**PROYECTO DATA WAREHOUSE**

**DATA WAREHOUSE**

**ANDRES SANCHEZ**

**LUIS FELIPE VELASCO TAO  
JUAN DAVID GONZALEZ**

**UNIVERSIDAD DE SAN BUENAVENTURA**

**BOGOTÁ**

**11 DE AGOSTO**

**2021**

Contenido

[Bona Health EPS 5](#_Toc87196081)

[Contexto de la Bona Health EPS 5](#_Toc87196082)

[Misión 5](#_Toc87196083)

[Visión 5](#_Toc87196084)

[KPI 5](#_Toc87196085)

[Requerimientos de negocio 9](#_Toc87196086)

[Fuentes elegidas 10](#_Toc87196087)

[Mortalidad VIH 2010 a 2016 10](#_Toc87196088)

[Mortalidad cáncer de pulmón 2010 a 2016 11](#_Toc87196089)

[Mortalidad cáncer de mama 2010 a 2016 12](#_Toc87196090)

[Departamentos y municipios de Colombia 20](#_Toc87196091)

[Modelo Entidad Relación 22](#_Toc87196092)

[DISEÑO CONCEPTUAL 24](#_Toc87196093)

[Diseño de Dimensiones y Medidas 24](#_Toc87196094)

[Modelo CMDM: relaciones dimensionales 25](#_Toc87196095)

[Modelo CMDM - Defunciones por fecha 25](#_Toc87196096)

[Modelo CMDM – Asistencia médica 25](#_Toc87196097)

[Modelo CMDM – Defunciones por municipio 26](#_Toc87196098)

[Modelo CMDM – Muertes 27](#_Toc87196099)

[Modelo Multidimensional 27](#_Toc87196100)

[Modelos DF 28](#_Toc87196101)

[Modelo DF – Defunciones por fecha 28](#_Toc87196102)

[Modelo DF – Asistencia médica 29](#_Toc87196103)

[Modelo DF – Defunciones por municipio 29](#_Toc87196104)

[Modelo DF – Muertes 30](#_Toc87196105)

[DISEÑO LOGICO 31](#_Toc87196106)

[Modelo ROLAP 31](#_Toc87196107)

[DDL del Data Ware House 32](#_Toc87196108)

[ETL 35](#_Toc87196109)

[Diseño de ETL 35](#_Toc87196110)

[Secuencia de ETL 37](#_Toc87196111)

[Modelo relacional ODS 39](#_Toc87196112)

[Programa ETL 45](#_Toc87196113)

[Muestra de datos del Data Warehouse 56](#_Toc87196114)

[Fact\_asistencia\_medica 57](#_Toc87196115)

[Fact\_defunciones\_fecha 57](#_Toc87196116)

[Fact\_defunciones\_municipio 58](#_Toc87196117)

[Fact\_muertes 58](#_Toc87196118)

[CREACIÓN Y PUBLICACIÓN DEL CUBO 59](#_Toc87196119)

[Entorno de trabajo 59](#_Toc87196120)

[Pentaho Server 59](#_Toc87196121)

[Pentaho Schema Workbench 63](#_Toc87196122)

[Pentaho Report Desginer 64](#_Toc87196123)

[Creación del cubo 66](#_Toc87196124)

[Dim\_fecha 69](#_Toc87196125)

[Dim\_Municipio 73](#_Toc87196126)

[Dim\_Enfermedad 76](#_Toc87196127)

[Ejemplo de creacion de una medida 78](#_Toc87196128)

[Asistencia\_medica 80](#_Toc87196129)

[Defun\_por\_municipio 80](#_Toc87196130)

[Defunciones\_fecha 81](#_Toc87196131)

[Prueba 82](#_Toc87196132)

[Publicación del cubo 84](#_Toc87196133)

[Dashboard del cubo en Pentaho Server 86](#_Toc87196134)

[Conexión al Data Warehouse 86](#_Toc87196135)

[Configuración del Schema 90](#_Toc87196136)

[Despliegue del Dashboard 91](#_Toc87196137)

[Creación de reportes 93](#_Toc87196138)

[Reporte 1 93](#_Toc87196139)

[Reporte 2 93](#_Toc87196140)

[Reporte 3 93](#_Toc87196141)

[Tabla de ilustraciones 94](#_Toc87196142)

# Bona Health EPS

A continuación, se van a definir los elementos corporativos de la empresa y se definirán las fuentes de datos elegidas para el desarrollo de la actividad.

## Contexto de la Bona Health EPS

Bona Health EPS es una entidad promotora de salud colombiana la cual tienen presencia a nivel nacional, principalmente en las ciudades capitales, en las cueles nuestros afiliados acceden a los servicios de salud prestados por medio de las IPS asociadas, entre las cuales se tienen hospitales, centro de optometría, odontología, radiografía y de atención para distintas afecciones medicas de los colombianos. Nuestra empresa nació en el 2015, teniendo como punto de partida la ciudad de Bogotá, pasando año tras año a tener presencia en Cali, Medellín, Cartagena, Bucaramanga y demás ciudades principales del país. En la actualidad, debido a la pandemia, Bona Health EPS ha tenido inconvenientes con los procesos de afiliación y especialmente con el manejo de citas y tratamientos de nuestros afiliados con enfermedades como VIH, cáncer y demás problemas de salud, llegando hasta el limite de notar cierto aumento en la mortalidad de muchos afiliados con ciertas enfermedades.

## Misión

Asegurarles a los colombianos desde el primer día que se encuentren afiliados el acceso a servicios de salud de calidad, a tiempo y siempre buscando su bienestar.

## Visión

Para el 2030, Bona Health EPS espera ser una EPS que tenga una cobertura del 80% en municipios apartados en el país con el fin de acercar a más a los pueblos y a sus habitantes servicios de salud de calidad, sea medio del régimen contributivo o subsidiado.

## KPI

Con el fin de proveer un servicio que sea capaz de atender a su debido tiempo las necesidades de los colombianos, es necesario determinar la tasa de mortalidad de determinados padecimientos en un intervalo de tiempo, y la frecuencia de defunciones en un lugar con respecto a dicho padecimiento.

Es por esto por lo que se establecieron los siguientes indicadores para determinar el tipo de servicio que debe ofrecerse en los departamentos más afectados por un determinado padecimiento.

1. **Gráfico, Histograma

   Descripción generada automáticamenteGráfico, Gráfico de barras

   Descripción generada automáticamenteGráfico, Gráfico de barras

   Descripción generada automáticamenteTasa de defunciones por VIH, cáncer de mama y cáncer de pulmón**. A continuación, se muestran los datos de las defunciones presentadas en el periodo del 2010 a 2016 con el fin de definir medidas para la prevención y atención oportuna de dichas enfermedades.

Ilustración 1 Defunciones por VIH

Ilustración 2 Defunciones por cáncer de mama

Ilustración 3 Defunciones por cáncer de pulmón

1. **Tasa de atención médica:** en las defunciones registradas. Se quiere definir si los fallecidos tuvieron o no atención medica en el momento, de modo tal se pueda definir acciones que mejoren la atención medica en futuros casos.

Gráfico, Gráfico circular

Descripción generada automáticamente

Gráfico, Gráfico circular

Descripción generada automáticamenteIlustración 4 Asistencia medida prestada a los fallecidos por VIH

Ilustración 5 Atención medica prestada a los fallecidos por cáncer de mama

Gráfico, Gráfico circular

Descripción generada automáticamente

Ilustración 6 Atención medica prestada a los fallecidos por cáncer de pulmón

1. **Gráfico

   Descripción generada automáticamenteDefunciones por municipio**: con el fin definir medidas y mecanismos de acción, se quiere medir cuantas personas fallecieron en cada uno de los municipios registrados.

Ilustración 7 defunciones por VIH clasificadas por municipio

Gráfico

Descripción generada automáticamente

Ilustración 8 defunciones por cáncer de pulmón clasificadas por municipio

Imagen que contiene Tabla

Descripción generada automáticamente

Ilustración 9 defunciones por cáncer de mama clasificadas por municipio

## Requerimientos de negocio

Bona Health EPS busca ofrecer un servicio accesible que cubra las zonas con mayor tasa de mortalidad por padecimientos clínicos en el país, tales como el VIH, el cáncer de Mama y el cáncer de pulmón. Con el fin de realizar estos objetivos, se plantean los siguientes requerimientos.

1. Analizar la tasa de mortalidad por padecimiento clínico con respecto a los departamentos del país para proveer cobertura en los municipios en dónde se requieran los servicios de la entidad.
2. Establecer un histórico de defunciones por padecimientos clínicos en cada departamento para la asignación del personal adecuado en las instalaciones que tengan cobertura de la entidad.
3. Determinar el número de difuntos que recibieron atención médica en una institución clínica con el fin de poder asignar los recursos correspondientes a las instituciones cubiertas por la entidad.

## Fuentes elegidas

Para la correcta toma de decisiones con relación a la mejora en la prestación de servicios de salud que prevengan la mortalidad de los afiliados con enfermedades criticas como el VIH, el cáncer de mama y de pulmón se seleccionaron las siguientes fuentes de datos, las cuales son los reportes emitidos de defunción relacionados con los afiliados con estas enfermedades, además de requerir de cierta información para definir apropiadamente el lugar (departamento y municipio) donde falleció la persona:

### Mortalidad VIH 2010 a 2016

Este dataset es suministrado por Observatorio Social de Salud Pública en el cual se entregan datos relacionados con las defunciones de las personas con VIH en el periodo del 2010 a 2016, en el cual se pueden ver los datos relacionados con el lugar en donde falleció la persona, como lo es municipio, departamento institución en la que falleció, también se presentan datos básicos del fallecido como su sexo y condición socioeconómica y finalmente se define una descripción de la condición médica de la persona al fallecer, incluyendo sus antecedentes de salud y datos relacionados con el estado de salud de la persona.

**Fecha de creación**: 21 de abril de 2017

**Sector**: Salud y protección social

**Cantidad de registros**:445

**Cantidad de columnas**: 105

Tabla

Descripción generada automáticamente

Ilustración 10 Muestra de datos del dataset de mortalidad por VIH

**Fuente**: [Mortalidad VIH 2010 A 2016 | Datos Abiertos Colombia](https://www.datos.gov.co/Salud-y-Protecci-n-Social/Mortalidad-VIH-2010-A-2016/yht4-twf4/data)

### Mortalidad cáncer de pulmón 2010 a 2016

Este dataset es suministrado por Observatorio Social de Salud Pública en el cual se entregan datos relacionados con las defunciones de las personas con cáncer de pulmón en el periodo del 2010 a 2016, en el cual se pueden ver los datos relacionados con el lugar en donde falleció la persona, como lo es municipio, departamento institución en la que falleció, también se presentan datos básicos del fallecido como su sexo y condición socioeconómica y finalmente se define una descripción de la condición médica de la persona al fallecer, incluyendo sus antecedentes de salud y datos relacionados con el estado de salud de la persona.

**Fecha de creación**: 20 de abril de 2017

**Sector**: Salud y protección social

**Cantidad de registros**:1025

**Cantidad de columnas**: 105

Tabla

Descripción generada automáticamente

Ilustración 11 Muestra de datos del dataset de mortalidad por cáncer del pulmón

**Fuente**: [Mortalidad Cáncer de Pulmón 2010 A 2016 | Datos Abiertos Colombia](https://www.datos.gov.co/Salud-y-Protecci-n-Social/Mortalidad-C-ncer-de-Pulm-n-2010-A-2016/qfuz-6igc)

### Mortalidad cáncer de mama 2010 a 2016

Este dataset es suministrado por Observatorio Social de Salud Pública en el cual se entregan datos relacionados con las defunciones de las personas con cáncer de mama en el periodo del 2010 a 2016, en el cual se pueden ver los datos relacionados con el lugar en donde falleció la persona, como lo es municipio, departamento institución en la que falleció, también se presentan datos básicos del fallecido como su sexo y condición socioeconómica y finalmente se define una descripción de la condición médica de la persona al fallecer, incluyendo sus antecedentes de salud y datos relacionados con el estado de salud de la persona.

**Fecha de creación**: 20 de abril de 2017

**Sector**: Salud y protección social

**Cantidad de registros**:503

**Cantidad de columnas**: 105

Tabla

Descripción generada automáticamente

Ilustración 12 Muestra de datos del dataset de mortalidad por cáncer de mama

**Fuente**: [Mortalidad Cáncer Mama 2010 A 2016 | Datos Abiertos Colombia](https://www.datos.gov.co/Salud-y-Protecci-n-Social/Mortalidad-C-ncer-Mama-2010-A-2016/r39m-rntv)

Debido a que los datasets se encuentran estructurados bajo las mismas columnas, se realizó un análisis de las columnas de los datasets y se brinda una descripción de los siguientes datos relevantes.

|  |  |  |
| --- | --- | --- |
| **Nombre de Columna** | **Descripción** | **Tipo** |
| **COD\_DPTO** | Identificador del departamento donde falleció la persona | Número |
| **COD\_MUNIC** | Identificador del municipio donde falleció la persona. Este valor se puede repetir, ya que, dos o más municipios pueden tener el mismo código, la diferencia es el código del departamento | Número |
| **A\_DEFUN** | Área del municipio donde falleció la persona (rural, cabecera municipal, etc.) | Texto simple |
| **SIT\_DEFUN** | Sitio en donde la persona falleció, como su casa o un hospital/clínica | Texto simple |
| **COD\_INST** | Identificador de la institución prestadora de servicios de salud donde falleció la persona. (Si la persona fallece en su casa, este campo se encuentra vacío) | Número |
| **NOM\_INST** | Nombre de la institución prestadora de servicios de salud en donde la persona halla fallecido. | Texto simple |
| **FECHA\_DEF** | Fecha y hora en la cual falleció la persona, expresada de la siguiente forma: 2012 Sep 14 12:00:00 AM | Fecha y hora |
| **ANO** | Año en el que falleció la persona | Número |
| **MES** | Mes en el que falleció la persona | Número |
| **HORA** | Hora en la que falleció la persona | Número |
| **MINUTOS** | Minuto en que falleció la persona | Número |
| **SEXO** | Cadena de caracteres con el sexo de la persona (Masculino/Femenino) | Texto simple |
| **FECHA\_NAC** | Fecha (y hora la cual no es un dato relevante) de nacimiento del fallecido | Fecha y hora |
| **EST\_CIVIL** | Estado civil del fallecido | Texto simple |
| **EDAD** | Edad del fallecido | Número |
| **NIVEL\_EDU** | Nivel educativo del fallecido | Texto simple |
| **MUERTEPORO** | Campo de texto en donde se define si la persona murió o no por omisión | Texto simple |
| **OCUPACION** | Ocupación o profesión del fallecido. | Texto simple |
| **CODPTORE** | Nombre del departamento donde residía el fallecido | Texto simple |
| **CODMUNRE** | Nombre del municipio donde residía el fallecido | Texto simple |
| **AREA\_RES** | Área del municipio donde residía la persona (rural, cabecera municipal, etc.) | Texto simple |
| **BARRIOFAL** | Nombre del barrio donde residía el fallecido | Texto simple |
| **SEG\_SOCIAL** | Tipo de seguridad social por medio de la cual se le presto los servicios de salud al fallecido. | Texto simple |
| **IDADMISALU** | Identificador de la EPS (este dato se obvia para la naturaleza del ejercicio) | Número |
| **IDCLASADMI** | EPS a la cual el fallecido estaba afiliado (este dato se obvia para la naturaleza del ejercicio) | Texto simple |
| **PMAN\_MUER** | Tipo de muerte | Texto simple |
| **MU\_PARTO** | Campo en donde se define si la muerte fue en un parto o no | Texto simple |
| **T\_PARTO** | Si la muerte fue por un parto, se define el tipo de parto. Se definen otros datos relacionados con muertes en partos | Texto simple |
| **ASIS\_MED** | Campo en donde se define si la persona tuvo asistencia médica en el momento de su muerte | Texto simple |
| Los siguientes datos son descripción de la condición médica del fallecido | | |
| **N\_DIR1**  **C\_DIR1**  **C\_DIR12**  **N\_ANT1**  **C\_ANT1**  **C\_ANT12**  **N\_ANT2**  **C\_ANT2**  **C\_ANT22**  **N\_ANT3**  **C\_ANT3**  **C\_ANT32**  **N\_PAT1**  **C\_PAT1**  **N\_PAT2**  **C\_PAT2** | Datos sobre antecedentes, signos vitales y condición médica del fallecido | Texto simple |
| **N\_BAS1** | Enfermedad de base del fallecido | Texto simple |

**TOTAL, DE REGISTROS A MANIPULAR**: 1975

Con base al análisis a cada una de las fuentes de datos, la descripción brindada a las columnas de interés y la definición de los requerimientos y KPIS, se definirá un único modelo entidad relación en el cual se puedan estructurar los datos vistos.

### Departamentos y municipios de Colombia

Este dataset es suministrado por el Departamento Administrativo Nacional de Estadística, por sus siglas, DANE de entregan los datos de codificación o identificadores de los municipios y departamentos del país, los cuales son empleados por las entidades publicas y privadas en sus procesos de registro de actividades departamentales.

**Fecha de creación**: 11 de mayo de 2017

**Sector**: estadísticas

**Cantidad de registros**:1123

**Cantidad de columnas**: 5



Ilustración 13 Muestra de datos del dataset de departamentos y municipios de Colombia

**Fuente**: [Departamentos y municipios de Colombia](https://www.datos.gov.co/Mapas-Nacionales/Departamentos-y-municipios-de-Colombia/xdk5-pm3f)

A continuación, se mostrara una tabla con la identificación de los datos presentados en este dataset:

|  |  |  |
| --- | --- | --- |
| **Nombre de Columna** | **Descripción** | **Tipo** |
| **REGIÓN** | Nombre de la región a la que pertenece cada municipio y departamento | Texto simple |
| **CÓDIGO DANE DEL DEPARTAMENTO** | Código identificador para cada uno de los departamentos de Colombia | Número |
| **DEPARTAMENTO** | Nombre del departamento asociado al código anterior | Texto simple |
| **CÓDIGO DANE DEL MUNICIPIO** | Código identificador para cada uno de los municipios de Colombia, conformado por el código del departamento al que pertenece y los últimos tres dígitos hacen referencia al identificador del municipio dentro del departamento | Numero |
| **MUNICIPIO** | Nombre del municipio asociado al código anterior | Texto simple |

## Modelo Entidad Relación

Debido a como las tres fuentes de datos se encuentran estructuradas bajo las mismas columnas, se obtuvo un único modelo entidad relación en el cual se definieron las siguientes tablas:

* **Lugar**: se definen los datos del lugar donde la persona falleció, y se estructura por medio de los siguientes datos:
  + Id\_lugar: dato identificador del lugar\*
  + Tipo\_lugar: se define si la persona murió o no en un hospital o clínica.
  + Departamento: este dato se obtendrá de la tabla departamento, y se definiría el departamento en el cual falleció la persona.
  + Municipio: este dato se obtendrá de la tabla municipio, y se definiría el municipio en el cual falleció la persona.
  + Institución: con base al tipo lugar, se define la institución obtenida de la tabla del mismo nombre.
* **Fecha**: se definen los datos del la fecha y hora registrada del fallecimiento de la persona, esto por medio de los siguientes datos:
  + Id\_fecha: fecha y hora formateada del fallecimiento de la persona.
  + Año: valor numérico del año del fallecimiento de la persona.
  + Mes: valor numérico del mes del fallecimiento de la persona.
  + Día: valor número del día del fallecimiento de la persona.
  + Hora: valor numérico de la hora de fallecimiento de la persona
  + Minutos: valor numérico del minuto de fallecimiento de la persona
* **Persona**: en esta tabla se definen los datos básicos de la persona, de modo tal se definen los siguientes datos:
  + Id\_persona: identificador de la persona.
  + Fecha\_nacimiento: fecha de nacimiento del fallecido
  + Edad: valor numero de la edad del fallecido
  + Sexo: cadena de caracteres con el sexo de la persona.
  + Seguridad\_social: cadena de caracteres en donde se define el régimen por medio del cual se le prestó atención a la persona (contributivo, subsidiado)
* **Enfermedad**: En esta tabla solo se definirá el nombre de la enfermedad de base que produjo el fallecimiento de la persona, por lo que se define los siguientes datos:
  + Nombre\_enfermedad: nombre de la enfermedad de base que genero el fallecimiento de la persona.
  + Observación: observación brindada por los médicos al momento del fallecimiento de la persona.

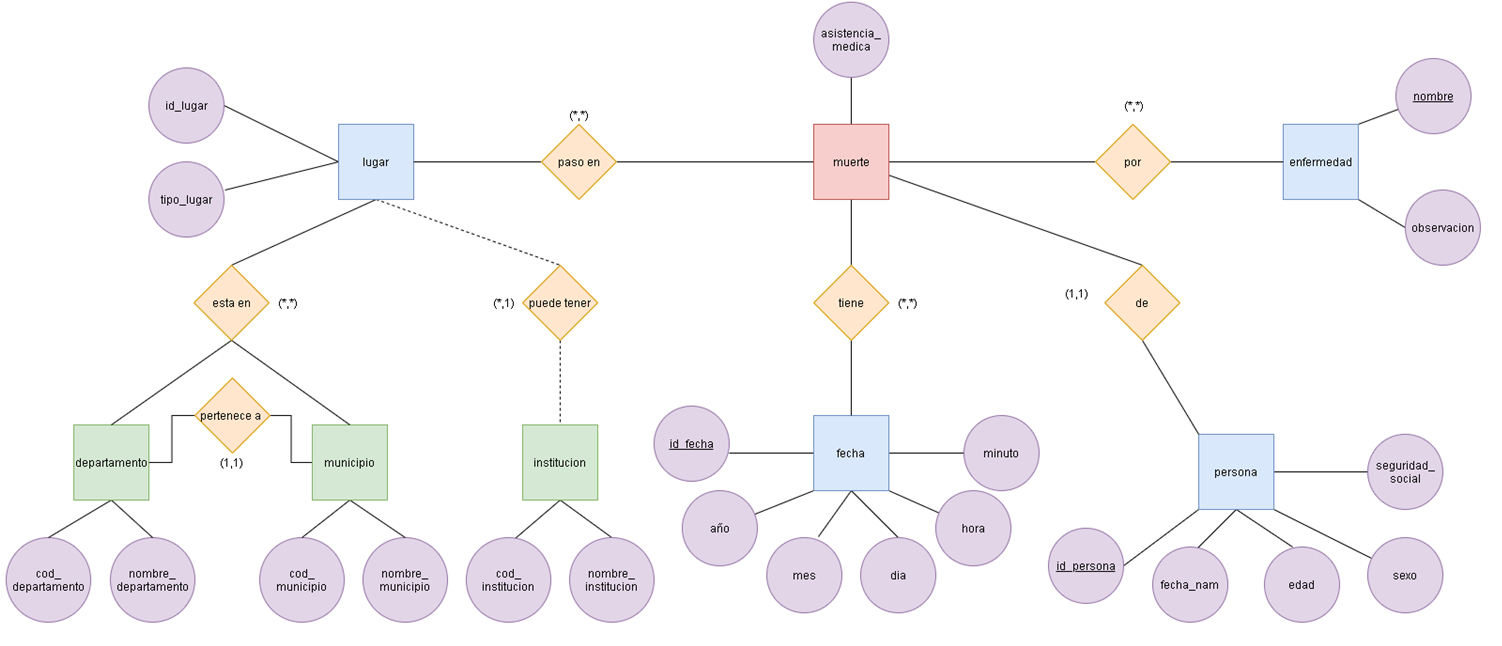
A continuación, se muestra el modelo resultante:

Ilustración 14 Modelo entidad relación de las fuentes de datos

# DISEÑO CONCEPTUAL

En esta fase se definen los elementos de dimensiones, hechos, medidas y la forma en que estas se relacionaran en el Data Warehouse, todo esto contemplando los requerimientos del negocio y las fuentes de datos a emplear en el desarrollo e implementación del almacén de datos. Para esta fase se contemplará el desarrollo de los diagramas de dimensiones y medidas, de relaciones y los modelos multidimensionales y de dimensiones y hechos.

## Diseño de Dimensiones y Medidas

Aquí se definen las dimensiones por medio de las cuales se obtendrá la información requerida por el negocio, en las cuales se definen los niveles y cierta jerarquía por medio de la cual se podrá ordenar los elementos de cada dimensión. Además, se deben definir las medidas, los datos de especial interés por parte de la empresa, por medio de los cuales se definirán más adelante las tablas de hechos.

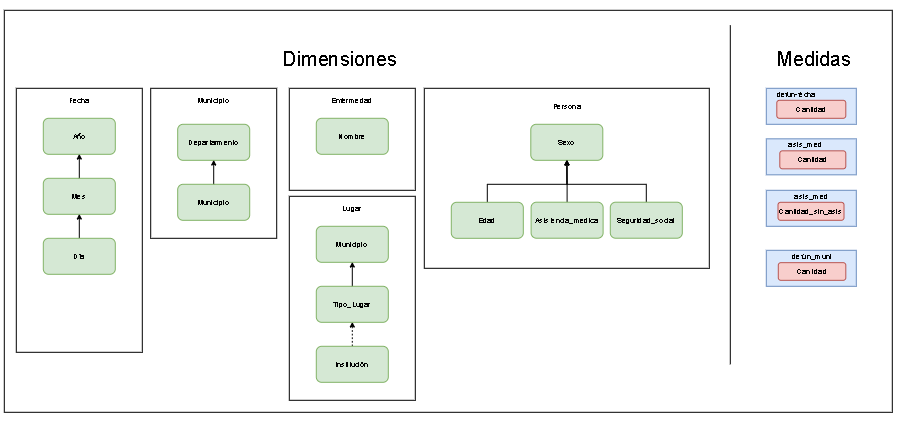


Ilustración 15 Diseño de dimensiones y medidas

Bajo los requerimientos del negocio, en la imagen anterior, se definieron las siguientes dimensiones compuestas por sus respectivas jerarquías y miembros:

* **Fecha**: Se tienen los datos para conocer la fecha de defunción de una persona, necesaria para hacer los análisis dispuestos por los KPI
* **Municipio**: con el fin de saber el municipio donde falleció cada persona se define esta dimensión con los datos de departamento y municipio, a su vez. Esta dimensión será empleada por la dimensión de lugar.
* **Lugar**: esta dimensión determina el lugar con municipio, tipo de lugar y institución en donde falleció una persona, este último dato dependerá de si la persona pudo ser o no atendida al momento de su fallecimiento.
* **Enfermedad**: Es la causa de muerte de los difuntos que contempla el sistema. Cada enfermedad tiene un **nombre** que tendrá un identificador para ser reconocida.
* **Persona**: Son los datos pertinentes al difunto, entre los que se incluyen el **sexo, la edad, su seguridad social** y si recibió **asistencia médica**.

## Modelo CMDM: relaciones dimensionales

Con base a las dimensiones y medidas previamente especificadas, se definieron los siguientes cubos por medio de las relaciones multidimensionales que le den forma a cada uno de los hechos. A continuación, se especificarán cada uno de los modelos.

### Modelo CMDM - Defunciones por fecha

Para conocer los datos con respecto a las defunciones generadas por cada enfermedad, se definió el siguiente vivo el cual usa las dimensiones de enfermedad y fecha, con el fin de saber cuántos fallecimientos se ocasionaron por cada enfermedad en cada fecha.



Ilustración 16 Modelo CMDM – Defunciones por fecha

### Modelo CMDM – Asistencia médica

Con el fin de conocer la asistencia medica brindada en cada uno de los fallecimientos registrados se tomará en cuenta las dimensiones de lugar/municipio, fecha y enfermedad, de modo tal se pueda conocer cuántos de los fallecimientos registrados de cada enfermedad en cada fecha recibieron asistencia medica y cuantos no la pudieron recibir. Esto con el fin de saber si los servicios de salud de cada lugar responder de forma óptima.

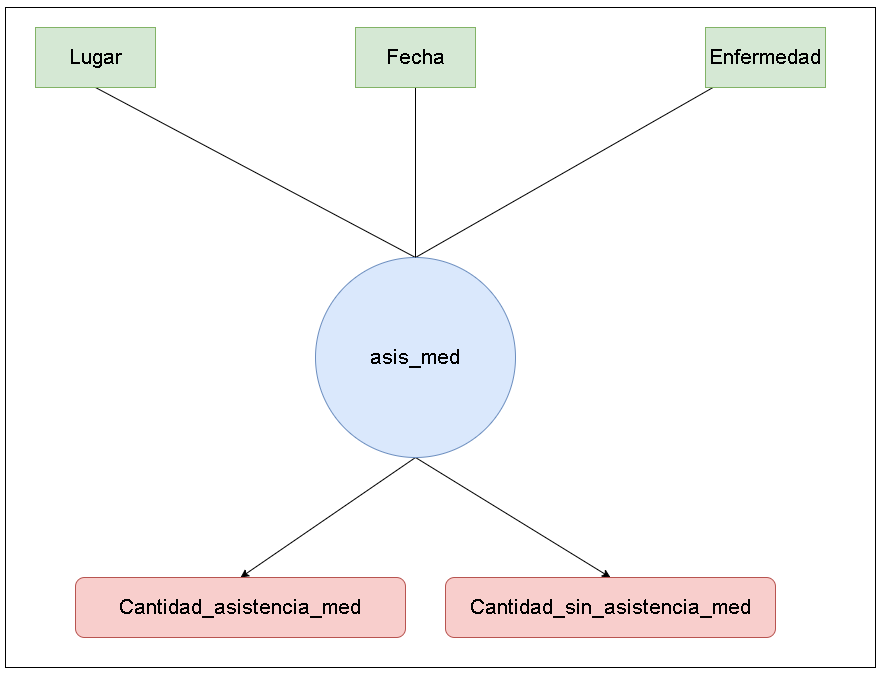


Ilustración 17 Modelo CMDM – Asistencia médica

### Modelo CMDM – Defunciones por municipio

Para tener presente los municipios con más o menos casos de defunciones de las enfermedades a trabajar en el Data Warehouse, se definen las dimensiones de enfermedad, lugar/municipio y fecha para poder conocer la cantidad de muertes registradas en cada municipio, fecha y por cada enfermedad

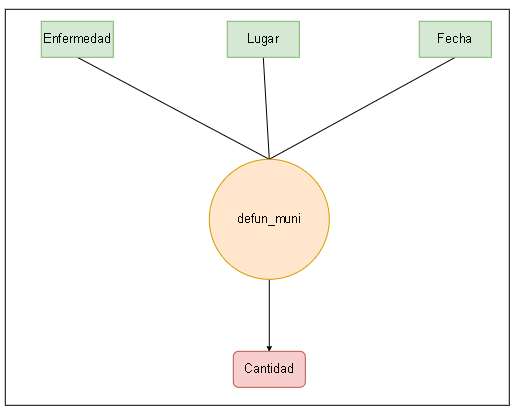


Ilustración 18 Modelo CMDM – Defunciones por municipio

### Modelo CMDM – Muertes

En este cubo se contemplarán las dimensiones de persona, lugar, fecha y enfermedad, esto con el fin de permitir el posterior análisis de los datos de la forma que se requiera. Aquí no se contará con ninguna medida.

Diagrama

Descripción generada automáticamente

Ilustración 19 Modelo CMDM – Muertes

## Modelo Multidimensional

Al tener presente cada una de las definiciones de las dimensiones involucradas en cada uno de los cubos, se realizo el siguiente modelo multidimensional el cual permite ver las dimensiones y hechos definidos.

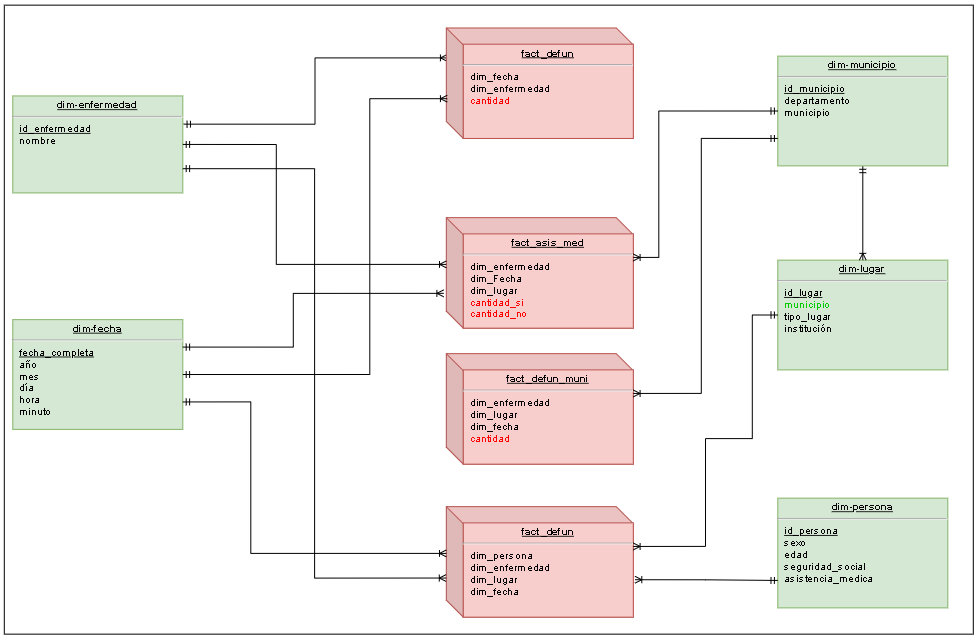


Ilustración 7 Modelo Multidimensional

En la anterior imagen se puede ver las relaciones entre los hechos, los cuales se encuentran en el siguiente orden:

* **Fact\_defuncion**: se define la relación entre las dimensiones de fecha y enfermedad, y se deja como medida la cantidad de muertes registradas de cada enfermedad en cada fecha.
* **Fact\_asis\_medica**: en este hecho se ve la relación entre las dimensiones de fecha, municipio y enfermedad, y teniendo como medidas la cantidad de fallecimientos que si recibieron asistencia médica y aquellos que lastimosamente no pudieron recibir asistencia médica.
* **Fact\_defunciones\_municipio**: en este hecho se vuelven a emplear las dimensiones de fecha, municipio y enfermedad, enfocándose en generar como medida la cantidad de defunciones registradas en cada municipio y fecha de cada una de las tres enfermedades a analizar.
* **Fact\_muerte**: en este hecho se verán todos los registros de fallecimientos, pero enfocados en las dimensiones de lugar, fecha, enfermedad y persona, esto en el caso que se requiera analizar particularidades presentadas con los fallecimientos y las personas.

## Modelos DF

En estos modelos se busca definir las jerarquías presentes en cada una de las dimensiones, de modo tal se conozca como se ira avanzando por los miembros de cada una de estas, dando un mapa de guía para las futuras operaciones que se van a realizar con el Data Warehouse, además de definir las tablas de hechos y las medidas que harán parte de cada una de estas. A continuación, se definirán los modelos resultantes en base a cada uno de los cubos visualizando la jerarquía de cada una de las dimensiones que los componen.

### Modelo DF – Defunciones por fecha

En este modelo se puede ver la jerarquía de las dimensiones involucradas en el hecho fact\_defun, de modo tal se pueda conocer la forma en que podemos realizar las consultas sobre el cubo y sus dimensiones. Este hecho involucra las dimensiones de fecha, en donde se ve como esta tiene en su nivel más bajo el día y en el más alto el año del fallecimiento, por otro lado, se contempla la dimensión de enfermedad que solo se compone por el nombre de la enfermedad. No se debe olvidar la medida de este hecho que es la cantidad de fallecimientos registrados asociados a cada enfermedad y fecha.

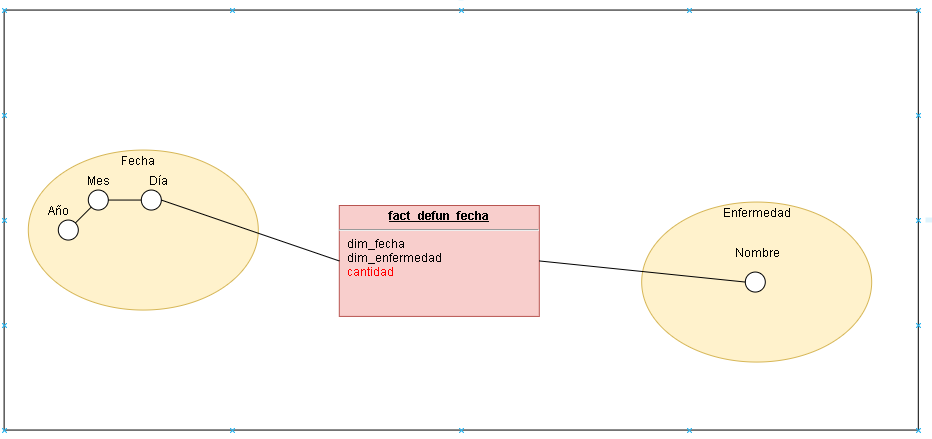


Ilustración 8 Modelo DF – defunciones\_fecha

### Modelo DF – Asistencia médica

En base a la definición de miembros de las dimensiones de Fecha, Lugar y Enfermedad, se obtuvieron las siguientes jerarquías para cada una de las dimensiones y se define la tabla de hechos fact\_asis\_med en la cual se encuentra la medida de cantidad de asistencias médicas prestadas.



Ilustración 9 Modelo DF – asistencia\_medica

### Modelo DF – Defunciones por municipio

En base a la definición de miembros de las dimensiones de Fecha, Lugar y Enfermedad, se definieron las siguientes jerarquías en base a las dimensiones previamente definidas, visualizando finalmente la relación entre estas en la tabla de hechos de defun\_municipio, la cual cuenta como medida la cantidad de fallecimientos registrados en cada municipio y fecha de las tres enfermedades empleadas en este ejercicio.

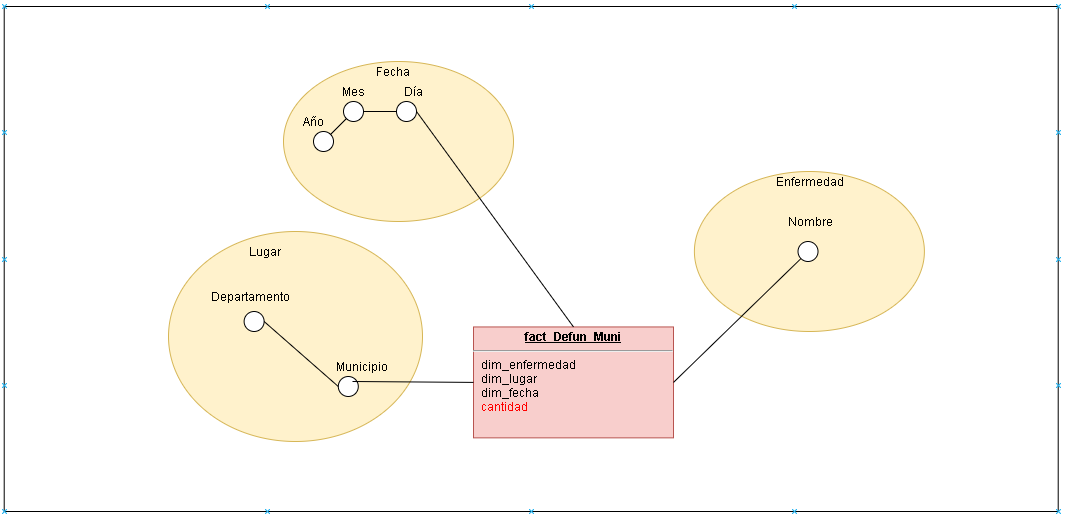


Ilustración 10 Modelo DF – Defunciones\_municipio

Nota: se define la dimensión de lugar, hasta este momento, con los miembros de departamento y municipio por practicidad, ya que igualmente hacen referencia a la dimensión de municipio definida en el modelo multidimensional.

### Modelo DF – Muertes

En este ultimo modelo se define el cubo de muertes el cual contempla las dimensiones de persona, fecha, luchar y enfermedad, sin definir nunca medida, esto con el fin de poder analizar cada uno de los fallecimientos registrados y sus particularidades.

Diagrama

Descripción generada automáticamente

Ilustración 20 Muertes

# DISEÑO LOGICO

Con todos los modelos que nos permitan conocer mucho más a fondo los elementos que compondrían al Data Warehouse destinado a satisfacer las necesidades de la empresa, se diseñara el modelo ROLAP, el cual permitirá la posterior creación de la base de datos.

## Modelo ROLAP

A continuación, se expone el modelo relacional del Data Warehouse el se realiza en base a todo el proceso de diseño conceptual desarrollado previamente.

se expone el modelo relacional del Data Warehouse el se realiza en base a todo el proceso de diseño conceptual desarrollado previamente.

Diagrama

Descripción generada automáticamente

Ilustración 21 Modelo ROLAP

## DDL del Data Ware House

Por medio de MySQL Workbench se generó el siguiente script en el cual se encuentran las sentencias DDL para cada una de las tablas definidas en el modelo:

-- -----------------------------------------------------

-- Schema Bona\_Health\_DW

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `Bona\_Health\_DW` DEFAULT CHARACTER SET utf8 ;

-- -----------------------------------------------------

-- Schema ods\_bona\_health

-- -----------------------------------------------------

USE `Bona\_Health\_DW` ;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`dim\_enfermedad`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`dim\_enfermedad` (

`nombre` VARCHAR(45) NOT NULL,

`descript` VARCHAR(45) NOT NULL,

PRIMARY KEY (`nombre`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`dim\_fecha`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`dim\_fecha` (

`fecha\_completa` DATE NOT NULL,

`anio` INT NOT NULL,

`mes` VARCHAR(45) NOT NULL,

`dia` INT(2) NOT NULL,

PRIMARY KEY (`fecha\_completa`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`fact\_defunciones\_fecha`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`fact\_defunciones\_fecha` (

`enfermedad` VARCHAR(45) NOT NULL,

`fecha` DATE NOT NULL,

`cantidad\_fallecidos` INT NOT NULL,

PRIMARY KEY (`enfermedad`, `fecha`),

INDEX `fk\_Defun\_Fecha1\_idx` (`fecha` ASC) ,

CONSTRAINT `fk\_Defun\_Enfermedad1`

FOREIGN KEY (`enfermedad`)

REFERENCES `Bona\_Health\_DW`.`dim\_enfermedad` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Defun\_Fecha1`

FOREIGN KEY (`fecha`)

REFERENCES `Bona\_Health\_DW`.`dim\_fecha` (`fecha\_completa`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`dim\_municipio`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`dim\_municipio` (

`id\_municipio` VARCHAR(45) NOT NULL,

`departamento` VARCHAR(45) NOT NULL,

`municipio` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_municipio`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`fact\_asis\_medica`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`fact\_asis\_medica` (

`fecha` DATE NOT NULL,

`municipio` VARCHAR(45) NOT NULL,

`sum\_asistencia\_med` INT NOT NULL,

`sum\_sin\_asistencia\_med` INT NOT NULL,

PRIMARY KEY (`fecha`, `municipio`),

INDEX `fk\_Asis\_med\_Fecha1\_idx` (`fecha` ASC) ,

INDEX `fk\_fact\_asis\_medica\_dim\_municipio1\_idx` (`municipio` ASC) ,

CONSTRAINT `fk\_Asis\_med\_Fecha1`

FOREIGN KEY (`fecha`)

REFERENCES `Bona\_Health\_DW`.`dim\_fecha` (`fecha\_completa`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_asis\_medica\_dim\_municipio1`

FOREIGN KEY (`municipio`)

REFERENCES `Bona\_Health\_DW`.`dim\_municipio` (`id\_municipio`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-------------------------------------------------------

-- Table `Bona\_Health\_DW`.`fact\_defun\_por\_municipio`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`fact\_defun\_por\_municipio` (

`fecha` DATE NOT NULL,

`enfermedad` VARCHAR(45) NOT NULL,

`municipio` VARCHAR(45) NOT NULL,

`cant\_defun` INT NOT NULL,

PRIMARY KEY (`fecha`, `enfermedad`, `municipio`),

INDEX `fk\_Defun\_muni\_Enfermedad1\_idx` (`enfermedad` ASC) ,

INDEX `fk\_fact\_defun\_por\_lugar\_dim\_municipio1\_idx` (`municipio` ASC) ,

CONSTRAINT `fk\_Defun\_muni\_Fecha`

FOREIGN KEY (`fecha`)

REFERENCES `Bona\_Health\_DW`.`dim\_fecha` (`fecha\_completa`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Defun\_muni\_Enfermedad1`

FOREIGN KEY (`enfermedad`)

REFERENCES `Bona\_Health\_DW`.`dim\_enfermedad` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_defun\_por\_lugar\_dim\_municipio1`

FOREIGN KEY (`municipio`)

REFERENCES `Bona\_Health\_DW`.`dim\_municipio` (`id\_municipio`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`dim\_persona`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`dim\_persona` (

`id\_persona` VARCHAR(45) NOT NULL,

`sexo` VARCHAR(45) NOT NULL,

`edad` INT NOT NULL,

`seguridad\_social` VARCHAR(45) NOT NULL,

`asistencia\_medica` VARCHAR(2) NOT NULL,

PRIMARY KEY (`id\_persona`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`dim\_lugar`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`dim\_lugar` (

`id\_lugar` VARCHAR(45) NOT NULL,

`tipo\_lugar` VARCHAR(45) NOT NULL,

`institucion` VARCHAR(45) NULL,

`municipio` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id\_lugar`),

INDEX `fk\_dim\_lugar\_dim\_municipio1\_idx` (`municipio` ASC) ,

CONSTRAINT `fk\_dim\_lugar\_dim\_municipio1`

FOREIGN KEY (`municipio`)

REFERENCES `Bona\_Health\_DW`.`dim\_municipio` (`id\_municipio`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `Bona\_Health\_DW`.`fact\_muerte`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `Bona\_Health\_DW`.`fact\_muerte` (

`persona` VARCHAR(45) NOT NULL,

`lugar` VARCHAR(45) NOT NULL,

`nombre` VARCHAR(45) NOT NULL,

`fecha\_completa` DATE NOT NULL,

INDEX `fk\_fact\_muerte\_dim\_persona1\_idx` (`persona` ASC) ,

INDEX `fk\_fact\_muerte\_dim\_lugar1\_idx` (`lugar` ASC) ,

INDEX `fk\_fact\_muerte\_dim\_enfermedad1\_idx` (`nombre` ASC) ,

INDEX `fk\_fact\_muerte\_dim\_fecha1\_idx` (`fecha\_completa` ASC) ,

PRIMARY KEY (`persona`, `lugar`, `nombre`, `fecha\_completa`),

CONSTRAINT `fk\_fact\_muerte\_dim\_persona1`

FOREIGN KEY (`persona`)

REFERENCES `Bona\_Health\_DW`.`dim\_persona` (`id\_persona`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_muerte\_dim\_lugar1`

FOREIGN KEY (`lugar`)

REFERENCES `Bona\_Health\_DW`.`dim\_lugar` (`id\_lugar`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_muerte\_dim\_enfermedad1`

FOREIGN KEY (`nombre`)

REFERENCES `Bona\_Health\_DW`.`dim\_enfermedad` (`nombre`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_muerte\_dim\_fecha1`

FOREIGN KEY (`fecha\_completa`)

REFERENCES `Bona\_Health\_DW`.`dim\_fecha` (`fecha\_completa`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

# ETL

Con la creación del Data Warehouse por medio del script expuesto anteriormente, a continuación, se expondrán los procesos ETL por medio de los cuales se dejará finalmente el Data Warehouse cargado y listo para ser usado. Se mostrarán las actividades relacionadas con el diseño de los ETL, secuencia de los procesos, creación de la base de datos ODS y el programa desarrollado en Python en el cual se integrará el diseño previamente expuesto.

## Diseño de ETL

Antes de desarrollar el componente de software destinado para los procesos de extracción de la información de los archivos CSV en donde se encuentra los datos relacionados con las defunciones, transformación o limpieza de los datos y carga de los datos, se realizo el siguiente diagrama en el cual se exponen los elementos involucrados en el proceso, definiendo las fuentes de datos, la forma en que se extraerán los datos de estas, métodos y clases requeridos para la conexión y ejecución de sentencias SQL y un listado previo de las trasformaciones a realizar. En este diagrama se hace un vistazo a lo que se podría implementar en el programa, pero puede que algunos métodos o elementos no se implementen, esto ya dependerá exclusivamente de los inconvenientes que se presenten sobre el desarrollo.

Escala de tiempo

Descripción generada automáticamente

Ilustración 22 Diagrama de Clases y componentes del sistema

A continuación, se definen algunos elementos presentados en el anterior diagrama:

* Fuentes de datos: en este apartado se define la carpeta en la cual se encontrarán los archivos en formato csv con los datos de las defunciones y los códigos de municipios y departamentos requeridos para este ejercicio.
* Proyecto en Python: en este apartado se definen las clases y librerías a emplear en el programa. Para este ejercicio se empleo a Python en su versión 3.9.
  + Pandas: librería enfocada en la manipulación de datagramas los cuales son estructuras de datos las cuales almacenan los datos extraídos de cada uno de los archivos csv a usar.
  + MySQLdb: librería destinada para la conexión a bases de datos MySQL.
  + Archivo\_Mort: clase la cual se encargará de la lectura de los archivos csv y la extracción de los datos para cada una de las tablas definidas en la base de datos ODS.
  + MySQLDB: clase encargada de la creación de las conexiones a las bases de datos MySQL, además de contener métodos para la extracción y carga de datos.
  + Transformaciones: clase con los de limpieza y transformación de los datos extraídos de la base de datos ODS, para la posterior inserción en cada una de las tablas definidas para el Data Warehouse.
* MySQL: en este apartado se encuentran las dos bases de datos que serán el destino de los datos manipulados dentro del programa en Python.
  + ODS Bona Health EPS: base de datos creada en base al modelo entidad – relación de las fuentes de datos a emplear en este ejercicio.
  + Bona Health EPS DW: base de datos para el Data Warehouse de este ejercicio.

## Secuencia de ETL

Con los componentes definidos a emplear en este proyecto, ahora se realizara un listado o secuencia de las actividades a realizar para la extracción de los datos de los archivos csv, carga de estos en la base de datos ODS, la extracción de los datos del ODS, la transformación de estos y su posterior carga al Data Warehouse.

Imagen que contiene Diagrama

Descripción generada automáticamente

Ilustración 23 Secuencia de procesos ETL

A grandes rasgos, a continuación, se expondrán los principales procesos definidos en el diagrama:

1. **Abertura y creación de Dataframes**: este proceso contempla la lectura de los archvis csv y su conformación como Dataframes manipulables en el programa. Este proceso se realizará con cada uno de los archivos, contemplando que la extracción de los datos se realizara de forma individual con los archivos que contienen los datos sobre las defunciones, y se prevé que se requiera el uso de un Dataframe en donde se encuentre la información de los departamentos y municipios.
2. **Conexión al ODS**: este proceso contempla el ingreso de los datos requeridos para la conexión a la base de datos ODS en la cual se almacenarán los datos extraídos de los archivos.

Si este proceso no resulta exitoso, es decir, con la conexión al ODS correctamente, se detiene todo, con el fin de evitar errores en la ejecución.

1. **Extracción de datos de archivos CSV y carga al ODS**: este proceso contemplara un conjunto de subprocesos, cada uno destinado para la extracción de un conjunto de datos en específico, el curado de los datos extraídos y la posterior carga de estos a la base de datos ODS. En este proceso se contemplan las primeras trasformaciones las cuales facilitaran posteriormente la carga de los datos al Data Warehouse. El orden de estos subprocesos se basa en las dependencias de tablas del ODS.
2. **Conexión al Data Warehouse**: este proceso contempla el ingreso de los datos requeridos para la conexión a la base de datos definida para el Data Warehouse en la cual se almacenarán los datos extraídos de la base de datos ODS en la cual se encuentran los datos extraídos d ellos archivos csv.

Si este proceso no resulta exitoso, es decir, con la conexión al Data Warehouse correctamente, se detiene todo, con el fin de evitar errores en la ejecución.

1. **ETL (Extracción, trasformaciones y carga al Data Warehouse)**: este proceso contemplara un conjunto de subprocesos secuenciados en los cuales se realizará la extracción de los datos contenidos en el ODS, la transformación de estos que permita su posterior inserción en el Data Warehouse. El orden en que se cargan cada uno de los datos se hace en base a las dimensiones definidas, la dependencia entre estas y dejando siempre al final la carga de los datos en las tablas de hechos, esto con el fin de evitar conflictos de claves foráneas. Debido a que, al momento que se cargaron los datos de los archivos csv al ODS se realizaron ya algunas transformaciones y conociendo la estructura de las tablas del ODS y el data Warehouse, se prevé que, para la carga de los datos de ciertas tablas de dimensiones y hechos no se requería implementar métodos para que los datos encajen con la estructura definida en estas tablas.
2. 7. **Cierre de conexiones**: en estos dos procesos se cerrarán las dos conexiones empleadas para la extracción y carga de datos en el ODS y el almacén de datos.

## Modelo relacional ODS

Antes del desarrollo del componente de software encargado de la extracción, transformación y carga de datos, se realiza el modelado de la base de datos ODS en la cual se almacenarán los datos que se encuentran en los archivos csv.

Diagrama

Descripción generada automáticamente

Con este modelo, en el cual se definen las tablas para conocer la ubicación en donde ocurrió el fallecimiento, la fecha donde ocurrió, los datos personales de la persona fallecida y la enfermedad que causo el descenso, se genero el siguiente script con las sentencias DDL para la creación de la base de datos y las tablas:

--

-- Base de datos: `ods\_bona\_health`

--

CREATE SCHEMA IF NOT EXISTS ` ods\_bona\_health` DEFAULT CHARACTER SET utf8 ;

-- -----------------------------------------------------

-- Schema ods\_bona\_health

-- -----------------------------------------------------

USE ` ods\_bona\_health ` ;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `departamento`

--

CREATE TABLE `departamento` (

`cod\_departamento` int(11) NOT NULL,

`nombre` varchar(45) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `enfermedad`

--

CREATE TABLE `enfermedad` (

`nombre` varchar(45) NOT NULL,

`observacion` varchar(45) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `fallecimiento`

--

CREATE TABLE `fallecimiento` (

`lugar` varchar(45) NOT NULL,

`fecha` date NOT NULL,

`persona` varchar(45) NOT NULL,

`enfermedad` varchar(45) NOT NULL,

`asistencia\_medica` varchar(2) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `fecha`

--

CREATE TABLE `fecha` (

`fecha\_completa` date NOT NULL,

`año` int(4) NOT NULL,

`mes` varchar(45) NOT NULL,

`dia` int(2) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `institucion`

--

CREATE TABLE `institucion` (

`cod\_institucion` varchar(45) NOT NULL,

`nombre` varchar(60) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `lugar`

--

CREATE TABLE `lugar` (

`id\_lugar` varchar(45) NOT NULL,

`tipo\_lugar` varchar(45) NOT NULL,

`departamento` int(11) NOT NULL,

`municipio` int(11) NOT NULL,

`institucion` varchar(45) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `municipio`

--

CREATE TABLE `municipio` (

`cod\_municipio` int(11) NOT NULL,

`cod\_departamento` int(11) NOT NULL,

`nombre` varchar(45) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `persona`

--

CREATE TABLE `persona` (

`id\_persona` varchar(45) NOT NULL,

`fecha\_nam` date NOT NULL,

`edad` int(2) NOT NULL,

`sexo` varchar(45) NOT NULL,

`seg\_social` varchar(45) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `departamento`

--

ALTER TABLE `departamento`

ADD PRIMARY KEY (`cod\_departamento`);

--

-- Indices de la tabla `enfermedad`

--

ALTER TABLE `enfermedad`

ADD PRIMARY KEY (`nombre`);

--

-- Indices de la tabla `fallecimiento`

--

ALTER TABLE `fallecimiento`

ADD PRIMARY KEY (`lugar`,`fecha`,`persona`,`enfermedad`),

ADD KEY `fk\_fallecimiento\_lugar1\_idx` (`lugar`),

ADD KEY `fk\_fallecimiento\_fecha1\_idx` (`fecha`),

ADD KEY `fk\_fallecimiento\_persona1\_idx` (`persona`),

ADD KEY `fk\_fallecimiento\_enfermedad1\_idx` (`enfermedad`);

--

-- Indices de la tabla `fecha`

--

ALTER TABLE `fecha`

ADD PRIMARY KEY (`fecha\_completa`);

--

-- Indices de la tabla `institucion`

--

ALTER TABLE `institucion`

ADD PRIMARY KEY (`cod\_institucion`);

--

-- Indices de la tabla `lugar`

--

ALTER TABLE `lugar`

ADD PRIMARY KEY (`id\_lugar`),

ADD KEY `fk\_lugar\_institucion1\_idx` (`institucion`),

ADD KEY `fk\_lugar\_municipio1\_idx` (`municipio`,`departamento`);

--

-- Indices de la tabla `municipio`

--

ALTER TABLE `municipio`

ADD PRIMARY KEY (`cod\_municipio`,`cod\_departamento`),

ADD KEY `fk\_municipio\_departamento\_idx` (`cod\_departamento`);

--

-- Indices de la tabla `persona`

--

ALTER TABLE `persona`

ADD PRIMARY KEY (`id\_persona`);

--

-- Restricciones para tablas volcadas

--

--

-- Filtros para la tabla `fallecimiento`

--

ALTER TABLE `fallecimiento`

ADD CONSTRAINT `fk\_fallecimiento\_enfermedad1` FOREIGN KEY (`enfermedad`) REFERENCES `enfermedad` (`nombre`) ON DELETE NO ACTION ON UPDATE NO ACTION,

ADD CONSTRAINT `fk\_fallecimiento\_fecha1` FOREIGN KEY (`fecha`) REFERENCES `fecha` (`fecha\_completa`) ON DELETE NO ACTION ON UPDATE NO ACTION,

ADD CONSTRAINT `fk\_fallecimiento\_lugar1` FOREIGN KEY (`lugar`) REFERENCES `lugar` (`id\_lugar`) ON DELETE NO ACTION ON UPDATE NO ACTION,

ADD CONSTRAINT `fk\_fallecimiento\_persona1` FOREIGN KEY (`persona`) REFERENCES `persona` (`id\_persona`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--

-- Filtros para la tabla `lugar`

--

ALTER TABLE `lugar`

ADD CONSTRAINT `fk\_lugar\_institucion1` FOREIGN KEY (`institucion`) REFERENCES `institucion` (`cod\_institucion`) ON DELETE NO ACTION ON UPDATE NO ACTION,

ADD CONSTRAINT `fk\_lugar\_municipio1` FOREIGN KEY (`municipio`,`departamento`) REFERENCES `municipio` (`cod\_municipio`, `cod\_departamento`) ON DELETE NO ACTION ON UPDATE NO ACTION;

--

-- Filtros para la tabla `municipio`

--

ALTER TABLE `municipio`

ADD CONSTRAINT `fk\_municipio\_departamento` FOREIGN KEY (`cod\_departamento`) REFERENCES `departamento` (`cod\_departamento`) ON DELETE NO ACTION ON UPDATE NO ACTION;

COMMIT;

## Programa ETL

Con la base de datos ODS, el Data Warehouse, las clases previamente definidas y la secuencia de procesos creada, se realizo el siguiente programa en Python, el cual cuenta con comentarios para comprender su funcionamiento. Este programa se entregará en un archivo comprimido, con el fin de prevenir problemas con las librerías implementadas. Además, dentro del directorio del programa, se encontrara la carpeta DATASETS en donde estarán los archivos csv con los datos a manipular por el programa.

from datetime import datetime # Libreria para la manipulacion de fechas  
  
import MySQLdb.connections as con # Libreria para la conexion con bases MySQL  
import pandas as pd # Libreria para la manipulacion de archivos CSV  
from dateutil.parser import parse # Metodo para formatear fechas en Strings sin indicar el formato de destino  
from dateutil.relativedelta import relativedelta # Metodo para encontrar la diferencia entre dos fechas  
  
  
# Clase destinada para la conexion a bases de datos MySQL  
# von los metodos que permiten la carga y extraccion de datos  
class MySQLCon:  
  
 # Contructor que recibe los datos necesarios para la conexion  
 # a una base de datos con sus respectivas credenciales  
 # y que a su vez inicia la conexion a la base de datos  
 # Para las consultas e inserciones se crean listas con las sentencias requeridas  
 def \_\_init\_\_(self, hostname, port, user, password, db):  
 self.hostname = hostname  
 self.port = port  
 self.username = user  
 self.password = password  
 self.database = db  
 self.condb = None  
 self.connect()  
 self.insertsODS = [("ods\_bona\_health.departamento (cod\_departamento,nombre) VALUES (%s, %s)", "DEPARTAMENTO"),  
 ("ods\_bona\_health.institucion ( cod\_institucion,nombre) VALUES (%s, %s)", "INSTITUCION"),  
 ("ods\_bona\_health.municipio (cod\_municipio,cod\_departamento,nombre) VALUES (%s, %s, %s)",  
 "MUNICIPIO"),  
 ("ods\_bona\_health.lugar (id\_lugar,tipo\_lugar,departamento,municipio,institucion) VALUES (%s, %s, %s,%s, %s)",  
 "LUGAR"),  
 ("ods\_bona\_health.fecha (fecha\_completa,año,mes,dia) VALUES (%s, %s, %s,%s)", "FECHA"),  
 ("ods\_bona\_health.persona (id\_persona,fecha\_nam,edad,sexo,seg\_social) VALUES (%s, %s, %s,%s,%s)",  
 "PERSONA"),  
 (  
 "ods\_bona\_health.fallecimiento (lugar,fecha,persona,enfermedad,asistencia\_medica) VALUES (%s,%s, %s,%s,%s)",  
 "FALLECIMIENTO")]  
 self.selectsODS = ["\* FROM fecha;", "\* FROM enfermedad;",  
 "municipio.\*, departamento.nombre from municipio INNER JOIN departamento ON municipio.cod\_departamento = departamento.cod\_departamento;",  
 "persona.\*, fallecimiento.asistencia\_medica from persona INNER JOIN fallecimiento ON persona.id\_persona = fallecimiento.persona;",  
 "\* FROM lugar;",  
 "fallecimiento.fecha, fallecimiento.enfermedad, lugar.departamento, lugar.municipio FROM fallecimiento INNER JOIN lugar ON fallecimiento.lugar = lugar.id\_lugar;",  
 "fallecimiento.fecha, fallecimiento.enfermedad, lugar.departamento, lugar.municipio FROM fallecimiento INNER JOIN lugar ON fallecimiento.lugar = lugar.id\_lugar;",  
 "fallecimiento.fecha, fallecimiento.enfermedad, lugar.departamento, lugar.municipio,fallecimiento.asistencia\_medica FROM fallecimiento INNER JOIN lugar ON fallecimiento.lugar = lugar.id\_lugar;",  
 "fallecimiento.persona, fallecimiento.lugar, fallecimiento.enfermedad, fallecimiento.fecha FROM `fallecimiento`;"]  
 self.insertBHDW = [("dim\_fecha (fecha\_completa,anio,mes,dia) VALUES (%s,%s,%s,%s);", "DIM FECHA"),  
 ("dim\_enfermedad (nombre, descript) VALUES (%s,%s);", "DIM ENFERMEDAD"),  
 ("dim\_municipio (id\_municipio,departamento,municipio) VALUES (%s,%s,%s);", "DIM MUNICIPIO"),  
 (  
 "dim\_persona (id\_persona,sexo,fecha\_nac,edad,seguridad\_social,asistencia\_medica) VALUES (%s,%s,%s,%s,%s,%s);",  
 "DIM PERSONA"),  
 ("dim\_lugar (id\_lugar,municipio,tipo\_lugar,institucion) VALUES (%s,%s,%s,%s);", "DIM LUGAR"),  
 ("fact\_defun\_por\_municipio (fecha,enfermedad,municipio,cant\_defun) VALUES (%s,%s,%s,%s);",  
 "FACT\_DEFUN\_POR\_MUNICIPIO"),  
 ("fact\_defunciones\_fecha (fecha,enfermedad, cantidad\_fallecidos) VALUES (%s,%s,%s);",  
 "FACT\_DEFUN\_POR\_FECHA"),  
 (  
 "fact\_asis\_medica (fecha,municipio,enfermedad,sum\_asistencia\_med,sum\_sin\_asistencia\_med) VALUES (%s,%s,%s,%s,%s);",  
 "FACT\_ASIS\_MEDICA"),  
 ("fact\_muerte (persona,lugar,enfermedad,fecha\_completa) VALUES (%s,%s,%s,%s);",  
 "FACT\_MUERTE")]  
  
 # Metodo para la conexion a la base de datos con la libreria  
 # MySQLdb con los datos almancenados en el cosntructor de la instancia  
 def connect(self):  
 try:  
 self.condb = con.Connection(host=self.hostname, port=self.port,  
 user=self.username,  
 passwd=self.password,  
 db=self.database)  
 cursor = self.condb.cursor()  
 cursor.execute("SET lc\_time\_names = 'es\_CO';")  
 print("Conexión exitoso a " + self.database)  
 except Exception as e:  
 print("Error de conexión", e)  
  
 # Metodo para la carga de datosen la BD, el cual recibe parte de la sentencia para la carga de los datos,  
 # una lista con la tuplas de los datos a cargar y una cadena con el nombre de la tabla. La carga de los datos se  
 # realiza con el metodo executemany, el cual usa la sentencia SQL y la lista que iterara para cargar los datos  
 def load(self, sql, data, table):  
 cursor = self.condb.cursor()  
 try:  
 cursor.executemany("INSERT IGNORE INTO " + sql, data)  
 print("TABLA ", table, " CARGADA CON ", len(data),  
 " REGISTROS (Los registros pueden ser repetidos y por ende haber sido ignroados)")  
 self.condb.commit()  
 except Exception as e:  
 print("Error de insercion", e)  
 self.condb.rollback()  
  
 # Metodo para la extraccion de datos de la BD, el cual un String para complementar la sentencia SELECT  
 # y retornara los datos obtenidos por el cursor, los cuales vienen en una lista de tuplas  
 def extract(self, sql):  
 cursor = self.condb.cursor()  
 try:  
 cursor.execute("SELECT " + sql)  
 return cursor.fetchall()  
 except Exception as e:  
 print("Error de consulta", e)  
  
 # Metodo encargado de cerrar la conexion con la base de datos,  
 # conexion que se encuentra alamcenada en la variable condb  
 def closeconection(self):  
 try:  
 self.condb.close()  
 print('Desconexión exitosa a ' + self.database)  
 except Exception as e:  
 print("Error de desconexión", e)  
  
  
## Clase encarga de la manipulacion de los archivos csv  
# haciendo uso de la libreria Pandas  
class ArchivoMort:  
  
 # Contructor de la clase, el cual recibe el nombre del archivo ubicado en la carpeta  
 # DATASETS y el nombre de la enfermedad asociada a este. Dentro de este metodo se incluyen  
 # parametros como separador de datos, nombres de las columnas y se crea el dataframe a manipular  
 def \_\_init\_\_(self, nombre, enfermedad):  
 self.ruta = 'DATASETS/' + nombre  
 self.sep = ';'  
 self.names = ['COD\_DPTO', 'COD\_MUNIC', 'A\_DEFUN', 'COD\_INSP', 'SIT\_DEFUN', 'OTRSITIODE',  
 'COD\_INST', 'NOM\_INST', 'TIPO\_DEFUN', 'FECHA\_DEF', 'ANO', 'MES', 'HORA', 'MINUTOS',  
 'SEXO', 'FECHA\_NAC', 'EST\_CIVIL', 'EDAD', 'NIVEL\_EDU', 'ULTCURFAL', 'MUERTEPORO',  
 'SIMUERTEPO', 'OCUPACION', 'IDPERTET', 'IDPUEBIN', 'N\_IDPUEBIN', 'CODPRES', 'CODPTORE',  
 'CODMUNRE', 'AREA\_RES', 'BARRIOFAL', 'COD\_LOCA', 'CODIGO', 'VEREDAFALL', 'SEG\_SOCIAL',  
 'IDADMISALU', 'IDCLASADMI', 'PMAN\_MUER', 'CONS\_EXP', 'MU\_PARTO', 'T\_PARTO', 'TIPO\_EMB',  
 'T\_GES', 'PESO\_NAC', 'EDAD\_MADRE', 'N\_HIJOSV', 'N\_HIJOSM', 'EST\_CIVM', 'NIV\_EDUM',  
 'ULTCURMAD', 'EMB\_FAL', 'EMB\_SEM', 'EMB\_MES', 'MAN\_MUER', 'COMOCUHEC', 'CODOCUR',  
 'CODMUNOC', 'DIROCUHEC', 'LOCALOCUHE', 'C\_MUERTE', 'C\_MUERTEB', 'C\_MUERTEC', 'C\_MUERTED',  
 'C\_MUERTEE', 'ASIS\_MED', 'N\_DIR1', 'T\_DIR1', 'M\_DIR1', 'C\_DIR1', 'C\_DIR12', 'N\_ANT1', 'T\_ANT1',  
 'M\_ANT1', 'C\_ANT1', 'C\_ANT12', 'N\_ANT2', 'T\_ANT2', 'M\_ANT2', 'C\_ANT2', 'C\_ANT22', 'N\_ANT3',  
 'T\_ANT3',  
 'M\_ANT3', 'C\_ANT3', 'C\_ANT32', 'N\_PAT1', 'T\_PAT1', 'M\_PAT1', 'C\_PAT1', 'N\_PAT2', 'C\_PAT2',  
 'N\_BAS1',  
 'C\_BAS1', 'N\_MCM1', 'C\_MCM1', 'CAUSA\_666', 'IDPROFCER', 'DD\_EXP', 'MM\_EXP', 'FECHA\_EXP',  
 'FECHAGRA',  
 'CAU\_HOMOL', 'GRU\_ED1', 'GRU\_ED2', 'HORA\_SE']  
 # Dataframe de los datos del csv relacionados con la mortalidad de una enfermedad  
 self.dataframe = pd.read\_csv(self.ruta, sep=self.sep, header=None, names=self.names)  
 # Dataframe con los codigos de los municipios y departamentos del pais  
 self.dataframe\_lugar = pd.read\_csv("DATASETS/Departamentos\_y\_municipios\_de\_Colombia.csv", sep=self.sep,  
 header=None,  
 names=['REGION', 'CÓDIGO DANE DEL DEPARTAMENTO', 'DEPARTAMENTO',  
 'CÓDIGO DANE DEL MUNICIPIO', 'MUNICIPIO'])  
 # Tupla con los nombres de los meses en español  
 self.meses = (  
 "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre",  
 "Noviembre",  
 "Diciembre")  
 self.enfermedad = enfermedad  
  
 # Metodo para la verificación de valores vacios extraidos del dataframe  
 def isNaN(self, num):  
 return num != num  
  
 # Metodo para el llenado de digitos de los numeros  
 def llenar(self, param):  
 return str(('0' \* (3 - len(str(param)))) + str(param))  
  
 # Metodo encargado de la extraccion del nombre de un departamento con base en su codigo  
 def search\_departament(self, clave):  
 for i in self.dataframe\_lugar.index:  
 if self.dataframe\_lugar['CÓDIGO DANE DEL DEPARTAMENTO'][i] == clave:  
 return self.dataframe\_lugar['DEPARTAMENTO'][i]  
 else:  
 return "SIN NOMBRE"  
  
 # Metodo encargado de la extraccion de la fecha de nacimiento de una persona, permitiendo la limpieza  
 # de los valores y haciendo uso de los valores de fecha de defuncion y edad en los casos requeridos  
 # en donde no se tenga la fecha de nacimiento de la persona, además de arreglar problemas de digitación  
 def fecha\_nam(self, i):  
 clave, fechad, edad = self.dataframe['FECHA\_NAC'][i], self.dataframe['FECHA\_DEF'][i], self.dataframe['EDAD'][i]  
 if type(clave) is str:  
 if not self.isNaN(clave):  
 dt2 = parse(fechad)  
 return dt2 + relativedelta(years=-int(edad))  
 else:  
 dt, dt2 = parse(clave), parse(fechad)  
 if dt > dt2:  
 dt = dt + relativedelta(years=-100)  
 return dt.date()  
 else:  
 dt2 = parse(fechad)  
 return dt2 + relativedelta(years=-int(edad))  
  
 # Metodo encargado de la extraccion del nombre de un municipio con base en su codigo  
 def search\_municipe(self, clave):  
 for i in self.dataframe\_lugar.index:  
 if self.dataframe\_lugar['CÓDIGO DANE DEL MUNICIPIO'][i] == clave:  
 return self.dataframe\_lugar['MUNICIPIO'][i]  
 else:  
 return "SIN NOMBRE"  
  
 # Metodo para la extraccion de los datos de los departamentos en donde ocurriendo las defunciones  
 # en donde se crean tuplas con el codigo y nombre del departamento, para ser almacenados en  
 # la lista. Se controlan los duplicados con una lista auxiliar en donde se almacenen los  
 # codigos de los departamentos  
 def extract\_departament(self):  
 lista, pk = [], []  
 for i in self.dataframe.index:  
 clave = self.dataframe['COD\_DPTO'][i]  
 if clave not in pk:  
 pk.append(clave)  
 aux = (clave, self.search\_departament(clave))  
 lista.append(aux)  
 return lista  
  
 # Metodo para la extraccion de los datos de las instituciones en donde ocurriendo las defunciones  
 # en donde se crean tuplas con el codigo y nombre de la institucion, para ser almacenados en  
 # la lista. Se controlan los duplicados con una lista auxiliar en donde se almacenen los  
 # codigos de las intituciones. Esta columna tienen valores vacios, que se cnontrolar con el metodo isNaN  
 def extract\_institutes(self):  
 lista, pk = [], []  
 for i in self.dataframe.index:  
 if not self.isNaN(self.dataframe['COD\_INST'][i]):  
 clave = str(self.dataframe['COD\_INST'][i]).replace(';', ".")  
 if clave not in pk:  
 pk.append(clave)  
 aux = (clave, self.dataframe['NOM\_INST'][i])  
 lista.append(aux)  
 return lista  
  
 # Metodo para la extraccion de los datos de los departamentos en donde ocurriendo las defunciones  
 # en donde se crean tuplas con el codigo del municipio, nombre del municipio y codigo del departamento, para ser almacenados en  
 # la lista. Se controlan los duplicados con una lista auxiliar en donde se almacenen los una clave compuesta por el codigo  
 # del departamento y del municipio  
 def extract\_municipes(self):  
 lista, pk = [], []  
 for i in self.dataframe.index:  
 clave = int(str(self.dataframe['COD\_DPTO'][i]) + (self.llenar(self.dataframe['COD\_MUNIC'][i])))  
 if clave not in pk:  
 pk.append(clave)  
 aux = (self.dataframe['COD\_MUNIC'][i], self.dataframe['COD\_DPTO'][i], self.search\_municipe(clave))  
 lista.append(aux)  
 return lista  
  
 # Metodo para la extraccion de los lugares de defuncion de las personas, en donde se crean tuplas con una clave compuesta  
 # por el codigo del dep, municipio, y si ocurrio en una institucion, esta se añade. Se realiza un control del valor  
 # vacio de la institucion y de los egistros duplicados por medio de la clave compuesta.  
 def extract\_place(self):  
 lista, pk, ins = [], [], ""  
 for i in self.dataframe.index:  
 if not self.isNaN(self.dataframe['COD\_INST'][i]):  
 ins = str(self.dataframe['COD\_INST'][i]).replace(';', ".")  
 clave = str(self.dataframe['COD\_DPTO'][i]) + "-" + str(self.dataframe['COD\_MUNIC'][i]) + "-" + ins  
 else:  
 ins = None  
 clave = str(self.dataframe['COD\_DPTO'][i]) + "-" + str(self.dataframe['COD\_MUNIC'][i])  
 if clave not in pk:  
 pk.append(clave)  
 aux = (  
 clave, self.dataframe['SIT\_DEFUN'][i], self.dataframe['COD\_DPTO'][i],  
 self.dataframe['COD\_MUNIC'][i],  
 ins)  
 lista.append(aux)  
 return lista  
  
 # Metodo para la extracion de las fechas de las defunciones registradas, en donde se crean tuplas con la fecha completa,  
 # el año, nombre del mes y día, se controlan los valores duplicados por medio de una lista auxiliar con las fechas  
 # completas  
 def extract\_date(self):  
 l, pk = [], []  
 for i in self.dataframe.index:  
 fechad = self.dataframe['FECHA\_DEF'][i]  
 dt2 = parse(fechad).date()  
 if dt2 not in pk:  
 aux = (dt2, dt2.year, self.meses[dt2.month - 1], dt2.day)  
 pk.append(dt2)  
 l.append(aux)  
 return l  
  
 # Extraccion de los datos personas de los fallecidos, creacion tuplas con ña fecha de nacimiento, edad, sexo y seguridad  
 # social de la persona, además de una clave compuesta usando todos los datos anteriomente expuestos  
 def extract\_person(self):  
 l, pk = [], []  
 for i in self.dataframe.index:  
 f, edad, sexo, segsoc = self.fecha\_nam(i), self.dataframe['EDAD'][i], self.dataframe['SEXO'][i], \  
 self.dataframe['SEG\_SOCIAL'][i]  
 clave = str(f.year) + str(f.month) + str(f.day) + "-" + str(edad) + "-" + sexo[0] + "-" + segsoc  
 if clave is not pk:  
 pk.append(clave)  
 aux = (clave, f, edad, sexo, segsoc)  
 l.append(aux)  
 return l  
  
 # Metodo encargado de la extraccion de los datos principales del fallecimiento de una persona, contemplando  
 # la fecha, creacion de la clave compuesta para el lugar y la persona y la cinluscion del nombre de la enfermedad  
 # y el dato de si la persona recibio o no asistencia medica  
 def extract\_deaths(self, enf):  
 lista, pk, lugar = [], [], ""  
 for i in self.dataframe.index:  
 if not self.isNaN(self.dataframe['COD\_INST'][i]):  
 ins = str(self.dataframe['COD\_INST'][i]).replace(';', ".")  
 lugar = str(self.dataframe['COD\_DPTO'][i]) + "-" + str(self.dataframe['COD\_MUNIC'][i]) + "-" \  
 + ins  
 else:  
 lugar = str(self.dataframe['COD\_DPTO'][i]) + "-" + str(self.dataframe['COD\_MUNIC'][i])  
 fechad = self.dataframe['FECHA\_DEF'][i]  
 df = parse(fechad).date()  
 f, edad, sexo, segsoc = self.fecha\_nam(i), self.dataframe['EDAD'][i], self.dataframe['SEXO'][i], \  
 self.dataframe['SEG\_SOCIAL'][i]  
 persona = str(f.year) + str(f.month) + str(f.day) + "-" + str(edad) + "-" + sexo[0] + "-" + segsoc  
 am = str(self.dataframe['ASIS\_MED'][i]).upper()  
 if am == "IG" or am == "SÍ":  
 am = "SI"  
 aux = (lugar, df, persona, enf, am)  
 lista.append(aux)  
 return lista  
  
  
# Clase encargada de las tranformaciones necesarias para la carga de los datos extraidos de la ODS  
# para el DW.  
class Tranformaciones:  
  
 # Constructora sin parametros  
 def \_\_init\_\_(self):  
 pass  
  
 # Método encargado de la búsqueda de la posición de un item en una lista de tuplas, de modo tal se comparar las  
 # Posiciones de la tupla ingresada con las posiciones de cada una de las tuplas de la lista. Se retorna la posición  
 # de la tupla con el numero K de posiciones de la tupla. Se envía -1 si no se encuentra la tupla en la lista  
 def search\_item(self, l, c, lim):  
 x = 0  
 for i in l:  
 k = 0  
 for j in range(len(i) - lim):  
 if i[j] == c[j]:  
 k = k + 1  
 else:  
 break  
 if k == len(i) - lim:  
 return x  
 x = x + 1  
 return -1  
  
 # Metodo para la trabformacion de los valores de municipios en donde se crea una tupla con una clase compuesta por  
 # el cod del departamento y municipio (al igual que en el listado del DANE) y se incluye el nombre del departamento  
 # y municipio. No se controlan duplicados  
 def transform\_municipe(self, lista):  
 l = []  
 for i in lista:  
 clave = str(i[1]) + "." + str(i[0])  
 aux = (clave, str(i[3]), str(i[2]))  
 l.append(aux)  
 return l  
  
 # Metodo encargado de la creación de las tuplas ordenadas de los daros personales de los fallecidos  
 def transform\_person(self, lista):  
 l = []  
 for i in lista:  
 aux = (i[0], i[3], i[1], i[2], i[4], i[5])  
 l.append(aux)  
 return l  
  
 # Metodo encargado de la tranformación de las tuplas con los datos de los lugares de fallecimiento de las personas  
 # en donde se congirua la clave del municipio y se organizan los datos extraidos del ODS  
 def transform\_place(self, lista):  
 l = []  
 for i in lista:  
 clave = str(i[2]) + "." + str(i[3])  
 aux = (i[0], clave, i[1], i[4])  
 l.append(aux)  
 return l  
  
 # Metodo para la tranformacion de los datos de las defunciones por muncipio, en donde se contempla la creacion de  
 # listas con el codigo del municipio, fecha y enfermedad, en donde se controlan los valores duplicados y se acunulan  
 # los registros duplicados en la ultima posicion de cada lista.  
 def transform\_fact\_def\_mun(self, lista):  
 l = []  
 for i in lista:  
 mun = str(i[2]) + "." + str(i[3])  
 aux = [i[0], i[1], mun, 1]  
 pos = self.search\_item(l, aux, 1)  
 if pos == -1:  
 l.append(aux)  
 else:  
 a = l[pos]  
 a[3] = a[3] + 1  
 l[pos] = a  
 return l  
  
 # Metodo para la tranformación de los datos de defunciones por fecha, creando listas con la fecha y enfermedad,  
 # controlando los valores duplicados y acumulandolos en la ultima posicion de la lista  
 def transform\_fact\_def\_fecha(self, lista):  
 l = []  
 for i in lista:  
 aux = [i[0], i[1], 1]  
 pos = self.search\_item(l, aux, 1)  
 if pos == -1:  
 l.append(aux)  
 else:  
 a = l[pos]  
 a[2] = a[2] + 1  
 l[pos] = a  
 return l  
  
 # Metodo para la tranformacion de los datos de asistencia medica, creando listas con los valores de fecha, muncipio y  
 # enfermedad, controlando si la persona recibio o no asistencia medica en las dos ultimas posiciones de cada lista.  
 def transform\_fact\_asis\_medica(self, lista):  
 l = []  
 for i in lista:  
 mun = str(i[2]) + "." + str(i[3])  
 aux = [i[0], mun, i[1], 0, 0]  
 pos = self.search\_item(l, aux, 2)  
 if pos == -1:  
 if i[4] == "SI":  
 aux[3] = aux[3] + 1  
 else:  
 aux[4] = aux[4] + 1  
 l.append(aux)  
 else:  
 a = l[pos]  
 if i[4] == "SI":  
 a[3] = a[3] + 1  
 else:  
 a[4] = a[4] + 1  
 l[pos] = a  
 return l  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 # Creacion de la conexion con ods\_bona\_health  
 ods = MySQLCon('localhost', 3306, 'root', 'password', 'ods\_bona\_health')  
 if ods.condb is not None:  
 # Creacion de los dataframes para los csv con los datos de mortalidad  
 arc1, arc2, arc3 = ArchivoMort("Mort\_Can\_Mama.csv", "Cancer de Mama"), ArchivoMort("Mort\_Can\_Pulmon.csv",  
 "Cancer de Pulmón"), ArchivoMort(  
 "Mort\_VIH.csv", "VIH")  
 print("INICIO CARGA DE DATOS ODS", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))  
 # Lista de enfermedades a cargar  
 lista = [(arc3.enfermedad, "Virus de inmunodeficiencia humana"), (arc1.enfermedad, ""), (arc2.enfermedad, "")]  
 # carga de la tabla enfermedad del ODS  
 ods.load("ods\_bona\_health.enfermedad (nombre,observacion) VALUES (%s, %s)", lista, "ENFERMEDAD")  
 # Ciclo for para la extracion de datos de los csv y su porterior carga al ODS  
 for i in range(7):  
 if i == 0:  
 lista = arc1.extract\_departament() + arc2.extract\_departament() + arc3.extract\_departament()  
 elif i == 1:  
 lista = arc1.extract\_institutes() + arc2.extract\_institutes() + arc3.extract\_institutes()  
 elif i == 2:  
 lista = arc1.extract\_municipes() + arc2.extract\_municipes() + arc3.extract\_municipes()  
 elif i == 3:  
 lista = arc1.extract\_place() + arc2.extract\_place() + arc3.extract\_place()  
 elif i == 4:  
 lista = arc1.extract\_date() + arc2.extract\_date() + arc3.extract\_date()  
 elif i == 5:  
 lista = arc1.extract\_person() + arc2.extract\_person() + arc3.extract\_person()  
 elif i == 6:  
 lista = arc1.extract\_deaths("Cancer de Mama") + arc2.extract\_deaths(  
 "Cancer de Pulmón") + arc3.extract\_deaths("VIH")  
 # Carga de los datos a la tabla especificada para cada iteración  
 ods.load(ods.insertsODS[i][0], lista, ods.insertsODS[i][1])  
 print("FIN DE CARGA DE DATOS ODS", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))  
 # Creacion de la conexion a bona\_health\_dw  
 bonaheathl\_dw = MySQLCon('localhost', 3306, 'root', 'password', 'bona\_health\_dw')  
 if bonaheathl\_dw.condb is not None:  
 # Instancia de la clase con las tranformaciones  
 tranform = Tranformaciones()  
 print("INICIO DE CARGA DE DATOS A BONA\_HEALTH\_DW: ", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))  
 # ciclo for para la extracion de los datos con las sentencias Select para el ODS  
 # se definen condiciones para los casos de tranformaciones requeridas  
 for i in range(9):  
 lista = ods.extract(ods.selectsODS[i])  
 if i == 2:  
 lista = tranform.transform\_municipe(lista)  
 elif i == 3:  
 lista = tranform.transform\_person(lista)  
 elif i == 4:  
 lista = tranform.transform\_place(lista)  
 elif i == 5:  
 lista = tranform.transform\_fact\_def\_mun(lista)  
 elif i == 6:  
 lista = tranform.transform\_fact\_def\_fecha(lista)  
 elif i == 7:  
 lista = tranform.transform\_fact\_asis\_medica(lista)  
 # Carga de los datos a la tabla especificada para cada iteración  
 bonaheathl\_dw.load(bonaheathl\_dw.insertBHDW[i][0], lista, bonaheathl\_dw.insertBHDW[i][1])  
 # Cierre de las conexiones al ODs y DW  
 bonaheathl\_dw.closeconection()  
 ods.closeconection()  
 print("FIN DE CARGA DE DATOS A BONA\_HEALTH\_DW: ", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))  
 else:  
 print("NO SE PUDO CONECTAR A ", bonaheathl\_dw.database)  
 else:  
 print("NO SE PUDO CONECTAR A ", ods.database)

El programa, el cual se desarrollo en Pycharm, genera la siguiente salida:

Texto

Descripción generada automáticamente

Ilustración 24 Salida en consola del programa ETL

## Muestra de datos del Data Warehouse

A continuación, se mostrarán pantallazos los cuales evidencian que el Data Warehouse destinado para este proyecto se encuentra cargado, presentando las tablas de hechos por medio de PHPMyAdmin, en donde se evidencia por medio del color de los datos que estos están relacionados con las tablas de dimensiones:

### Fact\_asistencia\_medica

Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente

Ilustración 25 Muestra de datos Fact\_asistencia\_medica

### Fact\_defunciones\_fecha

Tabla

Descripción generada automáticamente

Ilustración 26 Muestra de datos Fact\_defunciones\_fecha

### Fact\_defunciones\_municipio

Tabla

Descripción generada automáticamente

Ilustración 27 Muestra de datos fact\_defunciones\_municipio

### Fact\_muertes

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Ilustración 28 Muestra de datos fact\_muertes

# CREACIÓN Y PUBLICACIÓN DEL CUBO

En esta sección se explicara todo el proceso realizado para el despliegue de un cubo OLAP basado en el Data Warehouse creado previamente.

## Entorno de trabajo

Para la realización de esta actividad, se emplearon herramientas de software presentadas por Pentaho, las cuales están destinadas para la gestión de los cubos, y como parte de esta actividad, se mostrará el proceso de descarga, despliegue y configuración de dichas herramientas.

### Pentaho Server

Esta herramienta nos permitirá la gestión de los cubos publicados y la manipulación de ciertas consultas. Esta herramienta esta desarrollada en Java y será manipulada por medio de un cliente al que se tendrá acceso por medio del navegador web. Para la descarga de esta herramienta, se debe acceder a la pagina de Source Force en la cual se encuentran los archivos destinados para la instalación de estas herramientas, en el caso de Pentaho Server, se debe acceder al siguiente enlace el cual descargara la herramienta en formato zip. (<https://sourceforge.net/projects/pentaho/files/Pentaho-9.2/server/pentaho-server-ce-9.2.0.0-290.zip/download>)

Tabla

Descripción generada automáticamente

Con este archivo comprimido descargado, se creará una carpeta en el directorio raíz del disco duro principal en la cual se almacenaran todas las herramientas de Pentaho, en este caso, se creo una carpeta llamada Pentaho y se descomprimirá el archivo dentro de esta carpeta:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Para asegurar que la herramienta se conecte apropiadamente a MySQL y la base de datos correspondiente al Data Warehouse, se requiere contar con el conector en formato Jar, en este caso, se empleó al archivo mysql-connector-java-5.1.49 descargado desde Maven Repository en el siguiente enlace: <https://repo1.maven.org/maven2/mysql/mysql-connector-java/5.1.49/mysql-connector-java-5.1.49.jar>

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Con este archivo descargado, se debe incluir en la carpeta Tomcat/ Lib dentro de Pentaho Server. Este conectar se empleará e la adecuación de las siguientes herramientas.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Antes de iniciar el servidor, se requiere contar con las variables de entorno, para esto, nos dirigimos a la carpeta principal de Pentaho Server y se ejecuto el archivo **set-pentaho-env.bat** el cual se encargará de implementar las variables de entorno requeridas.

Captura de pantalla de computadora

Descripción generada automáticamente

Finalmente, para iniciar el servidor y teniendo en cuenta todas las actividades previas para la configuración del entorno de este, se ejecutará el archivo **start-pentaho.bat** el cual desplegará una instancia de la consola en la cual se podrá evidenciar el inicio del servidor. Este proceso es demorado, pero hay que prestar atención a como este se comporta y las posibles alertas de error en su ejecución. Verifique que no cuente con otro servicio que este operando en el puerto 8080.

Texto

Descripción generada automáticamente

Para acceder al servidor, solo se requiere ingresar al siguiente enlace <http://localhost:8080/pentaho/Login> en el cual se encuentra el Login del servidor, y se elegirán las credenciales de acceso como administrador.

Gráfico

Descripción generada automáticamente

Finalmente, nos encontraremos en el Dashboard del servidor, lo cual indica, hasta el momento, que el proceso de despliegue de la herramienta se realizó correctamente. Se recomienda acceder al servidor solo después de que, en la ventana de consola, nos presente el mensaje **org.apache.catalina.startup.Catalina.start Server startup in [n] milliseconds.**

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

### Pentaho Schema Workbench

Esta herramienta de Pentaho será la encarga de permitirnos el modelado de los cubos a publicar en el servidor, por lo que esta herramienta requiere de la conexión con la base de datos correspondiente al Data Warehouse. Para la descarga de esta herramienta, nos dirigimos a la página correspondiente a las herramientas de Pentaho y descargamos el archivo comprimido desde el siguiente enlace <https://sourceforge.net/projects/pentaho/files/Pentaho-9.2/client-tools/psw-ce-9.2.0.0-290.zip/download> .

Tabla

Descripción generada automáticamente

El archivo comprimido descargado también se debe de descomprimir en la carpeta Pentaho previamente creada en el directorio raíz del disco duro principal, y como actividad final para el adecuamiento de la herramienta, se debe incluir el conector de MySQL dentro de las carpetas lib y drivers.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Finalmente, para iniciar la herramienta, nos dirigimos al directorio principal de Schema Workbench y ejecutaremos el archivo **workbench.bat**, el cual iniciara el cliente grafico de la herramienta y una ventana de conola de comandos de Windows, la cual no se debe cerrar mientras la herramienta se tenga abierta.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

### Pentaho Report Desginer

Esta herramienta nos permitirá realizar las actividades correspondientes al diseño de los reportes requeridos por la empres por medio de la creación de consultas y representaciones graficas de la información requerida. Para descargar esta herramienta, nos dirigiremos a la pagina de SourceForge de donde hemos descargado los anteriores recursos y descargaremos el archivo pdr-ce-9.2.0.0.290.zip del siguiente enlace <https://sourceforge.net/projects/pentaho/files/Pentaho-9.2/client-tools/prd-ce-9.2.0.0-290.zip/download> .

Tabla

Descripción generada automáticamente

Con el archivo comprimido descargado, lo descomprimiremos dentro de la carpeta Pentaho en el directorio raíz, y como actividad final para el adecuamiento de la herramienta, se debe incluir el conector de MySQL dentro de las carpetas lib y drivers.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Con estas adecuaciones. La herramienta estaría lista para su uso posteriormente.

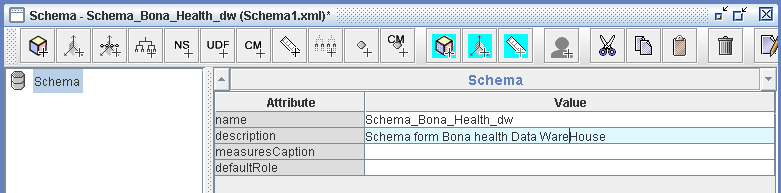
## Creación del cubo

Para la creación del cubo se empleará la herramienta Schema Workbech, y se realizara con cada una de las tablas de hechos definidas, de modo tal se crearán las dimensiones asacadas a cada una de estas con la jerarquía que las compone. Primero, se creará el esquema en el cual se contendrán los cubos asociados a las tablas de hechos, para esto se dará clic en File/ New / Schema.

Interfaz de usuario gráfica, Aplicación

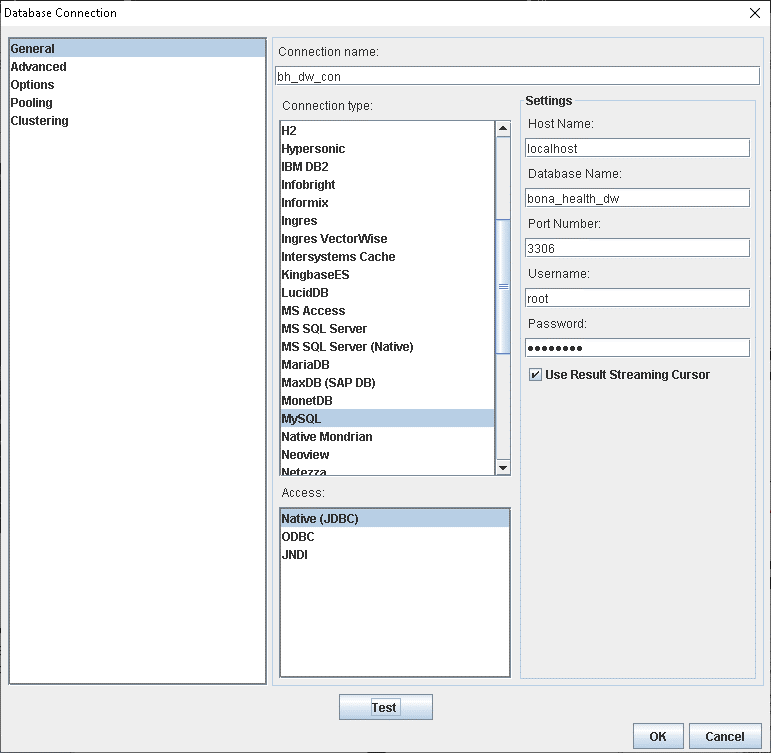
Descripción generada automáticamente

Con el esquema creado, se le dará un nombre a este, para esto se le da clic el Item **Schema** en la columna izquierda y en la ventana central, se ingresará el nombre del esquema y una descripción a este.

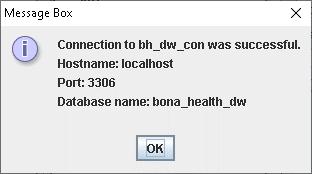


Luego, se requiere configurar la conexión con la base de datos con el fin de modelar cada uno de los cubos, para esto, no0s dirigiremos a Options / Connections y creamos una conexión con los siguientes datos:

* Connection Name: bh\_dw\_con
* Connection Type: MySQL
* Access: Native (JDBC)
* Hostname: localhost o dirección IP del equipo con la base datos.
* Database Name: bona\_health\_dw
* Port Number: 3306
* Username: root
* Password: contraseña del usuario root
* Use Result Streaming Cursor: debe estar marcado.



Con estos datos definidos, se realiza una prueba de la conexión a la base de datos correspondiente al Data Warehouse dando clic al botón test y debemos obtener la siguiente respuesta si ingresamos todos los datos correctamente. De lo contrario, por favor revisar los datos correspondientes a la base de datos en su equipo.

|

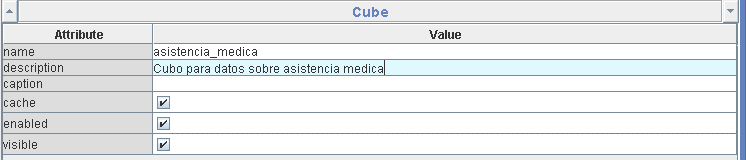
Para la creación de cad auno de los cubos se deben seguir los siguientes pasos:

1. Dentro del esquema, dar clic en el icono de cubo ubicado en la parte superior:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. En la ventana desplegada, definir el nombre del cubo el cual sera el nombre de la tabla de hechos que asociaremos al cubo.

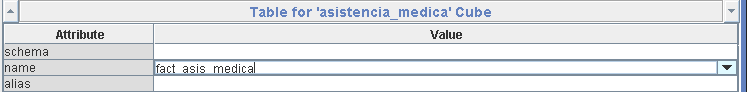


1. Para asociar al cubo recien creado a su tabla de hechos correspondiente, dar clic derecho sobre el cubo y seleccionar la opcion **Add Table**.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

En la ventana lateral, se define la tabla, para esto damos clic en la lista desplegable de name y seleccionamos el nombre de la tabla de hechos y definimos un alias a la tabla, la cual puede serl el mismo nombre de la tabla.



Con estos pasos definidos, se realizo la creación de cada uno de los cubos requeridos, mostrabdo acontinaucón la evidencia de estos, teniendo en cuenta que los erores que se reportan es debido a que no existen dimenciones aun en n ingun cubo.

Interfaz de usuario gráfica, Texto, Aplicación

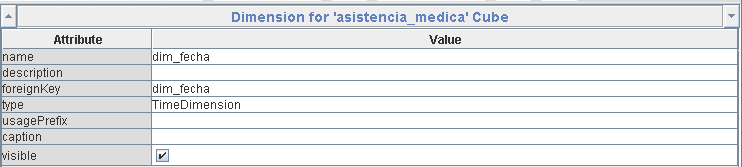
Descripción generada automáticamente

A continuación se mostrara la creación de cada una de las dimenciones, esto teniendo en cuenta que el proceso de creacion de las dimenciones es repetitivo y que solo se debe de replicar con base a las dimenciones involucradas en cada uno de los cubos.

### Dim\_fecha

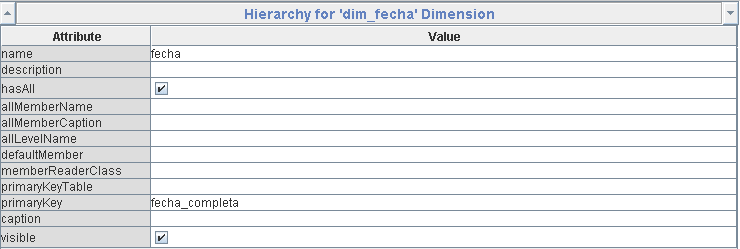
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: dim\_fecha
* **ForeingKey**: dim\_fecha
* **Type**: TimeDimesion

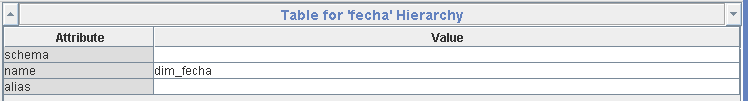


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: fecha
* **PrimaryKey**: dim\_fecha – fecha\_completa



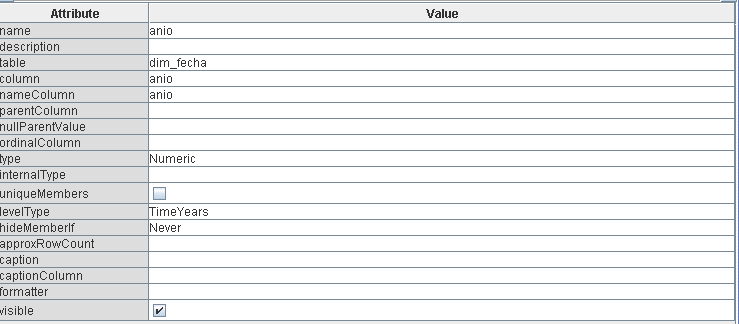
Con la jerarquía creada, se debe asociar la tabla correspondiente a esta, por lo que daremos clic derecho sobre la jerarquía y seleccionaremos la opción **Add Table**, en la ventana que se nos genera definiremos el dato name como dim\_fecha.



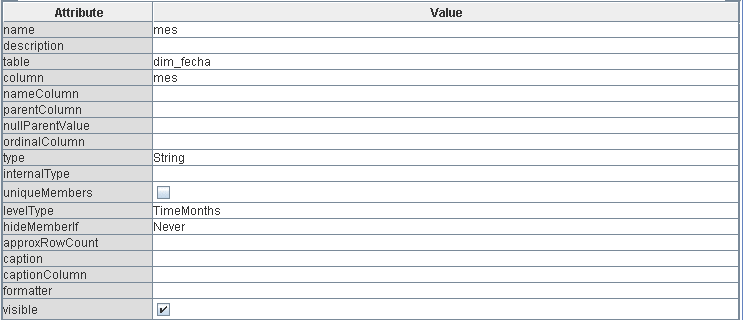
**Nota**: si Schema Workbench marca algún error en la creación de la jearquia con sus datos o no permite ingresar la Primary Key, se requiere crear la jerarquía vacía y luego añadirle la tabla, esto con el fin de poder acceder apropiadamente a los atributos de la tabla en cuestión.

A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso el año, hasta el menor nivel, que sería la fecha completa, a continuación, se mostrara la creación de los niveles de esta jerarquía:

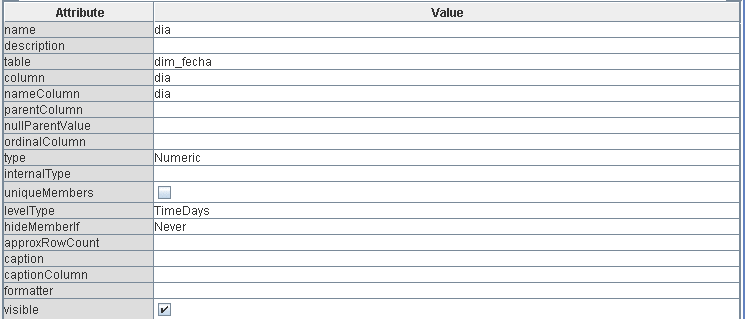
* Año: para este nivel se definen los siguientes datos:
  + Name: anio
  + Table: dim\_fecha\*
  + Column: anio
  + NameColumn: anio
  + Type: Numeric
  + LevelType: TimeYears



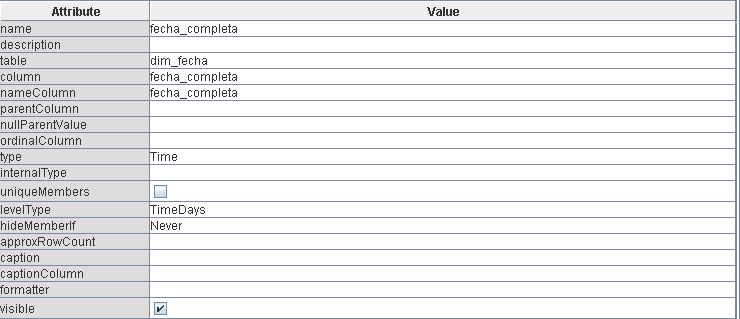
* Mes: para este nivel se definen los siguientes datos:
  + Name: mes
  + Table: dim\_fecha\*
  + Column: mes
  + NameColumn: mes
  + Type: String
  + LevelType: TimeMonths



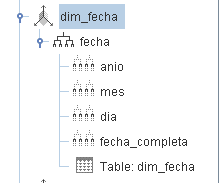
* Dia: para este nivel se definen los siguientes datos:
  + Name: día
  + Table: dim\_fecha\*
  + Column:dia
  + NameColumn: dia
  + Type: Numeric
  + LevelType: TimeDays



* Fecha\_completa: para este nivel se definen los siguientes datos:
  + Name: fecha\_completa
  + Table: dim\_fecha\*
  + Column: fecha\_completa
  + NameColumn: fecha\_completa
  + Type: Time
  + LevelType: TimeDays



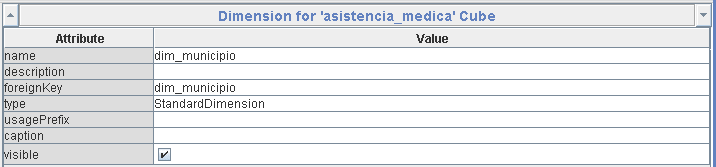
Con la dimensión, su respectiva jerarquía definida y los niveles de esta declarados, sin olvidar la asociación de esta jerarquía a la correspondiente tabla dentro del Data Warehouse de la dimensión, se obtiene la siguiente estructura para la dimensión de fecha:



### Dim\_Municipio

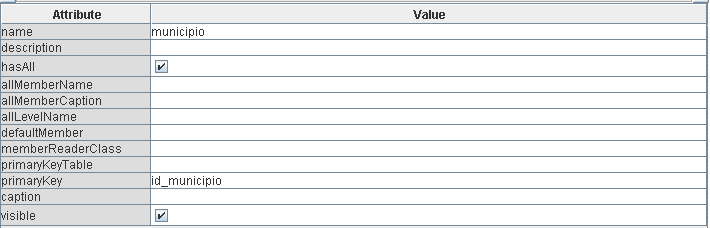
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: dim\_municipio
* **ForeingKey**: dim\_municipio
* **Type**: StandardDimesion

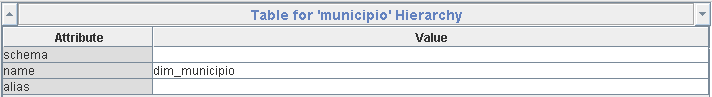


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: municipio
* **PrimaryKey**: dim\_municipio – id\_municipio

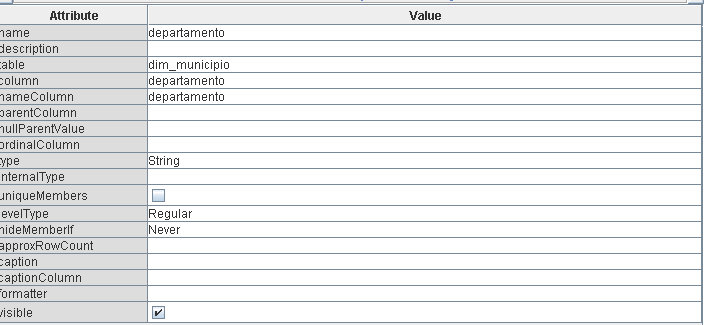


Con la jerarquía creada, se debe asociar la tabla correspondiente a esta, por lo que daremos clic derecho sobre la jerarquía y seleccionaremos la opción **Add Table**, en la ventana que se nos genera definiremos el dato name como dim\_municipio.

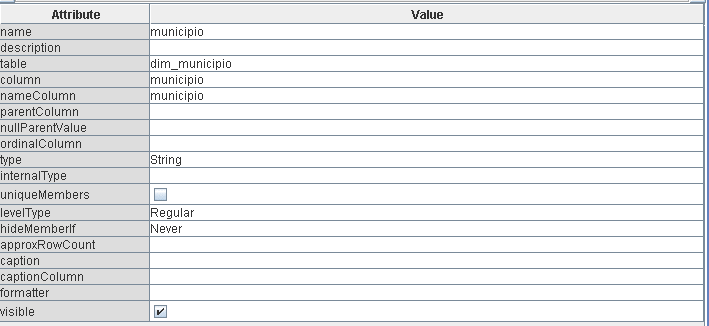


A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso el departamento, hasta el menor nivel, que sería id\_municipio, a continuación, se mostrara la creación de los niveles de esta jerarquía:

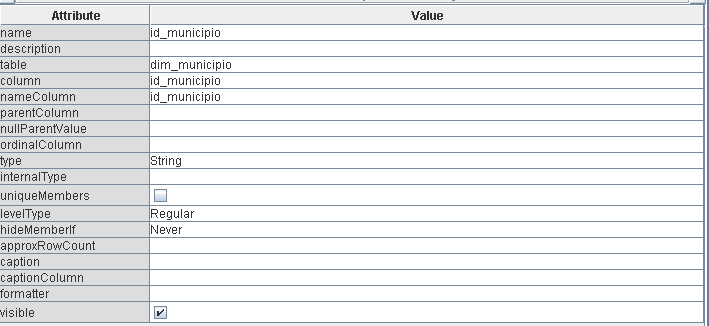
* Departamento: para este nivel se definen los siguientes datos:
  + Name: departamento
  + Table: dim\_municipio
  + Column: departamento
  + NameColumn: departamento
  + Type: String
  + LevelType: Regular



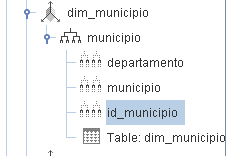
* Municipio: para este nivel se definen los siguientes datos:
  + Name: municipio
  + Table: dim\_municipio
  + Column: municipio
  + NameColumn: municipio
  + Type: String
  + LevelType: Regular



* Id\_ municipio: para este nivel se definen los siguientes datos:
  + Name: id\_ municipio
  + Table: dim\_ municipio
  + Column: id\_ municipio
  + NameColumn: id\_ municipio
  + Type: String
  + LevelType: Regular



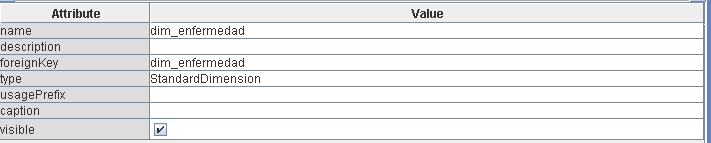
Con la dimensión, su respectiva jerarquía definida y los niveles de esta declarados, sin olvidar la asociación de esta jerarquía a la correspondiente tabla dentro del Data Warehouse de la dimensión, se obtiene la siguiente estructura para la dimensión de municipio:



### Dim\_Enfermedad

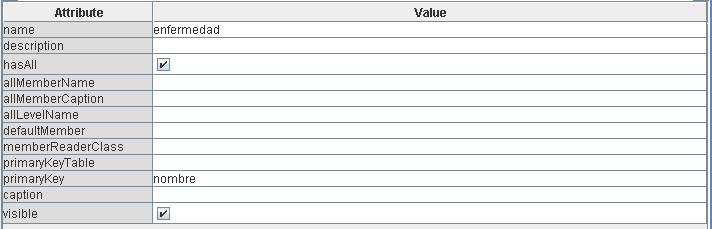
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: dim\_enfermedad
* **ForeingKey**: dim\_enfermedad
* **Type**: StandardDimesion

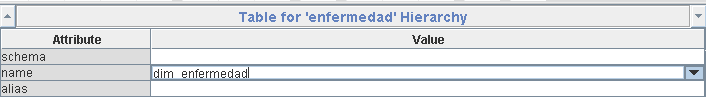


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: enfermedad
* **PrimaryKey**: dim\_enfermedad – nombre

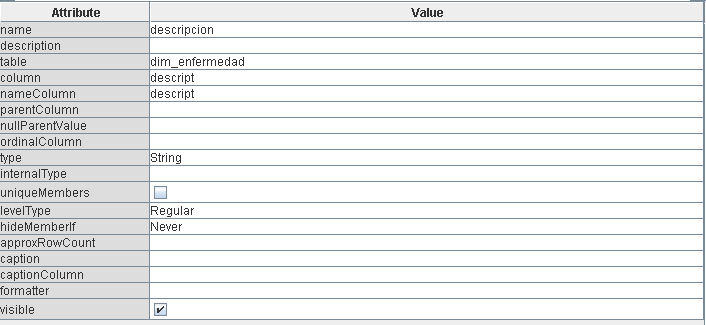


Con la jerarquía creada, se debe asociar la tabla correspondiente a esta, por lo que daremos clic derecho sobre la jerarquía y seleccionaremos la opción **Add Table**, en la ventana que se nos genera definiremos el dato name como dim\_enfermedad.

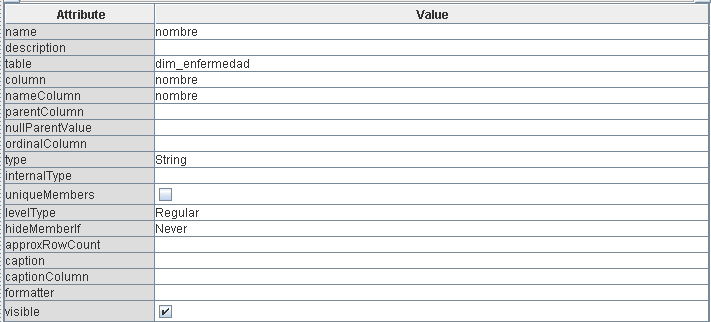


A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso la descripción, hasta el menor nivel, que sería el nombre de la enfermedad, a continuación, se mostrara la creación de los niveles de esta jerarquía:

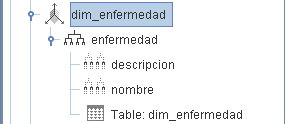
* Descripción: para este nivel se definen los siguientes datos:
  + Name: descripción
  + Table: dim\_enfermedad
  + Column: descrip
  + NameColumn: descrip
  + Type: String
  + LevelType: regular



* Nombre: para este nivel se definen los siguientes datos:
  + Name: nombre
  + Table: dim\_enfermedad
  + Column: nombre
  + NameColumn: nombre
  + Type: String
  + LevelType: regular



Con la dimensión, su respectiva jerarquía definida y los niveles de esta declarados, sin olvidar la asociación de esta jerarquía a la correspondiente tabla dentro del Data Warehouse de la dimensión, se obtiene la siguiente estructura para la dimensión de enfermedad:



### Ejemplo de creacion de una medida

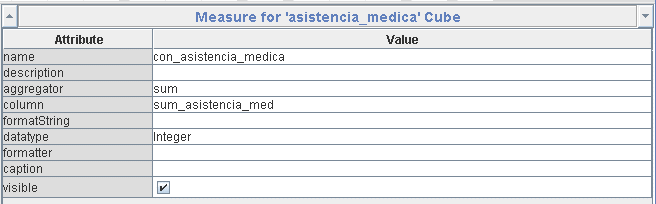
Antes de continuar con la especificacion de cada uno de los cubos, se requiere tener presente la creacion de medidas para cada uno de estos, este proceso de realiza dando clic derecho sobre el cubo correspondiente y seleccionando la opcion **Add Measure.**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Al dar clic en esta, se nos generara la medida y una ventana en la cual definimos los siguientes datos, en este caso, se creara la medida de Con\_asistencia\_medica del cubo asistencia medica.

* Name: con\_asistencia\_medica
* Aggregator: sum
* Column: sum\_asistencia\_med
* Datatype: integer

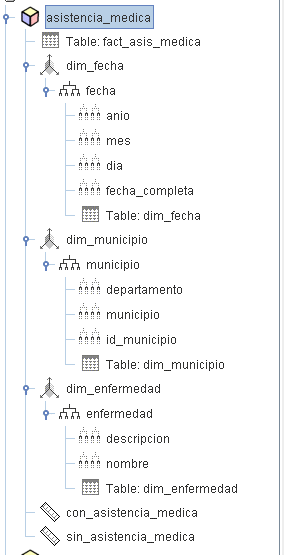


Este proceso se repitira con cada una de las medidas de cada una de las tablas de hechos definidas en la base de datos.

Con las dimenciones previamente definidas, teniendo presente la forma de creacion de las medidas, se continuara a definir cada uno de los cubos resultantes, las dimenciones que lo componen y sus rectivias medidas.

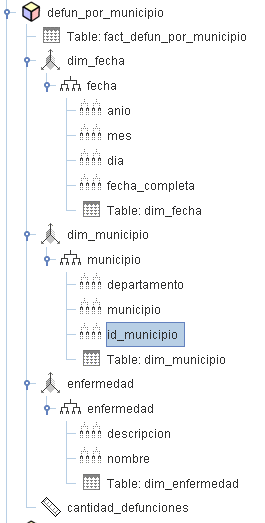
### Asistencia\_medica

En este cubo se emplearon las dimensiones de fecha, municipio y enfermedad, empleando las medidas de fallecidos con y sin asistencia, obteniendo la siguiente estructura al finalizar el modelado del cubo.



### Defun\_por\_municipio

En este cubo se emplearon las dimensiones de fecha, municipio y enfermedad, empleando la medida de cantidad de defunciones registradas, obteniendo la siguiente estructura al finalizar el modelado del cubo.



### Defunciones\_fecha

En este cubo se emplearon las dimensiones de fecha y enfermedad, empleando la medida de cantidad de defunciones, obteniendo la siguiente estructura al finalizar el modelado del cubo.

Tabla

Descripción generada automáticamente con confianza baja

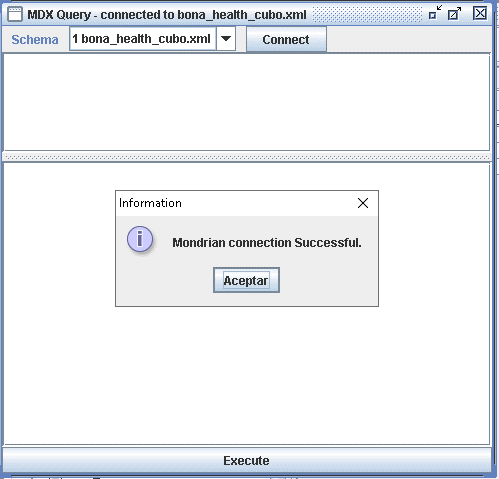
### Prueba

Con todos los cubos definidos correctamente, para saber que estos no generen ningún conflicto, se realizara una prueba de ejecución de consulta MDX, para esto, nos dirigimos al icono de New y elegiremos la opción MDX Query.

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Si el esquema logra sincronizarse correctamente, obtendremos el siguiente mensaje, de lo contrario, revisar la advertencia que el programa nos presente y realizar las modificaciones pertinentes.



Dentro de la ventana desplegada, se ejecutará la siguiente sentencia la cual retornará la cantidad de defunciones en el año 2010, si esta sentencia funciona correctamente, se podrá continuar con la publicación del cubo, de lo contrario, se realizarán las correcciones necesarias.

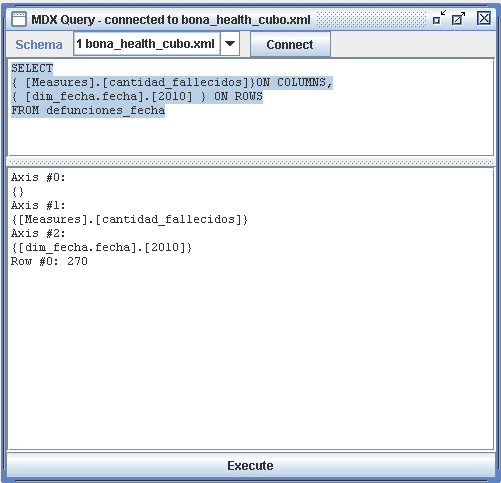
SELECT

{ [Measures].[cantidad\_fallecidos]}ON COLUMNS,

{ [dim\_fecha.fecha].[2010] } ON ROWS

FROM defunciones\_fecha

Al ejecutar esta sentencia se obtuvo una respuesta positiva al retornar los valores esperados:



### Publicación del cubo

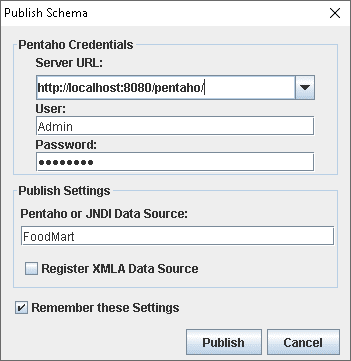
Finalmente, para realizar la publicación del cubo, se debe tener en cuenta que Pentaho server debe estar activo, para finalmente realizar la publicación del cubo, se debe realizar el siguiente proceso:

1. Clic en New / Publish

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. En la siguiente ventana verificaremos que el path de acceso a pentaho server sea el correcto, en el cual se encuentre el puerto 8080 definido, ingresaremos las credenciales de administrador (Admin – password) y daremos clic en Publish



1. Si el proceso de publicación del cubo se realizó correctamente, obtendremos el siguiente mensaje.

Interfaz de usuario gráfica, Texto, Aplicación

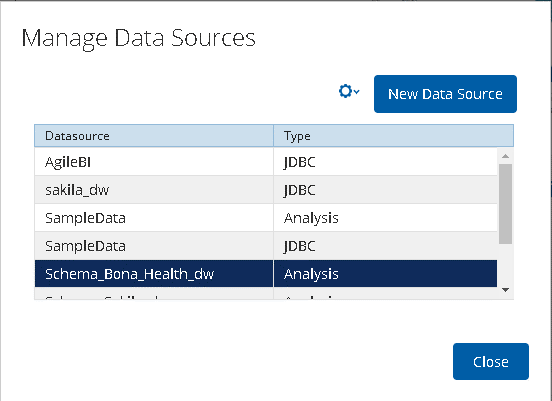
Descripción generada automáticamente

1. Para verificar finalmente que el cubo se encuentra disponible ya en el servidor, dentro de Pentaho Server, reiniciaremos el servidor, ingresaremos a este y daremos Clic en **Manage Data Source**.

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Dentro de la ventana que se nos despliegue buscaremos schema\_sakila\_dw, si lo encontramos significa que el proceso de publicación se realizó correctamente, de lo contrario, se recomienda reiniciar Pentaho Server y/o realizar de nuevo la publicación desde Schema Workbench.



En la siguiente parte, se realizará las adecuaciones necesarias para el despliegue de un Dashboard destinado para el cubo publicado, todo esto a través de Pentaho Server.

## Dashboard del cubo en Pentaho Server

Para poder desplegar operaciones de consulta y manipulación del cubo recién creado, Pentaho y su servidor presenta un entorno llamado Dashboard el cual nos permitirá realizar las tareas previamente mencionadas, para esto proceso, primero debemos realizar la configuración del entorno.

### Conexión al Data Warehouse

Es importante tener presente que el cubo que nosotros publicamos es la estructuración de la base de datos correspondiente al Data Warehouse, es una estructura vacía y para llenarla o asociarla a los datos, requerimos de incluir una conexión JDBC, para esto se realizaran los siguientes pasos:

1. Dentro del Home de Pentaho server, daremos clic en la opción de Manage Data Sources

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

1. Dentro de la ventana desplegada, daremos clic en el icono de configuración, y en la lista desplegada vamos a dar clic en la opción New Connection

Tabla

Descripción generada automáticamente

1. En la ventana desplegada, vamos a configurar la conexión a la base de datos correspondiente al Data Warehouse, para esto definimos los siguientes datos:
   * Connection name: Bona\_health\_dw
   * Database Type: MySQL
   * Access: Native (JDBC)
   * Host name: localhost
   * Database name. bona\_heatlh\_dw
   * Port number: 3306
   * Username: root
   * Password: contraseña del usuario root

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. Para comprobar que los parámetros ingresados son correctos, daremos clic en el botón Test y deberemos obtener la siguiente respuesta que verifique que todo está correcto.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Finalmente se da clic en Ok para guardar la conexión

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

1. El listado de Data Source se actualizará y en este encontraremos la conexión creada lista para ser usada.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

### Configuración del Schema

Con la conexión empleada y teniendo en cuenta que previamente ya hayamos publicado el cubo en el servidor, vamos a realizar una simple configuración al Schema la cual posibilitara el acceso a los datos dentro del Data Warehouse. Para esto, dentro del listado de Data Sources vamos a seleccionar el Schema del cubo, daremos click en el icono de configuración y elegiremos la opción de Edit.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Dentro de la ventana que se nos genera, vamos a definir la fuente de datos, para esto elegiremos la opción **Select from avaible data sources** y en la lista de data sources seleccionaremos a **Bona\_health\_dw**

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Con estos elementos definidos, daremos clic en **Save** y tendremos definida la fuente de datos correctamente asociada al Data Warehouse.

### Despliegue del Dashboard

Para realizar el despliegue del Dashboard del cubo, dentro de la pagina de home, daremos clic en la opción Create New:

Interfaz de usuario gráfica, Sitio web

Descripción generada automáticamente

Elegiremos la opción de crear un nuevo CDE Dashboard, el cual es un entorno en el cual se nos permitirá la manipulación y creación de consultas MDX sobre el cubo creado:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El proceso de carga del Dashboard en un poco demorado, pero al momento de que se nos despliegue la siguiente interfaz indicara que el proceso ha terminado. En esta interfaz daremos clic en el icono de Datasource Panel.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Dentro de la ventana que se nos genera nos dirigiremos a Wizards y daremos clic en la opción OLAP Selector Wizard.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Si todo el proceso de publicación y configuración del cubo se realizó correctamente, en la ventana que se nos genera, seleccionaremos del listado de Catalog el Schema de Bona Heatlh y debe desplegarnos los cubos, dimensiones y medidas asociados a este. En esta interfaz podremos realizar la manipulación grafica de consultas MDX sobre el cubo.

Interfaz de usuario gráfica, Aplicación, Teams

Descripción generada automáticamente

Con esto último terminaríamos con todo el proceso de despliegue de un cubo empleando la base de datos asociada al Data Warehouse.

# Creación de reportes

En esta sección se explicara el proceso de creación de reportes, los cuales permitan la visualización de los datos almacenados en el Data Warehouse teniendo en cuenta las necesidades de la organización usando cada uno de los cubos/tablas de hechos creados previamente.

## Reporte 1

## Reporte 2

## Reporte 3

# Tabla de ilustraciones

[Ilustración 1 Defunciones por VIH 5](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361515)

[Ilustración 2 Defunciones por cáncer de mama 5](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361516)

[Ilustración 3 Defunciones por cáncer de pulmón 5](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361517)

[Ilustración 4 Asistencia medida prestada a los fallecidos por VIH 6](#_Toc84361518)

[Ilustración 5 Atención medica prestada a los fallecidos por cáncer de mama 6](#_Toc84361519)

[Ilustración 6 Atención medica prestada a los fallecidos por cáncer de pulmón 7](#_Toc84361520)

[Ilustración 7 defunciones por VIH clasificadas por municipio 7](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361521)

[Ilustración 8 defunciones por cáncer de pulmón clasificadas por municipio 8](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361522)

[Ilustración 9 defunciones por cáncer de mama clasificadas por municipio 8](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361523)

[Ilustración 10 Muestra de datos del dataset de mortalidad por VIH 10](#_Toc84361524)

[Ilustración 11 Muestra de datos del dataset de mortalidad por cáncer del pulmón 11](#_Toc84361525)

[Ilustración 12 Muestra de datos del dataset de mortalidad por cáncer de mama 12](#_Toc84361526)

[Ilustración 13 Muestra de datos del dataset de departamentos y municipios de Colombia 19](#_Toc84361527)

[Ilustración 14 Modelo entidad relación de las fuentes de datos 22](file:///C:\Users\user\Downloads\Bona%20Health%20EPS.docx#_Toc84361528)

[Ilustración 15 Diseño de dimensiones y medidas 23](#_Toc84361529)

[Ilustración 16 Modelo CMDM – Defunciones por fecha 24](#_Toc84361530)

[Ilustración 17 Modelo CMDM – Asistencia médica 25](#_Toc84361531)

[Ilustración 18 Modelo CMDM – Defunciones por municipio 26](#_Toc84361532)

[Ilustración 19 Modelo CMDM – Muertes 26](#_Toc84361533)

[Ilustración 20 Muertes 30](#_Toc84361534)

[Ilustración 21 Modelo ROLAP 31](#_Toc84361535)

[Ilustración 22 Diagrama de Clases y componentes del sistema 36](#_Toc84361536)

[Ilustración 23 Secuencia de procesos ETL 37](#_Toc84361537)

[Ilustración 24 Salida en consola del programa ETL 57](#_Toc84361538)

[Ilustración 25 Muestra de datos Fact\_asistencia\_medica 58](#_Toc84361539)

[Ilustración 26 Muestra de datos Fact\_defunciones\_fecha 58](#_Toc84361540)

[Ilustración 27 Muestra de datos fact\_defunciones\_municipio 59](#_Toc84361541)

[Ilustración 28 Muestra de datos fact\_muertes 59](#_Toc84361542)