Travail Pratique 1

Les transactions sous Oracle 10g

La norme SQL92 propose quatre niveaux d'isolation (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ et SERIALIZABLE). Oracle implémente uniquement deux : READ COMMITTED et SERIALIZABLE.

Le niveau d'isolation d'une transaction est spécifié comme suit : SET TRANSACTION ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED };

Le niveau d'isolation peut être défini pour la durée d'une session active : ALTER SESSION SET ISOLATION_LEVEL { SERIALIZABLE | READ COMMITTED };

Sous Oracle, une instruction DML (SELECT, INSERT, UPDATE, DELETE), DDL (CREATE, ALTER, DROP) ou DCL (GRANT, REVOKE, COMMIT, ROLLBACK) démarre une transaction qui se termine par un COMMIT si tout s'est bien passé.

Dans Oracle on peut définir le mode d'une transaction (lecture seule ou lecture écriture) par l'instruction : SET TRANSACTION { READ ONLY | READ WRITE }.

Par défaut, le mode d'une transaction est READ WRITE et son niveau d'isolation est READ COMMITTED.

Voir la documentation Oracle https://docs.oracle.com/cd/E25054_01/server.1111/e25789/consist.htm

Dans la suite vous allez réaliser des expériences qui vous permettront de comprendre et d'analyser la manière comment Oracle gère la concurrence.

Initialisation de l'environnement

Chaque binôme utilisera 2 comptes oracle (chaque étudiant utilisera le compte qui lui a été attribué). Le TP est à faire avec une session par compte.

Ne pas ajouter d'autres commandes pendant les expériences. Entre chaque expérience vous pouvez faire un SELECT * FROM T suivi d'un COMMIT.

Session 1	Session 2
Drop table T cascade constraints; create table T (A integer, B integer); grant select, update, insert, delete on T to compte2; insert into T values (0,0); insert into T values (2,3); insert into T values (0,1); commit;	Drop synonym T ; create synonym T for compte1.T;

Expérience 1

Session 1	Session 2
select * from T; select * from T; select * from T; commit;	update T set A=A+1; commit ;

Question. Quel problème est mis en évidence ?

Expérience 2

Session 1	Session 2

update T set A=A+1;
select * from T;
 	
commit ;
	
	insert into T values(3,7);
	select * from T;
	delete from T where A=2;
	delete from T where A=1;
	
	commit ;

Qu'est-ce que vous pouvez déduire de cette exécution ? Est-ce qu'elle est sérialisable ?

A la fin de l'expérience 2 :

Select * from T ;
A | B
---
3 | 7
2 | 0
4 | 3
2 | 1
Commit ;

Expérience 3

Session 1	Session 2

update T set A=3 where A=2;
 	
select * from T;
 	
update T set B=2 where B=3 ;
 	
commit ;
	
	update T set B=2 where B=3;
	select * from T;
	
	update T set A=3 where A=2;
	
	commit ;

A quel niveau de granularité sont posés les verrous ? Comment se terminent ces transactions ? Pourquoi ?

A la fin de l'expérience 3 :

Select * from T ;
A | B
---
3 | 7
3 | 0
4 | 2
3 | 1
Commit ;

Expérience 4

Session 1	Session 2

Select * from T where B=7 for update ;
 	
update T set B=6 where B=7 ;
commit ;
	
	update T set B=10 where B=0;
	select * from T;
	update T set A=5 where A=3;
	
	commit ;
	select * from T;
	commit ;

Quelle a été l'utilité de la clause select for update ?

A la fin de l'expérience 4 :

Select * from T ;
A | B
-----
5 | 6
5 | 10
4 | 2
5 | 1
Commit ;

Expérience 5

Session 1	Session 2
Set transaction read only; Select * from T ; Select * from T ; Select * from T ; commit ; Select * from T ; commit ;	 select * from T; update T set A=A+1; commit ; select * from T; update T set A=A+1; commit ;

Qu'est-ce que vous avez remarqué ?

A la fin de l'expérience 5 :

Select * from T ;
A | B
-----

7 | 6
7 | 10
6 | 2
7 | 1
Commit ;

Expérience 6

Session 1	Session 2
lock table T in share mode; commit; lock table T in exclusive mode; commit;	 select * from T for update; update T set A=A+1; commit ; select * from T ; select * from T for update; commit ;

Expliquez les résultats obtenus.

A la fin de l'expérience 6 :

Select * from T ;
A | B
---
8 | 6
8 | 10
7 | 2
8 | 1
Commit ;

Expérience 7

Session 1	Session 2
alter session set isolation_level=serializable; select * from T ; select * from T ; select * from T ; commit; select * from T ; commit ;	alter session set isolation_level=serializable; update T set A=A+1; commit ; insert into T values (0,0) ; commit ; select * from T ; commit ;

Expliquez les résultats obtenus.

Expérience 8

Session 1	Session 2

select * from T ;
 	
update T set A=A+1 where A=8;
 	
commit ;
	
	select * from T;
	
	delete from T where A=9;
	delete from T where B=2;

Expliquez le problème rencontré.

Expérience 8 bis

Session 1	Session 2
Alter table T add constraint PK primary key (B) ; Grant references on T to compte2 ; Select * from T ; Delete from T where A=0 ; Commit ;	 Create table S(B number references T, C number) ; Select * from T ; Insert into S values(0,1) ; commit ;

Qu'est-ce que vous avez remarqué dans cette exécution ? Comment résoudre le problème rencontré ?

Expérience 9

Maintenant, c'est à vous de jouer. Il y a d'autres caractéristiques transactionnelles que nous n'avons pas analysé, par exemple, les variations de verrouillage d'une table (lock table) à savoir « share mode » et « exclusive mode » avec « no wait », ou « in row share mode », « in row exclusive mode ». Vous pouvez également exécuter les premières expériences du TP ave le mode serialisable.

Expérience 10

Pour terminer, dans cette expérience vous allez constater l'utilité des clauses ROLLBACK et SAVEPOINT

Session 1	Session 2
Insert into T values (0,0) ; Insert into T values (0,0) ; Rollback ; Select * from T ; Insert into T values (0,0) ; Insert into T values (0,0) ; Commit ; Rollback ;	update T set B=B+1; savepoint A ; insert into T values(10,10) ; savepoint B ; select * from T ; rollback to savepoint B select * from T ;

Expliquez les résultats obtenus.