

Université de Sciences et Technologies Houari Boumedien

Faculty d'Informatique
Master 1 RSD

TP ASGBD

Belmouloud Mustapha Abdellah 212131092524

Amine Chouial 222231579814

Merazi Wail 212133045973

Mohamed Amine Khoucha 212131052028

December 26, 2025

1 TP1

1. create tablespace belmouloud_tbs datafile '/opt/oracle/oradata/XE/belmouloud_tbs.dbf' SIZE 100M AUTOEXTEND ON;

```
SQL> create tablespace belmouloud_tbs datafile '/opt/oracle/oradata/XE/belmouloud_tbs.dbf' SIZE 100M AUTOEXTEND ON;
Tablespace created.
```

create temporary tablespace belmouloud2_temptbs tempfile '/opt/oracle/oradata/XE/belmouloud2_temptbs.dbf' SIZE 100M AUTOEXTEND ON;

```
SQL> create temporary tablespace belmouloud2_temptbs tempfile '/opt/oracle/oradata/XE/belmouloud2_temptbs.dbf' SIZE 100M AUTOEXTEND ON;
Tablespace created.
```

2. create user tp1_belmouloud identified by musmus3108 default tablespace belmouloud_tbs temporary tablespace belmouloud2_temptbs;

```
SQL> create user tp1_belmouloud identified by musmus3108 default tablespace belmouloud_tbs temporary tablespace belmouloud2_temptbs;
User created.
```

3. GRANT ALL privileges to tp1_belmouloud;

```
SQL> GRANT ALL privileges to tp1_belmouloud;
Grant succeeded.
```

4. les tableaux proposés:

user(id primary key, name)

sellers(id primary key, name)

products(id primary key, name, seller_id foreign key, price(entre 20 et 100), category (tshirt, hoodie, jacket))

orders(product_id foreign key, user_id foreign key, primarykey(product_id, use_id))

order_comments (id primary key, product_id foreign key, user_id foreign key,comments)

seller_reviews (id primary_key, seller_id foeignkey,user_id foreignkey,rating)

5. CREATE TABLE users (id NUMBER, name VARCHAR2(50), CONSTRAINT pk_users PRIMARY KEY (id));

```
CREATE TABLE sellers ( id NUMBER, name VARCHAR2(50), CONSTRAINT
pk_sellers PRIMARY KEY (id) );
```

```
CREATE TABLE products ( id NUMBER, name VARCHAR2(20), seller_id NUM-
BER, price NUMBER, category VARCHAR2(10), CONSTRAINT pk_products PRI-
MARY KEY (id), CONSTRAINT chk_price CHECK (price BETWEEN 20 AND
100), CONSTRAINT chk_category CHECK (category IN ('tshirt', 'hoodie', 'jacket')),
CONSTRAINT fk_product_seller FOREIGN KEY (seller_id) REFERENCES sellers
(id) );
```

```
CREATE TABLE orders ( product_id NUMBER, user_id NUMBER, CONSTRAINT
pk_orders PRIMARY KEY (product_id, user_id), CONSTRAINT fk_order_product
FOREIGN KEY (product_id) REFERENCES products (id), CONSTRAINT fk_order_user
FOREIGN KEY (user_id) REFERENCES users (id) );
```

```
CREATE TABLE order_comments ( id NUMBER, product_id NUMBER, user_id
NUMBER, commenttxt VARCHAR2(100), CONSTRAINT pk_order_comments PRI-
MARY KEY (id), CONSTRAINT fk_comment_order1 FOREIGN KEY (product_id)
REFERENCES products(id), CONSTRAINT fk_comment_order2 FOREIGN KEY
(user_id) REFERENCES users(id) );
```

```
CREATE TABLE seller_reviews ( id NUMBER, seller_id NUMBER, user_id NUM-
BER, rating NUMBER, CONSTRAINT pk_seller_reviews PRIMARY KEY (id),
CONSTRAINT chk_rating CHECK (rating BETWEEN 1 AND 5), CONSTRAINT
fk_review_seller FOREIGN KEY (seller_id) REFERENCES sellers (id), CONSTRAINT
fk_review_user FOREIGN KEY (user_id) REFERENCES users (id) );
```

```
SQL> CREATE TABLE users (
id NUMBER,
name VARCHAR2(50),
CONSTRAINT pk_users PRIMARY KEY (id)
); 2 3 4 5

Table created.

SQL> CREATE TABLE sellers (
id NUMBER,
name VARCHAR2(50),
CONSTRAINT pk_sellers PRIMARY KEY (id)
); 2 3 4 5

Table created.

SQL> CREATE TABLE products (
id NUMBER,
name VARCHAR2(20),
seller_id NUMBER,
price NUMBER,
category VARCHAR2(10),
CONSTRAINT pk_products PRIMARY KEY (id),
CONSTRAINT chk_price CHECK (price BETWEEN 20 AND 100),
CONSTRAINT chk_category CHECK (category IN ('tshirt', 'hoodie', 'jacket')),
CONSTRAINT 2 fk_product_seller FOREIGN KEY (seller_id) REFERENCES sellers (id)
); 3 4 5 6 7 8 9 10 11

Table created.
```

```
SQL> CREATE TABLE orders (
product_id NUMBER,
user_id     NUMBER,
CONSTRAINT pk_orders PRIMARY KEY (product_id, user_id),
CONSTRAINT fk_order_product FOREIGN KEY (product_id) REFERENCES products (id),
CONSTRAINT fk_order_user FOREIGN KEY (user_id) REFERENCES users (id)
); 2 3 4 5 6 7

Table created.
```

```
SQL> CREATE TABLE order_comments (
id          NUMBER,
product_id  NUMBER,
user_id     NUMBER,
commenttxt  VARCHAR2(100),
CONSTRAINT pk_order_comments PRIMARY KEY (id),
CONSTRAINT fk_comment_order1 FOREIGN KEY (product_id) REFERENCES products(id),
CONSTRAINT fk_comment_order2 FOREIGN KEY (user_id) REFERENCES users(id)
); 2 3 4 5 6 7 8 9

Table created.

SQL> CREATE TABLE seller_reviews (
id          NUMBER,
seller_id   NUMBER,
user_id     NUMBER,
rating      NUMBER,
CONSTRAINT pk_seller_reviews PRIMARY KEY (id),
CONSTRAINT chk_rating CHECK (rating BETWEEN 1 AND 5),
CONSTRAINT fk_review_seller FOREIGN KEY (seller_id) REFERENCES sellers (id),
CONSTRAINT fk_review_user FOREIGN KEY (user_id) REFERENCES users (id)
); 2 3 4 5 6 7 8 9 10

Table created.
```

6. alter table users add lastname varchar2(20);

```
SQL> alter table users add lastname varchar(20);

Table altered.

SQL> desc users;
Name                               Null?    Type
-----
ID                                  NOT NULL NUMBER
NAME                                VARCHAR2(50)
LASTNAME                           VARCHAR2(20)
```

7. alter table users modify name varchar2(50) not null;
alter table sellers modify name varchar2(50) not null;

```
SQL> desc sellers;
Name                                         Null?      Type
-----
ID                                           NOT NULL   NUMBER
NAME                                         VCHAR2(50)

SQL> desc users;
Name                                         Null?      Type
-----
ID                                           NOT NULL   NUMBER
NAME                                         VCHAR2(50)
LASTNAME                                     VCHAR2(20)

SQL> alter table users modify name varchar(50) not null;

Table altered.

SQL> alter table sellers modify name varchar(50) not null;

Table altered.
```

8. alter table users modify name varchar2(100);

```
SQL> alter table users modify name varchar(100);

Table altered.

SQL> desc users;
Name                                         Null?      Type
-----
ID                                           NOT NULL   NUMBER
NAME                                         NOT NULL   VCHAR2(100)
LASTNAME                                     VCHAR2(20)
```

9. alter table users drop column lastname;

```
SQL> desc users;
Name                                         Null?      Type
-----
ID                                           NOT NULL   NUMBER
NAME                                         NOT NULL   VCHAR2(100)
LASTNAME                                     VCHAR2(20)

SQL> alter table users drop column lastname;

Table altered.

SQL> desc users;
Name                                         Null?      Type
-----
ID                                           NOT NULL   NUMBER
NAME                                         NOT NULL   VCHAR2(100)
```

10. alter table users rename column name to fullname;

```
SQL> desc users;
  Name                               Null?    Type
  -----
  ID                                NOT NULL NUMBER
  NAME                             NOT NULL VARCHAR2(100)

SQL> alter table users rename column name to fullname;

Table altered.

SQL> desc users;
  Name                               Null?    Type
  -----
  ID                                NOT NULL NUMBER
  FULLNAME                         NOT NULL VARCHAR2(100)
```

11. alter table orders add constraint fk_order_seller foreign key (seller_id) references sellers(id);

```
SQL> alter table orders add seller_id number;

Table altered.

SQL> alter table orders add constraint fk_order_seller foreign key (seller_id) references sellers(id);

Table altered.
```

12. alter table products add constraint dlp check(discount < price);

```
SQL> alter table products add discount number;

Table altered.

SQL> alter table products add constraint dlp check(discount < price);

Table altered.
```

13. CREATE OR REPLACE TRIGGER potr BEFORE INSERT OR UPDATE ON orders FOR EACH ROW DECLARE vdiscount products.discount BEGIN SELECT discount INTO vdiscount FROM products WHERE id = :NEW.product_id; IF :NEW.discounted != vdiscount THEN RAISE_APPLICATION_ERROR(-20001, 'Order discount must match product discount'); END IF; END;

```

SQL> CREATE OR REPLACE TRIGGER potr
BEFORE INSERT OR UPDATE ON orders
FOR EACH ROW
DECLARE
    vdiscount products.discount%TYPE;
BEGIN
    SELECT discount
    INTO vdiscount
    FROM products
    WHERE id = :NEW.product_id;
    IF :NEW.discounted != v 2 3 discount THEN
        RAISE_APPLICATION_ERROR(-20001, 'Order discount must match product discount');
    END IF;
END;
/ 4 5 6 7 8 9 10 11 12 13 14 15

Trigger created.

```

14. INSERT INTO users (id, fullname) VALUES (1, 'Alice Johnson');
 INSERT INTO users (id, fullname) VALUES (2, 'Bob Smith');
 INSERT INTO users (id, fullname) VALUES (3, 'Charlie Brown');
 INSERT INTO users (id, fullname) VALUES (4, 'Diana Prince');
 INSERT INTO users (id, fullname) VALUES (5, 'Ethan Hunt');
 INSERT INTO users (id, fullname) VALUES (6, 'Fiona Apple');
 INSERT INTO users (id, fullname) VALUES (7, 'George Martin');

INSERT INTO sellers (id, name) VALUES (1, 'CoolTees');
 INSERT INTO sellers (id, name) VALUES (2, 'HoodieHub');
 INSERT INTO sellers (id, name) VALUES (3, 'JacketWorld');
 INSERT INTO sellers (id, name) VALUES (4, 'UrbanWear');
 INSERT INTO sellers (id, name) VALUES (5, 'FashionFiesta');
 INSERT INTO sellers (id, name) VALUES (6, 'TrendSetters');
 INSERT INTO sellers (id, name) VALUES (7, 'StreetStyle');

INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (1, 'Classic Tee', 1, 25, 'tshirt', 5);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (2, 'Premium Hoodie', 2, 60, 'hoodie', 10);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (3, 'Leather Jacket', 3, 90, 'jacket', 15);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (4, 'Graphic Tee', 1, 30, 'tshirt', 7);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (5, 'Zip Hoodie', 2, 55, 'hoodie', 12);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (6, 'Winter Jacket', 3, 95, 'jacket', 20);
 INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (7, 'Summer Tee', 1, 22, 'tshirt', 0);

INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (1, 1, 1,

```

5);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (2, 2, 2,
10);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (3, 3, 3,
15);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (4, 4, 1,
7);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (5, 5, 2,
12);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (6, 6, 3,
20);
INSERT INTO orders (product_id, user_id, seller_id, discounted) VALUES (7, 7, 1,
0);

```

```

INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (1,
1, 1, 'Great quality!');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (2,
2, 2, 'Very cozy hoodie. ');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (3,
3, 3, 'Love the jacket!');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (4,
4, 4, 'Nice graphic design. ');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (5,
5, 5, 'Good value for price. ');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (6,
6, 6, 'Perfect for winter. ');
INSERT INTO order_comments (id, product_id, user_id, commenttxt) VALUES (7,
7, 7, 'Light and comfortable. ');

```

```

INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (1, 1, 1, 5);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (2, 2, 2, 4);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (3, 3, 3, 5);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (4, 1, 4, 3);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (5, 2, 5, 4);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (6, 3, 6, 5);
INSERT INTO seller_reviews (id, seller_id, user_id, rating) VALUES (7, 1, 7, 4);

```

la requete "insert into orders (product_id, user_id, seller_id, discounted) values (1, 2, 1, 999);" vas echouer a cause de trigger deja déclaré qui necessite que discount = discounted

```

SQL> insert into orders (product_id, user_id, seller_id, discounted) values (1, 2, 1, 999);
insert into orders (product_id, user_id, seller_id, discounted) values (1, 2, 1, 999)
      *
ERROR at line 1:
ORA-20001: Order discount must match product discount
ORA-06512: at "SYSTEM.POTR", line 9
ORA-04088: error during execution of trigger 'SYSTEM.POTR'

```

la requete "INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (8, 'Expensive Tee', 1, 150, 'tshirt', 10); " vas echouer a cause constarint check qui necessite price;discount

```
SQL> INSERT INTO products (id, name, seller_id, price, category, discount)
VALUES (8, 'Expensive Tee', 1, 150, 'tshirt', 10);
  2  INSERT INTO products (id, name, seller_id, price, category, discount)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.CHK_PRICE) violated
```

INSERT INTO products (id, name, seller_id, price, category, discount) VALUES (9, 'Fancy Pants', 2, 50, 'pants', 5); vas echouer car la category doit etre tshirt hoodie ou jacket

```
SQL> INSERT INTO products (id, name, seller_id, price, category, discount)
VALUES (9, 'Fancy Pants', 2, 50, 'pants', 5);
  2  INSERT INTO products (id, name, seller_id, price, category, discount)
*
ERROR at line 1:
ORA-02290: check constraint (SYSTEM.CHK_CATEGORY) violated
```

15. alter table products disable constraint chk_price;
16. delete from products where id=1;
contrainte d'integrité car on a pas specifié de qui fair on delete.

```
SQL> delete from products where id=1;
delete from products where id=1
*
ERROR at line 1:
ORA-02292: integrity constraint (SYSTEM.FK_COMMENT_ORDER1) violated - child
record found
```

17. assuré
18. select * from products, orders, users where orders.product_id = products.id and orders.user_id = users.id and products.discount <10;

```
SQL> select * from products, orders, users where orders.product_id = products.id and orders.user_id = users.id and products.discount >10;
```

ID	NAME	SELLER_ID	PRICE	CATEGORY	DISCOUNT
3	Leather Jacket	3	90	jacket	15
3		3	15	3	
Charlie Brown					
5	Zip Hoodie	2	55	hoodie	12
5		2	12	5	
Ethan Hunt					
6	Winter Jacket	3	95	jacket	20
6		3	20	6	
Fiona Apple					

19. select seller_id , avg(discounted) as average_discount from orders group by seller_id;

```
SQL> select seller_id , avg(discounted) as average_discount from orders group by seller_id;
```

SELLER_ID	AVERAGE_DISCOUNT
1	4
2	11
3	17.5

20. create view qst20 as select seller_id , avg(discounted) as average_discount from orders group by seller_id;

```
SQL> create view qst20 as select seller_id , avg(discounted) as average_discount from orders group by seller_id;
```

View created.

```
SQL> select * from qst20;
```

SELLER_ID	AVERAGE_DISCOUNT
1	4
2	11
3	17.5

2 TP2

1. create user gerertp2 identified by musmus3108;
2. on peut pas connecter car l'utilisateur n'a pas le droit de creer une session

```
SQL> create user gerertp2 identified by musmus3108;

User created.

SQL> commit;

Commit complete.

SQL> exit;
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
bash-4.2$ sqlplus gerertp2/musmus3108@XEPDB1

SQL*Plus: Release 21.0.0.0.0 - Production on Fri Dec 26 15:25:10 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

ERROR:
ORA-01045: user GERERTP2 lacks CREATE SESSION privilege; logon denied
```

3. grant create session to gerertp2;

```
SQL> grant create session to gerertp2;

Grant succeeded.

SQL> exit;
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
bash-4.2$ sqlplus gerertp2/musmus3108@XEPDB1

SQL*Plus: Release 21.0.0.0.0 - Production on Fri Dec 26 15:33:23 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> █
```

4. grant create table, create user to gerertp2;

```
SQL> grant create table, create user to gerertp2;

Grant succeeded.
```

```

SQL> exit;
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
bash-4.2$ sqlplus gerertp2/musmus3108@XEPDB1

SQL*Plus: Release 21.0.0.0.0 - Production on Fri Dec 26 15:37:48 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Last Successful login time: Fri Dec 26 2025 15:33:23 +00:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> create table randomtable(name string);
create table randomtable(name string)
                        *
ERROR at line 1:
ORA-00902: invalid datatype

SQL> create table randomtable(name varchar2(50));

Table created.

SQL> create user randomuser;

User created.

```

5. la table order a été crée pa system donc la requete retourne que le tableau gerertp2.orders n'exxiste pas car il n y a pas une table nommé orders crée par gerertp2, d'autre par randomtable a été crée par gerertp2 donc il peut le lire.

```

SQL> select * from gerertp2.orders;
select * from gerertp2.orders
                        *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> select * from gerertp2.randomtable;

no rows selected

```

6. REMARQUE: JE SAIS PAS SI GERERTP2T EST UNE FAUTE DE FRAPPE OU PAS, J'ASSUME QUE LE BUTE DE LA QUESTION EST DE GERER LES

DROITS D'ACCÉES DES TABLES DONC EN RESUMÉ POUR QUE USER1 PEUT FAIRE SELECT SUR UNE TABLES DE USER2 IL FAUT D'ABORD LUI DONNER LE PREVILEGE AVEC GRANT SELECT ON TABLE TO USER1 ET DEPUIS USER1 SELECT * FROM USER2.TABLE, L'EXEMPLE SUIVANT EST AVEC SYSTEM ET GERERTP2;

```
SQL> grant select on system.orders to gerertp2;

Grant succeeded.
```

```
SQL> select * from system.orders;

PRODUCT_ID    USER_ID    SELLER_ID    DISCOUNTED
-----
          1             1             1             5
          2             2             2            10
          3             3             3            15
          4             4             1             7
          5             5             2            12
          6             6             3            20
          7             7             1             0

7 rows selected.
```

7. revoke select on system.orders from gerertp2;
revoke create session, create table, create user from gerertp2;

```
SQL> revoke select on system.orders from gerertp2;

Revoke succeeded.

SQL> revoke create session, create table, create user from gerertp2;

Revoke succeeded.
```

8. SELECT grantee, granted_role FROM dba_role_privs WHERE grantee = 'GERERTP2';

```
SQL> SELECT grantee, privilege
FROM dba_sys_privs
WHERE grantee = 'GERERTP2';
      2      3
no rows selected
```

9. CREATE PROFILE Gerer_DroitTP2
LIMIT
SESSIONS_PER_USER 3
CPU_PER_CALL 3000
CONNECT_TIME 30
LOGICAL_READS_PER_CALL 1500
PRIVATE_SGA 25
IDLE_TIME 40
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LIFE_TIME 80
PASSWORD_REUSE_TIME 60
PASSWORD_LOCK_TIME 1
PASSWORD_GRACE_TIME 25;

```
SQL> CREATE PROFILE Gerer_DroitTP2
LIMIT
  SESSIONS_PER_USER      3
  CPU_PER_CALL            3000
  CONNECT_TIME            30
  LOGICAL_READS_PER_CALL 1500
  PRIVATE_SGA             25
  IDLE_TIME               40
  FAILED_LOGIN_ATTEMPTS   3
  PASSWORD_LIFE_TIME      80
  PASSWORD_REUSE_TIME     60
  PASSWORD_LOCK_TIME      1
  PASSWORD_GRACE_TIME     25;
      2      3      4      5      6      7      8      9     10     11     12     13
Profile created.
```

10. alter user gerertp2 prodile gerer_DroitTP2

```
SQL> ALTER USER gerertp2 PROFILE Gener_DroitTP2;

User altered.
```

11. on peut pas car l'utilisateur n'a pas de privileges sur cet tables, il faut donner ce privilege aux profile

```
SQL> alter table system.orders add nc varchar(20);
alter table system.orders add nc varchar(20)
*
ERROR at line 1:
ORA-00942: table or view does not exist
```

12. grant alter on system.orders to gerertp2;
alter table system.orders add rn varchar(50);

```
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
bash-4.2$ sqlplus gerertp2/musmus3108@xepdb1

SQL*Plus: Release 21.0.0.0.0 - Production on Fri Dec 26 16:36:00 2025
Version 21.3.0.0.0

Copyright (c) 1982, 2021, Oracle. All rights reserved.

Last Successful login time: Fri Dec 26 2025 16:31:13 +00:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> alter table system.orders add rn varchar2(50);

Table altered.

SQL> desc system.orders;
Name                                Null?    Type
-----
PRODUCT_ID                          NOT NULL NUMBER
USER_ID                             NOT NULL NUMBER
SELLER_ID                           NUMBER
DISCOUNTED                         NUMBER
RN                                   VARCHAR2(50)
```

13. create role gestiontp2;
grant select, insert, update, delete, alter to gestiontp2;

```
SQL> show user
USER is "SYSTEM"
SQL> create role gestiontp2;

Role created.
```

```
SQL> grant select on system.users to gestiontp2;

Grant succeeded.

SQL> grant select on system.sellers to gestiontp2;

Grant succeeded.

SQL> grant select, insert, update, delete on system.products to gestiontp2;

Grant succeeded.
```

14. `grant gestiontp2 to gerertp2; SELECT granted_role FROM dba_role_privs WHERE grantee = 'GERERTP2';`

```
SQL> grant gestiontp2 to gerertp2;

Grant succeeded.
```

```
SQL> select * from system.users where rownum=1;
```

ID
1
Alice Johnson

```
SQL> select * from system.sellers where rownum=1;
```

ID	NAME
1	CoolTees

```
SQL> desc system.products;
```

Name	Null?	Type
ID	NOT NULL	NUMBER
NAME		VARCHAR2(20)
SELLER_ID		NUMBER
PRICE		NUMBER
CATEGORY		VARCHAR2(10)
DISCOUNT		NUMBER

```
SQL> insert into system.products values(10,'yo',1,20,'tshirt', 2);
```

1 row created.

```
SQL> desc system.products;
```

Name	Null?	Type
ID	NOT NULL	NUMBER
NAME		VARCHAR2(20)
SELLER_ID		NUMBER
PRICE		NUMBER
CATEGORY		VARCHAR2(10)
DISCOUNT		NUMBER

```
SQL> SELECT granted_role
FROM dba_role_privs
WHERE grantee = 'GERERTP2';
      2      3
GRANTED_ROLE
-----
GESTIONTP2
```

15. create index ind on sellers(name);
l'index a été crée correctement

```
SQL> create index ind on sellers(name);

Index created.
```

16. grant create any index to gerertp2; quand on essay de creer le meme index encore avec gerertp2 , oracle nous dit que l'index deja exist, on constate que les indexes ne sont pas isolés entre les utilisateurs

```
SQL> GRANT CREATE ANY INDEX TO gerertp2;

Grant succeeded.
```

```
SQL> create index ind on system.sellers(name);
create index ind on system.sellers(name)
                                     *
ERROR at line 1:
ORA-01408: such column list already indexed
```

3 TP3

1. le catalogue DICT contient 3245 lignes et sa structure est TABLE_NAME, COMMENTS

```
3245 rows selected.
```

```
SQL> desc dict;
```

Name	Null?	Type
TABLE_NAME		VARCHAR2(128)
COMMENTS		VARCHAR2(4000)

2. 1/ALL_TAB_COLUMNS: ça donne tout les columns que l'utilisateur courent a le droit a y acceder

```
SQL> desc ALL_TAB_COLUMNS;
```

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(128)
TABLE_NAME	NOT NULL	VARCHAR2(128)
COLUMN_NAME	NOT NULL	VARCHAR2(128)
DATA_TYPE		VARCHAR2(128)
DATA_TYPE_MOD		VARCHAR2(3)
DATA_TYPE_OWNER		VARCHAR2(128)
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(1000)
HIGH_VALUE		RAW(1000)
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)
DEFAULT_ON_NULL		VARCHAR2(3)
IDENTITY_COLUMN		VARCHAR2(3)
EVALUATION_EDITION		VARCHAR2(128)
UNUSABLE_BEFORE		VARCHAR2(128)
UNUSABLE_BEGINNING		VARCHAR2(128)
COLLATION		VARCHAR2(100)

- 2/USER_USERS: contient des informations sure l'utilisateur courent

```
SQL> desc USER_USERS
```

Name	Null?	Type
-----	-----	-----
USERNAME	NOT NULL	VARCHAR2(128)
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)
LOCAL_TEMP_TABLESPACE		VARCHAR2(30)
CREATED	NOT NULL	DATE
INITIAL_RSRC_CONSUMER_GROUP		VARCHAR2(128)
EXTERNAL_NAME		VARCHAR2(4000)
PROXY_ONLY_CONNECT		VARCHAR2(1)
COMMON		VARCHAR2(3)
ORACLE_MAINTAINED		VARCHAR2(1)
INHERITED		VARCHAR2(3)
DEFAULT_COLLATION		VARCHAR2(100)
IMPLICIT		VARCHAR2(3)
ALL_SHARD		VARCHAR2(3)
EXTERNAL_SHARD		VARCHAR2(3)
PASSWORD_CHANGE_DATE		DATE
MANDATORY_PROFILE_VIOLATION		VARCHAR2(3)

3/ALL_CONSTRAINTS: ça affiche tout les contraintes sur les tables que l'utilisateur courant peut y accéder

```
SQL> desc ALL_CONSTRAINTS;
```

Name	Null?	Type
-----	-----	-----
OWNER		VARCHAR2(128)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(128)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(128)
SEARCH_CONDITION		LONG
SEARCH_CONDITION_VC		VARCHAR2(4000)
R_OWNER		VARCHAR2(128)
R_CONSTRAINT_NAME		VARCHAR2(128)
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(128)
INDEX_NAME		VARCHAR2(128)
INVALID		VARCHAR2(7)
VIEW_RELATED		VARCHAR2(14)
ORIGIN_CON_ID		NUMBER

4/USER_TAB_PRIVS: affiche tout les privileges d'utilisateur courant

```
SQL> desc USER_TAB_PRIVS;
  Name                                         Null?    Type
-----
GRANTEE                                       VARCHAR2(128)
OWNER                                        VARCHAR2(128)
TABLE_NAME                                   VARCHAR2(128)
GRANTOR                                       VARCHAR2(128)
PRIVILEGE                                     VARCHAR2(40)
GRANTABLE                                    VARCHAR2(3)
HIERARCHY                                    VARCHAR2(3)
COMMON                                       VARCHAR2(3)
TYPE                                         VARCHAR2(24)
INHERITED                                    VARCHAR2(3)
```

3. select username from USER_USERS;

```
SQL> select username from USER_USERS;

USERNAME
-----
SYSTEM
```

4. ALL_TAB_COLUMNS affiche tout les columns l'utilisateur a le droit a y acceder et USER_TAB_COLUMNS affiche tout les columns qui appartient a l'utilisateur (owner)

5. il faut just interoger USER_TAB_COLUMNS

```
SQL> select unique table_name from user_tab_columns;

TABLE_NAME
-----
ORDER_COMMENTS
SELLER_REVIEWS
USERS
PRODUCTS
ORDERS
SELLERS

6 rows selected.
```

6. system a 151 tableaux, notre utilisateur a 6 seulement

```
SQL> select unique table_name from user_tab_columns;

TABLE_NAME
-----
ORDER_COMMENTS
SELLER_REVIEWS
USERS
PRODUCTS
ORDERS
SELLERS

6 rows selected.
```

```
TABLE_NAME
-----
MVIEW$_ADV_GC
LOGMNR_ATTRCOL$
LOGSTDBY$SKIP
LOGMNR_SHARD_TS
SQLPLUS_PRODUCT_PROFILE
LOGMNR_CON$
ORDER_COMMENTS
LOGMNR_TABPART$
MVIEW$_ADV_ELIGIBLE
LOGMNR_SHARD_TS
MVIEW$_ADV_AJG

TABLE_NAME
-----
SELLERS
MVIEW$_ADV_ROLLUP
LOGMNR_MDDL$
PRODUCTS
USERS
MVIEW$_ADV_LEVEL
LOGMNR_DBNAME_UID_MAP
REPL_VALID_COMPAT

151 rows selected.
```

7. SELECT table_name, column_name, data_type, data_length, nullable FROM user_tab_columns WHERE table_name IN ('USERS', 'PRODUCTS') ORDER BY table_name, column_id;
8. on peut exploiter ALL_CONSTRAINTS
9. select constraint_name from USER_CONSTRAINTS ;

```
SQL> select constraint_name from USER_CONSTRAINTS ;
```

```
CONSTRAINT_NAME
```

```
-----  
PK_USERS  
PK_SELLERS  
CHK_PRICE  
CHK_CATEGORY  
PK_PRODUCTS  
FK_PRODUCT_SELLER  
PK_ORDERS  
FK_ORDER_PRODUCT  
FK_ORDER_USER  
PK_ORDER_COMMENTS  
FK_COMMENT_ORDER1
```

```
CONSTRAINT_NAME
```

```
-----  
FK_COMMENT_ORDER2  
CHK_RATING  
PK_SELLER_REVIEWS  
FK_REVIEW_SELLER  
FK_REVIEW_USER
```

```
16 rows selected.
```

10. on peut interoger USER_CONSTRAINTS pour les contraintes et interoger USER_TAB_COLUMNS pour avoir les noms et les types de données
11. SELECT owner, table_name, privilege, grantable FROM role_tab_privs WHERE role = 'GESTIONTP2';

```
SQL> SELECT owner, table_name, privilege, grantable
FROM role_tab_privs
WHERE role = 'GESTIOTP2';
  2      3
```

```
OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SYSTEM
PRODUCTS
DELETE                                NO

SYSTEM
PRODUCTS
INSERT                                NO

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SYSTEM
USERS
SELECT                                NO

SYSTEM
SELLERS

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SELECT                                NO

SYSTEM
PRODUCTS
SELECT                                NO

SYSTEM

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
```

12. SELECT grantee, granted_role, admin_option, default_role FROM dba_role_privs
WHERE grantee = 'GERERTP2';

```

SQL> SELECT grantee, granted_role, admin_option, default_role
FROM dba_role_privs
WHERE grantee = 'GERERTP2';

```

GRANTEE	GRANTED_ROLE	ADM DEF
GERERTP2	GESTIONTP2	NO YES

13. SELECT owner, table_name, privilege, grantable FROM role_tab_privs WHERE role = 'GESTIONTP2';

```
SQL> SELECT owner, table_name, privilege, grantable
FROM role_tab_privs
WHERE role = 'GESTIONTP2';
  2      3
```

```
OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SYSTEM
PRODUCTS
DELETE                                NO

SYSTEM
PRODUCTS
INSERT                                NO

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SYSTEM
USERS
SELECT                                NO

SYSTEM
SELLERS

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
SELECT                                NO

SYSTEM
PRODUCTS
SELECT                                NO

SYSTEM

OWNER
-----
TABLE_NAME
-----
PRIVILEGE                                GRA
-----
```

14. SELECT owner, table_name FROM all_tables WHERE table_name = 'ORDERS';

```

SQL> show user
USER is "SYSTEM"
SQL> SELECT owner, table_name
FROM all_tables
WHERE table_name = 'ORDERS';
   2      3
OWNER
-----
TABLE_NAME
-----
SYSTEM
ORDERS

TP1_BELMOULOU
ORDERS

```

15. SELECT segment_name, bytes / 1024 AS size_kb FROM user_segments WHERE segment_type = 'TABLE' AND segment_name = 'ORDERS';

```

16 rows selected.

SQL> SELECT segment_name, bytes / 1024 AS size_kb
FROM user_segments
WHERE segment_type = 'TABLE'
      AND segment_name = 'ORDERS';
   2      3      4
SEGMENT_NAME
-----
      SIZE_KB
-----
ORDERS
      64

```