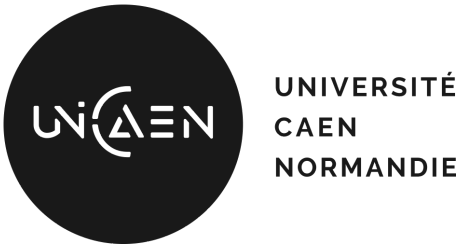

Master 1 Informatique
Bases de données avancées
Contrôle Continu 2

Romain ANDRES 21904263
Guillaume LEMONNIER 22007629

27 Octobre 2023

Contents

1 Bases de Données Avancées

1.1 Question 1

Le modèle en étoile:

- Entrepôt de données pour un ensemble de faits et de dimensions
- C'est un modèle relationnel
- Contient une unique table des faits avec:
  - contient un n-uplet par fait, décrit par ses mesures, ses attributs propres
  - une clé étrangère vers chacune des dimensions
- Et une unique table pour chaque dimension
- Chaque table de dimension contient les valeurs de cette dimension, décomposées en leurs constituants

On parle de modèle "maximalement dénormalisé".

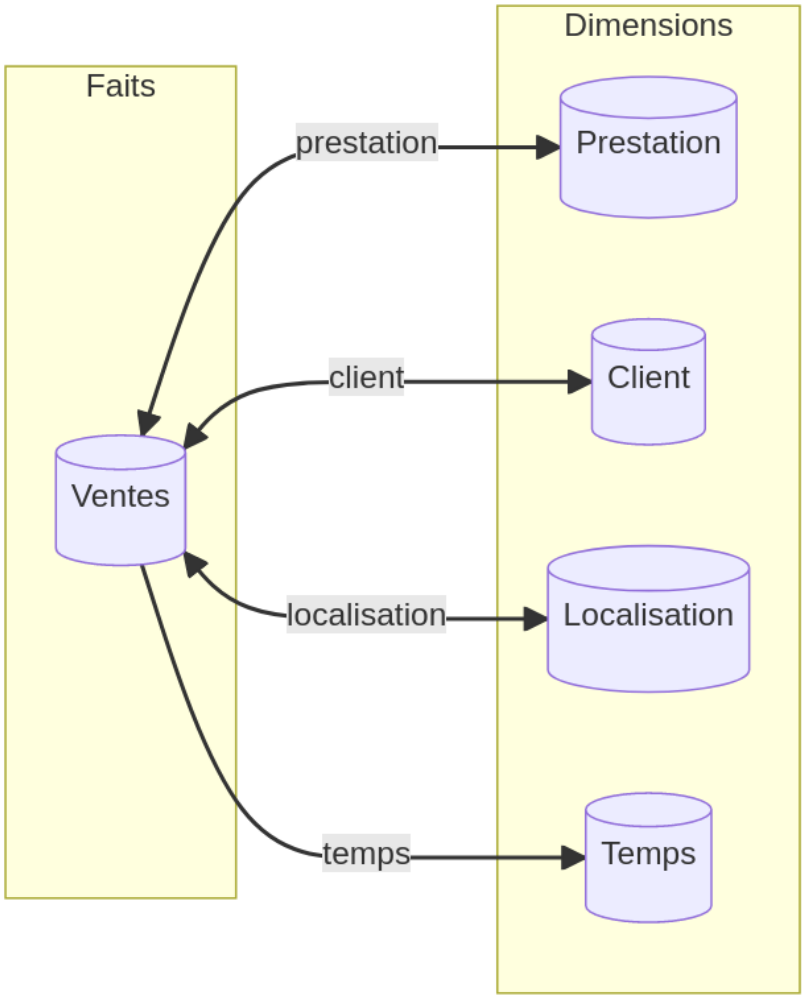

Figure 1: Une modélisation en étoile

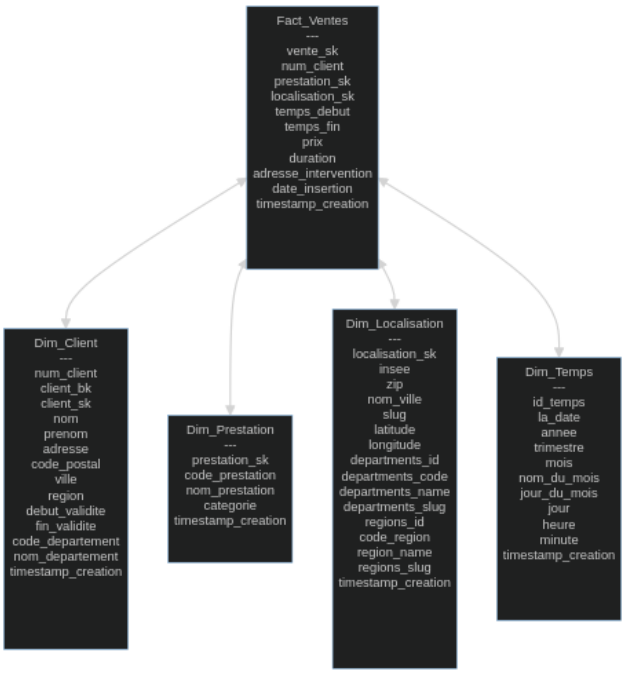

Figure 2: Premier aperçu des tables

1.2 Question 2 et 3

Voici notre script qui nous a permis de créer notre base de données SQLite. C'est la partie qui nous a donné le plus de difficultés durant ces TP, car nous avons dû la modifier à plusieurs reprises après avoir constaté des problèmes au fur et à mesure de nos manipulations sur Talend. Notre modélisation veille à insérer les éléments nécessaires pour que les données soient résilientes aux changements (adresses des clients) et pour permettre la traçabilité des données.

Dimension Client :
Nous avons introduit trois clés pour chaque client : une clé primaire, une clé de business, et une clé de substitution.

num_client sert de clé primaire pour identifier de manière unique chaque enregistrement dans la table.

client_bk est une clé métier pour identifier le client dans les systèmes d'entreprise.

client_sk est une clé de substitution qui permet de gérer les changements de dimension au fil du temps.

Ce choix nous permet de gérer les cas où un même client peut avoir différentes versions ou peut être représenté différemment dans d'autres systèmes.

Pour pallier au problème du client pouvant avoir plusieurs adresses, nous avons ajouté une colonne adresse dans la table Fact_Ventes.

Cela permet de conserver l'adresse spécifique à chaque transaction, garantissant ainsi une meilleure résilience aux changements de l'adresse du client.

Journalisation :
Nous avons utilisé le timestamp_creation dans toutes les tables. Cela permet de tracer quand chaque enregistrement a été créé, ce qui est essentiel pour le dépannage.

Pour finir nous avons essayé de respecter au mieux le principe du modèle en

étoile en répétant les données même si on trouvait cela très contre-intuitif. (Par exemple duration qui est une variable calculée au final) et nous sommes assez satisfaits du résultat final.
Vous pouvez retrouver ce fichier dans l'archive de rendu.

```
DROP TABLE IF EXISTS Fact_Ventes;
CREATE TABLE Fact_Ventes (
    vente_sk INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    num_client INT REFERENCES Dim_Client(num_client)
       NOT NULL,
    prestation_sk INT REFERENCES
       Dim_Prestation(prestation_sk) NOT NULL,
    localisation_sk INT REFERENCES
       Dim_Localisation(localisation_sk) NOT NULL,
    temps_debut INT REFERENCES Dim_Temps(id_temps) NOT
       NULL,
    temps_fin INT REFERENCES Dim_Temps(id_temps) NOT
       NULL,
    prix FLOAT NOT NULL,
    duration INTEGER NOT NULL,
    adresse_intervention TEXT NOT NULL,
    date_insertion TEXT NOT NULL,
    timestamp_creation TIMESTAMP DEFAULT
       CURRENT_TIMESTAMP
);


-- Table Dim_Client
DROP TABLE IF EXISTS Dim_Client;
CREATE TABLE Dim_Client (
    num_client INTEGER PRIMARY KEY AUTOINCREMENT NOT
       NULL,
    client_bk INT NOT NULL,
    client_sk INT NOT NULL,
    nom TEXT NOT NULL,
    prenom TEXT NOT NULL,
    adresse TEXT NOT NULL,
    code_postal TEXT NOT NULL,
    ville TEXT NOT NULL,
    region TEXT NOT NULL,
```

```
29      debut_validite DATETIME,
30      fin_validite DATETIME,
31      code_departement TEXT NOT NULL,
32      nom_departement TEXT NOT NULL,
33      timestamp_creation TIMESTAMP DEFAULT
            CURRENT_TIMESTAMP
34  );
35
36  -- Table Dim_Prestation
37  DROP TABLE IF EXISTS Dim_Prestation;
38  CREATE TABLE Dim_Prestation (
39      prestation_sk INTEGER PRIMARY KEY AUTOINCREMENT NOT
            NULL,
40      code_prestation TEXT NOT NULL,
41      nom_prestation TEXT NOT NULL,
42      categorie TEXT NOT NULL,
43      timestamp_creation TIMESTAMP DEFAULT
            CURRENT_TIMESTAMP
44  );
45
46  -- Table Dim_Localisation
47  DROP TABLE IF EXISTS Dim_Localisation;
48  CREATE TABLE Dim_Localisation (
49      localisation_sk INTEGER PRIMARY KEY AUTOINCREMENT
            NOT NULL,
50      insee TEXT NOT NULL,
51      zip TEXT NOT NULL,
52      nom_ville TEXT NOT NULL,
53      slug TEXT NOT NULL,
54      latitude FLOAT NOT NULL,
55      longitude FLOAT NOT NULL,
56      departments_id INT NOT NULL,
57      departments_code TEXT NOT NULL,
58      departments_name TEXT NOT NULL,
59      departments_slug TEXT NOT NULL,
60      regions_id INT NOT NULL,
61      code_region TEXT NOT NULL,
62      region_name TEXT NOT NULL,
63      regions_slug TEXT NOT NULL,
64      timestamp_creation TIMESTAMP DEFAULT
```

```
        CURRENT_TIMESTAMP
);

-- Table Dim_Temps
DROP TABLE IF EXISTS Dim_Temps;
CREATE TABLE Dim_Temps (
    id_temps INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
    la_date TEXT NOT NULL,
    annee INT NOT NULL,
    trimestre INT NOT NULL,
    mois INT NOT NULL,
    nom_du_mois TEXT NOT NULL,
    jour_du_mois TEXT NOT NULL,
    jour INT NOT NULL,
    heure INT NOT NULL,
    minute INT,
    timestamp_creation TIMESTAMP DEFAULT
        CURRENT_TIMESTAMP
);
```

1.3 Alimentation via Talend

Cette partie nous a donné beaucoup de difficultés. Il fallait déjà dompter le logiciel. Entre les erreurs de synchronisation qui devaient nous faire restart le logiciel 1 à 3 fois, les lignes des erreurs non indiquées initialement, etc. Malgré tout cela, nous avons su faire preuve de patience et déboguer nos erreurs. En regardant le code java généré, nous avons pu voir où étaient les erreurs, qui étaient au final souvent des erreurs de typages dans nos schémas et TMap.

Au final pour alimenter les dimensions c'était plutôt simple, nous avons juste du adapter les différents délimiteurs de CSV ($'|'$,$','$,$','$,$';'$).

Nous avons bien posé nos méta-données dans le logiciel et utilisé le mode référentiel pour ne pas avoir à tout refaire à chaque lancement de nos jobs. Pour les connexions aux bases de données SQLite, nous avons opté pour un tDBOutput ou tDBInput, ainsi qu'un tDBCommit on component ok pour écrire sur nos bases que si tout se déroulait bien, et ainsi qu'un tDBConnex-

ion afin d'assurer la connexion.
Certains groupes n'ont apparemment pas utilisé cela, on n'a pas essayé de faire sans car nous ne trouvons cela pas très propre.

Nous avons aussi vu un groupe qui avait eu le malheur d'avoir le logiciel en anglais et qui avaient des valeurs négatives pour leur durée en raison du format.

Vu qu'on a constaté différents soucis sur les schémas SQL comme mentionné à la section précédente, nous avons opté pour un petit script bash qui nous faisait automatiquement les commandes sqlite3 pour créer le fichier datawarehouse.db et executer le script SQL dessus en 1 seconde.

La pierre angulaire de notre alimentation sur Talend était le composant TMap qui nous a aidés à faire tout ce que nous voulions faire.
Il nous a permis de faire des manipulations de jointure simple et efficace pour alimenter nos tables avec des données communes.

Nous avons pu constater que le temps d'exécution des jobs pouvait s'avérer très long si nous demandions à Talend de parcourir trop de dimensions sur chaque élément de dimension. Ainsi nos jobs étaient optimisés et assez efficaces.

Nous avons rusé et mis les dates au format Text et utilisé un utilitaire de Talend afin de calculer les durées pour nous simplifier la vie.(Voir les figures 3 et 4)

L'alimentation de la table des faits nous a demandé plus de travail que pour les autres jobs en raison de son originalité de traitement. Nous avons dû utiliser la base de données opérationnelle contenant les données pour les ventes, mais aussi les données des précédents jobs dans notre base de donnée datawarehouse générée.
Nous ne pouvions pas ici simplement nous resservir des CSV en raison des clés que nous avions uniquement dans la table générée.

```
TalendDate.diffDate(TalendDate.parseDate("yyyy-MM-dd
HH:mm", row1.heure_fin), TalendDate.parseDate("yyyy-MM-dd
HH:mm", row1.heure_debut), "mm")
```

Figure 3: Calcul des durées

Figure 4: TMap de ventes

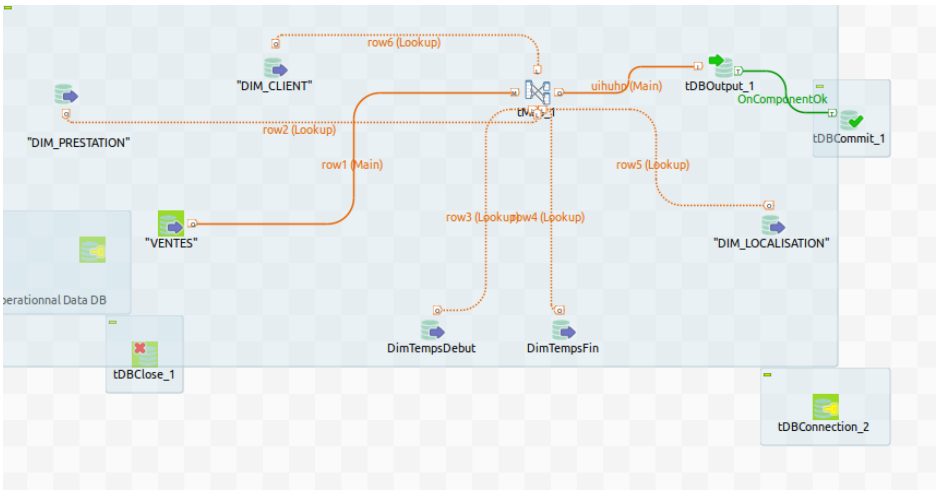

Figure 5: Job pour remplir la table des ventes

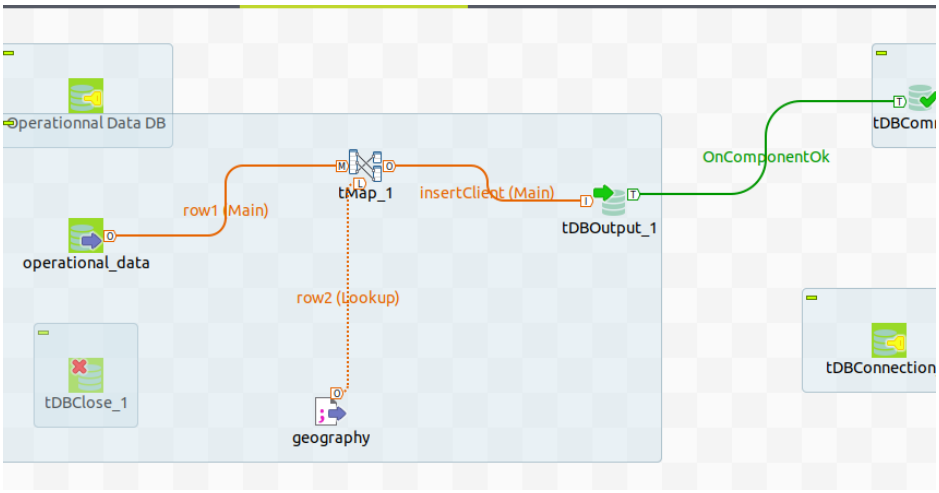

Figure 6: Job pour remplir la table des clients

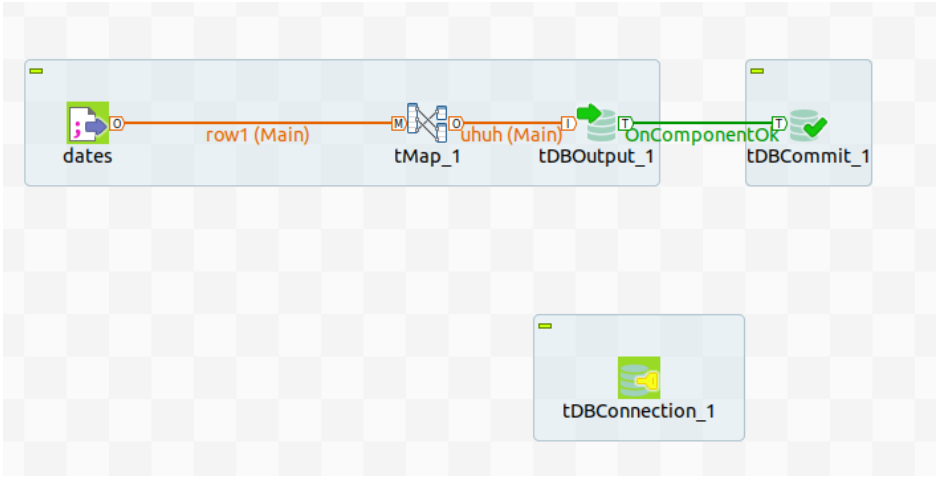

Figure 7: Job pour remplir les dates