Тема 27. Параллельная обработка данных

Основные понятия параллельной обработки

Параллельная обработка – одновременная работа с одними и теми же данными нескольких приложений или пользователей. Параллельная работа повышает эффективность обработки данных, но одновременно создает и различные проблемы. Введем ряд определений.

Транзакция (процесс) – отдельное выполнение группы операторов, переводящее базу данных из одного целостного состояния в другое.

База данных считается состоящей из отдельных элементов (**ресурсов**), которые можно брать во владение (**блокировать**) и освобождать (**разблокировать**). При этом элементом может быть и отдельный кортеж, и целое отношение.

Пример.

Программа: P: READ A; A:=A+1; WRITE A;

Пусть имеются две транзакции T_1 и T_2, представляющие два прогона программы P. Обе транзакции имеют доступ к элементу A и изменяют его значение на 1. При последовательном выполнении транзакций элемент A получит значение A_0+2 (A_0 - начальное значение A). Если транзакции выполняются параллельно, то A может принять как значение A_0+2, так и значение A_0+1. Это нежелательно.

Для того чтобы устранить подобные явления используются блокировки. Перед чтением транзакция должна блокировать элемент (установить монопольное владение элементом), а после записи значения снять блокировку (сделать элемент доступным для других транзакций). Будем считать, что блокирование и разблокирование элемента производятся операторами LOCK и UNLOCK.

Если транзакция пытается блокировать элемент, уже заблокированный другой транзакцией, ей приходится ждать снятия блокирования. Каждая транзакция в конце работы должна разблокировать все заблокированные ею элементы.

Программа с блокированиями имеет вид

P: LOCK A; READ A; A:=A+1; WRITE A; UNLOCK A;

Система параллельных транзакций может иметь такие нежелательные состояния как бесконечное ожидание и тупик.

Бесконечное ожидание имеет место, когда некоторая транзакция бесконечно долго не может получить доступа к элементу из-за того, что другие транзакции постоянно опережают ее (отталкивают от элемента). Бесконечное ожидание не является серьезной проблемой в БД. Оно легко устраняется введением подходящего порядка удовлетворения запросов на блокировки.

Тупик. Состояние системы параллельных транзакций, когда каждая из них ожидает, когда ей будет предоставлена возможность заблокировать элемент, уже заблокированный другой транзакцией, называется тупиком. Так как все транзакции находятся в состоянии ожидания, ни одна из них не может разблокировать элемент, необходимый для продолжения другой транзакции. Ожидание становится бесконечным. Тупик – серьезная проблема для параллельных транзакций.

Сериализуемость. Как показано выше, последовательное выполнение транзакций по одной и их параллельное выполнение могут давать разные результаты. Принято считать, что правильный результат обеспечивает именно последовательное выполнение транзакций. Т.е. параллельное выполнение нескольких транзакций корректно, если и только если их совместный результат будет таким же, как и при некотором последовательном выполнении.

Будем называть **расписанием** совокупности транзакций порядок, в котором выполняются элементарные шаги этих транзакций. Расписание считается последовательным, если все шаги каждой транзакции выполняются последовательно, и **сериализуемым**, если его результат эквивалентен результату некоторого последовательного расписания. Сериализуемость – важное свойство расписания. Корректное расписание совокупности параллельных транзакций обязательно должно быть сериализуемым.

Протоколы. Протокол – ограничения на последовательности шагов, которые могут выполнять транзакции. Протоколы используются для обеспечения сериализуемости расписаний и предотвращения тупиков.

Свойства транзакций

Транзакции характеризуются четырьмя свойствами: **атомарность, согласованность, изолированность, долговечность.**

Свойство атомарности выражается в том, что транзакция должна быть выполнена в целом или не выполнена вовсе.

Свойство согласованности гарантирует сохранение целостного состояния БД после выполнения каждой транзакции.

Свойство изолированности означает, что конкурирующие за доступ к БД транзакции физически обрабатываются последовательно, изолированно друг от друга.

Долговечность – обязательное сохранение всех изменений в данных, произведенных успешной транзакцией.

Транзакция может завершиться удачно или неудачно. В первом случае транзакция фиксируется – записывается на диск. Во втором – БД возвращается в исходное состояние (происходит откат транзакции).

Стандарт ANSI/ ISO SQL содержит два оператора (COMMIT и ROLLBACK), определяющие успешность транзакции.

Транзакция может завершиться:

- Оператором COMMIT – успешное завершение, транзакция фиксируется;
- Оператором ROLLBACK – неудачное завершение, происходит откат транзакции;
- Успешным завершением программы (неявный COMMIT);
- Ошибочным завершением программы (неявный ROLLBACK).

Восстановление БД

Для обеспечения надежного хранения данных в БД используются журналы транзакций, позволяющие восстанавливать согласованные состояния БД после любых аппаратных и программных сбоев.

Принципы восстановления БД:

- Зафиксированные транзакции должны быть сохранены в восстановленном состоянии БД;
- Незафиксированные транзакции должны отсутствовать в восстановленном состоянии БД.

Восстановление БД производится в следующих случаях:

- Индивидуальный откат транзакции (ROLLBACK, аварийное завершение программы, принудительный откат при взаимной блокировке);
- Мягкий сбой (потеря содержимого оперативной памяти);
- Жесткий сбой (поломка диска).

При индивидуальном откате устраняются все изменения в данных, произведенные транзакцией. Мягкий сбой устраняется с помощью хранящегося на диске журнала транзакций. Восстановление при жестком сбое требует использования архивных копий БД и неповрежденных журналов транзакций.

Журнал транзакций можно представлять в виде последовательного файла, в котором фиксируются все изменения БД.

Параллельное выполнение транзакций

СУБД должна обеспечивать корректную параллельную работу многих пользователей над одними и теми же данными таким образом, что БД, по мере выполнения транзакций, переходит из одного целостного состояния в другое.

Неограниченный параллелизм может приводить к следующим проблемам:

- **Пропавшие изменения**. Ситуация возникает при одновременном изменении записи несколькими транзакциями. Изменения, сделанные одной транзакцией, могут затираться другими.

- **Промежуточные данные**. Временные изменения, сделанные одной транзакцией, ошибочно становятся доступны другим транзакциям.
- **Несогласованные данные**. Повторное чтение транзакцией одной и той же записи может давать разные результаты.
- **Строки-фантомы**. Два одинаковых запроса к БД в одной транзакции могут давать разные результаты.

Для устранения подобных проблем выполнение параллельных транзакций должно удовлетворять следующим правилам:

- Пользователи не должны видеть несогласованные и промежуточные данные.
- Расписание параллельных транзакций должно быть сериализуемым.

Для реализации этих правил используются блокировки. Различают **разделяемые** (Shared) блокировки и **эксклюзивные** (монопольные, жесткие) блокировки (eXclusive).

Блокировки могут приводить к тупикам. Для устранения тупика выбирается и производится откат одной из транзакций (транзакции жертвы), в результате чего освобождаются требуемые ресурсы и тупик разрушается.

В языке SQL оператор блокирования имеет вид:

LOCK TABLE имя_таблицы IN {SHARED | EXCLUSIVE} MODE.

Сериализация расписаний достигается использованием двухфазного протокола блокирования ресурсов: первая фаза транзакции – накопление блокировок, вторая – разблокирование всех захваченных ресурсов.

Уровни изолированности пользователей

В тех случаях, когда приложению важны не точные данные, а скорость выполнения запросов, требование сериализации можно смягчить. Для этого вводится понятие уровней изолированности пользователя.

Имеются следующие уровни изолированности:

- SERIALIZABLE. Обеспечивает полную изоляцию транзакций.

- REPEATABLE READ (повторяющееся чтение). Транзакция не имеет доступа к промежуточным или окончательным результатам других транзакций. Возможны строки-призраки.
- READ COMMITED (подтвержденное чтение). Транзакция не имеет доступа к промежуточным результатам других транзакций. Возможны несогласованные данные и строки призраки.
- READ UNCOMMITED (грязное чтение). Транзакция видит промежуточные и несогласованные данные и строки-призраки. Предотвращаются пропавшие обновления.

В стандарте SQL 2 уровень изолированности задается оператором SET TRANSACTION:

SET TRANSACTION … ISOLATION LEVEL [{SERIALAZABLE | REPEATABLE READ | READ COMMITED | READ UNCOMMITED}] [{READ WRITE | READ ONLY}]

Гранулированные синхронизационные захваты

Объекты блокирования могут быть разного уровня: БД, файл, отношение, страница, кортеж. Чем крупнее объект, тем меньше блокировок потребуется программе, тем меньше накладные расходы. Одновременно возрастает вероятность конфликта транзакций, уменьшается степень их параллельности.

Для достижения разумного компромисса между требованиями минимизации расходов, уменьшения конфликтности и увеличения параллельности был предложен механизм гранулированных синхронизационных захватов. В соответствии с ним объект может быть захвачен в режимах S, X, IS, IX, SIX. Используется специальный протокол: перед захватом объекта в режиме S или X соответствующий объект более высокого уровня должен быть захвачен в режиме IS, IX или SIX.

Режим IS (Intended for Shared Lock) по отношению к составному объекту O означает намерение захватить в будущем некоторый входящий в O объект в S-режиме.

Режим IX по отношению к составному объекту O означает намерение захватить в будущем некоторый входящий в O объект в X-режиме.

Режим SIX – совместный захват объекта O с намерением впоследствии захватить входящие в него объекты в монопольном режиме.

Предикатные синхронизационные захваты

Метод гранулированных синхронизационных захватов не решает проблему фантомов. Эта проблема решается методом предикатных синхронизационных захватов.

Суть метода. Рассматривается захват двумя транзакциями одного отношения. Каждая транзакция содержит условие, определяющее множество обрабатываемых кортежей отношения. Если множества кортежей не пересекаются, то транзакции можно выполнять параллельно, в противном случае транзакции выполняются последовательно.