Управление распределенными данными

С управлением данными в распределенных системах связаны следующие две группы проблем:

- поддержка соответствия БД вносимым изменениям,
- обеспечение совместного доступа нескольких пользователей к общим данным.

Поддержка соответствия БД вносимым изменениям

В современных распределенных системах информация может храниться *централизовано* или *децентрализовано*.

Существуют две основные технологии децентрализованного управления БД: распределенных БД (Distributed Database) и тиражирования, или репликации, БД (Data Replication).

Распределенная БД состоит из нескольких фрагментов, размещенных на разных узлах сети и, возможно, управляемых разными СУБД. С точки зрения программ и пользователей, обращающихся к распределенной БД, последняя воспринимается как единая локальная БД. Пример - распределенный библиотечный каталог.

Для обеспечения корректного доступа к распределенной БД в современных системах чаще всего применяется протокол (метод) *двухфазной фиксации транзакций* (two-phase commit). Суть этого метода состоит в двухэтапной синхронизации выполняемых изменений на всех задействованных узлах.

- В узлах сети производятся изменения (пока обратимые) в их БД, о чем посылаются уведомления компоненту системы, управляющему обработкой распределенных транзакций.

- Получив от всех узлов сообщения о правильности выполнения операций (что свидетельствует об отсутствии сбоев и отказов аппаратно-программного обеспечения), управляющий компонент выдает всем узлам команду фиксации изменений. После этого транзакция считается завершенной, а ее результат - необратимым.

Достоинства модели распределенной БД:

- пользователи всех узлов (при исправных коммуникационных средствах) получают информацию с учетом всех последних изменений
- экономное использование внешней памяти компьютеров, что позволяет организовывать БД больших объемов.

Недостатки модели распределенной БД:

- жесткие требования к производительности и надежности каналов связи
- большие затраты коммуникационных и вычислительных ресурсов из-за их связывания на все время выполнения транзакций
- при интенсивных обращениях к распределенной БД, большом числе взаимодействующих узлов, низкоскоростных и ненадежных каналах связи обработка запросов по этой схеме становится практически невозможной.

Модель тиражирования данных, в отличие от технологии распределенных БД, предполагает дублирование данных (создание точных копий) в узлах сети. Данные всегда обрабатываются как обычные локальные. Поддержку идентичности копий друг другу в асинхронном режиме обеспечивает компонент системы, называемый *репликатором (replicator)*. При этом между узлами сети могут передаваться как отдельные изменения, так и группы изменений. В течение некоторого времени копии БД могут отличаться друг от друга.

Достоинства модели тиражирования БД:

- более высокая скорость доступа к данным
- существенное уменьшение передаваемого по каналам связи потока информации
- повышение надежности механизмов доступа к распределенным данным

Недостатки модели тиражирования БД:

- на некотором интервале времени возможно «расхождение» копий БД. Если отмеченный недостаток некритичен для прикладных задач, то предпочтительно иметь схему с тиражированием БД.

Доступ к общим данным

При обслуживании обращений к общим данным средства управления БД должны обеспечивать по крайней мере два основных метода доступа: *монопольный* и *коллективный*.

Монопольный доступ обычно используется в двух случаях:

- когда требуется исключить доступ к объектам со стороны других пользователей (например, при работе с конфиденциальной информацией);
- когда производятся ответственные операции с БД, не допускающие других действий, например, изменение структуры БД.

Коллективный доступ возможен, например, при одновременном просмотре таблиц. Попытки получить монопольный доступ к объектам коллективного доступа должны быть пресечены (например, в ситуации когда одни или несколько пользователей просматривают таблицу, а другой пользователь собирается удалить эту же таблицу).

Для организации коллективного доступа в СУБД применяется механизм блокировок. Суть блокировки состоит в том, что на время выполнения какой- либо операции в БД доступ к используемому объекту со стороны других потребителей временно запрещается или ограничивается. Например, при копировании таблицы она блокируется от изменения, хотя и разрешено просматривать ее содержимое.

Выделим четыре вида блокировок, перечисленные в порядке убывания строгости ограничений на возможные действия:

- полная блокировка;
- блокировка от записи;
- предохраняющая блокировка от записи;
- предохраняющая полная блокировка.

Полная блокировка: полное запрещение всяких операций над основными объектами (таблицами, отчетами и экранными формами). Обычно применяется при изменении структуры таблицы.

Блокировка от записи: накладывается в случаях, когда можно использовать таблицу, по без изменения ее структуры или содержимого. Применяется, например, при выполнении операции слияния данных из двух таблиц.

Предохраняющая блокировка от записи: предохраняет объект от наложения на него со стороны других операций полноты блокировки, либо блокировки от записи. Этот вид блокировки позволяет тому, кто раньше «захватил» объект, успешно завершить модификацию объекта. Применяется в режиме совместного редактирования таблицы несколькими пользователями.

Предохраняющая полная блокировка: предохраняет объект от наложения на него со стороны других операций только полной блокировки. Обеспечивает максимальный уровень совместного использования объектов. Может использоваться, например, для обеспечения одновременного просмотра несколькими пользователями одной таблицы.

В отношении перечисленных выше четырех блокировок действуют следующие правила совмещения:

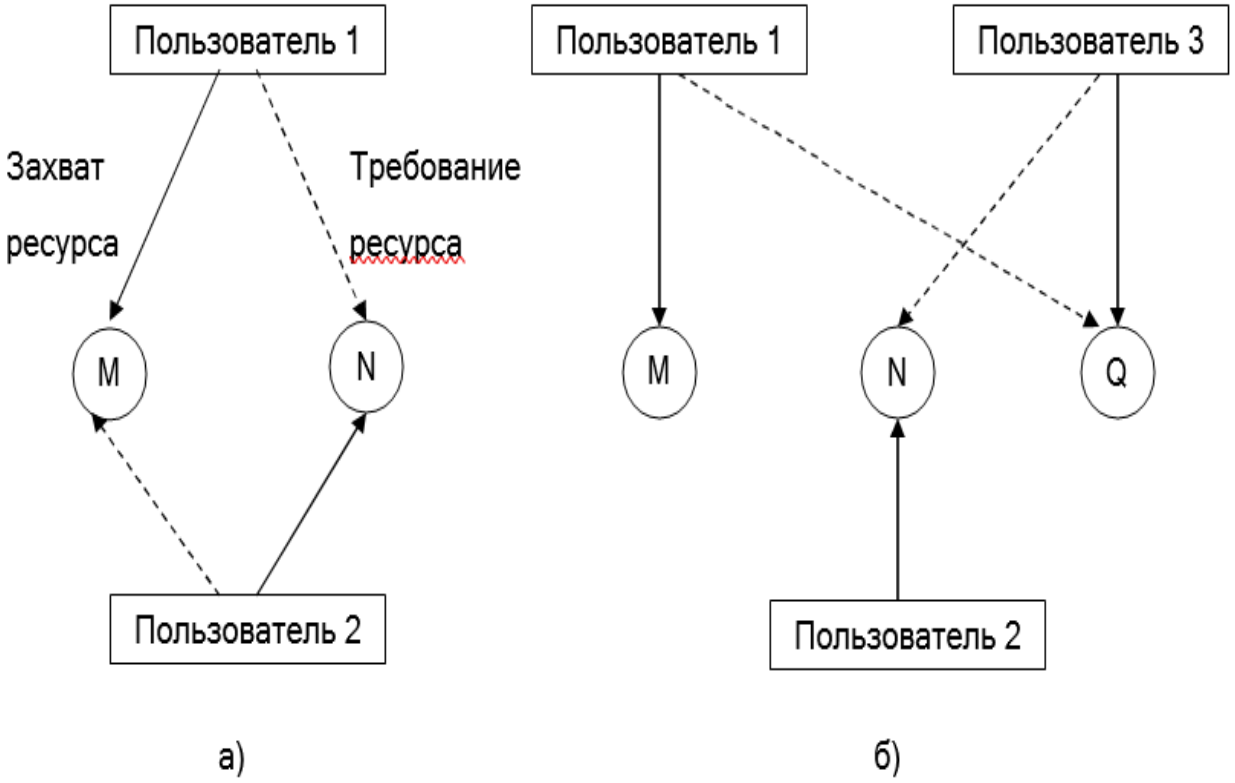
- при наличии полной блокировки над объектом нельзя производить операции, приводящие хотя бы к одному из видов блокировок (полная блокировка несовместима ни с какой другой блокировкой);
- блокировка от записи совместима с аналогичной блокировкой и предохраняющей полной блокировкой;
- предохраняющая блокировка от записи совместима с обеими видами предохраняющих блокировок;
- предохраняющая полная блокировка совместима со всеми блокировками, кроме полной.

Обычно в СУБД каждой из выполняемых с БД операций соответствует определенный вид блокировки, которую эта операция накладывает на объект.

Тупики

Существует два основных вида тупиков:

взаимные (deadlock) и *односторонние (livelock)*.



Примеры взаимных тупиков в распределенных БД

Простейшим случаем ***взаимного тупика*** является ситуация, когда каждый из двух пользователей стремится захватить данные, уже захваченные другим пользователем (рисунок 1а). В этой ситуации пользователь 1 ждет освобождения ресурса N, в то время как пользователь 2 ожидает освобождения от захвата ресурса M. Следовательно, никто из них не может продолжить работу. В действительности могут возникать и более сложные ситуации, когда выполняются обращения трех и более пользователей к нескольким ресурсам. ***Односторонний тупик*** возникает в случае требования получить монопольный доступ к некоторому ресурсу, как только он станет доступным и невозможности удовлетворить это требование.

Транзакции

Транзакцией называется последовательность операций, производимых над БД и переводящих базу данных из одного непротиворечивого (согласованного) состояния в другое непротиворечивое (согласованное) состояние. Транзакция рассматривается как некоторое неделимое действие над БД, осмысленное с точки зрения пользователя. В то же время это логическая единица работы системы.

Рассмотрим пример, связанный с принятием заказа в фирме на изготовление компьютера. Компьютер состоит из комплектующих, которые сразу резервируются за данным заказом в момент его формирования. Тогда транзакцией будет вся последовательность операций, включающая следующие этапы:

1) ввод нового заказа со всеми реквизитами заказчика;

2) изменение состояния для всех выбранных комплектующих на складе на «занято» с привязкой к определенному заказу;

3) подсчет стоимости заказа с формированием платежного документа;

4) включение нового заказа в производство.

С точки зрения работника, это единая последовательность операций; если она будет прервана, то БД потеряет свое целостное состояние.

Существуют различные модели транзакций, которые могут быть классифицированы на основании различных свойств, включающих структуру транзакции, параллельность внутри транзакции, продолжительность и т.д.

В настоящий момент выделяют следующие типы транзакций: ***классические (традиционные) транзакции***, ***цепочечные транзакции*** и ***вложенные транзакции***.

Традиционные транзакции характеризуются четырьмя основными свойствами: атомарности, согласованности, изолированности, долговечности (прочности) — ACID (Atomicity, Consistency, Isolation, Durability). Иногда традиционные транзакции называют ACID-транзакциями.

Упомянутые выше свойства означают следующее:

1. Свойство атомарности (Atomicity) выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.

2. Свойство согласованности (Consistency) гарантирует, что по мере выполнения транзакций данные переходят из одного согласованного состояния в другое — транзакция не разрушает взаимной согласованности данных.

3. Свойство изолированности (Isolation) 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.

4. Свойство долговечности (Durability) трактуется следующим образом: если транзакция завершена успешно, то те изменения в данных, которые были ею произведены, не могут быть потеряны ни при каких обстоятельствах (даже в случае последующих ошибок).

Возможны два варианта завершения транзакции. Если все операторы выполнены успешно и в процессе выполнения транзакции не произошло никаких сбоев программного или аппаратного обеспечения, транзакция фиксируется.

Фиксация транзакции — это действие, обеспечивающее запись на диск изменений в базе данных, которые были сделаны в процессе выполнения транзакции.

Если в процессе выполнения транзакции случилось нечто такое, что делает невозможным ее нормальное завершение, база данных должна быть возвращена в исходное состояние.

Откат транзакции — это действие, обеспечивающее аннулирование всех изменений данных, которые были сделаны операторами SQL в теле текущей незавершенной транзакции.

В стандарте ANSI/ISO SQL определены модель транзакций и функции операторов COMMIT и ROLLBACK. Стандарт определяет, что транзакция начинается с первого SQL-оператора, инициируемого пользователем или содержащегося в программе. Все последующие SQL-операторы составляют тело транзакции.

Транзакция завершается одним из четырех возможных путей:

1) оператор COMMIT означает успешное завершение транзакции; его использование делает постоянными изменения, внесенные в базу данных в рамках текущей транзакции;

2) оператор ROLLBACK прерывает транзакцию, отменяя изменения, сделанные в базе данных в рамках этой транзакции; новая транзакция начинается непосредственно после использования ROLLBACK;

3) успешное завершение программы, в которой была инициирована текущая транзакция, означает успешное завершение транзакции (как будто был использован оператор COMMIT);

4) ошибочное завершение программы прерывает транзакцию (как будто был использован оператор ROLLBACK).

Реализация в СУБД принципа сохранения промежуточных состояний, подтверждения или отката транзакций обеспечивается специальным механизмом, называемым журналом транзакций.

Общими принципами восстановления являются следующие:

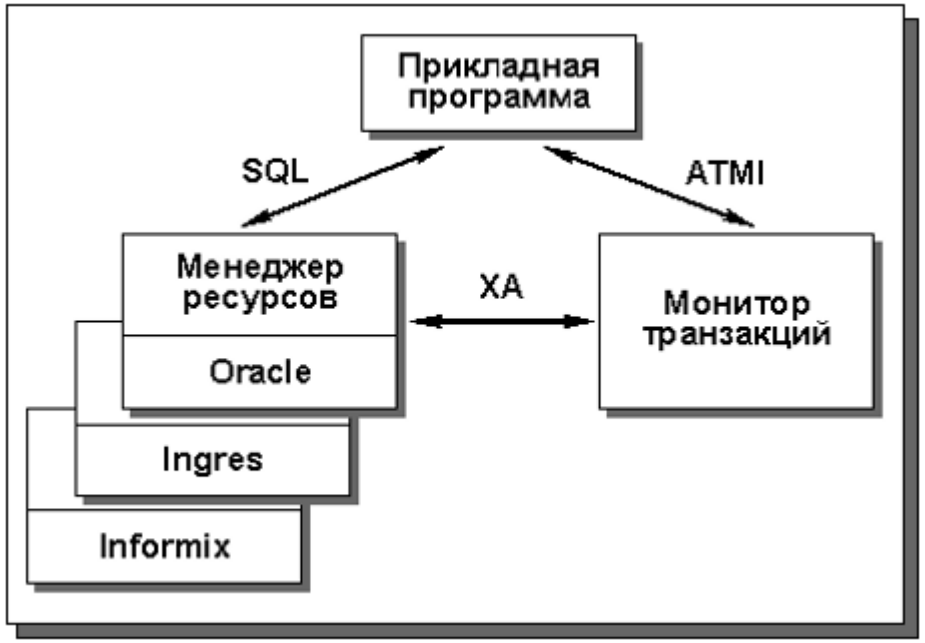
• результаты зафиксированных транзакций должны быть сохранены в восстановленном состоянии БД;

• результаты незафиксированных транзакций должны отсутствовать в восстановленном состоянии БД.

Это означает, что восстанавливается последнее по времени согласованное состояние БД.

Для оперативной обработки распределенных транзакций в неоднородных вычислительных системах (содержащих компьютеры, работающие под различными операционными системами) были созданы мониторы обработки транзакций (Transaction Processing Monitor — TPM), или мониторы транзакций. TPM опираются на трехзвенную модель клиент-сервер (модель сервера приложений или AS-модель).

Принципы организации обработки информации с помощью монитора транзакций описываются моделью монитора транзакций X/Open DTP (Distributed Transaction Processing — обработка распределенных транзакций). Эта модель (рисунок 1) включает в себя три объекта: прикладную программу, менеджер ресурсов (Resource Manager — RM) и монитор, или менеджер, транзакций (Transaction Manager — TM).



Модель обработки транзакций X/Open DTP

Интерфейс ATMI (Application Transaction Monitor Interface — интерфейс монитора транзакций приложения) позволяет вызывать функции TPM на некотором языке программирования, например, Си.

Функции менеджера ресурсов обычно выполняют серверы БД или СУБД. В задачах организации управления обработкой распределенных транзакций (транзакций, затрагивающих программные объекты вычислительной сети) TM взаимодействует с RM, который должен поддерживать протокол двухфазной фиксации транзакций и удовлетворять стандарту X/Open XA. Примерами СУБД, поддерживающих протокол двухфазной фиксации транзакции, являются Oracle 7.0, Open INGRES и Informix-Online 5.0.

Модель X/Open DTP не раскрывает структуру TM в деталях, а определяет состав компонентов распределенной системы обработки информации и как эти компоненты взаимодействуют друг с другом. Практические реализации этой модели, естественно, могут отличаться друг от друга. В числе примеров реализаций мониторов транзакций можно назвать ACMS, CICS, и TUXEDO System.

Для разработчиков приложений мониторы обработки транзакций TPM предоставляют удобства, связанные с возможностью декомпозиции приложений по нескольким функциональным уровням со стандартными интерфейсами, что позволяет создавать легко модифицируемые ИС со стройной архитектурой.

Для пользователей распределенных систем TPM позволяют улучшить показатели пропускной способности и времени отклика, снизить стоимость обработки данных в оперативном режиме на основе эффективной организации вычислительного процесса.

Администраторы распределенных систем, имея TPM, получают возможность легкого масштабирования ИС и увеличения производительности обработки информации. Здесь, кроме вертикального и горизонтального масштабирования, можно обеспечить так называемое матричное масштабирование. Суть его в том, что без остановки серверов приложений в любое время может быть добавлен, например, компьютер или менеджер ресурсов.