GMIN210 TP Visualisation de graphes de tables (à rendre)

1. Enoncé

L'objectif du travail demandé est de permettre une confrontation visuelle de schémas relationnels de bases de données possiblement hébergées sous des serveurs de données hétérogènes (SGBDs Oracle, Mysql et Postgresql notamment). Les rôles attendus d'un tel travail pourraient être de faciliter l'intégration de sources de données ou bien de faciliter l'interrogation de schémas de données fragmentés. A cet effet, les vues des catalogues de données seront consultés et livreront des renseignements attendus sur :

- les tables (nom, schéma, espace de table, ...)
- les attributs de ces tables et leur type de données
- les contraintes portant sur ces attributs et le type de ces contraintes (clé p rimaire, clé étrangère, contrainte d'unicité, de domaine, de non nullité, ...).

Vous trouverez en section annexe, des éléments informationnels sur les vues des catalogues de données proposés au sein d'Oracle, Mysql et Postgresql.

Le travail comprend différentes étapes qui vous sont listées ci-dessous :

1. **implantation de différentes bases de données** sous différents serveurs de bases de données (schémas fournis dans une section annexe). Les schémas relationnels sont associés logiquement et il peut donc s'avérer d'intérêt de composer de l'information provenant de l'intégration de ces différents schémas.
2. **définition d'une couche logique applicative**. La couche métier considérée va permettre de manipuler les principaux éléments structurels des schémas des bases de données relationnelles comme autant de classes métier et ainsi d'en faciliter l'accès aux données du schéma (métadonnées). Ces informations pourront êt re ensuite exploitées pour reconstruire une vision globale sur l'ensemble des schémas pro venant des bases de données distribuées.
   Un diagramme de classes UML vous est donné ci-dessous pour vous aider à définir les classes dont les objets pourront être rendus persistants. Ces classes peuvent suivre le modèle de programmation POJO (pour Plain Old Java Object).

3. **exploitation de la solution de projection Objet-Relationnel Hibernate** afin de disposer de tous les éléments structurels (métadonnées) des schémas de base de données comme autant d'objets distribués. Vous définirez à cet effet les fichiers de mapping nécessaires et vous proposerez les classes applicatives qui vont vous permettre de manipuler les objets Table, Attribut ou Contrainte.
4. **définition de classes Java de restitution et de visualisation** des graphes de relations (un graphe par schéma de base de données). Un exemple illustratif construit à l'aide des APIs Jung et SWING vous est fourni à titre d'exemple.
5. **Facultatif** Aller vers l'intégration ou la consultation de sources de données distribuées.

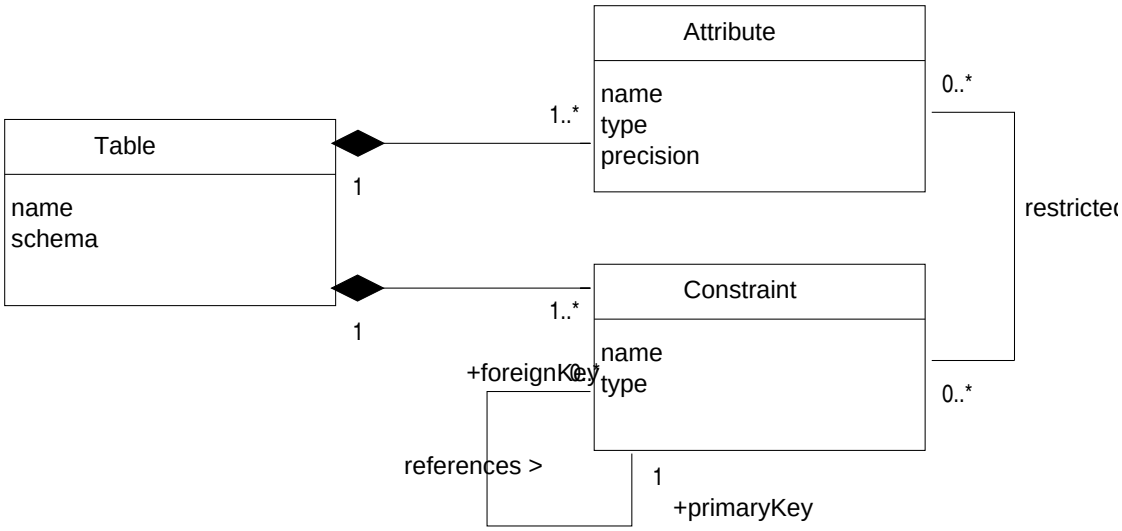

FIG. 1 – Diagramme de classes : éléments structurels BDR

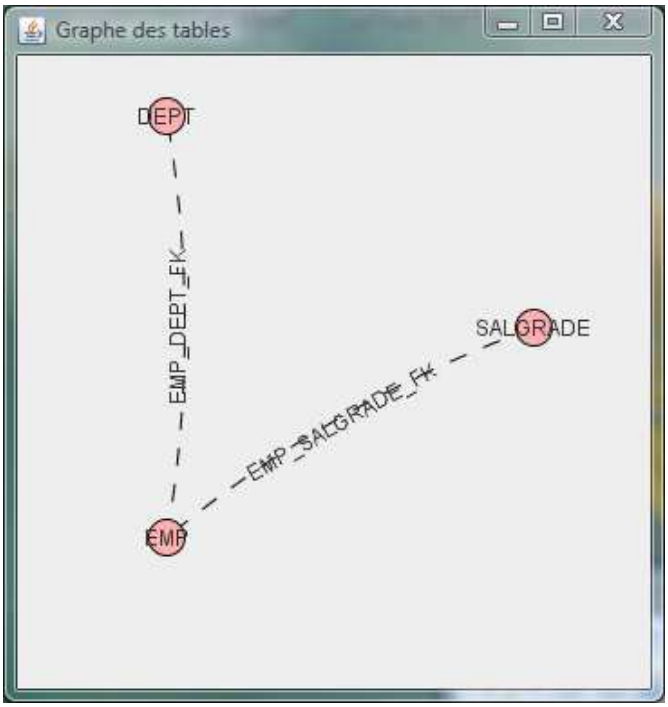

FIG. 2 – Exemple de restitution visuelle

Vous pouvez réfléchir à comment exploiter votre travail et mettre en oeuvre de quelques idées pour établir des passerelles entre les tables ou les attributs provenant de différents schémas ou bien pour faciliter la consultation concertée de plusieurs sources de données dont l'information est associée logiquement. Des pistes de travail sont proposées à titre d'exemple :
- procéder par mesure de similarité (distance Levenshtein par ex.) sur les noms des tables et/ou attributs
- définir une métrique sur la base du nombre d'attributs (et de leur type), du nombre de tuples, des contraintes posées sur chaque couple de tables comparées
- proposer des exemples de mise en oeuvre de consultation complexe dans le cadre de schémas fragmentés

Vous pouvez exploiter le modèle MVC dans votre logique de programmation.

2. Annexe : schémas relationnels

Trois schémas relationnels sont proposés. Vous vous doterez d'un jeu de tuples pertinent pour chacun des schémas :

2.1 Emp-Dept

- Dept (num, nom, budget, specialite, ville)
- Emp (num, nom, prenom, fonction, salaire, commission, n_sup, departement)
  Avec Emp(n_sup) $\subseteq$ Emp(num) et Emp(departement) $\subseteq$ Dept(num)

2.2 Ville-Departement-Region

- Region (nom, situationGeographique)
- Departement (num, nom, nbreHabitants, region)
  Avec Departement(region) $\subseteq$ Region(nom)
- Ville (nom, nbreHabitants, tempMoyenne, departement)
  avec Ville(departement) $\subseteq$ Departement(num)

2.3 Ville-Impôt ISF

- Ville (nom, nbreHabitantsImpot, annee, valeurMoyISF)

3. Annexe : catalogues de données

3.1 Oracle

Nous rappelons que le dictionnaire de données (ou méta-schéma) est un ensemble de tables dans lesquelles sont stockées les descriptions des objets de la base.
Les tables de ce dictionnaire peuvent être consultées au moyen du langage SQL. Des vues de ces tables permettent à l'utilisateur de voir les objets qui lui appartiennent (tables préfixées par USER) ou sur lesquels il a des droits (tables préfixées par ALL). L'administrateur a pour sa part accès à toutes les

vues (les tables précédentes ainsi que les tables préfixées par DBA).
`Quelques vues du dictionnaire de données :`

- USER_TABLES : tables et vues créées par l'utilisateur ;
- USER_CATALOG (ou CAT) : tables et vues sur lesquelles l'utilisateur a des droits à l'exception des tables et vues du dictionnaire de données ;
- USER_TAB_COLUMNS (ou COLS) : colonne de chaque table ou vue créée par l'utilisateur courant ;
- USER_CONSTRAINTS : définition des contraintes pour les tables des utilisateurs ;
- USER_CONS_COLUMNS : colonnes qui interviennent dans les définitions des contraintes ;
- USER_TAB_PRIVS : droits attribués et/ou reçus par l'utilisateur
- USER_SYS_PRIVS : privilèges donnés à l'utilisateur de manière générale ;
- USER_TAB_PRIVS_MADE : droits attribués par l'utilisateur ;
- USER_TAB_PRIVS_RECD : droits reçus par l'utilisateur ;
- ALL_CATALOG : liste de tous les objets accessibles par l'utilisateur ;
- ALL_TABLES Description des tables accessibles par l'utilisateur.

3.2 Mysql

Pour Mysql, le dictionnaire de données est disponible depuis le schéma information_schema. Les tables de ce schéma peuvent être consultées au moyen du langage SQL. `Quelques tables du dictionnaire de données`

- information_schema.tables
- information_schema.columns
- information_schema.key_column_usage
- information_schema.table_constraints

3.3 Postgresql

Pour Postgresql, les vues du dictionnaire de données (également information_schema), sont préfixées par "pg_". Les tables de ce schéma peuvent être consultées au moyen du langage SQL. `Quelques tables du dictionnaire de données`

- pg_tables
- pg_attribute
- pg_class
- pg_constraint