Распределенные базы данных

- Технология распределенных баз данных, способствует обратному переходу от централизованной обработки данных к децентрализованной и является симбиозом сетевых технологий и технологий баз данных.
- **Распределенные БД -** набор логически связанных между собой разделяемых данных (и их описаний), которые физически распределены в некоторой компьютерной сети.
- **Распределенная СУБД** – программный комплекс, предназначенный для управления распределенными базами данных и позволяющий сделать распределенность информации прозрачной для конечного пользователя.
- Система управления распределенными базами данных (СУРБД) состоит из единой логической базы данных, разделенной на некоторое количество *фрагментов.* Каждый фрагмент базы данных сохраняется на одном или нескольких компьютерах, которые соединены между собой линиями связи и каждый из которых работает под управлением отдельной СУБД. Любой из серверов способен независимо обрабатывать запросы пользователей, требующие доступа к локально сохраняемым данным (что создает определенную степень локальной автономии), а также способен обрабатывать данные, сохраняемые на других компьютерах сети.
- В распределенной СУБД должно существовать хотя бы одно глобальное приложение, поэтому любая СУРБД должна иметь следующие особенности:
  - Набор логически связанных разделяемых данных, разбитых на некоторое количество фрагментов.
  - Между фрагментами может быть организована репликация данных.
  - Фрагменты и их реплики распределены по различным серверам, связанным между собой сетевыми соединениями.
  - Работа с данными на каждом сервере управляется СУБД., способной поддерживать автономную работу локальных приложений и поддерживающей хотя бы одно глобальное приложение.

- Распределенные СУБД можно классифицировать как ***гомогенные и гетерогенные***.

- В *гомогенных* системах все серверы используют один и тот же тип СУБД. В *гетерогенных* системах на серверах могут функционировать различные типы СУБД, использующие разные модели данных.

- Гомогенные системы значительно проще проектировать и сопровождать. Кроме того, подобный подход позволяет поэтапно наращивать размеры системы, последовательно добавляя новые серверы к уже существующей распределенной системе. Дополнительно появляется возможность повышать производительность системы за счет организации на различных серверах параллельной обработки информации.

- Гетерогенные системы обычно возникают в тех случаях, когда независимые серверы, уже эксплуатирующие свои собственные системы с базами данных, интегрируются во вновь создаваемую распределенную систему. В гетерогенных системах для организации взаимодействия между различными типами СУБД потребуется организовать трансляцию передаваемых сообщений. Пользователи каждого из серверов должны иметь возможность вводить запросы на языке той СУБД, которая используется на данном сервере. В общем случае данные могут быть затребованы с другого сервера, который характеризуется такими особенностями, как: иной тип используемого оборудования; иной тип используемой СУБД; иной тип применяемых оборудования и СУБД.

•Двенадцать правил, сформулированные Дейтом для типичной РСУБД

•Правила построения распределенных систем

•1. Локальная автономность

•С точки зрения конечного пользователя распределенная система должна выглядеть в точности так, как и обычная, не распределенная система. Серверы в распределенной системе должны быть автономными. Автономность означает следующее: все локальные процессы остаются чисто локальными; все процессы на заданном сервере контролируются только этим сервером.

•2. Отсутствие опоры на центральный сайт

•В системе не должно быть ни одного сервера, без которого система не сможет функционировать. Не должно существовать центральных серверов таких служб, как управление транзакциями, выявление взаимных блокировок, оптимизация запросов и управление глобальным системным каталогом.

•3. Непрерывное функционирование

•В идеале, в системе никогда не должна возникать потребность в плановом останове ее функционирования для выполнения таких операций, как: добавление или удаление сервера из системы; динамическое создание или удаление фрагментов из одного или нескольких серверов.

•4. Независимость от расположения

•Пользователь должен получать доступ к базе данных с любого из серверов, причем так, как если бы они хранились на его локальном сервере.

•5.Независимость от фрагментации

•Пользователь должен получать доступ к данным независимо от способа их фрагментации.

•6. Независимость от репликации

•Пользователь не должен нуждаться в сведениях о наличии репликации данных. Это значит, что пользователь не должен заботиться об обновлении всех имеющихся копий элемента данных.

•7. Обработка распределенных запросов

•Система должна поддерживать обработку запросов, ссылающихся на данные, расположенные на более чем одном сервере.

•8. Обработка распределенных транзакций

•Система должна поддерживать выполнение транзакций, как единицы восстановления. Система должна гарантировать, что выполнение как глобальных, так и локальных транзакций будет происходить с сохранением четырех основных свойств транзакций, а именно: атомарности, согласованности, изолированности и продолжительности.

•9.Независимость от типа оборудования

•СУРБД должна быть способна функционировать на оборудовании с различными вычислительными платформами.

•10. Независимость от операционной системы

•СУРБД должна быть способна функционировать под управлением различных операционных систем.

•11. Независимость от сетевой архитектуры

•СУРБД должна быть способна функционировать в сетях с различной архитектурой и типами носителя.

•12.Независимость от типа СУБД

•СУРБД должна быть способна функционировать поверх различных локальных СУБД, возможно, с разным типом используемой модели данных (т.е. СУРБД должна поддерживать гетерогенность).

•Достоинства технологии распределенных систем:

•1. Отражение структуры организации.

•Географическая распределенность организации может быть отражена в распределении ее данных, причем пользователи одного сайта смогут получать доступ к данным, сохраняемым на других сайтах.

•2. Разделяемость и локальная автономность.

•Данные могут быть помещены на тот сервер, на котором пользователи их чаще всего используют. В результате заинтересованные пользователи получают локальный контроль над требуемыми им данными и могут устанавливать или регулировать локальные ограничения на их использование.

•3. Повышение доступности и надежности данных.

•В централизованных СУБД отказ центрального компьютера вызывает прекращение функционирования всей СУБД. Отказ одного из серверов СУРБД или линии связи между ними сделает недоступным лишь некоторые серверы, тогда как вся система в целом сохранит свою работоспособность. Распределенные СУБД проектируются таким образом, чтобы обеспечивать продолжение функционирования системы, несмотря на подобные отказы. Если организована репликация данных, в результате чего данные и их копии будут размещены на более чем одном сервере, отказ отдельного узла или линии связи между узлами не приведет к недоступности данных. Система сможет перенаправить запросы к отказавшему узлу в адрес другого сайта.

•4. Повышение производительности

•5. Модульность системы

•В распределенной среде расширение существующей системы осуществляется намного проще. Добавление в сеть нового сервера не оказывает влияния на функционирование уже существующих. Подобная гибкость позволяет организации легко расширяться. В централизованных СУБД рост размера базы данных может потребовать замены и оборудования и используемого программного обеспечения.

•Недостатки технологии распределенных систем

•1. Повышение сложности

•Распределенные СУБД, способные скрыть от конечных пользователей распределенную природу используемых ими данных и обеспечить необходимый уровень производительности, надежности и доступности, безусловно, являются более сложными программными комплексами, чем централизованные СУБД. Тот факт, что данные могут подвергаться репликации, также добавляет дополнительный уровень сложности в программное обеспечение СУРБД. Если репликация данных не будет поддерживаться на требуемом уровне, система будет иметь более низкий уровень доступности данных, надежности и производительности, чем централизованные системы, а все изложенные выше преимущества превратятся в недостатки.

•2. Увеличение стоимости

•3. Проблемы защиты

•В распределенных системах потребуется организовать контроль доступа не только к данным, реплицируемым на несколько различных сайтов, но и защиту сетевых соединений самих по себе.

•4. Усложнение контроля за целостностью данных.

•5. Отсутствие стандартов.

•Особенности распределенных СУБД

•Рассмотрим факторы, которые должны приниматься во внимание при разработке распределенных реляционных баз данных:

- **Фрагментация.** Любое отношение может быть разделено на некоторое количество частей, называемых фрагментами, которые затем распределяются по различным серверам. Существуют два основных типа фрагментов: *горизонтальные* и *вертикальные.* Горизонтальные фрагменты представляют собой подмножества кортежей, а вертикальные –подмножества атрибутов.

- **Распределение.** Каждый фрагмент сохраняется на сервере, выбранном с учетом «оптимальной» схемы их размещения.

- **Репликация.** СУРБД может поддерживать актуальную копию некоторого фрагмента на нескольких различных сайтах.

•*Репликацию* можно определить как процесс генерации и воспроизведения нескольких копий данных, размещаемых на одном или нескольких серверах. Использование репликации позволяет достичь многих преимуществ, включая повышение производительности (в тех случаях, когда централизованный ресурс оказывается перегруженным), надежности хранения и доступности данных, наличие «горячей» резервной копии на случай восстановления, а также возможность поддержки мобильных пользователей и хранилищ данных.

•**Фрагментация данных.**

•1. Локальность ссылок

•Везде, где только это возможно, данные должны храниться как можно ближе к местам их использования. Если фрагмент используется на нескольких серверах, может оказаться целесообразным разместить на них его копии.

•2. Повышение надежности и доступности.

•Надежность и доступность данных повышаются за счет использования механизма репликации. В случае отказа одного из сайтов всегда будет существовать копия фрагмента, сохраняемая на другом сайте.

•3. Приемлемый уровень производительности.

•Неверное распределение данных будет иметь следствием возникновение в системе узких мест. В этом случае некоторый сервер оказывается просто завален запросами, что может вызвать существенное снижение производительности всей системы. В то же время неправильное распределение может иметь следствием неэффективное использование ресурсов системы.

•4. Минимизация расходов на передачу данных.

•Следует тщательно учитывать стоимость выполнения в системе удаленных запросов. Однако при обновлении реплицируемых данных внесенные изменения потребуется распространить на все сайты, имеющие копию обновленного отношения, что вызовет увеличение затрат на передачу данных.

•Компоненты СУРБД

•1. Локальная СУБД

•Представляет собой стандартную СУБД, предназначенную для управления локальными данными на каждом из сайтов, входящих в состав распределенной БД. Имеет свой собственный системный каталог, в котором содержится информация о данных, сохраняемых на этом сайте.

•2. Компонент передачи данных

•Представляет собой программное обеспечение, позволяющее всем серверам взаимодействовать между собой. Он содержит сведения о существующих серверах и линиях связи между ними.

•3. Глобальный системный каталог

•Содержит информацию, специфическую для распределенной природы системы, например, схемы фрагментации и распределения.

•4. Распределенная СУБД

•Является управляющим компонентом по отношению ко всей системе элементом.

•Требования к распределенные СУБД :

•1. Должна позволять конечным пользователям воспринимать базу данных как единое логическое целое. Если СУРБД обеспечивает прозрачность распределенности, то пользователю не требуется каких-либо знаний о фрагментации данных (прозрачность фрагментации) или их размещении (прозрачность расположения).

•2. Прозрачность транзакций

•При выполнении любых распределенных транзакций гарантируется сохранение целостности и согласованности распределенной базы данных

•3. Работа в среде СУРБД должна выполняться аналогично работе в централизованной СУБД. СУРБД должна находить наиболее эффективные стратегии выполнения запросов.

•4. Типичная СУРБД должна обеспечивать как минимум тот же набор функциональных возможностей, что и централизованная СУБД.

•5. Должна иметь расширенные службы установки соединений должны обеспечивать доступ к удаленным серверам и позволять передавать запросы и данные между серверами, входящими в сеть.

•6. Должна иметь расширенные средства ведения каталога, позволяющие сохранять сведения о распределении данных в сети.

•• Должна иметь средства обработки распределенных запросов, включая механизмы оптимизации запросов и организации удаленного доступа.

•• Должна иметь функции, позволяющие поддерживать целостность реплицируемых данных.

•• Расширенные функции восстановления, учитывающие возможность отказов в работе отдельных серверов и отказов линий связи.

Репликация

Репликация SQL Server представляет собой набор технологий, с помощью которых данные или объекты баз данных можно скопировать и перенести из одной базы данных в другую, а затем синхронизировать эти базы данных для обеспечения согласованности.

Модель публикации репликации

Репликация использует терминологию издательской отрасли для представления компонентов в топологии репликации.

- Publisher (Издатель) - сервер или база данных, которая посылает данные на другой сервер или в другую базу данных.
- Subscriber (Подписчик) - сервер или база данных, которая получает данные от другого сервера или другой базы данных.
- Distributor (Распространитель) - сервер, который управляет потоком данных через систему репликации. Этот сервер содержит специализированную базу данных: Distribution database.

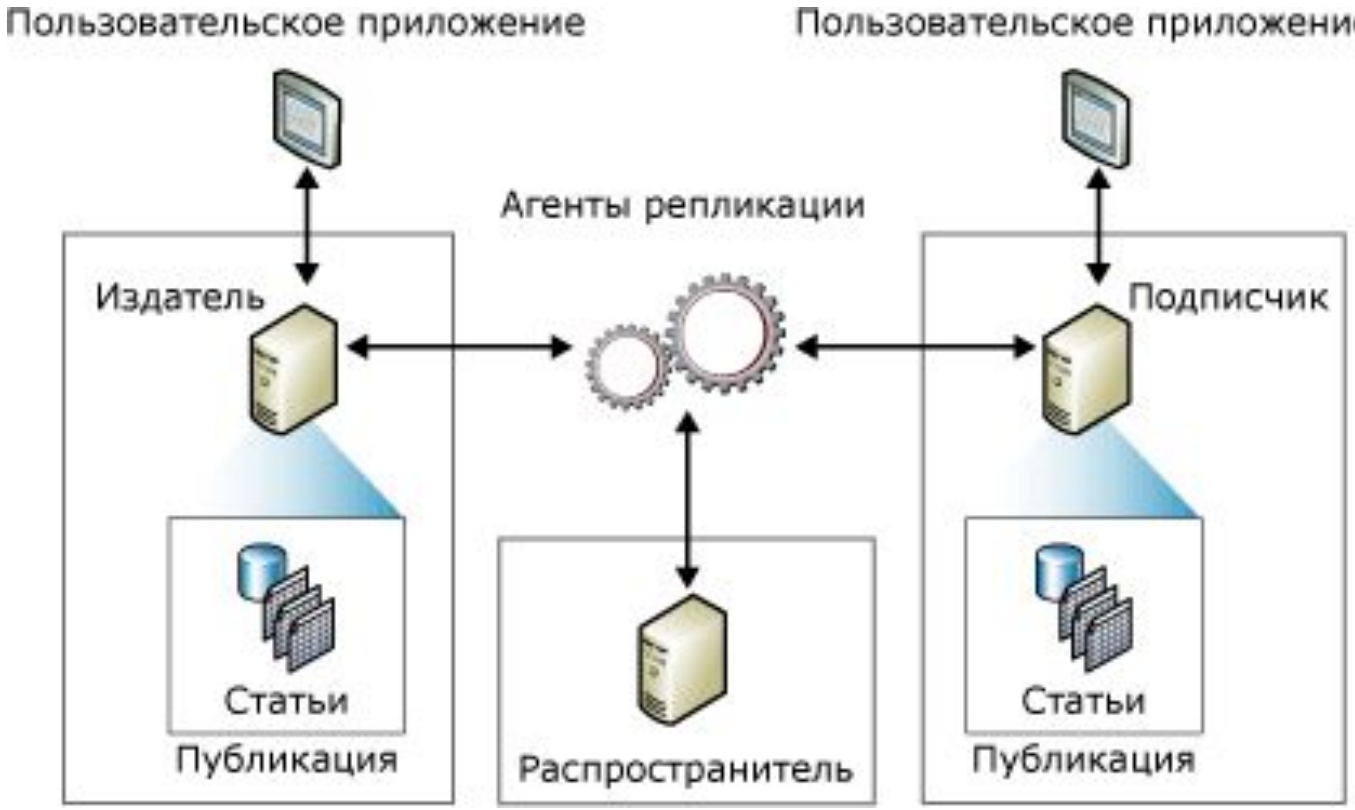
Publisher содержит публикацию/публикации. **Публикация** - это совокупность одной или более статей, которые посылаются серверу подписчику (subscriber) или базе данных. **Статья** (Article) - основной модуль репликации и это может быть таблица или подмножество таблицы. **Подписка** (subscriptions) - это группа данных, которые сервер или база данных получает.

Последовательность действий можно описать следующим образом.

- Издатель журнала производит одну или несколько публикаций
- Публикация содержит статьи
- Издатель или распространяет журнал напрямую, или использует распространитель
- Подписчики получают публикации, на которые они подписались

Хотя сравнение с журналом полезно для понимания репликации, следует отметить, что репликация SQL Server включает возможности, которые не представлены в данной метафоре, в частности возможность подписчика выполнять обновления и возможность издателя посылать дополнительные изменения в статьи публикации.

Топология репликации определяет отношения между серверами и копиями данных, и проясняет логику, определяющую порядок обмена данными между серверами. Существует несколько **процессов репликации (называемых *агентами*)**, которые отвечают за копирование и перемещение данных между издателем и подписчиками. Следующая иллюстрация представляет собой обзор компонентов входящих в репликацию.



Существуют три типа репликации:

- snapshot (репликация моментальных снимков),
- transactional (репликация транзакций)
- merge (репликация слиянием)

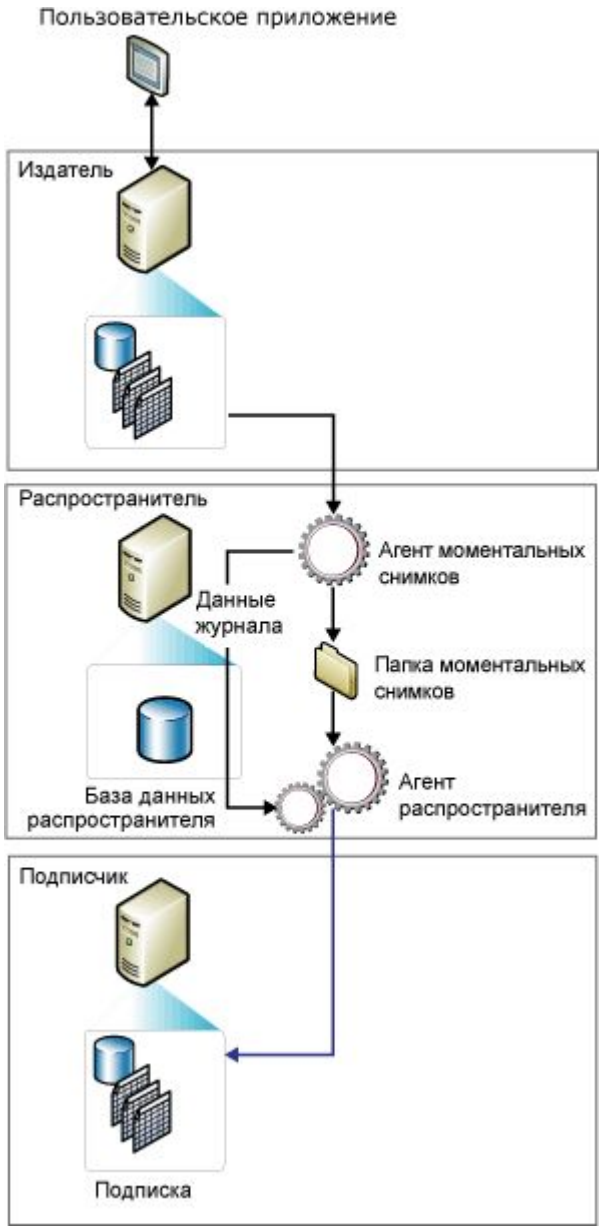
Репликация моментальных снимков распространяет данные точно в том виде, в котором они были представлены в определенный момент времени, и не контролирует обновления этих данных. Во время синхронизации формируется моментальный снимок и отсылается подписчикам целиком.

Использование репликации моментальных снимков самой по себе наиболее приемлемо, когда выполняется одно или несколько следующих условий:

- Данные изменяются редко.
- Допустимо на определенный период времени иметь копии данных, устаревших по отношению к издателю.
- Репликация небольших объемов данных.
- Большой объем изменений производится за короткий период времени.

Например, если торговая организация ведет прайс-лист на товары, и цены обновляются в одно и то же время раз или два раза в год, рекомендуется производить репликацию всего моментального снимка данных после их изменений. При наличии определенных типов данных наиболее подходящими могут быть более частые моментальные снимки. Например, если относительно небольшая таблица обновляется на издателе в течение дня, но допустима некоторая задержка, то изменения могут доставляться ночью в виде моментального снимка.

По умолчанию все три типа репликации для инициализации подписчиков используют моментальный снимок. Агент моментальных снимков SQL Server всегда создает файлы моментальных снимков, но агент, доставляющий файлы, меняется в зависимости от используемого типа репликации. Репликация моментальных снимков и репликация транзакций используют для доставки этих файлов **агент распространителя**, в то время как репликация слиянием использует **агент слияния** SQL Server. Агент моментальных снимков выполняется на распространителе. Агент распространителя и агент слияния выполняются на распространителе для принудительных подписок и на подписчиках для подписок по запросу.



Моментальные снимки могут создаваться и применяться или непосредственно после создания подписки, или в соответствии с расписанием, установленным при создании публикации.

Агент моментальных снимков подготавливает файлы моментального снимка, содержащие схему и данные опубликованных таблиц и объектов базы данных, сохраняет эти файлы в папке моментальных снимков для издателя и записывает данные слежения в базу данных распространителя на распространителе.

При настройке распространителя указывается папка моментальных снимков по умолчанию, но можно указать и другое расположение публикации вместо или дополнительно к расположению по умолчанию.

Основные компоненты репликации моментальных снимков.

Репликация транзакций обычно используется в серверных средах и пригодна в следующих случаях:

- Необходимо, чтобы добавочные изменения распространялись подписчикам без задержек по мере появления.
- Для приложения необходимы малые задержки между моментом внесения изменений на издателе и моментом прибытия изменений на подписчик.
- Для приложения необходим доступ к промежуточным состояниям данных. Например, если строка изменяется пять раз, репликация транзакций позволяет приложению реагировать на каждое изменение (например, срабатывание триггера), а не просто на итоговое изменение строки.
- На издателе выполняется очень большой объем вставок, обновлений и удалений.
- Издатель и подписчик являются базами данных, отличными от баз данных SQL Server (например, Oracle).

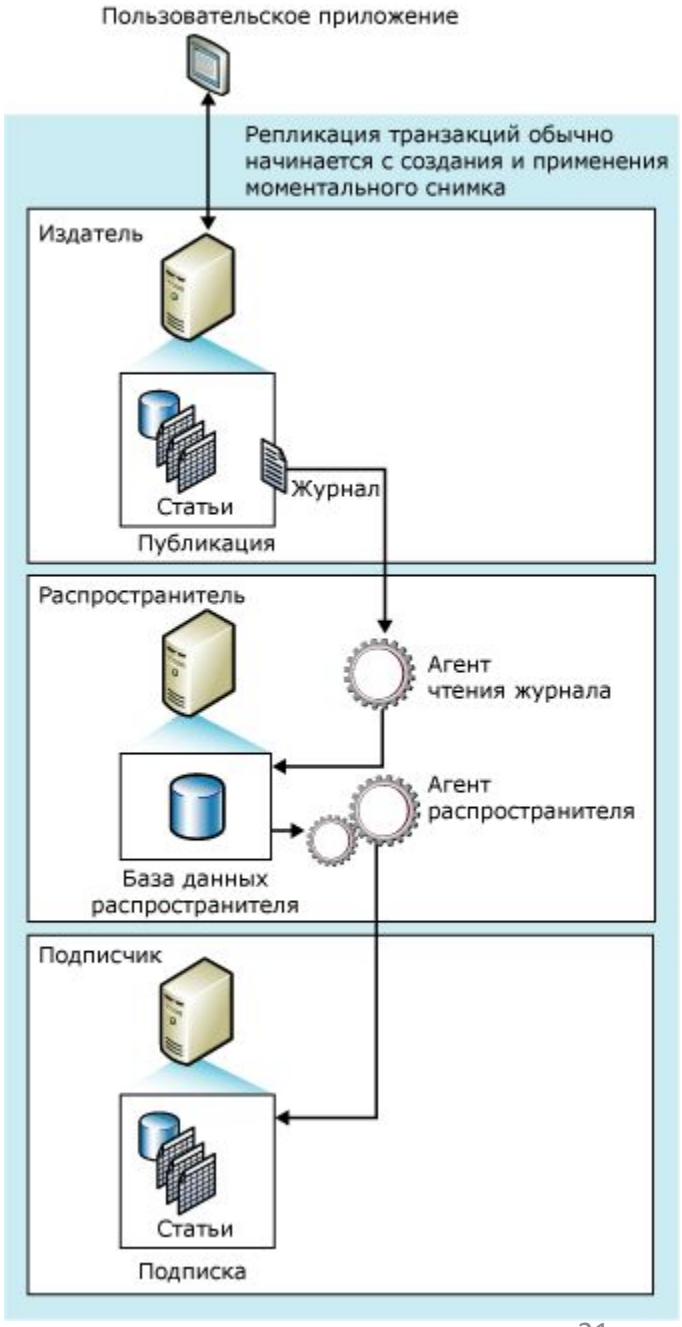
По умолчанию подписчики на публикации транзакций должны быть доступны только для чтения, так как изменения не распространяются обратно на издатель. Однако репликация транзакций предоставляет возможности, позволяющие выполнять обновления у подписчика.

Репликация транзакций реализуется агентом моментальных снимков, агентом чтения журналов и агентом распространителя SQL Server. Агент моментальных снимков готовит файлы моментальных снимков, содержащие схему, данные публикуемых таблиц и объекты базы данных, хранит файлы в папке моментальных снимков и записывает задания синхронизации в базу данных распространителя на распространителе.

Агент чтения журнала контролирует журнал транзакций всех баз данных, настроенных для репликации транзакций, и копирует транзакции, отмеченные для репликации, из журнала транзакций в базу данных распространителя, которая действует как надежная очередь с функциями хранения и переадресации данных. Агент распространителя копирует файлы исходного моментального снимка из папки моментальных снимков и транзакции, хранимые в таблицах базы данных распространителя, на подписчики.

Добавочные изменения, вносимые на издателе, поступают подписчикам согласно расписанию агента распространителя, который может выполняться непрерывно для достижения минимальной задержки либо в запланированные интервалы времени. Поскольку изменения данных должны вноситься на издателе (когда репликация транзакций используется без немедленного обновления и отложенного обновления), возникновение конфликтов обновления исключается.

В конечном счете, для всех подписчиков устанавливаются такие же значения, как и на издателе. Если с репликацией транзакций используется немедленное или отложенное обновление, в этом случае обновления могут вноситься на подписчике, и для отложенного обновления возможно возникновение конфликтов.



Репликация слиянием, как и репликация транзакций, как правило, начинается с моментального снимка объектов и данных базы данных публикации. Последующие изменения данных и схемы, произведенные на стороне издателя и подписчиков, отслеживаются при помощи триггеров. Подписчик синхронизируется с издателем при подключении к сети и обменивается с ним всеми строками, которые изменились со времени последней синхронизации издателя и подписчика.

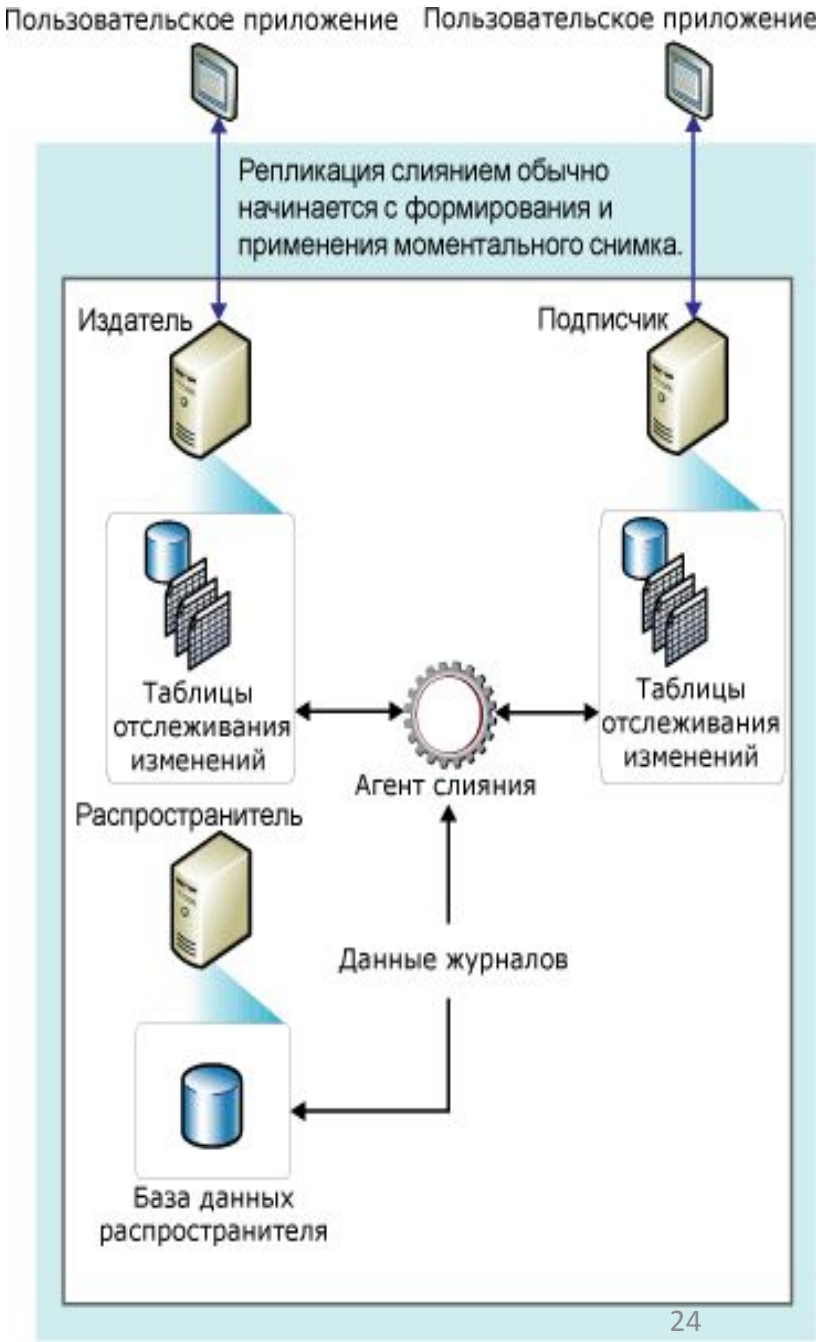
Как правило, репликация слиянием применяется в средах «сервер-клиент». Репликация слиянием подходит для любой из следующих ситуаций.

- Несколько подписчиков могут обновлять одни и те же данные в разное время и передавать эти изменения на издатель и на другие подписчики.
- Подписчикам нужно получить данные, внести изменения в автономном режиме и позднее синхронизировать изменения с издателем и другими подписчиками.

- Каждому подписчику нужна индивидуальная секция данных.
- Поскольку возможно возникновение конфликтов, необходимы средства по распознаванию и разрешению конфликтов.
- Приложению требуется конечное изменение данных, а не доступ к промежуточным состояниям данных. Например, если строка меняется пять раз на подписчике до его синхронизации с издателем, на издателе строка изменится только один раз для отображения конечного изменения данных (то есть, пятого значения).

Репликация слиянием позволяет различным узлам работать автономно, и позднее выполнить слияние обновлений в единый результат. Поскольку обновления выполняются на нескольких узлах, одни и те же данные могут быть обновлены издателем и несколькими подписчиками. Поэтому при слиянии обновлений могут возникать конфликты, и репликация слиянием предоставляет несколько способов обработки конфликтов.

Репликация слиянием реализуется агентом моментальных снимков SQL Server и агентом слияния. Если публикация не отфильтрована или использует статические фильтры, агент слияния создает один моментальный снимок. Если публикация использует параметризованные фильтры, то агент слияния создает моментальный снимок для каждой секции данных. Агент слияния применяет все исходные моментальные снимки на подписчиках. Он также объединяет добавочные изменения данных, которые возникли на издателе или подписчиках после создания исходного моментального снимка, выявляет и разрешает любые конфликты в соответствии с заданными вами правилами.



Компоненты, используемые при репликации слиянием.

Повышение производительности репликации

Перед настройкой репликации рекомендуется ознакомиться с факторами, влияющими на производительность репликации:

- серверное и сетевое аппаратное обеспечение;
- структура базы данных;
- конфигурация распространителя;
- структура и параметры публикации;
- структура фильтра и его использование;
- параметры подписки;
- параметры моментального снимка;
- параметры агентов.
- обслуживание

После настройки репликации рекомендуется разработать базовый уровень производительности, позволяющий определить работу репликации при рабочей нагрузке, типичной для используемых приложений и топологии. Следует применить монитор репликации и системный монитор для определения типичных значений следующих пяти измерений производительности репликации:

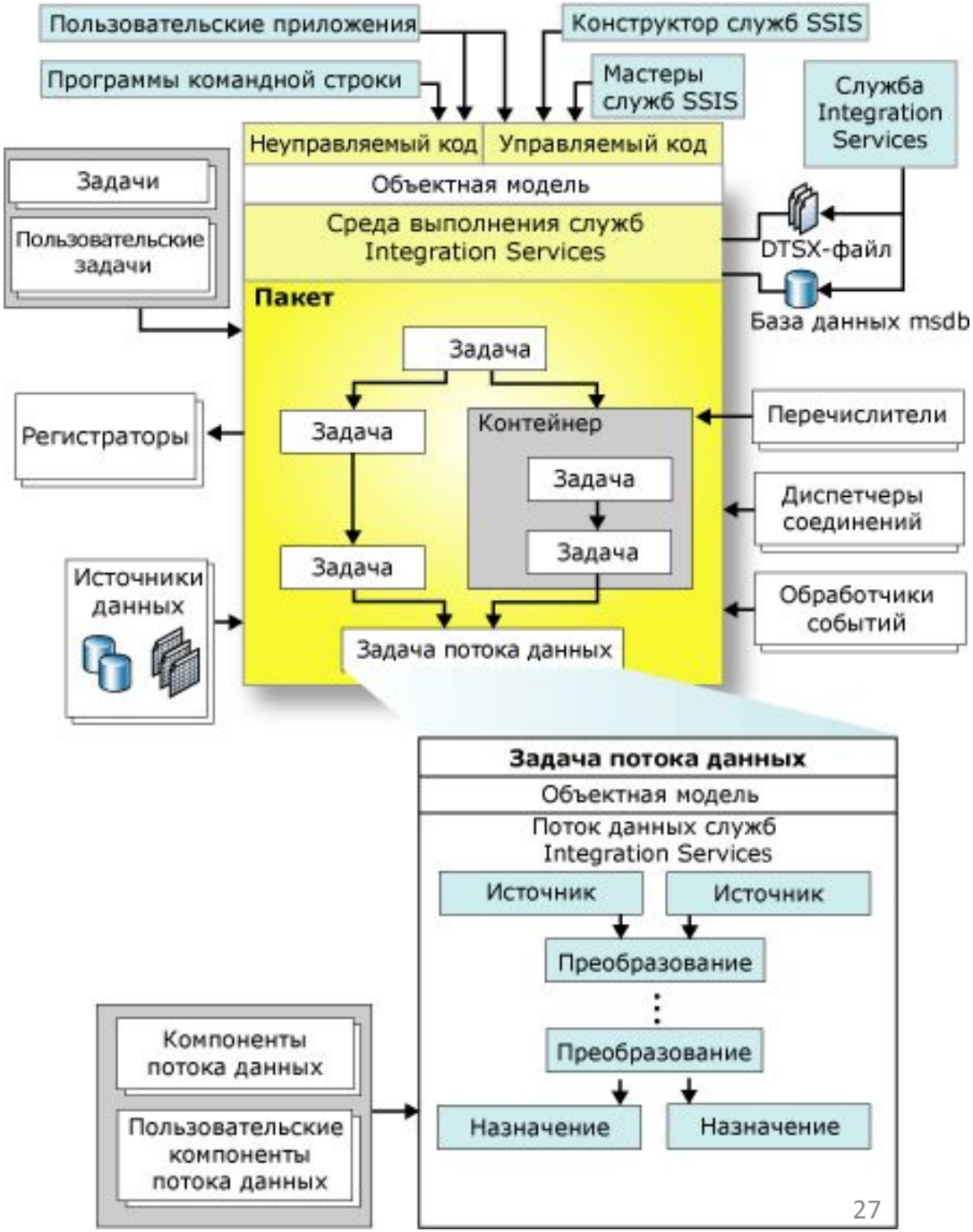
- Задержка: время распространения изменений данных между узлами в топологии репликации.
- Пропускная способность: величина репликационной активности (измеряемая в командах, доставленных за период времени), поддерживаемой системой.
- Параллелизм: число процессов репликации, которые могут выполняться системой одновременно.
- Длительность синхронизации: время, затрачиваемое на выполнение заданной синхронизации.
- Потребление ресурсов: аппаратные и сетевые ресурсы, используемые для обработки репликации.

Задержка и пропускная способность наиболее полно характеризуют репликацию транзакций, поскольку для систем, основанных на репликации транзакций, обычно требуется малая задержка и высокая пропускная способность.

Параллелизм и длительность синхронизации наиболее точно характеризуют репликацию слиянием, поскольку системы, основанные на репликации слиянием, часто имеют большое число подписчиков, а издатель может иметь значительное число параллельно выполняющихся синхронизаций с этими подписчиками.

SSIS (SQL Server Integration Services)

Службы Integration Services предлагают широкий выбор встроенных задач, контейнеров, преобразований и адаптеров обработки данных, поддерживающих разработку бизнес-приложений.



Компоненты службы Microsoft SQL Server Integration Services

Сценарии типичных случаев применения пакетов служб SSIS:

Слияние данных из разнородных хранилищ данных. Данные обычно хранятся во множестве различных систем хранилищ данных, и извлечение данных из всех источников и их слияние в единый согласованный набор данных может представлять собой довольно сложную задачу.

Заполнение хранилищ данных и витрин данных. Данные в хранилищах и витринах данных обновляются часто, а объем загружаемых данных обычно довольно велик.

Очистка и стандартизация данных. Перед загрузкой данных для OLTP или в базу данных OLAP, Excel или файл, данные необходимо очистить и стандартизировать.

Встраивание бизнес-аналитики в процесс преобразования данных. Процесс преобразования данных требует от встроенной логики динамичного реагирования на данные для обработки и доступа. Данным может требоваться суммирование, преобразование, распространение на основе их значений. Процесс может также отвергнуть данные на основе оценки значений столбцов.

Автоматизация административных функций и загрузки данных

Администраторы часто желают автоматизировать административные функции, такие как резервное копирование и восстановление баз данных, копирование баз данных SQL Server и содержащихся в них объектов, копирование объектов SQL Server и загрузку данных. Пакеты служб Integration Services могут выполнять эти функции.

SQL Server Service Broker

Компонент SQL Server Service Broker обеспечивает поддержку компонента SQL Server Database Engine для приложений обмена сообщениями и приложений с очередями сообщений. Это облегчает разработчикам создание сложных приложений, использующих компоненты Database Engine для связи между разнородными базами данных. Разработчики могут использовать компонент Service Broker для облегчения создания распределенных и надежных приложений.

Компонент Service Broker помогает создавать асинхронные слабосвязанные приложения, в которых независимые компоненты совместно выполняют ту или иную задачу. Эти компоненты обмениваются сообщениями, которые содержат данные, необходимые для выполнения задачи.

Приложения компонента Service Broker состоят из объектов базы данных Service Broker и одного или нескольких приложений, использующих эти объекты. Существует три типа компонентов Service Broker:

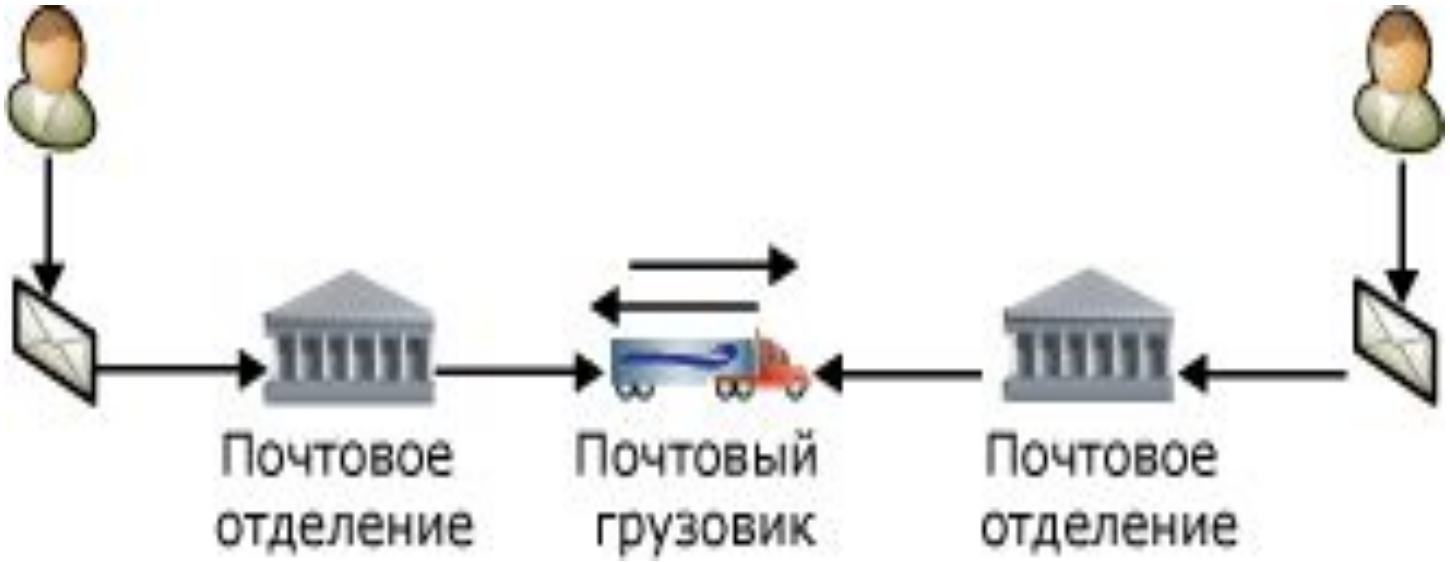
- Компоненты диалога. Структурные элементы времени исполнения диалога. В процессе диалога приложения обмениваются сообщениями.
- Объекты определения служб. Это компоненты времени проектирования, устанавливающие базовую конструкцию приложения. Эти компоненты определяют для приложения тип сообщений, поток диалога и хранение сообщений в базе данных.
- Компоненты маршрутизации и безопасности. Эти компоненты определяют инфраструктуру для обмена сообщениями между базами данных и экземплярами компонента Database Engine.

Работа компонента Service Broker основана на отправке и получении сообщений. Каждое сообщение является частью *диалога*. Каждый диалог — это надежный, постоянный канал коммуникации. Каждое сообщение и каждый диалог имеет конкретный тип, за соблюдением которого следит компонент Service Broker, помогая разрабатывать надежные приложения.

Приложение отправляет сообщения *службе,* которая представляет собой набор связанных задач. Приложение получает сообщения из *очереди* — представления внутренней таблицы.

Сообщения, адресованные одной и той же задаче, относятся к одному диалогу. При обслуживании каждого диалога компонент Service Broker гарантирует, что приложение получит каждое сообщение только один раз в соответствии с порядком отправления сообщений..

Для защиты конфиденциальных сообщений и управления доступом к службам применяется механизм обеспечения безопасности на основе сертификатов. По аналогии, компонент Service Broker можно считать похожим на почтовую службу. Для диалога с удаленным коллегой можно использовать письма, отправляя их при помощи почтовой службы. Эта служба сортирует и доставляет письма. Получив очередное письмо, вы достаете его из почтового ящика, читаете, пишете ответ и отправляете новое письмо. Ваш коллега делает то же самое, и это продолжается до завершения диалога. Доставка писем осуществляется «асинхронно» — пока вы с коллегой решаете другие задачи.



Программы могут использовать компонент Service Broker в качестве почтовой службы, чтобы поддерживать асинхронные диалоги с другими программами. Сообщения компонента Service Broker функционируют как письма. Обслуживаемая компонентом Service Broker служба — это адрес, по которому почтовая служба доставляет письма. Очереди — это почтовые ящики, в которых хранятся доставленные письма. Приложения получают сообщения, обрабатывают их и отправляют ответы.

Сообщение, отправленное приложением службе компонента Service Broker, является изолированным от деталей реализации приложения, находящегося на другом конце диалога. Принимающее приложение может быть динамически переконфигурировано или заменено новым кодом; при этом изменять отправляющее приложение нет необходимости. Принимающее приложение можно даже временно отключить; единственное, на что это повлияет — это на то, что компонент Service Broker будет продолжать добавлять новые сообщения в очередь, пока принимающее приложение не будет перезапущено.