Лекция 9.

Транзакции и блокировки в базах данных

Транзакции в СУБД

Транзакция – неделимая последовательность операторов манипулирования данными (чтения, удаления, вставки, модификации), приводящая к одному из двух возможных результатов:

- либо последовательность выполняется, если все операторы правильные,
- либо вся транзакция откатывается, если хотя бы один оператор не может быть успешно выполнен.

Транзакции в СУБД

Корректная поддержка транзакций:

1. Основа целостности базы данных.
2. Основа изолированности, параллельной работы нескольких пользователей. Параллелизм транзакций.

Команды SQL для управления транзакциями:

- START TRANSACTION (BEGIN TRANSACTION)
- SET TRANSACTION
- COMMIT
- ROLLBACK (вызывается явно или автоматически при возникновении ошибки)

ACID-свойства транзакций

ACID описывает требования к транзакционной системе, обеспечивающие наиболее надёжную и предсказуемую её работу.

ACID-свойства транзакций

- **A**tomicity. Транзакция **неделима**.
- **C**onsistency. Транзакция не нарушает логическую целостность данных, переводит БД из одного **согласованного** состояния в другое.
- **I**solation. Транзакция **изолирована**, ее результаты самодостаточны. Незаконченная (неподтвержденная) транзакция должна быть невидима извне.
- **D**urability. Транзакция **устойчива (долговечна)**, ее действие постоянно даже при сбое системы. После завершения транзакции изменения должны стать доступны всем и не быть потеряны.

Распределенные (NoSQL) системы могут не поддерживать одновременно все ACID-свойства.

Теорема CAP

Распределенная система не может гарантировать одновременного выполнения следующих трёх свойств:

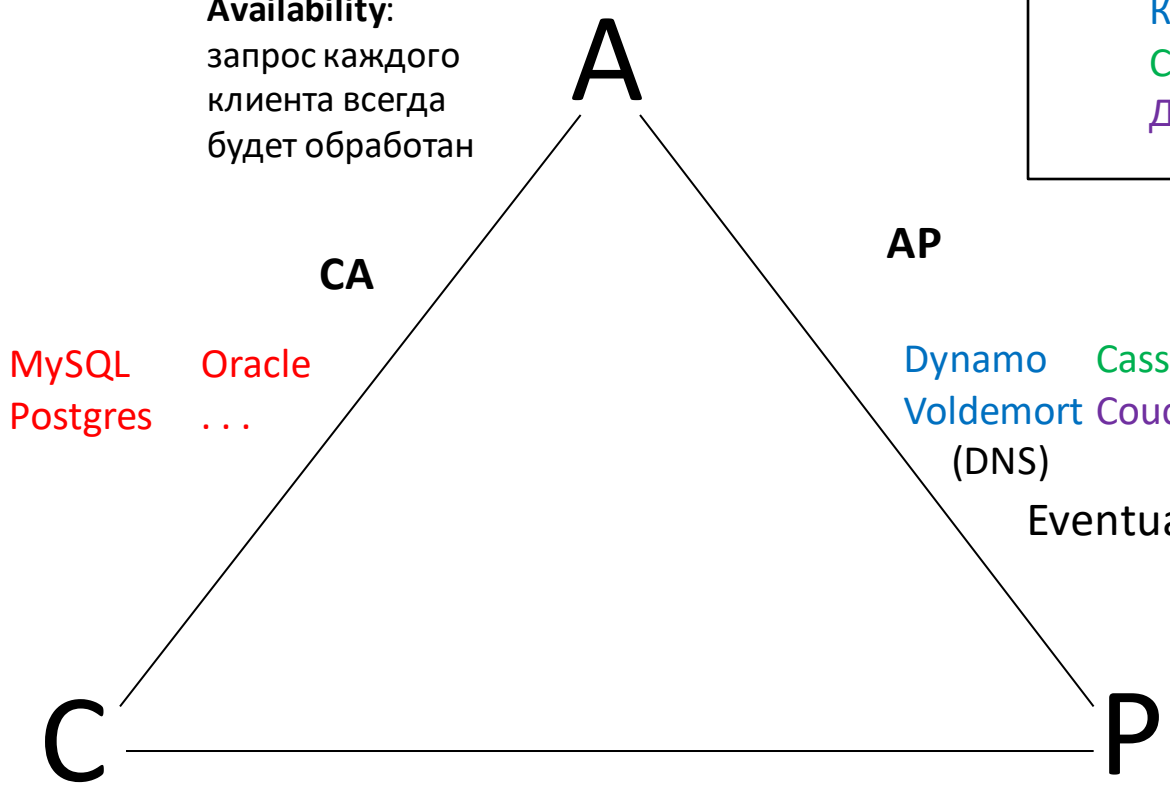
- **Consistency**. **Согласованность** - все пользователи в любой момент времени видят одинаковые данные.
- **Availability**. **Доступность** – при выходе из строя каких-либо узлов оставшиеся узлы продолжают функционировать.
- **Partition Tolerance**. **Устойчивость к разделению** – при распадении системы на отдельные группы узлов из-за сбоя сети каждая группа продолжает работать.

Теорема CAP

«Согласованность в конечном счете» (Eventual Consistency)

Теорема CAP

Модели данных:
- Реляционная
- Ключ-значение
- Столбцовая
- Документная



Availability:
запрос каждого клиента всегда будет обработан

CA

MySQL Oracle
Postgres . . .

AP

Dynamo Cassandra Riak
Voldemort CouchDB SimpleDB
(DNS)

Eventual consistency

Consistency:
все всегда видят одни и те же данные

CP

BigTable MongoDB Redis
Hbase TerraStore MemcacheDB

Partition tolerance:
система продолжает работать после физического разделения

ACID-свойства транзакций

- **Atomicity**. Транзакция **неделима**.
- **Isolation**. Транзакция **изолирована**, ее результаты самодостаточны. Незаконченная (неподтвержденная) транзакция должна быть невидима извне.
- **Durability**. Транзакция **устойчива (долговечна)**, ее действие постоянно даже при сбое системы. После завершения транзакции изменения должны стать доступны всем и не быть потеряны.

Журнал транзакций

Журнализация всех изменений в базе данных для атомарности и долговечности.

Упреждающая журнализация, Write-Ahead Log (WAL), Redo Log

В простейшем случае:

- идентификатор транзакции;
- объект, подвергшийся изменению;
- предыдущее состояние объекта и его новое состояние.

Журнал транзакций

Общий алгоритм:

- измения пишутся в журнал транзакций (состояние до и после) и изменяются страницы БД в памяти;
- в журнал транзакций сбрасывается маркер успешного завершения транзакции;
- журнал транзакций сбрасывается на диск;
- данные страниц БД пишутся на диск;
- данные страниц БД сбрасываются на диск.

Журнал транзакций

- Бэкапы
- Репликация

Блокировки

При выполнении транзакций СУБД накладывает на данные блокировки.

Блокировка - временное ограничение на выполнение некоторых операций обработки данных. Два вида:

- **блокировка записи** – транзакция блокирует строки в таблицах таким образом, что запрос другой транзакции к этим строкам будет отменен;
- **блокировка чтения** – транзакция блокирует строки так, что запрос со стороны другой транзакции на блокировку записи этих строк будет отвергнут, а на блокировку чтения – принят.

Проблема параллелизма

- Транзакция считывает строку таблицы → накладывает на нее блокировку чтения.
- Транзакция модифицирует строку данных → накладывает на нее блокировку записи.
- Запрашиваемая блокировка на строку отвергается из-за уже имеющейся блокировки → транзакция переводится в режим ожидания до снятия блокировки.
- Блокировка записи сохраняется до конца выполнения транзакции.

Deadlock

Ситуация в СУБД, при которой несколько транзакций ожидают освобождения ресурсов, занятых друг другом и ни одна из них не может продолжить свое выполнение.

Проблемы одновременного доступа к данным

Потерянное обновление – при одновременном изменении одного блока данных разными транзакциями одно из изменений теряется.

Транзакция 1	**Транзакция 2**
UPDATE tbl1 SET f2=f2+20 WHERE f1=1;	UPDATE tbl1 SET f2=f2+25 WHERE f1=1;

1. Обе транзакции одновременно читают текущее состояние поля.
2. Обе транзакции вычисляют новое значение поля.
3. Транзакции пытаются записать результат вычислений обратно в поле f2. Физически одновременно две записи выполнить невозможно, одна из операций записи будет выполнена раньше, другая позже. При этом вторая операция записи перезапишет результат первой. Первая транзакция «пропадет».

Проблемы одновременного доступа к данным

«Грязное» чтение – чтение данных, добавленных или изменённых транзакцией, которая впоследствии не подтвердится (откатится).

Транзакция 1

```
UPDATE tbl1 SET f2=f2+1 WHERE f1=1;

ROLLBACK;
```

Транзакция 2

```
SELECT f2 FROM tbl1 WHERE f1=1;
```

В транзакции 1 изменяется значение поля f2, а затем в транзакции 2 выбирается значение этого поля. После этого происходит откат транзакции 1. В результате значение, полученное второй транзакцией, будет отличаться от значения, хранимого в базе данных.

Проблемы одновременного доступа к данным

Неповторяющееся чтение – при повторном чтении в рамках одной транзакции ранее прочитанные данные оказываются изменёнными.

Транзакция 1	**Транзакция 2**
	SELECT f2 FROM tbl1 WHERE f1=1;
UPDATE tbl1 SET f2=f2+1 WHERE f1=1;	
COMMIT;	
	SELECT f2 FROM tbl1 WHERE f1=1;

В транзакции 2 выбирается значение поля f2, затем в транзакции 1 изменяется значение поля f2. При повторной попытке выбора значения из поля f2 в транзакции 2 будет получен другой результат.

Эта ситуация особенно неприемлема, когда данные считываются с целью их частичного изменения и обратной записи в базу данных.

Проблемы одновременного доступа к данным

Фантомное чтение – при повторном чтении в рамках одной транзакции одна и та же выборка дает разные множества строк.

Транзакция 1

```
INSERT INTO tbl1 (f1,f2) VALUES (15,20);
COMMIT;
```

Транзакция 2

```
SELECT SUM(f2) FROM tbl1;

SELECT SUM(f2) FROM tbl1;
```

В транзакции 2 выполняется SQL-оператор, использующий все значения поля f2. Затем в транзакции 1 выполняется вставка новой строки, приводящая к тому, что повторное выполнение SQL-оператора в транзакции 2 выдаст другой результат.

Проблемы одновременного доступа к данным

Аномалии сериализации – ситуация, когда параллельное выполнение транзакций приводит к результату, невозможному при последовательном выполнении тех же транзакций.

Транзакция 1

```
INSERT INTO tbl1 (f1,f2) VALUES (15,20);
COMMIT;
```

Транзакция 2

```
SELECT SUM(f2) FROM tbl1;

SELECT SUM(f2) FROM tbl1;
```

Уровни изоляции транзакций

Степень обеспечиваемой внутренними механизмами СУБД защиты от всех или некоторых видов несогласованностей данных, возникающих при параллельном выполнении транзакций.

Полная изоляция - результат параллельного выполнения транзакций совпадает с результатом их последовательного выполнения.

Уровни изоляции транзакций

Степень обеспечиваемой внутренними механизмами СУБД защиты от всех или некоторых видов несогласованностей данных, возникающих при параллельном выполнении транзакций.

Стандарт SQL-92 определяет четыре уровня изоляции:

1. **Read uncommitted** – чтение незафиксированных данных
2. **Read committed** – чтение фиксированных данных
3. **Repeatable read** – повторяемость чтения
4. **Serializable** – упорядочиваемость

Первый – самый слабый, последний — самый сильный, каждый последующий включает в себя все предыдущие.

Уровни изоляции транзакций

		Эффекты				
		Потерянное обновление	Грязное чтение	Неповтор-ся чтение	Фантомное чтение	Аномалии сериализации
Уровни изоляции	Read uncommitted	Нет	Допускается	Возможно	Возможно	Возможно
	Read committed	Нет	Нет	Возможно	Возможно	Возможно
	Repeatable read	Нет	Нет	Нет	Допускается	Возможно
	Serializable	Нет	Нет	Нет	Нет	

Повышение уровня изоляции: точность и согласованность данных растет, количество параллельно выполняемых транзакций уменьшается.

Установка уровня изоляции транзакций

MySQL:

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET SESSION TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET GLOBAL TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

Уровни изоляции транзакций

Read uncommitted – чтение незафиксированных данных

Гарантирует только отсутствие потерянных обновлений. Если несколько параллельных транзакций пытаются изменять одну и ту же строку таблицы, то в окончательном варианте строка будет иметь значение, определенное всем набором успешно выполненных транзакций.

При этом возможно считывание не только логически несогласованных данных, но и данных, изменения которых ещё не зафиксированы.

Уровни изоляции транзакций

Read committed – чтение зафиксированных данных

Если транзакция начала изменение данных, то никакая другая транзакция не сможет прочитать их до завершения первой.

Тем не менее, в процессе работы одной транзакции другая может быть успешно завершена и сделанные ею изменения зафиксированы. В итоге первая транзакция будет работать с другим набором данных.

Уровни изоляции транзакций

Repeatable read (snapshot isolation) – повторяемость чтения

Если транзакция считывает данные, то никакая другая транзакция не сможет их изменить. При повторном чтении они будут находиться в первоначальном состоянии.

Однако другие транзакции могут вставлять новые строки, соответствующие условиям поиска инструкций, содержащихся в текущей транзакции. При повторном запуске инструкции текущей транзакцией будут извлечены новые строки, что приведёт к фантомному чтению.

Уровни изоляции транзакций

Repeatable read (snapshot isolation) – повторяемость чтения

MultiVersion Concurrency Control (MVCC)

- Разные пользователи могут одновременно работать с одними и теми же данными;
- Каждый пользователь видит свой изолированный срез данных;
- Изменения, вносимые пользователем, никому не видны до завершения транзации.

Уровни изоляции транзакций

Serializable – упорядочиваемость (блокирует чтение)

Если транзакция обращается к данным, то никакая другая транзакция не сможет добавить новые или удалить имеющие строки, которые могут быть считаны при выполнении транзакции.

Такая блокировка накладывается не на конкретные строки таблицы, а на строки, удовлетворяющие определенному логическому условию.