Принципы построения распределенных баз данных

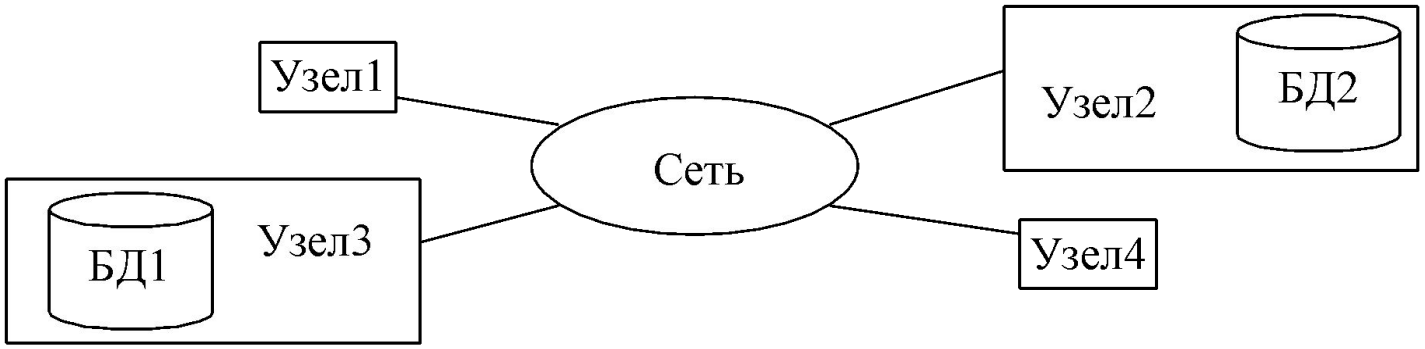
1. Методы поддержки распределенных данных
2. Основные понятия распределенных баз данных
3. Распределенные транзакции. Протокол двухфазной фиксации.

Общие принципы

Распределенная база данных (РБД) – это набор логически связанных между собой разделяемых данных, которые физически распределены по разных узлам компьютерной сети.

СУРБД – это программный комплекс (СУБД), предназначенный для управления РБД и позволяющий сделать распределенность прозрачной для конечного пользователя. **Прозрачность РБД** заключается в том, что с точки зрения конечного пользователя она должна вести себя точно также, как централизованная.

Логически единая БД разделяется на фрагменты, каждый из которых хранится на одном компьютере, а все компьютеры соединены линиями связи. Каждый из этих фрагментов работает под управлением своей СУБД.



Общие принципы

- В основе распределенных баз данных лежат две основные идеи:
- ✔ много организационно и физически распределенных пользователей, одновременно работающих с общей базой данной (пользователи с разными учетными записями, с разными полномочиями и задачами);
- ✔ логически и физически распределенные данные, составляющие и образующие единое взаимосогласованное целое - общую базу данных.

Основные принципы создания и функционирования распределенных баз данных

- Впервые задача об исследовании принципов построения и функционирования распределенных баз данных была поставлена К.Дейтом в рамках системы System R. Большую роль в исследовании принципов создания и функционирования распределенных баз данных внесли также и разработчики системы Ingres.

  Крис Дейт сформулировал основные принципы создания и функционирования распределенных баз данных.

Критерии распределенности (по К. Дейту)

- **Локальная автономность.** Локальные данные принадлежат локальным узлам и управляются администраторами локальных БД.

Локальные процессы в РБД остаются локальными. Все процессы на локальном узле контролируются только этим узлом.

Будучи *фрагментом* общего пространства данных, она, в то же время функционирует как полноценная локальная база данных; управление ею выполняется локально и независимо от других узлов системы.

- **Отсутствие опоры на центральный узел.**

В системе не должно быть узла, без которого система не может функционировать, т.е. не должно быть центральных служб.

Критерии распределенности (по К. Дейту)

В идеальной системе все узлы равноправны и независимы, а расположенные на них базы являются равноправными *поставщиками данных* в общее *пространство* данных. **База данных** на каждом из узлов самодостаточна - она включает полный собственный словарь данных и полностью защищена от несанкционированного доступа.

Критерии распределенности (по К. Дейту)

- **Непрерывное функционирование.**

  Удаление или добавление узла не должно требовать остановки системы в целом.

  Это качество можно трактовать как возможность непрерывного *доступа к данным* (известное "24 часа в сутки, семь дней в неделю") в рамках *DDB* вне зависимости от их расположения и вне зависимости от операций, выполняемых на локальных узлах. Это качество можно выразить лозунгом "данные доступны всегда, а операции над ними выполняются непрерывно".

Критерии распределенности (по К. Дейту)

- **Независимость от местоположения.**

  Пользователь должен получать доступ к любым данным в системе, независимо от того, являются эти данные локальными или удалёнными.

  Это свойство означает полную прозрачность расположения данных. Пользователь, обращающийся к *DDB*, ничего не должен знать о реальном, физическом размещении данных в узлах *информационной системы*. Все операции над данными выполняются без учета их местонахождения. *Транспортировка запросов* к базам данных осуществляется встроенными системными средствами.

Критерии распределенности (по К. Дейту)

- **Независимость от фрагментации.**

Доступ к данным не должен зависеть от наличия или отсутствия фрагментации и от типа фрагментации.

- **Независимость от репликации.**

Доступ к данным не должен зависеть от наличия или отсутствия реплик данных.

- **Прозрачность тиражирования.**

Тиражирование данных - это асинхронный процесс переноса изменений объектов исходной *базы данных* в базы, расположенные на других узлах распределенной системы. Данное свойство означает, что тиражирование возможно, прозрачно и достигается внутрисистемными средствами.

Критерии распределенности (по К. Дейту)

- **Обработка распределенных запросов.**

Система должна автоматически определять методы выполнения соединения (объединения) данных.

Это свойство *DDB* трактуется как возможность выполнения операций *выборки данных*, сформулированных в рамках обычного *запроса* на языке *SQL*. То есть операцию выборки из *DDB* можно сформулировать с помощью тех же языковых средств, что и операцию над локальной *базой данных*.

- **Обработка распределенных транзакций.**

Протокол обработки распределённой транзакции должен обеспечивать выполнение четырёх основных свойств транзакции:

- ✔ атомарность;
- ✔ согласованность;
- ✔ изолированность;
- ✔ продолжительность.

Свойства транзакций

- ***Свойство атомарности*** - транзакция должна быть выполнена в целом или не выполнена вовсе.

- ***Свойство согласованности*** **-** гарантирует, что по мере выполнения транзакции данные переходят из одного согласованного состояния в другое - транзакция не разрушает взаимной согласованности данных.

- ***Свойство изолированности*** означает, что конкурирующие за доступ к базе данных транзакции физически обрабатываются последовательно, изолированно друг от друга, но для пользователей это выглядит так, как будто они выполняются параллельно.

- ***Свойство долговечности*** : если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах.

Обработка распределенных транзакций

Это качество *DDB* можно трактовать как возможность выполнения операций обновления *распределенной базы данных* (*INSERT, UPDATE, DELETE*), не разрушающее целостность и согласованность данных. Эта цель достигается применением двухфазового или двухфазного протокола ***фиксации транзакций*** (two-*phase commit protocol*), ставшего фактическим стандартом обработки *распределенных транзакций*. Его применение гарантирует согласованное изменение данных на нескольких узлах в рамках распределенной (или, как ее еще называют, глобальной) транзакции.

Критерии распределенности (по К. Дейту)

- **Независимость от типа оборудования.**

  СУРБД должна функционировать на оборудовании с различными вычислительными платформами.

- **Независимость от операционной системы.**

  СУРБД должна функционировать под управлением различных ОС.

- **Независимость от сетевой архитектуры.**

  СУРБД должна быть способной функционировать в сетях с различной архитектурой, типами носителей и поддерживать любые сетевые протоколы.

- **Независимость от типа СУБД.**

  СУРБД должна быть способной функционировать поверх различных локальных СУБД, возможно, с различными моделями данных (требование гетерогенности).

Дейт, К., Дж. Введение в системы баз данных.

Издательский дом “Вильямс”, 2008. –630с.

Методы поддержки распределенных данных

Существуют различные методы поддержки распределенности:

1. **Фрагментация** – разбиение БД или таблицы на несколько частей и хранение этих частей на разных узлах РБД.
2. **Репликация** – создание и хранение копий одних и тех же данных на разных узлах РБД.
3. **Распределенные ограничения целостности** – ограничения, для проверки выполнения которых требуется обращение к другому узлу РБД.
4. **Распределенные запросы** – это запросы на чтение, обращающиеся более чем к одному узлу РБД.
5. **Распределенные транзакции** – команды на изменение данных, обращающиеся более чем к одному узлу РБД.

Фрагментация

Фрагментация – основной способ организации РБД.
Назначение: хранение данных на том узле, где они чаще используются.
Основные проблемы, которые при этом возникают:
– прозрачность написания запросов к данным;
– поддержка распределенных ограничений целостности.

Схема фрагментации отношения должна **удовлетворять трем условиям:**

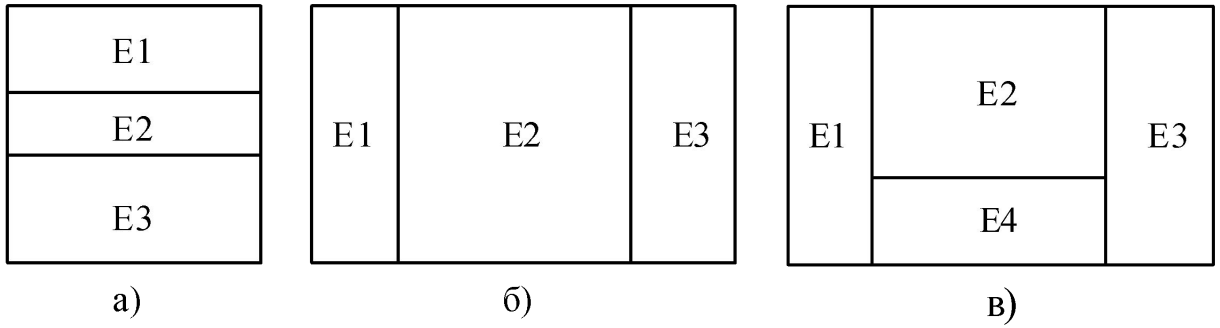
Полнота: если отношение R разбивается на фрагменты R1, R2,…, Rn, то
$U Ri = R$
(Каждый кортеж должен входить хотя бы в один фрагмент).

Восстановимость: должна существовать операция реляционной алгебры, позволяющая восстановить отношение R из его фрагментов. Это правило гарантирует сохранение функциональных зависимостей.

Непересекаемость: если элемент данных dj $\subset$ Ri, то он не должен присутствовать одновременно в других фрагментах. Исключение составляет первичный ключ при вертикальной фрагментации. Это правило гарантирует минимальную избыточность данных.

Фрагментация

Типы фрагментации:
а) горизонтальная;
б) вертикальная;
в) смешанная;
г) производная.



Производная фрагментация строится для подчиненного отношения на основе фрагментов родительского отношения. Например, для фрагментов отношения Emp (сотрудники) Ei подчиненное отношение "Дети" (Child), информацию о которых также целесообразно хранить в соответствующих узлах, имеет смысл разбить на три горизонтальных фрагмента:

C1 = C ►tabNo E1
C2 = C ►tabNo E2
C3 = C ►tabNo E3

где символ ► обозначает естественное полусоединение отношения C и фрагмента Ei (включает кортежи отношения C, которые могут быть соединены с соответствующим кортежем фрагмента Ei по значению внешнего ключа).

Фрагментация

- **Горизонтальная фрагментация** означает хранение строк одной таблицы на различных узлах (фактически, хранение строк одной логической таблицы в нескольких идентичных физических таблицах на различных узлах).
- **Вертикальная фрагментация** означает распределение столбцов логической таблицы по нескольким узлам.

Репликация данных

Репликация – это поддержание двух и более идентичных копий (реплик) данных на разных узлах РБД.

Реплика может включать всю базу данных (полная репликация), одно или несколько взаимосвязанных отношений или фрагмент отношения.

Достоинства репликации:

– повышение доступности и надежности данных;

– повышение локализации ссылок на реплицируемые данные.

Недостатки репликации:

– сложность поддержания идентичности реплик;

– увеличение объема памяти для хранения данных.

Поддержание идентичности реплик называется **распространение изменений** и реализуется **службой тиражирования**.

Служба тиражирования

Служба тиражирования должна выполнять следующие функции:

- ✔ Обеспечение масштабируемости, т.е. эффективной обработки больших и малых объемов данных.
- ✔ Преобразование типов и моделей данных (для гетерогенных РБД).
- ✔ Репликация объектов БД, например, индексов, триггеров и т.п.
- ✔ Инициализация вновь создаваемой реплики.
- ✔ Обеспечение возможности "подписаться" на существующие реплики, чтобы получать их в определенной периодичностью.

Для выполнения этих функций в языке, поддерживаемом СУБД, предусматривается наличие средств определения схемы репликации, механизма подписки и механизма инициализации реплик (создания и заполнения данными).

Распределенные запросы

Распределенным называется запрос, который обращается к двум и более узлам РБД, но не обновляет на них данные.

Запрашивающий узел должен определить, что в запросе идет обращение к данным на другом узле, выделить подзапрос к удаленному узлу и перенаправить его этому узлу.

Самой сложной проблемой выполнения распределенных запросов является **оптимизация**, т.е. поиск оптимального плана выполнения запроса. Информация, которая требуется для оптимизации запроса, распределена по узлам. Если выбрать центральный узел, который соберет эту информацию, построит оптимальный план и отправит его на выполнение, то теряется свойство локальной автономности.

Поэтому обычно распределенный запрос выполняется так: запрашивающий узел собирает все данные, полученные в результате выполнения подзапросов, у себя, и выполняет их соединение (или объединение), что может занять очень много времени.

Распределенные запросы. Пример

База данных "Агентство недвижимости", 2 филиала – в Лондоне и Глазго. Отношения:

Property (pNo, City, ...), 10 000 записей, хранится в Лондоне.

Renter (rNo, Max_price, ...), 100 000 записей, хранится в Глазго.

Viewing (pNo, rNo), 1 000 000 записей, хранится в Лондоне.

Время передачи = C_0 + (количество байт) / (скорость передачи), C_0 =1с

Запрос: получить список объектов в Абердине, которые осмотрены потенциальными покупателями, согласными заплатить не менее 200000.

```
select p.pNo
from property p, renter r, viewing v
where    p.pNo=v.pNo and r.rNo=v.rNo and
      p.City='Aberdine' and r.Max_price>=200000;
```

Распределенные запросы. Пример

Условия:

- ✔ скорость передачи 10000 б/с;
- ✔ задержка передачи – 1 с,
- ✔ все кортежи по 100 байт,
- ✔ существует 10 покупателей, согласных заплатить не менее 200 000,
- ✔ в Абердине было проведено 100 000 осмотров.

Проанализируем 6 стратегий выполнения этого запроса:

1. Переслать Renter в Лондон и выполнить обработку запроса там: 1+(100 000*100)/10 000 = **16,7 мин.**
2. Переслать Viewing и Property в Глазго и выполнить обработку запроса там: 2+((1000 000+10 000)*100)/10 000 = **2,8 ч**
3. Соединить Renter и Property в Лондоне и для каждого кортежа проверить покупателя: 100 000*(1+100/10 000) +1*100 000 = **2,3 дня**
4. Выбрать в Глазго нужных покупателей и проверить для каждого город: 10*(1+100/10000) +1*10 = **20 с**
5. Соединить Renter и Property в Лондоне, выполнить проекцию полей pNo и rNo и переслать её в Глазго: 1+(100 000*100)/10000 = **16,7 мин.**
6. Выбрать клиентов по Max_price и переслать в Лондон: 1+(10*100)/10000 = **1 с**

Распределенные ограничения целостности

Распределенные ограничения целостности возникают тогда, когда для проверки соблюдения какого-либо ограничения целостности системе необходимо обратиться к другому узлу.

Примеры:

1)База данных разделена на фрагменты таким образом, что родительская таблица находится на одном узле, а дочерняя, связанная с ней по внешнему ключу, – на другом. При добавлении записи в дочернюю таблицу система обратится к узлу, на котором расположена родительская таблица, для проверки наличия соответствующего значения ключа.

2)Разбиение одной таблицы на фрагменты и размещение этих фрагментов по разным узлам сети. Здесь будет необходима проверка соблюдения ограничений первичного ключа и уникальных ключей.

Распределенные транзакции

Распределенные транзакции обращаются к двум и более узлам и обновляют на них данные.

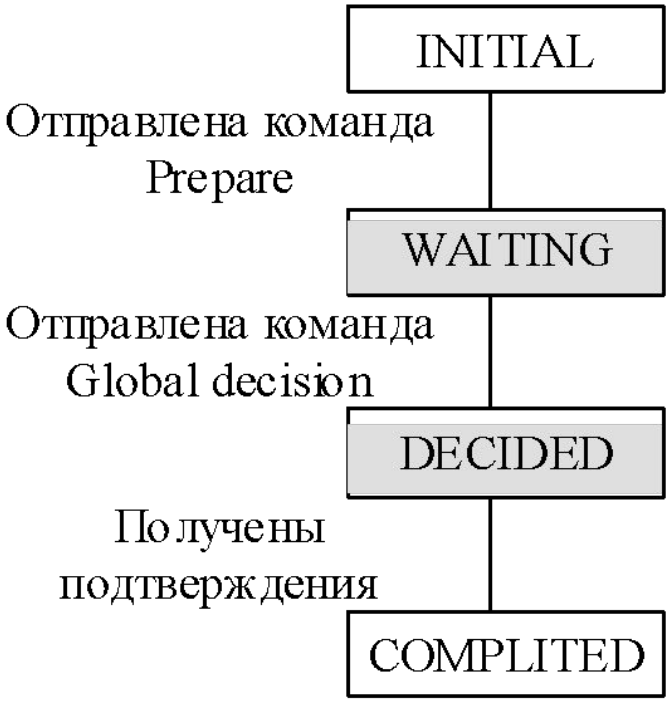
Основная проблема распределенных транзакций – соблюдение логической целостности данных. Транзакция на всех узлах должна завершиться одинаково: ***или фиксацией, или откатом.***

Выполнение распределенных транзакций осуществляется с помощью специального алгоритма, который называется **двухфазная фиксация** *(2ФФ).*

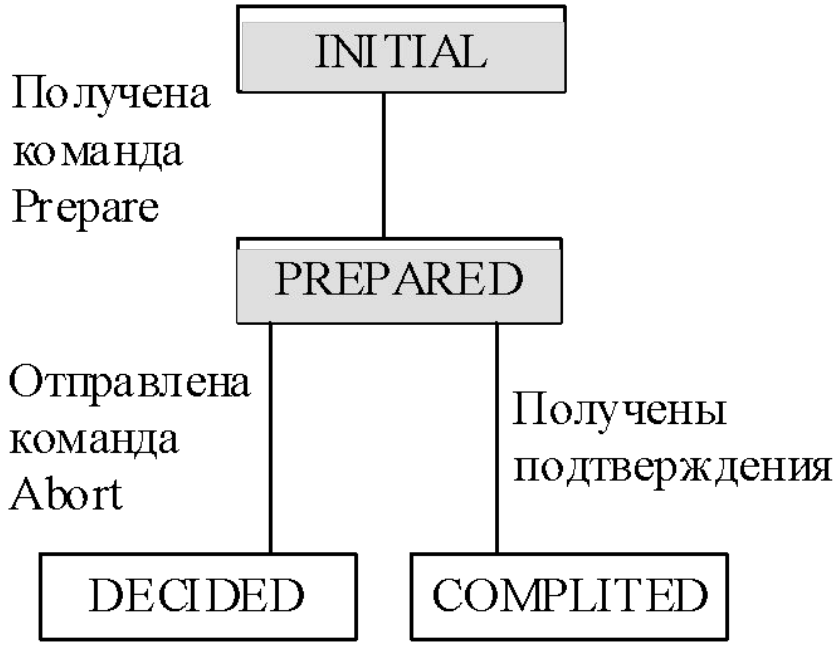
Координатор транзакции – узел, который контролирует выполнение этого протокола (обычно, тот узел, который инициирует данную транзакцию).

Остальные узлы, на которых выполняется транзакция, называются ***участниками транзакции***.

Протокол двухфазной фиксации



а) диаграмма состояний координатора



б) диаграмма состояний участника

Действия координатора транзакции

Координатор выполняет протокол 2ФФ по следующему алгоритму:

I. Фаза 1 (голосование).

Занести запись *begin_commit* в *системный журнал* и обеспечить ее перенос из буфера в ОП на ВЗУ. Отправить всем участникам команду PREPARE (приготовиться).

Ожидать ответов всех участников в пределах установленного тайм-аута.

Действия координатора транзакции

II. Фаза 2 (принятие решения).

- При поступлении сообщения ABORT: занести в *системный журнал* запись *abort* и обеспечить ее перенос из буфера в ОП на ВЗУ; отправить всем участникам сообщение GLOBAL_ABORT и ждать ответов участников (тайм-аут).
- Если участник не отвечает в течение установленного тайм-аута, координатор считает, что данный участник откатит свою часть транзакции и запускает протокол ликвидации.

Действия координатора транзакции

- Если все участники прислали COMMIT, поместить в *системный журнал* запись *commit* и обеспечить ее перенос из буфера в ОП на ВЗУ. Отправить всем участникам сообщение GLOBAL_COMMIT и ждать ответов всех участников.
- После поступления подтверждений о фиксации от всех участников: поместить в *системный журнал* запись *end_transaction* и обеспечить ее перенос из буфера в ОП на ВЗУ.
- Если некоторые узлы не прислали подтверждения фиксации, координатор заново направляет им сообщения о принятом решении и поступает по этой схеме до получения всех требуемых подтверждений.

Действия участника транзакции

Участник выполняет протокол 2ФФ по следующему алгоритму:

1. При получении команды PREPARE, если он готов зафиксировать свою часть транзакции, он помещает запись *ready_commit(готов завершить) в файл журнала транзакций* и отправляет координатору сообщение READY_COMMIT. Если он не может зафиксировать свою часть транзакции, он помещает запись *abort в файл журнала транзакций*, отправляет координатору сообщение ABORT и откатывает свою часть транзакции (не дожидаясь общего сигнала GLOBAL_ABORT).

Действия участника транзакции

- Если участник отправил координатору сообщение READY_COMMIT, то он ожидает ответа координатора в пределах установленного тайм-аута.
- При получении GLOBAL_ABORT участник помещает запись *abort в файл журнала транзакций*, откатывает свою часть транзакции и отправляет координатору подтверждение отката.

Действия участника транзакции

- При получении GLOBAL_COMMIT участник помещает запись *commit* в файл журнала транзакций, фиксирует свою часть транзакции и отправляет координатору подтверждение фиксации.
- Если в течение установленного тайм-аута участник не получает сообщения от координатора, он откатывает свою часть транзакции.

Протоколы ликвидации

Протокол ликвидации для координатора:

1. Тайм-аут в состоянии WAITING (ожидание): координатор не может зафиксировать транзакцию, потому что не получены все подтверждения от участников о фиксации. Ликвидация заключается в откате транзакции.
2. Тайм-аут в состоянии DECIDED(решение): координатор повторно рассылает сведения и принятом глобальном решении и ждет ответов от участников.

Протоколы ликвидации

- Простейший протокол ликвидации для участника заключается в блокировании процесса до тех пор, пока сеанс связи с координатором не будет восстановлен.

  Но в целях повышения производительности (и автономности) узлов могут быть предприняты и другие действия:
- Тайм-аут в состоянии INITIAL: участник не может сообщить о своем решении координатору и не может зафиксировать транзакцию. Но может откатить свою часть транзакции. Если он позднее получит команду PREPARE, он может проигнорировать ее или отправить координатору сообщение ABORT.

Протоколы ликвидации

- Тайм-аут в состоянии PREPARED: участник уже известил координатора о решении COMMIT, то он не может его изменить. Участник оказывается заблокированным.

Протоколы восстановления

Действия, которые выполняются на отказавшем узле после его перезагрузки, называются *протоколом восстановления*. Они зависят от того, в каком состоянии находился узел, когда произошел сбой, и какую роль выполнял этот узел в момент отказа: координатора или участника.

Протоколы восстановления

- **При отказе координатора:**
- ✔ В состоянии INITIAL: процедура 2ФФ еще не запускалась, поэтому после перезагрузки следует ее запустить.
- ✔ В состоянии WAITING: координатор уже направил команду PREPARE, но еще не получил всех ответов и не получил ни одного сообщения ABORT. В этом случае он перезапускает процедуру 2ФФ.
- ✔ В состоянии DECIDED: координатор уже направил участникам глобальное решение. Если после перезапуска он получит все подтверждения, то транзакция считается успешно зафиксированной. В противном случае он должен прибегнуть к протоколу ликвидации.

Протоколы восстановления

При отказе участника цель протокола восстановления – гарантировать, что после восстановления узел выполнит в отношении транзакции то же действие, которое выполнили другие участники, и сделает это независимо от координатора, т.е. по возможности без дополнительных подтверждений.

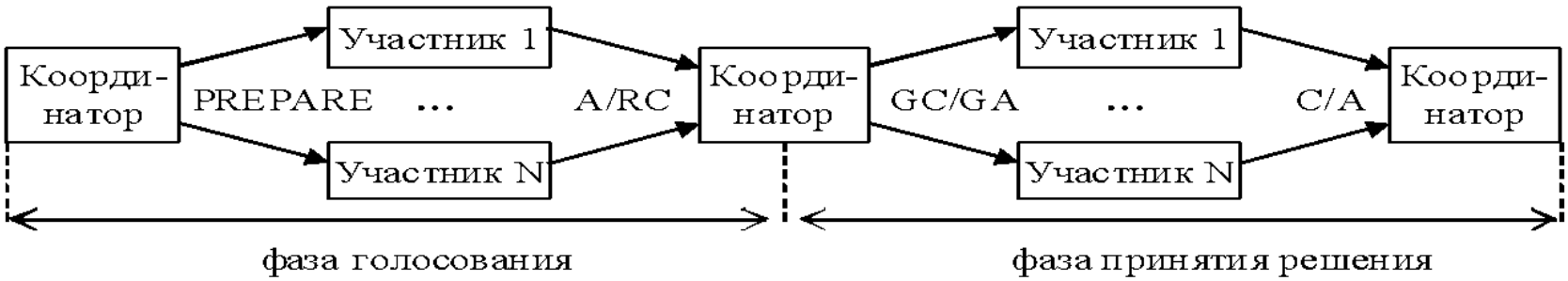
Протоколы восстановления

- Рассмотрим три возможных момента возникновения отказа:

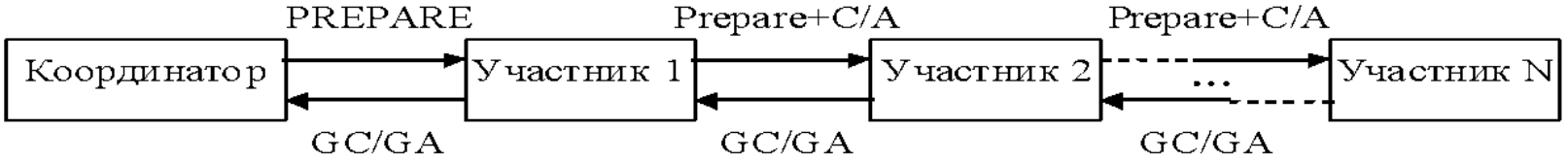
- ✔ В состоянии INITIAL: участник еще не успел сообщить о своем решении координатору, поэтому он может выполнить откат, т.к. координатор не мог принять решение о глобальной фиксации транзакции без голоса этого участника.
- ✔ В состоянии PREPARED: участник уже направил сведения о своем решении координатору, поэтому он должен запустить свой протокол ликвидации.
- ✔ В состоянии ABORTED/COMMITED: участник уже завершил обработку своей части транзакции, поэтому никаких дополнительных действий не требуется.

Реализация протокола 2ФФ

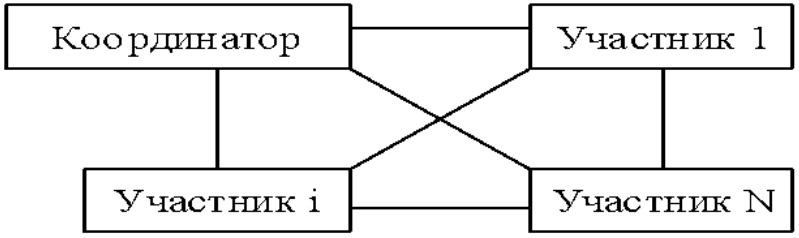
а) традиционная реализация протокола 2ФФ



б) линейная реализация протокола 2ФФ



в) распределенная реализация протокола 2ФФ



Поддержка распределенности в Oracle

1. Прозрачность распределенности.
2. Каждая часть данных, хранимых на одном компьютере в сети, оформлена как самостоятельная база данных.
3. Одна логическая таблица может быть распределена по разным узлам в сети.
4. Каждый сервер БД в системе распределенной базы данных (РБД) управляет доступом к своей локальной БД; за управление системой в целом не отвечает ни один сервер.
5. Поддержка целостности и согласованности данных осуществляется на уровне взаимодействия между серверами, что является расширением модели клиент-сервер (Distributed Oracle).
6. Связь осуществляется по сети с помощью программного средства Oracle – дополнительной утилиты Net8.
7. Распределенная БД может быть неоднородной, при этом один или несколько узлов должны быть Oracle-серверами, а связь с серверами других типов осуществляется через открытый шлюз (дополнительный программный пакет Open Gateways).

Связь в распределенной БД Oracle

Обращение к сервисам базы данных (серверу БД, очереди печати, серверу электронной почты и т.д.) происходит по уникальному имени (global name). Для БД оно состоит из основного имени $ORACLE_SID, назначаемого ей при создании (длиной не более 8-и символов) и сетевого домена БД, например:

sales.parts@east.compworld

– обращение к таблице PARTS базы данных SALES, расположенной на сервере east.compworld.

Для получения доступа к удаленной БД нужно установить связь с этой БД с помощью специальной команды языка SQL – LINK. При формировании связи могут учитываться учетные сведения пользователя для обеспечения безопасности данных, но это требует дополнительных усилий для распространения учетных сведений пользователя в сети: во-первых, нужно создать учетный раздел пользователя на удаленном сервере; во-вторых, пакеты регистрации в сети желательно шифровать, т.к. сеть не защищена от доступа посторонних лиц.

Связи в распределенной БД Oracle

Примеры.

Локальная база данных – HQ.ACME.COM.

Удаленная база данных – SALES.ACME.COM.

Создание общей связи баз данных к удаленной базе данных SALES:

CREATE PUBLIC DATABASE LINK sales.acme.com USING 'dbstring';

Создание личной связи баз данных для создателя этой связи:

CREATE DATABASE LINK sales CONNECT TO scott IDENTIFIED BY tiger;

Фраза CONNECT TO специфицирована явно. При установлении сессии в удаленной базе данных через эту связь баз данных будет использоваться идентификация SCOTT/TIGER.

Фраза USING опущена. Поэтому, когда используется эта личная связь баз данных, должна существовать одноименная общая или сетевая связь баз данных, содержащая строку базы данных для установления соединения с удаленной базой данных.

Работа в распределенной БД

Различает следующие виды обработки данных в РБД:

удаленный запрос – это оператор SELECT, считывающий информацию из одной или нескольких таблиц на одном из удаленных узлов сети;

распределенный запрос – это оператор SELECT, считывающий информацию из одной или нескольких удаленных таблиц, которые расположены на разных узлах;

удаленное обновление – это модификация, затрагивающая таблицы из одного удаленного узла;

распределенное обновление – это модификация данных двух или более узлов сети;

вызовы удаленных процедур – запуск процедуры, находящейся на удаленном сервере;

удаленная транзакция – это транзакция, содержащая хотя бы один удаленный запрос и относящаяся к одному удаленному узлу;

распределенная транзакция – это транзакция, содержащая одно или несколько распределенных обновлений или удаленные транзакции для разных серверов. За выполнение распределенных транзакций отвечает механизм двухфазной фиксации.

Моментальные снимки в Oracle

Oracle поддерживает два типа тиражирования:

- ✔ базовое – копия обеспечивает доступ "только для чтения".
- ✔ усовершенствованное – приложения могут считывать и обновлять тиражируемые копии таблиц по всей системе (поддерживается специальными средствами СУБД – **Replication Option**).

Базовое тиражирование осуществляется (после установления связи с удаленной БД) с помощью создания моментальных снимков (snapshot), например:

```
CREATE SNAPSHOT sales.parts AS
        SELECT * FROM sales.parts@central.compworld;
```

Моментальные снимки бывают:

- ✔ простые – создаются по однотабличному запросу SELECT, содержащему простые условия отбора.
- ✔ сложные – создаются по запросам, содержащим сложные условия отбора, фразы group by, having, обращающимся к двум и более таблицам и проч.

Моментальные снимки в Oracle

Примеры:

Моментальный снимок, основой которого является запрос

```
select * from employee@hr_link;
```

является простым.

Моментальный снимок, основанный на запросе

```
select dept, max(salary)
    from employee@hr_link
    group by dept;
```

сложный, так как в нем используются функции группирования.

С помощью моментального снимка в локальной базе данных будет создано несколько объектов, поэтому пользователь, создающий моментальный снимок, должен иметь привилегии CREATE TABLE, CREATE VIEW и CREATE INDEX.

Моментальные снимки в Oracle

Синтаксис создания моментального снимка:

create snapshot [*имя_схемы*.]*имя_снимка*
[{ pctfree *целое* | pctused *целое* | initrans *целое* |
maxtrans *целое* | tablespace *имя_табличной_области* |
storage *размер_памяти* }]
[cluster *имя_кластера* (*имя_столбца*[,…])]
[using index]
[{ pctfree *целое* | pctused *целое* | initrans *целое* |
maxtrans *целое* | tablespace *имя_табличной_области* |
storage *размер_памяти* }]
[refresh [{ fast | complete | force }]
[start with *дата_1*] [next *дата_2*]]
[for update]
as *запрос*;

Моментальные снимки в Oracle

Пример создания МС на локальном сервере:

```
create snapshot emp_dept_count
        pctfree 5
        tablespace snap
        storage (initial 100k next 100k pctincrease 0)
        refresh complete
        start with sysdate
        next sysdate+7
        as select deptno, count(*) dept_count
                from employee@hr_link
                group by deptno;
```

Моментальные снимки в Oracle

При создании моментального снимка в локальной базе данных создается:

- ✔ **таблица** для хранения записей, получаемых в результате выполнения запроса моментального снимка (с именем *SNAP$_имя_моментального_снимка)*;
- ✔ **представление** этой таблицы "только для чтения", называемое в соответствии с именем моментального снимка;
- ✔ **представление**, называемое MVIEW$_имя_моментального_снимка – для обращения к удаленной основной таблице (или таблицам). Это представление будет использоваться во время регенерации.

Для модификации снимка, например, с целью установки частоты автоматического изменения в 1 час можно воспользоваться командой ALTER SNAPSHOT:

```
alter snapshot emp_dept_count refresh complete
        start with sysdate next sysdate + 1/24;
```

Для удаления моментальных снимков применяется команда drop snapshot:

```
drop snapshot emp_dept_count;
```

Регенерация моментальных снимков Oracle

Возможны два варианта:

1. REFRESH FAST (**быстрая регенерация**).
2. REFRESH COMPLETE (**полная регенерация**).

Режим регенерации	Описание
COMPLETE	Таблицы моментального снимка полностью восстанавливаются с помощью его запроса и основных таблиц при каждой регенерации
FAST	Если применяется простой моментальный снимок, то для посылки только тех изменений, которые внесены в его таблицу, можно использовать журнал моментальных снимков
FORCE	Значение по умолчанию. Если это возможно, выполняется быстрая (FAST) регенерация, если нет – полная (COMPLETE) регенерация

Регенерация моментальных снимков Oracle

Для быстрой регенерации необходим **журнал моментальных снимков** (snapshot log) – это таблица, обеспечивающая регистрацию в моментальном снимке изменений, происшедших в основной таблице. Имя журнала (таблицы) – MLOG$_имя_таблицы.

Команда CREATE SNAPSHOT LOG. Пример:

```
create snapshot log on employee
        tablespace data
        storage (initial 10k next 10k pctincrease 0);
```

Изменения в журнал моментальных снимков попадают с помощью триггера AFTER типа FOR EACH ROW, который называется TLOG$_имя_таблицы.

В журнале моментальных снимков данные находятся очень непродолжительное время: записи вводятся в журнал моментальных снимков, используются во время регенерации, а затем удаляются из журнала автоматически.

Усовершенствованное тиражирование Oracle

Производится с помощью двух средств Oracle:

1. Многоабонентского тиражирования.
2. Узлов обновляемых моментальных снимков.

Распространение изменений:

1. на уровне строк: сервер записывает изменения, сделанные каждой DML-транзакцией, и рассылает эти изменения в удаленные узлы.
2. путем процедурного тиражирования: тиражируется вызов удаленной процедуры, выполняющей в удаленном узле те же изменения, что и в вызывающем.

Различают **асинхронное** и **синхронное** распространение изменений.

Внесение изменений в тиражируемые данные происходит в несколько этапов:

- ✔ локальный узел вносит изменения в свою копию данных (ОМС);
- ✔ локальный узел запускает отложенную транзакцию на основном узле;
- ✔ через некоторое время локальный узел выполняет быструю (или полную) регенерацию локальной копии данных, после чего приложение всегда может проверить, выполнена ли инициированная им транзакция. Если она не выполнена, то происходит рестарт транзакции и все повторяется.

Требования к распределенной базе данных

- **РБД должна обладать (требования):**
- ✔ локальными и глобальными (распределенными) средствами *доступа к данным*(*СУБД*);
- ✔ единообразной логикой *прикладных программ* во всех клиентских приложениях узлов сети;
- ✔ малым временем реакции на запросы пользователей;
- ✔ надежностью, исключающей разрушения *целостности системы* в случае выхода из строя ее отдельных компонент(узлов);
- ✔ открытостью, позволяющей наращивать объем *локальных БД* и добавлять новые клиентские приложения;

Требования к распределенной базе данных

- развитой системой управления *резервным копированием* и восстановления данных в случае сбоев;
- защищенностью, следящей за соблюдением *привилегий доступа* к данным;
- высокой эффективностью, за счет выбора оптимальных алгоритмов использования сетевых ресурсов;
- развитым репликационным механизмом, позволяющим размещать обновленные копии данных в сети оптимальным образом.

Принципы построения РБД

- *Минимизация* интенсивности обмена данными (сетевого трафика);
- *Оптимальное* размещение серверных и клиентских *приложений* в *сети;*
- *Декомпозиция* данных на часто и редко используемые *сегменты* (для правильной настройки *репликации* - размещение наиболее часто используемых данных на *АРМ* конечных пользователей);
- Периодическое сохранение копий данных и выполнение действий по *поддержке целостности* распределенной *информационной системы*.

Критерии построения РБД

- Всесторонний *анализ* информационных потребностей *предметной области* с выявлением объемов хранимых данных, их сложности, достоверности, взаимосвязанности.
- Моделирование предполагаемого сетевого трафика при работе РБД с различными моделями *репликации* данных.
- *Кластеризация элементов данных* и программ их обработки. Цель- добиться максимальной автономности и слабосвязанности *кластеров*.
- Привязка *кластеров* данных к *вероятным* пользователям или *АРМ*.
- Поддержка эталонной копии данных и ограничение репликационного *механизма*.
- Разработка и реализация правил приведения локальных и центральной *БД* в *непротиворечивое* состояние.

Компьютерные сети

Основное средством передачи данных в ИС - компьютерные сети, подразделяемые на:

- ✔ низкоскоростные;
- ✔ среднескоростные;
- ✔ высокоскоростные.

В сети с используется передача данных по:

- ✔ коммутируемым каналам связи;
- ✔ специально выделенным каналам связи.

Компьютерная сеть **-** совокупность взаимосвязанных через каналы передачи данных компьютеров, обеспечивающих пользователей средствами обмена информацией и коллективного использования аппаратных, программных и информационных ресурсов сети.

Локальные сети

По степени территориальной удаленности компьютерные сети классифицируются на:

- ✔ локальные;
- ✔ распределенные;
- ✔ глобальные.

Локальные сети ЭВМ связывают пользователей одной организации, расположенных в одном или нескольких близлежащих зданиях и удаленных друг от друга на расстояние не больше 10 км. Локальные сети обслуживают до 80%-90% потребности в передаче информации и только 10-20% требуют своего обслуживания региональной или глобальной сетью. Локальные сети могут иметь любую структуру, но чаще всего компьютеры в локальной сети связаны единым высокоскоростным каналом передачи данных, который является собственностью организации.

Региональные и глобальные сети

Региональные сети объединяют пользователей города, области, небольших стран и в качестве связи используют телефонные линии. Расстояние между узлами сети составляет **10-100 км.**

Глобальные сети объединяют пользователей, расположенных по всему миру, используют спутниковые каналы связи, позволяющие соединить узлы сети, находящиеся на расстоянии **10-15 тыс. км** друг от друга.

По способу установления соединений между пользователями (абонентами) сети делятся на:

- сети с коммутацией каналов, характеризующиеся установлением прямой связи с абонентом на некоторое время в пределах общей очереди;

- сети с коммутацией сообщений, которые характеризуются наличием узлов коммутации сообщений. Для таких узлов обеспечиваются технические средства получения и хранения сообщений. Задача ЭВМ, используемых для этих целей, - получить сообщение, запомнить его и в случае освобождения канала связи с абонентом по определенному адресу передать это сообщение;

- сети с коммутацией пакетов, позволяющие длинное сообщение на передающем пункте разбить на пакеты сообщений с заголовком, адресом и контрольным числом, а на принимающем пункте – сборку сообщения по идентификатору.

Основные понятия распределенных баз данных

- ***Сервером*** - называется процесс, обслуживающий информационные потребности клиента.
- ***Сервер***, обеспечивающий поиск и обновление в базе данных называется сервером базы данных.
- ***Сервер баз данных*** - СУБД, основанная на архитектуре «клиент-сервер».
- Сервер, обеспечивающий некоторую обработку данных, называется ***сервером приложений***.

Клиент в клиент-серверных ИС

- Приложение, посылающее запрос на обслуживание сервером, является ***клиентом.***
  - Задача клиента в клиент-серверных ИС:

1). инициирование связи с сервером;

2). определение вида запроса на обслуживание;

3). получение результата обслуживания;

4). подтверждение окончания обслуживания.

Основные понятия распределенных баз данных

- Если клиент и сервер располагаются на разных узлах локальной или глобальной вычислительной сети, то клиент-серверная архитектура является распределенной.
- ***SQL-сервер*** – собирательный термин, относящийся ко всем серверам баз данных, основанных на SQL.

Наиболее популярные SQL-серверы

СУБД	Производитель
Oracle	Oracle Corp
MS SQL Server	Microsoft
Informix	Informix
Sybase	Sybase
DB2	IBM

Свойства серверных СУБД

- реализация для различных платформ (UNIX, Windows, Linux);
- наличие административных утилит;
- резервное копирование и восстановление данных;
- обслуживание репликаций;
- параллельная обработка данных в многопроцессорных системах;
- поддержка OLAP и создания хранилища данных (OLAP-On-line Analytic Processing – оперативная аналитическая обработка данных);

- распределенные запросы и транзакции;
- использование средств проектирования БД (универсальных или ориентированных на конкретную СУБД);
- поддержка доступа к данным через Интернет.

Литература к лекции
(Интернет-ресурсы в свободном доступе)

Кузнецов С.Д. **Основы современных баз данных.** // Центр Информационных Технологий, http://www.citforum.ru/database/osbd/contents.shtml

Пушников А.Ю. **Введение в системы управления базами данных.** // Центр Информационных Технологий, http://www.citforum.ru/database/dblearn/index.shtml

Transact-SQL Reference. // Microsoft SQL Server 2000 Books Online. Microsoft Corp, 2000.