### **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. Также возможен вариант *с консолидацией данных,* при котором каждый узел владеет своей частью данных (например, отношения) и может ее обновлять, а на одном из узлов РБД эти части соединяются (или объединяются) вместе для образования консолидированной копии "только для чтения" (read only).

К основным достоинствам механизма репликации можно отнести повышение доступности и надежности данных и повышение локализации ссылок на реплицируемые данные. Основным недостатком репликации является сложность поддержания идентичности реплик: если в одну копию данных вносятся изменения, то эти изменения также должны быть внесены в другие копии. Это называется **распространение изменений** и реализуется **службой тиражирования**.

Служба тиражирования должна выполнять следующие функции:

1. Обеспечение масштабируемости, т.е. эффективная обработки больших и малых объемов данных.
2. Преобразование типов и моделей данных (для гетерогенных РБД).
3. Репликация объектов БД, например, индексов, триггеров и т.п.
4. Инициализация вновь создаваемой реплики.
5. 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).

Существуют различные подходы к организации репликации:

* **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. Например, при последовательной обработке заказов на поставку товаров сначала в отделе приема заказ принимается, затем на складе определяют, может ли он быть выполнен и сколько это будет стоить, затем эти данные передаются в финансовый отдел, а после оплаты – в отдел доставки. При репликации данных из узла Si на узел Si+1 вместе с новыми данными передается право на обновление реплики.

* **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.

Достоинствами синхронного распространения изменений являются полная согласованность копий и отсутствие конфликтов обновления. К недостаткам следует отнести трудоемкость и большую длительность модификации данных и низкую надежность работы системы: при выходе из строя одного узла все копии становятся недоступны для внесения изменений.

При использовании асинхронного режима распространения изменений могут возникать конфликты обновления. Можно выделить следующие конфликтные ситуации:

1. Добавление двух записей с одинаковыми первичными или уникальными ключами. Для предотвращения таких ситуаций обычно каждому узлу РБД выделяется свой диапазон значений ключевых (уникальных) полей.
2. Конфликты удаления: одна транзакция пытается удалить запись, которая в другой копии уже удалена другой транзакцией. Если такая ситуация считается конфликтом, то она разрешаются вручную.
3. 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.

Для разрешения конфликтов обновления применяются различные методы, например:

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.

Для настройки реакции на конфликты некоторые СУБД позволяют выделить в каждой таблице столбец (группу столбцов), и описать для них способ разрешения конфликтов (т.н. группа обновления).

Существуют различные варианты реализации распространения изменений. Один из них заключается в использовании триггеров. Напомним, что триггеры – это процедуры, которые срабатывают при наступлении определенных событий (например, insert, delete, update). 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).

Другой способ реализации механизма распространения изменений – 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:

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