|  |  |
| --- | --- |
| image001 | **Année : 2019/2020** |
|  | **TP-4** |
| **EFREI - L3A** | **Bases de Données Oracle** |
|  |  |
| **A. Lahlou** | [lahloukarim@free.fr](mailto:lahloukarim@free.fr) |

**TP-4**

**Cas Notes – SQL Avancé**

**Gestion des vues, des droits et transactions**

***Objectif :***

***SQL/SQL Avancé : Vues, Droits, transactions et concurrence d’accès***

1. Création et manipulation de vues mono-table ; multi-tables et contenant des groupements

Commandes CREATE VIEW ; RENAME, …

2. Gestion des droits d’accès

Commandes GRANT ; REVOKE.

3. Gestion des transactions

Commandes COMMIT ; ROLLBACK.

**I – Description de la base de données :**

Considérons la base de données de gestion de notes d’étudiants contenant les tables suivantes (script fourni en annexe).

Etudiant (**NumEtu**, Nom, Prenom, DateNaiss, Rue, CP, Ville)

Matiere (**CodeMat**, Libelle, Coef)

Epreuve (**numepreuve**, DateEpreuve, Lieu, #CodeMat)

Notation (**#NumEtu, #NumEpreuve,** Note)

Mettre le script fourni en Annexe1 dans un fichier .sql sous gedit ou notepad++ ou sqldevelopper ou datagrip ou autre et l’exécuter sous Oracle

**II – Interrogation :**

1. Vérifier la création de vos 4 tables et leur instanciation (Nombres de lignes par table par exemple).

2. Liste des notes supérieures ou égales à 10.

3. Liste des épreuves dont la date se situe entre le 1er janvier et le 30 juin 2004.

4. Nombre total d'épreuves.

5. Nombre de notes indéterminées (NULL).

6. Liste des notes en précisant pour chacune le nom et le prénom de l'étudiant qui l'a obtenue.

7. Moyennes des notes de chaque étudiant (indiquer le nom et le prénom), classées de la meilleure à la moins bonne (les notes à NULL sont à considérer comme 0).

8. Liste des étudiants qui ont obtenu la plus petite moyenne en Informatique

9. Liste des étudiants qui ont obtenu la meilleure moyenne

10. Liste des étudiants qui ont participé à toutes les épreuves

**III – Gestion des vues**

1. Créer la vue renfermant tous les étudiants ayant eu des épreuves en Informatique ainsi que les notes obtenues.

2. Donner la moyenne et le nombre d’épreuves en informatique de chaque étudiant ayant passé au moins une épreuve dans cette matière.

4. Donner les noms d’étudiants ainsi que leur moyenne ayant eu une moyenne supérieure ou égale à 10 en Informatique et classés par ordre de mérite.

5. Donner les noms d’étudiants qui n’ont passé aucune épreuve en Informatique.

**IV – Gestion de la confidentialité (Vues et Droits d’accès)**

1. Créer la vue EtudLyonnais renfermant tous les étudiants lyonnais (La vue obtenue contient tous les champs de la table Etudiant).

2. Donner la moyenne en informatique des étudiants lyonnais

3. L’étudiant Dupond a quitté l’établissement. Est-il possible de le supprimer depuis la vue EtudLyonnais ? Vérifier ?

4. L’étudiant Durand déménage lors de son stage à 1, Rue de Lyon Paris 75002. Est-il possible de mettre à jour ses informations depuis la vue EtudLyonnais ? Vérifier ?

5. Est-il possible d’ajouter l’étudiant (700, Gates, Bill, 1980-09-1, Rue de Paris, 69005, Lyon) depuis la vue EtudLyonnais ?  Vérifier ?

6. Donner la vue MoyLyonnais contenant la moyenne par matière des étudiants lyonnais.

7. Est-il possible de mettre à jour la moyenne en informatique de l’étudiant Dupont depuis la vue MoyLyonnais ?

8. Donnez un droit d’accès en consultation à la vue EtudLyonnais à votre binôme (Un second compte que vous allez créer si nécessaire sous Oracle). Vérifier le contenu de la vue depuis les 2 comptes.

9. Mettez à jour le nom de rue de Dupont qui passe à Rue de la Meuse, 69008, Lyon. Vérifier sur les 2 comptes si la mise à jour est faite. Le second compte essaye de remettre Dupont à son ancienne adresse. Que se passe –t-il ? Quoi faire ?

10. Votre binôme insère un nouvel étudiant lyonnais. Que doit-il faire pour que vous puissiez consulter ce nouvel étudiant ?

11. Une erreur de saisie entre 2 notes de 2 étudiants sur une même épreuve doit être corrigée. Comment procéder ?

12. Que doit-on faire pour éviter des mises à jour erronées des notes des étudiants par vous ou votre binôme (conflit de mise à jour) ?

**V – Transactions et reprise sur panne**

**5.1 – Les transactions sous Oracle**

1. Une transaction est une séquence d’ordres SQL exécutés tous ou aucun[[1]](#footnote-1).

2. Le début d’une transaction est le premier ordre qui suit :

(a) connexion au serveur, ou

(b) fin de transaction.

Les fins de transaction :

(a) annulation

(b) confirmation (*transaction dite alors validée*).

3. listes des confirmations :

(a) explicite : *commit*

(b) implicites :

i. tous ordres de mise à jour de schéma

ii. commande de déconnexion du mode interactif (ou application)

iii. *grant* (ordre de gestion de la confidentialité)

iv. en mode de confirmation automatique : toutes mises à jour de l’instance[[2]](#footnote-2)

effet : confirme toutes mises à jour de l’instance depuis le début de la transaction

4. listes des annulations :

(a) explicite : *rollback*

(b) implicite : déconnexion anormale (autre qu’ordre explicite de déconnexion interactif ou application)

effet : annule toutes mises à jour de l’instance depuis le début de la transaction

5. remarques :

(a) au cours d’une transaction : “table virtuelle locale”, différente de la table réelle visible par

tous les clients

(b) un ordre peut échouer sans annuler la transaction (ex : insertion avec colonnes incorrectes)

6. rapport avec la persistance : une mise à jour[[3]](#footnote-3) persiste si et seulement si elle fait partie d’une transaction validée

7. recommandations :

(a) ne pas mettre dans la même transaction des mises à jour non corrélées logiquement

(b) expliciter toutes les confirmations et annulations par des *commit* ou des *rollback*.

**5.2 – Analyse de programme**

On s’intéresse ici à prévoir le contenu de la base à chaque instant au cours du déroulement d’une séquence d’ordres SQL[[4]](#footnote-4).

On considère une table T(A INTEGER, B INTEGER) contenant à l’origine le(s) n-uplet(s) indiqué(s) au début de chaque tableau. Tout ordre dessus est autorisé à tout utilisateur. On considère un client p1 qui lance les séquences d’ordres suivantes, immédiatement après sa connexion, avec le mode de confirmation automatique non activé. Un autre client quelconque est représenté par p2. Compléter les tableaux suivants. Indiquez le début et la fin de chaque transaction. Vérifiez ensuite sur machine avec SQLPLUS.

|  |  |  |
| --- | --- | --- |
| ordres | T ds trans p1 | T vue par p2 |
|  | (1,2) | (1,2) |
| INSERT (3,4) |  |  |
| INSERT (1,2) |  |  |
| ROLLBACK |  |  |
| COMMIT |  |  |
| INSERT (1,2) |  |  |
| EXIT |  |  |

|  |  |  |
| --- | --- | --- |
| Ordres | T ds trans p1 | T vue par p2 |
|  | (1,2) | (1,2) |
| ROLLBACK |  |  |
| INSERT (1,2) |  |  |
| ALTER TABLE T ADD (C INTEGER) |  |  |
| UPDATE T SET B = 3 WHERE A = 1 |  |  |
| *panne : tapez ctl-\* |  |  |

|  |  |  |
| --- | --- | --- |
| ordres | T ds trans p1 | T vue par p2 |
|  | (1,2) | (1,2) |
| INSERT (3,4) |  |  |
| INSERT (5,6) |  |  |
| SET AUTOCOMMIT ON |  |  |
| INSERT (7,8) |  |  |
| ROLLBACK |  |  |
| INSERT (1,2,3,4) |  |  |
| DELETE FROM T WHERE A = 1 |  |  |
| *panne : tapez ctl-\* |  |  |

**VI – Introduction au contrôle de concurrence d’accès :**

Oracle gère des verrous pour contrôler les processus concurrents, au niveau des tables ou des n-uplets. Certains sont posés implicitement (sur les n-uplets lors des mises à jour d’instance, d’autres explicitement (sur les tables). Il n’est pas possible que deux processus possèdent des verrous globaux sur la même table ou locaux sur le même n-uplet. Les verrous sont relâchés en fin de transaction.

On s’initie ci-dessous à faire de l’intuition sur déroulement des ordres posant des verrous, puis à en poser nous-même là où il faut.

**6.1 – Accès concurrents :**

On va maintenant étudier le comportement concret d'Oracle en cas d'accès concurrents à la même ressource. Pour cela on va simuler l'exécution concurrente de programmes à l'aide du petit ensemble de lectures/écritures.

Soit la table Client définie comme suit : Client (**IDC**, Nom, Prenom, Age, Salaire, Solde)

* Créer 1 client nommé DUPONT Joe (attention aux majuscules/minuscules) de solde 12000 euros, âgé de 22 ans et de salaire 2800 €.
* Soient 3 fichiers, nommés Solde.sql, Depot.sql et Retrait.sql :

**Solde.sql**

PROMPT 'Solde de Joe';

SELECT Nom, Solde FROM client where Nom = 'DUPONT';

#### Depot.sql

PROMPT 'Un dépôt de 6000 euros est fait sur le compte de Joe' ;

UPDATE client SET solde = solde + 6000 WHERE Nom = 'DUPONT';

PROMPT 'Nouveau Solde de Joe';

SELECT Nom, Solde FROM client where Nom = 'DUPONT';

#### Retrait.sql

PROMPT 'Joe retire 4000 euros';

UPDATE client SET solde = solde - 4000 WHERE Nom = 'DUPONT';

PROMPT 'Nouveau Solde de Joe';

SELECT Nom, Solde FROM client where Nom = 'DUPONT';

Ensuite, soient 2 sessions utilisateur,nommées user1 et user2, en situation de concurrence. Dans tout ce qui suit, on note *INSTRi* l'exécution de l'instruction *INSTR* par l'utilisateur *i*. Par exemple Depot1 correspond à l'exécution du fichier Depot dans la première fenêtre par la commande @Depot. On note de même *ROLi* et *COMi* l'exécution des commandes rollback; et commit; dans la fenêtre *i*.

ORACLE pratique un verrouillage à deux phases assez libéral (degré d'isolation 1 – READ COMMITTED), qui ne garantit pas la sérialisabilité. De plus, un protocole multi-versions est utilisé. Du coup, aucun verrou n'est placé sur une ligne lors d'une lecture. Expérimentez les exécutions suivantes et essayez de comprendre le fonctionnement du verrouillage d'Oracle dans ce mode.

Simuler l’exécution des séquences d'instruction décrites ci-dessous.

1. L'utilisateur 1 (Joe) effectue un retrait. L'utilisateur 2 ne fait que consulter les soldes…

***COM*1,*COM*2,*Solde1, Solde2, Retrait1, Solde2, ROL1, Solde1, Solde2.***

2. idem, mais avec un *commit :*

***COM*1,*COM*2,*Solde1, Solde2, Retrait1, Solde2, COM*1*, Solde1, Solde2,***

3. L'utilisateur 1 (Joe) fait un retrait, alors que son solde est crédité par sa banque (utilisateur 2). Un blocage est apparu. Pourquoi ?

***COM*1,*COM*2,*Solde1, Solde2, Retrait1,Solde2, Depot*2, *Solde1, Solde2, COM*1, *COM2***

**6.2 – Analyse de programme**

On considère une table T(A integer, B integer) contenant au début les n-uplets (1,1) et (2,2). Tout ordre dessus est autorisé à tout utilisateur. On considère deux processus clients p1 et p2 qui effectuent les séquences d’ordres suivantes, immédiatement après leur connexion. Un autre client quelconque est représenté par p3.

Remplir les tableaux dans l’ordre dans lequel les ordres sont effectivement exécutés.

1. 1 (p1) : DELETE T WHERE B=1;

2 (p1) : COMMIT;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client | ordre | T ds trans p1 | T ds trans p2 | T vue par p3 |
|  |  | 11  22 | 11  22 | 11  22 |
|  |  |  |  |  |
|  |  |  |  |  |

2. 1 (p1) : DELETE T WHERE B=1;

2 (p2) : UPDATE T SET B=B-1;

3 (p1) : COMMIT;

4 (p2) : COMMIT;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client | ordre | T ds trans p1 | T ds trans p2 | T vue par p3 |
|  |  | 11  22 | 11  22 | 11  22 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

3. 1 (p2) : UPDATE T SET B=B-1;

2 (p1) : DELETE T WHERE B=1;

3 (p2) : COMMIT;

4 (p1) : COMMIT;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client | ordre | T ds trans p1 | T ds trans p2 | T vue par p3 |
|  |  | 11  22 | 11  22 | 11  22 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

4. 1 (p1) : UPDATE T SET A=1 WHERE B=2;

2 (p2) : DELETE T WHERE A=1;

3 (p1) : COMMIT;

4 (p2) : COMMIT;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Client | ordre | T ds trans p1 | T ds trans p2 | T vue par p3 |
|  |  | 11  22 | 11  22 | 11  22 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Annexe 1 :**

-- ============================================================

-- Nom de la base : BD\_Notation

-- Nom de SGBD : ORACLE

-- Date de création : Mars 2020

-- Auteur : Abdelkrim LAHLOU

-- ============================================================

-- ============================================================

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MM-YYYY' ;

-- ============================================================

-- Destruction des tables

DROP TABLE Etudiant CASCADE CONSTRAINTS;

DROP TABLE Matiere CASCADE CONSTRAINTS;

DROP TABLE Epreuve CASCADE CONSTRAINTS;

DROP TABLE Notation CASCADE CONSTRAINTS;

-- Creation de tables

CREATE TABLE Etudiant (

NumEtu NUMBER(8) PRIMARY KEY,

Nom VARCHAR2(20),

Prenom VARCHAR2(20),

DateNais DATE,

Rue VARCHAR2(50),

CP CHAR(5),

Ville VARCHAR2(25)

);

CREATE TABLE Matiere (

CodeMat CHAR(3) PRIMARY KEY,

Libelle VARCHAR2(20),

Coef NUMBER(3,2)

);

CREATE TABLE Epreuve (

NumEpreuve NUMBER(10) PRIMARY KEY,

DateEpreuve DATE,

Lieu VARCHAR2(20),

CodeMat CHAR(3)

);

CREATE TABLE Notation (

NumEtu NUMBER(8),

NumEpreuve NUMBER(10),

Note NUMBER(4,2),

PRIMARY KEY (NumEtu,NumEpreuve)

);

ALTER TABLE Epreuve ADD CONSTRAINT FkEpreuve1 FOREIGN KEY(CodeMat) REFERENCES Matiere(CodeMat);

ALTER TABLE Notation ADD CONSTRAINT FkNotation1 FOREIGN KEY(NumEtu) REFERENCES Etudiant(NumEtu);

ALTER TABLE Notation ADD CONSTRAINT FkNotation2 FOREIGN KEY(NumEpreuve) REFERENCES Epreuve(NumEpreuve);

-- Remplissage des tables

INSERT INTO Etudiant VALUES (110,'Dupont','Albert','01-06-1990','RuedeCrimée','69001','Lyon');

INSERT INTO Etudiant VALUES (222,'West','James','03-09-1993','Studio',NULL,'Hollywood');

INSERT INTO Etudiant VALUES (300,'Martin','Marie','05-06-1998','RuedesAcacias','69130','Ecully');

INSERT INTO Etudiant VALUES (421,'Durand','Gaston','15-11-1990','RuedelaMeuse','69008','Lyon');

INSERT INTO Etudiant VALUES (575,'Titgoutte','Justine','28-02-1995','CheminduChâteau','69630','Chaponost');

INSERT INTO Etudiant VALUES (667,'Dupond','Noémie','18-09-1997','RuedeDôle','69007','Lyon');

INSERT INTO Etudiant VALUES (999,'Phantom','Marcel','30-01-1990',NULL,NULL,NULL);

INSERT INTO Matiere VALUES ('STA','Statistique',0.4);

INSERT INTO Matiere VALUES ('INF','Informatique',0.4);

INSERT INTO Matiere VALUES ('ECO','Econométrie',0.2);

INSERT INTO Epreuve VALUES(11031,'15-12-2003','Salle191L','STA');

INSERT INTO Epreuve VALUES(11032,'01-04-2004','AmphiG','STA');

INSERT INTO Epreuve VALUES(21031,'30-10-2003','Salle191L','INF');

INSERT INTO Epreuve VALUES(21032,'01-06-2004','Salle192L','INF');

INSERT INTO Epreuve VALUES(31030,'02-06-2004','Salle05R','ECO');

INSERT INTO Notation VALUES (110,11031,10);

INSERT INTO Notation VALUES (110,11032,11.5);

INSERT INTO Notation VALUES (110,21031,8.5);

INSERT INTO Notation VALUES (110,21032, NULL);

INSERT INTO Notation VALUES (110,31030,13);

INSERT INTO Notation VALUES (222,11031,9);

INSERT INTO Notation VALUES (222,11032,14);

INSERT INTO Notation VALUES (222,21031,12);

INSERT INTO Notation VALUES (222,21032,16);

INSERT INTO Notation VALUES (222,31030,20);

INSERT INTO Notation VALUES (300,11031,14);

INSERT INTO Notation VALUES (300,11032,20);

INSERT INTO Notation VALUES (300,21031,20);

INSERT INTO Notation VALUES (300,21032,13.5);

INSERT INTO Notation VALUES (300,31030,16);

INSERT INTO Notation VALUES (421,11031,5.5);

INSERT INTO Notation VALUES (421,11032,17);

INSERT INTO Notation VALUES (421,21031,1.5);

INSERT INTO Notation VALUES (421,21032, NULL);

INSERT INTO Notation VALUES (421,31030,10);

INSERT INTO Notation VALUES (575,11031,13);

INSERT INTO Notation VALUES (575,11032,9);

INSERT INTO Notation VALUES (575,21031,12.5);

INSERT INTO Notation VALUES (575,21032,14);

INSERT INTO Notation VALUES (575,31030,7);

INSERT INTO Notation VALUES (667,11031,16);

INSERT INTO Notation VALUES (667,11032,20);

INSERT INTO Notation VALUES (667,21031,8.5);

INSERT INTO Notation VALUES (667,21032,9.5);

**Annexe-2**

Drop user User1 cascade;

create user User1 identified by L3appLSI

default tablespace USERS

temporary tablespace TEMP

quota 2 M on USERS

PROFILE DEFAULT

ACCOUNT UNLOCK;

GRANT ALTER SESSION TO User1;

GRANT CREATE DATABASE LINK TO User1;

GRANT CREATE SYNONYM TO User1;

GRANT CREATE VIEW TO User1;

GRANT CREATE TRIGGER TO User1;

GRANT DEBUG CONNECT SESSION TO User1;

GRANT UNLIMITED TABLESPACE TO User1;

GRANT CONNECT TO User1;

GRANT RESOURCE TO User1;

GRANT SELECT\_CATALOG\_ROLE TO User1;

**Manipulation de Base SQL\*PLUS :**

*Aide en ligne :*

**help <nom-commande>** : aide sur la commande citée (ex. : help exit)

*Dialogue avec le système hôte :*

**host <commande>** ou **! <commande>** : lance une commande du système d'exploitation.

*Travail sous sqlplus :*

**Sous l’interface SQLPLUS directement :** la plupart des commandes sql peuvent être tapées sur une ou plusieurs lignes et doivent être terminées par ; SQL ne fait pas la différence entre ses commandes minuscules et majuscules. Il est toutefois recommandé de mettre les mots SQL en majuscules, d'aller à la ligne et d'indenter les lignes de commande.

**A partir d'un script :** On écrit une suite de commandes (terminées par un ;) à l'aide d'un éditeur de texte (vi, emacs, xemacs, ...), bien qu'il y ait un buffer SQL et l'on sauvegarde le fichier dans le répertoire de travail, le nom du fichier devant avoir pour extension **.sql.** On exécute les commandes soit en faisant du "couper/coller" de l'éditeur vers SQL, soit en dialoguant avec le système hôte de la manière suivante :

**start** <*nomprog*> ou **@** <*nomprog*> : charge le fichier et lance les commandes contenues dans le fichier *nomprog.sql* (inutile de spécifier l’extension .sql)

*Memento SQL\*PLUS :*

**save filename.sql** sauve le contenu du buffer dans un fichier de nom filename.sql

**get filename.sql** charge le buffer avec le contenu du fichier de nom filename.sql

**start filename** charge le buffer et lance l'exécution du fichier script sql

**run**  lance l'exécution du contenu du buffer

**spool filename.txt** copie la sortie écran sur le fichier filename.txt

**spool off** suspend l'opération précédente

**help commande** pour obtenir de l'aide sur la commande donnée en argument

*Variables d'environnement :*

**set long 1024** pour voir la totalité des définitions de vues

**set pagesize 10** formate la sortie écran par blocs de 20 lignes

**set pause on** ne visualise la sortie qu'après un 2ième Return - bloc par bloc -

*Schéma d'information ( ou "méta-base") :*

**desc[ribe] tablename** donne le schéma de la relation tablename

**all\_catalog** relation donnant toutes les tables accessibles

**user\_catalog** relation donnant les seules tables du USER

**cat**  synonyme de la précédente

**user\_tables** relation donnant toutes les tables du USER

**tab** synonyme de la précédente

**all\_objects** relation donnant tous les objets accessibles

**user\_objects** relation donnant les seuls objets du USER

**obj**  synonyme de la précédente ..

**user\_sys\_privs** relation donnant les privilèges système du USER

**user\_tab\_privs** relation donnant les privilèges sur les objets accessibles

1. On considérera ici seulement les mises à jour de l’instance. [↑](#footnote-ref-1)
2. set autocommit on/off, show autocommit ; positionnement par défaut à la connexion [↑](#footnote-ref-2)
3. l’état de la base en résultant [↑](#footnote-ref-3)
4. Un exercice apparemment un peu scolaire, mais... incontournable pour bien utiliser ensuite les outils dans ses propres programmes. [↑](#footnote-ref-4)