RAPPORT BD50

P2012

G16 – Gestion du Tour de France

BENJAMIN BINI – YANNICK GUILLEMOT – JULIEN KLINGENMEYER – BASTIEN SCHEURER

SOMMAIRE

1. HISTORIQUE DU DOCUMENT

Version	Date	Auteur	Modifications du document
Etape 1	29/03	Yannick Guillemot	Création du dossier de conception et ajout des parties de l'étape 1
	14/04	Yannick Guillemot Julien Klingenmeyer	Contexte du projet, Domaine étudié, Fonctionnement du système, Les acteurs
Etape 2	18/05	Bastien Scheurer	Insertion du MEA
	19/05	Benjamin Bini	Modèle du dictionnaire
	20/05	Julien Klingenmeyer	Refonte du périmètre système et extensions possibles
Etape 3		Benjamin Bini Bastien Scheurer	MLR Normalisé
Etape 4		Benjamin Bini Julien Klingenmeyer	MLR Optimisé
		Julien Klingenmeyer	Optimisation Oracle
Etape 5		Yannick Guillemot	Maquettes
		Yannick Guillemot Bastien Scheurer	Scripts et Exemples

2. DESCRIPTION DU SUJET

a) Contexte du projet

Le but de ce projet est de créer un site web dans le cadre des matières BD50 et GL52. Ce portail devra être connecté à une base de données à modéliser, normaliser et optimiser dans le cadre de ce projet. L'interface client (frontend) devra s'accompagner d'une interface d'administration (backend).

b) Domaine étudié

Nous avons choisi de travailler sur la gestion du Tour de France. La course cycliste est réputée dans le monde entier, et de nombreux acteurs (journalistes, passionnés, chaînes de télévision,…) désirent pouvoir consulter les différents résultats ainsi que les caractéristiques clés des étapes. Tous ces éléments doivent pouvoir être consultés *via* le portail.

Pour se rapprocher au mieux de la réalité, nous nous sommes basés sur le fonctionnement officiel du Tour à l'aide notamment du règlement et du site internet de la course. Nous retrouvons ainsi dans notre modèle les mêmes types de classements et d'étapes que ceux établis par la Grande Boucle. Nous reviendrons plus bas sur le périmètre choisi pour notre système.

c) Fonctionnement du système

L'ensemble de la base de données est installé sur le système de gestion de bases de données (SGBD) Oracle 11G.

L'interface client est, elle, développée en PL/SQL *via* des procédures stockées incluses dans le schéma de données. Elles sont regroupées en packages. Il existe deux types de packages :

- les *packages* d'interface utilisateur
- les *packages* d'accès aux données

Chaque table dispose de son package d'accès aux données et éventuellement de son package d'interface utilisateur si un affichage de ses données est nécessaire.

Un DAD (Database Access Descriptor) doit également être installé sur la base Oracle pour permettre d'accéder à ces procédures via des requêtes http sur un navigateur internet.

Parallèlement à cela, des fichiers annexes (images, fichiers css ou js) sont stockés dans le serveur distant via un XMLDB.

Enfin, l'interface d'administration est entièrement construite via l'outil de développement fournie par Oracle : APEX. Il permet de développer très facilement une interface web d'accès et d'édition de données.

d) Les acteurs

Le logiciel est destiné tout d'abord destiné à être consultable par le plus grand nombre. On compte donc dans les utilisateurs « normaux », le grand public, les journalistes mais aussi les équipes et coureurs eux-mêmes qui s'en serviront pour consulter les enregistrements aussi complets soient-ils d'un Tour de France.

Il est nécessaire de distinguer un autre type d'utilisateur de la solution, l'administrateur. Dans notre cas, on peut l'identifier en tant que commissaire de course. C'est celui qui aurait tous les droits. On parle ici principalement des droits d'édition des informations comme l'ajout de nouvelles étapes dans un Tour de France, la composition des équipes ou l'enregistrement des différents classements à chaque fin d'étape ; ce dont l'utilisateur basique n'a bien sûr pas accès.

e) Les flux

Les flux de données à l'intérieur de notre système seront principalement les composantes d'une course, c'est-à-dire les étapes, les points kilométriques, … et les éléments concernant les coureurs : équipes, dossards, classement,…

f) Périmètre du système

L'application prévoit de contenir les informations de plusieurs courses du Tour de France sur plusieurs années. Il est possible de consulter les caractéristiques des coureurs et de leur équipe bien évidemment (de plus, le changement d'équipe par un joueur d'une année à l'autre est géré par notre système).

Pour chaque course, le détail du parcours est disponible avec les étapes ainsi que les points kilométriques qui les composent. L'attribution des points selon les différents barèmes établis par les organisateurs est également gérée entièrement par notre système.

Cependant, certains points du Tour ne sont pas pris en charge, notamment l'aspect « guide touristique » (description des points remarquables des communes franchies par exemple). Egalement, il n'est pas possible de rédiger des journaux d'étape comme ceux disponibles sur le site officiel du Tour de France. Enfin, une gestion minimale de la célèbre Caravane du Tour a été mise en place et nécessiterait un approfondissement.

g) Extensions possibles

Le temps limité qu'il nous était accordé pour la réalisation de ce projet ainsi que quelques lacunes dans la gestion d'une base de données ne nous ont pas permis de mettre en œuvre la totalité de nos idées initiales, c'est pourquoi certains éléments de notre système pourraient faire l'objet d'une amélioration.

Autour de la course en elle-même, de nombreux évènements périphériques doivent être gérés par les organisateurs et auraient pu être intégrés à notre système :

- Comme indiqué précédemment, la gestion de la Caravane du Tour pourrait notamment être revue et complétée par les heures et l'ordre de passage des différentes caravanes par exemple ;
- Une meilleure gestion des villes traversées par le Tour pourrait être implémentée ; cela permettrait la création d'un guide touristique (avec les lieux remarquables) et une gestion des fermetures de routes nécessaires pour faciliter la tâche aux organisateurs et municipalités ;
- Le contrôle antidopage étant une priorité pour les organisateurs, il aurait été intéressant de l'intégrer à notre système : heure des prises de sang, stockage des résultats, etc. ;
- Une gestion des équipes techniques et médias autorisés à circuler sur le Tour (voitures balai, caméramen, journalistes, médecins…) reste également à intégrer à notre solution.

3. DICTIONNAIRE DE DONNEES

Le dictionnaire des différentes données utilisées dans notre système est donné dans le tableau ci-dessous, classé par ordre lexicographique. *(I = Identifiant, O = Obligatoire, C = Valeur calculée)*

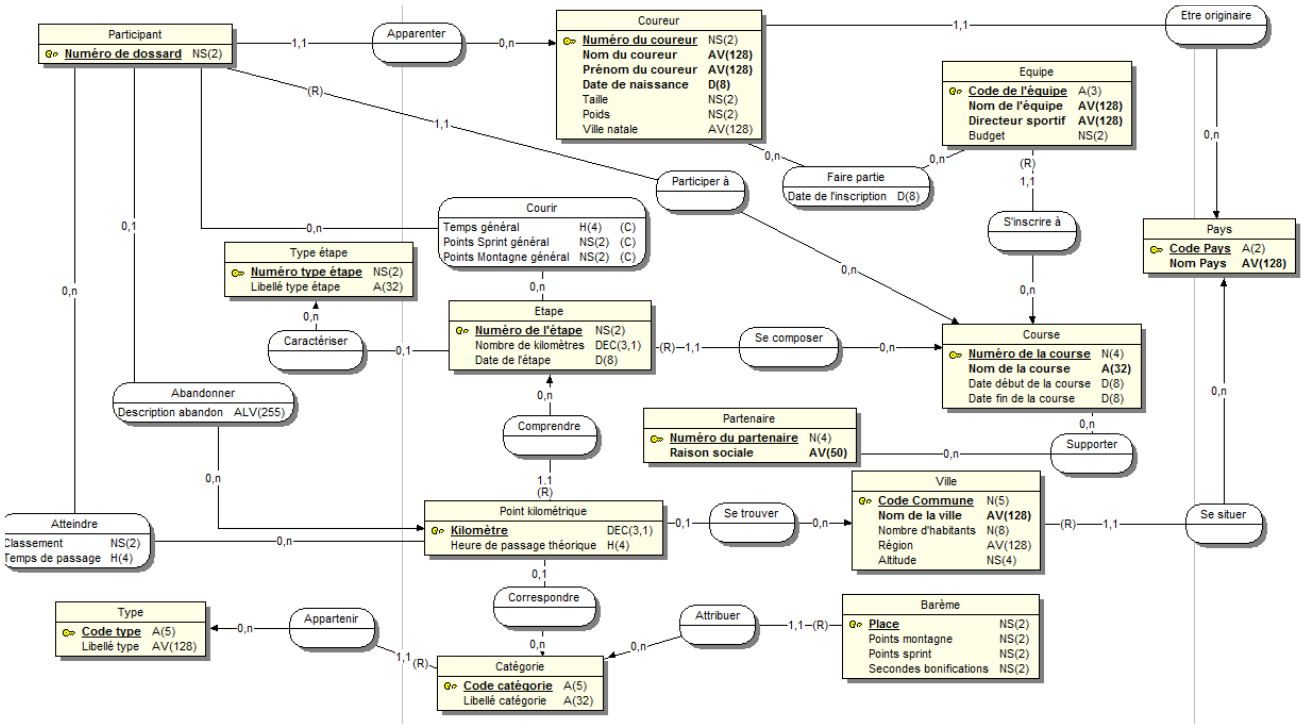
Nom conceptuel	Alias	Type WD	I	O	C	Entité
Altitude	ALTITUDE_VILLE	NS (4,)				Ville
Budget	BUDGET_EQUIPE	NS (2,)				Equipe
Classement	CLASSEMENT	NS (2,)				Atteindre
Code catégorie	CODE_CATEGORIE	A (5,)	**X**	**X**		Catégorie
Code Commune	CODE_COMMUNE	N (5,)	**X**	**X**		Ville
Code de l'équipe	CODE_EQUIPE	A (3,)	**X**	**X**		Equipe
Code Pays	CODE_PAYS	A (2,)	**X**	**X**		Pays
Code type	CODE_TYPE	A (5,)	**X**	**X**		Type
Date de l'étape	DATE_ETAPE	D (8,)				Etape
Date de l'inscription	DATE_INSCRIPTION_EQUIPE	D (8,)				Faire partie
Date de naissance	DATE_NAISSANCE_COUREUR	D (8,)		**X**		Coureur
Date début de la course	DEBUT_COURSE	D (8,)				Course
Date fin de la course	FIN_COURSE	D (8,)				Course
Description abandon	DESC_ABANDON	ALV (255,)				Abandonner
Directeur sportif	DIRECTEUR_SPORTIF_EQUIPE	AV (128,)		**X**		Equipe
Heure de passage théorique	HEURE_PASSAGE_POINT_KM	H (4,)				Point kilométrique
Kilomètre	KM	DEC (3,1)	**X**	**X**		Point kilométrique
Libellé catégorie	LIBELLE_CATEGORIE	A (32,)				Catégorie
Libellé type	LIBELLE_TYPE	AV (128,)				Type
Libellé type étape	LIBELLE_TYPE_ETAPE	A (32,)				Type étape
Nom de l'équipe	NOM_EQUIPE	AV (128,)		**X**		Equipe

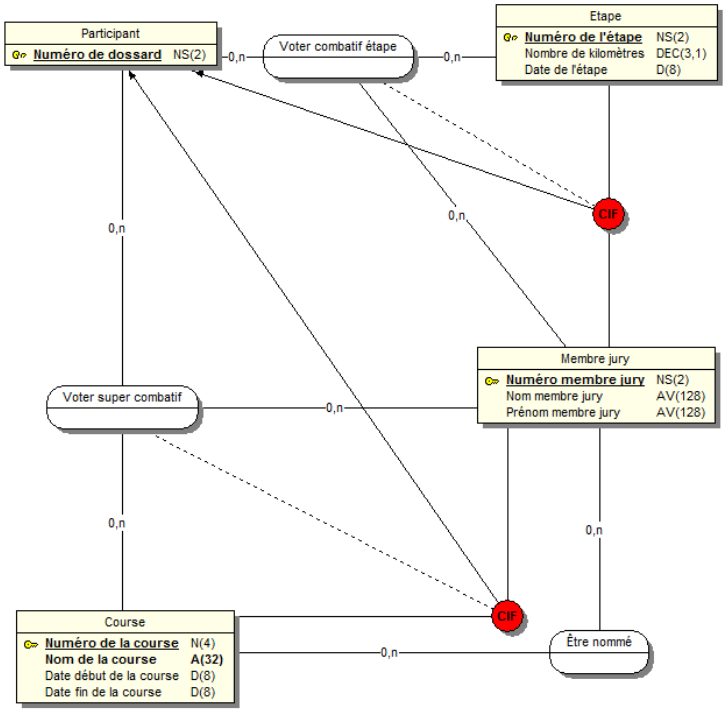
Nom de la course	NOM_COURSE	A (32,)		**O X**		Course
Nom de la ville	NOM_VILLE	AV (128,)		**O X**		Ville
Nom du coureur	NOM_COUREUR	AV (128,)		**O X**		Coureur
Nom membre jury	NOM_MEMBRE_JURY	AV (128,)		**O**		Membre jury
Nom Pays	NOM_PAYS	AV (128,)		**O X**		Pays
Nombre d'habitants	NB_HABITANTS_VILLE	N (8,)				Ville
Nombre de kilomètres	NB_KM_ETAPE	DEC (3,1)				Etape
Numéro de dossard	NUM_DOSSARD	NS (2,)	**I X**	**O X**		Participant
Numéro de l'étape	NUM_ETAPE	NS (2,)	**I X**	**O X**		Etape
Numéro de la course	NUM_COURSE	N (4,)	**I X**	**O X**		Course
Numéro du coureur	NUM_COUREUR	NS (2,)	**I X**	**O X**		Coureur
Numéro du partenaire	NUM_PARTENAIRE	N (4,)	**I X**	**O X**		Partenaire
Numéro membre jury	NUM_MEMBRE_JURY	NS (2,)	**I X**	**O X**		Membre jury
Numéro type étape	NUM_TYPE_ETAPE	NS (2,)	**I X**	**O X**		Type étape
Place	PLACE_BAREME	NS (2,)	**I X**	**X**		Barème
Poids	POIDS_COUREUR	NS (2,)				Coureur
Points montagne	PTS_MONTAGNE_BAREME	NS (2,)				Barème
Points Montagne général	POINTS_MONTAGNE_GENERAL	NS (2,)			**C X**	Courir
Points sprint	PTS_SPRINT_BAREME	NS (2,)				Barème
Points Sprint general	POINTS_SPRINT_GENERAL	NS (2,)			**C X**	Courir
Prénom du coureur	PRENOM_COUREUR	AV (128,)		**O X**		Coureur

Prénom membre jury	PRENOM_MEMBRE_JURY	AV (128,)			Membre jury
Raison sociale	NOM_PARTENAIRE	AV (50,)	**X**		Partenaire
Région	REGION_VILLE	AV (128,)			Ville
Secondes bonifications	SEC_BONIF_BAREME	NS (2,)			Barème
Taille	TAILLE_COUREUR	NS (2,)			Coureur
Temps de passage	TEMPS_PASSAGE	H (4,)			Atteindre
Temps general	TEMPS_GENERAL	H (4,)		**C** **X**	Courir
Ville natale	VILLE_NAISSANCE_COUREUR	AV (128,)			Coureur

4. MODELE ENTITE-ASSOCIATION

Le MEA modélisant notre système est présenté ci-dessous. Pour une meilleure lisibilité, nous avons retiré de ce schéma la partie qui concerne le jury et le vote des coureurs combatifs et super combatif. Cette partie est présentée dans un deuxième schéma page suivante.



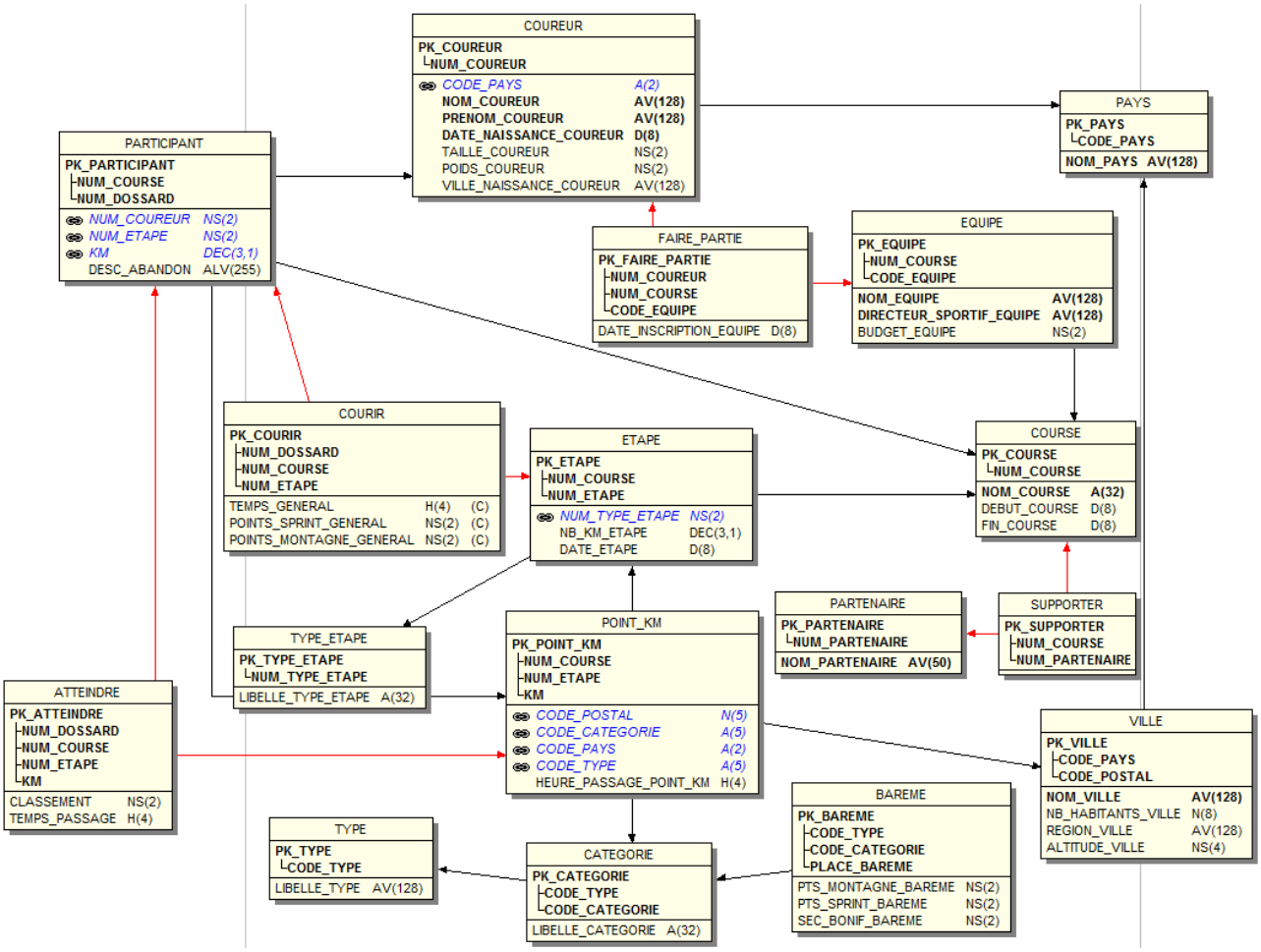


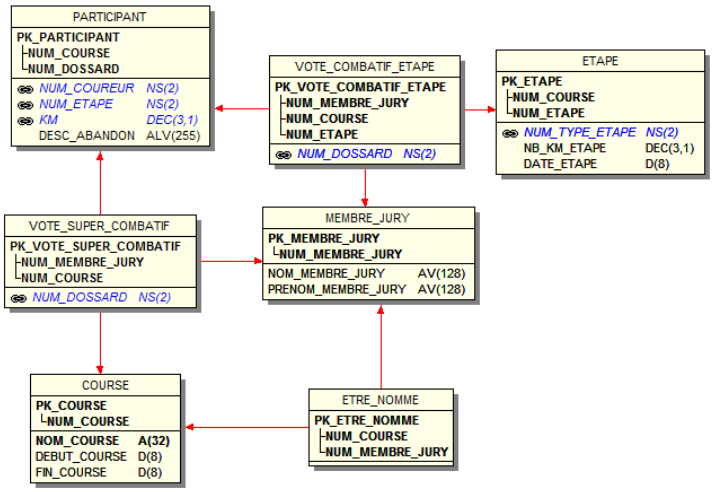
5. MODELISATION LOGIQUE

a) Modèle logique relationnel normalisé Oracle 10g/11g

Une fois le MCD validé, nous avons pu générer le MLR associé à l'aide de Win'Design pour Oracle. Nous avons tout de même apporté quelques modifications nécessaires à cette modélisation générée automatiquement, notamment au niveau de la conversion *types Win'Design > types Oracle*.

Ici encore, notre schéma a été divisé en deux parties, la seconde concernant le vote des combatifs et le jury.


COUREUR
PK_COUREUR
NUM_COUREUR
CODE_PAYS A(2)
NOM_COUREUR AV(128)
PRENOM_COUREUR AV(128)
DATE_NAISSANCE_COUREUR D(8)
TAILLE_COUREUR NS(2)
POIDS_COUREUR NS(2)
VILLE_NAISSANCE_COUREUR AV(128)
PAYS
PK_PAYS
CODE_PAYS
NOM_PAYS AV(128)
PARTICIPANT
PK_PARTICIPANT
NUM_COURSE
NUM_DOSSARD
NUM_COUREUR NS(2)
NUM_ETAPE NS(2)
KM DEC(3,1)
DESC_ABANDON ALV(255)
FAIRE_PARTIE
PK_FAIRE_PARTIE
NUM_COUREUR
NUM_COURSE
CODE_EQUIPE
DATE_INSCRIPTION_EQUIPE D(8)
EQUIPE
PK_EQUIPE
NUM_COURSE
CODE_EQUIPE
NOM_EQUIPE AV(128)
DIRECTEUR_SPORTIF_EQUIPE AV(128)
BUDGET_EQUIPE NS(2)
COURIR
PK_COURIR
NUM_DOSSARD
NUM_COURSE
NUM_ETAPE
TEMPS_GENERAL H(4) (C)
POINTS_SPRINT_GENERAL NS(2) (C)
POINTS_MONTAGNE_GENERAL NS(2) (C)
ETAPE
PK_ETAPE
NUM_COURSE
NUM_ETAPE
NUM_TYPE_ETAPE NS(2)
NB_KM_ETAPE DEC(3,1)
DATE_ETAPE D(8)
COURSE
PK_COURSE
NUM_COURSE
NOM_COURSE A(32)
DEBUT_COURSE D(8)
FIN_COURSE D(8)
PARTENAIRE
PK_PARTENAIRE
NUM_PARTENAIRE
NOM_PARTENAIRE AV(50)
SUPPORTER
PK_SUPPORTER
NUM_COURSE
NUM_PARTENAIRE
TYPE_ETAPE
PK_TYPE_ETAPE
NUM_TYPE_ETAPE
LIBELLE_TYPE_ETAPE A(32)
POINT_KM
PK_POINT_KM
NUM_COURSE
NUM_ETAPE
KM
CODE_POSTAL N(5)
CODE_CATEGORIE A(5)
CODE_PAYS A(2)
CODE_TYPE A(5)
HEURE_PASSAGE_POINT_KM H(4)
ATTEINDRE
PK_ATTEINDRE
NUM_DOSSARD
NUM_COURSE
NUM_ETAPE
KM
CLASSEMENT NS(2)
TEMPS_PASSAGE H(4)
VILLE
PK_VILLE
CODE_PAYS
CODE_POSTAL
NOM_VILLE AV(128)
NB_HABITANTS_VILLE N(8)
REGION_VILLE AV(128)
ALTITUDE_VILLE NS(4)
BAREME
PK_BAREME
CODE_TYPE
CODE_CATEGORIE
PLACE_BAREME
PTS_MONTAGNE_BAREME NS(2)
PTS_SPRINT_BAREME NS(2)
SEC_BONIF_BAREME NS(2)
TYPE
PK_TYPE
CODE_TYPE
LIBELLE_TYPE AV(128)
CATEGORIE
PK_CATEGORIE
CODE_TYPE
CODE_CATEGORIE
LIBELLE_CATEGORIE A(32)


PARTICIPANT
PK_PARTICIPANT
NUM_COURSE
NUM_DOSSARD
NUM_COUREUR NS(2)
NUM_ETAPE NS(2)
KM DEC(3,1)
DESC_ABANDON ALV(255)
VOTE_COMBATIF_ETAPE
PK_VOTE_COMBATIF_ETAPE
NUM_MEMBRE_JURY
NUM_COURSE
NUM_ETAPE
NUM_DOSSARD NS(2)
ETAPE
PK_ETAPE
NUM_COURSE
NUM_ETAPE
NUM_TYPE_ETAPE NS(2)
NB_KM_ETAPE DEC(3,1)
DATE_ETAPE D(8)
VOTE_SUPER_COMBATIF
PK_VOTE_SUPER_COMBATIF
NUM_MEMBRE_JURY
NUM_COURSE
NUM_DOSSARD NS(2)
MEMBRE_JURY
PK_MEMBRE_JURY
NUM_MEMBRE_JURY
NOM_MEMBRE_JURY AV(128)
PRENOM_MEMBRE_JURY AV(128)
COURSE
PK_COURSE
NUM_COURSE
NOM_COURSE A(32)
DEBUT_COURSE D(8)
FIN_COURSE D(8)
ETRE_NOMME
PK_ETRE_NOMME
NUM_COURSE
NUM_MEMBRE_JURY

b) Modèle logique relationnel optimisé

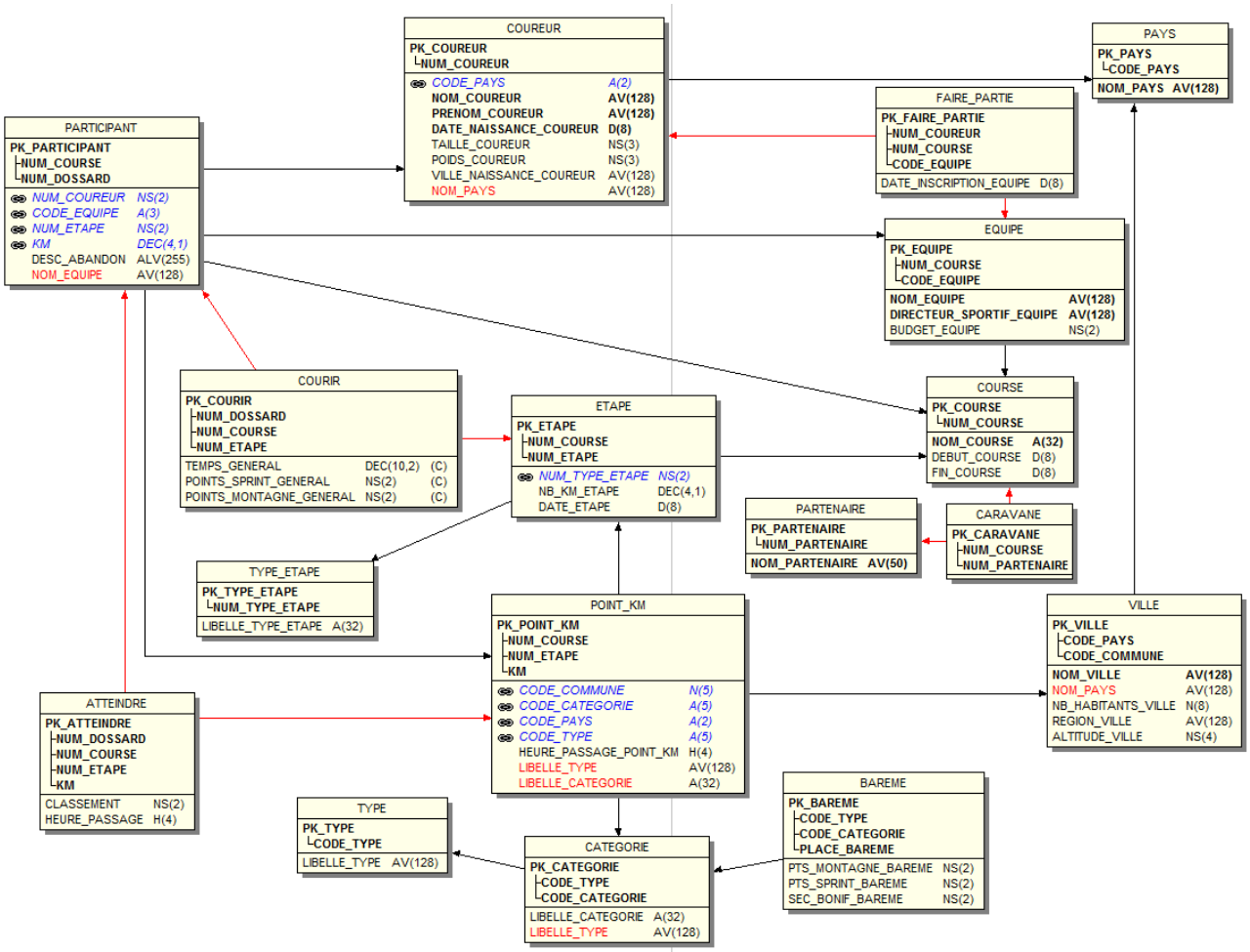
Un modèle normalisé pose généralement des problèmes de performances dans l'accès aux données de la base (insertion, consultation,…). C'est pourquoi il est nécessaire dans l'étape suivante de chercher à optimiser le MLR précédemment réalisé. C'est ainsi que nous obtenons le modèle suivant :

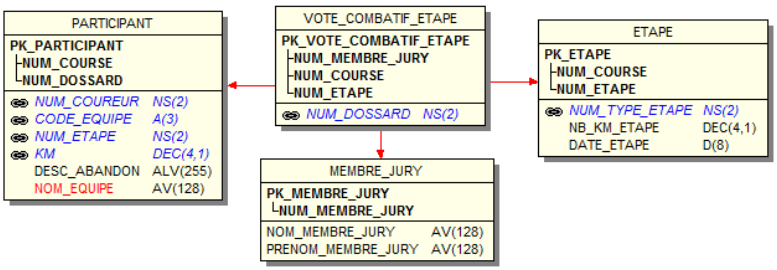
La table ETRE_NOMME a notamment été supprimée, elle ne contenait que des clés primaires d'autres tables (num_course et num_membre_jury).

De même pour la table VOTE_SUPER_COMBATIF, qui a été fusionnée avec VOTE_COMBATIF_ETAPE. L'étape numéro 0 sera tout simplement attribuée au numéro d'étape associé au vote du super combatif.

La redondance a été mise en place dans certains endroits stratégiques, comme nom_pays dans COUREUR, nom_equipe dans PARTICIPANT, libelle_type et libelle_categorie dans POINT_KM par exemple.

Les modèles suivants correspondent à la modélisation finale de notre système.





c) Optimisation spécifique à Oracle

Oracle propose des outils supplémentaires pour aller plus loin dans l'optimisation de la base. Nous avons mis en place par exemple un système de partitionnement sur les tables qui comportaient beaucoup de données. C'est le cas des tables ETAPE, COURIR, ATTEINDRE, POINT_KM et PARTICIPANT, que nous avons partitionnées en fonction du numéro de course (qui correspond à l'année du Tour), ce qui était le choix de partitionnement le plus logique.

Nous sommes allés encore plus loin avec les tables PARTICIPANT, POINT_KM et ATTEINDRE qui ont été créées en tables IOT.

6. INTERFACE HOMME-MACHINE

a) Maquettes

Les maquettes de notre système ont toutes été réalisées en html, ceci dans le but de pouvoir les réutiliser comme « base » pour notre système final. Trois maquettes sont présentées dans notre dossier projet, dans le sous-dossier Maquettes : *course.html*, *etape.html* et *leaderboards.html*. Elles s'appuient complètement sur la modélisation finale et optimisée de notre base.

b) Scripts PL/SQL

Concernant le code PL/SQL pour la création de l'interface homme-machine, les règles standards d'organisation ont été appliquées : les packages correspondant aux accès à la base ont été séparés des packages gérant l'affichage graphique (le nom du fichier du package porte la mention *ui* ou *db* pour déterminer rapidement s'il s'agit de l'un ou de l'autre). De plus, le corps des fonctions a été séparé de la déclaration des packages, c'est-à-dire que les instructions CREATE OR REPLACE PACKAGE et CREATE OR REPLACE PACKAGE BODY se trouvent dans des fichiers différents : ici encore, le contenu du nom de fichier sera différent, il contiendra *pqi* pour le premier, et *pqb* pour le second.

c) Exemples

Voici un aperçu du script de gestion des étapes. Il est décomposé en 4 fichiers :

- 710_create_pqi_db_tdf_etape : Accès à la base – en tête

```
  CREATE OR REPLACE PACKAGE "G16_TDF"."PA_ETAPE"
IS
       TYPE sqlcur IS REF CURSOR;
       FUNCTION GetAll return sqlcur;
       FUNCTION Get(cnum in number, enum in number) return ETAPE%rowtype;
  FUNCTION GetNbEtapesForCourse(cnum number) return number;
       FUNCTION GetForCourse(cnum in number) return sqlcur;
END;
```

- 710_create_pqb_db_tdf_etape : Accès à la base – corps

```
CREATE OR REPLACE PACKAGE BODY "G16_TDF"."PA_ETAPE" IS
       FUNCTION GetAll return sqlcur
       IS
       res sqlcur;
       BEGIN
              OPEN res FOR
                     SELECT *
                     FROM ETAPE
      ORDER BY NUM_ETAPE;
              return res;
       END;
       FUNCTION Get(cnum in number, enum in number) return ETAPE%rowtype
       IS
       res ETAPE%rowtype;
       BEGIN
              SELECT * into res
              FROM ETAPE
              WHERE NUM_COURSE = cnum
    AND NUM_ETAPE = enum
    ORDER BY NUM_ETAPE;
              return res;
       END;
  FUNCTION GetNbEtapesForCourse(cnum number) return number
  IS
  res number;
  BEGIN
    SELECT count(*) into res
    FROM ETAPE
    WHERE NUM_COURSE=cnum;
    return res;
  END;
       FUNCTION GetForCourse(cnum in number) return sqlcur
       IS
       res sqlcur;
       BEGIN
              OPEN res FOR
                     SELECT *
                     FROM ETAPE
      WHERE NUM_COURSE = cnum
      ORDER BY NUM_ETAPE;
    RETURN res;
       END;
END;
```

- 600_create_pqi_ui_tdf_etape : Interface graphique – en-tête

```
CREATE OR REPLACE PACKAGE "G16_TDF"."UI_ETAPE" IS
  TYPE sqlcur IS REF CURSOR;
  PROCEDURE AffTabForCourse(cnum number);
  PROCEDURE AffTabForEtape(num_etape number);
END UI_ETAPE;
```

- 600_create_pqb_ui_tdf_etape : Interface graphique – corps

```
CREATE OR REPLACE PACKAGE BODY "G16_TDF"."UI_ETAPE" IS
  PROCEDURE AffTabForCourse(cnum number)
  IS
    etapes sqlcur;
    rec_etape etape%rowtype;
    rec_typeetape type_etape%rowtype;
    rec_villedepart ville%rowtype;
    rec_villearrivee ville%rowtype;
  BEGIN
    htp.header(2, 'Liste des étapes du Tour ' || cnum || ' :');

    htp.tableOpen;
    htp.tableheader ('Etape');
    htp.tableheader ('Départ-Arrivée');
    htp.tableheader ('Type');
    htp.tableheader ('Nombre de KMs');
    htp.tableheader ('Date');

    etapes := pa_etape.GetForCourse(cnum);
    fetch etapes into rec_etape;
    while(etapes%FOUND)
    loop
      rec_typeetape := pa_type_etape.Get(rec_etape.num_type_etape);

      rec_villedepart := pa_ville.GetVilleDepartForEtape(cnum, rec_etape.num_etape);
      rec_villearrivee := pa_ville.GetVilleArriveeForEtape(cnum, rec_etape.num_etape);

      htp.tableRowOpen;
      htp.tableData( rec_etape.num_etape);
      htp.tableData( htf.anchor('etape_detail?num_etape=' || rec_etape.num_etape,
rec_villedepart.nom_ville || '-' || rec_villearrivee.nom_ville));
      htp.tableData( rec_typeetape.libelle_type_etape);
      htp.tableData( rec_etape.nb_km_etape );
      htp.tableData( 'Le ' || TO_CHAR(rec_etape.date_etape, 'DD/MM/YYYY') || ' à ' ||
TO_CHAR(rec_etape.date_etape, 'HH24:MI'));

      htp.tableRowClose;
      fetch etapes into rec_etape;
    end loop;

    htp.br;
    htp.tableclose;
  END;

  PROCEDURE AffTabForEtape(num_etape number)
  IS
    numcourse number(4);
    rec_typeetape type_etape%rowtype;
    rec_etape etape%rowtype;
    rec_villedepart ville%rowtype;
    rec_villearrivee ville%rowtype;
  BEGIN
    numcourse := TO_NUMBER(ui_commun.getcookievalue('course'));
```

```
    rec_etape := pa_etape.Get(numcourse, num_etape);

    rec_typeetape := pa_type_etape.Get(rec_etape.num_type_etape);
    rec_villedepart := pa_ville.GetVilleDepartForEtape(numcourse, num_etape);
    rec_villearrivee := pa_ville.GetVilleArriveeForEtape(numcourse, num_etape);

    htp.header(2, 'Etape ' || num_etape || ' : ' || rec_typeetape.libelle_type_etape);

    htp.bold('Date de début de l''étape : ');
    htp.print(TO_CHAR(rec_etape.date_etape, 'DD/MM/YYYY') || ' à ' ||
TO_CHAR(rec_etape.date_etape, 'HH24:MI'));
    htp.br;
    htp.bold('Nombre de kilomètres : ');
    htp.print(rec_etape.nb_km_etape || ' kilomètres');
    htp.br;
    htp.br;

    htp.header(3, 'Ville de départ : ' || rec_villedepart.nom_ville);
    htp.bold('Code postal : ');
    htp.print(rec_villedepart.code_commune);
    htp.br;
    htp.bold('Région : ');
    htp.print(rec_villedepart.region_ville);
    htp.br;
    htp.bold('Pays : ');
    htp.print(rec_villedepart.nom_pays);
    htp.br;
    htp.br;

    […]

    htp.header(3, 'Points kilométriques');
    UI_POINT_KM.AffTabForCourseForEtape(numcourse, num_etape);

    htp.header(3, 'Abandons');
    UI_PARTICIPANT.AffTabAbForCourseForEtape(numcourse, num_etape);
  END;
END UI_ETAPE;
```

CONCLUSION

Bien que notre système souffre de quelques imperfections, il permet néanmoins de gérer les éléments principaux qui constituent le Tour de France avec une certaine efficacité.

Il nous a permis de découvrir le système de gestion Oracle qui était inconnu pour tous les membres de notre groupe. Nous avons ainsi pu utiliser des technologies qui lui sont propres et qui sont un bon exemple de sa puissance, comme le PL/SQL, les tables IOT, ou encore Apex.

Mais avant de se lancer dans la création de base, nous avons pu voir que de nombreuses étapes sont préalablement nécessaires : le brainstorming bien sûr, afin de se mettre d'accord sur les bases de notre système, mais aussi et surtout les modélisations conceptuelle et logique. Ces dernières demandent du temps et une attention particulière afin que notre système soit bien conçu et s'adapte parfaitement au SGBD utilisé, Oracle dans notre cas.

Nous pouvons ainsi retenir de cette expérience qu'il est indispensable de ne griller aucune étape et ne pas aller dans la précipitation, car il est difficile de revenir en arrière une fois que les choix de conception ont été faits. Un modèle clair et bien conçu au départ facilite ensuite la création de l'interface homme-machine d'une part (que ce soit avec Apex ou le PL/SQL), et la prise en main pour les utilisateurs finaux d'autre part (l'administrateur du Tour de France qui va être chargé de modifier le contenu de la base *via* l'interface, ou bien les simples utilisateurs qui voudront consulter les informations qu'elle contient).