Распределенные транзакции

Распределенные транзакции

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

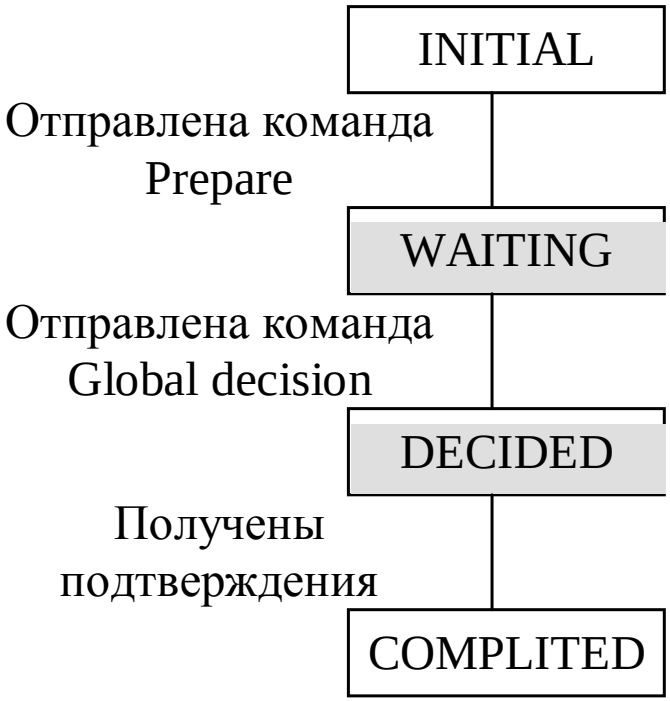
Основная проблема распределенных транзакций – соблюдение логической целостности данных. Транзакция на всех узлах должна завершиться одинаково: или фиксацией, или откатом.

Выполнение распределенных транзакций осуществляется с помощью специального алгоритма, который называется **двухфазная фиксация (Two Phase Commit, 2PC)**.

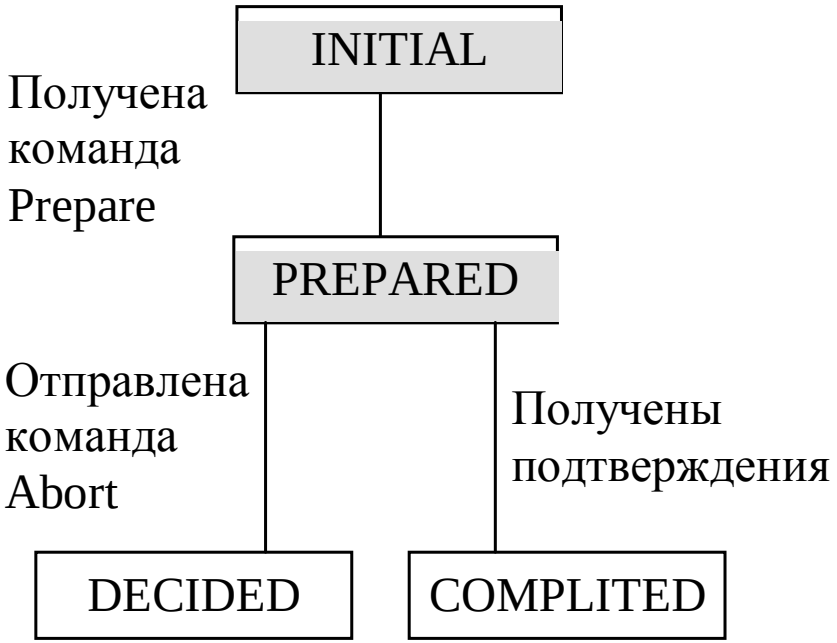
Координатор транзакции – узел, который контролирует выполнение этого протокола (обычно, тот узел, который инициирует данную транзакцию).

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.

Ожидать ответов всех участников в пределах установленного тайм-аута.

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.

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* 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).
2. Если участник отправил координатору сообщение READY_COMMIT, то он ожидает ответа координатора в пределах установленного тайм-аута.
3. При получении GLOBAL_ABORT участник помещает запись *abort* в файл журнала транзакций, откатывает свою часть транзакции и отправляет координатору подтверждение отката.
4. При получении GLOBAL_COMMIT участник помещает запись *commit* в файл журнала транзакций, фиксирует свою часть транзакции и отправляет координатору подтверждение фиксации.
5. 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: 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: координатор повторно рассылает сведения и принятом глобальном решении и ждет ответов от участников.

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:

1. 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.
2. Тайм-аут в состоянии PREPARED: участник уже известил координатор 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.

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

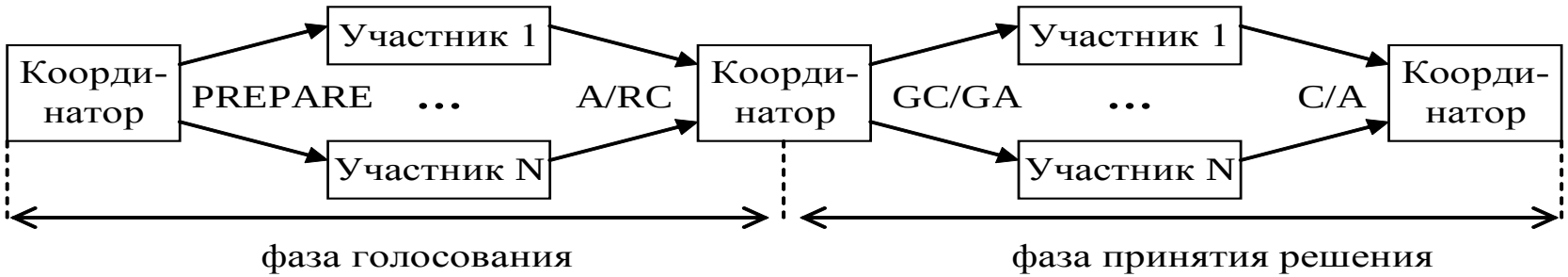
При отказе участника цель протокола восстановления – гарантировать, что после восстановления узел выполнит в отношении транзакции то же действие, которое выполнили другие участники, и сделает это независимо от координатора, т.е. по возможности без дополнительных подтверждений.

Рассмотрим три возможных момента возникновения отказа:

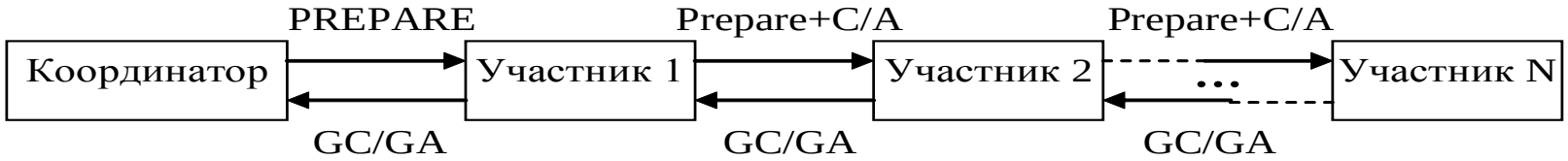
- ✓ В состоянии INITIAL: участник еще не успел сообщит о своем решении координатору, поэтому он может выполнить откат, т.к. координатор не мог принять решение о глобальной фиксации транзакции без голоса этого участника.
- ✓ В состоянии PREPARED: участник уже направил сведения о своем решении координатору, 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

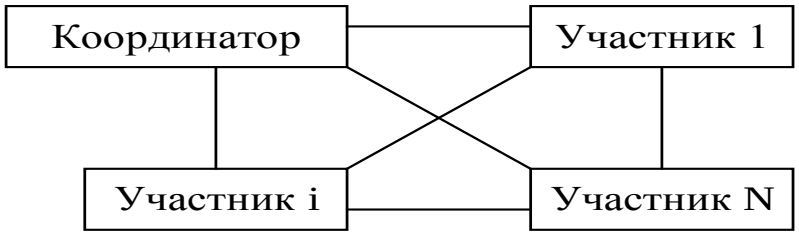
а) традиционная реализация протокола 2ФФ



б) линейная реализация протокола 2ФФ



в) распределенная реализация протокола 2ФФ



Протоколы блокировки

Централизованный протокол двухфазной блокировки

Во всей распределенной СУБД существует только один планировщик, или *диспетчер блокировок,* способный устанавливать и снимать блокировку с элементов данных. Координатор отвечает за соблюдение согласованности БД. Если транзакция предусматривает обновление реплицируемого элемента данных, координатор должен обеспечить обновление всех существующих реплик. Поэтому координатор должен установить исключительные блокировки на всех копиях обновляемого элемента данных, а затем освободить эти блокировки. Для чтения обновляемого элемента данных координатор может выбрать любую из существующих копий (обычно, локальную). Локальные диспетчеры транзакций, участвующие в выполнении глобальной транзакции, запрашивают и освобождают блокировки под управлением центрального диспетчера блокировок (ЦДБ). ЦДБ проверяет допустимость поступающих запросов на блокировку с учетом текущего состояния блокировки элементов данных. Если блокировка невозможна, запрос помещается в очередь. Этот протокол требует отправки не менее $2n + 3$ сообщений, в том числе:

- один запрос на блокировку;
- одно сообщение о предоставлении блокировки;
- n сообщений с требованием обновления;
- n подтверждений о выполненном обновлении;
- один запрос на снятие блокировки.

Протоколы блокировки

Двухфазная блокировка с первичными копиями (СУБД INGRES)
Каждый локальный диспетчер отвечает за управление блокировкой некоторого набора элементов данных. В процессе репликации для каждого копируемого элемента данных одна из копий выбирается в качестве *первичной копии* (primary copy), а все остальные рассматриваются как *вторичные* (slave copy). Выбор первичного узла может осуществляться по разным правилам, причем узел, который выбран для управления блокировкой первичной копии данных, не обязательно должен содержать саму эту копию. При обновлении элемента данных достаточно установить исключительную блокировку только его первичной копии. После того как первичная копия будет обновлена, внесенные изменения могут быть распространены на все вторичные копии. Это распространение должно быть выполнено с максимально возможной скоростью, чтобы предотвратить чтение другими транзакциями устаревших значений данных. Однако нет необходимости выполнять все обновления в виде одной элементарной операции. Данный протокол гарантирует актуальность значений только первичной копии данных.

Протоколы блокировки

Распределенный протокол двухфазной блокировки (System R*)

Диспетчер блокировок есть на каждом узле системы, и он отвечает только за блокировку локальных данных. Если данные не подвергаются репликации, этот протокол функционирует аналогично протоколу двухфазной блокировки с первичными копиями. В противном случае используется особый протокол управления репликацией – "чтение одной копии и обновление всех копий" (Read-One-Write-All – ROWA). В этом случае для операций чтения может использоваться любая копия копируемого элемента, но прежде чем можно будет обновить значение элемента, должны быть установлены исключительные блокировки на всех копиях. В такой схеме управление блокировками осуществляется децентрализованным способом, что позволят избавиться от недостатков, свойственных централизованному управлению. Однако данному подходу присущи свои недостатки, связанные с существенным усложнением методов выявления взаимоблокировок (из-за наличия многих диспетчеров блокировок) и возрастанием издержек на передачу данных (по сравнению с протоколом двухфазной блокировки с первичными копиями), которые вызваны необходимостью блокировать все копии каждого обновляемого элемента. Этот протокол требует передачи не менее 5n сообщений, в том числе:

- *n* сообщений с запросами на блокировку;
- *n* сообщений с предоставлением блокировки;
- *n* сообщений с требованием обновления элемента;
- *n* сообщений с подтверждением выполненного обновления;
- *n* сообщений с запросами на снятие блокировки

Протоколы блокировки

Блокировка большинства копий

Диспетчер блокировок имеется на каждом из узлов системы. Когда транзакции требуется считать или записать элемент данных, копии которого имеются на *n* узлах системы, она должна отправить запрос на блокировку этого элемента более чем на половину из этих *n* узлов. Транзакция не имеет права продолжать свое выполнение, пока не установит блокировки на большинстве копий элемента данных. Если ей не удастся это сделать за некоторый установленный промежуток времени, она отменяет свои запросы и информирует все узлы об отмене ее выполнения. Если большинство подтверждений будет получено, все узлы информируются о том, что требуемый уровень блокировки достигнут. Разделяемая блокировка на большинстве копий может быть установлена одновременно для любого количества транзакций, а исключительная блокировка на большинстве копий может быть установлена только для одной транзакции.

Недостатки: повышенная сложность алгоритма, усложнении процедур выявления взаимоблокировки, а также необходимости отправки не менее [(л + 1) /2] сообщений с запросами на установление блокировки и [*(п* + 1)/2] сообщений с запросами на отмену блокировки.

Протоколы с временными отметками

Проблема формирования временной отметки.
Общее правило: <локальная_отметка.идентификатор_узла>
Значение идентификатора узла имеет меньший весовой коэффициент, что гарантирует упорядочение событий по времени появления, а затем – по месту появления.
Чтобы предотвратить выработку более загруженными узлами больших значений временных отметок по сравнению с недогруженными узлами, необходимо использовать определенный механизм синхронизации значений временных отметок между узлами. Каждый узел помещает свою текущую временную отметку в сообщения, передаваемые на другие узлы. При получении сообщения узел-получатель сравнивает текущее значение его временной отметки с полученным и, если его текущая временная отметка оказывается меньше, меняет ее значение на некоторое другое, превосходящее то значение временной отметки, которое было получено им в сообщении. Например, если узел 1 с текущей временной отметкой <10, 1> передает сообщение на узел 2 с текущей временной отметкой <15, *2*>, то узел 2 не изменяет свою *временную* отметку. И наоборот, если текущая временная отметка на узле 2 равна <*5, 2*>, то узел 2 должен изменить свою *временную* отметку на <11, 2>.

Устранение взаимных блокировок в РБД

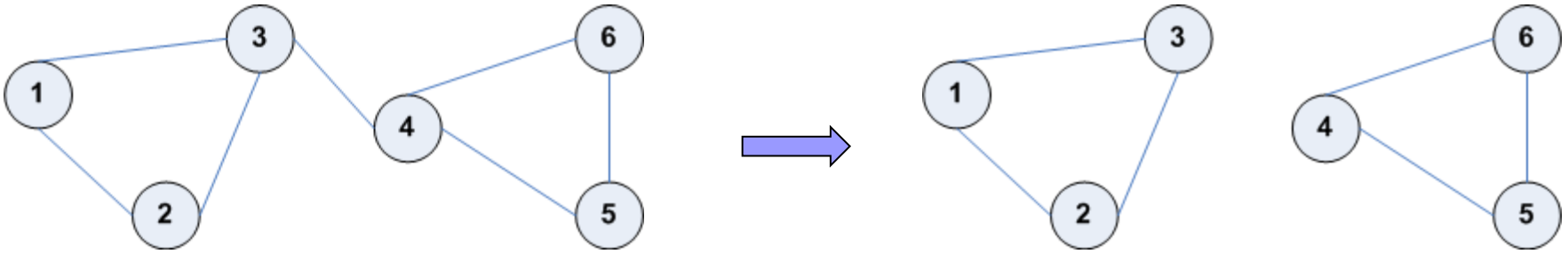
Проблемы выявление взаимных блокировок: локальные графы ожидания, как правило, не содержат циклов. Необходимо строить глобальный граф ожидания, представляющий собой объединение всех локальных графов ожидания. Существуют три основных метода выявления взаимоблокировок в распределенных СУБД:

- *централизованный:* один из узлов системы назначается координатором выявления взаимоблокировок. С определенным интервалом каждый диспетчер блокировок в системе направляет в адрес координатора свой локальный граф ожидания, содержащий внешние по отношению к узлу вершины (изменения в нём).
- *иерархический:* координаторы образуют иерархию, координатор нижнего уровня пересылает граф ожидания координатору верхнего уровня, если этот граф содержит внешнюю вершину.
- *распределенный:* существуют различные методы, но наиболее широко известен метод Обермарка. В соответствии с ним каждый узел пересылает свой граф ожидания тому узлу, на котором находится ресурс, необходимый данному узлу для продолжения транзакции. Для предотвращения взаимной пересылки каждому узлу назначается некоторая метка t, и пересылка от узла A к узлу B происходит, только если A.t < B.t.

Восстановление в РБД

Для распределенных БД характерны следующие отказы:
• потеря сообщения (устраняется с использованием соответствующих сетевых протоколов);
• отказ линии связи;
• аварийный останов одного из узлов;
• разделение сети на отдельные подсети.
Сложности в установлении причины (типа) отказа.
При отказе в РБД необходимо выполнить следующие действия:
• Аварийно завершить все транзакции, затронутые данным отказом.
• Отметить узел как отказавший, чтобы предотвратить любые попытки его использования другими узлами.
• Периодически проверять состояние отказавшего узла для восстановления его функционирования или ожидать поступления от этого узла широковещательного сообщения о восстановлении его нормальной работы.
• При перезапуске узла после отказа на нем должна выполняться процедура восстановления, предназначенная для отката любых транзакций, выполненных на момент отказа лишь частично.
• После завершения процедуры локального восстановления отказавший узел должен обновить свою копию базы данных, чтобы привести ее в соответствие с остальной частью системы.

Разделение сети: проблемы



Проблемы с обнаружением обновлений (например, две разные последовательности транзакций могут дать одинаковый результат).
Проблемы с обеспечением целостности (когда по отдельности две транзакции не нарушают ОЦ, а вместе приводят к нарушению).
Для принятия решения о том, можно ли продолжать работу в сети, разделенной на фрагменты, необходимо найти компромисс между доступностью данных и правильностью их использования. Абсолютной правильности можно проще всего достичь, если в случае разделения сети полностью запретить обработку каких-либо реплицируемых данных. С другой стороны, доступность данных в той же ситуации будет максимальной, если на обработку реплицируемых данных не накладывать никаких ограничений. В общем случае для сетей, разделенных на произвольные фрагменты, невозможно создать неблокирующий, соблюдающий свойство неразрывности протокол фиксации транзакций

Разделение сети: оптимистические протоколы

В оптимистических протоколах предпочтение отдается доступности данных, даже за счет возможной потери целостности базы данных. Кроме того, в этом случае используются оптимистические протоколы управления параллельным выполнением, позволяющие независимо выполнять любые обновления в каждом из фрагментов сети. В результате после возобновления связи в сети весьма вероятен переход базы данных в несогласованное состояние. Для выявления нарушений целостности базы данных может использоваться *граф предшествования,* содержащий сведения о взаимосвязях элементов данных. Он содержит сведения о том, какие элементы данных считывала и записывала каждая из выполнявшихся транзакций. Пока сеть находится во фрагментированном состоянии, обновления выполняются без каких-либо ограничений, но для каждого из фрагментов сети строится собственный граф предшествования. После возобновления связи в сети графы предшествования всех фрагментов сливаются. Нарушение целостности базы данных имеет место в том случае, если итоговый граф предшествования содержит циклы. Способ устранения нарушения целостности зависит от семантики выполнявшихся транзакций, поэтому в общем случае диспетчер восстановления не может восстановить целостность базы данных без вмешательства пользователей.

Модель распределенной обработки транзакций X/Open

Группа X/Open DTP (Distributed Transaction Processing))выпустила стандарт, определяющий три взаимодействующих компонента: приложение, диспетчер транзакций (Transaction Manager – TM) и диспетчер ресурсов (Resource Manager – RM).

Диспетчер ресурсов – любая подсистема, управляющая используемыми в транзакциях данными, например, СУБД или файловая система в сочетании с диспетчером сеансов. Диспетчер транзакций отвечает за:

- определение границ транзакции;
- присвоение каждой из транзакций уникального идентификатора;
- координацию работы остальных компонентов с целью определения результатов выполнения транзакции.

Начиная выполнение некоторой транзакции, приложение прежде всего обращается к диспетчеру транзакций, а затем к диспетчерам ресурсов с целью выполнения манипуляций с данными, необходимых для реализации алгоритмов обработки данных. По окончании обработки данных приложение вновь обращается к диспетчеру транзакций с требованием завершения данной транзакции. Для координации выполнения транзакции диспетчер транзакций взаимодействует с диспетчерами ресурсов.

Интерфейсы стандарта X/Open

Интерфейс TX
состоит из следующих процедур:

• tx_open и tx_close. Позволяют открыть и закрыть сеанс с помощью диспетчера транзакций.
• tx begin. Применяется для запуска новой транзакции.
• tx_commit и tx_abort. Предназначены для фиксации и аварийного завершения транзакции.

Интерфейс XA состоит **из** следующих процедур:

• xa_open и xa_close. Выполняют подключение и отключение от диспетчера ресурсов.
• xa_start и xa_end. Служат для запуска новой транзакции с указанным идентификатором транзакции и для ее завершения.
• xa_rollback. Выполняет откат транзакции с указанным идентификатором транзакции.

Интерфейсы стандарта X/Open

Интерфейс XA состоит **из** следующих процедур (продолжение):

• xa_prepare. Служит для подготовки транзакции с указанным идентификатором для глобальной фиксации/аварийного завершения.

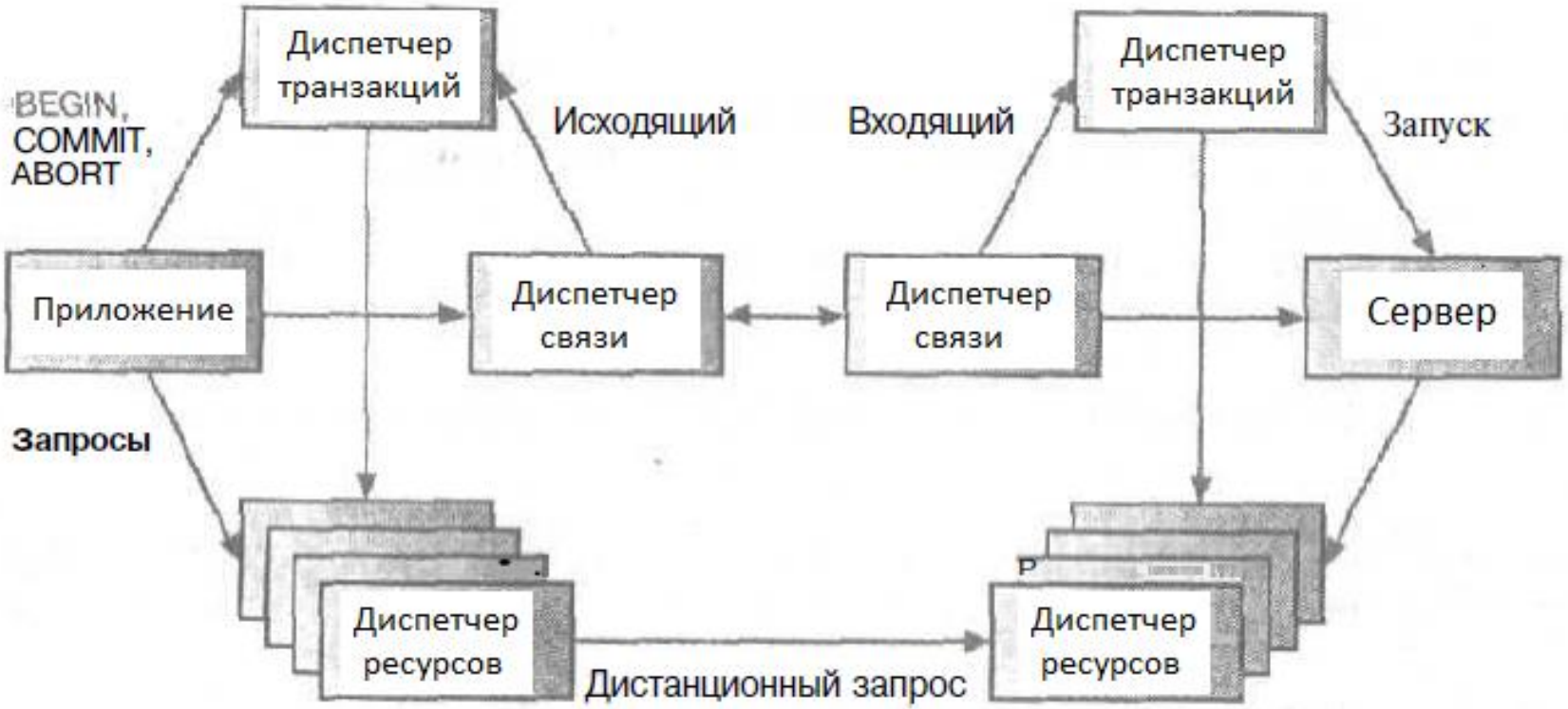
• xa_commit. Выполняет глобальную фиксацию транзакции с указанным идентификатором транзакции.

• xa_recover. Обеспечивает выборку списка подготовленных транзакций, зафиксированных или аварийно завершенных, с использованием эвристических алгоритмов. Если диспетчер ресурсов заблокирован, то при выполнении любого оператора транзакции может быть выработано эвристическое решение (как правило, предусматривающее аварийное завершение), позволяющее освободить заблокированные ресурсы. После возобновления нормальной работы диспетчера транзакций этот список транзакций может использоваться для передачи информации о фактически принятом решении тем транзакциям, состояние которых вызывает сомнение. Этот список ведется в виде журнала и позволяет также передать а приложение информацию о любых эвристических решениях, которые были приняты при обнаружении ошибки.

• xa_forget. Применяется для передачи диспетчеру ресурсов информации о том, что он должен удалить из памяти сведения об эвристической транзакции с указанным идентификатором транзакции.

X/Open в распределенной среде

В распределенной среде модель взаимодействия должна быть модифицирована таким образом, чтобы можно было применять транзакции, состоящие из нескольких отдельных субтранзакций, каждая из которых будет выполняться на удаленном узле и обращаться к удаленной базе данных.



X/Open в распределенной среде

Приложение взаимодействует с диспетчером передачи данных через один из типовых интерфейсов. В этом случае необходимо наличие двух механизмов – удаленного запуска и поддержки распределенных транзакций. Удаленный запуск может быть организован с помощью стандартизированных ISO механизмов ROSE (Remote Operations SErvice) или RFC (Remote Procedure Call). Для координации выполнения распределенных транзакций (интерфейс TM--TM) стандарт X/Open определяет коммуникационный протокол OSI-TP (Open Systems Interconnection Transaction Processing).

Модель X/Open DTP поддерживает выполнение не только плоских транзакций, но также последовательных и вложенных транзакций. В случае вложенных транзакций при отмене любой из субтранзакций будет отменена и вся глобальная транзакция.

Модель X/Open нашла широкую поддержку у производителей программного обеспечения. Несколько независимых компаний выпустили на рынок пакеты диспетчеров транзакций, поддерживающих интерфейс TX, а многие разработчики коммерческих СУБД реализовали в своих продуктах поддержку интерфейса XA. Примерами подобных систем являются мониторы транзакций CICS и Encina компании IBM (которые используются в основном на платформах IBM AIX или Windows NT, а теперь вошли в состав продуктов TXSeries компании IBM), диспетчер транзакций Tuxedo компании BEA Systems, а также СУБД Oracle, Informix и SQL Server.