Уровни изоляции транзакций

Графеева Н.Г.

2015

Свойства транзакций (ACID)

- Неделимость (**Atomicity**). Транзакция либо выполняется полностью, либо не выполняется.
- Согласованность (**Consistency**). Транзакция переводит базу данных из одного согласованного состояния в другое.
- Изолированность (**Isolation**). Результаты транзакции становятся доступны для других транзакций только после ее фиксации.
- Продолжительность (**Durability**). После фиксации транзакции изменения становятся постоянными.

- *Примечание: однако за свойством изолированности в действительности стоят весьма различающиеся понятия. Различаются, так называемые, уровни изоляции транзакций.*

Понятия, используемые для определения уровней изолированности транзакций

- • ***Грязное чтение (dirty read).*** *Допус*кается **чтение незафиксированных**, или "грязных", данных. При этом нарушается как целостность данных, так и требования внешнего ключа, а требования уникальности игнорируются.

- • ***Неповторяемость при чтении (non-repeatable read).*** Это означает, что если строка читается в момент времени T1, а затем перечитывается в момент времени T2, то за этот период она **может измениться**. Строка может исчезнуть, может быть обновлена и т.д.

- • ***Чтение фантомов (phantom read).*** Это означает, что если выполнить запрос в момент времени T1, а затем выполнить его повторно в момент времени T2, в базе данных могут появиться дополнительные строки. От неповторяемости при чтении это явление отличается тем, что прочитанные данные не изменились, но критериям запроса стало удовлетворять **больше данных, чем** прежде.

Уровни изолированности транзакций (стандарт SQL92)

Уровень изолированности	Грязное чтение	Неповторяемость при чтении	Чтение фантомов
READ UNCOMMITTED	Разрешено	Разрешено	Разрешено
READ COMMITTED		Разрешено	Разрешено
REPEATABLE READ			Разрешено
SERIALIZABLE			

- В большинстве промышленных СУБД по умолчанию поддерживается уровень READ COMMITTED!!!!

Механизмы поддержки уровней изолированностей

- Блокировка данных
- Многоверсионность данных

Механизм блокировки данных

- В основе механизма – использование двух видов замков на уровне записей:
  - SHARE-LOCK
  - EXCLUSIVE-LOCK
- Записям, которые программа читает, приписывается по умолчанию замок *SHARE-LOCK* (при этом другие пользователи не могут изменить эту запись, но могут читать). Записям, которые программа изменяет, приписывается по умолчанию замок *EXCLUSIVE-LOCK* (другие не могут даже читать). Записи могут быть прочитаны и без замка (если явно указать NO-LOCK).
- Замки снимаются по окончании транзакции.

3 режима работы с данными

- с замком *EXCLUSIVE-LOCK* — режим *EXCLUSIVE-LOCK*
- с замком *SHARE-LOCK* — режим *SHARE-LOCK*
- без учета замков — режим *NO-LOCK*

Согласование режимов

если на записи:	для других возможны:
NO-LOCK (нет замков)	EXCLUSIVE-LOCK SHARE-LOCK NO-LOCK
SHARE-LOCK	SHARE-LOCK NO-LOCK
EXCLUSIVE-LOCK	NO-LOCK

Пример

```
create table accounts
( account_number number primary key,
  account_balance number
);
```

Содержимое таблицы account

Строка	Номер счета	Баланс счета
1	123	500,00 $
2	234	250,00 $
3	345	400,00 $
4	456	100,00 $

Действия пользователей

- **Первый пользователь** выполняет запрос:
- select sum(account_balance) from accounts;

- **Второй пользователь** выполняет транзакцию, переводящую 400 $ со счета 123 на счет 456 (в тот момент времени, когда первый пользователь прочитал строку 2, но еще не прочитал остальные).

- **Как** будет получен очевидный результат результат – 1250?

Сценарий 1(исполнение транзакции с уровнем изоляции REPEATABLE READ на основе блокировок)

Время	Запрос	Транзакция
T1	Читает строку 1, sum = 500 $. На строке 1 устанавливается разделяемая блокировка.	
T2	Читает строку 2, sum = 750 $. На строке 2 устанавливается разделяемая блокировка	
T3		Пытается изменить строку 1, но эта попытка блокируется. Транзакция приостанавливается, пока не сможет установить исключительную блокировку.
T4	Читает строку 3	
T5	Читает строку 4, выдает результат суммирования	
T6	Фиксируется транзакция.	
T7		Изменяет строку 1, устанавливая на соответствующую строку исключительную блокировку. Теперь баланс в этой строке имеет значение 100 $.
T8		Изменяет строку 4 устанавливая на соответствующую строку исключительную блокировку. Теперь баланс в этой строке становится равным 500 $. Транзакция фиксируется.

Сценарий 2 (исполнение транзакции с уровнем изоляции REPEATABLE READ на основе многоверсионности данных)

Время	Запрос	Транзакция
T1	Читает строку 1, sum получает значение 500 $	
T2	Читает строку 2, sum получает значение 750 $	
T3		Изменяет строку 1, устанавливает исключительную блокировку на строку 1, предотвращая другие изменения. В строке 1 теперь хранится значение 100 $
T4	Читает строку 3, sum получает значение 1150 $	
T5		Изменяет строку 4, устанавливает исключительную блокировку на строку 4, предотвращая другие изменения (но не чтение). В строке 4 теперь хранится значение 500 $.
T6	Читает строку 4, определяет, что она была изменена. Выполняется откат блока до того состояния, которое он имел в момент времени T1. Запрос затем прочитает значение 100 $ из этого блока	
T7		Транзакция фиксируется

Взаимные блокировки пользователей

- Разделяемые блокировки могут не только снижать возможности параллельного доступа, но и приводить и к более трагическим последствиям – взаимным блокировкам.

Действия пользователей

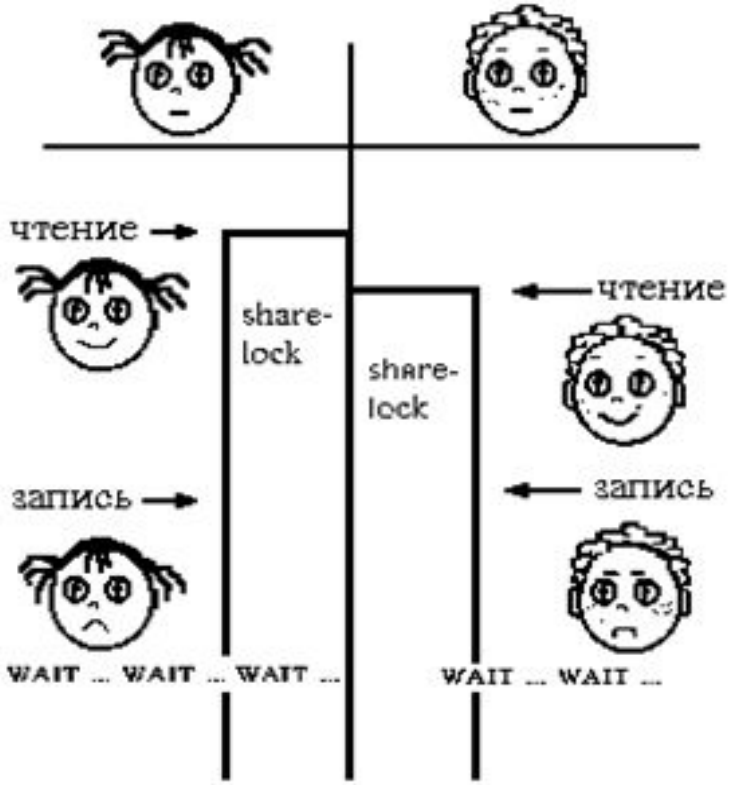
- **Первый пользователь** выполняет запрос:
- select sum(account_balance) from accounts;

- **Второй пользователь** выполняет транзакцию, переводящую 400 $ со счета 345 на счет 234 (в тот момент времени, когда первый пользователь прочитал строку 2, но еще не прочитал остальные).

Сценарий 3(возникновение взаимной блокировки во время исполнения транзакции REPEATABLE READ на основе блокировок)

Время	Запрос	Транзакция
T1	Читает строку 1. Устанавливает разделяемую блокировку на строку 1.	
T2	Читает строку 2. Устанавливает разделяемую блокировку на строку 2.	
T3		Редактирует строку 3. Устанавливает исключительную блокировку на строку 3.
T4	Пытается прочитать строку 3. Транзакция приостанавливается в ожидании ресурса.	
T5		Пытается отредактировать строку 2. Возникает взаимная блокировка.

Сценарий 4 (взаимная блокировка после чтения и попытки редактирования)



Как разрешается проблема взаимной блокировки?

- В различных СУБД по разному. Как правило ищется жертва…

Уровень изолированности SERIALIZABLE

- Этот уровень изолированности транзакции обычно считают наиболее сильным, именно он обеспечивает самую высокую степень изолированности. Транзакция с уровнем изолированности SERIALIZABLE работает в среде, где как бы нет других пользователей, изменяющих данные в базе данных; база данных "замораживается" на момент начала транзакции. Побочные эффекты (изменения) от других транзакций для такой транзакции невидимы, независимо от того, как долго она выполняется. Такой уровень изолированности нужно задавать явным образом.

Пример (ORACLE. Создание транзакции уровня SERIALIZABLE)

- create table a (x int) ;
- create table b (x int) ;

Пример (продолжение)

	Сеанс 1	Сеанс 2
T1	Alter session set isolation_level=serializable;	
T2		Alter session set isolation_level=serializable;
T3	insert into a select count(*) from b;	
T4		insert into b select count(*) from a;
T5	commit;	
T6		commit;

Пример (продолжение)

- Результат – две таблицы с значениями 0 и 0. Если бы не выполнялась команда

- **`alter session`**
- **`set isolation_level=serializable`**

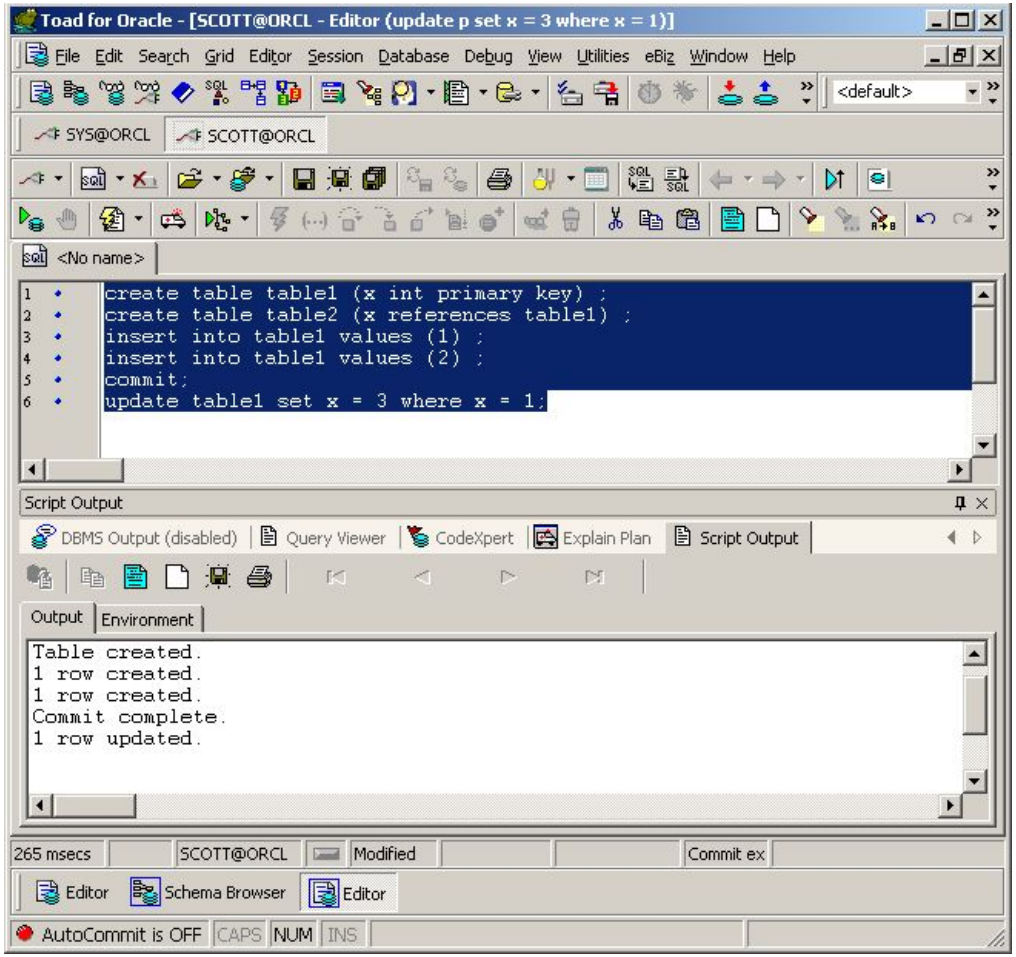
- результат был бы – 0 и 1.

Внешние ключи и взаимные блокировки

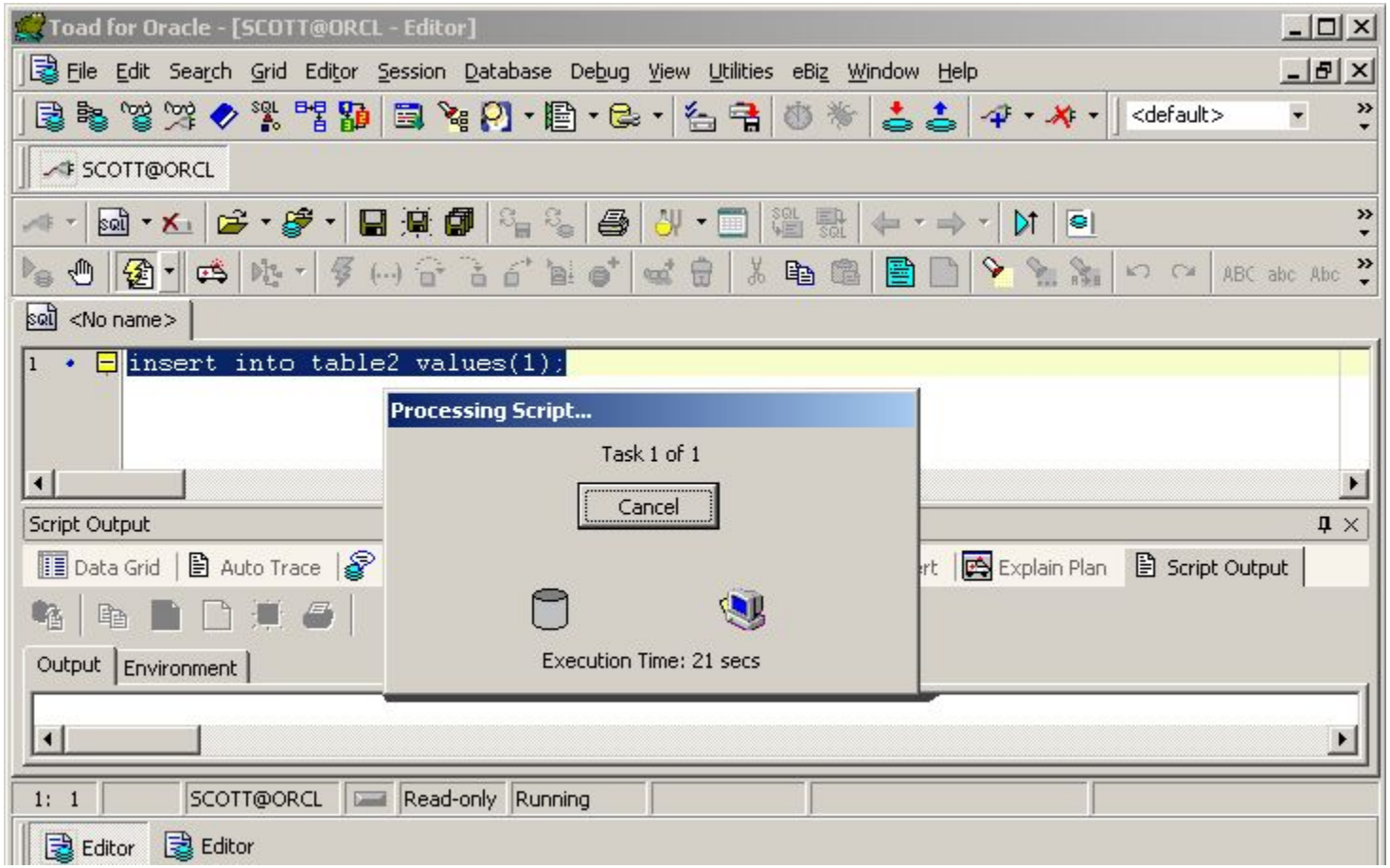
В Oracle также возникают взаимные блокировки. Основной причиной их возникновения являются неиндексированные внешние ключи. При изменении главной таблицы сервер Oracle **полностью** блокирует подчиненную таблицу в двух случаях:

- при изменении первичного ключа в главной таблице подчиненная таблица блокируется (при отсутствии индекса по внешнему ключу);
- при удалении строки в главной таблице подчиненная таблица также полностью блокируется (при отсутствии индекса по внешнему ключу).

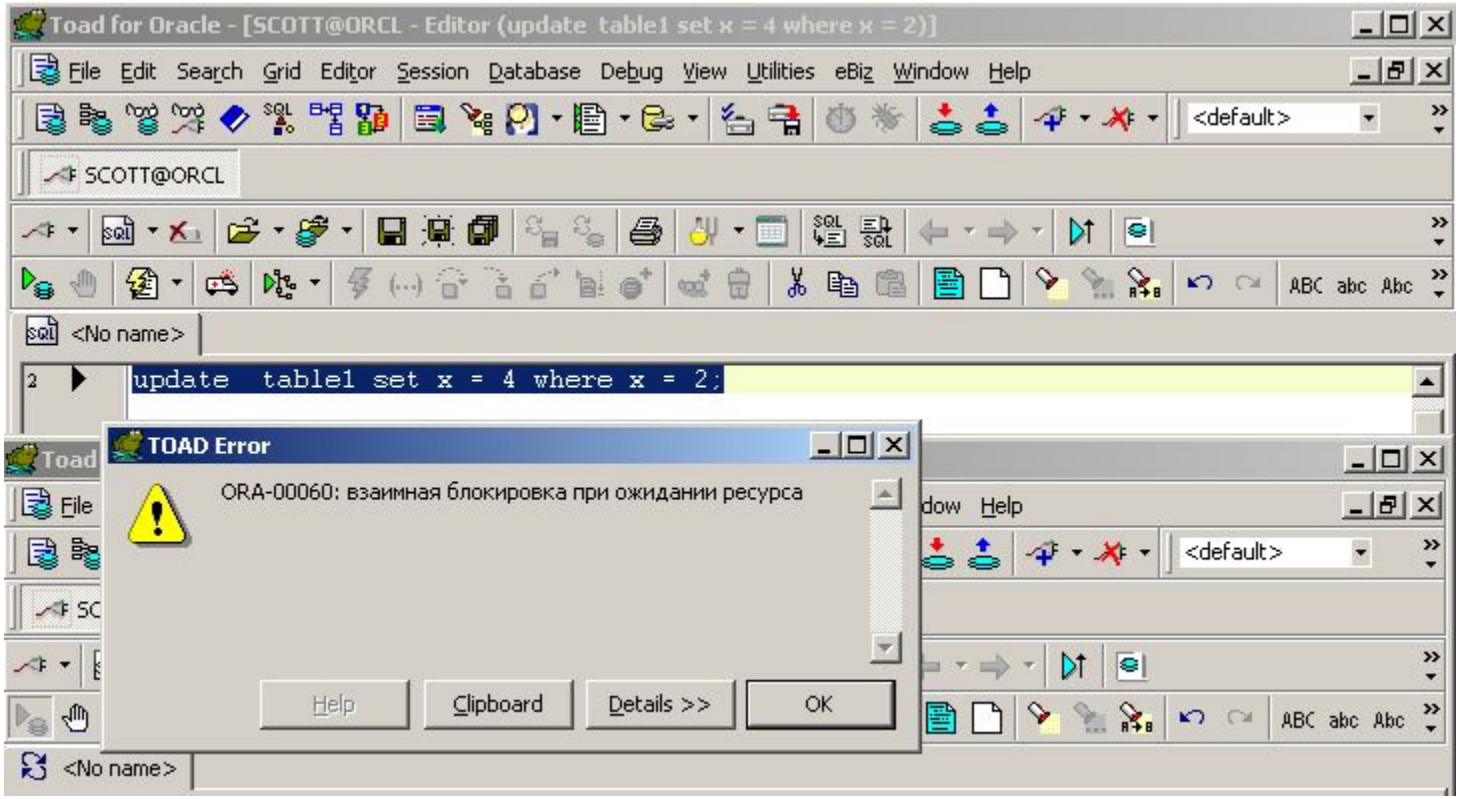
Пример. Шаг 1(сеанс 1)



Пример. Шаг 2 (сеанс 2)



Пример. Шаг 3 (сеанс 1)



Когда индекс по внешнему ключу не нужен?

- не удаляются строки из главной таблицы;
- не изменяется значение уникального/первичного ключа главной таблицы;
- не выполняется соединение главной и подчиненной таблиц

- А так практически не бывает…..

Домашнее задание 16(10 баллов)

- Создать конкурирующие jobs, провоцирующие неповторяемость при чтении и чтение фантомов. Результаты (с подробными комментариями) фиксировать в debug_log.

Результат отправьте по адресу N.Grafeeva@spbu.ru. Тема письма – DB_Applcation_2015_job16.

Примечание:задание должно быть отправлено в течение 2 недель. За более позднее отправление будут сниматься штрафные баллы (по баллу за каждые 2 недели).

За сдачу 21 ноября – дополнительные 5 баллов.