# Exercices sur les concepts de base

## Le dictionnaire des données

Exécuter le script `CreationBaseInfosoft.sql`, en utilisant l'utilisateur `INFOSOFT`. Il vous faudra donc le créer à
l'aide du script de création d'utilisateurs `create_user.sql`.

### Dictionnaire 1

Consulter la vue `USER_USERS` et afficher :

- Votre nom d'utilisateur en lettres minuscules, sauf la première lettre en majuscule;
- La date et l'heure de création de l'utilisateur;
- Le nom du tablespace par défaut;
- La date d'expiration de l'utilisateur, si cette date n'est pas spécifiée, ajoutez 2 ans à la date de création.

```
select INITCAP(USERNAME) as "Nom de l'utilisateur",
       CREATED           as "Date de creation",
       DEFAULT_TABLESPACE   "Tbl Sp Dfl",
       DEFAULT_TABLESPACE   "Date d'expiration"
from user_users;
```

Exécuter la commande `DESCRIBE USER_USERS` pour afficher les noms et les types des colonnes de la vue `USER_USERS`

```
DESCRIBE USER_USERS;
```

### Dictionnaire 2

Rechercher par ordre alphabétique toutes les vues accessibles. Ne sélectionnez que les vues dont le nom commence
par `USER_` et n'affichez que le nom des cinq premières vues `ALL_OBJECTS`
Remarque : l'évaluation de la commande `SELECT` se fait dans l'ordre `FROM, WHERE, SELECT, ORDER BY`.
Dans cet exercice
il faut d'abord trier avant de sélectionner 5 lignes.

> Attention au caractère `_`, il a une signification.

```
select *
from (select OBJECT_NAME
      from ALL_OBJECTS
      where upper(OBJECT_NAME) like 'USER@_%' ESCAPE '@'
          and upper(OBJECT_TYPE) = 'VIEW'
      order by OBJECT_NAME)
where ROWNUM <= 5;
```

OU

```
SELECT object_name
FROM all_objects
WHERE object_type = 'VIEW'
  AND object_name LIKE 'USER@_%' ESCAPE '@'
ORDER BY 1
    FETCH FIRST 5 ROWS ONLY;
```

## Dictionnaire 3

Afficher le nom de tous les index que vous avez créés. (`USER_INDEXES`). Spécifier le nom de la table sur lequel
l'index a été créé et spécifier `OUI` si les valeurs de l'index sont uniques ou `NON` si les valeurs de l'index ne sont
pas uniques. La colonne `uniqueness` de la vue `USER_INDEXES` peut contenir les valeurs `UNIQUE` ou `NONUNIQUE`. Le
résultat sera trié par le nom de la table.

Format du résultat demandé (respecter le nom des colonnes) :

| Nom de la table | Nom de l'index | Unicité |
|---|---|---|

```
select TABLE_NAME                                              as "Nom de la table",
       INDEX_NAME                                              as "Nom de l'index",
       case when UNIQUENESS = 'UNIQUE' then 'Oui' else 'Non' end as "Unicité"
from USER_INDEXES
order by TABLE_NAME;
```

## Dictionnaire 4

Afficher le nom, la date de création, le type et le statut des objets créés dans votre schéma.
(`USER_OBJECTS`).
Spécifier si l'objet est valide ou invalide par `YES` ou `NO` et spécifier la date sous la forme *jour-mois-année heure:*
*minute* soit, par exemple, `04-02-2019 08:40`.
Le résultat sera trié par type de l'objet et nom de l'objet

| Type de l'objet | Nom de l'objet | Date de Création | Statut |
|---|---|---|---|

```
select OBJECT_TYPE                                          as "Type de l'objet",
       OBJECT_NAME                                          as "Nom de l'objet",
       to_char(CREATED, 'DD-MM-YYYY HH24:MI')               as "Date de Création",
       case when STATUS = 'VALID' then 'YES' else 'NO' end as "Statut"
from USER_OBJECTS
order by OBJECT_TYPE, OBJECT_NAME;
```

## Dictionnaire 5

Consulter les vues `USER_CONSTRAINTS` et `USER_CONS_COLUMNS` pour afficher toutes les contraintes de colonnes spécifiées
sur une table, dont le début, du nom est donné en paramètre, en lettres majuscules ou lettres minuscules. On affichera :
Le nom de la colonne, le nom de la contrainte, le type de contrainte : `CHECK` pour `C`, `PRIMARY` pour `P`, `REFERENTIEL` pour `R`
et `UNIQUE` pour `U`, si la contrainte est différée ou non (`YES` / `NO`), le statut de la contrainte ainsi que la condition de la contrainte. Le résultat sera trié par le nom de la colonne.

| Nom de la colonne | Nom de la contrainte | Type | Différée | Statut | La condition |
|---|---|---|---|---|---|

```
SELECT USER_CONS_COLUMNS.column_name     as "Nom de la colonne",
       USER_CONSTRAINTS.constraint_name  as "Nom de la contrainte",
       (case
            when CONSTRAINT_TYPE like 'C' then 'CHECK'
            when CONSTRAINT_TYPE like 'P' then 'PRIMARY'
            when CONSTRAINT_TYPE like 'R' then 'REFERENTIEL'
            when CONSTRAINT_TYPE like 'U' then 'UNIQUE' end)
                                     as "Type",

       (case
            when DEFERRED = 'IMMEDIATE' then 'NO'
            else 'YES'
           end)                      as "Différée",
       status                        as "Statut",
       USER_CONSTRAINTS.search_condition AS "La condition"
from USER_CONSTRAINTS
         left join USER_CONS_COLUMNS on USER_CONSTRAINTS.constraint_name =
USER_CONS_COLUMNS.constraint_name
where upper(USER_CONS_COLUMNS.table_name) like upper('&EMPLOYES%')
order by "ColumnName";
```

## Dictionnaire 6

Générer dans un fichier `DropTable.sql`, les commandes qui permettent d'effacer toutes les tables de votre schéma. Le
fichier `DropTable.sql` ne doit contenir que les commandes `DROP`. Spécifier les commandes pour supprimer les entêtes et
les bas de page. Écrire les commandes et le afficher le contenu du fichier `DropTable.sql`.

```
SET
    ECHO OFF
SET FEEDBACK OFF
SET HEADING OFF
SET TERMOUT OFF
```

```
SET VERIFY OFF
SET NEWPAGE NONE
    spool DropTable.sq

SELECT ' DROP TABLE ' || table_name || ' CASCADE CONSTRAINTS;'
FROM user_tables;

SPOOL
OFF

SET FEEDBACK ON
SET HEADING ON
SET TERMOUT ON
SET VERIFY ON
SET ECHO ON
```

# L'architecture à niveaux d'une BD

Exécuter le fichier `ListeEmployesProjets.sql`. Celui-ci contient une procédure stockée. Cette procédure affiche le nom
et le *prénom* des employés ainsi que le *nom* du projet sur lequel ils travaillent.

## Concept de base 1

Afficher le statut, la date et l'heure de création de la procédure ListeEmpPro (Vue USER_OBJECTS).

```
SELECT SUBSTR(OBJECT_NAME, 1, 20)               AS "Nom de l'objet",
       TO_CHAR(CREATED, 'DAY DD/MM/YYYY HH:MI') AS "Date de création",
       STATUS
FROM USER_OBJECTS
WHERE upper(OBJECT_TYPE) = 'PROCEDURE'
  AND upper(OBJECT_NAME) = 'LISTEEMPPRO';
```

Ajouter une colonne à la table `Employes` et spécifier quel niveau du schéma de la base a été modifié.

```
ALTER TABLE Employes
    ADD Prenom2 VARCHAR2(30);
```

> Modification du niveau logique

Afficher le statut de la procédure ListeEmpPro et expliquer le résultat obtenu

```
SELECT STATUS
FROM USER_OBJECTS
WHERE upper(OBJECT_TYPE) = 'PROCEDURE'
  AND upper(OBJECT_NAME) = 'LISTEEMPPRO';
```

> Le statut est à nouveau `VALID`, car le serveur Oracle recompile implicitement tout objet `INVALID` lorsque ce dernier est à nouveau appelé.

Exécuter la procédure `ListeEmpPro` et expliquer pourquoi, après l'ajout de la colonne dans la table `Employes` et sans
modifier la procédure `ListeEmpPro` cette procédure affiche des résultats corrects.

> La procédure fait appel à trois champs des tables `employes`, `emppro` et `projets`. Ceux-ci n'ont pas été impactés par l'ajout d'un champ supplémentaire dans la table `Employes`. Indépendance logique : on peut modifier le schéma logique (dont les tables) sans modifier les programmes ! Il y a donc bien indépendance données/programmes.

## Concept de base 2

Créer un index pour spécifier l'unicité du *nom* et *prénom* de l'employé et spécifier quel niveau du schéma de la base
a été modifié.

```
CREATE UNIQUE INDEX IndexEmployesNomPrenom ON Employes (Nom, Prenom);
```

> On modifie le niveau physique

Pourquoi crée-t-on des index?

> Pour accélérer la recherche sur base de nom et prénom

Exécuter la procédure `ListeEmpPro` et expliquer pourquoi, après la création de l'index la procédure `ListeEmpPro`
affiche des résultats corrects.

> Parce que le nouvel index n'a pas d'influence sur les données et les champs auxquels la procédure accède. Le statut de
> la procédure est toujours VALID (même sans l'exécuter, simplement en regardant son statut dans `USER_OBJECTS`). Les index sont au niveau interne = schéma physique or, l'indépendance physique dit que l'on peut modifier le schéma physique sans modifier le schéma logique (et donc les programmes).

## Concept De Base 3

Ajouter une contrainte sur la table `Employes` et spécifier quel niveau du schéma de la base a été modifié.

```
ALTER TABLE employes
    ADD CONSTRAINT EmployesLongNom
        CHECK (LENGTH(NOM) > 3);
```

> Le niveau logique a été modifié. Les contraintes sont au même niveau que les tables, donc au niveau du schéma logique. Or, l'indépendance logique dit que l'on peut modifier le schéma logique sans modifier les programmes.

Exécuter la procédure `ListeEmpPro` et expliquer pourquoi, après la création de la contrainte, la procédure `ListeEmpPro` affiche des résultats corrects.

> Le statut est toujours `valid`. L'exécution se déroule de la même manière qu'au départ. Parce que la contrainte sera utilisée lors de l'insertion de valeurs dans les champs, elle n'a pas d'impact sur les données utilisées par la requête de la procédure.

## Concept de base 4

Supprimer la colonne `nom` de la table `Employes` et spécifier quel niveau du schéma de la base a été modifié.

```
ALTER TABLE Employes
    DROP COLUMN nom;
```

> Modification du niveau logique (c'est la que se situent les tables)

Afficher le statut de la procédure `ListeEmpPro` et expliquer le résultat obtenu. Expliquer pourquoi, après la suppression de la colonne `nom` de la table `Employes` la procédure `ListeEmpPro` affiche un message d'erreur.

> La procédure devient invalide La procédure ne peut plus fonctionner correctement, car un des champs utilisés dans la requête de la procédure n'existe plus dans la table.