READ UNCOMMITTED

READ UNCOMMITTED nie jest implementowane przez SZBD Oracle. Problem dirty read nie występuję w tym silniku co pokaże poniższy przykład.

READ COMMITTED

READ COMMITTED ANOMALIA : NIEPOWTARZALNY ODCZYT

1	**1. SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;**	
2	SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME 'UPDATEACCOUNT' ;	
3		SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME 'READACCOUNT';
4	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 2000 WHERE ACCOUNT_NUMBER = 11110005;	
5		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;
6	commit;	
7		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;

Wynik 1 : 1000
Wynik 2 : 1000 // NA POZIOMIE READ UNCOMMITTED WYNIK BYLBY 2000
Wynik 3 : 2000 // Tutaj widać zmian

READ COMMITTED ANOMALIA : FANTOMY

1	SELECT * FROM ACCOUNTS	
2	SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME ‘UPDATE_ACCOUNT’ ;	
3		SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME ‘INSERT_ACCOUNT’;
4		INSERT INTO ACCOUNTS (ACCOUNT_NUMBER, ACCOUNT_BALANCE) VALUES (SEQ_ACCOUNT_NUMBER.NEXTVAL, 10);
5		commit;
6	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = ACCOUNT_BALANCE * 1.1;	
7	commit;	
8	SELECT * FROM ACCOUNTS;	

Wynik 1:

```
ACCOUNT_NUMBER ACCOUNT_BALANCE
-------------- ---------------
      11110002            8000
      11110003           12000
      11110004          123000
      11110005            2000
      11110006            9000
      11110007            4400
      11110008           12000
      11110010           12203
      11110011          231220
```

Wynik 8:

```
ACCOUNT_NUMBER ACCOUNT_BALANCE
-------------- ---------------
      11110002            8800
      11110003           13200
      11110004          135300
      11110005            2200
      11110006            9900
      11110007            4840
      11110008           13200
      11110010         13423,3
      11110011          254342
```

11110012 11

Widać, że na krotce o największym ID : 11110012 został wykonany update, chociaż transakcja czerwona nie wiedziała ,że będzie updatować tę własnie krotkę.

READ COMMITTED ANOMALIA : LOST UPDATE (DIRTY WRITE)

1	SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME 'UPDATE_ACCOUNT' ;	
2		SET TRANSACTION ISOLATION LEVEL READ COMMITTED NAME 'INSERT_ACCOUNT';
3	SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110012 ;	
4		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110012 ;
5	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 20 WHERE ACCOUNT_NUMBER = 11110012;	
6	Commit ;	
7		UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 30 WHERE ACCOUNT_NUMBER = 11110012;
8		commit

Wynik 3: 11
Wynik 4: 11

Transakcja zielona update'uje stan konta nie wiedząc o jego aktualnym stanie. Po commicie w operacji 6, stan konta wynosi : 20 , a transakcja zielona odczytała wcześniej w operacji 4 wartość 11. Następuje update w operacji 7 , w tym miejscu tracimy update'a który został wykonany przez operacje czerwoną.

REPEATABLE READ

Oracle explicite wspiera tylko standardy READ COMMITTED oraz SERIALIZABLE.
Dodatkowo definiuje się poziom izolacji : READ ONLY

READ ONLY

Transakcje które działają na poziomie izolacji READ ONLY widzą tylko te zmiany które zostały zacommitowane przed rozpoczęciem transakcji. Jest to ekwiwalent poziomy REPEATABLE READ oraz SERIALIZABLE bez możliwości odczytu (źrodło : https://bit.ly/2jTrebT)

READ ONLY ANOMALIA : NIEPOWTARZALNY ODCZYT

1	**1. SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;**	
2	SET TRANSACTION READ ONLY NAME ‘READRED’ ;	
3		SET TRANSACTION READ ONLY NAME ‘READGREEN';
4	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 2000 WHERE ACCOUNT_NUMBER = 11110005;	
5		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;
6	commit;	
7		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;

Wynik 1: 2200
Wynik 5: 2200;
Wynik 6: 2200;

Transakcja czerwona nie zacommittowała swoich zmian przed rozpoczęciem transakcji zielonej stąd wyniki operacji 5 i 7 są takie same.

READ ONLY ANOMALIA : FANTOMY

1	SELECT * FROM ACCOUNTS	
2	SET TRANSACTION READ ONLY NAME ‘UPDATE_ACCOUNT’ ;	
3		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME ‘ser';
4		INSERT INTO ACCOUNTS (ACCOUNT_NUMBER, ACCOUNT_BALANCE) VALUES (SEQ_ACCOUNT_NUMBER.NEXTVAL, 666);
5		commit;
6	select * from accounts;	
7	commit;	
8	SELECT * FROM ACCOUNTS;	

Wynik 1 :

ACCOUNT_NUMBER	ACCOUNT_BALANCE
11110002	8800
11110003	13200
11110004	135300
11110005	2200
11110006	9900
11110007	4840
11110008	13200
11110010	13423,3
11110011	254342
11110012	11

Wynik 6:

ACCOUNT_NUMBER	ACCOUNT_BALANCE
11110002	8800
11110003	13200
11110004	135300
11110005	2200
11110006	9900
11110007	4840
11110008	13200
11110010	13423,3
11110011	254342
11110012	11

Wynik 8:

ACCOUNT_NUMBER	ACCOUNT_BALANCE
11110002	8800
11110003	13200
11110004	135300
11110005	2200
11110006	9900
11110007	4840
11110008	13200
11110010	13423,3
11110011	254342
11110012	11
11110013	666

Tu widać również, że nie zachodzi fantomowy odczyt. Mimo iż transakcja czerwona Zacommitowała dodanie krotki to transakcja zielona nie widziała ich podczas wykonywania swoich operacji.

SERIALIZABLE

Serializable jest najbardziej najgłębszym poziomem izolacji implementowanym przez system zarządzania baz jeśli chodzi o standard SQL. Według standardu nie powinna zachodzić żadna z anomalii zdefiniowanych w standardzie (dirty read, phantom read, repeatable read). Dirty read nie jest możliwe w szbd Oracle stąd wystarczy sprawdzić phantom read, repatable read oraz dirty read

SERIALIZABLE ANOMALIA : NIEPOWTARZALNY ODCZYT

1	**SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;**	
2	SET TRANSACTION ISOLATION LEVEL SERIALiZABLE NAME 'UPDATEACCOUNT' ;	
3		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME 'READACCOUNT';
4	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 3000 WHERE ACCOUNT_NUMBER = 11110005;	
5		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;
6	commit;	
7		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110005;

Wynik 1: 2200
Wynik 5: 2200
Wynik 7: 2200

Transakcja czerwona nie była niezacommitowana przed ropoczęciem transakcji zielonej , stąd Zawsze ten sam wynik, jeśli zacommitujemy zieloną to okaże się i rozpoczneimy taką samą transakcje to wówczas zobaczymy wynik 3000

SERIALIZABLE ANOMALIA : FANTOMY

1	SELECT * FROM ACCOUNTS	
2	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME ‘UPDATE_ACCOUNT’ ;	
3		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME ‘INSERT_ACCOUNT’;
4		INSERT INTO ACCOUNTS (ACCOUNT_NUMBER, ACCOUNT_BALANCE) VALUES (SEQ_ACCOUNT_NUMBER.NEXTVAL, 10000);
5		commit;
6	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = ACCOUNT_BALANCE * 1.1;	

Wynik 1:

ACCOUNT_NUMBER	ACCOUNT_BALANCE
11110002	8800
11110003	13200
11110004	135300
11110005	2200
11110006	9900
11110007	4840
11110008	13200
11110010	13423,3
11110011	254342
11110012	11
11110013	666

Wynik 6:
Error starting at line : 3 in command -
UPDATE ACCOUNTS SET ACCOUNT_BALANCE = ACCOUNT_BALANCE * 1.1
Error report -
ORA-08177: can't serialize access for this transaction

Transkacja zielona zablokowała transakcje czerwoną na update. Jeśli teraz zacommitujemy transakcje czerwoną, ponownie otworzymy taką samą transakcje to update będzie możliwy. Poziom serializable w zasadzie działa tak jakby użytkownik tego oczekiwał, ale pokazuje również jak bardzo nieoptymalny jest to poziom gdyż transakcje muszą czekać na zatwierdzenie zmian przez poprzedzające transakcje jeżeli operują na tych samych danych.

SERIALIZABLE ANOMALIA : LOST UPDATES (DIRTY READ)

1	SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME ‘UPDATE_ACCOUNT’ ;	
2		SET TRANSACTION ISOLATION LEVEL SERIALIZABLE NAME ‘UPT_ACCOUNT’;
3	SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110012 ;	
4		SELECT ACCOUNT_BALANCE FROM ACCOUNTS WHERE ACCOUNT_NUMBER = 11110012 ;
5	UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 20 WHERE ACCOUNT_NUMBER = 11110012;	
6	Commit ;	
7		UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 30 WHERE ACCOUNT_NUMBER = 11110012;
8		commit

Wynik 3: 12,1
Wynik 4: 12,1

Operacja 5 wykonuje się poprawnie, problem natomiast występuję w operacji 7 :
UPDATE ACCOUNTS SET ACCOUNT_BALANCE = 30 WHERE ACCOUNT_NUMBER = 11110012
Error report -
ORA-08177: can't serialize access for this transaction

Wynika z tego, że poziom Serializable zakłada blokadę na danej która jest update’owana