МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Московский авиационный институт”

(национальный исследовательский университет)

Факультет №3 «Системы управления, информатика и электроэнергетика»

Отчет по домашней работе №8

по курсу «**Управление базами данных**»

Выполнили:

студенты группы 3О-218М-19:

Пономарев Роман

Принял:

Доцент, к.т.н. Моргунов Е.П.

Москва, 2020

№2)

Транзакции, работающие на уровне изоляции Read Committed, видят только свои собственные обновления и обновления, зафиксированные параллельными транзакциями. При этом нужно учитывать, что иногда могут возникать ситуации, которые на первый взгляд кажутся парадоксальными, но на самом деле все происходит в строгом соответствии с этим принципом. Воспользуемся таблицей «Самолеты» (aircrafts) или ее копией. Предположим, что мы решили удалить из таблицы те модели, дальность полета которых менее 2 000 км. В таблице представлена одна такая модель — Cessna 208 Caravan, имеющая дальность полета 1 200 км. Для выполнения удаления мы организовали транзакцию. Однако параллельная транзакция, которая, причем, началась раньше, успела обновить таблицу таким образом, что дальность полета самолета Cessna 208 Caravan стала составлять 2 100 км, а вот для самолета Bombardier CRJ-200 она, напротив, уменьшилась до 1 900 км. Таким образом, в результате выполнения операций обновления в таблице по-прежнему присутствует строка, удовлетворяющая первоначальному условию, т. е. значение атрибута range у которой меньше 2000. Наша задача: проверить, будет ли в результате выполнения двух транзакций удалена какая-либо строка из таблицы. На первом терминале начнем транзакцию, при этом уровень изоляции Read Committed в команде указывать не будем, т. к. он принят по умолчанию:

```
SELECT *
  FROM aircrafts_tmp
  WHERE range < 2000;

 aircraft_code |        model       | range
---------------+--------------------+-------
 CN1           | Cessna 208 Caravan |  1200
(1 строка)

UPDATE aircrafts_tmp
  SET range = 2100
  WHERE aircraft_code = 'CN1';

UPDATE 1

UPDATE aircrafts_tmp
  SET range = 1900
  WHERE aircraft_code = 'CR2';

UPDATE 1
```

На втором терминале начнем вторую транзакцию, которая и будет пытаться удалить строки, у которых значение атрибута range меньше 2000.

```
BEGIN;

BEGIN
```

```
SELECT *
  FROM aircrafts_tmp
  WHERE range < 2000;

 aircraft_code |        model       | range
---------------+--------------------+-------
 CN1           | Cessna 208 Caravan |  1200
(1 строка)
```

```
DELETE FROM aircrafts_tmp WHERE range < 2000;
```

Введя команду DELETE, мы видим, что она не завершается, а ожидает, когда со строки, подлежащей удалению, будет снята блокировка. Блокировка, установленная командой UPDATE в первой транзакции, снимается только при завершении транзакции, а завершение может иметь два исхода: фиксацию изменений с помощью команды COMMIT (или END) или отмену изменений с помощью команды ROLLBACK.

Давайте зафиксируем изменения, выполненные первой транзакцией. На первом терминале сделаем так:

COMMIT;
COMMIT

Тогда на втором терминале мы получим такой результат от команды DELETE:
DELETE 0

Чем объясняется такой результат? Он кажется нелогичным: ведь команда SELECT, выполненная в этой же второй транзакции, показывала наличие строки, удовлетворяющей условию удаления. Объяснение таково: поскольку вторая транзакция пока еще не видит изменений, произведенных в первой транзакции, то команда DELETE выбирает для удаления строку, описывающую модель Cessna 208 Caravan, однако эта строка была заблокирована в первой транзакции командой UPDATE. Эта команда

изменила значение атрибута range в этой строке. При завершении первой транзакции блокировка с этой строки снимается (со второй строки — тоже), и команда DELETE во второй транзакции получает возможность заблокировать эту строку. При этом команда DELETE данную строку перечитывает и вновь вычисляет условие WHERE применительно к ней. Однако теперь условие WHERE для данной строки уже не выполняется, следовательно, эту строку удалять нельзя. Конечно, в таблице есть теперь другая строка, для самолета Bombardier CRJ-200, удовлетворяющая условию удаления, однако повторный поиск строк, удовлетворяющих условию WHERE в команде DELETE, не производится. В результате не удаляется ни одна строка. Таким образом, к сожалению, имеет место нарушение согласованности, которое можно объяснить деталями реализации СУБД. Завершим вторую транзакцию:

END;
COMMIT

Вот что получилось в результате:

SELECT * FROM aircrafts_tmp;

```
 aircraft_code |        model         | range
---------------+----------------------+-------
 773           | Boeing 777-300       | 11100
 763           | Boeing 767-300       |  7900
 SU9           | Sukhoi SuperJet-100  |  3000
 320           | Airbus A320-200      |  5700
 321           | Airbus A321-200      |  5600
 319           | Airbus A319-100      |  6700
 733           | Boeing 737-300       |  4200
 CN1           | Cessna 208 Caravan   |  2100
 CR2           | Bombardier CRJ-200   |  1900
(9 строк)
```

Задание. Модифицируйте сценарий выполнения транзакций: в первой транзакции вместо фиксации изменений выполните их отмену с помощью команды ROLLBACK и посмотрите, будет ли удалена строка и какая конкретно.

Решение

Терминал 1:
BEGIN;

SELECT *
FROM aircrafts_tmp
WHERE range < 2000;

UPDATE aircrafts_tmp
SET range = 2100

```
WHERE aircraft_code = 'CN1';

UPDATE aircrafts_tmp
SET range = 1900
WHERE aircraft_code = 'CR2';
```

Терминал 2:

```
BEGIN;

SELECT *
FROM aircrafts_tmp
WHERE range < 2000;

DELETE FROM aircrafts_tmp WHERE range < 2000;
```

Терминал 1:

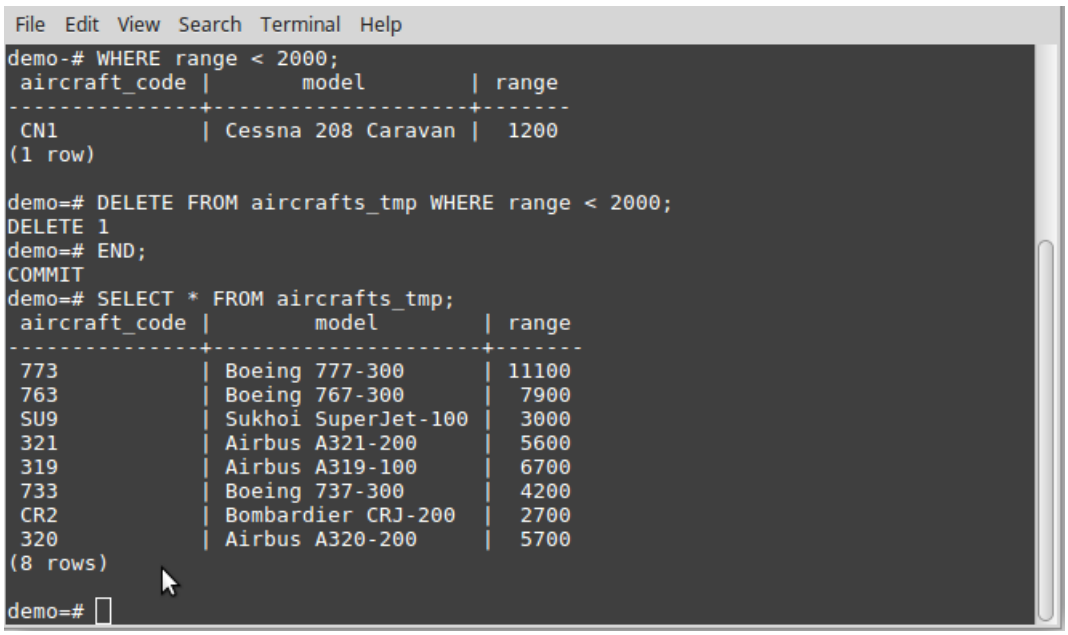
```
ROLLBACK;
```

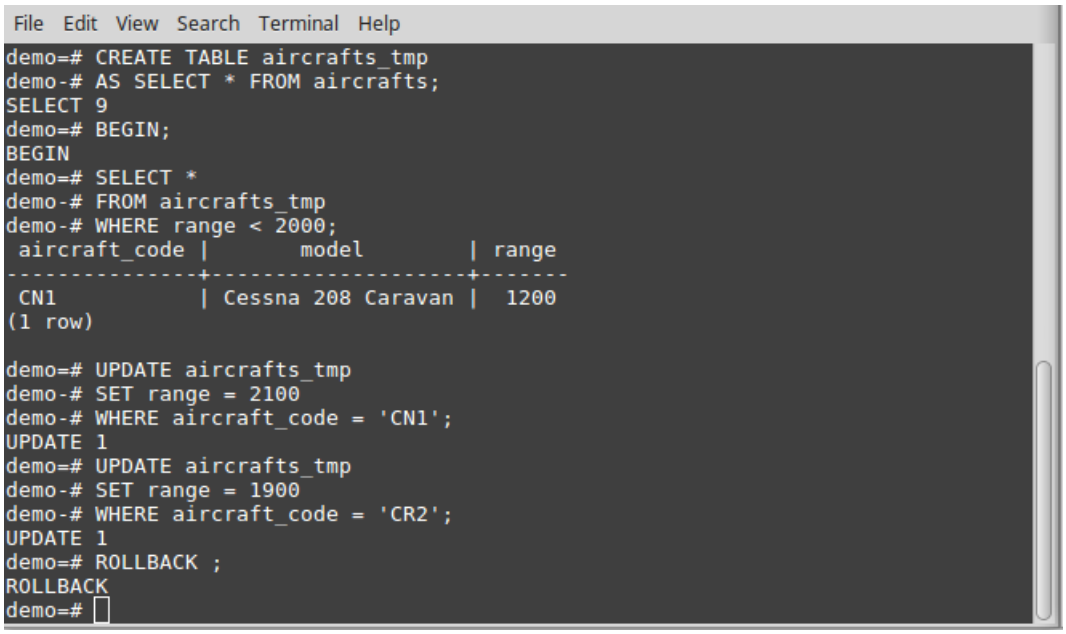
Терминал 2:

```
END;

SELECT * FROM aircrafts_tmp;
```

Удалилась строка, описывающая модель Cessna 208 Caravan;
Объяснение таково: поскольку вторая транзакция пока еще не видит изменений, произведенных в первой транзакции, то команда DELETE выбирает для
удаления строку, описывающую модель Cessna 208 Caravan, однако эта строка
была заблокирована в первой транзакции командой UPDATE. Эта команда изменила значение атрибута range в этой строке.
При завершении первой транзакции блокировка с этой строки снимается (со второй строки — тоже), и команда DELETE во второй транзакции получает возможность заблокировать эту строку. При этом команда DELETE данную строку
перечитывает и вновь вычисляет условие WHERE применительно к ней. Однако теперь условие WHERE для данной строки выполняется, так как к первой транзакции был применен ROLLBACK и изменения откатились, следовательно, эту строку необходимо удалять.





№3)

1. Когда говорят о таком феномене, как потерянное обновление, то зачастую в качестве примера приводится операция UPDATE, в которой значение какого-то атрибута изменяется с применением одного из действий арифметики. Например:

```
UPDATE aircrafts_tmp
    SET range = range + 200
    WHERE aircraft_code = 'CR2';
```

При выполнении двух и более подобных обновлений в рамках параллельных транзакций, использующих, например, уровень изоляции Read Committed,

будут учтены все такие изменения (что и было показано в тексте главы). Очевидно, что потерянного обновления не происходит. Предположим, что в одной транзакции будет просто присваиваться новое значение, например, так:

```
UPDATE aircrafts_tmp
    SET range = 2100
    WHERE aircraft_code = 'CR2';
```

А в параллельной транзакции будет выполняться аналогичная команда:

```
UPDATE aircrafts_tmp
    SET range = 2500
    WHERE aircraft_code = 'CR2';
```

Очевидно, что сохранится только одно из значений атрибута range. Можно ли говорить, что в такой ситуации имеет место потерянное обновление? Если оно имеет место, то что можно предпринять для его недопущения? Обоснуйте ваш ответ. Для получения дополнительной информации можно обратиться к фундаментальному труду К. Дж. Дейта, а также к полному руководству по SQL Дж. Гроффа, П. Вайнберга и Э. Оппеля. Библиографические описания этих книг приведены в списке рекомендуемой литературы.

Решение

Нет, нельзя говорить о потерянном обновлении, так как уровень READ COMMITED гарантирует отсутствие потерянных обновлений. Если же 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. Но в данном случае просто происходит перезапись поля, то есть предыдущее изменение в связи с особенностью операции просто остается незамеченным.