Master 2 BDIA - Informatique Décisionnelle

Année 2022-2023

TD et TP d'Informatique Décisionnelle • à rendre avant le 28 février 2023

Cadre général

Les TD et TP de ce module ont un unique sujet, qui servira de cas d'exemple afin d'avoir une vue d'ensemble de l'informatique décisionnelle, et permettra d'appliquer toutes les étapes, de la construction d'un *data warehouse*, à son exploitation et jusqu'à la présentation des résultats. Des propositions de technologies sont fournies pour chaque étape, mais il est possible d'en choisir une non citée.

Contraintes pour la réalisation

À l'issue des séances vous remettrez un rapport, le code produit et vous ferez une présentation orale de 15 minutes maximum de votre travail. Ces éléments seront évalués pour constituer la note d'épreuve pratique du module.

Dans le rapport, devront figurer votre analyse du sujet, vos choix justifiés, les schémas, les requêtes et les analyses réalisées. Le document est à rendre par **groupe de trois étudiants maximum**. Les aspects techniques, la lisibilité du code, la qualité de la rédaction seront évalués (dont les justifications des choix). Il conviendra de présenter, d'expliquer et de justifier les technologies retenues pour la réalisation si vous n'adoptez pas celles proposées.

Pour le document, les consignes **strictes** de présentation sont : format PDF, au moins 15 pages, police 12pts maximum, marges 2cm maximum. Le rapport doit comporter au moins les éléments suivants :

- **une analyse** du sujet, description détaillée des données disponibles, définition du cadre des analyses et des objectifs ;
- **la spécification du schéma du *data warehouse*** et des *data marts* ;
- **la justification** des choix envisagés et appliqués à toutes les étapes ;
- **une description de vos requêtes d'analyse**, les requêtes elles mêmes, leur restitution et les conclusions que vous en avez tiré ;
- **un retour d'expérience et des propositions d'amélioration** des choix qui ont été retenus ;
- **une évaluation / comparaison des performances** au travers de la mesure des temps d'exécution et de l'occupation mémoire ;
- une documentation pour compiler et exécuter votre code.

Il est fortement conseillé de rédiger le rapport en utilisant LaTeX car il permet d'inclure facilement des extraits de code source avec une mise en évidence des éléments syntaxiques. On trouve des applications permettant de rédiger collaborativement des documents LaTeX (`https://fr.sharelatex.com/`, `https://fr.overleaf.com/` ou encore avec le VPN pour l'accès à un serveur du département IEM `http://iemgit:9080/`). Pour le partage de code, l'outil git et son interface Web GitHub sont recommandés.

Dans le document que vous remettrez et qui sera évalué, il est conseillé d'utiliser des schémas pour illustrer vos propos.

Une archive [1] (tgz ou zip) de l'ensemble du code source de vos programmes et requêtes devra être produite et déposée sur Plubel dans la zone de dépôt prévue à cet effet.

L'évaluation sera faite sur la base de la présentation orale, du contenu du rapport et de la qualité technique de la réalisation.

1. Les autres formats d'archive ne sont pas acceptés

Jeu de données

Un jeu de données vous est fourni. Il s'agit de données réelles de l'application Yelp [2], qui regroupe des avis d'utilisateurs sur de nombreux commerces. C'est un jeu de données volumineux, de plusieurs Go, dispersé dans plusieurs sources. Nous les avons toutes mises à disposition sur le serveur `stendhal.iem` [3] accessible depuis le réseau des salles ou bien avec le VPN :

- un fichier csv "tip" est disponible sous `/data/M2BDIA-ID-Ressources/dataset`
- deux fichiers json "checkin" et "business" sont disponibles sous `/data/M2BDIA-ID-Ressources/dataset`
- une base de données PostgreSQL, contenant des informations sur :
  - les utilisateurs ou "user" (tables `yelp.user`, `yelp.friend` et `yelp.elite`)
  - les avis ou "review" (table `yelp.review`)

Globalement les données sont organisées en 5 grandes parties et décrivent :

- des commerces, leur localisation, leurs catégories et de nombreux attributs complémentaires des services offerts comme leurs horaires d'ouvertures mais aussi d'autres attributs servant à donner des détails sur le commerce, comme la présence d'un parking, le type de régime alimentaire pour un restaurant, la spécialisation pour un salon de coiffure, etc. ;
- des avis, avec une valeur, et un commentaire textuel, ainsi que les réactions des autres utilisateurs sur les avis ;
- des utilisateurs, leur prénom, leur id, leurs amis, la date de création du compte, des valeurs indiquant le type des évaluations, ainsi que des statistiques liées aux comptes, et les années pour lesquels un utilisateur était "élite" ;
- des tips, qui sont des avis donnés sous un format plus court, moins détaillé ;
- des checkins, qui sont des dates auxquelles un commerce a reçu la visite d'un utilisateur.

Références :

- `https://www.yelp.com/dataset/documentation/main`

Éléments techniques pour la connexion

- Connexion à la base de donnée hébergée par PostgreSQL :
  - port : `5432`
  - nom de la base : `tpid2020` (schéma `yelp`)
  - login : `tpid`
  - mot de passe : `tpid`
  - commande pour se connecter au shell SQL de Postgres : `psql -U tpid tpid2020`
- Connexion à la base de données Oracle pour construire votre *data warehouse* :
  - port : `1521`
  - base : `enss2022`
  - login : votre login des salles informatique IEM
  - mot de passe : identique au login
  - avant de lancer `sqlplus`, fixer les variables d'environnement pour Oracle : `. /opt/oraenv.sh`
  - sur votre poste de travail vous pouvez installer 3 outils qui vous seront potentiellement utiles : SQL Developper, Oracle Warehouse Builder et Oracle OLAP Analytic Workspace Manager, ces programmes sont disponibles gratuitement sur le site d'Oracle après avoir créé un compte, ou bien directement dans le répertoire `/data/M2BDIA-ID-Ressources/tools` du serveur stendhal. Avant des les utiliser penser à fixer les variables d'environnement pour le JDK 1.8.

2. `https://www.yelp.fr/paris`
3. ce serveur dispose de 48 cœurs, 256Go de RAM, il héberge le SGBD Oracle en version 19, le SGBD PostgreSQL en version 10.21

Objectif

Pour guider vos analyses, les questions suivantes peuvent être utilisées :
— quels sont les utilisateurs les plus actifs ?
— quels sont les types de commerces ?
— quelle est la moyenne des notes par type (suivant plusieurs attributs) ?
— peut-on distinguer une évolution temporelle des notes ?
— quelle est la moyenne des notes par zone géographique ?
— certaines zones concentrent-elles des restaurants très bien évalués ?
— quels types d'utilisateurs reçoivent le plus de réactions à leurs évaluations ?
— quels sont les commerces dont les notes augmentent le plus ?
— peut-on segmenter les utilisateurs en fonction des types de commerces qu'ils évaluent ?
— quels sont les commerces les plus visités ?
— quels types de commerces sont les plus notés, les mieux notés, les moins bien notés ?
— les caractéristiques annexes comme les menus végétariens dans les restaurants influent-elles sur la notation ?

Ces questions ne sont qu'un guide pour votre analyse des besoins. Il faut déterminer des différents types d'utilisateurs intéressés par les analyses, et décrire un ou plusieurs cas d'utilisation des données. Ce travail est à faire en dehors des séances de TP/TD et guidera vos analyses ainsi que la conception du schéma de l'entrepôt (*data warehouse*, *data mart*, *data cubes*).

Spécification du schéma du *data warehouse*

Choisir une approche (Kimball ou Inmon). En fonction des domaines d'analyse sélectionnés et des requêtes identifiées mais aussi en explorant les données disponibles, construire les schémas du *data warehouse* et des *data marts* (étoile, flocon, constellation), identifier plusieurs mesures ou tables de faits et les dimensions associées.

Attention : un schéma de *data warehouse* n'est pas un schéma de base de données transactionnelle. Il n'a donc pas besoin de suivre la 3^{e} forme normale qui est souvent contre-productive dans ce contexte. Il faut privilégier la simplicité des requêtes.

Intégration des données

Cette étape doit être réalisée en moins de 2 séances.

Le but de ce cas pratique est tout d'abord d'intégrer les données mises à disposition dans un unique *data warehouse*. Il convient de réfléchir et de modéliser le schéma en fonction des besoins et des attentes.

Pour réaliser cette étape, nous proposons de construire un ETL avec Spark en Scala. Il est également possible de l'utiliser en Java, mais la syntaxe est moins verbeuse en Scala. Les données doivent être extraites des différentes sources, transformées dans l'ETL, puis insérées dans le *data warehouse* cible dans un schéma propice à leur exploitation.

Lors de la construction d'un *data warehouse*, il faut garder à l'esprit que le schéma des données n'a pas forcément besoin d'être normalisé, étant donné que l'on cherche à optimiser l'exécution des requêtes sur un sujet donné. Les données peuvent donc être modifiées en conséquence, certaines parties jugées inutiles pour votre étude, peuvent ne pas être intégrées au *data warehouse*.

Références :
— `https://spark.apache.org/docs/latest/sql-getting-started.html`
— Exemple d'utilisation de Spark en ETL :
`https://medium.com/analytics-vidhya/etl-pipeline-using-spark-sql-746bbfae4d03`

Notes : les extractions (surtout depuis une base de données) ainsi que les insertions pouvant parfois prendre du temps, il convient de réfléchir au schéma avant de lancer un traitement complet, et

également de faire des tests sur des parties réduites des données, ou de faire tourner le programme uniquement pour une partie du schéma.

Des optimisations disponibles avec Spark peuvent également être utilisées, par exemple il est possible de paralléliser l'exécution d'une requête en fonction d'une colonne de la table de la manière suivante (la colonne doit être de type date ou numérique) :

```
spark.read
  .option("partitionColumn", "yelping_since")
  .option("lowerBound", "2004-10-12")
  .option("upperBound", "2019-12-13")
  .option("numPartitions", 50)
  .jdbc(urlPG, "yelp.\"user\"", connectionPropertiesPG)
```

Pour en faciliter l'utilisation, certaines tables de la base de données PostgreSQL possèdent une colonne `spark_partition`, contenant un entier de 0 à 99 équitablement répartis entre les tuples.

Oracle étant également légèrement capricieux concernant certains types (notamment les booléens), il peut être nécessaire de guider l'insertion avec Spark pour obtenir le résultat souhaité.

Une solution possible est d'implémenter la classe `JdbcDialect` de Spark pour redéfinir le mapping des types. L'exemple suivant modifie le mapping pour le type `Boolean` et pour le type `String` (par défaut un `VARCHAR2(255)` est créé, ce qui peut être insuffisant).

```
import org.apache.spark.sql.jdbc.{JdbcDialect, JdbcDialects, JdbcType}
import org.apache.spark.sql.types._

class OracleDialect extends JdbcDialect {
  override def getJDBCType(dt: DataType): Option[JdbcType] = dt match {
    case BooleanType => Some(JdbcType("NUMBER(1)", java.sql.Types.INTEGER))
    case IntegerType => Some(JdbcType("NUMBER(10)", java.sql.Types.INTEGER))
    case LongType => Some(JdbcType("NUMBER(19)", java.sql.Types.BIGINT))
    case FloatType => Some(JdbcType("NUMBER(19, 4)", java.sql.Types.FLOAT))
    case DoubleType => Some(JdbcType("NUMBER(19, 4)", java.sql.Types.DOUBLE))
    case ByteType => Some(JdbcType("NUMBER(3)", java.sql.Types.SMALLINT))
    case ShortType => Some(JdbcType("NUMBER(5)", java.sql.Types.SMALLINT))
    case StringType => Some(JdbcType("VARCHAR2(4000)", java.sql.Types.VARCHAR))
    case DateType => Some(JdbcType("DATE", java.sql.Types.DATE))
    case _ => None
  }

  override def canHandle(url: String): Boolean = url.startsWith("jdbc:oracle")
}
```

Puis pour dire à Spark de s'en servir, à appeler avant l'insertion dans Oracle :

```
val dialect = new OracleDialect
JdbcDialects.registerDialect(dialect)
```

Interrogation du *data warehouse*

Les deux étapes suivantes (interrogation et présentation) sont prévues pour 3 séances.

Une fois que les données ont été intégrées dans le *data warehouse*, puis organisées dans les *data marts*, il faut vérifier que les schémas permettent de répondre aux questions que l'on se pose sur le jeu de données (votre cas d'utilisation), et que les requêtes peuvent s'exécuter en un temps raisonnable. Dans ce cas, spécifier les cubes, les requêtes d'agrégations et les autres types de requêtes.

Après avoir exécuté les requêtes, interprétez les résultats et effectuer une étude des performances d'exécution des requêtes et éventuellement des optimisations de schéma.

Références :

- https://docs.oracle.com/en/database/oracle/oracle-database/19/olaug/index.html
- pour le calcul du rang, les top-k queries, les fenêtres temporelles : https://docs.oracle.com/cd/E11882_01/server.112/e25554/analysis.htm#DWHSG021
- pour des opérations plus avancées sur les cubes (notamment les cubes hiérarchiques) https://docs.oracle.com/cd/E11882_01/server.112/e25554/aggreg.htm#DWHSG020
- pour les optimisations :
  - sur les index : https://docs.oracle.com/cd/E11882_01/server.112/e25554/indexes.htm#DWHSG8139
  - les vues matérialisées : https://docs.oracle.com/cd/E11882_01/server.112/e25554/basicmv.htm#DWHSG00831
- il y a des extensions propriétaires pour la représentation des dimensions : https://docs.oracle.com/cd/E11882_01/server.112/e25554/dimen.htm#DWHSG8269

Présentation des résultats

Les résultats obtenus doivent ensuite être mis en valeur grâce à une présentation permettant de les comprendre facilement, y compris pour un public non informaticien et qui n'a pas connaissance de la structure des données.

Pour cela, une représentation sous forme de *dashboard* est souvent un bon moyen de faire passer des informations en mettant en valeur les points importants des données. Pour le TP, il est possible d'utiliser Metabase, qui est assez léger à mettre en place et permet de se connecter à une base de données Oracle, afin de lancer des requêtes ou d'interroger des vues puis effectuer des visualisations directement à partir du résultat obtenu.

La présentation des résultats doit permettre d'apporter une vision synthétique et pertinente de ce qu'il est possible d'extraire des données, sans en altérer le sens. Dans votre rapport, cette présentation devra figurer en annexe sous la forme de diaporama ou de document pdf de synthèse, sans oublier de décrire les détails techniques.

Références :

- Téléchargement de Metabase : https://www.metabase.com/start/oss/jar.html
- Ajout du plugin Oracle : https://www.metabase.com/docs/latest/administration-guide/databases/oracle.html