SAE BD/MATH.
SAE n°S4.03
Au delà du relationnel

1 La SAE

Le but de cette SAE est de vous fournir une vision générale des différents modèles de bases de données et des langages de requêtes utilisés pour les interroger. Cette SAE a pour objectif de vous amener à : (1) faire preuve d'initiative en recherchant des informations et en consultant la documentation technique, (2) appliquer les connaissances acquises dans la matière principale (la ressource R4.03), ainsi que dans d'autres domaines de votre formation, tels que les mathématiques et l'anglais.
La tâche centrale de cette SAE consiste à implémenter des requêtes dans différents modèles de données et les langages de requêtes associés.

2 L'organisation du travail

La SAE-S4.03 sera réalisée en **groupe de trois étudiants**, encadrés par les enseignants lors des séances de discussion. L'évaluation consistera en un devoir sur table portant sur les notions théoriques ou pratiques abordées pendant cette SAE. Ce contrôle sera individuel et permettra d'évaluer les connaissances acquises par chaque étudiant.

3 Base de données

Nous considérons une petite partie de la base de données que nous avons conçue en TD concernant les vols proposés par les compagnies aériennes. La partie de la spécification que nous allons considérer est la suivante :

> Les compagnies aériennes proposent plusieurs vols. Un vol est identifié par une compagnie, un numéro et une date-heure de départ. Pour chaque vol notre base stocke les villes, les pays, les aéroports et terminaux de départ et d'arrivée, les jours et les heures (prévues) d' arrivée. Par exemple, le vol Air France 0442 du 10/07/2023, est le vol Paris(FR)-Rio(BR) départ à 23h 30min de l'aéroport Charles de Gaulle, terminal 2E, arrivant à l'aéroport international Antonio Carlos Jobim le 11/07/2023 à 5h 30min, terminal 1. Nous ne considérons que des vols directs. Il est bon de rappeler que le nom d'un aéroport est unique.

3.1 Modèle relationnel

1. Créer votre base de données relationnelle à partir de cette petite spécification.
2. Nous souhaitons trouver les villes que nous pouvons atteindre en partant de Paris. Écrire en SQL (Oracle) les requêtes suivantes :
   (a) Donner les villes que nous pouvons atteindre par vols directs qui partent de Paris.

(b) En considérant les horaires des vols, veuillez fournir la liste des villes accessibles depuis Paris avec un vol comprenant UNE correspondance. L'objectif est de permettre aux passagers de réaliser leur correspondance.

(c) En considérant les horaires des vols, veuillez fournir la liste des villes accessibles depuis Paris avec un vol comprenant DEUX correspondances.

(d) Veuillez fournir la liste des villes accessibles depuis Paris, en tenant compte des horaires de vol, avec des vols directs ou un nombre quelconque de correspondances.

REMARQUE : Afin de permettre le traitement des requêtes récursives, des améliorations ont été apportées au langage SQL. Votre devez étudier ces enrichissements, en vous référant aux sources suivantes :

— `https://oracle-base.com/articles/11g/recursive-subquery-factoring-11gr2#basic`

— `https://www.youtube.com/watch?v=RIs9HaceYIc`

3.2 Modèle objet-relationnel

Nous souhaitons modéliser les informations de nos vols en utilisant un modèle objet-relationnel. Dans ce cadre, nous envisageons de faire une table qui contient une autre table (informant sur l'équipage) et une liste des trois indices de qualité, avec leur poids actuel. Chaque tuple de la table représente un vol. La figure 1 présente un exemple d'une petite instance du modèle.

NumVol	AeroDep	DateHeureDep	AeroArr	DateHeureArr	Equipage		IndicesQualite
					Nom	Fonction	
AF442	CDG	2024.06.11 13 :00	GIG	2024.06.11 19 :00	Goscinny Uderzo	Pilote Commissaire	[(carbone, 3, 4), (securite, 4, 5), (prix, 4, 3))]

FIGURE 1 – Exemple d'une instance dans le modèle objet-relationnel

1. Utiliser ORACLE pour implémenter votre base dans le modèle objet-relationnel. Vous devez définir les types complexes comme mentionné dans nos cours.
2. Écrire les requêtes suivantes
   (a) Pour chaque vol, donner le nombre de personnes de l'équipage, par fonction.
   (b) Pour chaque pilote, indiquer combien des vols lui sont associés.
   (c) L'*impact* d'un indice de qualité est donné par le produit de sa valeur et du poids que lui est attribué. Pour chaque vol, indiquer l'impact de chaque indice de qualité.
   (d) Pour chaque indice de qualité, calculer son impact moyen.

Dans les section suivantes, vous n'avez pas besoin de considérer les attributs concernant *Date-Heure* des vols de manière réaliste. Il suffit d'associer un entier comme valeur possible pour faire des comparaisons. Ainsi, vous pouvez dire que *DateHArr* d'un vol est 3 alors que *DateHDep* d'un autre vol est 5 et donc $DateHArr < DateHDep$.

3.3 Modèle Logique

Pour cet exercice, vous pouvez utiliser AbcDatalog, une implémentation de Datalog disponible à l'adresse suivante : `https://abcdatalog.seas.harvard.edu/`. Cette implémentation est simple à utiliser, mais certaines fonctionnalités ne sont pas disponibles, comme les

opérations de comparaison telles que $\geq$ ou $\leq$. Toutefois, vous pouvez toujours définir un prédicat, par exemple $plusPetit(1, 2)$ pour dire que 1 est plus petit que 2. Vous pouvez télécharger le fichier .jar correspondant depuis CELENE, sur le site de notre SAE. Avant de commencer, nous vous recommandons de lire attentivement les instructions sur la syntaxe de cette implémentation sur le site web d'AbcDatalog.

Considérons la base de données VOL qui contient des informations sur les vols directs.

1. Identifiez quels sont les prédicats extensionnels de votre base de données déductive.
2. Écrivez un programme Datalog qui permet de lister toutes les villes connectées par des vols (directs ou avec connexions) possibles à partir de d'une instance de la base VOL donnée. Les horaires de départ et d'arrivée du trajet complet doivent être indiqués.
3. Écrivez un programme Datalog qui permet de lister toutes les villes connectées par des vols avec un nombre impair de connexions. Les horaires de départ et d'arrivée du trajet complet doivent être indiqués.

3.4 Modèle graphe

Pour cet exercice, vous devez utiliser le SGBD Neo4J et le langage Cypher.

1. Proposez une modélisation de la base de données VOL dans le modèle graphe. Expliquez vos choix de modélisation.
2. Implémentez votre base de données en Neo4J.
3. Écrivez une requête en Cypher qui correspond à la requête 2d que vous avez écrite en SQL (Section 3.1). Faites différentes versions :
   (a) Version 1 : sans prendre en compte les horaires.
   (b) Version 2 : en prenant en compte les horaires.
   (c) Version 3 : pour les vols avec connexion, indiquez les villes intermédiaires.

   Remarque : lorsque vous écrivez vos requêtes en Cypher (versions 2 et 3), gardez à l'esprit que vous pouvez modifier le graphe. Prenez inspiration de vos requêtes Datalog pour vous aider à concevoir la requête dans le modèle graphe.

Vous trouverez sur CELENE des instructions pour installer Neo4J.

4 Évaluation

L'évaluation du projet SAE sera effectuée individuellement par un examen écrit, dans lequel chaque étudiant devra démontrer une maîtrise complète de toutes les parties du projet. L'examen portera sur les concepts manipulés pendant le projet SAE ainsi que sur les particularités du projet de chaque étudiant.

CALENDRIER : Contrôle le **06 avril 2023**

RENDUS :

— *Chaque étudiant* doit fournir un dossier en version papier qui contient toutes les requêtes réalisées, ainsi que les explications demandées dans le projet. Ce dossier doit être annexé à sa copie d'examen et pourra être consulté pendant l'évaluation pour aider l'étudiant à répondre aux questions.

— Chaque groupe doit soumettre les scripts des requêtes réalisées sur CELENE, accompagnés de leur jeu de tests, avant la date limite du 06/04/2023. Cela permettra éventuellement de les exécuter lors de l'évaluation. *Veuillez noter qu'un seul rendu par groupe est requis.*

pagebreak

5 Rappel : Instructions pour travailler sur Oracle

5.1 La routine de travail

1. Travailler avec un éditeur. Faites vos requêtes dans un fichier que vous nommez, par exemple, *tpX.sql*.
2. Ouvrir un terminal dans le même répertoire de votre fichier *tpX.sql* et lancer la connexion Oracle.
3. Vous pouvez lancer le script de *tpX.sql* en faisant : @*tpX* dans SQLPLUS ou faire copier-coller de chaque requête. Il est important d'avoir votre fichier avec vos requêtes (que vous sauvegardez).
4. Faire des fichiers séparés. Par exemple : un fichier *creationBD*1.*sql*, pour la création de la base ; un fichier *insertionBD*1.*sql* pour les insertion dans la base BD1 ; un fichier *reqBD*1.*sql* pour les requêtes sur la base BD1.

5.2 Des commandes en vrac

— Connectez vous à Oracle. À l'IUT vous devez faire :

```
sqlplus <nom>@ora12
```

ou mieux

```
rlwrap sqlplus <nom>@ora12
```

qui donnera plus de souplesse à l'édition en ligne.

— Changez votre mot de passe :

```
alter user <login> identified by <nouveauMotPasse>;
```

ou utiliser *passwd* comme sur un shell.

— Pour augmenter la taille des lignes : `set LINESIZE 500`
— Pour diminuer la taille d'une colonne : `column cname format A15`
— Pour voir l'ensemble de tables : `select * from tab;`
— Pour effacer les tables qui apparaissent après un drop : `purge recyclebin;`
— Pour voir le schéma d'une table : `describe` nomTable ;
— Pour faire la suppression de toutes vos tables (avec les contraintes). La requête

```
select 'drop table" '  || table_name || ' " cascade constraints; ' from use_table;
```

vous donne comme réponse la liste de tables à supprimer. Lancez ensuite le script généré.

— Database SQL Language Quick Reference
`https://docs.oracle.com/cd/E11882_01/server.112/e41085/sqlqr01001.htm#SQLQR110`