

Base de données avec Oracle DBA sur les championnats de moto

Pierre AYOUB et Maël ROUXEL

13 janvier 2019



**INSTITUT DES SCIENCES ET
TECHNIQUES DES YVELINES**

Résumé

Oracle Database est un système de gestion de bases de données relationnelles (SGBD) utilisé dans le monde entier. Très répandu en entreprise, tant pour ses performances que sa fiabilité, nous utilisons ce SGBD afin de créer une base de données et d'y effectuer des tâches d'administration. Plusieurs possibilités offertes par Oracle DBA seront explorées dans ce projet.

Table des matières

1	Introduction	4
2	Mise en place de la base de données	5
2.1	Schéma de la base de données	5
2.2	Jeu de données	8
2.3	Manipulation des données par requêtes SQL	10
2.4	Vues	10
3	SQL Avancé	13
3.1	Triggers	13
3.2	Méta-données	17
3.3	Analyse des requêtes	18
4	Conclusion	23

1 Introduction

Notre projet modélise une base de données concernant les championnats de moto. L'objectif de cette base est de stocker des informations non pas sur une seule saison de course, mais sur plusieurs saisons. De plus, on pourra stocker au sein d'une même base plusieurs championnats différents. Beaucoup d'informations techniques sont retenues concernant les motos et les résultats des pilotes sur chaque courses, ce qui permettra d'obtenir des statistiques poussées et diversifiées. La base de données possède quelques contraintes, listées ci-dessous (pour les moins évidentes) :

- Une saison d'un championnat dure une année.
- Par saison, un pilote peut participer à plusieurs championnats.
- Pour un championnat donné, un pilote ne peut faire partit que d'une team. Dans le cas où le pilote participe à plus d'un championnat sur une saison, alors il peut faire partit de plusieurs teams différentes concourantes sur différents championnats.
- Pour un championnat donné et une team donné, un pilote ne peut utiliser qu'une moto. Dans le cas où le pilote participe à plus d'un championnat différents sur une saison, alors il peut utiliser plusieurs motos différentes sur les différents championnats.
- Chaque pilote doit être sous contrat pour pouvoir courir dans un championnat. Un contrat est un CDD liant un pilote, un modèle de moto et une team pendant un temps donné (généralement, quelques années).
- La participation d'un pilote à un course correspond à une relation entre ladite course et le contrat d'un pilote.

2 Mise en place de la base de données

Dans cette section, nous vous présenterons la mise en place de la base de données. Pour la majorité du travail ci-dessous, cela concerne le langage de requête SQL ou l'utilisation de l'utilitaire de chargement de données Oracle SQL Loader.

2.1 Schéma de la base de données

La création du schéma de la base de données consiste à créer les tables en spécifiant les attributs, leurs types, et leurs contraintes d'intégrités basiques tel que les clés primaires et étrangères, ainsi que les contraintes check.

```
1      -- Création des tables.
2
3      -- 1. Marques de moto.
4      CREATE TABLE Marque
5      (
6          Nom          VARCHAR(32) NOT NULL,
7          Annee        DATE,          -- CHECK with a trigger.
8          Nationalite  CHAR(2),       -- CHECK with a trigger.
9          PRIMARY KEY (Nom)
10     );
11
12     -- 2. Teams concourrant aux championnats.
13     CREATE TABLE Team
14     (
15         Nom          VARCHAR(32) NOT NULL,
16         Marque       VARCHAR(32) NOT NULL,
17         PRIMARY KEY (nom)
18     );
19
20     -- 3. Modèles de moto.
21     CREATE TABLE Modele_moto
22     (
23         Marque       VARCHAR(32) NOT NULL,
24         Nom          VARCHAR(32) NOT NULL,
25         Annee        DATE          NOT NULL, -- CHECK with a trigger.
26         Cylindree    FLOAT         CHECK (Cylindree > 20 AND Cylindree <
27         ↪ 2000),
28         Couple       FLOAT         CHECK (Couple > 1      AND Couple <
29         ↪ 20),
30         Puissance    FLOAT         CHECK (Puissance > 1   AND Puissance <
31         ↪ 500),
32         Poids        FLOAT         CHECK (Poids > 30     AND Poids <
33         ↪ 500),
```

```

30     Prix          NUMBER(6)    CHECK (Prix > 100 AND Prix < 500000),
31     Genre         VARCHAR(40) NOT NULL CHECK (Genre IN ('Sportive',
32     ↪ 'Cafe Racer')),
33     PRIMARY KEY (nom, annee)
34 );
35
36 -- 4. Pilotes appartenant aux teams.
37 CREATE TABLE Pilote
38 (
39     Id             NUMBER(4)    NOT NULL, -- CHECK with a trigger.
40     Nom            VARCHAR(32) NOT NULL,
41     Prenom         VARCHAR(32) NOT NULL,
42     Age            NUMBER(3)    CHECK (Age BETWEEN 10 and 100),
43     Nationalite    CHAR(2),     -- CHECK with a trigger.
44     Sexe           CHAR(1)      CHECK (Sexe IN (NULL, 'H', 'F')),
45     Numero         NUMBER(2)    CHECK (Numero BETWEEN 0 and 99),
46     PRIMARY KEY (id)
47 );
48
49 -- 5. Championnats existants.
50 CREATE TABLE Championnat
51 (
52     Nom            VARCHAR(32) NOT NULL,
53     Annee          DATE         NOT NULL, -- CHECK with a trigger.
54     PRIMARY KEY (Nom, Annee)
55 );
56
57 -- 6. Circuits sur lesquels les courses se déroulent.
58 CREATE TABLE Circuit
59 (
60     Nom            VARCHAR(32) NOT NULL,
61     Pays           CHAR(2)      NOT NULL, -- CHECK with a trigger.
62     Longueur       FLOAT        CHECK (Longueur BETWEEN 0.5 AND 20),
63     PRIMARY KEY (Nom)
64 );
65
66 -- 7. Courses appartenants aux championnats.
67 CREATE TABLE Course_vitesse
68 (
69     Championnat    VARCHAR(32) NOT NULL,
70     Annee          DATE         NOT NULL,
71     Date_course    DATE         NOT NULL, -- CHECK with trigger.
72     Circuit        VARCHAR(32) NOT NULL,
73     Nb_tours       NUMBER(2)    CHECK (Nb_tours BETWEEN 1 AND 30),
74     Duree          FLOAT        CHECK (Duree BETWEEN 1 AND 100),
75     PRIMARY KEY (Championnat, Date_course)

```

```

75 );
76
77 -- 8. Participation d'un pilote à une course.
78 CREATE TABLE Participe (
79     Id_pilote      INT          NOT NULL,
80     Championnat    VARCHAR(32) NOT NULL,
81     Date_course    DATE         NOT NULL,
82     Modele_moto    VARCHAR(32) NOT NULL,
83     Annee_moto     DATE         NOT NULL,
84     Classement     NUMBER(2)    CHECK (Classement BETWEEN 0 AND
      ↪ 30),
85     Points_gagnes  NUMBER(2)    CHECK (Points_gagnes BETWEEN 0
      ↪ AND 25),
86     Vitesse_moy    FLOAT        CHECK (Vitesse_moy BETWEEN 0 AND
      ↪ 300),
87     Meilleur_tour  FLOAT        CHECK (Meilleur_tour BETWEEN 0
      ↪ AND 400),
88     PRIMARY KEY (Id_pilote, Championnat, Date_course,
      ↪ Modele_moto, Annee_moto)
89 );
90
91 -- 9. Contrats liants un pilote, une équipe et un modèle de
   ↪ moto.
92 CREATE TABLE Contrat (
93     Id_pilote      NUMBER(4)    NOT NULL,
94     Moto_modele    VARCHAR(32) NOT NULL,
95     Moto_annee     DATE         NOT NULL,
96     Team_nom       VARCHAR(32) NOT NULL,
97     Annee_debut    DATE         NOT NULL,
98     Annee_fin      DATE         NOT NULL,
99     PRIMARY KEY (Id_pilote, Moto_modele, Moto_annee, Team_nom,
      ↪ Annee_debut)
100 );
101
102 -- Configure les clés étrangères.
103
104 ALTER TABLE Team
105     ADD FOREIGN KEY (Marque) REFERENCES Marque (Nom);
106
107 ALTER TABLE Modele_moto
108     ADD FOREIGN KEY (Marque) REFERENCES Marque (Nom);
109
110 -- Désactivé car on utilise un trigger pour ces clés,
   ↪ conformément à la consigne.
111 -- ALTER TABLE Course_vitesse

```

```

112      -- ADD FOREIGN KEY (Championnat, Annee) REFERENCES
      ↪ Championnat (Nom, Annee);
113 ALTER TABLE Course_vitesse
114      ADD FOREIGN KEY (Circuit) REFERENCES Circuit (Nom);
115
116 ALTER TABLE Participe
117      ADD FOREIGN KEY (Id_pilote) REFERENCES Pilote (Id);
118 ALTER TABLE Participe
119      ADD FOREIGN KEY (Championnat, Date_course) REFERENCES
      ↪ Course_vitesse (Championnat, Date_course);
120 ALTER TABLE Participe
121      ADD FOREIGN KEY (Modele_moto, Annee_moto) REFERENCES
      ↪ Modele_moto (Nom, Annee);
122
123 ALTER TABLE Contrat
124      ADD FOREIGN KEY (Id_pilote) REFERENCES Pilote (Id);
125 ALTER TABLE Contrat
126      ADD FOREIGN KEY (Moto_modele, Moto_annee) REFERENCES
      ↪ Modele_moto (Nom, Annee);
127 ALTER TABLE Contrat
128      ADD FOREIGN KEY (Team_nom) REFERENCES Team (Nom);

```

Listing 1 – Code SQL permettant de mettre en place la base de données

2.2 Jeu de données

Pour le chargement du jeu de données, nous avons utilisé l'utilitaire spécialisé Oracle SQL Loader. Du fait que cela ne serait pas pertinent d'inclure l'intégralité du jeu de données dans le rapport, nous allons uniquement vous présenter un exemple d'un fichier de données CSV et d'un fichier de contrôle CTL. Le fichier CSV contient les valeurs des données allant s'intégrer dans les tables créées précédemment, tandis que le fichier CTL contient des informations sur la manière dont les données doivent être chargées depuis le fichier CSV. Par exemple, des précisions sur le type de données, tel que le format de la date.

```

1  LOAD DATA
2  INFILE './Data/Participe.csv'
3  TRUNCATE
4  INTO TABLE Participe
5  FIELDS TERMINATED BY ';'
6  TRAILING NULLCOLS
7  (
8      Id_pilote,
9      Championnat,

```



```

10     Date_course DATE "YYYY-MM-DD",
11     Modele_moto,
12     Annee_moto DATE "YYYY",
13     Classement,
14     Points_gagnes,
15     Vitesse_moy,
16     Meilleur_tour
17 )

```

Listing 2 – Code SQL Loader permettant de charger des données dans une table

```

1  Id_pilote;Championnat;Date_course;Modele_moto;Annee_moto;
   ↳ Classement;Points_gagnes;Vitesse_moy;Meilleur_tour
2  1;MotoGP;2016-03-20;M1;2016;1;25;167.1;114.543
3  0;MotoGP;2016-03-20;Desmosedici GP;2013;2;20;167.0;;
4  7;MotoGP;2016-03-20;RC213V;2012;3;16;167.0;;
5  4;MotoGP;2016-03-20;M1;2016;4;13;167.0;;
6  8;MotoGP;2016-03-20;RC213V;2012;5;11;166.2;;
7  5;MotoGP;2016-03-20;GSX-RR;2014;6;10;166.1;;
8  11;MotoGP;2016-03-20;RC213V-RS;2015;14;2;164.4;;
9  6;MotoGP;2016-03-20;RC213V;2012;0;0;165.0;;
10 10;MotoGP;2016-03-20;GSX-RR;2014;18;0;;
11 4;MotoGP;2016-04-24;M1;2016;1;25;157.5;100.090
12 [...]
13 13;Superbike;2015-02-22;GSX-R1000;2014;9;7;;;92.690
14 14;Superbike;2015-02-22;ZX-10R;2015;6;10;;;92.016

```

Listing 3 – Fichier CSV contenant des données à charger (extrait)

De plus, nous avons utilisé une séquence (initialisée à partir du nombre maximum déjà inséré dans la base de données) afin de générer des entiers naturels consécutifs en tant qu'identifiant pour nos pilotes :

```

1  -- Création des séquences.
2
3  -- 1. Création et configuration de la séquence des IDs des
   ↳ pilotes.
4  DECLARE
5      max_pilote_id NUMBER := 0;
6  BEGIN
7      SELECT MAX(Id) INTO max_pilote_id FROM Pilote;
8      EXECUTE IMMEDIATE 'CREATE SEQUENCE PiloteID INCREMENT BY 1
   ↳ START WITH ' || (max_pilote_id + 1);
9  END;

```

```

10 /
11 CREATE OR REPLACE TRIGGER pilote_set_id BEFORE INSERT ON Pilote
   ↪ FOR EACH ROW
12 DECLARE
13 BEGIN
14     :NEW.Id := PiloteID.NextVal;
15 END;
16 /

```

Listing 4 – Code SQL permettant d’initialiser et d’utiliser une séquence

2.3 Manipulation des données par requêtes SQL

TODO Maël

2.4 Vues

Notre base de données contient quelques vues permettant de visualiser des scores et des statistiques calculés à partir de notre jeu de données. De tels vues se destinerait à être inclus dans un site ou service web permettant de consulter des statistiques sur les championnats de moto, avec des paramètres dynamiques tel que, par exemple, l’année des scores pour ledit championnat.

```

1  -- Création des vues.
2
3  -- 1. Liste des scores des pilotes au MotoGP de 2016.
4  CREATE VIEW MotoGP_2016_Score_pilotes AS
5      SELECT Pi.Id, Pi.Numero, Pi.Nom, Pi.Prenom,
   ↪      SUM(Pa.Points_gagnes) AS Nombre_total_de_point
6      FROM Participe Pa, Pilote Pi
7      WHERE Pa.Id_pilote = Pi.Id
8             AND Pa.Championnat LIKE 'MotoGP'
9             AND TO_CHAR(Pa.Date_course, 'YYYY') LIKE '2016'
10     GROUP BY Pi.Id, Pi.Numero, Pi.Nom, Pi.Prenom
11     ORDER BY Nombre_total_de_point DESC
12     WITH READ ONLY;
13 GRANT SELECT ON MotoGP_2016_Score_pilotes to PUBLIC;
14
15 -- 2. Liste des scores des teams au MotoGP de 2016.
16 CREATE VIEW MotoGP_2016_Score_teams AS
17     SELECT C.Team_nom, SUM(Nombre_total_de_point) AS
   ↪     Nombre_total_de_point
18     FROM Contrat C, MotoGP_2016_Score_pilotes S

```

```

19         WHERE S.Id = C.Id_pilote
20             AND TO_DATE(2016, 'YYYY') BETWEEN C.Annee_debut AND
                ↳ C.Annee_fin
21         GROUP BY C.Team_nom
22         ORDER BY Nombre_total_de_point DESC
23         WITH READ ONLY;
24 GRANT SELECT ON MotoGP_2016_Score_teams to PUBLIC;
25
26 -- 3. Liste des scores des constructeurs au MotoGP de 2016.
27 CREATE VIEW MotoGP_2016_Score_construc AS
28     SELECT T.Marque, SUM(Nombre_total_de_point) AS
                ↳ Nombre_total_de_point
29     FROM Team T, MotoGP_2016_Score_teams S
30     WHERE T.Nom = S.Team_nom
31     GROUP BY T.Marque
32     ORDER BY Nombre_total_de_point DESC
33     WITH READ ONLY;
34 GRANT SELECT ON MotoGP_2016_Score_construc to PUBLIC;
35
36 -- 4. Statistiques diverses sur les pilotes du MotoGP.
37 CREATE VIEW MotoGP_Pilote_stat AS
38     SELECT Pi.Numero, Pi.Nom, Pi.Prenom, Pi.Age, Pi.Nationalite,
                ↳ Pi.Sexe,
39             SUM(Pa.Points_gagnes) AS Total_de_points_gagnes,
40             MIN(Pa.Classement) AS Meilleur_classement,
41             MAX(Pa.Vitesse_moy) AS Vitesse_moyenne,
42             MIN(Pa.Meilleur_tour) AS Meilleur_tour
43     FROM Pilote Pi, Participe Pa
44     WHERE Pa.Id_pilote = Pi.Id
45           AND Pa.Championnat LIKE 'MotoGP'
46     GROUP BY Pi.Numero, Pi.Nom, Pi.Prenom, Pi.Age,
                ↳ Pi.Nationalite, Pi.Sexe
47     WITH READ ONLY;
48 GRANT SELECT ON MotoGP_Pilote_stat to PUBLIC;
49
50 -- 5. Nombre de victoire des pilotes au MotoGP.
51 CREATE VIEW MotoGP_Pilote_win AS
52     SELECT Pi.Numero, Pi.Nom, Pi.Prenom, COUNT(*) AS
                ↳ Nombre_de_victoire
53     FROM Pilote Pi, Participe Pa
54     WHERE Pa.Id_pilote = Pi.Id
55           AND Pa.Championnat LIKE 'MotoGP'
56           AND Pa.Classement = 1
57     GROUP BY Pi.Numero, Pi.Nom, Pi.Prenom
58     ORDER BY Nombre_de_victoire DESC
59     WITH READ ONLY;

```

```
GRANT SELECT ON MotoGP_Pilote_win to PUBLIC;
```

Listing 5 – Code SQL permettant de créer les vues de la base de données

3 SQL Avancé

Cette deuxième section nous amènera à utiliser des fonctionnalités avancées d'Oracle DBA et de PL/SQL. Nous étudierons tout d'abord quelques triggers, puis opérerons à la récupération de méta-données afin d'obtenir des informations sur notre base de données, enfin nous effectuerons une rapide analyse de requête SQL.

3.1 Triggers

Les triggers sont utilisés, dans notre projet, afin d'implémenter des contraintes d'intégrités. Pour la plupart des triggers, ces contraintes ne peuvent être exprimées avec des clauses CHECK du fait qu'il faut récupérer les valeurs de certains tuples d'une autre table et qu'il est nécessaire d'avoir un contrôle de flux en fonction des résultats. Pour un des triggers ci-dessous, nous avons ré-implémenté le fonctionnement des clés étrangères, comme demandé dans le sujet.

```
1      -- Création des procédures.
2
3      -- 1. Vérifie qu'une date est inférieure à la date du jour.
4      ↪ Nécessite un
5      -- trigger/une procédure pour utiliser SYSDATE.
6      CREATE OR REPLACE PROCEDURE date_inferior_to_current_time
7      ↪ (new_date IN DATE) IS
8      BEGIN
9          IF new_date > SYSDATE THEN
10             RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
11             ↪ because the date is greater than today !');
12         END IF;
13     END;
14 /
15
16 -- 2. Vérifie qu'une nationalité est correct. On pourrait
17 ↪ utiliser un CHECK, mais
18 -- il faudrait copier-coller les valeurs de nationalités à la
19 ↪ main, alors que
20 -- l'on peut les regrouper dans une procédure.
21 CREATE OR REPLACE PROCEDURE check_nationalite (nat IN CHAR) IS
22 BEGIN
23     IF (nat NOT IN ('FR', 'GB', 'US', 'IT', 'ES', 'JP', 'CH',
24     ↪ 'DE', 'AU', 'QT', 'PB')) THEN
25         RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
26         ↪ because the nationality is incorrect !');
27     END IF;
28 END;
```

```

22 /
23
24 -- Création des triggers.
25
26 -- 1. Vérifie l'année de création et la nationalité d'une
    ↳ marque.
27 CREATE OR REPLACE TRIGGER marque_check BEFORE INSERT OR UPDATE ON
    ↳ Marque FOR EACH ROW
28 BEGIN
29     date_inferior_to_current_time(:new.Annee);
30     check_nationalite(:new.Nationalite);
31 END;
32 /
33
34 -- 2. Vérifie l'année d'un modèle de moto.
35 CREATE OR REPLACE TRIGGER modele_moto_check BEFORE INSERT OR
    ↳ UPDATE ON Modele_moto FOR EACH ROW
36 BEGIN
37     date_inferior_to_current_time(:new.Annee);
38 END;
39 /
40
41 -- 3. Vérifie la nationalité d'un pilote.
42 CREATE OR REPLACE TRIGGER pilote_check BEFORE INSERT OR UPDATE ON
    ↳ Pilote FOR EACH ROW
43 DECLARE
44     previous_pilote_id NUMBER := 0;
45 BEGIN
46     check_nationalite(:new.Nationalite);
47 END;
48 /
49
50 -- 4. Vérifie l'année d'un championnat de moto.
51 CREATE OR REPLACE TRIGGER championnat_check BEFORE INSERT OR
    ↳ UPDATE ON Championnat FOR EACH ROW
52 BEGIN
53     date_inferior_to_current_time(:new.Annee);
54 END;
55 /
56
57 -- 5. Vérifie le pays d'un circuit de moto.
58 CREATE OR REPLACE TRIGGER circuit_check BEFORE INSERT OR UPDATE
    ↳ ON Circuit FOR EACH ROW
59 BEGIN
60     check_nationalite(:new.Pays);
61 END;

```

```

62 /
63
64 -- 6. Vérifie la date d'une course de vitesse.
65 CREATE OR REPLACE TRIGGER course_vitesse_check BEFORE INSERT OR
66   ↳ UPDATE ON Course_vitesse FOR EACH ROW
67 BEGIN
68     date_inferior_to_current_time(:new.Date_course);
69 END;
70 /
71
72 -- 7. Vérifie la date d'un contrat et qu'un pilote n'est pas déjà
73   ↳ sous contrat valide
74 -- lors de la création d'un nouveau contrat.
75 CREATE OR REPLACE TRIGGER contrat_check BEFORE INSERT OR UPDATE
76   ↳ ON Contrat FOR EACH ROW
77 DECLARE
78     -- valide BOOLEAN := FALSE;
79     -- CURSOR Contrat IS
80     -- SELECT Annee_debut, Annee_fin
81     -- FROM Contrat C
82     -- WHERE Id_pilote = :NEW.Id_pilote;
83 BEGIN
84     date_inferior_to_current_time(:new.Annee_debut);
85     -- Désactivé car cela cause une erreur de table mutante. Je
86     -- n'ai pas trouvé
87     -- de résolution au problème, après avoir essayé de changer
88     -- la structure de
89     -- données ou utiliser une table temporaire...
90     -- FOR tuple IN Contrat LOOP
91     -- IF (:NEW.Annee_debut BETWEEN tuple.Annee_debut AND
92     --   ↳ tuple.Annee_fin)
93     --   OR (:NEW.Annee_fin BETWEEN tuple.Annee_debut
94     --   ↳ AND tuple.Annee_fin)) THEN
95     --     valide := TRUE;
96     -- END IF;
97     -- END LOOP;
98     -- IF valide = TRUE THEN
99     --   RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be
100     --   ↳ done because the pilot is/was already under a
101     --   ↳ contract during this new contract !');
102     -- END IF;
103 END;
104 /
105
106 -- 8. Vérifie qu'un pilote est bien sous un contrat valide lors
107   ↳ de sa participation

```

```

198      -- à une course et que la moto insérée est bien celle du
199      ↪ contrat.
200  CREATE OR REPLACE TRIGGER participe_check BEFORE INSERT OR UPDATE
201      ↪ ON Participe FOR EACH ROW
202  DECLARE
203      valide BOOLEAN := FALSE;
204      CURSOR Contrat IS
205          SELECT Annee_debut, Annee_fin, Moto_modele, Moto_annee
206          FROM Contrat C
207          WHERE Id_pilote = :NEW.Id_pilote;
208  BEGIN
209      FOR tuple IN Contrat LOOP
210          IF (:NEW.Date_course BETWEEN tuple.Annee_debut AND
211              ↪ tuple.Annee_fin) THEN
212              IF (:NEW.Modele_moto = tuple.Moto_modele AND
213                  ↪ :NEW.Annee_moto = tuple.Moto_annee) THEN
214                  valide := TRUE;
215              END IF;
216          END IF;
217      END LOOP;
218      IF valide = FALSE THEN
219          RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
220              ↪ because the pilot is/was not under a contract during
221              ↪ the race !');
222      END IF;
223  END;
224  /
225
226  -- 9. Vérifie les clés étrangères de la table Course_vitesse.
227  CREATE OR REPLACE TRIGGER verif_course_vitesse_cles BEFORE INSERT
228      ↪ OR UPDATE ON Course_vitesse FOR EACH ROW
229  DECLARE
230      nom Championnat.Nom%TYPE;
231  BEGIN
232      SELECT Nom INTO nom
233      FROM Championnat C
234      WHERE C.Nom = :NEW.Championnat
235            AND C.Annee = :NEW.Annee;
236
237      IF SQL%NOTFOUND THEN
238          RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
239              ↪ because the championship does not exist');
240      END IF;
241  END;
242  /

```

3.2 Méta-données

Les méta-données d'une base de données sont des données stockées à l'intérieur même de la base décrivant la base de données elle-même, elles ne font donc pas partie du jeu de données initial. Oracle DBA stock ces méta-données dans des tables pré-définies spécialement prévues à cet effet. Nous avons donc écrit une démonstration de la récupération de ces méta-données afin de pouvoir les visualiser et avoir des informations sur les contraintes actuellement en vigueur dans notre base. Cependant, vous pourrez constater dans les commentaires du code que nous nous sommes confrontés à un problème nous empêchant de mener notre expérience jusqu'au bout.

```
1      -- Liste toutes les contraintes de clés étrangères -> clés
      ↳ primaires, trié
2      -- par nom de table.
3      SELECT C1.TABLE_NAME AS Table_esclave, C2.TABLE_NAME AS
      ↳ Table_maitre, C1.CONSTRAINT_NAME AS Contrainte_id
4      FROM USER_CONSTRAINTS C1, USER_CONSTRAINTS C2
5      WHERE C1.R_CONSTRAINT_NAME = C2.CONSTRAINT_NAME
6             AND C1.CONSTRAINT_TYPE = 'R'
7      ORDER BY C1.TABLE_NAME ASC;
8
9      -- Censé lister les contraintes 'CHECK' de la table Participe.
      ↳ Cependant,
10     -- la table USER_CONSTRAINTS ne semble pas contenir les CHECK que
      ↳ l'on a
11     -- défini... Je ne comprends pas pourquoi.
12     SELECT C.TABLE_NAME, C.CONSTRAINT_NAME
13     FROM USER_CONSTRAINTS C
14     WHERE C.TABLE_NAME = 'Participe'
15            AND C.CONSTRAINT_TYPE = 'C'
16     ORDER BY C.TABLE_NAME ASC;
17
18     -- Censé lister les triggers de la table Participe. Cependant, la
      ↳ table
19     -- ALL_TRIGGERS ne semble pas contenir les triggers que l'on a
      ↳ défini... Je
20     -- ne comprends pas pourquoi.
21     SELECT T.TABLE_NAME, T.TRIGGER_NAME, T.TRIGGER_TYPE,
      ↳ T.TRIGGERING_EVENT
22     FROM ALL_TRIGGERS T
23     WHERE T.TABLE_NAME = 'Participe'
24     ORDER BY T.TABLE_NAME ASC;
```

Listing 7 – Code SQL permettant d’afficher les contraintes de notre base

1	TABLE_ESCLAVE	TABLE_MAITRE	CONTRAINTE_ID
2	-----	-----	-----
3	CONTRAT	PILOTE	SYS_C0041933
4	CONTRAT	TEAM	SYS_C0041935
5	CONTRAT	MODELE_MOTO	SYS_C0041934
6	COURSE_VITESSE	CIRCUIT	SYS_C0041929
7	MODELE_MOTO	MARQUE	SYS_C0041928
8	PARTICIPE	PILOTE	SYS_C0041930
9	PARTICIPE	COURSE_VITESSE	SYS_C0041931
10	PARTICIPE	MODELE_MOTO	SYS_C0041932
11	TEAM	MARQUE	SYS_C0041927
12			
13	no rows selected		
14			
15	no rows selected		

Listing 8 – Résultat de notre interrogation des méta-données

Nous pouvons observer, sur la première requête qui à bien fonctionné, que les contraintes d’intégrités entre clés primaires et clés étrangères sont bien listées : pour le vérifier, il suffit de mettre en corrélation ce résultat avec le script de création de la base, vu dans la Sous-Section 2.1. Nous avons donc la première colonne avec la table esclave, c’est-à-dire celle qui possède la clé étrangère, la table maître dans la deuxième colonne, c’est-à-dire celle qui possède la clé primaire. Enfin, dans la troisième colonne, nous avons l’identifiant de la contrainte qui est affiché.

3.3 Analyse des requêtes

Dans cette dernière section, nous avons utilisé l’outil Oracle Explain Plan afin de pouvoir analyser l’exécution de nos requêtes SQL. Cette analyse permet de modifier nos requêtes afin qu’elles soient interprétées d’une manière différente par l’optimiseur de requête (Cost Based Optimizer (CBO)) lors de la détection d’un soucis de performance. L’outil s’utilise à travers une clause SQL puis par interrogation d’une table prévue pour contenir les résultats d’Explain Plan. Ci-dessous sont listés nos requêtes permettant d’interroger l’outil et récupérer le résultat :

```

1  -- Création des plans d'exécution.
2
3  -- 1. Liste des scores des pilotes au MotoGP de 2016.
4  EXPLAIN PLAN FOR
```

```

5      SELECT Pi.Id, Pi.Numero, Pi.Nom, Pi.Prenom,
        ↪ SUM(Pa.Points_gagnes) AS Nombre_total_de_point
6      FROM Participe Pa, Pilote Pi
7      WHERE Pa.Id_pilote = Pi.Id
8             AND Pa.Championnat LIKE 'MotoGP'
9             AND TO_CHAR(Pa.Date_course, 'YYYY') LIKE '2016'
10     GROUP BY Pi.Id, Pi.Numero, Pi.Nom, Pi.Prenom
11     ORDER BY Nombre_total_de_point DESC;
12 SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY('PLAN_TABLE', null,
        ↪ null));

13
14 -- 2. Liste des scores des teams au MotoGP de 2016.
15 EXPLAIN PLAN FOR
16     SELECT C.Team_nom, SUM(Nombre_total_de_point) AS
        ↪ Nombre_total_de_point
17     FROM Contrat C, MotoGP_2016_Score_pilotes S
18     WHERE S.Id = C.Id_pilote
19            AND TO_DATE(2016, 'YYYY') BETWEEN C.Annee_debut AND
        ↪ C.Annee_fin
20     GROUP BY C.Team_nom
21     ORDER BY Nombre_total_de_point DESC;
22 SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY('PLAN_TABLE', null,
        ↪ null));

23
24 -- 3. Statistiques diverses sur les pilotes du MotoGP.
25 EXPLAIN PLAN FOR
26     SELECT Pi.Numero, Pi.Nom, Pi.Prenom, Pi.Age, Pi.Nationalite,
        ↪ Pi.Sexe,
27            SUM(Pa.Points_gagnes) AS Total_de_points_gagnes,
28            MIN(Pa.Classement) AS Meilleur_classement,
29            MAX(Pa.Vitesse_moy) AS Vitesse_moyenne,
30            MIN(Pa.Meilleur_tour) AS Meilleur_tour
31     FROM Pilote Pi, Participe Pa
32     WHERE Pa.Id_pilote = Pi.Id
33            AND Pa.Championnat LIKE 'MotoGP'
34     GROUP BY Pi.Numero, Pi.Nom, Pi.Prenom, Pi.Age,
        ↪ Pi.Nationalite, Pi.Sexe;
35 SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY('PLAN_TABLE', null,
        ↪ null));

```

Listing 9 – Code SQL permettant d'utiliser Oracle Explain Plan
 Ci-dessous le résultat de nos requêtes d'interrogation du plan d'exécution :

```

1 -- 1. Liste des scores des pilotes au MotoGP de 2016.
2

```

3 PLAN_TABLE_OUTPUT

4 -----
5 Plan hash value: 2815421071

6
7 -----
8 |Id|Operation|Name|Rows|Bytes|Cost(%CPU)|Time|
9 -----
10 | 0|SELECT STATEMENT| | 50| 5750| 9 (34)|00:00:01|
11 | 1| SORT ORDER BY| | 50| 5750| 9 (34)|00:00:01|
12 | 2| HASH GROUP BY| | 50| 5750| 9 (34)|00:00:01|
13 |*3| HASH JOIN| | 50| 5750| 7 (15)|00:00:01|
14 | 4| TABLE ACCESS FULL|PILOTE| 15| 930| 3 (0)|00:00:01|
15 |*5| TABLE ACCESS FULL|PARTICIPE| 50| 2650| 3 (0)|00:00:01|
16 -----

17
18 Predicate Information (identified by operation id):
19 -----

20 3 - access("PA"."ID_PILOTE"="PI"."ID")
21 5 - filter("PA"."CHAMPIONNAT"='MotoGP' AND
22 TO_CHAR(INTERNAL_FUNCTION
↪ ("PA"."DATE_COURSE"), 'YYYY')='2016')

23
24 Note

25 -----
26 - dynamic sampling used for this statement (level=2)

27
28 -- 2. Liste des scores des teams au MotoGP de 2016.

29
30 PLAN_TABLE_OUTPUT

31 -----
32 Plan hash value: 2104335280

33
34 -----
35 |Id|Operation|Name|Rows|Bytes|Cost(%CPU)|Time|
↪ |
36 -----
37 | 0|SELECT STATEMENT| | 40| 1240| 13
↪ (31)|00:00:01|
38 | 1| SORT ORDER BY| | 40| 1240| 13
↪ (31)|00:00:01|
39 | 2| HASH GROUP BY| | 40| 1240| 13
↪ (31)|00:00:01|
40 | 3| VIEW|VM_NWVW_1| 40| 1240| 11
↪ (19)|00:00:01|
41 | 4| HASH GROUP BY| | 40| 7040| 11
↪ (19)|00:00:01|

```

42 |*5|      HASH JOIN          |          |  40| 7040|  10
   ↳ (10)|00:00:01|
43 |*6|      HASH JOIN          |          |  12| 1476|   7
   ↳ (15)|00:00:01|
44 |*7|      TABLE ACCESS FULL|CONTRAT  |  12|  732|   3
   ↳ (0)|00:00:01|
45 | 8|      TABLE ACCESS FULL|PILOTE   |  15|   930|   3
   ↳ (0)|00:00:01|
46 |*9|      TABLE ACCESS FULL|PARTICIPE|  50| 2650|   3
   ↳ (0)|00:00:01|

```

```

47 -----

```

48

49 Predicate Information (identified by operation id):

```

50 -----

```

51

```

52   5 - access("PA"."ID_PILOTE"="PI"."ID")
53   6 - access("PI"."ID"="C"."ID_PILOTE")
54   7 - filter("C"."ANNEE_DEBUT"<=TO_DATE('2016','YYYY') AND
55          "C"."ANNEE_FIN">=TO_DATE('2016','YYYY'))
56   9 - filter("PA"."CHAMPIONNAT"='MotoGP' AND
57          TO_CHAR(INTERNAL_FUNCTION
58          ↳ ("PA"."DATE_COURSE"),'YYYY')='2016')

```

58

59 Note

```

60 -----

```

61 - dynamic sampling used for this statement (level=2)

62

63 -- 3. Statistiques diverses sur les pilotes du MotoGP.

64

65 PLAN_TABLE_OUTPUT

```

66 -----

```

67 Plan hash value: 2188167404

68

```

69 -----
70 |Id|Operation          |Name      |Rows |Bytes|Cost(%CPU)|Time      |
71 -----
72 | 0|SELECT STATEMENT    |          |  50| 8250|  8 (25)|00:00:01|
73 | 1|HASH GROUP BY       |          |  50| 8250|  8 (25)|00:00:01|
74 |*2|HASH JOIN           |          |  50| 8250|  7 (15)|00:00:01|
75 | 3|TABLE ACCESS FULL   |PILOTE    |  15| 1230|  3 (0)|00:00:01|
76 | 4|TABLE ACCESS FULL   |PARTICIPE |  50| 4150|  3 (0)|00:00:01|
77 -----

```

78

79 Predicate Information (identified by operation id):

```

80 -----

```

81

```

82      2 - access("PA"."ID_PILOTE"="PI"."ID")
83      4 - filter("PA"."CHAMPIONNAT"='MotoGP')
84
85 Note
86 -----
87      - dynamic sampling used for this statement (level=2)

```

Listing 10 – Résultat d'Oracle Explain Plan

Nous pouvons ainsi, pour chaque requête, observer toutes les actions effectuées par le SGBD. Pour chacune de ses actions nous sont affichés des informations tel que le nombre de colonnes traitées, la taille des données, le temps et la demande en puissance de traitement estimé pour effectuer l'action. Nous obtenons aussi des informations sur l'ordre d'exécution des différentes opérations constituant notre requête. Par exemple, pour effectuer une jointure entre Pilote et Participe, on observe que notre base demande d'abord un accès complet à ces deux tables. Ensuite, la restriction effectuée sur le produit cartésien de nos deux tables est opérée à l'aide d'une comparaison de hashes. Il en va de même pour le GROUP BY, qui groupe les hashes communs sur une certaine colonne. On remarque que l'attribut sur lequel porte la jointure est nommé comme étant un accès, tandis que les attributs sur lesquels portent des restrictions sont affichés en tant que filtres. Enfin, on observe bien que toutes les opérations sont estimées être très peu coûteuse, ce qui est évident compte tenu de la faible taille de notre base de données.

4 Conclusion

Ce projet nous aura beaucoup appris concernant Oracle DBA et PL/SQL. Concernant Oracle DBA, nous aurons constaté qu'il existe un large écosystème d'outil de développement autour de ce SGBD, des outils de débogage jusqu'à l'analyse de performance en passant par des utilitaires facilitant la manipulation des données. Par rapport à PL/SQL, nous avons pu expérimenter différentes utilisations du langage, par exemple pour gérer des contraintes avancées ou encore permettre d'automatiser certaines opérations de gestion des données nécessitant un programme dynamique. Pour conclure, ce projet aura été une bonne approche et une introduction intéressante à l'administration de base de données.