Репликация

Общие сведения и примеры реализации

Репликация данных

Репликация – это поддержание двух и более идентичных копий (реплик) данных на разных узлах РБД.

Реплика может включать всю базу данных (полная репликация), одно или несколько взаимосвязанных отношений или фрагмент отношения.

Достоинства репликации:

– повышение доступности и надежности данных;

– повышение локализации ссылок на реплицируемые данные.

Недостатки репликации:

– сложность поддержания идентичности реплик;

– увеличение объема памяти для хранения данных.

Поддержание идентичности реплик называется **распространение изменений** и реализуется **службой тиражирования**.

Служба тиражирования

Служба тиражирования должна выполнять следующие функции:

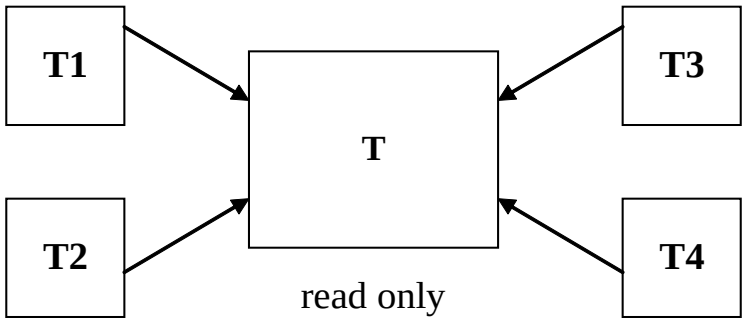
- ✓ Обеспечение масштабируемости, т.е. эффективной обработки больших и малых объемов данных.
- ✓ Преобразование типов и моделей данных (для гетерогенных РБД).
- ✓ Репликация объектов БД, например, индексов, триггеров и т.п.
- ✓ Инициализация вновь создаваемой реплики.
- ✓ Обеспечение возможности "подписаться" на существующие реплики, чтобы получать их в определенной периодичностью.

Для выполнения этих функций в языке, поддерживаемом СУБД, предусматривается наличие средств определения схемы репликации, механизма подписки и механизма инициализации реплик (создания и заполнения данными).

Репликация с основной копией

Существуют следующие варианты:

1. **Классический подход** заключается в наличии одной основной копии, в которую можно вносить изменения; остальные копии создаются с определением read only.
2. **Асимметричная репликация**: основная копия фрагментирована и распределена по разным узлам РБД, и другие узлы могут являться подписчиками отдельных фрагментов (read only).
3. **Рабочий поток**. При использовании этого подхода право обновления не принадлежит постоянно одной копии, а переходит от одной копии в другой в соответствии с потоком операций. В каждый момент времени обновляться может только одна копия**.**
4. **Консолидация данных:**



Репликация без основной копии

Симметричная репликация (без основной копии). Все копии реплицируемого набора могут обновляться одновременно и независимо друг от друга, но все изменения одной копии должны попасть во все остальные копии.

Существует два основных механизма распространения изменений при симметричной репликации:

- **синхронный:** изменения во все копии вносятся в рамках одной транзакции;
- **асинхронный:** подразумевает отложенный характер внесения изменений в удаленные копии.

Достоинство синхронного распространения изменений – полная согласованность копий и отсутствие конфликтов обновления.

Недостатки:

– трудоемкость и большая длительность модификации данных,

– низкая надежность работы системы.

Репликация без основной копии

Конфликтные ситуации:

- Добавление двух записей с одинаковыми первичными или уникальными ключами. Для предотвращения таких ситуаций обычно каждому узлу РБД выделяется свой диапазон значений ключевых (уникальных) полей.
- Конфликты удаления: одна транзакция пытается удалить запись, которая в другой копии уже удалена другой транзакцией. Если такая ситуация считается конфликтом, то она разрешаются вручную.
- Конфликты обновления: две транзакции в разных копиях обновили одну и ту же запись, возможно, по-разному, и пытаются распространить свои изменения. Для идентификации конфликтов обновления необходимо передавать с транзакцией дополнительную информацию: старое и новое содержимое записи. Если старая запись не может быть обнаружена, налицо конфликт обновления.

Репликация без основной копии

Методы разрешения конфликтов обновления:

1. Разрешение **по приоритету узлов**: для каждого узла назначается приоритет, и к записи применяется обновление, поступившее с узла с максимальным приоритетом.
2. Разрешение **по временной отметке**: все транзакции имеют временную отметку, и к записи применяется обновление с минимальной или максимальной отметкой. Использовать ли для этого минимальную или максимальную отметку – зависит от предметной области и, обычно, может регулироваться.
3. **Аддитивный метод** (add – добавить): может применяться в тех случаях, когда изменения основаны на предыдущем значении поля, например, salary = salary + X. При этом к значению поля последовательно применяются все обновления.
4. Использование **пользовательских процедур**.
5. Разрешение конфликтов **вручную**. Сведения о конфликте записываются в журнал ошибок для последующего анализа и устранения администратором.

Репликация без основной копии

Способы реализации распространения изменений:

1. Использование триггеров.
   Внутрь триггера помещаются команды, проводящие на других копиях обновления, аналогичные тем, которые вызвали выполнение триггера.
   Этот подход достаточно гибкий, но он обладает рядом недостатков:

- триггеры создают дополнительную нагрузку на систему;
- триггеры не могут выполняться по графику (время срабатывания триггера не определено);
- с помощью триггеров сложнее организовать групповое обновление связанных таблиц (из-за проблемы мутирующих таблиц).

2. Поддержка журналов изменений для реплицируемых данных. Рассылка этих изменений входит в задачу сервера СУБД или сервера тиражирования (входящего в состав СУБД). Основные принципы, которых необходимо придерживаться при этом:

- Для сохранения согласованности данных должен соблюдаться порядок внесения изменений.
- Информация об изменениях должна сохраняться в журнале до тех пор, пока не будут обновлены все копии этих данных.

Репликация с основной копией

Способы реализации распространения изменений:

1. Выгрузка (дамп) и повторная загрузка.

Данные извлекаются из единственного источника и загружаются в один или более приемников.

Недостатки:

- Время между появлением новых данных и их распространением может измеряться часами, днями и более длительными периодами.
- Этот подход обеспечивает одностороннюю репликацию без своевременного удаленного обновления.

2. Снимки (snapshot).

В определенные моменты времени делаются снимки базы данных, которые затем загружаются в один или более серверов-приемников.

Достоинства: простота реализации.

Недостатки:

- Получатели работают с относительно устаревшими данными, что делает данный подход неприемлемым в случаях, когда требуется информация в реальном времени.
- Этот способ не обеспечивает удаленного обновления.

Репликация в СУБД Oracle

Решаемые задачи

Репликация используется для решения следующих задач:

1) Распространение информации с одного сервера на несколько серверов (например, можно из центрального офиса передавать в филиалы новые цены и изменения в справочниках).
2) Сбор информации с нескольких серверов на один сервер (например, можно собирать отчеты о платежах, выполненные в филиалах).
3) Для снятия нагрузки с рабочего сервера и для повышения отказоустойчивости (можно установить резервный сервер и через определенные интервалы времени реплицировать на него данные с рабочего сервера).

Впрочем, последнюю задачу рациональнее решать с помощью технологий резервирования или опции Real Application Cluster.

Репликация имеет смысл, когда:

- Нужно синхронизировать несколько таблиц, а не всю БД.
- Данные синхронизируются с интервалом от 10 минут и больше.

Если заказчику необходимо обеспечивать полностью идентичные копии баз данных и/или данные нужно синхронизировать быстрее, то проще использовать технологии резервирования.

Репликация в СУБД Oracle

В СУБД Oracle репликация реализована несколькими технологиями:
1) Моментальные снимки (snapshot).
2) Oracle Streams.

Репликация на основе моментальных снимков

Существует два вида репликации, основанных на моментальных снимках: Basic и Advanced.

Особенности Basic репликации:

- ✓ Доступна во всех редакциях Oracle
- ✓ Реплицируются только данные
- ✓ Репликация производится только в одну сторону
- ✓ В исходной базе данных обычные таблицы
- ✓ В базе, в которую реплицируют, находятся не таблицы, а доступные только на чтение Snapshots (Read Only Materialized Views)
- ✓ Работает на основе триггеров
- ✓ После 16 неудачных попыток передачи данных подряд процесс останавливается и требует вручную перезапустить JOB

Моментальные снимки в Oracle

Oracle поддерживает два типа тиражирования:

- ✓ базовое – копия обеспечивает доступ "только для чтения".
- ✓ усовершенствованное – приложения могут считывать и обновлять тиражируемые копии таблиц по всей системе (поддерживается специальными средствами СУБД – **Replication Option**).

Базовое тиражирование осуществляется (после установления связи с удаленной БД) с помощью создания моментальных снимков (snapshot), например:

```
CREATE SNAPSHOT sales.parts AS
        SELECT * FROM sales.parts@central.compworld;
```

Моментальные снимки бывают:

- ✓ простые – создаются по однотабличному запросу SELECT, содержащему простые условия отбора.
- ✓ сложные – создаются по запросам, содержащим сложные условия отбора, фразы group by, having, обращающимся к двум и более таблицам и проч.

Моментальные снимки в Oracle

Примеры:

Моментальный снимок, основой которого является запрос

```
select * from employee@hr_link;
```

является простым.

Моментальный снимок, основанный на запросе

```
select dept, max(salary)
        from employee@hr_link
        group by dept;
```

сложный, так как в нем используются функции группирования.

С помощью моментального снимка в локальной базе данных будет создано несколько объектов, поэтому пользователь, создающий моментальный снимок, должен иметь привилегии CREATE TABLE, CREATE VIEW и CREATE INDEX.

Моментальные снимки в Oracle

Синтаксис создания моментального снимка:

create snapshot [*имя_схемы*.]*имя_снимка*
[{ pctfree *целое* | pctused *целое* | initrans *целое* |
maxtrans *целое* | tablespace *имя_табличной_области* |
storage *размер_памяти* }]
[cluster *имя_кластера* (*имя_столбца*[,…])]
[using index]
[{ pctfree *целое* | pctused *целое* | initrans *целое* |
maxtrans *целое* | tablespace *имя_табличной_области* |
storage *размер_памяти* }]
[refresh [{ fast | complete | force }]
[start with *дата_1*] [next *дата_2*]]
[for update]
as *запрос*;

Моментальные снимки в Oracle

Пример создания МС на локальном сервере:

```
create snapshot emp_dept_count
    pctfree 5
    tablespace snap
    storage (initial 100k next 100k pctincrease 0)
    refresh complete
    start with sysdate
    next sysdate+7
    as select deptno, count(*) dept_count
        from employee@hr_link
        group by deptno;
```

Моментальные снимки в Oracle

При создании моментального снимка в локальной базе данных создается:

- ✓ **таблица** для хранения записей, получаемых в результате выполнения запроса моментального снимка (с именем *SNAP$_имя_моментального_снимка)*;
- ✓ **представление** этой таблицы "только для чтения", называемое в соответствии с именем моментального снимка;
- ✓ **представление**, называемое MVIEW$_имя_моментального_снимка – для обращения к удаленной основной таблице (или таблицам). Это представление будет использоваться во время регенерации.

Для модификации снимка, например, с целью установки частоты автоматического изменения в 1 час можно воспользоваться командой ALTER SNAPSHOT:

```
alter snapshot emp_dept_count refresh complete
        start with sysdate next sysdate + 1/24;
```

Для удаления моментальных снимков применяется команда drop snapshot:

```
drop snapshot emp_dept_count;
```

Регенерация моментальных снимков Oracle

Возможны два варианта:

1. REFRESH FAST (**быстрая регенерация**).
2. REFRESH COMPLETE (**полная регенерация**).

Режим регенерации	Описание
COMPLETE	Таблицы моментального снимка полностью восстанавливаются с помощью его запроса и основных таблиц при каждой регенерации
FAST	Если применяется простой моментальный снимок, то для посылки только тех изменений, которые внесены в его таблицу, можно использовать журнал моментальных снимков
FORCE	Значение по умолчанию. Если это возможно, выполняется быстрая (FAST) регенерация, если нет – полная (COMPLETE) регенерация

Регенерация моментальных снимков Oracle

Для быстрой регенерации необходим **журнал моментальных снимков** (snapshot log) – это таблица, обеспечивающая регистрацию в моментальном снимке изменений, происшедших в основной таблице. Имя журнала (таблицы) – MLOG$_имя_таблицы.

Команда CREATE SNAPSHOT LOG. Пример:

```
create snapshot log on employee
        tablespace data
        storage (initial 10k next 10k pctincrease 0);
```

Изменения в журнал моментальных снимков попадают с помощью триггера AFTER типа FOR EACH ROW, который называется TLOG$_имя_таблицы.

В журнале моментальных снимков данные находятся очень непродолжительное время: записи вводятся в журнал моментальных снимков, используются во время регенерации, а затем удаляются из журнала автоматически.

Усовершенствованное тиражирование Oracle

Производится с помощью двух средств Oracle:

1. Многоабонентского тиражирования.
2. Узлов обновляемых моментальных снимков.

Распространение изменений:

1. на уровне строк: сервер записывает изменения, сделанные каждой DML-транзакцией, и рассылает эти изменения в удаленные узлы.
2. путем процедурного тиражирования: тиражируется вызов удаленной процедуры, выполняющей в удаленном узле те же изменения, что и в вызывающем.

Различают **асинхронное** и **синхронное** распространение изменений.

Внесение изменений в тиражируемые данные происходит в несколько этапов:

- ✓ локальный узел вносит изменения в свою копию данных (ОМС);
- ✓ локальный узел запускает отложенную транзакцию на основном узле;
- ✓ через некоторое время локальный узел выполняет быструю (или полную) регенерацию локальной копии данных, после чего приложение всегда может проверить, выполнена ли инициированная им транзакция. Если она не выполнена, то происходит рестарт транзакции и все повторяется.

Репликация в СУБД Oracle

Advanced-репликация поддерживает различные конфигурации: репликация в обе стороны, репликация со многими первичными серверами и т.д. Позволяет реплицировать не только данные, но и другие объекты базы данных. Доступна только в Enterprise Edition.

Advanced-репликация, начиная с Enterprise Edition 11g, позволяет передавать изменение структуры реплицируемых объектов в автоматическом режиме без остановки БД.

Репликация на основе Oracle Streams обладает следующими преимуществами:

- ✓ Работает быстрее (данные находятся в памяти: Streams pool).
- ✓ Репликация на уровне отдельных таблиц, схем, табличных пространств.
- ✓ Репликация между разными версиями базы данных и даже между разными платформами.
- ✓ Возможность фильтрации данных на основе правил.
- ✓ Поддерживает как одностороннюю, так и двухстороннюю репликацию.
- ✓ Поддерживает синхронную (начиная с версии 11g) или асинхронную репликацию.

Репликация в СУБД Oracle

Технология Oracle Streams

Oracle Streams – универсальный гибкий механизм обмена информацией между серверами в много серверной архитектуре (MTS). Позволяет одновременно реализовать репликацию, обмен сообщениями, загрузку хранилищ данных, работу с событиями, поддержку резервной БД. Данные следуют по определенным пользователем маршрутам и доставляются к месту назначения. В результате получается механизм, который обеспечивает большую функциональность и гибкость, чем традиционные решения для хранения и распространения данных, а также совместного их использования с другими базами данных и приложениями.

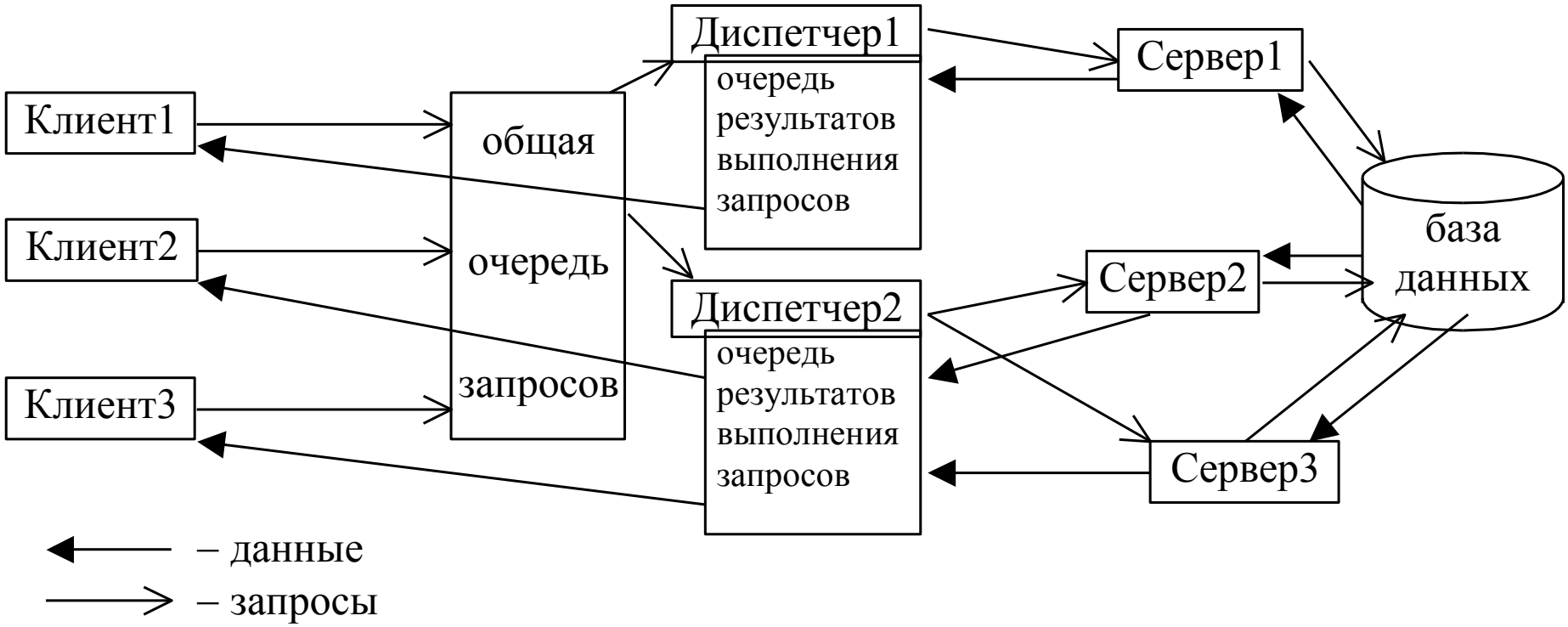
Oracle Streams – отдельная информационная инфраструктура, которая состоит из процессов *capture*, *propagation* и *apply* информации.

LCR, CR

В контексте Oracle Streams информационное представление любого изменения, сделанного в базе данных, называется *LCR* (logical change record). *LCR* – это обобщенное представление всех возможных изменений, представленных в базе данных.

CR (change record) – запись изменения, используется для того, чтобы обозначить конкретное изменение в базе.

Архитектура MTS



Многонитевая архитектура (MTS – Multi-Tread Server)

Репликация в СУБД Oracle

Rules and transformations

Пользователь имеет возможность определять соответствия между LCR и набором правил. Эти правила оценивают все изменения, произведенные в базе данных, и проводят фильтрацию несоответствующих LCR.

Например, следующее правило определяет только DML изменения таблицы SCOTT.EMP

:dml.get_object_owner()='SCOTT' and :dml.get_object_name()='EMP'

Точно также правила определяются и для DDL изменений.

Кроме того, к правилам могут быть привязаны трансформации. Трансформации используют пользовательские или системные хранимые процедуры и автоматически изменяют любой LCR , который удовлетворяет условиям используемого правила.

Queues

Очереди осуществляют хранение LCR, когда они двигаются в системе, т.е. находятся «между» процессами Oracle Streams.

Одна из первоочередных задач при настройке Oracle Streams – создать очереди и привязать их к процессам Oracle Streams. Для каждого процесса Oracle Streams может быть определен набор правил и связанных с этими правилами трансформаций для того, чтобы иметь возможность фильтровать информацию на «входе» и «выходе» процесса.

Очереди поддерживают три типа операций: enqueue – постановка LCR в очередь, browse – просмотр LCR и dequeue – удаление из очереди.

Репликация в СУБД Oracle

Capture, Propagation and Apply – три основных процесса Oracle Streams.
Основные задачи процесса ***capture***:

- ✓ считывание изменений, содержащихся в журналах транзакций;
- ✓ преобразование CR в LCR;
- ✓ постановка LCR в очередь.

Так же как и *capture*, процесс ***propagation*** выполняет 3 основных задачи:

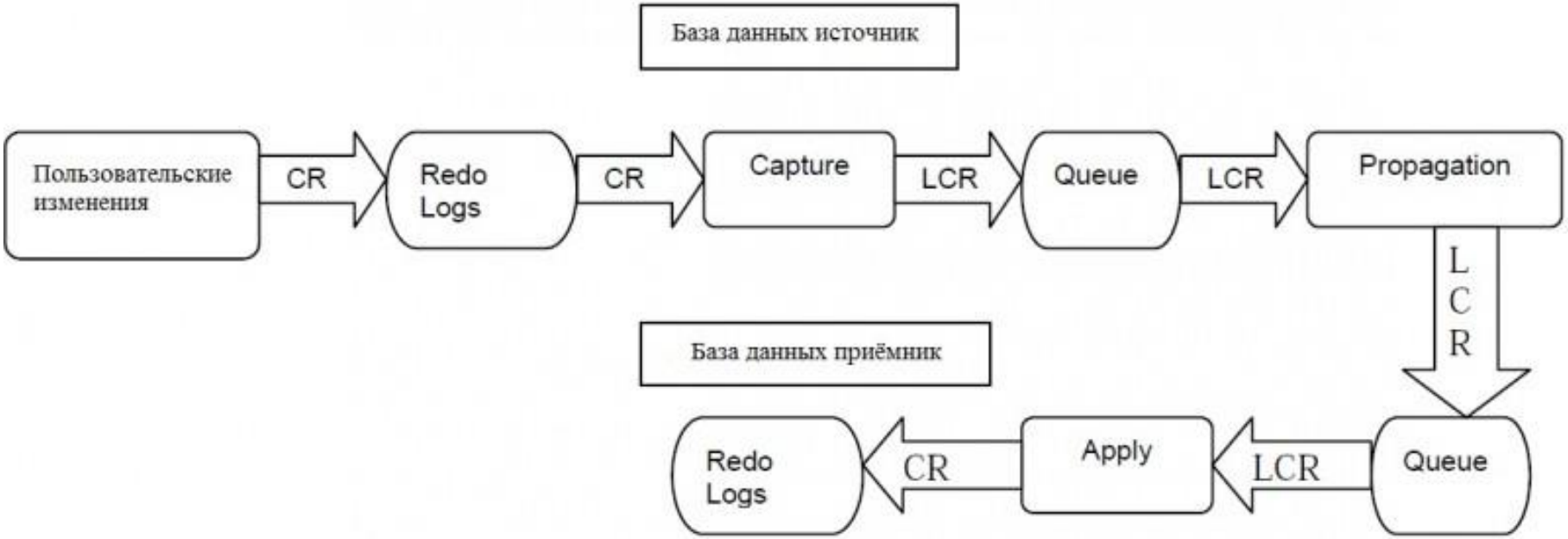
- ✓ просмотр LCR;
- ✓ передача LCR из одной очереди в другую, причем очереди могут находится как на одной базе данных, так и на разных;
- ✓ удаление LCR.

Apply процесс:

- ✓ извлекает принятые LCR из очереди;
- ✓ производит изменения с базой данных в соответствии с LCR;
- ✓ удаляет LCR из очереди.

В общем случае все изменения, сделанные с базой данных, записываются в журналы транзакций, но изменения, сделанные *apply* процессом, в журналы транзакций не пишутся, так как происходят в фоновом режиме незаметно для пользователя.

Репликация с помощью Oracle Streams



Репликация в СУБД SQL Server

Начиная с SQL Server 2000 эта СУБД поддерживает три типа репликации:

✓**Репликация моментальных снимков (snapshot replication)** – это периодическая репликация целостного набора данных, зафиксированного по состоянию на определенный момент времени, с локального сервера на удаленные. Применяется для репликации в БД, где количество реплицируемых данных невелико, а источник данных статичен. Можно предоставлять удаленным серверам ограниченную возможность обновления реплицированных данных.

✓**Репликация транзакций (transactional replication)** – это репликация начального моментального снимка данных на удаленные серверы, а также репликация отдельных транзакций, работающих на локальном сервере и выполняющих последовательные изменения данных в начальном моментальном снимке. Эти реплицированные транзакции выполняются над реплицируемыми данными на каждом удаленном сервере для синхронизации данных на удаленном сервере с данными локального сервера. Используется при необходимости постоянного обновления данных на удаленных серверах. Можно предоставлять удаленным серверам ограниченную возможность обновления реплицированных данных.

Репликация в СУБД SQL Server

✓**Репликация сведением (merge replication)** – это репликация начального моментального снимка данных на удаленные серверы, а также репликация изменений, происходящих на каком/либо удаленном сервере, обратно на локальный сервер с целью синхронизации, разрешения конфликтов и повторной репликации на удаленные серверы. Используется в случае, когда многочисленным изменениям подвергаются одни и те же данные, либо когда удаленные независимые компьютеры работают автономно, например, как в случае автономного пользователя.

Терминология репликации

Средства репликации SQL Server используют терминологию издательской деятельности для названий процессов и компонентов репликации. Сервер, реплицирующий сохраненную информацию на другие серверы, называется издателем (publisher). Реплицируемая информация состоит из одной или нескольких публикаций (publications). Каждая публикация представляет собой логически согласованный набор данных отдельной БД и состоит из одной или нескольких статей (articles). Статья может быть одним или несколькими объектами следующего типа:

- часть или целая таблица (с фильтрацией по столбцам и/или по строкам);
- хранимая процедура или определение представления;
- выполнение хранимой процедуры;
- представление;
- индексированное представление;
- пользовательская функция.

Репликация в СУБД SQL Server

Терминология (продолжение)

В процессе репликации каждый издатель взаимодействует с распространителем (distributor). Последний сохраняет публикуемые БД, историю событий и метаданные. Роль распространителя зависит от типа репликации. При этом распространитель может быть локальным (тот же экземпляр SQL Server) или удаленным (отдельный экземпляр SQL Server). Серверы, получающие реплицируемую информацию, называются подписчиками (subscribers). Они получают избранные публикации – подписки (subscriptions) – от одного или нескольких издателей. В зависимости от типа репликации, подписчикам может быть разрешено изменять реплицируемую информацию, а также реплицировать измененную информацию обратно издателю. Подписчики могут быть авторизованы, или могут быть анонимными (анонимная подписка используется при публикации данных в Интернете). Для больших публикаций использование анонимных серверов может повысить производительность системы.

Репликация в СУБД SQL Server

Агенты репликации (replication agents) автоматизируют процесс репликации. Как правило, агент репликации – это задание службы SQL Server Agent, сконфигурированное администратором для выполнения специфических задач по расписанию.

Существует некоторое число агентов репликации для различных задач репликации. Каждый агент сконфигурирован так для запуска по определенному расписанию. Различные типы репликации используют один или несколько таких агентов.

✓**Агент Snapshot** создает исходную мгновенную копию каждой реплицируемой публикации, включая информацию о схеме. Его используют все типы репликации. Можно иметь один такой агент на каждую публикацию.

✓**Агент Distribution** передает моментальный снимок данных и по следующие изменения от распространителя подписчикам. Этот агент используется при репликации моментальных снимков и репликации транзакций. По умолчанию для всех подписок на отдельную публикацию используется один агент Distribution. Такой агент называется разделяемым (shared). Однако можно настроить систему так, чтобы у каждого подписчика был личный, независимый (independent), агент Distribution.

Репликация в СУБД SQL Server

✓**Агент Log Reader** перемещает транзакции, помеченные для репликации, из журнала транзакций с сервера/издателя на сервер/распространитель. Этот агент используется при репликации транзакций. Каждая из помеченных для репликации БД будет иметь один агент Log Reader, запускающийся на распространителе и подключающийся к издателю.

✓**Агент Queue Reader** вносит в публикацию изменения, сделанные подписчиками в автономном режиме. Репликация мгновенных снимков и репликация транзакций используют этот агент в случае, если разрешена очередь обновлений. Агент запускается на распространителе, и существует только один экземпляр такого агента, обслуживающий всех издателей и публикации для конкретного распространителя.

✓**Агент Merge** передает моментальный снимок данных от распространителя подписчикам. Он также перемещает и контролирует изменения в реплицируемых данных между издателем и подписчиками. Этот агент дезактивирует подписки, информация которых не обновлялась в течение максимального срока хранения публикации (по умолчанию – 14 дней). Этот агент используется в случае репликации сведением. Каждая подписка на объединенную публикацию имеет свой объединяющий агент, который синхронизирует данные между сервером, публикующим данные, и серверами/подписчиками.

Репликация в СУБД SQL Server

✓**Агент History Clean Up** удаляет журнал событий агента из БД распространения, и используется для управления размером этой БД. Все типы репликации используют этот агент. По умолчанию он запускается каждые 10 минут.

✓**Агент Distribution Clean Up** удаляет реплицированные транзакции из БД распространения, и отключает неактивных подписчиков, данные которых не обновлялись в течение максимального периода хранения распространяемых данных (по умолчанию – 72 часа). Если разрешены анонимные подписки, реплицированные транзакции не удаляются до истечения максимального периода хранения. Репликация моментальных снимков и репликация транзакций используют этот агент. По умолчанию он запускается каждые 10 минут.

✓**Агент Expired Subscription Clean Up** выявляет и удаляет подписки с истекшим сроком хранения. Все типы репликации используют этот агент. По умолчанию он запускается один раз в день.

✓**Агент Reinitialize Subscriptions Having Data Validation Failures** повторно инициализирует все подписки, имеющие ошибки при проверке согласованности данных. По умолчанию этот агент запускается вручную.

✓**Агент Replication Agents Checkup** являет неактивных агентов репликации и заносит соответствующие записи в журнал приложений Windows. По умолчанию он запускается каждые 10 минут.

Репликация в СУБД SQL Server

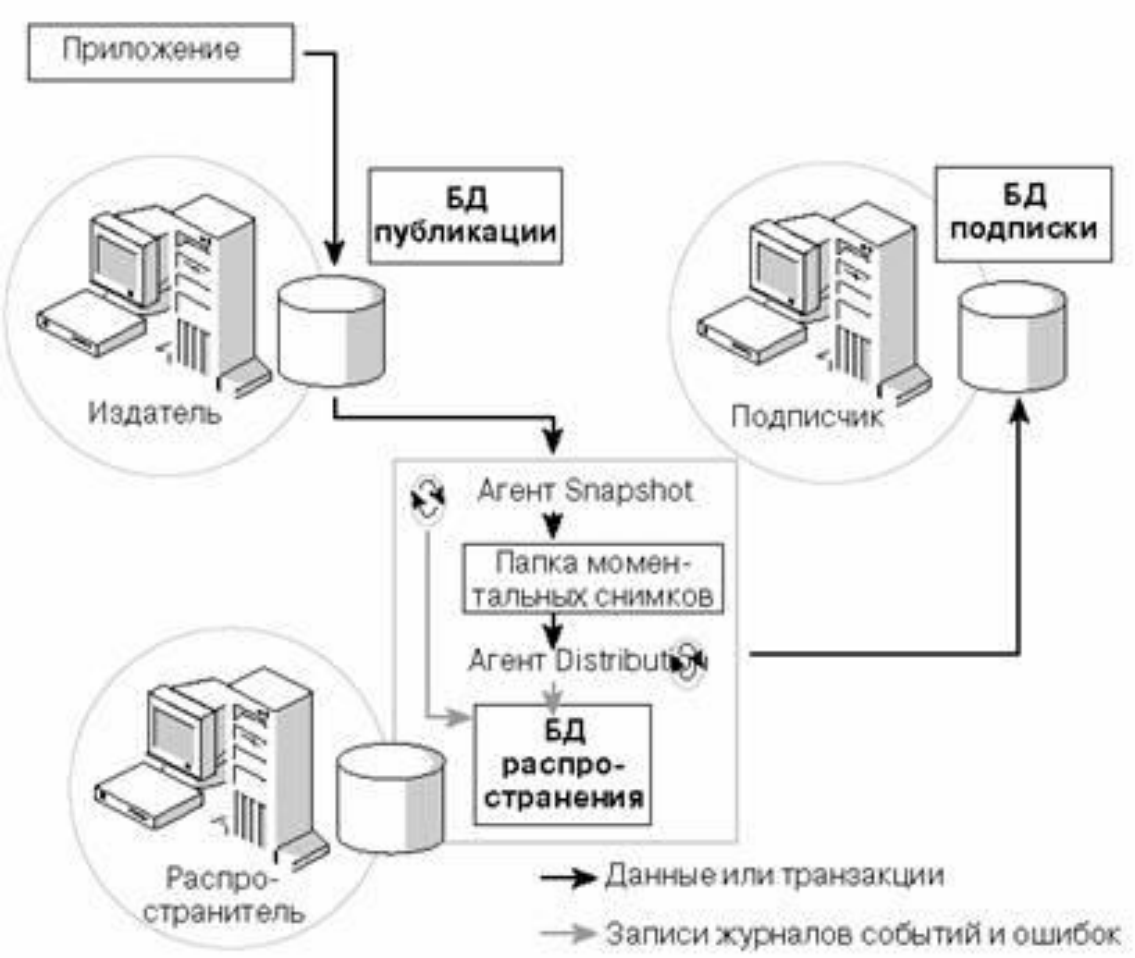
Репликация моментальных снимков

При репликации моментальных снимков агент Snapshot периодически (по заданному расписанию) копирует все помеченные для репликации данные с сервера/издателя в папку моментальных снимков распространителя. Агент Distribution периодически копирует все данные из папки моментальных снимков на каждый сервер/подписчик и, используя эти данные, полностью обновляет на нем публикацию. Агент Snapshot выполняется на распространителе, а агент Distribution может выполняться как на распространителе, так и на каждом сервере/подписчике. Оба агента записывают информацию журналов событий и журнала ошибок в БД распространения.

При репликации моментальных снимков подписчикам можно разрешить обновлять реплицированную информацию немедленно (Immediate Updating) и/или в порядке очереди (Queued Updating). Возможность обновления подписки (Updatable Subscription) полезна, когда подписчикам требуется изредка изменять последнюю. Если же подписку изменяют часто, лучше использовать репликацию сведением. Кроме того, в случае с обновляемыми подписками все обновления являются частью транзакции. Это означает, что обновление либо целиком подтверждается, либо откатывается, если происходит конфликт. При репликации сведением конфликты разрешаются построчно.

Репликация в СУБД SQL Server

Репликация моментальных снимков наиболее подходит для работы с не слишком интенсивно изменяемыми данными, для небольших публикаций, которые могут обновляться полностью без существенного увеличения нагрузки на сеть, а также для данных, которые не нужно постоянно поддерживать в актуальном состоянии (допустим, архивные данные об объемах продаж).



Репликация в СУБД SQL Server

Репликация моментальных снимков (продолжение)
Если используется немедленное обновление подписки, при любой попытке подписчика обновить реплицированные данные он сам или издатель инициируют транзакцию с двухэтапным подтверждением (two/phase commit, 2PC). 2PC/транзакция включает этап подготовки и этап подтверждения, и выполняется под управлением службы MS DTC, запущенной на подписчике и выступающей в качестве диспетчера транзакций. На подготовительном этапе MS DTC координирует действия служб SQL Server, запущенных на издателе и подписчике и играющих роль диспетчеров ресурсов, чтобы гарантировать успешное выполнение транзакции в обеих БД. На этапе подтверждения MS DTC получает от диспетчеров ресурсов уведомления об успешной подготовке, затем диспетчерам передается команда подтверждения и транзакция подтверждается на сервере/издателе и сервере/подписчике. Если на издателе имеется конфликт (конфликтующее обновление еще не было тиражировано на сервер/подписчик), транзакция, инициированная подписчиком, завершается неудачно. 2PC/транзакция гарантирует отсутствие конфликтов, поскольку издатель выявляет все конфликты до подтверждения транзакции.

Репликация в СУБД SQL Server

Репликация моментальных снимков (продолжение)

Если используется очередь обновлений (Queued Updating), сделанные подписчиком изменения помещаются в очередь и периодически передаются издателю. Изменения могут быть выполнены при отсутствии соединения с издателем. Изменения, которые находятся в очереди, пересылаются на данный сервер, когда устанавливается соединение. Очередь может храниться либо в БД SQL Server, либо можно выбрать использование Microsoft Message Queuing при работе в среде Windows 2000.

Так как обновления происходят не в реальном времени, то конфликты могут происходить, если другой подписчик или издатель изменили одни и те же данные. Конфликты разрешаются с использованием стратегии разрешения конфликтов, определяемой в момент создания публикации.

Если используются оба варианта обновления подписок, очередь обновлений выступает в качестве страховки на случай отказа немедленного обновления (например, из/за сбоев в работе сети). Это полезно, когда между издателем и подписчиком существует постоянное соединение, но при этом нобходимо убедиться, что подписчики могут совершать обновления в случае, если соединение разорвано.

Репликация в СУБД SQL Server

Репликация транзакций

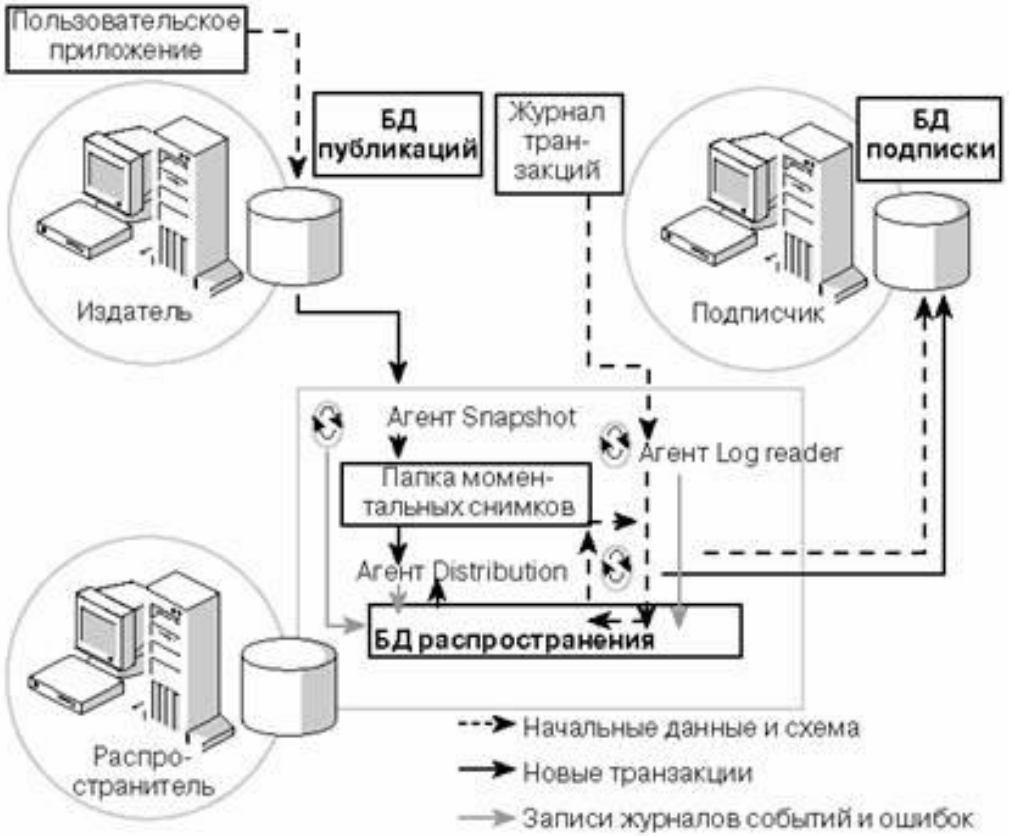
При репликации транзакций агент Snapshot создает исходный моментальный снимок данных, помеченных для репликации, и копирует его с сервера/издателя в папку моментальных снимков распространителя. Агент Distribution направляет полученный снимок каждому подписчику. Агент Log Reader следит за изменениями данных, участвующих в репликации, и фиксирует каждое изменение журнала транзакций в БД распространения на сервере/распространителе. Агент Distribution отправляет каждое изменение всем подписчикам в первоначальном порядке выполнения этих изменений. Если хранимая процедура используется для обновления большого количества записей, можно реплицировать эту процедуру, а не каждую обновленную строку. Все три этих агента репликации заносят информацию о событиях и ошибках в БД распространения.

Агент Distribution может работать постоянно, чтобы минимизировать задержку в обновлении данных между издателем и подписчиками, или может выполняться по заданному расписанию. Подписчики при наличии сетевого соединения с издателем могут получать изменения почти в реальном времени. После того как все подписчики получат реплицированные транзакции, агент Distribution Clean Up удаляет эти транзакции из БД распространения.

Репликация в СУБД SQL Server

Репликация транзакций

Если по окончании заданного периода хранения (по умолчанию – 72 часа) подписчик не получил реплицируемые транзакции, те удаляются из БД распространения и подписка дезактивируется. Это позволяет предотвратить чрезмерное увеличение размера БД распространения. Дезактивированная подписка может быть повторно активирована, и тогда подписчику с целью обновления его данных передается новый моментальный снимок.



Кроме того, репликацию транзакций, по аналогии со снимочной репликацией, можно настроить для поддержки обновляемых подписок.

Репликация в СУБД SQL Server

Репликация сведением

При репликации сведением агент Snapshot передает начальный моментальный снимок данных, участвующих в репликации, от издателя в папку моментальных копий распространителя. Агент Merge направляет полученный снимок каждому подписчику. Также он анализирует и объединяет изменения реплицируемых данных, выполняемые издателем и подписчиками. Если при объединении изменений происходит конфликт на издателе, агент Merge разрешает его, используя указанный администратором способ. Можно выбрать одно из существующих средств обнаружения конфликтов или создавать свое собственное.

Оба агента заносят информацию о событиях и ошибках в БД распространения (это единственная функция БД распространения в случае репликации сведением).

Чтобы различать записи отдельных копий реплицируемой таблицы и выявлять конфликты между записями, агент Merge использует специальный уникальный столбец реплицируемых таблиц. Если такого столбца нет, агент Snapshot добавляет его при создании публикации.

Репликация в СУБД SQL Server

Репликация сведением

При репликации сведением агент Snapshot передает начальный МС данных, участвующих в репликации, от издателя в папку моментальных копий распространителя. Агент Merge направляет полученный снимок каждому подписчику. Также он анализирует и объединяет изменения реплицируемых данных, выполняемые издателем и подписчиками. Если при объединении изменений происходит конфликт на издателе, агент Merge разрешает его, используя указанный администратором способ. Можно выбрать одно из существующих средств обнаружения конфликтов или создавать своё.

Оба агента заносят информацию о событиях и ошибках в БД распространения.

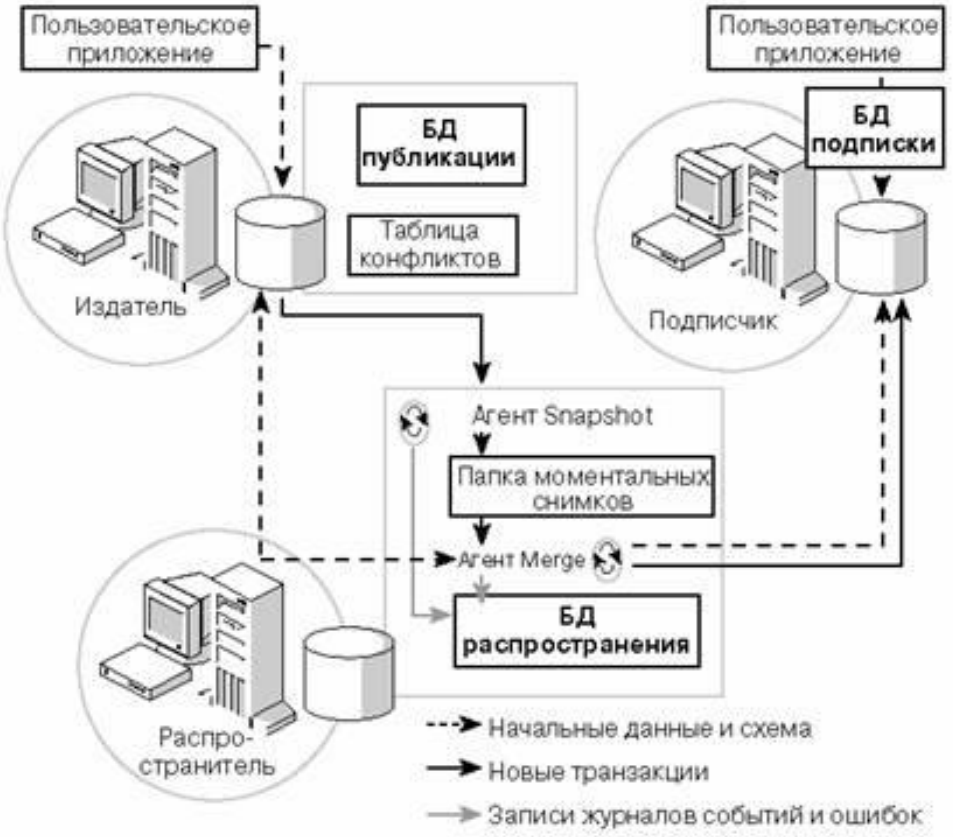
Чтобы различать записи отдельных копий реплицируемой таблицы и выявлять конфликты между записями, агент Merge использует специальный уникальный столбец реплицируемых таблиц. Если такого столбца нет, агент Snapshot добавляет его при создании публикации.

Кроме того, при создании публикации агент Snapshot создает на издателе триггеры. Они ведут мониторинг реплицированных записей и заносят информацию об изменениях в системные таблицы сведения. Агент Merge также создает идентичные триггеры на каждом сервере/подписчике, когда передает ему начальный моментальный снимок.

Репликация в СУБД SQL Server

Репликация сведением

Агент Snapshot может работать постоянно, чтобы минимизировать задержку в обновлении данных между издателем и подписчиками, или может выполняться по заданному расписанию. Подписчики при наличии сетевого соединения с издателем могут получать изменения почти в реальном времени. Если по окончании заданного периода хранения (по умолчанию – 14 дней) подписчик не получил реплицируемые транзакции, подписка дезактивируется. Дезактивированная подписка может быть повторно активирована, и тогда подписчику с целью обновления его данных передается новый моментальный снимок.



Репликация в СУБД SQL Server

Выбор модели репликации

Если применяется репликация моментальных снимков или репликацию транзакций, то приходится использовать удаленного распространителя. Он может предоставлять службы репликации одновременно нескольким издателям и подписчикам. Если объем реплицируемых данных невелик, распространителя и издателей нередко размещают на одном и том же компьютере. Вместо репликации данных нескольким подписчикам через подключение, имеющее низкую пропускную способность или высокую стоимость использования, можно опубликовать данные на удаленном подписчике, который распространит их другим подписчикам в своей области. Такой подписчик называется переиздающим (publishing subscriber, republisher).

В случае репликации сведением центральный подписчик часто используется для объединения информации, поступающей от нескольких региональных издателей. Для этой модели необходимо горизонтальное разбиение данных, чтобы избежать возможных конфликтов, и обычно используется специальный столбец для идентификации данных, поступивших из отдельных регионов. Такая модель также может использоваться при репликации моментальных снимков и при репликации транзакций. Кроме того, так как репликация сведением накладывает ограничения на использование БД распространения, издатель и распространитель часто размещаются на одном и том же компьютере.

Репликация в СУБД Sybase

Sybase Replication Server основан на распределенной архитектуре и в нем реализован набор возможностей, гарантирующих доставку изменений в данных. Этот пакет рассчитан на работу не только с СУБД Sybase Adaptive Server® Enterprise (ASE), но и с другими типами источников данных, в том числе Oracle, IBM DB2, Microsoft SQL Server. В состав СУБД Sybase входит Replication Server Manager – инструмент для управления сложными конфигурациями репликации.

Свойства **Sybase Replication Server**:

• Выделенная функциональность репликации.

Сопутствующий репликации большой объем операций ввода-вывода может неприемлемо замедлить работу сервера баз данных. Sybase Replication Server поставляется в форме специализированного дополнения, а не внешнего приложения. Эта архитектура минимизирует взаимодействие с исходной БД

• Непрерывный захват транзакционных данных в реальном времени по журналу.

Этот метод, при котором чтение выполняется непосредственно из онлайнового журнала для воспроизведения операций, быстрее, нежели триггерная репликация.

• Репликация команд SQL.

Для уменьшения объема данных, пересылаемх по сети, приемнику отправляются только команды SQL.

• Настраиваемая сетевая маршрутизация.

Sybase Replication Server позволяет администраторам задавать маршруты следования реплицируемой информации к удаленным приемникам.

• Повышенная скорость выполнения транзакций в БД назначения.

Replication Server работает с разными СУБД и содержит специализированные механизмы оптимизации пропускной способности и индивидуальные средства взаимодействия с каждой поддерживаемой платформой.

Репликация в СУБД Sybase

Асинхронное тиражирование транзакций

Механизм асинхронного тиражирования транзакций (репликации) гарантирует доставку измененных данных на вторичные серверы непосредственно после завершения транзакции, если сервер доступен, или сразу после подключения сервера к сети. Такой подход предполагает хранение дублирующей информации в различных узлах сети и обеспечивает, по сравнению с другими подходами к репликации, снижение трафика, уменьшение времени ответа системы, а также позволяет оптимизировать нагрузку на серверы.

Асинхронная репликация перекладывает передачу данных, обеспечение их целостности и ожидание при передаче данных с прикладной программы и пользователя на системный уровень.

Асинхронная репликация, в отличие от 2PC, не обеспечивает полной синхронности информации на всех серверах в любой момент времени. Синхронизация происходит через некоторый, обычно небольшой, интервал времени, величина которого определяется быстродействием соответствующего канала связи. Для большинства задач кратковременное наличие устаревших данных в удаленных узлах вполне допустимо.

Вместе с тем асинхронная репликация транзакций принципиально обеспечивает целостность данных, так как объектом обмена данными здесь является логическая единица работы – транзакция, а не просто данные из измененных таблиц.

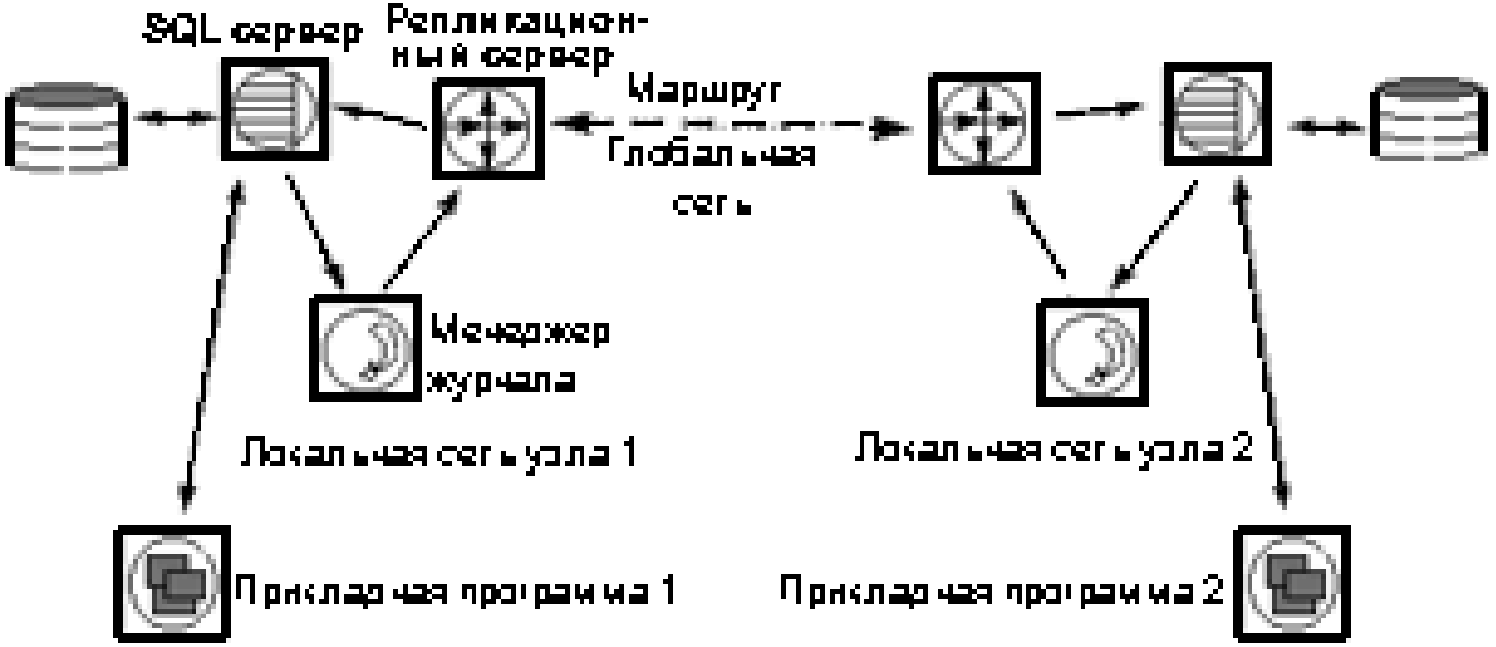
Репликация в СУБД Sybase

Репликационный сервер использует асинхронную модель репликации транзакций. При разработке модели данных проектируются и правила репликации. Затем проводится конфигурирование системы. При работе прикладной программы изменения данных отслеживаются системными средствами и в соответствии с конфигурацией требуемые данные передаются в удаленную СУБД. Репликационный сервер представляет собой отдельную задачу, запускаемую одновременно с СУБД. Он имеет свой входной язык и стандартный для продуктов Sybase сетевой интерфейс Open Server. Такое разделение снижает нагрузку на СУБД и делает систему в целом более открытой.

Репликация использует интуитивно понятный принцип "публикации" изменяемых данных и "подписки" на изменения. Транзакция может вносить изменения (т.е. добавлять, удалять и изменять записи) в одну или несколько таблиц БД. Выбранные для репликации таблицы специальным образом помечаются. Для каждой такой таблицы или группы ее строк, выбранной по заданному условию, определяется один узел (СУБД), в котором данные таблицы являются первичными. Это тот узел, в котором происходит наиболее активное обновление данных.

Репликация в СУБД Sybase

Репликационному серверу, обслуживающему БД с первичными данными, задается описание тиражирования (replication definition). В этом описании, в частности, могут быть заданы интервалы значений первичного ключа таблицы (или другое условие на первичный ключ), при выполнении которого измененные данные будут тиражироваться из этого узла к подписчикам. Если условие не задано, то описание тиражирования действует для всех записей таблицы. Возможность тиражирования группы записей таблицы означает, в частности, что часть записей таблицы может быть первичными данными в одном узле, а часть – в других.



Репликация в СУБД Sybase

В одном или нескольких узлах (СУБД), которым нужны измененные данные, в обслуживающем его репликационном сервере создается подписка (subscription) на соответствующее описание тиражирования. Здесь будет поддерживаться (с небольшой задержкой) копия первичных данных.

Репликация данных в Sybase использует журнал транзакций как источник информации о завершенных транзакциях.

В узле, содержащем первичные данные, для каждой тиражируемой базы данных запускается специальная компонента – репликационный агент (Replication Agent – RA). Он подключается к серверу БД и получает от него уведомления о завершении транзакций. Измененные данные передаются репликационному серверу, обслуживающему этот узел. Репликационный сервер в соответствии с описанием тиражирования и подписками отправляет данные в специальном эффективном протоколе по месту назначения – соответствующим репликационным серверам в удаленных узлах.

Именно в этом месте – между репликационными серверами – связь может быть медленной или недостаточно надежной. Передаваемые данные в составе транзакции при недоступности узла-получателя записываются в стабильные очереди на диске и затем передаются по мере возможности. Данные могут передаваться в удаленный узел по маршруту, содержащему несколько репликационных серверов. Данная возможность лежит в основе построения иерархических систем репликации.

Репликация в СУБД Sybase

По умолчанию репликационный сервер сохраняет смысл операций. Это значит, что удаление записи из первичной таблицы (выполнение оператора DELETE) приведет к выполнению такого же оператора DELETE в узле, хранящем копию таблицы; выполнение INSERT или UPDATE над первичной таблицей точно так же приведет соответственно к добавлению или обновлению записи в копии таблицы в результате работы системы репликации. Имеется гибкий механизм конфигурирования так называемых функциональных строк (function strings), которые переопределяют любую операцию на макроязыке с возможностью подстановки параметров.

В одной базе данных могут содержаться как первичные данные, так и данные-копии. Приложение-клиент, работающее со своей СУБД, может вносить изменения напрямую (операторами INSERT, DELETE, UPDATE) только в первичные данные. Для изменения копии данных предназначен механизм асинхронного вызова процедур.

Для работы механизма асинхронного вызова процедур в нескольких базах данных создаются процедуры с одинаковым именем и параметрами, но, возможно, с различным текстом. В одной базе данных процедура помечается как предназначенная к репликации. Вызов этой процедуры вместе со значениями параметров передается через журнал и механизм репликации к узлам-подписчикам. Затем в базах данных подписчиков вызывается одноименная процедура с теми же значениями параметров.

Репликация в СУБД Sybase

Таким образом, для обновления "чужих" для узла данных (копии данных) прикладная программа-клиент вместо выполнения оператора UPDATE вызывает заранее определенную в этом узле хранимую процедуру и передает ей параметры (например, значение первичного ключа и новые значения для обновляемых колонок). Тело этой процедуры пустое и она не выполняет никаких действий, однако ее вызов записывается в журнал. Механизм репликации обеспечивает вызов на узле, содержащем первичные данные, одноименной процедуры с подстановкой параметров. В теле этой процедуры может быть записан оператор UPDATE, обновляющий первичные данные. Тот же механизм репликации передаст изменения в данных узлу, инициировавшему операцию.

Репликационный сервер и Replication Agent реализованы в виде отдельных модулей и могут выполняться не на том же компьютере, что сервер базы данных. Включение в систему репликационного сервера практически не оказывает влияние на загрузку сервера первичной базы данных.

Репликация в СУБД Sybase

СУБД, хранящая вторичные данные, может быть любой СУБД, доступной через шлюз, в том числе Oracle, Informix, Ingres, DB2, RMS, ISAM, или даже приложение Open Server.

СУБД, хранящая первичные данные, требует наличия для нее RA. Сейчас RA имеется для Sybase SQL Server, Oracle, DB2, Sybase SQL Anywhere. Готовятся RA и для других СУБД. Интерфейс RA открыт и возможно создание RA для нестандартных источников данных.

Некоторые применения тиражирования данных:

- ✓сервер, выполняющий активное обновление данных (OLTP), разгружается от сложных запросов, связанных с поддержкой принятия решений (DSS);
- ✓консолидация данных от подразделений в центре;
- ✓обмен данными по медленным и/или ненадежным линиях связи;
- ✓поддержание резервной базы данных;
- ✓построение сети равноправных узлов, обменивающихся данными.

Важно, что репликационный сервер тиражирует транзакции, а не отдельные изменения в базе данных. Метод тиражирования транзакций гарантирует целостность внутри транзакции, и, как следствие, невозможность нарушения ссылочной целостности. Схема обновления первичных данных и копий данных исключает возможность возникновения конфликтов (конфликты могут быть вызваны только неправильным проектированием системы или сбоем).