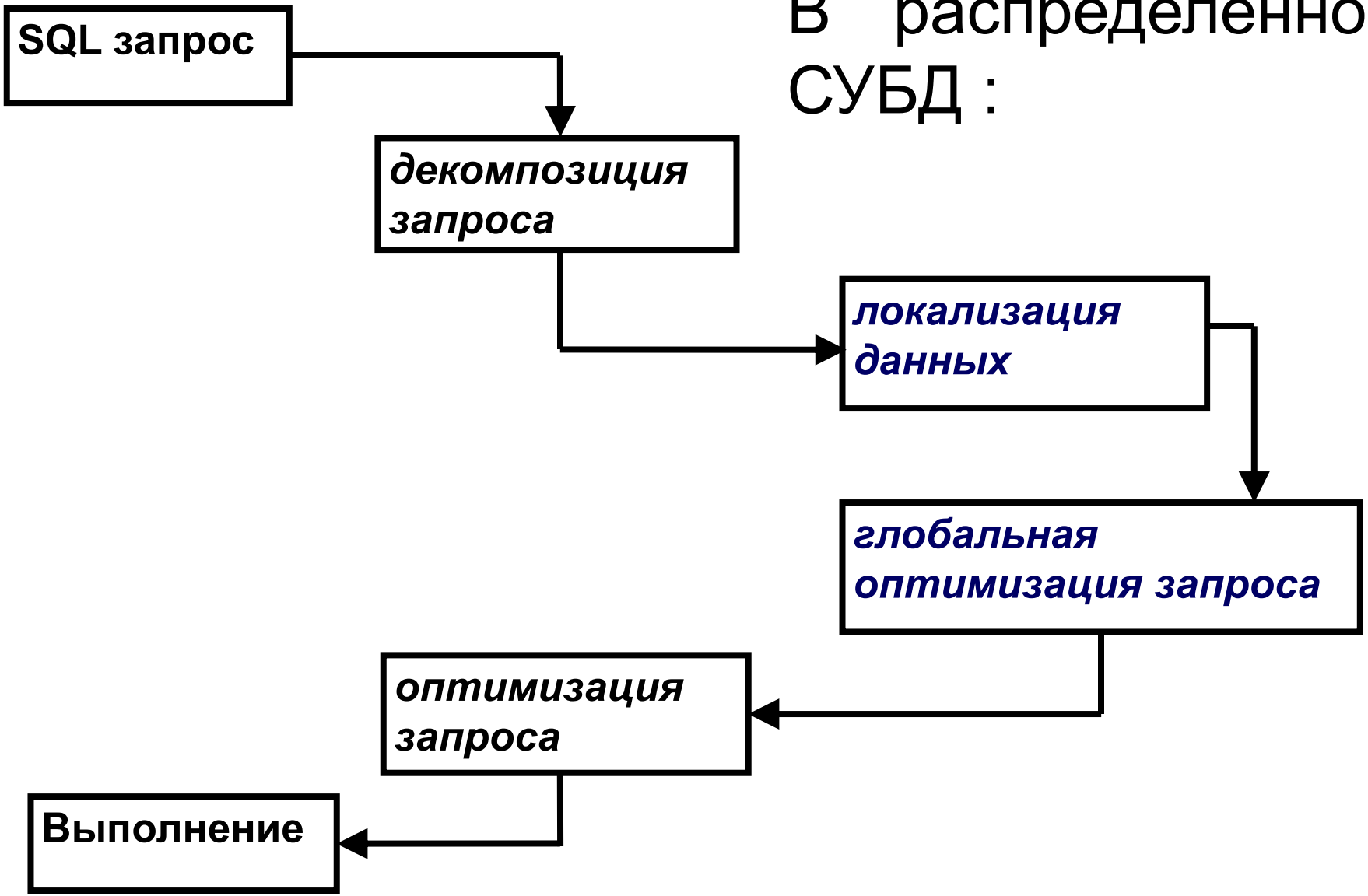
Декомпозиция запроса - это трансляция его с языка SQL в выражение реляционной алгебры. В ходе декомпозиции запрос подвергается семантическому анализу; при этом некорректные запросы отвергаются, а корректные упрощаются. Упрощенный запрос преобразуется в алгебраическую форму.

Распределенные и параллельные СУБД

"Качество" алгебраического выражения определяется исходя из объема затрат, необходимых для его вычисления. SQL-запрос транслируется в какое-нибудь выражение, а затем, применяя правила эквивалентных алгебраических преобразований, получают из него другие алгебраические преобразования, пока не будет найдено "наилучшее". При поиске "наилучшего" выражения используется функция стоимости, в соответствии с которой вычисляется сумма затрат, необходимых для выполнения запроса. Этот процесс и называется ***оптимизацией запросов***.

Распределенные и параллельные СУБД

В распределенной СУБД :



Распределенные и параллельные СУБД

Исходной информацией для локализации данных служит алгебраическое выражение, полученное на этапе декомпозиции запроса. В этом выражении фигурируют глобальные отношения без учета их фрагментации или распределения. Сущность данного шага заключается в том, чтобы локализовать участвующие в запросе данные, используя информацию об их распределении. При этом выявляются фрагменты, реально участвующие в запросе, а запрос преобразуется к форме, где операции применяются уже не к глобальным отношениям, а к фрагментам.

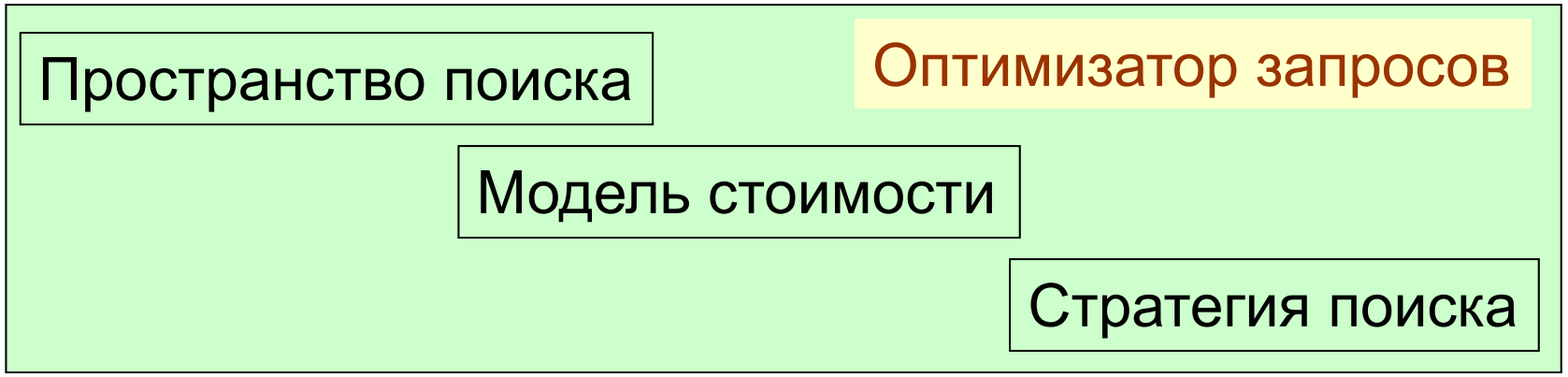
Распределенные и параллельные СУБД

Распределенные отношения реконструируются путем применения инверсии правил фрагментации. Это называется ***программой локализации***. Таким образом, на этапе локализации данных запрос заменяется программой локализации; фрагментный запрос затем упрощается и реструктурируется, пока не будет получено "хорошее" выражение. Как и в шаге декомпозиции, окончательный "хороший" фрагментный запрос может быть еще далек от оптимального; данный процесс лишь исключает "плохие" алгебраические выражения.

Распределенные и параллельные СУБД

Цель ***глобальной оптимизации*** - найти стратегию выполнения запроса, близкую к оптимальной. Стратегию выполнения распределенного запроса можно выразить в терминах *операций реляционной алгебры* и *коммуникационных примитивов* (операций "послать"/"получить"), описывающих пересылки данных между узлами. Однако проведенные оптимизации на предыдущих этапах не зависели от характеристик фрагментов; еще не учтены коммуникационные операции. Путем перестановок операций в рамках фрагментного запроса можно получить множество эквивалентных планов его выполнения. Оптимизация запроса заключается в

Распределенные и параллельные СУБД



Пространство поиска - это множество альтернативных планов выполнения исходного запроса.

Модель стоимости - это способ оценить стоимость реализации заданного плана.

Стратегия поиска - это способ обхода пространства поиска и выбора наилучшего плана.

Распределенные и параллельные СУБД

В распределенной среде ***функция стоимости***, часто определяемая в единицах времени, оценивает затраты вычислительных ресурсов, таких как дисковое пространство, число обменов с дисками, время CPU, коммуникации и т. д. Обычно это некоторая взвешенная сумма затрат ввода-вывода, CPU и коммуникаций.

В распределенных СУБД применяется упрощенный подход, когда в качестве наиболее значимых рассматриваются лишь коммуникационные затраты.

Распределенные и параллельные СУБД

Внутриоперационный параллелизм достигается за счет выполнения операции сразу на нескольких узлах многопроцессорной машины.

Множество узлов, на которых хранится отношение, называется ***домашним множеством***.

Домашним множеством узлов операции называется множество узлов, на которых она выполняется; оно должно совпадать с домашними множествами узлов ее операндов, для того чтобы операция имела доступ к своим исходным данным.

Распределенные и параллельные СУБД

Межоперационный параллелизм имеет место, когда одновременно выполняются две или более операции, независимые или связанные общим потоком данных. Термином ***поток данных*** мы обозначаем форму параллелизма, реализуемую методами *конвейерной обработки*.

При ***независимом*** параллелизме операции выполняются одновременно или в произвольном порядке. Независимый параллелизм возможен, только если операции не содержат в качестве операндов общих данных.

БАЗЫ ДАННЫХ
часть II

Распределенные и
параллельные системы
управления базами данных

Управление доступом

Распределенные и параллельные СУБД

Знакомые понятия:

- синхронизация доступа;
- сериализация;
- транзакция;
- свойство изолированности;
- механизм блокировок;
- двухфазовый протокол блокирования.

Распределенные и параллельные СУБД

Для распределенных СУБД возникает задача распространения свойства сериализуемости и алгоритмов управления одновременным доступом на распределенную среду. Сложность - на разных узлах упорядочение одного и того же множества транзакций может оказаться различным.

Свойство ***глобальной сериализуемости***.

Выполнение множества распределенных транзакций сериализуемо тогда и только тогда, когда:

1. выполнение этого множества транзакций сериализуемо на каждом узле;
2. упорядочение транзакций на всех узлах одинаково.

Распределенные и параллельные СУБД

В алгоритмах, основанных на блокировании, для этого применяется один из трех методов: ***централизованное блокирование***, ***блокирование первичной копии*** и ***распределенное блокирование***.

Распределенные и параллельные СУБД

При ***централизованном блокировании*** для всей распределенной базы данных поддерживается единая таблица блокировок. Эта таблица, располагаемая на одном из узлов, находится под управлением единого менеджера блокировок. Менеджер блокировок отвечает за установку и снятие блокировок от имени всех транзакций. Поскольку управление блокировками сосредоточено на одном узле, то оно аналогично централизованному управлению одновременным доступом, и глобальная сериализуемость обеспечивается достаточно легко.

Проблемы: 1) центральный узел может стать узким местом как из-за большого объема обработки данных; 2) надежность такой системы ограничена, поскольку отказ или недоступность центрального узла приводит к выходу из строя всей системы.

Распределенные и параллельные СУБД

Блокирование первичной копии - это алгоритм управления одновременным доступом, применяемый для баз данных с репликациями, где копии одних и тех же данных могут храниться на множестве узлов. Одна из таких копий выделяется как первичная, и для доступа к любому элементу данных необходимо установить блокировку на его первичную копию. Множество первичных копий элементов данных известно всем узлам распределенной системы, и запросы транзакций на блокирование направляются узлам, где хранятся первичные копии. Если в распределенной базе данных репликации не используются, то данный алгоритм сводится к алгоритму распределенного блокирования. Алгоритм блокирования первичных копий был предложен для прототипа распределенной версии Ingres.

Распределенные и параллельные СУБД

Алгоритм ***распределенного*** (или *децентрализованного*) ***блокирования*** предполагает распределение обязанностей по управлению блокировками между всеми узлами системы. Для выполнения транзакции необходимо участие и взаимная координация менеджеров блокировок на нескольких узлах. Блокировки устанавливаются на всех узлах, данные которых участвуют в транзакции.

Алгоритмы этого типа сложнее, а коммуникационные затраты, необходимые для установки всех требуемых блокировок, выше.

Распределенные и параллельные СУБД

Общий побочный эффект всех алгоритмов управления одновременным доступом посредством блокирования - возможность **тупиковых ситуаций.**

Задача обнаружения и преодоления тупиков особенно сложна в распределенных системах. Тем не менее, благодаря относительной простоте и эффективности алгоритмов блокирования, они имеют значительно большую популярность, чем альтернативные *алгоритмы, основанные на* ***временных метках***, а также *алгоритмы* ***оптимистичного управления одновременным доступом***.

Распределенные и параллельные СУБД

Протоколы обеспечения надежности

Распределенные и параллельные СУБД

В распределенной СУБД различаются четыре типа сбоев:

сбой транзакции,

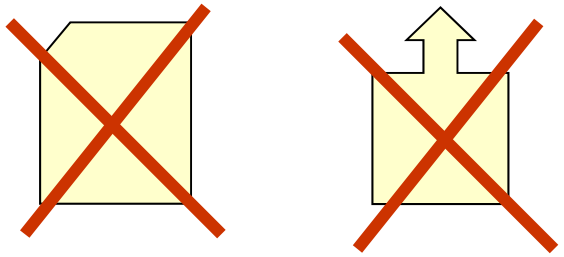
сбой узла *(системы)*,

сбой носителя *(диска)*,

сбой коммуникационной линии.

Распределенные и параллельные СУБД

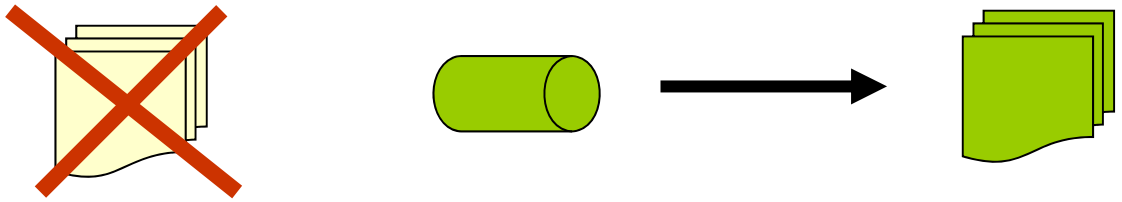
Причин ***сбоев транзакции*** может быть несколько: ошибки, вызванные неверными входными данными, обнаружение возникшего или возможного тупика. Обычный способ обработки таких сбоев заключается в том, чтобы прервать транзакцию и откатить базу данных к состоянию, предшествовавшему началу транзакции.



Распределенные и параллельные СУБД

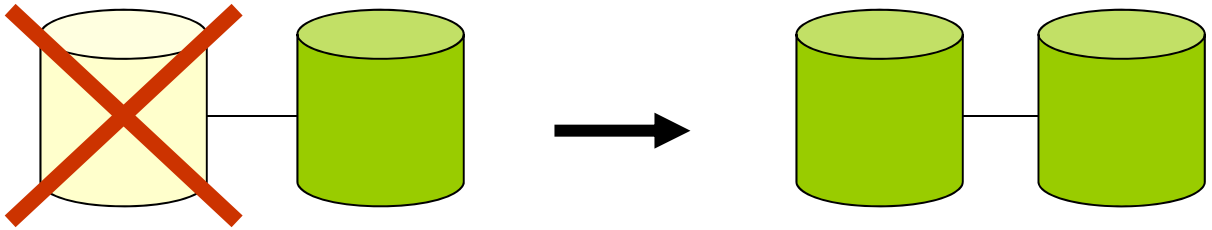
Сбои узлов (систем) могут быть вызваны аппаратными отказами или программными ошибками (в системном или прикладном коде). Системные сбои приводят к потере содержимого оперативной памяти (потеря всех элементов данных, находящиеся в буферах оперативной памяти - **неустойчивая база данных**). В то же время данные, находящиеся во вторичной памяти, остаются в сохранности (**стабильной базой данных**). Для поддержания сохранности данных обычно применяют протоколы журнализации.

В распределенной базе данных отказавший узел не может участвовать в транзакциях.



Распределенные и параллельные СУБД

Сбои носителей связаны с отказами устройств вторичной памяти, на которых хранится стабильная база данных. Обычно эта проблема решается путем применения дуплексных устройств и поддержания архивных копий базы данных. Сбои носителей рассматриваются обычно как локальная проблема узла, и специальных механизмов для их обработки в распределенных СУБД не предусматривается.



Распределенные и параллельные СУБД

Коммуникационные сбои являются специфической проблемой распределенных баз данных. Наиболее распространенные - ошибки в сообщениях, нарушение упорядоченности сообщений, потерянные сообщения, повреждения на линиях связи.

Ожидающий узел по истечении определенного промежутка времени просто решает, что узел-партнер стал недоступен.

Серьезным последствием повреждений на линиях связи может стать фрагментация сети, когда множество узлов распадается на группы, внутри которых имеется связь, а коммуникации между группами невозможны. В такой ситуации сложно обеспечить доступ пользователей к системе, сохраняя при этом ее ***непротиворечивость***.

Распределенные и параллельные СУБД

Протоколы обеспечения надежности поддерживают два свойства транзакций: **атомарность** и **долговечность.**

Атомарность означает…

Долговечность означает, что результат успешно завершенной (зафиксированной) транзакции сохраняется даже при последующих сбоях.

Распределенные и параллельные СУБД

Для обеспечения атомарности и долговечности необходимы *протоколы* ***атомарной фиксации*** и *протоколы* ***распределенного восстановления***.

Наиболее популярным из протоколов атомарной фиксации является протокол **двухфазовой фиксации транзакций**.

Протоколы восстановления надстраиваются над протоколами локального восстановления, которые зависят от режима взаимодействия СУБД с операционной системой.

Распределенные и параллельные СУБД

Двухфазовая фиксация (2PC - 2 Phase Commit) - каждый участвующий в ней узел, прежде чем зафиксировать транзакцию, подтверждает, что он готов сделать это.

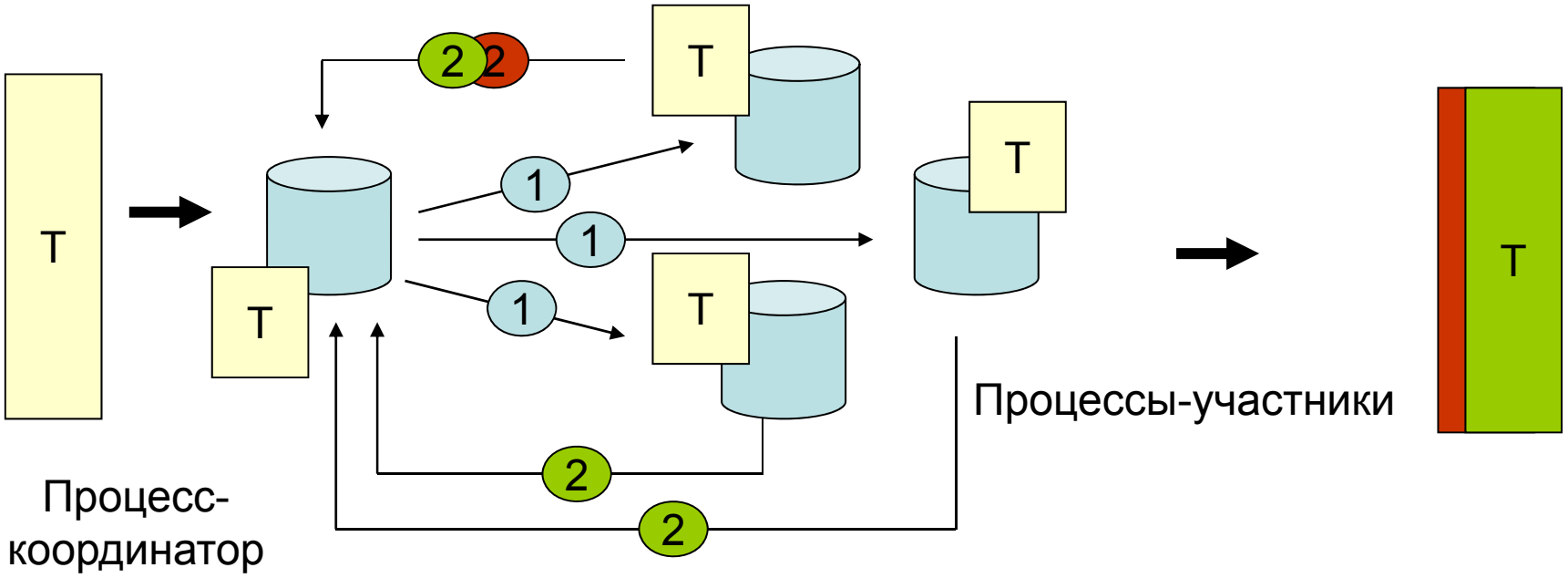
Если все узлы согласны зафиксировать транзакцию, то все относящиеся к ней действия реально выполняются; если один из узлов отказывается зафиксировать свою часть транзакции, то и все остальные узлы вынуждены прервать данную транзакцию.

2PC опирается на следующие фундаментальные правила:

1. Если хотя бы один узел отказывается зафиксировать транзакцию (голосует за ее прерывание), то распределенная транзакция прерывается на всех участвующих в ней узлах.
2. Если все узлы голосуют за фиксацию транзакции, то она фиксируется на всех участвующих в ней узлах.

Распределенные и параллельные СУБД

В простейшем варианте работа 2PC выглядит следующим образом:



1 - «приготовиться»

2 - «голосую за фиксацию »

2 - «голосую за прерывание»

Распределенные и параллельные СУБД

Существенная черта 2PC - блокирующий характер протокола. Отказы могут происходить, в частности, на стадии фиксации транзакции. Как уже отмечалось, единственный способ выявления сбоев - это ожидание сообщения в течение определенного промежутка времени. Если время ожидания исчерпано, то ждущий процесс (координатор или участник) следует ***протоколу терминирования***, который предписывает для такой ситуации способ обработки транзакции, находящейся в стадии завершения.

Распределенные и параллельные СУБД

Неблокирующий протокол фиксации - это такой протокол, терминирующая часть которого при любых обстоятельствах способна определить, что делать с транзакцией в случае сбоя. При использовании 2PC, если в период сбора голосов сбой происходит и на координирующем узле, и на одном из участников, оставшиеся узлы не способны решить между собой судьбу транзакции и вынуждены оставаться в заблокированном состоянии, пока не восстановится либо координатор, либо отказавший участник. В этот период невозможно снять установленные транзакцией блокировки.

Распределенные и параллельные СУБД

Обратной стороной терминирования является ***восстановление***. Протокол восстановления должен принять решение о том, как следует поступить с транзакцией, которую координировал узел. Возможны следующие случаи:

1. Сбой произошел до начала процедуры фиксации. Узел может начать процесс фиксации после восстановления.
2. Координатор отказал, находясь в состоянии готовности. После восстановления координатор перезапускает процедуру фиксации и рассылает команду "приготовиться". Если участники уже завершили транзакцию - сообщение об этом координатору. Если они были заблокированы - вновь отослать координатору свои голоса и возобновить процесс фиксации.
3. Сбой произошел после того, как координатор сообщил участникам о глобальном решении и завершил транзакцию. В этом случае ничего делать не нужно.

БАЗЫ ДАННЫХ
часть II

Параллельные архитектуры баз данных

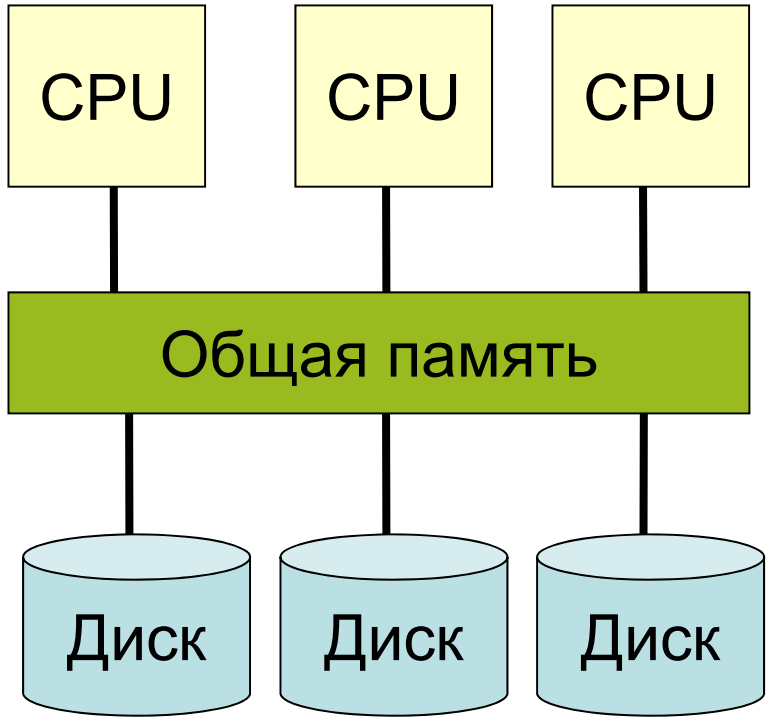
Параллельные архитектуры БД

Основные параллельные архитектуры

Параллельные архитектуры БД

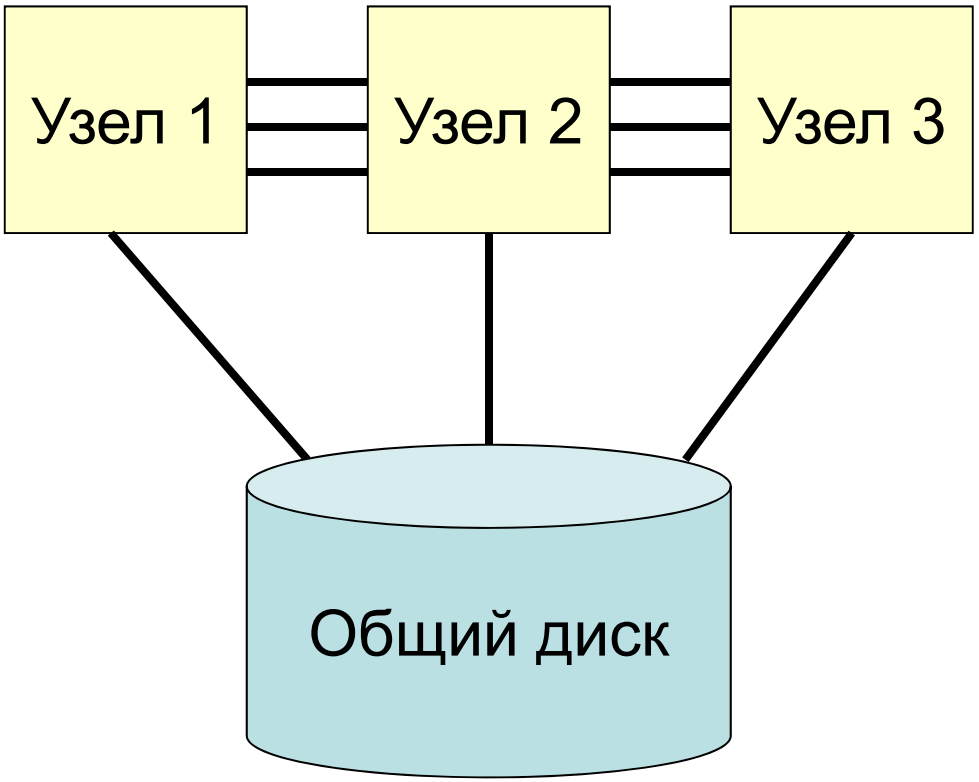
Фактически определились три архитектурных направления:

1. Симметричные многопроцессорные системы (SMP) - наиболее часто используемая форма сильносвязанных многопроцессорных систем, т.е. систем, разделяющих единую оперативную память и наиболее часто - дисковую подсистему.



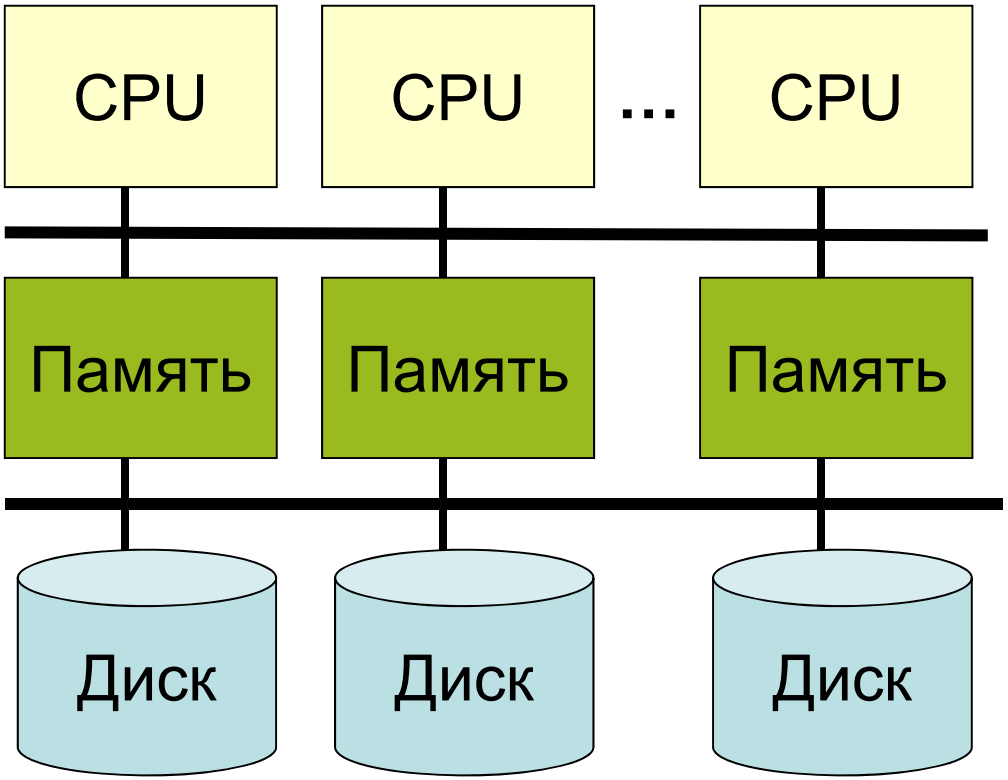
Параллельные архитектуры БД

2. Слабосвязанные многопроцессорные системы - совокупность самостоятельных компьютеров, объединенных в единую систему быстродействующей сетью и, возможно, имеющих общую дисковую подсистему, как, например, кластерные инсталляции;



Параллельные архитектуры БД

3. Системы с массовым параллелизмом (MPP) - системы с сотнями и даже тысячами процессоров, детали их реализации могут значительно различаться.



Параллельные архитектуры БД

Наиболее оптимальными с точки зрения стоимости и прозрачности наращивания можно считать симметричные многопроцессорные платформы (SMP). Добавление процессоров в них обходится относительно дешево, и при использовании соответствующих программ не требует изменения программного обеспечения или принципов администрирования, причем, начиная уже с однопроцессорных систем. Для более дорогостоящих и ответственных систем необходимый уровень резервирования может быть достигнут с помощью кластеров, в т.ч. состоящих из SMP-систем.

Параллельные архитектуры БД

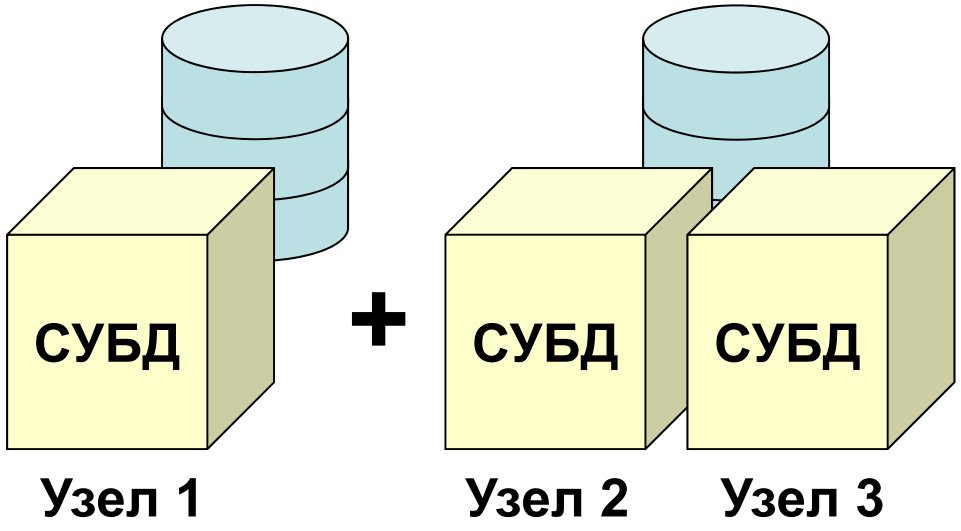
Можно выделить четыре группы требований, определяющих с технической точки зрения потребительские качества современной СУБД:

1. **масштабируемость;**
2. **производительность;**
3. **возможность смешанной загрузки разными типами задач;**
4. **обеспечение постоянной доступности данных.**

Параллельные архитектуры БД

Масштабируемость - такое свойство вычислительной системы, которое обеспечивает предсказуемый рост системных характеристик при добавлении к ней вычислительных ресурсов. В случае сервера СУБД можно рассматривать два способа масштабирования - вертикальный и горизонтальный.

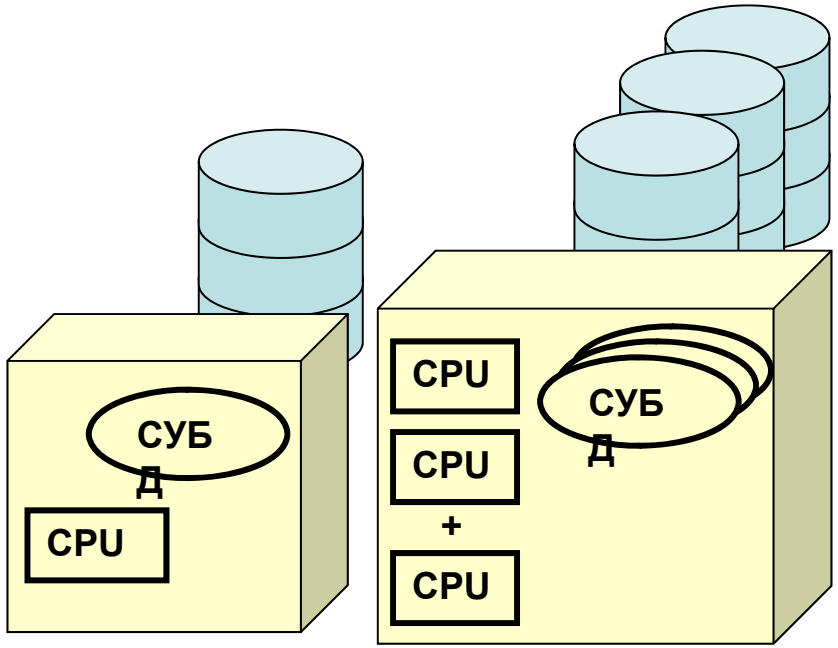
При горизонтальном подходе увеличивается число серверов СУБД, возможно, взаимодействующих друг с другом в прозрачном режиме, разделяя таким образом общую загрузку системы.



Параллельные архитектуры БД

Вертикальное масштабирование подразумевает увеличение мощности отдельного сервера СУБД. Хорошим примером может служить увеличение числа процессоров в симметричных многопроцессорных (SMP) платформах.

При этом программное обеспечение сервера не должно изменяться, например, требовать дополнительных модулей, т.к. это увеличило бы сложность администрирования и ухудшило предсказуемость системы.



Параллельные архитектуры БД

Для вертикального масштабирования все чаще используются симметричные многопроцессорные системы (SMP), поскольку в этом случае не требуется смены платформы, т.е. операционной системы, аппаратного обеспечения, а также навыков администрирования.

При оценке сервера СУБД на базе SMP платформы стоит обратить внимание на две основные характеристики расширямости архитектуры: адекватность и прозрачность.

Параллельные архитектуры БД

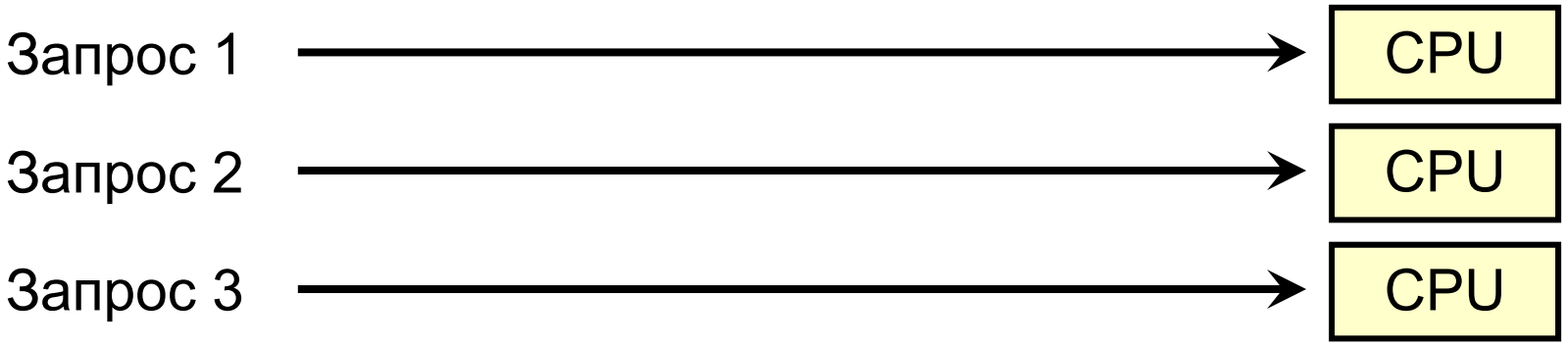
Свойство адекватности требует, чтобы архитектура сервера равно поддерживала один или десять процессоров без переинсталляции или существенных изменений в конфигурации, а также дополнительных программных модулей.

Приложение не должно учитывать подробности реализации аппаратной архитектуры - способы манипулирования данными и программный интерфейс доступа к базе данных обязаны оставаться одинаковыми и в равной степени эффективными (прозрачность).

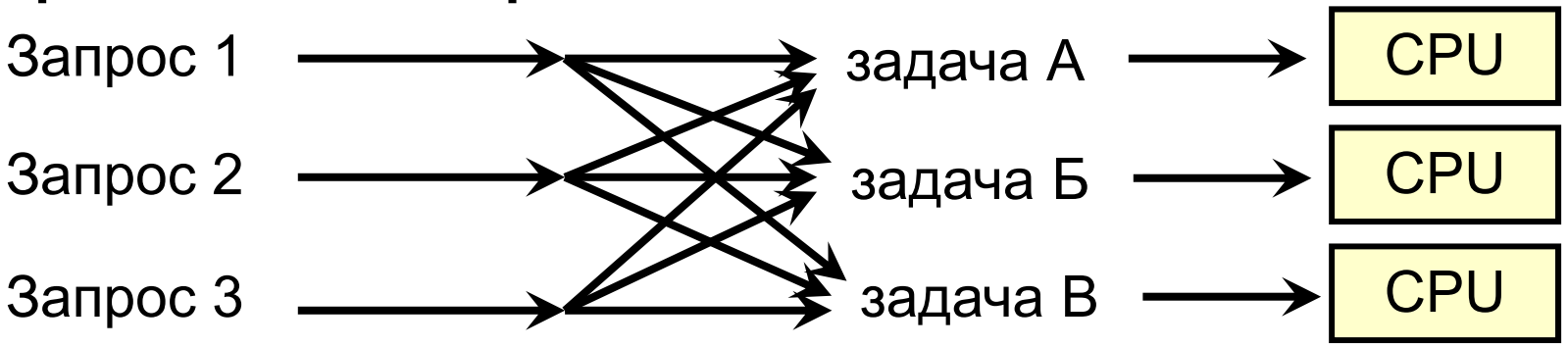
Параллельные архитектуры БД

Запросы могут обрабатываться последовательно несколькими задачами или разделяться на подзадачи, которые в свою очередь могут быть выполнены параллельно.

Параллельные транзакции



Параллельные запросы



Параллельные архитектуры БД

Среди многих факторов, влияющих на производительность СУБД, рассмотрим два, имеющие прямое отношение к нынешним параллельным вычислительным платформам:

1. поддержка параллелизма;
2. реализация многопотоковой архитектуры

Параллельные архитектуры БД

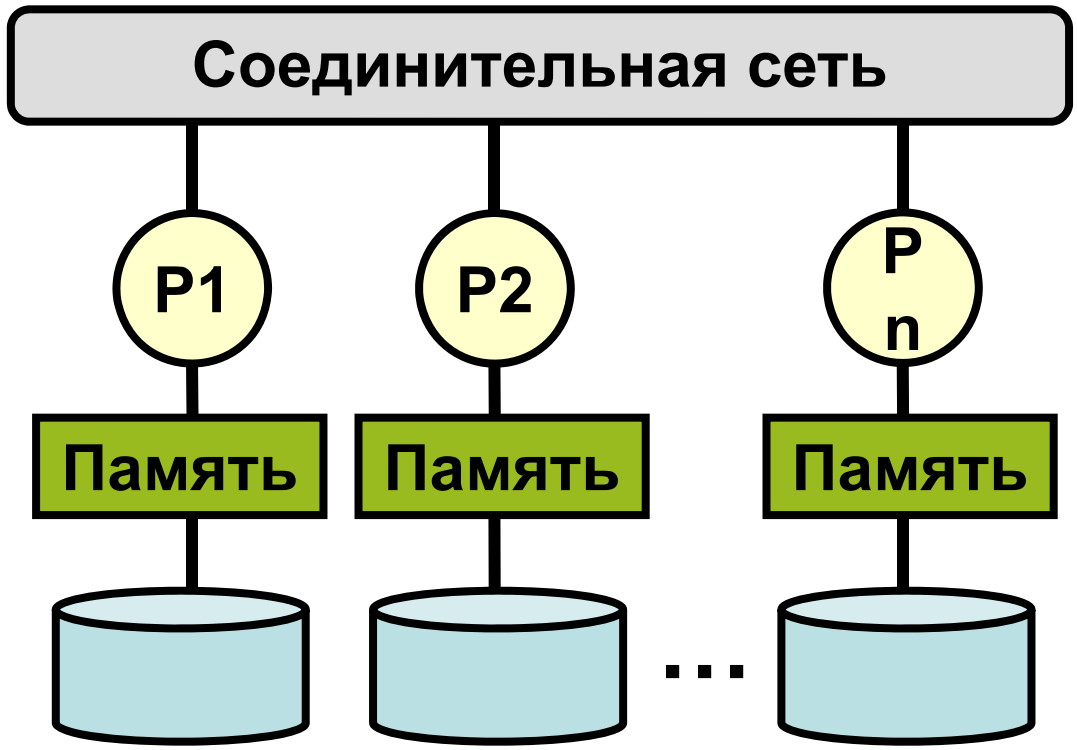
Одним из способов достижения более высокой производительности является использование алгоритмов распараллеливания заданий.

В СУБД существует три области применения таких алгоритмов:

1. параллельный ввод/вывод;
2. параллельные средства и утилиты администрирования;
3. параллельная обработка запросов к базе данных.

Параллельные архитектуры БД

Распараллеливание ввода/вывода в сочетании с оптимальным планированием заданий позволяет осуществить крайне эффективный одновременный доступ к фрагментированным таблицам и индексам, расположенным на нескольких физическим дисках.

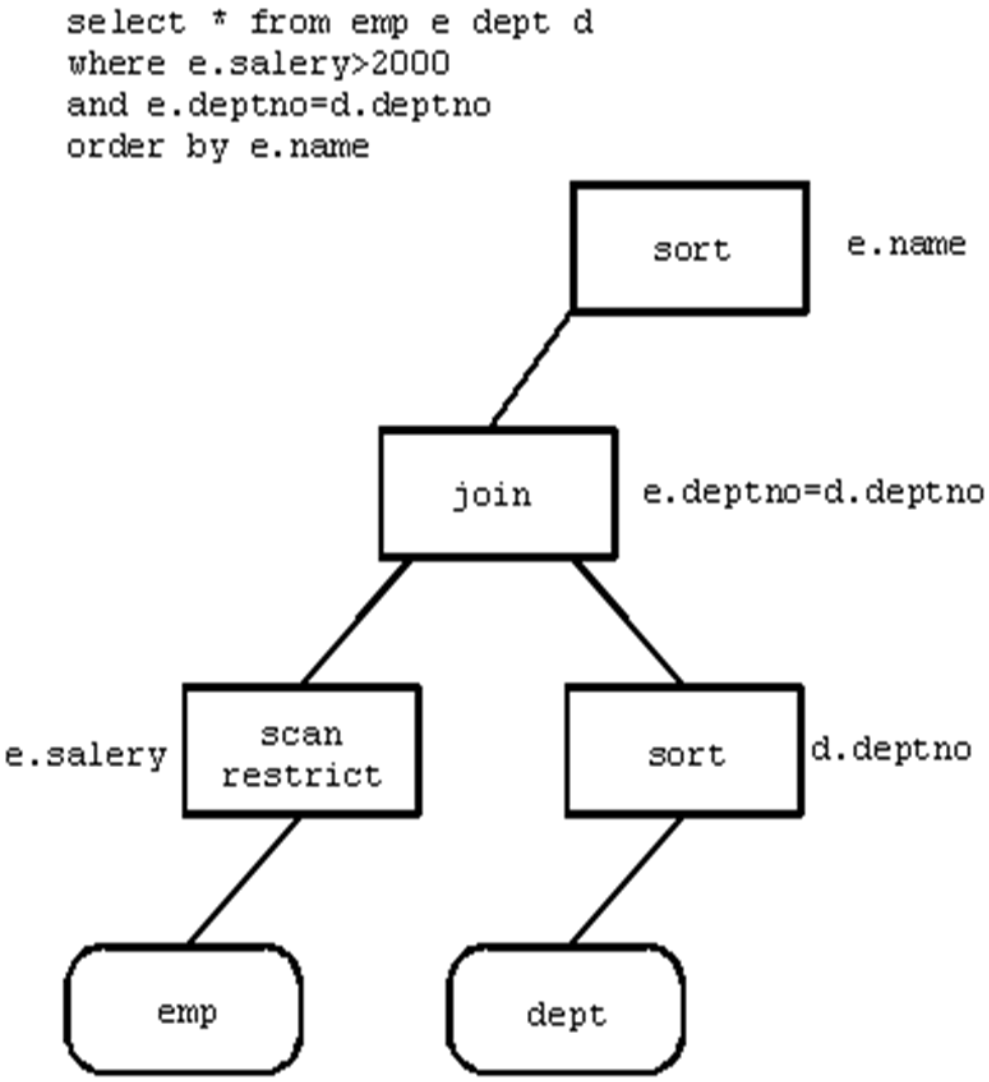


Параллельные архитектуры БД

Выполнение административных утилит (реорганизация данных, в т.ч. сортировки, построение индексов, загрузка и выгрузка данных, архивирование и восстановление баз данных) также должно быть полностью параллельным, включая распараллеливание операций ввода/вывода, хотя на практике это требует доработки лишь небольшого числа механизмов.

Параллельные архитектуры БД

Теоретическим обоснованием возможности распараллеливания запросов в реляционной СУБД являются свойство реляционного замыкания.
Параллелизм в обработке запросов подобен идее "разделяй и властвуй" или коллективному выполнению задания.



Параллельные архитектуры БД

Достижение этой цели требует, чтобы декомпозиция и параллельное выполнение запроса были осуществимы независимо от средств межзадачного взаимодействия, принятых в конкретной вычислительной системе.

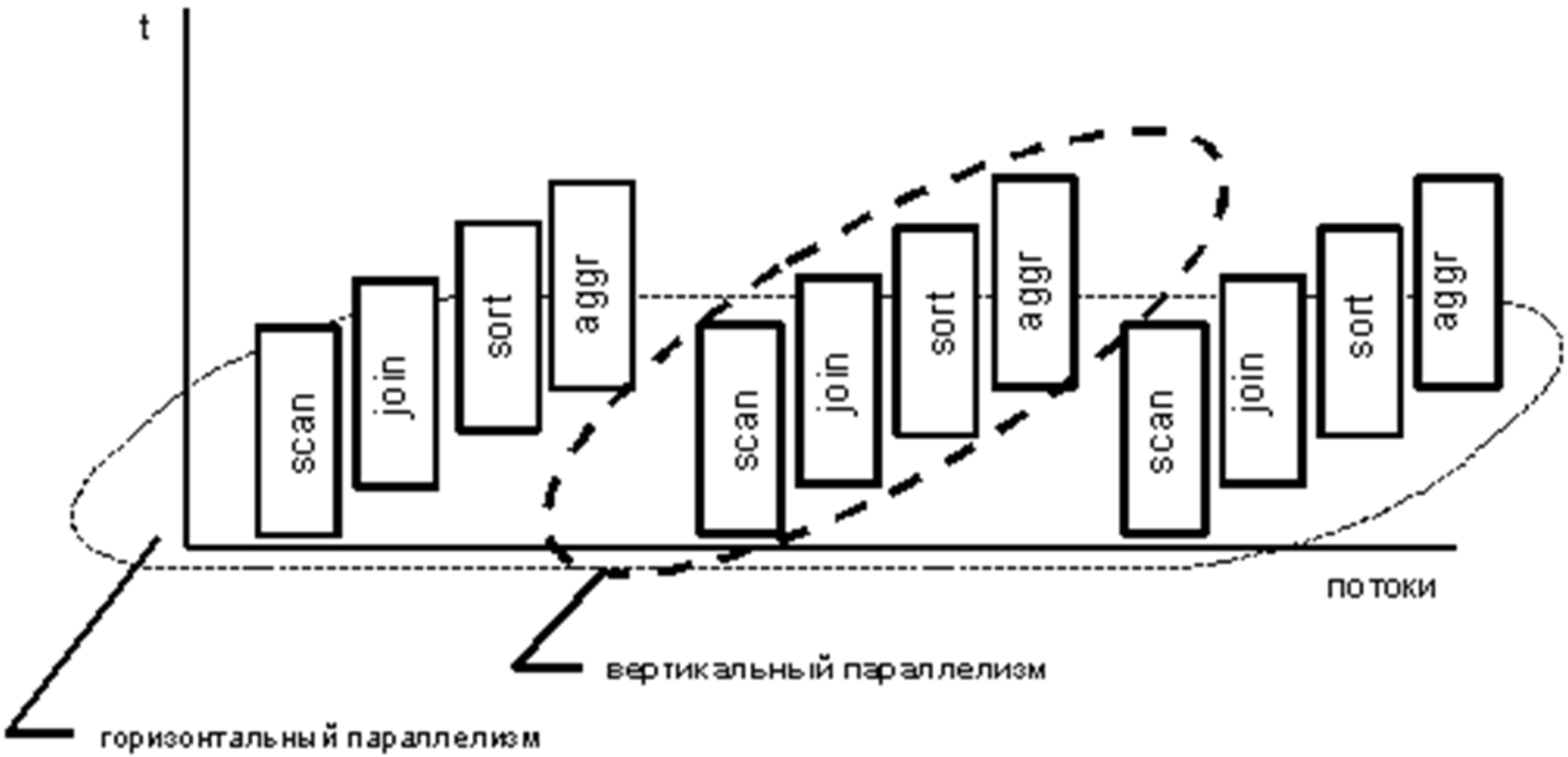
Создание множества тиражируемых копий одной атомарной операции, одновременно выполняющих одну и ту же задачу, можно назвать горизонтальным параллелизмом.

Параллельные архитектуры БД

Дополнительная степень параллелизма - вертикальный параллелизм - может быть достигнута при параллельном запуске нескольких, в общем случае различных атомарных операций, которые обычно должны были бы выполняться последовательно.

Такой подход, во многом сходный с конвейерной обработкой в современных процессорах, носит название "потока данных" (data flow) или "управления по требованию" (demand driven).

Параллельные архитектуры БД

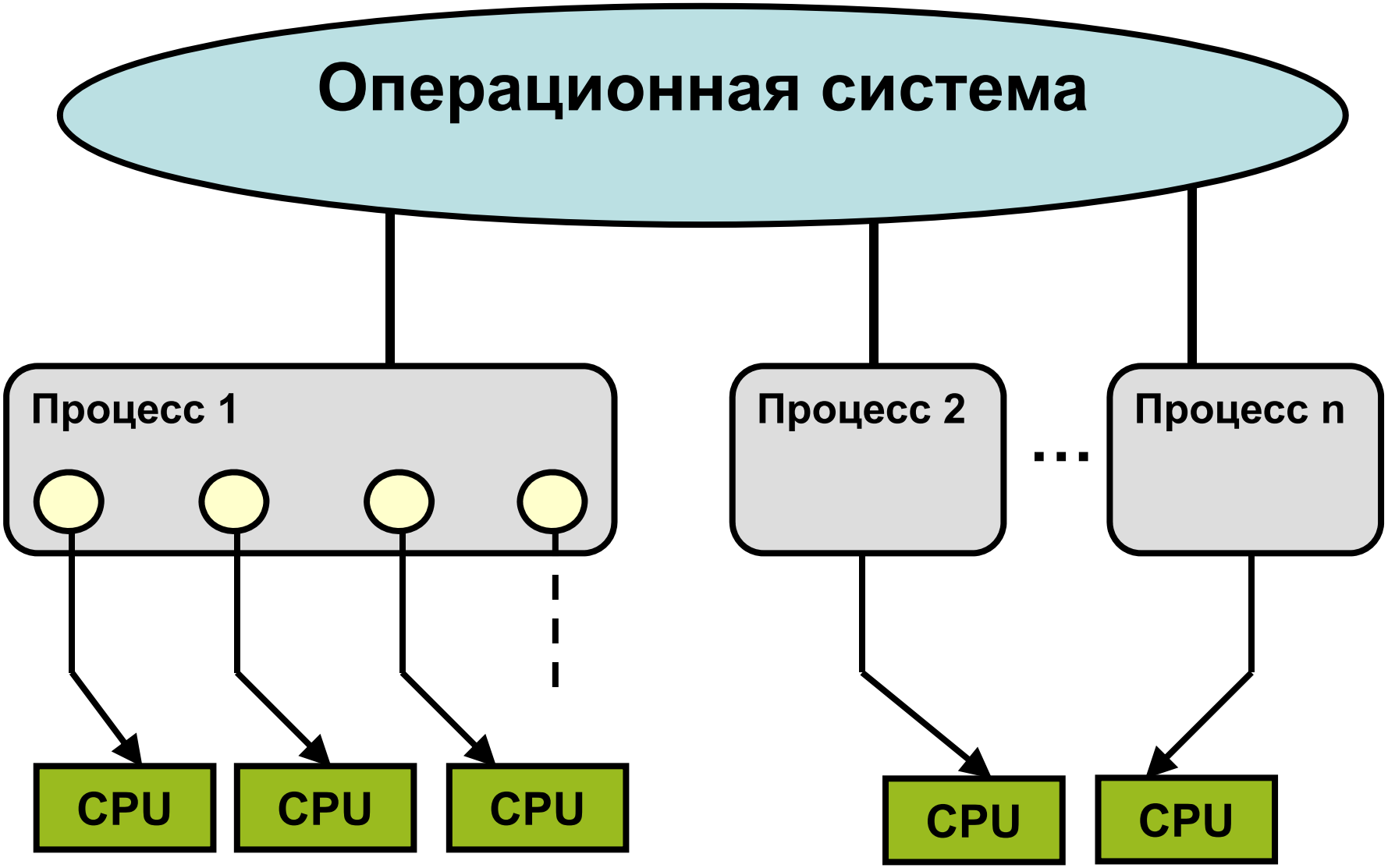


Параллельные архитектуры БД

Способ организации одновременной обработки множества запросов внутри сервера баз данных при минимальных накладных расходах на переключение контекста и максимальном разделении вычислительных ресурсов между ними называется истинно многопотоковой архитектурой .

Поток (thread) - это фрагмент контекста одного процесса, включающий только те данные (стек рабочих переменных и текущего состояния), которые необходимы для реализации выполняемых потоком функций.

Параллельные архитектуры БД



Параллельные архитектуры БД

Эволюция в области информационных систем все отчетливее направлена в сторону объединения трех видов задач: оперативной обработки транзакций (OLTP), поддержки принятия решений (DSS) и пакетной обработки (batch processing).

Одновременное исполнение задач смешанного характера, разделяющих общие вычислительные ресурсы и базы данных, называется "оперативной сложной обработкой данных" (OLCP - On Line Complex Processing).

Параллельные архитектуры БД

Основные факторы реализации универсальной системы со смешанной загрузкой можно определить как:

1. оптимизация запросов;
2. эффективное управление ресурсами;
3. параллельная обработка запросов.

Параллельные архитектуры БД

Оптимальное управлении ресурсами требует поддержки прозрачности доступа к ресурсам в целом и эффективного использования каждого ресурса в отдельности.

Один из наиболее важных ресурсов - оперативная память. Оптимальное управление распределением памяти настолько же критично для производительности системы, как и уменьшение загрузки ОС.

Параллельные архитектуры БД

Приложения СУБД используют разделяемую память для сохранения контекста подключения, а также кэширования запросов и результатов.

Разделяемая память

Кэш-область

Системная область, область коммуникаций

Динамическая область обслуживания потоков и обработки запросов

Параллельные архитектуры БД

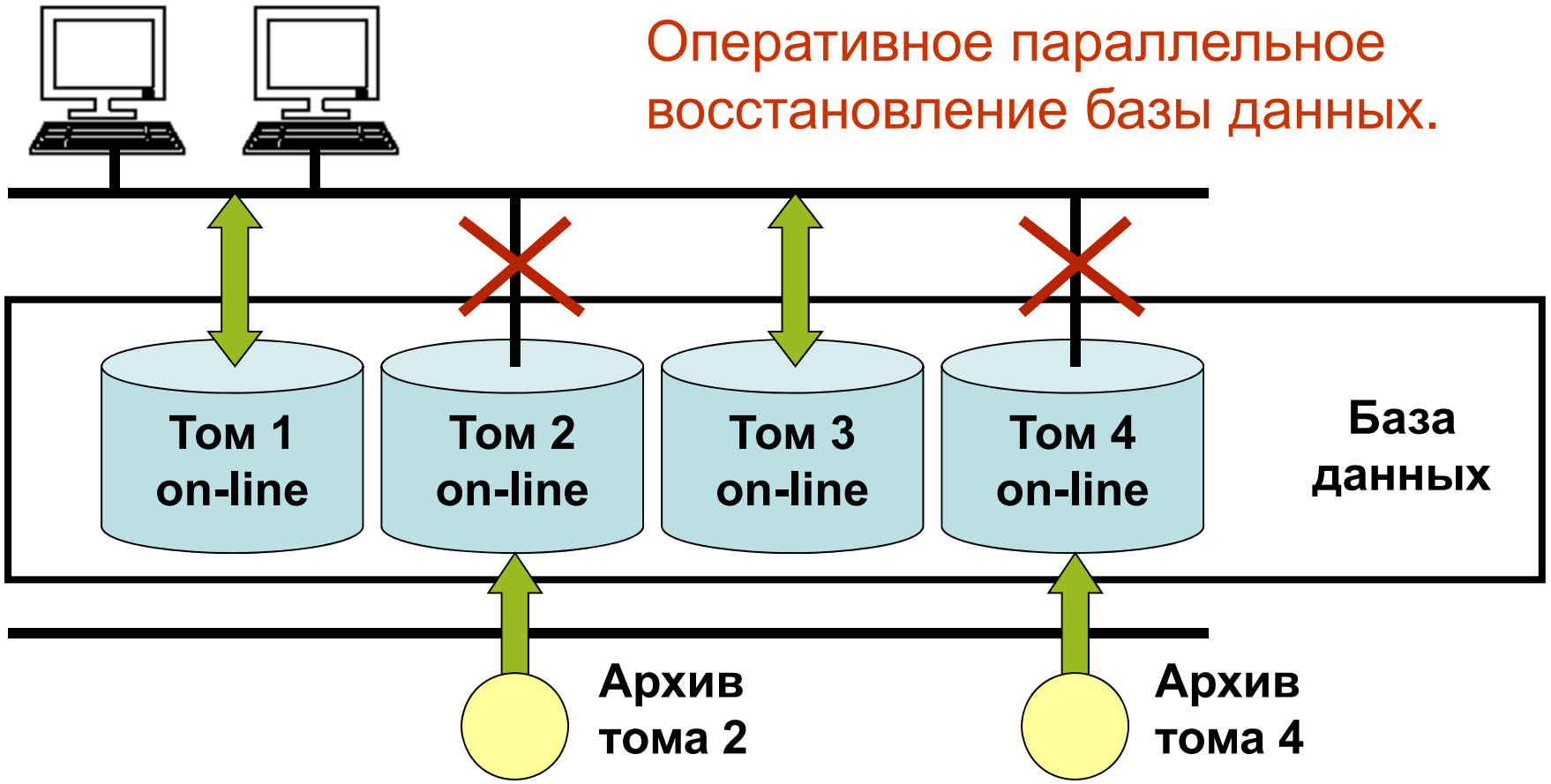
Основные моменты, обеспечивающие постоянную готовность современных СУБД можно определить, как:

1. оперативное администрирование;
2. функциональная насыщенность СУБД.

Параллельные архитектуры БД

Утилиты администрирования в идеале призваны поддерживать бесперебойное функцинирование СУБД, что подразумевает сведение к минимуму планируемых или сбойных простоев системы.

Оперативное параллельное восстановление базы данных.



Параллельные архитектуры БД

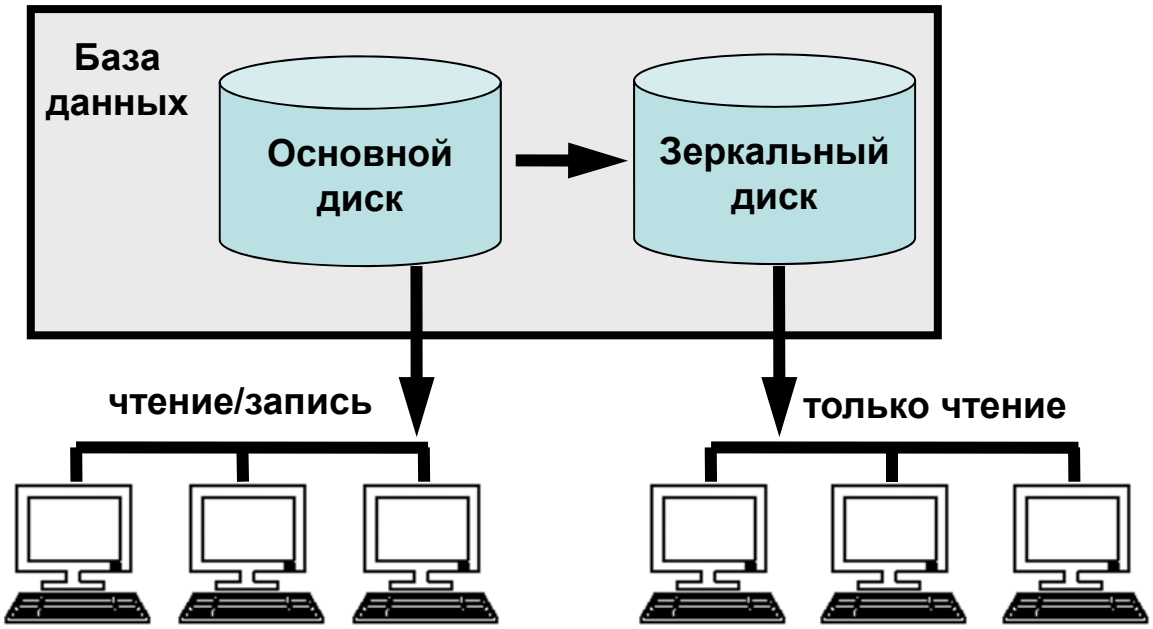
Общим термином "функциональная насыщенность" можно определить множество механизмов, используемых серверами баз данных (а) для уменьшения последствий системных сбоев и (б) для повышения прозрачности доступа к дублированным данным при нормальном функционировании.

Теоретически отказоустойчивая СУБД обязана обеспечивать доступ к данным по чтению и записи независимо от обстоятельств, будь то сбой аппаратной платформы или какого-то компонента СУБД.

Параллельные архитектуры БД

Системы, обеспечивающие непрерывный доступ к данным (fault tolerant) или почти непрерывный (high availability), обычно опираются на различные формы избыточности. Как правило, это системы дублирования аппаратного обеспечения и контролируемой избыточности данных.

Один из вариантов - зеркалирование дисков:



БАЗЫ ДАННЫХ
часть II

Объектно ориентированные базы данных

ООБД

В компьютерных технологиях сегодня отчетливо просматривается стремление с минимальными потерями перенести в виртуальный мир объекты мира реального. Объектно-ориентированная СУБД — именно то средство, которое обеспечивает запись объектов в базу данных «как есть». Данное обстоятельство стало решающим аргументом в пользу выбора ООСУБД для переноса семантики объектов и процессов реального мира в сферу информационных систем.

ООБД

Использование объектного подхода к проектированию систем поднимает роль ООБД как средства для наиболее естественного хранения и манипулирования создаваемыми объектами.

ООСУБД находят широкое применение в Internet — текст, картинки, видео и звук, из которых составляется Web-страница, хранятся в ООСУБД как набор объектов, подготовленный к передаче программе-клиенту, что позволяет добиться быстрой реакции сервера на запрос.

ООБД

Все большую популярность получают активные Web-серверы, на лету генерирующие страницы, используя язык Java. Практически все ведущие разработчики ООСУБД выбрали Java одним из основных для себя языков программирования.

ООБД

Единого мнения по поводу того, как конкретно следует организовывать ООСУБД, нет. Тем не менее, ряд свойств продекларированы в «Манифесте систем объектно-ориентированных баз данных», а впоследствии закреплены в документах ODMG, организации, объединяющей ведущих производителей ООСУБД.

Используемая далее терминология отражает требования стандарта ODMG 2.0.

ООБД

Модель данных

Объектная модель данных характеризуется следующими свойствами:

- Базовыми примитивами являются объекты и литералы. Каждый объект имеет уникальный идентификатор, литерал не имеет идентификатора.
- Объекты и литералы различаются по типу. Все элементы одного типа имеют одинаковый диапазон изменения состояния (множество свойств) и одинаковое поведение (множество определенных операций). Объект, на который можно установить ссылку, называется экземпляром; он хранит определенный набор данных.

Модель данных (продолжение)

- Состояние объекта определяется набором значений, реализуемых множеством свойств. Этими свойствами могут быть атрибуты объекта или связи между объектом и одним или несколькими другими объектами.
- Поведение объекта определяется набором операций, которые могут быть выполнены над объектом или самим объектом. Операции могут иметь список входных и выходных параметров строго определенного типа.
- База данных хранит объекты, позволяя совместно использовать их различным пользователям и приложениям. База данных основана на схеме данных, определяемой языком определения данных, и содержит экземпляры типов, определенных схемой.

Модель данных (продолжение)

Каждый тип имеет внешнюю спецификацию и одну или несколько реализаций. Спецификация определяет внешние характеристики типа: пользователю для работы с объектом предоставляется набор операций и набор атрибутов объекта, при помощи которых можно работать с реальными экземплярами. Реализация определяет внутреннее содержание объектов, например операции.

Тип также является объектом. Поддерживается иерархия супертипов и подтипов, реализуя стандартный механизм объектно-ориентированного программирования — наследование.

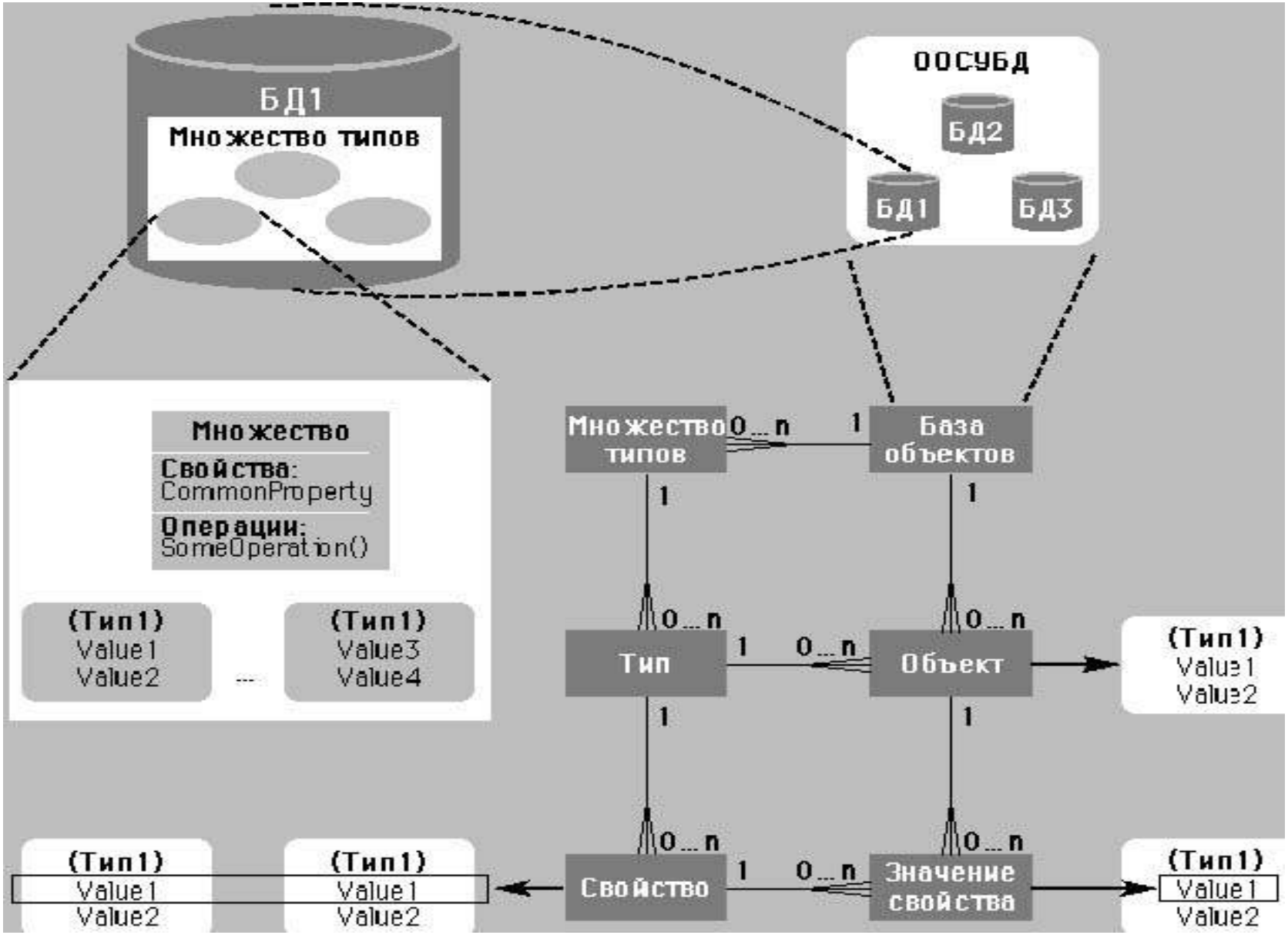
ООБД


Тип
Объект
Объект
...
Объект
Объект
Поведение
Операции
Состояние
Свойства
Атрибуты
Связи

Модель данных (продолжение)

ООСУБД обслуживает множество баз данных, каждая из которых содержит определенное множество типов. В базах данных могут содержаться объекты соответствующего типа из этого множества. Тип имеет набор свойств, а объект характеризуется состоянием в зависимости от значения каждого свойства. Операции, определяющие поведение типа, едины для всех объектов одного типа. Свойство едино для всего типа, а все объекты типа также имеют одинаковый набор свойств. Значение свойства относится к конкретному объекту.

ООБД


БД1
Множество типов
ООСУБД
БД2
БД1
БД3
Множество
Свойства:
CommonProperty
Операции:
SomeOperation()
(Тип1)
Value1
Value2
...
(Тип1)
Value3
Value4
Множество типов
0...n
1
База объектов
1
1
0...n
0...n
Тип
1
0...n
Объект
(Тип1)
Value1
Value2
1
1
0...n
0...n
(Тип1)
Value1
Value2
(Тип1)
Value1
Value2
Свойство
1
0...n
Значение свойства
(Тип1)
Value1
Value2

Идентификатор объекта

Каждый объект в базе данных уникален. Существует несколько подходов для идентификации объекта. Самый простой — присвоить ему уникальный номер (OID — object identificator) в базе и никогда больше не повторять этот номер, даже если предыдущий объект с таким номером уже удален.

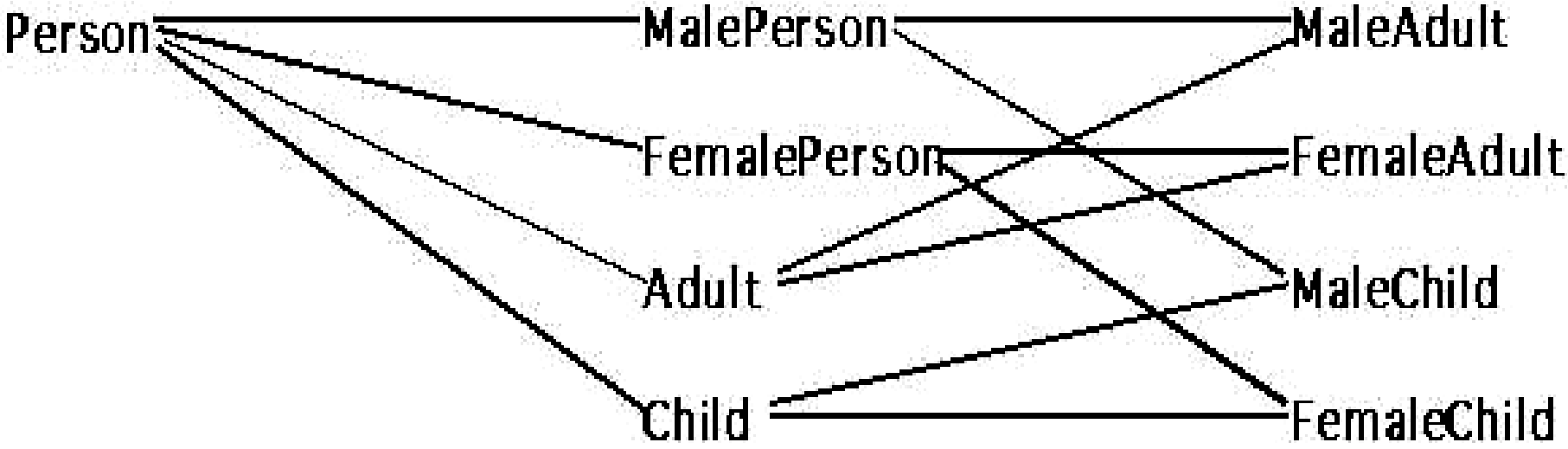
Недостаток – невозможно перенести объекты в другую базу без потери связности между ними. Решение этой проблемы заключается в использовании составного идентификатора. Например, xxxxxxxx:yyyyyyyy, где xxxxxxxx — идентификатор базы данных, yyyyyyyy — идентификатор объекта в базе.

Новые типы данных

Одним из принципиальных отличий объектных баз данных от реляционных является возможность создания и использования новых типов данных. Концептуально объект характеризуется поведением и состоянием. Определение типа заключается в определении поведения, т.е. операций, которые могут быть выполнены объектом или над состоянием объекта — набором атрибутов определенных типов (атрибут может иметь любой объявленный в базе тип).

Создание нового типа не требует модификации ядра базы и основано на принципах объектно-ориентированного программирования: инкапсуляции, наследовании, перегрузке операций и позднем связывании.

ООБД


Person
MalePerson
FemalePerson
Adult
Child
MaleAdult
FemaleAdult
MaleChild
FemaleChild

ООБД

Функционирование базы основано на схеме данных. Любой тип является объектом, следовательно, схемы данных являются уровнем интерпретации специфических служебных объектов, использующих свойства этих объектов как схему для создания новых типов.

Схема данных может быть как первичной для создания классов, так и вторичной, выделяемой из созданных на языке программирования (скажем, на C++) классов и загружаемой в базу. Язык ODL разработан ODMG как универсальный язык описания объектов и не претендует на то, чтобы называться полноценным языком программирования. Для целей разработки предусмотрены элементы расширения классических объектных языков C++, Smalltalk, Java, позволяющих описать структуру объектов, их связи и типы связей.

Оптимизация ядра СУБД

Ядро ООСУБД оптимизировано для операций с объектами. Естественными операциями для него являются кэширование объектов, ведение версий объектов, разделение прав доступа к конкретным объектам. Ядро объектно-реляционной СУБД остается реляционным, а «объектность» реализуется в виде специальной надстройки. Как следствие, ООСУБД свойственно более высокое быстродействие на операциях, требующих доступа и получения данных, упакованных в объекты, по сравнению с реляционными СУБД, для которых необходимость выборки связных данных ведет к выполнению дополнительных внутренних операций.

Язык запросов

Общепризнанны две группы вариантов языков запросов:

Первая объединяет языки, унаследованные от SQL и представляют собой разновидность OQL (Object Query Language), языка, стандартизованного ODMG. Объектно-реляционные СУБД используют различные варианты ограниченных объектных расширений SQL.

Вторая группа языков запросов базируется на XML. Собирательное название языков этой группы — XML QL (или XQL). Они могут применяться в качестве языков запросов в объектных и объектно-реляционных базах данных.

Транзакции

Транзакции в ООСУБД представляют логический блок, гарантирующий атомарность, целостность, изолированность и долговечность.

Атомарность: операции в рамках транзакции либо полностью выполняются, либо полностью не выполняются.
Целостность: транзакция, инициируемая в находящейся во внутренне логически связанном состоянии базе данных, приводит ее в другое логически связанное состояние.
Изолированность: ни один другой пользователь не сможет увидеть изменений, проводимых в рамках транзакции, пока не будет выполнена операция “commit”.
Долговечность: изменения, проведенные в рамках транзакции и сохраненные в базе данных, сохранятся даже в том случае, если произойдет сбой системы.

Транзакции (продолжение)

Короткие транзакции характеризуются малым временем выполнения; они могут существовать только в рамках сеанса работы с ООСУБД. Это наиболее простой вид транзакций, реализованный во всех современных СУБД. Все объекты подлежащие изменениям блокируются, а после принятия транзакции разблокируются, изменения же записываются в базу данных.

Транзакции (продолжение)

Длинные транзакции предназначены для увеличения производительности при групповой работе. При одновременной работе большого числа пользователей предлагается возможность организации персональной базы данных. Пользователи работают со своей базой, а объекты из нее синхронизируются с групповой базой данных. Пользователь, начав длинную транзакцию, тем самым отмечает объекты, с которыми предстоит работать в групповой базе данных (операция «поставить на контроль» — check out). Эти объекты копируются в его персональную базу, а в групповой базе блокируются.

Транзакции (продолжение)

В групповой базе создается объект, содержащий все данные о длинных транзакциях. В случае повреждения групповой базы или физического отключения сервера групповой базы пользователь сможет продолжать работу с объектами в своей персональной базе, а после восстановления групповой базы — синхронизировать объекты. Перед завершением длинной транзакции пользователь должен поместить все измененные объекты обратно в основную базу (операция «зарегистрировать» — check in). После этого объекты копируются в основную базу, а блокировка снимается. В случае аварийного завершения длинной транзакции все изменения будут потеряны.

Транзакции (продолжение)

Вложенные транзакции по принципу функционирования аналогичны коротким. В процессе выполнения одной транзакции формируются другие. Если в текущем сеансе работает один процесс, то создается стек, а если несколько процессов — дерево транзакций.

Блокировки

Короткие блокировки (short lock) предназначены для обеспечения последовательного доступа к данных при многопользовательском режиме работы. Они автоматически выполняются во время выполнения коротких транзакций.

Продолжительные блокировки (persistent lock) обеспечивают блокирование объектов на продолжительное время — часы, дни, недели. Применяются совместно с длинными транзакциями.

Объект может быть заблокирован несколькими способами: с исключением снятия другим процессом (hard lock); с возможностью снятия другим процессом (soft lock); по конкретным операциям.

Перемещение объектов

Миграция объектов: постоянное их перемещение, например в другую базу данных.

Постановка на контроль (check out): копирование объектов в персональную базу данных при выполнении длинной транзакции.

Регистрация объектов (check in): копирование объектов в групповую базу данных из персональной при выполнении длинной транзакции.

Ведение версий

Для всех объектов возможно сохранение всех версий их изменения. Создается граф происхождения версий, поддерживающий ряд свойств:

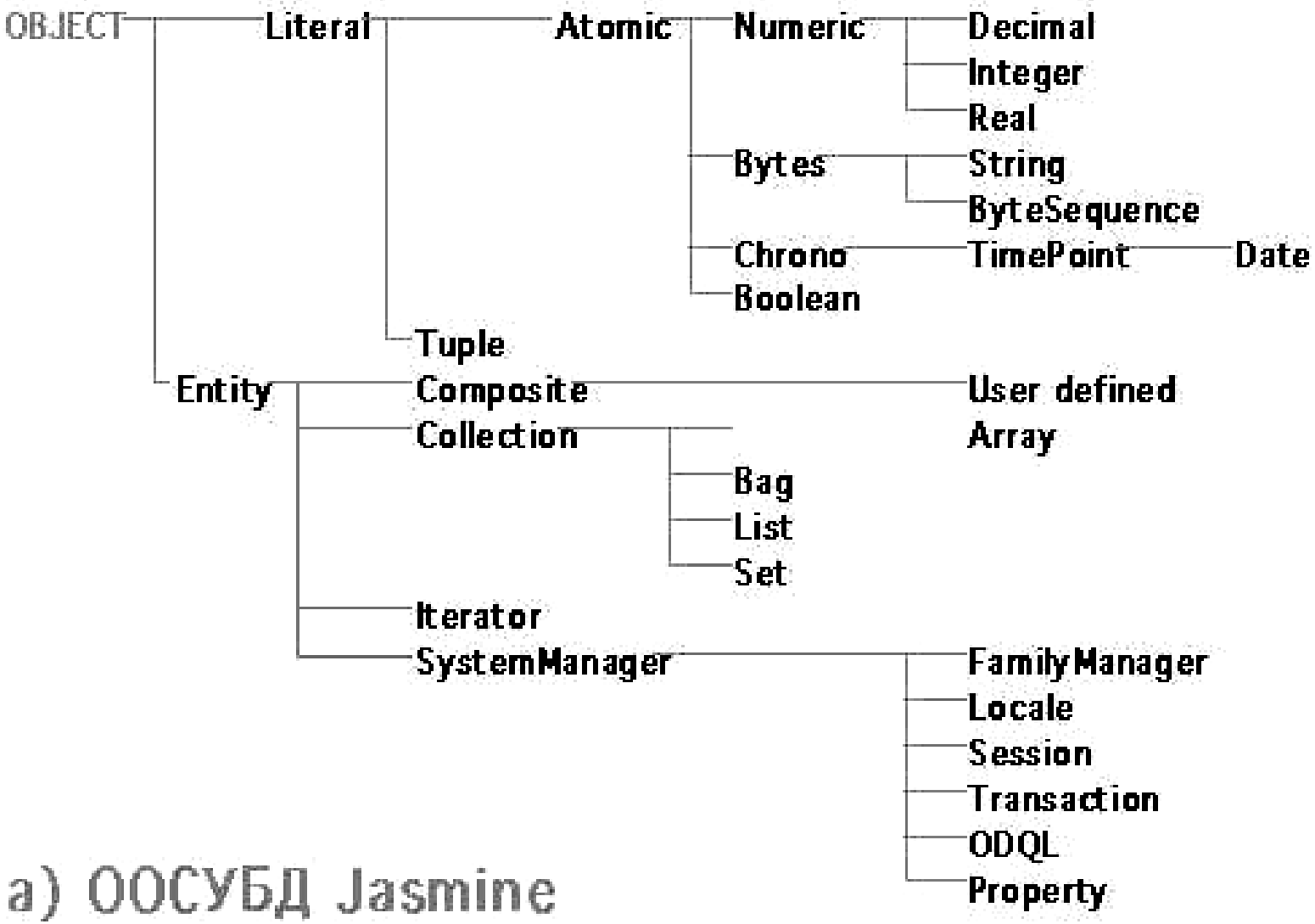
1. **Доступ к любой ранее сохраненной версии.** Благодаря этому свойству возможно извлечение из базы данных, например, случайно удаленных данных.
2. **Установка версии по умолчанию.** Любая порожденная версия создаст ветвь от установленной текущей, не последней версии.
3. **Удаление версии.** Существует необходимость удаления некоторых версий (например, промежуточных или случайно порожденных) из графа происхождения версий.

Физические хранилища

Один из ключевых моментов функционирования любой СУБД — хранилище данных. Выделяют три типа хранилищ:

1. **системное хранилище**, использующееся для хранения системы классов, создается на этапе формирования базы данных и содержит информацию о классах, о наличии и месторасположении пользовательских хранилищ;
2. **пользовательское хранилище** для хранения пользовательских объектов;
3. **служебное хранилище**, содержащее временную информацию, например сведения о заблокированных объектах, об активных транзакциях, различного вида списки запросов пользователей и т.д.

Иерархия типов



а) ООСУБД Jasmine

БАЗЫ ДАННЫХ
часть II

Многомерные базы данных

Многомерные БД

Если целью является именно анализ данных, а не выполнение транзакций, используется многомерная модель данных. Технология многомерных баз данных — ключевой фактор интерактивного анализа больших массивов данных с целью поддержки принятия решения. Подобные базы данных трактуют данные как многомерные кубы, что очень удобно именно для их анализа.

Многомерные БД

Многомерные модели рассматривают данные либо как факты с соответствующими численными параметрами, либо как текстовые измерения, которые характеризуют эти факты.

Многомерные БД

Многомерные модели данных имеют три важных области применения, связанных с проблематикой анализа данных:

1. Хранилища данных интегрируют для анализа информации из нескольких источников.
2. Системы оперативной аналитической обработки (online analytical processing — OLAP) позволяют оперативно получить ответы на запросы, охватывающие большие объемы данных в поисках общих тенденций.
3. Приложения добычи данных служат для выявления знаний за счет полуавтоматического поиска ранее неизвестных шаблонов и связей в базах данных.

Многомерные БД

Электронные таблицы не подходят для управления и хранения многомерных данных, поскольку они слишком жестко связывают данные с их внешним видом, не отделяя структурную информацию от желаемого представления информации.

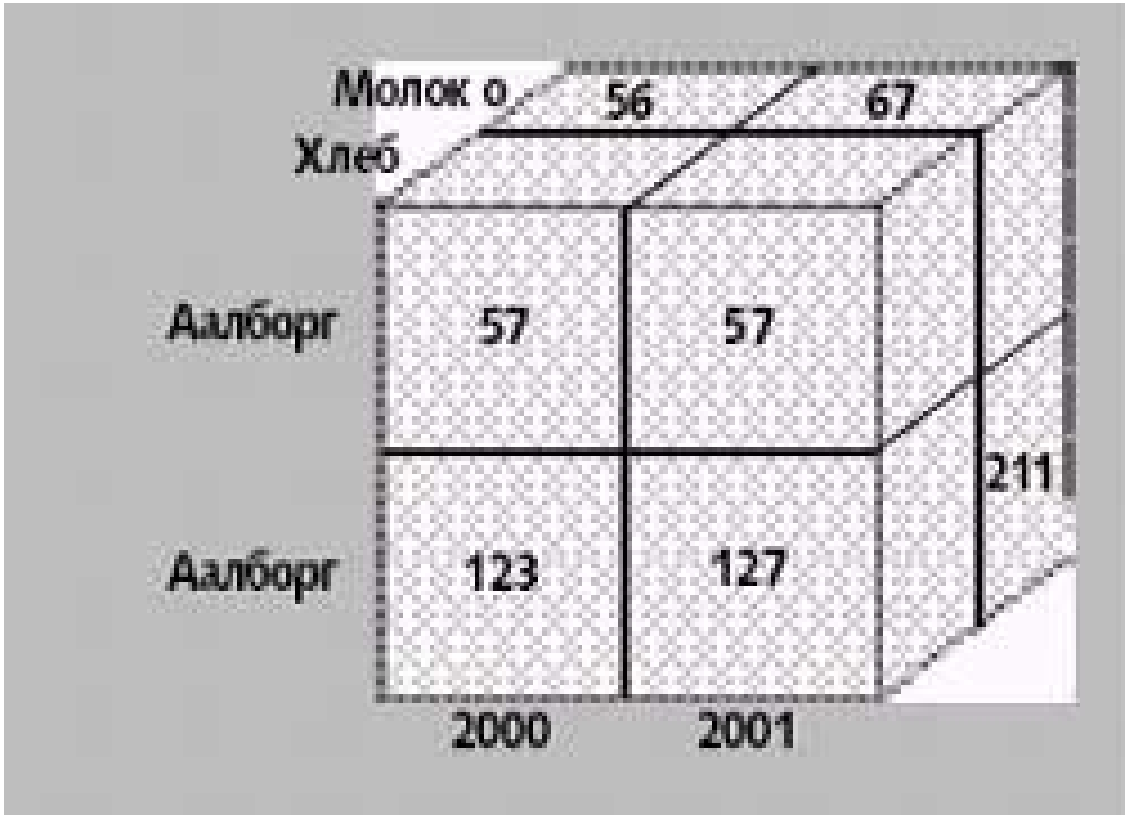
Использование баз данных, поддерживающих SQL, не позволяет сформулировать многие вычисления, такие как совокупные показатели (объем продаж за год к текущему моменту), сочетание итоговых и промежуточных результатов, ранжирование.

Многомерные БД

Многомерные базы данных рассматривают данные как кубы, которые являются обобщением электронных таблиц на любое число измерений. Кубы поддерживают иерархию измерений и формул без дублирования их определений. Набор соответствующих кубов составляет многомерную базу данных (или хранилище данных). Комбинации значений измерений определяют ячейки куба.

Многомерные БД

Пример куба, содержащего данные о продажах. В этом случае куб обобщает электронную таблицу, добавляя к ней третье измерение — время.



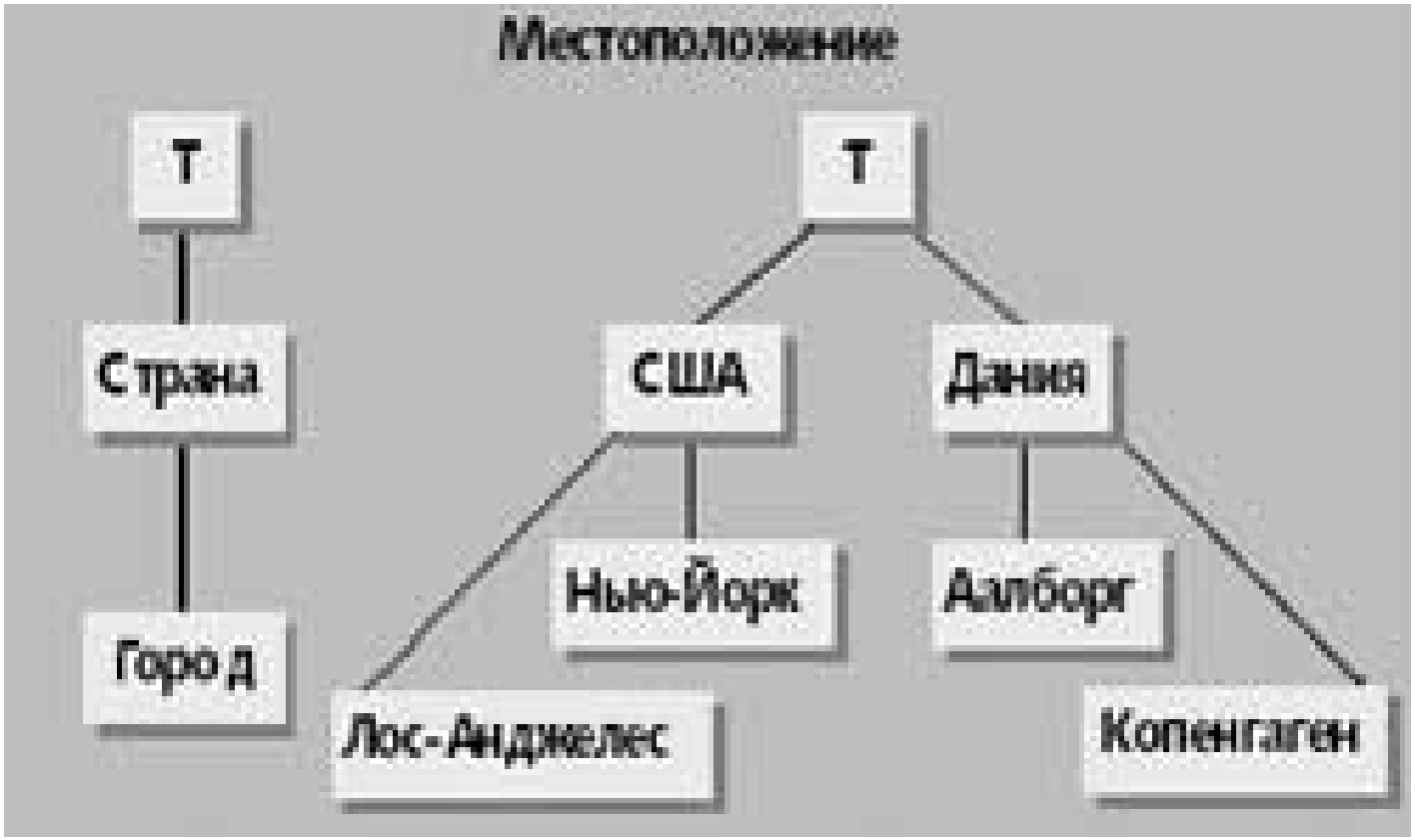
Многомерные БД

Измерения — ключевая концепция многомерных баз данных. Многомерное моделирование предусматривает использование измерений для предоставления максимально возможного контекста для фактов.

Измерения используются для выбора и агрегирования данных на требуемом уровне детализации. Измерения организуются в иерархию, состоящую из нескольких уровней, каждый из которых представляет уровень детализации, требуемый для соответствующего анализа.

Многомерные БД

Пример схемы измерений местоположения. Каждое значение размерности является частью значения Т.



Многомерные БД

В отличие от линейных пространств, с которыми имеет дело алгебра матриц, многомерные модели, как правило, не предусматривают функций упорядочивания или расстояния для значений измерения. Однако для некоторых измерений, таких как время, упорядоченность значений размерности может использоваться для вычисления совокупной информации. Большинство моделей требуют определения иерархии измерений для формирования сбалансированных деревьев — иерархии должны иметь одинаковую высоту по всем ветвям, а каждое значение не корневого уровня — только одного родителя.

Многомерные БД

Факты представляют субъект — некий шаблон или событие, которые необходимо проанализировать. В большинстве многомерных моделей данных факты однозначно определяются комбинацией значений измерений; факт существует только тогда, когда ячейка для конкретной комбинации значений не пуста.

Каждый факт обладает некоторой гранулярностью, определенной уровнями, из которых создается их комбинация значений измерений.

Многомерные БД

Хранилища данных, как правило, содержат следующие три типа фактов:

1. **События (event),** по крайней мере, на уровне самой большой гранулярности, как правило, моделируют события реального мира, при этом каждый факт представляет определенный экземпляр изучаемого явления.
2. **Мгновенные снимки (snapshot)** моделируют состояние объекта в данный момент времени.
3. **Совокупные мгновенные снимки (cumulative snapshot)** содержат информацию о деятельности организации за определенный отрезок времени.

Многомерные БД

Параметры состоят из двух компонентов:
численная характеристика факта, например, цена или доход от продаж;
формула, обычно простая агрегативная функция, скажем, сумма, которая может объединять несколько значений параметров в одно.
В многомерной базе данных параметры, как правило, представляют свойства факта, который пользователь хочет изучить.

Многомерные БД

1. **Аддитивные параметры** могут содержательным образом комбинироваться в любом измерении.
2. **Полуаддитивные параметры,** которые не могут комбинироваться в одном или нескольких измерениях.
3. **Неаддитивные параметры** не комбинируются в любом измерении, обычно потому, что выбранная формула не позволяет объединить средние значения низкого уровня в среднем значении более высокого уровня.

Многомерные БД

Многомерная база данных предназначена для определенных типов запросов:

1. **Запросы вида slice-and-dice** осуществляют выбор, сокращающий куб.
2. **Запросы вида drill-down и roll-up** — взаимообратные операции, которые используют иерархию измерений и параметры для агрегирования. Обобщение до высших значений соответствует исключению размерности.

Многомерные БД

3. **Запросы вида drill-across** комбинируют кубы, которые имеют одно или несколько общих измерений. С точки зрения реляционной алгебры такая операция выполняет слияние (join).
4. **Запросы вида ranking** возвращает только те ячейки, которые появляются в верхней или нижней части упорядоченного определенным образом списка.
5. **Поворот (rotating)** куба дает пользователям возможность увидеть данные, сгруппированные по другим измерениям.

Многомерные БД

Многомерные базы данных реализуют в двух основных формах:

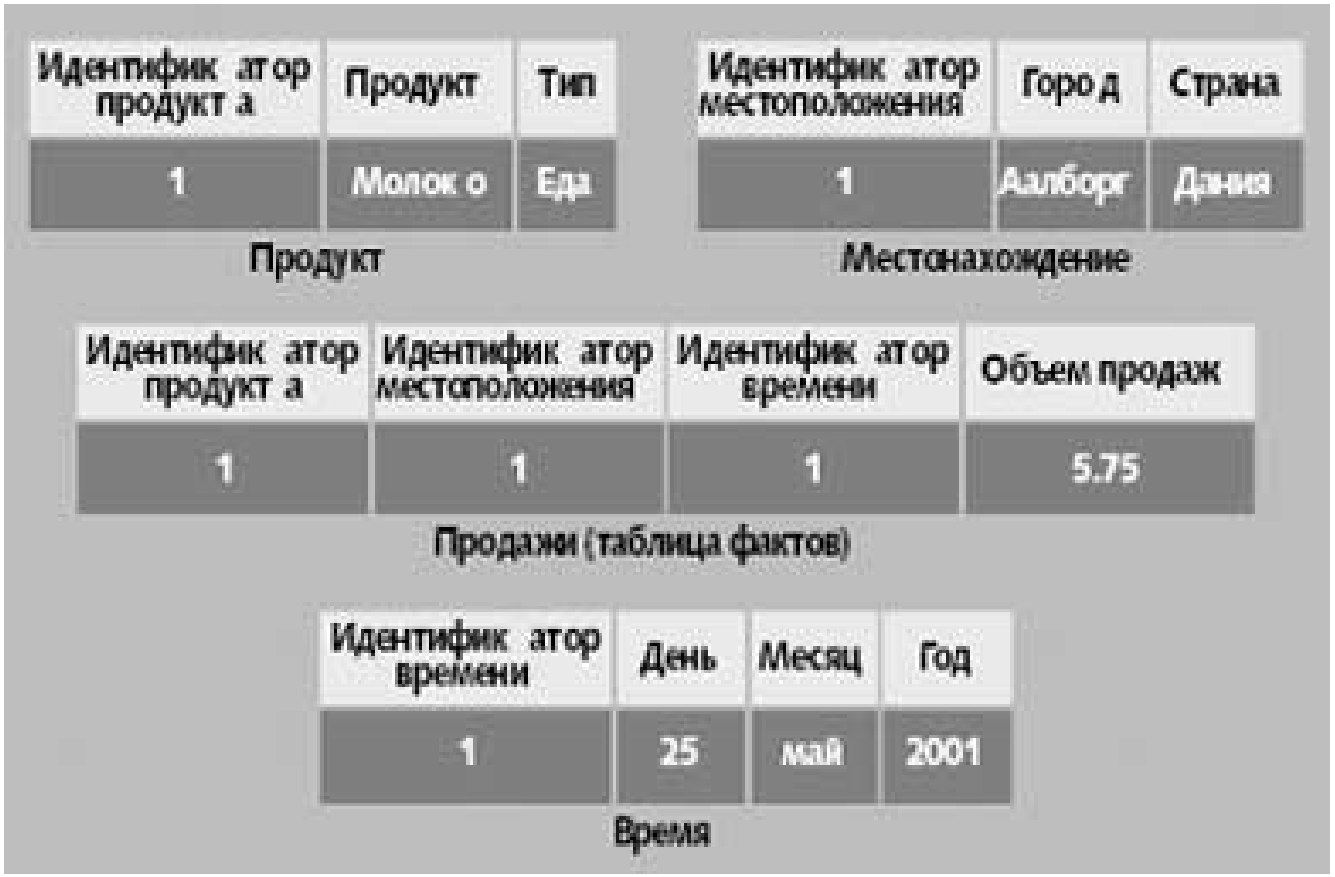
1. Системы многомерной оперативной аналитической обработки (MOLAP) хранят данные в специализированных многомерных структурах.
2. Реляционные системы OLAP (ROLAP) для хранения данных используют реляционные базы данных, а также применяют специализированные индексные структуры, такие как битовые карты, чтобы добиться высокой скорости выполнения запросов.

Многомерные БД

В ROLAP, как правило, используются схемы «звезда» и «снежинка», при которых данные хранятся в таблицах фактов и таблицах измерений. Таблица фактов содержит одну строку для каждого факта в кубе. Для каждого измерения отводится отдельный столбец, содержащий значение параметра для конкретного факта, а также столбец для каждого измерения, которое содержит внешний ключ, ссылающийся на таблицу измерений для конкретного измерения.

Многомерные БД

Схема «звезда» для куба продаж. Информация со всех уровней в измерении хранится в одной таблице измерений, например, названия продуктов и типы продуктов хранятся в таблице «Продукт».



Идентификатор продукта	Продукт	Тип
1	Молоко	Еда

Продукт

Идентификатор местоположения	Город	Страна
1	Аалборг	Дания

Местонахождение

Идентификатор продукта	Идентификатор местоположения	Идентификатор времени	Объем продаж
1	1	1	5.75

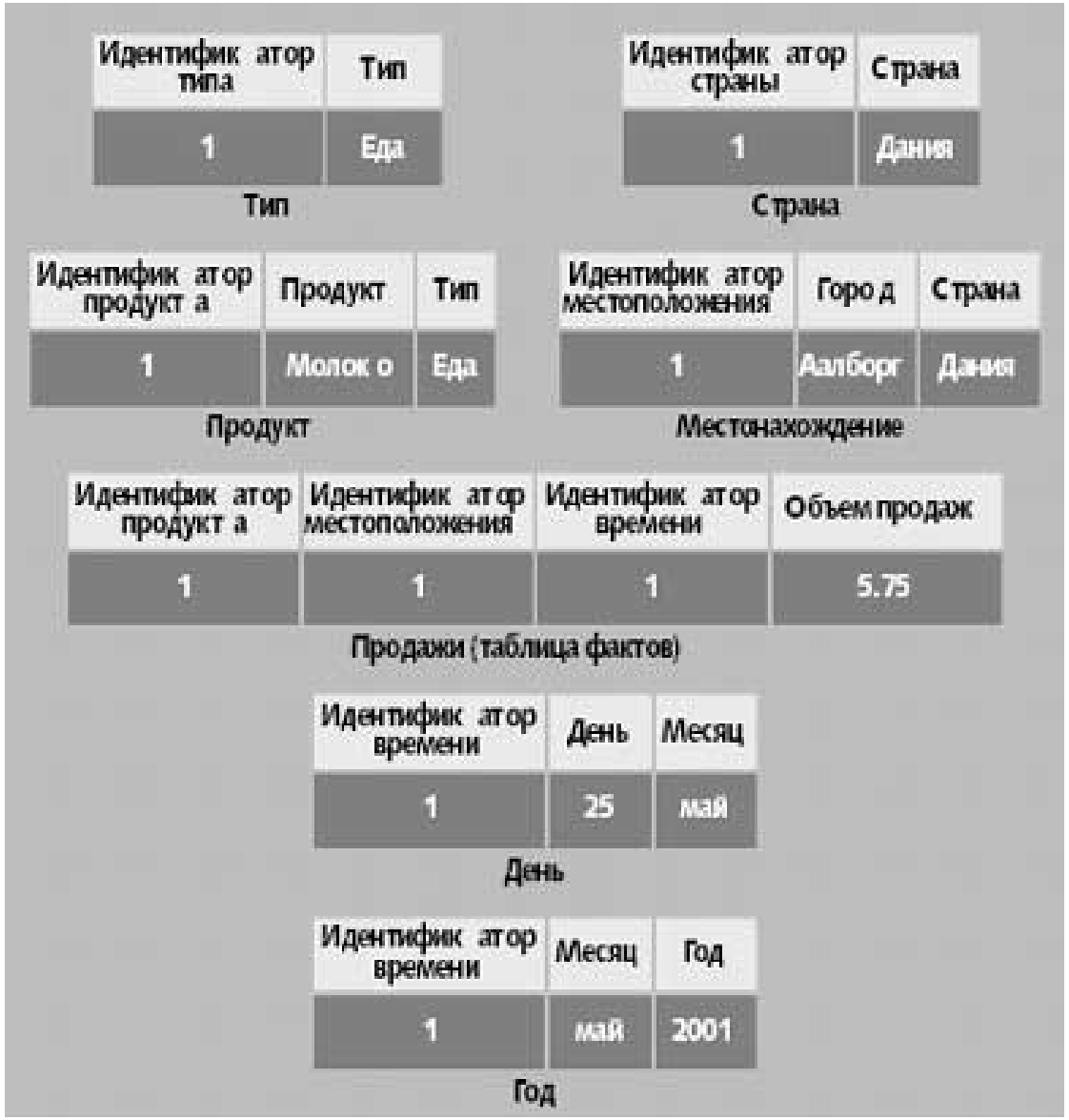
Продажи (таблица фактов)

Идентификатор времени	День	Месяц	Год
1	25	май	2001

Время

Многомерные БД

Схема «снежинка» для куба продаж. Информация из различных уровней в измерении хранится в различных таблицах. Например, названия продуктов и типы продуктов хранятся в таблицах «Продукт» и «Тип» соответственно.



Многомерные БД

За 30 лет с момента своего возникновения технология многомерных баз данных прошла серьезную эволюцию. С недавних пор она стала реализовываться в решениях, предназначенных для массового рынка, а ведущие производители теперь выпускают многомерные ядра вместе со своими реляционными базами данных.

Многомерные БД

Данные, которые необходимо анализировать, становятся все более распределенными. К примеру, это часто необходимо для выполнения анализа, при котором используются данные в формате XML, получаемые с определенных Web-сайтов. Растущая распределенность данных, в свою очередь, требует применения методов, которые позволяют легко добавлять новые данные в многомерные базы данных, тем самым, упрощая задачу создания интегрированного хранилища данных.

Многомерные БД

Технология многомерных баз данных также применяется к новым типам данных, которые современные технологии зачастую не в состоянии адекватно анализировать.

Наконец, технология многомерных баз данных все больше будет применяться там, где результаты анализа напрямую передаются в другие системы, тем самым, исключая участие человека в этом процессе.

БАЗЫ ДАННЫХ
часть II

Технологии базы данных для WWW

Даниэла Флореску, Алон Леви, Альберто Мендельсон **Технологии баз данных для World-Wide Web: обзор** // Системы управления базами данных - Издательство «Открытые Системы», #04-05, 1998 г.

БД для WWW

Популярность World-Wide Web (WWW) превратила его в главное средство распространения информации.

Основной возникающий здесь вопрос – как управлять большими объемами данных?

Новый контекст WWW вынуждает значительно расширить ранее используемые технологии.

Основная цель лекции состоит в том, чтобы классифицировать различные задачи, к решению которых применялись концепции баз данных.

БД для WWW

Сосредоточимся на трех классах задач, связанных с управлением информацией в среде WWW:

1. Моделирование и запросы в WWW.
2. Выборка и интеграция информации.
3. Разработка и реструктуризация Web-сайтов.

БД для WWW

Моделирование и запросы в WWW: Предположим, что мы рассматриваем Web как ориентированный граф, узлы которого являются страницами Web, а дуги – связями между страницами. Рассмотрим задачу формулировки запросов для поиска определенных страниц Web. При этом запросы могут быть основаны на содержании нужных страниц и на структуре связей, соединяющих эти страницы.

БД для WWW

Выборка и интеграция информации: Некоторые Web-сайты могут рассматриваться на более тонком уровне гранулярности, чем страницы, как контейнеры структурированных данных. В связи с ростом числа таких сайтов становятся актуальными две следующие задачи. **Первая** - осуществлять выборку данных, представленных в структурированном виде (например, множество кортежей) из HTML-страниц, их содержащих. Если мы рассматриваем сайты такого рода как автономные неоднородные базы данных, возникает **вторая** задача – формулировка запросов, которые требуют интеграции данных.

БД для WWW

Разработка и реструктуризация Web-сайтов: Другой аспект применения концепций и технологий баз данных – разработка и реструктуризация Web-сайтов, а также управление ими. Конструирование Web-сайтов может начинаться либо с некоторых исходных данных (хранимых в базах данных или в структурированных файлах), либо путем реструктуризации уже существующих Web-сайтов. Выполнение этой задачи требует использования каких-либо методов моделирования структуры Web-сайта и языков для реструктуризации данных таким образом, чтобы они соответствовали желаемой структуре.

БД для WWW

Представление данных для задач Web/DB

Создание систем для решения любой из указанных выше задач требует выбора какого-либо метода для моделирования предметной области. В частности, нам необходимо в этих задачах моделировать сам Web, структуру Web-сайтов, внутреннюю структуру страниц Web, и, наконец, содержание сайтов с более тонкой степенью гранулярности. Далее необходимо обсудить главные факторы, характеризующие модели данных, используемые в приложениях Web.

БД для WWW

Представление данных для задач Web/DB

Графовые модели данных: Нам необходимо моделировать множество страниц Web, а также связи между ними. Эти страницы могут полностью находится на нескольких сайтах либо на единственном сайте. Следовательно, естественный способ моделирования этих данных основан на модели данных ***помеченных графов***. В этой модели узлы представляют страницы Web (или внутренние компоненты страниц), а дуги – связи между страницами. Метки на дугах могут рассматриваться как имена атрибутов.

БД для WWW

Представление данных для задач Web/DB

Модели слабоструктурированных данных: Второй аспект моделирования данных для приложений Web заключается в том, что во многих случаях структура этих данных не является постоянной.

Слабоструктурированными называются данные, обладающие какими-либо из следующих характеристик:

1. Схема не задана заранее и может неявно содержаться в данных.
2. Схема сравнительно велика (в смысле объема данных) и может часто изменяться.
3. Схема является описательной, а не предписывающей.
4. Данные не являются строго типизированными, т.е., для различных объектов значения одного и того же атрибута могут иметь различные типы.

БД для WWW

Представление данных для задач Web/DB

Модели слабоструктурированных данных были основаны на помеченных ориентированных графах. В модели слабоструктурированных данных не налагается какого-либо ограничения на множество дуг, которые исходят от данного узла в графе, или на типы значений атрибутов.

В связи с упомянутыми выше характеристиками слабоструктурированных данных становится важной в этом контексте дополнительная возможность – запрашивать схему (т.е., метки дуг в графе).

БД для WWW

Представление данных для задач Web/DB

Другая отличительная черта моделей, используемых в приложениях Web/DB – присутствие специфических для Web конструкций в представлении данных. Например, в некоторых моделях различаются унарное отношение, идентифицирующее страницы, и бинарное отношение для связей между страницами. Кроме того, мы можем различать связи внутри Web-сайта и внешние связи.

БД для WWW

Представление данных для задач Web/DB

К свойствам второго порядка, которыми различаются обсуждаемые здесь модели данных, можно отнести: (1) способность моделирования некоторого порядка на множестве элементов в базе данных, (2) моделирование вложенных структур данных и (3) поддержка типов коллекций (множества, мультимножества, массивы).

БД для WWW

Представление данных для задач Web/DB

Важный аспект языков запросов данных в приложениях Web – необходимость генерировать сложные структуры в результате обработки запроса. Например, результат некоторого запроса в системе управления Web-сайтом может представлять собой граф, моделирующий этот Web-сайт. Следовательно, фундаментальная характеристика многих из языков запросов состоит в том, что их выражения запросов содержат компонент структурирования наряду с традиционным компонентом фильтрации данных.

БД для WWW

Моделирование Web и запросы

Первыми инструментальными средствами для обработки запросов в Web, были известные поисковые машины, которые теперь широко развернуты и активно используются. Они основаны на поисковых индексах слов и фраз, встречающихся в документах, обнаруженных «роботами» (crawler) Web. Совсем недавно были предприняты усилия, направленные на преодоление ограничений этой парадигмы за счет использования в запросах структуры связей. Прототип поисковой машины Web следующего поколения Google интенсивно использует структуру Web для повышения производительности функционирования «робота» и индексирования.

БД для WWW

Гипертекстовые/документальные языки запросов: Ряд моделей и языков запросов для структурированных документов и гипертекстов был предложен еще в период, предшествующий появлению Web. Новый аспект этого подхода заключается в возможности производить запросы относительно структуры с помощью переменных-путей.

БД для WWW

Графовые языки запросов: Исследования, связанные с использованием графов для моделирования баз данных, которые были стимулированы такими приложениями, как разработка программного обеспечения и управление компьютерными сетями, привели к созданию языков целого ряда языков, например, G, G+ и GraphLog, основанных на графах. Они поддерживают использование в запросах правильных выражений путей и графовых конструкций.

БД для WWW

Языки запросов для слабоструктурированных данных: Такие языки запросов для слабоструктурированных данных, как Lorel, UnQL и STRUDQL, также используют помеченные графы как гибкую модель данных. В отличие от графовых языков запросов, они делают акцент на возможности запрашивать схему и приспосабливаться к нерегулярностям в данных, таких, например, как опущенные или повторяющиеся поля, неоднородные записи.

БД для WWW

Язык WebSQL: В языке WebSQL предлагается моделировать Web как реляционную базу данных, состоящую из двух (виртуальных) отношений: *Документ* и *Якорь*. Отношение Документ содержит по одному кортежу для каждого документа из Web, а отношение Якорь – по одному кортежу для каждого якоря в каждом документе из Web. Такая реляционная абстракция Web позволяет использовать для формулировки запросов язык, подобный SQL.

Если бы *Документ* и *Якорь* были фактическими отношениями, можно было бы просто использовать SQL. Но поскольку эти отношения являются полностью виртуальными, нельзя оперировать ими непосредственно.

БД для WWW

Семантика WebSQL зависит от *материализации* частей этих отношений путем спецификации представляющих интерес документов во фразе FROM запроса. Основным способом материализации является навигация из известных URL. Для описания такой навигации используются правильные выражения путей.

Атом такого правильного выражения может иметь форму d1 = > d2, означающую, что документ d1 указывает на d2, и d2 хранится на ином сервере, чем d1.

Он может иметь также форму d1 – > d2, которая, в свою очередь, означает, что d1 указывает на d2, и d2 хранится на том же самом сервере, что и d1.

БД для WWW

Предположим, например, что мы хотим найти список триплетов вида (d1, d2, метка), где d1 – документ, хранимый на нашем локальном сайте, d2 – документ, хранимый где-либо еще, и d1 указывает на d2 с помощью связи, помеченной меткой. Допустим также, что все наши локальные документы достижимы из www.mysite.start. Тогда указанную задачу можно решить с помощью запроса:

```
SELECT d.url, e.url, a.label
FROM Document d SUCH THAT «www.mysite.start» -> d,
        Document e SUCH THAT  d => e,
        Anchor a SUCH THAT a.base = d.url
WHERE a.href = e.url
```

БД для WWW

Предложение FROM порождает экземпляры двух переменных, определенных на отношении *Документ* (d и e), и одной переменной a на отношении *Якорь*. Области определения переменной d принадлежит каждый локальный документ, а e принимает значения на множестве всех не локальных документов, достижимых непосредственно из d.

В свою очередь, значением переменной a может являться каждая связь, которая исходит из документа d. Дополнительное условие, предписывающее, чтобы целевым документом связи a был документ e, задается предложением WHERE.

БД для WWW

Язык W3QL подобен, по существу, WebSQL, с некоторыми значительными различиями: он использует внешние программы (аналогично определяемым пользователем функциям в объектно-реляционных языках) для спецификации условий, налагаемых на содержание файлов, а не формирование условий в синтаксисе языка, и это обеспечивает механизмы для обработки форм, встречающихся в процессе навигации.

БД для WWW

WQL, язык запросов проекта WebDB подобен WebSQL, но он в большей мере поддерживает функциональные возможности SQL, допуская, например, агрегацию и группирование, и, кроме того, обеспечивает ограниченную поддержку запросов внутридокументной структуры.

БД для WWW

Рассмотренные выше языки интерпретируют страницы Web как атомарные объекты с двумя свойствами: они могут содержать или не содержать некоторые текстовые образцы и они могут указывать на другие объекты. Опыт использования таких языков показывает, что имеется две основные области приложений, для которых они могут быть полезны: (1) создание оболочек (wrapping) для данных, трансформация и реструктуризация данных, (2) создание и реструктуризация Web-сайтов. В обеих этих областях приложений часто оказывается существенной возможность иметь доступ к внутренней структуре страниц Web из языка запросов.

БАЗЫ ДАННЫХ
часть II

Технологии базы данных для WWW

Языки манипулирования данными Web

Языки запросов второго поколения для Web превосходят языки первого поколения в двух важных аспектах.

- Прежде всего, они обеспечивают доступ к структуре объектов Web, которыми они манипулируют. Другими словами, они моделируют внутреннюю структуру документов Web, а также внешние связи, которые их соединяют. Они поддерживают связи для моделирования гиперссылок, а некоторые из них поддерживают также упорядоченные совокупности записей для более естественного представления данных.
- Во-вторых, эти языки обеспечивают возможности создания новых сложных структур в результате запроса. Поскольку данные в Web обычно являются слабоструктурированными, в этих языках придается особое значение поддержке возможностей для работы со слабоструктурированными данными.

Язык WebOQL

Язык WebOQL: Основная структура данных в WebOQL – гипердерево. *Гипердеревья* – это упорядоченные деревья с помеченными дугами, причем имеется два типа дуг – внутренние и внешние. Внутренние дуги используются для представления структурированных объектов, а внешние – для представления связей (обычно гиперссылок) между объектами. Дуги снабжаются метками, в качестве которых используются записи. Язык WebOQL позволяет манипулировать как отдельными гипердеревьями, так и Web в целом, и они (гипердеревья и Web) могут создаваться в результате обработки запроса.

Язык WebOQL

На рисунке показано гипердерево, содержащее описания публикаций нескольких исследовательских групп.

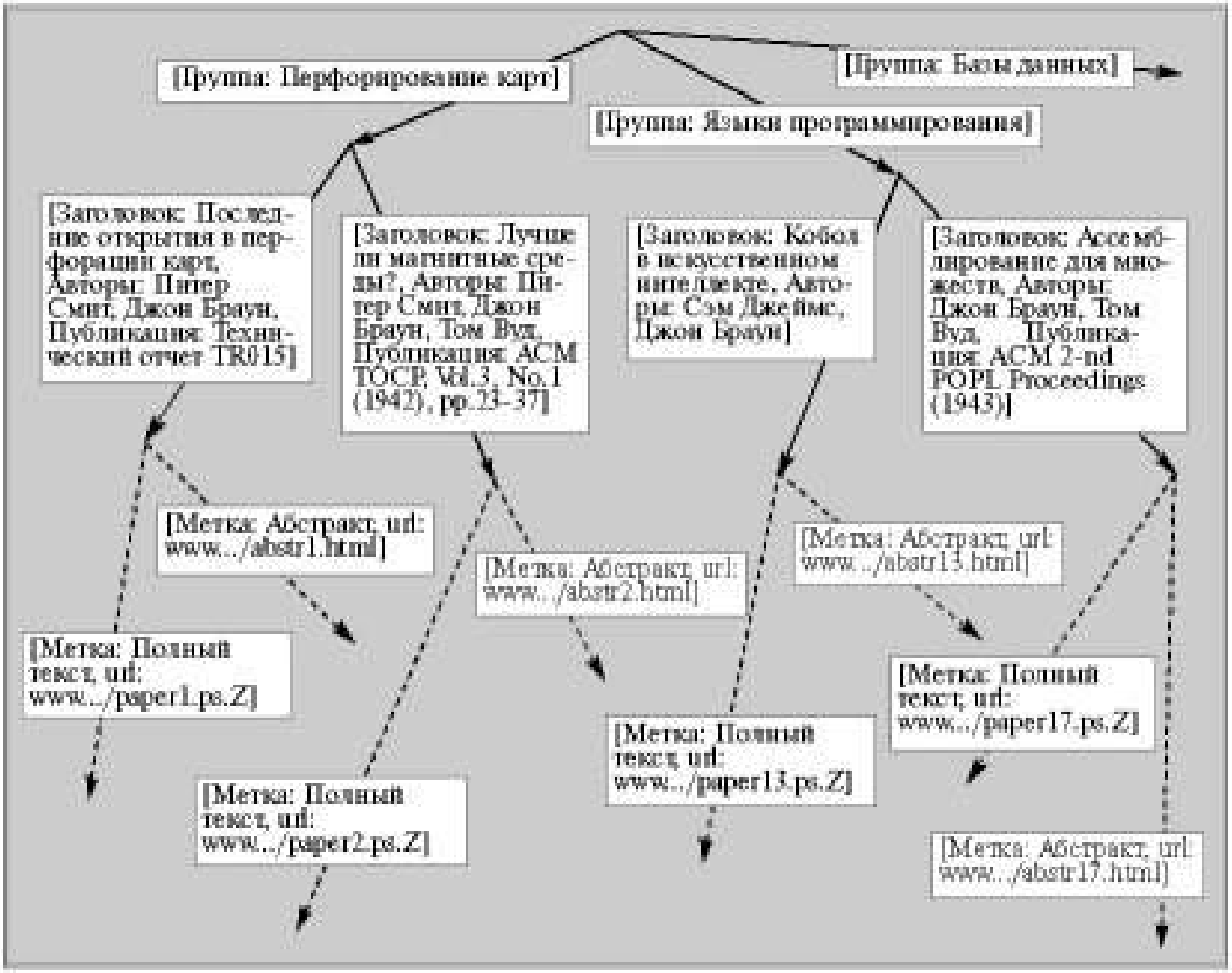


Рис. 1: Пример гипердерева

Язык WebOQL

Предположим, например, что база данных документов на рисунке выше имеет имя СтатьиПоИнформатике и что мы хотим осуществить из нее выборку названия и URL полных текстов статей Смита. Тогда нужно использовать следующий запрос:

```
select [y.Название, y'.Url]
from x in СтатьиПоИнформатике, y in x'
where y.Авторы ~ «Смит»
```

Язык WebOQL

В этом запросе переменная x определена на множестве простых деревьев базы данных СтатьиПоИнформатике, а при заданном значении x переменная y, в свою очередь, принимает значения на множестве простых деревьев x'. Переменная x' обозначает результат применения к дереву x оператора (‘), который возвращает первое поддерево его параметра. Тот же самый оператор используется для извлечения из дерева y его первого поддерева в y'.Url. Квадратные скобки обозначают оператор, который строит дугу, помеченную записью, образуемой аргументом (в приведенном примере предполагается, что запись включает поля с указанными именами). Наконец, тильда (~) представляет собой предикат сопоставления со строковым образцом: его левый аргумент – строка, а правый – образец.

Язык WebOQL

Рассмотренные выше запросы отображают гипердерево в другое гипердерево, или, если говорить в более общих терминах, запрос – это функция, которая отображает один Web в другой. Например, следующий запрос создает новую страницу для каждой исследовательской группы (использующей имя группы как URL). Каждая страница содержит публикации соответствующей группы.

```
select x' as x.Группа
from x in СтатьиПоИнформатике
```

Язык WebOQL

В общем случае фраза select имеет вид:

select q1 as s1, q2 as s2, ..., qm as sm

где каждое qi – это запрос, а каждое из si – или запрос строки или схема. Фразы «as» создают URL s1, s2,... , sm, которые присваиваются новым страницам, полученным в результате выполнения каждого из запросов qi.

БД для WWW

Шаблоны навигации: Шаблоны навигации – это правильные выражения в алфавите предикатов, определенных над записями. Они позволяют специфицировать структуру путей, по которым необходимо следовать для того, чтобы найти значения переменных.

Предположим, что мы имеем некоторый программный продукт, документация к которому представлена в формате HTML, и мы хотим сформировать полнотекстовый индекс для нее. Такие документы образуют сложный гипертекст, но можно просматривать их и последовательно, следуя по связям, помеченным меткой «Следующий». Мы можем получить эту информацию, используя следующий запрос:

```
select [ x.Url, x.Текст ]

from x in browse(«root.html»)

        via (^*[Текст ~ «Следующий»]>)*
```

Язык STRUQL

STRUQL – это язык запросов системы управления Web-сайтами STRUDEL. Хотя STRUQL был разработан в контексте специфического приложения Web, он является универсальным языком запросов для слабоструктурированных данных, основанным на модели данных помеченных ориентированных графов.

Результат запроса в STRUQL представляет собой граф в той же самой модели данных, что и входные графы. В системе STRUDEL язык STRUQL использовался для решения двух задач: для запросов к неоднородным источникам с тем, чтобы интегрировать их в граф данных сайта, и для запросов к этому графу данных с целью продуцирования графа сайта.

Язык STRUQL

Запрос в STRUQL представляет собой набор, возможно, вложенных блоков следующего вида:

[where C1,...,Ck] [create N1,...,Nn]
[link L1,...,Lp] [collect G1,...,Gq].

Фраза where может включать либо условия принадлежности множеству, либо условия, налагаемые на пары узлов, выражающие используемые правильные выражения путей. Фраза where продуцирует все связывания переменных-узлов и переменных-дуг со значениями из исходного графа. Оставшиеся фразы используют функции Сколема для построения нового графа из этих связываний.

Язык STRUQL

Для примера приведен запрос, определяющий некоторый Web-сайт, начиная с библиографического файла Bibtex, моделируемого как помеченный граф. Рассматриваемый Web-сайт будет состоять из страниц трех видов: страницы PaperPresentation для каждого источника в библиографии, страницы Year для каждого года, указывающей все статьи, опубликованные в этом году, и, наконец, страницы Root, указывающей на все страницы Year.

Язык STRUQL

```
// Создать страницу Root create RootPage()
// Создать представление для каждой
// публикации x
where Publications(x), x -> 1 -> v
create PaperPresentation(x)
link PaperPresentation(x) -> 1 -> v
{ // Создать страницу для каждого года
where 1 = «year»
create YearPage(v)
link YearPage(v) -> «Year» -> u
YearPage(v) -> «Paper» ->PaperPresentation(x),
// Связать корневую страницу с каждой
// страницей года
RootPage() -> «YearPage» -> YearPage(v) }
```

Язык STRUQL

Здесь выражение Publications(x) во фразе where обозначает, что x принадлежит совокупности публикаций Publications. В свою очередь, атом x -> 1 -> v обозначает, что существует связь в графе от x к v, и метка соответствующей дуги – 1. Такая же нотация используется во фразе link для того, чтобы специфицировать вновь созданные ребра в результирующем графе. После создания корневой страницы Root первый оператор create генерирует страницу для каждой публикации, обозначенную функцией Сколема PaperPresentation. Второй оператор create, вложенный во внешний запрос, генерирует страницу Year для каждого года и связывает ее со страницей Root, а также со страницами PaperPresentation тех публикаций, которые относятся к этому году. Отметим, что функция Сколема YearPage обеспечивает, чтобы страница Year для конкретного года создавалась только один раз, независимо от того, сколько статей было опубликовано в этом году.

Язык STRUQL

Запись того же самого запроса в WebOQL:

select unique [Url: x.year, Label:»YearPage»]
as «RootPage», [label: «Paper» / x]
as x.year from x in browse(«bibtex: myfile.bib») | *select [year: y.url] + y as y.url from y in «browse(RootPage)»*

Запрос в WebOQL состоит из двух подзапросов. Полученная в результате первого из них подструктура Web поступает в качестве исходных данных во второй запрос, что достигается с помощью использования оператора «|». Первый подзапрос строит страницы Root, Paper и Year, а второй переопределяет каждую страницу Year, добавляя к ней поле «year».

Интеграция информации

WWW содержит все возрастающее число информационных источников, которые могут просматриваться как контейнеры множеств кортежей. Эти «кортежи» могут быть либо встроенными в HTML-страницы, либо быть скрытыми за интерфейсами форм. Благодаря написанию специальных программ, называемых оболочками (wrapper), можно создать иллюзию, что данный Web-сайт обслуживает множества кортежей. Будем называть комбинацию такого Web-сайта и ассоциированной с ним оболочки Web-источником.

Интеграция информации

Задача системы интеграции информации, поддерживаемой средствами Web, состоит в том, чтобы отвечать на запросы, которые могут потребовать извлечения и комбинирования данных из множества Web-источников. Например, рассмотрим такую предметную область, как кино. Сайт Internet Movie Database содержит исчерпывающие данные о кинофильмах, составе исполнителей ролей, жанрах и руководителях съемки. Во множестве других Web-источников (например, на Web-сайтах большинства газет) могут быть найдены также рецензии на кинофильмы, а некоторые Web-источники предоставляют расписания показа кинофильмов. Комбинируя данные из этих источников мы можем отвечать на запросы типа: выдать мне какой-либо фильм с Фрэнком Синатрой в главной роли, который можно посмотреть сегодня вечером в Париже, время сеанса и рецензии на него.

Интеграция информации

Важные различия при построении систем интеграции данных, а, следовательно, и систем интеграции данных Web, возникают в связи с тем, принимается ли подход, основанный на хранилищах данных, или виртуальный подход. В случае использования первого подхода данные из множества Web-источников загружаются в хранилище данных, и далее все запросы будут обращены к этому хранилищу данных. В таком случае необходимо, чтобы при изменении данных в источниках обновлялось и хранилище данных. Однако преимущество состоит в том, что может быть гарантирована адекватная эффективность на стадии обработки запроса.

Интеграция информации

При виртуальном подходе данные остаются в Web-источниках, и запросы к системе интеграции данных декомпозируются на стадии исполнения на запросы к отдельным источникам. При таком подходе данные не тиражируются, и тем самым гарантируется их актуальность на стадии обработки запросов. С другой стороны, поскольку Web-источники автономны, для обеспечения адекватной эффективности необходимы более изощренные техника оптимизации запросов и исполнения. Виртуальный подход более уместен при построении таких систем, где число источников велико, данные изменяются часто, и имеется слабый контроль над Web-источниками.

Интеграция информации

Архитектура системы виртуальной интеграции данных имеет две главных особенности, отличающие такую систему от традиционной системы базы данных:

- система не взаимодействует непосредственно с локальными менеджерами хранения данных. Вместо этого для получения данных механизмы исполнения запросов взаимодействуют с множеством оболочек (wrapper);
- пользователь не формулирует запросы непосредственно в терминах той схемы, в соответствии с которой хранятся данные. Вместо этого пользователь формулирует запросы в терминах промежуточной схемы.

Создание и реструктуризация Web-сайтов

Web-сайты, по существу, обеспечивают доступ к сложным структурам информации. Поэтому естественно применить методы систем баз данных для создания и поддержки Web-сайтов. Можно выделить два общих класса задач создания Web-сайтов: создание Web-сайтов из некоторой совокупности внутренних источников данных и создание их путем реструктуризации существующих Web-сайтов.

Для обоих этих классов задач необходимы одни и те же методы.

Создание и реструктуризация Web-сайтов

С целью применения технологий баз данных для создания Web-сайтов было разработано несколько систем. Общая характерное свойство этих систем состоит в том, что они обеспечивают явное декларативное представление структуры Web-сайта. Структура Web-сайта рассматривается как представление, определенное над существующими данными. Однако, языки, используемые для создания этих представлений, дают в результате графы Web-страниц с гипертекстовыми связями, а не простые таблицы. Указанные выше системы различаются моделями данных, которую они используют, языками запросов, а также использованием или отказом от использования промежуточного логического представления Web-сайта наряду с наличием конечного представления в формате HTML.

Создание и реструктуризация Web-сайтов

Создание Web-сайта, использующего декларативное представление его структуры, обеспечивает несколько важных преимуществ.

- Так как структура и содержание Web-сайта определяются декларативно по запросу, а не процедурным образом с помощью программы, достаточно просто могут создаваться множественные версии такого сайта.

- Кроме того, декларативное представление структуры Web-сайта также позволяет легко поддерживать эволюцию этой структуры.

Создание и реструктуризация Web-сайтов

Архитектура прототипа системы с декларативно определенной структурой показана на рисунке.

На нижнем уровне системы осуществляется доступ к множеству источников данных, содержащих те данные, которые будут обслуживаться на Web-сайте. Эти данные могут храниться в базах данных, в структурированных файлах, или на уже существующих Web-сайтах.

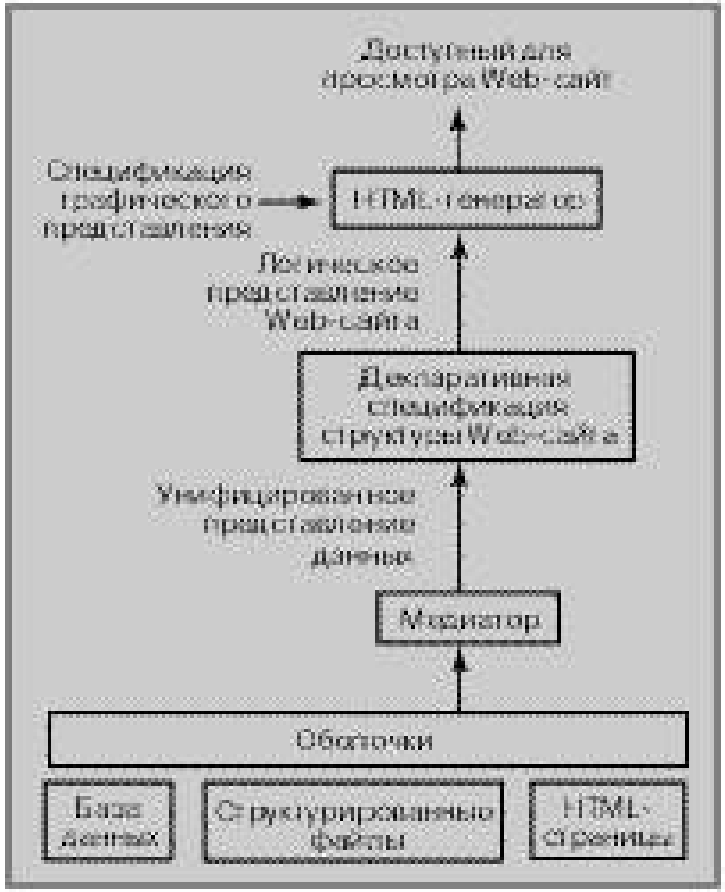


Рис. 3. Архитектура систем управления Web-сайтами

Создание и реструктуризация Web-сайтов

Данные представляются в системе средствами некоторой модели данных, и система обеспечивает унифицированный интерфейс к этим источникам данных, используя технологии, подобные описанным выше. Главным шагом в создании Web-сайта является спецификация выражения, которое в декларативной форме представляет структуру Web-сайта. Это выражение записывается на специальном языке запросов, предоставляемом системой. В результате применения этого запроса к внутренним данным формируется логическое представление Web-сайта в терминах модели данных системы (например, как помеченный ориентированный граф). Наконец, для фактического создания доступного для просмотра Web-сайта система содержит метод (например, HTML-шаблоны) для трансляции специфицированной логической структуры в набор HTML-файлов.

Продолжение следует…

- Значительное влияние на использование технологий баз данных для приложений Web оказывает появление и развитие языка XML.

- Существенно новые возможности, предоставляемые XML, и связанные с ним инициативы, касающиеся метаданных, несомненно, могут способствовать применимости концепций баз данных для Web, обеспечивая множество необходимых структур в широко принятом формате.

БАЗЫ ДАННЫХ
часть II

Кто такой Администратор Базы Данных?

Е.З. Зиндер **Администратор базы данных - кто он?** // Системы управления базами данных - Издательство «Открытые Системы», #02, 1995 г.

Кто такой АБД?

Кто такой Администратор Баз Данных - администратор в обычном понимании?

Технический эксперт, почти не выходящий из своего кабинета, говорящий с другими работниками предприятия редко, неохотно и на непонятном языке?

Главный хозяин всех данных в компьютерах предприятия?

Системный программист специфического профиля, находящийся под началом руководителя группы системных программистов отдела эксплуатации ВЦ предприятия?

Очень плохо, если на вашем предприятии сложился один из таких вариантов.

Кто такой АБД?

Функция "администрирования данных" стала активно рассматриваться и определяться как вполне самостоятельная с конца 60-х годов. Практическое значение это имело для предприятий, вынужденных использовать вычислительную технику в системах информационного обеспечения своей ежедневной основной деятельности.

Время шло, технологии развивались и усложнялись, специализация углублялась. Однако качественные изменения стали происходить с включением в использование так называемых **интегрированных баз данных**. Одна такая база данных (БД) создавалась для решения многих задач, каждая из которых могла использовать только небольшую, "свою" часть БД, обычно пересекающуюся с частями, используемыми в других задачах.

Таким образом, сформировалось **определение БД как общего информационного ресурса** предприятия. В этом смысле БД стала аналогична большому компьютеру, который одновременно используется многими пользователями с различными целями и должен быть все время работоспособен.

Кто такой АБД?

Как и для каждого общего ресурса значительной важности, БД стала требовать отдельного управления, причем:

- БД требует управления для обеспечения ее повседневной эксплуатации,
- БД развивается, отвечая изменениям в потребностях предприятия, и требуется управление ее развитием,
- БД и технология ее разработки и развития являются объектами высокой сложности, требующим специальных знаний, высокого уровня квалификации и строгой дисциплины разработки и эксплуатации.

Функция управления БД получила название "администрирование базы данных". Естественно, **лицо, ответственное за администрирование БД, получило название "Администратор базы данных", или АБД.**

Кто такой АБД?

При этом от непосредственного управления данными отстраняются программисты, выполняющие конкретные прикладные разработки, пользователи, которые не должны изменять или даже видеть не принадлежащие им данные, и другие сотрудники, которые, быть может, хотели бы это делать.

Кто такой АБД?

Классические подходы к наполнению содержанием понятия "АБД" стали формироваться после издания рабочего отчета группы по базам данных Американского Национального Института Стандартов ANSI/X3/SPARC в 1975 г. В этом отчете была описана трехуровневая архитектура СУБД, в которой выделялся уровень внешних схем данных, уровень концептуальной схемы данных и уровень схемы физического хранения данных. В соответствии с этой архитектурой определялись три роли АБД: администратор концептуальной схемы, администратор внешних схем и администратор хранения данных. Эти роли в случае очень маленькой системы мог играть один человек, в большой системе для выполнения каждой роли могла назначаться группа людей. Каждой роли соответствовал набор функций, а все эти функции вместе составляли функции АБД.

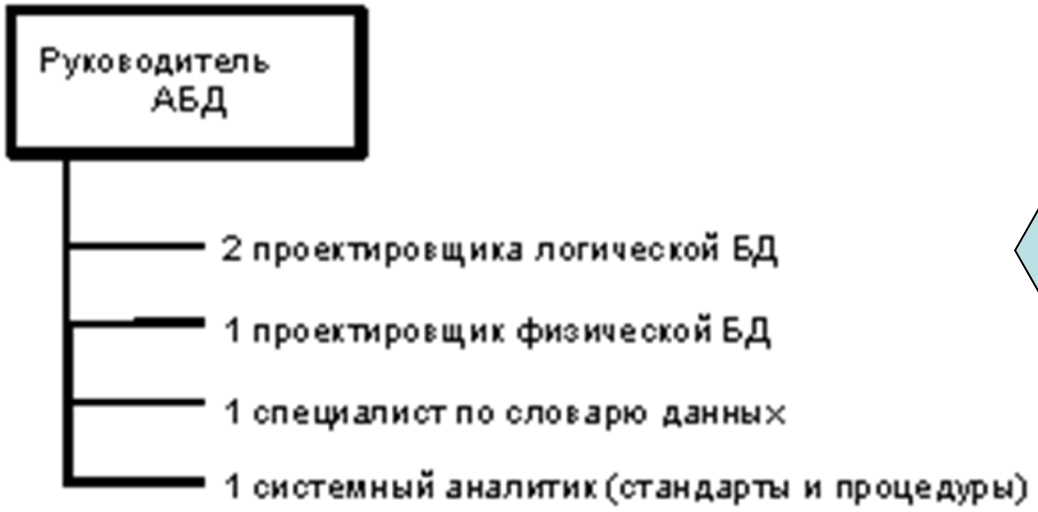
Кто такой АБД?

В 1980 - 1981 г. в американской литературе стало принятым включать в функции АБД:

- организационное и техническое планирование БД,
- проектирование БД,
- обеспечение поддержки разработок прикладных программ,
- управление эксплуатацией БД.

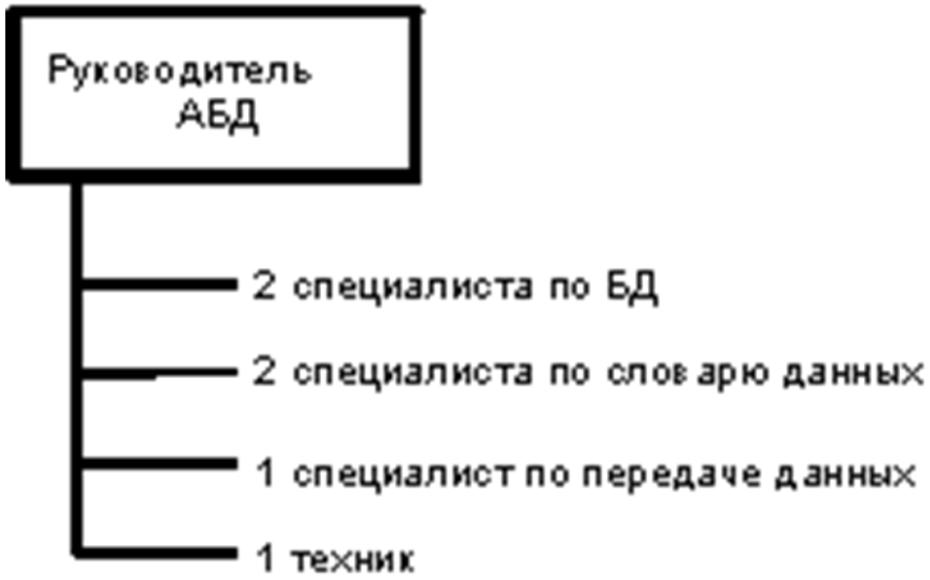
Видно, что функции АБД в общем случае были ориентированы и на разработку БД собственными силами, и на эксплуатацию БД, хотя рассматривались и варианты простых неструктурированных групп АБД, специализирующихся только на эксплуатации БД.

Кто такой АБД?



Начало 80-х: неструктурированная Группа АБД, специализирующаяся на проектировании

Начало 80-х: неструктурированная Группа АБД, специализирующаяся на эксплуатации базы данных



Кто такой АБД?

Начало 80-х: Организованная по функциональному признаку Группа АБД, обеспечивающая сопровождение СУБД

Кто такой АБД?

Первое определение АБД в ГОСТ-ах задало слишком узкий состав функций АБД:

- подготовка вычислительного комплекса к установке СУБД,
- участие в установке и приемке СУБД и самой БД с комплексом прикладных программ,
- управление эксплуатацией БД,
- подготовка словарей и другой НСИ - нормативно-справочной информации - к моменту начала испытания БД.

Существенно, что функции АБД были заужены и всегда ориентированы только на эксплуатацию БД; предполагалось, что разработка БД ведется силами специализированной организации.

Кто такой АБД?

Концепция управления базой данных как содержание основной деятельности АБД сохранилась и по сей день. Однако **объем функций АБД стал более четко определен и, в частности, отделен от так называемого стратегического планирования, концептуального и, чаще всего логического проектирования базы данных**, что связано с развитием технологий и специализированных инструментов разработки информационных систем и других автоматизированных систем с базами данных.

Кто такой АБД?

Основная работа по планированию информационных потребностей предприятия, проектированию концептуальной и логической схемы БД, внешних схем, используемых в отдельных процессах обработки информации, ложится теперь на группу проектирования Автоматизированной Системы (АС).

Эта группа выполняет несколько стадий проектных работ, например:

- стратегическое планирование развития АС,
- детальный анализ функциональных и информационных потребностей,
- проектирование таблиц БД, прототипов экранных форм, печатных отчетов и др.

Кто такой АБД?

Основу состава такой группы составляют аналитики: так называемые **прикладные аналитики** (business analysts) и **системные аналитики** (system analysts).

Аналитики с помощью CASE-системы в идеале (пока не всегда достижимом) получают такой вариант АС, который внешне выглядит так, как его должны или хотят видеть пользователи. Этот вариант АС, конечно, не рассчитан на эффективное функционирование, не реализует традиционных процедур поддержки сохранности БД и, может быть, не проводит множество пожеланий пользователей к тем или иным деталям сервиса, который трудно предусмотреть в обобщенных средствах CASE.

Кто такой АБД?

Обеспечение надежной и эффективной работы пользователей и программ с БД, поддержка разработчиков в их доступе к БД и средствам разработки реализует группа АБД.

Кто такой АБД?

В достаточно полный набор функций АБД входит:

а) консультирование аналитиков и программистов по особенностям используемой версии СУБД и инструментов разработки, участие - совместно с аналитиками по проектированию баз данных - в логическом проектировании БД в тех случаях, когда полезно учитывать специфические для СУБД или режимов обработки данных рекомендации по проектированию БД,

б) установка СУБД, программных инструментов разработки АС (языков программирования экранных приложений, генераторов отчетов, CASE-систем и др.) и инструментов пользователей для прямой работы с БД (средства запросов к БД, офисные системы, системы планирования производства и т.п.),

Кто такой АБД?

в) планирование использования запоминающих устройств компьютера (дисков, основной памяти, лент), участие - совместно с проектировщиками БД - в физическом проектировании таблиц БД,

г) организация работы с БД, находящейся на удаленном компьютере, работы с распределенной БД, т.е. размещенной на нескольких компьютерных центрах (узлах, "нодах"),

д) сбор статистики о работе СУБД, ее настройка и настройка АС в целом для эффективной обработки данных и обслуживания пользователей,

Кто такой АБД?

е) участие в планировании развития аппаратных и системных программных средств предприятия в связи с качественным и количественным ростом требований к АС,

ж) составление процедур использования штатных средств СУБД (программ-утилит и др.) для начальной загрузки данных, копирования и восстановления БД, реорганизации размещения данных и т.п.; передача этих процедур эксплуатационному персоналу,

з) подключение новых разработчиков и пользователей, приписывание им паролей, привилегий доступа к конкретным данным и др.,

и) участие в анализе попыток несанкционированного доступа к БД.

Кто такой АБД?

Из приведенного перечня ясно, что АБД:

- управляющий данными, а не их хозяин;
- системный программист определенного профиля, а также эксперт высшего уровня, обеспечивающий службу эксплуатации решениями по процедурам и регламентам работы;
- лицо, принимающее окончательные решения в своей области, и человек, обладающий способностями к общению, совместному планированию и компромиссам.

Надежность и достоверность - ключевые понятия в деятельности АБД. Он должен уметь вести тщательное документирование всех действий по управлению базой данных.

Кто такой АБД?

Большое значение имеют личностные данные АБД: как руководителя, так и других специалистов Группы.

Например, в понятной и доказательной форме они должны уметь проконсультировать как старшее руководство предприятия, так и разработчиков АС, пользователей и службу эксплуатации, понимать их нужды, принимать их справедливые требования.

Не годится на роль АБД специалист, который - пусть и на основе специальных знаний - без согласования с хозяевами информации (руководством предприятия, пользователями или др.) будет вносить изменения в содержание БД или в нарушение установленных регламентов станет менять режимы доступности данных.

Кто такой АБД?

АБД не всегда выполняет все функции, указанные выше. Их состав зависит от политики автоматизации, проводимой на предприятии. Рассмотрим три варианта политики:

1. **"САМООБЕСПЕЧЕНИЕ"**: предприятие полностью самостоятельно ведет разработку АС поддержки своей деятельности;
2. **"ЗАКАЗЫ"**: предприятие закупает полностью готовый проект АС и его дальнейшую адаптацию для себя, включая проект БД, процедуры ее сопровождения и дальнейшего развития;
3. **"СМЕШАННЫЙ"**, при котором начальная версия системы с ее настройкой на предприятие закупается, а дальнейшее, относительно небольшое развитие и приспособление делается на предприятии при поддержке разработчиков.

Кто такой АБД?

Для варианта "САМООБЕСПЕЧЕНИЕ" АБД выполняет наибольший набор функций, причем в стартовый момент они могут быть практически такими же, как и впоследствии (хотя возможен рост набора функций в связи с развитием самой АС).

В этом варианте практически всегда требуется несколько специалистов в Группе АБД. Кроме того, в большинстве случаев возможно и рекомендуется дальнейшее внутреннее деление Группы.

АБД такого "полного" вида играет активную роль на предприятии вплоть до участия в совещаниях самого высокого уровня.

Кто такой АБД?

АБД в структуре предприятия: вариант "САМООБЕСПЕЧЕНИЕ"

Кто такой АБД?

Для варианта "ЗАКАЗЫ" АБД выполняет наименьший набор функций, причем впоследствии они могут остаться такими же, как и в стартовый момент. В этом варианте часто требуется всего два-три специалиста в Группе АБД, особенно в сравнительно небольших, компактных по числу пользователей и территорий АС.

АБД такого "локального" вида обычно играет менее активную роль на предприятии. По существу это неверно, так как решения должны приниматься во многом по тем же вопросам, что и в случае "САМООБЕСПЕЧЕНИЯ", хотя их реализация в меньшей мере выполняется силами предприятия. Поэтому **и в варианте "ЗАКАЗЫ" АБД должен играть важную роль эксперта вплоть до участия в совещаниях самого высокого уровня.**

Кто такой АБД?

АБД в структуре предприятия: вариант "ЗАКАЗЫ"

Кто такой АБД?

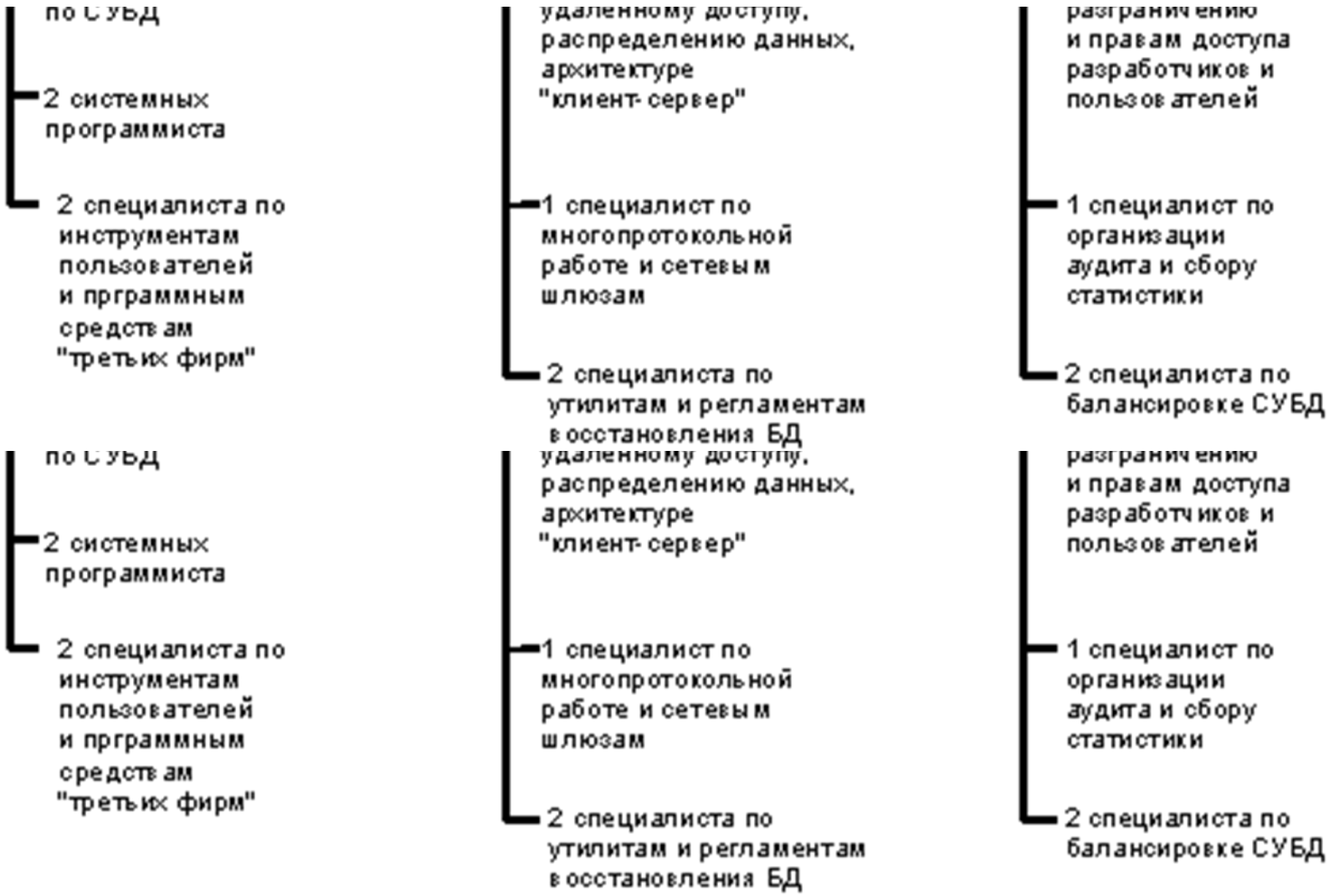
Для варианта "СМЕШАННЫЙ" АБД выполняет тот же набор функций, что и для "САМООБЕСПЕЧЕНИЯ", но их состав дорастает до такого состояния постепенно.

В стартовый момент они практически такие же, как и в варианте "ЗАКАЗЫ", хотя функции поддержки разработчиков, настройки СУБД и др. должны планироваться сразу, а, значит, должно сразу планироваться и обучение АБД в специализированных учебных центрах поставщика СУБД или разработчика АС.

АБД такого "растущего" вида также должен играть активную роль на предприятии вплоть до участия в совещаниях самого высокого уровня.

Кто такой АБД?

Функции Группы АБД "полного" вида



Кто такой АБД?

Функции Группы АБД "локального" вида

Кто такой АБД?

В этом случае АБД как минимум выполняет следующие функции:

д) сбор статистики о работе СУБД, ее настройка и настройка АС в целом для эффективной обработки данных и обслуживания пользователей,

е) участие в планировании развития аппаратных и системных программных средств предприятия в связи с качественным и количественным ростом требований к АС,

з) подключение новых разработчиков и пользователей, приписывание им паролей, привилегий доступа к конкретным данным и др.,

и) участие в анализе попыток несанкционированного доступа к БД.

Кто такой АБД?

Кроме того, АБД принимает участие в приемке готовой АС и в рамках этой работы, в объеме, соответствующем приемке, выполняет функции:

б) установка СУБД, программных инструментов разработки АС и инструментов пользователей для прямой работы с БД,

в) планирование использования запоминающих устройств компьютера (дисков, основной памяти, лент),
г) организация работы с БД, находящейся на удаленном компьютере, работы с распределенной БД,

ж) составление процедур для начальной загрузки данных, копирования и восстановления БД и т.п.; передача этих процедур эксплуатационному персоналу,

Кто такой АБД?

Функции Группы АБД "растущего" вида

Вариант "СМЕШАННЫЙ", при котором начальная версия системы с ее настройкой на предприятие закупается, а дальнейшее, относительно небольшое развитие и приспособление делается на предприятии при поддержке разработчиков.

В соответствии с описанием такого варианта, изложенным выше, а так же в соответствии с реальными объемами доработок, сложностью АС и т.п., Группа АБД и ее функции занимают промежуточное состояние между Группами "локального" и "полного" видов. Обычно, с течением времени происходит рост использования АС, процессы ее развития и эксплуатации усложняются, а сама Группа АБД развивается вплоть до зрелой Группы "полного" вида.

Кто такой АБД?

Существуют и другие виды администрирования, которые чаще всего рассматриваются отдельно от АБД, хотя и тесно с ним связаны. К таким функциям можно отнести:

- администрирование приложений, т.е. управление подключением пользователей к конкретным прикладным программам, входам в меню и т.п., управление расписанием выполнения процедур обработки данных, ведение нормативно-справочной информации и др.,

- администрирование конфигурации и ресурсов вычислительной установки данного узла или ВЦ, его связь с ресурсами СУБД, прикладных программ и пользователей,

- сетевое администрирование и его связь с сетевыми компонентами СУБД и удаленными БД,

Кто такой АБД?

- администрирование систем электронной почты и специальных видов передачи файлов, их связь с клиентами БД и обменами данными с БД,

- администрирование безопасностью для защиты различных данных на предприятии от несанкционированного получения и доступа любого вида (в том числе, защита от АБД, системных программистов и др.).

Эти функции выполняют соответствующие Администраторы и их группы. Но в случае работы с интегрированной БД все они обычно должны выполнять совместно с АБД работы по стыковке своих компонентов и текущему согласованию их взаимодействия.

Кто такой АБД?

Итак…

АБД - это одна из ключевых фигур в ежедневном обеспечении предприятия информацией, включая информацию для принятия самых оперативных или самых ответственных решений.

Это - эксперт или группа экспертов высшего уровня, обеспечивающие своими решениями и своими консультациями продуктивность и работоспособность и баз данных, и работающих с ними людей.

АБД и его группа должны занимать определенное, чаще всего - достаточно высокое положение в иерархии предприятия, принимать участие в решении многих важных вопросов, обладать соответствующими правами.

Кто такой АБД?

Итак…

В свою очередь, частные специальные функции АБД могут меняться при изменении применяемых технологий. Кроме того, функции АБД сильно зависят от принятого на предприятии варианта политики автоматизации.

Численный состав Группы АБД зависит в большей степени от компактности Автоматизированной Системы предприятия и способности специалистов Группы совмещать различные функции.

АБД должен подбираться как по уровню специальных знаний (и способности к их постоянному обновлению), так и по личностным данным.

БАЗЫ ДАННЫХ
часть II

Создание информационных систем на основе БД

Фролов А. В., Фролов Г. В. Базы данных в Интернете: практическое руководство по созданию Web-приложений с базами данных. — Изд. 2-ое, испр. — М.: Издатель-ско-торговый дом «Русская Редакция», 2000. — 448 с.

Связь приложений с базами данных

Для доступа к базам данных SQL Server можно использовать различные методы — программный интерфейс DB Library, программный интерфейс ODBC, объектный интерфейс RDO, объектный интерфейс OLE DB и объектный интерфейс ADO.

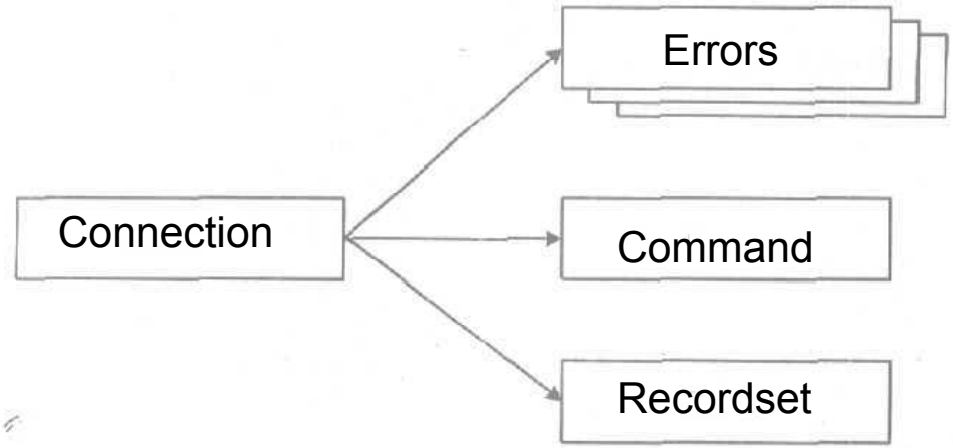
Посредством интерфейса ActiveX Data Objects (ADO) приложения (как обычные, так и ориентированные на использование технологий Интернета) могут подключаться к базам данных, извлекать, обрабатывать и обновлять информацию в них.

Связь приложений с базами данных

ADO представляет собой интерфейс уровня приложений, созданный поверх объектного интерфейса OLE DB. При этом интерфейс OLE DB обеспечивает универсальный доступ к данным. Такой доступ обеспечивается в свою очередь с помощью провайдеров, таких, как Microsoft OLE DB Provider для ODBC (MSDASQL) или Microsoft OLE DB Provider для SQL Server (SQLOLEDB).

Связь приложений с базами данных

Ключевыми элементами программной модели ADO является набор объектов, с помощью которых осуществляется соединение с базами данных, выполнение команд с параметрами, получение результата выполнения этих команд в виде переменных или наборов записей, обработка событий и ошибок.



Связь приложений с базами данных

Пример доступа к БД на языке Jscript с использованием интерфейса ADO:

```
var connect, rs, cmd, ClientID;
connect = Server.CreateObject("ADODB.Connection");
connect.ConnectionTimeout = 15;
connect.CommandTimeout = 10;
connect.Open("DSN=BookStore", "dbo", "password");
cmd = Server.CreateObject("ADODB.Command");
cmd.CommandText = "ListOrders";
cmd.CommandType = adCmdStoredProc;
cmd.ActiveConnection = connect;
cmd. Parameters.Append(cmd.CreateParameter(
"ClientID", adVarChar, adParamInput, 50, ClientID));
rs = cmd.Execute();
```

Связь приложений с базами данных

```
var fieldbooksID = 0;
var fieldAuthor = 1;
var fieldTitle = 2;
var fieldPublisher = 3;
var fieldPrice = 4;
%>
<HTML>
<BODY>
<h2>Вы отобрали для
покупки</h2>
<TABLE BORDER=1>
<%
while (!rs.EOF)
{%>
<tr>
<td>
<%=rs.Fields(fieldAuthor)%>. <%=rs.
Fields(fieldTitle)!H><br>
<X=rs,Fields(fieldPublisher)%></td>
<td>
<%=rs. Fields(f ieldPrice)%> y. e. </td>
</tr>
<%
rs.MoveNext();
}
%>
</TABLE>
%>
```

Здесь мы создаем таблицу и записываем в ее ячейки содержимое полей текущей записи (на которую указывает курсор), обращаясь к четырем из пяти столбцов.

Связь приложений с базами данных

Интерфейс **OLE DB** — открытый стандарт, разработанный специально для предоставления доступа приложениям к базам данных, как реляционных, так и нереляционных (таких, как серверы почты, базы данных VSAM и т. д.).

Создавая приложения OLE DB, Вы можете реализовать в нем как функции провайдера данных, так и функции потребителя данных.

Связь приложений с базами данных

Так же как и в случае только что рассмотренной объектной модели ADO, базовыми элементами программной модели OLE DB является набор объектов. Эти объекты применяются для установки соединения с базами данных и сеансов, выполнения команд с параметрами, получения результата выполнения этих команд в виде переменных или наборов записей, обработки событий и ошибок.

Работа OLE DB основана на модели компонентных объектов COM, поэтому сразу после начала своей работы приложение должно выполнить инициализацию системы COM.

Связь приложений с базами данных

```
IMalloc*            pIMalloc = NULL;
IDBInitialize*      pIDBInitialize = NULL;
IRowset*            pIRowset = NULL;
int main(int argc, TCHAR* argv[J, TCHAR* envp[ ])
{
        if(init())
        {
        if(startCommand())
        get_records();
        }
Else
{
if(pIRowset != NULL)
        pIRowset->Release();
        ifCpIDBInitialize != NULL)
        {
                pIDBInitialize->Uninitialize();
                pIDBInitialize->Release();
        }
        if(pIHalloc != NULL)
        pIMalloc->Release();
}
return 0;
}
```

Связь приложений с базами данных

Функция startCommsnd запускает команду SELECT, выбирающую из таблицы покупателей clients поля с именами ClientID, UserID, Password, RegisterDate и Email:

```
LPCTSTR wSQLString = OLESTR("SELECT ClientID, UserID, Password,
RegisterDate, Email FROM clients");
```

Связь приложений с базами данных

Обработка полей текущей строки выполняется в цикле:

```
for(nCurrentCol = 0; nCurrentCol < nColsCount; nCurrentCol++)
{
if(pColInfo[nCurrentCol].wType == DBTYPE_STR)
{
printf("*10s ", &pRowValues[pDBBind[nCurrentCol].obValue]);
}
else if(pColInfo[nCurrentCol].wType == DBTYPE_I4)
{
printf("X5d ", pRowValues[pDBBind[nCurrentCol].obValue]);
}
else if(pColInfo[nCurrentCol].wType ==
DBTYPE_DBTIMESTAMP)
{
DBTIMESTAMP- ts = (DBTIMESTAMP*)
C&pRowValu9s[pD8BInd[nCurrentCol].obValue]>;
prIntf('1X02d.X02d.X04d X02d: K02d: J!02d ",ts->day, ts->month,
ts->year,ts->hour, ts->minute, ts->second);
}
}
printf("\n");
```

Связь приложений с базами данных

Интерфейс **Open Database Connectivity (ODBC)** представляет собой набор предназначенных для доступа к базам данных функций программного интерфейса. Этот набор предполагает использование структурного языка запросов SQL.

Для того чтобы интерфейс ODBC стал доступен программам, необходимо установить драйвер ODBC. Такой драйвер имеется в составе Microsoft SQL Server, а также в составе многих других СУБД, рассчитанных на работу в среде операционных систем Microsoft Windows, Macintosh и некоторых версий Unix.

Связь приложений с базами данных

Программа, обращающаяся к базам данных посредством интерфейса ODBC, обычно выполняет следующие действия:

1. инициализирует среду выполнения;
2. подключается к источнику данных;
3. создает и выполняет команды;
4. обрабатывает результат выполнения команды;
5. освобождает ресурсы, полученные для работы с ODBC.

Связь приложений с базами данных

Прежде всего программа должна получить идентификатор среды типа SQL_HANDLE_ENV:

```
SQLHENV hEnv = SQL_NULL_HENV;
RETCODE rc;
rc = SQLAllOcHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE,
&hEnv);
```

На следующем этапе нам нужно установить атрибуты среды:

```
rc = SQLSetEnvAttr(hEnv, SQLJVTTR_ODBC_VERSION,
(SQLPOINTER)SQL_OV_ODBC3, SQL_IS,INTEGER);
```

На втором этапе инициализации надо получить идентификатор соединения:

```
SQLHDBC hDbC = SQL_NULL_HDBC;
rc = SQLAllocHandle(SQL_HANDLEJ)BC, hEnv, &hDbc);
```

Связь приложений с базами данных

Фрагмент текста программы, выполняющий соединение с источником данных BookStore:

```
UCHAR szDSN[SQL_MAX_DSN_LENGTH + 1] = "BookStore";
UCHAR szUserName[MAXNAME] = "dbo";
UCHAR szPasswor[fMAXNAME] = "";
rc = SQLConnect(hObc,
szDSN, (SWORD)strlen((const char*)szDSN),
szUserName, (SWORD)strlen( (const cnar*)szUserName),
szPassword, (SWORD)strlen((const char*)szPassword));
```

В случае успешного соединения функция SQLConnect вернет значение SQL_SUCCESS или SQL_SUCCESS_WITH_INFO, а при ошибке - значение SQL_INVALID_HANDLE или SQL_ERROR.

Связь приложений с базами данных

Получение идентификатора команды реализуется обычным способом с применением функции SQLAllocHandle:

```
SQLHSTMT hStmt = SQL_NULL_HSTMT;
rc = SQLAllocHandle(SQL_HANDLE_STMT, hDbc, fchStmt);
```

Для запуска команды на выполнение надо вызвать функцию SQLExecDirect:

```
rc = SQLExecDirect(hStmt, (unsigned char*)
"select ManagerID, Name, Password, Lastlogin, Rights from
managers", SOLENTS);
```

Связь приложений с базами данных

Создание приложений Web к базам данных Oracle в простейшем случае обеспечивает PL/SQL. Хранимые процедуры могут извлекать информацию из базы данных и форматировать ее в виде страниц HTML. Интерпретировать команды HTML позволяют стандартные пакеты htf и htp.

Связь приложений с базами данных

```
PROCEDURE book_list(setid IN NUMBER,cookie IN VARCHAR2
DEFAULT "0") IS
i INTEGER;
vbook   BOOK%ROWTYPE;
vset    BOOK_SET%ROWTYPE;
CURSOR cs_book IS
      SELECT * FROM book  WHERE book.setno=setid  ORDER BY
          book.num_in_list;
CURSOR cs_set IS
      SELECT * FROM book_set       WHERE   book_set.id=setid;
BEGIN
OPEN cs_set;
FETCH cs_set INTO vset;
CLOSE cs_set;
htp.htmlOpen;
htp.headOpen;
htp.title(vset.set_name);
htp.headClose;
htp.bodyOpen("/images/ppaper.jpg");
```

Связь приложений с базами данных

```
htp.htitle("Документация СУБД        ORACLE на русском языке:"
||htf.br||"комплект"   ||htf.italic(vset.set_name),     "2","center");
…
htp.br;
htp.hr;
htp.FormOpen (owa_util.get_owa_service_path
||"store.submit_books");
htp.tableOpen;
i := 1;
htp.tableRowOpen;
htp.tableData("");
htp.tableHeader("");
htp.tableHeader("Цена");
htp.tableHeader("Количество");
htp.tableRowClose;
…
htp.bodyClose;
htp.htmlClose;
END book_list;
```

Связь приложений с базами данных

Oracle WebServer представляет собой сервер HTTP с возможностью интеграции с базой данных. Oracle WebServer получает Uniform Resource Locator (URL) с помощью протокола HTTP от навигатора WWW и извлекает информацию, необходимую для ответа на запрос, из базы данных или файловой системы. Web Listener - обычный Web-сервер, который получает URL от клиента - навигатора WWW - и отвечает на запрос. При получении URL Web Listener определяет, требуется ли для выполнения запроса обращение к файловой системе, выполнение программы через интерфейс CGI или обращение к базе данных.

Связь приложений с базами данных

В последнем случае Listener передает запрос диспетчеру многопотокового сервера приложений Web Request Broker (WRB) и возвращается к состоянию ожидания новых запросов HTTP. Диспетчер WRB распределяет запрос между процессами, называемыми WRB Executable Engines (WRBX). Каждый из WRBX с помощью встроенного API обращается к специальным приложениям, т. н. WRB-картриджам.

Связь приложений с базами данных

WebServer 3.0 позволяет использовать следующие картриджи.

Картридж PL/SQL, позволяющий строить Web-страницы, основанные на информации, извлекаемой из базы данных. Динамические страницы создаются с помощью хранимых процедур PL/SQL, которые используют стандартный пакет функций и процедур, интерпретирующих команды языка HTML.

Связь приложений с базами данных

Картридж Java, представляющий собой интерпретатор языка Java, полностью интегрированный с базой данных Oracle.

Картридж ODBC позволяет иметь доступ к информации, хранящейся в базах данных различных типов через драйверы ODBC (Open DataBase Connectivity).

Картридж Perl дает возможность использования программ, написанных на языке Perl в архитектуре Web Request Broker.

WRB API позволяет разработчикам также создавать свои собственные картриджи, расширяющие возможности Oracle WebServer.

Связь приложений с базами данных

Альтернативные технологии для реализации ИС

- фактографические базы данных (Oracle DataBase Server 9i);
- WEB-сервер (Apache Tomcat 5.5.4) :
  - управление взаимодействием клиентов с АИС: Java 2 EE
  - формирование выходных данных для отображения в броузере: XSLT
  - формирование списков и отчетов в формате PDF: XSL:FO
  - Выходные данные: XHTML, CSS, JavaScript