

# TRAVAUX PRATIQUES DE BASES DE DONNEES

## PARTIE A : CONCEPTION DU MCD ET MLD

Cette partie est à réaliser sur papier uniquement. Il s'agit de réaliser un modèle conceptuel de données (MCD) suivi du modèle logique de données (MLD) dans le modèle relationnel. Le cas étudié est celui d'une petite entreprise dont les règles de gestion sont énoncées ci-dessous. Dans la suite du TP, une partie de ce MLD sera implémenté sur le SGBD cible (Oracle, Access, etc.)

### Méthodologie :

- Réaliser des **sous schéma** (par exemple le sous schéma des employés avec les divisions) en indiquant chaque fois les règles de gestion utilisées. Penser à indiquer les cardinalités.
- Faire alors l'union des sous schémas et vérifier la cohérence globale.
- Appliquer les règles de vérification vues en cours.
- Passer alors au MLD (rappel, une association de cardinalité 1-1 fusionne avec l'entité)
- **ATTENTION** : Si certaines règles ou précisions vous manquent, spécifiez-les....

### Règles de gestion :

**Règle 1** : L'entreprise est organisée en divisions identifiées par un numéro, désignées par un nom. Chaque division est dirigée par un employé de l'entreprise.

**Règle 2** : Un employé est identifié par un matricule, nommé par son nom et prénom. Il travaille dans une seule division où il exerce une fonction qui peut être Président (1 pour l'entreprise), Directeur (1 par division) ou Vendeur. Un employé perçoit un salaire. Les informations suivantes sur les employées seront stockées : adresse complète, téléphone, date de naissance.

**Règle 3** : Un employé perçoit une prime sur les ventes qu'il réalise directement (vendeur) ou indirectement (directeur ou président). La prime directe s'élève à 5% des ventes, les primes indirectes s'élèvent à 1% des ventes.

**Règle 4** : Les clients de l'entreprise sont identifiés par un numéro. Les informations suivantes sur les clients seront stockées : nom, prénom, adresse complète, téléphone.

**Règle 5** : Une vente est identifiée par un numéro de vente. Elle est réalisée par un employé pour un client. De plus les informations suivantes seront stockées : date de la vente, date de livraison, adresse complète de livraison (incluant nom, prénom, adresse et téléphone), frais de port.

**Règle 6** : Une vente concerne un ou plusieurs produits. Pour chaque produit vendu, on indiquera la quantité vendue. Une remise peut être effectuée en fonction de cette quantité. C'est l'employé (vendeur) qui décide au cas par cas du taux de cette remise.

**Règle 7** : Un produit est identifié par un numéro. Il possède un nom et un prix unitaire. Il est fourni par un seul fournisseur.

**Règle 8** : Un fournisseur est identifié par un numéro. Les informations suivantes sur les fournisseurs seront stockées : nom, prénom, adresse complète, téléphone.

**Règle 9** : Une vente est livrée par un transporteur (DHL, la poste, etc.). Un transporteur est identifié par un numéro, possède un nom, un prénom, une adresse complète et un téléphone.

## Partie B : Installer / Configurer Oracle Express Edition V.10

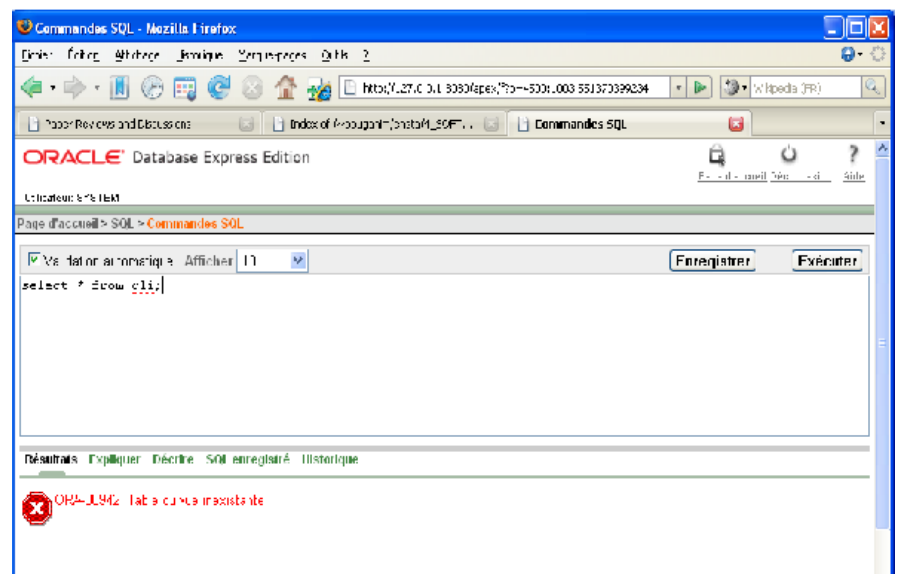
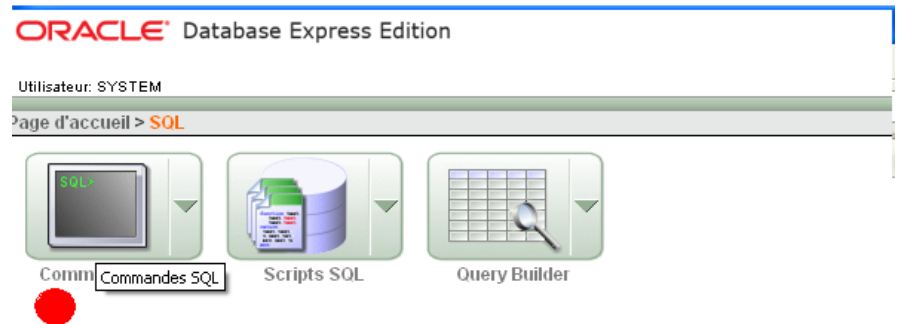
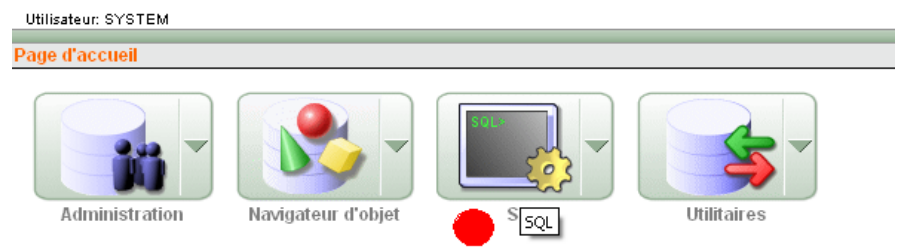
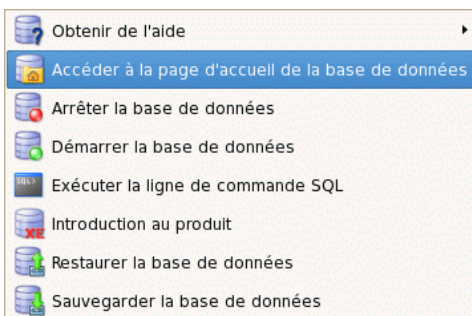
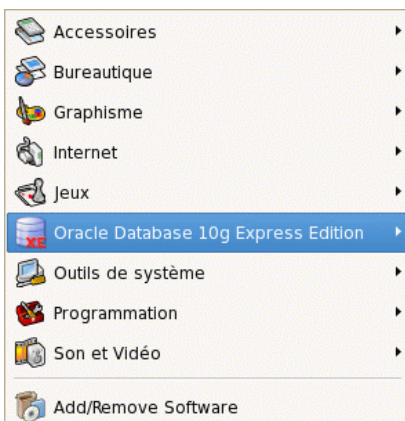
**Pour une installation windows :** Installer Oracle Express Edition V.10.

1. Récupérer le fichier d'installation
  - a. <http://www-smis.inria.fr/~bouganim/TP/oracle>
2. Installer : choisir toutes les options par défaut, utiliser comme password "oracle"
3. Créer l'utilisateur "TP" avec password "oracle"
  - a. Se connecter en tant qu'administrateur (login system – password oracle)
  - b. Créer l'utilisateur TP/oracle et lui donner tous les privilèges
  - c. Se déconnecter
4. Se connecter avec le login TP et password oracle

**IMPORTANT :** Pendant toute la durée du TP : Ouvrir une fenêtre éditeur (texte, word ou autre) et copier à chaque fois les requêtes envoyées à Oracle (dans leurs dernières versions) pendant le TP.

Le but est double :

- (1) Cela me permet de savoir ce que vous avez fait pour vous aider
- (2) Cela sert de compte rendu de TP.



## Partie C : Création / Modification du schéma / Insertion

Il s'agit de construire le schéma de la base. Nous nous intéresserons dans la suite du TP à une sous partie du schéma simplifié et restreint aux commandes. Pour minimiser la saisie, on utilisera comme noms des tables les noms abrégés (3 lettres).

Table <b>client</b> :	CLI	(NumCli, NomCli, Pays)
Table <b>commande</b> :	COM	(NumCom, NumCli, FraisPort, AnCom);
Table <b>détail commande</b> :	DET	(NumCom, NumPro, Qte, Remise)
Table <b>produit</b> :	PRO	(NumPro, NumFou, NomPro, TypePro)
Table <b>fournisseur</b> :	FOU	(NumFou, NomFou, Pays, Tel);

AnCom : Année de commande (ex 1992)  
FraisPort : Frais de port  
NomCli : Nom du client  
NomFou : Nom du fournisseur  
NomPro : Nom du produit

NumCli : Numéro de client  
NumCom : Numéro de la commande  
NumFou : Numéro de fournisseur  
NumPro : Numéro de produit  
Pays : Pays du client ou du fournisseur

Qte : Quantité commandée  
Remise : Remise effectuée  
Tel : Téléphone du fournisseur  
TypePro : Type de produit

- C1. Dessiner sur papier le schéma de la base. Faites ressortir les clés primaires (en souligné) et étrangères (en souligné et pointillé) – Il servira tout au long du TP.
- C2. Créer les relations de la base ci-dessus (pensez aux clés primaires et étrangères)
- C3. Quel ordre faut-il respecter lors de la création de ces tables ? Est-ce le seul possible ?
- C4. Vérifier vos créations : Utiliser la commande : **desc <nomtable>**
- C5. Modifier un type précédemment défini : NomCli devient char(30).
- C6. Ajouter un attribut Téléphone (Tel) pour les clients
- C7. Ajouter le prix unitaire (PrixUnit) pour les produits
- C8. Rajoutez une contrainte de non nullité à l'attribut NomPro
- C9. Insérer un jeu de données cohérent dans vos relations (un ou deux tuples par relation)
- C10. Essayez l'interface d'Oracle Express (cf. Navigateur d'objets)
- C11. Vider toutes vos tables. Quel ordre faut-il respecter? Est-ce le seul possible ?
- C12. Supprimer toutes vos tables. Quel ordre faut-il respecter? Est-ce le seul possible ?

- Exemple : **create table** PRO (  
    NumPro **number(5) constraint Cle\_Pro primary key**,  
    NumFou [**number(2)**] **constraint Pro\_Ref\_Fou references FOU**,  
    NomPro **char(20) constraint NomPro\_Not\_Null not null**);
- Les types disponibles sont : number(n), char(n), date, number(m,n)
- Faire attention à la cohérence de type des attributs de jointures (même domaine).
- Modifier le type des attributs : ALTER TABLE <nom table> MODIFY ( <attr> newtype, ...)
- Ajouter un attribut à une relation : ALTER TABLE <nom table> ADD ( <newattr> type, ...)

## Partie D : Insertion massive de données

**Mise en place :** Récupérer les fichiers CREATE.sql, CLI.ctl, COM.txt, DET.ctl, PRO.ctl, FOU.ctl  
(<http://www-smis.inria.fr/~bouganim/TP/insertion>)

### Création de la base à l'aide d'un script

D1. Exécutez le script CREATE.sql (Utilisez l'interface à cet effet)

### Chargement massif de données : Utilisation de SQL Loader \*

SQL loader est un utilitaire Oracle permettant d'importer massivement des données depuis un fichier au format texte. Une entête de fichier permet de préciser le format de ce dernier. Le chargement d'un fichier texte a les effets suivants :

- Chargement dans la base des données valides
- Création d'un fichier de log indiquant le déroulement des opérations
- Création éventuelle d'un fichier .bad contenant les données invalides

D2. Quatre des cinq fichiers fournis comportent l'entête (extension ctl). Rajoutez l'entête du cinquième fichier.

D3. Chargez les cinq relations en se basant sur l'exemple ci-dessous

**C:\oracle\app\oracle\product\10.2.0\server\BIN\sqlldr TP/oracle c:\TP\CLI.ctl c:\TP\CLI.log**

D4. Vérifier que vous avez l'intégralité des données :

**91 Clients, 1078 Commandes, 2809 Détails, 77 Produits et 29 Fournisseurs.**

D5. Trouvez les erreurs dans les fichiers, corrigez les

D6. Supprimez le contenu des tables incomplètes

D7. Rechargez les données valides.

#### Exemples :

<b>LOAD DATA</b>	<i>(Charger des données)</i>
<b>INFILE *</b>	<i>(les données se trouvent dans ce fichier)</i>
<b>APPEND</b>	<i>(On veut les ajouter dans la table...)</i>
<b>INTO TABLE CLI</b>	<i>(c'est dans la table CLI)</i>
<b>FIELDS TERMINATED BY ","</b>	<i>(Attention "" → quote/guillemet/quote)</i>
<b>OPTIONALLY ENCLOSED BY ""</b>	<i>(champs terminés par des ;)</i>
	<i>(... encadrés éventuellement par des " )</i>
<b>(CodeCli,NomCli,Pays,Tel)</b>	<i>(liste des attributs indiqués ci-dessous)</i>
<b>BEGINDATA</b>	<i>(les données commencent après cet inst°)</i>
<b>1, Maria,Pologne,4867427275</b>	<i>(1<sup>er</sup> tuple codecli = 1, nomcli = Maria, etc.)</i>
<b>2,"Ana",Autriche", "4905219433"</b>	<i>(2<sup>ème</sup> tuple...)</i>
<b>3,"Antonio", "Espagne", "7062001573"</b>	<i>(3<sup>ème</sup> tuple ...)</i>

## Partie E : Insertion / Mise à jour / suppression de données

- E1. Mettre en majuscule les noms des clients (fonction UPPER)
- E2. Multiplier par deux les remises associées aux commandes passées après 1996
- E3. Supprimer les clients français
- E4. Supprimer les clients qui ont commandé avant 1991
- E5. Récupérer le fichier CLEAN.sql sur <http://www-smis.inria.fr/~bouganim/TP/manip>
- E6. Exécutez le script CLEAN.sql (Utilisez l'interface à cet effet)

## Partie F : Manipulation des données

**Répondez aux questions suivantes en interrogeant votre base :**

- F1. Liste des clients français triés par ordre alphabétique ?
- F2. Noms des clients français ou anglais ?
- F3. Quels sont les produits vendus à "Antonio" ?
- F4. Quels sont les fournisseurs de produits de prix < 50 FF ?
- F5. Y a-t-il des fournisseurs de produits de prix < 50 FF et supérieur à 400 FF ?
- F6. Nom des fournisseurs qui fournissent le produit 'Chef' le moins cher ?
- F7. Y a-t-il des fournisseurs qui sont clients (même nom) ?
- F8. Quels sont les fournisseurs qui ne fournissent que des produits de prix < 50 FF ?
- F9. Quels sont les fournisseurs qui ne fournissent aucun produits de prix < 50 FF ?
- F10. Noms des clients qui n'ont pas passé de commandes ?
- F11. Couples des numéros de clients différents ?
- F12. Nombre de produits dans la base
- F13. Pour chaque type de produit, donner le prix moyen et le prix max. ?
- F14. Même question mais uniquement pour les types où il y a plus de 10 produits.
- F15. Donner pour chaque fournisseur la liste des noms des produits par ordre alphabétique
- F16. Donner pour chaque client, la liste des produits commandés et leur nombre pour l'ensemble des commandes passées.
- F17. Calculez, pour chaque ligne de commande, le prix total à payer. (Penser à la remise et à la quantité)
- F18. Calculez le total à payer par commande
- F19. Calculez le total à payer par client
- F20. Quels sont les clients qui ont commandé plus de 10000 FF ?

## Partie G : JDBC et Injection SQL

### Objectif :

L'objectif est de se familiariser avec 2 aspects importants pour le développement d'applications utilisant des bases de données.

Tout d'abord, l'appel d'une base de données depuis JAVA. Pour cela, un exemple simple, permettant de débiter est fourni. Sur la base de cet exemple, il vous est demandé de réaliser une application simpliste.

Ensuite, vous tenterez de réaliser une attaque au processus d'identification développé, montrant qu'une mauvaise programmation peut générer des failles importantes.

Docs et programmes sur <http://www-smis.inria.fr/~bouganim/TP/jdbc>

**Installation :** Si vous avez des outils pour développer en JAVA, vous pouvez les utiliser, sinon, vous pouvez utiliser Eclipse. Sur un PC windows, vous pouvez télécharger Eclipse (j2re-1\_4\_2\_13-windows-i586-p.exe). Si vous ne connaissez ni Java, ni Eclipse, lisez [http://www-smis.inria.fr/~bouganim/TP/jdbc/IMPORTANT\\_ECLIPSE.pdf](http://www-smis.inria.fr/~bouganim/TP/jdbc/IMPORTANT_ECLIPSE.pdf)

- G1. Copiez l'exemple 'TestDB' fourni – compilez le et testez le (après avoir notamment vérifié / modifié les paramètres de connexion)
- G2. Créez une table USER contenant deux attributs Login et Password
- G3. Insérez quelques tuples
- G4. En java, Sur le modèle de TestDB, construisez un petit programme qui
  - se connecte à la base de données comme l'utilisateur «TP»
  - demande un login (saisi au clavier) et un mot de passe (saisi au clavier cf. <http://www-smis.inria.fr/~bouganim/TP/jdbc/Exemples>)
  - va vérifier dans la table USER, l'existence du couple login/password
  - si le login/mot de passe existe, demande un type de produit et affiche le nom des produits correspondants.
- G5. Réalisez une attaque par injection de code SQL sur votre petit programme. Sans modifier le programme JAVA, contournez le mécanisme d'authentification créé en G4
- G6. Par injection, récupérez, sans modifier le programme JAVA, le nom des clients
- G7. Corrigez votre programme pour qu'il soit résistant à ces attaques.

## Partie H : Concurrence d'accès dans Oracle

### Préparation

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

Rajouter un attribut numérique à votre table CLI (champ nommé <Solde>)

Créer 1 client nommés Joe (attention au majuscules/minuscules)

Récupérez 3 fichiers, nommés Solde.sql, Depot.sql et Retrait.sql

<http://www-smis.inria.fr/~bouganim/TP/concurrence>

```
Solde.sql      PROMPT 'Solde de Joe';
               SELECT Solde FROM CLI where NomCLI= 'Joe';

Depot.sql      PROMPT 'Un dépôt de 6000 FF est fait sur le compte de Joe'
               UPDATE CLI SET solde = solde + 6000 WHERE NomCLI = 'Joe';
               PROMPT 'Nouveau Solde de Joe';
               SELECT Solde FROM CLI where NomCLI = 'Joe';

Retrait.sql    PROMPT 'Joe retire 4000 FF';
               UPDATE CLI SET solde = solde - 4000 WHERE NomCLI = 'Joe';
               PROMPT 'Nouveau Solde de Joe';
               SELECT Solde FROM CLI where NomCLI = 'Joe';
```

Ouvrez deux fenêtres SQLPLUS (C:\oraclexe\app\oracle\product\10.2.0\server\BIN\sqlplus).

Chaque session est considérée par ORACLE comme un utilisateur, et on a donc 2 utilisateurs, nommés 1 et 2, en situation de concurrence. Dans tout ce qui suit, on note *INSTR<sub>i</sub>* l'exécution de l'instruction *INSTR* par l'utilisateur *i*. Par exemple Depot<sub>1</sub> correspond à l'exécution du fichier Depot dans la première fenêtre par la commande @Depot. On note de même *ROL<sub>i</sub>* et *COM<sub>i</sub>* l'exécution des commandes rollback; et commit; dans la fenêtre *i*.

**Expérimentez les exécutions suivantes et essayez de comprendre le fonctionnement du verrouillage d'Oracle.**

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

**COM<sub>1</sub>, COM<sub>2</sub>, Solde<sub>1</sub>, Solde<sub>2</sub>, Retrait<sub>1</sub>, Solde<sub>2</sub>, ROL<sub>1</sub>, Solde<sub>1</sub>, Solde<sub>2</sub>.**

2. idem, mais avec un *commit* :

**COM<sub>1</sub>, COM<sub>2</sub>, Solde<sub>1</sub>, Solde<sub>2</sub>, Retrait<sub>1</sub>, Solde<sub>2</sub>, COM<sub>1</sub>, Solde<sub>1</sub>, Solde<sub>2</sub>,**

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

**COM<sub>1</sub>, COM<sub>2</sub>, Solde<sub>1</sub>, Solde<sub>2</sub>, Retrait<sub>1</sub>, Solde<sub>2</sub>, Depot<sub>2</sub>, Solde<sub>1</sub>, COM<sub>1</sub>, COM<sub>2</sub>.**

**Expérimentez les exécutions précédentes en spécifiant le mode suivant**  
**APRES CHAQUE COMMIT : SET TRANSACTION ISOLATION LEVEL SERIALIZABLE**

**Comparez et expliquez comment Oracle verrouille les données.**