



Ministère de l'Enseignement supérieur et de la Recherche scientifique
Université des Sciences et de la Technologie Houari Boumediene



Faculté d'Informatique
Master Ingénierie de Logiciels

Rapport de projet SQL3 BDA

Nom et Prénom : Elaidat Mohamed Redha

Matricule : 212132055546

Groupe : 02

Nom et Prénom : Benzaoucha Youcef

Matricule : 202037029967

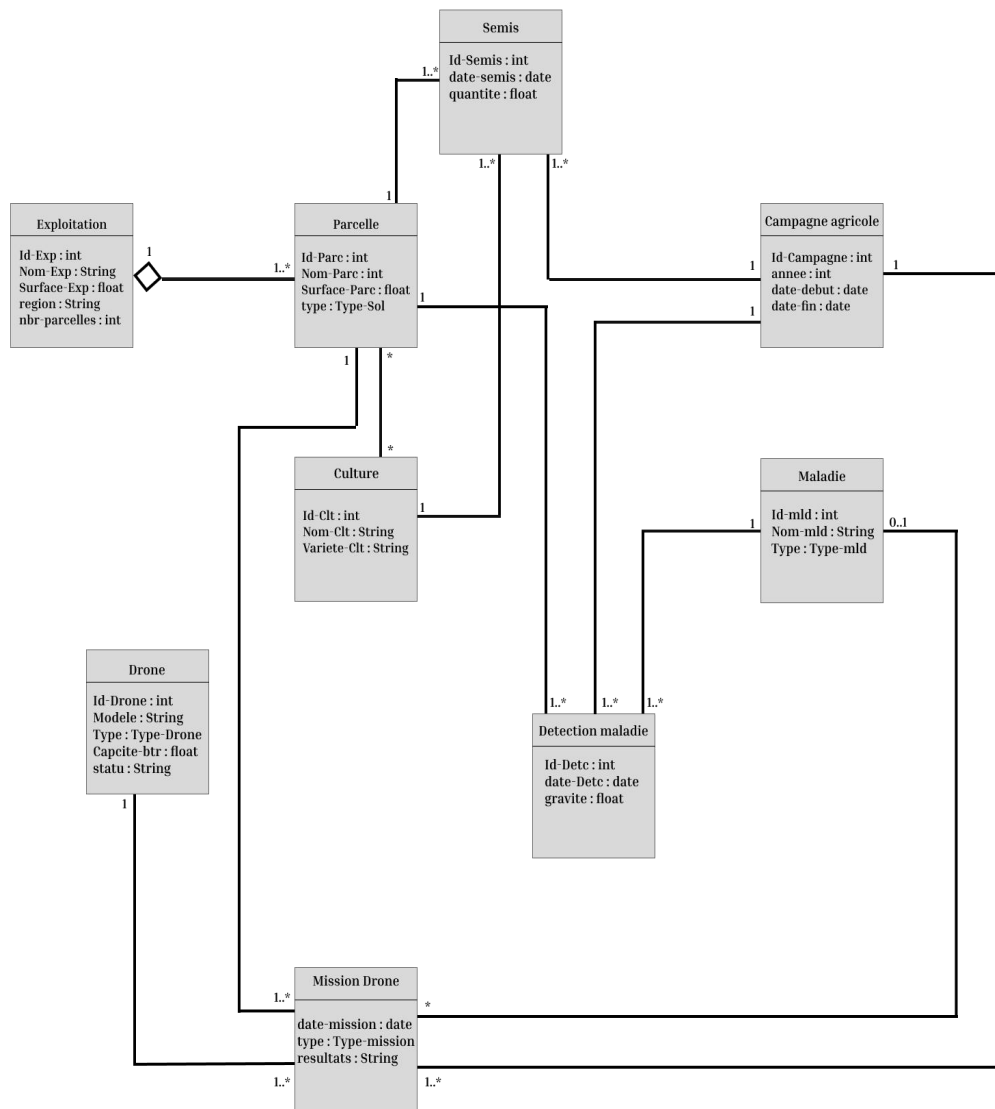
Groupe : 01

Sommaire

A. Modélisation orientée objet	01
1. UML (Diagramme de classes)	
B. Création des TableSpaces et utilisateurs	04
1. Création des TableSpace	
2. Création de l'utilisateur SQL3	
3. Attribution des privileges	
C. Langage des définition de données (LDD)	06
1. Définition des types dérivés	
2. Définition des méthodes	
3. Création des tables	
D. Langage de Manipulation de données (LMD)	15
1. Insertion des données	
E. Langage d'Interrogation de données (LID)	18
1. Liste des exploitations et de leurs parcelles	
2. Calcul du taux de maladies par parcelle et par campagne	
3. Liste des missions de drones de type « traitement »	
4. Historiques des cultures semées par parcelle et par exploitation	
5. Liste des drones disponibles pour une mission de surveillance	
6. Recherche de l'année avec le plus grand nombre de maladies détectées	


A. Modélisation orientée objet :

Diagramme de classes :




B. Création des TableSpaces et utilisateur :

1. Création des TableSpaces :




```
CREATE TABLESPACE SQL3_TBS  
DATAFILE 'C:\tbs\SQL3_TBS.dat'  
SIZE 100M  
AUTOEXTEND ON  
ONLINE;
```

a. SQL3_TBS (TableSpace de données) :



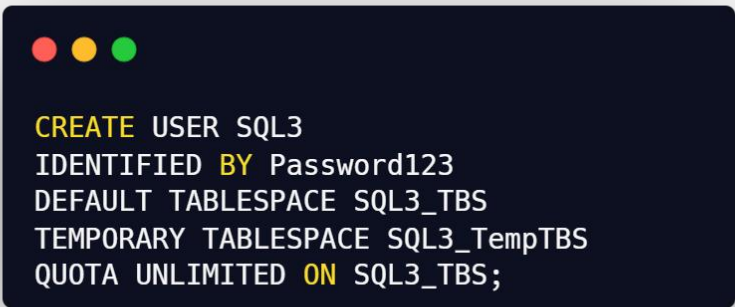
```
CREATE TABLESPACE SQL3_TempTBS  
DATAFILE 'C:\tbs\SQL3_TempTBS.dat'  
SIZE 100M  
AUTOEXTEND ON  
ONLINE;
```

b. SQL3_TempTBS (TableSpace temporaire) :



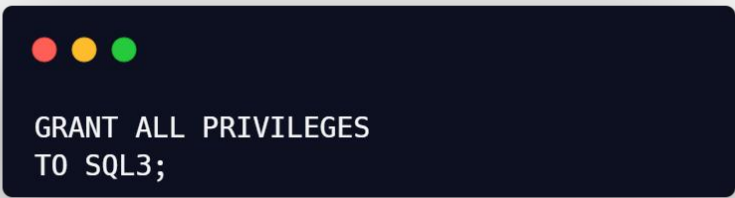
```
CREATE TABLESPACE SQL3_TempTBS  
DATAFILE 'C:\tbs\SQL3_TempTBS.dat'  
SIZE 100M  
AUTOEXTEND ON  
ONLINE;
```

2. Création de l'utilisateur SQL3 :



```
CREATE USER SQL3  
IDENTIFIED BY Password123  
DEFAULT TABLESPACE SQL3_TBS  
TEMPORARY TABLESPACE SQL3_TempTBS  
QUOTA UNLIMITED ON SQL3_TBS;
```

3. Attribution des privilèges :



```
GRANT ALL PRIVILEGES  
TO SQL3;
```

C. Langage de définition de données :

1. Définition des types dérivés :

1.1. Définition des Types Objets et Collections Référentielles :

```
CREATE TYPE TExploitation;  
/  
CREATE TYPE TParcelle;  
/  
CREATE TYPE TCulture;  
/  
CREATE TYPE TCampagne;  
/  
CREATE TYPE TSemis;  
/  
CREATE TYPE TMaladie;  
/  
CREATE TYPE TDetection_Maladie;  
/
```

```
CREATE TYPE TDrone;  
/  
CREATE TYPE TMission_Drone;  
/  
CREATE TYPE tset_ref_Parcelle AS TABLE  
OF ref TParcelle  
/  
CREATE TYPE tset_ref_Semis AS TABLE OF  
ref TSemis  
/  
CREATE TYPE tset_ref_Mission_Drone AS  
TABLE OF ref TMission_Drone  
/  
CREATE TYPE tset_ref_Detection_Maladie AS  
TABLE OF ref TDetection_Maladie  
/
```

1.2. Création des Types Objets Représentant les Entités Métier :

```
CREATE OR Replace TYPE TExploitation AS
OBJECT(
    id_exploitation      CHAR(6),
    nom_exploitation     VARCHAR2(50),
    superficie_exploitation NUMBER(38),
    region               VARCHAR2(50),
    nbr_parcelles         NUMBER(38),
    Exploitation_Parcelle
tset_ref_Parcelle
);
/
CREATE OR Replace TYPE TParcelle AS
OBJECT (
    id_parcelle          CHAR(4),
    nom_parcelle         VARCHAR(50),
    superficie_parcelle  INT,
    type_sol             VARCHAR(50),
    Parcelle_Exploitation ref
TExploitation,
Parcelle_Semis
tset_ref_Semis,
Parcelle_Maladie
tset_ref_Detection_Maladie,
Parcelle_Mission
tset_ref_Mission_Drone
);
/
```

```
CREATE OR Replace TYPE TCulture AS
OBJECT (
    id_culture           CHAR(6),
    nom_culture          VARCHAR(50),
    variete_culture      VARCHAR(50)
);
/
CREATE OR Replace TYPE TCampagne AS
OBJECT (
    id_campagne          CHAR(6),
    annee                INT,
    date_debut           DATE,
    date_fin             DATE,
    Campagne_Semis
tset_ref_Semis,
Campagne_Maladie
tset_ref_Detection_Maladie,
Campagne_Mission
tset_ref_Mission_Drone
);
/
CREATE OR Replace TYPE TSemis AS OBJECT
(
    id_semis             CHAR(4),
    date_semis           DATE,
    quantite_semis       INT,
    semis_parcelle       ref
TParcelle,
semis_culture           ref
TCulture,
semis_campagne          ref
TCampagne
);
/
```

```
CREATE OR Replace TYPE TMaladie AS
OBJECT (
    id_maladie           CHAR(6),
    nom_maladie          VARCHAR(50),
    type_maladie         VARCHAR(50)
);
/
CREATE OR Replace TYPE TDetection_Maladie
AS OBJECT (
    id_detection         CHAR(4),
    date_detection       DATE,
    gravite              VARCHAR(10),
    maladie_parcelle     ref
TParcelle,
maladie_campagne       ref
TCampagne,
maladie_maladie        ref TMaladie
);
/
```

```
CREATE OR Replace TYPE TDrone AS OBJECT
(
    id_drone             CHAR(6),
    modele               VARCHAR(50),
    type_drone           VARCHAR(50),
    capacite_batterie    INT,
    statut_drone         VARCHAR(20),
    Drone_Mission
tset_ref_Mission_Drone
);
/
CREATE OR Replace TYPE TMission_Drone AS
OBJECT (
    id_mission           CHAR(6),
    date_mission         DATE,
    type_mission         VARCHAR(50),
    resultats            VARCHAR(255),
    mission_drone        ref TDrone,
    mission_parcelle     ref
TParcelle,
mission_campagne        ref
TCampagne,
mission_maladie         ref TMaladie
);
/
```

2. Définition des méthodes :

2.1. Surface Totale des Parcelles par Exploitation

```
ALTER TYPE TExploitation ADD MEMBER
FUNCTION Surface_Totale_Parcelles RETURN
NUMBER CASCADE;
/
CREATE OR REPLACE TYPE BODY TExploitation
AS
    MEMBER FUNCTION
    Surface_Totale_Parcelles RETURN NUMBER
    IS
        total NUMBER := 0;
        p      TParcelle;
    BEGIN
        IF Exploitation_Parcelle IS NOT NULL
        THEN
            FOR i IN 1 ..
            Exploitation_Parcelle.COUNT LOOP
                SELECT
                Deref(Exploitation_Parcelle(i)) INTO p
                FROM DUAL;
                total := total +
                p.superficie_parcelle;
            END LOOP;
        END IF;
        RETURN total;
    END;
END;
/
```


2.2. Cultures Semées pendant une Campagne Agricole :

```
CREATE TYPE tset_ref_Culture AS TABLE OF
ref TCulture;
/
ALTER TYPE TExploitation ADD MEMBER
FUNCTION Cultures_Semees_Campagne(id_camp
CHAR) RETURN tset_ref_Culture CASCADE;
/
```

```

MEMBER FUNCTION
Cultures_Semees_Campagne(id_camp CHAR)
RETURN tset_ref_Culture IS
    cultures tset_ref_Culture :=
tset_ref_Culture();
    p      TParcelle;
    s      TSemis;
    c      TCampagne;
BEGIN
    IF Exploitation_Parcelle IS NOT NULL
THEN
        FOR i IN 1 ..
Exploitation_Parcelle.COUNT LOOP
            SELECT
DEREF(Exploitation_Parcelle(i)) INTO p
FROM DUAL;

            IF p.Parcelle_Semis IS NOT NULL
THEN
                FOR j IN 1 ..
p.Parcelle_Semis.COUNT LOOP
                    SELECT
DEREF(p.Parcelle_Semis(j)) INTO s FROM
DUAL;

                    IF s.semis_campagne IS NOT
NULL THEN
                        SELECT
DEREF(s.semis_campagne) INTO c FROM
DUAL;

                        IF c.id_campagne = id_camp
THEN
                            cultures.EXTEND;
                            cultures(cultures.COUNT)
:= s.semis_culture;
                            END IF;
                            END IF;

                        END LOOP;
                    END IF;

                END LOOP;
            END IF;

        RETURN cultures;
    END;
END;
/

```

2.3. Cultures sur Parcelle durant la Campagne

```
CREATE TYPE tset_ref_Culture AS TABLE OF
ref TCulture;
/
ALTER TYPE TParcelle ADD MEMBER FUNCTION
Cultures_Presentes_Campagne(id_camp CHAR)
RETURN tset_ref_Culture CASCADE;
/
```

```
CREATE OR REPLACE TYPE BODY TParcelle
AS

    MEMBER FUNCTION
    Cultures_Presentes_Campagne(id_camp
    CHAR) RETURN tset_ref_Culture IS
        cultures tset_ref_Culture :=
        tset_ref_Culture(); -- initialiser le
        tableau vide
        s          TSemis;
    BEGIN
        -- Vérification si la parcelle a
        des semis
        IF Parcelle_Semis IS NOT NULL THEN
            FOR i IN 1 ..
            Parcelle_Semis.COUNT LOOP
                SELECT Deref(Parcelle_Semis(i))
                INTO s FROM DUAL;

                -- Vérifier si le semis
                appartient à la campagne donnée
                IF s.semis_campagne.id_campagne
                = id_camp THEN
                    cultures.EXTEND;
                    cultures(cultures.COUNT) :=
                    s.semis_culture; -- Ajouter la culture
                    à la liste
                END IF;

            END LOOP;
            END IF;
            RETURN cultures;
        END;

    END;
/
```

2.4. Maladies Détectées avec Gravité Forte sur Parcelle

```
CREATE TYPE tset_ref_Maladie AS TABLE OF
ref TMaladie;
/
ALTER TYPE TParcelle ADD MEMBER FUNCTION
Maladies_Fortes RETURN tset_ref_Maladie
CASCADE;
/
```

```
CREATE OR REPLACE TYPE BODY TParcelle AS

MEMBER FUNCTION Maladies_Fortes RETURN
tset_ref_Maladie IS
    maladies tset_ref_Maladie :=
tset_ref_Maladie();
    d          TDetection_Maladie;
BEGIN
    IF parcelle_maladie IS NOT NULL THEN
        FOR i IN 1 ..
parcelle_maladie.COUNT LOOP
            SELECT
DEREF(parcelle_maladie(i)) INTO d FROM
DUAL;

            IF d.gravite = 'forte' THEN
                maladies.EXTEND;
                maladies(maladies.COUNT) :=
d.maladie_maladie;
            END IF;
        END LOOP;
    END IF;
    RETURN maladies;
END;
/
```

2.5. Missions par Type de Drone

```
CREATE TYPE tset_ref_Mission_Drone AS
TABLE OF ref TMission_Drone;
/
ALTER TYPE TDrone ADD MEMBER FUNCTION
Missions_Par_Type(type_mission_in
VARCHAR2) RETURN tset_ref_Mission_Drone
CASCADE;
/
```

```
CREATE OR REPLACE TYPE BODY TDrone AS

  MEMBER FUNCTION
  Missions_Par_Type(type_mission_in
VARCHAR2) RETURN tset_ref_Mission_Drone
IS
  missions tset_ref_Mission_Drone :=
tset_ref_Mission_Drone();
  m_ref    REF TMission_Drone;
  m        TMission_Drone;
  BEGIN
    IF Drone_Mission IS NOT NULL THEN
      FOR i IN 1 .. Drone_Mission.COUNT
  LOOP
        m_ref := Drone_Mission(i);

        SELECT Deref(m_ref) INTO m FROM
  DUAL;

        IF m.type_mission =
type_mission_in THEN
          missions.EXTEND;
          missions(missions.COUNT) :=
m_ref;
        END IF;

      END LOOP;
    END IF;
    RETURN missions;
  END;

END;
/
```

3. Création des tables :

```
CREATE TABLE Exploitation OF
TExploitation( CONSTRAINT
pk_id_exploitation PRIMARY KEY
(id_exploitation))

    NESTED TABLE Exploitation_Parcelle
STORE AS NESTED_Exploitation_Parcelle
;
CREATE TABLE Parcelle OF TParcelle(
CONSTRAINT pk_id_parcelle PRIMARY KEY
(id_parcelle),

CONSTRAINT fk_Exploitation FOREIGN
KEY(Parcelle_Exploitation) REFERENCES
Exploitation,

CONSTRAINT ck_type_sol CHECK (type_sol
IN ('argileux', 'sableux',
'limoneux', 'calcaire', 'humifère', 'tourb
eux'))))

NESTED TABLE Parcelle_Semis  STORE AS
NESTED_Parcelle_Semis,

NESTED TABLE Parcelle_Maladie STORE AS
NESTED_Parcelle_Maladie,

NESTED TABLE Parcelle_Mission STORE AS
NESTED_Parcelle_Mission
;
```

```
CREATE TABLE Culture OF TCulture(
CONSTRAINT PK_id_culture PRIMARY KEY
(id_culture))
;
CREATE TABLE Campagne OF TCampagne(
CONSTRAINT pk_id_campagne PRIMARY KEY
(id_campagne),

CONSTRAINT ck_date CHECK (date_debut <=
date_fin))

NESTED TABLE Campagne_Semis  STORE AS
NESTED_Campagne_Semis,

NESTED TABLE Campagne_Maladie STORE AS
NESTED_Campagne_Maladie,

NESTED TABLE Campagne_Mission STORE AS
NESTED_Campagne_Mission
;
CREATE TABLE Semis OF TSemis(
CONSTRAINT pk_id_semis PRIMARY
KEY(id_semis),

CONSTRAINT fk_parcelle FOREIGN
KEY(semis_parcelle) REFERENCES
Parcelle,

CONSTRAINT fk_culture  FOREIGN
KEY(semis_culture) REFERENCES Culture,

CONSTRAINT fk_campagne FOREIGN
KEY(semis_campagne) REFERENCES
Campagne)
;
```



```

CREATE TABLE Maladie OF
TMaladie(CONSTRAINT pk_id_maladie
PRIMARY KEY(id_maladie),

CONSTRAINT ck_type_maladie
CHECK(type_maladie IN
('fongique','bactérienne','virale','par
asitique','physiologique')))
;
CREATE TABLE DETECTIONMALADIE OF
TDetection_Maladie ( CONSTRAINT
pk_id_detection PRIMARY
KEY(id_detection),

CONSTRAINT ck_gravite
CHECK(gravite IN ('faible', 'moyenne',
'forte'))),

CONSTRAINT
fk_detection_parcelle FOREIGN
KEY(maladie_parcelle) REFERENCES
Parcelle,

CONSTRAINT
fk_detection_maladie FOREIGN
KEY(maladie_maladie) REFERENCES
Maladie,

CONSTRAINT
fk_detection_campagne FOREIGN
KEY(maladie_campagne) REFERENCES
Campagne)
;

```

```

CREATE TABLE Drone OF TDrone (
CONSTRAINT pk_id_drone PRIMARY
KEY(id_drone),

CONSTRAINT ck_type_drone CHECK
(type_drone IN ('multirotor','iles
fixes','hybride','à voilure
tournante','autonome')),

CONSTRAINT ck_statut_drone
CHECK(statut_drone IN ('Disponible',
'En Maintenance', 'En Mission'))
NESTED
TABLE Drone_Mission STORE AS
NESTED_Drone_Mission
;
CREATE TABLE Mission_Drone OF
TMission_Drone ( CONSTRAINT
pk_id_mission PRIMARY KEY(id_mission),

CONSTRAINT ck_type_mission
CHECK (type_mission IN
('surveillance','traitement','cartograp
hie','analyse thermique')),

CONSTRAINT fk_mission_parcelle
FOREIGN KEY(mission_parcelle)
REFERENCES Parcelle,

CONSTRAINT fk_mission_maladie
FOREIGN KEY(mission_maladie)
REFERENCES Maladie,

CONSTRAINT fk_mission_campagne
FOREIGN KEY(mission_campagne)
REFERENCES Campagne,

CONSTRAINT fk_mission_drone
FOREIGN KEY(mission_drone)
REFERENCES Drone)
;

```

D. Langage de manipulation de données :

1. Listes des Exploitations :

```
SQL> select id_exploitation from Exploitation ;

ID_EXP
-----
EXP001
EXP002
EXP003
EXP004
EXP005
EXP006
EXP007
EXP008
EXP009
EXP010
```

2. Listes des parcelles :

```
ID_P
----
P001
P002
P003
P004
P005
P006
P007
P008
P009
P010
P011
```

```
ID_P
----
P012
P013
P014
P015
P016
P017
P018
P019
P020
P021
P022

ID_P
----
P023
P024

24 rows selected.
```

3. Listes des cultures :

```
SQL> select id_culture from Culture;

ID_CUL
-----
CUL001
CUL003
CUL004
CUL005
CUL006
CUL007
CUL008
CUL009
CUL010

9 rows selected.
```

4. Listes des campagnes :

```
SQL> select id_campagne from Campagne;

ID_CAM
-----
CA2022
CA2023
CA2024
```

5. Listes des semis :

```
SQL> select id_semis from Semis;

ID_S
----
S001
S002
S003
S004
S005
S006
S007
S008
S009
S010
S011

ID_S
----
S012
```


6. Listes des maladies :

```
SQL> select id_maladie from Maladie;

ID_MAL
-----
M001
M002
M003
M004
M005
M006
M007
M008
M009
M010
M011

11 rows selected.
```

7. Listes des détections maladie :

```
SQL> select id_detection from DETECTIONMALADIE;

ID_D
----
D001
D002
D003
D004
D005
D006
D007

7 rows selected.
```

8. Listes des drones :

```
SQL> select id_drone from Drone;

ID_DRO
-----
DR001
DR002
DR003
DR004
DR005
DR006

6 rows selected.
```

9. Listes des missions des drones :

```
SQL> select id_mission from Mission_Drone;

ID_MIS
-----
MS001
MS002
MS003
MS004
MS005
MS006
MS007
MS008

8 rows selected.
```

E. Langage d'interrogation de données :

1. Association des exploitations à leurs parcelles:

```
SELECT e.id_exploitation, p.id_parcelle
FROM Exploitation e,
TABLE(e.Exploitation_Parcelle) ep,
Parcelle p
WHERE VALUE(ep) = REF(p)
ORDER BY e.id_exploitation,
p.id_parcelle;
```

```
SQL> SELECT e.id_exploitation, p.id_parcelle
  2 FROM Exploitation e, TABLE(e.Exploitation_Parcelle) ep, Parcelle p
  3 WHERE VALUE(ep) = REF(p)
  4 ORDER BY e.id_exploitation, p.id_parcelle;

ID_EXP ID_P
-----
EXP001 P001
EXP001 P002
EXP001 P003
EXP002 P004
EXP002 P005
EXP003 P006
EXP003 P007
EXP003 P008
EXP003 P009
EXP004 P010
EXP004 P011

ID_EXP ID_P
-----
EXP005 P012
EXP005 P013
EXP006 P014
EXP006 P015
EXP006 P016
EXP007 P017
EXP008 P018
EXP008 P019
EXP008 P020
EXP009 P021
EXP009 P022

ID_EXP ID_P
-----
EXP010 P023
EXP010 P024

24 rows selected.
```

2. Taux de maladies par parcelle et par campagne Agricole :

```
SQL> SELECT
  2   p.id_parcelle,
  3   p.nom_parcelle,
  4   (SELECT COUNT(*) FROM TABLE(p.Parcelle_Semis)) AS nombre_semis,
  5   (SELECT COUNT(*) FROM TABLE(p.Parcelle_Maladie)) AS nombre_detections,
  6   (SELECT COUNT(*) FROM TABLE(p.Parcelle_Mission)) AS nombre_missions,
  7   CASE
  8     WHEN (SELECT COUNT(*) FROM TABLE(p.Parcelle_Semis)) > 0 THEN
  9       (SELECT COUNT(*) FROM TABLE(p.Parcelle_Maladie)) /
10       (SELECT COUNT(*) FROM TABLE(p.Parcelle_Semis)) * 100
11     ELSE 0
12   END AS taux_maladies_par_semis,
13   CASE
14     WHEN (SELECT COUNT(*) FROM TABLE(p.Parcelle_Mission)) > 0 THEN
15       (SELECT COUNT(*) FROM TABLE(p.Parcelle_Maladie)) /
16       (SELECT COUNT(*) FROM TABLE(p.Parcelle_Mission)) * 100
17     ELSE 0
18   END AS taux_maladies_par_mission
19 FROM
20   Parcelle p
21 ORDER BY
22   p.id_parcelle;
```

ID_P	NOM_PARCELLE	NOMBRE_SEMIS	NOMBRE_DETECTIONS	NOMBRE_MISSIONS	TAUX_MALADIES_PAR_SEMIS	TAUX_MALADIES_PAR_MISSION
P001	Nord-1	1	1	2	100	50
P002	Sud-2	1	1	0	100	0
P003	Est-3	0	0	0	0	0
P004	Ouest-1	1	0	0	0	0
P005	Central	1	1	1	100	100
P006	Vergers-1	1	0	0	0	0
P007	Vergers-2	1	0	0	0	0
P008	Plaine	0	0	0	0	0
P009	Hauteur	0	0	0	0	0
P010	Test-1	1	1	1	100	100
P011	Oasis	0	0	0	0	0
P012	Bio-Est	1	0	0	0	0
P013	Bio-Ouest	1	0	1	0	0
P014	Sahara-1	1	1	1	100	100
P015	Sahara-2	1	1	1	100	100
P016	Palmeraie	1	1	1	100	100
P017	Montagne	0	0	0	0	0
P018	El-Feth-1	0	0	0	0	0
P019	El-Feth-2	0	0	0	0	0
P020	El-Feth-3	0	0	0	0	0
P021	Vergers-Nord	0	0	0	0	0
P022	Vergers-Sud	0	0	0	0	0
P023	Terra-1	0	0	0	0	0
P024	Terra-2	0	0	0	0	0

24 rows selected.

3. Missions de drones de type "traitement":

```
SQL> SELECT
  2   d.id_drone,
  3   d.type_drone,
  4   d.statut_drone,
  5   (SELECT COUNT(*) FROM TABLE(d.Drone_Mission)) AS nombre_missions
  6 FROM
  7   Drone d
  8 ORDER BY
  9   d.id_drone;
```

ID_DRO	TYPE_DRONE	STATUT_DRONE	NOMBRE_MISSIONS
DR001	multirotor	Disponible	1
DR002	iles fixes	En Maintenance	1
DR003	hybride	En Mission	2
DR004	iles fixes	Disponible	1
DR005	multirotor	Disponible	2
DR006	iles fixes	En Mission	1

6 rows selected.

4. Historique des cultures semées par exploitation et par parcelle

```
SQL> SELECT
2   e.ID_EXPLOITATION,
3   e.NOM_EXPLOITATION,
4   e.REGION,
5   p.ID_PARCELLE,
6   c.ANNEE AS ANNEE_CAMPAGNE,
7   cul.ID_CULTURE,
8   cul.NOM_CULTURE,
9   cul.VARIETE_CULTURE,
10  s.DATE_SEMIS,
11  s.QUANTITE_SEMIS
12 FROM
13   Exploitation e,
14   TABLE(e.EXPLOITATION_PARCELLE) ep,
15   PARCELLE p,
16   semis s,
17   culture cul,
18   campagne c
19 WHERE
20   VALUE(ep) = REF(p)
21   AND s.SEMIS_PARCELLE = REF(p)
22   AND s.SEMIS_CULTURE = REF(cul)
23   AND s.SEMIS_CAMPAGNE = REF(c)
24 ORDER BY
25   e.ID_EXPLOITATION,
26   p.ID_PARCELLE,
27   c.ANNEE,
28   s.DATE_SEMIS;
```

ID_EXP	NOM_EXPLOITATION	REGION	ID_P	ANNEE_CAMPAGNE	ID_CUL	NOM_CULTURE	VARIETE_CULTURE	DATE_SEMIS	QUANTITE_SEMIS
EXP001	Ferme des Códres	Kabylie	P001	2023	CUL001	Blé	Blé dur	2023-02-15	120
EXP002	Oasis Verte	Biskra	P004	2023	CUL003	Tomate	Roma	2023-03-05	80
EXP002	Oasis Verte	Biskra	P005	2024	CUL004	Pomme de terre	Spunta	2024-02-18	110
EXP003	Domaine El-Kheir	Oran	P006	2023	CUL007	Olive	Chemlal	2023-02-25	90
EXP003	Domaine El-Kheir	Oran	P007	2024	CUL001	Blé	Blé dur	2024-02-20	130
EXP004	Agro-Sud	Ghardaïa	P010	2024	CUL006	Carotte	Nantaise	2024-03-01	95
EXP005	Bio Champs	Tlemcen	P012	2024	CUL005	Ail	Violet de Provence	2024-03-10	70
EXP005	Bio Champs	Tlemcen	P013	2024	CUL008	Dattier	Deglet Mour	2024-03-15	85
EXP006	Ferme Saharienne	Adrar	P014	2024	CUL009	Laitue	Batavia	2024-03-20	90
EXP006	Ferme Saharienne	Adrar	P015	2024	CUL010	Pois chiche	Kabuli	2024-03-22	75
EXP006	Ferme Saharienne	Adrar	P016	2024	CUL003	Tomate	Roma	2024-03-25	95

11 rows selected.

5. Drones disponibles pour les missions de surveillance

```
SQL> SELECT
2   d.id_drone,
3   d.modele,
4   d.type_drone,
5   d.capacite_batterie,
6   d.statut_drone
7 FROM
8   Drone d
9 WHERE
10  d.statut_drone = 'Disponible' AND
11  NOT EXISTS (
12    SELECT 1
13    FROM Mission_Drone md
14    WHERE md.mission_drone = REF(d) AND md.type_mission = 'surveillance' AND md.date_mission > SYSDATE
15  )
16 ORDER BY
17   d.id_drone;
```

ID_DRO	MODELE	TYPE_DRONE	CAPACITE_BATTERIE	STATUT_DRONE
DR001	DJI Agras T30	multirotor	10000	Disponible
DR004	eBee SQ	iles fixes	9000	Disponible
DR005	Matrice 300 RTK	multirotor	11000	Disponible

6. Année avec le plus grand nombre de maladies détectées

```
SQL> SELECT
  2     annee,
  3     nombre_de_detections
  4 FROM (
  5     SELECT
  6         EXTRACT(YEAR FROM d.date_detection) AS annee,
  7         COUNT(*) AS nombre_de_detections
  8     FROM
  9         DETECTIONMALADIE d
 10     GROUP BY
 11         EXTRACT(YEAR FROM d.date_detection)
 12     ORDER BY
 13         nombre_de_detections DESC
 14 )
 15 WHERE ROWNUM = 1;
```

ANNEE	NOMBRE_DE_DETECTIONS
2024	7

