Département Informatique et Statistique, Faculté de SEG, Université Lumière Lyon 2
M1 Informatique – Année 2010-2011

Langages de requêtes – TD n° 6

J. Darmont (*http://eric.univ-lyon2.fr/~jdarmont/*), 05/01/11

Exercice 1 : Sécurité/intégrité des données – Transactions

1. Se connecter au même compte (le vôtre !) depuis deux connexions différentes (une sous *SQL Developer* et une sous *APEX*, deux connexions différentes dans *SQL Developer*…), simultanément. Sous *APEX*, décocher l'option « Autocommit ».
2. Avec l'une des deux connexions, créer la table TRANSACTION (TID, LIB) où la clé primaire TID est un nombre à deux chiffres et LIB une chaîne de 20 caractères maximum. Vérifier en consultant la vue système TAB depuis les deux connexions que la table est bien créée.
3. Dans chacune des deux connexions, insérer un n-uplet différent dans la table TRANSACTION. Consulter la table depuis chaque connexion. Que constate-t-on ?
4. Annuler l'une des insertions précédentes à l'aide de la commande `ROLLBACK` (F12 sous *SQL Developer*). Consulter la table depuis chaque connexion. Que se passe-t-il ?
5. Recommencer les insertions d'un n-uplet différent depuis chaque connexion, puis valider l'une des insertions à l'aide de la commande `COMMIT` (F11 sous *SQL Developer*). Consulter la table depuis chaque connexion. Valider dans l'autre connexion. Consulter la table depuis chaque connexion. Conclusion ?
6. Dans chacune des deux connexions, insérer un n-uplet identique, cette fois, dans la table TRANSACTION. Que se passe-t-il ?
7. Annuler l'insertion dans la première connexion. Que se produit-il dans la deuxième ?
8. Essayer de ré-insérer le même n-uplet depuis la première connexion. Que se passe-t-il ?
9. Valider l'insertion dans la deuxième connexion. Que se produit-il dans la première ?
10. Insérer un nouveau n-uplet depuis une des deux connexions et quitter le client correspondant ou interrompre la connexion. Consulter la table depuis l'autre connexion. Conclusion ?
11. Insérer un nouveau n-uplet depuis la connexion restante, puis vous déconnecter. Ouvrir une nouvelle connexion. La dernière transaction a-t-elle été validée ?
12. Insérer deux nouveaux n-uplets, puis émettre un `ROLLBACK`. Que sont devenus les deux n-uplets insérés ?
13. Conclusion ? Comment est validée une transaction ?

Exercice 2 : Sécurité – Privilèges

Travailler par groupe de deux ordinateurs : vous et vos voisins (se coordonner pour avancer dans les questions).

1. Donner à vos voisins le droit de consulter votre table TRANSACTION. Consulter la-leur.
2. Insérer un n-uplet dans la table TRANSACTION de vos voisins. Que se passe-t-il ?
3. Donner à vos voisins le droit d'insertion dans votre table TRANSACTION. Insérer un n-uplet dans la-leur.

4. Afficher la liste des privilèges que vous avez accordés à l'aide de la vue système USER_TAB_PRIVS(GRANTEE, TABLE_NAME, GRANTOR, PRIVILEGE…).

Travailler par groupe de trois ordinateurs : vous et vos deux voisins (se coordonner pour avancer dans les questions).

1. Donner à vos premiers voisins le droit de consulter votre table TRANSACTION, ainsi que le droit de transmettre ce privilège.
2. Transmettre à vos premiers voisins le privilège transmis par les seconds. Consulter la table TRANSACTION de tous vos voisins. Réafficher la liste des privilèges que vous avez accordés.
3. Supprimer à vos premiers voisins le droit de consulter votre table TRANSACTION. Vos seconds voisins peuvent-ils toujours consulter votre table ?
4. Supprimer à tous les autres utilisateurs le droit de consulter votre table TRANSACTION. Vos seconds voisins peuvent-ils toujours consulter votre table ?

NB : Pour réaliser la partie à trois, utiliser la technique de la « permutation circulaire ».

Exercice 3 complémentaire : Révision vues

Soit le schéma relationnel suivant, représentant une table de pièces détachées, leurs fournisseurs et les associations entre ces deux tables (cotations et commandes).

PARTS (partno, description, qonhand, qonorder)
SUPPLIERS (suppno, name, address)
QUOTATIONS (suppno#, partno#, price, responsetime)
ORDERS (suppno#, partno#, quantity, orderdate)

I. Recopier ces quatre tables depuis le compte de l'utilisateur `DARMONT`.

II. Créer les vues suivantes et vérifier grâce aux requêtes fournies que le résultat obtenu est correct.

1. *fast_quotes :* contient toutes les cotations dont le temps de réponse est inférieur à dix jours.

```
select * from fast_quotes;

SUP PART     PRICE RESPONSETIME
--- ---- --------- ------------
S54 P222        75            7
```

2. *low_prices :* fournit le prix minimum coté pour chaque pièce.

```
select * from low_prices;

PART  MINPRICE
---- ---------
P207       950
P209      1250
P221        35
P222        15
P231        20
P232         5
P285      3250
P295      1900
```

3. *old_orders :* contient les commandes qui datent de plus de deux ans en précisant le nom du fournisseur et la description de la pièce.

```
select * from old_orders;

PART  QUANTITY ORDERDAT NAME                 DESCRIPTION
---- --------- -------- -------------------- -----------
P207        20 15/06/97 Parts Are We         Gear
P209        10 20/06/97 ABC Parts Company    Cam
P221       200 01/07/97 Partco Inc.          Big Bolt
P231       200 01/07/97 Partco Inc.          Big Nut
P295        25 28/06/97 Quality Parts        Belt
```

4. *all_parts :* contient les pièces détachées, leur description et le total de la quantité disponible et de la quantité en commande.

```
select * from all_parts;

PART DESCRIPTION        TOTALQ
---- --------------- ---------
P207 Gear                 95
P209 Cam                  10
P221 Big Bolt            850
P222 Small Bolt         1250
P231 Big Nut             200
P232 Small Nut          1100
P250 Big Gear              8
P285 WheelBelt           350
P295 Belt                 25
```

5. À travers la vue *all_parts*, changer la description de la pièce P231 à « Bigger Nut ».

6. À travers la vue *all_parts*, affecter à l'attribut TOTALQ de la pièce P231 la valeur 900.

7. Valider la transaction. Insérer le n-uplet suivant dans la vue *fast_quotes* :
('S51', 'P221', 3000, 20)

8. Annuler la transaction précédente. Créer une vue *fast_quotes2* identique à *fast_quotes*, mais en rajoutant la clause « WITH CHECK OPTION » à la fin de l'instruction SQL. Essayer de ré-insérer le n-uplet précédent dans *fast_quotes2*.

Exercice 4 complémentaire : Révision requêtes

Soit le schéma relationnel suivant, représentant le fournisseur d'un produit pour un projet en une certaine quantité.

FOUR (CodeF, Ville)
PROD (CodeP, Poids, Couleur, Parent#) NB : Parent référence CodeP
PROJ (CodeJ, Ville)
FJP (CodeF#, CodeP#, CodeJ#, Qte)

I. Recopier ces quatre tables depuis le compte de l'utilisateur `DARMONT`.

II. Formuler en SQL les requêtes suivantes. Vérifier à chaque fois que le résultat obtenu est correct et sans doublon.

1. Propriétés des projets.
2. Propriétés des projets de Londres.
3. Produit(s) de poids minimal.
4. Fournisseurs du projet J1.
5. Produits fournis par le fournisseur F1 au projet J6.
6. Couleurs des produits fournis par le fournisseur F1.
7. Fournisseurs des projets J1 et J2.
8. Projets utilisant un produit rouge.
9. Produits fournis aux projets de Londres.
10. Fournisseurs d'un produit rouge aux projets de Londres ou de Paris.
11. Produits fournis à Paris par un fournisseur de Paris.
12. Projets dont au moins un fournisseur n'est pas dans la ville du projet.
13. Projets qui n'ont ni produit rouge ni fournisseur à Londres.
14. Projets utilisant au moins un produit fourni par le fournisseur F2.
15. Fournisseurs d'au moins un produit fourni par au moins un fournisseur d'au moins un produit rouge.
16. Couples (ville d'un fournisseur, ville d'un projet du même fournisseur).
17. Triplets (ville, code produit, ville) tels qu'un fournisseur dans la première ville fournit un produit donné à un projet dans la seconde ville.
18. Projets fournis entièrement par le fournisseur F1.
19. Projets n'utilisant que des produits fournis par le fournisseur F2.
20. Fournisseurs qui fournissent tous les produits.
21. Produits fournis à tous les projets de Bruxelles.
22. Nombre de projets fournis par le fournisseur F3.
23. Quantité totale de produit P3 fournie par le fournisseur F2.
24. Code produit, code projet et somme des quantités pour chaque produit fourni à un projet.
25. Niveau hiérarchique et code produit de tous les produits.
26. Niveau hiérarchique et code produit de tous les produits sauf P2.
27. Niveau hiérarchique et code produit de tous les produits sauf P2 et les produits qui composent (« qui sont fils de ») P2.
28. Nombre de fournisseurs et de projets distincts par niveau hiérarchique de produits.

Correction Exercice 1

Connexion 1	Connexion 2
create table transaction(tid number(2) primary key, lib varchar(20));	
select count(*) from tab where tname='TRANSACTION';	select count(*) from tab where tname='TRANSACTION';
insert into transaction values(10, 'T1');	insert into transaction values(20, 'T2');
select * from transaction;	select * from transaction;
rollback;	
select * from transaction;	select * from transaction;
insert into transaction values(30, 'T3');	insert into transaction values(40, 'T4');
commit;	
select * from transaction;	select * from transaction;
insert into transaction values(50, 'T5');	insert into transaction values(50, 'T5');
rollback;	
insert into transaction values(50, 'T5');	
	commit;
insert into transaction values(60, 'T6');	
	select * from transaction;
	insert into transaction values(70, 'T7');

```
select * from transaction;
insert into transaction values(80, 'T8');
insert into transaction values(90, 'T9');
rollback;
select * from transaction;
```

Correction Exercice 2

```
grant select on transaction to voisins1;
select * from voisins1.transaction;
insert into voisins1.transaction values (80, 'T8');
grant insert on transaction to voisins1;
insert into voisins1.transaction values (80, 'T8');
select grantee, table_name, grantor, privilege from user_tab_privs;

grant select on transaction to voisins1 with grant option;
grant select on voisins2.transaction to voisins1;
select grantee, table_name, grantor, privilege from user_tab_privs;
revoke select on transaction from voisins1;
revoke select on transaction from public;
```

Correction Exercice 3

```
-- 1
create view fast_quotes as
select * from quotations where responsetime < 10;

-- 2
create view low_prices as
select partno, min(price) as minprice from quotations group by partno;

-- 3
create view old_orders as
select o.partno, quantity, orderdate, name, description
from orders o, suppliers s, parts p
where o.partno=p.partno and o.suppno=s.suppno
and months_between(sysdate, orderdate) > 24;
```



```
-- 4
create view all_parts as
select partno, description, qonhand + qonorder as totalq from parts;

-- 5
update all_parts set description='Bigger Nut' where partno='P231';

-- 6
update all_parts set totalq=900 where partno='P231';
-- ORA-01733: les colonnes virtuelles ne sont pas autorisées ici
-- (champ calculé : impossibilité de mettre à jour)

--7
commit;
insert into fast_quotes values('S51','P221',3000,20);

-- 8
rollback;
create view fast_quotes2 as
select * from quotations where responsetime < 10
with check option;
insert into fast_quotes2 values('S51','P221',3000,20);
-- Ne fonctionne plus
```

Correction Exercice 4

```
--  1
select * from proj;

--  2
select * from proj where ville='Londres';

--  3
select codep from prod where poids=(select min(poids) from prod);

--  4
select codef from fjp where codej='J1';

--  5
select codep from fjp where codef='F1' and codej='J6';

--  6
select distinct couleur from fjp f, prod p where f.codep=p.codep and codef='F1';

--  7
select distinct codef from fjp where codej='J1' or codej='J2';

--  8
select distinct codej from fjp f, prod p
where f.codep=p.codep and couleur='Rouge';

--  9
select distinct codep from fjp f, proj j
where f.codej=j.codej and ville='Londres';

-- 10
select distinct codef from fjp f, prod p, proj j
where f.codep=p.codep and f.codej=j.codej
and (ville='Londres' or ville='Paris');
```

```
-- 11
select distinct codep from fjp f, four r, proj j
where f.codef=r.codef and f.codej=j.codej
and r.ville='Paris' and j.ville='Paris';

-- 12
select distinct j.codej from proj j, four r, fjp f
where r.codef=f.codef and f.codej=j.codej and r.ville<>j.ville;

-- 13
select distinct j.codej from proj j, four r, prod p, fjp f
where r.codef=f.codef and f.codej=j.codej and p.codep=f.codep
and (couleur<>'Rouge' or r.ville<>'Londres');

-- 14
select distinct codej from fjp where codep in
 (select codep from fjp where codef='F2');

-- 15
select distinct codef from fjp where codep in
 (select codep from fjp where codef in
  (select codef from prod p, fjp f where p.codep=f.codep and couleur='Rouge'));

-- 16
select distinct r.ville, j.ville from four r, fjp f, proj j
where r.codef=f.codef and f.codej=j.codej;

-- 17
select distinct r.ville, p.codep, j.ville from four r, proj j, prod p, fjp f
where r.codef=f.codef and f.codej=j.codej and p.codep=f.codep;

-- 18
select codej from proj j where not exists
 (select * from fjp where codef<>'F1' and fjp.codej=j.codej);

-- 19
select codej from proj j where not exists (
     select * from fjp where fjp.codej=j.codej and codep not in (
          select distinct codep from fjp where codef='F2'));

-- 20
select codef from four r where not exists
 (select * from prod p where not exists
  (select * from fjp f where f.codef=r.codef and f.codep=p.codep));

-- 21
select codep from prod p where not exists
 (select * from proj j where ville='Bruxelles' and not exists
  (select * from fjp f where f.codep=p.codep and f.codej=j.codej));

-- 22
select count(codep) from fjp where codef='F3';

-- 23
select sum(qte) from fjp where codef='F2' and codep='P3';

-- 24
select codep, codej, sum(qte) from fjp group by codep, codej;

-- 25
select level, codep from prod
connect by parent = prior codep
start with parent is null;
```



```
-- 26
select level, codep from prod
where codep <> 'P2'
connect by parent = prior codep
start with parent is null;

-- 27
select level, codep from prod
connect by parent = prior codep and codep <> 'P2'
start with parent is null;

-- 28
select level, count(distinct codef), count(distinct codej)
from prod, fjp
where prod.codep = fjp.codep
connect by parent = prior prod.codep
start with parent is null
group by level;
```