Reporte de Laboratorio Nro.2

Jhandry Zambrano[L00380367]

Universidad de las Fuerzas Armadas
jvzambrano3@espe.edu.ec

Tema: Modelamiento multidimensional

Resumen

En el presente Laboratorio se va a realizar el proceso de modelación de una estructura multidimensional para un caso de la vida cotidiana real. Se realiza esto con el objetivo de aprender a abstraer conocimientos acerca del modelado multidimensional. Además, poder solventar problemas de analítica de datos relacionados en un entorno de forma real. Por ende, primer se realiza la creación de dimensiones y carga de datos, también se realiza la creación de otras dimensiones necesarias teniendo en cuenta aspectos como el tiempo. Por ende, la creación de las dimensiones debe ser limpias es decir no tener valores erróneos o en blanco. Para ello se va a realizar un ETL que permita la extracción, transformación y carda de datos en algún Datawarehouse. Para ello se debe generar todo manualmente por comandos y código. Finalmente conocer la importancia de todas estas implementaciones en la vida cotidiana.

1. Introducción

En el laboratorio se realiza primeramente la creación de las dimensiones establecidas. Esto se realiza con el objetivo de aprender a crear estructuras multidimensionales para casos en la vida real. Para ello es necesario tener conocimientos en base de datos en comandos, consultas y realización de un ETL. Con ello se podrá aprender a solucionar los problemas mediante un análisis previo para la toma de decisiones y las necesidades que requiera cualquier empresa. Además de analizar algunas de las dimensiones con sus características necesarias como la dimensión tiempo que debe estar correctamente definida al igual que las demás dimensiones.

Para el modelamiento se va hacer uso del esquema estrella debido a que es uno de los esquemas que permite relaciones directas hacia la tabla de hechos que es la tabla principal. Para seguir con su desarrollo debemos tener en cuenta varios aspectos como la normalización y desnormalización de la misma siempre teniendo en cuenta las necesidades de la empresa. Los modelos dimensionales nos permiten optimizar el tiempo para las consultas en caso de tener una base de datos grande como muchas de las empresas. Como ingenieros debemos tener la capacidad de analizar la situación y tomar la decisión de definir que opción es más beneficiosa .

El crear dimensiones para un datawarehose y ETL para la manipulación de sus datos. Nos permite aprender del mismo obteniendo datos sólidos. Para ello se realiza en una base de datos como SQLITE y Python. Por ende, aprenderemos la importancia del almacén de los datos y su

diseño. En el transcurso del tiempo se puede ver la eficiencia de este procedimiento que en la empresa el tiempo demanda ganancias por ende mientras más fácil sea de manipular u obtener datos de una base de datos más optima será su extracción. Una de las ventajas principales de realizar un esquema con la tabla de hechos puede ser su simplicidad.

2. Método

Para empezar con la práctica de laboratorio primero abrimos nuestro sistema de bases de datos. Para ello se hace uso de SQLite donde procedemos a crear nuestro datawarehouse. Todos y cada uno de los procesos serán establecidos mediante el uso de de comandos.Primeramente creamos nuestra Datawarehouse donde mediante comandos como se muestra en la figura 1

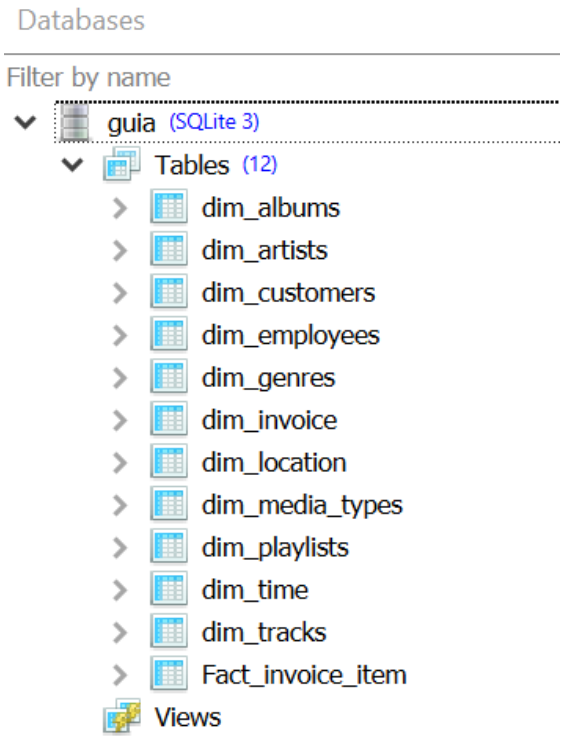

Figura 1: Creación de modelo dimensional

Para crear cada una de las dimensiones se hizo uso de los siguientes comandos. Para ello se establece las dimensiones y otras tablas que no se incluyen en chinook como parte del desarrollo mismo. En la siguiente imagen se crea los script de creación de la dimension album,artist y customers definiendo y estableciendo cada llave primaria ademas del tipo de datos. A continuación en como se muestra en figura 2.

```sql
--Creación de la dimension albums
CREATE TABLE dim_albums (
    AlbumId INTEGER        PRIMARY KEY AUTOINCREMENT
                           NOT NULL,
    Title   NVARCHAR (160) NOT NULL
);

--Creación de la dimension artist
CREATE TABLE dim_artists (
    ArtistId INTEGER        PRIMARY KEY AUTOINCREMENT
                            NOT NULL,
    Name     NVARCHAR (120)
);


--Creación de la dimension customers
CREATE TABLE dim_customers (
    CustomerId   INTEGER        PRIMARY KEY AUTOINCREMENT
                                NOT NULL,
    FirstName    NVARCHAR (40) NOT NULL,
    LastName     NVARCHAR (20) NOT NULL,
    Company      NVARCHAR (80),
    Address      NVARCHAR (70),
    City         NVARCHAR (40),
    State        NVARCHAR (40),
    Country      NVARCHAR (40),
    PostalCode   NVARCHAR (10),
    Phone        NVARCHAR (24),
    Fax          NVARCHAR (24),
    Email        NVARCHAR (60) NOT NULL,
    SupportRepId INTEGER        NOT NULL
                                REFERENCES dim_employees (EmployeeId)
);
```

Figura 2: Creación de tres dimensiones con sus caracteristicas.

Para crear las dimensiones se hizo uso de los siguientes comandos. Para ello se establece las dimensiones y otras tablas que no se incluyen en chinook como parte del desarrollo mismo. En la siguiente imagen se crea los script de creación de la employee,genre e invoice definiendo y estableciendo cada llave primaria ademas del tipo de datos. A continuación en como se muestra en figura 3.

```sql
--Creación de la dimension employees
CREATE TABLE dim_employees (
    EmployeeId INTEGER       PRIMARY KEY AUTOINCREMENT
                             NOT NULL,
    LastName   NVARCHAR (20) NOT NULL,
    FirstName  NVARCHAR (20) NOT NULL,
    Title      NVARCHAR (30),
    ReportsTo  INTEGER       REFERENCES dim_employees (EmployeeId),
    BirthDate  DATETIME,
    HireDate   DATETIME,
    Address    NVARCHAR (70),
    City       NVARCHAR (40),
    State      NVARCHAR (40),
    Country    NVARCHAR (40),
    PostalCode NVARCHAR (10),
    Phone      NVARCHAR (24),
    Fax        NVARCHAR (24),
    Email      NVARCHAR (60)
);

--Creación de la dimension genres
CREATE TABLE dim_genres (
    GenreId INTEGER        PRIMARY KEY AUTOINCREMENT
                           NOT NULL,
    Name    NVARCHAR (120)
);

--Creación de la dimension invoice
CREATE TABLE dim_invoice (
    InvoiceID INTEGER         PRIMARY KEY AUTOINCREMENT,
    Total     NUMERIC (10, 2) NOT NULL
);
```

Figura 3: Creación de tres dimensiones employee, genre e invoice

De esta forma de crea las otras tablas siempre y cuando teniendo en cuenta la guia. Además

aqui se crea la tabla de hechos la cual es fundamental para el modelamiento multidimensional de nuestra base de datos con esquema estrella. Ademas debemos ser minuciósos analizadando la relación de cad auna de las tablas.Entonces se debe tener en cuenta que todo debe ser realizado de forma manual como se muestra acontinuación en la figura 4.

```
--Creación de la dimension tracks
CREATE TABLE dim_tracks (
    TrackId      INTEGER         PRIMARY KEY AUTOINCREMENT
                                 NOT NULL,
    Name         NVARCHAR (200)  NOT NULL,
    MediaTypeId  INTEGER         NOT NULL
                                 REFERENCES dim_media_types (MediaTypeId),
    GenreId      INTEGER         REFERENCES dim_genres (GenreId),
    Composer     NVARCHAR (220),
    Milliseconds INTEGER         NOT NULL,
    Bytes        INTEGER,
    UnitPrice    NUMERIC (10, 2) NOT NULL
);

--Creación de la tabla de hechos fact invoice item
CREATE TABLE Fact_invoice_item (
    Id         INTEGER PRIMARY KEY AUTOINCREMENT
                       NOT NULL,
    InvoiceID  INTEGER REFERENCES dim_invoice (InvoiceID)
                       NOT NULL,
    CustomerID INTEGER NOT NULL
                       REFERENCES dim_customers (CustomerId),
    TimeID     INT     REFERENCES dim_time (TimeID)
                       NOT NULL,
    LocationID INTEGER REFERENCES dim_location (LocationID)
                       NOT NULL,
    PlaylistID INTEGER REFERENCES dim_playlists (PlaylistId)
                       NOT NULL,
    ArtistID   INTEGER REFERENCES dim_artists (ArtistId)
                       NOT NULL,
    AlbumID    INTEGER REFERENCES dim_albums (AlbumId)
                       NOT NULL,
    Quantity   INTEGER NOT NULL
);
```

Figura 4: Creación de tres dimensiones employee, genre e invoice

Luego de haber realizado todos estos procesos se procede a la parte donde se efectua un ETL. El ETL va a permitir cargar los datos que tenemos en nuestra base de datos chinook hacia la datawarehaouse y ademas tener otras tablas necesarias. Para ello primeramente realizamos la importación de cada una de las librerias necesarias. A continuación se muestra las librerias que se usaran como se muestra en la figura 5.

```
#librerias
import sqlalchemy
from sqlalchemy import create_engine
from sqlalchemy import Table, Column, Integer, String, MetaData, ForeignKey
from sqlalchemy import inspect
```

Figura 5: Importación de librerias

En consecuencia se realiza la conexión de las bases de datos. Esto implica la base de datos chinook y la nueva que se denomia guia segun se establecia el modelo de laboratorio.A continuación se muestra el en la figura 6.

```
#Conectar el motor al archivo de la base de datos
engine = create_engine('sqlite:///chinook.db')
engineDWH = create_engine('sqlite:///guia.db')
```

Figura 6: Conexión a la base de datos.

Luego se procede a crear los metadatos donde obtenenos la tabla cliente. Por ende, se requiere de metodos que instancien la tabla, esta se encuentra conectado al motor. Despues realizamos

un breve inspección para verificar que todo este funcionando correctamente. A continuación se muestra la programación usada en la figura 7.

```
# SQL permite crear metadatos que contienen objetos que definen la tabla de clientes
metadata = MetaData()

# Este método instancia las tablas que ya
# existe en la base de datos, a la que está conectado el motor.
metadata.create_all(engine)
metadata.create_all(engineDWH)
# Revisando esto, podemos ver la estructura de la tabla y los tipos de variables
inspector = inspect(engine)
inspectorWH = inspect(engineDWH)

#Tablas de db tranlacinal chinook - invoice_items
#print('Chinook-invoice_items')
#print(inspector.get_columns('invoice_items'))
#Tablas de echo - fact_sales
#print('\n WH - Fact_sales')
#print(inspectorWH.get_columns('Fact_sales'))
```

Figura 7: Proceso extracción de datos.

Se realiza el proceso de inspección el cual nos permitira verificar si los datos han sido extraidos con exitos. Este proceso es realizado con cada una de las tablas. A continuación se muestra en proceso en la figura 8.

```
#datos de db
inspector.get_columns('employees')
inspector.get_columns('albums')
inspector.get_columns('media_types')
inspector.get_columns('genres')
inspector.get_columns('playlists')
inspector.get_columns('artists')
inspector.get_columns('invoices')
inspector.get_columns('customers')
inspector.get_columns('tracks')
```

Figura 8: Inspección de datos.

En consiguiente se realiza el proceso de transformación de los datos. Es el segundo paso del ETL donde los datos son transformados para ello hacemos la importación de la libreria pandas. Estableciendo un dataframe de cada tabla. A continuación se meuestra en la figura 9.

Transform

```
[ ]  import pandas as pd
     # dim_employees
     df_employees=pd.read_sql_query("""SELECT LastName, FirstName, Title, ReportsTo, BirthDate, HireDate, Address,
     City, State, Country, PostalCode, Phone, Fax, Email
     FROM employees;
      """, con=engine.connect())
     df_employees.head()
```

Figura 9: Proceso de transformación

Finalmente el proceso ETL termina con la carga donde los datos de la base de datos. Al realizar esta acción practicamente ya tenemos los datos almacenados en el datawarehouse sin embargo se debe verificar que todo funcione correctamente. Esto se realiza mediante el uso de condicionales como se muestra en el algoritmo de la siguiente figuta 10.

Funciones de carga

```
[ ]  def updateData(name_table,data_db,name_columns):
       print(name_columns)
       aux_values='?'
       aux_data=[]
       aux_columns=name_columns[0]
       aux_data.append(list(data_db[name_columns[0]]))
       entities=[]
       aux_entities=[]
       for i in range(len(name_columns)-1):
         aux_values+=',?'
         aux_columns+=','+name_columns[i+1]
         aux_data.append(list(data_db[name_columns[i+1]]))
       for i in range(len(aux_data[0])):
         #aux_entities.append(i)
         for j in range(len(aux_data)):
           aux_entities.append(aux_data[j][i])
         entities.append(aux_entities)
         aux_entities=[]
       with engineDWH.connect() as con:
         for i in entities:
           #print('INSERT INTO '+name_table+'('+aux_columns+') VALUES('+aux_values+')')
           #print(i)
           con.execute('INSERT INTO '+name_table+'('+aux_columns+') VALUES('+aux_values+')', i)
       con.close()
```

Figura 10: Proceso de carga

Ahora realizamos el proceso de actualización de datos. Para realizar esto hacemos uso de codigo de python que nos permite actualizar los datos de cada una de las tablas. A continuación se muestra en la figura 11.

dim_employees

```
[ ] updateData('dim_employees',df_employees,list(df_employees.columns.values))
    ['LastName', 'FirstName', 'Title', 'ReportsTo', 'BirthDate', 'HireDate', 'Address', 'City', 'State', 'Country', 'PostalCode', 'Phone', 'Fax', 'Email']
```

dim_albums

```
[ ] updateData('dim_albums',df_albums,list(df_albums.columns.values))
    ['AlbumId', 'Title']
```

dim_media_types

```
[ ] updateData('dim_media_types',df_mediatypes,list(df_mediatypes.columns.values))
    ['MediaTypeId', 'Name']
```

dim_genres

```
    updateData('dim_genres',df_genres,list(df_genres.columns.values))
    ['GenreId', 'Name']
```

Figura 11: UpdateData

Se establece algunas de las consultas de algunas tablas. Para ello es necesario tener en cuenta que es indispensable saber programación en python debido a que se hace uso de for. A continuación se muestra las consultas en la figura 12.

Realiza la consulta donde posteriormente carga a la tabla de hechos. Se muestra en la ejecución donde abstrae los datos y los agrega donde muestra las id de cada una de las tablas. A continuación se muestra en la figura 13.

Fact_invoice_item

```
[ ]
    #fac_invoce_items
    #Columnas Aux
    df_AuxInvoiceId=pd.read_sql_query("""SELECT InvoiceId
    FROM invoice_items;
     """, con=engine.connect())
    df_AuxInvoiceId.head()

    df_CustInv=pd.read_sql_query("""SELECT InvoiceId, CustomerId
    FROM invoices;
     """, con=engine.connect())

    aux=list(df_AuxInvoiceId['InvoiceId'])
    auxCustomer=[]
    for i in aux:
        aux2=df_CustInv.index[df_CustInv['InvoiceId'] == i].tolist()
        auxCustomer.append(df_CustInv['CustomerId'][aux2[0]])

    df_timeInv=pd.read_sql_query("""SELECT InvoiceId, InvoiceDate
    FROM invoices;
     """, con=engine.connect())
    aux=list(df_AuxInvoiceId['InvoiceId'])
    auxTime1=[]
    for i in aux:
        aux22=df_CustInv.index[df_CustInv['InvoiceId'] == i].tolist()
        auxTime1.append(df_timeInv['InvoiceDate'][aux22[0]])
    df_dim_time=pd.read_sql_query("""SELECT TimeID, InvoiceDate
    FROM dim_time;
     """, con=engineDWH.connect())
    auxTime=[]
```

Figura 12: UpdateData

Fact_invoice_items

```
[ ]  columnas=['InvoiceID','CustomerID','TimeID','LocationID','TrackID','PlaylistID','ArtistID','AlbumID','Quantity']
     df = pd.DataFrame(list(zip(list(df_AuxInvoiceId['InvoiceId']),auxCustomer,auxTime,auxLocation,
                                list(df_AuxTrackId['TrackId']),auxPlaylist,auxArtist,auxAlbum,
                                 list(df_Quantity['Quantity']))), columns = columnas)
     updateData('Fact_invoice_item',df,list(df.columns.values))

     ['InvoiceID', 'CustomerID', 'TimeID', 'LocationID', 'TrackID', 'PlaylistID', 'ArtistID', 'AlbumID', 'Quantity']
```

Figura 13: UpdateData

3. Results and Analysis

Al desarrollar toda la practica podemos se obtuvo el resultado final. Aqui se obtiene la tabla de hechos y las dimensiones creadas para que en consecuencia puedan ser cargadas mediante el proceso ETL.

4. Discusión

En el desarrollo de este laboratorio se implemento las dimensiones sdc esto se produce en CostumerID y TranckID. Por lo tanto, para ello se unen a varias de las dimensiones de tipo 1 como las dimensiones de localización y playlist. Para realización del modelo dimensional hacemos uso

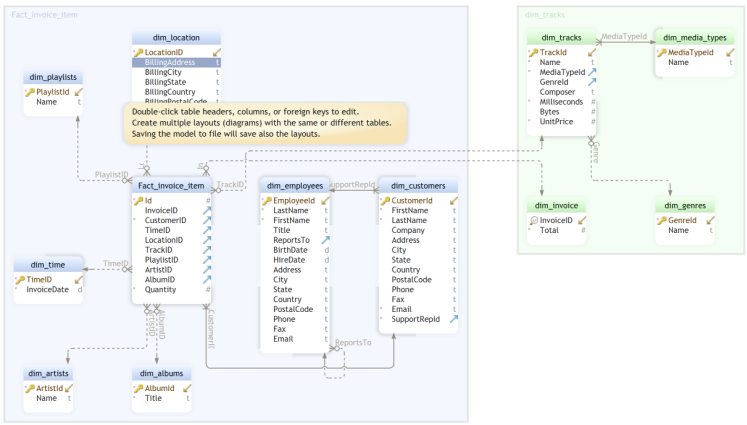

Figura 14: Modelamiento multidimensional de una base de datos con el esquema estrella.

del modelo estrella donde la tabla de hechos se denomina tabla factvoiceitem. Esta será la que nos permite realizar el proceso de transacción.

El almacén de datos en si es muy importarte para las organizaciones porque permiten tener los datos necesarios. Esto implica que los datos sean solitos y estén listos para ser consultados. Por ende, con ellos se puede realizar análisis y saber cualquier tipo de información necesaria para mejor algún aspecto de la empresa. Un ejemplo claro es analizar los datos de una empresa identificando que productos son los que le generan mayor ganancia para poder adquirir más y generar más ganancia aún. Teniendo en cuenta que esto se denomina inteligencia de negocios debido a que toda empresa aparte de prestar servicios lo que requiere es generar ingresos.

Cambiar el diseño de un almacén de datos es algo complejo debido a que existen algunas instancias de base de datos. Es decir, cada una de las instancias de alas bases de datos son podrían cambiar en el transcurso del tiempo, pero su esquema es mas complejo. Se dice que es complejo porque su esquema es estático por su estructura. Los diseños de los datos almacenados de cada empresa obedecen al administrador de base de datos que es la persona que esta capacitada para realizar cualquier cambio. Teniendo en cuenta varios factores como los datos, los cambios realizados y los beneficios. En muchas bases de datos de algunas instituciones pueden llegar a encontrar la dimensión tiempo la cual es clave en las tablas. La mayor parte de los data Warehouse tienen la dimensión tiempo. El modelo de la tabla hechos suele ser bastante clara debido a que ahi es mas facil hacer algunas cosas [1].

5. Conclusión

Se logro desarrollar las actividades del laboratorio donde se creó cada una de las dimensiones necesarias de la base de datos para ellos se establece una tabla de hechos basándonos en el esquema estrella. Para ello se realiza un análisis y verifica todos los datos y además agregamos otras tablas como la de tiempo. Se tuvo en cuenta que para poder realizar todos estos procesos es realmente necesarios tener conocimientos en bases de datos, manejo de consultas, y saber programar en Python.

Se logro aplicar muchos de los conceptos y desarrollo de ejemplos aprendidos en el transcurso de la materia. Esto implica saber los que se está haciendo como el proceso de ETL que implica el uso de librerías necesarias para la carga de los datos. También se tuvo en cuenta que todos estos procesos en si sirven para poder aplicar a alguna empresa o entidad que maneje problemas similares donde sea fundamental realizar este tipo de proceso.

Es importante saber manejas todas estas herramientas porque realmente en la actualidad varias de las empresas tienen inmensas bases de datos que es difícil realizar algunas consultas. Por lo tanto, produce algún tipo de cuello de botella donde la información que se requiere no es optima en respecto al tiempo. Además, varias de las empresas focalizan sus datos como algo esencial para obtener ganancias donde la aplicación de los modelos multidimensionales es de gran ayuda. Teniendo en cuenta que cada una de sus dimensiones pueden organizarse de forma jerárquica de agregación.

Referencias

[1] DATASolution. Data warehouse (dw): El almacén de datos usable y con objetivos de negocio - lis data solutions. https://www.lisdatasolutions.com/blog/data-warehouse-dw-el-almacen-de-datos-usable-y-con-objetivos-de-negocio/, 15 2017. (Accessed on 02/19/2022).