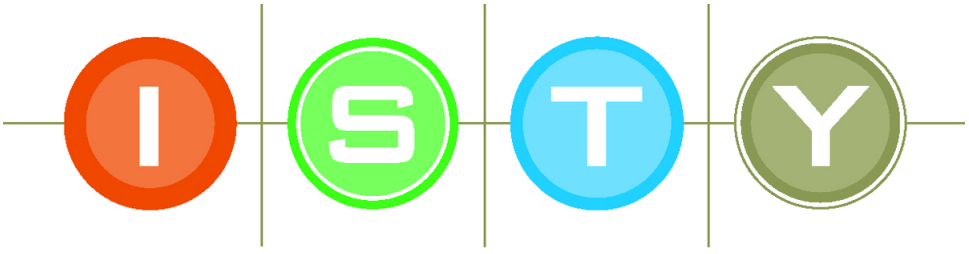
Base de données avec Oracle DBA sur les championnats de moto

Pierre AYOUB et Maël ROUXEL

17 janvier 2019



INSTITUT DES SCIENCES ET TECHNIQUES DES YVELINES

Résumé

Oracle Database est un système de gestion de base de données relationnelle (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

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 course, 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 partie que d'une team. Dans le cas où le pilote participe à plus d'un championnat sur une saison, alors il peut faire partie 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 différent 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 une 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'intégrité basiques telles 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 <
        ↪ 2000),
27      Couple      FLOAT       CHECK (Couple > 1     AND Couple <
        ↪ 20),
28      Puissance   FLOAT       CHECK (Puissance > 1  AND Puissance <
        ↪ 500),
29      Poids       FLOAT       CHECK (Poids > 30     AND Poids <
        ↪ 500),
```

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

```
/
CREATE OR REPLACE TRIGGER pilote_set_id BEFORE INSERT ON Pilote
↪ FOR EACH ROW
DECLARE
BEGIN
    :NEW.Id := PiloteID.NextVal;
END;
/
```

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

2.3 Manipulation des données par requêtes SQL

Afin de pouvoir interagir avec la base de données, nous avons imaginé et élaboré 20 requêtes qui pourrait être émises dans une situation vraisemblable (pas forcément commune cela dit) tout en tâchant de faire varier les fonctionnalités utilisées. Certaines de ces requêtes pourraient correspondre à des demandes simples de la part d'un utilisateur consultant un site web ayant enregistré les données, mais d'autres requêtes correspondent plutôt à des actions uniques qui seraient demandées dans une situation particulière.

Voici donc la liste de ces 20 requêtes :

```
--1. On voudrait connaître la liste des championnats différents
↪ pour chaque année.

SELECT Cv.championnat, EXTRACT(YEAR FROM Cv.date_course) AS Annee
FROM Course_vitesse Cv
GROUP BY Cv.championnat, EXTRACT(YEAR FROM Cv.date_course);

--2. On voudrait connaître la liste des pilotes et de leur score
↪ total au championnat MotoGP de 2016, autrement dit, le
↪ classement de ce championnat

SELECT Pi.prenom, Pi.nom, Pi.numero, SUM(Pa.points_gagnes) AS
↪ Nombre_total_de_point
FROM Pilote Pi, Participe Pa
WHERE Pa.id_pilote = Pi.id
    AND Pa.championnat LIKE 'MotoGP'
    AND EXTRACT(YEAR FROM Pa.date_course) LIKE 2016
GROUP BY Pi.prenom, Pi.nom, Pi.numero
ORDER BY SUM(Pa.points_gagnes) DESC;

-- 3.  Dans le même genre, on voudrait connaître le nombre de
↪ points total gagnés par les pilotes dans tous les
↪ championnats confondus, afin de faire un classement général
```

```sql

SELECT P.Nom, P.Prenom, SUM(PA.points_gagnes) as total_point
FROM Pilote P,Participe PA
WHERE P.Id = PA.Id_pilote
GROUP BY P.Nom,P.Prenom
ORDER BY SUM(PA.points_gagnes) DESC;

--4. On veut maintenant connaître le classement des équipes au MotoGP de 2016, en additionnant les points remportés par chaque pilote courant pour cette équipe. On va ainsi faire une somme des résultats de la requête n°2.

SELECT C.Team_nom, SUM(Nombre_total_de_point_pilote) AS Nombre_total_de_point
FROM Contrat C, (
    SELECT Pi.id, SUM(Pa.points_gagnes) AS Nombre_total_de_point_pilote
    FROM Pilote Pi, Participe Pa
    WHERE Pa.id_pilote = Pi.id
        AND Pa.championnat LIKE 'MotoGP'
        AND EXTRACT(YEAR FROM Pa.date_course) LIKE 2016
    GROUP BY Pi.id
) score_pilote
WHERE score_pilote.id = C.id_pilote
    AND EXTRACT(YEAR FROM C.annee_debut) <= 2016
    AND (
        EXTRACT(YEAR FROM C.annee_fin) >= 2016
        OR C.annee_fin IS NULL
    )
GROUP BY C.team_nom
ORDER BY SUM(Nombre_total_de_point_pilote) DESC;

--5. Et enfin, on veut connaître le classement des marques pour le MotoGP. Étant donné que chaque équipe est affiliée à une marque, cela signifie qu'on va faire une somme des résultats de la requête n°4.

SELECT T.marque, SUM(Nombre_total_de_point_team) AS Nombre_total_de_point
FROM Team T, (
    SELECT C.Team_nom, SUM(Nombre_total_de_point_pilote) AS Nombre_total_de_point_team
    FROM Contrat C, (
        SELECT Pi.id, SUM(Pa.points_gagnes) AS Nombre_total_de_point_pilote
        FROM Pilote Pi, Participe Pa
```

```
53              WHERE Pa.id_pilote = Pi.id
54                  AND Pa.championnat LIKE 'MotoGP'
55                  AND EXTRACT(YEAR FROM Pa.date_course ) LIKE 2016
56              GROUP BY Pi.id
57          ) score_pilote
58          WHERE score_pilote.id = C.id_pilote
59              AND EXTRACT(YEAR FROM C.annee_debut) <= 2016
60              AND (
61                  EXTRACT(YEAR FROM C.annee_fin) >= 2016
62                  OR C.annee_fin IS NULL
63              )
64          GROUP BY C.team_nom
65      ) score_team
66      WHERE T.nom = score_team.team_nom
67      GROUP BY T.marque
68      ORDER BY Nombre_total_de_point DESC;
69
70      --6. Intéressés par les performances des pilotes espagnols, on
        ↪  désire connaître la moyenne de points obtenus par les pilotes
        ↪  espagnols pour chaque course du MotoGP.
71
72      SELECT Pi.prenom, Pi.nom, Pi.numero, AVG(Pa.points_gagnes) AS
        ↪  Moyenne
73      FROM Pilote Pi, Participe Pa
74      WHERE Pa.id_pilote = Pi.id
75          AND Pa.championnat LIKE 'MotoGP'
76          AND Pi.nationalite LIKE 'ES'
77      GROUP BY Pi.prenom, Pi.nom, Pi.numero;
78
79      --7. On voudrait savoir combien de pilotes de moins 30 ans sont
        ↪  actuellement sous contrat avec une moto de type "sportive".
80
81      SELECT COUNT(*) as nb_pilotes_moins_30_ans_moto_sportive
82      FROM Pilote P, Contrat C, Modele_moto M
83      WHERE P.Id = C.Id_pilote
84          AND C.Moto_modele = M.Nom
85          AND C.Moto_annee = M.Annee
86          AND M.Genre = 'Sportive'
87          AND P.Age <= 30
88          AND EXTRACT(YEAR FROM C.annee_fin) >= 2019;
89
90      --8. Curieux de connaître les pilotes les moins performants, on
        ↪  veut obtenir la liste des pilotes n'ayant jamais participé à
        ↪  une course, ainsi que la liste des pilotes n'ayant jamais
        ↪  remporté une course.
91
```

```sql
    SELECT Pi.prenom, Pi.nom, 0 AS Nombre_victoire
    FROM Pilote Pi
    WHERE Pi.id NOT IN (
        SELECT Pa.id_pilote
        FROM Participe Pa
    )
    UNION
    SELECT Pi.prenom, Pi.nom, COUNT(*) AS Nombre_victoire
    FROM Pilote Pi, Participe Pa
    WHERE Pi.id = Pa.id_pilote
        AND Pa.classement = 1
    GROUP BY Pi.prenom, Pi.nom
    HAVING COUNT(*) = 0;

    --9. Une marque voudrait obtenir des contrats auprès de pilotes,
    ↪  et pour cela elle désire développer un modèle de moto qui
    ↪  s'inspire des modèles ayant été les plus populaires. Il va
    ↪  alors demander la liste classée des modèles de motos ayant
    ↪  été utilisés plus de six fois lors de courses (pas forcément
    ↪  la même).

    SELECT Modele_moto, EXTRACT(YEAR FROM Annee_moto) as Annee,
    ↪  count(*) as Participations
    FROM Participe
    GROUP BY Modele_moto, Annee_moto
    HAVING count(*) >= 6
    ORDER BY count(*) DESC;

    -- 10. Une nouvelle loi a décrétée par l'organisation des courses
    ↪  de moto: tous les contrats terminant cette année et ayant une
    ↪  durée inférieure à 3 ans seront prolongés jusqu'en 2021. On
    ↪  définit une requête qui répond à cette demande.

    UPDATE Contrat
    SET annee_fin = TO_DATE('2021-12-31', 'YYYY-MM-DD')
    WHERE EXTRACT(YEAR FROM annee_fin) = 2019
        AND EXTRACT(YEAR FROM annee_fin) - EXTRACT(YEAR FROM
        ↪  annee_debut) < 3;

    -- 11. Suite au Brexit, les marques britanniques ont décidé
    ↪  d'augmenter de 10% le prix de tous leurs modèles de moto.

    UPDATE Modele_moto
    SET Prix = Prix * 1.1
    WHERE Marque = (SELECT Nom FROM Marque M
                    WHERE Marque = M.nom
```

```sql
                    AND Nationalite = 'GB');

-- 12. On désire faire une étude de stratégie des marques vis à vis de la puissance de leur modèle. On demande une liste donnant le modèle le plus puissant de chaque marque ayant été utilisé au moins une fois, et on donne le meilleur temps moyen obtenu avec ce modèle.

SELECT MMA.Marque, MMA.Nom, MMA.Annee, MMA.Puissance, P.Vitesse_moy
FROM Modele_moto MMA, Participe P
WHERE Puissance = (SELECT MAX(Puissance)
                   FROM Modele_moto MMB
                   WHERE MMB.Marque = MMA.Marque)
    AND P.Vitesse_moy = (SELECT MAX(Vitesse_moy)
                         FROM Participe PB
                         WHERE PB.Modele_moto = P.Modele_moto
                         AND PB.Annee_moto = P.Annee_moto)
    AND MMA.Nom = P.Modele_moto
    AND MMA.Annee = P.Annee_moto;

-- 13. Suite à un scandale ayant éclaté après la révélation de pots au vin parmi les pilotes britanniques lors du championnat Superbike de 2015, l'organisation a décidé de supprimer de la base de données toutes les participations de ces pilotes lors de ce championnat.

DELETE FROM Participe
WHERE Id_pilote = (SELECT Id FROM Pilote P
                   WHERE Nationalite = 'GB'
                   AND Id_pilote = P.Id)
AND Championnat = 'Superbike'
AND EXTRACT(YEAR FROM Date_course) = 2015;

-- 14. Suite à une nouvelle législation espagnole, les motos de type Cafe Racer de plus de 200kg sont interdites sur les circuits espagnols. Le gouvernement a fait pression sur l'organisation pour qu'elle supprime également toutes les participations passées considérées comme maintenant interdites.

DELETE FROM Participe P
WHERE P.Modele_moto = (SELECT MM.Nom FROM Modele_moto MM
                   WHERE Genre = 'Cafe Racer'
                       AND Poids >= 200
                       AND MM.nom = P.Modele_moto
```

```
159                         AND MM.annee = P.annee_moto)
160     AND Date_course = (SELECT Date_course FROM Course_vitesse CVT
161                 WHERE CVT.Date_course = P.Date_course
162                         AND Circuit = (SELECT Nom FROM circuit
163                                         WHERE Pays = 'ES'));
164
165     -- 15. A l'occasion du salon de la moto, les marques ont décidé
        ↪  de sur-renchérir un peu en augmentant de 1000€ le prix de
        ↪  tous leurs modèles ayant déjà remporté une première place
        ↪  lors d'une course.
166
167     UPDATE Modele_moto
168     SET Prix = Prix + 1000
169     WHERE (Nom,Annee) = (SELECT Modele_moto,Annee_moto FROM Participe
170                         WHERE Classement = 1
171                             AND Participe.Modele_moto =
                                ↪  Modele_moto.Nom
172                             AND Participe.Annee_moto =
                                ↪  Modele_moto.Annee
173                             GROUP BY Modele_moto,Annee_moto);
174
175     -- 16. L'organisation des championnats a décrété que dû aux
        ↪  conditions de température difficiles en Australie, les
        ↪  comptes de points sont rééquilibrés en augmentant d'un point
        ↪  tous les scores obtenus lors de courses sur un circuit
        ↪  Australien.
176
177     UPDATE Participe P
178     SET points_gagnes = points_gagnes + 1
179     WHERE date_course = (SELECT date_course as DCB FROM
        ↪  Course_vitesse CV
180                         WHERE CV.Date_course = P.date_course
181                             AND CV.Circuit = ( SELECT Nom from
                                ↪  Circuit C
182                             WHERE C.Nom = CV.Circuit
183                                                 AND C.Pays =
                                                    ↪  'AU'));
184
185     -- 17. On désire savoir quels sont les modèles de motos ayant été
        ↪  le plus souvent la cible d'un contrat. On dresse le compte du
        ↪  nombre de contrats effectués pour chaque modèle de moto.
186
187     SELECT Moto_modele, EXTRACT(YEAR FROM Moto_annee) as Annee,
        ↪  count(*) AS Nb_of_contracts
188     FROM Contrat
189     GROUP BY Moto_modele,Moto_annee
```

```
190     ORDER BY count(*) DESC;
191
192     -- 18. On voudrait voir quel pilote est le plus efficace sur quel
        ↪  circuits. On dresse le classements du plus grand nombre de
        ↪  victoire par pilote pour chaque circuit.
193
194     SELECT C.Nom , P.Nom, P.Prenom, COUNT(*) as nb_win
195     FROM Circuit C, Pilote P, Participe PA
196     WHERE PA.Id_pilote = P.Id
197         AND PA.Date_course = (SELECT CV.Date_course FROM
            ↪  Course_vitesse CV
198                               WHERE CV.Circuit = C.Nom
199                               AND PA.Date_course = CV.date_course)
200                               AND PA.Classement = 1
201     GROUP BY C.Nom , P.Nom, P.Prenom
202     ORDER BY count(*);
203
204     --19. Pour la nouvelle saison, on met à jour l'âge de tous les
        ↪  pilotes toujours actifs (encore sous contrats) en
        ↪  l'augmentant de 1 an.
205
206     UPDATE Pilote P
207     SET Age = Age + 1
208     WHERE Id = (SELECT Id_pilote FROM contrat C
209             WHERE P.Id = C.Id_pilote
210                   AND EXTRACT(YEAR FROM annee_fin) >= 2019);
211
212     -- 20. Une équipe désire connaître la liste des pilotes les plus
        ↪  actifs récemment en vue de leur proposer des futurs contrats.
        ↪  On fait une requête donnant la liste des pilotes encore sous
        ↪  contrat, avec la date de leur dernière course, et la date
        ↪  d'expiration de leur contrat actuel.
213
214     SELECT P.Nom,P.prenom,PA.Date_course, EXTRACT(YEAR FROM
        ↪  C.annee_fin) as fin_contrat
215     FROM Pilote P, Participe PA, Contrat C
216     WHERE P.Id = PA.id_pilote
217         AND P.Id = C.id_pilote
218         AND EXTRACT(YEAR FROM C.annee_fin) >= 2019
219         AND PA.Date_course = (SELECT MAX(Date_course) FROM Participe
            ↪  PAB
220                       WHERE PA.Date_course = PAB.Date_course
221                       AND PA.id_pilote = PAB.id_pilote)
222     GROUP BY P.Nom,P.prenom,PA.Date_course,EXTRACT(YEAR FROM
        ↪  C.annee_fin);
```

Listing 5 – Code SQL permettant d'exécuter les requêtes sur la base de données

2.4 Vues

Notre base de données contient quelques vues permettant de visualiser des scores et des statistiques calculées à partir de notre jeu de données. De tels vues se destineraient à être incluses 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;
60      GRANT SELECT ON MotoGP_Pilote_win to PUBLIC;
```

Listing 6 – 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'intégrité. 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.
    ↪  Nécéssite un
4   -- trigger/une procédure pour utiliser SYSDATE.
5   CREATE OR REPLACE PROCEDURE date_inferior_to_current_time
    ↪  (new_date IN DATE) IS
6   BEGIN
7       IF new_date > SYSDATE THEN
8           RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
            ↪  because the date is greater than today !');
9       END IF;
10  END;
11  /
12
13  -- 2. Vérifie qu'une nationalité est correct. On pourrait
    ↪  utiliser un CHECK, mais
14  -- il faudrait copier-coller les valeurs de nationalités à la
    ↪  main, alors que
15  -- l'on peux les regrouper dans une procédure.
16  CREATE OR REPLACE PROCEDURE check_nationalite (nat IN CHAR) IS
17  BEGIN
18      IF (nat NOT IN ('FR', 'GB', 'US', 'IT', 'ES', 'JP', 'CH',
        ↪  'DE', 'AU', 'QT', 'PB')) THEN
19          RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
            ↪  because the nationality is incorrect !');
20      END IF;
21  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
        ↪  UPDATE ON Course_vitesse FOR EACH ROW
66      BEGIN
67          date_inferior_to_current_time(:new.Date_course);
68      END;
69      /
70
71      -- 7. Vérifie la date d'un contrat et qu'un pilote n'est pas déjà
        ↪  sous contrat valide
72      -- lors de la création d'un nouveau contrat.
73      CREATE OR REPLACE TRIGGER contrat_check BEFORE INSERT OR UPDATE
        ↪  ON Contrat FOR EACH ROW
74      DECLARE
75          --  valide BOOLEAN := FALSE;
76          --  CURSOR Contrat IS
77              --  SELECT Annee_debut, Annee_fin
78              --  FROM Contrat C
79              --  WHERE Id_pilote = :NEW.Id_pilote;
80      BEGIN
81          date_inferior_to_current_time(:new.Annee_debut);
82          --  Désactivé car cela cause une erreur de table mutante. Je
            ↪  n'ai pas trouvé
83          --  de résolution au problème, après avoir essayé de changer
            ↪  la structure de
84          --  données ou utiliser une table temporaire...
85          --  FOR tuple IN Contrat LOOP
86              --  IF ((:NEW.Annee_debut BETWEEN tuple.Annee_debut AND
                ↪  tuple.Annee_fin)
87                      --  OR (:NEW.Annee_fin BETWEEN tuple.Annee_debut
                        ↪  AND tuple.Annee_fin)) THEN
88                  --  valide := TRUE;
89              --  END IF;
90          --  END LOOP;
91          --  IF valide = TRUE THEN
92              --  RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be
                ↪  done because the pilot is/was already under a
                ↪  contract during this new contract !');
93          --  END IF;
94      END;
95      /
96
97      -- 8. Vérifie qu'un pilote est bien sous un contrat valide lors
        ↪  de sa participation
```

```
98      -- à une course et que la moto insérée est bien celle du
        ↪  contrat.
99      CREATE OR REPLACE TRIGGER participe_check BEFORE INSERT OR UPDATE
        ↪  ON Participe FOR EACH ROW
100     DECLARE
101         valide BOOLEAN := FALSE;
102         CURSOR Contrat IS
103             SELECT Annee_debut, Annee_fin, Moto_modele, Moto_annee
104             FROM Contrat C
105             WHERE Id_pilote = :NEW.Id_pilote;
106     BEGIN
107         FOR tuple IN Contrat LOOP
108             IF (:NEW.Date_course BETWEEN tuple.Annee_debut AND
                ↪  tuple.Annee_fin) THEN
109                 IF (:NEW.Modele_moto = tuple.Moto_modele AND
                    ↪  :NEW.Annee_moto = tuple.Moto_annee) THEN
110                     valide := TRUE;
111                 END IF;
112             END IF;
113         END LOOP;
114         IF valide = FALSE THEN
115             RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
                ↪  because the pilot is/was not under a contract during
                ↪  the race !');
116         END IF;
117     END;
118     /
119
120     -- 9. Vérifie les clés étrangères de la table Course_vitesse.
121     CREATE OR REPLACE TRIGGER verif_course_vitesse_cles BEFORE INSERT
        ↪  OR UPDATE ON Course_vitesse FOR EACH ROW
122     DECLARE
123         nom Championnat.Nom%TYPE;
124     BEGIN
125         SELECT Nom INTO nom
126         FROM Championnat C
127         WHERE C.Nom = :NEW.Championnat
128             AND C.Annee = :NEW.Annee;
129
130         IF SQL%NOTFOUND THEN
131             RAISE_APPLICATION_ERROR(-20001, 'Insertion cannot be done
                ↪  because the championship does not exist');
132         END IF;
133     END;
134     /
```

Listing 7 – Code PL/SQL programmant les triggers de notre base de données

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 stocke 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 contraites 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 8 – Code SQL permettant d'afficher les contrainres de notre base

```
TABLE_ESCLAVE    TABLE_MAITRE     CONTRAINTE_ID
-------------------------------- ------------------------------
CONTRAT          PILOTE           SYS_C0041933
CONTRAT          TEAM             SYS_C0041935
CONTRAT          MODELE_MOTO      SYS_C0041934
COURSE_VITESSE   CIRCUIT          SYS_C0041929
MODELE_MOTO      MARQUE           SYS_C0041928
PARTICIPE        PILOTE           SYS_C0041930
PARTICIPE        COURSE_VITESSE   SYS_C0041931
PARTICIPE        MODELE_MOTO      SYS_C0041932
TEAM             MARQUE           SYS_C0041927

no rows selected

no rows selected
```

Listing 9 – Résultat de notre interrogation des méta-données

Nous pouvons observer, sur la première requête qui à bien fonctionné, que les contraintes d'intégrité 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 requêtes (Cost Based Optimizer (CBO)) lors de la détection d'un souci 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 listées nos requêtes permettant d'interroger l'outil et récupérer le résultat :

```
-- Création des plans d'exécution.

-- 1. Liste des scores des pilotes au MotoGP de 2016.
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 10 – 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
```

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 2815421071

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

Predicate Information (identified by operation id):
---------------------------------------------------
3 - access("PA"."ID_PILOTE"="PI"."ID")
5 - filter("PA"."CHAMPIONNAT"='MotoGP' AND
           TO_CHAR(INTERNAL_FUNCTION
           ↪  ("PA"."DATE_COURSE"),'YYYY')='2016')

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

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

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------
Plan hash value: 2104335280

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

```
|*5|     HASH JOIN           |         |   40| 7040|  10
  ↪  (10)|00:00:01|
|*6|       HASH JOIN         |         |   12| 1476|   7
  ↪  (15)|00:00:01|
|*7|        TABLE ACCESS FULL|CONTRAT  |   12|  732|   3
  ↪  (0)|00:00:01|
| 8|        TABLE ACCESS FULL|PILOTE   |   15|  930|   3
  ↪  (0)|00:00:01|
|*9|       TABLE ACCESS FULL |PARTICIPE|   50| 2650|   3
  ↪  (0)|00:00:01|
-------------------------------------------------------------------------

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

   5 - access("PA"."ID_PILOTE"="PI"."ID")
   6 - access("PI"."ID"="C"."ID_PILOTE")
   7 - filter("C"."ANNEE_DEBUT"<=TO_DATE('2016','YYYY') AND
              "C"."ANNEE_FIN">=TO_DATE('2016','YYYY'))
   9 - filter("PA"."CHAMPIONNAT"='MotoGP' AND
              TO_CHAR(INTERNAL_FUNCTION
               ↪  ("PA"."DATE_COURSE"),'YYYY')='2016')

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

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

PLAN_TABLE_OUTPUT
-------------------------------------------------------------------------
Plan hash value: 2188167404

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

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

```
   2 - access("PA"."ID_PILOTE"="PI"."ID")
   4 - filter("PA"."CHAMPIONNAT"='MotoGP')

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

Listing 11 – Résultat d'Oracle Explain Plan

Nous pouvons ainsi, pour chaque requête, observer toutes les actions effectuées par le SGBD. Pour chacune de ces actions nous sont affichées des informations telles que le nombre de colonnes traitées, la taille des données, le temps et la demande en puissance de traitement estimée 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 hashs. Il en va de même pour le GROUP BY, qui groupe les hashs 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.