



جامعة هواري بومدين للعلوم والتكنولوجيا

Université des Sciences et de Technologie Houari Boumediene
Faculté d'Electronique et d'Informatique
Département d'Informatique

Travaux Pratiques

Bases de Données

2 année Licence

Enseignants:

BOUZIANE Nabila

AHMED NACER Mehdi

21/02/2019

Objectif des travaux pratiques de base de données

L'objectif des travaux pratiques de base de données est tout d'abord de maîtriser et d'utiliser l'environnement d'Oracle, d'apprendre et de manipuler le langage de requêtes SQL. Oracle est un Système de Gestion de Bases de Données relationnel (SGBDR). Il utilise le langage SQL (*Structured Query Language*), langage de définition, de manipulation et de contrôle de bases de données relationnelles.

Ces travaux pratiques sont scindés donc en deux principales parties. La première partie est axée sur l'environnement oracle à savoir la création de tables, manipulation, interrogation des bases de données. La seconde partie vise à utiliser l'environnement de développement Eclipse avec java afin de concevoir une application permettant l'affichage des résultats des requêtes SQL ainsi que la manipulation des tuples via ses composants.

L'objectif de ces travaux pratiques étant de maîtriser le langage SQL et l'environnement d'Oracle et non pas de maîtriser le développement d'applications avec Eclipse. Dans le cadre de ces travaux pratiques, tout autre environnement de développement d'applications aurait pu être utilisé afin de faciliter la création d'applications.

Les travaux pratiques de base de données regroupent les TPs suivants:

N° du TP	Objectif	Nombre de séances de 1 h 30
1	L'éditeur de commandes SQL <ul style="list-style-type: none">• Télécharger et installer un système de gestion de base de données tel que Oracle 11g ou plus.• Se connecter et déconnecter du serveur Oracle.• Utiliser les commandes de l'éditeur SQLPLUS .	1
2	Création des tables et des indexes <ul style="list-style-type: none">• Créer des tables avec des contraintes d'intégrités,• Créer des indexes,• Modifier les structures des tables et les contraintes.	1

3	Insertions et modifications dans la Base de données <ul style="list-style-type: none"> Insérer des tuples dans la bases de données. Modifier les tuples. 	1
4	Recherche dans la Base de données <ul style="list-style-type: none"> Interroger la Base de données par des instructions SQL simples puis complexes. 	2
5	Gestion des utilisateurs et des privilèges <ul style="list-style-type: none"> Créer des utilisateurs, leurs accorder des privilèges Supprimer des privilèges et des utilisateurs. 	1
6	Réalisation et interrogation de la base de données "Étudiants" <ul style="list-style-type: none"> Consolider les acquis des cinq premiers TPs. 	1+1/2
7	Les fonctions d'agrégat	1
8	Création et manipulation des vues	1/2
9	Eclipse et WindowBuilder <ul style="list-style-type: none"> Se familiariser avec WindowBuilder sous eclipse Créer une interface graphique. La création des composants se fait simplement à travers une palette graphique, le code java est généré automatiquement. 	1
10	Connectivité avec la base de données Oracle à travers Eclipse Accéder au serveur Oracle à partir d'une interface WindowBuilder afin d'afficher, de manipuler et d'interroger les tables de la base de données à partir des formes Swing.	1
11	Projet Final	2
Séances nécessaires		13

Table des matières

TP N° 1 : L'éditeur de commandes SQL.....	5
TP N° 2 : Création des tables et des indexes	9
TP N° 3 : Insertions et modifications dans la Base de données.....	12
TP N° 4 : Gestion des utilisateurs et des privilèges	14
TP N° 5 : Recherche dans la Base de données	16
TP N° 6 : Réalisation et interrogation de la base de données "Étudiants"	18
TP N° 7 : Les fonctions d'agrégat	21
TP N° 8 : Création et manipulation des vues.....	22
TP N° 9 : Eclipse et WindowBuilder	23
TP N° 10 : Connectivité avec la base de données Oracle à travers Eclipse	28
TP N° 11 : Projet final	30
Annexe A : Langage de Contrôle de Données.....	31
Annexe B : <i>Langage de Définition de Données</i>	33
Annexe C : Langage de Manipulation de Données	39
Annexe D: Fichiers TPOracleScript.sql et donnees.txt	41
Annexe E: Solutions des TPs	45

TP N° 1 : L'éditeur de commandes SQL

SQLPlus ou **Run SQL Command Line** sont des éditeurs de commandes SQL du SGBD Oracle. Ces éditeurs permettent de se connecter à une base de données Oracle afin d'y exécuter des ordres SQL ou des procédures PL/SQL. Les procédures PL/SQL ne seront pas étudiées dans ce TP.

Les types de commandes de l'éditeur SQLPlus

SQLPlus accepte trois types de commandes :

- Les commandes SQLPlus : commandes internes de l'éditeur SQLPlus, permettant entre autre de le paramétrer. Ces commandes ne peuvent pas manipuler les données de la base de données. Nous testerons ces commandes dans ce TP.
- Les commandes SQL : le langage de requête utilisé pour extraire et manipuler les données de la base.
- Les commandes PL/SQL : langage procédural propre à Oracle, utilisé pour intégrer des requêtes SQL à des traitements procéduraux (c'est-à-dire en utilisant des variables, des boucles, des structures de contrôles, etc.)

Instruction importante

- Télécharger les fichiers "[TPOracleScript.sql](http://perso.usthb.dz/~nbouziane/TPBDD/)" et "[donnees.txt](http://perso.usthb.dz/~nbouziane/TPBDD/)" à partir du site : <http://perso.usthb.dz/~nbouziane/TPBDD/>. Ces deux fichiers seront utilisés dans tous les TPs.

Saisie des commandes SQL

- Dès que le prompt SQL> est affiché, les commandes SQL peuvent être saisies.
- Ne laisser pas une ligne vide sous SQLPLUS pour une même instruction SQL.
- SQLPlus ne fait pas de différence entre les majuscules et les minuscules concernant les commandes.
- SQLPlus fait la différence entre les majuscules et les minuscules concernant les chaînes de caractères. Exemple : 'exemple' différent de 'EXEMPLE'
- La saisie d'une instruction SQL peut être effectuée sur plusieurs lignes sans ponctuation particulière ce qui permet une meilleure lisibilité.

- L'instruction SQL est stockée dans un buffer et les lignes sont numérotées.
- Pour arrêter la saisie et débiter l'interprétation et donc l'exécution de la commande SQL, vous devez:
 - finir la ligne par un point-virgule,
 - appuyer sur le retour à la ligne « entrée ».
- Après chaque exécution de la commande SQL, SQLPlus doit indiquer le résultat de cette exécution.

Tâche 1: Lancement de l'éditeur SQLPlus

1. L'éditeur SQLPlus se trouve sur le bureau windows ou parmi les menus "Programme" de Windows. Ce TP explique l'utilisation de SQLPlus, ce qui est identique à **Run SQL Command Line** sauf qu'il faut taper l'instruction **connect** au démarrage ce dernier.
2. Connecter vous à Oracle avec l'utilisateur «**system**» ayant le mot de passe "**orcl1**", pré-installé avec Oracle.
3. L'invite de commande **SQL >** signifie que l'on s'adresse à Oracle.
4. Les commandes entrées par la suite sont des ordres SQL destinés au SGBDR.
5. Écrire et sauvegarder toutes les instructions SQL dans un fichier ayant l'extension « .sql » avec le logiciel **NOTEPAD++** et sélectionner le type du langage (SQL) dans le menu « Langage » de NOTEPAD++.

Tâche 2: Manipulation des commandes SQLPlus

1. Tester les commandes SQLPlus listées dans le tableau ci-dessous. Ces commandes peuvent être abrégées exemple CONNECT devient CONN.

Commande SQLPlus	Description
CONNECT	Réalise une nouvelle connexion. Syntaxe: CONN[ECT] « utilisateur » « mot_de_passe »
DISCONNECT	Se déconnecter de la base de données Syntaxe: DISC[ONNECT]
QUIT	Quitter l'outil SQLPlus et se déconnecter de la Base de données Syntaxe: QUIT
RUN .	La commande « RUN » ou « / » affiche le contenu du tampon et exécute l'instruction stockée dans le tampon mémoire Syntaxe: R[UN] ou « / »

EDIT	Ouvrir un fichier. Syntaxe: ED[IT] fichier [.extension]
SHOW USER	Affiche l'utilisateur connecté. Syntaxe: SHO[W] USER

Tâche 3: Exécution d'un script SQL sous SQLPlus

- Un script SQL est un fichier contenant des commandes SQL.
 - Exemple: le fichier utilisé pour ce TP "TPOracleScript.sql".
- Pour exécuter un script, taper sous SQLPlus: (Le chemin du fichier ne doit pas contenir de caractère blanc (espace))

```
SQL > @ suivi_du_chemin_et_du_nom_de_fichier_avec_son_extension
```

- Pour interrompre la commande SQL en cours: appuyer sur **ctrl+C** ou **del**.
- Créer le répertoire TPBDD dans le disque c:
 - Sauvegarder les fichiers TPOracleScript.sql et donnees.txt dans TPBDD.
 - Lancer l'éditeur SQLPlus.
 - Taper le nom de l'utilisateur : **system** et le mot de passe : **orcl1**
 - Exécuter le script "TPOracleScript.sql".

Tâche 4: Sauvegarde de la session de travail

- Pour lancer la sauvegarde des commandes qui seront tapées par l'utilisateur, et les réponses d'Oracle dans un fichier, il suffit d'utiliser l'instruction Spool (le fichier de sauvegarde sera créé par la commande ci-dessous, il n'existe pas avant).

```
SPOOL Chemin_et_repertoire_de_sauvegarde_suivi_du_nom_du_fichier_à_créer_sans_extension
```

- Lancer le spool suivant:

```
SPOOL c:/TPBDD/sauvegarde
```

Le fichier ayant l'extension « .LST » sera créé. (sauvegarde.LST)

- Arrêter la sauvegarde:

```
SPOOL OFF
```

3. Analyser le contenu du fichier sauvegarde.LST.

4. Que contient le fichier sauvegarde.LST.

5. Relancer le spool.

6. Relancer l'exécution du script "TPOracleScript.sql".

7. Arrêter la sauvegarde.

8. Ouvrir le fichier sauvegarde.LST et analyser son contenu.

9. Que contient le fichier sauvegarde.LST.

TP N° 2 : Création des tables et des indexes

Le Langage SQL

SQL " *Structured Query Language* " est à la fois un langage de :

- LDD (*Langage de Définition de Données*): ensemble d'instructions permettant de créer des tables, des indexes, des vues et d'associer à une définition des contraintes.
- LMD (*Langage de Manipulation de Données*) : ajout, suppression, modification et interrogation des données
- LCD (*Langage de Contrôle de Données*) : gestion des protections d'accès.

Tâche 1: Analyse du fichier TPOracleScript.sql

1. Lancer l'interpréteur SQLPlus (Utilisateur : system, mot de passe : orcl**1**),
2. Après vérification du nom d'utilisateur et du mot de passe, l'invite SQL > s'affiche, alors il est possible de lancer l'ensemble des commandes SQL qui seront associées à cet utilisateur.
3. Télécharger les fichiers "TPOracleScript.sql" et "donnees.txt" à partir du site : <http://perso.usthb.dz/~nbouziane/TPBDD/>
4. Ouvrir le fichier TPOracleScript.sql avec Notepad++.
5. Écrire et sauvegarder toutes les instructions SQL dans un fichier ayant l'extension « .sql » avec le logiciel **NOTEPAD++** et sélectionner le type du langage (SQL) dans le menu « Langage » de NOTEPAD++.
6. Lancer un spool dans le répertoire c:/TPBDD.
7. Exécuter le fichier TPOracleScript.sql.
8. Vérifier que toutes les instructions SQL du fichier TPOracleScript.sql ont été interprétées par SQLPLUS. Ces instructions sont elles des LMD, LDD ou LCD?

-
9. Quelles sont les différentes tables de cette base de données, préciser leurs attributs en utilisant la commande SQLPlus "DESCRIBE", quelles sont les clés primaires?
-
-

10. Quelles sont les relations entre ces tables?

11. A quoi servent les indexes créés dans cette base de données? Quels sont ces indexes?

Tâche 2: Création de tables avec contraintes d'intégrités

1. Ajouter les tables ci-dessous à la base de données de la tâche1.
2. Ajouter la table **SEANCE** avec les attributs ci-dessous sans oublier les contraintes d'intégrités.
 - TITRE Chaîne de 40 caractères not null (clé primaire et clé étrangère avec la table film)
 - NOM_SALLE not null Chaîne de 30 caractères, (clé primaire)
 - HEURE_DEBUT entier, not null (clé primaire)
 - VERSION Chaîne de 10 caractères.
3. Ajouter la table **VU** avec les attributs ci-dessous sans oublier les contraintes d'intégrités.
 - NOM_SPECTATEUR Chaîne de 30 caractères not null, (clé primaire),
 - TITRE Chaîne de 40 caractères not null (clé primaire et clé étrangère avec la table film)
4. Ajouter la table **AIME** avec les attributs ci-dessous:
 - NOM_AMATEUR Chaîne de 30 caractères,
 - TITRE Chaîne de 20 caractères.
5. Les instructions SQL de création des tables SEANCE, AIME et VU doivent être ajoutées et sauvegardées dans le fichiers TPOracleScript.sql.
6. Créer un synonyme de la table **SEANCE**.

Tâche 3: Création d'indexes

1. Créer un index pour la table **SEANCE** sur l'attribut TITRE dans l'ordre croissant.
2. Créer un index pour la table **VU** sur l'attribut TITRE dans l'ordre décroissant.
3. Créer un index pour la table **AIME** sur l'attribut TITRE dans l'ordre croissant.
4. Les instructions SQL de création des indexes précédents doivent être ajoutées et sauvegardées dans le fichiers TPOracleScript.sql.

Tâche 4: Modification du schéma et des contraintes

- Toutes les instructions SQL ci-dessous doivent être ajoutées et sauvegardées dans le fichiers TPOracleScript.sql.
1. Modifier le type de l'attribut TITRE de la table AIME de manière à l'augmenter de 20 caractères.
 2. Ajouter une contrainte sur la table SEANCE pour que HEURE_DEBUT soit toujours supérieure à 13,
 3. Ajouter une contrainte sur la table SEANCE pour que NOM_SALLE soit "IBN ZAIDOUN" ou "IBN KHALDOUN".
 4. Ajouter une contrainte sur la table SEANCE pour que VERSION soit par défaut "VO" ie version originale.
 5. Ajouter une contrainte sur la table AIME pour que NOM_AMATEUR commence obligatoirement par "R".
 6. Supprimer les 4 contraintes précédentes.
 7. Changer le type de l'attribut HEURE_DEBUT de la table SEANCE au type Chaîne de 6 caractères.
 8. Ajouter l'attribut PRENOM_AMATEUR de type Chaîne de 30 caractères à la table AIME.
 9. Supprimer l'attribut PRENOM_AMATEUR de la table AIME.
 10. Retirer la contrainte NOT NULL à l'attribut HEURE_DEBUT de la table SEANCE.
 11. Ajouter la contrainte NOT NULL à l'attribut HEURE_DEBUT de la table SEANCE.
 12. Ajouter les attributs NOM_AMATEUR et TITRE de la table AIME comme clés primaires.
 13. Ajouter l'attribut TITRE de la table AIME comme clé étrangère avec la table film.
 14. Supprimer les clés primaires et étrangère de la table VU.
 15. Remettre les clés primaires et étrangère de la table VU.

TP N° 3 : Insertions et modifications dans la Base de données

Tâche 1: Analyse du fichier donnees.txt

1. Lancer l'interpréteur SQLPlus (Utilisateur : system, mot de passe : orcl**1**)
2. Ouvrir le fichier donnees.txt avec Notepad++.
3. Écrire et sauvegarder toutes les instructions SQL dans un fichier ayant l'extension « .sql » avec le logiciel **NOTEPAD++** et sélectionner le type du langage (SQL) dans le menu « Langage » de NOTEPAD++.
4. Lancer un spool.
5. Exécuter le fichier TPOracleScript.sql. Vérifier que toutes les instructions SQL du fichier TPOracleScript.sql ont été interprétées
6. Exécuter le fichier donnees.txt.
7. Analyser le contenu du fichier donnees.txt.
8. Supprimer la table FILM de la base de données. Y a t il un problème? pourquoi?

9. Quelle est la solution pour supprimer la table FILM.

Tâche 2: Insertion de tuples

1. Exécuter le fichier TPOracleScript.sql.
2. Relancer le script donnees.txt et valider les insertions avec l'instruction COMMIT.
3. Insérer les tuples ci-dessous dans la base de données:

Table	Tuple à inséré
SEANCE	'Perfect World', 'Beta', '19:00', 'VO'
PRODUIT	'Adam Moore', 'ET'

4. Y a t il un problème lors des insertions précédentes?

5. Insérer les tuples ci-dessous dans la base de données:

Table	Tuple à inséré
JOUE	'Kevin Costner', 'Waterworld'
PRODUIT	'Kevin Costner', 'Waterworld'
SEANCE	'Waterworld', 'Alpha', '15:00', 'VF'
SEANCE	'Waterworld', 'Alpha', '17:00', 'VO'
SEANCE	'Waterworld', 'Beta', '15:00', 'VF'
VU	'Kevin Costner', 'Waterworld'

6. Y a t il un problème lors des insertions précédentes? pourquoi?

7. Insérer le tuple suivant dans la table FILM: 'Waterworld', 250, 'US', 'Rickley Moore'

8. Réinsérer les tuples du tableau précédent.

9. Valider les insertions avec l'instruction SQL "COMMIT".

10. Sauvegarder toutes les instructions d'insertion dans le fichier donnees.txt.

Tâche 3: Modification et suppression de tuples

1. Ajouter une heure (60minutes) à la durée du film 'Waterworld'.

2. Modifier toutes les versions des films projetés en version originale "VO".

3. Supprimer tous les films projetés qui débutent à "17:00".

4. Supprimer le film 'Waterworld' de la table film. Y a t il un problème? pourquoi? donner une solution.

TP N° 4 : Gestion des utilisateurs et des privilèges

Tâche 1:

1. Lancer l'interpréteur SQLPlus (Utilisateur : system, mot de passe : orcl¹)
2. Exécuter le fichier TPOracleScript.sql. Vérifier que toutes les instructions SQL du fichier TPOracleScript.sql ont été exécutées.
3. Exécuter le fichier donnees.txt.
4. Créer un nouvel utilisateur "TP4user1" ayant le mot de passe "TP41".
5. Afficher l'utilisateur connecté.
6. Déconnecter l'utilisateur connecté.
7. Accorder à l'utilisateur "TP4user1" tous les privilèges et connecter cet utilisateur au SGBD.
8. Exécuter l'instruction `SELECT * FROM FILM` pour afficher le contenu de la table FILM. En cas d'échec, expliquer, corriger et re-exécuter de nouveau;

9. Supprimer tous les privilèges accordés à l'utilisateur "TP4user1".
10. Supprimer l'utilisateur "TP4user1".

Tâche 2:

1. Créer un nouvel utilisateur "TP4user2" ayant le mot de passe "TP42".
2. Accorder le privilège « SELECT » de la table SEANCE à l'utilisateur "TP4user2" .
3. Accorder le privilège « INSERT » dans la table VU à l'utilisateur "TP4user2".
4. Connecter l'utilisateur "TP4user2". En cas d'échec, expliquer, corriger et reconnecter de nouveau;

5. Tester tous les privilèges accordés à l'utilisateur "TP4user2".

6. Supprimer tous les privilèges accordés à l'utilisateur "TP4user2".
7. Supprimer l'utilisateur "TP4user2".
8. Accorder le privilège « SELECT » dans la table VU à tous les utilisateurs.
9. Supprimer le privilège «SELECT » dans la table VU à tous les utilisateurs.

Tâche 3:

1. Créer un nouvel utilisateur "TP4user3" ayant le mot de passe "TP43".
2. Accorder à l'utilisateur "TP4user3" tous les privilèges (sauf CREATE et DROP) sur les tables de la BDD du TPOracleScript.sql.
3. Tester tous les privilèges accordés à l'utilisateur "TP4user3".
4. Supprimer tous les privilèges accordés à l'utilisateur "TP4user3" sur la table VU.
5. Supprimer l'utilisateur "TP4user3".

TP N° 5 : Recherche dans la Base de données

Tâche 1:

- Lancer l'interpréteur SQLPus (Utilisateur : system, mot de passe : orcl1)
- Ré-exécuter le fichier "TPOracleScript.sql".
- Ré-exécuter le fichier "donnees.txt".
- Exécuter des requêtes SQL pour répondre à ces questions .
 1. Quels sont les titres des films qui contiennent la chaîne suivante 'World' ?
 2. Quels sont les titres des films dont la durée est inférieure à 60 minutes ?
 3. Quels sont les titres des films dont la durée est supérieure à 140 minutes ?

Tâche 2:

- Exécuter des requêtes SQL pour répondre à ces questions:
 1. Quel est le nom du réalisateur du film 'Waterworld'?
 2. Où peut-on voir jouer 'Kevin Costner' en version originale "VO" ?
 3. Quels sont les titres des films réalisés par le réalisateur de 'Waterworld'?
 4. Quels sont les titres de films réalisés par 'Spielberg' non projetés actuellement ?
 5. Où peut-on voir un film dans lequel jouent 'Tom Cruise' et 'Nicole Kidman'?

Tâche 3:

- Exécuter des requêtes SQL pour répondre à ces questions:
 1. Quels sont les noms des acteurs qui jouent dans un film qu'ils produisent et réalisent ?
 2. Quels sont les titres de films dans lesquels 'Tom Cruise' n'a pas 'Nicole Kidman' comme partenaire?
 3. Quels producteurs ne produisent **aucun** film réalisé par 'Spielberg'?
 4. Quels sont les noms des acteurs qui jouent dans tous les films réalisés par 'Spielberg'?
 5. Quels acteurs ne jouent **que dans tous** les films réalisés par 'Spielberg' ?
 6. Quels sont les acteurs qui voient **tous** les films dans lesquels ils jouent ?
 7. Quels sont les acteurs **qui ne voient que tous** les films dans lesquels ils jouent?

Tâche 4: Consultation de quelques tables systèmes.

- Repérer dans les tables ci-dessous les informations intéressantes.

1. dba_tables

2. dba_constraints

3. dba_indexes

4. dba_users

TP N° 6 : Réalisation et interrogation de la base de données "Étudiants"

Après une étude conceptuelle concernant la gestion des étudiants, les tables ci-dessous ont été déduites. Les clés primaires sont soulignées et les clés étrangères sont désignées du signe *. Les nombres d'heures sont comptabilisés par semaine.

Etudiant (matricule_etu, nom_etu, prenom_etu, date_naissance)

Unité (code_Unité, libelle, nbr_heures, matricule_ens*)

Enseignant (matricule_ens, nom_ens, prenom_ens, âge)

EtudiantUnité (matricule_etu*, code_Unité*, note_CC, note_TP, note_examen)

- Sauvegarder toutes les instructions de ce TP dans un fichier nommé : TPscriptEtudiant.sql sous le répertoire c:/TPBDD

Tâche 1: Création des tables avec les contraintes d'intégrité et des indexes.

1. Créer l'utilisateur "BDDAdmin" avec le mot de passe "TPAdmin".
2. Accorder à "BDDAdmin" tous les privilèges.
3. Se connecter avec l'utilisateur "BDDAdmin".
4. Créer les tables précédentes avec les contraintes d'intégrités.
5. Créer un index de la table Etudiant sur l'attribut nom_etu dans l'ordre croissant.
6. Créer un index de la table Enseignant sur l'attribut nom_ens dans l'ordre décroissant.

Tâche 2: Création des utilisateurs

1. Créer l'utilisateur "Etudiant" avec le mot de passe "TPEtudiant".
2. Accorder à "Etudiant" uniquement le privilèges SELECT sur la table Etudiant.
3. Créer l'utilisateur "Enseignant" avec le mot de passe "TPEnseignant".
4. Accorder à "Enseignant" le privilèges SELECT et INSERT sur la table Enseignant.

Tâche 3: Modification du schéma et ajout de contraintes

1. Ajouter l'attribut Adresse à la table Etudiant de type chaînes de 100 caractères.
2. Supprimer l'attribut âge de la table Enseignant.
3. Exiger que tous les matricules des étudiant doivent être entre 20190000 et 20199999.
4. Augmenter le type de l'attribut prénom_etu de 5 caractères.

Tâche 4: Insertions et modifications des tuples.

1. Insérer les tuples ci-dessous.

Etudiant

matricule_etu	nom_etu	prenom_etu	date_naissance	Adresse
20190001	BOUSSAI	MOHAMED	12/01/2000	Alger
20190002	CHAID	LAMIA	01/10/1999	Batna
20190003	BRAHIMI	SOUAD	18/11/2000	Sétif
20190004	LAMA	SAID	23/05/1999	Oran

Enseignant

matricule_ens	nom_ens	prenom_ens
20000001	HAROUNI	AMINE
19990011	FATHI	OMAR
19980078	BOUZIDANE	FARAH
20170015	ARABI	ZOUBIDA

EtudiantUnite

Matricule_etu	code_Unite	note_CC	note_TP	note_examen
20190001	FEI0001	10	15	9
20190002	FEI0001	20	13	10
20190004	FEI0001	13	17	16
20190002	FEI0002	10	16	17
20190003	FEI0002	9	8	15
20190004	FEI0002	15	9	20
20190002	FEI0004	12	18	14
20190003	FEI0004	17	12	15
20190004	FEI0004	12	13	20

Unité

code_Unité	libelle	nbr_heures	matricule_ens
FEI0001	POO	6	20000001
FEI0002	BDD	6	19990011
FEI0003	RESEAU	3	20170015
FEI0004	SYSTEME	6	19980078

2. Augmenter la note_CC de 2 pour tous les étudiants dont le nom commence par 'B'.
3. Remettre toutes les notes d'examen de l'unité "SYSTEME" à 0 pour tous les étudiants.

Tâche 5: Interrogation de la base de données

1. Afficher les noms et prénoms des étudiants ayant obtenus des notes d'examens égales à 20.
2. Afficher les noms et prénoms des étudiants qui ne sont pas inscrits dans l'unité « POO ».
3. Afficher les libellés des unités d'enseignement dont aucun étudiant n'est inscrit.
4. Afficher les noms des étudiants qui ne sont inscrits que dans l'unité d'enseignement« POO ».

TP N° 7 : Les fonctions d'agrégat

Tâche 1:

- Lancer SQLPlus et ré-exécuter le script TPscriptEtudiant.sql
- Exécuter des requêtes SQL pour:
 1. Afficher pour chaque unité d'enseignement le nombre d'étudiants inscrits (les étudiants ayant été notés).
 2. Afficher, [pour chaque étudiant et pour chaque unité d'enseignement, sa moyenne (moyenne = $(\text{note de CC} + \text{note TP} + \text{note d'examen} * 2) / 4$)] , le matricule d'étudiant, le libellé de l'unité d'enseignement et sa moyenne.
 3. Afficher , pour chaque libellé d'unité d'enseignement, le matricule des étudiants ayant obtenus une moyenne supérieure à 15,
 4. Afficher , pour chaque libellé d'unité d'enseignement, le matricule des étudiants ayant obtenus une moyenne strictement inférieure à 10,
 5. Pour les unités d'enseignement ayant le plus grand nombre d'étudiants inscrits, afficher les libellés des unités et le nombre d'étudiants.

Afficher toutes les informations des étudiants inscrits dans le plus grand nombre d'unités d'enseignement et le nombre de ces d'unités.

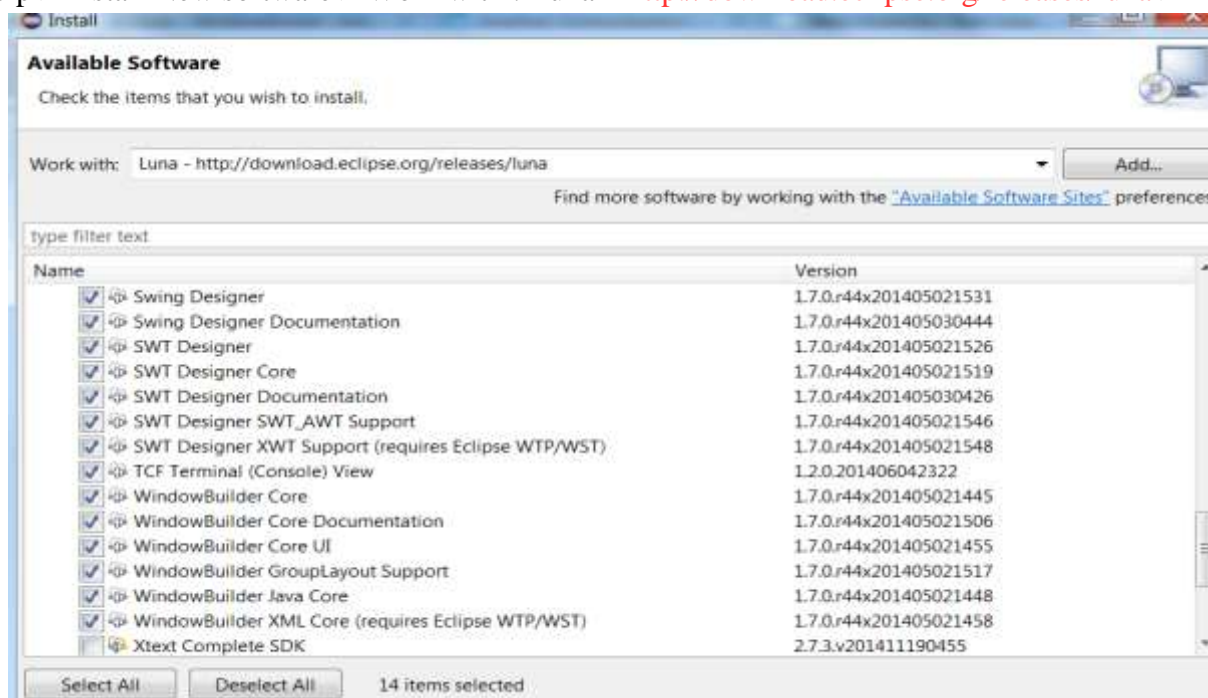
 6. Afficher toutes les informations des étudiants inscrits dans le moins d'unités d'enseignement et le nombre de ces d'unités.
 7. Afficher toutes les informations des étudiants et leurs moyennes générale.

TP N° 8 : Création et manipulation des vues

TP N° 9 : Eclipse et WindowBuilder

Tâche 1: Installation et premiers pas sur WindowBuilder d'Eclipse

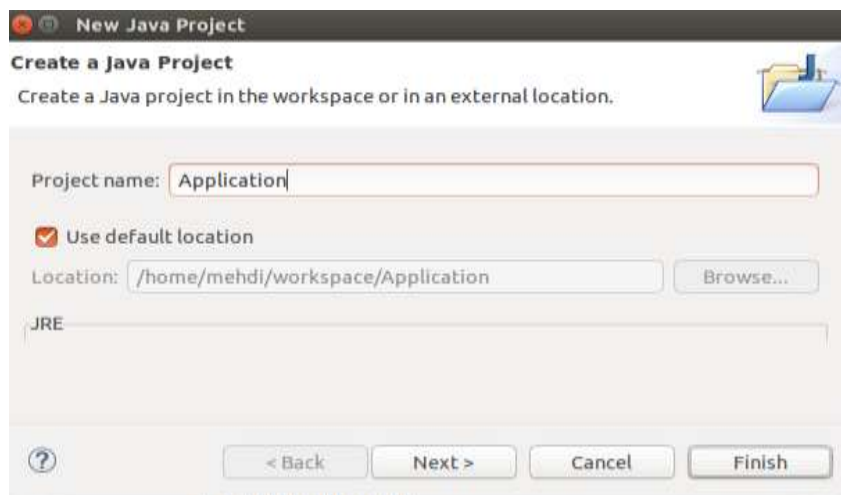
Help > Install New software > Work with : Luna – <http://download.eclipse.org/releases/luna> >



cocher tout entre *swing designer* et *swt windowbuilder* (voir ci-dessous) > Next > Finish.

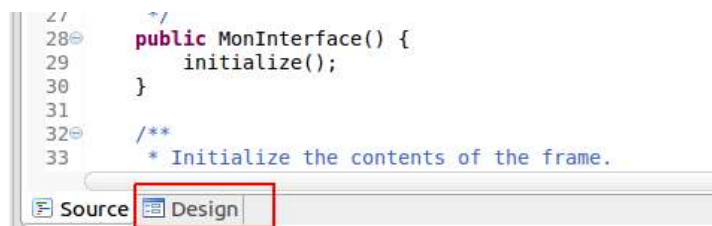
Tâche 2: Création du Projet Java

File → New Java Project → Nom de projet : Application → Finish



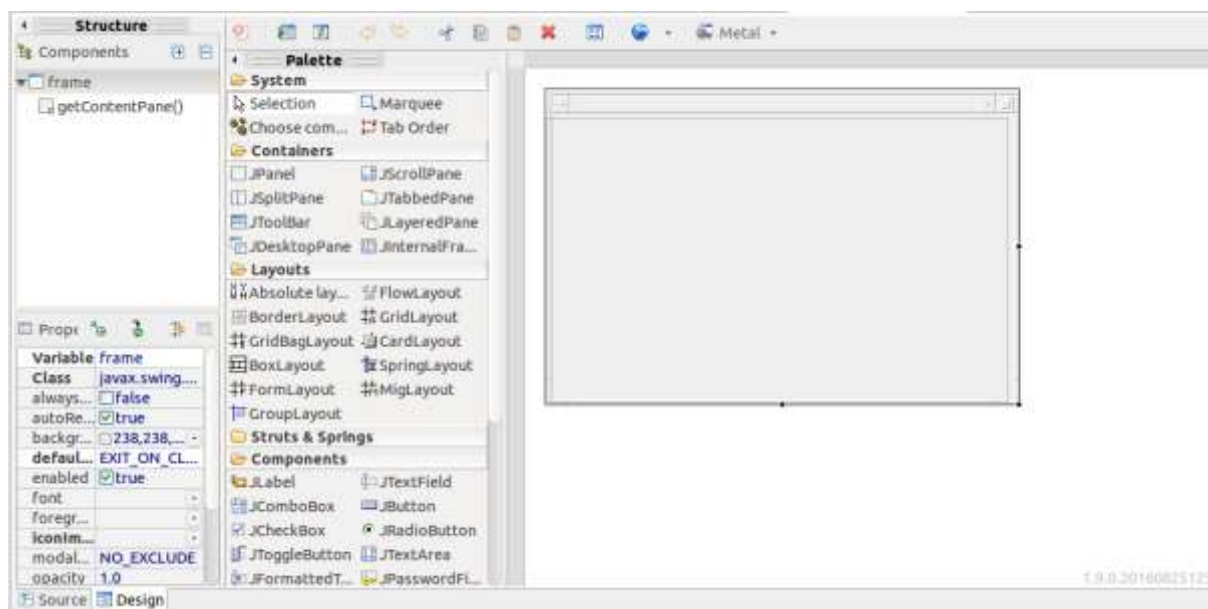
Tâche 3: Création d'une application sous WindowBuilder

Clic droit sur le Project Application : New → other → windowbuilder → SWING Designer → application window → Name : monInterface . En bas à gauche cliquer sur l'onglet design qui apparait à côté de source (cadre rouge ci-dessous).




L'interface de l'application WindowBuilder (dans la figure ci-dessous) est composée de :

1. Frame : C'est la fenetre principale qui regroupe tous les composants. Elle est générée automatiquement à la création du projet.
2. Composants : c'est les éléments qui constituent notre application telles que les boutons, les labels, les zones de textes, les barres de menu, etc.
3. Palette des composants: regroupe l'ensemble des composants offerts par WindowBuilder regroupés suivants leurs fonctionnalités.
4. Propriété: Représente les caractéristiques des composants telles que le titre, la couleur, la taille, etc.
5. Source : Permet de visualiser le code source de notre application. Ce code source est généré automatiquement à la création ou à la modification des composants dans l'interface graphique.

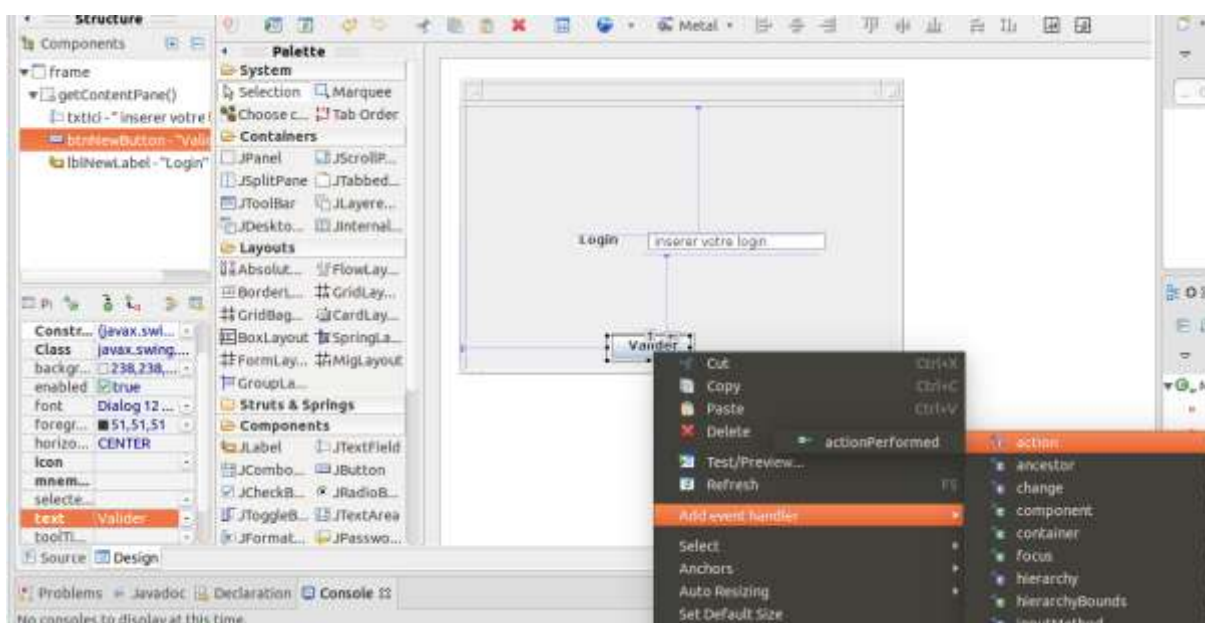


Tâche 4: Manipulation des composants: *JLabel*, *TextField* et *Button*

- Glisser le composant *JLabel* dans la fenêtre principale (frame).
- Dans la partie propriété, observer les différents caractéristiques du composant.
- Changer le texte de votre label et donner un nom à votre variable.
- Glisser le composant *TextField* dans la fenêtre principale (frame).
- Dans la partie propriété, observer les différents caractéristiques du composant.
- Ecrire « inserer votre login» dans la propriété texte de votre zone de texte et donner un nom à votre variable (exemple *monLogin*).
- Glisser le composant *JButton* dans la fenêtre principale (frame).
- Dans la partie propriété, observer les différents caractéristiques du composant.
- Ecrire «Valider» dans la propriété texte de votre bouton donner un nom à votre variable.
- Cliquer sur l'icône 

Tâche 5: Les actions

- Ajouter une action à votre bouton. Cliquez droit sur le bouton → add event handler → action → actionPerformed. Voir la figure ci-dessous.



```
JButton btnNewButton = new JButton("Valider");  
btnNewButton.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
    }  
});
```

- Une fois l'action est ajoutée, une fonction *actionPerformed* est inséré dans le code source (à vérifier).
- Dans cette fonction, nous allons définir l'événement à exécuter lorsqu'on clique sur le bouton.

```
public void actionPerformed(ActionEvent e) {  
    String login = monLogin.getText(); //monLogin est le nom de la variable JTextField  
    System.out.println("Le texte que vous avez inserer est "+login);  
}
```

- Afficher un message en cliquant sur le bouton

```
public void actionPerformed(ActionEvent e) {  
    System.out.println("Vous avez cliquer sur le bouton");  
}
```

- Afficher le message inséré dans JTextField en cliquant sur le bouton

Tâche 6: Manipulation du composant *JRadioButton*

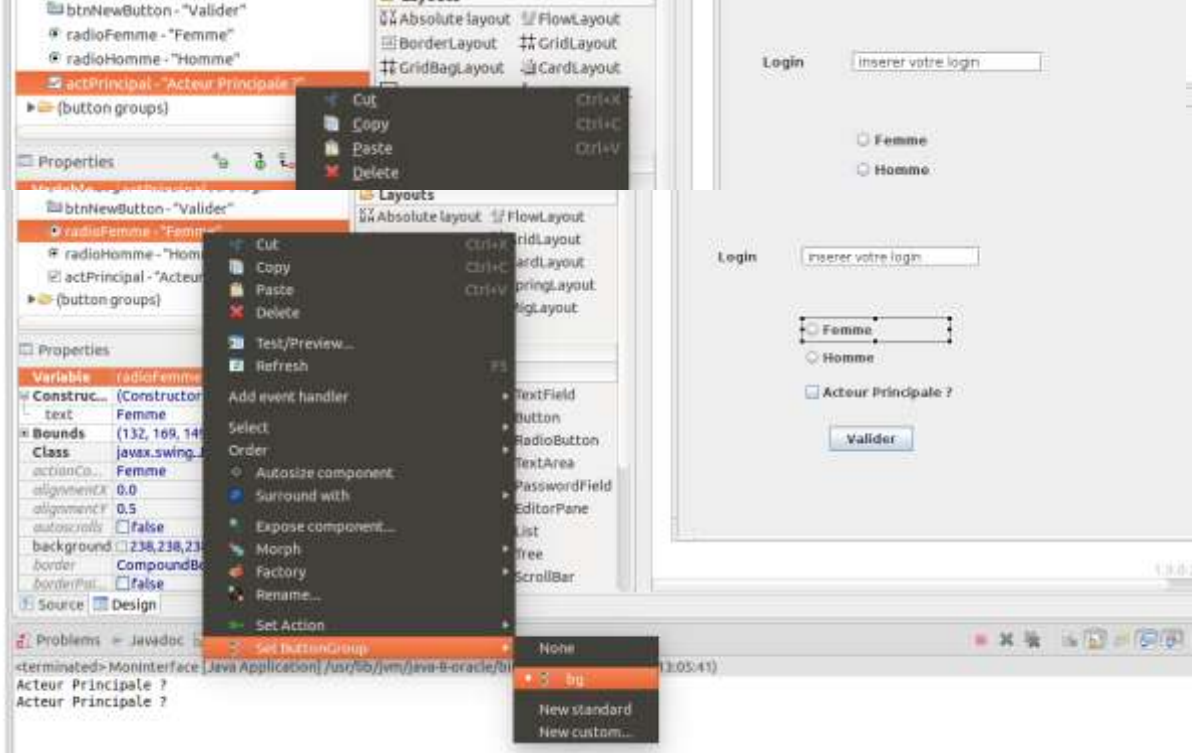
- Glisser le composant *JRadioButton* dans la fenêtre principale.
- Dans la partie propriété, observer les différents caractéristiques du composant.
- Changer le texte de votre *JRadio* et donner un nom à votre variable.
- Glisser deux composant *JRadioButton* dans la fenêtre principale.
- Changer le texte de votre *JRadio* et donner un nom à vos variables (radioFemme et radioHomme). Vous remarquez qu'on peut choisir les deux *JRadio* à la fois ! Pour pouvoir choisir une option, et UNE SEULE, vous devriez mettre les deux Jradio dans le même groupe.

Tâche 7: Manipulation du composant Groupe JRadio

- Dans source, créer la variable suivante.

```
private ButtonGroup bg= new ButtonGroup();
```

- Ajouter ensuite, les deux JRadios dans ce groupe comme suit :



```
public void actionPerformed(ActionEvent e) {
    Enumeration<AbstractButton> buttons = bg.getElements();
    while (buttons.hasMoreElements()){
        AbstractButton button = buttons.nextElement();
        if (button.isSelected())
            System.out.println(button.getText());
    }
}
```

- Tester votre Application.

Tâche 8: Manipulation du composant CheckBox

- Glisser le composant *JCheckBox* dans la fenetre principale (frame).
- Dans la partie propriété, observer les différents caractéristiques du composant.
- Changer le texte de votre *JCheckBox* et donner un nom à votre variable (actPrincipal).
- Rendre la variable *actPrincipal* une *variable de classe*
- Vérifier dans source que la variable de classe est bien créée

```
private JCheckBox actPrincipal;
```

- Afficher le texte de votre JcheckBox à l'exécution de l'événement du bouton

```
public void actionPerformed(ActionEvent e) {
    if(actPrincipal.isSelected())
    {
        System.out.println(actPrincipal.getText());
    }
}
```

TP N° 10 : Connectivité avec la base de données Oracle à travers Eclipse

Tâche 1: Connexion avec Oracle

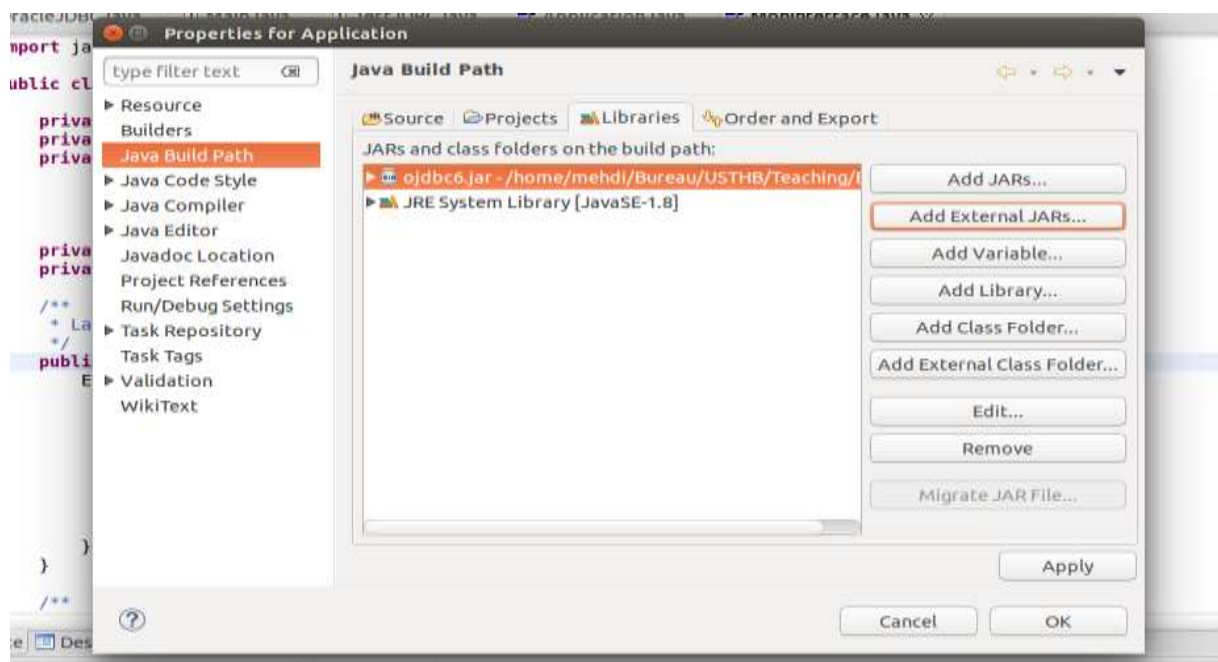
- Télécharger les deux bibliothèques: *ojdbc6.jar* pour la connexion et *rs2xml.jar* pour l'affichage du résultat.
- Déclarer les trois variables (Connection, Statement et ResultSet) dans source de la Class *MonInterface*
- Ajouter le code source suivant dans la méthode initialize.

```
--  
19 public class MonInterface {  
20  
21     private Connection connection=null;  
22     private Statement stmt = null;  
23     private ResultSet rs = null;  
--
```

- Remplacer le Login et le Password par vos login et mot de passe de Oracle

```
private void initialize() {  
    frame = new JFrame();  
    frame.getContentPane().setEnabled(false);  
  
    try {  
        Class.forName("oracle.jdbc.driver.OracleDriver");  
        connection = DriverManager.getConnection("jdbc:oracle:thin:LOGIN/PASSWORD@localhost");  
        stmt = connection.createStatement();  
    } catch (Exception e) {  
        e.printStackTrace();  
    }  
}
```

- Ajouter les librairies *ojdbc6* à votre projet. Cliquez droit sur votre projet → Build path → Configure Build Path → L'ongle Libraries → Add external Jars → Chercher le *ojdbc6.jar* → OK



Tâche 2: Exécution des requêtes

- L'exécution d'une requête se fait avec la méthode *executeQuery* lors de l'exécution de l'action déclenchée par le clique sur le bouton.

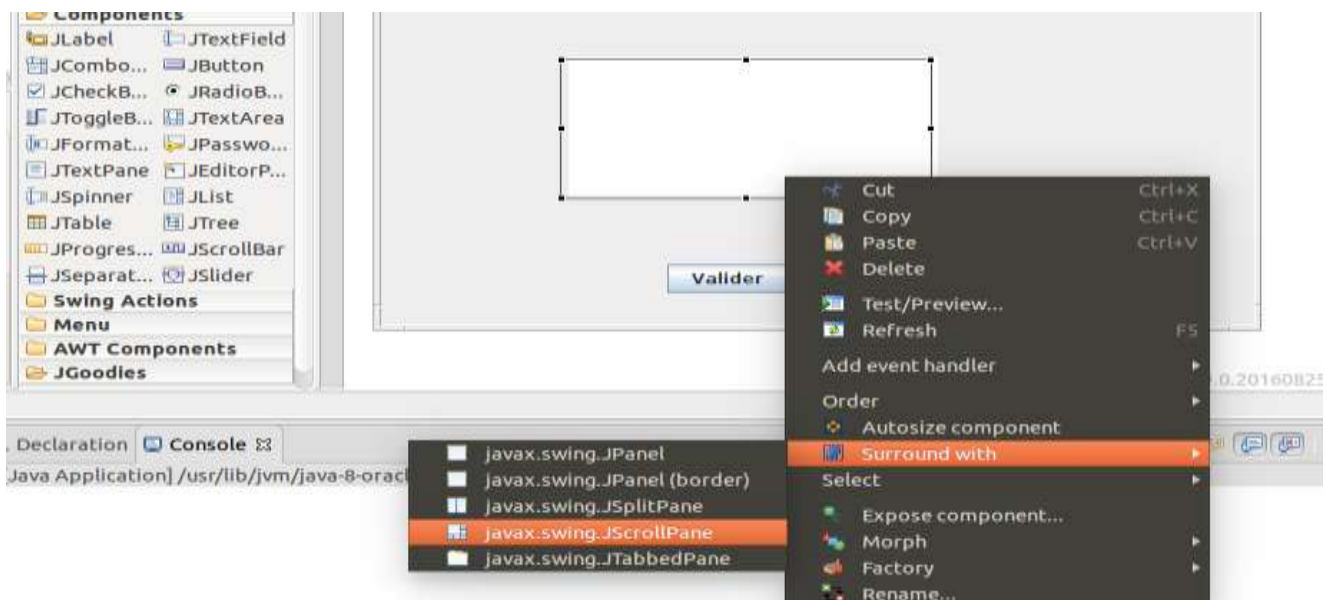
```
public void actionPerformed(ActionEvent e) {

    rs = stmt.executeQuery("SELECT * FROM film");
```

- L'exécution de la requête retourne un résultat de type *ResultSet*

Tâche 3: Afficher le résultat d'une requête

- Ajouter le composant *JTable* à partir de la palette dans votre interface graphique.
- Mettre le composant *JTable* dans le conteneur *JScrollPane*.
- Clique droit sur le *JTable* → Surround with → *JScrollPane*.



- Ajouter le résultat de la requête dans *JTable*.
- Ajouter les librairies *rs2xml* à votre projet. Cliquez droit sur votre projet → Build path → Configure Build Path → L'onglet Libraries → Add external Jars → Chercher le *rs2xml.jar*

```
public void actionPerformed(ActionEvent e) {
    try {
        rs = stmt.executeQuery("SELECT * FROM film");
        table.setModel(DbUtils.resultSetToTableModel(rs));
```

→ OK

Tâche 4:

- Créer l'interface suivante :



The image shows a Java Swing window with a title bar containing standard OS window controls (close, maximize, minimize). The window has a light gray background. Inside, the label 'Requete' is positioned to the left of a text input field. The input field contains the placeholder text 'Mettez votre requete SQL'. Below the input field is a large, empty rectangular area, likely intended for displaying the query results. At the bottom center of the window is a button labeled 'Valider'.

- Ecrire la requête SQL dans votre JTextField
- Afficher le résultat de la requête votre JTabl

TP N° 11 : Projet final

Tâche 1:

Tâche 2:

Annexe A : Langage de Contrôle de Données

Création d'utilisateur

```
CREATE USER nom_d'utilisateur IDENTIFIED BY mot_de_passe ;
```

Exemple

```
CREATE USER utilisateur1 IDENTIFIED BY tp1 ;
```

- **Remarque importante**

Le mot de passe et le *nom_d'utilisateur* doivent commencer par un caractère alphabétique.

Modification de mot de passe

```
ALTER USER nom_d'utilisateur IDENTIFIED BY mot_de_passe ;
```

Exemple :

```
ALTER USER utilisateur1 IDENTIFIED BY test ;
```

Suppression d'utilisateur

```
DROP USER nom_d'utilisateur ;
```

Exemple

```
DROP USER utilisateur1 ;
```

Transmission de privilèges

Lorsqu'un utilisateur crée une table ou une vue, il en est le propriétaire exclusif. Les autres utilisateurs n'y ont pas accès. Il peut cependant en donner un droit d'utilisation à d'autres par la commande GRANT.

```
GRANT privilège1 , privilège2, ... | ALL PRIVILEGES  
[ON table/vue ]  
TO utilisateur1 , utilisateur2 , ... | PUBLIC  
[ WITH GRANT OPTION ] ;
```

Cette instruction accorde des privilèges d'accès « *privilège i* » ou tous les privilèges (ALL PRIVILEGES) sur la *table* ou *vue* aux utilisateurs « *utilisateur i* » ou à tous les utilisateurs (PUBLIC).

La clause WITH GRANT OPTION permet aux utilisateurs ayant reçu les privilèges de les transmettre à leur tour à d'autres utilisateurs.

Exemple

```
GRANT ALL PRIVILEGES TO utilisateur1 ;
```

L'utilisateur « utilisateur1 » peut tout faire, il peut même accéder aux tables des autres utilisateurs.

- **Exemples de Privilèges :**

SELECT : lecture, INSERT : insertion, UPDATE : mise à jour, DELETE : suppression, ALL : tous les privilèges, ALTER : destruction, INDEX : construction d'index

Exemples :

- Pour accorder à tous un droit de lecture sur une table T d'un utilisateur U, le propriétaire U accordera les droits nécessaires.

```
GRANT SELECT ON nom_table TO PUBLIC;
```

- Si on veut accorder à l' *utilisateur1* un droit de lecture sur une table T d'un utilisateur U, le propriétaire U accordera les droits nécessaires.

```
GRANT SELECT ON film TO utilisateur1;  
GRANT SELECT, INSERT ON seance TO utilisateur1;
```

- **Exemples de privilèges systeme :**

CREATE SESSION : “ce privilège est utilisé pour ouvrir une session sous SQL*PLUS (permet donc d'ouvrir une fenêtre SQL*Plus et de se connecter à Oracle”.

CREATE TABLE, CREATE USER, CREATE VIEW, CREATE SESSION

Exemple

```
GRANT CREATE SESSION TO utilisateur3;
```

Suppression de privilèges

```
REVOKE privilège ON table|vue FROM user|PUBLIC;
```

Annexe B : Langage de Définition de Données

1. Les commentaires

- Pour insérer des commentaires: il faut les précéder par : /* ... */. Exemple :

```
/* commentaire TP BDD */
```

2. Les symboles [] et |

- Les crochets [] indiquent que la commande est optionnelle.

Exemple

L'instruction :

```
CREATE [UNIQUE] INDEX nom_index ON nom_table (attribut [ASC|DESC], ...);
```

Sous SQLPLUS avec le mot clé UNIQUE :

```
CREATE UNIQUE INDEX nom_idx ON nom_table ( nom_etudiant ASC );
```

Ou bien sous SQLPLUS sans le mot clé UNIQUE :

```
CREATE INDEX nom_idx ON nom_table (nom_etudiant DESC);
```

- Ce symbole « | » indique le choix.

Exemple

```
CREATE INDEX nom_index ON nom_table (attribut [ASC | DESC], ...);
```

Sous SQLPLUS : dans l'ordre ASC

```
CREATE INDEX nom_index ON nom_table (attribut ASC);
```

Ou bien l'ordre DESC

```
CREATE INDEX nom_index ON nom_table (attribut DESC);
```

3. Création d'une table

```
CREATE TABLE nom_table ( Liste d'attributs suivis de leurs types séparer par des virgules );
```

Type des données :

- NUMBER(n) : Entier à n chiffres
- NUMBER(n, m) : Réel à n chiffres au total (virgule comprise), m après la virgule
- INTEGER
- VARCHAR2(n) : Chaîne de n caractères (entre ' ')
- DATE : Date au format 'JJ-MM-AAAA'

- Un attribut déclaré "NOT NULL" doit nécessairement être renseigné (rempli) lors de l'insertion d'un tuple.
- Ne pas mettre de virgule avant la dernière parenthèse fermante de l'instruction CREATE.

Exemple

```
CREATE TABLE JOUE (
  NOM_ACTEUR    VARCHAR2(30)      NOT NULL ,
  TITRE         VARCHAR2(40)      NOT NULL
);
```

4. Définition de la contrainte "CHECK"

Cette contrainte est utilisée pour limiter les valeurs qu'un attribut peut contenir. Si l'on définit la contrainte CHECK sur un attribut seules certaines valeurs seront autorisées pour cet attribut. Si peut également définir la contrainte CHECK pour limiter les valeurs d'un attribut en se basant sur les valeurs d'un autre attribut.

Exemple

```
CREATE TABLE Films (
  TITRE          VARCHAR2(40)      NOT NULL ,
  DUREE          INTEGER CHECK ( DUREE>0 and DUREE <10) ,
  NATIONALITE    VARCHAR2(30) ,
  NOM_REALISATEUR VARCHAR2(30) CHECK (NOM_REALISATEUR LIKE '%r')
);
```

5. Définition de la contrainte "DEFAULT"

La contrainte DEFAULT est utilisée pour insérer une valeur par défaut dans une colonne. La valeur par défaut sera ajoutée à tous les nouveaux tuples, si aucune autre valeur n'est spécifiée.

Exemple

```
CREATE TABLE Films (
  TITRE          VARCHAR2(40)      NOT NULL ,
  DUREE          INTEGER CHECK ( DUREE>0 and DUREE <10) ,
  NATIONALITE    VARCHAR2(30) DEFAULT 'Algérienne',
  NOM_REALISATEUR VARCHAR2(30) check (NOM_REALISATEUR LIKE '%r')
);
```

6. Définitions des contraintes d'intégrité: Clé primaire

Il est préférable de nommer de la contrainte clé primaire de cette façon: PK_ *nom_table*

```
CONSTRAINT nom_contrainte PRIMARY KEY (attribut_clé [, attribut_clé2, ...])
```

7. Définitions des contraintes d'intégrité: Clé étrangère

Il est préférable de nommer la contrainte clé étrangère de cette façon: FK_nom_table1_nom_table2

```
CONSTRAINT nom_contrainte FOREIGN KEY (attributs) REFERENCES table (attribut)
```

De plus, pour supprimer un tuple et que sa suppression se répercute dans les autres table liées par "REFERENCES" alors il faut ajouter "ON DELETE CASCADE". Avec cette fonction la suppression se fera dans toute les tables liées.

```
CONSTRAINT nom_contrainte FOREIGN KEY (attributs) REFERENCES table (attribut) ON DELETE CASCADE
```

L'instruction de création de table devient alors:

```
CREATE TABLE nom_table1 ( liste d'attributs avec leurs types séparer par des virgules
CONSTRAINT nom_contrainte PRIMARY KEY (liste1 d'attributs) ,
CONSTRAINT nom_contrainte FOREIGN KEY (liste2 d'attributs) REFERENCES
nom_table2 (liste2 d'attributs) ) ;
```

Exemple

```
CREATE TABLE JOUE (
  NOM_ACTEUR      VARCHAR2(30)          NOT NULL ,
  TITRE           VARCHAR2(40)          NOT NULL ,
  CONSTRAINT PK_JOUE PRIMARY KEY ( NOM_ACTEUR , TITRE ) ,
  CONSTRAINT FK_JOUE_FILM FOREIGN KEY (TITRE) REFERENCES FILM (TITRE)
);
```

8. Synonyme d'une table

- Création d'un Synonyme d'une table

```
CREATE PUBLIC SYNONYM le_nom_du_synonyme FOR nom_table ;
```

Exemple

```
CREATE PUBLIC SYNONYM toto FOR system.film;
```

```
SELECT * FROM toto;
/*****      identique à *****/
SELECT * FROM system.film ;
```

Exemple

```
GRANT SELECT ON system.film TO PUBLIC;  
/*** ou bien *****/  
GRANT SELECT ON toto TO PUBLIC;
```

9. Ajout d'attributs dans une table déjà créée

```
ALTER TABLE nom_table ADD (attribut TYPE, ...);
```

10. Modifications du type d'un attribut existant

```
ALTER TABLE nom_table MODIFY (attribut TYPE, ...);
```

11. Ajout de contrainte

```
ALTER TABLE nom_table ADD CONSTRAINT nom_contrainte ;
```

Exemple

```
ALTER TABLE SEANCE ADD CONSTRAINT VERIFIER_titre CHECK ( titre like  
'%R');
```

12. Suppression de contraintes

```
ALTER TABLE nom_table DROP CONSTRAINT nom_contrainte ;
```

13. Description d'une table créée

- Utiliser DESC pour voir la description d'une table déjà créée.

```
DESC nom_table ;
```

Exemple

```
DESC film ;
```

14. Création d'un index

```
CREATE [ UNIQUE ] INDEX nom_index ON nom_table (attribut [ASC|DESC], ...);
```

Tel que : UNIQUE → pas de double
ASC/DESC → ordre croissant ou décroissant

Exemple

```
CREATE INDEX JOUER_FK ON joue ( TITRE ASC );
```

15. Suppression d'index ou de table

- Pour supprimer une table

```
DROP TABLE nom_table ;
```

- **Pour supprimer un index**

```
DROP INDEX nom_index ;
```

16. Ajout de tuples

```
INSERT INTO nom_table VALUES (valeur_de_attribut1, valeur_de_attribut2, ...);
```

```
COMMIT ; /*validation des insertions*/
```

- Il faut mettre les chaînes de caractères entre côtes exemple 'test BDD' .
- Utiliser le mot clé **NULL** pour laisser un attribut vide.
- Il ne faut pas oublier de valider les insertions en exécutant la commande « *commit* »
- Après chaque exécution l'instruction INSERT une réponse de SQL*Plus est affichée
 - **SQL> 1 Ligne créée**
- Après exécution de l'instruction COMMIT une réponse de SQL*Plus est affichée
 - **SQL> validation effectuée**

17. Mise à jour d'un attribut

```
UPDATE nom_table SET attribut = valeur [ WHERE condition ];
```

Exemple

```
UPDATE film SET duree = 60 WHERE titre = 'Waterworld';
```

18. Suppression de tuples

```
DELETE FROM nom_table [ WHERE condition ];
```

Exemples

```
DELETE FROM film WHERE titre = 'waterworld';
```

```
DELETE FROM film ;
```

19. Les Vues

Une vue est une table virtuelle calculée à partir d'autres tables grâce à une requête. L'intérêt des vues est de :

- Simplifier l'accès aux données en masquant les opérations de jointure.
- Sauvegarder des requêtes complexes.

- Présenter de mêmes données sous différentes formes adaptées aux différents usagers particuliers. Renforcer de la sécurité des données par masquage des lignes et des colonnes sensibles aux usagers non habilités

20. Définition d'une vue

```
CREATE VIEW nom_vue AS requête;
```

21. Problèmes de mise à jour, restrictions

La mise à jour de données via une vue pose des problèmes et la plupart des systèmes impose d'importantes restrictions.

- Le mot clé DISTINCT doit être absent.
- La clause FROM doit faire référence à une seule table.
- La clause SELECT doit faire référence directement aux attributs de la table concernée (pas d'attribut dérivé).
- Les clauses GROUP BY et HAVING sont interdites.

22. Catalogue du système

Contient sous forme relationnelle la définition de tous les objets créés par le système et les usagers. Ces tables sont accessibles avec SQL (en mode consultation uniquement).

```
USER_CATALOG (TABLE_NAME , TABLE_TYPE)
USER_TAB_COLUMNS (TABLE_NAME, COLUMN_NAME, ...)
USER_IND_COLUMNS (INDEX_NAME, TABLE_NAME, COLUMN_NAME, ...)
ALL_TABLES (TABLE_NAME, OWNER, ...)
ALL_USERS (USERNAME , USER_ID , CREATED)
```

Annexe C : Langage de Manipulation de Données

Après la création des tables, il est alors possible de récupérer les données stockées grâce à la commande SELECT :

```
SELECT [ALL|DISTINCT] attribut(s) FROM nom_table [ WHERE condition]
      [GROUP BY attribut(s) [HAVING condition]]
      [ORDER BY attribut(s) [ASC|DESC]] ;
```

- **Pour afficher tous les attributs et les tuples d'une table**

```
SELECT * FROM nom_table ;
```

Exemple : afficher tout le contenu de la table film

```
SELECT * FROM film;
```

Exemple : afficher deux attributs de la table film

```
SELECT titre , nationalite FROM film;
```

- **Pour ordonner les tuples**

- ORDER BY attribut ASC : ordre ascendant
- ORDER BY attribut DESC : ordre descendant.

Exemple

```
SELECT * FROM Film
ORDER BY nationalite ASC;
```

- **Pour afficher un ensemble d'attribut d'une table (Projection)**

Exemple

```
SELECT titre , nationalite FROM film;
```

- **Pour afficher un ensemble de tuples vérifiant une condition parmi les tuples d'une table. (Restriction)**

Exemple

```
SELECT * FROM film WHERE nationalite = 'US' ;
```

Dans le "WHERE" on peut :

1. Employer les opérateurs de comparaison (>, <, ≤, ≥, =) ;

2. Vérifier l'appartenance d'un attribut à un intervalle

```
WHERE attribut BETWEEN valeur1 AND valeur2 ;
```

3. Vérifier qu'un attribut possède comme valeur une chaîne de caractère qui se termine par un ensemble de caractère précis (chaîne) :

```
WHERE attribut LIKE '%chaîne'
```

4. Vérifier qu'un attribut commence par un ensemble de caractère précis (chaîne)

```
WHERE attribut LIKE 'chaîne%'
```

5. Vérifier qu'un attribut contient une chaîne de caractère (chaîne) :

```
WHERE attribut LIKE '%chaîne%'
```

6. Préciser qu'un attribut appartient à un ensemble

Exemple : vérifier si la valeur de l'attribut est égale à x, y ou z

```
WHERE attribut IN (x, y , z)
```

7. Utiliser la négation pour tous ces prédicats : NOT BETWEEN, NOT NULL, NOT LIKE, NOT IN.

```
WHERE attribut NOT LIKE '%chaîne%'
```

• **Fonctions d'agrégat** : Elles opèrent sur un ensemble de valeurs.

- AVG(), VARIANCE(), STDDEV() : moyenne, variance et écart-type des valeurs ;
- SUM() : somme des valeurs ;
- MIN(), MAX() : valeur minimum, valeur maximum ;
- COUNT() : nombre de valeurs.

Annexe D: Fichiers TPOracleScript.sql et donnees.txt**Fichiers TPOracleScript.sql**

```
*=====*/  
/* Database name: TP BD_CINEMA */  
/*=====*/
```

```
drop index AIMER_FK  
/
```

```
drop index JOUER_FK  
/
```

```
drop index PRODUIR_FK  
/
```

```
drop index PROJETER_FK  
/
```

```
drop index VOIR_FK  
/
```

```
drop table AIME cascade constraints  
/
```

```
drop table VU cascade constraints  
/
```

```
drop table SEANCE cascade constraints  
/
```

```
drop table PRODUIT cascade constraints  
/
```

```
drop table JOUE cascade constraints  
/
```

```
drop table FILM cascade constraints  
/
```

```
/*=====*/
/* Table: FILM */
/*=====*/
create table FILM (
  TITRE          VARCHAR2(40)          not null,
  DUREE          INTEGER,
  NATIONALITE     VARCHAR2(30),
  NOM_REALISATEUR VARCHAR2(30),
  constraint PK_FILM primary key (TITRE)
)
/
```

```
/*=====*/
/* Table: JOUE */
/*=====*/
create table JOUE (
  NOM_ACTEUR      VARCHAR2(30)          not null,
  TITRE           VARCHAR2(40)          not null,
  constraint PK_JOUE primary key (NOM_ACTEUR, TITRE),
  constraint FK_JOUE_JOUER_FILM foreign key (TITRE)
    references FILM (TITRE)
)
/
```

```
/*=====*/
/* Index: JOUER_FK */
/*=====*/
create index JOUER_FK on JOUE (
  TITRE ASC
)
/
```

```
/*=====*/
/* Table: PRODUIT */
/*=====*/
create table PRODUIT (
  NOM_PRODUCTEUR  VARCHAR2(30)          not null,
  TITRE           VARCHAR2(40)          not null,
  constraint PK_PRODUIT primary key (NOM_PRODUCTEUR, TITRE),
  constraint FK_PRODUIT_PRODUIR_FILM foreign key (TITRE)
    references FILM (TITRE)
)
/
```

```
/*=====*/
/* Index: PRODUIR_FK */
/*=====*/
create index PRODUIR_FK on PRODUIT ( TITRE ASC
)
/
```

Fichier donnees.txt

```
INSERT INTO Film VALUES ('Body Guard', 120, 'US', 'Mark Gordon');
INSERT INTO Film VALUES ('Les Incorruptibles', 160, 'US', 'Rickley Moore');
INSERT INTO Film VALUES ('Perfect World', 150, 'US', 'Clint Eastwood');
INSERT INTO Film VALUES ('Le Dernier Chateau', 132, 'US', 'Rod Lurie');
INSERT INTO Film VALUES ('Le Negociateur', 140, 'US', 'David Auberman');
INSERT INTO Film VALUES ('Le Destin De Will Hunting', 120, 'US', 'Gus Van Sant');
INSERT INTO Film VALUES ('High Crimes', 115, 'US', 'Carl Francklin');
INSERT INTO Film VALUES ('Le Parrain', 168, 'US', 'Francis Ford Coppola');
INSERT INTO Film VALUES ('Spy Games', 127, 'US', 'Tony Scott');
INSERT INTO Film VALUES ('Nous Etions Soldat', 139, 'US', 'Randall Wallace');
INSERT INTO Film VALUES ('Top Gun', 127, 'US', 'Tony Scott');
INSERT INTO Film VALUES ('Men', 140, 'US', 'Rickley Moore');
INSERT INTO Film VALUES ('Le Pacificateur', 135, 'US', 'Ashley Jude');
INSERT INTO Film VALUES ('Il Faut Sauver Le soldat Ryan', 162, 'US', 'Spielberg');
INSERT INTO Film VALUES ('ET', 165, 'US', 'Spielberg');
INSERT INTO Film VALUES ('Indiana Jones', 170, 'US', 'Spielberg');
```

```
INSERT INTO Joue VALUES ('Kevin Costner', 'Perfect World');
INSERT INTO Joue VALUES ('Kevin Costner', 'Les Incorruptibles');
INSERT INTO Joue VALUES ('Kevin Costner', 'Body Guard');
INSERT INTO Joue VALUES ('Clint Eastwood', 'Perfect World');
INSERT INTO Joue VALUES ('Tom Cruise', 'Top Gun');
INSERT INTO Joue VALUES ('Tom Cruise', 'Men');
INSERT INTO Joue VALUES ('Nicole Kidman', 'Men');
INSERT INTO Joue VALUES ('Nicole Kidman', 'Le Pacificateur');
INSERT INTO Joue VALUES ('Harrison Ford', 'Indiana Jones');
INSERT INTO Joue VALUES ('Tom Hanks', 'Il Faut Sauver Le soldat Ryan');
INSERT INTO Joue VALUES ('Aaron Smith', 'Indiana Jones');
INSERT INTO Joue VALUES ('Aaron Smith', 'Il Faut Sauver Le soldat Ryan');
INSERT INTO Joue VALUES ('Aaron Smith', 'ET');
INSERT INTO Joue VALUES ('Amanda Peet', 'Indiana Jones');
INSERT INTO Joue VALUES ('Amanda Peet', 'Il Faut Sauver Le soldat Ryan');
INSERT INTO Joue VALUES ('Amanda Peet', 'ET');
INSERT INTO Joue VALUES ('Amanda Peet', 'Top Gun');
```

```
INSERT INTO Produit VALUES ('Clint Eastwood', 'Perfect World');
INSERT INTO Produit VALUES ('Adam Moore', 'Men');
INSERT INTO Produit VALUES ('Alex May', 'Top Gun');
INSERT INTO Produit VALUES ('Roger Taylor', 'Indiana Jones');
```

```
INSERT INTO Seance VALUES ('Indiana Jones', 'Alpha', '13:00', 'VF');
INSERT INTO Seance VALUES ('Body Guard', 'Alpha', '21:00', 'VO');
INSERT INTO Seance VALUES ('Les Incorruptibles', 'Ibn Zeidoun', '13:00', 'VO');
INSERT INTO Seance VALUES ('ET', 'Ibn Zeidoun', '11:00', 'VO');
INSERT INTO Seance VALUES ('Top Gun', 'Ibn Zeidoun', '15:00', 'VO');
INSERT INTO Seance VALUES ('Top Gun', 'Alpha', '09:00', 'VF');
```

```
INSERT INTO Seance VALUES ('Men', 'Beta', '11:00', 'VF');  
INSERT INTO Seance VALUES ('Le Pacificateur', 'Beta', '23:00', 'VF');
```

```
INSERT INTO Vu VALUES ('Kevin Costner', 'Perfect World');  
INSERT INTO Vu VALUES ('Kevin Costner', 'Les Incorruptibles');  
INSERT INTO Vu VALUES ('Kevin Costner', 'Body Guard');  
INSERT INTO Vu VALUES ('Kevin Costner', 'ET');  
INSERT INTO Vu VALUES ('Tom Cruise', 'Top Gun');  
INSERT INTO Vu VALUES ('Tom Cruise', 'Men');  
INSERT INTO Vu VALUES ('Tom Hanks', 'Men');
```

Annexe E: Solutions des TPs

Solution TP 1: L'éditeur de commandes SQL

Tâche 3: Exécution d'un script SQL sous SQLPlus

5. Exécuter le script "TPOracleScript.sql".

```
SQL> @ c:/TPBDD/TPOracleScript.sql
```

Tâche 4: Sauvegarde de la session de travail

4. Que contient le fichier sauvegarde.LST.

```
SQL> spool off
```

5. Relancer le spool.

```
SPOOL C:/TPBDD/sauvegarde
```

6. Relancer l'exécution du script "TPOracleScript.sql".

```
SQL> @ c:/TPBDD/TPOracleScript.sql
```

7. Arrêter la sauvegarde.

```
spool off
```

9. Cette fois le fichier sauvegarde.LST contient tous les messages d'exécution du fichier TPOracleScript.sql.

Solution TP 2: Création des tables et des indexes

Tâche 1: Analyse du fichier TPOracleScript.sql

6. Lancer un spool dans le répertoire c:/TPBDD.

```
SQL> SPOOL C:/TPBDD/sauvegarde2
```

7. SQL> @ C:/TPBDD//TPOracleScript.sql

8. Ces instructions sont des LDD.

9. Les attributs soulignés ci-dessous sont des clés primaires et les clés étrangères sont identifiées par le signe *.

```
SQL> DESC FILM;
```

Name	Null?	Type
<u>TITRE</u>	NOT NULL	VARCHAR2(40)
DUREE		NUMBER(38)
NATIONALITE		VARCHAR2(30)
NOM_REALISATEUR		VARCHAR2(30)

```
SQL> DESC JOUE;
```

Name	Null?	Type
<u>NOM_ACTEUR</u>	NOT NULL	VARCHAR2(30)
<u>TITRE</u> *	NOT NULL	VARCHAR2(40)

```
SQL> DESC PRODUIT;
```

Name	Null?	Type
<u>NOM_PRODUCTEUR</u>	NOT NULL	VARCHAR2(30)
<u>TITRE</u> *	NOT NULL	VARCHAR2(40)

10. Les clés étrangères précisées précédemment.

11. Accélérer la recherche dans les tables.

L'index JOUER_FK et l'index PRODUIR_FK.

Tâche 2: Création de tables avec contraintes d'intégrités

```
2. CREATE TABLE SEANCE (  
  TITRE          VARCHAR2(40)          NOT NULL,  
  NOM_SALLE      VARCHAR2(30)          NOT NULL,  
  HEURE_DEBUT    INTEGER                NOT NULL,  
  VERSION        VARCHAR2(10),  
  CONSTRAINT PK_SEANCE PRIMARY KEY (TITRE, NOM_SALLE, HEURE_DEBUT),  
  CONSTRAINT FK_SEANCE_PROJETER_FILM FOREIGN KEY (TITRE)  
    REFERENCES FILM (TITRE)  
);  
  
3. CREATE TABLE VU (  
  NOM_SPECTATEUR VARCHAR2(30)          NOT NULL,  
  TITRE          VARCHAR2(40)          NOT NULL,  
  CONSTRAINT PK_VU PRIMARY KEY (NOM_SPECTATEUR, TITRE),  
  CONSTRAINT FK_VU_VOIR_FILM FOREIGN KEY (TITRE)  
    REFERENCES FILM (TITRE)  
);  
  
4. CREATE TABLE AIME (  
  NOM_AMATEUR    VARCHAR2(30)          ,  
  TITRE          VARCHAR2(40)  
);  
  
6. CREATE PUBLIC SYNONYM toto FOR system. SEANCE;
```

Tâche 3: Création d'indexes

```
1. CREATE INDEX PROJETER_FK ON SEANCE ( TITRE ASC);  
2. CREATE INDEX VOIR_FK ON VU ( TITRE DESC);  
3. CREATE INDEX AIMER_FK ON AIME ( TITRE ASC);
```

Tâche 4: Modification du schéma et des contraintes

```
1. ALTER TABLE AIME MODIFY( TITRE VARCHAR(40));  
  
2. ALTER TABLE SEANCE ADD CONSTRAINT VERIFY_HEURE_DEBUT CHECK (  
  HEURE_DEBUT > 13 );
```

```
3. ALTER TABLE SEANCE ADD CONSTRAINT VERIFY_NOM_SALLE CHECK
(NOM_SALLE IN( 'IBN ZAIDOUN', 'IBN KHALDOUN' ));

4. ALTER TABLE SEANCE MODIFY VERSION DEFAULT 'VO';

5.ALTER TABLE AIME ADD CONSTRAINT VERIFY_NOM_AMATEUR CHECK
(NOM_AMATEUR LIKE '%R');

6.ALTER TABLE SEANCE DROP CONSTRAINT VERIFY_HEURE_DEBUT ;

ALTER TABLE SEANCE DROP CONSTRAINT VERIFY_NOM_SALLE ;

ALTER TABLE SEANCE MODIFY VERSION DEFAULT NULL ;

ALTER TABLE AIME DROP CONSTRAINT VERIFY_NOM_AMATEUR ;

7. ALTER TABLE SEANCE MODIFY HEURE_DEBUT VARCHAR2(6) ;

8. ALTER TABLE AIME ADD( PRENOM_AMATEUR VARCHAR(30));

9. ALTER TABLE AIME DROP COLUMN PRENOM_AMATEUR ;

10. ALTER TABLE SEANCE MODIFY HEURE_DEBUT NULL;

11. ALTER TABLE SEANCE MODIFY HEURE_DEBUT NOT NULL;

12.ALTER TABLE AIME ADD CONSTRAINT PK_AIME PRIMARY KEY(
NOM_AMATEUR,TITRE);

13.ALTER TABLE AIME ADD CONSTRAINT FK_AIME FOREIGN KEY(TITRE)
REFERENCES FILM(TITRE);

14.ALTER TABLE VU DROP CONSTRAINT FK_VU_VOIR_FILM;

ALTER TABLE VU DROP CONSTRAINT PK_VU;

15. ALTER TABLE VU ADD (CONSTRAINT PK_VU PRIMARY KEY
(TITRE,NOM_SPECTATEUR), CONSTRAINT FK_VU_VOIR_FILM FOREIGN KEY
(TITRE)REFERENCES FILM (TITRE));
```

Solution TP 3: Insertions et modifications dans la Base de données

Tâche 1: Analyse du fichier donnees.txt

4. SQL> SPOOL C:/TPBDD/sauvegarde3

5. SQL> @ C:/TPBDD//TPOracleScript.sql

6. SQL> @ C:/TPBDD/donnees.txt.

8. SQL> DROP TABLE FILM;

drop table film

*

ERROR at line 1:

ORA-02449: unique/primary keys in table referenced by foreign keys

Problème de clé étrangère.

9. DROP TABLE FILM CASCADE CONSTRAINTS;

Tâche 2: Insertion de tuples

3. SQL> INSERT INTO Seance VALUES ('Perfect World', 'Beta', '19:00', 'VO');

SQL> INSERT INTO Produit VALUES ('Adam Moore', 'ET');

4. Non pas de problème.

5.

INSERT INTO Joue VALUES ('Kevin Costner', 'Waterworld');

INSERT INTO Produit VALUES ('Kevin Costner', 'Waterworld');

INSERT INTO Seance VALUES ('Waterworld', 'Alpha', '15:00', 'VF');

INSERT INTO Seance VALUES ('Waterworld', 'Alpha', '17:00', 'VO');

INSERT INTO Seance VALUES ('Waterworld', 'Beta', '15:00', 'VF');

INSERT INTO Vu VALUES ('Kevin Costner', 'Waterworld');

6. Violation des contraintes de clés étrangères.

7. INSERT INTO Film VALUES ('Waterworld', 250, 'US', 'Rickley Moore');

Tâche 3: Modification et suppression de tuples

1. UPDATE film SET duree = duree+60 where titre = 'Waterworld';

1 row updated.

2. UPDATE SEANCE SET version = 'VO';

12 rows updated.

3. DELETE FROM SEANCE WHERE HEURE_DEBUT='17:00';

1 row deleted.

4. SQL> DELETE FROM FILM WHERE TITRE='Waterworld';

DELETE FROM FILM WHERE TITRE='Waterworld'

*

ERROR at line 1:

ORA-02292: integrity constraint (SYSTEM.FK_JOUE_JOUER_FILM) violated - child record found

Solution : Ajouter ON DELETE CASCADE

create table JOUE (

NOM_ACTEUR VARCHAR2(30) not null,

TITRE VARCHAR2(40) not null,

constraint PK_JOUE primary key (NOM_ACTEUR, TITRE),

constraint FK_JOUE_JOUER_FILM foreign key (TITRE)

references FILM (TITRE) **ON DELETE CASCADE**

);

Solution TP 4: Gestion des utilisateurs et des privilèges

Tâche 1:

4. CREATE USER TP4user1 IDENTIFIED BY TP41;
5. SHOW USER;
6. DISC;
7. GRANT ALL PRIVILEGES TO TP4user1;
DISC ;
CONNECT TP4user1;
8. SQL> SELECT * FROM FILM ;
ERROR at line 1:
ORA-00942: table or view does not exist (la table film appartient à l'utilisateur system)
Solution :
SQL> SELECT * FROM system.FILM ;
9. REVOKE all privileges FROM TP4user1;
10. DISC;
CONNECT system;
DROP USER TP4user1;

Tâche 2:

1. CREATE USER TP4user2 IDENTIFIED BY TP42;
2. GRANT SELECT ON SEANCE TO TP4user2;
3. GRANT INSERT ON VU TO TP4user2;
4. SQL> connect TP4user2
Enter password:
ERROR:
ORA-01045: user TP4USER2 lacks CREATE SESSION privilege; logon denied
Solution :
GRANT CREATE SESSION TO TP4user2;
5. CONNECT TP4user2;
SELECT * FROM system.SEANCE;
INSERT INTO system.VU VALUES ('Said', 'Men');

```
DISC;  
CONNECT system;  
SELECT * FROM system.VU;
```

6. REVOKE SELECT ON SEANCE FROM TP4user2;
REVOKE INSERT ON VU FROM TP4user2;
7. DROP USER TP4user2;
8. GRANT SELECT ON VU TO PUBLIC;
9. REVOKE SELECT ON VU FROM PUBLIC;

Tâche 3:

1. CREATE USER TP4user3 IDENTIFIED BY TP43;
2. GRANT SELECT,INSERT,UPDATE,DELETE ON FILM TO TP4user3;
GRANT SELECT,INSERT,UPDATE,DELETE ON SEANCE TO TP4user3;
GRANT SELECT,INSERT,UPDATE,DELETE ON VU TO TP4user3;
GRANT SELECT,INSERT,UPDATE,DELETE ON AIME TO TP4user3;
GRANT SELECT,INSERT,UPDATE,DELETE ON PRODUIT TO TP4user3;
GRANT SELECT,INSERT,UPDATE,DELETE ON JOUE TO TP4user3;
4. REVOKE SELECT,INSERT,UPDATE,DELETE ON VU FROM TP4user3;
5. DROP USER TP4user3;

Solution TP 5: Recherche dans la Base de données

Tâche 1 :

1. SQL> SELECT titre from film where titre like '%World%';
TITRE

Perfect World

2. SQL> SELECT titre from film where duree < 60 ;

no rows selected

3. SQL> SELECT titre from film where duree > 140;

TITRE

Waterworld
Les Incorruptibles
Perfect World
Le Parrain
Il Faut Sauver Le soldat Ryan
ET
Indiana Jones

7 rows selected.

Tâche 2:

1. Quel est le nom du réalisateur du film 'Waterworld'?

```
SELECT nom_realisateur  
FROM film  
WHERE titre='Waterworld';
```

Résultat:

Rickley Moore

2. Où peut on voir jouer 'Kevin Costner' en version original 'VO' ?

a)

```
SELECT NOM_SALLE  
FROM SEANCE  
WHERE VERSION ='VO' AND TITRE IN ( SELECT TITRE  
FROM JOUE  
WHERE NOM_ACTEUR='Kevin Costner');
```

b)

```
SELECT NOM_SALLE
FROM SEANCE , JOUE
WHERE SEANCE.TITRE = JOUE.TITRE
AND SEANCE.VERSION = 'VO'
AND JOUE.NOM_ACTEUR='Kevin Costner';
```

Résultat:

NOM_SALLE

Alpha
Ibn Zeidoun
Beta
Alpha

3. Quels sont les titres des films réalisés par le réalisateur du film 'Waterworld'?

```
SELECT titre
from Film
WHERE Nom_Réalisateur IN
(SELECT Nom_Réalisateur FROM Film WHERE Titre='Waterworld');
```

Résultat:

TITRE

Waterworld
Les Incorruptibles
Men

4. Quels sont les titres de films réalisés par 'Spielberg' non projetés actuellement ?

a)

```
SELECT titre
from film
WHERE Nom_Réalisateur = 'Spielberg'
AND titre NOT IN ( SELECT titre FROM SEANCE);
```

b)

```
SELECT titre
from film
WHERE Nom_Réalisateur = 'Spielberg'
MINUS
SELECT titre
FROM SEANCE;
```

c)

```
SELECT titre
from film
WHERE Nom_Réalisateur = 'Spielberg'
AND
```

NOT EXISTS (SELECT * FROM SEANCE WHERE SEANCE.TITRE = FILM.TITRE);

Résultat:

TITRE

Il Faut Sauver Le soldat Ryan

5. Où peut on voir un film dans lequel jouent 'Tom Cruise' et 'Nicole Kidman'?

a)

```
SELECT NOM_SALLE
FROM SEANCE
WHERE TITRE IN
    ( SELECT TITRE
      FROM JOUE
      WHERE NOM_ACTEUR='Tom Cruise')
AND
    TITRE IN
    ( SELECT TITRE
      FROM JOUE
      WHERE NOM_ACTEUR='Nicole Kidman');
```

b)

```
SELECT NOM_SALLE
FROM SEANCE
WHERE TITRE IN
    ( SELECT TITRE
      FROM JOUE
      WHERE NOM_ACTEUR='Tom Cruise'
      INTERSECT
      SELECT TITRE
      FROM JOUE
      WHERE NOM_ACTEUR='Nicole Kidman');
```

Résultat:

NOM_SALLE

Beta

Tâche 3:

8. Quels sont les noms des acteurs qui jouent dans un film qu'ils produisent et réalisent ?

```
SELECT Nom_Acteur
FROM Joue, Film, Produit
WHERE Joue.Titre=Film.Titre
AND Joue.Titre=produit.Titre
AND Joue.Nom_Acteur=Film.Nom_Realisateur
```

AND Joue.Nom_Acteur=Produit.Nom_Producteur;

Résultat:

NOM_ACTEUR

Clint Eastwood

9. Quels sont les titres de films dans lesquels 'Tom Cruise' n'a pas 'Nicole Kidman' comme partenaire?

```
SELECT titre
from Joue
WHERE joue.NOM_ACTEUR='Tom Cruise'
AND TITRE NOT IN
      (SELECT TITRE
       FROM joue
       WHERE joue.Nom_Acteur ='Nicole Kidman');
```

Résultat

TITRE

Top Gun

10. Quels producteurs ne produisent **aucun** film réalisé par 'Spielberg'?

```
SELECT Nom_Producteur
from Produit
where Nom_Producteur NOT IN
(SELECT Nom_Producteur
 from Produit , FILM
 where Produit.titre = Film.TITRE
 AND Nom_Realisateur = 'Spielberg' ) ;
```

Résultat

NOM_PRODUCTEUR

Clint Eastwood
Kevin Costner
Alex May

11. Quels sont les noms des acteurs qui jouent dans tous les films réalisés par 'Spielberg'?

```
SELECT DISTINCT NOM_ACTEUR
from JOUE j
WHERE NOT EXISTS
      (SELECT *
```

```
FROM film f
WHERE f.Nom_Réalisateur = 'Spielberg'
AND NOT EXISTS
  (SELECT *
   FROM Joue g
   WHERE f.titre = g.titre
   AND j.NOM_ACTEUR = g.NOM_ACTEUR));
```

Résultat

NOM_ACTEUR

Amanda Peet

Aaron Smith

12. Quels acteurs ne jouent **que dans tous** les films réalisés par 'Spielberg' ?

```
SELECT NOM_ACTEUR
from JOUE X
WHERE NOT EXISTS
  (SELECT *
   FROM film Y
   WHERE Y.Nom_Réalisateur = 'Spielberg'
   AND NOT EXISTS
     (SELECT *
      FROM Joue Z
      WHERE Z.titre = Y.titre
      AND Z.NOM_ACTEUR= X.NOM_ACTEUR))
```

MINUS

```
SELECT Nom_Acteur
from Joue , FILM
where Joue.titre = Film.TITRE
AND Film.Nom_Réalisateur <> 'Spielberg' ;
```

Résultat

NOM_ACTEUR

Aaron Smith

13. Quels sont les acteurs qui voient **tous** les films dans lesquels ils jouent ?

Ces acteurs doivent voir tous les films dans lesquels ils jouent mais ils peuvent voir des films dans lesquels ils ne jouent pas.

```
SELECT DISTINCT NOM_ACTEUR
from JOUE X
```

WHERE NOT EXISTS

```
(SELECT *  
  FROM vu Y  
  WHERE X.Nom_Acteur = Y.Nom_spectateur  
  AND NOT EXISTS  
    (SELECT *  
      FROM Joue Z  
      WHERE Z.titre = Y.titre  
      AND  Z.NOM_ACTEUR = X.NOM_ACTEUR));
```

Résultat

NOM_ACTEUR

Nicole Kidman
Amanda Peet
Aaron Smith
Tom Cruise
Harrison Ford
Clint Eastwood

6 rows selected.

14. Quels sont les acteurs **qui ne voient que tous** les films dans lesquels ils jouent?

```
SELECT NOM_ACTEUR  
from JOUE X  
WHERE NOT EXISTS  
  (SELECT *  
    FROM vu Y  
    WHERE X.Nom_Acteur = Y.Nom_spectateur  
    AND NOT EXISTS  
      (SELECT *  
        FROM Joue Z  
        WHERE Z.titre = Y.titre  
        AND  Z.NOM_ACTEUR = X.NOM_ACTEUR))
```

MINUS

```
SELECT Nom_Acteur  
from Joue X  
WHERE EXISTS  
  (SELECT *  
    FROM Vu Y  
    WHERE X.Nom_Acteur = Y.Nom_Spectateur  
    AND NOT EXISTS  
      (SELECT *  
        FROM Joue Z
```

```
WHERE Z.titre = Y.titre
AND  Z.Nom_Acteur = X.NOM_ACTEUR));
```

Résultat

NOM_ACTEUR

```
-----
Aaron Smith
Amanda Peet
Clint Eastwood
Harrison Ford
Nicole Kidman
Tom Cruise
```

6 rows selected.

Tâche 4: Consultation de quelques tables systèmes.

1. dba_tables

```
DESC dba_tables;
```

```
select table_name from dba_tables where owner='SYSTEM';
```

2. dba_constraints

```
DESC dba_constraints;
```

```
SELECT TABLE_NAME, CONSTRAINT_NAME  from dba_constraints where owner='SYSTEM';
```

3. dba_indexes

```
DESC  dba_indexes;
```

```
SELECT INDEX_NAME, TABLE_NAME  from dba_indexes where owner='SYSTEM';
```

4. dba_users

```
DESC  dba_users;
```

```
SELECT USERNAME, PASSWORD from dba_users;
```

Solution TP 6: Projet "De l'étude conceptuelle à la création de la base de données"**Tâche 1: Création des tables avec les contraintes d'intégrité et des indexes.**

1. CREATE USER BDDAdmin IDENTIFIED BY TPAdmin;
2. GRANT ALL PRIVILEGES TO BDDAdmin;
4. Créer les tables précédentes avec les contraintes d'intégrités.

```
CREATE TABLE ETUDIANT (  
  MATRICULE_ETU      INTEGER          NOT NULL,  
  NOM_ETU            VARCHAR2(40)      NOT NULL,  
  PRENOM_ETU         VARCHAR2(40)      NOT NULL,  
  DATE_NAISSANCE     VARCHAR2(10)     NOT NULL,  
  CONSTRAINT PK_ETUDIANT PRIMARY KEY (MATRICULE_ETU)  
);
```

```
CREATE TABLE ENSEIGNANT (  
  MATRICULE_ENS      INTEGER NOT NULL,  
  NOM_ENS            VARCHAR2(30)      NOT NULL,  
  PRENOM_ENS         VARCHAR2(30)      NOT NULL,  
  AGE                INTEGER           NOT NULL,  
  CONSTRAINT PK_UNITE PRIMARY KEY (MATRICULE_ENS )  
);
```

```
CREATE TABLE UNITE (  
  CODE_UNITE         VARCHAR2(30)      NOT NULL,  
  LIBELLE            VARCHAR2(30)      NOT NULL,  
  NBR_HEURES         INTEGER           NOT NULL,  
  MATRICULE_ENS      INTEGER           NOT NULL,  
  CONSTRAINT PK_UNITES PRIMARY KEY (CODE_UNITE),  
  CONSTRAINT FK_UNITE_ENSEIGNANT FOREIGN KEY (MATRICULE_ENS )  
    REFERENCES ENSEIGNANT (MATRICULE_ENS)  
);
```

```
CREATE TABLE ETUDIANTUNITE (  
  MATRICULE_ETU      INTEGER          NOT NULL,  
  CODE_UNITE         VARCHAR2(30)      NOT NULL,  
  NOTE_CC            INTEGER           NOT NULL,  
  NOTE_TP            INTEGER           NOT NULL,  
  NOTE_EXAMEN        INTEGER           NOT NULL,
```

```
CONSTRAINT PK_ETUDIANTUNITE PRIMARY KEY (MATRICULE_ETU
, CODE_UNITE),
CONSTRAINT FK_ETUDIANTUNITE_ETUDIANT FOREIGN KEY (MATRICULE_ETU
REFERENCES ETUDIANT (MATRICULE_ETU),
CONSTRAINT FK_ETUDIANTUNITE_UNITE FOREIGN KEY (CODE_UNITE
REFERENCES UNITE (CODE_UNITE
);
```

```
5. CREATE INDEX Etudiant_FK ON Etudiant ( nom_etu ASC);
```

```
6. CREATE INDEX Enseignant_FK ON Enseignant ( nom_ens DESC);
```

Tâche 2: Création des utilisateurs.

1. CREATE USER Etudiant IDENTIFIED BY TPEtudiant;
2. GRANT SELECT ON Etudiant TO Etudiant;
3. CREATE USER Enseignant IDENTIFIED BY TPEnseignant;
4. GRANT SELECT, INSERT ON Enseignant TO Enseignant;

Tâche 3: Modification du schéma et ajout de contraintes

1. ALTER TABLE Etudiant ADD (Adresse VARCHAR2(100));
2. ALTER TABLE Enseignant DROP COLUMN age ;
3. ALTER TABLE Etudiant ADD CONSTRAINT VERIFY_MATRICULE CHECK (MATRICULE_ETU > 20190000 AND MATRICULE_ETU < 20199999);
4. ALTER TABLE Etudiant MODIFY PRENOM_ETU VARCHAR2(45) ;

Tâche 4: Insertions et modifications des tuples.

1. Insérer les tuples ci-dessous.

```
INSERT INTO Etudiant VALUES (20190001,'BOUSSAI','MOHAMED','12/01/2000','Alger');
```

```
INSERT INTO Etudiant VALUES (20190002,'CHAIID','LAMIA', '01/10/1999','Batna');
```

```
INSERT INTO Etudiant VALUES (20190003,'BRAHIMI','SOUAD','18/11/2000','Sétif');
```

```
INSERT INTO Etudiant VALUES (20190004,'LAMA','SAID','23/05/1999','Oran');
```

```
INSERT INTO Enseignant VALUES (20190001,'HAROUNI','AMINE');
```

```
INSERT INTO Enseignant VALUES (19990011,'FATHI','OMAR');
```

```
INSERT INTO Enseignant VALUES (19980078,'BOUZIDANE','FARAH');
```

```
INSERT INTO Enseignant VALUES (20170015,'ARABI','ZOUBIDA');
```

```
INSERT INTO Unite VALUES ( 'FEI0001','POO',6,20190001);
```

```
INSERT INTO Unite VALUES ('FEI0002','BDD',6,19990011);
```

```
INSERT INTO Unite VALUES ('FEI0003','RESEAU',3,20170015);
```

```
INSERT INTO Unite VALUES ('FEI0004','SYSTEME',6,19980078);
```

```
INSERT INTO EtudiantUnite VALUES (20190001,'FEI0001',10,15,9);
```

```
INSERT INTO EtudiantUnite VALUES (20190002,'FEI0001',20,13,10);
```

```
INSERT INTO EtudiantUnite VALUES (20190004,'FEI0001',13,17,16);
```

```
INSERT INTO EtudiantUnite VALUES (20190002,'FEI0002',10,16,17);
```

```
INSERT INTO EtudiantUnite VALUES (20190003,'FEI0002',9,8,15);
```

```
INSERT INTO EtudiantUnite VALUES (20190004,'FEI0002',15,9,20);
```

```
INSERT INTO EtudiantUnite VALUES (20190002,'FEI0004',12,18,14);
```

```
INSERT INTO EtudiantUnite VALUES (20190003,'FEI0004',17,12,15);
```

```
INSERT INTO EtudiantUnite VALUES (20190004,'FEI0004',12,13,20);
```

```
COMMIT;
```

2. UPDATE EtudiantUnite

```
SET note_CC = note_CC +2
```

```
WHERE MATRICULE_ETU IN ( SELECT MATRICULE_ETU FROM ETUDIANT  
WHERE NOM_ETU LIKE 'B%' );
```

3. UPDATE EtudiantUnite

```
SET NOTE_EXAMEN = 0
```

```
WHERE CODE_UNITE IN ( SELECT CODE_UNITE FROM UNITE  
WHERE LIBELLE = 'SYSTEME' );
```

Tâche 5: Interrogation de la base de données

```
1. SELECT  nom_etu , prenom_etu
  FROM  ETUDIANT  WHERE
    matricule_etu IN ( SELECT  matricule_etu FROM EtudiantUnite
                      WHERE  note_examen = 20 ) ;
```

Résultat

LAMA SAID

```
2. SELECT  nom_etu , prenom_etu
  FROM  ETUDIANT  Y
  WHERE NOT EXISTS (SELECT *
                    FROM EtudiantUnite X
                    WHERE X. matricule_etu = Y. matricule_etu
                    AND  code_Unite IN ( SELECT code_unite FROM UNITE
                                         WHERE libelle = 'POO') ) ;
```

Résultat

BRAHIMI SOUAD

```
3. SELECT libelle FROM UNITE X
   WHERE NOT EXISTS ( SELECT * FROM EtudiantUnite Y
                     WHERE X.code_unite = Y.code_unite);
```

Résultat

LIBELLE

RESEAU

```
4. SELECT  nom_etu , prenom_etu
  FROM  ETUDIANT  Y
  WHERE  EXISTS (SELECT *
                FROM EtudiantUnite X
                WHERE X. matricule_etu = Y. matricule_etu
                AND  code_Unite IN ( SELECT code_unite FROM UNITE
                                     WHERE libelle = 'POO') )
```

```
AND NOT EXISTS (SELECT *  
                FROM EtudiantUnite W  
                WHERE Y. matricule_etu = W. matricule_etu  
                AND code_Unite IN ( SELECT code_unite FROM UNITE  
                                   WHERE libelle <> 'POO') );
```

Résultat

BOUSSAI MOHAMED

Solution TP 7: Les fonctions d'agrégat

Solution TP 8: Création et manipulation des vues

Solution TP 9: Eclipse et WindowBuilder

Solution TP 10: Connectivité avec la base de données Oracle à travers Eclipse