

Nom: Akrouchi Rihab
Matricule: 202031050231
groupe: Master 2 IL groupe 2

TP SDED 1

TP 1

Introduction:

Ce TP vise à créer une base de données pour la gestion des trajets de véhicules, des pannes et des localisations géographiques. À travers des commandes SQL et des blocs PL/SQL, nous allons établir des tables et des relations, tout en insérant des données simulées pour modéliser efficacement ce système.

1.Création de compte avec privileges :

```
Session altered.  
  
SQL> create user master identified by psw ;  
  
User created.  
  
SQL> grant all privileges to master ;  
  
Grant succeeded.
```

2.Les modèles relationnels :

Marque(CodeMarque, Marque)

Modèle(CodeMod, NomMod, **CodeMarque***)

Véhicule(NumVehicule, **CodeMod***)

Chauffeur(CodeChauffeur, NomChauffeur)

Trajet(CodeTrajet, DateTrajet, DureeTrajet, KilometreParcouru, CarburantConsomme, **NumVehicule***, **CodeChauffeur***)

Panne(CodePanne, NiveauGravite, **CodeTrajet***, **CodeTypePanne***)

TypePanne(CodeTypePanne, TypePanne)

PointDépart(CodeVille, Localisation, GPS)

PointArrivée(CodeVille, Localisation, GPS)

3.Les tables :

D'abord on se connecte avec le compte Master :

```
SQL> connect master/psw  
Connected.  
SQL>
```

Création des tables :

```
SQL Plus
SQL> CREATE TABLE Marque (
2     CodeMarque NUMBER PRIMARY KEY,
3     Marque CHAR(20)
4 );

Table created.

SQL> CREATE TABLE Chauffeur (
2     CodeChauffeur NUMBER PRIMARY KEY,
3     NomChauffeur CHAR(30)
4 );

Table created.

SQL> CREATE TABLE TypePanne (
2     CodeTypePanne NUMBER PRIMARY KEY,
3     TypePanne CHAR(20)
4 );

Table created.

SQL> CREATE TABLE PointDepart (
2     CodeVille NUMBER PRIMARY KEY,
3     Localisation CHAR(50),
4     GPS CHAR(50)
5 );

Table created.

SQL>
SQL> CREATE TABLE PointArrivee (
2     CodeVille NUMBER PRIMARY KEY,
3     Localisation CHAR(50),
4     GPS CHAR(50)
5 );
```

```
SQL Plus
Table created.

SQL> CREATE TABLE Modele (
2     CodeMod NUMBER PRIMARY KEY,
3     NomMod CHAR(10),
4     CodeMarque NUMBER,
5     FOREIGN KEY (CodeMarque) REFERENCES Marque(CodeMarque)
6 );

Table created.

SQL> CREATE TABLE Vehicule (
2     NumVehicule NUMBER PRIMARY KEY,
3     CodeModele NUMBER,
4     FOREIGN KEY (CodeModele) REFERENCES Modele(CodeMod)
5 );

Table created.

SQL> CREATE TABLE Trajet (
2     CodeTrajet NUMBER PRIMARY KEY,
3     DateTrajet DATE,
4     DureeTrajet NUMBER,
5     KilometreParcoursu NUMBER,
6     CarburantConsomme NUMBER,
7     NumVehicule NUMBER,
8     CodeChauffeur NUMBER,
9     FOREIGN KEY (NumVehicule) REFERENCES Vehicule(NumVehicule),
10    FOREIGN KEY (CodeChauffeur) REFERENCES Chauffeur(CodeChauffeur)
11 );

Table created.
```

```
SQL Plus
SQL> CREATE TABLE Trajet (
2   CodeTrajet NUMBER PRIMARY KEY,
3   DateTrajet DATE,
4   DureeTrajet NUMBER,
5   KilometreParcoursu NUMBER,
6   CarburantConsomme NUMBER,
7   NumVehicule NUMBER,
8   CodeChauffeur NUMBER,
9   FOREIGN KEY (NumVehicule) REFERENCES Vehicule(NumVehicule),
10  FOREIGN KEY (CodeChauffeur) REFERENCES Chauffeur(CodeChauffeur)
11 );

Table created.

SQL> CREATE TABLE Panne (
2   CodePanne NUMBER PRIMARY KEY,
3   NiveauGravite NUMBER,
4   CodeTrajet NUMBER,
5   CodeTypePanne NUMBER,
6   FOREIGN KEY (CodeTrajet) REFERENCES Trajet(CodeTrajet),
7   FOREIGN KEY (CodeTypePanne) REFERENCES TypePanne(CodeTypePanne)
8 );

Table created.

SQL> create table pays (
2   codepays number primary key ,
3   nompays varchar (50)
4 );

Table created.

SQL> |
```

4. pl/sql :

Remplir les tables avec pl/sql :

```
SQL Plus
SQL> DECLARE
2   CodeMarque NUMBER;
3   Marque CHAR(20);
4 BEGIN
5   FOR CodeMarque IN 1..38 LOOP
6     SELECT dbms_random.string('A', 10) INTO Marque FROM dual;
7     INSERT INTO Marque (CodeMarque, Marque) VALUES (CodeMarque, Marque);
8   END LOOP;
9   COMMIT;
10 END;
11 /

PL/SQL procedure successfully completed.

SQL> DECLARE
2   CodeMod NUMBER;
3   NomMod CHAR(10);
4   CodeMarque NUMBER;
5 BEGIN
6   FOR CodeMod IN 1..120 LOOP
7     SELECT dbms_random.string('U', 8) INTO NomMod FROM dual;
8     SELECT FLOOR(dbms_random.value(1, 38)) INTO CodeMarque FROM dual;
9     INSERT INTO Modele (CodeMod, NomMod, CodeMarque) VALUES (CodeMod, NomMod, CodeMarque);
10  END LOOP;
11  COMMIT;
12 END;
13 /

PL/SQL procedure successfully completed.

SQL> DECLARE
2   NumVehicule NUMBER;
3   CodeModele NUMBER;
4 BEGIN
5   FOR NumVehicule IN 1..600 LOOP
6     SELECT FLOOR(dbms_random.value(1, 120)) INTO CodeModele FROM dual;
7     INSERT INTO Vehicule (NumVehicule, CodeModele) VALUES (NumVehicule, CodeModele);
8   END LOOP;
9   COMMIT;
10 END;
11 /

PL/SQL procedure successfully completed.

SQL> |
```

Les résultats :

Affichage des tables avec l'instruction select :

```
SQL> select * from vehicule ;
```

NUMVEHICULE	CODEMODELE
1	78
2	40
3	102
4	50
5	101
6	37
7	3
8	27
9	87
10	46
11	23

NUMVEHICULE	CODEMODELE
12	67
13	72
14	28
15	26
16	35
17	118
18	1
19	64
20	44
21	11
22	113

NUMVEHICULE	CODEMODELE
23	83

```
SQL> select * from modele ;
```

CODENOD	NOMMOD	CODENARQUE
1	QEEZESTM	25
2	TRHPLEST	33
3	VMPXVLHL	5
4	JCJDCKSF	27
5	FEQWIMKB	23
6	AMQOHEU	19
7	UYQHIMBW	22
8	OMKFLMYQ	30
9	EXFVPVNY	21
10	KKQMYBNA	20
11	AHAVLUEP	19

CODENOD	NOMMOD	CODENARQUE
12	CYZFUIFK	27
13	SRMESMEY	30
14	QCHLESTAU	3
15	PIVKIXPC	28
16	XPXPQPKYS	28
17	HEGWMOP	30

```
SQL> select * from marque ;
```

CODENARQUE	MARQUE
1	QWZl8ahncq
2	lhkfhz3d0H
3	hF4eF1dCt
4	z4p111LbyX
5	45YzyjprS
6	KC0p3xLd
7	VHjrytBwWG
8	ngcm1E3jg
9	mFpRdgaT4
10	X3piPRKZVM
11	ENHvrtLSie

```

SQL Plus
3 HFzIcFLDCL
4 ZwPIITLhyX
5 iSVZyYjPbL
6 KCQzhjclLD
7 YHjrytBwNG
8 xgcMIEJxg
9 nVFRMgpaTu
10 XJpiPhMZVM
11 EHVfvtLSie

CODEMARQUE MARQUE
12 PQhtUIMQs
13 nYGCnWomD
14 UIQSLjJnQs
15 lwiaLrSDyz
16 zAwbDtuITh
17 vFbyxbzuHE
18 bzMetZQYgn
19 mMeKfJCLrf
20 QlwMRiXIVs
21 wuCKssZQAV
22 ghvLRaPeJz

CODEMARQUE MARQUE
23 GQdEqshJAO
24 RMTiFFMyg
25 F15xnLJqjb
26 iFyymhnut
27 GqbldeMBVQ
28 GUMcxyWQs
29 UQNZKodas
30 yMEZAlMFzI
31 zQhCDfXXkN
32 UHAHSTFBwM
33 wLWZqpeDZhw

CODEMARQUE MARQUE
34 CxCxJXvkwX
35 buNtARFJba
36 FJgTApXCLL
37 caUnKefasO
38 QhwoQMcsJR

38 rows selected.

```

TP2

Étape 1 : code pl/sql

```

SQL> DECLARE
2   CodeTypePanne NUMBER;
3   TypePanne VARCHAR2(100);
4 BEGIN
5   FOR i IN 1..20 LOOP
6     CodeTypePanne := i;
7     TypePanne := 'TypePanne_' || dbms_random.string('U', 8);
8     INSERT INTO TypePanne (CodeTypePanne, TypePanne) VALUES (CodeTypePanne, TypePanne);
9   END LOOP;
10  COMMIT;
11 END;
12 /

PL/SQL procedure successfully completed.

```

Création de la table localisation et la remplir avec pl/sql

```

SQL> CREATE TABLE Localisation (
2   CodeLoc NUMBER PRIMARY KEY,          -- Code unique pour chaque localisation
3   GPS VARCHAR2(20)                     -- Coordonnées GPS de la localisation
4 );

Table created.

SQL>
SQL> DECLARE
2   CodeLoc NUMBER;
3   GPS VARCHAR2(20);
4 BEGIN
5   FOR i IN 1..127344 LOOP
6     CodeLoc := i;
7     GPS := 'GPS ' || i;
8     INSERT INTO Localisation (CodeLoc, GPS)
9     VALUES (CodeLoc, GPS);
10  END LOOP;
11  COMMIT;
12 END;
13 /

PL/SQL procedure successfully completed.

```

```

SQL> DECLARE
2   NumVehicule NUMBER;
3   CodeModele NUMBER;
4 BEGIN
5   -- Remplacer 1276 par la valeur maximale actuelle + 1
6   FOR i IN 1277..2552 LOOP -- Insérer 1276 nouveaux enregistrements
7     NumVehicule := i;
8     CodeModele := FLOOR(DBMS_RANDOM.VALUE(1, 120)); -- Modèle aléatoire entre 1 et 120
9
10    -- Insertion dans la table Vehicule
11    INSERT INTO Vehicule (NumVehicule, CodeModele)
12    VALUES (NumVehicule, CodeModele);
13  END LOOP;
14  COMMIT;
15 END;
16 /

PL/SQL procedure successfully completed.
SQL>

```

Affichage du résultat pour la table vehicule :

```
SQL> select * from vehicule ;
```

NUMVEHICULE	CODEMODELE
1	78
2	40
3	102
4	50
5	101
6	37
7	3
8	27
9	87
10	46
11	23

NUMVEHICULE	CODEMODELE
12	67
13	72
14	28
15	26
16	35
17	118
18	1
19	64
20	44
21	11
22	113

Code sql pour remplir les tables :

Pour la table trajet :

```

SQL> DECLARE
2   CodeTrajet NUMBER;
3   DateTrajet DATE;
4   DureeTrajet INTERVAL DAY TO SECOND;
5   KilometrageParcours NUMBER;
6   CarburantConsomme NUMBER;
7   CodeVehicule NUMBER;
8   CodeDepart NUMBER;
9   CodeArrivee NUMBER;
10  CodeChauffeur NUMBER;
11 BEGIN
12  FOR i IN 1..2502545 LOOP
13    CodeTrajet := i;
14
15    DateTrajet := TO_DATE('2023-01-01','YYYY-MM-DD') + TRUNC(dbms_random.value(0, 365));
16
17    DureeTrajet := NUMTODSINTERVAL(TRUNC(dbms_random.value(1, 100)), 'HOUR');
18    KilometrageParcours := dbms_random.value(10, 1000);
19    CarburantConsomme := dbms_random.value(10, 500);
20    CodeVehicule := TRUNC(dbms_random.value(1, 600));
21    CodeDepart := TRUNC(dbms_random.value(1, 127344));
22    CodeArrivee := TRUNC(dbms_random.value(1, 127344));
23    CodeChauffeur := TRUNC(dbms_random.value(1, 2003));
24
25    INSERT INTO Trajet (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcours, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur)
26    VALUES (CodeTrajet, DateTrajet, DureeTrajet, KilometrageParcours, CarburantConsomme, CodeVehicule, CodeDepart, CodeArrivee, CodeChauffeur);
27  END LOOP;
28  COMMIT;
29 END;
30 /

PL/SQL procedure successfully completed.

```

Pour la table typepanne:

```

SQL> DECLARE
2   CodeTypePanne NUMBER;
3   TypePanne VARCHAR2(100);
4   BEGIN
5   FOR i IN 1..20 LOOP
6       CodeTypePanne := i;
7       TypePanne := 'TypePanne_' || dbms_random.string('U', 8);
8       INSERT INTO TypePanne (CodeTypePanne, TypePanne) VALUES (CodeTypePanne, TypePanne);
9   END LOOP;
10  COMMIT;
11  END;
12  /

PL/SQL procedure successfully completed.

```

table panne:

```

SQL> DECLARE
2   CodePanne NUMBER;
3   NiveauGravite VARCHAR2(50);
4   Observation VARCHAR2(255);
5   CodeTypePanne NUMBER;
6   CodeTrajet NUMBER;
7   BEGIN
8   FOR i IN 1..160022 LOOP
9       CodePanne := i;
10      NiveauGravite := 'Gravite_' || dbms_random.string('U', 4);
11      Observation := 'Observation_' || dbms_random.string('U', 8);
12      CodeTypePanne := TRUNC(dbms_random.value(1, 20));      CodeTrajet := TRUNC(dbms_random.value(1, 2502545));
13
14      INSERT INTO Panne (CodePanne, NiveauGravite, Observation, CodeTypePanne, CodeTrajet)
15      VALUES (CodePanne, NiveauGravite, Observation, CodeTypePanne, CodeTrajet);
16  END LOOP;
17  COMMIT;
18  END;
19  /

PL/SQL procedure successfully completed.

```

Pour la table ville :

```

SQL> DECLARE
2   CodeVille NUMBER;
3   NomVille VARCHAR2(100);
4   CodePays NUMBER;
5   BEGIN
6   FOR i IN 1..9076 LOOP
7       CodeVille := i;
8       NomVille := 'Ville_' || dbms_random.string('U', 8);
9
10      -- Utilisez TRUNC pour obtenir un entier entre 1 et 25
11      CodePays := TRUNC(dbms_random.value(1, 26)); -- Notez que dbms_random.value(1, 26) génère des nombres entre 1 et 25
12
13      INSERT INTO Ville (CodeVille, NomVille, CodePays)
14      VALUES (CodeVille, NomVille, CodePays);
15  END LOOP;
16  COMMIT;
17  END;
18  /

PL/SQL procedure successfully completed.

```

Localisation:

```

SQL> DECLARE
2   GPS NUMBER;
3   CodeVille NUMBER;
4   BEGIN
5   FOR i IN 1..127344 LOOP
6       GPS := i;
7
8       -- Utilisez TRUNC pour générer un entier aléatoire pour CodeVille
9       CodeVille := TRUNC(dbms_random.value(1, 9077)); -- dbms_random.value(1, 9077) génère des nombres entre 1 et 9076 inclus
10
11      INSERT INTO Localisation (GPS, CodeVille)
12      VALUES (GPS, CodeVille);
13  END LOOP;
14  COMMIT;
15  END;
16  /

PL/SQL procedure successfully completed.

```

Pour la table chauffeur :

```

SQL> DECLARE
2   CodeChauffeur NUMBER;
3   NomChauffeur VARCHAR2(100);
4   CodeVille NUMBER;
5 BEGIN
6   FOR i IN 1..2003 LOOP
7     CodeChauffeur := i;
8     NomChauffeur := 'Chauffeur_' || dbms_random.string('U', 8);
9     CodeVille := TRUNC(dbms_random.value(1, 9077)); -- Génère des entiers entre 1 et 9076 inclus
10    INSERT INTO Chauffeur (CodeChauffeur, NomChauffeur, CodeVille)
11    VALUES (CodeChauffeur, NomChauffeur, CodeVille);
12  END LOOP;
13  COMMIT;
14 END;
15 /

PL/SQL procedure successfully completed.

```

Pour la table pays :

```

SQL> DECLARE
2   CodePays NUMBER;
3   NomPays VARCHAR2(100);
4 BEGIN
5   FOR i IN 1..25 LOOP
6     CodePays := i;
7     NomPays := 'Pays_' || dbms_random.string('U', 8);
8     INSERT INTO Pays (CodePays, NomPays) VALUES (CodePays, NomPays);
9   END LOOP;
10  COMMIT;
11 END;
12 /

PL/SQL procedure successfully completed.

```

Étape 2: Requêtes demandées

a. Nombre de trajets effectués et kilométrage parcouru entre le 01/01/2023 et le 30/01/2024

```

SQL> SELECT COUNT(*) AS nombre_trajets,
2         SUM(KilometrageParcours) AS total_kilometrage,
3         p.NomPays
4 FROM Trajet t
5 JOIN Localisation l_depart ON t.CodeDepart = l_depart.GPS
6 JOIN Ville v_depart ON l_depart.CodeVille = v_depart.CodeVille
7 JOIN Pays p ON v_depart.CodePays = p.CodePays
8 WHERE t.DateTrajet BETWEEN TO_DATE('2023-01-01', 'YYYY-MM-DD')
9        AND TO_DATE('2024-01-30', 'YYYY-MM-DD')
10 GROUP BY p.NomPays;

```

Affichage de la table trajet et le résultat :

NOMBRE_TRAJETS	TOTAL_KILOMETRAGE

NOMPAYS	

101137	51054633
Pays_OKONLAEJ	
102361	51830538.2
Pays_GDMLIGLW	

Le nombre de trajets est élevé et le kilométrage parcouru conséquent, cela indique une bonne utilisation de la flotte de véhicules

b. Nombre de trajets sans pannes par année


```
SQL> SELECT EXTRACT(YEAR FROM t.DateTrajet) AS annee,
2         COUNT(*) AS nombre_trajets_sans_pannes
3 FROM Trajet t
4 LEFT JOIN Panne p ON t.CodeTrajet = p.CodeTrajet
5 WHERE p.CodeTrajet IS NULL -- Pas de panne associée au trajet
6 GROUP BY EXTRACT(YEAR FROM t.DateTrajet)
7 ORDER BY annee;
```

Affichage du résultat:

ANNEE	NOMBRE_TRAJETS_SANS_PANNES
2023	2347512

Ce grand nombre de trajets sans pannes est un signe positif qui reflète une bonne maintenance des véhicules et des conditions optimales de conduite.

```
Elapsed: 00:00:00.13
Execution Plan
-----
Plan hash value: 1356978527

-----
| Id | Operation | Name | Rows | Bytes | Cost |
|----|-----|-----|-----|-----|-----|
| 0 | SELECT STATEMENT | | 1 | 152 | 7 |
| (15) | | | | | |
| 1 | HASH GROUP BY | | 1 | 152 | 7 |
| (15) | | | | | |
| 2 | NESTED LOOPS | | 1 | 152 | 6 |
| (0) | | | | | |
| 3 | NESTED LOOPS | | 1 | 152 | 6 |
| (0) | | | | | |
| 4 | NESTED LOOPS | | 1 | 87 | 5 |
| (0) | | | | | |

-----
Predicate Information (identified by operation id):
-----

6 - filter("T"."DATETRAJET">=TO_DATE(' 2023-01-01 00:00:00', 'syyy-mm-dd
hh24:mi:ss') AND "T"."DATETRAJET"<=TO_DATE(' 2024-01-30 00:00:00',
'syyy-mm-dd
hh24:mi:ss'))
8 - access("T"."CODEDEPART"="L_DEPART"."GPS")
10 - access("L_DEPART"."CODEVILLE"="V_DEPART"."CODEVILLE")
11 - access("V_DEPART"."CODEPAYS"="P"."CODEPAYS")

Note
-----
- dynamic statistics used: dynamic sampling (level=2)
- this is an adaptive plan
```

c. Modèle de véhicule ayant fait le plus grand kilométrage sans panne :

```
SQL>
SQL>
SQL> SELECT
2     v.NumVehicule,
3     SUM(t.KilometrageParcoursu) AS KilometrageTotal
4 FROM
5     Vehicule v,
6     Trajet t
7 WHERE
8     v.NumVehicule = t.CodeVehicule
9     AND t.CodeTrajet NOT IN (SELECT CodeTrajet FROM Panne)
10 GROUP BY
11     v.NumVehicule
12 HAVING
13     SUM(t.KilometrageParcoursu) = (
14     SELECT MAX(KilometrageTotal)
15     FROM (
16     SELECT SUM(t1.KilometrageParcoursu) AS KilometrageTotal
17     FROM Trajet t1
18     WHERE t1.CodeTrajet NOT IN (SELECT CodeTrajet FROM Panne)
19     GROUP BY t1.CodeVehicule
20     )
21 );
NUMVEHICULE KILOMETRAGETOTAL
-----
562 8695367

Ecoulu : 00 :00 :02.08
Plan d'exécution
-----
Plan hash value: 3653919417

-----
| Id | Operation | Name | Rows | Bytes | TempSpc | Cost | (%CP
U) | Time |
|----|-----|-----|-----|-----|-----|-----|
| 0 | SELECT STATEMENT | | 1 | 152 | 0 | 7 | 00:00:00.13 |
| 1 | HASH GROUP BY | | 1 | 152 | 0 | 7 | 00:00:00.13 |
| 2 | NESTED LOOPS | | 1 | 152 | 0 | 6 | 00:00:00.13 |
| 3 | NESTED LOOPS | | 1 | 152 | 0 | 6 | 00:00:00.13 |
| 4 | NESTED LOOPS | | 1 | 87 | 0 | 5 | 00:00:00.13 |
```

Le véhicule ayant fait le plus grand kilométrage sans panne est 562 avec un kilométrage 8695367 .

Conclusion de l'analyse :

Ces résultats apportent des informations importantes sur la gestion de la flotte de véhicules. En identifiant les véhicules les plus performants et en analysant les trajets sans pannes, on peut non seulement optimiser les coûts liés à la maintenance, mais aussi améliorer la gestion des trajets et maximiser l'utilisation des véhicules.

Conclusion:

Pour résumer, ce TP nous a permis d'acquérir des compétences pratiques en SQL et PL/SQL en développant une base de données fonctionnelle. Cette expérience nous a préparés à gérer des informations complexes et à appliquer nos connaissances dans des scénarios réels de gestion des données.