

Used Database : **Oracle 11g XE**

Used software : **Oracle sql developer** (pour mieux visualizer les tables et les données)

1) Lister le catalogue « DICT ». Il contient combien d'instances ? Donner sa structure :

```
DESC dict ;
SELECT * FROM dict ;
SELECT COUNT(*) FROM dict ;
```

La table DICT contient deux colonnes , **Table_name** varchar (30) et **comment** varchar (4000) qui donne une description sur le rôle de la table

```

Name          Null? Type
-----
TABLE_NAME    VARCHAR2(30)
COMMENTS      VARCHAR2(4000)
```

Le catalogues contient tous les vues de meta base du SGBD ORACLE , **2551 instances**

TABLE_NAME	COMMENTS
1 USER_CONS_COLUMNS	Information about accessible columns in constraint definitions
2 ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
3 DBA_CONS_COLUMNS	Information about accessible columns in constraint definitions
4 USER_LOG_GROUP_COLUMNS	Information about columns in log group definitions
5 ALL_LOG_GROUP_COLUMNS	Information about columns in log group definitions
6 DBA_LOG_GROUP_COLUMNS	Information about columns in log group definitions
7 USER_LOBS	Description of the user's own LOBs contained in the user's own tables
8 ALL_LOBS	Description of LOBs contained in tables accessible to the user
9 DBA_LOBS	Description of LOBs contained in all tables
10 USER_CATALOG	Tables, Views, Synonyms and Sequences owned by the user
11 ALL_CATALOG	All tables, views, synonyms, sequences accessible to the user
12 DBA_CATALOG	All database Tables, Views, Synonyms, Sequences
13 USER_CLUSTERS	Descriptions of user's own clusters
14 ALL_CLUSTERS	Description of clusters accessible to the user
15 DBA_CLUSTERS	Description of all clusters in the database
16 USER_CLU_COLUMNS	Mapping of table columns to cluster columns

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

- ALL_TAB_COLUMNS :

- *Role :*

Cette vue contient **tous** les colonnes que l'utilisateur peut **accéder** (celles qu'il a créer + celles qu'il peut voir a travers des roles ou grant)

- *Structure :*

En exécutant la requête **DESC dict ;**

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(120)
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 BYTE)
HIGH_VALUE		RAW(32 BYTE)
DENSITY		NUMBER
NUM_NULLS		NUMBER
NUM_BUCKETS		NUMBER
LAST_ANALYZED		DATE

Tel que **Table_name** est le nom de la table de la colonne , **OWNER** est le propriétaire de la table , et d'autres attributs qui décrivent la colonne

Par exemple en exécutant la requête **Select * from all_tab_columns ;** de la part de **gereachat** qu'on a créé dans le TP 2 et qu'on lui a donné le privilège de **lecture sur la tables produit**

OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH
DBACOMPTOIRS_SAMI	PRODUIT	NOMPROD	VARCHAR2	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	NOFOUR	NUMBER	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	CODECATEG	NUMBER	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	QTEPARUNIT	VARCHAR2	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	PRIXUNIT	NUMBER	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	UNITESSTOCK	NUMBER	(null)	(null)	
DBACOMPTOIRS_SAMI	PRODUIT	UNITESCOM	NUMBER	(null)	(null)	

- USER_USERS

- Rôle :

Cette vue permet de donner les **caractéristique** de l'**utilisateur en courant** , son nom , ID , date de création et d'expiration et aussi les tables spaces qu'ils utilise .

- Structure :

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)

Par exemple en exécutant la requête sur l'utilisateur crée au TP 1 , **select * from dbacomptoirs_sami ;**

Script Output x Query Result x

 All Rows Fetched: 1 in 0.003 seconds

USERNAME	USER_ID	ACCOUNT_STATUS	LOCK_DATE	EXPIRY_DATE	DEFAULT_TABLESPACE	TEMPORARY_TABLESPACE	CREATED
1 DBACOMPTOIRS_SAMI	48	OPEN	(null)	04-MAY-24	COMPTOIRS_TBS	COMPTOIRS_TEMPTBS	06-NOV-23 D

- ALL_CONSTRAINTS

- Rôle :

Cette vue contient tous les contraintes sur tous les tables que l'utilisateur peut **accéder** .

- Structure :

Name	Null?	Type
-----	-----	-----
OWNER		VARCHAR2 (120)
CONSTRAINT_NAME	NOT NULL	VARCHAR2 (30)
CONSTRAINT_TYPE		VARCHAR2 (1)
TABLE_NAME	NOT NULL	VARCHAR2 (30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2 (120)
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)

Les informations qu'elle contient sont , **le nom de la contrainte** , son **type** , la **table** dont elle appartient et le **owner** de la contrainte.

Par exemple , lorsque on exécute la requête **select * from all_constraints** de la part de **gereachat** on peut lister les contraintes de la tables produits et commandes dont l'utilisateur a le droit de lecture .

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	R_OWNER	R_CONSTRAINT_NAME
1 DBACOMPTOIRS_SAMI	FK_COMMANDE_CLIENT	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_CLIENT
2 DBACOMPTOIRS_SAMI	FK_COMMANDE_MESSAGER	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_MESSAGER
3 DBACOMPTOIRS_SAMI	FK_COMMANDE_EMPLOY	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_EMPLOYE
4 DBACOMPTOIRS_SAMI	SYS_C007337	C	COMMANDE	"DESTINATAIRE" IS NOT NULL	(null)	(null)
5 DBACOMPTOIRS_SAMI	SYS_C007338	C	COMMANDE	"ADRLIV" IS NOT NULL	(null)	(null)
6 DBACOMPTOIRS_SAMI	FK_PRODUIT_FOURNISSEUR	R	PRODUIT	(null)	DBACOMPTOIRS_SAMI	PK_FOURNISSEUR

• USER_TAB_PRIVS

➤ Rôle :

Cette vue donne la liste des **privileges** dont l'utilisateur est le **garantor** , **garantee** , ou le **owner** de l'objet .

➤ Structure :

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)

Tel que , le **Garantee** est l'utilisateur ou le rôle qui reçoit le privilège , **Garantor** est celui qui le donne , **Owner** est le propriétaire de table , et on spécifie le **type** de privilège (update , select , alter , index ...)

Par exemple en exécutant la requête **select * from user_tab_privs** de la part de user **dbacomptoirs_sami** On pourra voir les privilèges qu'on a donné dans le TP 2

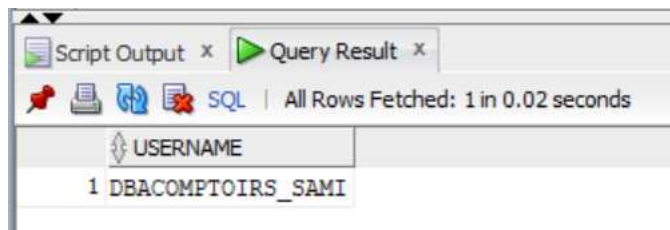
GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
1 GESTIONAPPRO	DBACOMPTOIRS_SAMI	COMMANDE	DBACOMPTOIRS_SAMI	UPDATE	NO	NO
2 GESTIONAPPRO	DBACOMPTOIRS_SAMI	FOURNISSEUR	DBACOMPTOIRS_SAMI	SELECT	NO	NO
3 GERERACHAT	DBACOMPTOIRS_SAMI	PRODUIT	DBACOMPTOIRS_SAMI	INDEX	NO	NO
4 GESTIONAPPRO	DBACOMPTOIRS_SAMI	PRODUIT	DBACOMPTOIRS_SAMI	SELECT	NO	NO
5 GERERACHAT	DBACOMPTOIRS_SAMI	PRODUIT	DBACOMPTOIRS_SAMI	UPDATE	NO	NO
6 GERERACHAT	DBACOMPTOIRS_SAMI	PRODUIT	DBACOMPTOIRS_SAMI	SELECT	NO	NO

3) Trouver le nom d'utilisateur avec lequel vous êtes connecté :

On utilise la table **USER_USERS** et on affiche la colonne **USERNAME**

```
SELECT username FROM user_users ;
```

Résultat :



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying the results of the SQL query. The window title is 'SQL | All Rows Fetched: 1 in 0.02 seconds'. The results are shown in a table with one column, 'USERNAME', and one row, '1 DBACOMPTOIRS_SAMI'.

	USERNAME
1	DBACOMPTOIRS_SAMI

4) Comparer la structure et le contenu des tables ALL_TAB_COLUMNS et USER_TAB_COLUMNS :

- Structure :

```
DESC 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(120)
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 BYTE)
HIGH_VALUE		RAW(32 BYTE)
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)

```
DESC 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(120)
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 BYTE)
HIGH_VALUE		RAW(32 BYTE)
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)

On voit bien que tous les attributs des deux tables sont identiques , sauf le premier attribut **OWNER** que la vue **ALL_TAB_COLUMNS** possède.

Car la vue **ALL_TAB_PRIVS** contient les colonnes que l'utilisateur a accès mais pas forcément créer donc on garde trace du **propriétaire** de la table concerné , par contre la vue **USER_TAB_COLUMNS** contient les colonnes des tables dont le **propriétaire** est l'utilisateur qui exécute la commande.

- Contenu :

On utilise le user dbacomptoirs_sami du TP 1

```
SELECT * FROM user_tab_columns ;
```

On liste les colonnes des tables qu'il a cree

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NULLABLE
1 CATEGORIE	DESCRIPTION	VARCHAR2	(null)	(null)	100	(null)	(null)	Y
2 CATEGORIE	NOMCATEG	VARCHAR2	(null)	(null)	40	(null)	(null)	Y
3 CATEGORIE	CODECATEG	NUMBER	(null)	(null)	22	(null)	0	N
4 CLIENT	FAX	VARCHAR2	(null)	(null)	20	(null)	(null)	Y
5 CLIENT	TEL	VARCHAR2	(null)	(null)	25	(null)	(null)	Y
6 CLIENT	PAYS	VARCHAR2	(null)	(null)	20	(null)	(null)	Y
7 CLIENT	CODEPOSTAL	VARCHAR2	(null)	(null)	20	(null)	(null)	Y
8 CLIENT	REGION	VARCHAR2	(null)	(null)	40	(null)	(null)	Y
9 CLIENT	VILLE	VARCHAR2	(null)	(null)	40	(null)	(null)	Y
10 CLIENT	ADRESSE	VARCHAR2	(null)	(null)	40	(null)	(null)	Y
11 CLIENT	FONCTION	VARCHAR2	(null)	(null)	40	(null)	(null)	Y
12 CLIENT	CONTACT	VARCHAR2	(null)	(null)	30	(null)	(null)	Y
13 CLIENT	SOCIETE	VARCHAR2	(null)	(null)	30	(null)	(null)	Y
14 CLIENT	CODECLI	VARCHAR2	(null)	(null)	20	(null)	(null)	N
15 COMMANDE	DATECOM	DATE	(null)	(null)	7	(null)	(null)	Y
16 COMMANDE	NOEMP	NUMBER	(null)	(null)	22	(null)	0	Y
17 COMMANDE	ALIVAVANT	DATE	(null)	(null)	7	(null)	(null)	Y
18 COMMANDE	NOMESS	NUMBER	(null)	(null)	22	(null)	0	Y

```
SELECT * FROM all_tab_columns ;
```

On liste les colonnes dont il a accès

OWNER	TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH	DATA_PRECISION	DATA_SCALE
1 APPQOSSYS	WLM_CLASSIFIER_PLAN	CHKSUM	NUMBER	(null)	(null)	22	(null)	(null)
2 APPQOSSYS	WLM_CLASSIFIER_PLAN	TIMESTAMP	DATE	(null)	(null)	7	(null)	(null)
3 APPQOSSYS	WLM_CLASSIFIER_PLAN	SEQNO	NUMBER	(null)	(null)	22	(null)	(null)
4 APPQOSSYS	WLM_CLASSIFIER_PLAN	ACTIVE	CHAR	(null)	(null)	1	(null)	(null)
5 APPQOSSYS	WLM_CLASSIFIER_PLAN	CLPCSTR	VARCHAR2	(null)	(null)	4000	(null)	(null)
6 APPQOSSYS	WLM_CLASSIFIER_PLAN	NCLSR	NUMBER	(null)	(null)	22	(null)	(null)
7 APPQOSSYS	WLM_CLASSIFIER_PLAN	OPER	NUMBER	(null)	(null)	22	(null)	(null)
8 APPQOSSYS	WLM_METRICS_STREAM	NEGATIVE_INTERVAL	NUMBER	(null)	(null)	22	(null)	(null)
9 APPQOSSYS	WLM_METRICS_STREAM	PC	VARCHAR2	(null)	(null)	31	(null)	(null)
10 APPQOSSYS	WLM_METRICS_STREAM	TIMESTAMP	DATE	(null)	(null)	7	(null)	(null)
11 APPQOSSYS	WLM_MPA_STREAM	RISKLEVEL	NUMBER	(null)	(null)	22	(null)	(null)
12 APPQOSSYS	WLM_MPA_STREAM	SERVERORPOOL	VARCHAR2	(null)	(null)	8	(null)	(null)
13 APPQOSSYS	WLM_MPA_STREAM	NAME	VARCHAR2	(null)	(null)	4000	(null)	(null)
14 APPQOSSYS	WLM_VIOLATION_STREAM	VIOLATION	VARCHAR2	(null)	(null)	4000	(null)	(null)
15 APPQOSSYS	WLM_VIOLATION_STREAM	SERVERPOOL	VARCHAR2	(null)	(null)	4000	(null)	(null)
16 APPQOSSYS	WLM_VIOLATION_STREAM	TIMESTAMP	DATE	(null)	(null)	7	(null)	(null)
17 DBSNMP	BSLN_BASELINES	LAST_COMPUTE_DATE	DATE	(null)	(null)	7	(null)	(null)
18 DBSNMP	BSLN_BASELINES	STATUS	VARCHAR2	(null)	(null)	16	(null)	(null)

5) Vérifiez que les tables du TP1 ont été réellement créées et donner toutes les informations sur ces tables :

Il suffit d'exécuter un select de la table **USER_TABLES** car cette dernière contient tous les tables que l'utilisateur a créée

```
SELECT * FROM user_tables ;
```

Query Result x

All Rows Fetched: 9 in 0.15 seconds

	TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS	MAX_TRANS	INITIAL_EXTENT	NE
1	TABLEERREURS	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
2	FOURNISSEUR	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
3	CLIENT	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
4	CATEGORIE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
5	PRODUIT	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
6	MESSAGER	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
7	EMPLOYE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
8	COMMANDE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
9	DETAILCOMMANDE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	

Parmi les informations qu'on peut tirer :

En faisant une jointure entre **USER_TABLES** , **USER_TAB_COLUMNS**, **USER_INDEXES**

- la **cardinalité** (Num_rows) , nom de la table space (tablespace_name) , Nom de l'utilisateur (Table_owner)
le **degré** et d'autres informations

```
SELECT T.table_name , T.tablespace_name , I.table_owner , I.index_name , T.num_rows
AS "cardinalite" , COUNT( C.column_name) AS "degre"
FROM user_tables T, user_tab_columns C , user_indexes I
WHERE T.table_name = C.table_name AND T.table_name=I.table_name
GROUP BY T.table_name, T.tablespace_name , I.table_owner , I.index_name
, T.num_rows ;
```

All Rows Fetched: 8 in 0.355 seconds

	TABLE_NAME	TABLESPACE_NAME	TABLE_OWNER	INDEX_NAME	Cardinalite	Degre
1	DETAILCOMMANDE	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_DC	569	5
2	EMPLOYE	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_EMPLOYE	10	15
3	CLIENT	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_CLIENT	90	11
4	CATEGORIE	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_CATEGORIE	8	3
5	PRODUIT	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_PRODUIT	71	10
6	FOURNISSEUR	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_FOURNISSEUR	31	12
7	MESSAGER	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_MESSAGER	3	3
8	COMMANDE	COMPTOIRS_TBS	DBACOMPTOIRS_SAMI	PK_COMMANDE	242	14

6) Lister les tables de l'utilisateur « system » et celles de l'utilisateur DBACOMPTOIRS_VotreNom :

- Dbacomptoirs_sami :

```
SELECT * FROM user_tables ;
```

Query Result x

All Rows Fetched: 9 in 0.15 seconds

TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS	MAX_TRANS	INITIAL_EXTENT	NE
1 TABLEERREURS	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
2 FOURNISSEUR	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
3 CLIENT	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
4 CATEGORIE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
5 PRODUIT	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
6 MESSENGER	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
7 EMPLOYE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
8 COMMANDE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	
9 DETAILCOMMANDE	COMPTOIRS_TBS	(null)	(null)	VALID	10	(null)	1	255	65536	

- System :

```
SELECT * FROM user_tables ;
```

TABLE_NAME	TABLESPACE_NAME	CLUSTER_NAME	IOT_NAME	STATUS	PCT_FREE	PCT_USED	INI_TRANS	MAX_TRANS	INITIAL_EXTENT	NE
1 LOGMNR_SESSION_EVOLVE\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
2 LOGMNR_GLOBAL\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
3 LOGMNR_GT_TAB_INCLUDE\$	(null)	(null)	(null)	VALID	10	40	1	255		
4 LOGMNR_GT_USER_INCLUDE\$	(null)	(null)	(null)	VALID	10	40	1	255		
5 LOGMNR_GT_XID_INCLUDE\$	(null)	(null)	(null)	VALID	10	40	1	255		
6 LOGMNR_UID\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
7 LOGMNR_DBNAME_UID_MAP	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
8 LOGMNR_LOG\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
9 LOGMNR_PROCESSED_LOG\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
10 LOGMNR_SPILL\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
11 LOGMNR_AGE_SPILL\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
12 LOGMNR_RESTART_CKPT_TXINFO\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
13 LOGMNR_ERROR\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
14 LOGMNR_RESTART_CKPT\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
15 LOGMNR_INTEGRATED_SPILL\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		
16 LOGMNR_FILTER\$	SYSAUX	(null)	(null)	VALID	10	(null)	1	255		

7) Donner la description des attributs des tables CLIENT et PRODUIT (Exploiter la table USER_TAB_COLUMNS) :

Les informations qu'on trouve concernant les colonnes des tables sont : nom de colonne , nom de la table , type de donnée , taille de donnée et ID de la colonne

```
SELECT * FROM user_tab_columns WHERE table_name='PRODUIT';
SELECT * FROM user_tab_columns WHERE table_name='CLIENT';
```

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NULLABLE	COLUMN_ID
1 PRODUIT	REFPROD	NUMBER	(null)	(null)	22	(null)	0 N		1
2 PRODUIT	NOMPROD	VARCHAR2	(null)	(null)	40	(null)	(null) Y		2
3 PRODUIT	NOFOUR	NUMBER	(null)	(null)	22	(null)	0 Y		3
4 PRODUIT	CODECATEG	NUMBER	(null)	(null)	22	(null)	0 Y		4
5 PRODUIT	QTEPARUNIT	VARCHAR2	(null)	(null)	30	(null)	(null) Y		5
6 PRODUIT	PRIXUNIT	NUMBER	(null)	(null)	22	(null)	(null) Y		6
7 PRODUIT	UNITESSTOCK	NUMBER	(null)	(null)	22	(null)	0 Y		7
8 PRODUIT	UNITESCOM	NUMBER	(null)	(null)	22	(null)	0 Y		8
9 PRODUIT	NIVEAUREAP	NUMBER	(null)	(null)	22	(null)	0 Y		9
10 PRODUIT	INDISPONIBLE	NUMBER	(null)	(null)	22	(null)	0 Y		10

TABLE_NAME	COLUMN_NAME	DATA_TYPE	DATA_TYPE_MOD	DATA_TYPE_OWNER	DATA_LENGTH	DATA_PRECISION	DATA_SCALE	NULLABLE	COLUMN_ID
1 CLIENT	CODECLI	VARCHAR2	(null)	(null)	20	(null)	(null) N		1
2 CLIENT	SOCIETE	VARCHAR2	(null)	(null)	30	(null)	(null) Y		2
3 CLIENT	CONTACT	VARCHAR2	(null)	(null)	30	(null)	(null) Y		3
4 CLIENT	FONCTION	VARCHAR2	(null)	(null)	40	(null)	(null) Y		4
5 CLIENT	ADRESSE	VARCHAR2	(null)	(null)	40	(null)	(null) Y		5
6 CLIENT	VILLE	VARCHAR2	(null)	(null)	40	(null)	(null) Y		6
7 CLIENT	REGION	VARCHAR2	(null)	(null)	40	(null)	(null) Y		7
8 CLIENT	CODEPOSTAL	VARCHAR2	(null)	(null)	20	(null)	(null) Y		8
9 CLIENT	PAYS	VARCHAR2	(null)	(null)	20	(null)	(null) Y		9
10 CLIENT	TEL	VARCHAR2	(null)	(null)	25	(null)	(null) Y		10
11 CLIENT	FAX	VARCHAR2	(null)	(null)	20	(null)	(null) Y		11

8) Comment peut-on vérifier qu'il y a une référence de clé étrangère entre les tables CLIENT et COMMANDE :

Pour cela on exploite la table USER_CONSTRAINTS

les clé étrangere sont sauvegardé dans cette table avec CONSTRAINT_TYPE = R

```
SELECT * FROM user_constraints WHERE constraint_type='R';
```

ou bien on teste si l'attribut R_constraint_name correspond bien a la clé primaire de la relation CLIENT

```
SELECT *
FROM user_constraints
WHERE table_name = 'COMMANDE' AND r_constraint_name IN ( SELECT index_name FROM
user_indexes WHERE table_name = 'CLIENT' ) ;
```

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	R_OWNER	R_CONSTRAINT_NAME	
1	DBACOMPTOIRS_SAMI	FK_PRODUIT_FOURNISSEUR	R	PRODUIT	(null)	DBACOMPTOIRS_SAMI	PK_FOURNISSEUR	1
2	DBACOMPTOIRS_SAMI	FK_COMMANDE_CLIENT	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_CLIENT	1
3	DBACOMPTOIRS_SAMI	FK_PRODUIT_CATEGORIE	R	PRODUIT	(null)	DBACOMPTOIRS_SAMI	PK_CATEGORIE	1
4	DBACOMPTOIRS_SAMI	FK_DETAILCOMMANDE_PRODUIT	R	DETAILCOMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_PRODUIT	1
5	DBACOMPTOIRS_SAMI	FK_COMMANDE_MESSAGER	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_MESSAGER	1
6	DBACOMPTOIRS_SAMI	FK_EMPLOY_EMPLOY	R	EMPLOYE	(null)	DBACOMPTOIRS_SAMI	PK_EMPLOYE	1
7	DBACOMPTOIRS_SAMI	FK_COMMANDE_EMPLOY	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_EMPLOYE	1
8	DBACOMPTOIRS_SAMI	FK_DETAILCOMMANDE_COMMANDE	R	DETAILCOMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_COMMANDE	1

Tel que , **TABLE_NAME** est la table de la **clé étrangère** et **R_CONSTRAINT_NAME** est le nom de la clé primaire référencié

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

On peut lister la liste de tous les contraintes en utilisant la table **USER_CONSTRAINTS**

Cette table contient des informations sur tous les contraintes tel que : propriétaire (**OWNER**) , **nom** , **type** de contrainte(**R** , **P** , **C**) pour respectivement (**FK** , **PK** , **check**) , status (enabled / disabled) , date du dernier changement

Elle contient aussi des informations qui change selon le type de la contrainte :

- Check : l'attribut **search_condition** décrit la condition du check
- Foreign Key : l'attribut **R_CONSTRAINT_NAME** donne le nom de la clé primaire référencié et **R_OWNER** donne le propriétaire de la table de cette clé primaire.

```
DESC user_constraints ;
```

Name	Null?	Type
OWNER		VARCHAR2 (120)
CONSTRAINT_NAME	NOT NULL	VARCHAR2 (30)
CONSTRAINT_TYPE		VARCHAR2 (1)
TABLE_NAME	NOT NULL	VARCHAR2 (30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2 (120)
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)

On exécute cette requête par le user dbacomptoirs_sami

```
SELECT * FROM user_constraints ;
```

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	R_OWNER	R_CONSTRAINT_NAME
47	DBACOMPTOIRS_SAMI	BIN\$rH2A/K8sRyC2jKp62u/...	P	BIN\$mJoodJhmT2...	(null)	(null)	(null)
48	DBACOMPTOIRS_SAMI	PK_MESSAGER	P	MESSAGER	(null)	(null)	(null)
49	DBACOMPTOIRS_SAMI	CK_TITRECOURTOISIE	C	EMPLOYE	"TITRECOURTOISIE"='Mlle' OR "TITRE...	(null)	(null)
50	DBACOMPTOIRS_SAMI	CK_DATECOMPARE	C	EMPLOYE	datenaissance < dateembauche	(null)	(null)
51	DBACOMPTOIRS_SAMI	PK_EMPLOYE	P	EMPLOYE	(null)	(null)	(null)
52	DBACOMPTOIRS_SAMI	FK_EMPLOY_EMPLOY	R	EMPLOYE	(null)	DBACOMPTOIRS_SAMI	PK_EMPLOYE
53	DBACOMPTOIRS_SAMI	PK_COMMANDE	P	COMMANDE	(null)	(null)	(null)
54	DBACOMPTOIRS_SAMI	FK_COMMANDE_CLIENT	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_CLIENT
55	DBACOMPTOIRS_SAMI	FK_COMMANDE_EMPLOY	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_EMPLOYE
56	DBACOMPTOIRS_SAMI	FK_COMMANDE_MESSAGER	R	COMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_MESSAGER
57	DBACOMPTOIRS_SAMI	CK_REMISE	C	DETAILCOMMANDE	remise >= 0 AND remise <=0.4	(null)	(null)
58	DBACOMPTOIRS_SAMI	PK_DC	P	DETAILCOMMANDE	(null)	(null)	(null)
59	DBACOMPTOIRS_SAMI	FK_DETAILCOMMANDE_COMMANDE	R	DETAILCOMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_COMMANDE
60	DBACOMPTOIRS_SAMI	FK_DETAILCOMMANDE_PRODUIT	R	DETAILCOMMANDE	(null)	DBACOMPTOIRS_SAMI	PK_PRODUIT
61	DBACOMPTOIRS_SAMI	SYS_C007337	C	COMMANDE	"DESTINATAIRE" IS NOT NULL	(null)	(null)
62	DBACOMPTOIRS_SAMI	SYS_C007338	C	COMMANDE	"ADRLIV" IS NOT NULL	(null)	(null)
63	DBACOMPTOIRS_SAMI	CK_OTF	C	DETAILCOMMANDE	ote >0	(null)	(null)

10) Retrouver toutes les informations permettant de recréer la table FOURNISSEUR :

Les informations permettant de recréer une table sont :

- La structure (les attributs) :

on recupere le nom , numero de la colonne , et la taille de chaque attribut à partir de la table **user_tab_columns**

- L'index primaire :

une jointure entre **user_constraints** et **user_indexes**

- Les contraintes :

On utilise la table **user_constraints** , afin de trouver tous les contraintes (primary key , foreign key et check) et les attributs concerné par ses contraintes

Dans notre cas la table ne contient que la contrainte de la clé primaire qui correspond aussi a l'index

- Les triggers :

user_triggers

Dans notre cas , la table fournisseur n'a aucun trigger associé

- Les privileges :

user_tab_privs , afin de recuperer les roles ou users qui ont des privileges sur cette table

- Les données :

on recupere les tuples par un select de la table fournisseur original par un select

```
SELECT column_name , column_id , data_type , data_length, nullable
FROM user_tab_columns
WHERE table_name = 'FOURNISSEUR';

SELECT index_name FROM user_indexes WHERE table_name = 'FOURNISSEUR' ;

SELECT * FROM user_constraints WHERE table_name = 'FOURNISSEUR' ;

SELECT * FROM user_triggers WHERE table_name = 'FOURNISSEUR' ;
```

Dans notre cas ,

La table fournisseur contient **12 attributs** , aucune contrainte sauf **PK** , aucun trigger , et le role **GESTIONAPPRO** a le droit de **lecture** sur cette table

COLUMN_NAME	COLUMN_ID	DATA_TYPE	DATA_LENGTH	NULLABLE
NOFOUR	1	NUMBER	22	N
SOCIETE	2	VARCHAR2	40	Y
CONTACT	3	VARCHAR2	40	Y
FONCTION	4	VARCHAR2	40	Y
ADRESSE	5	VARCHAR2	30	Y
VILLE	6	VARCHAR2	30	Y
REGION	7	VARCHAR2	30	Y
CODEPOSTAL	8	VARCHAR2	20	Y
PAYS	9	VARCHAR2	30	Y
TEL	10	VARCHAR2	15	Y
FAX	11	VARCHAR2	20	Y
PAGEACCUEIL	12	VARCHAR2	30	Y

SQL | All Rows Fetched: 1 in 0.047 seconds

OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME	SEARCH_CONDITION	R_OWNER	R_CONSTRAINT_NAME	DELETE_RULE	STATUS
DBACOMPTOIRS_SAMI	PK_FOURNISSEUR	P	FOURNISSEUR	(null)	(null)	(null)	(null)	ENABLED

GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
GESTIONAPPRO	DBACOMPTOIRS_SAMI	FOURNISSEUR	DBACOMPTOIRS_SAMI	SELECT	NO	NO

11) Trouver tous les privilèges accordés à GestionAppro :

On utilise la table all_tab_privs , et on prend les tuples ayant le grantee = gestionappro

```
SELECT * FROM all_tab_privs WHERE grantee = 'GESTIONAPPRO';
```

	GRANTOR	GRANTEE	TABLE_SCHEMA	TABLE_NAME	PRIVILEGE	GRANTABLE	HIERARCHY
1	DBACOMPTOIRS_SAMI	GESTIONAPPRO	DBACOMPTOIRS_SAMI	COMMANDE	UPDATE	NO	NO
2	DBACOMPTOIRS_SAMI	GESTIONAPPRO	DBACOMPTOIRS_SAMI	FOURNISSEUR	SELECT	NO	NO
3	DBACOMPTOIRS_SAMI	GESTIONAPPRO	DBACOMPTOIRS_SAMI	PRODUIT	SELECT	NO	NO

12) Trouver les rôles donnés à l'utilisateur GererAchat :

On utilise la table **DBA_ROLE_PRIVS** à partir de user system ou bien **USER_ROLE_PRIVS** à partir de gererachat

```
SELECT * FROM user_role_privs ;  
  
SELECT * FROM dba_role_privs WHERE grantee = 'GERERACHAT' ;
```

	USERNAME	GRANTED_ROLE	ADMIN_OPTION	DEFAULT_ROLE	OS_GRANTED
1	GERERACHAT	GESTIONAPPRO	NO	YES	NO

13) Trouver tous les objets appartenant à GererAchat :

On utilise la table **USER_OBJECTS** ou **DBA_OBJECTS** qui donne tous les objets créés (Table , Index , Triggers)

```
SELECT * FROM user_objects ;  
  
SELECT * FROM dba_objects WHERE owner = 'GERERACHAT' ;
```

Dans notre cas l'utilisateur gererachat n'a créé aucun objet donc le résultat est vide

par contre si on utilise le user **dbacomptoirs**

7	PK_CATEGORIE	(null)	21821	21821 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:20:20	VALID
8	PRODUIT	(null)	21822	21822 TABLE	13-NOV-23	25-NOV-23	2023-11-13:11:01:43	VALID
9	PK_PRODUIT	(null)	21823	21823 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:20:41	VALID
10	MESSAGER	(null)	21824	21824 TABLE	13-NOV-23	13-NOV-23	2023-11-13:11:04:51	VALID
11	PK_MESSAGER	(null)	21825	21825 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:20:46	VALID
12	EMPLOYE	(null)	21826	21826 TABLE	13-NOV-23	13-NOV-23	2023-11-13:11:13:13	VALID
13	PK_EMPLOYE	(null)	21827	21827 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:20:54	VALID
14	COMMANDE	(null)	21828	21828 TABLE	13-NOV-23	25-NOV-23	2023-11-13:11:26:23	VALID
15	PK_COMMANDE	(null)	21829	21829 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:20:59	VALID
16	DETAILCOMMANDE	(null)	21830	21830 TABLE	13-NOV-23	13-NOV-23	2023-11-13:09:21:04	VALID
17	PK_DC	(null)	21831	21831 INDEX	13-NOV-23	13-NOV-23	2023-11-13:09:21:04	VALID
18	QTE_DETAILCOMMANDE	(null)	21832	(null) TRIGGER	13-NOV-23	13-NOV-23	2023-11-13:13:57:10	INVALID

14) L'administrateur cherche le propriétaire de la table EMPLOYE, comment pourra-t-il le trouver :

On utilise la table **DBA_TABLES** et on récupère l'attribut **owner**

```
SELECT OWNER FROM dba_tables  
WHERE table_name = 'EMPLOYE';
```

	OWNER
1	DBACOMPTOIRS_SAMI

15) Donner la taille en Ko de la table EMPLOYE :

On utilise la table **USER_SEGMENTS** ou **DBA_SEGMENTS**

```
SELECT segment_name ,BYTES/1024 KO  
FROM USER_SEGMENTS  
WHERE SEGMENT_NAME='FOURNISSEUR' ;
```

	SEGMENT_NAME	KO
1	FOURNISSEUR	64