



République Algérienne Démocratique et
Populaire



Ministère de l'Enseignement Supérieur et de la
Recherche Scientifique

Université de sciences et de la technologie de Houari Boumediene

Faculté d'Informatique Département IA_SD

Spécialité:

Master Systèmes Informatiques Intelligents (M1)

Architecture de S.G.B.D. relationnels Dictionnaire Oracle

Rapport de TP3 présenté par **FERKOUS Sarah**

Matricule d'étudiante : **191931043867**

Année universitaire **2022-2023**

Introduction Générale

1. Objectif du TP

Ce TP va nous permettre d'étudier la structure centralisée (dictionnaire de données) sur laquelle repose **Oracle**. La compréhension du fonctionnement du dictionnaire est essentielle à l'administration des bases de données **Oracle**. Elle sera utile pour contrôler notre travail lors de la création d'un schéma de base de données **Oracle**. Au cours de ce TP, nous allons nous intéresser plus particulièrement aux vues de type **ALL** et **USER**. Pour vous guider dans vos travaux, vous trouvez dans le tableau suivant l'ensemble des vues du dictionnaire de données Oracle de type **USER**.

2. Présentation : Dictionnaire de données

Le dictionnaire de données **ORACLE** est une structure centralisée qui contient la description de tous les objets (tables, vues, utilisateurs, ...) gérés par le **SGBD**. Il présente la particularité d'être lui-même organisé comme une base de données (on parle de **méta-base**) et d'être par conséquent accessible directement à travers SQL. Pour des raisons d'intégrité, la plupart des tables du dictionnaire ne sont pas modifiables par les utilisateurs (seul l'administrateur de la base **SYS**, à accès à ces dernières). A l'opposé, l'accès à leur contenu est généralement autorisé.

- Les tables du dictionnaire des données

Le dictionnaire de données est un ensemble de tables appartenant à l'utilisateur **SYS**. Ces tables contiennent la définition complète de la base de données. Il est maintenu à jour automatiquement par le SGBD Oracle lors de l'exécution d'ordres **DDL** (create, drop, alter,...).

- Les vues du dictionnaire des données

Le contenu des tables du dictionnaire de données est accessible par l'intermédiaire de vues. Elles sont de quatre grands types :

- ✓ Vues relatives aux objets d'un utilisateur : **USER_***
- ✓ Vues relatives aux objets accessibles à un utilisateur : **ALL_***
- ✓ Vues relatives aux administrateurs : **DBA_***
- ✓ Vues relatives au suivi des performances : **V\$***

USER : l'environnement propre de l'utilisateur, il ne peut voir que ce qui lui appartient

Exemple : SELECT sur une vue USER du dictionnaire.

ALL : l'environnement propre de l'utilisateur plus les objets auquel il a accès. Exemple :

SELECT sur une vue ALL du dictionnaire.

DBA : informations générales sur la base, ne peuvent être accédés que par un utilisateur de type DBA et doivent être préfixées par SYS.DBA. Exemple : SELECT sur une vue DBA du dictionnaire.

Réponse aux questions

1. Connecter en tant que « System ». Lister le catalogue « **DICT** ». Il contient combien d'instances ? Donner sa structure ?

Requête :

On se connecte en tant que « **system** » après on exécute la requête :

Select * from DICT ;

Résultat :

Cette requête donne un résultat de plus de 1000 lignes, voici donc une petite partie du résultat

```
Enter user-name: system
Enter password:

Connected to:
Oracle Database 10g Express Edition Release 10.2.0.1.0 - Production

SQL> select * from DICT;

TABLE_NAME
-----
COMMENTS
-----
DBA_ROLES
All Roles which exist in the database

DBA_PROFILES
Display all profiles and their limits

USER_RESOURCE_LIMITS
Display resource limit of the user

TABLE_NAME
-----
COMMENTS
-----
USER_PASSWORD_LIMITS
Display password limits of the user

USER_CATALOG
Tables, Views, Synonyms and Sequences owned by the user

ALL_CATALOG
All tables, views, synonyms, sequences accessible to the user

TABLE_NAME
-----
COMMENTS
-----
DBA_CATALOG
All database Tables, Views, Synonyms, Sequences

USER_CLUSTERS
```

[Requête](#) :

Select TABLE_NAME from DICT;

[Résultat](#) :

La liste des tables qui peuvent exister:

```
SQL>
SQL> select TABLE_NAME from DICT;

TABLE_NAME
-----
ALL_XML_SCHEMAS
ALL_XML_SCHEMAS2
DBA_ROLES
DBA_PROFILES
USER_RESOURCE_LIMITS
USER_PASSWORD_LIMITS
USER_CATALOG
ALL_CATALOG
DBA_CATALOG
USER_CLUSTERS
ALL_CLUSTERS

TABLE_NAME
-----
DBA_CLUSTERS
USER_CLU_COLUMNS
DBA_CLU_COLUMNS
USER_COL_COMMENTS
ALL_COL_COMMENTS
DBA_COL_COMMENTS
USER_COL_PRIVS
ALL_COL_PRIVS
DBA_COL_PRIVS
USER_COL_PRIVS_MADE
ALL_COL_PRIVS_MADE

TABLE_NAME
-----
USER_COL_PRIVS_RECD
ALL_COL_PRIVS_RECD
DBA_ENCRYPTED_COLUMNS
ALL_ENCRYPTED_COLUMNS
USER_ENCRYPTED_COLUMNS
USER_DB_LINKS
ALL_DB_LINKS
DBA_DB_LINKS
DBA_EXP_OBJECTS
DBA_EXP_VERSION
DBA_EXP_FILES
```

[Requête](#) :

Select count(*) from DICT;

[Résultat](#) :

Il contient **1821** instances.

```
SQL> select count(*) from DICT;

COUNT(*)
-----
      1821
```

[Requête](#) :

DESC DICT ;

[Résultat](#) :

Le dictionnaire contient t les noms de toutes les structures de la base de données ainsi que des commentaires sur ces dernières : **TABLE_NAME** et **COMMENTS** de type VARCHAR.

```
SQL>
SQL> DESC DICT;
Name                                Null?    Type
-----
TABLE_NAME                          VARCHAR2(30)
COMMENTS                            VARCHAR2(4000)
SQL>
```

Détaille de la question

i Le dictionnaire de données **Oracle** représente le cœur de la base de données. Il s'agit d'un ensemble de tables systèmes contenant les informations relatives à la structure de la base de données :

- Utilisateurs de la base (ainsi que leurs privilèges et leur rôle)
- Noms et caractéristiques des objets contenus dans la base (tables, vues, index, clusters, triggers, packages, ...)
- Contraintes d'intégrité
- Ressources physiques allouées à la base
- ...

le dictionnaire est créé au moment de la création de la base et est mis à jour.

Il appartient à l'utilisateur **SYS**, mais l'utilisateur **SYSTEM**, c'est-à-dire l'administrateur de la base, possède des droits de lecture sur des vues du dictionnaire. Enfin le dictionnaire de données est conservé dans le **tablespace SYSTEM**.

Le dictionnaire de données sert principalement dans deux situations :

- afin de vérifier la syntaxe et les privilèges sur une requête **SQL** (pour le **DDL**, *Data Definition Language*)
- Afin d'obtenir des informations sur la structure de la base de données

2. Donner le rôle et la structure des tables (ou vues) suivantes : **ALL_TAB_COLUMNS**, **USER_USERS**, **ALL_CONSTRAINTS** et **USER_TAB_PRIVS**.

On exécute d'abord la commande **DESCRIBE** pour chaque table, a fin de visualiser leur structure.

❖ **ALL_TAB_COLUMNS :**

Décrit soit les colonnes d'une table, soit les valeurs actuelles, ainsi que les valeurs par défaut des propriétés de zone de préparation d'une table.

```
SQL>
SQL> DESCRIBE ALL_TAB_COLUMNS;
Name                                         Null?    Type
-----
OWNER                                         NOT NULL VARCHAR2(30)
TABLE_NAME                                   NOT NULL VARCHAR2(30)
COLUMN_NAME                                  NOT NULL VARCHAR2(30)
DATA_TYPE                                     VARCHAR2(106)
DATA_TYPE_MOD                                VARCHAR2(3)
DATA_TYPE_OWNER                             VARCHAR2(30)
DATA_LENGTH                                  NOT NULL NUMBER
DATA_PRECISION                               NUMBER
DATA_SCALE                                  NUMBER
NULLABLE                                     VARCHAR2(1)
COLUMN_ID                                    NUMBER
DEFAULT_LENGTH                              NUMBER
DATA_DEFAULT                                LONG
NUM_DISTINCT                                NUMBER
LOW_VALUE                                    RAW(32)
HIGH_VALUE                                   RAW(32)
DENSITY                                      NUMBER
NUM_NULLS                                    NUMBER
NUM_BUCKETS                                  NUMBER
LAST_ANALYZED                               DATE
SAMPLE_SIZE                                 NUMBER
CHARACTER_SET_NAME                          VARCHAR2(44)
CHAR_COL_DECL_LENGTH                        NUMBER
GLOBAL_STATS                                VARCHAR2(3)
USER_STATS                                  VARCHAR2(3)
AVG_COL_LEN                                  NUMBER
CHAR_LENGTH                                  NUMBER
CHAR_USED                                    VARCHAR2(1)
V80_FMT_IMAGE                              VARCHAR2(3)
DATA_UPGRADED                              VARCHAR2(3)
HISTOGRAM                                    VARCHAR2(15)
```

❖ USER_USERS :

Contient les informations sur tous les utilisateurs.

```
SQL>
SQL> DESCRIBE USER_USERS;
Name                               Null?    Type
-----
USERNAME                           NOT NULL VARCHAR2(30)
USER_ID                            NOT NULL NUMBER
ACCOUNT_STATUS                     NOT NULL VARCHAR2(32)
LOCK_DATE                          DATE
EXPIRY_DATE                        DATE
DEFAULT_TABLESPACE                 NOT NULL VARCHAR2(30)
TEMPORARY_TABLESPACE              NOT NULL VARCHAR2(30)
CREATED                           NOT NULL DATE
INITIAL_RSRC_CONSUMER_GROUP        VARCHAR2(30)
EXTERNAL_NAME                      VARCHAR2(4000)
```

❖ ALL_CONSTRAINTS :

Contient les informations sur toutes les contraintes définies dans la base de données.

```
SQL>
SQL> DESCRIBE ALL_CONSTRAINTS;
Name                               Null?    Type
-----
OWNER                             NOT NULL VARCHAR2(30)
CONSTRAINT_NAME                   NOT NULL VARCHAR2(30)
CONSTRAINT_TYPE                   VARCHAR2(1)
TABLE_NAME                       NOT NULL VARCHAR2(30)
SEARCH_CONDITION                  LONG
R_OWNER                          VARCHAR2(30)
R_CONSTRAINT_NAME                 VARCHAR2(30)
DELETE_RULE                       VARCHAR2(9)
STATUS                           VARCHAR2(8)
DEFERRABLE                       VARCHAR2(14)
DEFERRED                         VARCHAR2(9)
VALIDATED                        VARCHAR2(13)
GENERATED                        VARCHAR2(14)
BAD                              VARCHAR2(3)
RELY                             VARCHAR2(4)
LAST_CHANGE                      DATE
INDEX_OWNER                      VARCHAR2(30)
INDEX_NAME                      VARCHAR2(30)
INVALID                         VARCHAR2(7)
VIEW_RELATED                    VARCHAR2(14)
```

❖ USER_TAB_PRIVS :

Rassemble les informations concernant les privilèges des utilisateurs sur les tables de la base de données.

```
SQL>
SQL> DESCRIBE USER_TAB_PRIVS;
Name                                         Null?    Type
-----
GRANTEE                                     NOT NULL VARCHAR2(30)
OWNER                                       NOT NULL VARCHAR2(30)
TABLE_NAME                                 NOT NULL VARCHAR2(30)
GRANTOR                                     NOT NULL VARCHAR2(30)
PRIVILEGE                                  NOT NULL VARCHAR2(40)
GRANTABLE                                  VARCHAR2(3)
HIERARCHY                                  VARCHAR2(3)
```

Le rôle des 4 tables :

	Name	rôle
TABLE 1	ALL_TAB_COLUMNS	description des colonnes des tables, views et clusters accessible à l'utilisateur actuel
TABLE 2	USER_USERS	caractéristiques générales du user : Nom, tablespace par défaut, tablespace temporaire
TABLE 3	ALL_CONSTRAINTS	décrit les définitions de contraintes sur les tables accessibles à l'utilisateur actuel.
TABLE 4	USER_TAB_PRIVS	Des privilèges donnés ou reçus : Bénéficiaire, propriétaire, créateur

3. Trouver le nom d'utilisateur avec lequel vous êtes connecté ?

Comme on a vu dans la question précédèrent la table **user_users** contient la colonne **username** qui va stocker le nom d'utilisateur connecté.

Dans ce cas on est connecté avec l'utilisateur **system**.

Requête :

On se connecte en tant que « **system** » après on exécute la requête :

Select username from user_users ;

Résultat :

Cette requête affiche l'utilisateur system.

```
SQL>
SQL> select username from user_users;

USERNAME
-----
SYSTEM
```


4. Comparer la structure et le contenu des tables `ALL_TAB_COLUMNS` et `USER_TAB_COLUMNS` ?

Requête :

Afin de pouvoir comparer ces 2 tables on commence par les visualiser comme première étape avec la commande **DESCRIBE nom-table** (voir leurs structure), après on affiche le contenu des 2 table avec la commande **select * from nom_table**.

Remarque :

Avec **select *** le résultat ne sera pas vraiment visible c'est pour ça on choisit quelque colonnes à afficher : **owner, table_name, column_name**.

Les 2 tables contiennent la même structure sauf que la table `ALL_TAB_COLUMNS` a une colonne en plus qui est **owner**, et pas le même contenu,

Résultat :

```
SQL>
SQL> describe ALL_TAB_COLUMNS;
Name                                     Null?      Type
-----
OWNER                                   NOT NULL   VARCHAR2(30)
TABLE_NAME                             NOT NULL   VARCHAR2(30)
COLUMN_NAME                             NOT NULL   VARCHAR2(30)
DATA_TYPE                               VARCHAR2(106)
DATA_TYPE_MOD                           VARCHAR2(3)
DATA_TYPE_OWNER                         VARCHAR2(30)
DATA_LENGTH                             NOT NULL   NUMBER
DATA_PRECISION                           NUMBER
DATA_SCALE                              NUMBER
NULLABLE                               VARCHAR2(1)
COLUMN_ID                               NUMBER
DEFAULT_LENGTH                           NUMBER
DATA_DEFAULT                             LONG
NUM_DISTINCT                             NUMBER
LOW_VALUE                                RAW(32)
HIGH_VALUE                               RAW(32)
DENSITY                                 NUMBER
NUM_NULLS                                NUMBER
NUM_BUCKETS                              NUMBER
LAST_ANALYZED                           DATE
SAMPLE_SIZE                             NUMBER
CHARACTER_SET_NAME                       VARCHAR2(44)
CHAR_COL_DECL_LENGTH                     NUMBER
GLOBAL_STATS                            VARCHAR2(3)
USER_STATS                              VARCHAR2(3)
AVG_COL_LEN                              NUMBER
CHAR_LENGTH                              NUMBER
CHAR_USED                                VARCHAR2(1)
V80_FMT_IMAGE                           VARCHAR2(3)
DATA_UPGRADED                            VARCHAR2(3)
HISTOGRAM                                VARCHAR2(15)

SQL>
```

```
SQL> describe USER_TAB_COLUMNS ;
```

Name	Null?	Type
TABLE_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
DATA_TYPE		VARCHAR2(106)
DATA_TYPE_MOD		VARCHAR2(3)
DATA_TYPE_OWNER		VARCHAR2(30)
DATA_LENGTH	NOT NULL	NUMBER
DATA_PRECISION		NUMBER
DATA_SCALE		NUMBER
NULLABLE		VARCHAR2(1)
COLUMN_ID		NUMBER
DEFAULT_LENGTH		NUMBER
DATA_DEFAULT		LONG
NUM_DISTINCT		NUMBER
LOW_VALUE		RAW(32)
HIGH_VALUE		RAW(32)
DENSITY		NUMBER
NUM_NULLS		NUMBER
NUM_BUCKETS		NUMBER
LAST_ANALYZED		DATE
SAMPLE_SIZE		NUMBER
CHARACTER_SET_NAME		VARCHAR2(44)
CHAR_COL_DECL_LENGTH		NUMBER
GLOBAL_STATS		VARCHAR2(3)
USER_STATS		VARCHAR2(3)
AVG_COL_LEN		NUMBER
CHAR_LENGTH		NUMBER
CHAR_USED		VARCHAR2(1)
V80_FMT_IMAGE		VARCHAR2(3)
DATA_UPGRADED		VARCHAR2(3)
HISTOGRAM		VARCHAR2(15)

La table **ALL_TAB_COLUMNS** contient : **45639** lignes.

La table **USER_TAB_COLUMNS** contient : **1404** lignes.

```
SQL> select count(*) from ALL_TAB_COLUMNS;

COUNT(*)
-----
45639

SQL> select count(*) from USER_TAB_COLUMNS;

COUNT(*)
-----
1404

SQL>
```

Détail de la question

- i** « **ALL_TAB_COLUMNS** » contient des informations sur toutes les tables de la base de données, tandis que « **USER_TAB_COLUMNS** » contient des informations sur les colonnes des tables d'un utilisateur précis.

	Column	Datatype	NULL	Description
COL1	OWNER	VARCHAR2 (30)	NOT NULL	Owner of the table, view, or cluster
COL2	TABLE_NAME	VARCHAR2 (30)	NOT NULL	Name of the table, view, or cluster
COL3	COLUMN_NAME	VARCHAR2 (30)	NOT NULL	Column name
COL4	DATA_TYPE	VARCHAR2 (106)		Datatype of the column
COL5	DATA_TYPE_MOD	VARCHAR2 (3)		Datatype modifier of the column
COL6	DATA_TYPE_OWNER	VARCHAR2 (30)		Owner of the datatype of the column
COL7	DATA_LENGTH	NUMBER	NOT NULL	Length of the column (in bytes)
COL8	DATA_PRECISION	NUMBER		Decimal precision for <code>NUMBER</code> datatype; binary precision for <code>FLOAT</code> datatype, null for all other datatypes
COL9	DATA_SCALE	NUMBER		Digits to right of decimal point in a number

5. Vérifiez que les tables du TP1 ont été réellement créées ? Donner toutes les informations sur ces tables ?

Requête :

Connect dbaintervention/sarah;
select TABLE_NAME from USER_TABLES;

Résultat :

Ici on sélectionne toutes les tables créent par l'utilisateur **dbaintervention**.

```
SQL> Connect dbintervention/sarah;
Connected.
SQL> select TABLE_NAME from USER_TABLES;

TABLE_NAME
-----
TABLEERREURS
INTERVENTIONS
EMPLOIE
MARQUE
MODELE
CLIENT
VEHICULE
INTERVENANTS

8 rows selected.
```

Affichons maintenant les informations sur ces tables :

Requête:

Connect dbintervention/sarah;

Select * from USER_TABLES

SELECT TABLE_NAME, TABLESPACE_NAME CLUSTER_NAME FROM USER_TABLES ;

Résultat:

```
SQL> connect dbintervention/sarah;
Connected.
SQL> SELECT TABLE_NAME, TABLESPACE_NAME CLUSTER_NAME FROM USER_TABLES ;

TABLE_NAME                                CLUSTER_NAME
-----
TABLEERREURS                             INTERVENTION_TBS
INTERVENTIONS                           INTERVENTION_TBS
EMPLOIE                                  INTERVENTION_TBS
MARQUE                                   INTERVENTION_TBS
MODELE                                  INTERVENTION_TBS
CLIENT                                  INTERVENTION_TBS
VEHICULE                                 INTERVENTION_TBS
INTERVENANTS                             INTERVENTION_TBS

8 rows selected.
```

On ne va pas afficher le résultat de cette requête, vu la taille de ce dernier.

On va préciser des informations à afficher, par exemple les **noms des tables** et le nom de la **TableSpaces** à laquelle elles appartiennent

6. Lister les tables de l'utilisateur « **system** » et celles de l'utilisateur **DBAINTERVENTION** (l'utilisateur de TP1).

Les tables de l'utilisateur « **system** » :

Requête :

Connect system/sarah;

SELECT DISTINCT TABLE_NAME FROM ALL_TABLES WHERE OWNER='SYSTEM';

Résultat :

```
SQL> connect system/sarah;
Connected.
SQL> SELECT DISTINCT TABLE_NAME FROM ALL_TABLES WHERE OWNER='SYSTEM';

TABLE_NAME
-----
DEF$_LOB
LOGMNR_DBNAME_UID_MAP
LOGMNR_AGE_SPILL$
LOGSTDBY$PARAMETERS
LOGSTDBY$APPLY_MILESTONE
LOGSTDBY$SCN
REPCAT$_FLAVORS
REPCAT$_SNAPGROUP
REPCAT$_CONFLICT
REPCAT$_OBJECT_TYPES
REPCAT$_USER_PARM_VALUES

TABLE_NAME
-----
REPCAT$_EXCEPTIONS
REPCAT$_SITES_NEW
OL$NODES
LOGMNR_TYPE$
LOGMNR_LOBFRAG$
LOGMNR_INDCOMPART$
LOGMNR_FILTER$
LOGMNR_SPILL$
OL$HINTS
LOGMNR_GSII
LOGMNR_COL$

TABLE_NAME
-----
LOGMNR_TABCOMPART$
LOGMNR_ATTRIBUTE$
LOGMNR_LOB$
LOGMNR_CDEF$
MVIEW$_ADV_PRETTY
MVIEW$_ADV_TEMP
MVIEW$_ADV_ROLLUP
MVIEW$_ADV_GC
LOGMNR_PROCESSED_LOG$
REPCAT$_KEY_COLUMNS
```

Remarque :

En ce qui concerne les tables de « **system** » on a pris juste une partie du résultat.

Les tables de l'utilisateur « **dbaintervention** » :

Requête :

Connect system/sarah;

SELECT DISTINCT TABLE_NAME FROM ALL_TABLES WHERE OWNER='DBAINTERVENTION';

Résultat :

```
SQL> connect system/sarah;
Connected.
SQL> SELECT DISTINCT TABLE_NAME FROM ALL_TABLES WHERE OWNER='DBAINTERVENTION';

TABLE_NAME
-----
MODELE
INTERVENANTS
TABLEERREURS
VEHICULE
MARQUE
CLIENT
EMPLOYE
INTERVENTIONS

8 rows selected.
```

Ou bien :

Requête :

Connect dbaintervention/sarah;

SELECT TABLE_NAME FROM USER_TABLES ;

Résultat :

```
SQL>
SQL> connect dbaintervention/sarah;
Connected.
SQL> select TABLE_NAME FROM USER_TABLES;

TABLE_NAME
-----
TABLEERREURS
INTERVENTIONS
EMPLOYE
MARQUE
MODELE
CLIENT
VEHICULE
INTERVENANTS

8 rows selected.
```

- Donner la description des attributs des tables **VEHICULE** et **INTERVENTIONS** (Exploiter la table **USER_TAB_COLUMNS**)

Pour consulter ce résultat on exécute ces 2 commandes **sql** :

Requête :

select * from USER_TAB_COLUMNS where TABLE_NAME = 'VEHICULE';

On choisit ces colonnes uniquement :

**select COLUMN_NAME, DATA_TYPE, DATA_LENGTH from USER_TAB_COLUMNS
where TABLE_NAME = 'VEHICULE';**

Résultat :

```
SQL>
SQL> select COLUMN_NAME, DATA_TYPE, DATA_LENGTH from USER_TAB_COLUMNS where TABLE_NAME = 'VEHICULE';

COLUMN_NAME
-----
DATA_TYPE
-----
DATA_LENGTH
-----
NUMVEHICULE
NUMBER
          22

NUMCLIENT
NUMBER
          22

COLUMN_NAME
-----
DATA_TYPE
-----
DATA_LENGTH
-----

NUMMODELE
NUMBER
          22

NUMIMMAT
NUMBER

COLUMN_NAME
-----
DATA_TYPE
-----
DATA_LENGTH
-----
          22

ANNEE
NUMBER
          22
```

Requête :

select * from USER_TAB_COLUMNS where TABLE_NAME = 'INTERVENTIONS';

On choisit ces colonnes uniquement :

**select COLUMN_NAME, DATA_TYPE, DATA_LENGTH from USER_TAB_COLUMNS
where TABLE_NAME = 'INTERVENTIONS';**

Résultat :

```
SQL> select COLUMN_NAME, DATA_TYPE, DATA_LENGTH from USER_TAB_COLUMNS where TABLE_NAME = 'INTERVENTIONS';
```

COLUMN_NAME	DATA_TYPE	DATA_LENGTH
NUMINTERVENTION	NUMBER	22
NUMVEHICULE	NUMBER	22
TYPEINTERVENTION	VARCHAR2	20
DATEDEBINTERV	DATE	
DATEFININTERV	DATE	
COUTINTERV	FLOAT	22

6 rows selected.

8. Comment peut-on vérifier qu'il y a une référence de clé étrangère entre les tables **VEHICULE** et **INTERVENTIONS**?

Il suffit d'afficher les contraintes de la table «**INTERVENTIONS**» et «**VEHICULE**»

Requête :

```
SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE  
TABLE_NAME= 'INTERVENTIONS';
```


SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME= 'VEHICULE';

Résultat :

```
SQL>
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME= 'INTERVENTIONS';

CONSTRAINT_NAME          C
-----
ATTINTERV                 P
FOREIGNINTER              R

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS WHERE TABLE_NAME= 'VEHICULE';

CONSTRAINT_NAME          C
-----
ATTVEHICULE               P
FOREIGNVEH                R
FOREIGNVEH2               R
```

- Donner toutes les contraintes créées lors du **TP1** et les informations qui les caractérisent (Exploitez la table **USER_CONSTRAINTS**)

Requête :

Pour afficher toutes les caractéristiques :

SELECT * FROM USER_CONSTRAINTS ;

On ne va afficher que table_name, constraint_name et constraint_type

SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE FROM USER_CONSTRAINTS;

Résultat :

```
SQL>
SQL> SELECT TABLE_NAME, CONSTRAINT_NAME, CONSTRAINT_TYPE FROM
2 USER_CONSTRAINTS;

TABLE_NAME                CONSTRAINT_NAME          C
-----
INTERVENTIONS            FOREIGNINTER             R
BIN$sm6o+9Z/TfqasnaoUuKFpg==$0 BIN$dseyKU5IShuvC2zHBg6k6g==$0 C
EMPLOYE                   SYS_C004092             C
EMPLOYE                   CKSALAIRE                C
INTERVENANTS              FOREIGNINTERVENANTSCD    R
VEHICULE                  FOREIGNVEH2              R
VEHICULE                  FOREIGNVEH               R
CLIENT                   ATTCIV                   C
MODELE                   FOREIGNMARQ              R
EMPLOYE                   CHECKCATE                C
BIN$1QXCafIoTvyzJ502qSeZHw==$0 BIN$W8HOKTvmR1KF7Au49E4HNA==$0 P

TABLE_NAME                CONSTRAINT_NAME          C
-----
INTERVENTIONS            ATTINTERV                P
BIN$sm6o+9Z/TfqasnaoUuKFpg==$0 BIN$LmBELHvcR7qJdU2k/yo53Q==$0 P
EMPLOYE                   ATTPRIMEMP               P
MARQUE                   ATTMARQ                  P
MODELE                   ATTMODELE                P
CLIENT                   ATTPRIM                  P
VEHICULE                  ATTVEHICULE              P
INTERVENANTS              ATTINTERVENANT           P
BIN$1QXCafIoTvyzJ502qSeZHw==$0 BIN$mhnv6fPKTqOwJeFiIJ1zJg==$0 C

20 rows selected.
```

10. Retrouver toutes les informations permettant de recréer la table **INTERVENTIONS**.

On a besoin de connaître les attributs (Noms, types, tailles) et les contraintes définies sur la table (Noms, types, et les colonnes sur lesquelles elles sont définies)

❖ *Les attributs :*

Requête :

**SELECT COLUMN_NAME, DATA_TYPE, DATA_LENGTH FROM USER_TAB_COLUMNS
WHERE TABLE_NAME=' INTERVENTIONS ';**

Résultat :

```
SQL> SELECT COLUMN_NAME, DATA_TYPE, DATA_LENGTH FROM USER_TAB_COLUMNS WHERE TABLE_NAME='INTERVENTIONS';  
  
COLUMN_NAME  
-----  
DATA_TYPE  
-----  
DATA_LENGTH  
-----  
NUMINTERVENTION  
NUMBER  
          22  
  
NUMVEHICULE  
NUMBER  
          22  
  
COLUMN_NAME  
-----  
DATA_TYPE  
-----  
DATA_LENGTH  
-----  
  
TYPEINTERVENTION  
VARCHAR2  
          20  
  
DATEDEBINTERV  
DATE  
  
COLUMN_NAME  
-----  
DATA_TYPE  
-----  
DATA_LENGTH  
-----  
          7  
  
DATEFININTERV  
DATE  
          7  
  
COUTINTERV
```

```

COUTINTERV
COLUMN_NAME
-----
DATA_TYPE
-----
DATA_LENGTH
-----
FLOAT
      22

6 rows selected.

```

❖ *Les contraintes*

Requête :

```

SELECT  U.CONSTRAINT_NAME, U.CONSTRAINT_TYPE, A.COLUMN_NAME
FROM    USER_CONSTRAINTS  U, ALL_CONS_COLUMNS  A  WHERE
A.TABLE_NAME='INTERVENTIONS' AND
U.CONSTRAINT_NAME=A.CONSTRAINT_NAME;

```

Résultat :

```

SQL> SELECT U.CONSTRAINT_NAME, U.CONSTRAINT_TYPE, A.COLUMN_NAME FROM USER_CONSTRAINTS U, ALL_CONS_COLUMNS A WHERE A.TABLE_NAME='INTERVENTIONS' AND U.CONSTRAINT_NAME=A.CONSTRAINT_NAME;

CONSTRAINT_NAME      C
-----
COLUMN_NAME
-----
ATTINTERV            P
NUMINTERVENTION
FOREIGNINTER          R
NUMVEHICULE

```

11. Trouver tous les privilèges accordés à **Admin**.

- Les privilèges système de l'admin :

Requête :

Si on n'est pas connecté avec « **ADMIN** »

Select privilege from dba_sys_privs where grantee='ADMIN';

Résultat :

```

SQL> connect system/sarah;
Connected.
SQL> Select privilege from dba_sys_privs where grantee='ADMIN';

PRIVILEGE
-----
CREATE SESSION

```

Ou bien :

Requête :

Si on est connecté en tant que « **ADMIN** »

Select * from USER_SYS_PRIVS;

[Résultat :](#)

```
SQL> connect admin/admin;
Connected.
SQL> SELECT * FROM USER_SYS_PRIVS;
```

USERNAME	PRIVILEGE	ADM
ADMIN	CREATE SESSION	NO

- Les privilèges objets de l'admin :

[Requête :](#)

Si on n'est pas connecté avec « **ADMIN** »

Select privilege from dba_tab_privs where grantee=upper ('admin');

[Résultat :](#)

```
SQL> connect system/sarah;
Connected.
SQL> Select privilege from dba_tab_privs where grantee=upper ('admin');

no rows selected
```

Ou bien :

[Requête :](#)

Si on est connecté en tant que « **ADMIN** »

Select * from USER_TAB_PRIVS;

[Résultat :](#)

```
SQL> connect admin/admin;
Connected.
SQL> SELECT * FROM USER_TAB_PRIVS;

no rows selected
```

Détaille de la question

- **i** Les utilisateurs peuvent se voir accorder des privilèges système qui sont visibles dans la section **DBA_SYS_PRIVS**.
- Ils peuvent se voir accorder des privilèges d'objet qui sont visibles dans l'application **DBA_TAB_PRIVS**.
- Ou bien on peut utiliser directement **USER_SYS_PRIVS** et **USER_TAB_PRIVS** pour l'utilisateur connecté.

12. Trouver les rôles donnés à l'utilisateur **Admin**.

Requête :

Si on n'est pas connecté avec « **ADMIN** »

Select * from dba_role_privs where grantee=upper ('admin');

Résultat :

```
SQL> connect system/sarah
Connected.
SQL> select * from dba_role_privs where GRANTEE =upper('admin');
```

GRANTEE	GRANTED_ROLE	ADM	DEF
ADMIN	GESTIONNAIRE_DES_INTERVENTIONS	NO	YES

Ou bien :

Requête :

Si on est connecté en tant que « **ADMIN**»

Select * from USER_TAB_PRIVS;

Résultat :

```
SQL> connect admin/admin;
Connected.
SQL> SELECT * FROM USER_ROLE_PRIVS;
```

USERNAME	GRANTED_ROLE	ADM	DEF	OS_
ADMIN	GESTIONNAIRE_DES_INTERVENTIONS	NO	YES	NO

Détaille de la question

- i** Les utilisateurs peuvent se voir attribuer des rôles qui sont visibles dans la base de données de l'UE : **DBA_ROLE_PRIVS**
- Mais ces rôles peuvent, à leur tour, se voir accorder des privilèges système, des privilèges d'objet et des rôles supplémentaires qui peuvent être consultés en regardant dans **ROLE_SYS_PRIVS**, **ROLE_TAB_PRIVS** et **ROLE_ROLE_PRIVS**

13. Trouver tous les objets appartenant à **Admin**.

Pour afficher toutes les informations sur les objets on exécute la requête :

SELECT * FROM USER_OBJECTS;

Requête :

Si on n'est pas connecté en tant que « **ADMIN**» :

- Select OBJECT_NAME from dba_objects where owner='ADMIN' ;
- Select OBJECT_NAME from all_objects where owner='ADMIN' ;
- Select OBJECT_NAME from all_objects where owner='ADMIN' ;

Résultat :

```
SQL> Select object_name from dba_objects where owner='ADMIN' ;
```

```
OBJECT_NAME
```

```
-----
```

```
TEST
```

```
NOMEMP_IX
```

```
SQL> Select OBJECT_NAME from all_objects where owner='ADMIN' ;
```

```
OBJECT_NAME
```

```
-----
```

```
TEST
```

```
NOMEMP_IX
```

```
SQL>
```

```
SQL> Select OBJECT_NAME from dba_objects where owner='ADMIN' ;
```

```
OBJECT_NAME
```

```
-----
```

```
TEST
```

```
NOMEMP_IX
```

Ou bien :

Requête :

Ici je suis dans **admin** :

Select object_name from user_objects ;

Résultat :

```
SQL> connect admin/admin;
```

```
Connected.
```

```
SQL> select object_name from user_objects;
```

```
OBJECT_NAME
```

```
-----
```

```
TEST
```

```
NOMEMP_IX
```

Objet (ADMIN)	type
TEST	table
NOMEMP_IX	index

14. L'administrateur cherche le propriétaire de la table **INTERVENTIONS**, comment il pourra le trouver ?

Requête :

Select table_name, owner from all_tables where table_name = '**INTERVENTIONS**' ;

Résultat :

```
SQL> Select table_name, owner from all_tables where table_name = 'INTERVENTIONS' ;
```

TABLE_NAME	OWNER
INTERVENTIONS	DBAINTERVENTION

Détaille de la question

i **ALL_TABLES** décrit les tables relationnelles accessibles à l'utilisateur courant.

15. Donner la taille en **Ko** de la table **INTERVENTIONS**.

Requête :

```
SELECT BYTES/1024 AS SIZE_TABLE_KO  
FROM USER_SEGMENTS  
WHERE SEGMENT_NAME='INTERVENTIONS';
```

Résultat :

```
SQL> connect dbaintervention/sarah;  
Connected.  
SQL> SELECT BYTES/1024 AS SIZE_TABLE_KO  
2 FROM USER_SEGMENTS  
3 WHERE SEGMENT_NAME='INTERVENTIONS';
```

SIZE_TABLE_KO
64

16. Vérifier l'effet produit par chacune des commandes de définition de données du TP1 sur le dictionnaire

1. Créons un nouvel utilisateur :

```
SQL>
SQL> CREATE TABLESPACE TP_3_TBS DATAFILE 'C:\tbsTP3_xxx.dat' SIZE 100M AUTOEXTEND ON ONLINE;

Tablespace created.

SQL> CREATE TEMPORARY TABLESPACE TP_3_TempTBS TEMPFILE 'C:\tempTP3_xxx.dat' SIZE 100M AUTOEXTEND ON;

Tablespace created.

SQL> Create User user3 Identified by sarah3 Default Tablespace TP_3_TBS Temporary Tablespace TP_3_TempTBS;

User created.

SQL> GRANT ALL privileges to user3;

Grant succeeded.

SQL> connect user3/sarah3;
Connected.
SQL>
```

2. Vérifions ce qu'il y a dans le dictionnaire avant de créer les tables

```
SQL>
SQL> SELECT TABLE_NAME FROM USER_TABLES;

no rows selected

SQL> SELECT COLUMN_NAME FROM USER_TAB_COLUMNS;

no rows selected

SQL> SELECT CONSTRAINT_NAME FROM USER_CONSTRAINTS;

no rows selected

SQL>
```


3. Créons désormais les tables du TP1 et vérifions encore le contenu du dictionnaire en utilisant les requêtes précédentes :

```
SQL> create table CLIENT (  
  2  NUMCLIENT INTEGER primary key,  
  3  CIV varchar(3),  
  4  PRENOMCLIENT varchar(60),  
  5  NOMCLIENT varchar(60),  
  6  DATEDENAISSANCE date,  
  7  ADRESSE varchar(100),  
  8  TELPROF varchar(60),  
  9  TELPRIV varchar(60),  
 10  FAX VARCHAR(30),  
 11  constraint check_civ check (CIV in ('M.', 'Mle', 'Mme'))  
 12 );
```

Table created.

```
SQL> create table EMPLOYE (  
  2  NUMEMPLOYE INTEGER primary key,  
  3  NOMEMP varchar(3),  
  4  PRENOMEMP varchar(60),  
  5  CATEGORIE varchar(60),  
  6  SALAIRE float,  
  7  constraint check_categorie check (CATEGORIE in ('Mécanicien', 'Assistant'))  
  8 );
```

Table created.

```
SQL> create table MARQUE (  
  2  NUMMARQUE INTEGER primary key,  
  3  MARQUE varchar(3),  
  4  PAYS varchar(60)  
  5 );
```

Table created.

```
SQL> create table MODELE (  
  2  NUMMODELE INTEGER primary key,  
  3  NUMMARQUE INTEGER ,  
  4  MODELE varchar(30),  
  5  constraint fk_nummarque FOREIGN key (NUMMARQUE) REFERENCES MARQUE (NUMMARQUE)  
  6 );
```

Table created.

```

SQL> create table VEHICULE (
  2  NUMVEHICULE INTEGER primary key,
  3  NUMCLIENT INTEGER ,
  4  NUMMODELE INTEGER ,
  5  NUMIMMAT INTEGER,
  6  ANNEE INTEGER,
  7  constraint fk_numclient FOREIGN key (NUMCLIENT) REFERENCES CLIENT (NUMCLIENT),
  8  constraint fk_nummodele FOREIGN key (NUMMODELE) REFERENCES MODELE (NUMMODELE)
  9  );

Table created.

SQL> create table INTERVENTIONS (
  2  NUMINTERVENTION INTEGER primary key,
  3  NUMVEHICULE INTEGER ,
  4  TYPEINTERVENTION varchar(20) ,
  5  DATEDEBUTINTERV date,
  6  DATEFININTERV date,
  7  COUTINTERV float ,
  8  constraint fk_numvehicule FOREIGN key (NUMVEHICULE) REFERENCES VEHICULE (NUMVEHICULE)
  9  );

Table created.

SQL> create table INTERVENANTS (
  2  NUMINTERVENTION INTEGER,
  3  NUMEMPLOYE INTEGER,
  4  DATEDEBUT date,
  5  DATEFIN date,
  6  constraint fk_numinter FOREIGN key (NUMINTERVENTION) REFERENCES INTERVENTIONS (NUMINTERVENTION),
  7  constraint fk_numemploye FOREIGN key (NUMEMPLOYE) REFERENCES EMPLOYE (NUMEMPLOYE),
  8  constraint pk_INTERVENANTS
  9  primary key(NUMINTERVENTION,NUMEMPLOYE)
  10 );

Table created.

```

Vérification du contenu du dictionnaire après création des tables :
 Le dictionnaire contient désormais les informations sur les tables créées, les colonnes, les contraintes...

```

SQL> select table_name from USER_TABLES;

TABLE_NAME
-----
CLIENT
EMPLOYE
MARQUE
MODELE
VEHICULE
INTERVENTIONS
INTERVENANTS
7 rows selected.

SQL> select column_name from USER_TAB_COLUMNS;

COLUMN_NAME
-----
DATEDENAISSANCE
NOMCLIENT
DATEFININTERV
ADRESSE
CIV
NUMINTERVENTION
NUMCLIENT
TYPEINTERVENTION
TELPFOP
DATEFIN
NUMIMMAT

COLUMN_NAME
-----
NUMEMPLOYE
NUMVEHICULE
NUMMODELE
NUMVEHICULE
NUMMARQUE
NUMINTERVENTION
PRENOMEMP
NUMEMPLOYE
MODELE
FAX
PRENOMCLIENT

```

```

COLUMN_NAME
-----
DATEDEBUT
COUTINTERV
NOMEMP
NUMMARQUE
TELPRIV
ANNEE
NUMCLIENT
DATEDEBUTINTERV
SALAIRE
NUMMODELE
MARQUE

COLUMN_NAME
-----
CATEGORIE
PAYS

35 rows selected.

```

5. Vérifions l'effet de la requête « Alter » sur le dictionnaire :

```

SQL> ALTER TABLE EMPLOYE ADD DATAINSTALLATION date;

Table altered.

SQL> ALTER TABLE CLIENT rename column ADRESSE TO ADRESSECLIENT;

Table altered.

```

Vérifions maintenant le contenu du dictionnaire : on remarque dans le résultat que la colonne «DATAINSTALATION » a bien été ajoutée, et la colonne « ADRESSE » a été renommée en «ADRESSECLIENT »

```

SQL> select COLUMN_NAME from USER_TAB_COLUMNS;

COLUMN_NAME
-----
DATEDENAISSANCE
DATAINSTALLATION
NOMCLIENT
DATEFININTERV
ADRESSECLIENT
CIV
NUMINTERVENTION
NUMCLIENT
TYPEINTERVENTION
TELPREF
DATEFIN

```

6. Enfin, vérifions l'effet de l'ajout d'une contrainte sur le dictionnaire :

- Avant l'ajout de la contrainte :

```
SQL> select CONSTRAINT_NAME from USER_CONSTRAINTS;

CONSTRAINT_NAME
-----
FK_NUMEMPLOYE
FK_NUMINTER
FK_NUMVEHICULE
FK_NUMMODELE
FK_NUMCLIENT
FK_NUMMARQUE
CHECK_CATEGORIE
CHECK_CIV
SYS_C004109
SYS_C004111
SYS_C004112

CONSTRAINT_NAME
-----
SYS_C004113
SYS_C004115
SYS_C004118
PK_INTERVENANTS

15 rows selected.
```

- Après l'ajout de la contrainte :

```
SQL>
SQL> ALTER TABLE INTERVENTIONS ADD constraint cf_INTERVENTIONS check(DATEDEBUTINTERV < DATEFININTERV);

Table altered.

SQL> select CONSTRAINT_NAME from USER_CONSTRAINTS;

CONSTRAINT_NAME
-----
cf_INTERVENTIONS
FK_NUMEMPLOYE
FK_NUMINTER
FK_NUMVEHICULE
FK_NUMMODELE
FK_NUMCLIENT
FK_NUMMARQUE
CHECK_CATEGORIE
CHECK_CIV
SYS_C004109
SYS_C004111

CONSTRAINT_NAME
-----
SYS_C004112
SYS_C004113
SYS_C004115
SYS_C004118
PK_INTERVENANTS

16 rows selected.
```

