TD1 BD (Module HMIN112M) : Modélisation conceptuelle avec UML

1. Préambule

Nous nous contenterons de modéliser les aspects structurels des univers considérés. Vous aurez donc surtout à construire des diagrammes de classes et éventuellement des diagrammes d'objets.

2. Bars à bières

Modéliser une base réunissant l'information relative à des buveurs, des bières et des bars. Un buveur aime certaines bières, un bar sert un certain nombre de bières et enfin un buveur fréquente certains bars.

3. Cabinet d'assurances

Il s'agit de modéliser une partie de la gestion d'une compagnie d'assurances dont les clients sont possesseurs d'un ou de plusieurs véhicules.

1. Quelques éléments de modélisation vous sont apportés
   - chaque client est caractérisé par son numéro de sécurité sociale, son nom et son adresse,
   - chaque véhicule est caractérisé par son numéro d'immatriculation, son modèle et son année de mise en circulation,
   - chaque véhicule fait l'objet d'un certain nombre de sinistres dont on connaît la date et le montant des dégâts,
2. Nous voulons maintenant ajouter la potentialité d'un changement de propriétaire et d'immatriculation
   - la possession d'un véhicule par un propriétaire est définie par un intervalle date-début, date-fin. Comment est représenté le possesseur actuel d'un véhicule ?
   - un véhicule est caractérisé par son numéro de chassis. Comment lui attacher une ou des immatriculations ?

4. Pièces de précision

Une société spécialisée dans l'usinage de pièces de précision a à sa tête un responsable qui dirige tous les employés (le responsable est vu lui même comme un employé). Les employés appartiennent à divers secteurs d'activités (production, administratif, communication, qualité, achat, recherche, ...) et se divisent en cadres et non-cadres. Un employé possède un nom, un prénom, un âge, une date d'embauche, une fonction et un salaire). Les cadres se voient attribuer une prime annuelle en fonction de leur ancienneté mais ne peuvent pas bénéficier du paiement d'heures supplémentaires. Inversement,

les non-cadres ne bénéficient pas de primes mais peuvent effectuer un nombre d'heures supplémentaires qui leur seront payées par l'entreprise. Un secteur d'activité possède un nom, une localisation dans l'usine et dispose d'un budget global.

— proposer un diagramme de classes satisfaisant la description précédente.
— rajouter dans le modèle la possibilité pour un employé de changer de secteur d'activité tout en restant dans l'entreprise (garder un historique pour l'employé des secteurs dans lesquels il a travaillé).

5. Liens matrimoniaux et de parenté

Modéliser de diverses manières les relations matrimoniales et de parenté entre les types d'entités Homme et Femme (et Personne éventuellement). Comparer les mérites de chacune de ces différentes modélisations.

6. Bibliothèque

Une bibliothèque gère le stockage, l'indexation et l'emprunt d'un ensemble de livres.
Vous vous baserez sur les éléments suivants pour construire votre modèle UML.

— Chaque livre est un exemplaire d'un ouvrage ; il peut exister plusieurs exemplaires d'un même ouvrage.
— Un ouvrage est caractérisé par son numéro ISBN, un titre, ses auteurs éventuels, l'année de parution, l'éditeur et une liste de mots-clés qui le caractérisent
— Les auteurs sont connus par leur nom, prénom et leur organisation.
— Chaque livre est identifié par sa cote et sa localisation dans la bibliothèque.
— Les emprunteurs sont identifiés par un numéro d'abonné et possèdent un nom, un prénom, une adresse et un numéro de téléphone.
— Les emprunts passés ne sont pas mémorisés.

Enrichissez le modèle en mémorisant les emprunts passés.

7. Habitation (exercice maison)

Un syndic assure la gestion d'un ensemble important de logements et doit répartir des charges relatives à une période (semestre) entre les logements.
Chaque logement (numéroté) est situé dans un immeuble (nommé et ayant une adresse). Chaque immeuble est situé dans un quartier. La répartition des charges d'un immeuble entre ses différents logements se fait proportionnellement à la surface des logements. La répartition des charges d'un quartier entre ses différents immeubles se fait proportionnellement à la somme des surfaces des logements de chaque immeuble multipliée par un coefficient traduisant la catégorie de l'immeuble (grand standing : 3, standing moyen : 2, normal : 1, etc...).
La société paye les factures d'entretien, chauffage, eau ... Ces factures identifiées par un numéro sont relatives à un semestre déterminé et unique mais peuvent concerner un ou plusieurs logements suivant certains critères :

— Une facture peut concerner un logement unique, par exemple une réparation incombant au locataire
— Une facture peut concerner tous les logements d'un même immeuble, par exemple les dépenses

d'eau
- Une facture peut concerner tous les logements de tous les immeubles d'un quartier, par exemple l'entretien des espaces verts
- Contrainte : Une facture doit obligatoirement se rapporter soit à un et un seul quartier, soit à un et un seul immeuble, soit à un et un seul logement.

Donner le diagramme de classes associé.

8. Traduction en modèle relationnel

La suite de ce TD consistera à proposer un modèle relationnel pour chacun des diagrammes de classes définis précédemment au sein des différents exercices de ce TD

TD2 BD (HMIN112M) : Calcul Relationnel

1. Exercice 1

Deux tables représentent la même relation si l'une peut être obtenue à partir de l'autre par permutation de lignes et/ou de colonnes, à condition que l'attribut désignant chaque colonne soit déplacé avec le contenu de la colonne.
Si le schéma de la relation comporte m attributs et si une instance de la relation comporte p n-uplets, combien peut-on construire de représentations tabulaires de cette même instance ?

2. Exercice 2

Soient R et S les relations suivantes :

A	B
a	b
c	b
d	e
a	c

FIGURE 1 – R

B	C
b	c
c	a
b	d
c	b

FIGURE 2 – S

On pose aussi S'(A,B)=S(B,C)

1. R $\bigcup$ S'
2. R $-$ S'
3. R $\bowtie$ S
4. R $\ltimes$ S ($\Pi_{attde R}$(R $\bowtie$ S)
5. Π_A(R) et Π_B(R)
6. R $\bowtie$ S avec comme condition de rapprochement A=C
7. R $\bowtie$ S avec comme condition de rapprochement R.B<C (ordre alphabétique sur les lettres)

3. Exercice 3

Soient R et S les deux relations suivantes :

A	B	C
a	b	c
c	d	e
b	e	f
d	a	h

FIGURE 3 – R

A	B	D
a	b	c
a	e	f
b	e	f
e	b	a
d	a	b

FIGURE 4 – S

On pose aussi S'(A,B,C)=S(A,B,D)

1. R $\bigcup$ S'
2. S' − R
3. R $\bowtie$ S
4. R $\ltimes$ S ($\Pi_{attde R}$(R $\bowtie$ S)

4. Exercice 4

Calculer Q=R ÷ S avec S=S1 puis S2 puis S3

A	B
a1	b1
a2	b2
a2	b1
a3	b3

FIGURE 5 – R

B
b1

FIGURE 6 – S1

B
b1
b2

FIGURE 7 – S2

B
b1
b2
b3

FIGURE 8 – S3

5. Exercice 5

Calculer Q=R ÷ S avec S=S1 puis S2 puis S3

A	B	C	D
a1	b1	c1	d1
a1	b1	c2	d3
a1	b2	c2	d3
a2	b2	c2	d2
a2	b1	c1	d1
a2	b1	c3	d3
a2	b2	c1	d1
a1	b1	c2	d2

FIGURE 9 – R

A	B
a1	b1
a2	b1
a2	b2

FIGURE 10 – S1

A	B
a1	b1
a2	b2

FIGURE 11 – S2

A
a1
a2

FIGURE 12 – S3

TD3 BD (HMIN112M) : Algèbre relationnelle et SQL en tant que LMD

1. Enoncé

Le schéma relationnel (très particulier) d'une base de données est le suivant :

DRAGONS(Dragon, Sexe, Longueur, NombreEcailles, CracheduFeu, ComportementAmoureux)

AIME(DragonAimant, DragonAimé, Force)

NOURRITURES(Produit, Calories)

REPAS(Dragon, Produit, Quantité)

Une réalisation en est donnée ci-dessous à titre illustratif :

DRAGONS	Dragon	Sexe	Longueur	NombreEcailles	CracheduFeu	ComportementAmoureux
	Sméagol	M	152	1857	oui	macho
	Birdurh	M	258	4787	non	timide
	Négueth	F	128	1581	oui	sincère
	MissToc	F	183	2781	non	superflu
	Bolong	M	213	2751	oui	macho
	Miloch	M	83	718	oui	timide
	Nessie	M	168	1721	non	absent
	Tarak	F	123	851	oui	timide
	Solong	M	173	1481	oui	sincère

Figure 1 – Instance de DRAGONS

NOURRITURES	Produit	Calories
	pomme	7
	cacahuète	10
	orange	25
	oeuf	15
	ver	3
	poisson	35

Figure 2 – Instance de NOURRITURES

REPAS	Dragon	Produit	Quantité
	Sméagol	cacahuète	1000
	Sméagol	pomme	16
	Birdurh	oeuf	4
	Négueth	orange	6
	Négueth	oeuf	1
	Miloch	ver	53
	Miloch	cacahuète	100
	Nessie	poisson	20
	Tarak	pomme	10
	Tarak	orange	10
	Solong	oeuf	6
	Solong	poisson	1
	Solong	orange	2

FIGURE 3 – Instance de REPAS

AIME	DragonAimant	DragonAimé	Force
	Sméagol	Négueth	passionnément
	Birdurh	Négueth	beaucoup
	Négueth	Miloch	à la folie
	Bolong	Négueth	à la folie
	Tarak	Bolong	un peu
	Solong	Tarak	beaucoup

FIGURE 4 – Instance de AIME

2. Questions

2.1 Rétroconception

Donnez le diagramme de classes correspondant à ce schéma relationnel. Proposez également un diagramme d'objets conforme au diagramme de classes construit (à partir des tuples de l'exercice).

2.2 Langage algébrique et SQL en tant que LMD

Traduire les requêtes suivantes en langage algébrique et en SQL

2.2.1 Sélection-Projection

1. Noms des dragons qui crachent du feu ?
2. Noms des dragon mâles qui crachent du feu ?
3. Noms des dragons femelles qui ne crachent pas de feu ?

4. Noms des dragons amoureux ?
5. Qui aime qui passionnément ?
6. Noms des dragons qui mangent des oeufs ?

2.2.2 Jointure-Différence-Union-Intersection

1. Noms des dragons qui ne sont pas amoureux ?
2. Liste des couples des dragons qui s'aiment mutuellement ?
3. Noms des dragons femelles qui ne crachent pas de feu ?
4. Noms des dragons qui ne mangent pas ?
5. Noms des dragons qui n'aiment personne et qui ne mangent pas ?

2.2.3 Division

1. Noms des dragons qui mangent de tout ?
2. Noms des dragonnes qui sont aimées par tous les dragons machos ?

TD4 BD (HMIN112M) : Théorie de la normalisation

1. Exercice 1

Enumérer les dépendances fonctionnelles (DFs) satisfaites par la relation R instance de $\mathcal{R}$(ABC) présentée ci-dessous :

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c3

FIGURE 1 – Réalisation R

2. Exercice 2

Soit U={A,B,C,D}, un ensemble d'attributs et un schéma relationnel $\mathcal{R}$ défini sur U.
Considérons les ensembles de dépendances fonctionnelles suivants :

1. F1={A $\rightarrow$ B, A $\rightarrow$ C}
2. F2={BD $\rightarrow$ A, D $\rightarrow$ C}

Il est maintenant envisagé de créer une relation R instance de $\mathcal{R}$ en insérant des n_uplets dans l'ordre suivant :

(a1,b1,c1,d1)
(a1,b1,c1,d2)
(a1,b1,c3,d2)
(a2,b2,c4,d4)
(a3,b2,c1,d2)
(a1,b2,c1,d1)
(a2,b2,c1,d1)

Déterminer pour chaque ensemble de DFs, les insertions éventuelles qui enfreignent les contraintes définies

3. Exercice 3

Soit F1, un ensemble de DFs défini comme suit :
F1={A $\rightarrow$ BC, B $\rightarrow$ E, C $\rightarrow$ F, EF $\rightarrow$ I, F $\rightarrow$ H}

Calculer A^+, B^+, BC^+, E^+ et F^+ par rapport à F1 en appliquant l'algorithme de fermeture vu en cours

4. Exercice 4

Soit le schéma relationnel $\mathcal{R}$(A,B,C,D,E,F) soumis à l'ensemble de DFs :
F1={A → B, C → D, ABC → E, AC → F, BC → FA}

Question 1 : Calculer A^+, B^+, C^+, AC^+ et BC^+ par rapport à F1 en appliquant l'algorithme de fermeture vu en cours

Question 2 : Calculer toutes les clés de $\mathcal{R}$ soumis à F1

Question 3 : Calculer une couverture minimale irredondante (CIM) de $\mathcal{R}$ soumis à F1

5. Exercice 5

Soit le schéma relationnel Employe(nom,salaire,chef,departement) soumis à l'ensemble de DFs, F1={nom → salaire, nom → departement, departement → chef}

Question 1 Donner la sémantique de ce schéma soumis à DFs.

Question 2 Montrer, à partir d'une instance du schéma, que ce schéma est redondant. Que proposez vous pour réduire cette redondance ?

6. Exercice 6

Le schéma relationnel $\mathcal{R}$ est défini sur les attributs CPHSEA (C pour Cours, P pour Professeur, H pour Heure, S pour Salle, E pour Etudiant et A pour Année).
$\mathcal{R}$ est soumis à l'ensemble de DFs :
F={C → P, HS → C, HP → S,CE → A, HE → S}

Donner la sémantique de ce schéma soumis à DFs.

Calculer la ou les clés de R.

Donner une décomposition en 3NF ou en BCNF

7. Exercice 7

Pour chacun des schémas de relation suivants :
— Vous indiquerez une violation de la propriété BCNF
— Vous donnerez une décomposition en forme normale BCNF
— Vous indiquerez une violation de la propriété 3NF
— Vous donnerez une décomposition en 3NF

a) $\mathcal{R}$1(ABCD), F1={AB → C, C → D, D → A }
b) $\mathcal{R}$1(ABCD), F1={B → C, B → D }
c) $\mathcal{R}$1(ABCDE), F1={AB → C, C → D, D → B, D → E }

TP1 BD (HMIN112M) : Client-Fournisseur

1. Connexion à Oracle

Le serveur Oracle est hébergé sur une machine serveur (nommée venus) sur laquelle on ne se connecte pas (a priori). Chaque machine sous linux possède un client nommé `sqlplus` (interpréteur de commandes SQL et module d'Oracle). Comme l'éditeur de commandes sous `sqlplus` est "ligne à ligne", il est conseillé d'utiliser un éditeur externe (exemple `xemacs` ou `gedit`).

Pour se connecter (sur une base déjà créée et nommée `master`), vous aurez à :

1. aller sur le site Web du SIF/DSIN et vous connecter afin de créer un compte oracle personnel sur la base master,
2. puis lancer le client `sqlplus` dans une fenêtre `xterm` par la commande (les paramètres identifiants et motDePasse sont à modifier en fonction du compte oracle que vous venez de créer)

   ```
   sqlplus identifiant/motDePasse@venus/master
   Après le  signe d'invite  SQL>
        saisir les commandes SQL.
   Les minuscules ne sont pas différenciées des majuscules pour ce qui concerne
   la syntaxe réservée.
   ```

3. l'utilitaire rlwrap (readLine wrapper) permet l'ajout de fonctionnalités d'édition, préférez donc la commande

   ```
   rlwrap sqlplus identifiant/motDePasse@venus/master
   ```

2. Session SQL interactif

L'exécutable `sqlplus` est un interpréteur/éditeur de commandes, chaque commande SQL pouvant être saisie sur une ou plusieurs lignes, le point virgule obligatoire termine la commande. Attention à la syntaxe qui est très rigoureuse. Voir l'exemple suivant : attention aux parenthèses, virgules et *surtout au point-virgule* qui termine une commande.

```
SQL>  create table fournisseur
  2  (nomf varchar(20),
  3  adrsf varchar(50));
Table created.
```

3. Client - Fournisseur

Soit les schémas relationnels suivants :

- FOURNISSEUR(nomf, adrsf) # catalogue des fournisseurs ; clé primaire nomf
- PRODUIT(nomp, nomf, cout) # catalogue donnant les produits de nomp, de leur fournisseur nomf et leur coût ; clé primaire le couple (nomp, nomf)
- COMMANDE(ncom,nomc,nomp,nomf,qte) # chaque commande de numéro ncom est passée par un client nomc et concerne une quantité qte de produit (nomp, nomf) ; clé primaire ncom
- CLIENT(nomc, adrsc, solde) # chaque client nomc a une adresse adrsc et un solde ; clé primaire nomc

Autres contraintes d'intégrité :

1. Les coûts des produits proposés dans le schéma PRODUIT sont > 0.
2. Les noms de fournisseurs dans le schéma PRODUIT sont des noms de fournisseurs référencés dans le schéma FOURNISSEUR.
3. Les clients qui passent des commandes sont des clients à part entière.
4. Les commandes concernent des produits (nomp, nomf) connus dans le schéma PRODUIT
5. Les quantités commandées sont > 0.

Vous allez créer les tables et les contraintes afférentes (de domaine, de clés primaires et d'intégrité référentielles). Pour ce faire, vous allez éditer un fichier de script contenant vos ordres SQL. Sauvegardez votre fichier avec l'extension .sql. Des tuples vous sont donnés qui serviront à alimenter vos tables. Exécuter ce script dans la session SQL interactive par la commande start "votreNomDeFichier". Vérifier la définition et le contenu des tables créées.

4. Requêtes

Répondre aux requêtes suivantes :

4.1 Requêtes de consultation

- Affichage du contenu de chaque table créée ;
- Nom des clients dont le solde est négatif ;
- Nom des fournisseurs qui proposent le produit "brique" ou le produit "parpaing" ;
- Nom et adresse des clients dont le nom commence par un "J" (attention à la casse de la police de caractères) ;
- Nom et adresse des clients ayant commandé du produit "brique", la quantité commandée étant comprise entre 5 et 10 ;
- Nom et coût moyen des articles proposés par les fournisseurs ;

4.2 Requêtes de définition et de mises à jour

- Insérer les tuples correspondant aux commandes suivantes :
  `6,paul,ciment,Samaco,12`
  `7,pierre,parpaing,Abounayan,8`
- Modifier l'adresse du client Jean qui devient 83000 Toulon ;
- Les produits du fournisseur Samaco ont augmenté de 15%, réaliser la mise à jour ;
- Le fournisseur Samaco est racheté par la firme Technal d'adresse 69005 Lyon (changement de nom et d'adresse) ;

4.2.1 Exercices facultatifs (en non présentiel)

4.2.2 Rétro-conception

À partir du schéma relationnel de la base de données, vous donnerez le diagramme de classes UML correspondant. Vous proposerez également un diagramme d'objets conforme au diagramme de classes, en prenant pour exemple les tuples de la base de données.

4.2.3 Requêtes

Vous écrirez et testerez les requêtes suivantes en SQL. Vous pouvez aussi écrire les requêtes 1 et 5 en algèbre relationnelle

1. Nom des fournisseurs qui proposent le produit "brique" mais pas le produit "parpaing" ;
2. Nom et coût moyen des produits proposés par au moins deux fournisseurs ;
3. Nom des produits et de leurs fournisseurs qui sont proposés par ces mêmes fournisseurs à des coûts supérieurs à leurs coûts moyens.
4. Nom des produits dont le prix est $>$ au coût moyen de tous les produits ;
5. Nom des fournisseurs qui proposent tous les produits commandés par Jean (à l'instant de l'exécution de la requête) ;

TP2 BD (HMIN112M) : SQL en tant que Langage de Manipulation de Données (LMD)

1. Création d'une base

Vous disposez d'un fichier Tp2MastereCreation.sql contenant un script de création d'une base de données nommée "Entreprise", que vous sauvegarderez dans un répertoire de travail.
Le schéma initial comporte les schémas relationnels suivants :

EMP (nom, **num**, fonction, n_sup, embauche, salaire, comm, n_dept)
un employé de numéro *num* possède un *nom* et occupe l'emploi *fonction*, cet employé a aussi un responsable identifié par le numéro *n_sup*, une date d'embauche *embauche*, un *salaire*, une commission *comm* et un département de rattachement *n_dept*
DEPT(**n_dept**, nom, lieu)
un Département de numéro *n_dept*, de nom *nom*, situé à *lieu*

Créer la base et vérifier le contenu des tables créées. Vous noterez que seules les clés primaires ont été posées sur les tables et non les clés d'intégrité référentielle. Vous rajouterez ces contraintes au cours du TP3.

2. Requêtes

Vous traduirez en SQL les requêtes suivantes :

1. Donner les nom, fonction et date d'embauche de tous les employés,
2. Donner les numéros, nom et salaire des employés dont le salaire est <= 2000 euros,
3. Donner la liste des employés ayant une commission, classée par commission décroissante,
4. Donner le nom des personnes embauchées depuis janvier 1991,
5. Donner pour chaque employé son nom et son lieu de travail,
6. Donner pour chaque employé le nom de son supérieur hiérarchique,
7. Quels sont les employés ayant la même fonction que "CODD" ?
8. Quels sont les employés gagnant plus que tous les employés du département 30 ?
9. Quels sont les employés ne travaillant pas dans le même département que leur supérieur hiérarchique ?
10. Quels sont les employés travaillant dans un département qui a procédé à des embauches depuis le début de l'année 98,
11. Donner le nom, la fonction et le salaire de l'employé (ou des employés) ayant le salaire le plus élevé,

12. Donner le total des salaires, le nombre de salariés, ainsi que le salaire minimal, moyen et maximal pour l'ensemble des salariés de chaque département,
13. Donner le ou les départements ayant le plus d'employés,
14. Donner les départements qui ne possèdent pas d'employés exerçant la fonction d'ingénieur,
15. Donner les départements possédant des employés exerçant l'ensemble des fonctions référencées au sein de la société.

2.0.1 Exercices facultatifs (en non présentiel)

2.0.2 Rétro-conception

À partir du schéma relationnel de la base de données, vous donnerez le diagramme de classes UML correspondant. Vous proposerez également un diagramme d'objets conforme au diagramme de classes, en prenant pour exemple les tuples de la base de données.

2.0.3 Requêtes

Vous écrirez et testerez les requêtes suivantes en SQL.

1. Nom et année d'embauche des salariés qui ont été embauchés en septembre (toutes années confondues)
2. Nom de la fonction qui est la mieux valorisée au sein de l'entreprise (la fonction pour laquelle la moyenne des salaires des employés exerçant cette fonction, est supérieure à la moyenne des salaires des employés exerçant les autres fonctions) ;
3. Nom du département qui est le plus valorisé au sein de l'entreprise (le département pour lequel la moyenne des salaires des employés travaillant dans ce département, est le plus élevé) ;
4. ajouter le tuple suivant dans EMP

   ```
   'MIRANDA',16099,'commercial',27047,'01-JUN-2019',3000,2000,20
   ```

5. Nom, salaire et prime des employés gagnant plus que leurs supérieurs (considérez dans cette question le cumul du salaire et de la prime pour comparer les gains des salariés et de leurs supérieurs)
6. Modifier le nom du département 40 qui de "fabrication", devient "production"
7. Ajouter un attribut (colonne) nommé budget à la table DEPT (type de données FLOAT) et lui associer une valeur par défaut de 10 000 euros.

TP3 BD (HMIN112M) : Contraintes et évolution de schéma

1. Schéma de base de données

Le schéma initial du TP précédent vous est rappelé (le script de création des tables du TP2 est à nouveau exploité) :

EMP (nom, **num**, fonction, n_sup, embauche, salaire, comm, n_dept)
un employé de numéro *num* possède un *nom* et occupe l'emploi *fonction*, cet employé a aussi un responsable identifié par le numéro *n_sup*, une date d'embauche *embauche*, un *salaire*, une commission *comm* et un département de rattachement *n_dept*
DEPT(**n_dept**, nom, lieu)
un Département de numéro *n_dept*, de nom *nom* est situé à *lieu*

2. Contraintes

2.1 Définition des Contraintes

Le schéma initial ne comporte aucune contrainte et le concepteur constate qu'il a omis de déclarer des contraintes d'intégrité essentielles. Les contraintes ont en effet un rôle de garant dans la qualité et la cohérence des données. Vous noterez à ce sujet, que certaines données seront en non conformité face à certaines de ces contraintes et causeront donc quelques problèmes qu'il vous faudra résoudre. Il s'agit de mettre en œuvre ces contraintes en les créant et en les nommant au travers de la clause *ALTER TABLE ...*.

- Sur la table DEPT rajouter la contrainte *dept_pk* définissant n_dept comme clé primaire.
- Sur la table EMP rajouter les contraintes suivantes :
  1. *emp_pk* définissant num comme clé primaire,
  2. *nom_u* définissant nom comme nom unique,
  3. *responsable* définissant n_sup comme Foreign_key référençant num (CI intra table)
  4. *dept* définissant n_dept comme Foreign_key référençant n_dept de la table DEPT (CI inter table)
  5. *commission* définissant un controle tel que seuls les employés dont la fonction est commercial aient une commission comm non nulle (null).

2.2 Vérification

Vérifier que les contraintes ont été créées en consultant la vue du dictionnaire de données nommée *user_constraints*.
Lorsque certaines contraintes sont introduites dans la conception du schéma sans les nommer, le

système leur donne un nom interne.
Vérifier que ces contraintes sont actives en essayant de les transgresser en réalisant des insertions ou des effacements (donner un exemple de transgression).
Tout ordre sql s'effectue en fait au sein d'une *transaction*. Toute transaction peut être validée par l'ordre *commit* ou annulée par l'ordre *rollback*.

2.3 Désactiver/Activer des Contraintes

Dans certains cas, l'administrateur ou l'utilisateur qui a défini une contrainte peut souhaiter, pour effectuer des mises à jour qui la transgresse, désactiver momentanément cette contrainte.

1. désactiver la contrainte commission ;
2. insérer des n-uplets dans EMP transgressant la contrainte commission ;
3. essayer de rétablir la contrainte. Conclusion ? ;
4. détecter les n-uplets qui provoquent l'erreur afin de pouvoir les supprimer et rétablir la contrainte. Pour cela, créer une table que vous nommez par exemple REJETS :

```
CREATE TABLE REJETS
       (ROW_ID  ROWID,
        OWNER VARCHAR2(30),
        TABLE_NAME VARCHAR2(30),
        CONSTRAINT VARCHAR2(30));
```

Réactiver la contrainte commission en demandant la sauvegarde des n-uplets *erreurs* dans la table REJETS par la commande

```
alter table EMP enable constraint commission exceptions into REJETS
```

Vérifier le contenu de la table REJETS, puis supprimer de la table EMP les n-uplets *erreurs*. Rétablissez enfin la contrainte commission

2.4 Exploiter les vues du dictionnaire de données

Les vues du dictionnaire de données peuvent être consultées de la même manière que les tables de votre schéma utilisateur.
Vous exprimerez les ordres SQL suivants :

- Donner le nom et le type de chacune des contraintes posées sur la table EMP (vue user_constraints). Vous vérifierez également que ces contraintes sont actives et valides.
- Donner le nom des contraintes et le nom des attributs auquelles elles s'appliquent (vue user_cons_columns)
- Donner le nom de vos tables, ainsi que le nombre de tuples insérés dans ces tables (vue user_tables)
- Donner le nom des tables auquelles vous avez accès, ainsi que le nom de leurs propriétaires (vue all_tables)
- Donner le nom de toutes les tables (et de leurs propriétaires) de l'ensemble des schémas utilisateurs de la base de données master (vue dba_tables)

2.5 Suppression des contraintes

Supprimer les contraintes de la table EMP.

TP4 BD (HMIN112M) : Transactions et gestion des droits utilisateurs

1. Dictionnaire de données

Nous rappelons que le dictionnaire de données (ou méta-schéma) est un ensemble de tables dans lesquelles sont stockées les descriptions des objets de la base.
Les tables de ce dictionnaire peuvent être consultées au moyen du langage SQL. Des vues de ces tables permettent à l'utilisateur de voir les objets qui lui appartiennent (tables préfixées par USER) ou sur lesquels il a des droits (tables préfixées par ALL). L'administrateur a pour sa part accès à toutes les vues (les tables précédentes ainsi que les tables préfixées par DBA).
`Quelques vues et tables du dictionnaire de données :`

- USER_TABLES : tables et vues créées par l'utilisateur ;
- USER_CATALOG (ou CAT) : tables et vues sur lesquelles l'utilisateur a des droits à l'exception des tables et vues du dictionnaire de données ;
- USER_TAB_COLUMNS (ou COLS) : colonne de chaque table ou vue créée par l'utilisateur courant ;
- USER_CONSTRAINTS : définition des contraintes pour les tables des utilisateurs ;
- USER_CONS_COLUMNS : colonnes qui interviennent dans les définitions des contraintes ;
- USER_TAB_PRIVS : droits attribués et/ou reçus par l'utilisateur
- USER_SYS_PRIVS : privilèges donnés à l'utilisateur de manière générale ;
- USER_TAB_PRIVS_MADE : droits attribués par l'utilisateur ;
- USER_TAB_PRIVS_RECD : droits reçus par l'utilisateur ;
- ALL_CATALOG : liste de tous les objets accessibles par l'utilisateur ;
- ALL_TABLES Description des tables accessibles par l'utilisateur.

Connectez vous (les tables du TP précédent (sur Employé) étant créées) et répondez aux requêtes (portant sur le méta-schéma) suivantes :

1. Donner toutes les informations sur les tables sur lesquelles vous avez des droits
2. Donner le nombre d'attributs de la table EMP
3. Donner la liste des contraintes (avec leur statut) créées au cours du TP précédent
4. Donner les informations sur les contraintes de type clé primaire que vous aviez créées au cours de ce TP
5. Utilisez à bon escient la table USER_TAB_COLUMNS de manière à avoir un maximum d'informations sur une table donnée (partageant des similarités avec la commande `desc` de SQL*Plus).

2. Privilèges d'accès à la base de données

Oracle permet à plusieurs utilisateurs de travailler sur la même base de données en toute sécurité. Deux commandes sont à ce titre particulièrement importantes : `GRANT` et `REVOKE` et permettent de

définir les droits de chaque utilisateur sur les objets de la base.
Tout utilisateur accède à la base à l'aide de son nom utilisateur et de son mot de passe. C'est le nom utilisateur qui permet de déterminer les droits d'accès aux objets de la base de données.
Les utilisateurs ont été créés par le DBA et sont autorisés à se connecter à la base Oracle MASTER. Ils ont aussi les privilèges de créer des objets de schéma de base de données utilisateur (tables, vues, contraintes etc).
Au cours des TP précédents vous avez travaillé sous le nom d'un utilisateur et vous n'avez donc été en concurrence qu'avec vous-même.
Nous allons vérifier que le SGBD gère la concurrence d'accès à des objets de la base entre plusieurs utlisateurs différents.
Tout utilisateur qui crée des objets est propriétaire de ces objets (`table_name`, `owner` de `USER_tables` dans le dictionnaire des données). A ce titre, tout objet dont vous êtes propriétaire est préfixé par votre nom utilisateur. Le créateur d'un objet peut décider de donner (ou de supprimer) certains droits d'accès à cet objet à tout autre utilisateur de sa connaissance.

2.1 L'ordre GRANT

```
GRANT privilege ON table/vue TO utilisateur [WITH GRANT OPTION]
```

Cet ordre permet de donner le privilège concerné sur la table ou la vue à l'utilisateur.
Exemple : X a créé la table EMP et veut autoriser Y à lire cette table.
Il passe alors l'ordre `GRANT SELECT ON EMP TO Y;`
Les privilèges qui peuvent être donnés sont les suivants :

1. SELECT : droit de lecture ;
2. INSERT : droit d'insertion de lignes ;
3. UPDATE : droit de modification de lignes ;
4. DELETE : droit de suppression de lignes ;
5. ALTER : droit de modification de la définition de la table ;
6. INDEX : droit de création d'index ;
7. ALL : tous les droits ci-dessus.

Un utilisateur ayant reçu un privilège avec la mention facultative WITH GRANT OPTION peut les transmettre à son tour à un autre utilisateur.

Pour la suite du TP vous allez donc fonctionner par paire d'étudiants (X connecté sur la machine m1 et Y connecté sur la machine m2).

- X (Y) donne les droits de lecture de sa table concernant les employés à l'utilisateur Y (X)
- X (Y) donne les droits de modification de sa table DEPT à l'utilisateur Y (X)
- Vérifier que les privilèges ont été bien accordés en interrogeant les tables du méta-schéma appropriées.
- Testez vos nouveaux droits (les objets que vous interrogerez et dont vous n'êtes pas propriétaire sont désignés par leur nom complet nompropriétaire.nomobjet).
- Créez une vue sur une table sur laquelle X (Y) vous a donné des droits.

2.2 L'ordre REVOKE

Un utilisateur ayant accordé un privilège peut le reprendre à tout moment à l'aide de l'ordre REVOKE.

```
REVOKE privilege ON table/vue FROM utilisateur
```

— Enlever les privilèges précédemment accordés
— Vérifier que les privilèges ont bien été supprimés, notamment en interrogeant les tables sur lesquelles vous n'avez plus de droits et en consultant également les tables du méta-schéma comme USER_TAB_PRIVS.
— Que constatez vous par rapport à la vue précédemment créée ?

3. Gestion des accès concurrents

Une transaction (ensemble d'ordres SQL) est atomique c'est-à-dire qu'elle ne peut se terminer que par un succés (elle est alors validée) ou un échec (tous ses effets sont alors détruits).
En conséquence, en contexte multi-utilisateurs, les modifications effectuées par une transaction réalisée par un utilisateur ne sont connues des autres utilisateurs que lorsque la transaction a été confirmée par un `COMMIT`.
Oracle gère automatiquement les accès concurrents. Si une transaction est en train de modifier les lignes d'une table, les autres transactions peuvent accéder aux données telles qu'elles étaient avant ces dernières modifications (pas de temps d'attente pour la lecture).
Pour rester "simple" nous dirons que toute transaction pose des verrous sur les objets qu'elle manipule et que deux grands types de verrous existent :

— en lecture (verrou passant plusieurs lectures simultanées peuvent avoir lieu)
— en écriture (verrou bloquant la première écriture bloque les autres jusqu'à ce que le verrou soit relaché)

`Commandes qui provoquent un blocage implicite sur les tables et les lignes impliquées:`
DELETE, INSERT, UPDATE, ...

— Faites des sélections sur les mêmes lignes des mêmes tables avec deux noms utilisateurs différents
Par exemple, X et Y réalisent la même requête : Donnez le nom et la date d'embauche des employés sur la table EMP.
— Réessayez le même exercice mais avec des commandes provoquant des blocages
Par exemple, X modifie la table EMP : "Modifiez le nom de l'employé BALIN en BALUN". Puis X et Y réalisent la même requête : "Donnez le nom et la date d'embauche des employés" sur la table EMP. Constatations.
Tester avec COMMIT et ROLLBACK.

— Etreinte mortelle (DEADLOCK) :
- X fait un UPDATE sur le tuple i de la table EMP
- Y fait un UPDATE sur le tuple j de la table DEPT
- X fait un UPDATE sur le tuple j de la table DEPT
- Y fait un UPDATE sur le tuple i de la table EMP

Constatations. Quelle est la solution ? Quelles sont les opérations qui ont été effectivement effectuées sur les tables concernées ?