

TRUONG Vo Ky  
NGUYEN Stéphane  
QUAGLIAROLI Jean-Baptiste  
ADJAHO Prince

# **PROJET GLPI**

**TRAITEMENT ET ADMINISTRATION  
DE BASES DE DONNÉES**

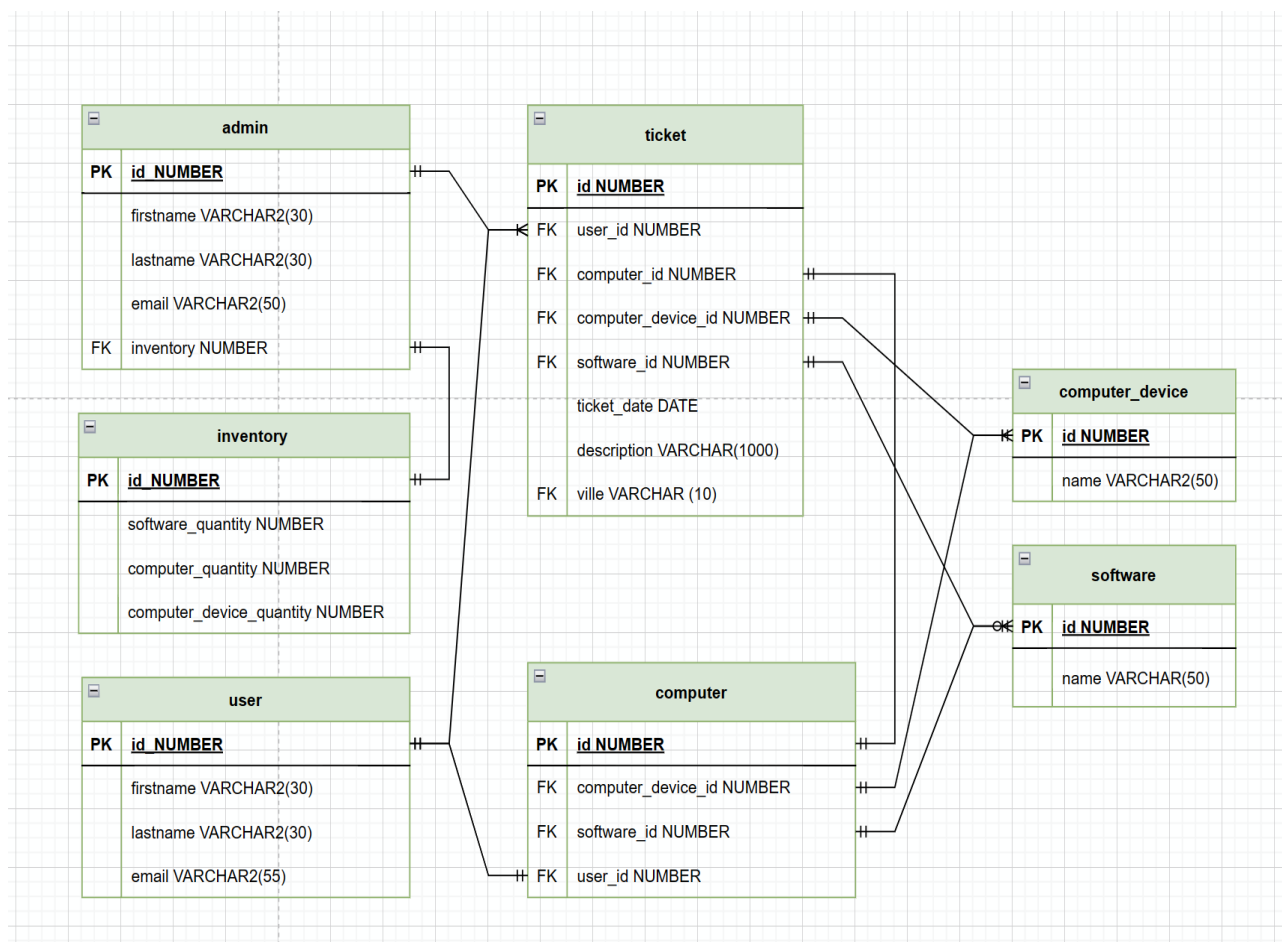


**Mai 2023**

# **TABLE DES MATIÈRES**

TABLE DES MATIÈRES	2
<b>Diagramme de relations d'entités (ERD)</b>	<b>3</b>
<b>Base de données réparties (BDDR)</b>	<b>4</b>
<b>Explication du code</b>	<b>5</b>
<b>USERS / RÔLES / TABLESPACES</b>	<b>5</b>
Création des tables de Cergy	6
<b>Clusters</b>	<b>6</b>
Création des tables de Pau	8
<b>Vues matérialisées de Pau</b>	<b>9</b>
<b>Vue matérialisée de Cergy</b>	<b>11</b>
<b>Procédures</b>	<b>12</b>
Procédure d'affichage	12
Procédure d'affichage avec <b>CURSEUR</b>	12
Procédure d'affichage avec <b>CURSEUR</b>	13
Procédure d'affichage	14
Procédure d'affichage	14
Procédure de création avec <b>EXCEPTION</b>	14
Procédure d'affichage	16
<b>Fonctions</b>	<b>16</b>
<b>Indexes</b>	<b>19</b>
<b>Plan de requêtes</b>	<b>21</b>
<b>Triggers</b>	<b>24</b>
Remarque du trigger	24
Remarque du trigger	25
Remarque du trigger	26
<b>Séquences</b>	<b>27</b>
Remarque de la séquence	29
<b>Notes</b>	<b>30</b>
Lien du code GITHUB	30
Exécution du code	30

# Diagramme de relations d'entités (ERD)



Nous disposons de deux sites : Cergy et Pau. Cergy est le site principal, il gère les **logiciels**, utilise un **dashboard** qui liste les **tickets** des deux sites et gère l'**inventaire** des **deux sites**.

Pau aura donc une vue matérialisée sur l'**inventaire** de Cergy, les **softwares** de Cergy (car on considère que les softwares sont des éléments dématérialisés et qui ne sont stockées que dans la base de données de Cergy), et les infos de l'**administrateur** (qui se trouve à Cergy).

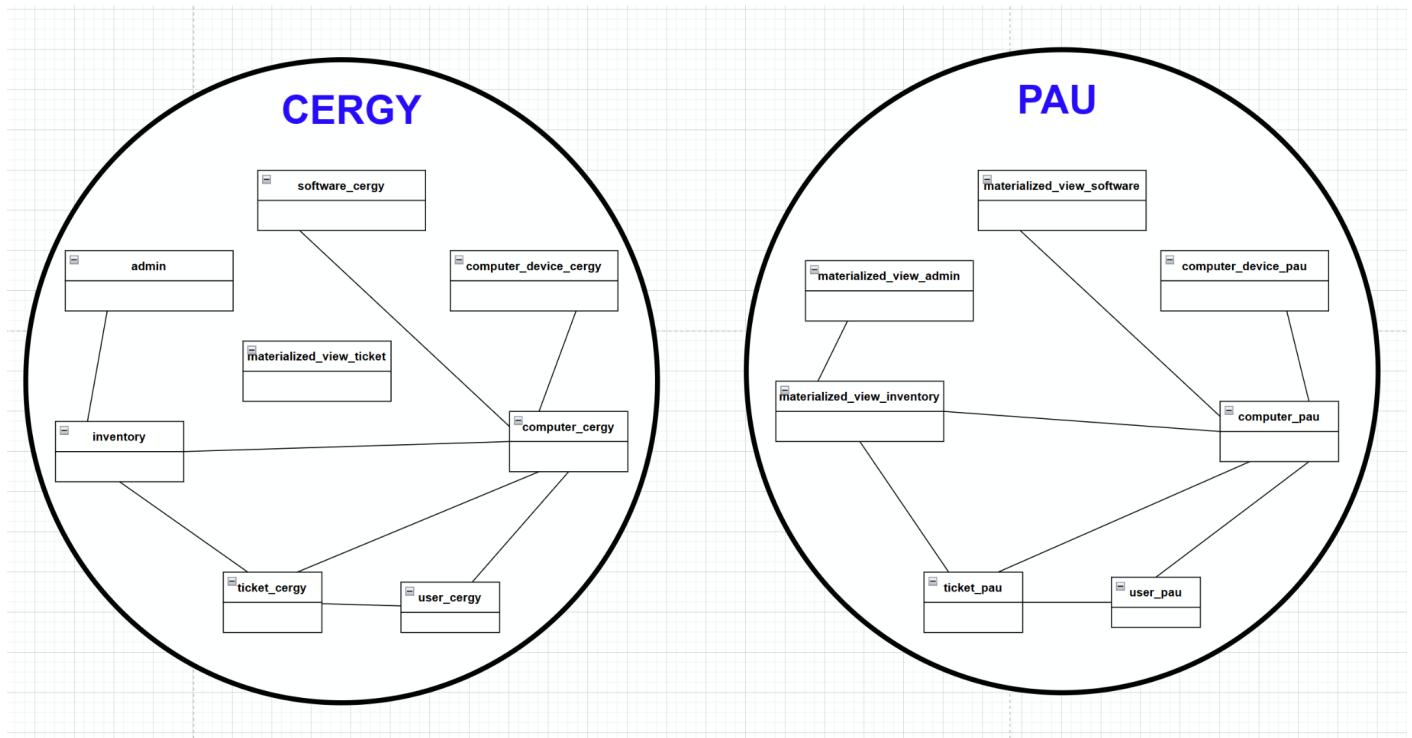
On part du diagramme de relations d'entités (ERD) qui fait office de schéma global. Ce schéma n'existe que dans la conception, il n'est pas implémenté à 100%.

Nous appliquons la méthode Top-Down pour réaliser la base de données répartie (BDDR).

Pour représenter les deux sites, nous avons créé 2 users : user\_cergy et user\_pau qui ont leur propre tablespace.

Pour créer une table sur un des deux sites, il faut se connecter sur un des deux utilisateurs.

## Base de données réparties (BDDR)



Des vues matérialisées ont été créées afin d'améliorer les performances et le stockage. Ce sont des tables temporaires qui utilisent les résultats des requêtes d'une table. Ainsi, sur le site de Pau nous avons créé des vues (par exemple la vue Inventory car c'est Cergy qui gère les stocks du matériel). Il suffit d'accéder à la vue pour voir le matériel disponible et pas besoin de lancer une requête sur la table Inventory.

La vue matérialisée `materialized_view_ticket` est créée pour diminuer les coûts réseau car `ticket_pau` et `ticket_cergy` ne sont pas sur le même site. C'est l'union des tables `ticket_cergy` et `ticket_pau`.

Des fragmentations horizontales sur les tables suivantes selon leur ville pour améliorer les performances et diminuer le temps des requêtes :

- on sépare la table `user` en `user_cergy`, `user_pau`
- on sépare la table `computer` en `computer_cergy`, `computer_pau`
- on sépare la table `computer_device` en `computer_device_cergy`, `computer_device_pau`
- on sépare la table `ticket` en `ticket_cergy`, `ticket_pau`
- on sépare la table `user` en `user_cergy`, `user_pau`

# Explication du code

## USERS / RÔLES / TABLESPACES

Pour représenter les deux sites, nous avons créé 2 users : user\_cergy et user\_pau qui ont leur propre tablespace.

```
01_database > 01_db_system.sql
1  /** ===== USERS AND TABLESPACES CREATION ===== */
2  CREATE USER user_cergy IDENTIFIED BY user_cergy;
3  CREATE USER user_pau IDENTIFIED BY user_pau;
4  GRANT CONNECT, RESOURCE, CREATE VIEW, CREATE MATERIALIZED VIEW, CREATE DATABASE LINK TO user_cergy;
5  GRANT CONNECT, RESOURCE, CREATE VIEW, CREATE MATERIALIZED VIEW, CREATE DATABASE LINK TO user_pau;
6
7  /** ===== TABLESPACE LINK CREATION ===== */
8  -- Link cergy to pau from Pau database
9  drop database link database_cergy_to_pau;
10 create database link database_cergy_to_pau connect to user_cergy identified by user_cergy using 'XE';
11
12 drop database link database_pau_to_cergy;
13 create database link database_pau_to_cergy connect to user_pau identified by user_pau using 'XE';
14
```

Pour créer une table sur un des deux sites, il faut se connecter sur un des deux utilisateurs.

On va commencer par celui de Cergy.

## Création des tables de Cergy

On se connecte sur Cergy. On crée les tables :

```
01_db_system.sql 02_db_cergy.sql x 03_db_pau.sql 05_2_procedures_create.sql 06_sequences

01_database > 02_db_cergy.sql

15  /** ===== TABLE CREATION ===== */
16
17  -- CERGY INFRASTRUCTURE
18  --Creation d'un cluster user/ticket/computer
19  DROP TABLE ticket_cergy CASCADE CONSTRAINTS;
20  DROP TABLE computer_cergy CASCADE CONSTRAINTS;
21  DROP CLUSTER my_cluster;
22  CREATE CLUSTER my_cluster
23  (user_id NUMBER)
24  SIZE 1024
25  TABLESPACE users;
26
27  -- Creation des tables
28  DROP TABLE admin CASCADE CONSTRAINTS;
29  CREATE TABLE admin (
30      id NUMBER NOT NULL PRIMARY KEY,
31      firstname VARCHAR2(15) NOT NULL,
32      lastname VARCHAR2(15) NOT NULL,
33      email VARCHAR2(30) NOT NULL,
34      inventory NUMBER NOT NULL
35  );
36
37  DROP TABLE inventory CASCADE CONSTRAINTS;
38  CREATE TABLE inventory (
39      id NUMBER NOT NULL PRIMARY KEY,
40      software_quantity NUMBER NOT NULL,
41      computer_quantity NUMBER NOT NULL,
42      computer_device_quantity NUMBER NOT NULL
43  );
44
45  DROP TABLE user_cergy CASCADE CONSTRAINTS;
46  CREATE TABLE user_cergy (
47      id NUMBER NOT NULL PRIMARY KEY,
48      firstname VARCHAR2(15) NOT NULL,
49      lastname VARCHAR2(15) NOT NULL,
50      email VARCHAR2(30) NOT NULL
51  );
52
53  -- Si le ticket ne concerne que l'ordinateur et pas les softwares ou les devices
54  -- alors on autorise les valeurs NULL pour les colonnes software_id et computer_device_id
55  DROP TABLE ticket_cergy CASCADE CONSTRAINTS;
56  CREATE TABLE ticket_cergy (
57      id NUMBER NOT NULL PRIMARY KEY,
58      user_id NUMBER NOT NULL,
59      computer_id NUMBER,
60      computer_device_id NUMBER,
61      software_id NUMBER,
62      ticket_date DATE NOT NULL,
63      description VARCHAR(30) NOT NULL,
64      city VARCHAR(10) NOT NULL
65  )CLUSTER my_cluster (user_id);
66
67  DROP TABLE computer_cergy CASCADE CONSTRAINTS;
68  CREATE TABLE computer_cergy (
69      id NUMBER NOT NULL PRIMARY KEY,
70      computer_device_id NUMBER,
71      software_id NUMBER,
72      user_id NUMBER
73  )CLUSTER my_cluster(user_id);
```

## Clusters

Les clusters que nous avons ici lors de la création des tables de Cergy, sont utilisés lorsque nous avons des tables qui sont demandées fréquemment ensemble dans les requêtes. Ils servent à regrouper les tables sur le disque pour réduire le coût des opérations des requêtes ainsi améliorer les performances des requêtes.

On n'oublie pas les FOREIGN KEY :

```
91
92 ALTER TABLE admin ADD CONSTRAINT fk_inventory FOREIGN KEY (inventory) REFERENCES inventory (id);
93
94 ALTER TABLE ticket_cergy ADD CONSTRAINT fk_ticket_user_id FOREIGN KEY (user_id) REFERENCES user_cergy (id);
95 ALTER TABLE ticket_cergy ADD CONSTRAINT fk_ticket_computer_device_id FOREIGN KEY (computer_device_id) REFERENCES computer_device_cergy (id);
96 ALTER TABLE ticket_cergy ADD CONSTRAINT fk_ticket_software_id FOREIGN KEY (software_id) REFERENCES software (id);
97 ALTER TABLE ticket_cergy ADD CONSTRAINT fk_ticket_computer_id FOREIGN KEY (computer_id) REFERENCES computer_cergy (id);
98
99 ALTER TABLE computer_cergy ADD CONSTRAINT fk_computer_user_id FOREIGN KEY (user_id) REFERENCES user_cergy (id);
100 ALTER TABLE computer_cergy ADD CONSTRAINT fk_computer_computer_device_id FOREIGN KEY (computer_device_id) REFERENCES computer_device_cergy (id);
101 ALTER TABLE computer_cergy ADD CONSTRAINT fk_computer_software_id FOREIGN KEY (software_id) REFERENCES software (id);
102
```

On insère une base de données avec des INSERT INTO :

```
119
120 /** ===== INSERTIONS DE DONNEES CERGY ===== */
121
122 -- Lignes de code à entrer dans le terminal dès le début pour AFFICHER CORRECTEMENT :
123 set wrap off;
124 set linesize 250;
125
126 -- Users Cergy
127 DELETE FROM user_cergy;
128 INSERT INTO user_cergy (id, firstname, lastname, email) VALUES (1001, 'Jean', 'Dupont', 'jean.dupont@cergy.fr');
129 INSERT INTO user_cergy (id, firstname, lastname, email) VALUES (1002, 'Marie', 'Martin', 'marie.martin@cergy.fr');
130 INSERT INTO user_cergy (id, firstname, lastname, email) VALUES (1003, 'Luc', 'Dubois', 'luc.dubois@cergy.fr');
131 INSERT INTO user_cergy (id, firstname, lastname, email) VALUES (1004, 'Sophie', 'Lecomte', 'sophie.lecomte@cergy.fr');
132 INSERT INTO user_cergy (id, firstname, lastname, email) VALUES (1005, 'Pierre', 'Morel', 'pierre.morel@cergy.fr');
133
134 -- Computer Device Cergy
135 DELETE FROM computer_device_cergy;
136 INSERT INTO computer_device_cergy (id, name) VALUES (3001, 'souris');
137 INSERT INTO computer_device_cergy (id, name) VALUES (3002, 'clavier');
138 INSERT INTO computer_device_cergy (id, name) VALUES (3003, 'écran');
139 INSERT INTO computer_device_cergy (id, name) VALUES (3004, 'cable');
140 INSERT INTO computer_device_cergy (id, name) VALUES (3005, 'imprimante');
141
142 -- Software
143 DELETE FROM software;
144 INSERT INTO software (id, name) VALUES (4001, 'Windows');
145 INSERT INTO software (id, name) VALUES (4002, 'Ubuntu');
146 INSERT INTO software (id, name) VALUES (4003, 'Office');
147 INSERT INTO software (id, name) VALUES (4004, 'Adobe');
148 INSERT INTO software (id, name) VALUES (4005, 'Antivirus');
149
150 -- Computer Cergy
151 DELETE FROM computer_cergy;
152 INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2001, 3001, 4001, 1001);
153 INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2002, 3002, 4002, 1002);
154 INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2003, 3003, 4003, 1003);
155 INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2004, 3004, 4004, 1004);
156 INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2005, 3005, 4005, 1005);
157
158 -- Inventory
159 DELETE FROM inventory;
160 INSERT INTO inventory (id, software_quantity, computer_quantity, computer_device_quantity) VALUES (1001, 100, 50, 70);
161
162 -- Admin
163 DELETE FROM admin;
164 INSERT INTO admin (id, firstname, lastname, email, inventory) VALUES (1001, 'VoKy', 'TRUONG', 'voky@cergy.fr', 1001);
165
166 -- Ticket Cergy
167 DELETE FROM ticket_cergy;
168 INSERT INTO ticket_cergy (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5001,
169 INSERT INTO ticket_cergy (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5002,
170 INSERT INTO ticket_cergy (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5003,
171 INSERT INTO ticket_cergy (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5004,
172 INSERT INTO ticket_cergy (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5005,
173
174 -- Affichage des tickets
175 select * from ticket_cergy;
176
177
```

Et enfin, voici un exemple d'affichage des tickets de Cergy :

```
SQL>
SQL> -- Affichage des tickets
SQL> select * from ticket_cergy;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5001	1001		3001		31-MAR-23	souris ne marche pas	CERGY
5002	1002		3002		31-MAR-23	clavier ne marche pas	CERGY
5003	1003		3001		31-MAR-23	souris ne marche pas	CERGY
5004	1004			4003	01-APR-23	besoin du logiciel office	CERGY
5005	1005	2005			01-APR-23	ordinateur en panne	CERGY

Nous faisons maintenant pareil pour Pau.

## Création des tables de Pau

```
12 */
13 drop database link database_cergy_to_pau;
14 create database link database_cergy_to_pau connect to user_cergy identified by user_cergy using 'XE';
15
16 /** ===== PAU INFRASTRUCTURE ===== */
17
18 DROP TABLE user_pau CASCADE CONSTRAINTS;
19 CREATE TABLE user_pau (
20     id NUMBER NOT NULL PRIMARY KEY,
21     firstname VARCHAR2(15) NOT NULL,
22     lastname VARCHAR2(15) NOT NULL,
23     email VARCHAR2(30) NOT NULL
24 );
25
26 DROP TABLE ticket_pau CASCADE CONSTRAINTS;
27 CREATE TABLE ticket_pau (
28     id NUMBER NOT NULL PRIMARY KEY,
29     user_id NUMBER NOT NULL,
30     computer_id NUMBER,
31     computer_device_id NUMBER,
32     software_id NUMBER,
33     ticket_date DATE NOT NULL,
34     description VARCHAR(30),
35     city VARCHAR(10) NOT NULL
36 );
37
38 DROP TABLE computer_pau CASCADE CONSTRAINTS;
39 CREATE TABLE computer_pau (
40     id NUMBER NOT NULL PRIMARY KEY,
41     computer_device_id NUMBER,
42     software_id NUMBER,
43     user_id NUMBER
44 );
45
46 DROP TABLE computer_device_pau CASCADE CONSTRAINTS;
47 CREATE TABLE computer_device_pau (
48     id NUMBER NOT NULL PRIMARY KEY,
49     name VARCHAR2(20) NOT NULL
50 );
51
52 ALTER TABLE ticket_pau ADD CONSTRAINT fk_ticket_user_id FOREIGN KEY (user_id) REFERENCES user_pau (id);
53 ALTER TABLE ticket_pau ADD CONSTRAINT fk_ticket_computer_device_id FOREIGN KEY (computer_device_id) REFERENCES computer_device_pau (id);
54 ALTER TABLE ticket_pau ADD CONSTRAINT fk_ticket_computer_id FOREIGN KEY (computer_id) REFERENCES computer_pau (id);
55
56 ALTER TABLE computer_pau ADD CONSTRAINT fk_computer_user_id FOREIGN KEY (user_id) REFERENCES user_pau (id);
57 ALTER TABLE computer_pau ADD CONSTRAINT fk_computer_computer_device_id FOREIGN KEY (computer_device_id) REFERENCES computer_device_pau (id);
58
59
```



## Vues matérialisées de Pau

Comme expliqué à la page n°5, nous générons des vues matérialisées à partir de Pau, qui se connectera à la base de données de Cergy, sur les tables suivantes : **software**, **admin**, **inventory**.

```
60
61  /** ===== PAU MATERIALIZED VIEW ===== */
62  -- Materialized view from software cergy
63  DROP MATERIALIZED VIEW materialized_view_software;
64  CREATE MATERIALIZED VIEW materialized_view_software AS
65  SELECT * FROM software@database_cergy_to_pau;
66
67  -- Materialized view from inventory cergy
68  DROP MATERIALIZED VIEW materialized_view_inventory;
69  CREATE MATERIALIZED VIEW materialized_view_inventory AS
70  SELECT * FROM inventory@database_cergy_to_pau;
71
72  -- Materialized view from admin cergy
73  DROP MATERIALIZED VIEW materialized_view_admin;
74  CREATE MATERIALIZED VIEW materialized_view_admin AS
75  SELECT * FROM admin@database_cergy_to_pau;
76
77
78  -- tester la requête
79  select * from materialized_view_software;
80  select * from materialized_view_inventory;
81  select * from materialized_view_admin;
82
83
```

Voici l'affichage :

```
SQL> select * from materialized_view_software;

  ID NAME
-----
 4001 Windows
 4002 Ubuntu
 4003 Office
 4004 Adobe
 4005 Antivirus

SQL> select * from materialized_view_inventory;

  ID SOFTWARE_QUANTITY COMPUTER_QUANTITY COMPUTER_DEVICE_QUANTITY
-----
 1001                100                50                70

SQL> select * from materialized_view_admin;

  ID FIRSTNAME  LASTNAME  EMAIL  INVENTORY
-----
 1001 VoKy      TRUONG   voky@cergy.fr  1001
```

Les vues matérialisées de Pau étant créées, on n'oublie pas maintenant de faire les **INSERT INTO**, qui se servent des vues matérialisées pour créer la base de données de Pau.

```
91
92 -- Users Pau
93 DELETE FROM user_pau;
94 INSERT INTO user_pau (id, firstname, lastname, email) VALUES (1001, 'John', 'Doe', 'john.doe@pau.com');
95 INSERT INTO user_pau (id, firstname, lastname, email) VALUES (1002, 'Jane', 'Doe', 'jane.doe@pau.com');
96 INSERT INTO user_pau (id, firstname, lastname, email) VALUES (1003, 'Bob', 'Smith', 'bob.smith@pau.com');
97 INSERT INTO user_pau (id, firstname, lastname, email) VALUES (1004, 'Alice', 'Johnson', 'alice.johnson@pau.com');
98 INSERT INTO user_pau (id, firstname, lastname, email) VALUES (1005, 'David', 'Lee', 'david.lee@pau.com');
99
100 -- Computer Device Pau
101 DELETE FROM computer_device_pau;
102 INSERT INTO computer_device_pau (id, name) VALUES (3001, 'souris');
103 INSERT INTO computer_device_pau (id, name) VALUES (3002, 'clavier');
104 INSERT INTO computer_device_pau (id, name) VALUES (3003, 'écran');
105 INSERT INTO computer_device_pau (id, name) VALUES (3004, 'cable réseau');
106 INSERT INTO computer_device_pau (id, name) VALUES (3005, 'imprimante');
107
108 -- Computer Pau
109 DELETE FROM computer_pau;
110 INSERT INTO computer_pau (id, computer_device_id, software_id, user_id) VALUES (2001, 3001, 4001, 1001);
111 INSERT INTO computer_pau (id, computer_device_id, software_id, user_id) VALUES (2002, 3002, 4002, 1002);
112 INSERT INTO computer_pau (id, computer_device_id, software_id, user_id) VALUES (2003, 3003, 4003, 1003);
113 INSERT INTO computer_pau (id, computer_device_id, software_id, user_id) VALUES (2004, 3004, 4004, 1004);
114 INSERT INTO computer_pau (id, computer_device_id, software_id, user_id) VALUES (2005, 3005, 4005, 1005);
115
116 -- Ticket Pau
117 DELETE FROM ticket_pau;
118 INSERT INTO ticket_pau (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5001,
119 INSERT INTO ticket_pau (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5002,
120 INSERT INTO ticket_pau (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5003,
121 INSERT INTO ticket_pau (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5004,
122 INSERT INTO ticket_pau (id, user_id, computer_id, computer_device_id, software_id, ticket_date, description, city) VALUES (5005,
123
124 -- affichage des tickets
125 select * from ticket_pau;
```

Affichage des tickets de Pau :

```
SQL> select * from ticket_pau;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5001	1001		3005		31-MAR-23	imprimante ne marche pas	PAU
5002	1002		3004		31-MAR-23	cable réseau ne marche pas	PAU
5003	1003	2003			31-MAR-23	ordinateur ne marche pas	PAU
5004	1004			4003	01-APR-23	besoin du logiciel office	PAU
5005	1005			4004	01-APR-23	besoin du logiciel adobe	PAU

```
SQL>
```

## Vue matérialisée de Cergy

Les deux bases de données étant créées, on ajoute maintenant la vue matérialisée de Cergy, qui permet d'avoir une supervision des tickets de Cergy et également de Pau.

```
20
21  /** ===== CERGY MATERIALIZED VIEW ===== */
22  -- DASHBOARD is CERGY : Display all tickets from Cergy and Pau
23  DROP MATERIALIZED VIEW materialized_view_tickets;
24  CREATE MATERIALIZED VIEW materialized_view_tickets
25  AS
26  SELECT * FROM ticket_cergy
27  UNION ALL
28  SELECT * FROM ticket_pau@database_cergy_to_pau;
29
30  -- tester la requête
31  select * from materialized_view_tickets;
32
```

Cela affiche :

```
SQL> select * from materialized_view_tickets;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5001	1001		3001		31-MAR-23	souris ne marche pas	CERGY
5002	1002		3002		31-MAR-23	clavier ne marche pas	CERGY
5003	1003		3001		31-MAR-23	souris ne marche pas	CERGY
5004	1004			4003	01-APR-23	besoin du logiciel office	CERGY
5005	1005	2005			01-APR-23	ordinateur en panne	CERGY
5001	1001		3005		31-MAR-23	imprimante ne marche pas	PAU
5002	1002		3004		31-MAR-23	cable réseau ne marche pas	PAU
5003	1003	2003			31-MAR-23	ordinateur ne marche pas	PAU
5004	1004			4003	01-APR-23	besoin du logiciel office	PAU
5005	1005			4004	01-APR-23	besoin du logiciel adobe	PAU

```
10 rows selected.
```

## Vues logiques

Nous avons réalisé quelques vues logiques, qui affichent tous les utilisateurs, tout en filtrant quelques unes de leurs données.

```
create or replace view vue as select u.firstname, u.lastname, cc.id as
"Computer id Cergy" from user_cergy u, computer_cergy cc where u.id =
cc.user_id;

select * from vue;
```

```
SQL> select * from vue;
```

FIRSTNAME	LASTNAME	Computer id Cergy
Jean	Dupont	2001
Marie	Martin	2002
Luc	Dubois	2003
Sophie	Lecomte	2004
Pierre	Morel	2005

```
create or replace view vuemail
as
select email from user_cergy uc
union all
select email from user_pau@database_pau_to_cergy up;
select * from vuemail;
```

```
EMAIL
-----
jean.dupont@cergy.fr
marie.martin@cergy.fr
luc.dubois@cergy.fr
sophie.lecomte@cergy.fr
pierre.morel@cergy.fr
john.doe@pau.com
jane.doe@pau.com
bob.smith@pau.com
alice.johnson@pau.com
david.lee@pau.com

10 rows selected.
```

## Procédures

Nous avons créé différentes sortes de procédures.  
Voici leurs codes et leurs résultats :

### Procédure d'affichage

```

/*****
-- PROCEDURE : afficher le prénom de l'utilisateur de Cergy qui a l'id 1001
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afficher_user_cergy_1001 IS
-- DECLARE
    v_first_name user_cergy.firstname%TYPE;
BEGIN
    SELECT firstname INTO v_first_name FROM user_cergy WHERE id = 1001;
    DBMS_OUTPUT.PUT_LINE('Le prénom de l'utilisateur 1001 est : ' || v_first_name);
END;
/
EXECUTE afficher_user_cergy_1001;
*****/
```

Procedure created.

```
SQL> EXECUTE afficher_user_cergy_1001;
Le prénom de l'utilisateur 1001 est : Jean
```

PL/SQL procedure successfully completed.

### Procédure d'affichage avec CURSEUR

```

/*****
-- PROCEDURE CURSEUR / BOUCLE FOR : afficher les tickets de Cergy et de Pau
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afficher_tickets_cergy_et_pau IS
BEGIN
    FOR cur IN (SELECT * FROM materialized_view_tickets) LOOP
        DBMS_OUTPUT.PUT_LINE('Ticket ID: ' || cur.id || ', Description: ' || cur.description || ', City: ' ||
cur.city);
    END LOOP;
END;
/
EXECUTE afficher_tickets_cergy_et_pau;
*****/
```

```
SQL> EXECUTE afficher_tickets_cergy_et_pau;
Ticket ID: 5001, Description: souris ne marche pas, City: CERGY
Ticket ID: 5002, Description: clavier ne marche pas, City: CERGY
Ticket ID: 5003, Description: souris ne marche pas, City: CERGY
Ticket ID: 5004, Description: besoin du logiciel office, City: CERGY
Ticket ID: 5005, Description: ordinateur en panne, City: CERGY
Ticket ID: 5001, Description: imprimante ne marche pas, City: PAU
Ticket ID: 5002, Description: cable réseau ne marche pas, City: PAU
Ticket ID: 5003, Description: ordinateur ne marche pas, City: PAU
Ticket ID: 5004, Description: besoin du logiciel office, City: PAU
Ticket ID: 5005, Description: besoin du logiciel adobe, City: PAU

PL/SQL procedure successfully completed.
```

## Procédure d'affichage avec CURSEUR

```

/*****
-- PROCEDURE CURSEUR/FETCH : afficher les tickets de Cergy et de Pau
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afficher_tickets_cergy_et_pau IS
    CURSOR curseur IS SELECT id, description, city, ticket_date FROM materialized_view_tickets;
    v_id NUMBER;
    v_description VARCHAR2(100);
    v_city VARCHAR2(100);
    v_date DATE;

    CURSOR curseur_cergy IS SELECT id, firstname, lastname FROM user_cergy;
    v_id_cergy user_cergy.id%TYPE;
    v_first_name_cergy user_cergy.firstname%TYPE;
    v_last_name_cergy user_cergy.lastname%TYPE;

    CURSOR curseur_pau IS SELECT id, firstname, lastname FROM user_pau@database_pau_to_cergy;
    v_id_pau NUMBER;
    v_first_name_pau VARCHAR2(100);
    v_last_name_pau VARCHAR2(100);

BEGIN
    OPEN curseur;
    OPEN curseur_cergy;
    OPEN curseur_pau;

    LOOP
        FETCH curseur INTO v_id, v_description, v_city, v_date;
        EXIT WHEN curseur%NOTFOUND;

        DBMS_OUTPUT.PUT('Ticket ID: ' || v_id || CHR(9));

        IF v_city = 'CERGY' THEN
            FETCH curseur_cergy INTO v_id_cergy, v_first_name_cergy, v_last_name_cergy;
            DBMS_OUTPUT.PUT(', First Name : ' || v_first_name_cergy || CHR(9) || ', Last Name : ' || v_last_name_cergy || CHR(9));
        END IF;
        IF v_city = 'PAU' THEN
            FETCH curseur_pau INTO v_id_pau, v_first_name_pau, v_last_name_pau;
            DBMS_OUTPUT.PUT(', First Name : ' || v_first_name_pau || CHR(9) || ', Last Name : ' || v_last_name_pau || CHR(9));
        END IF;

        DBMS_OUTPUT.PUT_LINE(' Description: ' || v_description || CHR(9) || ', City: ' || v_city || CHR(9) || ', Date: ' ||
v_date);

    END LOOP;

    -- AUTRES FACONS DE FAIRE : WHILE / FOR
    -- while curseur%FOUND LOOP
    --     FETCH curseur INTO v_id, v_description, v_city;
    -- end loop;

    -- FOR cur IN (SELECT * FROM materialized_view_tickets) LOOP
    --     DBMS_OUTPUT.PUT_LINE('Ticket ID: ' || cur.id || ', Description: ' || cur.description || ', City: ' || cur.city);
    -- END LOOP;
END;
/
EXECUTE afficher_tickets_cergy_et_pau;
*****/
```

Procedure created.

```
SQL> EXECUTE afficher_tickets_cergy_et_pau;
Ticket ID: 5001 , First Name : Jean      , Last Name : Dupont    , Description: souris ne marche pas    , City: CERGY, Date: 31-MAR-23
Ticket ID: 5002 , First Name : Marie    , Last Name : Martin    , Description: clavier ne marche pas   , City: CERGY, Date: 31-MAR-23
Ticket ID: 5003 , First Name : Luc      , Last Name : Dubois    , Description: souris ne marche pas    , City: CERGY, Date: 31-MAR-23
Ticket ID: 5004 , First Name : Sophie   , Last Name : Lecomte   , Description: besoin du logiciel office , City: CERGY      , Date: 01-APR-23
Ticket ID: 5005 , First Name : Pierre   , Last Name : Morel     , Description: ordinateur en panne     , City: CERGY, Date: 01-APR-23
Ticket ID: 5001 , First Name : John     , Last Name : Doe       , Description: imprimante ne marche pas , City: PAU
      , Date: 31-MAR-23
Ticket ID: 5002 , First Name : Jane     , Last Name : Doe       , Description: cable réseau ne marche pas , City: PAU , Date: 31-MAR-23
Ticket ID: 5003 , First Name : Bob      , Last Name : Smith     , Description: ordinateur ne marche pas , City: PAU
      , Date: 31-MAR-23
Ticket ID: 5004 , First Name : Alice    , Last Name : Johnson   , Description: besoin du logiciel office , City: PAU , Date: 01-APR-23
Ticket ID: 5005 , First Name : David    , Last Name : Lee       , Description: besoin du logiciel adobe , City: PAU
      , Date: 01-APR-23

PL/SQL procedure successfully completed.
```

## Procédure d'affichage

```

/*****
-- PROCEDURE : afficher un compteur qui augmente en fonction du nombre de ligne de tickets
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afficher_tickets_cergy_et_pau IS
    i NUMBER;
    v_count NUMBER;
BEGIN
    select (count(*)) into v_count from materialized_view_tickets;
    FOR i in 1..v_count LOOP
        DBMS_OUTPUT.PUT_LINE(i);
    END LOOP;
END;
/
EXECUTE afficher_tickets_cergy_et_pau;
*****/
```

```
SQL> EXECUTE afficher_tickets_cergy_et_pau;
1
2
3
4
5
6
7
8
9
10

PL/SQL procedure successfully completed.
```

## Procédure d'affichage

```

/*****
-- PROCEDURE : afficher les user de Cergy
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE afficher_users_cergy IS
BEGIN
    FOR cur IN (SELECT * FROM user_cergy) LOOP
        DBMS_OUTPUT.PUT_LINE(cur.id || CHR(9) || ' ' || cur.firstname || CHR(9) || ' ' || cur.lastname ||
CHR(9) || ' ' || cur.email);
    END LOOP;
END;
/
EXECUTE afficher_users_cergy;
*****/
```

Procedure created.

```
SQL> EXECUTE afficher_users_cergy;
1001    Jean    Dupont    jean.dupont@cergy.fr
1002    Marie   Martin   marie.martin@cergy.fr
1003    Luc     Dubois    luc.dubois@cergy.fr
1004    Sophie  Lecomte   sophie.lecomte@cergy.fr
1005    Pierre  Morel     pierre.morel@cergy.fr

PL/SQL procedure successfully completed.
```

## Procédure de création avec EXCEPTION

```

/*****
-- PROCEDURE : CREATION TICKET
CREATE OR REPLACE PROCEDURE create_ticket(
    p_id IN ticket_cergy.id%TYPE,
    p_user_id IN ticket_cergy.user_id%TYPE,
    p_description IN ticket_cergy.description%TYPE,
    p_ticket_date IN ticket_cergy.ticket_date%TYPE,
    p_computer_id IN ticket_cergy.computer_id%TYPE,
    p_computer_device_id IN ticket_cergy.computer_device_id%TYPE,
    p_software_id IN ticket_cergy.software_id%TYPE,
    p_city IN ticket_cergy.city%TYPE
) AS
BEGIN
    INSERT INTO ticket_cergy(id, user_id, description, ticket_date, computer_id, computer_device_id,
software_id, city)
    VALUES(p_id, p_user_id, p_description, p_ticket_date, p_computer_id, p_computer_device_id, p_software_id,
p_city);

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Ticket created successfully');

EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('A ticket with the id has already been created');
        DBMS_OUTPUT.PUT_LINE('Solution : enter a different p_ticket_id');
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error creating ticket: ' || SQLERRM);
END;
/

```

```

-- EXECUTION QUI MARCHE : on utilise un id qui n'existe pas
DELETE FROM ticket_cergy WHERE id = 5050;
EXECUTE create_ticket(5050, 1001, 'Probleme de clavier', SYSDATE, NULL, 3002, NULL, 'Cergy');

-- On vérifie que le ticket a bien été créé
select * from ticket_cergy;
/****

```

```
SQL> EXECUTE create_ticket(5050, 1001, 'Probleme de clavier', SYSDATE, NULL, 3002, NULL, 'Cergy');
Ticket created successfully
```

```
PL/SQL procedure successfully completed.
```

```
SQL> select * from ticket_cergy;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5001	1001		3001		31-MAR-23	souris ne marche pas	CERGY
5002	1002		3002		31-MAR-23	clavier ne marche pas	CERGY
5003	1003		3001		31-MAR-23	souris ne marche pas	CERGY
5004	1004			4003	01-APR-23	besoin du logiciel office	CERGY
5005	1005	2005			01-APR-23	ordinateur en panne	CERGY
5050	1001		3002		06-APR-23	Probleme de clavier	Cergy

```
6 rows selected.
```



```
-- EXECUTION QUI NE MARCHE PAS ET LANCE UNE EXCEPTION : on utilise un id qui existe déjà
EXECUTE create_ticket(5001, 1001, 'Probleme de clavier', SYSDATE, NULL, NULL, NULL, 'Cergy');
```

```
SQL> -- EXECUTION QUI NE MARCHE PAS ET LANCE UNE EXCEPTION : on utilise un id qui existe déjà
SQL> EXECUTE create_ticket(5001, 1001, 'Probleme de clavier', SYSDATE, NULL, NULL, NULL, 'Cergy');
A ticket with the id has already been created
Solution : enter a different p_ticket_id

PL/SQL procedure successfully completed.
```

## Procédure d’affichage

```
-- PROCEDURE : afficher le nombre total d'objets dans l'inventaire
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE total_quantity_inventory
IS
    v_software_quantity NUMBER;
    v_computer_quantity NUMBER;
    v_computer_device_quantity NUMBER;
    v_inventaire_total_quantity NUMBER;
BEGIN
    SELECT software_quantity INTO v_software_quantity FROM inventory WHERE id = 1001;
    SELECT computer_quantity INTO v_computer_quantity FROM inventory WHERE id = 1001;
    SELECT computer_device_quantity INTO v_computer_device_quantity FROM inventory WHERE id
= 1001;

    v_inventaire_total_quantity := v_software_quantity + v_computer_quantity +
v_computer_device_quantity;
    DBMS_OUTPUT.PUT_LINE('Total quantity in the inventory : ' ||
v_inventaire_total_quantity);
END;
/
EXECUTE total_quantity_inventory;
```

(petit rappel : nous avons fait cet INSERT INTO dans le fichier “02\_db\_cergy.sql”)

```
INSERT INTO inventory (id, software_quantity, computer_quantity, computer_device_quantity)
VALUES (1001, 100, 50, 70);
```

```
SQL> EXECUTE total_quantity_inventory;
Total quantity in the inventory : 220

PL/SQL procedure successfully completed.
```

Le résultat 220 est correct.

## Fonctions

Quelques fonctions pour nous aider à retourner des ID ou des noms selon ce que l'on recherche.

On a pris l'exemple de deux fonctions, qui permettent de

- retourner le nom d'un software en fonction de l'ID
- retourner l'ID d'un device en fonction du nom

```

/*****
-- fonction qui retourne le SOFTWARE_NAME grace à l'id du Software.
CREATE OR REPLACE FUNCTION get_SoftwareName_by_SoftwareId(p_SoftwareId IN NUMBER)
RETURN VARCHAR2
AS
    v_name VARCHAR2(20);
BEGIN
    SELECT name INTO v_name FROM software WHERE id = p_SoftwareId;
    RETURN v_name;
END;
/

--TEST
DECLARE
    result VARCHAR2(15);
BEGIN
    result := get_SoftwareName_by_SoftwareId(4001);
    DBMS_OUTPUT.PUT_LINE('Le software numéro ''4001'' est : ' || result);
END;
/
*****/
```

```

SQL> DECLARE
2     result VARCHAR2(15);
3 BEGIN
4     result := get_SoftwareName_by_SoftwareId(4001);
5     DBMS_OUTPUT.PUT_LINE('Le software numéro ''4001'' est : ' || result);
6 END;
7 /
Le software numéro '4001' est : Windows

PL/SQL procedure successfully completed.
```

```

/*****
-- fonction qui retourne le SOFTWARE_NAME grace à l'id du Software.
CREATE OR REPLACE FUNCTION get_DeviceId_by_DeviceName(p_DeviceName VARCHAR2)
RETURN NUMBER
AS
    v_DeviceId VARCHAR2(20);
BEGIN
    SELECT id INTO v_DeviceId FROM computer_device_cergy WHERE name = p_DeviceName;
    RETURN v_DeviceId;
END;
/

--TEST
DECLARE
    result NUMBER;
BEGIN
    result := get_DeviceId_by_DeviceName('clavier');
    DBMS_OUTPUT.PUT_LINE('Le device nommé 'clavier' a l''ID suivant : ' || result);
END;
/
*****/

```

```

SQL> DECLARE
2     result NUMBER;
3 BEGIN
4     result := get_DeviceId_by_DeviceName('clavier');
5     DBMS_OUTPUT.PUT_LINE('Le device nommé 'clavier' a l''ID suivant : ' || result);
6 END;
7 /
Le device nommé 'clavier' a l'ID suivant : 3002

PL/SQL procedure successfully completed.

```

## Indexes

Des index ont été implémentés pour que les requêtes soient plus performantes: à la place de parcourir toute la table, on a un bloc d'info précis de ce qu'on recherche, pour ne lire qu'uniquement ce bloc.

```
-- Create an index on the ticket_date column
drop index index_mv_tickets;
CREATE INDEX index_mv_tickets ON materialized_view_tickets (ticket_date);
-- Get all tickets from a certain ticket_date
SELECT * FROM materialized_view_tickets
WHERE ticket_date >= TO_DATE('2023-04-01', 'YYYY-MM-DD');
```

Index created.

```
SQL> -- Get all tickets from a certain ticket_date
SQL> SELECT * FROM materialized_view_tickets
  2  WHERE ticket_date >= TO_DATE('2023-04-01', 'YYYY-MM-DD');
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5004	1004			4003	01-APR-23	besoin du logiciel office	CERGY
5004	1004			4003	01-APR-23	besoin du logiciel office	PAU
5005	1005	2005			01-APR-23	ordinateur en panne	CERGY
5005	1005			4004	01-APR-23	besoin du logiciel adobe	PAU

```
-- Get all tickets related to computer
drop index index_mv_tickets;
CREATE INDEX index_mv_tickets ON materialized_view_tickets (computer_id);
SELECT * FROM materialized_view_tickets
WHERE computer_id IS NOT NULL;
```

Index created.

```
SQL> SELECT * FROM materialized_view_tickets
  2  WHERE computer_id IS NOT NULL;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5003	1003	2003			31-MAR-23	ordinateur ne marche pas	PAU
5005	1005	2005			01-APR-23	ordinateur en panne	CERGY

```
-- Get all tickets related to computer device
drop index index_mv_tickets;
CREATE INDEX index_mv_tickets ON materialized_view_tickets (computer_device_id);
SELECT * FROM materialized_view_tickets
WHERE computer_device_id IS NOT NULL;
```

Index created.

```
SQL> SELECT * FROM materialized_view_tickets
2 WHERE computer_device_id IS NOT NULL;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5001	1001		3001		31-MAR-23	souris ne marche pas	CERGY
5003	1003		3001		31-MAR-23	souris ne marche pas	CERGY
5002	1002		3002		31-MAR-23	clavier ne marche pas	CERGY
5002	1002		3004		31-MAR-23	cable réseau ne marche pas	PAU
5001	1001		3005		31-MAR-23	imprimante ne marche pas	PAU

```
-- Get all tickets related to software
drop index index_mv_tickets;
CREATE INDEX index_mv_tickets ON materialized_view_tickets (software_id);
SELECT * FROM materialized_view_tickets
WHERE software_id IS NOT NULL;
```

Index created.

```
SQL> SELECT * FROM materialized_view_tickets
2 WHERE software_id IS NOT NULL;
```

ID	USER_ID	COMPUTER_ID	COMPUTER_DEVICE_ID	SOFTWARE_ID	TICKET_DA	DESCRIPTION	CITY
5004	1004			4003	01-APR-23	besoin du logiciel office	CERGY
5004	1004			4003	01-APR-23	besoin du logiciel office	PAU
5005	1005			4004	01-APR-23	besoin du logiciel adobe	PAU

SQL>

## Plan de requêtes

Des query plans ont été implémentés pour voir les exécutions et informations liées aux requêtes comme leur coût.

```
EXPLAIN PLAN FOR
SELECT COUNT(*) FROM user_cergy;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

```
SQL> EXPLAIN PLAN FOR
  2  SELECT COUNT(*) FROM user_cergy;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

PLAN\_TABLE\_OUTPUT

Plan hash value: 3426230978

Id	Operation	Name	Rows	Cost (%CPU)	Time
0	SELECT STATEMENT		1	2 (0)	00:00:01
1	SORT AGGREGATE		1		
2	INDEX FAST FULL SCAN	SYS_C009922	5	2 (0)	00:00:01

Note

PLAN\_TABLE\_OUTPUT

- dynamic statistics used: dynamic sampling (level=2)

13 rows selected.

```
EXPLAIN PLAN FOR
SELECT COUNT(*) FROM ticket_cergy;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

```
SQL> EXPLAIN PLAN FOR
  2 SELECT COUNT(*) FROM ticket_cergy;

Explained.

SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

PLAN\_TABLE\_OUTPUT

-----

Plan hash value: 3948282300

Id	Operation	Name	Rows	Cost (%CPU)	Time
0	SELECT STATEMENT		1	2 (0)	00:00:01
1	SORT AGGREGATE		1		
2	INDEX FAST FULL SCAN	SYS_C009928	6	2 (0)	00:00:01

Note

PLAN\_TABLE\_OUTPUT

-----

- dynamic statistics used: dynamic sampling (level=2)

13 rows selected.

```

EXPLAIN PLAN FOR
SELECT firstname, lastname
FROM user_cergy
WHERE id = 1;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

```

```

SQL> EXPLAIN PLAN FOR
  2  SELECT firstname, lastname
  3  FROM user_cergy
  4  WHERE id = 1;

```

Explained.

```
SQL> SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

PLAN\_TABLE\_OUTPUT

-----

Plan hash value: 3020935787

Id		Operation	Name	Rows	Bytes	Cost (%CPU)	Time	
0		SELECT STATEMENT		1	31	1 (0)	00:00:01	
1		TABLE ACCESS BY INDEX ROWID	USER_CERGY	1	31	1 (0)	00:00:01	
* 2		INDEX UNIQUE SCAN	SYS_C009922	1		1 (0)	00:00:01	

Predicate Information (identified by operation id):

PLAN\_TABLE\_OUTPUT

-----

2 - access("ID"=1)

14 rows selected.



## Triggers

Des triggers ont été implémentés lorsque la table ticket est utilisée. Ce sont des blocs de code qui s'exécutent lorsqu'une requête spécifique a été demandée. Les codes des triggers sont exécutés dans le même fichier que la création des tables. Ainsi, les changements des tables prendra en compte les mises à jour effectués par les triggers.

```
DROP TRIGGER computer_device_inventory;
CREATE OR REPLACE TRIGGER computer_device_inventory
AFTER INSERT OR DELETE ON computer_device_cergy
FOR EACH ROW
BEGIN
    IF inserting THEN
        UPDATE inventory SET computer_device_quantity = computer_device_quantity + 1;
    ELSE
        UPDATE inventory SET computer_device_quantity = computer_device_quantity - 1;
    END IF;
END;
/
-- Pour afficher les erreurs du trigger
SHOW ERRORS TRIGGER computer_device_inventory;

-- Commandes pour tester le trigger
select * from inventory;
delete from computer_device_cergy where id = 3050;
INSERT INTO computer_device_cergy (id, name) VALUES (3050, 'projecteur');
select * from inventory;
delete from computer_device_cergy where id = 3050;
```

```
SQL> select * from inventory;

      ID SOFTWARE_QUANTITY COMPUTER_QUANTITY COMPUTER_DEVICE_QUANTITY
-----
    1001             100             50                70

SQL> delete from computer_device_cergy where id = 3050;
0 rows deleted.

SQL> INSERT INTO computer_device_cergy (id, name) VALUES (3050, 'projecteur');
1 row created.

SQL> select * from inventory;

      ID SOFTWARE_QUANTITY COMPUTER_QUANTITY COMPUTER_DEVICE_QUANTITY
-----
    1001             100             50                71
```

### Remarque du trigger

Nous pouvons constater que notre **trigger** marche très bien :  
Au début, l'**inventaire** contient **70 computer\_device\_quantity**.

Et lorsqu'on ajoute un **computer\_device** à la table de **Cergy**, alors **computer\_device\_quantity** **augmente de 1**, soit il vaut **71** maintenant.

```

DROP TRIGGER computer_inventory_trigger;
CREATE OR REPLACE TRIGGER computer_inventory_trigger
AFTER INSERT OR DELETE ON computer_cergy
FOR EACH ROW
BEGIN
    IF inserting THEN
        UPDATE inventory SET computer_quantity = computer_quantity + 1;
    ELSE
        UPDATE inventory SET computer_quantity = computer_quantity - 1;
        -- IF computer_quantity <= 0 THEN
        --     UPDATE inventory SET computer_quantity = 0;
        --     RAISE_APPLICATION_ERROR(-20001, 'Computer is out of stock!');
        -- END IF;
    END IF;
END;
/
-- Pour afficher les erreurs du trigger
SHOW ERRORS TRIGGER computer_inventory_trigger;

-- Commandes pour tester le trigger
select * from inventory;
delete from computer_cergy where id = 2050;
INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2050,
3001, 4001, 1001);
select * from inventory;
delete from computer_cergy where id = 2050;

```

```

SQL> select * from inventory;

```

ID	SOFTWARE_QUANTITY	COMPUTER_QUANTITY	COMPUTER_DEVICE_QUANTITY
1001	100	50	70

```

SQL> delete from computer_cergy where id = 2050;

0 rows deleted.

SQL> INSERT INTO computer_cergy (id, computer_device_id, software_id, user_id) VALUES (2050, 3001, 4001, 1001);

1 row created.

SQL> select * from inventory;

```

ID	SOFTWARE_QUANTITY	COMPUTER_QUANTITY	COMPUTER_DEVICE_QUANTITY
1001	100	51	70

### Remarque du trigger

De même que pour l'exemple précédent, on voit que l'état 1 est de 50 computer\_quantity, et 51 pour l'état suivant. Explication : à chaque **ajout** d'un **computer**, l'**inventory** se met à jour et ajoute "+1" ou "-1" selon l'**ajout** ou la **suppression** d'ordinateur.

```

/*****
CREATE OR REPLACE TRIGGER software_inventory_trigger
AFTER INSERT OR DELETE ON software
FOR EACH ROW
BEGIN
    IF inserting THEN
        UPDATE inventory
        SET software_quantity = software_quantity + 1;
    ELSE
        UPDATE inventory
        SET software_quantity = software_quantity - 1;
        -- IF inventory.software_quantity <= 0 THEN
        --     UPDATE inventory SET software_quantity = 0;
        --     RAISE_APPLICATION_ERROR(-20001, 'Software is out of stock!');
        -- END IF;
    END IF;
END;
/
-- Commandes pour tester le trigger
INSERT INTO software (id, name) VALUES (4050, 'Linux');
select * from inventory;
delete from software where id = 4050;
select * from inventory;
*****/

```

```

SQL> select * from inventory;

      ID SOFTWARE_QUANTITY COMPUTER_QUANTITY COMPUTER_DEVICE_QUANTITY
-----
    1001              101              50              70

SQL> delete from software where id = 4050;

1 row deleted.

SQL> select * from inventory;

      ID SOFTWARE_QUANTITY COMPUTER_QUANTITY COMPUTER_DEVICE_QUANTITY
-----
    1001              100              50              70

```

### Remarque du trigger

Pour changer un peu, ici on teste la suppression.  
La mise à jour s'est bien faite : on passe de 101 à 100.

## Séquences

Des séquences pour créer un identifiant unique pour créer un utilisateur ou insérer au hasard un matériel informatique (clavier, souris...) dans la table computer\_device\_cergy.

```

/*****
-- Génération d'utilisateurs au hasard dans la table user_cergy
DROP SEQUENCE user_cergy_seq;
CREATE SEQUENCE user_cergy_seq START WITH 1006 INCREMENT BY 1 MAXVALUE 1015 NOCYCLE
NOCACHE;
BEGIN
  FOR i IN 1006..1015 LOOP
    DELETE FROM computer_device_cergy WHERE id = i;
    INSERT INTO user_cergy (id, firstname, lastname, email) VALUES
(user_cergy_seq.NEXTVAL, 'UTILISATEURS', 'SEQUENCES', 'UTILISATEURS@SEQUENCES.fr');
  END LOOP;
END;
/
--Vérification
SELECT * FROM user_cergy;
*****/
```

```
SQL> --Vérification
SQL> SELECT * FROM user_cergy;
```

ID	FIRSTNAME	LASTNAME	EMAIL
1001	Jean	Dupont	jean.dupont@cergy.fr
1002	Marie	Martin	marie.martin@cergy.fr
1003	Luc	Dubois	luc.dubois@cergy.fr
1004	Sophie	Lecomte	sophie.lecomte@cergy.fr
1005	Pierre	Morel	pierre.morel@cergy.fr
1006	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1007	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1008	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1009	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1010	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1011	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1012	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1013	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1014	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr
1015	UTILISATEURS	SEQUENCES	UTILISATEURS@SEQUENCES.fr

15 rows selected.

```

/*****
-- Insertion d'outils au hasard dans la table computer_device_cergy
DROP SEQUENCE seq_tool_cergy;
CREATE SEQUENCE seq_tool_cergy START WITH 3006 INCREMENT BY 1 MAXVALUE 3015 NOCYCLE
NOCACHE;
DECLARE
    tool_name VARCHAR2(50);
BEGIN
    FOR i IN 3006..3015 LOOP
        DELETE FROM computer_device_cergy WHERE id = i;
        CASE round(dbms_random.value(1, 5))
            WHEN 1 THEN tool_name := 'souris';
            WHEN 2 THEN tool_name := 'clavier';
            WHEN 3 THEN tool_name := 'écran';
            WHEN 4 THEN tool_name := 'cable';
            WHEN 5 THEN tool_name := 'imprimante';
        END CASE;
        INSERT INTO computer_device_cergy (id, name) VALUES (seq_tool_cergy.NEXTVAL,
tool_name);
    END LOOP;
END;
/
--Vérification
SELECT * FROM computer_device_cergy;

```

```

SQL> --Vérification
SQL> SELECT * FROM computer_device_cergy;

```

ID	NAME
3001	souris
3002	clavier
3003	écran
3004	cable
3005	imprimante
3007	clavier
3006	clavier
3008	cable
3009	cable
3010	cable
3011	écran
ID	NAME
3012	cable
3013	souris
3014	cable
3015	clavier

15 rows selected.

## Remarque de la séquence

Sachant que c'est une séquence, et que l'on a choisi de lui attribuer au hasard des données, alors elle ne va pas générer la même donnée à chaque tour de séquence.

Exemple, si l'on relance le programme, ça donne des résultats différents. Voici plusieurs résultats différents :

```
SQL> --Vérification
SQL> SELECT * FROM computer_device_cergy;
```

ID	NAME
3001	souris
3002	clavier
3003	écran
3004	cable
3005	imprimante
3006	cable
3007	clavier
3008	clavier
3009	imprimante
3010	écran
3011	écran

ID	NAME
3012	imprimante
3013	clavier
3014	souris
3015	imprimante

15 rows selected.

```
SQL> --Vérification
SQL> SELECT * FROM computer_device_cergy;
```

ID	NAME
3001	souris
3002	clavier
3003	écran
3004	cable
3005	imprimante
3006	clavier
3007	cable
3008	imprimante
3009	clavier
3010	écran
3011	écran

ID	NAME
3012	écran
3013	clavier
3014	cable
3015	cable

15 rows selected.

```
SQL> --Vérification
SQL> SELECT * FROM computer_device_cergy;
```

ID	NAME
3001	souris
3002	clavier
3003	écran
3004	cable
3005	imprimante
3006	écran
3007	écran
3008	clavier
3009	clavier
3010	souris
3011	souris

ID	NAME
3012	écran
3013	écran
3014	cable
3015	clavier

15 rows selected.

```
SQL> --Vérification
SQL> SELECT * FROM computer_device_cergy;
```

ID	NAME
3001	souris
3002	clavier
3003	écran
3004	cable
3005	imprimante
3006	cable
3007	imprimante
3008	écran
3009	écran
3010	clavier
3011	imprimante

ID	NAME
3012	clavier
3013	clavier
3014	écran
3015	cable

15 rows selected.

## **Notes**

### **Lien du code GITHUB**

# **GitHub - PROJET GLPI**

## **Exécution du code**

Il ne faut pas tout exécuter tout le code d'un coup dans le système.

Dans les sections attribuées à un site il faut se connecter à la tablespace dédiée.

Sinon, il va y avoir des erreurs car les tables sur les 2 sites portent le même nom.

Ci-dessous les étapes d'exécution du code :

1. se connecter sur une fenêtre SQL PLUS avec l'utilisateur SYSTEM, lancer db\_system.sql
2. se connecter sur une fenêtre SQL PLUS avec l'utilisateur user\_cergy, lancer db\_cergy.sql
3. se connecter sur une fenêtre SQL PLUS avec l'utilisateur user\_pau, lancer db\_pau.sql
4. dans user\_cergy, lancer materialized\_view\_cergy.sql

Ensuite nous pouvons lancer les procédures, index ...

Dans user\_cergy, lancer une par une les procédures du fichier 05\_01\_procedures\_display.sql.

Lancer une par une les procédures du fichier 05\_02\_procedures\_create.sql, lancer les query plans, les séquences, les indexes, les triggers.