**PROYECTO IMPLEMENTACION DE UN DATAMART DE UN CASO DE ESTUDIO**

**Elaborado por:**

**Profesor:**

**Semestre:**

El informe deberán ser impresos en Papel Bond Blanco tamaño A4 y una copia en digital, los márgenes serán de 3cm por lado, la letra empleada será Arial 10.

**El informe FINAL deberá contener:**

1. Carátula
2. Índice de contenidos
3. Índice de Figuras
4. Índice de tablas
5. **Planeación y administración del proyecto**

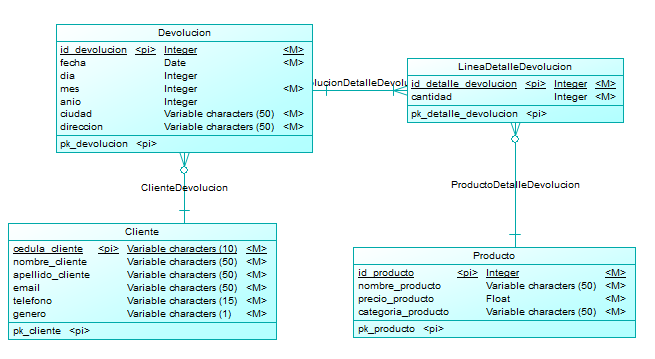
* Introducción Descripción de la organización (Razón Social, ubicación, rubro económico, clientes, competidores, misión, organigrama.
* Titulo del proyecto
* Calendario tentativo
* Recursos : Personal, Hardware, Software, Costos
* Autorización del proyecto

1. **Análisis de requerimientos** 
   1. El Negocio
      1. Objetivos del negocio
      2. Los problemas del Negocio
      3. Descripción de los procesos a modelar
      4. Priorización de procesos
      5. Necesidades de Información de alto nivel
      6. Matriz Procesos vs. Entidades del negocio
      7. Definición de Indicadores de Rendimiento, Dimensiones
      8. Fuentes de datos
2. **el modelo logico: modelo dimensional**
   1. Selección del Proceso sobre el que se construirá el Data Mart
   2. Definir la granularidad
   3. Definir las dimensiones
   4. Definir las tablas de hechos
      1. Mapear las medidas del negocio en las tablas de hechos
      2. Identificar formulas
   5. Modelo de Estrella
   6. Diseño de base de datos intermedia
   7. Diccionario de datos
3. **Diseño técnico de la arquitectura**
   1. Infraestructura: servidores, equipos.
   2. Flujo técnico de la arquitectura: back room, front room.
4. **Procesos de extracción, transformación y carga**
   1. Identificar fuentes y destinos detallados

**Fuentes**

La fuente de datos principal será una base de datos de PostgreSQL.

Cuyo esquema conceptual es el siguiente:

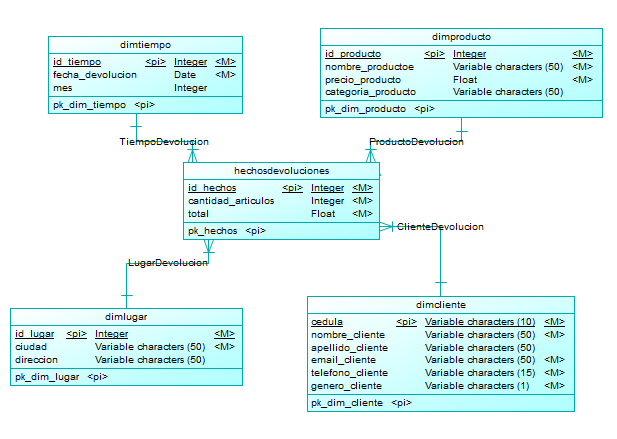


En esta fuente se almacena las devoluciones que realizan los clientes de uno o varios productos registrando la fecha, dirección y cantidad por producto devuelto.

**Destinos**

El destino será un Datamart compuesto de dimensiones y una tabla de hechos cuyo indicador principal es el número de artículos devueltos por cada producto y el dinero total que se perdió por esa devolución.

Su esquema conceptual es el siguiente:



* 1. Descripción de herramienta de ETL

La herramienta de Extracción-Transformación-Carga a utilizarse es PENTAHO.

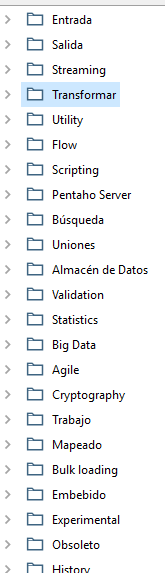
Pentaho es una plataforma opensource con componentes que incluyen tecnologías como Big Data o Internet de las Cosas (IoT).

Algunas de estas herramientas o componentes más conocidos son Pentaho Business Analytics (Pentaho BA) para ejecutar recursos como los informes o cuadro de mandos, CTools para crear y gestionar Dashboards, o Pentaho Data Integration (PDI), tema principal de este artículo.

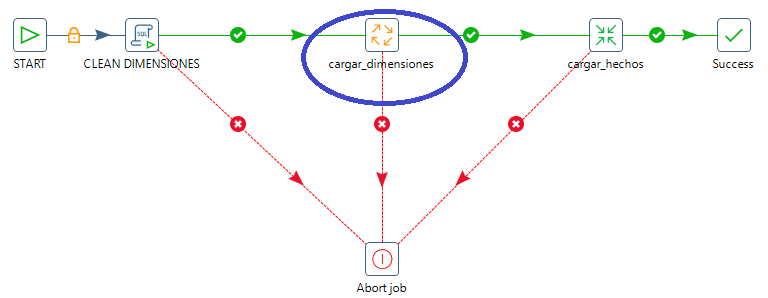


Pentaho Data Integration (PDI), cuyo nombre en clave es Kettle, es una de las herramientas o componentes de Pentaho Suite que permite que se utilicen técnicas ETL, es decir, poder implementar procesos de extracción, transformación y carga de datos. Kettle, además, ofrece datos analíticos muy precisos, eliminando las complejidades involucradas en la codificación al proporcionar bibliotecas en profundidad para el mismo.

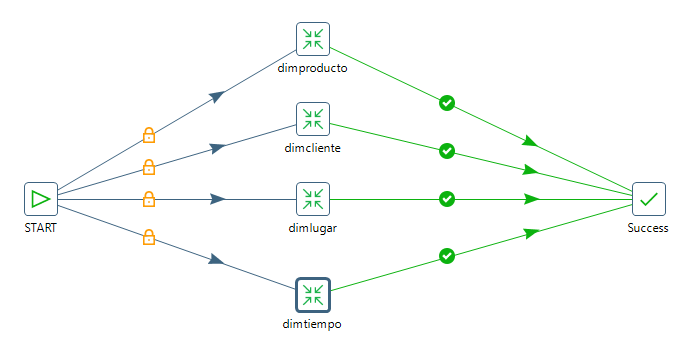
Esta herramienta cuenta con módulos encargados de gestionar la entrada, salida, transformaciones, búsquedas, etc. Como se aprecia en la siguiente figura.



* 1. Cargar dimensiones



El proceso se detallará en el punto V.6 de este documento.



**Primera dimensión: producto**

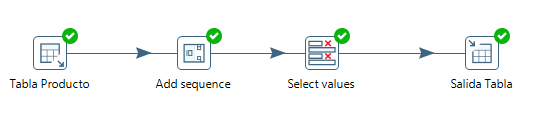
Para la carga de esta dimensión:

1. Se extrae la tabla “producto” de la base de datos “Devoluciones” como entrada.

2. Se agrega un identificador único extra llamado ‘id’

3. Se seleccionan las columnas importantes para la dimensión (nombre, precio, categoría)

4. Se carga a la tabla “dimproducto” los datos transformados de la entrada.



**Segunda dimensión: cliente**

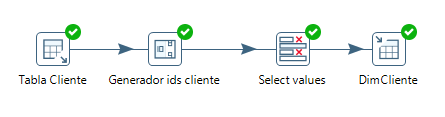
Para la carga de esta dimensión:

1. Se extrae la tabla “cliente” de la base de datos “Devoluciones” como entrada.

2. Se agrega un identificador único extra llamado ‘id’

3. Se seleccionan las columnas importantes para la dimensión (cedula, nombre, apellido, email, teléfono, genero)..

4. Se carga a la tabla “dimcliente” los datos transformados de la entrada.



**Tercera dimensión: lugar**

Para la carga de esta dimensión:

1. Se extrae la tabla “lugar” de la base de datos “Devoluciones” como entrada.

2. Se agrega un identificador único extra llamado ‘id’

3. Se seleccionan las columnas importantes para la dimensión (ciudad,dirección).

4. Se carga a la tabla “dimlugar” los datos transformados de la entrada.



**Cuarta dimensión: tiempo**

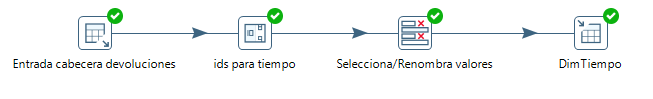
Para la carga de esta dimensión:

1. Se extrae la tabla “cabeceradevolucion” de la base de datos “Devoluciones” como entrada.

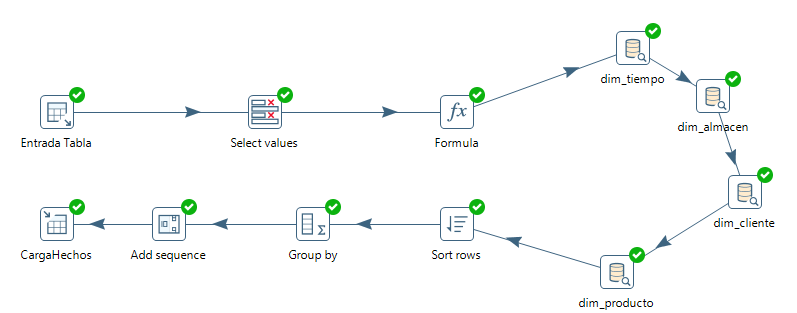
2. Se agrega un identificador único extra llamado ‘id’

3. Se seleccionan las columnas importantes para la dimensión (fecha, mes).

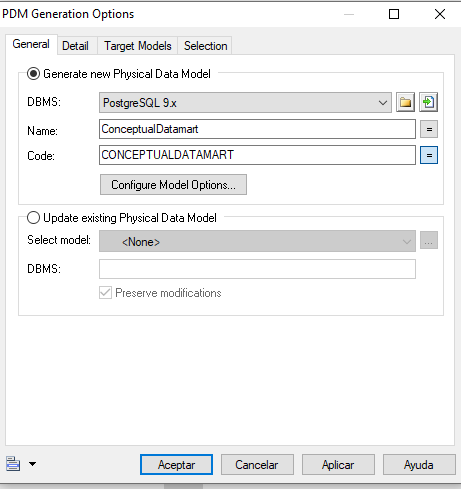
4. Se carga a la tabla “dimtiempo” los datos transformados de la entrada.



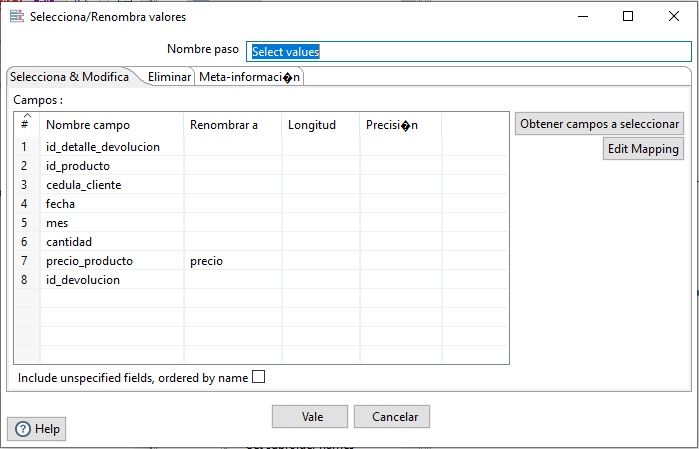
* 1. Cargar tabla de hechos



* 1. Carga dimensiones y tabla de hechos incrementales.
  2. Documentación de procesos



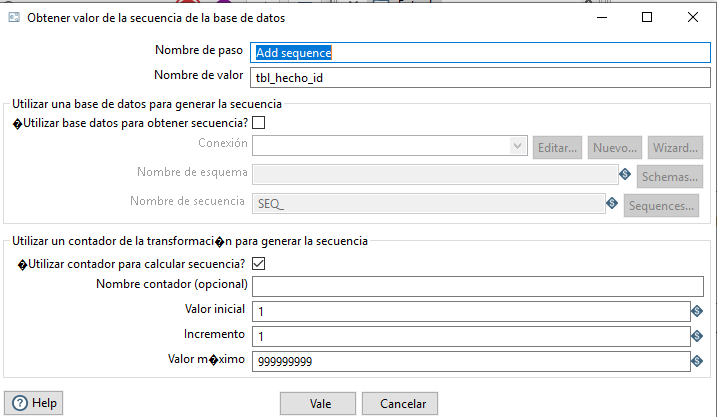
En la siguiente figura se aprecia como se filtra una gran cantidad de campos y se renombre ‘precio\_producto’ a ‘precio’.



3 Añadir secuencia de valores

Dado que al trabajar con diversas fuentes muchas pueden tener identificadores repetidos, es necesario crear una secuencia nueva para etiquetar a los datos que ingresan a las dimensiones y a los hechos

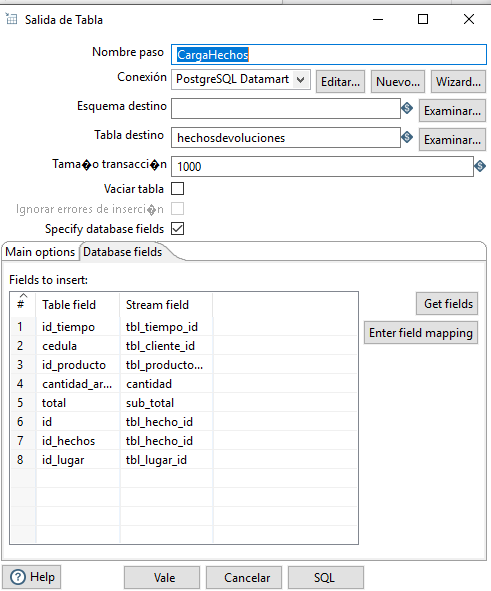
Por ejemplo, en la siguiente figura se aprecia una inserción de identificadores desde el número 1 hasta 999999999. La secuencia deberá ser adecuada a la cantidad de tuplas que se vaya a insertar.



4 Salida de datos

En la salida de datos se apunta hacia el destino que en este caso será la tabla de hechos del Datamart.

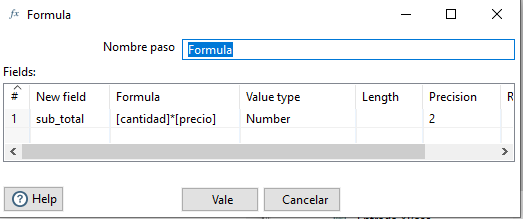
Es importante recalcar como los nombres de los campos entre la entrada y la salida difieren a pesar de referirse al mismo ‘tipo’ de información.



5 Fórmula

La fórmula sirve para incluir una nueva columna en base a los datos que se reciba para mejorar el análisis de datos.

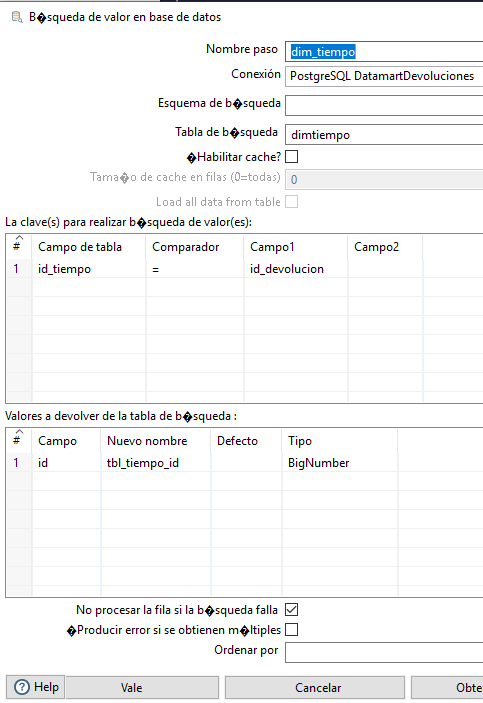
Para este trabajo se utiliza la fórmula de cantidad multiplicado por precio de producto para generar una nueva columna con el subtotal de la devolución para en un futuro averiguar cuanto dinero le ha costado a la empresa esa devolución.



6 Búsqueda de valor en la base de datos

Este proceso ayuda a comparar identificadores únicos entre las dimensiones y la nueva consulta realizada a la fuente de datos de forma que la integridad y congruencia de datos se mantenga en la relación de hechos.

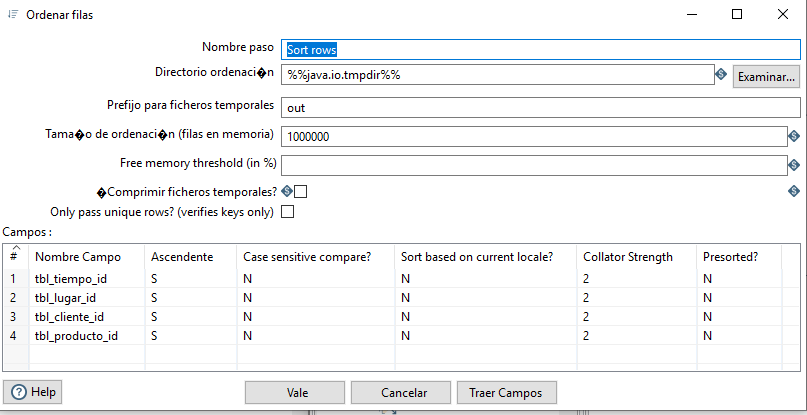
La siguiente figura ilustra el procedimiento de comparación y la reasignación de nombre para el identificador que se incluirá en la tabla de hechos.



7 Ordenar filas

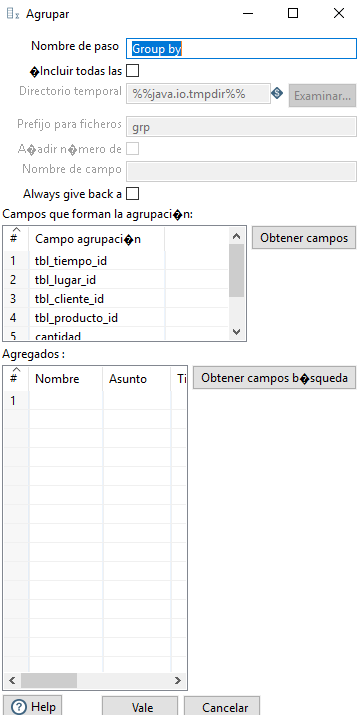
Como su nombre lo indica, permite realizar un ordenamiento de forma ascendente para los identificadores de forma que la información se vaya organizando en el tiempo.

En el trabajo se ordena con la siguiente jerarquía: tiempo, lugar, cliente y finalmente producto.



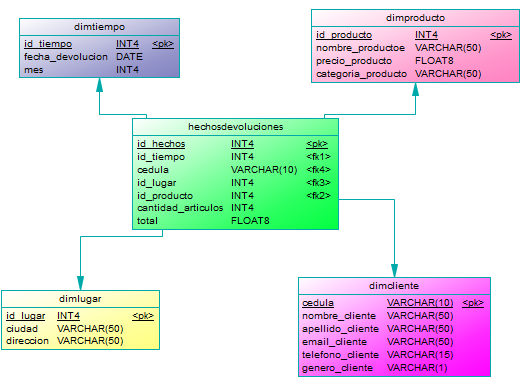
8 Agrupación

Esta operación se debe utilizar siempre que se realice agregaciones sobre un mismo campo de forma que la información no se replique innecesariamente, tiene una semejanza con el group by del lenguaje de consultas SQL.



1. **diseño fisico**
   1. Trasladar el modelo dimensional al modelo físico. Aplicando nombres y estándares de la base de datos, identificar el tipo de datos y la longitud de las columnas, etiquetar atributos como Nulos y No nulos.

El siguiente modelo corresponde a uno a exportarse para PostgreSQL, con sus tipos de datos primitivos.



* 1. Determinar la longitud de la BD del DWH
  2. Determinar la estrategia de indexación.
  3. Sentencias SQL para crear el modelo

/\*==============================================================\*/

/\* Table: DIMCLIENTE \*/

/\*==============================================================\*/

create table DIMCLIENTE (

CEDULA VARCHAR(10) not null,

NOMBRE\_CLIENTE VARCHAR(50) not null,

APELLIDO\_CLIENTE VARCHAR(50) null,

EMAIL\_CLIENTE VARCHAR(50) not null,

TELEFONO\_CLIENTE VARCHAR(15) not null,

GENERO\_CLIENTE VARCHAR(1) not null,

constraint PK\_DIMCLIENTE primary key (CEDULA)

);

/\*==============================================================\*/

/\* Index: DIMCLIENTE\_PK \*/

/\*==============================================================\*/

create unique index DIMCLIENTE\_PK on DIMCLIENTE (

CEDULA

);

/\*==============================================================\*/

/\* Table: DIMLUGAR \*/

/\*==============================================================\*/

create table DIMLUGAR (

ID\_LUGAR INT4 not null,

CIUDAD VARCHAR(50) not null,

DIRECCION VARCHAR(50) null,

constraint PK\_DIMLUGAR primary key (ID\_LUGAR)

);

/\*==============================================================\*/

/\* Index: DIMLUGAR\_PK \*/

/\*==============================================================\*/

create unique index DIMLUGAR\_PK on DIMLUGAR (

ID\_LUGAR

);

/\*==============================================================\*/

/\* Table: DIMPRODUCTO \*/

/\*==============================================================\*/

create table DIMPRODUCTO (

ID\_PRODUCTO INT4 not null,

NOMBRE\_PRODUCTOE VARCHAR(50) not null,

PRECIO\_PRODUCTO FLOAT8 not null,

CATEGORIA\_PRODUCTO VARCHAR(50) null,

constraint PK\_DIMPRODUCTO primary key (ID\_PRODUCTO)

);

/\*==============================================================\*/

/\* Index: DIMPRODUCTO\_PK \*/

/\*==============================================================\*/

create unique index DIMPRODUCTO\_PK on DIMPRODUCTO (

ID\_PRODUCTO

);

/\*==============================================================\*/

/\* Table: DIMTIEMPO \*/

/\*==============================================================\*/

create table DIMTIEMPO (

ID\_TIEMPO INT4 not null,

FECHA\_DEVOLUCION DATE not null,

MES INT4 null,

constraint PK\_DIMTIEMPO primary key (ID\_TIEMPO)

);

/\*==============================================================\*/

/\* Index: DIMTIEMPO\_PK \*/

/\*==============================================================\*/

create unique index DIMTIEMPO\_PK on DIMTIEMPO (

ID\_TIEMPO

);

/\*==============================================================\*/

/\* Table: HECHOSDEVOLUCIONES \*/

/\*==============================================================\*/

create table HECHOSDEVOLUCIONES (

ID\_HECHOS INT4 not null,

ID\_TIEMPO INT4 not null,

CEDULA VARCHAR(10) not null,

ID\_LUGAR INT4 not null,

ID\_PRODUCTO INT4 not null,

CANTIDAD\_ARTICULOS INT4 not null,

TOTAL FLOAT8 not null,

constraint PK\_HECHOSDEVOLUCIONES primary key (ID\_HECHOS)

);

/\*==============================================================\*/

/\* Index: HECHOSDEVOLUCIONES\_PK \*/

/\*==============================================================\*/

create unique index HECHOSDEVOLUCIONES\_PK on HECHOSDEVOLUCIONES (

ID\_HECHOS

);

/\*==============================================================\*/

/\* Index: TIEMPODEVOLUCION\_FK \*/

/\*==============================================================\*/

create index TIEMPODEVOLUCION\_FK on HECHOSDEVOLUCIONES (

ID\_TIEMPO

);

/\*==============================================================\*/

/\* Index: PRODUCTODEVOLUCION\_FK \*/

/\*==============================================================\*/

create index PRODUCTODEVOLUCION\_FK on HECHOSDEVOLUCIONES (

ID\_PRODUCTO

);

/\*==============================================================\*/

/\* Index: LUGARDEVOLUCION\_FK \*/

/\*==============================================================\*/

create index LUGARDEVOLUCION\_FK on HECHOSDEVOLUCIONES (

ID\_LUGAR

);

/\*==============================================================\*/

/\* Index: CLIENTEDEVOLUCION\_FK \*/

/\*==============================================================\*/

create index CLIENTEDEVOLUCION\_FK on HECHOSDEVOLUCIONES (

CEDULA

);

alter table HECHOSDEVOLUCIONES

add constraint FK\_HECHOSDE\_CLIENTEDE\_DIMCLIEN foreign key (CEDULA)

references DIMCLIENTE (CEDULA)

on delete restrict on update restrict;

alter table HECHOSDEVOLUCIONES

add constraint FK\_HECHOSDE\_LUGARDEVO\_DIMLUGAR foreign key (ID\_LUGAR)

references DIMLUGAR (ID\_LUGAR)

on delete restrict on update restrict;

alter table HECHOSDEVOLUCIONES

add constraint FK\_HECHOSDE\_PRODUCTOD\_DIMPRODU foreign key (ID\_PRODUCTO)

references DIMPRODUCTO (ID\_PRODUCTO)

on delete restrict on update restrict;

alter table HECHOSDEVOLUCIONES

add constraint FK\_HECHOSDE\_TIEMPODEV\_DIMTIEMP foreign key (ID\_TIEMPO)

references DIMTIEMPO (ID\_TIEMPO)

on delete restrict on update restrict;

1. **Aplicaciones para usuarios finales: características**
   1. Descripción de la herramienta para el procesamiento analítico.
   2. Listar los resúmenes de información requeridos por los usuarios.
   3. Determinar los cubos.
   4. Diseñar los reportes a mostrar.
   5. Implementación
      1. Crear las tablas dinámicas.
      2. Construir los gráficos dinámicos.
      3. Crear interface para Internet Explorer.

**VII. Referencias**

**VIII. ANEXOS**



1. **Determinar la longitud de la BD del DWH**

La longitud para la base de datos del Datawarehouse se compone de la longitud de las dimensiones y de la tabla de hechos.

Para la dimensión de tiempo:

Id\_tiempo               -> Integer -> 4 bytes

Fecha\_devolución -> Date -> 8 bytes

Mes                        -> Integer -> 4 bytes

Fecha\_compra      -> Date -> 8 bytes

Para la dimensión de producto:

id\_producto            -> Integer -> 4 bytes

Nombre\_producto  -> varchar(50)-> 50 bytes

precio\_producto     -> float-> 4 bytes

categoria\_producto-> varchar(50) -> 50 bytes

Para la dimensión de almacen (lugar):

id\_lugar    -> Integer -> 4 bytes

Ciudad     -> varchar(50)-> 50 bytes

dirección  -> varchar(50) -> 50 bytes

Para la dimensión de cliente:

Cedula                -> varchar(10)-> 10 bytes

Nombre\_cliente  -> varchar(50)-> 50 bytes

apellido\_cliente   -> varchar(50)-> 50 bytes

Email\_cliente      -> varchar(50) -> 50 bytes

telefono\_cliente   -> varchar(15) -> 15 bytes

genero\_cliente     -> varchar(1) -> 1 bytes

Finalmente, para Hechos de devoluciones:

Identificadores de las dimensiones -> Integer \* 4 -> 16 bytes

Id\_hechos               -> Integer -> 4 bytes

Cantidad\_articulos -> Integer -> 4 bytes

Total\_dinero            -> Float -> 4 bytes

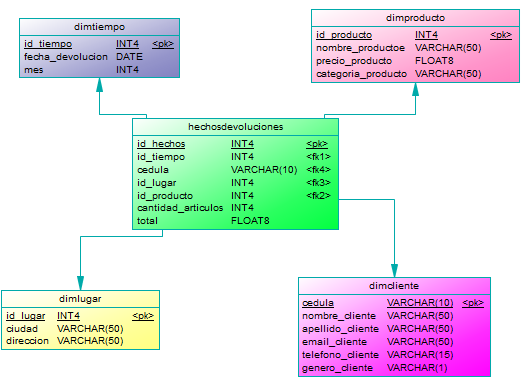
Dias\_entre\_compra\_devolución -> Float -> 4 bytes

**En total la longitud para una inserción con todas sus relaciones es de 444 bytes.**

De forma que para insertar un millon de datos se requerirá 0,4135 GB aproximadamente. Para una base de datos estándar hosteada de 25GB con precio de 10$ al més, se podrán almacenar mas de 60 millones de registros hasta requerir una nueva expansión.

1. **Trasladar el modelo dimensional al modelo físico. Aplicando nombres y estándares de la base de datos, identificar el tipo de datos y la longitud de las columnas, etiquetar atributos como nulos y no nulos.**

El siguiente modelo corresponde a uno a exportarse para PostgreSQL, con sus tipos de datos primitivos.



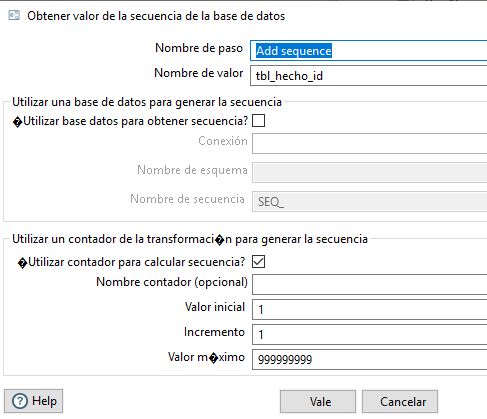
Al pasar del modelo conceptual al físico todas las relaciones de 1:N trasladan su llave primaria hacia la tabla de ‘N’ transformándolas en llaves foráneas.

Para este trabajo se consideró la cantidad de artículos y el total de dinero perdido por devolución en un tiempo. Indicadores de crecimiento como capacitación a los empleados para manejar el tema de las devoluciones se omite de forma que la tabla de hechos tenga información suficiente para que permita un análisis profundo de datos con alta escalabilidad.

En el diagrama del modelo conceptual se ha establecido los que son mandatorios ‘<M>’ que en este caso son todos, al transformarlo al modelo de PostgreSQL estas características se han puesto ocultas.

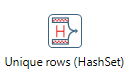
1. **Carga dimensiones y tabla de hechos incrementales.**

Para una carga de dimensiones de forma incremental es importante utilizar la transformación ‘Add secuence’ de forma que desde el propio dominio del administrador se agreguen los identificadores únicos con una secuencia lineal.



Utilizar una columna con valor auto incremental es también una excelente opción siempre y cuando la base de datos utilizada permita la desactivación del chequeo de ‘identity’, dado que para alguans transformaciones puede ser útil manipular esos identificadores únicos.

De todas formas se aplica también una operación de valores únicos



Con el fín de salvaguardar los valores únicos y no sobreponer a la tabla.

Para la carga continua se puede realizar de forma manual como se ha hecho en este trabajo, es decir realizar la consulta a la base de datos que actúa como fuente cada X tiempo. Se considera ese ‘X’ como un valor de 1mes a 6 meses dependiendo que tanta inserción de datos ocurra en las fuentes, para no sobre cargar al Datamart en una fecha determinada se recomienda cargar mensualmente y aprovechar esa granularidad.

Algo que no se hizo, pero queda para futuros trabajos es automatizar la carga utilizando ‘scripts’ escritos en PHP que se ejecuten a cierta hora del mes, para que empiecen a ordenar los datos pero bloqueando la lectura de forma que se conseve la integridad de los datos.