Изоляция транзакций

Фролков Иван

www.postgrespro.ru

Что это такое и зачем надо

- Почти все слышали
- Мало кто знает, что это такое
  - Где и когда обращать внимание
  - Какие проблемы могут возникнуть

Стандартные уровни изоляции

- Read uncommited
  - Можно увидеть изменения незавершенных транзакций
- Read commited
  - Можно увидеть изменения только завершенных транзакций
- Repeatable read
  - То, что прочитано однажды, будет прочитано еще раз
- Serializable
  - Полная иллюзия последовательного выполнения, в частности, отсутствие неожиданно появляющихся строк

Откуда это все появилось

- Двухфазная блокировка
  - Во время работы транзакции блокировки накапливаются и снимаются только во время завершения работы транзакции
  - Виды блокировок
    - FOR UPDATE/exclusive
    - FOR SHARE/shared

Как все это некогда выглядело

	Свои shared	Чужие exclusive для чтения
Read uncommited	Не накладывает	Игнорирует
Read commited	Не накладывает	Соблюдает
Repeatable read	Накладывает на каждую прочитанную строку	Соблюдает
Serializable	Накладывает на всю таблицу	Соблюдает

Postgres PROFESSIONAL

В документации

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Serialization Anomaly
Read uncommitted	Allowed, but not in PG	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible	Possible
Repeatable read	Not possible	Not possible	Allowed, but not in PG	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Как все это выглядит в Postgres

- xmin, xmax
- clog
- txid_current
- txid_current_snapshot
  - 100:110,101,102,107
- max_pred_locks_per_transaction и т.п.

Как это все выглядит в Postgres

- Read uncommited — уровень отсутствует
- Read commited — по умолчанию, shapshot берется на каждый запрос
- Repeatable read — есть, и без фантомов, shapshot берется на всю транзакцию
- **Serializable — он настоящий!** - snapshot на всю транзакцию + предикатные блокировки (SILocks)

Это все, конечно, хорошо...

- Уровень изоляции по умолчанию — с виду очень хороший: большинство использует именно его и даже об этом не задумывается.
- А если так, то зачем это надо?
  - Ну вообще-то неплохо знать, почему это происходит
  - Некоторые ошибки различного уровня неприятности
  - Не все так просто

Postgres PROFESSIONAL

База данных не только хранилище, но и средство синхронизации бизнес-процессов

Примеры ошибок read commited

- Комментарии к посту
  - Изучение логов показало, что иногда при добавлении комментария возникает нарушение внешнего ключа.
  - Причина: пост был удален после начала транзакции, добавляющей комментарий.
- Что делать?
  - Блокировать пост (for share/for update)
  - Поставить нужный уровень изоляции

Примеры ошибок read commited

- Двойные выплаты
  - Две таблицы client(id), client_payment(id, client_id)
  - Одному клиенту было направлено две выплаты. Почему?
  - Причина: две одновременно работающих транзакции не нашли произведенных выплат payment
- Что делать?
  - Блокировать клиента (for update/for share)
  - Уникальный ключ на client_payment(client_id)
  - Уровень изоляции

Примеры ошибок read commited

- Сложные условия и read commited
  - Таблица balance(user_id, acc_no, amt, flag)=[(1,1,99,false),[1,2,100,false])
  - Транзакция 1:
    - update balance set amt=amt+1 where user_id=1
  - Транзакция 2:
    - Update balance set flag=**true** where user_id=1 and amt=100
  - Казалось бы…
    - balance(user_id, acc_no, amt, flag)=[(1,1,100,**false**),[1,2,101,**false**])
    - Почему?!!
    - Потому что в **при изменении данных при поиске подходящих строк сначала читаются старые значения и используются в вычислении условия, а при собственно обновлении после получения блокировки условия перепроверяются**: первая строка пропущена, потому что прочитано 99, а вторая — потому что после получения блокировки она уже 101

Примеры ошибок repeatable read

- Таблица t(v int) = [1,2,3]
- Запрос – insert into t select sum(v) from t
- Два параллельных запроса в других БД **могут** дать неверный результат:
  t=[1,2,3,6,6]

Примеры ошибок repeatable read

- Черное и белое
- Таблица, у половины строк колонка равна «черное», у другой половины — «белое»
- Транзакции «все черное сделать белым», и «все белое — черным»
- Две параллельных транзакции поменяют черное и белое местами
- Две последовательных — поставят в один цвет
- Результат выполнения непредсказуем!

Примеры ошибок repeatable read

- Две таблицы, заказы `order(id,batch_num references batch(num),…)` и пачки `batch(num)`
- Две процедуры: `place_orders`(берет batch.num) и `increment_batch_num`
- Отчет `build_order` по `order` с `batch_num=batch.num-1`
- При последовательности операций ниже получаем неверный отчет:

place_orders	report	increment_batch_num
`select batch.num;`		
		`update batch set num=num+1;` `commit`

	`select batch.num into bn;`
	`select count(*) from order`
	`where batch_num=bn-1;`
	`commit;`	

`insert into orders…;` `commit;`		

Что делать?

(Если проблема может случиться, она случится — это лишь вопрос времени)

- Понимать причины
- Уметь их устранять
  - Или знать, что это практически невозможно
- Способы устранения
  - Блокировки вручную
  - Уровни изоляции
- В сложных случаях лучше использовать serializable
  - ERROR: could not serialize access due to concurrent update
  - SQLSTATE value of '40001'
  - Как?
    - Триггер с проверкой уровня изоляции
    - Вся важная бизнес-логика в функциях
      - select current_setting('transaction_isolation') && security definer;
  - Могут быть проблемы с памятью
  - До 12 версии serializable не параллелился
- Замечание в сторону: всегда надо проверять количество удаленных/обновленных/вставленных строк – могут быть сюрпризы

Postgres PROFESSIONAL

Вопросы?

Заголовок

- Текст

Заголовок

- Текст