Cel badania -> izolacja: *READ_ COMMITED*, cecha: *Niezatwierdzony odczyt*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Create_Account';
2 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=3000 WHERE ACCOUNT_NUMBER=1110002;
3
4 ROLLBACK;
```

Transakcja 2

```
1
2 SET TRANSACTION READ ONLY NAME 'Balances';
3 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Otrzymane wyniki:

Przed rozpoczęciem testów pole o id=1110002 miało wartość 20 000. Transakcja 2, zwróciła wartość 20 000.

Wnioski

Baza danych przy domyślnym poziomie izolacji: READ_COMMITED jest odporna na niezatwierdzony odczyt.

Cel badania -> izolacja: *READ_ COMMITED*, cecha: *Niepowtarzalny odczyt*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
3 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=3000 WHERE ACCOUNT_NUMBER=1110002;
4 COMMIT;
5 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Transakcja 2

```
1 SET TRANSACTION READ WRITE NAME 'Read_Balace';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
3
4 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Otrzymane wyniki:

Przed rozpoczęciem testów pole o id=1110002 miało wartość 20 000. Transakcja 1 oraz 2, zwróciła wartość 20 000. Transakcja 1 dokonała updatu rekordu id=1110002. Po zatwierdzeniu zmian obie Transakcje zwróciły te same wartości 3000

Wnioski Baza przy poziomie izolacji Read_Commited nie jest odporna na niepowtarzalny odczyt.

Cel badania -> izolacja: *READ_ COMMITED*, cecha: *Fantomy*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT SUM(ACCOUNT_BALANCE) FROM ACCOUNTS;
3
4
5 UPDATE ACCOUNT SET ACCOUNT_BALANCE = ACCOUNT_BALANCE*1.5;
6 COMMIT;
```

Transakcja 2

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT SUM(ACCOUNT_BALANCE) FROM ACCOUNTS;
3 INSERT INTO ACCOUNTS (ACCOUNT_NUMBER, ACCOUNT_BALANCE) VALUES (SEQ_ACCOUNT_NUMBER.
      NEXTVAL, 10000);
4 COMMIT;
```

Otrzymane wyniki:

Zapytanie SELECT Transakcji 1 oraz 2, zwróciło wartość 33 000. Transakcja 2 wprowadziła nowy rekord do bazy danych. Po commitcie Transakcji 2, Transakcja 1 dokonała updatu wartości kolumny ACCOUNT_BALANCE powiększając wszystkie wartości o 50%. Łącznie z nowo dodanym. **Wnioski** Baza przy poziomie izolacji Read_Commited nie jest odporna na fantomy.

Cel badania -> izolacja: *SERIALIZABLE*, cecha: *Niezatwierdzony odczyt*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Create_Account';
2 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=3000 WHERE ACCOUNT_NUMBER=1110002;
3
4 ROLLBACK;
```

Transakcja 2

```
1
2 SET TRANSACTION READ ONLY NAME 'Balances';
3 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Otrzymane wyniki:

Przed rozpoczęciem testów pole o id=1110002 miało wartość 4500. Transakcja 2, zwróciła wartość 4500.

Wnioski

Baza danych przy izolacji: SERIALIZABLE jest odporna na niezatwierdzony odczyt.

Cel badania -> izolacja: *SERIALIZABLE*, cecha: *Niepowtarzalny odczyt*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
3 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=3000 WHERE ACCOUNT_NUMBER=1110002;
4 COMMIT;
5 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Transakcja 2

```
1 SET TRANSACTION READ WRITE NAME 'Read_Balace';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
3
4 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Otrzymane wyniki:

Przed rozpoczęciem testów pole o id=1110002 miało wartość 4500. Transakcja 1 oraz 2, zwróciła wartość 4500. Transakcja 1 dokonała updatu rekordu id=1110002. Po zatwierdzeniu zmian Transakcja 2 zwróciła wartość 4500, natomiast Transakcja 2 zwróciła wartośc 3000

Wnioski Baza przy poziomie izolacji SERIALIZABLE jest odporna na niepowtarzalny odczyt. Update transakcji nr 1 nie został zacommitowany, więc transakcja 2 zwróciła 2 razy tę samą wartość.

Cel badania -> izolacja: *SERIALIZABLE*, cecha: *Fantomy*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT SUM(ACCOUNT_BALANCE) FROM ACCOUNTS;
3
4
5 UPDATE ACCOUNT SET ACCOUNT_BALANCE = ACCOUNT_BALANCE*1.5;
6 COMMIT;
```

Transakcja 2

```
1 SET TRANSACTION READ WRITE NAME 'Update_Account';
2 SELECT SUM(ACCOUNT_BALANCE) FROM ACCOUNTS;
3 INSERT INTO ACCOUNTS (ACCOUNT_NUMBER, ACCOUNT_BALANCE) VALUES (SEQ_ACCOUNT_NUMBER.
      NEXTVAL, 10000);
4 COMMIT;
```

Otrzymane wyniki:

Zapytanie SELECT Transakcji 1 oraz 2, zwróciło wartość 63 000. Transakcja 2 wprowadziła nowy rekord do bazy danych. Po commitcie Transakcji 2, Transakcja 1 chciała dokonać updatu wartości kolumny ACCOUNT_BALANCE powiększając wszystkie wartości o 50%. Baza danych zwróciła error blokując UPDATE. **Wnioski** Baza danych jest odporna na fantomowy odczyt.

Cel badania -> izolacja: *READ_ COMMITED + FOR UPDATE*, cecha: *Niezatwierdzony odczyt*

Wykonane instrukcje:

Transakcja 1

```
1 SET TRANSACTION READ WRITE NAME 'trans1';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002 FOR UPDATE;
3 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=9000 WHERE ACCOUNT_NUMBER=1110002;
4 COMMIT;
```

Transakcja 2

```
1 SET TRANSACTION READ WRITE NAME 'trans2';
2 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002 FOR UPDATE;
3
4 UPDATE ACCOUNTS SET ACCOUNT_BALANCE=5000 WHERE ACCOUNT_NUMBER=1110002;
5 COMMIT;
6 SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER=1110002;
```

Otrzymane wyniki:

Transakcja 2 czekała na zakończenie transakcji 1. Transakcja 1 pierwszy odczyt danej był 3000, pierwszy odczyt transakcji 2 był 9000, kolejny SELECT zwrócił wartośc 5000. **Wnioski** Baza danych przy dodaniu do zapytania FOR UPDATE zachowuje się podobnie jak przy poziomie izolacji SERIALIZABLE, gdzie jedna transakcja czeka na wykonanie się poprzedzającej transakcji.