Projet de SGBD

Gestion des associations

Auteurs :
Gabin GAUDRÉ
Mehdy SALIMI
Julien BURÉ

Table des matières

1 Contexte

L'objet de ce projet est la réalisation d'une base de données et d'une interface permettant sa consultation. Cette base de données a pour but de rassembler l'ensemble des informations relatives aux associations de l'ENSEIRB-MATMECA et aux événements qu'elles organisent.

On y retrouve ainsi les différentes associations avec leurs objectifs, leur bureau et leurs sources de financement. Viennent ensuite les adhérents qui cotisent dans une ou plusieurs associations, on dispose pour chacun d'entre eux de leur nom et prénom, de leur date de cotisation, de leur filière et promotion, de leur adresse mail ainsi que d'un login et un mot de passe. Les événements organisés par les différentes associations sont aussi enregistrés avec leur titre, le lieu dans lequel ils se déroulent, le nombre places disponibles, l'association organisatrice, le responsable de l'événement, la fréquence de l'événement en question, les dates de début et de fin, le coût de l'événement et le prix de la place pour les différents participants en fonction de leur adhérence ou non à l'association organisatrice. Les participants peuvent de plus laisser des commentaires sur les événements et leur attribuent une note entre 1 et 10. On trouve enfin des news publiées par les membres du bureau des associations et qui comportent un titre, une date de publication et un texte. Les adhérents d'une association peuvent commenter les news publiées par celle-ci.

Notre base de données doit contenir et organiser entre elles toutes ces informations afin de permettre un ensemble de requêtes de consultations et de requêtes statistiques.

2 Modèle entité-association

La première étape de la conception de notre base de données consiste en la construction d'un modèle entités associations. Le contexte dans lequel s'inscrit notre base de données nous a permis de construire les entités en présence naturellement. Nous avons ensuite recensé les différentes associations liant ces entités entre elles.

2.1 Normalisation du modèle entités-associations

Il s'agit ensuite de normaliser cette première ébauche du modèle conceptuel. En appliquant les règles de normalisation nous avons pu vérifier que notre modèle entités-associations était correctement conçu. En effet il vérifie chacune des règles de normalisation et ne comporte donc pas d'associations fantômes ou de redondances. De même les attributs non identifiants des différentes associations dépendent directement de l'identifiant de l'entité concernée tandis que les identifiants des entités ne dépendent jamais des autres attributs de ces entités. On en conclu donc que notre modèle entités-associations respecte bien la 3^{me} forme normale de Boyce-Codd.

Un point particulier peut en revanche sembler discutable. Il s'agit de la présence de deux tarifs et de deux dates dans l'entité *ÉVÉNEMENTS*. L'usage et les règles de normalisation voudraient que ces deux types d'information forment chacun une nouvelle entité qui serait mise en relation avec l'entité *ÉVÉNEMENTS* *via* de nouvelles associations. Nous avons décidé de procéder autrement en conservant deux dates et deux tarifs dans l'entité *ÉVÉNEMENTS* car étant donné les informations qui seront effectivement stockées on réalise que la redondance de l'information ne sera que très faible et qu'elle ne présente ici pas de risque de nuire à l'intégrité et la stabilité de notre base de données. En effet les tarifs et les dates de début de de fin d'un événement sont propres à celui-ci et l'information n'est stockée que dans l'entité *ÉVÉNEMENTS*. La redondance aurait en revanche été problématique si la même information avait été stockée à deux endroits différents, on aurait alors pu ne modifier qu'une seule des deux occurrences de cette information, ce qui nuirait évidemment à la stabilité de la base de données dans le temps, mais ce n'est, comme nous l'avons vu, pas le cas ici.

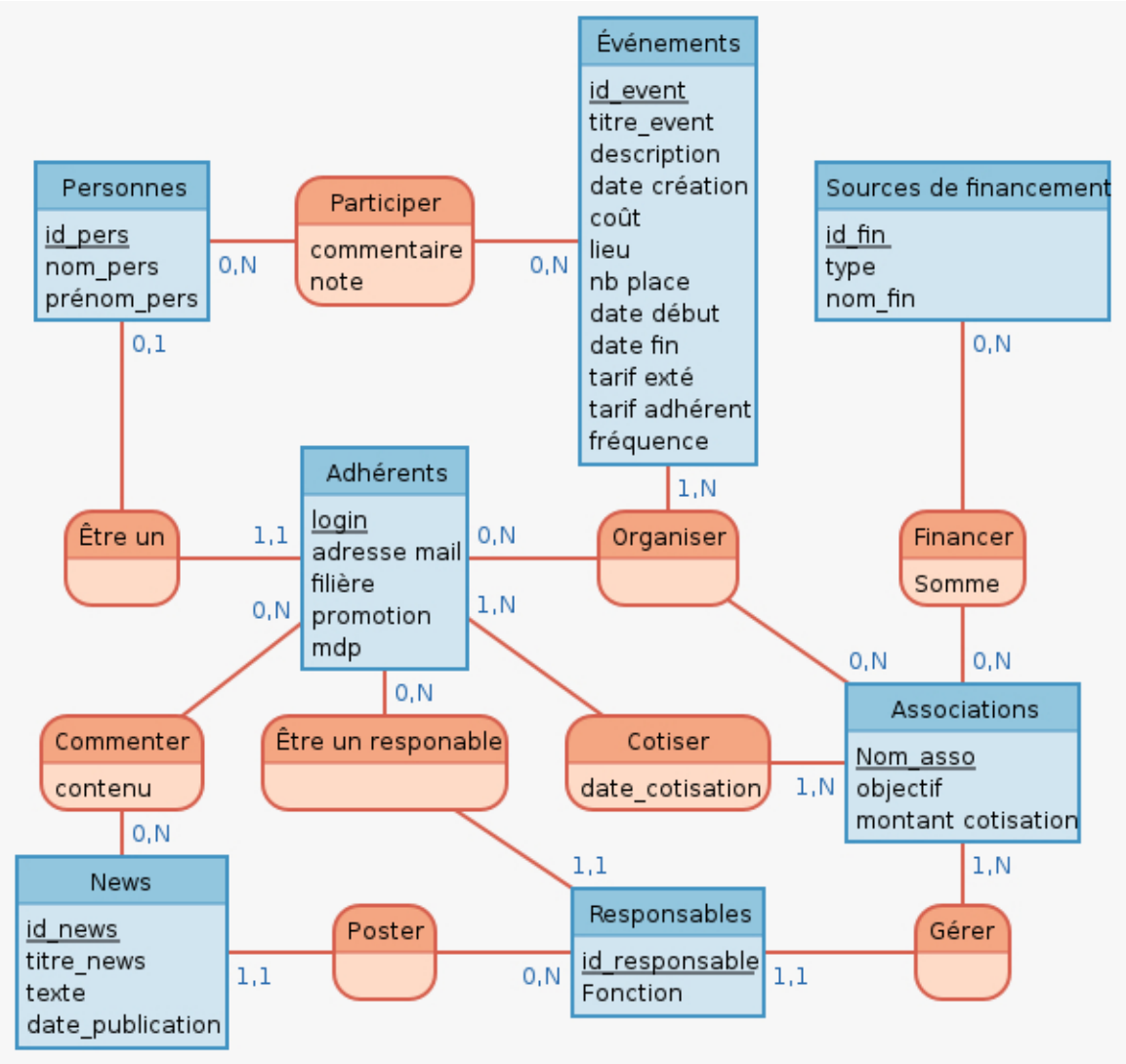

FIGURE 1 – Modèle entités-associations

2.2 Liste des opérations

Il s'agit ensuite de définir l'ensemble des opérations qui doivent être réalisables sur notre base de données. Nous différencierons les requêtes de consultation des requêtes statistiques et des opérations permettant la création et la modification de la base de donnnées.

Requêtes de consultation :
- de la liste des lignes d'une table choisie par l'utilisateur.
- des informations sur les adhérents d'une association à une date donnée.
- des informations sur les événements à venir.
- du bureau d'une association.
- de la liste des commentaires à propos d'un événement.
- de la liste des news d'une association.
- de la liste des commentaires d'une news.
- des dernières news.
- de la liste des organisateurs d'un événement.
- de la liste des personnes y participant
- du nombre de commentaires postés en réaction à une news.

Requêtes statistiques :
- la moyenne du nombre de commentaires donnés par chaque adhérent pour une news.
- la moyenne des notes données par les participants à un événement.
- le classement des événements en fonction de la note moyenne donnée par les participants à chacun de ces événements.
- le classement des adhérents participant le plus aux événements.
- le classement des financement d'une association à une date donnée en comptant les cotisations.

Mis à jour de la base de données :
- Ajout d'un adhérent, d'une personne, d'un événement etc...
- Modification du mot de passe d'un adhérent.
- Suppression d'un adhérent.
- Ajout d'une association.

3 Schéma relationnel

Dans cette partie nous allons nous intéresser à la construction d'un schéma relationnel à partir du modèle entités-associations présenté à la figure 1.

3.1 Passsage au relationnel

Comme nous venons de le présenter, notre modèle relationnel se construit à partir du modèle entités-associations en lui appliquant une série de transformations.

La première d'entre elles consiste à attribuer une clé primaire à toutes les entités. C'est l'identifiant de chaque entité qui revêt ce rôle.

Il s'agit ensuite de caractériser chacune des associations présentes sur le modèle entités-associations. On ne s'intéresse qu'aux cardinalités maximales des associations. On obtient le résultat suivant :
- *Participer* est de type N :M
- *Être un* est de type 1 :1
- *Financer* est de type N :M
- *Gérer* est de type 1 :N

— *Poster* est de type 1 :N
— *Commenter* est de type N :M
— *Être responsable* est de type 1 :N
— *Cotiser* est de type N :M
— *Organiser* est une association non binaire et sera par conséquent traitée différemment.

Il s'agit maintenant de traiter chaque type d'association de façon à constituer les tables manquantes en leur attribuant une clé primaire et à attribuer des clés étrangères aux différentes tables.

Les associations de type 1 :N disparaissent, substituées par une clé étrangère dans la table du côté (0,1) ou (1,1) de l'association. Ainsi :

— L'association *Gérer* disparaît et *nom_ asso*, clé primaire de la table *ASSOCIATIONS*, devient une clé étrangère dans la table *RESPONSABLES*.
— L'association *Poster* disparaît et *id_ responsable*, clé primaire de la table *RESPONSABLES*, devient une clé étrangère dans la table *NEWS*. Celle-ci devra nécessairement être non vide.
— L'association *Être responsable* disparaît et *login*, clé primaire de la table *ADHÉRENTS*, devient une clé étrangère dans la table *RESPONSABLES*.

Les associations de type N :M deviennent des tables dont la clé primaire est constituée de deux clés primaires des tables concernées par l'association. Les attributs de l'association deviennent les attributs de la nouvelle table. On obtient donc :

— Une nouvelle table *PARTICIPER*, dont la clé primaire est constitué des clés étrangères *id_ pers* et *id_ event* issues respectivement des tables *PERSONNES* et *ÉVÉNEMENTS*. Les attributs *commentaire* et *note* de l'association deviennent ceux de la nouvelle table.
— Une nouvelle table *FINANCER*, dont la clé primaire est constitué des clés étrangères *id_ fin* et *nom_ asso* issues respectivement des tables *SOURCES DE FINANCEMENT* et *ASSOCIATIONS*. L'attribut *somme* de l'association devient celui de la nouvelle table.
— Une nouvelle table *COMMENTER*, dont la clé primaire est constitué des clés étrangères *id_ news* et *login* issues respectivement des tables *NEWS* et *ADHÉRENTS*. L'attribut *contenu* de l'association devient celui de la nouvelle table.
— Une nouvelle table *COTISER*, dont la clé primaire est constitué des clés étrangères *nom_ asso* et *login* issues respectivement des tables *ASSOCIATIONS* et *ADHÉRENTS*. L'attribut *date_ cotisation* de l'association devient celui de la nouvelle table.

L'association *Être un*, de type 1 :1, est traitée comme une association de type 1 :N. La clé étrangère symbolisant la relation est placée dans la table *ADHÉRENTS* et référence la clé primaire *id_ pers* de la table *PERSONNES*.

Enfin l'association non binaire *Organiser* engendre une ultime table supplémentaire. La clé primaire de cette table est constituée des clés primaires des tables *ÉVÉNEMENTS*, *ADHÉRENTS* et *ASSOCIATIONS*, à savoir *nom_ asso*, *login* et *id_ event*.

3.2 Schéma relationnel

Vous trouverez à la figure 2 une représentation du modèle relationnel élaboré à partir de notre modèle entités-associations présenté à la figure 1. Les clés primaires des différentes tables sont soulignées tandis que les clés étrangère y sont précédées d'un comme le veut la convention. Ce schéma a été construit en partant d'un modèle entités-associations respectant la 3_{me} forme normale. Les contraintes sur les clés primaires et étrangères sont elles aussi respectées.

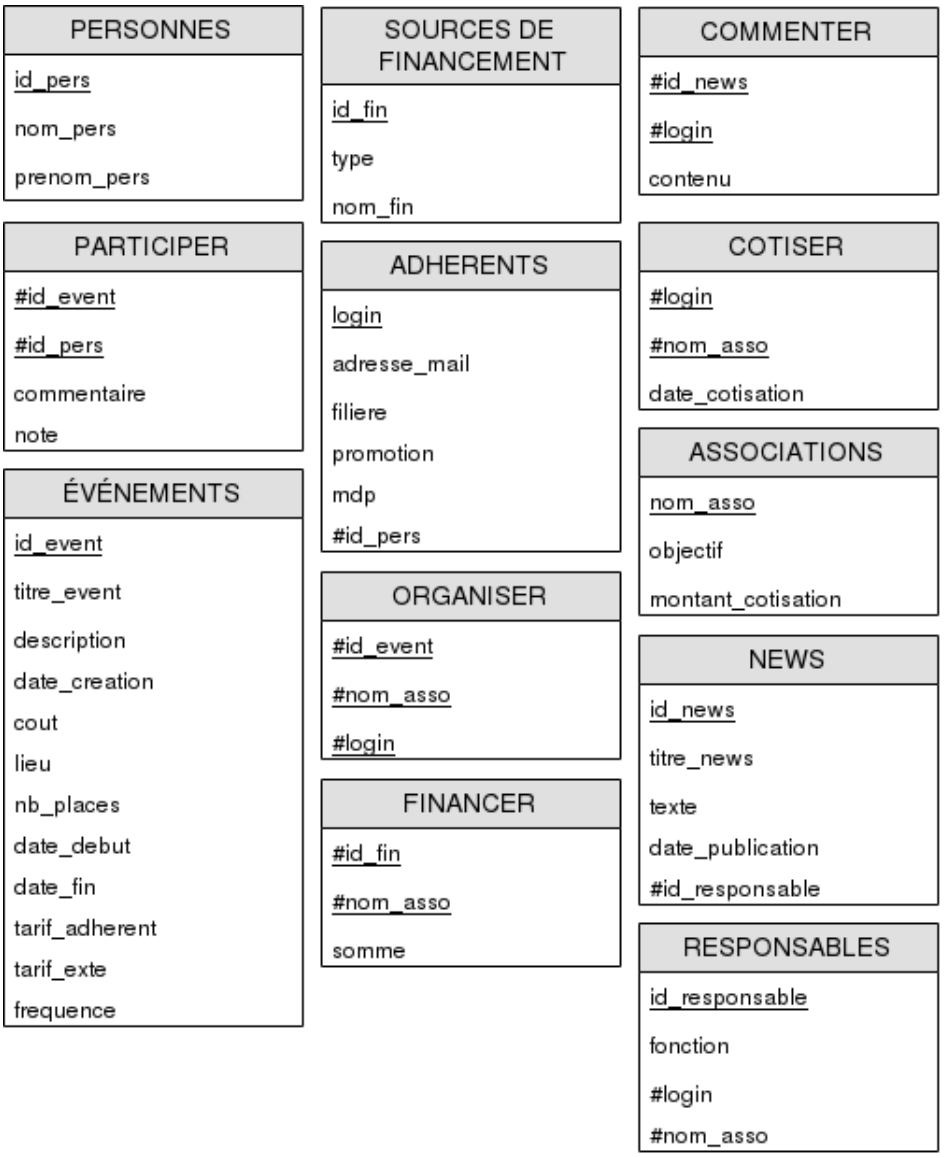

Figure 2 – Modèle relationnel

4 Choix de la base SQL

Notre projet est basé le système de gestion de données Oracle et nous utiliserons SQL*Plus comme langage de requêtes. Ce choix a été déterminé par le fait que nous étions davantage habitués à Oracle qu'à MySQL. Cependant notons qu'à le recul nous nous sommes rendu compte que MySQL simplifiait largement l'utilisation avec PHP.

4.1 Création de la base de données

La création de notre base de données s'appuie naturellement sur le modèle relationnel que nous avons construit plus haut. Il s'agit donc d'écrire un script permettant de créer l'ensemble des tables de la base de données. Une condition préalable à cela est de supprimer l'ensemble des tables préexistantes et les objets qui en dépendent afin de nous assurer que la base qui sera obtenue après exécution correspondra bien à celle que l'on cherche à créer. On évite ainsi qu'une table préexistante ne bloque la création de la nouvelle table lorsque les deux portent

le même nom. Cela s'effectue au moyen de la commande : `drop table <NOM_TABLE> cascade constraints`. De même on supprime les séquences qui ont été définies grâce à la commande `drop sequence <ATTRIBUT> _SEQ`.
On peut ensuite générer sereinement les tables que l'on a construites avec la commande :

```
create table <TABLE>
(
    <attribut_1>            <TYPE>(<NB>)          not null / default <valeur> ,
    <attribut_2>            <TYPE>(<NB>)          not null / default <valeur> ,
    .
    .
    .
    <attribut_n>            <TYPE>(<NB>)          not null / default <valeur> ,
    constraint <nom_cle_primaire> primary key (<attribut_i>)
);
```

Dans certains cas on peut souhaiter restreindre les valeurs qu'un attribut peut prendre. On ajoute donc `check (type in ('CHOIX_1', 'CHOIX_2', 'CHOIX_3', ...))` après le motif `<TYPE>(<NB>)`.

Afin de ne pas avoir de doublons dans les identifiants ou d'avoir à tenir le compte par soi-même nous avons réalisé des séquences qui permettent de générer les identifiants de façon automatisée. Celles-ci s'écrivent de la manière suivante :

```
create sequence ID_PERS_SEQ
    minvalue 0
    maxvalue 99999
    start with 0
    increment by 1
    cache 20;
```

Le cache permet de garder quelques valeurs dans une zone mémoire particulière ce qui permet un accès plus rapide à ces données.

4.2 Requêtes SQL

Comme nous l'avons vu plus haut les requêtes sur notre base de données se divisent entre les requêtes de consultation et les requêtes statistiques. Nous ne détaillerons ici que les requêtes les plus intéressantes ou complexes. Nous aborderons aussi dans cette partie l'ajout et la suppression de nouveaux éléments dans la base de données et la modification des lignes qui peuvent l'être.

4.2.1 Requêtes de sélection

La plupart des requêtes de sélection ne présentent pas de point d'intérêt particulier qui nous pousserait à les développer davantage. Nous nous contenterons donc ici d'en présenter une. Il s'agit de la requête permettant d'obtenir les événements à venir. Elle s'énonce comme suit :

```
SELECT TITRE_EVENT, LIEU, DATE_DEBUT, NOM_ASSO, LOGIN_ADHERENT
FROM EVENEMENTS JOIN ORGANISER ON EVENEMENTS.ID_EVENT = ORGANISER.ID_EVENT
WHERE DATE_DEBUT >= (select SYSDATE + 206 from DUAL)
ORDER BY DATE_DEBUT ASC;
```

4.2.2 Requêtes statistiques

Les requêtes statistiques présentent quelques difficultés supplémentaires. Le premier exemple remarquable est la requête permettant d'obtenir le nombre moyen de commentaires donnés par chaque adhérent à une news en particulier. Celle-ci est construite par imbrication de requêtes :

```
SELECT AVG(NB_COMMENTAIRE) AS "Moyenne"
FROM (SELECT COUNT(LOGIN_ADHERENT) AS "NB_COMMENTAIRE"
      FROM COMMENTER
      WHERE ID_NEWS = (SELECT ID_NEWS FROM NEWS WHERE TITRE_NEWS = &news)
      GROUP BY LOGIN_ADHERENT);
```

On commence par définir clairement l'ensemble sur lequel on désire effectuer des calculs, en l'occurrence un calcul de moyenne. Cette construction a l'avantage de faciliter la compréhension de la requête et on sait donc plus facilement ce sur quoi on calcule.

La seconde requête intéressante est celle présentant le classement des financements d'une association à une date donnée. La difficulté principale consiste ici à tenir compte des cotisations dans ce classement. L'idée consiste en la construction d'une table temporaire permettant de stocker la somme totale réunie par une association par le biais des cotisations en tenant compte de la date.

```
create table COTIS_TMP
(
    SOURCE        VARCHAR2(40)    NOT NULL,
    ARGENT_RECU   NUMBER(6)       NOT NULL
);

insert into COTIS_TMP values ('Cotisations',(
    SELECT COUNT(*)*(
        SELECT MONTANT_COTISATION FROM ASSOCIATIONS
        WHERE NOM_ASSO = 'EIRBWARE'
    )
    FROM COTISER
    WHERE (DATE_INSCRIPTION < &date) AND (NOM_ASSO = &var )
    GROUP BY NOM_ASSO)
);
```

On réalise ensuite deux requêtes imbriquées l'une dans l' autre telles que la première génère la liste des sources de financement d'une association associé au montant reçu par l'association par ces sources de financement. La seconde sélectionne ensuite les sources de financement et les sommes reçues dans une union avec la table temporaire mentionnée plus tôt que l'on classe en fonction de la somme reçue. Cela nous permet d'obtenir la liste complète des financements d'une association avant de les classer selon la somme d'argent reçue par l'association en question.

```
SELECT SOURCE, ARGENT_RECU
FROM
(SELECT NOM_FIN AS SOURCE, SUM(SOMME) AS ARGENT_RECU
FROM FINANCER JOIN SOURCES_FINANCEMENT ON FINANCER.ID_FIN = SOURCES_FINANCEMENT.ID_FIN
WHERE (NOM_ASSO = &asso)
GROUP BY SOURCES_FINANCEMENT.NOM_FIN)
UNION
SELECT * FROM COTIS_TMP
ORDER BY ARGENT_RECU DESC;
```

Le reste des requêtes statistiques consistent généralement en une simple utilisation de fonction mathématique comme `AVG(<attribut>)` parfois accompagnée d'un `ORDER BY` dans des requêtes de la forme :

```
SELECT TITRE_EVENT, AVG(NOTE) AS NOTE
```

```
FROM PARTICIPER JOIN EVENEMENTS ON PARTICIPER.ID_EVENT = EVENEMENTS.ID_EVENT
GROUP BY TITRE_EVENT
ORDER BY NOTE DESC;
```

4.2.3 Requêtes d'ajout, de suppression et de modification

Comme nous l'avons vu plus tôt il nous faut pouvoir modifier certains attributs dans certaines tables voire pouvoir ajouter de nouvelles lignes à certaines tables.

L'ajout d'une nouvelle ligne dans une table pose la question de la mise à jour de l'identifiant. Nous avons donc implémenté des séquences permettant la mise à jour automatique des identifiants. Il nous suffit alors d'appeler cette séquence pour mettre à jour une variable utilisée dans la requête d'ajout concernée. Par exemple, l'ajout d'un nouvel adhérent se présente sous la forme suivante :

```
PROMPT "Ajout d'un adhérent"
define id_personne = ID_PERS_SEQ.nextval
insert into PERSONNES values (&id_personne, &nom_pers, &prenom_pers);
insert into ADHERENTS values (&login, &e-mail , &filiere, &promo, &mdp, &id_personne);
```

Enfin la modification d'un attribut dans une ligne doit être possible dans certains cas précis. Par exemple il faut pouvoir modifier le mot de passe d'un adhérent :

```
PROMPT "Modification du mdp de l'adhérent"
update ADHERENTS set MDP = &nouveau_mdp where LOGIN_ADHERENT = &login;
```

5 Utilisation

Nous avons choisi de réaliser une interface texte dans le terminal. Vous trouverez dans cette partie les commandes à effectuer avant de pour pouvoir utiliser notre base de données suivi du mode d'emploi de notre base de données une fois que vous êtes connectés à Oracle.

5.1 Prérequis

Avant de pouvoir consulter notre base de données il faut se connecter à la machine Oracle. Si vous travaillez depuis votre ordinateur portable vous devez au préalable vous connecter en ssh au réseau de l'ENSEIRB-MATMECA *via* :

```
ssh login@ssh.enseirb-matmeca.fr
```

Connectez vous ensuite à la machine Oracle avant de définir les variables d'environnement nécessaires à l'utilisation de la base de données *via* :

```
ssh oracle
export PATH=$PATH:/home/oracle/app/oracle/product/11.2.0/dbhome_1/bin
export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1
export ORACLE_SID=oracle
export ORACLE_HOME_LISTNER=$ORACLE_HOME
```

Vous pouvez vous passez de définir ces variables à chaque fois en ajoutant les `export` ci-dessus dans le .bashrc. On se connecte ensuite à SQL*Plus *via* la commande `sqlplus`. Rentrez vote mot de passe, défini par défaut égal à voter login.

Il reste à créer la base de données et à remplir les tables avant de pouvoir manipuler la base. On tape pour cela les deux commandes `@base` puis `@donnees`. On peut maintenant utiliser notre base.

5.2 Interface utilisateur

Maintenant que la machine est opérationnelle nous allons voir l'ensemble des opérations qui seront disponibles. Pour afficher l'ensemble des commandes disponibles tapez `@help`. Vous disposerez ensuite d'un ensemble d'opérations vous permettant de consulter ou de modifier la base. L'interface vous demandera alors de saisir différentes valeurs selon ce que vous souhaitez consulter ou modifier. Notez qu'il faut mettre des ' ' autour des chaînes de caractères et des dates. Ces dernières sont par ailleurs au format '11-DEC-18'. Vous disposez à présent de toutes les informations nécessaires pour faire fonctionner notre base de données.

Commentaires

Nous voulions à l'origine implémenter une interface web utilisant SQL*Plus d'Oracle et PHP. Nous avons cependant abandonner l'idée faute de temps et de maîtrise de ces outils. Cela nous a toutefois poussé à nous pencher sur ces questions, engendrant un travail de recherches conséquent et enrichissant bien qu'il ne découle sur aucun résultat tangible. Nous nous sommes en effet rendus compte que choisir MySQL au lieu de SQL*Plus nous aurait largement facilité la tâche sur ce plan.

Webographie

Nous avons utilisé les sources suivantes pour nous documenter :

— https ://moodle.bordeaux-inp.fr/pluginfile.php/71724/mod_resource/content/1/cours_conception.pdf