Secteur Tertiaire Informatique

Filière étude - développement

Activité Développer la persistance des données

Rapport de base de données

Romain Heller

Afpa 2014.

Un grand MERCI à toutes les personnes qui m'ont permis de revenir vers ce métier passionnant qu'est l'Informatique. Sans eux, je livrerai encore des journaux, quelque soit le temps !

Merci au Personnel de l'Afpa, qui ont su gérer nos petits tracas, et qui ont facilité notre quotidien.

Merci à Gilbert dont l'enseignement m'a redonné l'envie d'apprendre tout en me confortant dans mon désir de re-devenir informaticien, pour sa vision globale et parfois engagée des choses de la vie, ainsi que pour toutes ces heures passées à essayer de comprendre nos erreurs !

Merci à ma fille, dont un simple regard repose mes yeux, même après 12 heures devant un écran !

Table des matières

1 Cahier des charges

Une association de groupes folkloriques souhaite mettre en place une application permettant de gérer les groupes qui lui sont affiliés ainsi que les rencontres culturelles où les groupes peuvent participer.

Vous trouverez le Cahier des charges en Annexe cpp1-1.

2 Modéliser les données

2.1 Le dictionnaire des données

Nom	***Type***	***Rattachement***
adr_cp	AV(12,)	adresses
adr_label	AV(128,)	adresses
adr_pays	A(32,)	adresses
adr_ville	AV(128,)	adresses
adresseType_label	AV(32,)	adresseTypes
adresseType_usage	AV(128,)	adresseTypes
caracteristiques	AV(32,)	groupes
civilite_abrv	AV(32,)	civilites
civilite_label	AV(128,)	civilites
civilite_usage	AV(256,)	civilites
date_apparition_oeuvre	D(8,)	oeuvres
date_debut	D(8,)	rencontres
date_entree	D(8,)	est_membre
date_fin	D(8,)	rencontres
date_naissance	D(8,)	individus
date_passage	A(32,)	programme
date_sortie	D(8,)	est_membre
denomination	AV(32,)	groupes
duree_oeuvre	H(4,)	oeuvres
duree_prevue	H(4,)	se_compose_de
email_label	AV(128,)	emails
emailType_label	AV(128,)	emailTypes
emailType_usage	AV(128,)	emailTypes

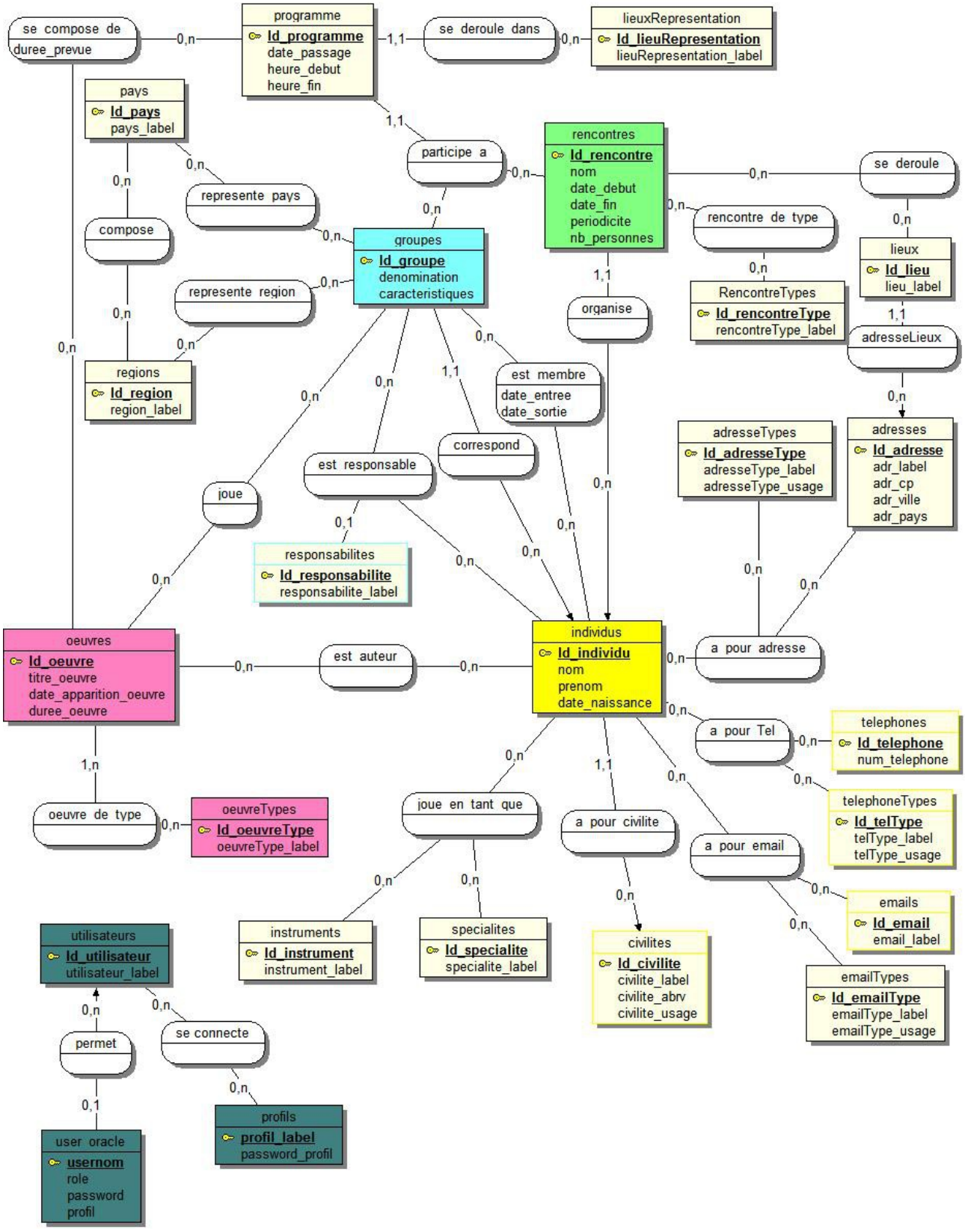
Nom	***Type***	***Rattachement***
heure_debut	H(4,)	programme
heure_fin	H(4,)	programme
Id_adresse	NS(2,)	adresses
Id_adresseType	NS(2,)	adresseTypes
Id_civilite	NS(2,)	civilitesi
Id_email	NS(2,)	emails
Id_emailType	NS(2,)	emailTypes
Id_groupe	NS(2,)	groupes
Id_individu	NS(2,)	individus
Id_instrument	NS(2,)	instruments
Id_lieu	NS(2,)	lieux
Id_lieuRepresentation	NS(2,)	lieuxRepresentation
Id_oeuvre	NS(32,)	oeuvres
Id_oeuvreType	NS(2,)	oeuvreTypes
Id_pays	NS(3,)	pays
Id_programme	NS(2,)	programme
Id_region	NS(2,)	regions
Id_rencontre	NS(2,)	rencontres
Id_rencontreType	NS(2,)	RencontreTypes
Id_responsabilite	NS(2,)	responsabilites
Id_specialite	NS(2,)	specialites
Id_telephone	NS(2,)	telephones
Id_telType	NS(2,)	telephoneTypes
Id_utilisateur	NS(2,)	utilisateurs
instrument_label	AV(32,)	instruments
lieu_label	AV(32,)	lieux
lieuRepresentation_label	AV(32,)	lieuxRepresentation
nb_personnes	NS(8,)	rencontres
usernom	A(32,)	user_oracle
nom	AV(32,)	individus
nom	AV(32,)	rencontres
num_telephone	AV(15,)	telephones
oeuvreType_label	AV(32,)	oeuvreTypes

Nom	***Type***	***Rattachement***
password	A(32,)	user_oracle
password_profil	A(32,)	profils
pays_label	AV(32,)	pays
periodicite	AV(12,)	rencontres
prenom	AV(32,)	individus
profil	A(32,)	user_oracle
profil_label	AV(128,)	profils
region_label	AV(32,)	regions
rencontreType_label	AV(32,)	RencontreTypes
responsabilite_label	AV(32,)	responsabilites
role	A(32,)	user_oracle
specialite_label	AV(32,)	specialites
telType_label	AV(12,)	telephoneTypes
telType_usage	AV(128,)	telephoneTypes
titre_oeuvre	AV(128,)	oeuvres
utilisateur_label	AV(32,)	utilisateurs

2.2 Les règles de gestion

- Interrogation des groupes jouant un titre donné.
- Interrogation des rencontres où un titre a été interprété.
- Interrogation des membres ayant une spécialité donnée pour une rencontre donnée.
- Interrogation des titres de plus de x minutes pour un pays ou une région donnés.
- Interrogation des rencontres ayant eu n groupes participants.
- Interrogation des rencontres où on a joué d'un instrument donné

2.3 MCD


se compose de
duree_prevue
programme
Id_programme
date_passage
heure_debut
heure_fin
se deroule dans
lieuxRepresentation
Id_lieuRepresentation
lieuRepresentation_label
pays
Id_pays
pays_label
participe a
rencontres
Id_rencontre
nom
date_debut
date_fin
periodicite
nb_personnes
se deroule
represente pays
compose
groupes
Id_groupe
denomination
caracteristiques
rencontre de type
RencontreTypes
Id_rencontreType
rencontreType_label
lieux
Id_lieu
lieu_label
represente region
organise
adresseLieux
regions
Id_region
region_label
est membre
date_entree
date_sortie
adresseTypes
Id_adresseType
adresseType_label
adresseType_usage
adresses
Id_adresse
adr_label
adr_cp
adr_ville
adr_pays
correspond
est responsable
joue
responsabilites
Id_responsabilite
responsabilite_label
oeuvres
Id_oeuvre
titre_oeuvre
date_apparition_oeuvre
duree_oeuvre
est auteur
individus
Id_individu
nom
prenom
date_naissance
a pour adresse
a pour Tel
telephones
Id_telephone
num_telephone
telephoneTypes
Id_telType
telType_label
telType_usage
oeuvre de type
oeuvreTypes
Id_oeuvreType
oeuvreType_label
joue en tant que
a pour civilite
a pour email
emails
Id_email
email_label
instruments
Id_instrument
instrument_label
specialites
Id_specialite
specialite_label
civilites
Id_civilite
civilite_label
civilite_abrv
civilite_usage
emailTypes
Id_emailType
emailType_label
emailType_usage
utilisateurs
Id_utilisateur
utilisateur_label
permet
se connecte
profils
profil_label
password_profil
user oracle
usernom
role
password
profil

2.3.1 Contraintes d'intégrité

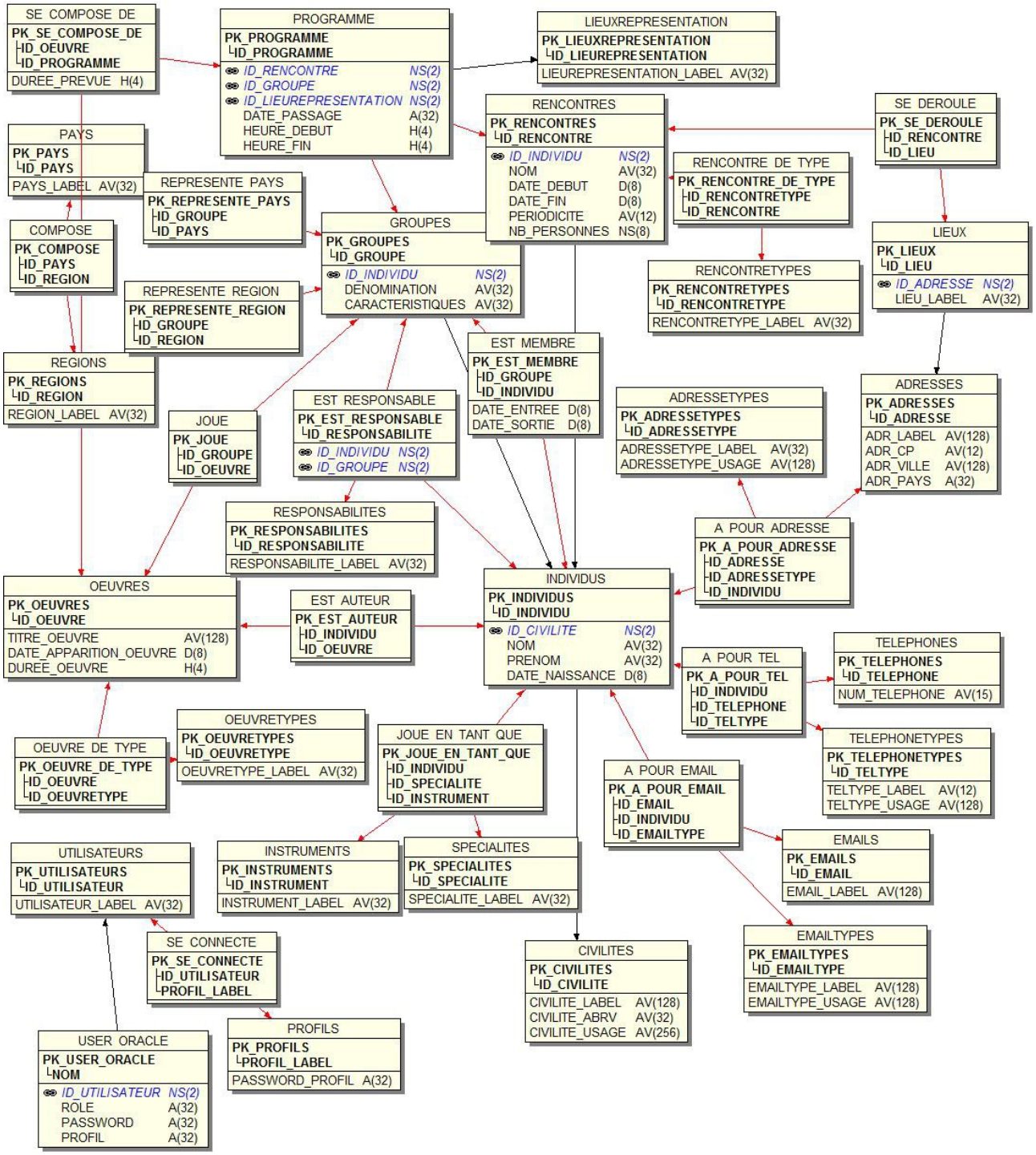
La modélisation essaye de respecter au maximum les formes normales. En particulier :

- Aucun des attributs participant à une clef primaire ne peut avoir la valeur null.
- Toutes les valeurs d'une clef étrangère se retrouvent comme valeur de la clef primaire de la relation référée (ensemble des valeurs de la clef étrangère est **inclus** au sens large dans l'ensemble des valeurs de la clef primaire).

2.3.2 Dépendances fonctionnelles

Le respect de la 3ème forme normale nous protège des dépendances entre les attributs ne faisant pas partie de la clé primaire.

2.4 MLR


SE COMPOSE DE
PK_SE_COMPOSE_DE
ID_OEUVRE
ID_PROGRAMME
DUREE_PREVUE H(4)
PROGRAMME
PK_PROGRAMME
ID_PROGRAMME
ID_RENCONTRE NS(2)
ID_GROUPE NS(2)
ID_LIEUREPRESENTATION NS(2)
DATE_PASSAGE A(32)
HEURE_DEBUT H(4)
HEURE_FIN H(4)
LIEUXREPRESENTATION
PK_LIEUXREPRESENTATION
ID_LIEUREPRESENTATION
LIEUREPRESENTATION_LABEL AV(32)
RENCONTRES
PK_RENCONTRES
ID_RENCONTRE
ID_INDIVIDU NS(2)
NOM AV(32)
DATE_DEBUT D(8)
DATE_FIN D(8)
PERIODICITE AV(12)
NB_PERSONNES NS(8)
SE DEROULE
PK_SE_DEROULE
ID_RENCONTRE
ID_LIEU
PAYS
PK_PAYS
ID_PAYS
PAYS_LABEL AV(32)
REPRESENTE PAYS
PK_REPRESENTE_PAYS
ID_GROUPE
ID_PAYS
RENCONTRE DE TYPE
PK_RENCONTRE_DE_TYPE
ID_RENCONTRETYPE
ID_RENCONTRE
COMPOSE
PK_COMPOSE
ID_PAYS
ID_REGION
GROUPES
PK_GROUPES
ID_GROUPE
ID_INDIVIDU NS(2)
DENOMINATION AV(32)
CARACTERISTIQUES AV(32)
LIEUX
PK_LIEUX
ID_LIEU
ID_ADRESSE NS(2)
LIEU_LABEL AV(32)
RENCONTRETYPES
PK_RENCONTRETYPES
ID_RENCONTRETYPE
RENCONTRETYPE_LABEL AV(32)
REPRESENTE REGION
PK_REPRESENTE_REGION
ID_GROUPE
ID_REGION
EST MEMBRE
PK_EST_MEMBRE
ID_GROUPE
ID_INDIVIDU
DATE_ENTREE D(8)
DATE_SORTIE D(8)
REGIONS
PK_REGIONS
ID_REGION
REGION_LABEL AV(32)
ADRESSETYPES
PK_ADRESSETYPES
ID_ADRESSETYPE
ADRESSETYPE_LABEL AV(32)
ADRESSETYPE_USAGE AV(128)
ADRESSES
PK_ADRESSES
ID_ADRESSE
ADR_LABEL AV(128)
ADR_CP AV(12)
ADR_VILLE AV(128)
ADR_PAYS A(32)
JOUE
PK_JOUE
ID_GROUPE
ID_OEUVRE
EST RESPONSABLE
PK_EST_RESPONSABLE
ID_RESPONSABILITE
ID_INDIVIDU NS(2)
ID_GROUPE NS(2)
RESPONSABILITES
PK_RESPONSABILITES
ID_RESPONSABILITE
RESPONSABILITE_LABEL AV(32)
A POUR ADRESSE
PK_A_POUR_ADRESSE
ID_ADRESSE
ID_ADRESSETYPE
ID_INDIVIDU
OEUVRES
PK_OEUVRES
ID_OEUVRE
TITRE_OEUVRE AV(128)
DATE_APPARITION_OEUVRE D(8)
DUREE_OEUVRE H(4)
EST AUTEUR
PK_EST_AUTEUR
ID_INDIVIDU
ID_OEUVRE
INDIVIDUS
PK_INDIVIDUS
ID_INDIVIDU
ID_CIVILITE NS(2)
NOM AV(32)
PRENOM AV(32)
DATE_NAISSANCE D(8)
A POUR TEL
PK_A_POUR_TEL
ID_INDIVIDU
ID_TELEPHONE
ID_TELTYPE
TELEPHONES
PK_TELEPHONES
ID_TELEPHONE
NUM_TELEPHONE AV(15)
OEUVRETYPES
PK_OEUVRETYPES
ID_OEUVRETYPE
OEUVRETYPE_LABEL AV(32)
OEUVRE DE TYPE
PK_OEUVRE_DE_TYPE
ID_OEUVRE
ID_OEUVRETYPE
JOUE EN TANT QUE
PK_JOUE_EN_TANT_QUE
ID_INDIVIDU
ID_SPECIALITE
ID_INSTRUMENT
TELEPHONETYPES
PK_TELEPHONETYPES
ID_TELTYPE
TELTYPE_LABEL AV(12)
TELTYPE_USAGE AV(128)
A POUR EMAIL
PK_A_POUR_EMAIL
ID_EMAIL
ID_INDIVIDU
ID_EMAILTYPE
UTILISATEURS
PK_UTILISATEURS
ID_UTILISATEUR
UTILISATEUR_LABEL AV(32)
INSTRUMENTS
PK_INSTRUMENTS
ID_INSTRUMENT
INSTRUMENT_LABEL AV(32)
SPECIALITES
PK_SPECIALITES
ID_SPECIALITE
SPECIALITE_LABEL AV(32)
EMAILS
PK_EMAILS
ID_EMAIL
EMAIL_LABEL AV(128)
SE CONNECTE
PK_SE_CONNECTE
ID_UTILISATEUR
PROFIL_LABEL
CIVILITES
PK_CIVILITES
ID_CIVILITE
CIVILITE_LABEL AV(128)
CIVILITE_ABRV AV(32)
CIVILITE_USAGE AV(256)
EMAILTYPES
PK_EMAILTYPES
ID_EMAILTYPE
EMAILTYPE_LABEL AV(128)
EMAILTYPE_USAGE AV(128)
USER ORACLE
PK_USER_ORACLE
NOM
ID_UTILISATEUR NS(2)
ROLE A(32)
PASSWORD A(32)
PROFIL A(32)
PROFILS
PK_PROFILS
PROFIL_LABEL
PASSWORD_PROFIL A(32)

3 Les utilisateurs et les droits

On peut se connecter à la Base de Donnée avec trois comptes utilisateurs ayant chacun des droits différents.

Les droits sont gérés par le mécanisme fournit par Oracle : Utilisateur / Profil / Rôle./ Privilèges

Un utilisateur se connecte à la Base selon les modalités définies par son profil, et pourra manipuler le contenu de la Base en fonction des privilèges donnés à son Rôle.

Ici, l'utilisation des rôles peut sembler redondante, mais cela simplifiera la gestion des droits si l'application doit se décliner dans plusieurs version (différentes plate-formes, par exemple client Java, site Web, application mobile, ...).

3.1 Les profils

Sous *Oracle*, un *profile* définit les règles d'accès aux ressources de la base.

Les trois profils Administrateur, Utilisateur et Consultant seront donc assez proches, les différences notables portant sur la durée de connexion et le nombre d'essais possibles.

```
-- ----------------------------------------
-- Profiles
-- ----------------------------------------
DROP PROFILE Air2Java_Admin CASCADE;
CREATE PROFILE Air2Java_Admin    LIMIT
        SESSIONS_PER_USER          UNLIMITED
        CPU_PER_SESSION            UNLIMITED
        CPU_PER_CALL               3000
        CONNECT_TIME               45
        LOGICAL_READS_PER_SESSION  DEFAULT
        LOGICAL_READS_PER_CALL     1000
        PRIVATE_SGA                15K
        COMPOSITE_LIMIT            5000000
        ;
```

Pour Air2Java_User on ajoutera :

```
        FAILED_LOGIN_ATTEMPTS 5
        PASSWORD_LOCK_TIME 1
```

Pour Air2Java_Consult on ajoutera :

```
        CONNECT_TIME               5
```

3.2 Les utilisateurs

La création des utilisateurs doit être fait par le SYSDBA, après la création du *TableSpace* Air2Java.

3.2.1 L'Administrateur

Cet utilisateur a tout les droits sur la Base.

Il s'agit en fait du propriétaire du shémas Oracle

3.2.2 Le Manager

Cet utilisateur possède le droit d'éditer des données dans la base, mais ne peut pas en modifier la

structure.

3.2.3 Le Consultant

Cet utilisateur n'a que le droit de lire certaines données de la base.

3.2.4 Scripts sql de création des utilisateurs

Les 3 utilisateurs sont crées par le SysDBA avec les même instructions :

```
-- ----------------------------
-- Creation de l'admin
-- ----------------------------
CREATE USER Air2Java
  IDENTIFIED BY Air2Java
    DEFAULT TABLESPACE Air2Java
     QUOTA UNLIMITED ON Air2Java
      TEMPORARY TABLESPACE TEMP
        ACCOUNT UNLOCK;
```

```
-- ----------------------------
-- Creation du manager
-- ----------------------------
CREATE USER Air2JavaUser
  IDENTIFIED BY Air2JavaUser
    DEFAULT TABLESPACE Air2Java
      TEMPORARY TABLESPACE TEMP
        ACCOUNT UNLOCK;
```

```
-- ----------------------------
-- Creation du consultant
-- ----------------------------
CREATE USER Air2JavaConsult
  IDENTIFIED BY Air2JavaConsult
    DEFAULT TABLESPACE Air2Java
      TEMPORARY TABLESPACE TEMP
        ACCOUNT UNLOCK;
```

3.3 Les Rôles

Nous avons besoin de créer trois Rôles, et de leur donner les privilèges en fonction des droits voulus.

La création des rôles ainsi que les privilèges du rôle administrateurs doivent être faites par le SysDBA,
L'assignation des privilèges pour les autres rôles se fait depuis un utilisateur ayant le rôle d'administration (user Air2Java).

```
-- ----------------------------
-- Creation des Roles
-- ----------------------------
CREATE ROLE R_Air2JavaAdmin;
CREATE ROLE R_Air2JavaUser;
CREATE ROLE R_Air2JavaConsult;
```

```
-- ------------------------------------------------
-- Assignation des privileges au role R_Air2JavaAdmin
```

```
  -- ------------------------------------------------
  GRANT
    CONNECT, RESOURCE,
    CREATE SESSION,
    CREATE TABLE,
    CREATE PROCEDURE,
    CREATE TRIGGER,
    CREATE SEQUENCE,
    CREATE VIEW
    TO R_Air2JavaAdmin;
  -- ------------------------------------------------
  -- Assignation des privileges au role R_Air2JavaUser
  -- ------------------------------------------------
  GRANT
    CONNECT,
    CREATE SESSION
    TO R_Air2JavaUser;
  -- Privileges sur les tables
  GRANT SELECT, INSERT, UPDATE, DELETE ON Air2Java.ADRESSES TO
R_Air2JavaUser;
  GRANT SELECT ON Air2Java.ADRESSETYPES TO R_Air2JavaUser;
  GRANT SELECT, INSERT, UPDATE, DELETE ON Air2Java.A_POUR_ADRESSE TO
R_Air2JavaUser;
  [...]
```

```
  -- ------------------------------------------------
  -- Assignation des privileges au role R_Air2JavaConsult
  -- ------------------------------------------------
  GRANT
    CONNECT,
    CREATE SESSION
    TO R_Air2JavaConsult;
  -- Privileges sur les tables
  GRANT SELECT ON Air2Java.ADRESSES TO R_Air2JavaConsult;
  GRANT SELECT ON Air2Java.ADRESSETYPES TO R_Air2JavaConsult;
  GRANT SELECT ON Air2Java.A_POUR_ADRESSE TO R_Air2JavaConsult;
  [...]
```

```
  -- ------------------------------------
  -- Assignation des roles aux utilisateurs
  -- ------------------------------------
  GRANT R_Air2JavaAdmin TO Air2Java;
  GRANT R_Air2JavaUser TO Air2JavaUser;
  GRANT R_Air2JavaConsult TO Air2JavaConsult;
```

4 La mise en place de la base de données

La base de donnée sera installée sur un serveur *Oracle 11gR2* sous *Windows 7*, choix imposé par le commanditaire.

4.1 Création de la base

La base devra être installée par un administrateur système du serveur Oracle.

4.1.1 Création du Schéma

Sous *Oracle*, un *Schema* est créé pour chaque *User*.
La base sera donc installée dans le schéma de l'utilisateur Air2Java.

4.1.2 Création du Tablespace

L'application a son propre espace de travail

```
-- ------------------------------------
-- Creation du tablespace Air2Java
-- ------------------------------------
CREATE TABLESPACE Air2Java
DATAFILE 'TS_Air2Java.DBF' SIZE 500M
ONLINE;
```

4.2 Peuplement de la base

Une fois la structure de la base installée, il faut y importer des données.

Ce sont toutes les informations qui sont nécessaires pour le bon fonctionnement de la base.

Il faudra faire attention à l'ordre dans lequel les données sont importées à cause des dépendances liées aux clés étrangères.

4.2.1 Les données applicatives

Les données applicatives n'ont pas vocation à être modifiées lors de l'utilisation de l'application.

Ces données sont le plus normées possible de manière a faciliter l'import/export dans un format définit pour d'autres applications (synchronisation de carnet d'adresse pour téléphone mobile ou messagerie, localisation gps, …)

Ce sont par exemple les types de numéro de téléphone ou d'adresse (domicile, portable, bureau), ou encore la liste des pays du monde, des villes et des codes postaux.

La mise à jour de ces données se fera au moment de la mise à jour de l'application par l'éditeur.

4.2.2 Les données métier

Les données métiers ont été fournies par le commanditaire, et sont installées dans la version de départ.

Elles pourront être modifiées occasionnellement par l'utilisateur lors de l'utilisation de l'application,

ces modifications pouvant être intégrées à de futures mise à jour de l'application.

Ce sont par exemple les types d'œuvres (musique, danse, conte), la liste des instruments de musique, les spécialités des musiciens, ...

4.2.3 Le jeu de test

Pour être sûr du bon fonctionnement de la base, et pour que l'utilisateur puisse se familiariser avec le fonctionnement de l'application, on fournira une série de données fictives.

On choisira celles-ci de manière a pouvoir contrôler rapidement les résultats des requêtes testées (par exemple, on prendra les 26 noms de familles les plus courant, un par lettre de l'alphabet, et de même 26 prénoms féminin et masculins, pour composer 52 individus, ainsi, si lors d'une requête, le résultat fournit une identification nom/prénom ne commençant pas par la même lettre, l'erreur sera facile a voir).

Pour être complet, le jeu de test devra peupler toutes les tables de la base, et répondre à toutes les cardinalités envisagées dans la conception.

4.3 Scripts d'installation de la base

La base est créée par une série de scripts SQL interprétés par SQL*Plus. Le lancement du script d'installation se fait simplement par :

```
$ sqlplus Air2Java/Air2Java@localhost/XE @Air2Java.sql
```

Ce script va se charger d'appeler tout les autres dans le bon ordre.

Air2Java.sql (extraits)

```
-- On efface tout
@@DROPS\DROP.sql
-- On génère des séquences
@@SEQUENCES\SEQ_ADRESSES.sql
@@SEQUENCES\SEQ_ADRESSETYPES.sql
-- On génère la structure des tables
@@TABLES\ADRESSES.sql
@@TABLES\ADRESSETYPES.sql
@@TABLES\A_POUR_ADRESSE.sql
-- On génère les vues
@@VIEWS\VIEWOEUVRES.sql
-- On génère le contenu des tables
@@DATA_TABLE\ADRESSES.sql
@@DATA_TABLE\ADRESSETYPES.sql
@@DATA_TABLE\A_POUR_ADRESSE.sql
-- On régénère les index
@@INDEXES\PK_ADRESSETYPES.sql
@@INDEXES\PK_ADRESSES.sql
@@INDEXES\I_FK_A_POUR_ADRESSE_ADRESSETYP.sql
@@INDEXES\I_FK_A_POUR_ADRESSE_ADRESSES.sql
@@INDEXES\PK_A_POUR_ADRESSE.sql
@@INDEXES\I_FK_A_POUR_ADRESSE_INDIVIDUS.sql
-- On ajoute les contraintes
@@CONSTRAINTS\ADRESSES.sql
@@CONSTRAINTS\A_POUR_ADRESSE.sql
@@CONSTRAINTS\ADRESSETYPES.sql
@@REF_CONSTRAINTS\A_POUR_ADRESSE.sql
-- On installe les packages de procédures
@@PACKAGES\PACK_RENCONTRES.sql
@@PACKAGE_BODIES\PACK_RENCONTRES.sql
```

4.4 Sauvegarde et restauration

Pour sauvegarder et restaurer la base, on utilise les utilitaires IMP et EXP fournies par Oracle

On se référera à la documentation de Microsoft pour automatiser les sauvegardes.

4.4.1 Exemples

Export complet de la base :

```
$ exp userid=Air2Java/Air2Java owner=Air2Java file=export_full.dump
log=export_full.log
```

Import complet de la base :

```
$ imp userid=Air2Java/Air2Java owner=Air2Java file=export_full.dump
log=import_full.log
```

Copie de la base dans un autre schéma :

```
$ imp userid=system/system file=export_full.dump log=import_full.log
fromuser=Air2Java touser=Air2JavaSave
```

On pourra aussi exporter le résultat d'une requête (ici, la liste des emails invalides)

```
$ exp system/pass file=exp_emails.dmp tables=Air2Java.emails query="'where
email_label not like '%@%.%'"
```

et importer cette liste dans une autre base :

```
$ imp system/pass file=exp_emails.dmp fromuser=Air2Java
touser=Air2JavaSave tables=emails_nonvalides log=imp.log
```

4.4.2 Scripts de sauvegarde et restauration

Les paramètres des import/export seront gérées dans des fichiers de configuration *parfile* :

Air2Java.prm

```
userid=Air2Java/Air2Java
file=export_full.dmp
log=export_full.log
owner=Air2Java
```

On utilisera ensuite les commandes suivantes :

```
$ exp parfile=Air2Java.prm
$ imp parfile=Air2Java.prm
```

5 Manipuler les données avec SQL

5.1 Requêtes SQL

SQL est le langage d'interrogation de la base de donnée.

Voici quelques exemples de requêtes utilisées pour extraire des données :

5.1.1 Afficher tout les Individus contenus dans la base

```
SELECT i.id_individu
    , email_label
```

```
     , emailType_label as email_type
     , nom
     , prenom
     , adr_label
     , adr_cp
     , adr_ville
     , num_telephone
     , teltype_label as tel_type
FROM individus i
   LEFT OUTER JOIN a_pour_email ae
        ON i.Id_Individu = ae.Id_individu
        JOIN emails e
        USING(id_email)
        LEFT OUTER JOIN emailTypes et
        ON ae.Id_emailType = et.Id_emailType
   LEFT OUTER JOIN a_pour_adresse aa
        ON i.Id_Individu = aa.Id_individu
        JOIN adresses adr
        USING(Id_adresse)
   LEFT OUTER JOIN a_pour_tel at
        ON i.Id_Individu = at.Id_individu
        JOIN telephones t
        USING(Id_telephone)
        LEFT OUTER JOIN telephoneTypes tt
        ON at.Id_telType = tt.Id_telType
;
```

5.1.2 Les Groupes jouant un titre donné

```
-- Interrogation des groupes jouant un titre donné.
SELECT g.denomination
 FROM joue j
 INNER JOIN groupes g
 ON j.Id_groupe = g.Id_groupe
 INNER JOIN oeuvres o
 ON j.Id_oeuvre = o.Id_oeuvre
 WHERE o.titre_oeuvre LIKE '%&oeuvr%'
;
```

5.1.3 Les e-mails non conformes

```
-- Liste des emails mal construits (nom@domaine.ext).
SELECT *
 FROM emails
 WHERE email_label NOT LIKE '%@%.%'
;
```

5.2 XML

Oracle permet de sérialiser les réponses des requêtes. On obtient donc une représentation des données dans un format XML que l'on pourra utiliser directement.

Oracle renverra un nœud par tuple, que l'on pourra ajouter à un document DOM par exemple.

Il faudra formater le Select pour construire le modèle des nœuds à renvoyer.

5.2.1 Les Individus

Ici, on souhaite récupérer des Elements Individus ayant leur Id en argument et leur Nom, Prénom et date de naissance en nœuds enfants :

```
 SELECT
 XMLSERIALIZE(
         CONTENT XMLELEMENT(
                 NAME "Individu",
                 XMLATTRIBUTES(
                         i.id_individu "id"
                 ),
                 XMLFOREST(
                         nom "nom",
                         prenom "prenom",
                         date_naissance "date"
                 )
         )
 ) AS Individus
 FROM individus i
/
```

On obtiendra des tuples formatés comme on le désire :

```
<Individu id="65"><nom>Heller</nom><prenom>Romain</prenom><date>1977-09-
02</date></Individu>
```

5.2.2 Les Pays représentés par un groupe donné

```
 SELECT
 XMLSERIALIZE(
         CONTENT XMLELEMENT(
                 NAME "pays",
                 XMLATTRIBUTES(
                         id_pays "id"
                 ),
                 XMLFOREST(
                         pays_label "label"
                 )
         )
 ) AS Pays
 FROM pays
  RIGHT JOIN represente_pays
  USING (id_pays)
  WHERE id_groupe = &group
 /
```

```
Pour le Groupe 17 (Hop Cha Cha !)

<pays id="67"><label>Bulgarie</label></pays>
<pays id="97"><label>France</label></pays>
<pays id="104"><label>Grèce</label></pays>
<pays id="242"><label>Turquie</label></pays>
```

5.3 HTML

Bien sûr, si on peut sortir les données au format XML, il est facile d'obtenir du HTML !

5.3.1 Liste de tout les pays en

```
select XMLSERIALIZE
(
        CONTENT
        XMLELEMENT
        (
                NAME "div",
                XMLATTRIBUTES(id_pays "id"),
                XMLFOREST(pays_label "span")
        )
)
AS Pays
from pays
order by pays_label
/
```

```
<div id="36"><span>Afghanistan</span></div>
<div id="37"><span>Afrique du Sud</span></div>
[…]
<div id="97"><span>France</span></div>
[…]
<div id="249"><span>Yémen</span></div>
<div id="250"><span>Zambie</span></div>
<div id="251"><span>Zimbabwe</span></div>
```

5.3.2 Liste des Pays pour un formulaire <select>

```
select XMLSERIALIZE
(
        CONTENT
        XMLELEMENT
        (
                NAME "option",
                XMLATTRIBUTES(id_pays "id"),
                pays_label
        )
)
AS Pays
from pays
order by pays_label
/
```

```
<option id="36">Afghanistan</option>
<option id="37">Afrique du Sud</option>
[...]
<option id="250">Zambie</option>
<option id="251">Zimbabwe</option>
```

Il ne restera qu'a insérer le résultat de la requête entre les balises <select name='Pays'> et </select>.

6 Programmer dans le langage du SGBD

Les SGBD permettent en général la programmation de procédures embarquées qui sont écécutées directement par le système. Cela permet en général de gagner en rapidité d'exécution puisqu'on est au plus près des données traitées, mais ce au détriment de la portabilité, puisque chaque SGBD utilise un langage qui lui est propre.

Oracle utilise usuellement le langage PL/SQL, dérivé de ADA, qui permet une approche fonctionnelle structuré en packages.

6.1.1 Exemples

Quelques fonctions embarquées qui peuvent être utiles pour la gestion des Groupes :

```
  -- Package Groupes
  -- Pour tout ce qui touche a la gestion des Groupes
  CREATE OR REPLACE PACKAGE Pack_Groupes IS
          -- Déclaration de types globaux
          TYPE tgroupe IS RECORD (
                  groupe_id       groupes.id_groupe%TYPE
                , groupe_nom      groupes.denomination%TYPE
          );
          PROCEDURE groupe_not_in_rencontre(r_Id rencontres.id_rencontre
%TYPE);
          FUNCTION groupe_not_in_rencontre(r_Id rencontres.id_rencontre%TYPE)
                  RETURN Pack_Groupes.tgroupe ;
          FUNCTION get_id_last_rencontre
                  RETURN rencontres.id_rencontre%TYPE ;
  END Pack_Groupes ;
  /
  CREATE OR REPLACE PACKAGE BODY Pack_Groupes IS
  -- Procédure groupe_not_in_rencontre
  -- Affiche les groupes @TODO Refaconner en Fonction !
          PROCEDURE groupe_not_in_rencontre(r_Id rencontres.id_rencontre
%TYPE) IS
                  individ_id      individus.id_individu%TYPE       ;
                  CURSOR C IS
                          SELECT id_groupe, denomination
                          FROM groupes
                          WHERE groupes.id_groupe NOT IN (
                          -- Groupes qui participent à la rencontre r_id :
                                  SELECT UNIQUE g.id_groupe
                                  FROM programme p
                                  INNER JOIN rencontres r
                                          ON p.id_rencontre = r_id
                                  INNER JOIN groupes g
                                          ON p.id_groupe = g.id_groupe
                          )
                          ORDER BY denomination
                          ;
                  vgroupe  tgroupe ;
          BEGIN
                  OPEN C;
                  LOOP
                          fetch C into vgroupe ;
                          EXIT WHEN C%NOTFOUND ;
```

```
                        dbms_output.put_line(
                                '- ' || vgroupe.groupe_id
                            || ' : ' || vgroupe.groupe_nom
                            || ' '
                        );
                END LOOP;
                CLOSE C;
        END groupe_not_in_rencontre ;
        FUNCTION groupe_not_in_reunion(r_Id rencontres.id_rencontre%TYPE)
        RETURN tgroupe IS
                individ_id      individus.id_individu%TYPE      ;
                CURSOR C IS
                        SELECT id_groupe, denomination
                        FROM groupes
                        WHERE groupes.id_groupe NOT IN (
                        -- Groupes qui participent à la rencontre r_id :
                                SELECT UNIQUE g.id_groupe
                                FROM programme p
                                INNER JOIN rencontres r
                                        ON p.id_rencontre = r_id
                                INNER JOIN groupes g
                                        ON p.id_groupe = g.id_groupe
                        )
                        ORDER BY denomination
                        ;
                vgroupe  tgroupe ;
        BEGIN
                OPEN C;
                LOOP
                        fetch C into vgroupe ;
                        EXIT WHEN C%NOTFOUND ;
                        dbms_output.put_line(
                                '- ' || vgroupe.groupe_id
                            || ' : ' || vgroupe.groupe_nom
                            || ' '
                        );
                END LOOP;
                CLOSE C;
        RETURN(vgroupe);
        END;
-- Fonction get_id_last_rencontre
-- Retourne l'Id de la dernière rencontre
        FUNCTION get_id_last_rencontre
        RETURN rencontres.id_rencontre%TYPE IS
                id_      rencontres.id_rencontre%TYPE       ;
                CURSOR C IS
                        SELECT max(Id_rencontre) FROM rencontres
                ;
        BEGIN
                OPEN C;
                LOOP
                                fetch C into id_ ;
                                EXIT WHEN C%NOTFOUND ;
        /*                      dbms_output.put_line(
                                        '- ' || id_
                                );
        */
                        END LOOP;
```

```
                CLOSE C;
        RETURN (id_);
        END get_id_last_rencontre;
  END Pack_Groupes ;
  /
```

6.1.2 Tests

```
  /*
     set linesize 1400;
     set pagesize 40;
     set serveroutput on;
     set showmode on;
  */
  exec Pack_Groupes.groupe_not_in_rencontre(&rid) ;
  -- Tests id_last_rencontre
  select Pack_Groupes.get_id_last_rencontre() from dual ;
  select * from rencontres where id_rencontre =
Pack_Groupes.get_id_last_rencontre() ;
```