**EJERCICIO SAKILA**

**DATA WAREHOUSE**

**ANDRES SANCHEZ**

**LUIS FELIPE VELASCO TAO**

**JUAN DAVID GONZALEZ DIMATÉ**

**UNIVERSIDAD DE SAN BUENAVENTURA**

**FACULTAD DE INGENIERIA**

**INGENIERIA DE SISTEMAS**

**BOGOTÁ**

**14 DE SEPTIEMBRE**

**2021**

# EJERCICIO SAKILA

## Introducción

En este documento se busca realizar el análisis de una base de datos SAKILA que se utilizara como prototipo para el desarrollo de este caso práctico, al cual se le construirá o desarrollara una solución de Data Warehouse.

De esta manera se deberá presentar una posible solución de acuerdo con la base de datos entregada, con el fin de poner en práctica todos los conceptos y herramientas de Diseño Conceptual vistas en clase.

## Modelo de negocio

La empresa la cual se le desarrollará una solución de inteligencia de negocio será CLUBFIML S.A.

La empresa Sakila tiene como objetivo principal el alquiler de películas de videos, manejando una gran variedad de películas con el fin de satisfacer todas las necesidades de sus clientes.

La empresa Sakila tiene varias sucursales, donde sus clientes podrán encontrar las mejores películas del mercado distribuidas por, lenguaje, actor, título, categoría, de la mejor manera posible, para su fácil acceso dentro de sus tiendas.

El modelo de negocio de la empresa CLUBFILM S.A se compone de los siguientes procesos según el análisis de la base de datos:

* **Datos Clientes:** Donde se almacenan los datos de los clientes que por lo menos una vez, han prestado los servicios ofrecidos en las tiendas de CLUBFILM S.A.
* **Inventario:** Donde se almacena todos los datos sobre los productos (películas) que se tienen en las sucursales, para la administración y el buen funcionamiento de las tiendas.
* **Proceso del Negocio:** En este proceso del negocio es donde se encuentran los datos del negocio.

## Indicadores de negocio

* **Cantidad de venta (sales):** la empresa requiere conocer cuantas ventas se realizan esto en base a distintos factores como lo es las tiendas y el tiempo en que se realizan dichas compras de las películas, esto con el fin de tomar decisiones que ayuden al incremento de las ventas y el abastecimiento de las tiendas a tiempo.
* **Cantidad de películas en inventario (inventory):** con el fin de saber cuántas películas hay en inventario en cada tienda, se requiere para el correcto abastecimiento de películas de conocer dicha cantidad.
* **Cantidad de rentas (rentals):** la empresa requiere conocer la cantidad de rentas y los horarios en que se realizan en cada una de las tiendas.

## Identificación de Tablas de la fuente

La empresa nos entrega un listado con las tablas de la fuente de datos que se emplearan en la construcción e implementación del almacén de datos, clasificando que tablas son en las cuales se encuentran las actividades relacionadas con los indicadores del negocio, en las cuales se podrá evidenciar la actividad comercial de la empresa, dichas tablas son:

* **Payment-sales**: en esta tabla se almacenan las ventas de las películas realizadas en cada una de las tiendas.
* **Rental**: en esta tabla se almacenan el alquiler de las películas realizadas en cada una de las tiendas.
* **Inventory**: en esta tabla se almacenan la revisión de las películas que hay en cada una de las tiendas.

Por otro lado, se contemplan como tablas no transaccionales en las cuales se almacenan los datos necesarios para dejar evidencia de las actividades económicas de la empresa en las tablas transaccionales, estas tablas son:

* **Customer**: en esta tabla se almacenan los datos relacionados con los clientes que compran y rentan películas.
* **Address**: en esta tabla se almacenan las direcciones relacionadas con las tiendas, trabajadores y los clientes de la empresa.
* **City**: en esta tabla se almacenan las ciudades relacionadas con las direcciones almacenadas en la tabla address.
* **Country**: en esta tabla se almacenan los países relacionados con las ciudades almacenadas en la tabla city.
* **Staff**: en esta tabla se almacenan los datos de los trabajadores o empleados de la empresa, contemplando sus datos personales y credenciales de acceso.
* **Store**: en esta tabla se almacenan los datos sobre el encargado de la tienda y la dirección de esta.
* **Film**: en esta tabla se almacenan los datos relacionados con cada una de las películas que posee la empresa para su venta o renta, contemplando datos como el idioma de la película, duración, descripción, entre otros.
* **Film\_category**: en esta tabla se relacionarán las películas con las categorías almacenadas, esto contemplando el hecho de que una película puede estar catalogada en varias categorías.
* **Category**: en esta tabla se almacenan las categorías a las cuales se pueden asociar las películas disponibles para su venta o renta.
* **Language**: en esta tabla se almacenan los datos relacionados con el idioma de las películas que posee la empresa.
* Actor: en esta tabla se almacenan los datos relacionados con los actores participes de las películas que posee la empresa.
* **Film\_actor**: en esta tabla se relacionan las películas con los actores registrados en la base de datos.
* **Film\_text**: en esta tabla se almacena la descripción de la película al momento de realizar el inventario.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 1 Identificación de las tablas transaccionales y no transaccionales

La mayoría de las tablas contienen un campo por medio del cual se evidencia la fecha en la que el registro en cuestión fue almacenado, de modo tal, al definir las fechas a emplear en el almacén de datos, se debe diferenciar estos campos de los cuales si definen el flujo de la actividad de la empresa. A continuación, se vera la forma en que las tablas anteriormente mencionadas se encuentran relacionadas entre sí, para de esta forma comprender el funcionamiento del negocio y así comenzar el correcto diseño del almacén de datos.

## Modelo Entidad relación de la Fuente

La empresa nos entregó la base de datos a emplear en el despliegue del almacén de datos, de modo tal se puedan observar las relaciones entre las tablas y comprender de mejor forma la naturaleza del negocio y la forma por medio de la cual se podrán obtener los indicadores del negocio.

Este modelo se divide en tres apartados los cuales tienen una función específica en el negocio:

* **Customer Data**: se almacenan los datos relacionados con el cliente, estructurando en la tabla customer los datos básicos del cliente y haciendo uso de la tabla address y sus relaciones para definir la ubicación de residencia del cliente. En base a esta ultima tabla, se va a definir la ubicación de las tiendas de la empresa en el apartado de Business.
* **Inventory**: en este apartado se define todo lo relacionado con los datos de las películas como lo es su categoría, lenguaje y actores, resultando en el indexado de las películas por tienda en la tabla inventory.
* **Business**: en este apartado nos encontraremos con la actividad comercial de la empresa, en la cual se consignan los datos de los empleados, tiendas y todos los registros relacionados con la compra y alquiler de las películas, de modo tal este apartado reúne los datos consignados de los clientes y las películas definidos en los apartados de Inventory y Customer Data.

Escala de tiempo

Descripción generada automáticamente

Ilustración 2 Modelo entidad-relación de la fuente de datos de la empresa

Con base a la definición y análisis de la fuente de datos, los indicadores del negocio y la actividad que la empresa desempeña, se realizara el diseño del almacén 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 contemplarán 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.

Diagrama

Descripción generada automáticamente

Ilustración 3 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:

* **Film**: se definen los datos de las películas que vende, renta y tiene en inventario, usando los datos del **nombre**, **categoría** e **idioma** de las películas para ordenar los registros de las películas.
* **Store**: se contemplarán los datos de la ubicación de las tiendas de la empresa, de modo tal se conocerá el país, ciudad y dirección de cada sucursal de la empresa.
* **Staff**: hace relación a los empleados de cada una de las sucursales de la empresa, esto con el fin de conocer la cantidad de ventas y alquiler de películas que ha realizado cada uno, por lo que se requiere conocer la tienda en la que trabaja cada empleado y su nombre.
* **Customer**: para conocer a los compradores o clientes de la empresa, se requiere conocer datos sobre su ubicación y su estado.
* **Time**: para la gestión de todos los registros, se requiere conocer cuando ocurrieron las ventas, alquileres y revisión del inventario.

## 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 - Ventas

Para conocer cómo se comportan las ventas y tomar decisiones con relación a como estas se dan se requiere conocer al cliente, la tienda en donde se realizan las ventas y los empleados que las realizan, esto siempre contemplando la dimensión del tiempo, esto con el fin de saber en qué horarios se realizan más o menos ventas en cada una de las tiendas.

Diagrama

Descripción generada automáticamente

Ilustración 4 Modelo CMDM – Ventas

Para este modelo se definieron las dimensiones de Customer, Time, Store y Staff las cuales se unirán a en la tabla de hechos de Sales que tendrá como medida la cantidad de ventas realizadas.

### Modelo CMDM - Rentas

Para conocer la forma en que se comporta el alquiler de películas se requiere conocer al cliente, las tiendas a las que estos acuden para alquilar películas, las películas que alquilan, los empleados que realizan el alquiler de las películas y los horarios en que se realiza dicho alquiler, esto con el fin de conocer información de cuales películas son las que más se alquilan, los días o horas en que se realiza más el alquiler de películas y demás información de utilidad que ayude a la empresa a mejorar su actividad comercial.

Diagrama

Descripción generada automáticamente

Ilustración 5 Modelo CMDM – Rentas

Para este modelo se definieron las dimensiones de Customer, Time, Store, Staff y Film las cuales se unirán a en la tabla de hechos de Rentals que tendrá como medida la cantidad de adquiéreles de películas realizados.

### Modelo CMDM - Inventario

Para conocer el inventario de las tiendas, se requiere tener presente las películas que maneja la empresa, a las tiendas en donde estas películas se venden o alquilan y el tiempo, esto con el fin de tomar decisiones relacionadas con el abastecimiento de las tiendas. En este cubo se deben contemplar las dimensiones de time, Store y Film, con el fin de definir las fechas en que se realiza el inventario de las películas en cada una de las tiendas, teniendo en la tabla de hechos de Inventory la medida de la cantidad de películas.

Diagrama

Descripción generada automáticamente

Ilustración 6 Modelo CMDM - Inventario

## Modelo Multidimensional

Con la definición de cada uno de los cubos, las dimensiones y medidas involucradas en cada uno de estos, se define el siguiente modelo multidimensional el cual se definen todas las relaciones que tendrán las tablas de hechos definidas en color rojo e iniciando por la palabra “fact”, y por otro lado las tablas de dimensiones, definidas en color verde y las cuales inician por la palabra “dim”, en las cuales se definen las jerarquías especificadas en el diseño de dimensiones y medidas.

**Aclaración**: debido a que en la dimensión de staff se requiere conocer la tienda a la que pertenece el empleado en cuestión, se definió la relación que permita el acceso a este elemento, y a su vez, que las tablas de hechos que requieran hacer uso de la dimensión de Staff tengan una relación con esta.

Diagrama, Esquemático

Descripción generada automáticamente

Ilustración 7 Modelo Multidimensional

## 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.

### Modelo DF – Sales

En base a la definición de miembros de las dimensiones de Date/Time, Customer y Store, se obtuvieron las siguientes jerarquías para cada una de las dimensiones y se define la tabla de hechos fact\_sales en la cual se encuentra la medida de cantidad de ventas.

Diagrama

Descripción generada automáticamente con confianza media

Ilustración 8 Modelo DF – Sales

**Aclaración**: debido a la relación definida entre las dimensiones de Staff y Store, se puede ver como en los cubos en que se requiere de la dimensión de staff se hace esta un miembro de la jerarquía de la dimensión de store, debido a como un empleado en parte de n tiendas. Por otro lado, se define la jerarquía de la dimensión Customer de modo tal se pueda clasificar al cliente en base a su ubicación y su estado como cliente.

### Modelo DF – Rentals

En base a la definición de miembros de las dimensiones de Date/Time, Customer, Film y Store, se obtuvieron las siguientes jerarquías para cada una de las dimensiones y se define la tabla de hechos fact\_sales en la cual se encuentra la medida de cantidad de alquileres. Como particularidad, en la dimensión de Film se puede ver como a partir del nombre de una película se puede clasificar en base a su categoría y/o idioma, caso muy similar al definido en la dimensión de Customer en la que se clasifican los clientes por su ubicación y estado como cliente.

Diagrama

Descripción generada automáticamente con confianza media

Ilustración 9 Modelo DF – Rentals

### Modelo DF – Inventory

En base a la definición de miembros de las dimensiones de Date/Time, Film y Store, se obtuvieron las siguientes jerarquías para cada una de las dimensiones y se define la tabla de hechos fact\_sales en la cual se encuentra la medida de cantidad de alquileres. Como particularidad, en la dimensión de Film se puede ver como a partir del nombre de una película se puede clasificar en base a su categoría y/o idioma.

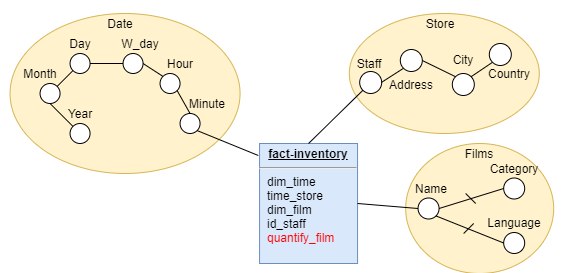


Ilustración 10 Modelo DF – Inventory

# 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 cual se realizó con base a todo el proceso de diseño conceptual desarrollado previamente.

Diagrama

Descripción generada automáticamente

Ilustración 7 Modelo ROLAP de SAKILA

## DDL de Data Warehouse

Con el modelo previamente expuesto se generaron las sentencias DDL que permitan la creación de la base de datos y de cada una de las tablas que representaran a las dimensiones, bridges y facts que componente al Data Warehouse.

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

-- Schema sakila\_dw

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

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

-- Schema sakila\_dw

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

CREATE SCHEMA IF NOT EXISTS ` sakila\_dw` DEFAULT CHARACTER SET utf8 ;

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

-- Schema sakila\_dw

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

USE ` sakila\_dw` ;

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

-- Table ` sakila\_dw`.`dim\_film`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_film` (

`id\_dim\_film` INT NOT NULL,

`title` VARCHAR(45) NULL,

`language` VARCHAR(45) NULL,

PRIMARY KEY (`id\_dim\_film`))

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_time`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_time` (

`complete\_date` DATE NOT NULL,

`year` INT NULL,

`month` VARCHAR(45) NULL,

`day` INT NULL,

`week\_day` VARCHAR(45) NULL,

PRIMARY KEY (`complete\_date`))

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_store`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_store` (

`id\_dim\_store` INT NOT NULL,

`country` VARCHAR(45) NULL,

`city` VARCHAR(45) NULL,

`address` VARCHAR(45) NULL,

PRIMARY KEY (`id\_dim\_store`),

UNIQUE INDEX `address\_UNIQUE` (`address` ASC) )

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`fact\_inventory`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`fact\_inventory` (

`dim\_film` INT NOT NULL,

`dim\_time` DATE NOT NULL,

`dim\_store` INT NOT NULL,

`quantity\_film` INT(6) NOT NULL DEFAULT 1,

INDEX `fk\_fact\_inventory\_dim\_film\_idx` (`dim\_film` ASC) ,

INDEX `fk\_fact\_inventory\_dim\_time1\_idx` (`dim\_time` ASC) ,

INDEX `fk\_fact\_inventory\_dim\_store1\_idx` (`dim\_store` ASC) ,

PRIMARY KEY (`dim\_store`, `dim\_time`, `dim\_film`),

CONSTRAINT `fk\_fact\_inventory\_dim\_film`

FOREIGN KEY (`dim\_film`)

REFERENCES ` sakila\_dw`.`dim\_film` (`id\_dim\_film`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_inventory\_dim\_time1`

FOREIGN KEY (`dim\_time`)

REFERENCES ` sakila\_dw`.`dim\_time` (`complete\_date`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_inventory\_dim\_store1`

FOREIGN KEY (`dim\_store`)

REFERENCES ` sakila\_dw`.`dim\_store` (`id\_dim\_store`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_customer`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_customer` (

`id\_dim\_customer` INT NOT NULL,

`name` VARCHAR(45) NULL,

`status` TINYINT NULL,

`address` VARCHAR(45) NULL,

`city` VARCHAR(45) NULL,

`country` VARCHAR(45) NULL,

PRIMARY KEY (`id\_dim\_customer`))

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_staff`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_staff` (

`id\_dim\_staff` INT NOT NULL,

`name` VARCHAR(45) NULL,

`id\_store` INT NULL,

PRIMARY KEY (`id\_dim\_staff`),

INDEX `fk\_dim\_staff\_dim\_store1\_idx` (`id\_store` ASC) ,

CONSTRAINT `fk\_dim\_staff\_dim\_store1`

FOREIGN KEY (`id\_store`)

REFERENCES ` sakila\_dw`.`dim\_store` (`id\_dim\_store`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`fact\_rentals`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`fact\_rentals` (

`dim\_customer` INT NOT NULL,

`dim\_film` INT NOT NULL,

`dim\_store` INT NOT NULL,

`dim\_time` DATE NOT NULL,

`dim\_staff` INT NOT NULL,

`quantity\_rentals` INT(6) NOT NULL DEFAULT 1,

INDEX `fk\_fact\_rentals\_dim\_customer1\_idx` (`dim\_customer` ASC) ,

INDEX `fk\_fact\_rentals\_dim\_film1\_idx` (`dim\_film` ASC) ,

INDEX `fk\_fact\_rentals\_dim\_store1\_idx` (`dim\_store` ASC) ,

INDEX `fk\_fact\_rentals\_dim\_time1\_idx` (`dim\_time` ASC) ,

INDEX `fk\_fact\_rentals\_dim\_staff1\_idx` (`dim\_staff` ASC) ,

PRIMARY KEY (`dim\_staff`, `dim\_time`, `dim\_store`, `dim\_film`, `dim\_customer`),

CONSTRAINT `fk\_fact\_rentals\_dim\_customer1`

FOREIGN KEY (`dim\_customer`)

REFERENCES ` sakila\_dw`.`dim\_customer` (`id\_dim\_customer`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_rentals\_dim\_film1`

FOREIGN KEY (`dim\_film`)

REFERENCES ` sakila\_dw`.`dim\_film` (`id\_dim\_film`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_rentals\_dim\_store1`

FOREIGN KEY (`dim\_store`)

REFERENCES ` sakila\_dw`.`dim\_store` (`id\_dim\_store`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_rentals\_dim\_time1`

FOREIGN KEY (`dim\_time`)

REFERENCES ` sakila\_dw`.`dim\_time` (`complete\_date`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_rentals\_dim\_staff1`

FOREIGN KEY (`dim\_staff`)

REFERENCES ` sakila\_dw`.`dim\_staff` (`id\_dim\_staff`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`fact\_sales`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`fact\_sales` (

`dim\_customer` INT NOT NULL,

`dim\_time` DATE NOT NULL,

`dim\_store` INT NOT NULL,

`dim\_staff` INT NOT NULL,

`quantity\_sales` INT(6) NOT NULL DEFAULT 1,

INDEX `fk\_fact\_sales\_dim\_customer1\_idx` (`dim\_customer` ASC) ,

INDEX `fk\_fact\_sales\_dim\_time1\_idx` (`dim\_time` ASC) ,

INDEX `fk\_fact\_sales\_dim\_store1\_idx` (`dim\_store` ASC) ,

INDEX `fk\_fact\_sales\_dim\_staff1\_idx` (`dim\_staff` ASC) ,

PRIMARY KEY (`dim\_customer`, `dim\_time`, `dim\_store`, `dim\_staff`),

CONSTRAINT `fk\_fact\_sales\_dim\_customer1`

FOREIGN KEY (`dim\_customer`)

REFERENCES ` sakila\_dw`.`dim\_customer` (`id\_dim\_customer`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_sales\_dim\_time1`

FOREIGN KEY (`dim\_time`)

REFERENCES ` sakila\_dw`.`dim\_time` (`complete\_date`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_sales\_dim\_store1`

FOREIGN KEY (`dim\_store`)

REFERENCES ` sakila\_dw`.`dim\_store` (`id\_dim\_store`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_fact\_sales\_dim\_staff1`

FOREIGN KEY (`dim\_staff`)

REFERENCES ` sakila\_dw`.`dim\_staff` (`id\_dim\_staff`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_category`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_category` (

`id\_category` TINYINT NOT NULL,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`id\_category`))

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`bridge\_film\_category`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`bridge\_film\_category` (

`id\_category` TINYINT NOT NULL,

`id\_dim` INT NOT NULL,

INDEX `fk\_bridge\_film\_category\_dim\_category1\_idx` (`id\_category` ASC) ,

INDEX `fk\_bridge\_film\_category\_dim\_film1\_idx` (`id\_dim` ASC) ,

CONSTRAINT `fk\_bridge\_film\_category\_dim\_category1`

FOREIGN KEY (`id\_category`)

REFERENCES ` sakila\_dw`.`dim\_category` (`id\_category`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_bridge\_film\_category\_dim\_film1`

FOREIGN KEY (`id\_dim`)

REFERENCES ` sakila\_dw`.`dim\_film` (`id\_dim\_film`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`dim\_actor`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`dim\_actor` (

`id\_actor` TINYINT NOT NULL,

`name` VARCHAR(45) NULL,

PRIMARY KEY (`id\_actor`))

ENGINE = InnoDB;

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

-- Table ` sakila\_dw`.`bridge\_film\_actor`

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

CREATE TABLE IF NOT EXISTS ` sakila\_dw`.`bridge\_film\_actor` (

`id\_actor` TINYINT NOT NULL,

`id\_film` INT NOT NULL,

INDEX `fk\_bridge\_film\_customer\_dim\_actor1\_idx` (`id\_actor` ASC) ,

INDEX `fk\_bridge\_film\_customer\_dim\_film1\_idx` (`id\_film` ASC) ,

CONSTRAINT `fk\_bridge\_film\_customer\_dim\_actor1`

FOREIGN KEY (`id\_actor`)

REFERENCES ` sakila\_dw`.`dim\_actor` (`id\_actor`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_bridge\_film\_customer\_dim\_film1`

FOREIGN KEY (`id\_film`)

REFERENCES ` sakila\_dw`.`dim\_film` (`id\_dim\_film`)

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, y el programa desarrollado en Python en el cual se integrará el diseño previamente expuesto, además de mostrar la evidencia del Data Warehouse cargado.

## Diseño ETL

Antes de desarrollar el componente de software destinado para los procesos de extracción de la información desde la base de datos de Sakila en donde se encuentra los datos relacionados con la actividad de la empresa, transformación o limpieza de los datos y carga de los datos, se realizó 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.

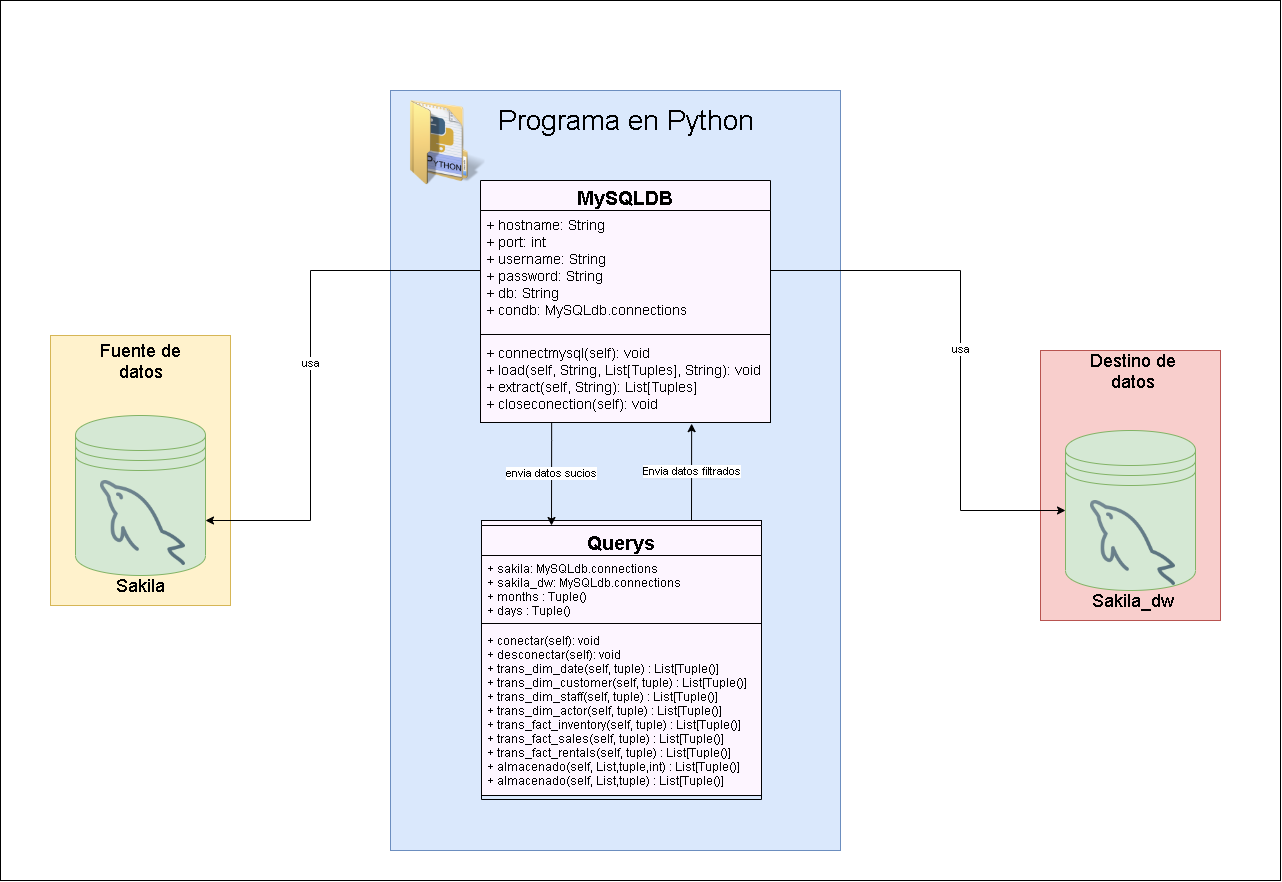


Ilustración 8 Diseño Programa ETL

## Secuencia ETL

Con un diseño previo del programa, en el cual se contemplan las clases, sus variables y métodos a implementar, se define la siguiente secuencia de funcionamiento del programa el cual contemple el respeto a las dependencias de las tablas que representan las dimensiones, hechos y bridges del data Warehouse, por lo que se define primero la conexión a la fuente de datos y al Data Warehouse y los procesos de extracción de datos desde la base de datos Sakila, las transformaciones requeridas y la carga a la tabla correspondiente en el Data Warehouse, para finalmente cerrar las conexiones realizadas a la fuente de datos y el Data Warehouse.

Interfaz de usuario gráfica

Descripción generada automáticamente

Ilustración 9 Secuencia del programa ETL

## Programa ETL

Con todo el proceso de diseño del programa encargado de los ETL y la definición de la secuencia de actividades a seguir por este, se creo el siguiente programa en Python en su versión 3.9, en el cual se puede ver como se implementaron todos los elementos previamente expuestos.

from datetime import datetime

import mysql.connector

class MysqlDB:

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

def connectmysql(self):

try:

self.condb = mysql.connector.connect(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)

def closeconection(self):

try:

self.condb.close()

print('Desconexión exitosa a ' + self.database)

except Exception as e: \

print("Error de desconexión", e)

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)

def load(self, c, tabla, sql):

cursori = self.condb.cursor()

try:

cursori.executemany("INSERT IGNORE INTO " + sql, c)

print("TABLA ", tabla, " CARGADA CON ", len(c),

" 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()

class Querys:

def \_\_init\_\_(self):

## Conexion a las bases de datos

self.sakila = MysqlDB('localhost', 3306, 'root', 'password', 'sakila')

self.sakila\_dw = MysqlDB('localhost', 3306, 'root', 'password', 'sakila\_dw')

self.months = (

"Enero", "Febrero", "Marzo", "Abri", "Mayo", "Junio", "Julio", "Agosto", "Septiembre", "Octubre",

"Noviembre",

"Diciembre")

self.days = ("Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo")

def desconectar(self):

self.sakila.closeconection()

self.sakila\_dw.closeconection()

def conectar(self):

self.sakila.connectmysql()

self.sakila\_dw.connectmysql()

def trans\_dim\_date(self, c):

l, f = [], []

for i in c:

a = i[0].strftime('%Y-%m-%d')

if a not in f:

f.append(a)

aux = (a, i[0].year, self.months[i[0].month - 1], i[0].day, self.days[i[0].weekday() - 1])

l.append(aux)

return l

def trans\_dim\_customer(self, c):

l = []

for i in c:

name = i[1] + " " + i[2]

aux = (i[0], name, i[3], i[4], i[5])

l.append(aux)

return l

def trans\_dim\_staff(self, c):

l = []

for i in c:

aux = (i[0], i[1] + " " + i[2], i[3])

l.append(aux)

return l

def trans\_dim\_actor(self, c):

l = []

for i in c:

aux = (i[0], i[1] + " " + i[2])

l.append(aux)

return l

def trans\_fact\_inventory(self, c):

l = []

for i in c:

a = i[0].strftime('%Y-%m-%d')

aux = [a, i[1], i[2]]

p = self.almacenado\_inv(l, aux)

if p == -1:

aux.append(1)

l.append(aux)

else:

a = l[p]

a[3] = a[3] + 1

l[p] = a

return l

def trans\_fact\_sales(self, c):

l = []

for i in c:

a = i[1].strftime('%Y-%m-%d')

aux = [i[0], a, i[2], i[3], 1, i[4]]

p = self.almacenado(l, aux, 4)

if p == -1:

l.append(aux)

else:

a = l[p]

a[4] = a[4] + 1

a[5] = a[5] + aux[5]

l[p] = a

return l

def almacenado(self, l, c, r):

x = 0

for i in l:

k = 0

for j in range(r):

if i[j] == c[j]:

k = k + 1

if k == r:

return x

x = x + 1

return -1

def almacenado\_inv(self, l, c):

x = 0

for i in l:

k = 0

for j in range(1, 3):

if i[j] == c[j]:

k = k + 1

if k == 2:

return x

x = x + 1

return -1

def trans\_fact\_rentals(self, c):

l = []

for i in c:

a = i[3].strftime('%Y-%m-%d')

aux = [i[0], i[1], i[2], a, i[4]]

p = self.almacenado(l, aux, 5)

if p == -1:

aux.append(1)

l.append(aux)

else:

a = l[p]

a[5] = a[5] + 1

l[p] = a

return l

if \_\_name\_\_ == '\_\_main\_\_':

## Creacion de las conexiones a la base de datos

print("INICIO DE CARGA DE DATOS A SAKILA\_DW: ", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))

## Inicio de clase dedicada a las consultas e inserciones

query = Querys()

query.conectar()

### Ejecucion de las sentencias SQL

## DIM\_TIME

c = query.sakila.extract("payment\_date FROM payment;") + query.sakila.extract("rental\_date FROM rental;") + query.sakila.extract(

"last\_update FROM inventory;")

c = query.trans\_dim\_date(c)

query.sakila\_dw.load(c, "DIM\_TIME", "dim\_time (complete\_date, year, month, day, week\_day) VALUES (%s, %s, %s, %s, %s)")

## DIM\_CUSTOMER

c = query.sakila.extract(

"customer.customer\_id,customer.first\_name,customer.last\_name,customer.active , city.city, country.country FROM customer "

"INNER JOIN address ON customer.address\_id = address.address\_id "

"INNER JOIN city ON address.city\_id = city.city\_id "

"INNER JOIN country ON city.country\_id = country.country\_id;")

c = query.trans\_dim\_customer(c)

query.sakila\_dw.load(c, "DIM\_CUSTOMER",

"dim\_customer (id\_customer, name, status, city, country) VALUES (%s, %s, %s, %s, %s)")

## DIM\_STORE

c = query.sakila.extract(

"store.store\_id , address.address, city.city , country.country FROM store "

"INNER JOIN address ON store.address\_id = address.address\_id "

"INNER JOIN city ON address.city\_id = city.city\_id "

"INNER JOIN country ON city.country\_id = country.country\_id;")

query.sakila\_dw.load(c, "DIM\_STORE", "dim\_store (id\_store, address, city, country) VALUES (%s, %s, %s, %s)")

## DIM\_STAFF

c = query.sakila.extract("staff.staff\_id, staff.first\_name, staff.last\_name, staff.store\_id FROM staff;")

c = query.trans\_dim\_staff(c)

query.sakila\_dw.load(c, "DIM\_SATFF", "dim\_staff (id\_staff, name, id\_store) VALUES (%s, %s, %s)")

## DIM\_ACTOR

c = query.sakila.extract("actor\_id, first\_name, last\_name FROM actor;")

c = query.trans\_dim\_actor(c)

query.sakila\_dw.load(c, "DIM\_ACTOR", "dim\_actor (id\_actor, name) VALUES (%s, %s)")

## DIM\_CATEGORY

c = query.sakila.extract("category\_id, name FROM category;")

query.sakila\_dw.load(c, "DIM\_CATEGORY", "dim\_category (id\_category, name) VALUES (%s, %s)")

## DIM\_FILM

c = query.sakila.extract(

"film.film\_id, film.title, language.name FROM film "

"INNER JOIN language ON film.language\_id = language.language\_id;")

query.sakila\_dw.load(c, "DIM\_FILM", "dim\_film (id\_film, title, language) VALUES (%s, %s, %s)")

## FACT\_INVENTORY

c = query.sakila.extract("inventory.last\_update, inventory.film\_id, inventory.store\_id FROM sakila.inventory;")

c = query.trans\_fact\_inventory(c)

query.sakila\_dw.load(c, "FACT\_INVENTORY",

"fact\_inventory (dim\_time,dim\_film, dim\_store, quantity\_film) VALUES (%s, %s, %s, %s)")

## FACT RENTALS

c = query.sakila.extract(

"rental.customer\_id, inventory.film\_id,store.store\_id,rental.rental\_date,rental.staff\_id "

"FROM sakila.rental "

"INNER JOIN sakila.staff ON rental.staff\_id = staff.staff\_id "

"INNER JOIN sakila.store ON staff.store\_id = store.store\_id "

"INNER JOIN sakila.inventory ON rental.inventory\_id = inventory.inventory\_id;")

c = query.trans\_fact\_rentals(c)

query.sakila\_dw.load(c, "FACT\_RENTALS",

"fact\_rentals (dim\_customer,dim\_film, dim\_store,dim\_time,dim\_staff, quantity\_rentals) VALUES (%s, %s, %s,%s, %s, %s)")

## FACT\_SALES

c = query.sakila.extract(

"payment.customer\_id,payment.payment\_date,store.store\_id, payment.staff\_id ,payment.amount FROM sakila.payment "

"INNER JOIN sakila.staff ON payment.staff\_id = staff.staff\_id "

"INNER JOIN sakila.store ON staff.store\_id = store.store\_id;")

c = query.trans\_fact\_sales(c)

query.sakila\_dw.load(c, "FACT\_SALES",

"fact\_sales(dim\_customer,dim\_time,dim\_store,dim\_staff,quantity\_sales, cost) VALUES (%s, %s, %s,%s, %s, %s)")

## BRIDGE\_FILM\_CATEGORY

c = query.sakila.extract("film\_category.category\_id, film\_category.film\_id FROM film\_category;")

query.sakila\_dw.load(c, "BRIDGE\_FILM\_CATEGORY", "bridge\_film\_category (id\_category,id\_film) VALUES (%s, %s)")

## BRIDGE\_FILM\_ACTOR

c = query.sakila.extract("film\_actor.actor\_id, film\_actor.film\_id FROM film\_actor;")

query.sakila\_dw.load(c, "BRIDGE\_FILM\_ACTOR", "bridge\_film\_actor (id\_actor,id\_film) VALUES (%s, %s)")

##Desconexion a las bases de datos

query.desconectar()

print("FIN DE CARGA DE DATOS A SAKILA\_DW: ", datetime.today().strftime('%Y-%m-%d %H:%M:%S'))

## Muestra de datos Data Warehouse

Con la ejecución del programa previamente, se mostrará a continuación como evidencia del correcto funcionamiento una muestra de los datos almacenados en las tablas de hechos, en las cuales se pueda visualizar como los datos ingresados en esta se relacionan a los datos almacenados en las tablas correspondientes a las dimensiones.

### Fact\_inventory

Tabla

Descripción generada automáticamente

Ilustración 10 Muestra de datos Fact\_inventory

### Fact\_rentals

Tabla

Descripción generada automáticamente

Ilustración 11 Muestra de datos Fact\_rentals

### Fact\_sales

Tabla

Descripción generada automáticamente

Ilustración 12 Muestra de datos Fact\_sales

### Muestra general del Data Warehouse

Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

Ilustración 13 Muestra general del Data Warehouse

# CREACIÓN Y PUBLICACIÓN DEL CUBO

## 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 realizo 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

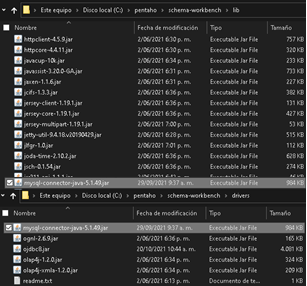
### Pentaho Schema Workbench

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.



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

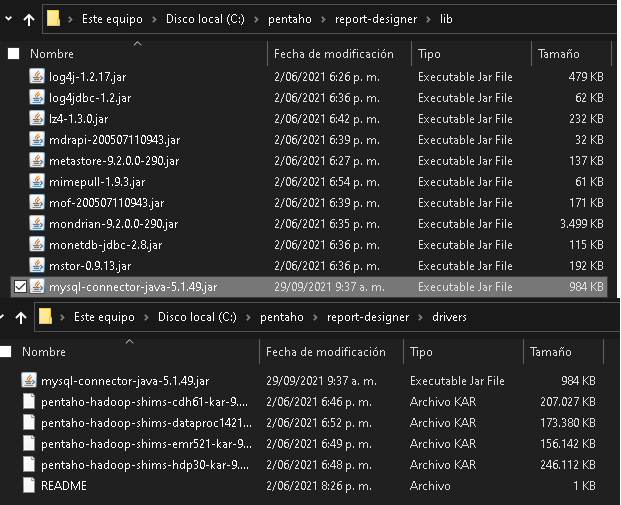
### Pentaho Report Desginer

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.



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.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

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.

|Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

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.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

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, Texto, Aplicación

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.

Interfaz de usuario gráfica, Texto, Aplicación, Tabla

Descripción generada automáticamente

Con estos pasos definidos, se realizo la creación de cada uno de los cubos requeridos, 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\_Time

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**: time
* **ForeingKey**: dim\_time
* **Type**: TimeDimesion

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

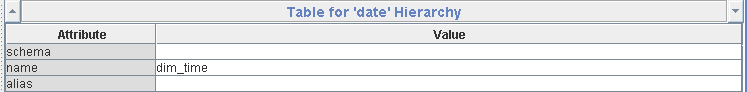
Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: date
* **PrimaryKey**: dim\_time - complete\_date

Tabla

Descripción generada automáticamente

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\_time.



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:

* Year: para este nivel se definen los siguientes datos:
  + Name: year
  + Table: dim\_time\*
  + Column: year
  + NameColumn: year
  + Type: Numeric
  + LevelType: TimeYears

Tabla

Descripción generada automáticamente

* Month: para este nivel se definen los siguientes datos:
  + Name: month
  + Table: dim\_time\*
  + Column: month
  + NameColumn: month
  + Type: String
  + LevelType: TimeMonths

Tabla

Descripción generada automáticamente

* Month\_day: para este nivel se definen los siguientes datos:
  + Name: month\_day
  + Table: dim\_time\*
  + Column: month\_day
  + NameColumn: month\_day
  + Type: Numeric
  + LevelType: TimeDays

Tabla

Descripción generada automáticamente

* Day\_of\_week: para este nivel se definen los siguientes datos:
  + Name: week\_day
  + Table: dim\_time\*
  + Column: week\_day
  + NameColumn: week\_name
  + Type: String
  + LevelType: TimeDays

Tabla

Descripción generada automáticamente

* Complete\_date: para este nivel se definen los siguientes datos:
  + Name: complete\_date
  + Table: dim\_time\*
  + Column: complete\_date
  + NameColumn: complete\_date
  + Type: Time
  + LevelType: TimeDays

Tabla

Descripción generada automáticamente

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

Imagen que contiene Texto

Descripción generada automáticamente

### Dim\_Store

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**: store
* **ForeingKey**: dim\_store
* **Type**: StandardDimesion

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

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**: location
* **PrimaryKey**: dim\_store – id\_store

Tabla

Descripción generada automáticamente con confianza media

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\_store.

Aplicación

Descripción generada automáticamente con confianza media

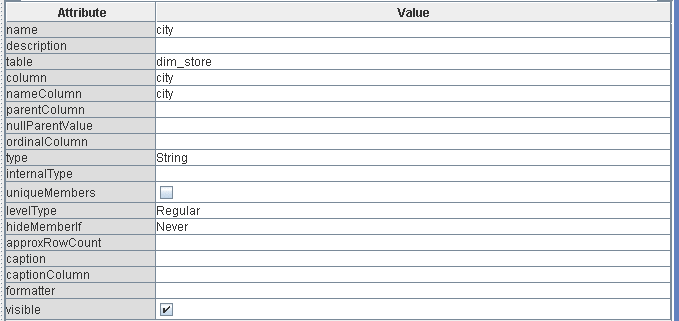
A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso el país, hasta el menor nivel, que sería id\_store, a continuación, se mostrara la creación de los niveles de esta jerarquía:

* Country: para este nivel se definen los siguientes datos:
  + Name: country
  + Table: dim\_store
  + Column: country
  + NameColumn: country
  + Type: String
  + LevelType: Regular

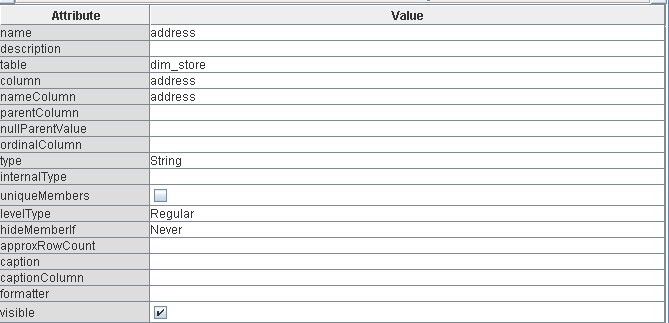
Tabla

Descripción generada automáticamente

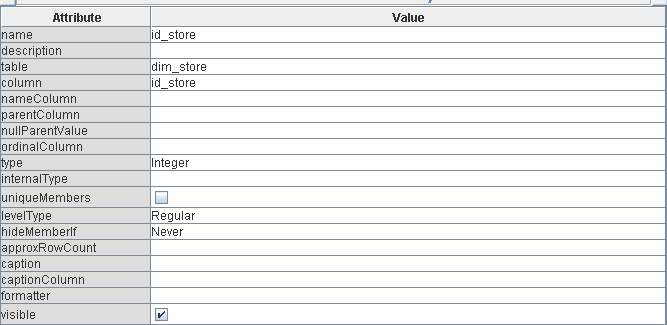
* City: para este nivel se definen los siguientes datos:
  + Name: city
  + Table: dim\_store
  + Column: city
  + NameColumn:
  + Type: String
  + LevelType: Regular



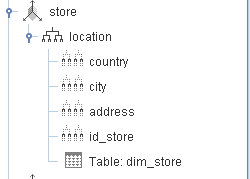
* Address: para este nivel se definen los siguientes datos:
  + Name: address
  + Table: dim\_store
  + Column: address
  + NameColumn:
  + Type: String
  + LevelType: Regular



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



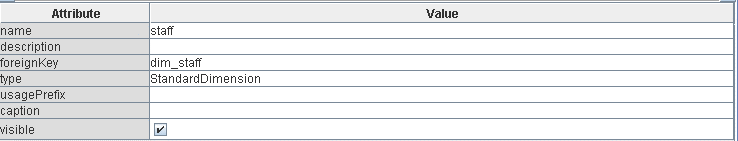
Con la dimensión, su respectiva jerarquia definida y los niveles de esta declarados, sin olvidar la asociación de esta jerarquia a la correspondiente tabla dentro del Data Warehouse de la dimensión, se obtiene la siguiente estructura para la dimensión de tienda:



### Dim\_Staff

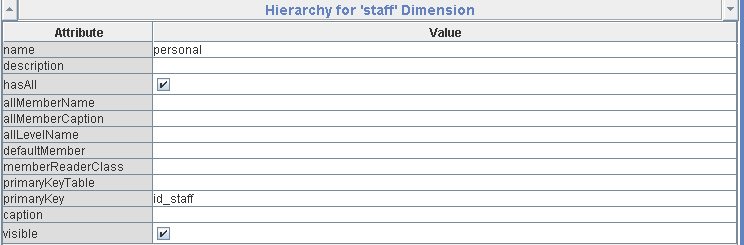
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: staff
* **ForeingKey**: dim\_staff
* **Type**: StandardDimesion

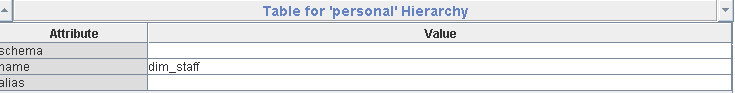


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: personal
* **PrimaryKey**: dim\_staff – id\_staff

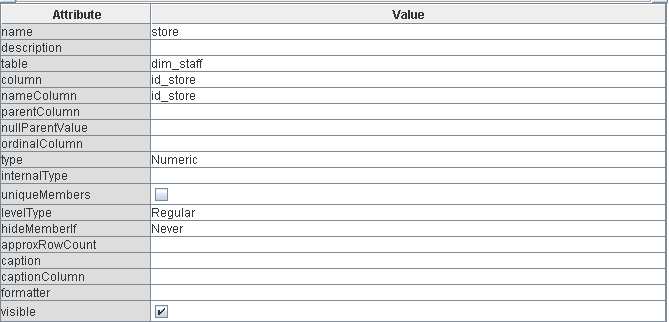


Con la jerarquía creada, se debe asociar la tabla correspondiente a esta, por lo que daremos clic derecho sobre la jerarquía y seleccionaremos la opción **Add Table**, en la ventana que se nos genera definiremos el dato name como dim\_staff.

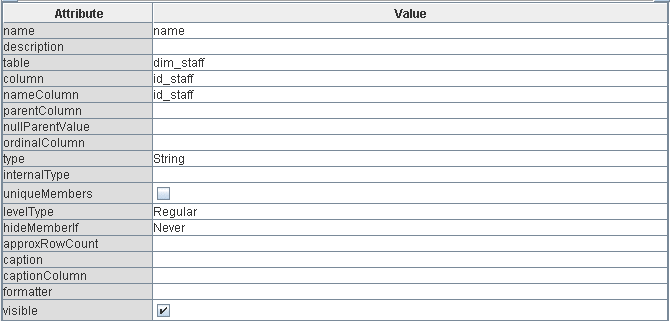


A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso la tienda, hasta el menor nivel, que sería id\_staff, a continuación, se mostrara la creación de los niveles de esta jerarquía:

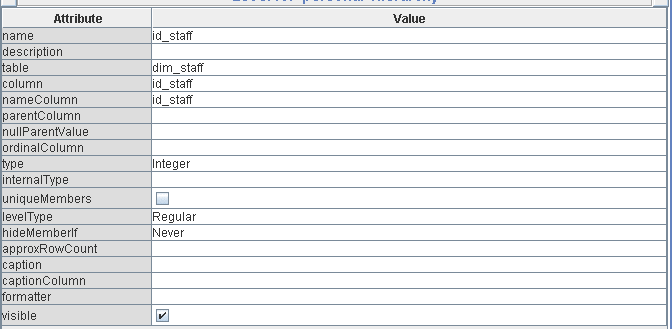
* Store: para este nivel se definen los siguientes datos:
  + Name: store
  + Table: dim\_staff
  + Column: store
  + NameColumn: store
  + Type: numeric
  + LevelType: regular



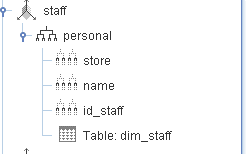
* Name: para este nivel se definen los siguientes datos:
  + Name: name
  + Table: dim\_staff
  + Column: name
  + NameColumn: name
  + Type: String
  + LevelType: regular



* Id\_staff: para este nivel se definen los siguientes datos:
  + Name: id\_staff
  + Table: dim\_staff
  + Column: id\_staff
  + NameColumn: id\_staff
  + Type: numeric
  + LevelType: regular



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



### Dim\_Film

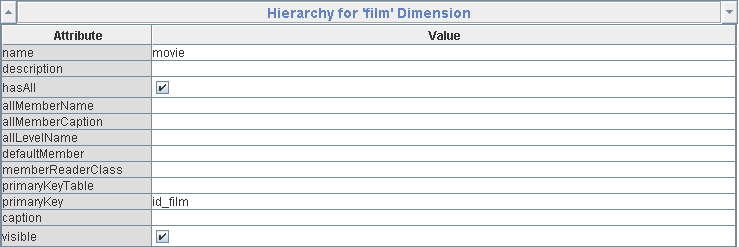
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: film
* **ForeingKey**: dim\_film
* **Type**: StandardDimesion

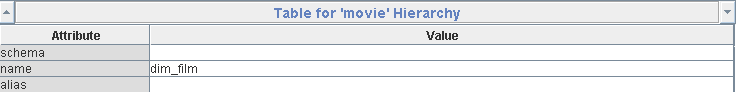


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: movie
* **PrimaryKey**: dim\_film – id\_film

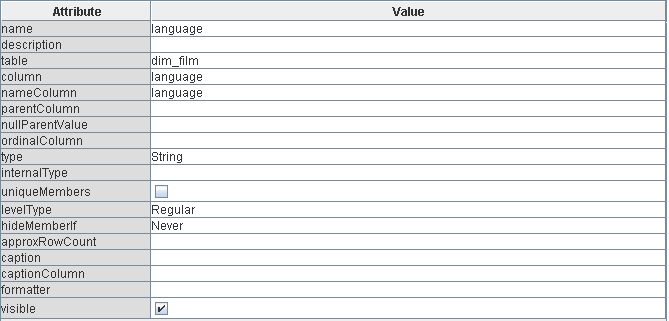


Con la jerarquía creada, se debe asociar la tabla correspondiente a esta, por lo que daremos clic derecho sobre la jerarquía y seleccionaremos la opción **Add Table**, en la ventana que se nos genera definiremos el dato name como dim\_film.

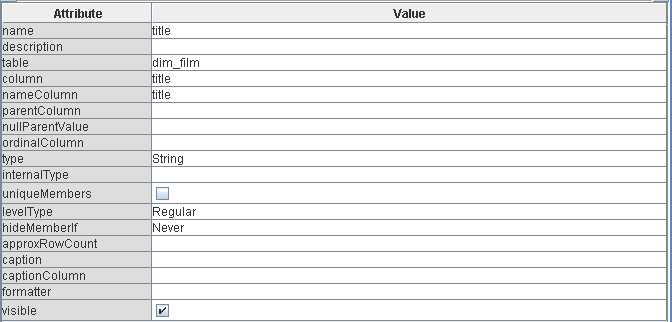


A continuación, se crearán los miembros que la conforman esta jerarquía desde el mayor nivel, en este caso el lenguaje, hasta el menor nivel, que sería id\_film, a continuación, se mostrara la creación de los niveles de esta jerarquía:

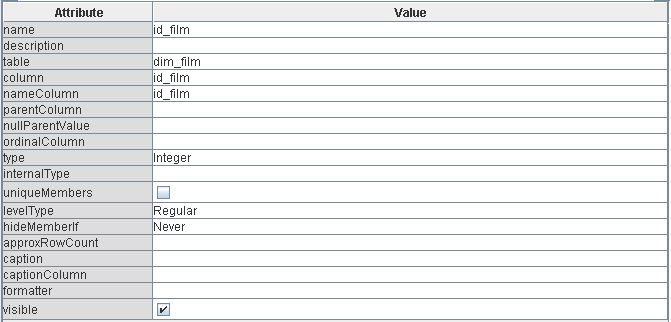
* Language: para este nivel se definen los siguientes datos:
  + Name: languaje
  + Table: dim\_film
  + Column: languaje
  + NameColumn: languaje
  + Type: String
  + LevelType: Regular



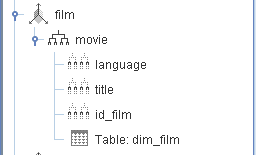
* Title: para este nivel se definen los siguientes datos:
  + Name: titlle
  + Table: dim\_film
  + Column: titlle
  + NameColumn: titlle
  + Type: String
  + LevelType: Regular



* Id\_film: para este nivel se definen los siguientes datos:
  + Name: id\_film
  + Table: dim\_film
  + Column: id\_film
  + NameColumn: id\_film
  + Type: integer
  + LevelType: regular



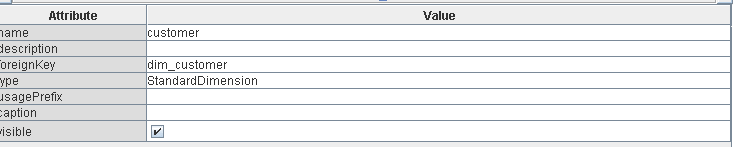
Con la dimensión, su respectiva jerarquia definida y los niveles de esta declarados, sin olvidar la asociación de esta jerarquia a la correspondiente tabla dentro del Data Warehouse de la dimensión, se obtiene la siguiente estructura para la dimensión de película:



### Dim\_Customer

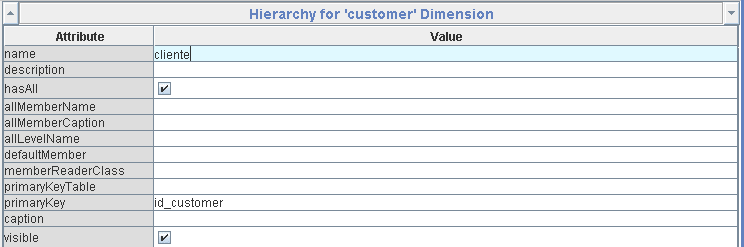
Para la creación de esta dimensión, en cada cubo se dará clic derecho sobre el cubo y se seleccionará la opción **Add Dimension**, en la ventana que se nos genera, definiremos los siguientes datos para esta dimensión:

* **Name**: customer
* **ForeingKey**: dim\_customer
* **Type**: StandardDimesion

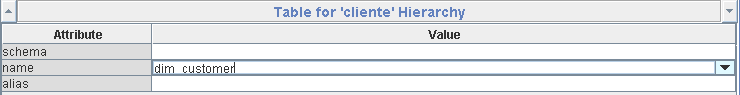


Con estos datos definidos, se creará la jerarquía correspondiente para esta dimensión, para esto, daremos clic derecho sobre la dimensión previamente creada y seleccionaremos la opción **Add Herarchy,** en la ventana que se nos genera definiremos los siguientes datos para la jerarquía:

* **Name**: cliente
* **PrimaryKey**: dim\_customer – id\_customer



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\_customer.



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

* Country: para este nivel se definen los siguientes datos:
  + Name: country
  + Table: dim\_customer
  + Column: country
  + NameColumn: country
  + Type: String
  + LevelType: regular

Tabla

Descripción generada automáticamente

* City: para este nivel se definen los siguientes datos:
  + Name: city
  + Table: dim\_customer
  + Column: city
  + NameColumn: city
  + Type: String
  + LevelType: regular

Tabla

Descripción generada automáticamente

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

Tabla

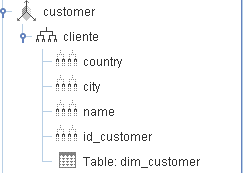
Descripción generada automáticamente

* Id\_customer: para este nivel se definen los siguientes datos:
  + Name: id\_customer
  + Table: dim\_customer
  + Column: id\_customer
  + NameColumn: id\_customer
  + Type: Numeric
  + LevelType: regular

Tabla

Descripción generada automáticamente

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



### 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 Costo del cubo Fact\_sales.

* Name: cost
* Aggregator: sum
* Column: fact\_sales - cost
* Datatype: numeric

Tabla

Descripción generada automáticamente con confianza media

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.

### Fact\_sales

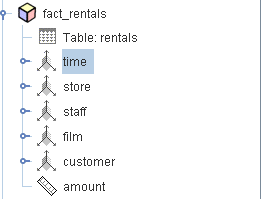
En este cubo se emplearon las dimensiones de tiempo, tienda, staff y cliente, empleando las medidas de costo y cantidad de ventas, obteniendo la siguiente estructura al finalizar el modelado del cubo.

Diagrama

Descripción generada automáticamente con confianza media

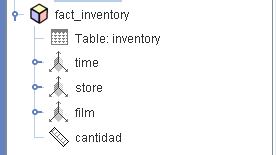
### Fact\_rentals

En este cubo se emplearon las dimensiones de tiempo, tienda, staff, película y cliente, empleando las medidas cantidad de rentas, obteniendo la siguiente estructura al finalizar el modelado del cubo.



### Fact\_inventory

En este cubo se emplearon las dimensiones de tiempo, tienda y película, empleando la medida de cantidad de películas, obteniendo la siguiente estructura al finalizar el modelado del cubo.



### 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, Texto, Aplicación

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.

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Dentro de la ventana desplegada, se ejecutará la siguiente sentencia la cual retornara la cantidad de ventas y el total de estas en el año 2006, si esta sentencia funciona correctamente, se podrá continuar con la publicación del cubo, de lo contrario, se realizaran las correcciones necesarias.

SELECT

{ [Measures].[amount], [Measures].[cost] } ON COLUMNS,

{ [time.date].[2006] } ON ROWS

FROM sales

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

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

### Publicación del cubo

Finalmente, para realizar la publicación del cubo, se debe tener en cuenta que Pentaho server debe estar “”, 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

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Si el proceso de publicación del cubo se realizo 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.

Tabla

Descripción generada automáticamente

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 el servidor