Projet de BDA

COMPTE RENDU

Rémi KEBAILI - Pascal NGUYEN - Angélique ETIENNE – Juexiao ZHANG

TP3B2 | 2016

Introduction

Dans le cadre du projet tutoré de S3 concernant la gestion des stages, la mise en place d'une base de données est indispensable. Notre groupe s'est réparti les différentes tâches selon le plan suivant :

Plan :

1. Création de types et tables
2. Insertion de données
3. Fonctions stockées & requêtes
4. JDBC
5. Triggers et Statistiques

Néanmoins, chaque membre du groupe a testé les différentes parties du projet. Chaque personne ne se limitait pas uniquement à la tâche qui lui était attribuée à la base. En cas de besoin, plusieurs personnes pouvaient travailler sur un même problème pour tenter de le corriger.

Répartition des taches

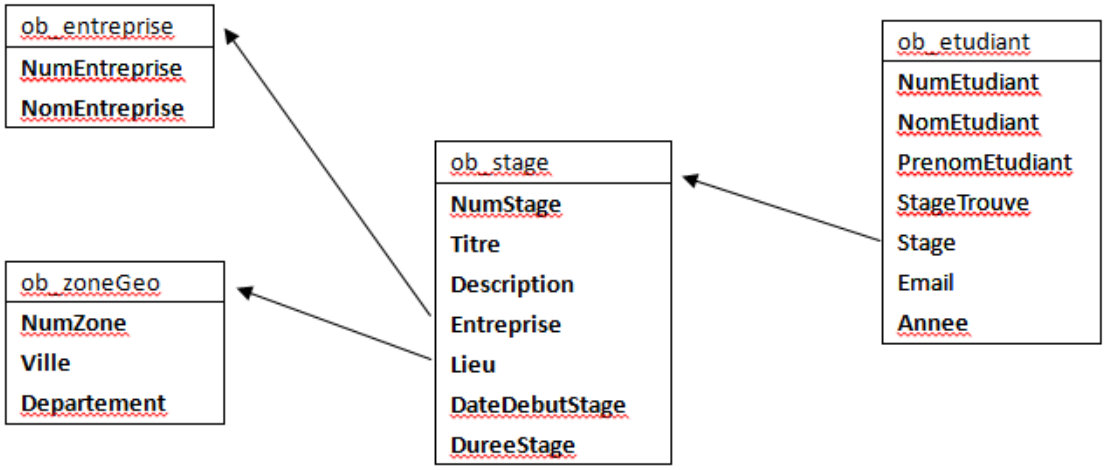
La conception de la base de données et la marche à suivre pour réaliser le projet ont été établies par tous les membres du groupe. Chacun suivait et contrôlait toutes les tâches effectuées et lorsqu'un problème était rencontré nous nous concertions pour trouver une solution. Elles ont cependant principalement été attribuées sous cette forme :

Angélique ETIENNE : création de type et de tables
Juexiao ZHANG : insertions de données
Rémi KEBAILI : JDBC et Fonctions stockées/Requêtes SQL (70%)
Pascal NGUYEN : Trigger, Statistique et Fonctions stockées/Requêtes SQL (30%)

1. Création de la structure de la base de données

Dans un premier temps, il a fallu commencer par la structure de notre base de données. Le choix des tables a été fait en analysant les requêtes qu'il faudra pouvoir réaliser et les différentes contraintes.

Voici un schéma simplifié de notre base de données :



La base de données a été créée selon le principe de base de données orientée objet.

Pour créer ces tables, nous créons tout d'abord des types :

ob_entreprise_ty
ob_zoneGeo_ty
ob_stage_ty
ob_etudiant_ty

Les tables ont ensuite été créées le même ordre :

ob_entreprise
ob_zoneGeo
ob_stage
ob_etudiant

Exemple de création d'une table :

```
--*****************************************
--*           ZONE GEOGRAPHIQUE           *
--*****************************************
-- Objet zone géographique
CREATE OR REPLACE TYPE ob_zoneGeo_ty AS OBJECT(
NumZone INTEGER,
Ville VARCHAR(50),
Departement NUMBER(3));

CREATE TABLE ob_zoneGeo OF ob_zoneGeo_ty
(NumZone PRIMARY KEY);

--*****************************************
--*                STAGE                  *
--*****************************************
-- Objet stage
CREATE OR REPLACE TYPE ob_stage_ty AS OBJECT(
NumStage INTEGER,
Titre VARCHAR2(250),
Description VARCHAR2(800),
Disponible NUMBER(1), -- 1 pour true et 0 pour false
entreprise REF ob_entreprise_ty,
lieu REF ob_zoneGeo_ty,
dateDebutStage DATE,
dureeStage INTEGER);

-- Table ob_stage
CREATE TABLE ob_stage OF ob_stage_ty
(NumStage PRIMARY KEY);
```

Notons que la table ob_stage possède une entreprise de type ob_entreprise_ty et un lieu de type ob_zoneGeo_ty.

2. Insertion de données

Pour les données, on utilise des données réalistes et utilisables (160 étudiants pour 200 stages, répartis sur 4 années) :

NumZone	Ville	Departement
1	Orsay	91
2	Paris	75
3	Massy	91
4	Longjumeau	91
5	Villeneuve-Saint-Georges	94
6	Strasbourg	67
7	Nantes	44

NumStage	Titre	Description	Disponible	Entreprise	Lieu	DateDebut	Duree
1	Assistant développeur	Une description de stage	0	3	5	TO_DATE('15/04/2016','DD/MM/YYYY')	66
2	Développeur web	Une description de stage	0	14	7	TO_DATE('05/04/2016','DD/MM/YYYY')	49
3	Analyste-programmeur	Une description de stage	1	9	2	TO_DATE('04/04/2016','DD/MM/YYYY')	65
4	Administrateur réseaux	Une description de stage	0	12	4	TO_DATE('01/04/2016','DD/MM/YYYY')	39
5	R&D Développement	Une description de stage	0	3	4	TO_DATE('01/04/2016','DD/MM/YYYY')	41

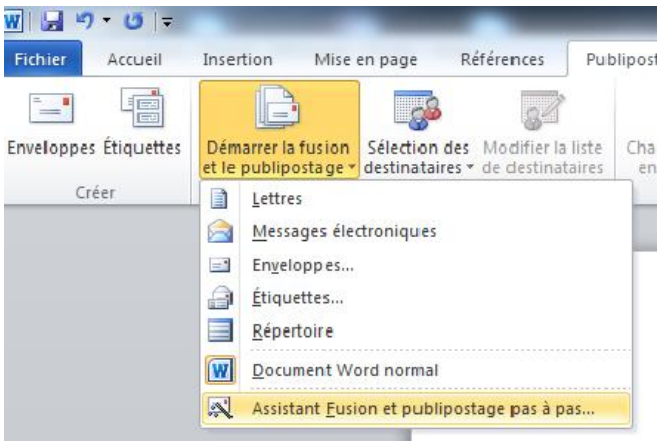
NumEtudiant	NomEtudiant	PreEtudiant	StageTrouve	Stage	Email	Annee
1	Andre	Johnny	1	REF(S)	Johnny.Andre@u-psud.fr	2016
2	Aouri	Barbara	0	REF(S)	Barbara.Aouri@u-psud.fr	2016
3	Bellik	Doyle	1	REF(S)	Doyle.Bellik@u-psud.fr	2016
4	Benissan	Stewart	1	REF(S)	Stewart.Benissan@u-psud.fr	2016
5	Binello	Mohammed	1	REF(S)	Mohammed.Binello@u-psud.fr	2016
6	Biswas	Rémi	1	REF(S)	Rémi.Biswas@u-psud.fr	2016

Des exemples du code d'insertion :

```
INSERT INTO ob_stage
SELECT 1, 'Assistant développeur', 'Une description de stage', 0, REF(E), REF(L),
TO_DATE('15/04/2016','DD/MM/YYYY'), 87
FROM ob_entreprise E, ob_zoneGeo L
WHERE NumEntreprise = 2
AND NumZone = 3;

INSERT INTO ob_zoneGeo
VALUES (1, 'Orsay', 91);
```

De plus, nous avons choisi d'utiliser le publipostage (une fonction de Word) de manière à écrire le code plus rapidement mais aussi à éviter les erreurs :




```
INSERT INTO ob_entreprise
VALUES (1, Peugeot) ;

INSERT INTO ob_entreprise
VALUES (2, Air France) ;

INSERT INTO ob_entreprise
VALUES (3, ChocoStart) ;

INSERT INTO ob_entreprise
VALUES (4, Android) ;

INSERT INTO ob_entreprise
VALUES (5, Société générale) ;

INSERT INTO ob_entreprise
VALUES (6, BNP Paribas) ;

INSERT INTO ob_entreprise
VALUES (7, Pear Store) ;

INSERT INTO ob_entreprise
VALUES (8, Systemis) ;

INSERT INTO ob_entreprise
VALUES (9, Starbreeze) ;
```

3. Fonctions stockées & requêtes

- ***Fonctions pour récupérer une valeur particulière (comme une moyenne ou une somme)***

  - récupérer le nombre d'étudiants avec stage cette année

```
create or replace FUNCTION nbEtudiantsAvecStage
return INTEGER
As
nbEtudiants INTEGER;
Begin
SELECT COUNT(*) into nbEtudiants
FROM ob_etudiant o
Where StageTrouve = 1
and Annee = to_number(to_char(sysdate, 'yyyy'));
return nbEtudiants;
End nbEtudiantsAvecStage;
```

  - récupérer le nombre d'étudiants sans stage cette année

```
create or replace FUNCTION nbEtudiantsSansStage
return INTEGER
As
nbEtudiants INTEGER;
Begin
SELECT COUNT(*) into nbEtudiants
FROM ob_etudiant
Where StageTrouve = 0
and Annee = to_number(to_char(sysdate, 'yyyy'));
return nbEtudiants;
End nbEtudiantsSansStage;
```

  - récupérer le nombre d'étudiants sans stage à une certaine date pour une année précédente choisie par l'utilisateur

```
CREATE OR REPLACE FUNCTION NbEtudiantsSansStageAnnee
(ann IN ob_etudiant.annee %TYPE)
RETURN INTEGER
AS
nbEtudiants INTEGER;
BEGIN
SELECT COUNT(*) INTO nbEtudiants
FROM ob_etudiant E
WHERE StageTrouve = 0
AND Annee = ann;
RETURN nbEtudiants;
END NbEtudiantsSansStageAnnee;
```

- ***3 fonctions pour le cas du nombre de stages par zone géographique :***

  - le nombre de stages par zone géographique choisie par l'utilisateur (département, ville)

Pour les villes

```
create or replace FUNCTION nbStagesParVille
(uneVille in ob_zoneGeo.ville % type )
return INTEGER
As
nbStagesVille INTEGER;
Begin
SELECT COUNT(*) into nbStagesVille
FROM ob_stage o, ob_zoneGeo z
Where o.lieu = REF(z)
And z.ville = uneVille;
return nbStagesVille;
End nbStagesParVille;
```

Pour les départements

```
create or replace FUNCTION nbStagesParDepartement
(unDepartement in ob_zoneGeo.departement % type )
return INTEGER
As
nbStagesDepartement INTEGER;
Begin
SELECT COUNT(*) into nbStagesDepartement
FROM ob_stage o, ob_zoneGeo z
Where o.lieu = REF(z)
And z.departement = unDepartement;
return nbStagesDepartement;
End nbStagesParDepartement;
```

- le nombre de stages pour toutes les zones géographiques (département, ville) *(c'est-à-dire tous les stages)*

```
create or replace FUNCTION nbStages
return INTEGER
As
nbStages INTEGER;
Begin
SELECT COUNT(*) into nbStages
FROM ob_stage ;
return nbStages;
End nbStages;
```

- ***Requêtes SQL pour les affichages***

- le nombre de stagiaires pris par chaque entreprise durant les *n* dernières années

A mettre dans un PreparedStatement dans le code Java

```
SELECT en.nomEntreprise, COUNT(et.stage.numStage)
as nombre_de_stagiaires
FROM ob_stage s, ob_entreprise en, ob_etudiant et
WHERE s.entreprise = REF(en)
and et.stage = REF(s)
and Annee >= to_number(to_char(sysdate, 'yyyy')) + 1 - 5
GROUP BY en.numEntreprise, en.nomEntreprise;
```

Le « 5 » (à la ligne 6) a été remplacé par une variable contenant le

nombre d'années saisie par l'utilisateur pour n année.

- le nombre moyen de stagiaires encadrés par les entreprises dans les *n* dernières années

```
CREATE OR REPLACE FUNCTION nbMoyenStagiaire (ann IN INTEGER)
RETURN INTEGER
AS
nbStagiaires INTEGER;
BEGIN
SELECT COUNT(*)/ann INTO nbStagiaires
FROM ob_etudiant E
WHERE StageTrouve = 1
AND Annee >= to_number(to_char(sysdate, 'yyyy')) + 1 - ann;
RETURN nbStagiaires;
END nbMoyenStagiaire;
```

- récupérer toutes les entreprises et leur contact ayant eu au moins un stage dans les *n* dernières années

A mettre dans un PreparedStatement dans le code Java

```
SELECT DISTINCT s.entreprise.NomEntreprise,
et.nometudiant, et.preetudiant
from ob_stage s, ob_etudiant et
where et.stage = REF(s)
and to_number(to_char(dateDebutStage, 'yyyy')) >=
    to_number(to_char(sysdate, 'yyyy')) + 1 - 5
order by s.entreprise.NomEntreprise,
         et.nometudiant, et.preetudiant;
```

Le « 5 » (à la ligne 6) a été remplacé par une variable contenant le nombre d'années saisie par l'utilisateur pour n année.

4. JDBC

CallableStatement : Pour les procédures stockées

CallableStatement cst = myconnexion.prepareCall("{? = call nbEtudiantsAvecStage}");

Lorsque l'on cherchait à obtenir le résultat d'un calcul précis comme une moyenne ou une somme, nous avons utilisé des fonctions stoquées.

PreparedStatement : Pour les requêtes

Nombre de stagiaires pris par chaque entreprise durant les nbAnnéeSaisi dernières années :

```
String req = "SELECT en.nomEntreprise, COUNT(et.stage.numStage) as nombre_de_stagiaires
"+
        "FROM ob_stage s, ob_entreprise en, ob_etudiant et "+
        "WHERE s.entreprise = REF(en) "+
```

```
                "AND et.stage = REF(s) "+
                "AND Annee >= to_number(to_char(sysdate, 'yyyy')) + 1 - '"+nbAnnéeSaisi+"' "+
                "GROUP BY en.numEntreprise, en.nomEntreprise";
PreparedStatement psm = myconnexion.prepareStatement (req);
```

Ici on utilise « PreparedStatement » pour une éventuelle amélioration de l'application, par exemple si on veut mettre une interface graphique plus tard avec un bouton pour chaque chose qu'on veut savoir (c'est à dire un bouton par question) on stoque les SELECT dans un prepareStatement et on peut les réutiliser quand on veut.
Lorsque l'on cherchait à obtenir un affichage de valeurs en fonction d'un attribut (nombre de stagiaires par entreprise par exemple), nous avons utilisé des requêtes SQL simples.

5. Triggers et Statistiques

Une table Statistique contenant une unique ligne, avec chaque colonne correspondant au résultat d'une fonction stockée, a été créée.
Des triggers ont également été mis en place afin de mettre à jour les données de la table Statistique, à chaque insertion, suppression ou mise à jour de ligne de la table Etudiant ou de la table Stage.

Exemple : Trigger sur la table Stage

```
CREATE OR REPLACE TRIGGER miseAJourStatistiqueStage
AFTER INSERT OR UPDATE OR DELETE
ON ob_stage
BEGIN
   UPDATE Statistique
   SET nbStageMassy = (SELECT NbStagesParVille('Massy')
                              FROM ob_stage
                              WHERE ROWNUM = 1)
   WHERE ROWNUM = 1;

   UPDATE Statistique
   SET nbStage91 = (SELECT NbStagesParDepartement(91)
                           FROM ob_stage
                           WHERE ROWNUM = 1)
   WHERE ROWNUM = 1;

   UPDATE Statistique
   SET nbTotalStage = (SELECT NbStages
                              FROM ob_stage
                              WHERE ROWNUM = 1)
   WHERE ROWNUM = 1;
END;
/
```

La table Statistique étant créée avec par défaut avec 1 seule ligne, chaque modification de colonne est un update, appelant une fonction stockée réalisée préalablement. De plus, les valeurs initiales de chaque colonne de Statistique étant NULL, il est nécessaire d'insérer un nouveau stage et un nouvel étudiant pour que la table se mette à jour, après sa création.