Репликации

Для многих предприятий и организаций важно, чтобы данные, используемые различными приложениями, были бы доступны в любом месте и в любое время. Подобная среда для данных может быть предоставлена несколькими базами данных, которые включают множество копий одной и той же информации.

Переезжающие с места на место менеджеры представляют хороший пример того, как используется среда распределенных данных. В течение дня продавец обычно использует дорожный компьютер для запроса всей необходимой информации из базы данных (например, цены и наличие товара) для информирования на месте покупателей. Позже, в номере отеля, продавец снова использует дорожный компьютер - на этот раз для пересылки данных о проданных товарах в центральный офис предприятия.

Пример иллюстрирует тот факт, что среда распределенных данных имеет несколько преимуществ по сравнению с централизованной обработкой данных:

- данные доступны пользователям в нужном месте и в нужное время;
- локальный пользователь может автономно оперировать данными;
- система сокращает сетевой трафик;
- технологический безостановочный процесс существенно дешевле.

С другой стороны, среда распределенных данных является гораздо более сложной системой, чем соответствующая централизованная модель, и поэтому требует больше сил на планирование и администрирование.

Методы распределения данных

Существуют два основных метода распределения данных на множество серверов базы данных – это распределенные транзакции и репликация данных.

Распределенная транзакция - это транзакция, где все изменения во всех местах, где хранятся распределенные данные, собираются вместе и выполняются синхронно. Распределенные системы баз данных используют метод, называемый двухфазовым подтверждением, для реализации распределенных транзакций.

Каждая база данных, вовлеченная в распределенную транзакцию, имеет собственную технику восстановления, которая используется в случае ошибки. Менеджер глобального восстановления (называемый координатором) координирует обе фазы распределенного процесса.

На первой фазе этого процесса координатор проверяет, все ли участвующие элементы готовы выполнять их часть распределенной транзакции. Вторая фаза состоит из фактического выполнения транзакции на всех участвующих элементах. В течение этого процесса любая ошибка на любом элементе приводит к тому, что координатор останавливает транзакцию. В этом случае он отправляет сообщение каждому менеджеру локального восстановления, чтобы он отменил ту часть транзакции, которая уже была выполнена на этом элементе.

Microsoft Distributed Transaction Coordinator (*DTC*) поддерживает распределенные транзакции с использованием двухфазового подтверждения.

DTC – компонент *Microsoft Windows*, предназначенный для координации изменения данных на двух или более сетевых компьютерных системах. Он основан на технологии *COM+* и включает в себя: менеджер транзакций; журнал транзакций; прокси (реализует интерфейсы *DTC*); утилиты администрирования и заголовочные файлы *API.*

Каждый компьютер, участвующий в выполнении распределённых транзакций, имеет локальный менеджер транзакций, который взаимодействует с приложениями и локальными менеджерами ресурсов (такими, как базы данных, файловые системы, системы хранения документов, очереди сообщений). При получении запроса на выполнение транзакции между парами систем устанавливаются отношения *вышестоящая - подчинённая*. Каждая система может иметь несколько подчинённых систем, но не более одной вышестоящей. Это отношение справедливо для каждой конкретной транзакции, при выполнении других транзакций роли менеджеров могут меняться.

При запросе фиксации или отката транзакции менеджером транзакций выполняется двухфазный протокол фиксации. Во время первой фазы менеджеру

ресурсов посылается запрос на подготовку к завершению, во время второй - на фиксацию или откат транзакции. По дереву, образованному вышестоящими и подчинёнными системами, рассылаются сообщения для подготовки к завершению, фиксации или отката. Любой узел дерева может прервать транзакцию до подтверждения подготовки к завершению. После того, как узел подтвердил подготовку, он остаётся в этом состоянии до фиксации или отката транзакции вышестоящим узлом. В случае сбоя и перезагрузки компьютера менеджер транзакций запрашивает вышестоящий узел о дальнейшей судьбе подготовленных к завершению транзакций.

Репликация (*replication*) – это механизм синхронизации содержимого нескольких копий базы данных. Это процесс копирования измененных данных из одного источника на другой (или на множество других) и наоборот. При репликации изменения, сделанные в одной копии базы данных, могут быть распространены в другие копии.

Репликация делается автоматически, администратору не надо делать вручную экспорт, а потом импорт данных.

Другими словами, это синхронизация между несколькими серверами. Например, есть сервер с *БД* сайта, к которому в минуту обращаются 5000 посетителей, и этот сервер заметно подтормаживает. Тогда рекомендуется одну и туже *БД* расположить на двух серверах чтобы посетителей поделить - по 2500 на каждый сервер. А для того чтобы посетители получали одну и туже информацию с разных серверов их надо синхронизировать, вот такая синхронизация и называется репликацией.

В течение процесса *репликации данных* копии данных распределяются от исходной базы данных к одной или более целевым базам данных, размещенным на отдельных компьютерах. По этой причине репликация данных отличается от распределенных транзакций в двух аспектах, а именно: промежутком времени и задержкой по времени.

В отличие от метода распределенных транзакций, где все данные являются

одними и теми же на всех участвующих элементах, репликация данных позволяет элементам иметь различные данные в одно и то же время. Кроме того, репликация данных является асинхронным процессом. Это означает, что существует некоторая задержка по времени, когда все копии данных становятся одинаковыми на всех участвующих в процессе элементах. Эта задержка может изменяться в диапазоне от единиц секунд до нескольких дней или недель.

Выбор метода распределения данных

Репликация данных в большинстве случаев является лучшим решением, чем использование распределенных транзакций, потому что она дешевле и надежнее. Эксперименты с двухфазовым подтверждением транзакций показали, что администрирование становится более сложным при увеличении участвующих в этом процессе элементов. Кроме того, увеличение участвующих элементов уменьшает надежность, потому что вероятность того, что локальная часть распределенной транзакции завершится со сбоем, увеличивается при увеличении количества узлов. Если один локальный элемент дает сбой, то вся распределенная транзакция также даст сбой.

Другой причиной использования репликации данных вместо централизации данных является производительность: пользователи элемента с репликацией данных, имеют повышенную производительность, потому что у них есть возможность получать доступ к данным локально, а не при использовании сети для соединения с центральным сервером базы данных.

Общие сведения о репликации

Вообще, репликация основывается на двух различных концепциях: использование протокола транзакций и использование триггеров.

Database Engine хранит все значения измененных строк (значения как "до", так и "после") в системных файлах, называемых протоколами транзакций. Если для выбранных строк нужно выполнить репликацию, система запускает новый

процесс, который читает данные из протокола транзакций и отправляет их одной или более целевым базам данных.

Второй метод основан на триггерах. Изменение таблицы, содержащей данные, для которых должна быть выполнена репликация, вызывает соответствующий триггер, который по порядку создает новую таблицу с данными и запускает процесс репликации.

Обе концепции имеют как преимущества, так и недостатки. Репликация, основанная на протоколе, характеризуется улучшенной производительностью, поскольку процесс, который читает данные из протокола транзакций, выполняется асинхронно и оказывает очень небольшое влияние на производительность всей системы. С другой стороны, реализация репликации, основанной на протоколе транзакций, является весьма сложной для компаний, создающих системы баз данных, потому что система базы данных должна не только управлять дополнительными процессами и буферизацией, но и разрешать проблемы одновременного доступа между системой и процессами репликации, которые обращаются к протоколу транзакций.

Database Engine использует обе концепции: метод протокола транзакции для процесса репликации на основе транзакций и триггеры для слияния обработанных репликаций.

Издатели, распространители и подписчики

Репликация *Database Engine* основывается на понятии издатель-подписчик. Оно описывает различные роли, которые могут играть серверы в процессе репликации. Один или более серверов публикуют данные, на которые могут подписываться другие серверы. Между ними существует распространитель, который сохраняет изменения и передает их дальше подписчикам. Следовательно, отдельный сервер может иметь три роли в технологии репликации.

- Издатель (или публикующий сервер). Поддерживает свои исходные

базы данных, делает данные доступными для репликации и отправляет измененные данные распространителю.

- Распространитель (или распределяющий сервер). Получает от издателя все изменения для данных, подлежащих репликации, сохраняет и отправляет их соответствующим подписчикам.
- Подписчик (или получающий сервер). Получает и сохраняет опубликованные данные.

Сервер базы данных может играть несколько ролей в процессе репликации. Например, сервер в одно и то же время может работать как издатель и как распространитель. Этот сценарий подходит для случая с небольшими репликациями и с небольшим количеством подписчиков. Если же существует множество подписчиков для информации издателя, то распространитель должен размещаться на собственном сервере.

Допустимы репликации только для пользовательских баз данных.

Единица публикуемых данных называется публикацией. Статья содержит данные из таблицы и/или одной или более хранимых процедур. Таблица статьи может быть целой таблицей или подмножеством данных таблицы. Хранимая процедура статьи может содержать одну или более хранимых процедур, которые существуют в базе данных на момент публикации.

Публикация содержит одну или более статей. Каждая публикация может содержать данные только из одной базы данных.

Публикация является основой подписки. Это означает, что нельзя подписываться напрямую к статье, потому что статья является частью публикации.

Фильтр является процессом, который ограничивает информацию, создавая подмножество. Поэтому публикация содержит один или более следующих элементов, которые задают типы статей таблицы: таблица, вертикальный фильтр, горизонтальный фильтр и комбинация вертикальных и горизонтальных фильтров. Вертикальный фильтр содержит подмножество столбцов таблицы. Горизонтальный фильтр содержит подмножество строк таблицы.

Подписка может быть инициализирована двумя различными способами - через принудительную подписку и через подписку по запросу.

При принудительной подписке все управление созданными подписками выполняется издателем в процессе определения публикации. Принудительные подписки упрощают и централизуют управление, потому что обычный сценарий репликации содержит одного издателя и множество подписчиков. Преимуществом принудительной подписки является высокий уровень безопасности, потому что процесс инициализации управляется из одного места. С другой стороны, производительность распространителя может пострадать, потому что полное распространение подписок выполняется немедленно.

При подписке по запросу подписчик инициализирует подписку. Подписка по запросу является более избирательной, чем принудительная подписка, потому что подписчик может выбирать публикации для подписки. В отличие от принудительной подписки подписка по запросу должна быть использована для публикаций с низким уровнем секретности и большим количеством подписчиков.

Загрузка данных через Интернет является типичным примером подписки по запросу.

Типы репликации

Database Engine предоставляет следующие типы репликации: репликация транзакций, репликация мгновенного снимка, репликация слияния и одноранговая репликация.

Репликация транзакций. Здесь для репликации данных используется протокол транзакций системы. Все транзакции, которые содержат данные, подлежащие репликации, отмечаются как транзакции для репликации. Компонент с именем *Log Reader Agent* отыскивает отмеченные транзакции и копирует их из протокола транзакций издателя в базу данных *distribution*. Другой компонент - *Distribution Agent* - перемещает транзакции подписчику, где они применяются для целевых таблиц в подписанных базах данных.

Все таблицы, опубликованные с использованием репликации транзакций, должны явно содержать первичный ключ. Первичный ключ требуется для уникальной идентификации строк в опубликованной таблице, потому что строка является единицей перемещения в репликации транзакций.

Репликация транзакций может выполнять репликацию таблиц (или части таблиц) и одной или более хранимых процедур. Использование хранимых процедур в репликации транзакций повышает производительность, потому что объем данных, пересылаемых через сеть, обычно бывает значительно меньше. Вместо репликации данных подписчику пересылаются только хранимые процедуры, где они и выполняются. Можно сконфигурировать время задержки синхронизации между издателем с одной стороны и подписчиком с другой стороны в процессе выполнения репликации транзакций. Все эти изменения распространяются компонентами *Log Reader Agent* и *Distribution Agent*.

База данных *distribution* является системной базой данных, которая инсталлируется на распределяющий сервер, когда запускается процесс репликации. Эта база данных содержит все реплицированные транзакции из публикаций и издателей, которые должны быть отправлены подписчикам. Это интенсивно используется только в репликации транзакций.

Прежде чем репликация транзакций может начаться, каждому подписчику должна быть направлена копия полной базы данных. Это осуществляется при выполнении мгновенного снимка.

Репликация мгновенного снимка. Это самый простой тип репликации копирует опубликованные данные от издателя всем подписчикам. Различием между репликацией мгновенного снимка и репликацией транзакций является то, что в первом случае подписчику пересылаются все опубликованные данные, а во втором случае - только измененные данные.

Репликация мгновенного снимка тесно связана с компонентом, называемым *Snapshot Agent*. Этот компонент генерирует схему и данные из опубликованных таблиц и сохраняет их в файлах. Схема таблицы и соответствующий

файл данных создают синхронизированный набор, который представляет мгновенный снимок таблицы в конкретный момент времени. Создает ли агент новые файлы мгновенных снимков каждый раз при его выполнении, зависит от типа репликации и выбранных опций.

Репликация транзакций и репликация мгновенного снимка являются однонаправленными репликациями, это означает, что изменения реплицируемых данных создаются только на публикующем сервере. Поэтому данные на всех получающих серверах являются данными только для чтения за исключением тех, изменения которых были выполнены в процессе репликации.

В отличие от репликации транзакций репликация мгновенного снимка не требует для таблиц присутствия первичного ключа. Причина очевидна: единицей пересылки в репликации мгновенного снимка является файл мгновенной копии, а не строка таблицы. Другим отличием между этими двумя типами репликации является задержка во времени - репликация мгновенного снимка выполняется периодически, что означает значительную задержку во времени, потому что все публикуемые данные (измененные и неизмененные) пересылаются от издателя к подписчикам.

Репликация мгновенного снимка не использует напрямую базу данных *distribution*. Однако база данных *distribution* содержит информацию состояния и другие детали, которые применяются в репликации мгновенного снимка.

Репликация слияния. В репликации транзакций и в репликации мгновенного снимка издатель отправляет данные, а подписчик их получает. Не существует такой возможности, чтобы подписчик отправлял реплицированные данные издателю. Репликация слияния позволит издателю и подписчикам изменять реплицируемые данные. По этой причине могут возникать конфликты в процессе репликации.

После создания публикации на публикующем сервере *Snapshot Agent* подготавливает файлы, содержащие схему таблицы и данные, и сохраняет их в ра-

бочем каталоге на существующем распределяющем элементе. В процессе слияния репликации база данных *distribution* содержит только лишь состояние процесса репликации. Затем используется синхронизирующее задание для другого компонента - *Merge Agent*, который отправляет все измененные данные на другой элемент. *Merge Agent* может отправлять реплицированные данные как подписчикам, так и издателям. Перед тем как стартует процесс пересылки, *Merge Agent* сохраняет соответствующую информацию, которая позволяет отслеживать конфликты изменений.

Когда используется сценарий слияния репликаций, система выполняет три важных изменения в схеме для публикуемой базы данных: она идентифицирует уникальный столбец для каждой реплицируемой строки, добавляет несколько системных таблиц и создает триггеры таблиц, для которых выполняется репликация данных.

Database Engine создает или находит уникальный столбец в таблице для реплицируемых данных. Если базовая таблица уже содержит столбец с типом данных *uniqueidentifier* и со свойством *rowguidcol*, то система использует этот столбец для идентификации каждой реплицируемой строки. Если в таблице не существует подобного столбца, то система добавляет столбец *rowguid* с типом данных *uniqueidentifier* и свойством *rowguidcol*.

Столбец *uniqueidentifier* может содержать множество экземпляров значений. Свойство *rowguidcol* дополнительно указывает, что значение столбца с типом данных *uniqueidentifier* уникально идентифицирует строку в таблице. Поэтому столбец с типом данных *uniqueidentifier* со свойством *rowguidcol* содержит уникальное значение для каждой строки среди всех компьютеров в сети во всем мире и поэтому гарантирует уникальность реплицируемой строки среди множества копий таблицы издателя и подписчиков.

Добавление новых системных таблиц предоставляет способ определения и разрешения любых конфликтов обновления. Для разрешения конфликта

Database Engine сохраняет все изменения, связанные с реплицируемыми данными, в системных таблицах слияния *msmerge_contents* и *msmerge_tombstone* и соединяет их (используя свойство *rowguid* существующего столбца с типом данных *uniqueidentifier*) с таблицей, которая содержит реплицируемые данные.

Database Engine для всех таблиц, содержащих реплицируемые данные, создает триггеры на всех элементах для отслеживания изменения данных в каждой реплицируемой строке. Эти триггеры определяют, какие были выполнены изменения в таблице и записывают их в системные таблицы msmerge_contents и *msmerge_tombstone*.

Выявление конфликта выполняется *Merge Agent* при использовании происхождения столбца в системной таблице *msmerge contents*. Разрешение конфликта может быть выполнено на основе приоритета или на основе пользователя. Разрешение конфликта на основе приоритета означает, что любой конфликт между новым и старым значениями в реплицируемой строке разрешается автоматически на основе назначенных приоритетов. Особым случаем метода на основе приоритета является метод "первый побеждает", при котором первое по времени изменение реплицируемой строки становится победителем. Метод на основе приоритета является методом по умолчанию. Метод на основе пользователя применяет пользовательский триггер, основанный на бизнес-правилах, определенных администратором базы данных для разрешения конфликтов.

Одноранговая репликация транзакций. Она является еще одной формой репликации транзакций, при которой каждый сервер в одно и то же время является издателем, распространителем и подписчиком одних и тех же данных. Иными словами, все серверы содержат одни и те же данные, но каждый сервер ответственный за изменение собственной части данных.

Одноранговую репликацию лучше всего объяснить примером. Предположим, что компания имеет несколько офисов филиалов в различных городах, и каждый офисный сервер имеет такой же набор данных, что и все другие серверы. С другой стороны, все данные разделены на подмножества, и каждый офисный

сервер может изменять только собственное подмножество данных. Когда данные изменяются на одном офисном сервере, эти изменения реплицируются на все другие серверы (подписчики) в одноранговой сети. Пользователи в каждом офисе могут читать данные без каких-либо ограничений.

Преимуществами этой формы репликации являются: вся система хорошо масштабируется; вся система предоставляет высокую доступность.

Система, которая поддерживает одноранговую репликацию, хорошо масштабируется, потому что каждый сервер обслуживает только локальных пользователей. Пользователи могут изменять только те части данных, которые принадлежат их локальному серверу. Для операций чтения все данные также хранятся локально. Высокая доступность основывается на том факте, что если один или более серверов отключаются от сети, все другие серверы могут продолжать работу, поскольку все данные, необходимые им для операций чтения и записи, хранятся локально. Когда отключенный сервер снова включается в сеть, заново стартует процесс репликации, и сервер получает все модификации данных, которые были выполнены на других сайтах.

Определение конфликтов в одноранговой репликации. В случае одноранговой репликации можено изменять данные на любом узле. Если строка изменяется более чем на одном узле, то это может привести к конфликту.

Одноранговая репликация в *SQL Server 2008* вводит опцию для возможности определения конфликта в одноранговой топологии. При включенной такой опции конфликтное изменение данных трактуется как критическая ошибка, которая приводит к завершению *Distribution Agent*. В случае конфликта сценарий остается в несогласованном состоянии, пока конфликт не будет разрешен и данные не будут приведены в согласованное состояние на всех участвующих серверах. Можно включить определение конфликтов, используя системные процедуры *sp_addpublication* или *sp_configure_peerconflictdetection*.

Конфликты в одноранговой репликации определяются хранимыми процедурами, которые применяют изменения к каждому узлу, основываясь на скрытом

столбце в каждой опубликованной таблице. Этот скрытый столбец хранит идентификатор, комбинируемый вместе с идентификатором, который вы указываете для каждого узла, и с версией строки. Эти процедуры выполняются из *Distribution Agent*, они применяют операции добавления, изменения и удаления с других участников сети. Если одна из процедур определяет конфликт при чтении значений скрытого столбца, то она вызывает ошибку 22815.

Скрытый столбец может быть доступен только тому пользователю, который соединился через *Dedicated Administrator Connection*.

Когда появляется конфликт одноранговой репликации, то выдается предупреждающее сообщение "*Peer-to-peer conflict detection alert*". Необходимо сконфигурировать это предупреждающее сообщение так, чтобы вы были соответствующим образом проинформированы при появлении конфликта.

Модели репликации

Типы репликации (транзакционная, мгновенный снимок, слияние и одноранговая) предоставляют функциональность для обработки реплицируемых данных. Репликационные модели используются в компании для проектирования их собственной репликации данных. Каждая модель репликации может быть реализована с использованием одного или более типов репликации. Тип репликации и репликационная модель обычно определяются в одно и то же время.

В зависимости от требований может быть использовано несколько моделей репликации. Тремя базовыми моделями репликации являются следующие: центральный издатель с распространителем; центральный подписчик с множеством издателей; множество издателей с множеством подписчиков.

Центральный издатель с распространителем. В модели центрального издателя с распространителем существует один издатель и обычно один распространитель. Издатель создает публикации, которые распространитель распределяет нескольким подписчикам. Эта модель является стандартной моделью.

Если объем публикуемых данных не является слишком большим, то издатель и распространитель могут располагаться на одном сервере. В противном случае рекомендуется использование двух отдельных серверов для повышения производительности. Если есть большой объем публикуемых данных, то распространитель обычно является узким местом в процессе.

Публикации, созданные в этой модели и полученные подписчиком, обычно бывают только для чтения. По этой причине в большинстве случаев репликация транзакций является предпочтительным типом репликации для этой модели, хотя здесь может быть также использована и репликация мгновенного снимка.

Центральный подписчик с множеством издателей. Сценарий про путешествующего менеджера, который передает данные в главный офис, является типичным примером центрального подписчика с множеством издателей. Данные собираются на центральном подписчике, и несколько издателей направляют туда свои данные.

Для этой модели можно использовать тип репликации транзакций или слияния в зависимости от использования реплицируемых данных. Если издатель публикует (и, значит, изменяет) одни и те же данные для подписчика, то должна быть использована репликация слияния. Если каждый издатель имеет для публикации собственные данные, то должна быть использована репликация транзакций или одноранговая репликация. В этом случае опубликованные таблицы будут отфильтровываться горизонтально, а каждый издатель будет исключительным владельцем отдельного фрагмента таблицы.

Множество издателей с множеством подписчиков. Модель репликации, в которой несколько серверов или все серверы, участвующие в репликации данных, играют роль издателей, а также подписчиков, известна как модель множества издателей с множеством подписчиков. В большинстве случаев эта модель включает несколько распространителей, которые обычно располагаются на каж-

дом издателе. Эта модель может быть реализована только при использовании репликации слияния, потому что публикации изменяются на сервере издателя. Только единственным другим способом реализации этой модели является использование распределенных транзакций с двухфазным подтверждением.

Управление репликацией

Все серверы, являющиеся участниками репликации, должны быть зарегистрированы. После регистрации серверов распределяющий сервер, сервер (серверы) издателя и сервер (серверы) подписчика должны быть настроены. Далее описывается конфигурирование этих процессов с помощью соответствующих мастеров.

Конфигурирование распределяющего и публикующего серверов. Прежде чем устанавливать публикуемые базы данных, необходимо установить распределяющий сервер и сконфигурировать распределяемые базы данных. Можно настроить распределяющий сервер, используя мастер *Configure Distribution Wizard*. Этот мастер позволяет сконфигурировать распространитель и распределяемую базу данных и сделать доступным издателя (издателей). Основные функции этого мастера:

- ♦ сконфигурировать определенный сервер в качестве распространителя так, чтобы его могли использовать другие издатели;
- ♦ сконфигурировать определенный сервер в качестве издателя так, чтобы он мог работать и как собственный распространитель;
- ♦ сконфигурировать определенный сервер в качестве издателя, чтобы он использовал другой сервер как распространителя.

Microsoft SQL Server 2008 Express может служить подписчиком для всех типов репликации, но не может служить издателем или распространителем.