

9. Управление распределенными данными

С управлением данными в распределенных системах связаны следующие две группы проблем: поддержка соответствия БД вносимым изменениям и обеспечение совместного доступа нескольких пользователей к общим данным.

§9.1 Поддержка соответствия БД вносимым изменениям

В современных распределенных системах информация может храниться *централизованно* или *децентрализованно*. В первом случае проблемы идентичности представления информации для всех пользователей не существует, так как все последние изменения хранятся в одном месте. На практике чаще информация изменяется одновременно в нескольких узлах распределенной вычислительной системы. В этом случае возникает проблема контроля за всеми изменениями информации и предоставления ее в достоверном виде всем пользователям.

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. В качестве примера можно привести распределенный библиотечный каталог, фактически объединяющий в себе несколько каталогов отдельных библиотек (которые могут быть расположены даже в разных городах). В частности, так организован каталог ГПНТБ СО РАН.

Информация о местоположении каждой из частей распределенной БД и другая служебная информация хранится в так называемом *глобальном словаре данных*. В общем случае этот словарь может храниться на одном из узлов или тоже быть распределенным.

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

На втором этапе, 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.

Основным достоинством модели распределенной БД является то, что пользователи всех узлов (при исправных коммуникационных средствах) получают информацию с учетом всех последних изменений. Второе достоинство состоит в экономном использовании внешней памяти компьютеров, что позволяет организовывать БД больших объемов.

К недостаткам модели распределенной БД относится следующее: жесткие требования к производительности и надежности каналов связи, а также 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.

К основным достоинствам модели тиражирования БД (в сравнении с предыдущей моделью) относятся: более высокая скорость доступа к данным, так как они всегда есть в узле; существенное уменьшение передаваемого по каналам связи потока информации, поскольку происходит передача не всех операций доступа к данным, а только изменений в БД; повышение надежности механизмов доступа к распределенным данным, поскольку нарушение связи не приводит к потере работоспособности системы (предполагается буферизация потока изменений, позволяющая корректно возобновить работу после восстановления связи).

Основной недостаток модели тиражирования БД заключается в том, что на некотором интервале времени возможно «расхождение» копий БД. Если отмеченный недостаток не критичен для прикладных задач, то предпочтительно иметь схему с тиражированием БД.

§9.2 Доступ к общим данным

При обслуживании обращений к общим данным средства управления БД должны обеспечивать по крайней мере два основных метода доступа: монопольной и коллективный. Основными объектами доступа в различных системах могут быть целиком БД, отдельные таблицы, записи, поля записей. В СУБД, предоставляющих возможность разработки, объектами доступа также

могут выступать спецификации отчетов и экранных форм, запросы и программы.

Монопольный доступ обычно используется в двух случаях:

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

В первом случае пользователь с помощью диалоговых средств СУБД или прикладной программы устанавливает явную *блокировку*. Во втором случае пользователь тоже может установить явную блокировку, либо положиться на СУБД. Последняя обычно автоматически устанавливает неявную (без ведома пользователя или приложения) блокировку, если это необходимо.

В режиме **коллективного доступа** полная блокировка на используемые объекты, как правило, не устанавливается. 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.

Рассмотрим некоторый типичный набор блокировок. В конкретных программах схемы блокирования объектов могут отличаться от описываемой. 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. Означает полное запрещение всяких операций над основными объектами (таблицами, отчетами и экранными формами). Этот вид блокировки обычно применяется при изменении структуры таблицы.

Блокировка от записи. Накладывается в случаях, когда можно использовать таблицу, но без изменения ее структуры или содержимого. Такая блокировка 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. Предохраняющая блокировка от записи совместима с аналогичной блокировкой (предохраняющей блокировкой от записи), а также с предохраняющей полной блокировкой. Примером необходимости использования этой блокировки является режим 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. Такая блокировка может использоваться, например, для обеспечения одновременного просмотра несколькими пользователями одной таблицы. В группе пользователей, работающих с одной таблицей, эта блокировка не позволит никому изменить структуру общей таблицы.

При незавершенной операции с некоторым объектом и запросе на выполнение новой операции с этим же объектом производится попытка эти операции совместить. Совмещение возможно тогда, когда совместимыми оказываются блокировки, накладываемые конкурирующими операциями.

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. Пользователям современных СУБД, работающим в интерактивном режиме, не нужно помнить все тонкости механизма блокировки, поскольку система достаточно «разумно» осуществляет автоматическое блокирование во всех

случаях, когда это требуется. При этом система сама стремится предоставить всем пользователям наиболее свободный доступ к объектам. При необходимости пользователь и программист может воспользоваться командными или языковыми средствами явного определения блокировок. Например, в СУБД Paradox для явного блокирования отдельной записи во время редактирования таблицы используется команда Record Lock.

§9.3 Тупики

Если не управлять доступом к совместно используемым объектам, то между потребителями ресурсов могут возникать тупиковые ситуации (клинчи, «смертельные объятия» или блокировки). Следует отличать понятие блокировки в смысле контроля доступа к объектам от блокировки в смысле тупикового события.

Существует два основных вида тупиков: *взаимные* (deadlock) и *односторонние* (livelock).

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 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. Пример одной из таких ситуаций приведен на рисунке 1б.

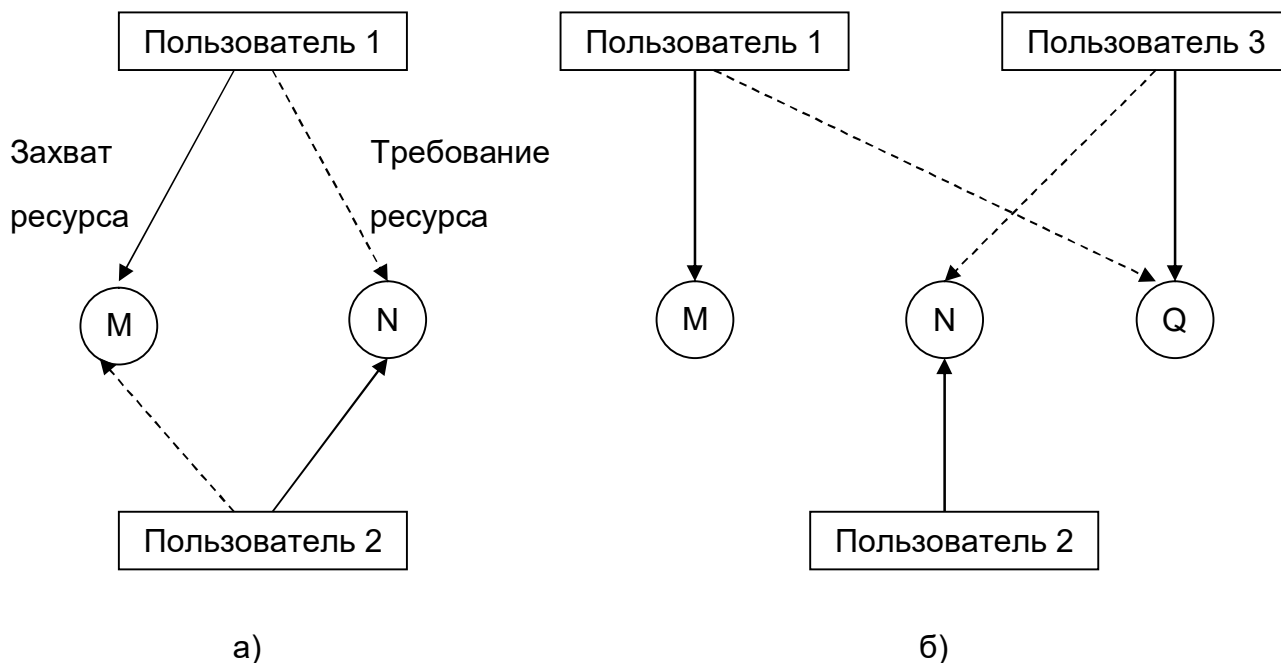


Рисунок 1 — Примеры взаимных тупиков в распределенных БД

Односторонний тупик возникает в случае требования получить монопольный доступ к некоторому ресурсу как только он станет доступным и невозможности удовлетворить это требование.

Системы управления распределенными БД, очевидно, должны иметь соответствующие средства обнаружения или предотвращения конфликтов, а также разрешения возникающих конфликтов. Одной из наиболее сложных является задача устранения конфликтов в неоднородных системах в случае, если некоторая программа не обрабатывает или обрабатывает некорректно сигналы (уведомления) о наличии конфликтов. При этом важно не только сохранить целостность и достоверность данных в распределенных БД, но и восстановить вычислительный процесс, иногда парализующий пользователей и программы ожиданием чего-то.

Пользователи и разработчики приложений в распределенной среде должны предусматривать обработку сигналов о тупиках.