TD N° 5
Gestion d'une chaîne d'hôtels (suite)

Exercice 1 - CONCURRENCE ET CONTRAINTE D'INCLUSION

Soit la contrainte d'inclusion :

(C1) Il est impossible qu'une réservation porte sur un type de chambre et sur un hôtel qui ne possède aucune chambre de ce type.

Qui est implantée par les triggers suivants :

```
create or replace trigger C1_Ajout_Reservations
after insert or update of NumTy, NumHo on Reservations
for each row
declare
   N binary_integer;
begin
   select 1 into N from Chambres
   where NumHo = :New.NumHo and NumTy = :New.NumTy;
exception
   when too_many_rows then null;
   when no_data_found then
      raise_application_error(-20000,'PB C1');
end;

create or replace trigger C1_Supp_Chambres
after delete or update of NumHo, NumTy on Chambres
declare
   N binary_integer;
begin
   select 1 into N from Reservations
   where (NumHo, NumTy) not in
      (select NumHo, NumTy from Chambres);
   raise too_many_rows;
exception
   when no_data_found then null;
   when too_many_rows then
      raise_application_error(-20000,'PB C1');
end;
```

1) Analyser l'interférence liée à plusieurs transactions qui s'exécutent simultanément.

2) Proposer une gestion de la concurrence avec des verrous de table. Et faire une évaluation de degré de blocage des transactions (i.e. du degré de concurrence).

3) Proposer une gestion de la concurrence avec des verrous de ligne. Et faire une évaluation de degré de blocage des transactions.

Exercice 2 - CONCURRENCE ET CONTRAINTE D'UNICITÉ

Soit la contrainte d'unicité :

(C2) Il est impossible qu'une chambre soit occupée 2 fois en même temps par deux clients différents ou par le même client.

Qui est implantée par le trigger suivant :

```
create or replace trigger C2
after insert or update on Occupations
declare
   N binary_integer;
begin
   select 1 into N from Occupations O1, Occupations O2
   where O1.NumCh = O2.NumCh and O1.NumHo = O2.NumHo
   and O1.DateA <> O2.DateA
   and (O1.DateA, O1.DateD) overlaps (O2.DateA, O2.DateD);
exception
   when no_data_found then null;
   when too_many_rows then
     raise_application_error(-20000,'PB C2');
end;
```

1) Analyser l'interférence liée à plusieurs transactions qui s'exécutent simultanément.

2) Proposer une gestion de la concurrence avec des verrous. Et faire une évaluation de degré de blocage des transactions.

3) En utilisant une table temporaire, proposer une nouvelle implantation qui améliore la gestion de la concurrence. Et faire une évaluation de degré de blocage des transactions.

Gestion de la concurrence en Oracle

1 - Gestion par défaut :

- Les lectures ne posent aucun verrou, mais les lectures sales sont impossibles grâce au mécanisme de lecture consistante.
- Verrouillage automatique en mode exclusif de chaque ligne concernée par une instruction de mise à jour (**`INSERT, DELETE ou UPDATE`**) et en mode **`ROW EXCLUSIVE`** de la table.
- Le niveau **READ COMMITED** de la norme SQL92 est assuré : les écritures et les lectures sales sont impossibles.

2 - Instructions SQL pour gérer la concurrence :

- Isolation par snapshot au niveau de la transaction avec :

```
SET TRANSACTION READ ONLY
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
```

Les lectures sont réalisées sur l'état de la base avant le début de la transaction.

- Verrouillage explicite de lignes en mode exclusif :

```
SELECT ... FOR UPDATE [OF col,...,col]
```

- Verrouillage explicite d'une table :

```
LOCK TABLE table IN mode MODE [NOWAIT]
```

Mode de verrouillage :

```
EXCLUSIVE, SHARE, ROW SHARE, SHARE UPDATE
ROW EXCLUSIVE, SHARE ROW EXCLUSIVE
```

- Compatibilités des verrous :

compatible acquis	Exclusive	Share	Row Share Share Update Select for update	Row Ex. Insert Delete, Update	Sh. Row Ex
Exclusive (X)	**Non**	**Non**	**Non**	**Non**	**Non**
Share (S)	**Non**	**Oui**	**Oui**	**Non**	**Non**
Row Share (RS) Share Update (RS) Select for update	**Non**	**Oui**	**Oui**	**Oui**	**Oui**
Row Ex. (RX) Insert, Delete, Update	**Non**	**Non**	**Oui**	**Oui**	**Non**
Sh. Row Ex (SRX)	**Non**	**Non**	**Oui**	**Non**	**Non**