ТРАНЗАКЦИИ

Бежать быстро – это значит
делать медленные шаги
без перерывов между ними.

Б. Джонсон

Технологии баз данных

Содержание

- Понятие транзакции
- АСИД-свойства транзакций
- Управление параллельными транзакциями

Транзакция

Accounts

AcctNo	Balance	…
000374	25000	
…		
123456	10000	
…		

Перевод средств

Accounts

AcctNo	Balance	…
000374	20000	
…		
123456	15000	
…		

- Банковская транзакция состоит из двух операций:
  1. update Accounts
     set Balance=Balance-5000
     where AccNo='000374';
  2. update Accounts
     set Balance=Balance+5000
     where AccNo='123456';
- Что произойдет в случае сбоя после выполнения первого шага транзакции?

Транзакция

4

- *Транзакция* – набор операций над базой данных, которые рассматриваются как *одна неделимая* операция и выполняют перевод базы данных из одного согласованного состояния в другое.
- В ходе выполнения транзакции согласованность базы данных может временно нарушаться.
- Результатом транзакции может быть *фиксация (COMMIT)* или *откат (ROLLBACK)* **всех** входящих в нее операций.

Транзакция

5

```
BEGIN TRANSACTION
  update Accounts
  set Balance=Balance-5000
  where AccNo='000374';
  if SQLState<>'00000' goto UNDO;
  update Accounts
  set Balance=Balance+5000
  where AccNo='123456';
   if SQLState<>'00000' goto UNDO;
  COMMIT;
  goto Finish;
  Undo: ROLLBACK;
  Finish: ;
END TRANSACTION
```


Диспетчер транзакций

6

- В состав СУБД входит *диспетчер транзакций*.
- Команды диспетчера транзакций
  - *COMMIT (зафиксировать)*
    - сигнал диспетчеру транзакций об *успешном окончании* транзакции: логическая единица работы успешно завершена, база данных вновь находится в непротиворечивом состоянии
    - *все обновления транзакции необходимо внести в базу данных*.
  - *ROLLBACK (откатить)*
    - сигнал диспетчеру транзакций о *неудачном окончании* транзакции: в процессе выполнения логической единицы работы произошла ошибка, вследствие которой база данных находится в противоречивом состоянии
    - *все обновления транзакции необходимо отменить.*

Журнал транзакций

7

- Диспетчер транзакций осуществляет ведение *журнала транзакций*, в котором отмечаются операции, производимые транзакциями и результаты их завершения.
  - Журнал транзакций заполняется по *принципу опережающей записи WAL (Write Ahead Log)* – данные об операции сначала записываются в журнал, а затем производится операция.
  - Журнал транзакций используется для восстановления базы данных после сбоев.

АСИД-свойства транзакции

8

- Транзакция должна обладать *АСИД-свойствами*.
  - ***А**томарность (**A**tomicity)* – выполняются все операторы транзакции или ни один.
  - ***С**огласованность (**C**onsistency)* – перевод базы данных из одного согласованного состояния в другое.
  - ***И**золированность (**I**solation)* – параллельные транзакции не могут повлиять друг на друга.
  - ***Д**олговечность (**D**urability)* – изменения, произведенные зафиксированной транзакцией, не могут быть потеряны ни при каких обстоятельствах.

Транзакции в SQL

9

- Транзакция начинается
  - *явно* оператором START TRANSACTION
  - *неявно* первым выполняемым оператором SQL*.
- Транзакция завершается
  - *явно*
    - *фиксируется* оператором COMMIT
    - *откатывается* оператором ROLLBACK
  - *неявно*
    - *фиксируется* оператором модификации объекта схемы (CREATE, DROP, ALTER) или в случае завершения сессии пользователем
    - *откатывается* в случае аварийного завершения сессии пользователя

* Исполнение триггеров, активируемых оператором транзакции, является частью той же транзакции.

Выполнение оператора vs фиксация транзакции

10

- Успешное выполнение оператора транзакции не гарантирует сохранение результатов его выполнения в случае отката всей транзакции.

```
SQL>
create table emp (id number primary key, name char(20), age number);
insert into emp values (1, 'Иванов', 40);
insert into emp values (2, 'Петрова', 30);
insert into emp values (3, 'Сидоров', 50);
rollback;
select * from emp;
ID      NAME          AGE
нет строк
```


Откат оператора vs откат транзакции

10

- Неуспешное выполнение одного оператора транзакции не означает откат всей транзакции.

```
SQL>
create table emp (id number primary key, name char(20), age number);
insert into emp values (1, 'Иванов', 40);
insert into emp values (2, 'Петрова', 30);
insert into emp values (2, 'Сидоров', 50);
Ошибка! Дубликат первичного ключа.
commit;
select * from emp;
ID      NAME          AGE
1       Иванов        40
2       Петрова       30
```


Точки сохранения транзакции

12

- *Точка сохранения (savepoint)* – текстовая метка внутри транзакции.
- Точки сохранения используются для разбиения длинной транзакции на небольшие части.
- *Откат до точки сохранения (rollback to savepoint)* позволяет откатить не всю транзакцию, а только изменения после указанной точки до текущей точки транзакции.
  - Откатываются только операторы транзакции, выполненные *после* точки сохранения.
  - Указанная точка сохранения остается, но все точки сохранения, установленные после указанной, теряются.

Точки сохранения транзакции

13

```
savepoint update_cur_data;
    update emp set …;
    update dept set …;
    update job set …;
savepoint delete_old_data;
    delete from emp …;
    delete from dept …;
    delete from job …;
savepoint insert_new_data;
    insert into emp values (…);
    insert into dept values (…);
    insert into job values (…);
rollback to savepoint delete_old_data;
    delete from job …;
savepoint insert_new_data;
    insert into emp values (…);
    insert into dept values (…);
    insert into job values (…);
commit;
```

```
savepoint update_cur_data;
    update emp set …;
    update dept set …;
    update job set …;
savepoint delete_old_data;

    delete from job …;
savepoint insert_new_data;
    insert into emp values (…);
    insert into dept values (…);
    insert into job values (…);
commit;
```


Режимы доступа транзакции к данным

14

- Транзакция может быть запущена в одном из двух режимов: READ-WRITE (по умолчанию) и READ-ONLY.
- В *режиме READ-WRITE* транзакция
  - может модифицировать объекты базы данных
  - видит изменения, вносимые в базу данных другими транзакциями, – после фиксации этих транзакций.
- В *режиме READ-ONLY* транзакция
  - не может модифицировать объекты базы данных
  - не видит изменений, вносимых в базу данных другими транзакциями.

Проблемы параллельного выполнения транзакций

15

- Правильная сама по себе транзакция может выдать неправильный ответ, если в ее работу каким-либо образом вмешаются другие (также сами по себе правильные) транзакции.
- Проблемы параллельного выполнения транзакций
  - Потерянное обновление
  - Зависимость от незафиксированных результатов
  - Анализ несовместимости

Проблема потерянного обновления

16

Транзакция A	Время	Транзакция B
-		-
retrieve *tuple*	t1	-
-		-
-	t2	retrieve *tuple*
-		-
update *tuple*	t3	-
-		-
-	t4	update *tuple*
-		-

- Транзакция A *теряет обновление* в момент времени t4.

Проблема зависимости от незафиксированных результатов (1)

17

Транзакция A	Время	Транзакция B
-		-
-	t1	update *tuple*
-		-
retrieve *tuple*	t2	retrieve *tuple*
-		-
-	t3	rollback

- Транзакция A *зависит от незафиксированного обновления* в момент времени t2.

Проблема зависимости от незафиксированных результатов (2)

18

Транзакция A	Время	Транзакция B
-		-
-	t1	update *tuple*
-		-
update *tuple*	t2	retrieve *tuple*
-		-
-	t3	rollback

- Транзакция A *обновляет незафиксированное изменение* в момент времени t2 и теряет это обновление в момент времени t3.

Проблема анализа несовместимости

acct1: balance=40; acct2: balance=50; acct3: balance=30.

Транзакция A	Время	Транзакция B
retrieve *acct1* sum=40	t1	-
retrieve *acct2* sum=90	t2	-
-	t3	retrieve *acct3*
-	t4	update *acct3* balance:=20
-	t5	retrieve *acct1*
-	t6	update *acct1* balance:=50
-	t7	commit
retrieve *acct3* **sum=110 (не 120!)**	t8	-

- Транзакция A – подсчет остатков на счетах *acct1*, *acct2*, *acct3*. Транзакция B – перевод суммы 10 со счета *acct3* на счет *acct1*.
- Транзакция A выполнила *анализ несовместимости*.

Блокировка

- Проблемы параллельного выполнения могут быть устранены с помощью механизма блокировок.
- Если некоторой транзакции необходимо, чтобы определенный объект базы данных (как правило, кортеж отношения) не изменился без ее ведома, то она должна *наложить блокировку* на этот объект.
- *Блокировка (lock)* – ограничение (или запрет) доступа к объекту со стороны других транзакций.

Виды блокировок

- *Блокировки записи* (исключительные блокировки, X-блокировки)
- *Блокировки чтения* (разделяемые блокировки, S-блокировки)
- Матрица совместимости блокировок

	X	**S**	**-**
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

Протокол доступа к данным

22

- Транзакция, в которой требуется выполнить выборку кортежа, должна вначале *наложить S-блокировку* на этот кортеж.
- Транзакция, в которой требуется выполнить обновление кортежа, должна вначале *наложить X-блокировку* на этот кортеж.
- Если транзакция ранее уже наложила S-блокировку на этот кортеж (когда выполняется последовательность операций выборки и обновления), то эта транзакция должна *повысить уровень блокировки до уровня X*.
- Если запрос на блокировку от транзакции *B* не может быть немедленно удовлетворен из-за конфликта с блокировкой, которая уже наложена транзакцией *A*, то *B* переходит в *состояние ожидания*. Транзакция *B* ожидает до тех пор, пока у СУБД не появится возможность удовлетворить ее запрос на блокировку (не раньше, чем транзакция *A* освободит блокировку).

Тупики

23

- Ситуация бесконечного ожидания транзакции – *активный тупик (livelock),* или *истощение ресурсов (starvation).*
- СУБД должна обеспечить отсутствие активных тупиков. Стандартный способ – очередь запросов на блокировку ("первым поступил –первым обслуживается").
- Блокировки освобождаются по завершении транзакции (COMMIT или ROLLBACK).

Решение проблемы потерянного обновления

24

Транзакция A	Время	Транзакция B
-		-
retrieve *tuple* **lockS** *tuple* ✓	t1	-
-		-
-	t2	retrieve *tuple* **lockS** *tuple* ✓
-		-
update *tuple* **lockX** *tuple* ×	t3	-
-		-
-	t4	update *tuple* **lockX** *tuple* ×
wait *B*		**wait** *A*

- Транзакция A *НЕ теряет обновление* в момент времени t4, но в этот момент возникает *взаимоблокировка (deadlock).*

Решение проблемы зависимости от незафиксированных результатов

25

Транзакция A	Время	Транзакция B
-		-
-	t1	update *tuple* **lockX** *tuple* ✓
-		-
retrieve *tuple* **lockS** *tuple* ×	t2	
-		-
wait *B*	t3	commit/rollback **release** *tuple*
-		-
retrieve *tuple* **lockS** *tuple* ✓	t4	

- Транзакция A *НЕ зависит от незафиксированного обновления* в момент времени t2.

Проблема зависимости от незафиксированных результатов

26

Транзакция A	Время	Транзакция B
-		-
-	t1	update *tuple* **lockX** *tuple* ✓
-		-
update *tuple* **lockX** *tuple* ×	t2	
-		-
wait *B*	t3	commit/rollback **release** *tuple*
-		-
update *tuple* **lockX** *tuple* ✓	t4	

- Транзакция A *НЕ теряет обновление* в момент времени t3.

Решение проблемы анализа несовместимости

27

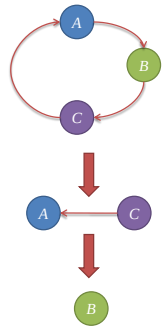
Транзакция A	Время	Транзакция B
retrieve *acct1* **lockS** *acct1* ✓ sum=40	t1	-
retrieve *acct2* **lockS** *acct2* ✓ sum=90	t2	
-	t3	retrieve *acct3* **lockS** *acct3* ✓
-	t4	update *acct3* **lockX** *acct3* ✓ balance:=20
-	t5	retrieve *acct1* **lockS** *acct1* ✓
-	t6	update *acct1* **lockX** *acct1* ×
retrieve *acct3* **lockX** *acct3* ×	t7	**wait** *A*
wait *B*	t8	**wait** *A*

- Транзакция A – подсчет остатков на счетах *acct1*, *acct2*, *acct3*. Транзакция B – перевод суммы 10 со счета *acct1* на счет *acct3*.
- Транзакция A *НЕ выполняет анализ несовместимости*, однако в момент времени t7 возникает *взаимоблокировка (deadlock)*.

Устранение взаимоблокировки

28

- СУБД поддерживает граф ожидания транзакций. Если в этом графе имеется цикл, значит имеет место взаимоблокировка.
- Чтобы разорвать взаимоблокировку, СУБД выбирает одну из транзакций, которые входят в состав цикла в графе ожидания, в качестве "жертвы" и насильно выполняет ее откат. Это позволяет освободить ее блокировки и дать возможность другим транзакциям продолжить свою работу.



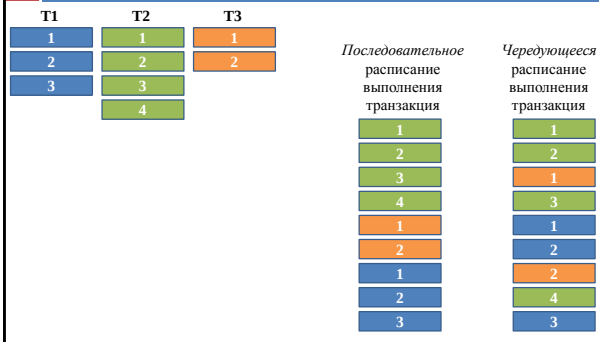
Предотвращение взаимоблокировки

29

- Модифицированные протоколы блокировки, позволяющие исключить возникновение взаимоблокировок.
  - Каждая транзакция обозначается уникальной отметкой времени ее начала.
  - Если транзакция *A* запрашивает блокировку кортежа, который уже заблокирован транзакцией *B*, то
    - *Протокол "ожидание-отмена"*
      - Если транзакция *A* началась раньше, чем *B*, то *A* переходит в состояние ожидания.
      - Иначе откат и перезапуск транзакции *A*.
    - *Протокол "Отмена-ожидание"*
      - Если транзакция *A* началась позже, чем *B*, то *A* переходит в состояние ожидания.
      - Иначе откат и перезапуск транзакции *B*.
  - Перезапускаемой транзакции присваивается ее первоначальная отметка времени.

Организация изолированного выполнения множества транзакций

30



Сериализация транзакций

31

- Транзакция переводит базу данных из одного согласованного состояния в другое ⇒ любое последовательное расписание транзакций является корректным (переводит базу данных в согласованное состояние).
- Чередующееся расписание является *сериализуемым* (корректным) ⇔ оно эквивалентно некоторому последовательному расписанию (гарантирует получение идентичных результатов независимо от начального состояния базы данных).

Теорема двухфазной блокировки

32

- Любое чередующееся расписание двухфазных транзакций является сериализуемым.
- Транзакция является *двухфазной*, если она подчиняется протоколу двухфазной блокировки.
- *Протокол двухфазной блокировки*
  1. Перед выполнением операции с любым объектом транзакция должна наложить блокировку на этот объект.
  2. После освобождения любой блокировки транзакция больше не должна налагать какие-либо дополнительные блокировки.

Уровни изоляции транзакций в SQL

33

Проблема / Уровень изоляции	Грязное чтение	Неповторяемое чтение	Фантомы
READ UNCOMMITTED	+	+	+
READ COMMITTED	–	+	+
REAPEATABLE READ	–	–	+
SERIALIZABLE	–	–	–

Заключение

- Транзакция – набор операций над базой данных, которые рассматриваются как одна неделимая операция и выполняют перевод базы данных из одного согласованного состояния в другое. Результат транзакции – фиксация (COMMIT) или откат (ROLLBACK) всех входящих в нее операций.
- АСИД-свойства транзакций: атомарность, согласованность, изолированность, долговечность.
- Проблемы параллельного выполнения транзакций: потерянное обновление, зависимость от незафиксированных результатов, анализа несовместимости.
- Блокировки помогают в решении проблем параллельного выполнения транзакций. X-блокировки, S-блокировки. Протокол доступа к данным на основе блокировок.
- Сериализация транзакций. Теорема двухфазной блокировки.