

E C O L E D E S S C I E N C E S D E L ' I N F O R M A T I O N

**R É A L I S É P A R**

SOUMAYA EL AMRANI

N : 18

RAPPORT FINAL DU PROJET BUSINESS INTELLIGENCE

C O NC E P T IO N E T R É AL IS AT IO N

# Contexte fonctionnel :

Les plateformes de streaming de musique offrent un service basé sur le Web qui permet aux utilisateurs de diffuser des chansons en continu sur leur ordinateur ou leur appareil mobile. Ces services peuvent permettre aux utilisateurs de télécharger des chansons pour une lecture hors ligne ainsi que de télécharger leur propre collection de musique sur le Cloud. Des exemples bien connus de ces plateformes sont Spotify, Pandora, Napster, Google Play Music, YouTube Music, Apple Music, iTunes, iTunes Match. Ces plateformes stockent forcément les données des achats de leurs clients pour atteindre plusieurs objectifs qu’on détaillera par la suite. A ce stade, les bases de données opérationnelles alimentent les entrepôts des données de ces entreprises par des données détaillées concernant les achats des clients. Dans ce projet, on procédera au traitement de ces données selon la chaine Business Intelligence connue : ETL – Analyse – Reporting.

## Dataset utilisé

La base de données Chinook représente les données d’un service de streaming de musique. Elle comprend des tables pour artistes, albums, pistes médiatiques, factures et clients. Il est possible de voir le modèle de données Chinook ici <https://github.com/cwoodruff/ChinookDatabase>.

Des données liées aux médias ont été créées à l’aide de données réelles provenant d’une bibliothèque

iTunes.

## Outil utilisé

**SQL Server Integration Services** : SSIS est un composant du logiciel de base de données Microsoft SQL Server qui peut être utilisé pour effectuer un large éventail de tâches d'extraction, de transformation et de chargement de données

**Power BI** : Power BI est une solution de Business Intelligence développée par Microsoft pour permettre

aux entreprises d’agréger, d’analyser et de visualiser les données en provenance de sources multiples.

## Objectif métier

L’objectif est de mieux Comprendre le comportement des clients et avoir des insights sur les facteurs qui

entrainent plus de rentabilité pour la plateforme de streaming digital.

* Assurer le suivi informationnel des activités.
* Avoir un aperçu régulier sur la situation des achats.
* Mieux connaître leur clientèle (pouvoir d’achat, tendances et attentes…).
* Avoir une idée sur les produits (chansons, genre, albums etc) qui connaissent plus de succès.

# Descriptif détaillé des indicateurs choisis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Indicateurs choisis** | | | **Explication** | **Justification** |
| Chansons plus achetées par  genre | | | Compter les quantités consommées des  chansons par genre. | Déterminer Les genres les  plus écoutés. |
| Chansons plus achetées par  pays | | | Compter les quantités consommées des  chansons par pays. | Déterminer Les genres les  plus écoutés par pays. |
| Total d’achats par genre et par pays | | | Calculer le total payé pour chaque genre. | Déterminer Les genres les plus rentables. |
| Nombre d’achats par genre et  par pays | | | Compter les quantités consommées des  chansons par pays et par genre. | Déterminer Les genres les  plus rentables par pays. |
| Durée moyenne des chansons achetés | | | Compter la moyenne des durées des chansons écoutés par les utilisateurs. | Avoir une idée sur la  distribution des durées des chansons achetées. |
| TOP 5 | Chanson | Du mois | Chanson/Album/Artiste/genre plus achetés par mois/semaine/année. | Savoir Le top 5 des chansons, des albums, des artistes et des albums qui chiffrent le plus sur différentes périodes de temps et cela à intervalles réguliers. |
| Album |
| Artiste |
| Genre |
| Chanson | De la semai ne |
| Album |
| Artiste |
| Genre |
| Chanson | De  l’anné  e |
| Album |
| Artiste |
| Genre |
| Artiste Plus rentable dans  chaque genre | | | L’artiste avec la contribution majeur à la  plateforme. | Savoir Les artistes les plus  achetés pour chaque genre. |
| Chiffre d’affaire par année | | | La somme des recettes générées par la plateforme. | Avoir une idée précise sur la valeur exacte du chiffre  d’affaires de la plateforme. |
| Evolution temporelle des  revenus de chaque genre | | | Valeur du revenu généré par chaque  genre par jour ou semaine. | Pouvoir comparer le revenu  généré par différents genres. |

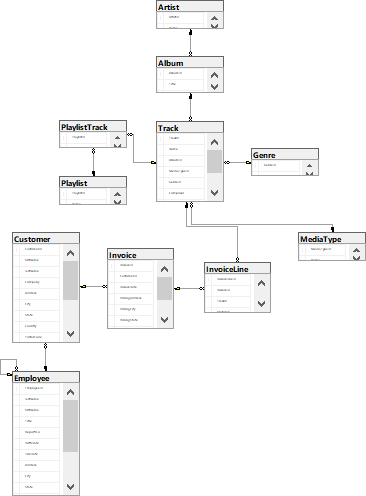
1. Matrice dimensionnelle d'analyse des indicateurs

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Indicateurs choisis** | | | **Track** | **Genre** | **Album** | **Custom**  **er** | **Invoice** | **Artist** |
| Chansons plus achetées par genre | | | x | x |  |  | x |  |
| Chansons plus achetées par pays | | | x |  |  |  | x |  |
| Total d’achats par genre et par pays | | | x | x |  |  | x |  |
| Nombre d’achats par genre et par  pays | | |  |  |  |  | x |  |
| Durée moyenne des chansons  achetés | | | x |  |  |  |  |  |
| TOP 5 | Chanson | Du mois | x |  |  |  | x |  |
| Album |  |  | x |  | x |  |
| Artiste |  |  |  |  | x | x |
| Genre |  | x |  |  | x |  |
| Chanson | De la  semaine | x |  |  |  | x |  |
| Album |  |  | x |  | x |  |
| Artiste |  |  |  |  | x | x |
| Genre |  | x |  |  | x |  |
| Chanson | De  l’année | x |  |  |  | x |  |
| Album |  |  | x |  | x |  |
| Artiste |  |  |  |  | x | x |
| Genre |  | x |  |  | x |  |
| Artiste Plus rentable dans chaque  genre | | | x | x |  |  | x |  |
| Chiffre d’affaire par année | | |  |  |  |  | x |  |
| Evolution temporelle des revenus de  chaque genre | | |  | x |  |  | x |  |

# Conception du datawarehouse

## Conception initiale

Comme déjà précisé, la base de données initiales sur laquelle je me suis basé est en format .bak. Ainsi, son schéma initial est le suivant :



La table de fait est InvoiceLine, elle contient les achats granulaires des chansons. Les dimensions initiales sont les suivantes :

La table des employés stocke les données relatives aux employés telles que l'identité de l'employé, son nom, son prénom, etc. Elle comporte également un champ appelé ReportsTo pour spécifier qui fait rapport à qui.

**Customers** : stocke les données relatives aux clients.

**Invoices** : La table invoices stocke les données d'en-tête de la facture.

**Artists** : stocke les données relatives aux artistes. Il s'agit d'une simple table qui ne contient que l'identifiant et le nom de l'artiste.

**Albums** : stocke les données relatives à une liste de titres. Chaque album appartient à un artiste. Toutefois, un artiste peut avoir plusieurs albums.

**Media\_types** : stocke les types de médias tels que les fichiers audio MPEG et AAC.

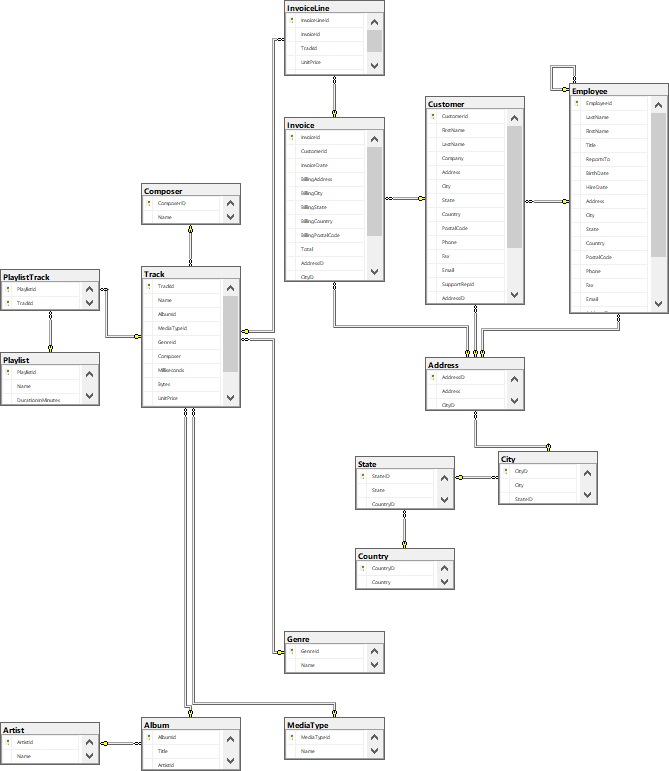
**Genre** : stocke les types de musique tels que le rock, le jazz, le métal, etc.

**Track** : stocke les données des chansons. Chaque piste appartient à un album.

**Playlist & playlist\_track** : la table playlists stocke les données des playlists. Chaque playlist contient une liste de pistes. Chaque piste peut appartenir à plusieurs listes de lecture. La relation entre la table des playlists et la table des pistes est multiple. La table playlist\_track est utilisée pour refléter cette relation.

## Conception normalisée

La nouvelle conception proposée est une conception qui normalise les colonnes Composer et les colonnes d’addresses (Address, City, State, Country) à des dimensions à part et assure le lien entre la table de faits et ses dimensions par des clés étrangères. Le but de cette normalisation est d’éviter la redondance des données en suivant les trois formes de normalisations standards. Dans notre cas, les 2 premières formes sont assurées, donc notre but est d’atteindre la 3ème forme normale. Le résultat de ce processus de reconception, dont les étapes de réalisation seront détaillées ultérieurement dans le présent rapport, est le suivant :



# Partie ETL :

Avant de commencer le pipeline ETL, un scan général a été réalisé sur SQL Server Management Studio (SSMS) pour vérifier :

* La présence des données manquantes.
* La pertinence des types de données.
* La cohérence des colonnes.

La commande T-SQL utilisée est : **sp\_help** nom\_de\_la\_table.

Le format .bak de la base de données initiale a facilité l’intégration de cette base de données dans le nouveau Data Warehouse à travers l’outil de l’ETL SQL Server Integration Services.

Le processus ETL suivi sert à réaliser par ordre les taches principales suivantes :

* La création des tables du nouveau Data Warehouse.
* L’ajout des colonnes des ID secondaires dans les Tables Invoice, Track, Customer et Employee qui

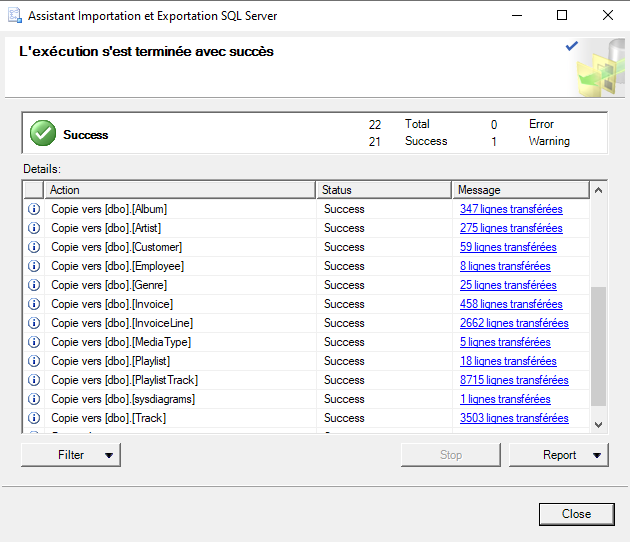
serviront de relier ces derniers par les nouvelles dimensions créées à la suite de la normalisation.

* La population des dimensions résultantes de la normalisation.
* La mise à jour des tables par les clés secondaires qui renvoient vers les dimensions nouvellement crées.
* La création de la colonne calculée « Minutes » dans la table Track pour avoir la durée en minutes de chaque Chansons vu qu’on a déjà cette valeur mais en millisecondes.

## Création du Data Warehouse :

Le nouveau Datawarehouse nommée DWChinook est une version normalisée de la base de données initiale nommée Chinook. Donc, pour le créer, j’ai commencé par répliquer la base de données initiale remplie des données et ensuite j’ai veillé à créer les nouvelles dimensions dont on a besoin.

Etape de réplication :



Etape de création des dimensions et la liaison avec leurs tables d’origines par des clés étrangères (étape

réalisée dans SSMS) :

Je liste ci-dessous les scripts SQL utilisés pour créer les nouvelles dimensions (Composer, Address, City, State, Country) tout en initialisant les clés étrangères nécessaires. Je tiens à préciser que j’ai respecté le typage uniforme des données de tel façons que les colonnes de clés primaires soient conformes avec les clés étrangères auxquelles elles correspondent. Il est nécessaire de préciser que les tables suivantes ont des clés primaires auto-incrémentalles.

CREATE TABLE Composer ( ComposerID INTEGER NOT NULL,

Composer nchar(40), PRIMARY KEY (ComposerID),

)

CREATE TABLE Country ( CountryID INTEGER NOT NULL,

Country nchar(40), CountryID INTEGER, PRIMARY KEY (StateID),

FOREIGN KEY (CountryID) REFERENCES Country(CountryID)

)

CREATE TABLE State ( StateID INTEGER NOT NULL,

State nchar(40), CountryID INTEGER, PRIMARY KEY (StateID),

FOREIGN KEY (CountryID) REFERENCES Country(CountryID)

)

CREATE TABLE City ( CityID INTEGER NOT NULL,

City nchar(40), StateID INTEGER, PRIMARY KEY (CityID),

FOREIGN KEY (StateID) REFERENCES State(StateID)

)

CREATE TABLE Address ( AddressID INTEGER NOT NULL,

Address nchar(70), CityID INTEGER,

PRIMARY KEY (AddressID),

FOREIGN KEY (CityID) REFERENCES City(CityID)

)

Les tables des dimensions sont maintenant créées, on passe maintenant à l’étape de Création des clés secondaires et laLiaison des customers, Employee et Invoices avec l’address qui est à son tour développée dans les dimensions city, state, country.

ALTER TABLE Invoice

ADD AddressId INTEGER,

FOREIGN KEY(AddressId) REFERENCES Address(AddressId); ALTER TABLE Customer

ADD AddressId INTEGER,

FOREIGN KEY(AddressId) REFERENCES Address(AddressId); ALTER TABLE Employee

ADD AddressId INTEGER,

FOREIGN KEY(AddressId) REFERENCES Address(AddressId);

ALTER TABLE Employee

ADD COLUMN CountryID INTEGER, ADD COLUMN CityID INTEGER, ADD COLUMN StateID INTEGER;

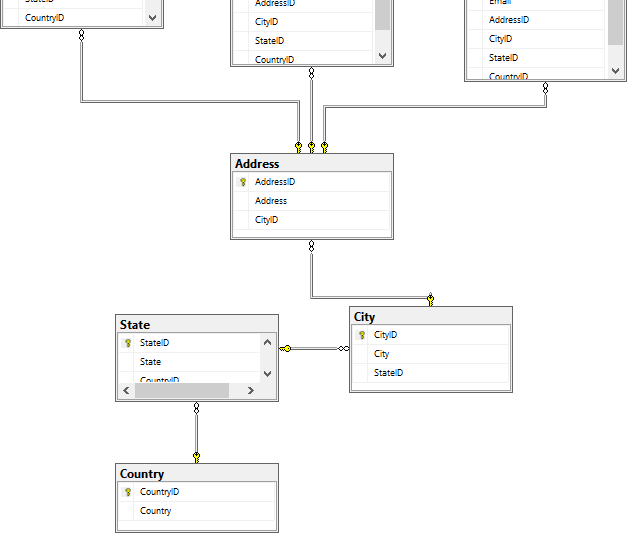
ALTER TABLE Customer

ADD COLUMN CountryID INTEGER, ADD COLUMN CityID INTEGER, ADD COLUMN StateID INTEGER;

ALTER TABLE Invoice

ADD COLUMN CountryID INTEGER, ADD COLUMN CityID INTEGER, ADD COLUMN StateID INTEGER;

Le Schéma Résultant est le suivant :

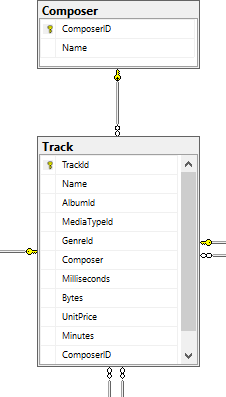


De même pour la table Track avec la clé secondaire ComposerID

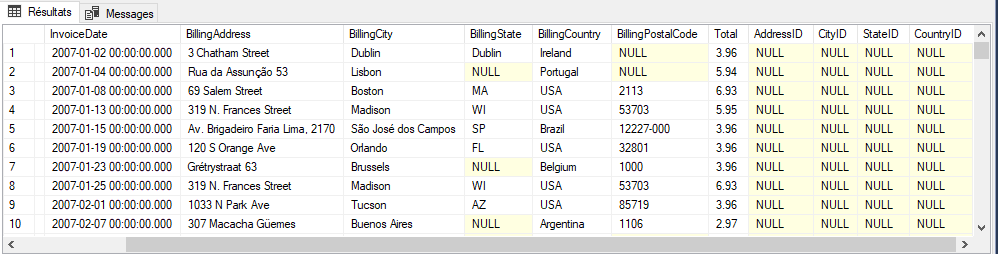
ALTER TABLE Track

ADD ComposerID INTEGER; FOREIGN

KEY(ComposerId) REFERENCES Composer(ComposerId);



Voici un aperçu des clés secondaires résultantes (la table Invoice comme exemple) :

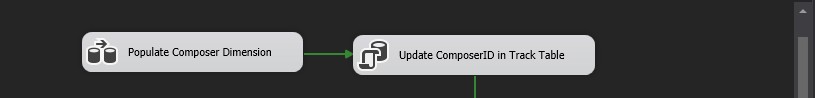


## Pipeline ETL

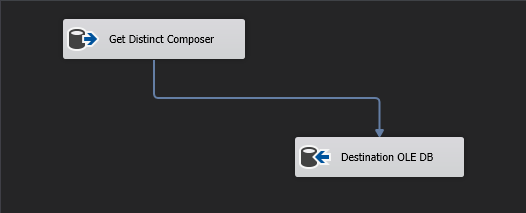
On passe à présent à l’outil SSIS pour remplir ces dimensions.

L’idée est de sélectionner les valeurs uniques de chaque colonne qu’on souhaite normaliser est les mettre dans les dimensions qui lui sont réservées. Ensuite on passe à la liaison des tables d’origine avec les dimensions qui viennent d’être remplie.

On fournit l’exemple de cette opération pour la dimension Composer :



La tâche de flux de données Populate Composer Dimension

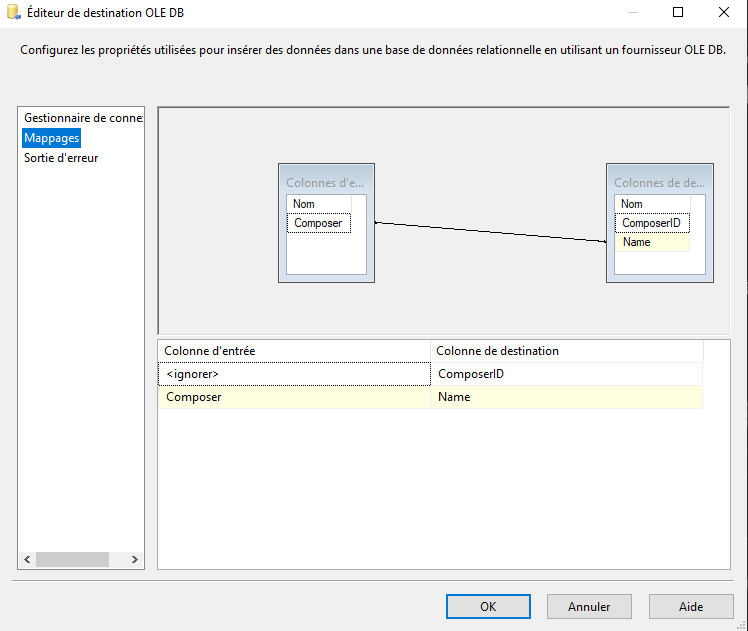


Get Distinct Composer :

SELECT DISTINCT Composer FROM Track

ORDER BY Composer ASC

La destination OLE DB remplit la dimension Composer par le résultat de la requête précédente. Je tiens à préciser que l’ID de la table Composer s’autoincrémente.



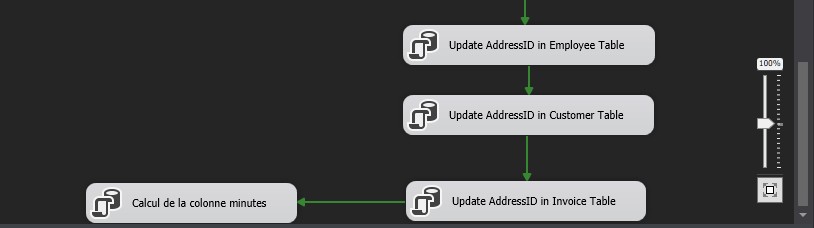
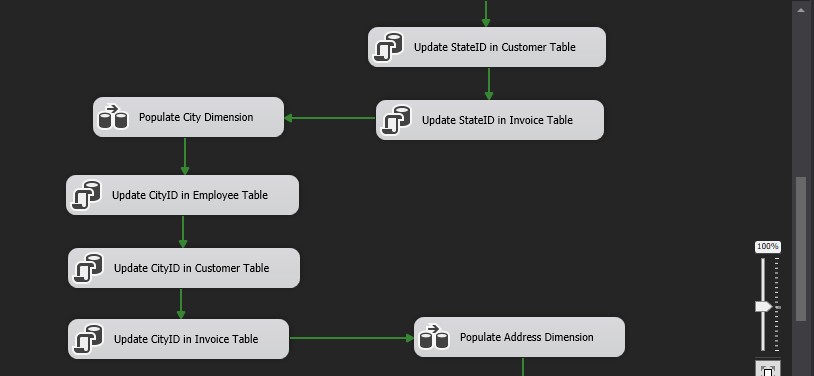
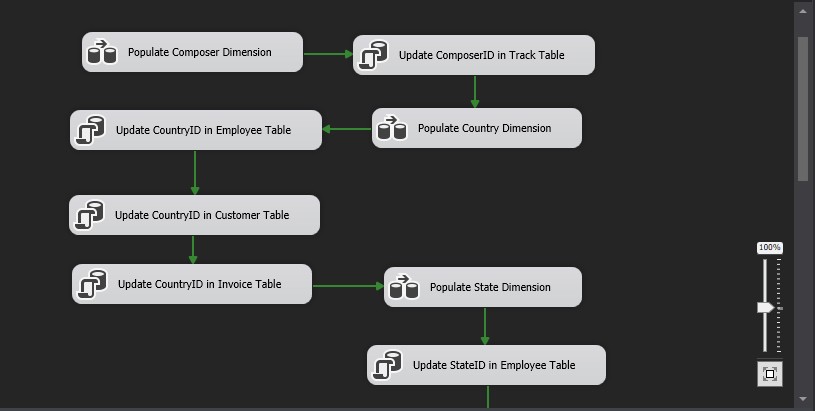
Update ComposerID in Track Table : Cette Tâche SQL consiste à mettre à jour la clé etrangère ComposerID de la table Track par les ID correspondants au nom du compositeur. Le code SQL de cette tâche est le suivant :

UPDATE Track

SET Track.composerID = Composer.ComposerID FROM Track

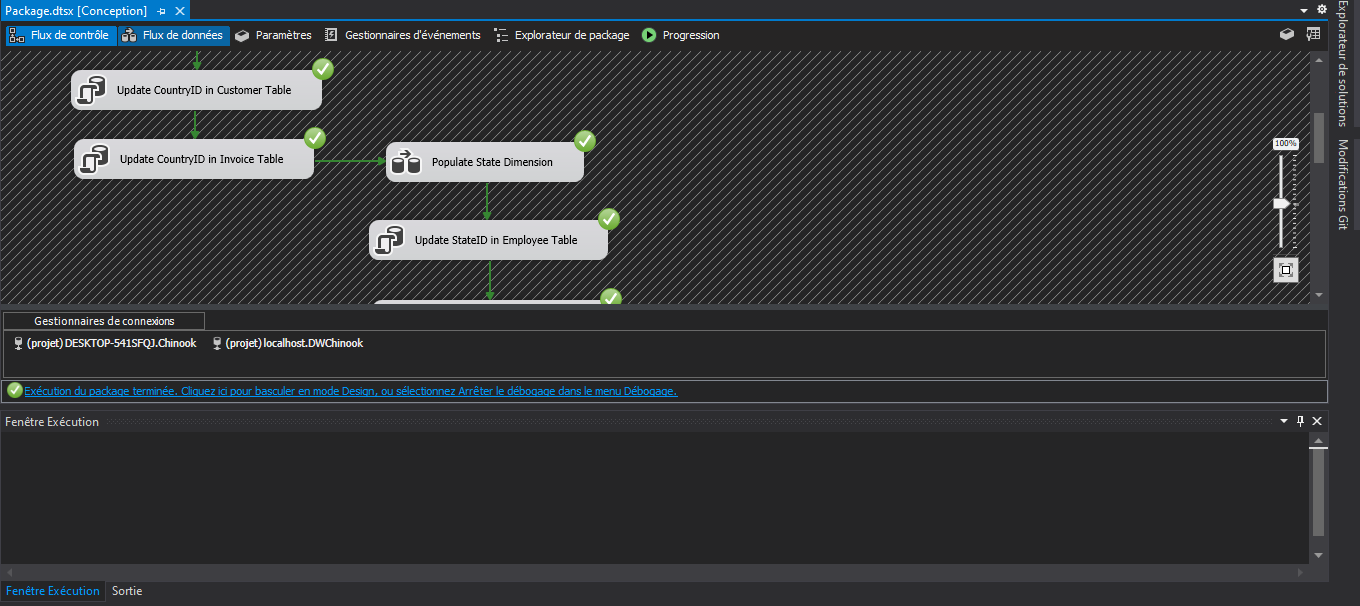
INNER JOIN Composer ON Track.Composer = Composer.Name

Concernant les dimensions Country, State, City et Address, le processus change un peu vu qu’il y a plusieurs dimensions interreliées à remplir. On explique, La dimension Address contient l’id de la ville de cette adresse, du coup, on ne peut remplir correctement la dimension address qu’après avoir rempli la dimension ville qui à son tour nécessite que la table State doit être remplie vu que la table City comporte l’ID du State de cette ville et ainsi de suite. Donc, la première dimension à remplir est celle des Pays, la dimension Country. On présente alors dans les captures suivantes le processus ETL correspondants à ces étapes consécutives. Qui consiste à remplir la dimension concernée et mettre à jour dans les Tables Cutomers, Employees et Invoice les clés secondaires qui correspondent aux clés primaires des dimensions qui leurs correspondent. Cette opération est répétée jusqu’à ce que toutes les dimensions et toutes les clés primaires sont bien remplies.



En annexe, vous trouverez l’ensemble des requêtes SQL utilisées dans ce pipeline.

Après l’exécution avec succès de l’ETL

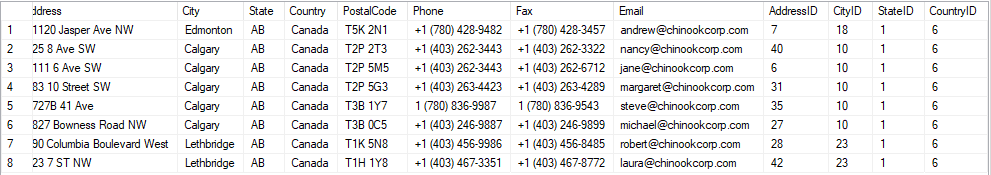


Tables Customer, Invoice et Employee après la mise à jour des clès secondaires dans l’ETL :

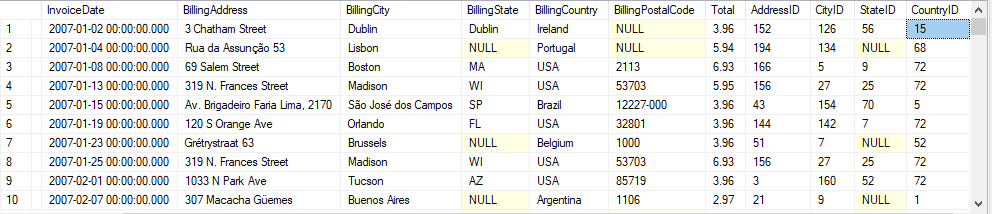
**Customer**



**Employee**



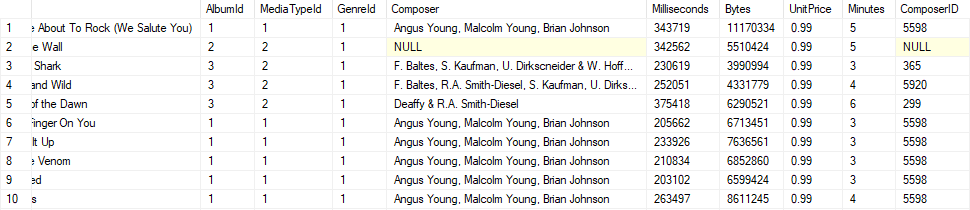
**Invoice**



La dernière tache de ce pipeline consiste à remplir la colonne « Minutes » de la table Track par la colonne de la durée convertie des millisecondes en minutes, c’est une requête SQL qui s’annoce comme suit :

UPDATE Track

SET Track.Minutes = Track.Milliseconds / (1000\*60) Le résultat est le suivant dans la table Track

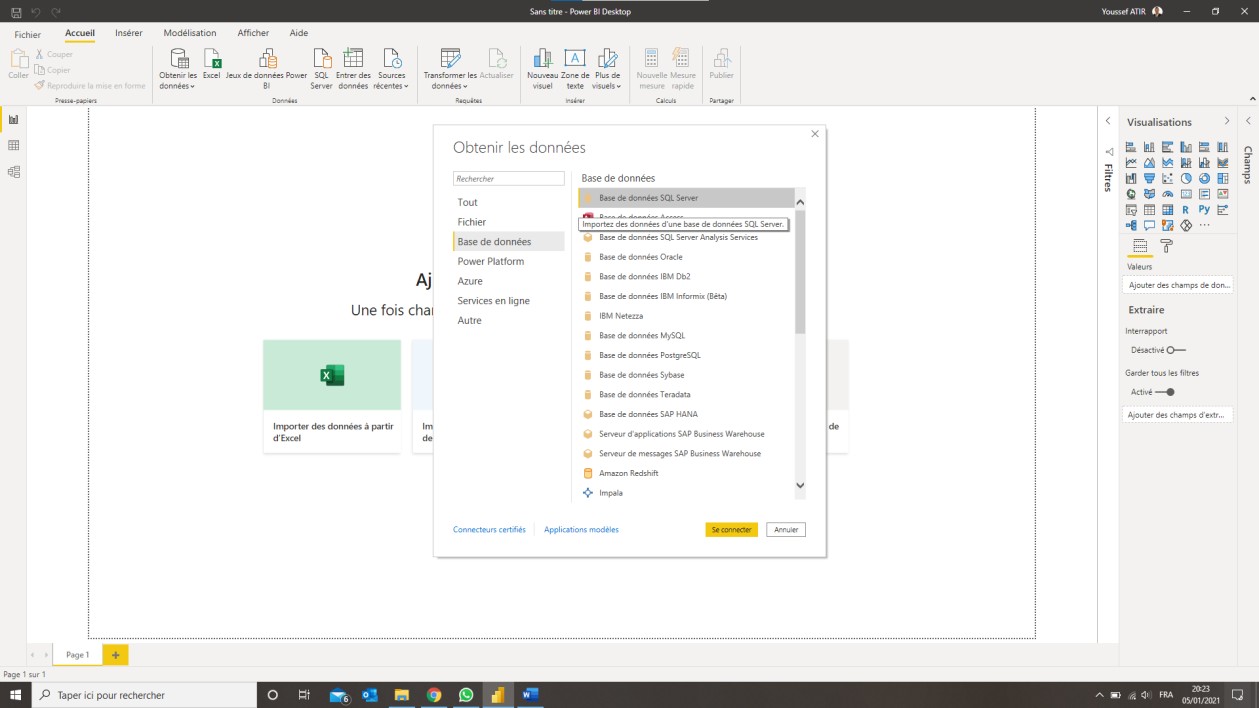


# Partie Analyse, Visualisation et Reporting des KPI

La partie analyse multidimensionnelle a été réalisé via les matrices et les tables Power BI. Quant à la partie Visualisation et Reporting, elle a été réalisée par les différentes visualisations qu’offre la même solution.

On commence d’abord par alimenter Power BI par le Data Warehouse issu de l’ETL :

**Choix de la connexion à une base de données SQL Server**



**Chargement des données :**