Tecnológico Nacional de México

Instituto Tecnológico de Culiacán

Materia: Temas Selectos de Base de Datos

Profesor. Dr Clemente Garcia Gerardo

Reporte del tercer proyecto: DW de importaciones y ventas.

Alumno. Olan Castro Angel Eduardo

Domingo, 14 de abril de 2024

Contenido

Fuentes de datos

Trabajo ETL realizado

Las fuentes de datos para realizar las 2 actividades del negocio fueron 4 archivos con extensión CVS (separado por comas).

- ImportacionAutos111.
- synergy_logistics_database111.
- TicketsH.
- TicketsD.

Una gran parte del trabajo ETL fue realizado en java y lo demás en TSQL.

La extracción de los registros contenidos en los archivos se facilitó en gran parte, pues en trabajos pasados había codificado una clase extractor, la cual procesa un archivo y con un método puedes ir sacando tupla por tupla, para analizarla, limpiarla y luego insertarla a la base de datos.

```
public class Extractor{
    private final BufferedReader bf;
    public Extractor(String src) throws FileNotFoundException {
        bf = new BufferedReader(new FileReader(src));
    }

    public boolean hasNextTuple() throws IOException {
        return bf.ready();
    }

    public String[] nextTuple() throws IOException {
        String line = bf.readLine();
        String[] tuple = line.split(",");
        return tuple;
    }
}
```

Para el ETL de importaciones limpie los archivos por separado, pues al ser muy distintos, se necesitaba hacerlo de esa forma, todas las instrucciones del cargado a la base de datos están contenidas en una transacción, por lo que si en el procesamiento de alguno de los 2 archivos algo sale mal, se hace un rollback, el código del Main es sencillo:

```
SQLServerConnection db;
Connection conn;
try {
    db   =   new   SQLServerConnection("localhost",   "Importaciones",   "sa",
"Aa252328");
  conn = db.getConnection();
} catch (Exception e) {
  System.out.println(e.getMessage());
  return;
}

db.beginTran();
System.out.println("Se inicia la transaccion");

try {
  ETLArchivo1(db, conn);
  ETLArchivo2(db, conn);
  db.commitTran();
  System.out.println("COMMIT");
} catch (Exception e) {
  System.out.println(e.getMessage());
  System.out.println("Rollback");
  db.rollbackTran();
}
```

Ahora, para el primer archivo, se creó el extractor correspondiente, así la sentencia de inserción, la cual servirá para preparar el enunciado, asimismo, se declaró un contador el cual luego será explicado..

```
Extractor                    extractor                    =                    new
Extractor("ExtraccionLimpiezaCarga/Datos/ImportacionAutos111.csv");
extractor.nextTuple();

Integer cnt = 0;

String insertStatement = "INSERT INTO OLTP.Importaciones" +
                         "(estado, ciudad, marca, modelo, año, fecha, precio,
medioDeTransporte, paisDeOrigen)" +
                          " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
PreparedStatement ps = conn.prepareStatement(insertStatement);
```

Recorremos cada uno de los registros dentro del archivo con los métodos que nos provee el extractor y los procesamos.

Hay muchas partes, así que vayamos desde el inicio, extraemos el registro y pasamos a normalizar cada uno de los campos, esto se refiere a quitar espacios al inicio y al final y dejar un formato en donde por cada palabra, la primera sea mayúscula y las siguientes en minúsculas.

```
while (extractor.hasNextTuple()) {
  String tuple[] = extractor.nextTuple();
  for (int i = 0; i < tuple.length; i++) {
    tuple[i] = Rutinas.normalizarAtributo(tuple[i]);
  }
```

Este es el método usado:

```
public static String normalizarAtributo(String atributo) {
  atributo = atributo.trim().toLowerCase();
  StringBuilder cd = new StringBuilder(atributo);
  for (int i = 0; i < cd.length(); i++) {
    if (cd.charAt(i) == ' ') {
      cd.setCharAt(i + 1, Character.toUpperCase(cd.charAt(i + 1)));
    }
  }
  cd.setCharAt(0, Character.toUpperCase(cd.charAt(0)));
  return cd.toString();
}
```

Posteriormente, obtenemos el país de origen de la importación. Como en el archivo no se indica explícitamente, por medio de unas reglas de negocio impuestas, calculamos el país de origen tomando en cuenta el país de procedencia de la marca del vehículo, cabe recalcar que como no había ninguna de Japón, se decidió acomodar a Dina (mexicana) y Vw (alemana), como una.

```
  String paisOrigen = Rutinas.getPaisOrigen(tuple[2]);


public static String getPaisOrigen(String marcaVehiculo) {
  return switch (marcaVehiculo) {
    case "Ford", "Dodge", "Chevrolet" -> "Usa";
    case "Dina", "Vw" -> "Japan";
    default -> "??";
  };
}
```

De igual manera, nos aseguraremos de normalizar el estado, el significado de esto es que en el archivo el formato del estado es muy variante, siendo que, en algunas ocasiones, por ejemplo, se cuenta con Sinaloa y en otras SIN. Para erradicar eso, se utilizó el método apropiado, identificando esas irregularidades y normalizándolas.

```
  String estadoNormalizado = Rutinas.normalizarEstado(tuple[0]);

public static String normalizarEstado(String estado) {
  return switch (estado) {
    case "Dur" -> "Durango";
    case "Bc" -> "Baja California";
    case "Bcs" -> "Baja California Sur";
    case "Son" -> "Sonora";
    case "Sin" -> "Sinaloa";
    case "Nay" -> "Nayarit";
    case "Chi" -> "Chiapas";
    case "Nl" -> "Nuevo Leon";
    default -> estado;
  };
}
```

Después se fueron asignando los parámetros que necesita el PreparedStatement, además de las conversiones ya hechas, se hicieron algunas más.

```
  ps.setString(1, estadoNormalizado);
  ps.setString(2, Rutinas.normalizarCiudad(tuple[1]));
  ps.setString(3, tuple[2]);
  ps.setString(4, tuple[3]);
  ps.setString(5, tuple[4]);
  ps.setString(6, Rutinas.normalizarFecha(tuple[5]));
           ps.setString(7,           Rutinas.convertirDivisaAMXN(paisOrigen,
Double.parseDouble(tuple[6])));
    ps.setString(8,   Rutinas.getModoDeTransporte(Integer.parseInt(tuple[4]),
estadoNormalizado, cnt));
  ps.setString(9, paisOrigen);
  ps.executeUpdate();
}
```

El formato de la fecha, que estaba en DD-MM-YYYY no era el mejor que se puede manejar, por lo que se pasó a YYYY-MM-DD. De igual forma, algunas fechas tenían días de más, por lo que se regularizó mandándolo al último día del mes. Otros tipos de cambios también fueron efectuados, como completar ceros o el 20 del año.

```
public static String normalizarFecha(String fecha) {
  String[] separado = fecha.split("/");

  if (separado[0].length() == 1) {
    separado[0] = "0" + separado[0];
  }
  if (separado[1].length() == 1) {
    separado[1] = "0" + separado[1];
  }
  if (separado[2].length() == 2) {
    separado[2] = "20" + separado[2];
  }

  if (separado[1].equals("02") && Integer.parseInt(separado[0]) > 28) {
    separado[0] = "28";
  }

  if ((separado[1].equals("04") || separado[1].equals("06") ||
            separado[1].equals("09")    ||    separado[1].equals("11"))    &&
separado[0].equals("31")) {
        separado[0] = "30";
      }

  return separado[2] + separado[1] + separado[0];
}
```

La divisa que se indica en los registros pertenece a la del país de Origen, por lo que es visible que se necesita una normalización de cada uno de esos precios a una divisa en específico, en este caso, usé el peso mexicano (MXN) como base.

```
public static String convertirDivisaAMXN(String paisOrigen, double valor)
  double convertido = valor * valorDivisaMXN(paisOrigen);
  return String.format("%.4f", convertido);
}
private static double valorDivisaMXN(String paisOrigen) {
  return switch(paisOrigen) {
    case "Canada" -> 12.01;
    case "Usa" -> 16.44;
    case "South Korea" -> 0.012;
    case "Netherlands", "Spain", "Germany", "Italy" -> 17.63;
    case "China" -> 2.27;
    case "Japan" -> 0.11;
    case "Brazil" -> 3.23;
    case "Australia" -> 10.75;  }; }
```

Otra regla de negocio era la asignación del transporte utilizado, la cual se implementó de la siguiente forma:

```
public static String getModoDeTransporte(int year, String estado, int cnt) {
  return switch (year) {
    case 2020 -> esEstadoDelNorte(estado) ? "Road" : "Rail";
    case 2021 -> getMedioDeTransporteRandom();
    case 2022 -> getMedioDeTransporteEquilibrado(cnt);
    default -> "?";
  };
}


public static boolean esEstadoDelNorte(String estado) {
  return estado.equals("Baja California") ||
          estado.equals("Baja California Sur") || estado.equals("Sonora") ||
          estado.equals("Sinaloa") || estado.equals("Nuevo Leon") ||
          estado.equals("Coahuila") || estado.equals("Tamaulipas") ||
          estado.equals("Durango") || estado.equals("Chihuahua");
}

public static String getMedioDeTransporteRandom() {
  Random o = new Random();
  int i = o.nextInt(4);
  String[] medios = {"Road", "Sea", "Air", "Rail"};
  return medios[i];
}

public static String getMedioDeTransporteEquilibrado(Integer cnt) {
  String[] medios = {"Road", "Sea", "Air", "Rail"};
  String medio = medios[cnt];
  cnt = (cnt + 1) % 4;
  return medio;
}
```

Aquí se puede apreciar la importancia de nuestro contador, pues para asegurarnos de que los medios de transporte se distribuyan con 25% cada uno, se puede utilizar una estrategia muy sencilla, digamos que enumeramos los medios como números del 0 al 3, entonces, a como vayamos leyendo tuplas del 2022, podemos ir asignando los transportes de la siguiente forma: 0-1-2-3-0-1-2-3-0-1-...

Y hasta aquí es todo de la limpieza del primer archivo, al final de asignar estos parámetros, se ejecuta el insert y si al final todo sale bien, se continua con el proceso del siguiente archivo.

Para el siguiente análisis, el proceso era un poco similar, partiendo por crear el extractor y preparando la sentencia de insert.

```
Extractor extractor = new Extractor("ExtraccionLimpiezaCarga/Datos/synergy_logistics_database111.csv");
extractor.nextTuple();

System.out.println("Extractor 2 creado");

String insertStatement = "INSERT INTO OLTP.Importaciones" +
                         "(estado, ciudad, marca, modelo, año, fecha, precio, medioDeTransporte, paisDeOrigen)" +
                          " VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
PreparedStatement ps = conn.prepareStatement(insertStatement);
```

Lo interesante es que este archivo nos es demasiado útil, pues muchos de las cosas que anteriormente teníamos que hacer, no se requieren, se cuentan con más datos de los que agarrar por cada registro.

```
// 1 - direction, 2 - origin, 3 - destination, 4 - year, 5 - date, 6 - product, 7 - transport_mode, 8 - company_name, 9 - total_value
while (extractor.hasNextTuple()) {
  String tuple[] = extractor.nextTuple();
  if (!tuple[3].equals("Mexico")) {
    continue;
  }

  for (int i = 0; i < tuple.length; i++) {
    tuple[i] = Rutinas.normalizarAtributo(tuple[i]);
  }

  String[] destino = Rutinas.generarDestinoAleatoriamente();
  ps.setString(1, destino[0]);
  ps.setString(2, destino[1]);
  ps.setString(3, tuple[8]);
  ps.setString(4, tuple[6]);
  ps.setString(5, tuple[4]);
  ps.setString(6, Rutinas.normalizarFecha(tuple[5]));
            ps.setString(7,            Rutinas.convertirDivisaAMXN(tuple[2], Double.parseDouble(tuple[9])));
  ps.setString(8, tuple[7]);
  ps.setString(9, tuple[2]);
  ps.executeUpdate(); }
```

Aun así, claramente hay cosas que se requieren. Como de rutina, los atributos se normalizan y posteriormente se limpian.

Lo único con lo que no contamos en los registros es el destino, por lo que, tomando en cuenta unas reglas de negocio, se realizó eligiendo de forma aleatoria uno de los 32 estados de la republica mexicana y posteriormente, igual de manera aleatoria, se selecciona una ciudad de dicho estado, como el análisis es con fines educativos, para no hacer tan pesado el programa, se decidió usar solo 3 ciudades por estado.

Los estados son representados por un arreglo de String, mientras que las ciudades son una matriz, donde cada posición indica las ciudades del estado i.

```
public static String[] generarDestinoAleatoriamente() {
  String[] estados = {
    "Aguascalientes", "Baja California", "Baja California Sur", "Campeche",
    "Chiapas", "Chihuahua", "Coahuila", "Colima", "Durango", "Guanajuato",
    "Guerrero", "Hidalgo", "Jalisco", "Mexico", "Michoacan", "Morelos",
     "Nayarit", "Nuevo Leon", "Oaxaca", "Puebla", "Queretaro", "Quintana
Roo",
    "San Luis Potosi", "Sinaloa", "Sonora", "Tabasco", "Tamaulipas",
    "Tlaxcala", "Veracruz", "Yucatan", "Zacatecas", "Ciudad de Mexico"
  };

  String[][] ciudades = {
    { "Aguascalientes", "Calvillo", "Jesus Maria", "Pabellon de Arteaga" },
    { "Mexicali", "Tijuana", "Ensenada", "Tecate" },
    { "La Paz", "Los Cabos", "Loreto", "Santa Rosalia" },
     { "San Francisco de Campeche", "Ciudad del Carmen", "Champoton",
"Calkiní" },
     { "Tuxtla Gutierrez", "San Cristobal de las Casas", "Tapachula",
"Comitan" },
    { "Chihuahua", "Ciudad Juarez", "Parral", "Cuauhtemoc" },
    { "Saltillo", "Torreon", "Monclova", "Piedras Negras" },
    { "Colima", "Manzanillo", "Tecoman", "Villa de Alvarez" },
    { "Victoria de Durango", "Gomez Palacio", "Lerdo", "Durango" },
    { "Guanajuato", "Leon", "Celaya", "Irapuato" },
    { "Chilpancingo de los Bravo", "Acapulco", "Iguala", "Taxco" },
    { "Pachuca de Soto", "Tulancingo", "Tizayuca", "Tepeapulco" },
    { "Guadalajara", "Zapopan", "Tlaquepaque", "Tlajomulco de Zuniga" },
    { "Toluca de Lerdo", "Ecatepec de Morelos", "Nezahualcoyotl", "Naucalpan
de Juarez" },
    { "Morelia", "Uruapan", "Zamora", "Lázaro Cárdenas" },
    { "Cuernavaca", "Cuautla", "Jiutepec", "Temixco" },
    { "Tepic", "Xalisco", "Compostela", "Nayarit" },
    { "Monterrey", "Guadalupe", "San Nicolas de los Garza", "Apodaca" },
```

```
    { "Oaxaca de Juarez", "Santa Cruz Xoxocotlan", "San Bartolo Coyotepec",
"Tlacolula" },
    { "Puebla de Zaragoza", "Cholula", "Tehuacan", "Atlixco" },
     { "Santiago de Queretaro", "Corregidora", "El Marques", "San Juan del
Rio" },
    { "Chetumal", "Cancun", "Playa del Carmen", "Cozumel" },
    { "San Luis Potosi", "Soledad de Graciano Sanchez", "Matehuala", "Ciudad
Valles" },
    { "Culiacan", "Mazatlan", "Los Mochis", "Guasave" },
    { "Hermosillo", "Nogales", "Ciudad Obregon", "Guaymas" },
    { "Villahermosa", "Cardenas", "Comalcalco", "Paraiso" },
    { "Ciudad Victoria", "Reynosa", "Matamoros", "Nuevo Laredo" },
    { "Tlaxcala de Xicohtencatl", "Apizaco", "Huamantla", "Calpulalpan" },
    { "Xalapa-Enriquez", "Veracruz", "Boca del Rio", "Orizaba" },
    { "Merida", "Valladolid", "Progreso", "Tizimin" },
    { "Zacatecas", "Guadalupe", "Fresnillo", "Jerez de Garcia Salinas" },
     { "Ciudad de Mexico", "Iztapalapa", "Gustavo A. Madero", "Venustiano
Carranza" }
  };

  Random rnd = new Random();
  int i = rnd.nextInt(32), j = rnd.nextInt(3);

  String[] destino = {estados[i], ciudades[i][j]};
  return destino;
}
```

Al igual que con el otro archivo, la fecha se normaliza y la divisa es adaptada al MXN. Por último, se efectúa la inserción, y si todo sale bien, se regresará al main y realizará un commit a la transacción que estaba envolviendo el ETL de esta actividad. Si algo falla en el proceso, automáticamente se realizará un rollback.

El ETL de la segunda actividad del negocio, Ventas, prácticamente se había realizado en trabajos anteriores, sin embargo, de igual forma, se explicará brevemente.

Se inicia igual que en la anterior actividad, con la diferencia que aquí se analizan los dos archivos en el mismo método.

```
SQLServerConnection db;
Connection conn;
try {
  db = new SQLServerConnection("localhost", "Proyecto", "sa", "sa");
  conn = db.getConnection();
} catch (Exception e) {
  System.out.println(e.getMessage());
  return;
}

db.beginTran();

try {
  insertarRegistros(db);
  db.commitTran();
  System.out.println("COMMIT");
} catch (Exception e) {
  System.out.println(e.getMessage());
  System.out.println("Rollback");
  db.rollbackTran();
}
```

Empezaremos con TicketH, se crea su extractor y se prepara el statement.

```
Extractor extractorTicketH = new Extractor("ExtraccionLimpiezaCarga/Datos/TicketH.csv");
extractorTicketH.nextTuple();

Connection conexion = db.getConnection();

String query = "INSERT INTO OLTP.TicketsH VALUES (?,?,?,?,?,?)";
PreparedStatement st = conexion.prepareStatement(query);
```

El enfoque para analizar los registros es igual al hecho con Importaciones, recorrer cada uno de los registros, analizarlo y posteriormente realizar una inserción a la base de datos, con los datos limpios, normalizados y útiles para nuestro propósito.

La verdad no hay mucho que explicar, las inconsistencias surgidas por la fecha del ticket son iguales a las de Importaciones, así que simplemente se corrigen, este archivo es bastante limpio, ignorando ese minúsculo detalle.

```
while (extractorTicketH.hasNextTuple()) {
  String[] tupla = extractorTicketH.nextTuple();
  tupla[1] = Rutinas.corregirFecha(tupla[1]);
  for (int i = 0; i < 6; i++) {
    st.setString(i + 1, tupla[i]);
  }
  st.executeUpdate();
}
```

Ahora, moviéndonos al análisis de TicketD, se realiza la rutina normal, sin embargo, ahora la sentencia es diferente. Resulta que en los registros las reglas de normalización no se respetan del todo, una llave primaria puede aparecer en más de un registro, para eso, se optó por aislar los 2 casos: si ya está presente en la base de datos una tupla con esa llave primaria, se realiza un update aumentando las unidades que se tienen en dicha tupla, de otro modo, se agrega una tupla nueva.

```
Extractor extractorTicketD = new Extractor("ExtraccionLimpiezaCarga/Datos/TicketD.csv");
extractorTicketD.nextTuple();

query = "IF EXISTS(SELECT TICKET FROM OLTP.TICKETSD WHERE TICKET = ? AND producto = ?) " +
         "UPDATE OLTP.TICKETSD SET UNIDADES = UNIDADES + ? WHERE TICKET = ? AND producto = ? " +
        "ELSE " +
        "INSERT INTO OLTP.TICKETSD VALUES(?,?,?,?)";
st = conexion.prepareStatement(query);
```

Se recorre cada registro y se limpian los campos, como se vio anteriormente, de igual forma, casi todos los campos están limpios, por lo que la transformación es casi mínima. Lo que se tuvo que cambiar era el precio unitario que se tenía, pues según información empresarial, el precio de los tickets era erróneo y era mejor calcularlo como la división del precio total entre las unidades compradas.

```
while (extractorTicketD.hasNextTuple()) {
  String[] tupla = extractorTicketD.nextTuple();
  for (int i = 0; i < 5; i++) {
    tupla[i] = Rutinas.limpiarCampo(tupla[i]);
```

```
  }
      tupla[3]      =      String.valueOf(Integer.parseInt(tupla[4])      /
Integer.parseInt(tupla[2]));

  st.setString(1, tupla[0]);
  st.setString(2, tupla[1]);
  st.setString(3, tupla[2]);
  st.setString(4, tupla[0]);
  st.setString(5, tupla[1]);
  for (int i = 0; i < 4; i++) {
    st.setString(i + 6, tupla[i]);
  }
  st.executeUpdate();
}
```

Al igual que con la limpieza de la primera actividad, si todo sale bien, se realiza un commit y si en alguna parte, falla el proceso, automáticamente se realiza un rollback.

Todo el proceso realizado en java fue para extraer los datos de los archivos CVS y insertarlos a tablas en una base de datos, la cual fungirá como la de un sistema OLTP de la empresa.

El siguiente paso fue meter registros a las tablas dimensiones, en el caso de la dimensión Fecha esto fue muy fácil, haciendo uso de unas funciones y procedimientos almacenados codificados.

```
create proc sp_AgregarFecha @fecha date
as begin
  insert into Dimension.Fecha(fecha, año, nombreMes, diaMes, diaAño,
semanaAño, semanaMes,
              nombreDiaSemana, mesesBimestre, mesesTrimestre,
mesesCuatrimestre,
              mesesSemestre, nombreTemporada, esFinDeSemana,
estaEnAñoBisiesto)
    values (@fecha, year(@fecha), datename(month, @fecha), day(@fecha),
datepart(dayofyear, @fecha),
        datepart(wk, @fecha), (day(@fecha) - 1) / 7 + 1, datename(weekday,
@fecha),
        dbo.GetNmestre(2, month(@fecha)), dbo.GetNmestre(3, month(@fecha)),
        dbo.GetNmestre(4, month(@fecha)), dbo.GetNmestre(6, month(@fecha)),
        dbo.getNombreTemporada(month(@fecha), day(@fecha)),
```

```
        dbo.EsFinDeSemana(@fecha), dbo.EsAñoBisiesto(year(@fecha)))
end
go

create proc dbo.sp_AgregarRangoDeFechas @fechaInicio date, @fechaFin date
as
begin
  while @fechaInicio < @fechaFin
  begin
    exec sp_AgregarFecha @fechaInicio
    set @fechaInicio = dateAdd(day, 1, @fechaInicio)
  end
end
go
exec dbo.sp_AgregarRangoDeFechas '2015-01-01', '2024-01-01'
go
```

Básicamente, recorremos cada día en un intervalo de fechas e insertamos una tupla que corresponda a los datos de ese día. Es fácil de apreciar que para conseguir dichos datos se usaron algunas funciones, las cuales se verán a continuación. Lo único que posiblemente merezca explicación es la función GetNMestre, esta función te retorna un intervalo de meses, digamos que N = 2 y mes = 3, entonces te retorna 'Marzo-Abril' pues ese mes está en el bimestre de marzo a abril, es decir, el que va desde el mes 3 al 4. Como se manejan bimestres, trimestres, etc. Se creó la función.

```
create function GetNombreMes(@mes int)
returns nvarchar(20)
as begin
  return (select dateName(month, dateAdd(month, @mes, -1)))
end
go

create function GetNmestre(@N int, @mes int)
returns nvarchar(40)
as begin
  declare @numero int = (@mes - 1) / @N
  declare @mesInicio nvarchar(20) = dbo.GetNombreMes(@numero * @N + 1)
  declare @mesFin nvarchar(20) = dbo.GetNombreMes((@numero + 1) * @N)
  return @mesInicio + '-' + @mesFin
end
go

create function GetNombreTemporada(@mes int, @dia int)
```

```
returns nvarchar(40)
as begin
  if (@mes in (1, 2) or (@mes = 12 and @dia >= 21) or (@mes = 3 and @dia <
20))
    return 'Invierno'
  if (@mes in (4, 5) or (@mes = 3 and @dia >= 21) or (@mes = 6 and @dia <
21))
    return 'Primavera'
  if (@mes in (7, 8) or (@mes = 6 and @dia >= 21) or (@mes = 9 and @dia <
23))
    return 'Verano'
  return 'Otoño'
end
go

create function dbo.EsFinDeSemana(@fecha date)
returns nvarchar(40)
as begin
  if (datepart(weekday, @fecha) in (1, 7))
    return 'Es fin de semana'
  return 'No es fin de semana'
end
go

alter function dbo.EsAñoBisiesto(@año int)
returns nvarchar(40)
as begin
  if ((@año % 4 = 0 and @año % 100 != 0) or
    (@año % 100 = 0 and @año % 400 = 0))
    return 'Es año bisiesto'
  return 'No es año bisiesto'
end
go
```

Para las demás dimensiones los datos fueron agregados manualmente, en conjunto con un generador de inserts. Se mostraran algunos de estos inserts para dar idea.

```
INSERT INTO Dimension.PaisDeOrigen (nombre, continente, divisa,
densidadDePoblacion, cultura, clima, costoDeVida) VALUES
('China', 'Asia', 'Yuan', 'Pais muy poblado', 'Pais ateo', 'Clima templado',
'Costo de vida normal'),
('India', 'Asia', 'Rupia', 'Pais muy poblado', 'Pais hindu', 'Clima
tropical', 'Costo de vida bajo'),
```

```
('USA', 'America', 'Dolar', 'Pais con poblacion normal', 'Pais cristiano',
'Clima variado', 'Costo de vida alto'),
('Indonesia', 'Asia', 'Rupia indonesia', 'Pais muy poblado', 'Pais
musulman', 'Clima tropical', 'Costo de vida bajo'),
('Pakistan', 'Asia', 'Rupia pakistani', 'Pais muy poblado', 'Pais musulman',
'Clima arido', 'Costo de vida normal'),


INSERT INTO Dimension.MedioDeTransporte (nombre, tipo, seguridad, costo,
duracionTraslado, impactoAmbiental, capacidadCarga, tipoCombustible,
seguimientoEnvios) VALUES
('Road', 'Transporte terrestre', 'Transporte seguro', 'Costo de transporte
normal', 'Traslado normal', 'Transporte sin impacto ambiental', 'Mucha
capacidad de carga', 'Gasolina', 'Seguimiento alto'),
('Rail', 'Transporte terrestre', 'Transporte con seguridad promedio', 'Costo
de transporte normal', 'Traslado normal', 'Transporte sin impacto
ambiental', 'Capacidad de carga normal', 'Electricidad', 'Seguimiento
alto'),
('Air', 'Transporte aereo', 'Trasporte seguro', 'Costo de transporte caro',
'Traslado rapido', 'Traslado con impacto ambiental', 'Poca capacidad de
carga', 'Combustible de aviacion', 'Seguimiento alto'),
('Sea', 'Transporte naval', 'Transporte poco seguro', 'Costo de transporte
economico', 'Traslado lento', 'Traslado con impacto ambiental', 'Capacidad
de carga normal', 'Combustible marino', 'Seguimiento limitado');


INSERT INTO Dimension.Estado (id, nombre, zona, clima, ambiente,
giroPrincipal, culturaDominante, musicaDominante, criminalidad) VALUES
(1, 'Aguascalientes', 'Zona centro', 'Templado semiseco', 'Rural',
'Agricultura', 'Cultura mestiza', 'Se escucha musica nortena', 'Estado con
criminalidad baja'),
(2, 'Baja California', 'Zona norte', 'Seco desertico', 'Urbano', 'Industria
manufacturera', 'Cultura mestiza', 'Se escucha musica nortena', 'Estado con
criminalidad alta'),
(3, 'Baja California Sur', 'Zona sur', 'Seco desertico', 'Costero',
'Turismo', 'Cultura mestiza', 'Se escucha musica tropical', 'Estado con
criminalidad baja'),
(4, 'Campeche', 'Zona sur', 'Calido humedo', 'Costero', 'Petroleo', 'Cultura
maya', 'Se escucha musica tropical', 'Estado con criminalidad baja'),
(5, 'Coahuila', 'Zona norte', 'Arido', 'Rural', 'Mineria', 'Cultura
mestiza', 'Se escucha musica nortena', 'Estado con criminalidad media'),
(6, 'Colima', 'Zona centro', 'Calido subhumedo', 'Costero', 'Agricultura',
'Cultura mestiza', 'Se escucha musica tropical', 'Estado con criminalidad
baja'),
```

Una vez hayan sido llenadas las dimensiones, el último y no tan complicado paso es llenar las vistas materializadas (VM), es decir hacer el cargado final. Para esto, se creó cada tupla de las VM utilizando las tablas OLTP llenadas anteriormente por los programas en Java, posteriormente, enlazamos las VM agregando restricciones de llave foránea a los parámetros que así lo sean, pues la sentencia select into que usamos para poblar las VM es limitada y no nos permite hacer eso.

```
create proc sp_CargarVMImportaciones
as begin
  select paisDeOrigen, medioDeTransporte, fecha, importe = SUM(precio),
unidades = COUNT(*)
  into VistaMaterializada.Importaciones
  from OLTP.Importaciones
  group by paisDeOrigen, medioDeTransporte, fecha

  alter table VistaMaterializada.Importaciones
  add constraint PK_VMImportaciones primary key(PaisDeOrigen,
MedioDeTransporte, Fecha)

  alter table VistaMaterializada.Importaciones
  add constraint FK_VMImportaciones_PaisDeOrigen foreign key(paisDeOrigen)
references Dimension.paisDeOrigen(nombre)
  alter table VistaMaterializada.Importaciones
  add constraint FK_VMImportaciones_MedioTransporte  foreign
key(medioDeTransporte) references Dimension.medioDeTransporte(nombre)
  alter table VistaMaterializada.Importaciones
  add constraint FK_VMImportaciones_Fecha  foreign key(Fecha) references
Dimension.fecha(fecha)
end

create proc sp_CargarVMVentas
as begin
  select TH.estado, TD.producto, fecha, importe = SUM(precio * unidades),
mayorCantidadUnidadesVendidas = max(unidades)
  into VistaMaterializada.Ventas
  from OLTP.TicketsH TH
  inner join OLTP.TicketsD TD on TH.ticket = TD.ticket
  group by TH.estado, TD.producto, TH.fecha

  alter table VistaMaterializada.Ventas
  add constraint PK_VMVentas primary key(estado, producto, Fecha)

  alter table VistaMaterializada.Ventas
```

```
  add constraint FK_VMVentas_Estado foreign key(Estado) references
Dimension.Estado(id)
  alter table VistaMaterializada.Ventas
  add constraint FK_VMVentas_Producto  foreign key(Producto) references
Dimension.Producto(id)
  alter table VistaMaterializada.Ventas
  add constraint FK_VMVentas_Fecha  foreign key(Fecha) references
Dimension.fecha(fecha)
end
```

Tablas dimensión.

Las tablas dimensión son un pilar de los sistemas OLAP, ya que de ellas depende, en su mayoría, que podamos hacer un análisis robusto, con la granularidad deseada.

Si bien en muchas de estas se tienen atributos de tipo entero, fecha o decimal. Lo que predominó fueron aquellos que utilizaban valores de tipo nominal, pues nos permite agrupar la información de mejor manera, a continuación, se podrán observar ejemplos de esto, como el bimestre, el cual es más fácil de entender como 'Enero-Febrero' que como 1, o en la población de un país, la cual se agrupó en distintos valores como 'Muy poblado', 'Poblado normal', 'Poco poblado'.

Fecha.

Como en todo sistema OLAP, esta dimensión no podía faltar.

Para dar un amplio espectro de opciones para analizar, se agregaron diversos campos, los cuales surgen con naturalidad al querer analizar el rendimiento de una actividad del negocio en un periodo determinado.

Esta dimensión es utilizada en las 2 vistas materializadas.

```
create table Dimension.Fecha (
  fecha date primary key not null,
  año int not null,
  nombreMes nvarchar(40) not null,
  diaMes int not null,
  diaAño int not null,
  semanaAño int not null,
  semanaMes int not null,
  nombreDiaSemana nvarchar(40) not null,
  mesesBimestre nvarchar(40) not null,
  mesesTrimestre nvarchar(40) not null,
  mesesCuatrimestre nvarchar(40) not null,
  mesesSemestre nvarchar(40) not null,
  nombreTemporada nvarchar(40) not null,
  esFinDeSemana nvarchar(20) not null,
  estaEnAñoBisiesto nvarchar(20) not null
)
```

Importaciones

Para esta actividad el negocio se consideraron útiles 2 dimensiones.

País de origen.

Siendo el país de origen de la importación, la cual contiene atributos que serán de gran utilidad para realizar un análisis completo. Algunos de estos, la divisa que maneja el país, la facilidad del comercio entre los países, con el TCL, y mucho más.

```
create table Dimension.PaisDeOrigen(
  nombre nvarchar(40) not null primary key,
  continente nvarchar(40) not null,
  divisa nvarchar(40) not null, --
  densidadDePoblacion nvarchar(40) not null,
  cultura nvarchar(40) not null,
  clima nvarchar(40) not null,
  ambiente nvarchar(40) not null,
  costoDeVida nvarchar(40) not null,
  sistemaPolitico nvarchar(40) not null,
  TLC nvarchar(40) not null,
  idiomaOficial nvarchar(40) not null,
  zonaHoraria nvarchar(40) not null
)
```

Medio de transporte

Representa como fueron transportados los paquetes de importación, también cuenta con atributos muy útiles para su análisis, de hecho, todos los atributos de esta clase son nominales, pues al haber una amplia gama de valores para cada uno de ellos, es necesario crear grupos para que el análisis tenga sentido.

```
create table Dimension.MedioDeTransporte(
  nombre nvarchar(40) not null primary key,
  tipo nvarchar(40) not null,
  seguridad nvarchar(40) not null,
  costo nvarchar(40) not null,
  duracionTraslado nvarchar(40) not null,
  impactoAmbiental nvarchar(40) not null,
  capacidadCarga nvarchar(40) not null,
  tipoCombustible nvarchar(40) not null,
  seguimientoEnvios nvarchar(40) not null
)
```

Ventas

Para esta actividad del negocio se consideraron útiles 2 dimensiones.

Producto

El objetivo de haber utilizado esta dimensión es tener la capacidad de analizar como distintos grupos de productos aportan a la ganancia del negocio, agregando atributos muy útiles como si es un producto nacional, el rango de peso, entre otros.

Cabe decir que esta dimensión fue de las mas tardadas en llenar, pues se cuentan con aproximadamente 1000 productos distintos y para mantener congruentes los datos, en vez de utilizar un script de automatización, se generó una sentencia insert.

```
create table Dimension.Producto(
  id int primary key not null,
  nombre nvarchar(80) not null,
  tipo nvarchar(80) not null,
  marca nvarchar(80) not null,
  proveedor nvarchar(80) not null,
  esPerecedero nvarchar(80) not null,
  edadObjetivo nvarchar(80) not null,
  esNacional nvarchar(80) not null,
  añoIngreso nvarchar(80) not null,
  peso nvarchar(80) not null,
  esFragil nvarchar(80) not null
)
```

Estado

De igual forma, es de gran utilidad poder extraer información de las ventas en un estado, por lo que se optó por crear dicha dimensión, contando con atributos que pueden ser muy útiles, como la música dominante, que los estados que escuchan algún genero musical en específico, prefieren un tipo de producto. O analizar en que condiciones ambientales se obtiene un mayor beneficio.

```
create table Dimension.Estado(
  id int primary key not null,
  nombre nvarchar(40) not null,
  zona nvarchar(40) not null,
  clima nvarchar(40) not null,
  ambiente nvarchar(40) not null,
  giroPrincipal nvarchar(40) not null,
  culturaDominante nvarchar(40) not null,
  musicaDominante nvarchar(40) not null,
  criminalidad nvarchar(40) not null,
)
```

Tablas de hechos.

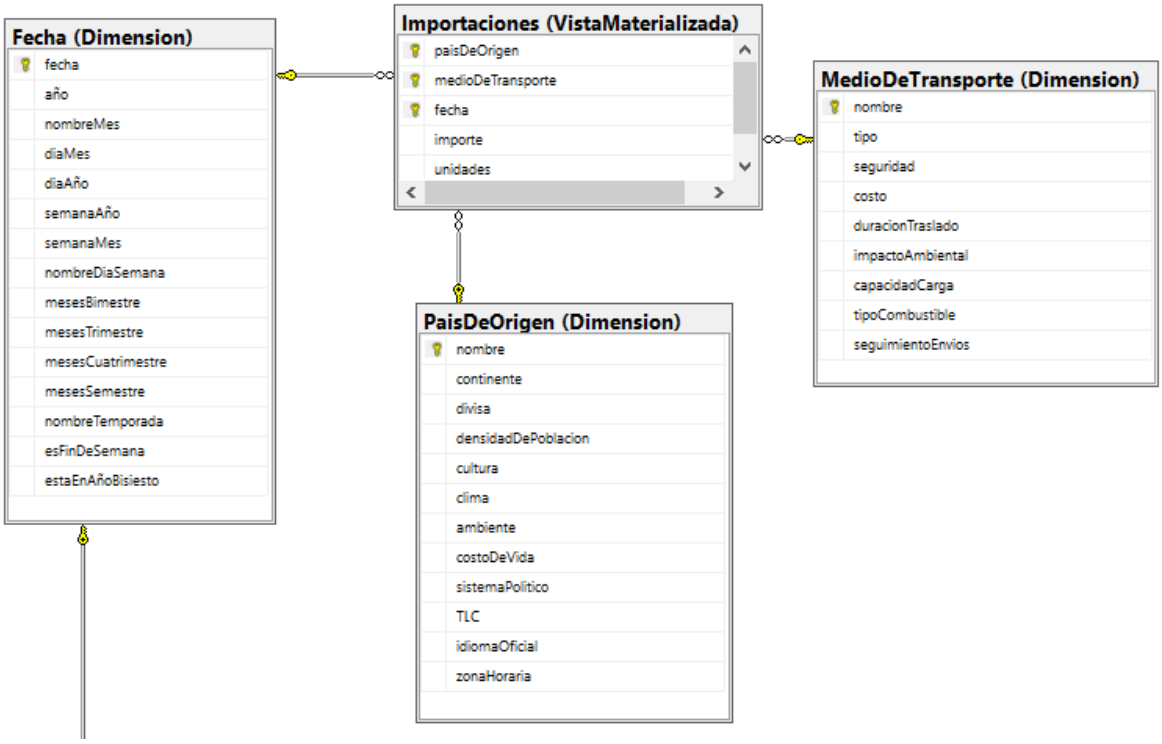
Como se mencionó anteriormente, una vez llenadas las tablas de dimensión y extraídos los datos esenciales, el siguiente paso es poblar la vista materializada

Importaciones

En este caso, esta se define de la siguiente manera:

```
create table VistaMaterializada.Importaciones(
	paisDeOrigen nvarchar(40) not null foreign key references
Dimension.PaisDeOrigen(nombre),
	medioDeTransporte nvarchar(40) not null foreign key references
Dimension.MedioDeTransporte(nombre),
	fecha date not null foreign key references Dimension.Fecha(fecha),
	importe decimal(16, 4),
	unidades int,
	primary key(paisDeOrigen, medioDeTransporte, fecha)
)
```

Es necesario aclarar que esa sentencia créate es solo con fines ilustrativos, ya que esta table se crea y llena a través de la sentencia insert into select, explicada en secciones anteriores. El diagrama es el siguiente:



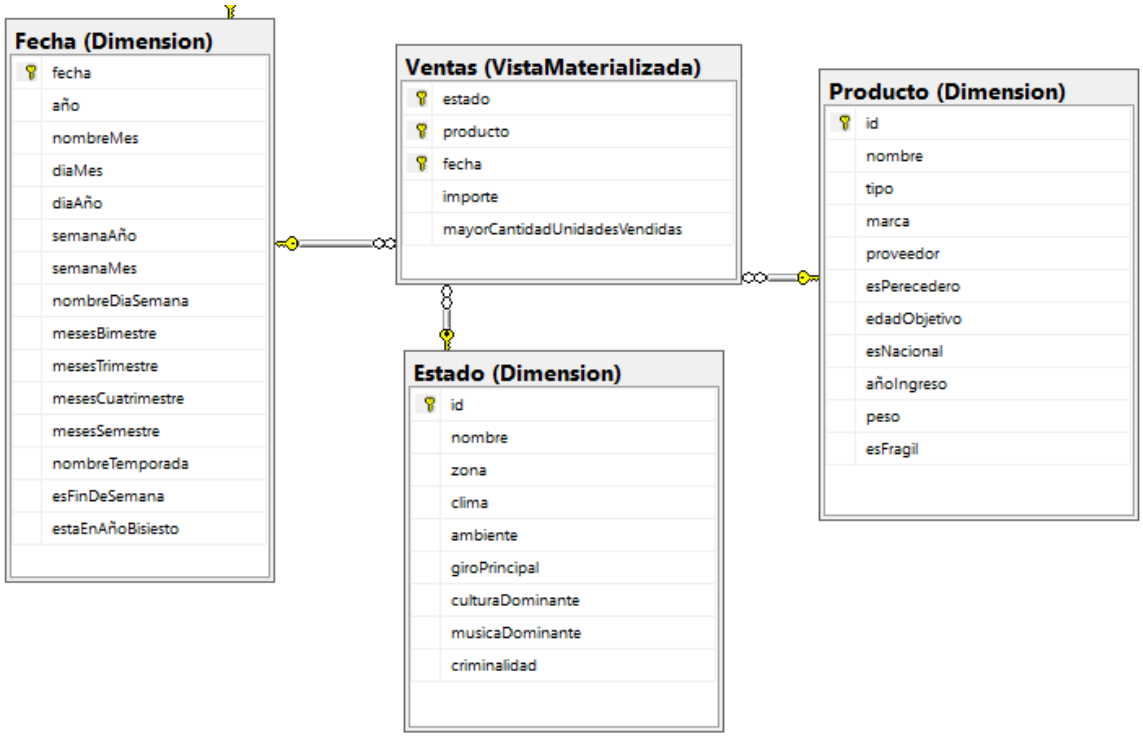
Ventas

Se define de la siguiente manera:

```
create table VistaMaterializada.Ventas(
	estado int not null foreign key references Dimension.Estado(id),
	producto int not null foreign key references Dimension.Producto(id),
	fecha date not null foreign key references Dimension.Fecha(fecha),
	importe decimal(16, 4),
	mayorCantidadUnidadesVendidas int,
	primary key(estado, producto, fecha)
)
```

De igual forma esta sentencia es mostrada con fines ilustrativos.

La razón de usar en las dimensiones Estado y Producto identificadores en lugar de simplemente el nombre es que desde la información extraída, se tienen esos identificadores y para evitar trabajo extra, se dejaron. Este es el diagrama:



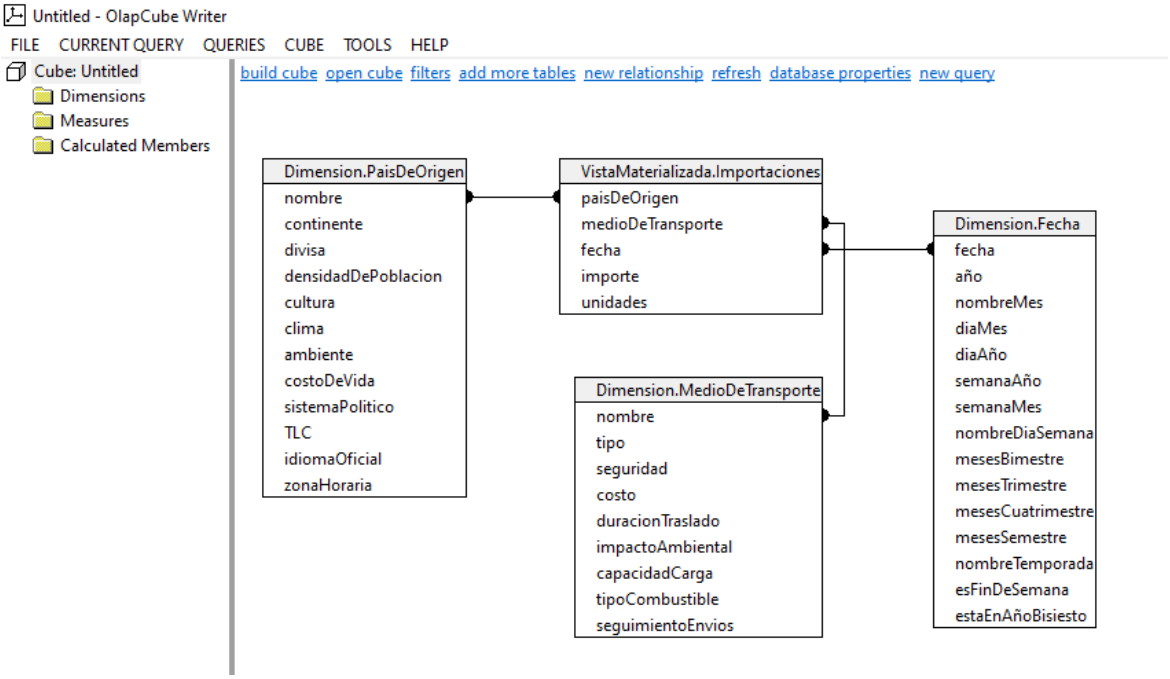
Herramienta OLAP.

La herramienta OLAP utilizada fue OlapCube, un software antiguo, pero bastante potente para la generación de cubos, además, es gratuito.

Actividad del negocio: importaciones

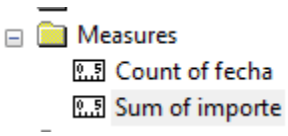
Esquema en estrella

La generación del esquema fue sencilla en su totalidad, simplemente se necesita conectarse a la base de datos local ingresando las credenciales y seleccionar en la base de datos deseada las tablas que se mostrarán en el diagrama.



Medidas utilizadas

Se usaron las medidas importe y unidades, las cuales fueron calculadas al sumarizar la información con el group by, cuando se poblaron las VM.



Jerarquías, creación del cubo y consultas

Con respecto a la creación del cubo, es muy sencillo, una vez tengamos asignadas las medidas y las jerarquías a utilizar, se presiona el botón construir cubo y posteriormente seleccionaremos que herramienta utilizaremos para consultar sobre él, en este caso, se optó por elegir a Excel.

Esta jerarquía fue muy interesante pues se puede rescatar información como que los países asiáticos que tienen como divisa el Yen, realizan más importaciones a México en el periodo de enero-marzo en el invierno.

- continente
- divisa
- nombreTemporada
- mesesTrimestre

Etiquetas de fila	Sum of importe	Count of fecha
⊞ America	$ 42,126,369,747,645.20	3117
⊟ Asia		
⊞ Won	$ 18,108,420.00	161
⊟ Yen		
⊟ Invierno		
January-March	$ 40,838,722,875.50	683
October-December	$ 5,516,348,059.09	77
⊞ Otoño	$ 46,283,111,951.78	631
⊞ Primavera	$ 47,005,988,711.66	843
⊞ Verano	$ 47,800,666,738.55	705
⊞ Yuan	$ 28,361,380,000.00	350
⊞ Europa	$ 60,717,367,400.00	316
⊞ Oceania	$ 903,000,000.00	51
Total general	$ 42,403,814,441,801.80	6934

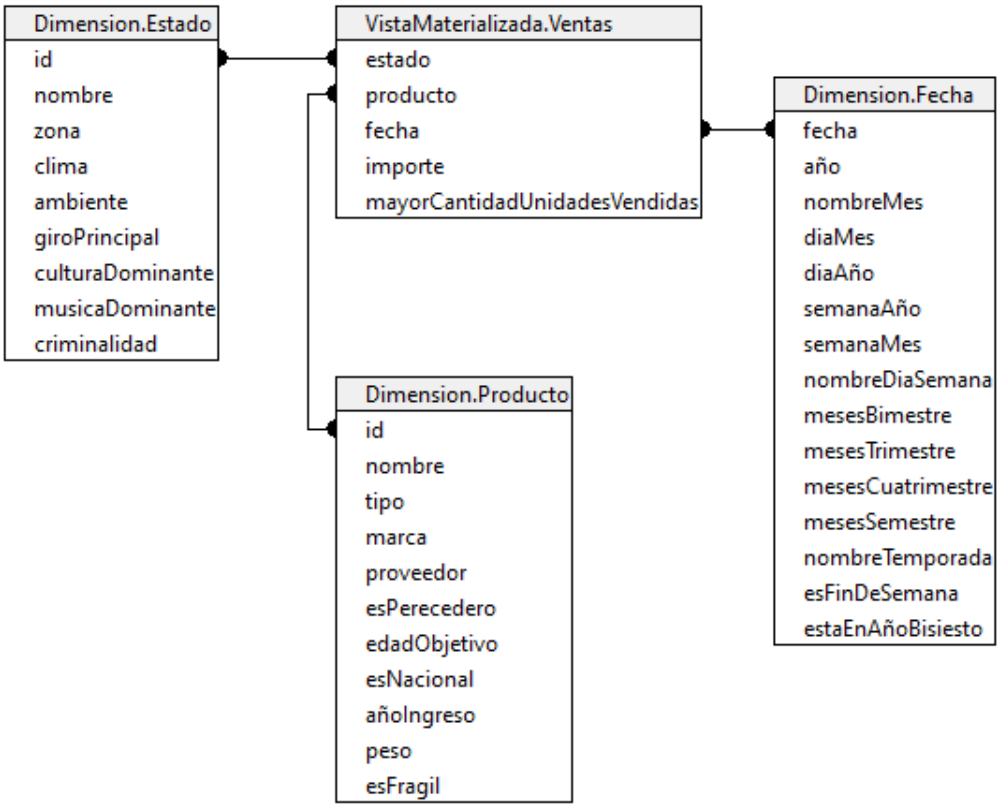
Otra jerarquía bastante curiosa es la siguiente. La cual nos permite conocer datos como que en la primavera de 2020 los principales importadores eran Países en los que el sistema de gobierno era una monarquía constitucional o una república federal, así como que suelen enviar menos paquetes de capacidad de carga grande.

- año
- nombreTemporada
- sistemaPolitico
- capacidadCarga

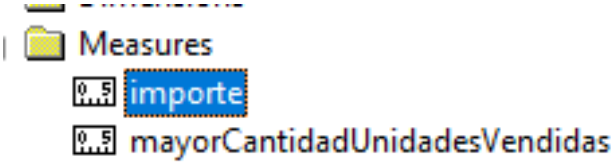
Etiquetas de fila	Sum of importe	Count of fecha
⊞ 2015	$ 314,527,490.00	34
⊞ 2016	$ 10,180,940,150.00	156
⊞ 2017	$ 52,897,480,000.00	405
⊞ 2018	$ 83,647,691,830.00	450
⊞ 2019	$ 34,529,148,190.00	403
⊟ 2020		
⊞ Invierno	$ 3,506,331,352,070.53	515
⊞ Otoño	$ 3,481,217,071,247.61	363
⊟ Primavera		
⊟ Monarquía Constitucional		
Capacidad de carga normal	$ 51,741,219,289.71	184
Mucha capacidad de carga	$ 12,421,632,212.22	92
⊞ República	$ 300.00	46
⊟ República Federal		
Capacidad de carga normal	$ 698,520,143,802.48	92
Mucha capacidad de carga	$ 2,795,668,802,378.40	92
⊞ República Socialista	$ 6,889,450,000.00	65
⊞ Verano	$ 3,597,175,045,438.82	387
⊞ 2021	$ 14,036,209,501,099.60	2920
⊞ 2022	$ 14,036,070,436,302.40	730
Total general	$ 42,403,814,441,801.80	6934

Actividad del negocio: ventas

Esquema en estrella



Medidas utilizadas



Jerarquías, creación del cubo y consultas

Esa es la primera jerarquía que me resulta muy interesante pues explora distintos atributos de los productos, intervalos y zonas geográficas.

- esPerecedero
- marca
- año
- mesesCuatrimestre
- zona

Podemos obtener mucha información, de entrada, podemos darnos cuenta de que los productos no perecederos son los que más ingresos dan al negocio.

Etiquetas de fila	importe	mayorCantidadUnidadesVendidas
⊞ Es un producto no perecedero	$ 4,271,132,105.00	4980942
⊞ Es un producto perecedero	$ 81,197,615.00	95188
Total general	**$ 4,352,329,720.00**	**5076130**

Y adentrándonos un poco más nos podemos percatar de que Amazon tiene una mayor venta en la zona centro del país, siendo el cuatrimestre enero-abril el mejor.

⊟ **Es un producto no perecedero**		
⊞ **Ace**	**4266584**	**4914**
⊞ **Adidas**	**4363728**	**4961**
⊟ **Amazon**		
⊞ 2020	1404549	1711
⊟ 2021		
⊟ **January-April**		
Zona centro	254753	292
Zona norte	155441	177
Zona sur	163821	203
⊟ **May-August**		
Zona centro	234001	180
Zona norte	146401	153
Zona sur	130944	162
⊟ **September-December**		
Zona centro	186144	207
Zona norte	143835	173
Zona sur	159635	195
⊞ 2022	1557501	1795

Además, que Bimbo es el rey de los productos perecederos.

Etiquetas de fila	importe	mayorCantidadUnidadesVendidas
⊞ Es un producto no perecedero	**4271132105**	**4980942**
⊟ Es un producto perecedero		
⊞ Bimbo	46743514	54643
⊞ Coca-Cola	4073139	4814
⊞ Corona	4253169	4968
⊞ Dunkin'	4211466	4973
⊞ Gamesa	4487948	5281
⊞ Heineken	9193310	10830
⊞ Lala	3789442	4480
⊞ San Juan	4445627	5199
Total general	**4352329720**	**5076130**

Así como que, desde que empezó la pandemia, los compradores de cerveza Corona suelen adquirir más en tiempos a finales de año.

Etiquetas de fila	importe	mayorCantidadUnidadesVendidas
⊞ Es un producto no perecedero	**4271132105**	**4980942**
⊟ Es un producto perecedero		
⊞ Bimbo	**46743514**	**54643**
⊞ Coca-Cola	**4073139**	**4814**
⊟ Corona		
⊟ 2020		
⊞ January-April	**513013**	**591**
⊞ May-August	**410300**	**479**
⊞ September-December	**435880**	**526**
⊟ 2021		
⊞ January-April	**478646**	**630**
⊞ May-August	**465766**	**523**
⊞ September-December	**515536**	**566**
⊟ 2022		
⊞ January-April	**438119**	**527**
⊞ May-August	**456216**	**505**
⊞ September-December	**539693**	**621**

Pero esta otra jerarquía no es menos interesante, veamos las consultas.

- zona
- musicaDominante
- año
- nombreTemporada
- tipo

Podemos darnos cuenta que los disfrutadores de la música banda suelen adquirir mayormente maquillaje, viendo las tendencias del norte y centro del país en el 2022

Zona norte		
Se escucha musica banda		
2020	42820743	50798
2021	43413459	49936
2022		
Invierno		
Accesorios	250144	282
Alimento	13761	13
Bebé	9437	9
Bebidas	79866	127
Componentes de computadora	43263	55
Componentes de red	1677	3
Cuidado de la piel	932866	1089
Cuidado del cabello	6072	17
Cuidado personal	1409643	1747
Electrodoméstico	459052	558
Electrónica de seguridad	7841	14
Electrónico	11301	17
Electronicos	789255	955
Electrónicos	843308	969
Granos	100814	110
Higiene personal	1861803	2280
Lacteos	24898	31
Limpieza	177633	239
Maquillaje	2580580	3203
Oficina	114367	161

Zona centro		
Se escucha musica banda		
2020	51871093	60719
2021	48077928	55375
2022		
Invierno	**12436148**	**15454**
Otoño	**12488672**	**14746**
Primavera		
Accesorios	313827	320
Alimento	9960	9
Bebé	4796	4
Bebidas	67991	81
Componentes de computadora	87082	101
Componentes de red	2342	4
Cuidado de la piel	765700	884
Cuidado del cabello	30480	21
Cuidado personal	1191158	1332
Electrodoméstico	434962	525
Electrónica de seguridad	8811	9
Electrónico	9614	20
Electronicos	668041	817
Electrónicos	686543	847
Granos	72610	103
Higiene personal	1606415	1875
Lacteos	28807	39
Limpieza	156159	169
Maquillaje	2033546	2378

O que en la zona centro del país, las ciudades que escuchan música ranchera producen más ingreso para la compañía.

Zona centro		
Se escucha musica banda	142921618	166961
Se escucha musica huapango	405966583	472153
Se escucha musica mariachi	128857226	150582
Se escucha musica nortena	281952562	330197
Se escucha musica pop	261734630	304635
Se escucha musica ranchera	564723264	658220
Se escucha musica tropical	271053260	316978

También se obtuvo que los habitantes de la zona sur del país en donde se escucha música tropical tienden a cuidarse mucho la piel, comprando en su mayoría artículos relacionados al cuidado personal.

Zona sur		
Se escucha musica oaxaquena	**137548558**	**161200**
Se escucha musica tropical		
2020	356568468	415107
2021	361445807	423123
2022		
Invierno		
Accesorios	2935127	3548
Alimento	118224	144
Bebé	146913	144
Bebidas	1026325	1246
Componentes de computadora	532059	632
Componentes de red	119208	164
Cuidado de la piel	8097110	10049
Cuidado del cabello	196228	256
Cuidado personal	13963380	17132
Electrodoméstico	4609734	5677
Electrónica de seguridad	150536	159
Electrónico	101783	154
Electronicos	7640207	9339
Electrónicos	7242891	8826
Granos	894878	1098
Higiene personal	18369552	22417
Lacteos	166537	225
Limpieza	1626100	2090
Maquillaje	23649309	29364