Universidad Nacional Experimental del Táchira
Vice-Rectorado Académico
Decanato de Docencia
Departamento de Informática
Base de Datos II

Datawarehouse Alquiler Videos

Integrantes:

Chacón Sánchez, Anny Esmeralda C.I. V-23.136.684

Sánchez García, Gregory Erasmo C.I. V-20.999.771

Suarez Silva, Julia Estefany C.I V-25.023.358

Valladares Guerrero, Yuleidy Rossana C.I V.-24.147.599

Zambrano Campitelli, Loreana Isabel C.I V.-24.356.109

San Cristóbal, marzo de 2018

Índice

1. Análisis de Requerimientos

a. Identificar Preguntas

- Se trabajará con el proceso de alquiler y recepción de películas de las diferentes sucursales de la empresa
- La empresa desea saber:
    - Promedio de días que dura el alquiler de una película por su categoría en un tiempo dado.
    - La gerencia necesita saber el desempeño histórico del personal de las tiendas, para ello quisiera tomar indicadores como la cantidad y monto de alquileres que gestionan.
    - Cantidad de alquileres, disponibilidad y proporción por película y por día.
    - Cantidad de veces que una película a alquilar no está en stock por más de 3 horas antes alquilarla.
    - Cantidad de películas alquiladas en una tienda por categoría.
    - Cantidad de alquileres por categoría de película según el cliente y su ciudad.

b. Identificar indicadores y perspectivas

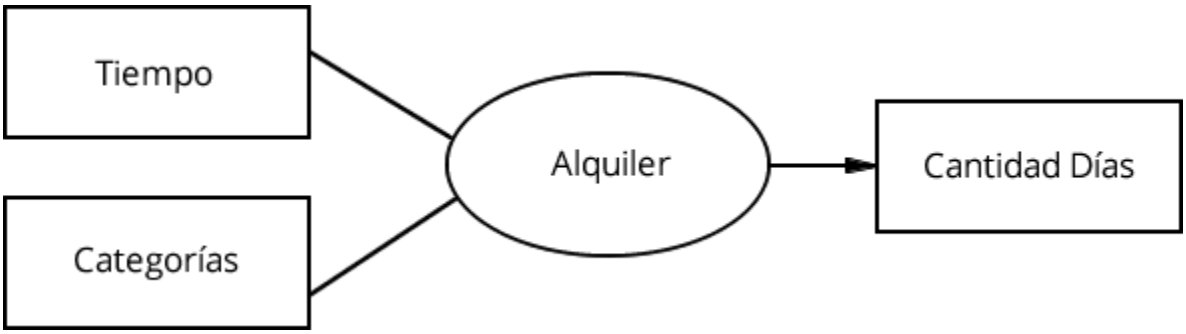
Indicadores / Perspectivas

- Promedio de días que dura el alquiler de una película por su categoría en un tiempo dado
- La gerencia necesita saber el desempeño histórico del personal de las tiendas, para ello quisiera tomar indicadores como la cantidad y monto de alquileres que gestionan en un tiempo determinado.
- Cantidad de alquileres, disponibilidad y proporción por película en un tiempo determinado.
- Cantidad de veces que una película a alquilar no está en stock por más de 3 horas antes de alquilarla en un tiempo determinado.

- Cantidad de películas alquiladas en una tienda por categoría en un tiempo determinado.
- Cantidad de alquileres por categoría de película según el cliente y su ciudad en un tiempo determinado.

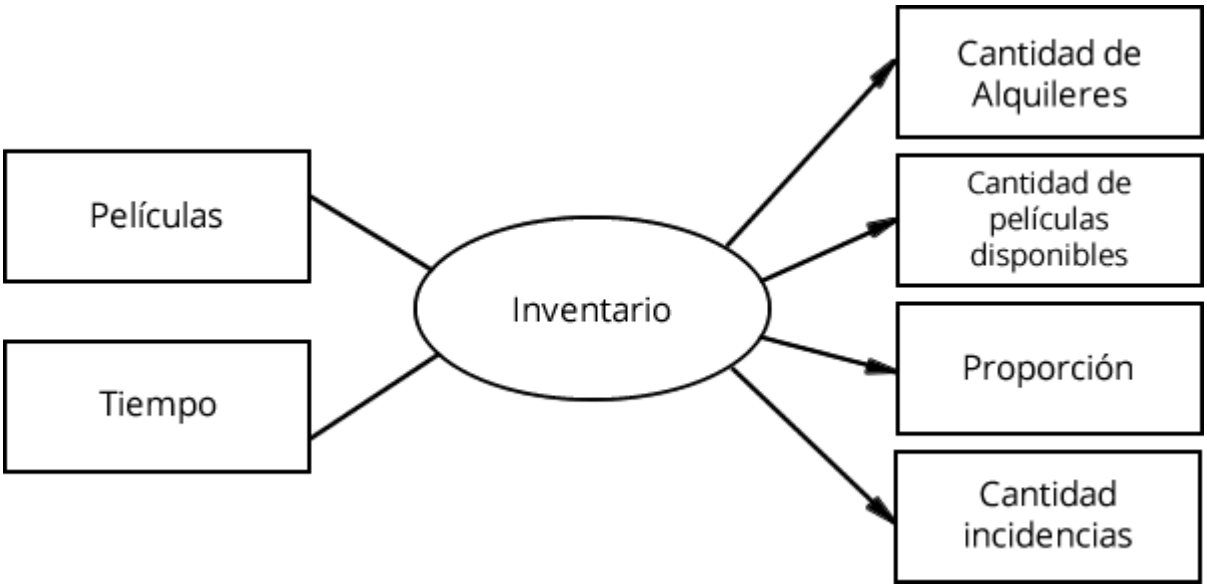
c. Modelo conceptual

"Promedio de días que dura el alquiler de una película por su categoría en un tiempo dado"

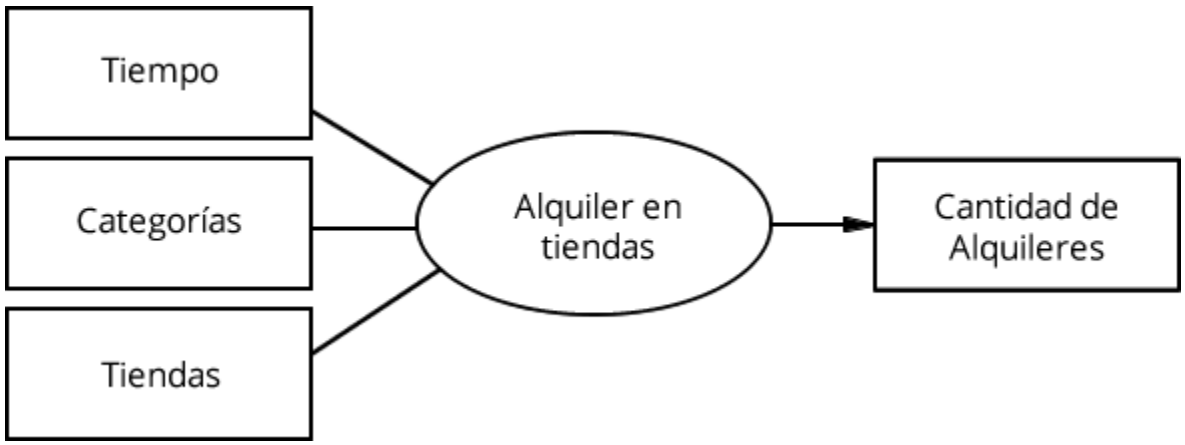


"La gerencia necesita saber el desempeño histórico del personal de las tiendas, para ello quisiera tomar indicadores como la cantidad y monto de alquileres que gestionan en un tiempo determinado."

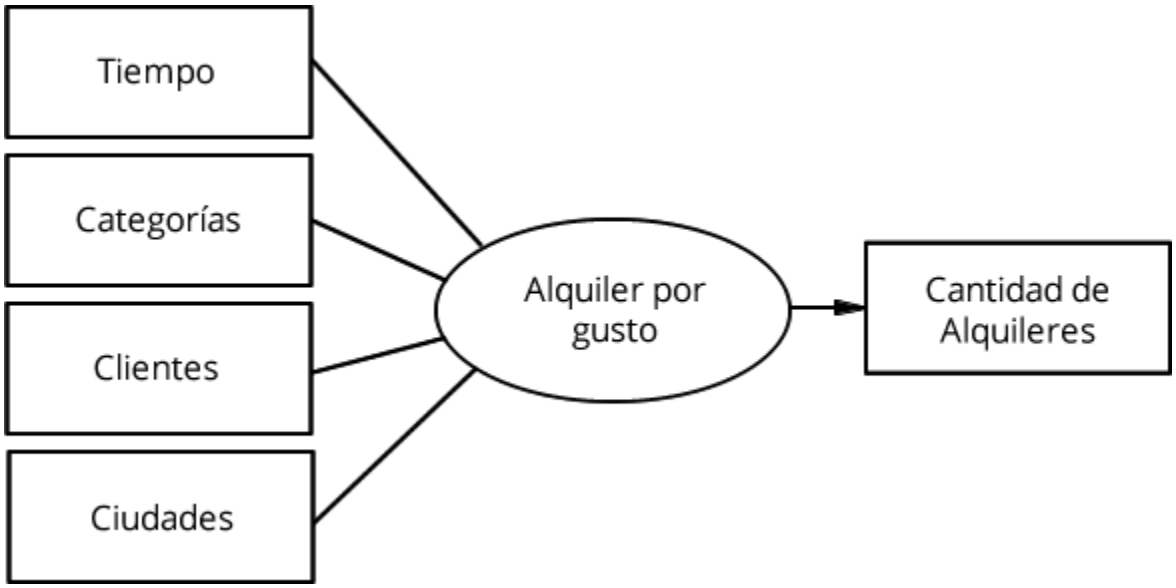
“Cantidad de alquileres, disponibilidad y proporción por película en un tiempo determinado.” “Cantidad de veces que una película a alquilar no está en stock por más de 3 horas antes de alquilarla en un tiempo determinado.”



“Cantidad de películas alquiladas en una tienda por categoría en un tiempo determinado.”



"Cantidad de alquileres por categoría de película según el cliente y su ciudad en un tiempo determinado."



2. Análisis de los OLTP

a. Conformar indicadores

DataMart "Alquiler":

- Cantidad Días:
  - Hechos: Fecha de retorno - Fecha de renta
  - Función de sumatoria: SUM
  - Aclaración: El indicador "Cantidad Días" representa los días de alquiler por las diferentes categorías de las películas

DataMart "Desempeño del personal":

- Cantidad de Alquileres:
  - Hechos: Alquiler
  - Función agregación: COUNT

    - Aclaración: El indicador “Cantidad Alquileres” representa la cantidad de alquileres gestionados por cada empleado.

- Monto por Alquiler:
    - Hechos: Monto del pago de los Alquileres
    - Función agregación: SUM
    - Aclaración: El indicador “Monto por Alquileres” representa la suma total de los pagos de cada alquiler gestionado por cada empleado.

DataMart “Inventario”

- Cantidad de Alquileres
    - Hechos: Alquiler
    - Función agregación: COUNT
    - Aclaración: El indicador “Cantidad Alquileres” representa la cantidad de alquileres realizados en un día determinado

- Cantidad de Películas Disponibles
    - Hechos: Película disponible
    - Función agregación: COUNT
    - Aclaración: El indicador “Cantidad de películas disponibles” representa la cantidad de películas disponibles en un día determinado

- Proporción alquiler
    - Hechos: Cantidad de alquileres/Total de películas
    - Aclaración: El indicador “Proporción” representa la cantidad de películas alquiladas entre el total de películas en un día determinado

- Cantidad Incidencia
    - Hechos: Incidencia
    - Función agregación: COUNT

- Aclaración: El indicador “Cantidad de incidencias” representa la cantidad de veces que, al alquilar una película, esta no ha estado en stock por más de 3 horas consecutivas

DataMart “Alquiler en tiendas”

- Cantidad de Alquileres
  - Hechos: Alquiler
  - Función agregación: COUNT
  - Aclaración: El indicador “Cantidad Alquileres” representa la cantidad de alquileres por categorías, en una tienda determinada realizados en un día.

DataMart “Alquiler por gusto”

- Cantidad de Alquileres
  - Hechos: Alquiler
  - Función agregación: COUNT
  - Aclaración: El indicador “Cantidad Alquileres” representa la cantidad de alquileres por categorías de un cliente específico y su ciudad en un día determinado.

b. Establecer correspondencias

- La tabla “Category” se relaciona con la perspectiva “Categorías” en todos los DataMarts.
- La tabla “City” se relaciona con la perspectiva “Ciudades” en todos los DataMarts.
- La tabla “Customer” se relaciona con la perspectiva “Clientes” en todos los Datamarts.
- La tabla “Film” se relaciona con la perspectiva “Películas” en todos los DataMarts.
- La tabla “Store” se relaciona con la perspectiva “Tiendas” en todos los DataMarts.
- La tabla “Staff” se relaciona con la perspectiva “Personal” en todos los DataMarts.
- El campo “rental_date” de la tabla “rental” corresponde a la perspectiva “Tiempo” del DataMart “Alquiler”, “Inventario”, “Alquiler en tiendas” y “Alquiler por gusto”.

- El campo “rental_date” de la tabla “rental” unido al dato “payment_date” de la tabla “payment” corresponde a la perspectiva “TiempoPago” del DataMart “Desempeño del personal”.
- El campo “return_date” menos el campo “rental_date”, ambos de la tabla “rental”, conforman el parámetro para calcular la cantidad de días correspondiente al campo “Días” del DataMart “Alquiler”.
- El indicador "Cantidad Alquileres" del DataMart “Desempeño del personal” corresponde al conteo total de registros en la tabla "rental" por cada "staff_id" de la tabla "staff".
- El indicador "Monto por Alquileres" del DataMart "Desempeño del personal" corresponde con la sumatoria del campo "amount" de la tabla "payment" para cada "staff_id".
- El indicador "Cantidad Alquileres" corresponde a la suma total de registros en la tabla "rental" por cada película alquilada del DataMart ”Inventario”.
- El indicador “Cantidad de películas disponibles” corresponde a la sumatoria total de registros de la tabla “rental” cuyo campo “return_date” sea null, en el DataMart inventario.
- El indicador “Proporción” del DataMart “Inventario”, corresponde a la división entre el indicador “Cantidad Alquileres” y la cantidad de registros por película en la tabla “Inventory”.
- El indicador “Cantidad Incidencia” del DataMart “Inventario”, corresponde a la sumatoria de registros por películas de la tabla “Rental” en conjunto con la sumatoria de la tabla “Inventory”, las cuales eran 0 hasta 3 horas antes del campo “rental_date”.
- El indicador “Cantidad Alquileres” del DataMart “Alquiler en Tiendas” corresponde a la sumatoria de todos los registros de la tabla “rental” por cada categoría de películas en cada tienda.

- El indicador “Cantidad Alquileres” del DataMart “Alquiler por gusto” corresponde a la sumatoria de todos los registros de la tabla “rental” por cada categoría de películas, en una ciudad determinada de un cliente.

c. Nivel de granularidad

Perspectiva “Categorías”

- Con respecto a la perspectiva “Categorías”, los datos disponibles son los siguientes:
    - category_id: es la clave primaria de la tabla ”Category”, y representa unívocamente a una categoría en particular.
    - name: es el nombre de la categoría.
    - last_update: Última fecha de actualización de una categoría.

- En la perspectiva “Categorías” los datos que se van a utilizar son los siguientes:
    - original_categoria_id: “category_id” de la tabla ”Category”, ya que representa a una categoría en particular.
    - nombre: “name” de la tabla ”Category”, ya que representa el nombre de la categoría.

Perspectiva “Tiempo”

- En la perspectiva “Tiempo”, los datos disponibles son los siguientes:
    - rental_date: Es la fecha en que se alquiló una película, en la tabla “rental”
    - Año.
    - Semestre.
    - Trimestre.
    - Número de mes.
    - Nombre del mes.
    - Semana
    - Nombre del día.

- En la perspectiva “Tiempo” los datos que se van a utilizar son los siguientes:
    - Fecha.
    - Año.
    - Número de mes.
    - Nombre del mes.
    - Semana.
    - Nombre del día.

Perspectiva “TiempoPago”

- En la perspectiva “**TiempoPago**”, los datos disponibles son los siguientes:
    - rental_date: Es la fecha en que se alquiló una película, en la tabla “rental”
    - Año.
    - Semestre.
    - Trimestre.
    - Número de mes.
    - Nombre del mes.
    - Semana
    - Nombre del día.

- En la perspectiva “**TiempoPago**” los datos que se van a utilizar son los siguientes:
    - Fecha.
    - Año.
    - Número de mes.
    - Nombre del mes.
    - Semana.
    - Nombre del día.

Perspectiva “Ciudades”

- Con respecto a la perspectiva “Ciudades”, los datos disponibles son los siguientes:
    - city_id: es la clave primaria de la tabla “City”, y representa unívocamente a una ciudad en particular.
    - city: es el nombre de la ciudad en la tabla “City”
    - country_id: es la clave foránea de la tabla “Country” en la tabla “City” y representa el país al que pertenece una ciudad
    - country: es el nombre del país al que pertenece la ciudad en la tabla “City” unida a “Country”
    - last_update: Última fecha de actualización de una ciudad.

- En la perspectiva “Ciudades” los datos que se van a utilizar son los siguientes:
    - original_ciudad_id: “city_id” de la tabla “City”, ya que representa a una ciudad en particular.
    - ciudad: “city” es el nombre de la ciudad en la tabla “City”
    - pais_id: “country_id” es la clave foránea de la tabla “Country” en la tabla “City” y representa el país al que pertenece una ciudad
    - país: “country” es el nombre del país al que pertenece la ciudad en la tabla “City” unida a “Country”

Perspectiva “Clientes”

- Con respecto a la perspectiva “Clientes”, los datos disponibles son los siguientes:
    - customer_id: es la clave primaria de la tabla “Customer”, y representa unívocamente a un cliente en particular.
    - store_id: representa a través de una clave foránea la tienda donde se registró el cliente.
    - first_name: nombre del cliente.
    - last_name: apellido del cliente.
    - email: correo electrónico del cliente.

- address_id: representa a través de una clave foránea la dirección que posee el cliente.
- activebool: estado del cliente, si está activo es “true” si no, es “false”
- create_date: fecha de creación de la instancia cliente en el sistema.
- last_update: Última fecha de actualización de un cliente.

- En la perspectiva “Clientes”, los datos que se van a utilizar son los siguientes:
  - original_cliente_id: “customer_id” de la tabla “Customer”, ya que representa a un cliente en particular.
  - nombre: “first_name” y “last_name” de la tabla “Customer”, ya que representa nombre completo del cliente.
  - correo: “email” de la tabla “Customer”, ya que representa correo electrónico del cliente.

Perspectiva “Películas”

- Con respecto a la perspectiva “Películas”, los datos disponibles son los siguientes:
  - film_id: es la clave primaria de la tabla “Film”, y representa unívocamente una película en particular.
  - title: es el título de la película.
  - description: es la descripción de la película.
  - release_year: es el año de lanzamiento de la película.
  - language_id: representa a través de una clave foránea el lenguaje de la película.
  - rental_duration: es la duración máxima de días que puede durar el alquiler de la película.
  - rental_rate: es la tasa de arrendamiento de la película.
  - length: duración de la película en minutos.
  - replacement_cost: es el costo de reemplazo de la película.
  - rating: es la clasificación de la película
  - special_feactures: características especiales incluidas en la película.

- last_update: última fecha de actualización de una película.

- En la Perspectiva “Películas”, los datos que se van a utilizar son los siguientes:
  - original_pelicula_id: “film_id” de la tabla “Film”, ya que representa a una película en particular.
  - titulo: “title” de la tabla “Film”, ya que representa el título de la película.
  - duracion_alquiler: “rental_duration” de la tabla “Film”, ya que representa la cantidad de días máxima que se puede alquilar la película.

Perspectiva “Tiendas”

- Con respecto a la perspectiva “Tiendas”, los datos disponibles son los siguientes:
  - store_id: es la clave primaria de la tabla “Store”, y representa unívocamente a una tienda en particular.
  - manager_staff_id: representa a través de una clave foránea el gerente de la tienda.
  - address_id: representa a través de una clave primaria la dirección de la tienda.
  - last_update: Última fecha de actualización de la tienda.

- En la perspectiva “Tiendas”, los datos que se van a utilizar son los siguientes:
  - original_tienda_id: “store_id” de la tabla “Store”, ya que representa a una tienda en particular.
  - gerente_id: “manager_staff_id” de la tabla “Store”, ya que representa al gerente de la tienda.
  - direccion_id: “address_id” de la tabla “Store”, ya que representa la dirección de la tienda.
  - direccion: “address” es la dirección de la tienda que pertenece la tabla “Store” unida a “Address”.
  - ciudad: “city” es la ciudad donde se encuentra la tienda que pertenece la tabla “Store” unida a “Address” y a “City”.

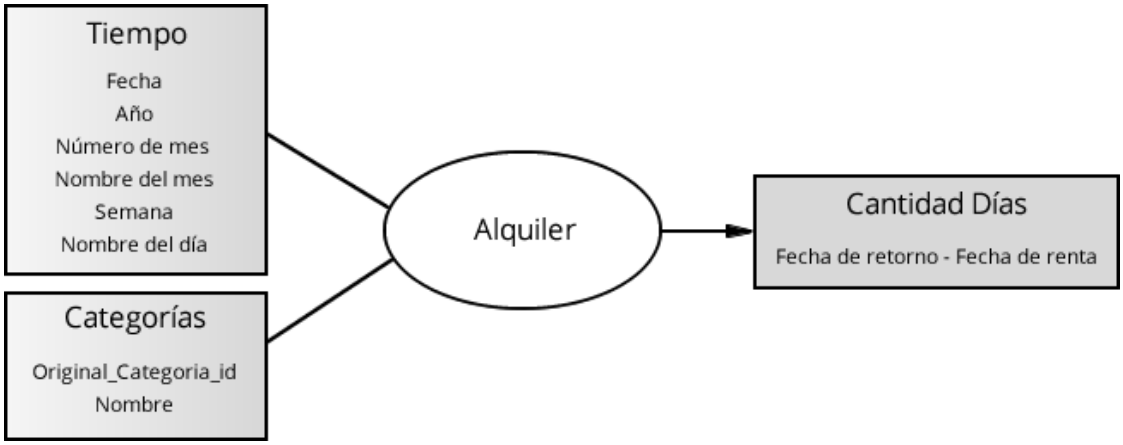
Perspectiva "Personal"

- Con respecto a la perspectiva "Personal", los datos disponibles son los siguientes:
  - staff_id: es la clave primaria de la tabla "Staff", y representa unívocamente a un empleado en particular.
  - first_name: nombre del empleado.
  - last_name: apellido del empleado.
  - email: correo electrónico del empleado.
  - store_id: representa a través de una clave foránea la tienda donde trabaja el empleado.
  - active: estado del empleado, si está activo es "true" si no, es "false".
  - username: usuario del empleado para ingresar al sistema.
  - password: contraseña del empleado para ingresar al sistema
  - picture: foto del empleado guardada en bytes.
  - last_update: Última fecha de actualización de un empleado.

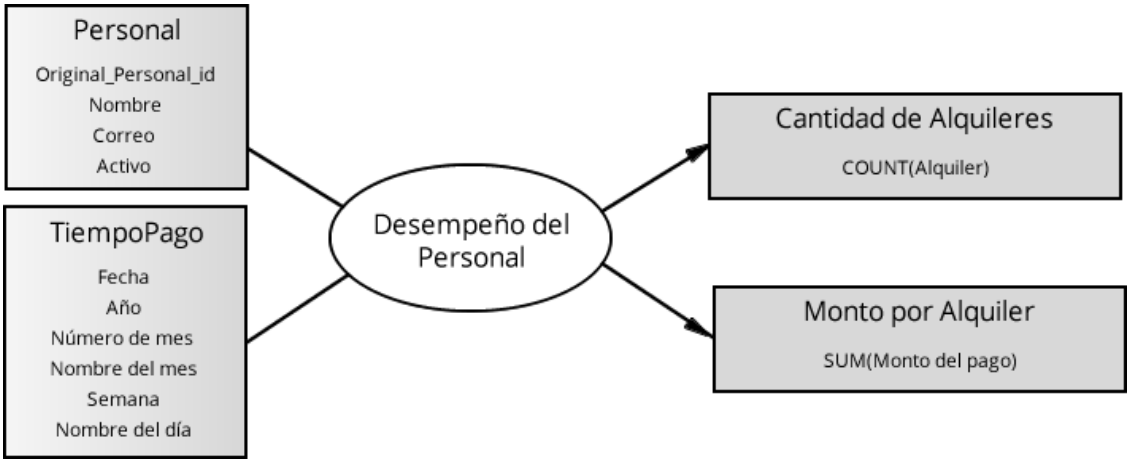
- En la perspectiva "Personal" los datos que se van a utilizar son los siguientes:
  - original_personal_id: "staff_id" es la clave primaria de la tabla "Staff", ya que representa a un empleado en particular.
  - nombre: "first_name" de la tabla "Staff", ya que representa el nombre del empleado y "last_name" de la tabla "Staff", ya que representa el apellido del empleado.
  - correo: "email" de la tabla "Staff", ya que representa el correo electrónico del empleado.

d. Modelo conceptual ampliado

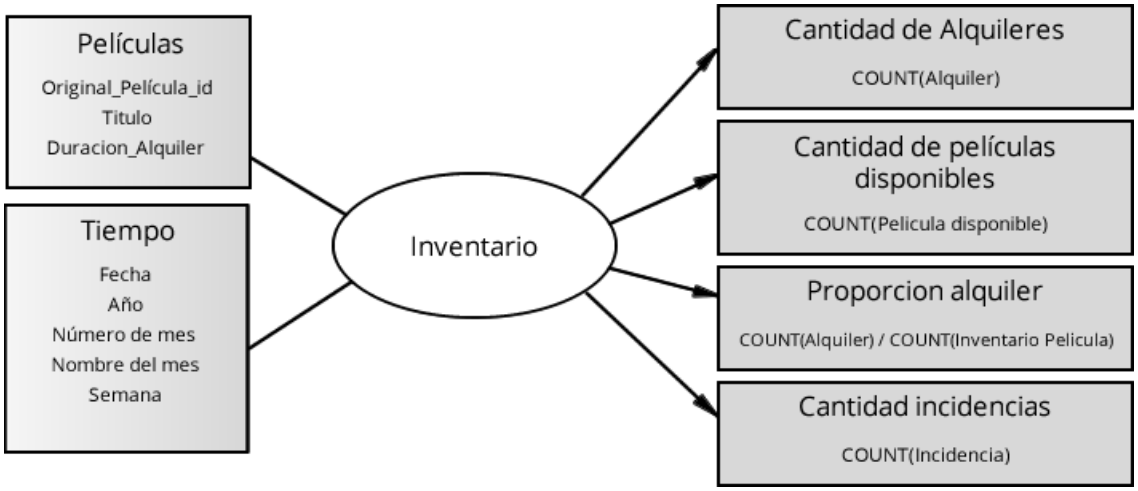
"Promedio de días que dura el alquiler de una película por su categoría en un tiempo dado"



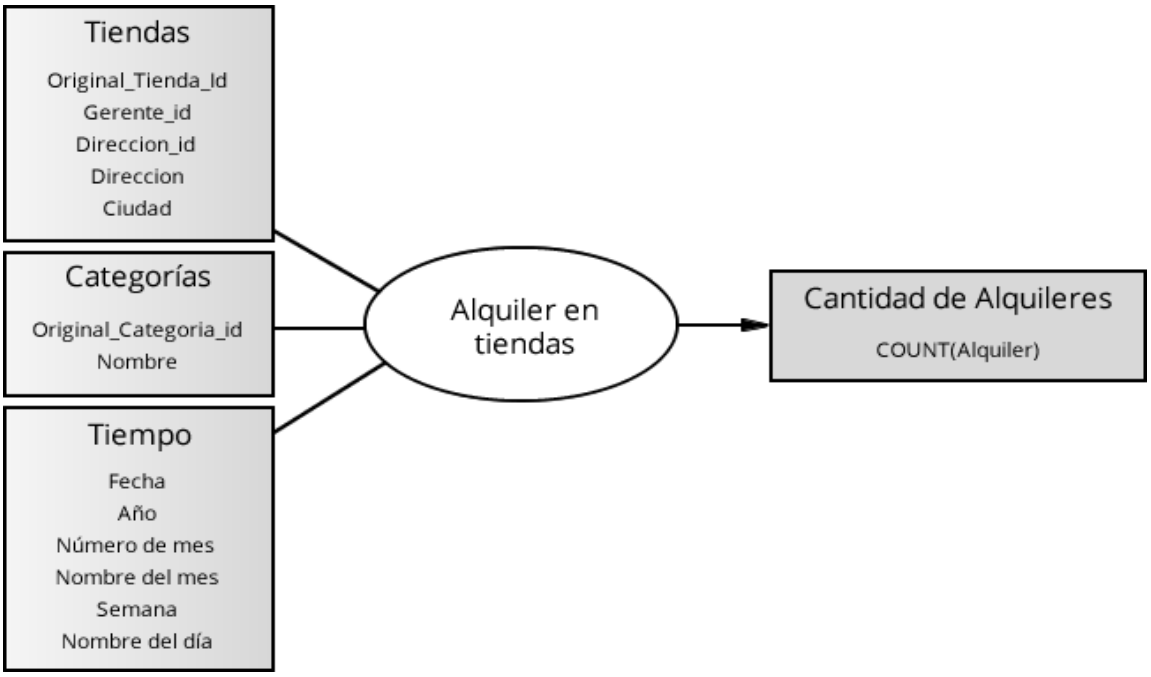
“La gerencia necesita saber el desempeño histórico del personal de las tiendas, para ello quisiera tomar indicadores como la cantidad y monto de alquileres que gestionan en un tiempo determinado.”



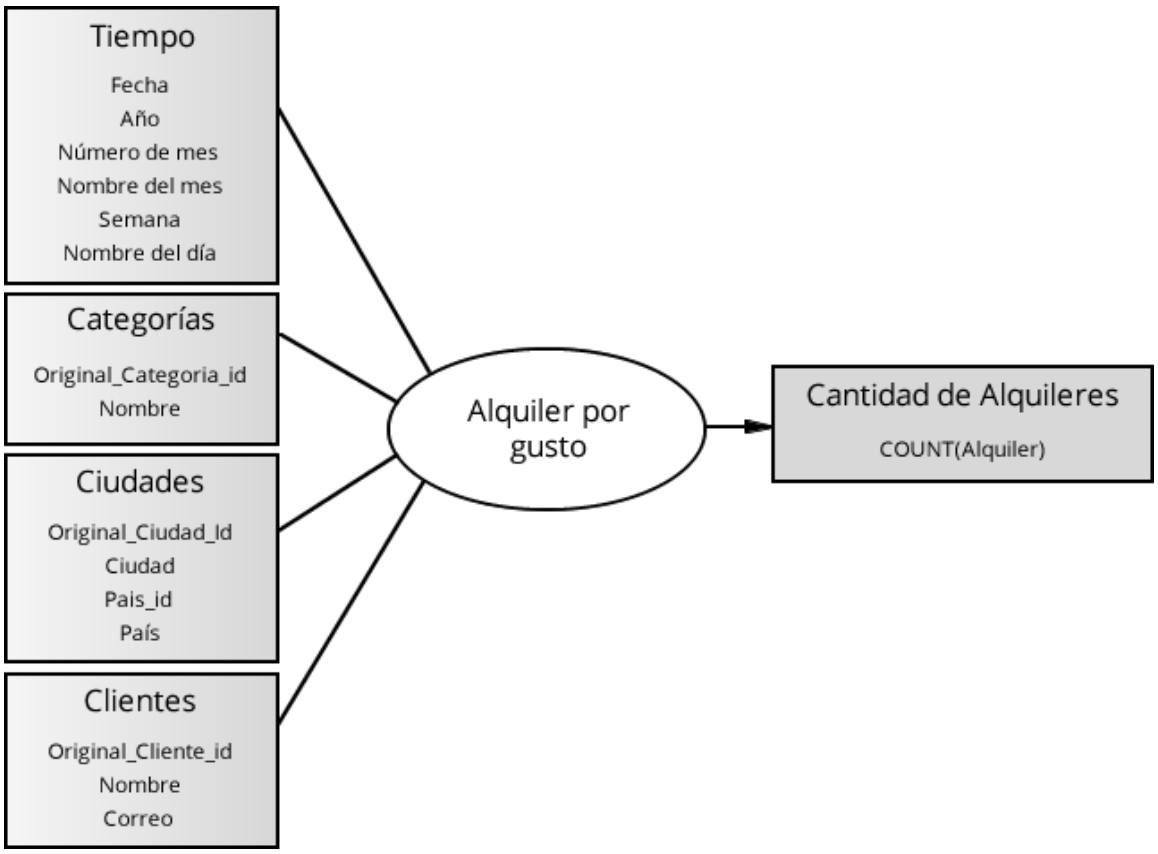
“Cantidad de alquileres, disponibilidad y proporción por película en un tiempo determinado.” “Cantidad de veces que una película a alquilar no está en stock por más de 3 horas antes de alquilarla en un tiempo determinado.”



“Cantidad de películas alquiladas en una tienda por categoría en un tiempo determinado.”



“Cantidad de alquileres por categoría de película según el cliente y su ciudad en un tiempo determinado.”

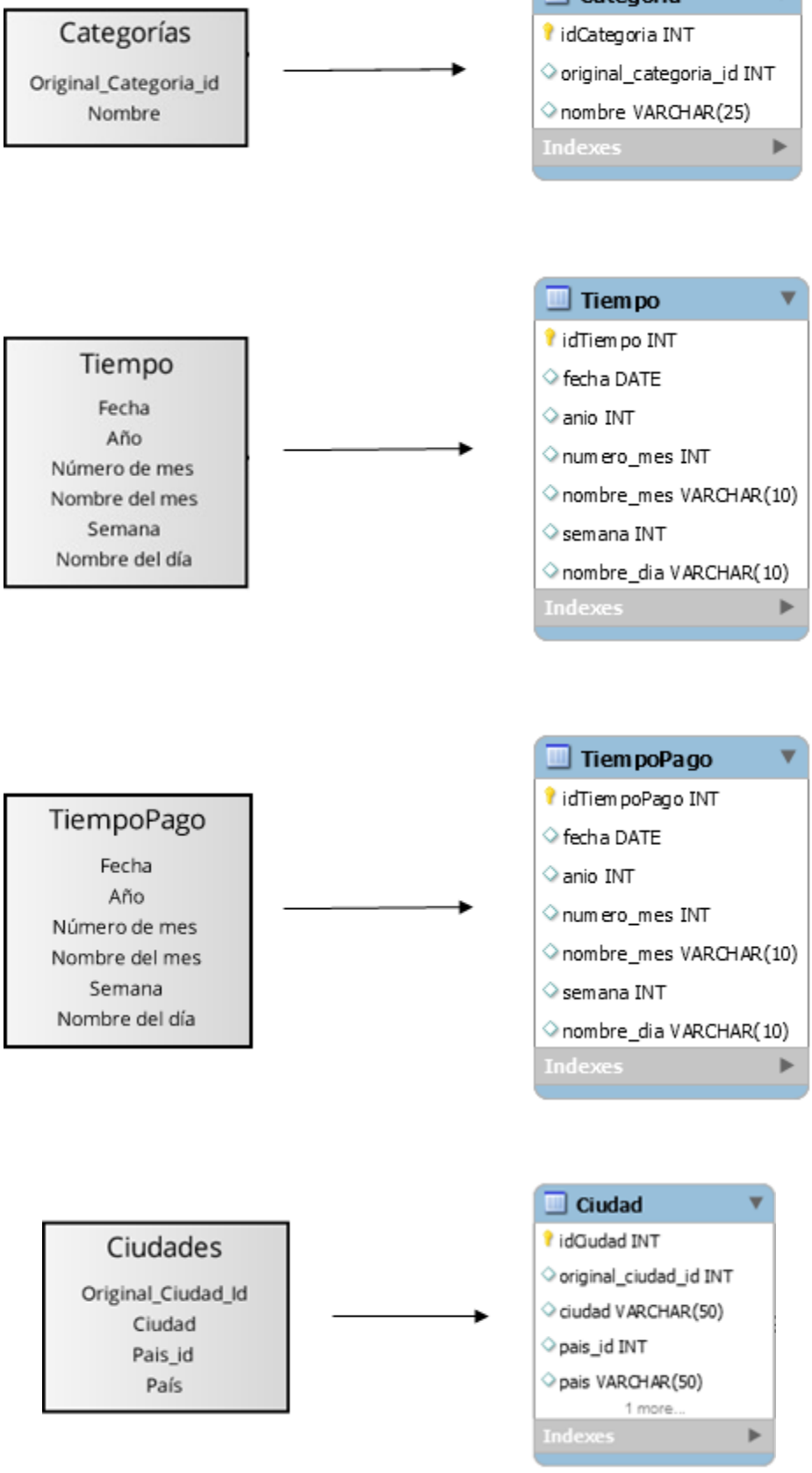


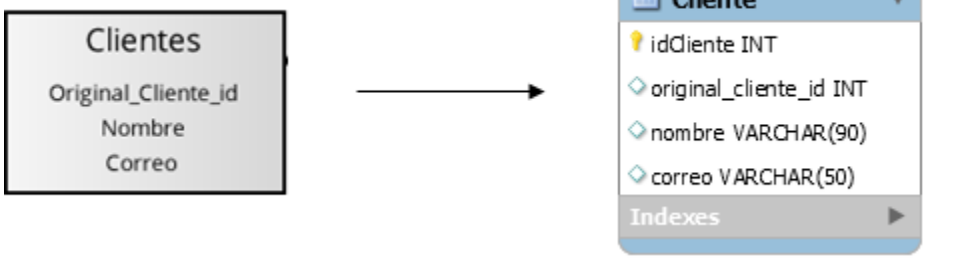
3. Modelo lógico del DWH

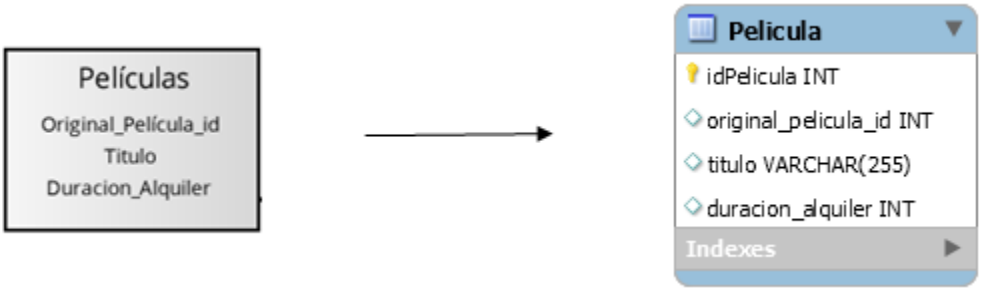
a. Tipo del modelo lógico del DWH

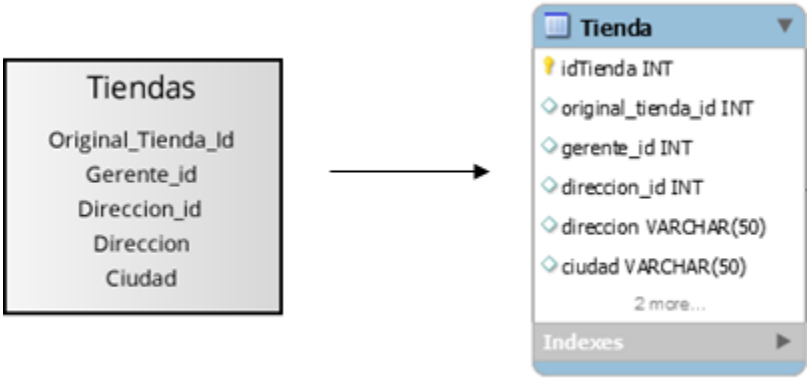
El esquema que se utilizará será en estrella, debido a sus características, ventajas y diferencias con los otros esquemas. Se utilizarán varios datamart en esquema de estrella, lo que llevará a tener una constelación.

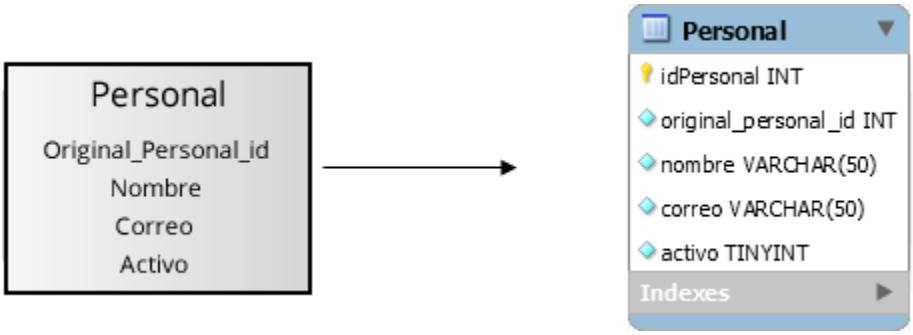
b. Tablas de dimensiones


Categorías
Original_Categoria_id
Nombre
Categoria
idCategoria INT
original_categoria_id INT
nombre VARCHAR(25)
Indexes
Tiempo
Fecha
Año
Número de mes
Nombre del mes
Semana
Nombre del día
Tiempo
idTiempo INT
fecha DATE
anio INT
numero_mes INT
nombre_mes VARCHAR(10)
semana INT
nombre_dia VARCHAR(10)
Indexes
TiempoPago
Fecha
Año
Número de mes
Nombre del mes
Semana
Nombre del día
TiempoPago
idTiempoPago INT
fecha DATE
anio INT
numero_mes INT
nombre_mes VARCHAR(10)
semana INT
nombre_dia VARCHAR(10)
Indexes
Ciudades
Original_Ciudad_Id
Ciudad
Pais_id
País
Ciudad
idCiudad INT
original_ciudad_id INT
ciudad VARCHAR(50)
pais_id INT
pais VARCHAR(50)
1 more...
Indexes

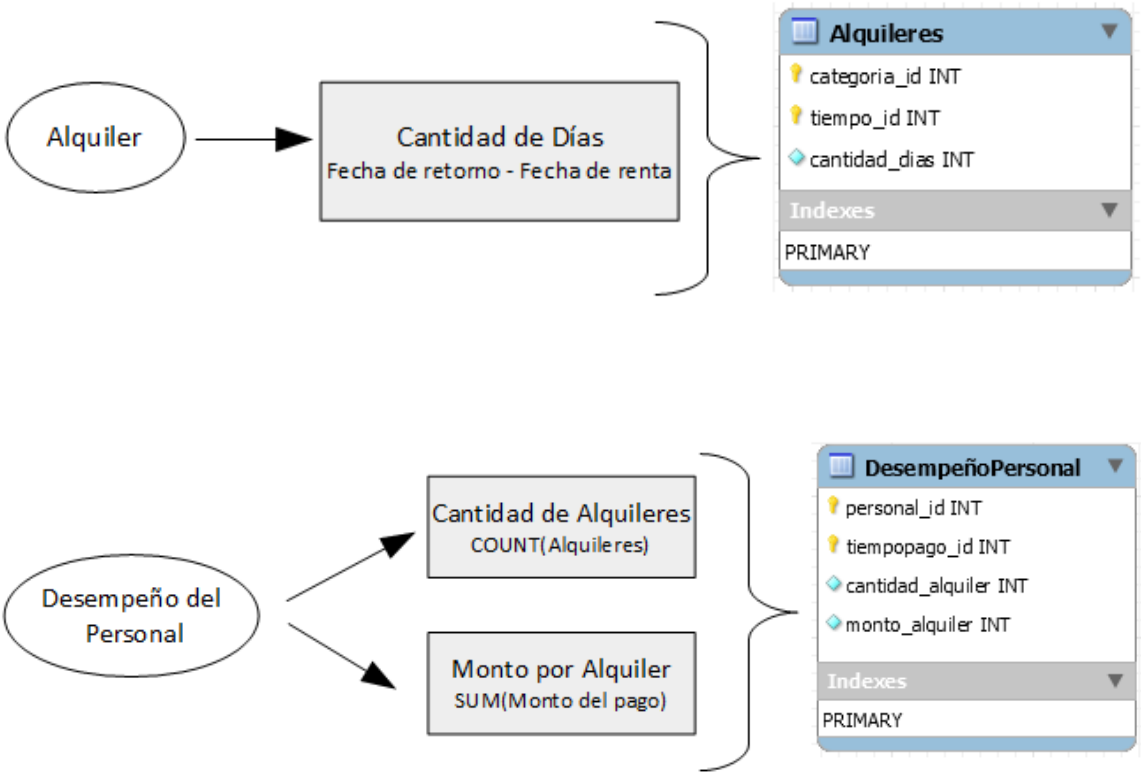

Clientes
Original_Cliente_id
Nombre
Correo
Cliente
idCliente INT
original_cliente_id INT
nombre VARCHAR(90)
correo VARCHAR(50)
Indexes

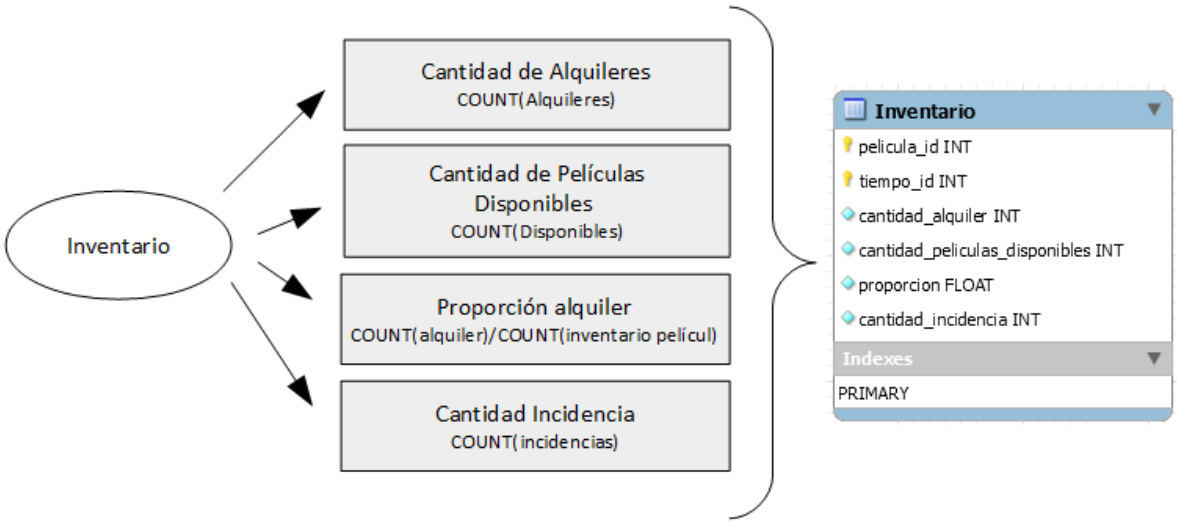

Películas
Original_Película_id
Titulo
Duracion_Alquiler
Pelicula
idPelicula INT
original_pelicula_id INT
titulo VARCHAR(255)
duracion_alquiler INT
Indexes

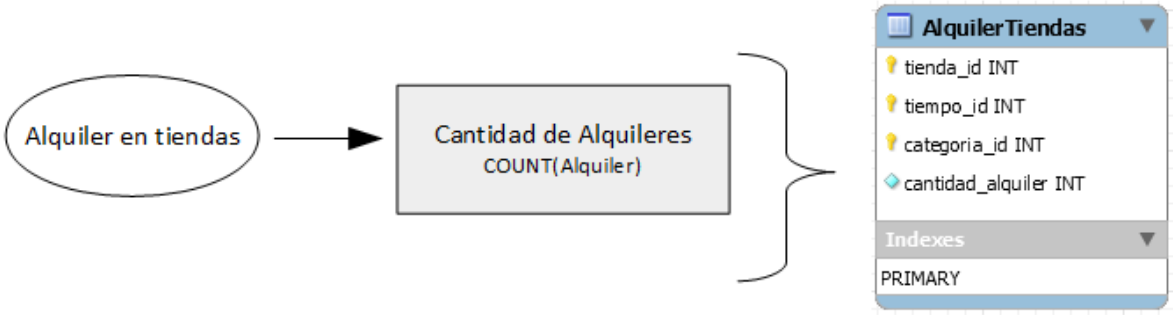

Tiendas
Original_Tienda_Id
Gerente_id
Direccion_id
Direccion
Ciudad
Tienda
idTienda INT
original_tienda_id INT
gerente_id INT
direccion_id INT
direccion VARCHAR(50)
ciudad VARCHAR(50)
2 more...
Indexes

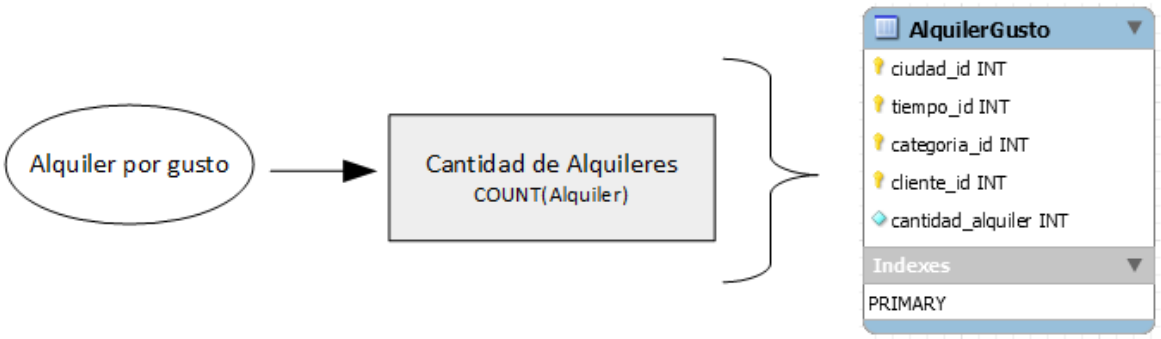

Personal
Original_Personal_id
Nombre
Correo
Activo
Personal
idPersonal INT
original_personal_id INT
nombre VARCHAR(50)
correo VARCHAR(50)
activo TINYINT
Indexes

c. Tablas de hechos



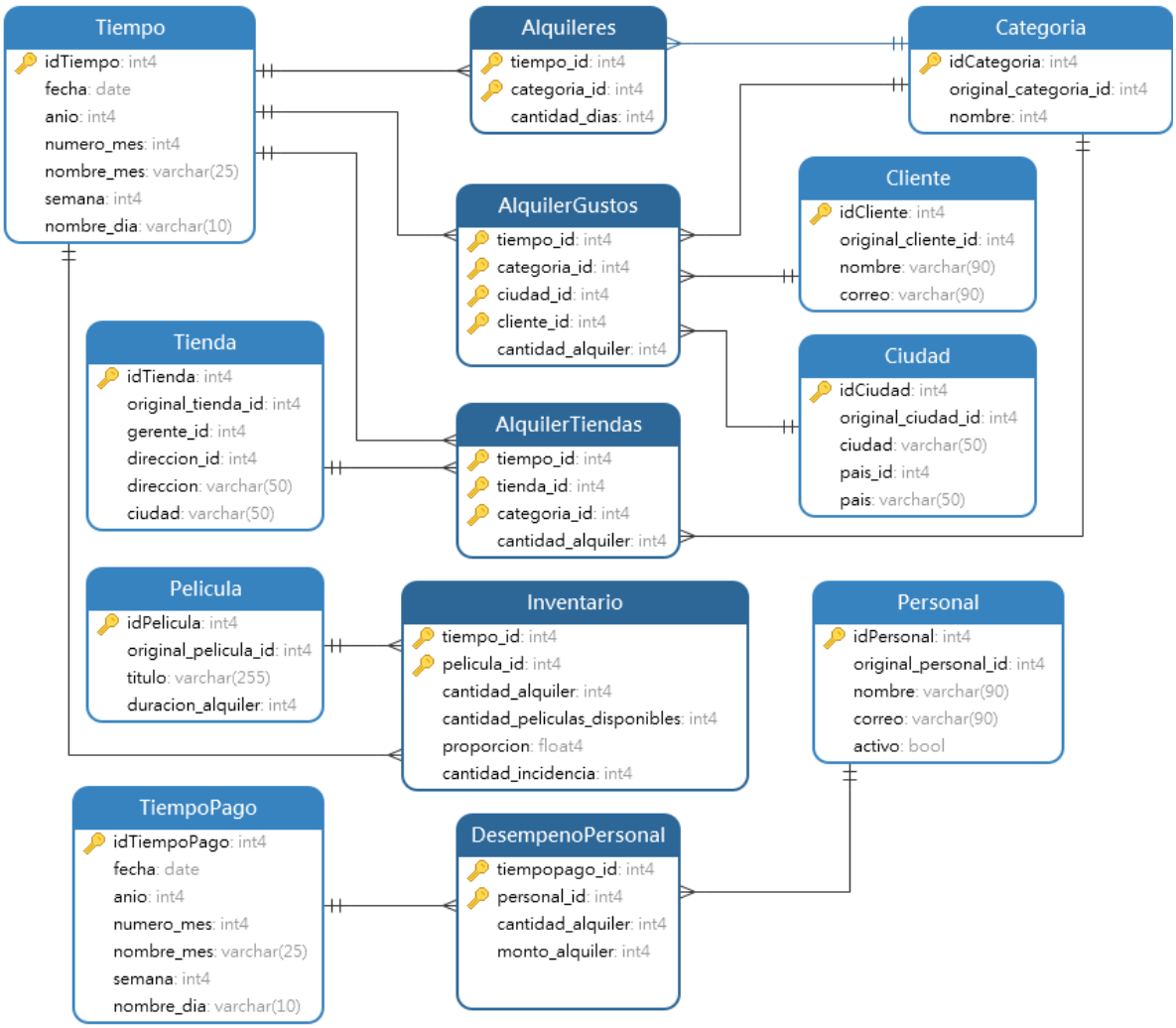

Inventario
Cantidad de Alquileres
COUNT(Alquileres)
Cantidad de Películas Disponibles
COUNT(Disponibles)
Proporción alquiler
COUNT(alquiler)/COUNT(inventario películ)
Cantidad Incidencia
COUNT(incidencias)
Inventario
pelicula_id INT
tiempo_id INT
cantidad_alquiler INT
cantidad_peliculas_disponibles INT
proporcion FLOAT
cantidad_incidencia INT
Indexes
PRIMARY


Alquiler en tiendas
Cantidad de Alquileres
COUNT(Alquiler)
AlquilerTiendas
tienda_id INT
tiempo_id INT
categoria_id INT
cantidad_alquiler INT
Indexes
PRIMARY


Alquiler por gusto
Cantidad de Alquileres
COUNT(Alquiler)
AlquilerGusto
ciudad_id INT
tiempo_id INT
categoria_id INT
cliente_id INT
cantidad_alquiler INT
Indexes
PRIMARY

d. Uniones

En el diagrama expuesto a continuación se presenta el modelo físico del DWH, en el centro se pueden apreciar las tablas de hechos y a los costados las dimensiones.



4. Integración de datos

a. Carga inicial

Con el programa Pentaho Data Integration se realizó el proceso ETL para la carga y actualización del Data Warehouse, se describirán los pasos llevados a cabo para cada dimensión, tabla de hechos y Data Mart descritos en el modelo físico anterior.

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Categoría".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Categoría.
  - Se tomará como fuente de entrada la tabla “Category” del OLTP.
  - A continuación, la consulta SQL:

```
SELECT category_id as original_categoria_id,
    name as nombre
FROM category;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Ciudad".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Ciudad.
  - Se tomarán como fuentes de entrada las tablas “City” y “Country” del OLTP.
  - A continuación, la consulta SQL:

```sql
SELECT c.city_id AS original_ciudad_id,
    c.city AS ciudad,
    p.country_id AS pais_id,
    p.country AS pais
FROM city c
JOIN country p ON (c.country_id = p.country_id);
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Cliente".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Cliente.
  - Se tomará como fuente de entrada la tabla “Customer” del OLTP.
  - A continuación, la consulta SQL:

```sql
SELECT customer_id as original_cliente_id,
    CONCAT(first_name, ' ', last_name) as nombre,
    email as correo
FROM customer;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Película".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Película.
  - Se tomará como fuente de entrada la tabla “Film” del OLTP.
  - A continuación, la consulta SQL:

```sql
SELECT film_id as original_pelicula_id,
    title as titulo,
    rental_duration as duracion_alquiler
FROM film;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Personal".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Personal.
  - Se tomará como fuente de entrada la tabla “Staff” del OLTP.
  - A continuación, la consulta SQL:

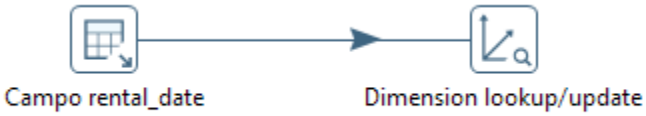
```sql
SELECT staff_id as original_personal_id,
    CONCAT(first_name, ' ', last_name) as nombre,
    email as correo,
    active as activo
FROM staff;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Tienda".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Tienda.
  - Se tomarán como fuentes de entrada las tablas “Store”, “Address” y “City” del OLTP.
  - A continuación, la consulta SQL:

```sql
SELECT s.store_id AS original_tienda_id,
    manager_staff_id AS gerente_id,
    a.address_id AS direccion_id,
    a.address AS direccion,
    c.city AS ciudad
FROM store s JOIN address a ON ( s.address_id = a.address_id)
JOIN city c ON (a.city_id = c.city_id);
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión Tiempo".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión Tiempo.
  - Se tomará como fuente de entrada “rental_date” de la tabla “rental” del OLTP.
  - A continuación, la consulta SQL:

```sql
SELECT DISTINCT rental_date::date as fecha,
    date_part('year',rental_date) as anio,
    date_part('month',rental_date) as numero_mes,
    to_char(rental_date,'TMMonth') as nombre_mes,
    date_part('week',rental_date) as semana,
    to_char(rental_date,'TMDay') as nombre_dia
FROM rental;
```



- A continuación, se especificarán las tareas llevadas a cabo por "Carga de Dimensión tiempoPago".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la dimensión TiempoPago.

- Se tomará como fuente de entrada “rental_date” de la tabla “rental” y “payment_date” de la tabla “payment” del OLTP.
- A continuación, la consulta SQL:

```
SELECT DISTINCT fecha::date as fecha,
    date_part('year',fecha) as anio,
    date_part('month',fecha) as numero_mes,
    to_char(fecha,'TMMonth') as nombre_mes,
    date_part('week',fecha) as semana,
    to_char(fecha,'TMDay') as nombre_dia
FROM (SELECT rental_date as fecha
    FROM rental UNION (SELECT payment_date as fecha FROM payment))x;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de tabla de Hechos AlquilerTiendas".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la tabla de hechos AlquilerTiendas.
  - Para la confección de la tabla de hechos, se tomaron como fuente las tablas “Rental”, “Inventory”, “Store” y “Film_Category”.
  - A continuación, la consulta SQL:

```
SELECT tp.idtiempo,
    t.idtienda,
    c.idcategoria,
    count(*) as cantidad_alquiler
FROM rental r
JOIN inventory i ON ( r.inventory_id = i.inventory_id)
JOIN dwh.tienda t ON (i.store_id = t.original_tienda_id)
JOIN film_category fc ON (fc.film_id = i.film_id)
JOIN dwh.categoria c ON (c.original_categoria_id = fc.category_id)
JOIN dwh.tiempo tp ON ( tp.fecha = r.rental_date::date )
GROUP BY tp.idtiempo, t.idtienda, c.idcategoria
ORDER BY 1 asc, 2 asc, 3 asc;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de tabla de Hechos AlquilerGustos".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la tabla de hechos AlquilerGustos.
  - Para la confección de la tabla de hechos, se tomaron como fuente las tablas “Rental”, “Inventory”, “Film”, “Film_Category”, “Customer” y “Address”.
  - A continuación, la consulta SQL:

```
SELECT t.idtiempo as tiempo_id,
    cli.idcliente as cliente_id,
    ciu.idciudad as ciudad_id,
    cat.idcategoria as categoria_id,
    COUNT(r.rental_id) as cantidad_alquiler
FROM rental r
JOIN inventory inv ON (r.inventory_id = inv.inventory_id)
JOIN film f ON(inv.film_id = f.film_id)
JOIN film_category fc ON(fc.film_id = f.film_id)
JOIN customer cus ON(cus.customer_id = r.customer_id)
JOIN address adr ON(adr.address_id = cus.address_id)
JOIN dwh.tiempo t ON (t.fecha = r.rental_date::date)
JOIN dwh.cliente cli ON(cli.original_cliente_id = cus.customer_id)
JOIN dwh.ciudad ciu ON(ciu.original_ciudad_id = adr.city_id)
JOIN dwh.categoria cat ON(cat.original_categoria_id=fc.category_id)
GROUP BY t.idtiempo, cli.idcliente, ciu.idciudad, cat.idcategoria;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de tabla de Hechos DesempenoPersonal".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la tabla de hechos DesempenoPersonal.
  - Para la confección de la tabla de hechos, se tomaron como fuente las tablas “Rental”, y “Payment”.
  - A continuación, la consulta SQL:

```
SELECT dp.idpersonal as personal_id,
    dt.idtiempopago as tiempopago_id,
    desempenno.fecha::date,
    sum((CASE WHEN (desempenno.pago = 0 and desempenno.renta<>-1) THEN 1
            ELSE 0 END)) as cantidad_alquiler,
    sum(desempenno.pago) as monto_alquiler
FROM (SELECT rental_date as fecha,
        staff_id as staff,
        0 as pago,
        rental_id as renta
        FROM rental UNION (SELECT payment_date,
                                  staff_id,
                                  amount,
                                  -1
                                  FROM payment)) as desempenno
JOIN dwh.personal dp on(desempenno.staff=dp.original_personal_id)
JOIN dwh.tiempopago dt on(desempenno.fecha::date=dt.fecha)
GROUP BY dp.idpersonal, dt.idtiempopago, desempenno.fecha::date
ORDER BY desempenno.fecha::date;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de tabla de Hechos Alquiler".
  - Obtener a través de una consulta SQL los datos del OLTP necesarios para cargar la tabla de hechos Alquiler.

- Para la confección de la tabla de hechos, se tomaron como fuente las tablas “Rental”, “Inventory”, “Film” y “Film_Category”.
- A continuación, la consulta SQL:

```
SELECT dt.idTiempo AS tiempo,
    dc.idCategoria AS categoria,
    SUM(nt.rrdate::date-dt.fecha::date) AS dias
FROM (SELECT fc.category_id AS category,
        r.rental_date AS rdate,
        r.return_date AS rrdate, *
      FROM rental AS r
      JOIN inventory AS i
      ON r.inventory_id=i.inventory_id
      JOIN film AS f
      ON i.film_id=f.film_id
      JOIN film_category AS fc
      ON fc.film_id = f.film_id
      WHERE r.return_date IS NOT NULL) AS nt
JOIN dwh.categoria AS dc
ON dc.original_categoria_id=nt.category
JOIN dwh.tiempo AS dt
ON dt.fecha=nt.rdate::date
GROUP BY dt.idTiempo, dc.idCategoria;
```

- A continuación, se especificarán las tareas llevadas a cabo por "Carga de tabla de Hechos Inventario".
  - Obtener a través de una función PL/pgSQL los datos del OLTP necesarios para cargar la tabla de hechos Inventario.
  - Para la confección de la tabla de hechos, se tomaron como fuente las tablas “Rental”, “Inventory”, “Film”.
  - A continuación, la función PL/pgSQL:

```
DROP FUNCTION IF EXISTS inventario_hechos();
CREATE OR REPLACE FUNCTION inventario_hechos() RETURNS SETOF RECORD AS $$
DECLARE
    rent_cursor CURSOR FOR
    select t.fecha,t.idtiempo, p.original_pelicula_id, count(*)
    from rental r
    join inventory i on r.inventory_id = i.inventory_id
    join dwh.tiempo t on r.rental_date::date = t.fecha
    join dwh.pelicula p on i.film_id = p.original_pelicula_id
    group by t.fecha, p.original_pelicula_id,t.idtiempo order by t.fecha,p.original_pelicula_id;
    v_fecha TIMESTAMP;
    v_idtiempo INTEGER;
    v_original_pelicula_id INTEGER;
    v_alquileres INTEGER;
    v_rentadas INTEGER;
    v_disponible INTEGER;
    v_total INTEGER;
    v_proporcion INTEGER;
    v_incidencias INTEGER;
    row RECORD;

BEGIN
    OPEN rent_cursor;
    LOOP
     FETCH rent_cursor INTO v_fecha,v_idtiempo,v_original_pelicula_id,v_alquileres;
     EXIT WHEN NOT FOUND;

     --Para calcular el total de copias en inventario
     SELECT COUNT(*) INTO v_total
     FROM inventory i
     WHERE i.film_id=v_original_pelicula_id;

     --Para calcular cuantas estan rentadas en el momento y poder calcular las disponibles: Total-rentadas
     SELECT COUNT(*) INTO v_rentadas
     FROM rental r
     JOIN inventory i on r.inventory_id = i.inventory_id
     AND i.film_id=v_original_pelicula_id
     WHERE r.rental_date::date<=v_fecha
     and (r.return_date::date>v_fecha OR r.return_date is null);

     --Calcular cuantas veces al rentar una pelicula en un determinado dia, estaba sin stock 3 horas antes
     SELECT COUNT(*) FILTER (
         WHERE
        (v_total-
            (SELECT COUNT(*)
            FROM rental ren
            JOIN inventory inv on ren.inventory_id = inv.inventory_id
            AND inv.film_id=v_original_pelicula_id
            WHERE ren.rental_date<=r.rental_date-interval '3 hours'
            and (ren.return_date>r.rental_date-interval '3 hours' OR ren.return_date is null)
            )
        )=0) INTO v_incidencias
     FROM rental r JOIN inventory i on r.inventory_id = i.inventory_id AND i.film_id=v_original_pelicula_id
     WHERE r.rental_date::date=v_fecha group by r.rental_date::date,i.film_id;
```

```
    --Se retorna la primera fila

    SELECT v_idtiempo,
       v_original_pelicula_id,
       v_alquileres, (v_total-v_rentadas),
       round(v_alquileres::decimal/v_total,2)::float,
       v_incidencias into row;
     RETURN next row;
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

b. Actualización

Los datos de las tablas dimensiones “Categoría, Ciudad, Cliente, Película, Personal, Tienda, Tiempo y Tiempopago” y tablas hechos “Alquiler, AlquilerGustos, AlquileresTiendas, Inventario y DesempenoPersonal ” se cargarán de manera incremental teniendo en cuenta las actualizaciones que se realizaron.

El proceso ETL para la actualización del DW es muy similar al de Carga Inicial, pero cuenta con las siguientes diferencias:

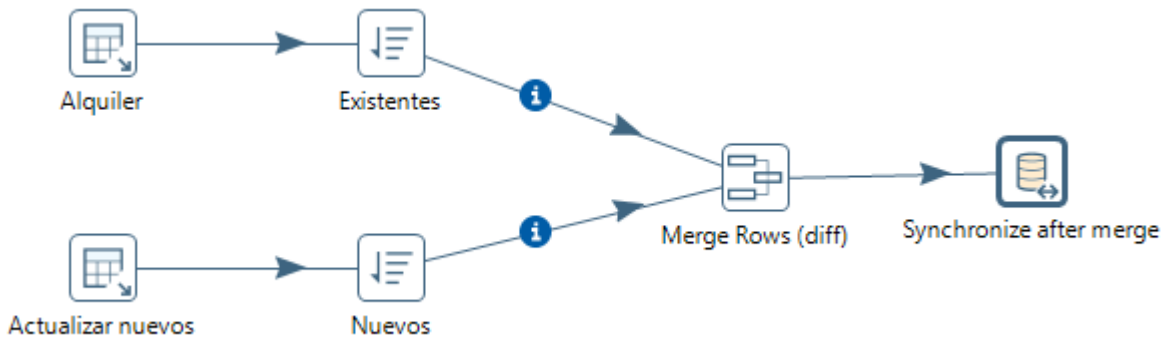
- Para las dimensiones se usará el mismo código de la carga.
- Actualización de la tabla hechos Alquiler: Se procederá a comparar los nuevos datos con los antiguos. Haciendo una consulta a la tabla hechos Alquiler y otra al OLTP, cada consulta se ordenada por las claves primarias y luego se procederá a unir los datos antiguos con los nuevos.

Datos Existentes

```
SELECT
    tiempo_id,
    categoria_id,
    cantidad_dias
FROM dwh.alquiler;
```

Datos Nuevos

```
SELECT dt.idTiempo AS tiempo,
    dc.idCategoria AS categoria,
    SUM(nt.rrdate::date-dt.fecha::date) AS dias
FROM (SELECT fc.category_id AS category,
        r.rental_date AS rdate,
        r.return_date AS rrdate, *
      FROM rental AS r
      JOIN inventory AS i
      ON r.inventory_id=i.inventory_id
      JOIN film AS f
      ON i.film_id=f.film_id
      JOIN film_category AS fc
      ON fc.film_id = f.film_id
      WHERE r.return_date IS NOT NULL) AS nt
JOIN dwh.categoria AS dc
ON dc.original_categoria_id=nt.category
JOIN dwh.tiempo AS dt
ON dt.fecha=nt.rdate::date
GROUP BY dt.idTiempo, dc.idCategoria;
```



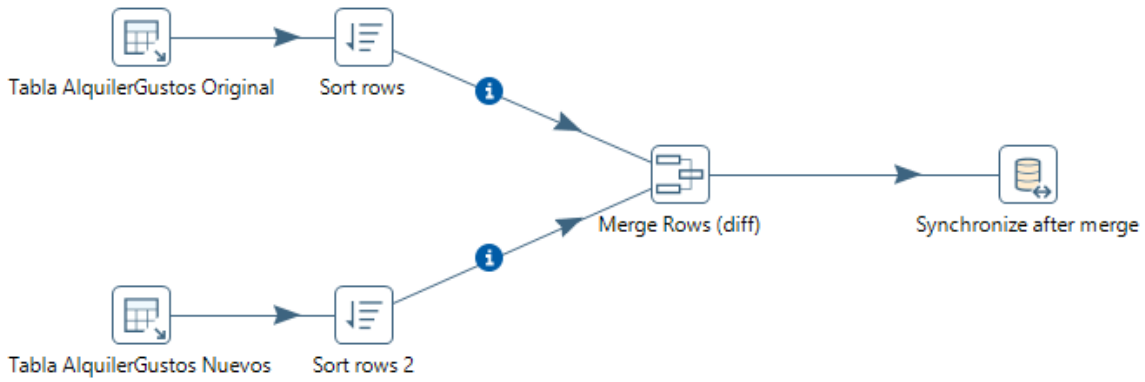
- Actualización de la tabla hechos AlquilerGustos: Se procederá a comparar los nuevos datos con los antiguos. Haciendo una consulta a la tabla hechos AlquilerGustos y otra al OLTP, cada consulta se ordenada por las claves primarias y luego se procederá a unir los datos antiguos con los nuevos.

Datos Existentes

```
SELECT *
FROM dwh.alquilergustos;
```

Datos Nuevos

```
SELECT t.idtiempo as tiempo_id,
    cli.idcliente as cliente_id,
    ciu.idciudad as ciudad_id,
    cat.idcategoria as categoria_id,
    COUNT(r.rental_id) as cantidad_alquiler
FROM rental r
JOIN inventory inv ON (r.inventory_id = inv.inventory_id)
JOIN film f ON(inv.film_id = f.film_id)
JOIN film_category fc ON(fc.film_id = f.film_id)
JOIN customer cus ON(cus.customer_id = r.customer_id)
JOIN address adr ON(adr.address_id = cus.address_id)
JOIN dwh.tiempo t ON (t.fecha = r.rental_date::date)
JOIN dwh.cliente cli ON(cli.original_cliente_id = cus.customer_id)
JOIN dwh.ciudad ciu ON(ciu.original_ciudad_id = adr.city_id)
JOIN dwh.categoria cat ON(cat.original_categoria_id=fc.category_id)
GROUP BY t.idtiempo, cli.idcliente, ciu.idciudad, cat.idcategoria;
```



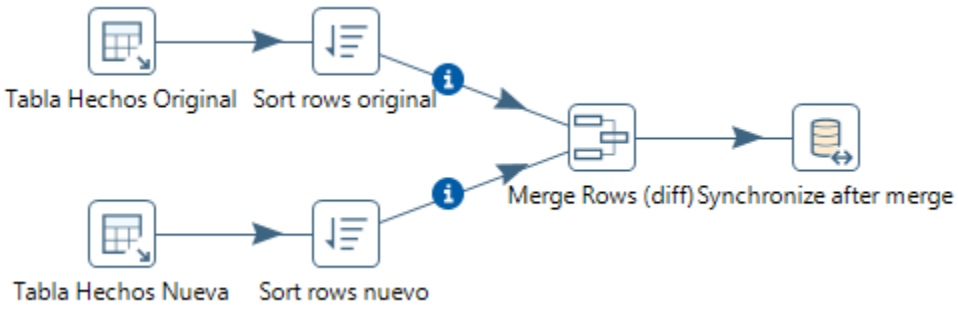
- Actualización de la tabla hechos AlquilerTiendas: Se procederá a comparar los nuevos datos con los antiguos. Haciendo una consulta a la tabla hechos AlquilerTiendas y otra al OLTP, cada consulta se ordenada por las claves primarias y luego se procederá a unir los datos antiguos con los nuevos.

Datos Existentes

```
SELECT
  idtiempo
, idtienda
, idcategoria
, cantidad_alquiler
FROM dwh.alquilertiendas;
```

Datos Nuevos

```
SELECT tp.idtiempo,
    t.idtienda,
    c.idcategoria,
    count(*) as cantidad_alquiler
FROM rental r
JOIN inventory i ON ( r.inventory_id = i.inventory_id)
JOIN dwh.tienda t ON (i.store_id = t.original_tienda_id)
JOIN film_category fc ON (fc.film_id = i.film_id)
JOIN dwh.categoria c ON (c.original_categoria_id = fc.category_id)
JOIN dwh.tiempo tp ON ( tp.fecha = r.rental_date::date )
GROUP BY tp.idtiempo, t.idtienda, c.idcategoria
ORDER BY 1 asc, 2 asc, 3 asc;
```



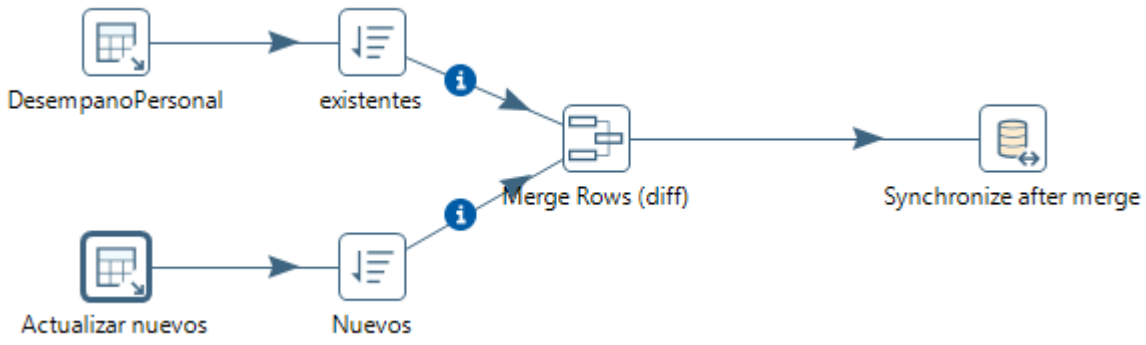
- Actualización de la tabla hechos DesempenoPersonal: Se procederá a comparar los nuevos datos con los antiguos. Haciendo una consulta a la tabla hechos DesempenoPersonal y otra al OLTP, cada consulta se ordenada por las claves primarias y luego se procederá a unir los datos antiguos con los nuevos.

Datos Existentes

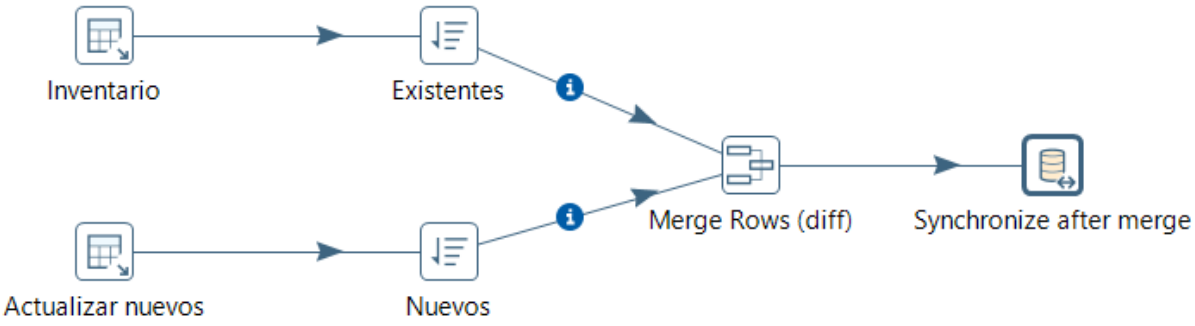
```
SELECT
  personal_id
, tiempopago_id
, cantidad_alquiler
, monto_alquiler
FROM dwh.hechos_desempeno;
```

Datos Nuevos

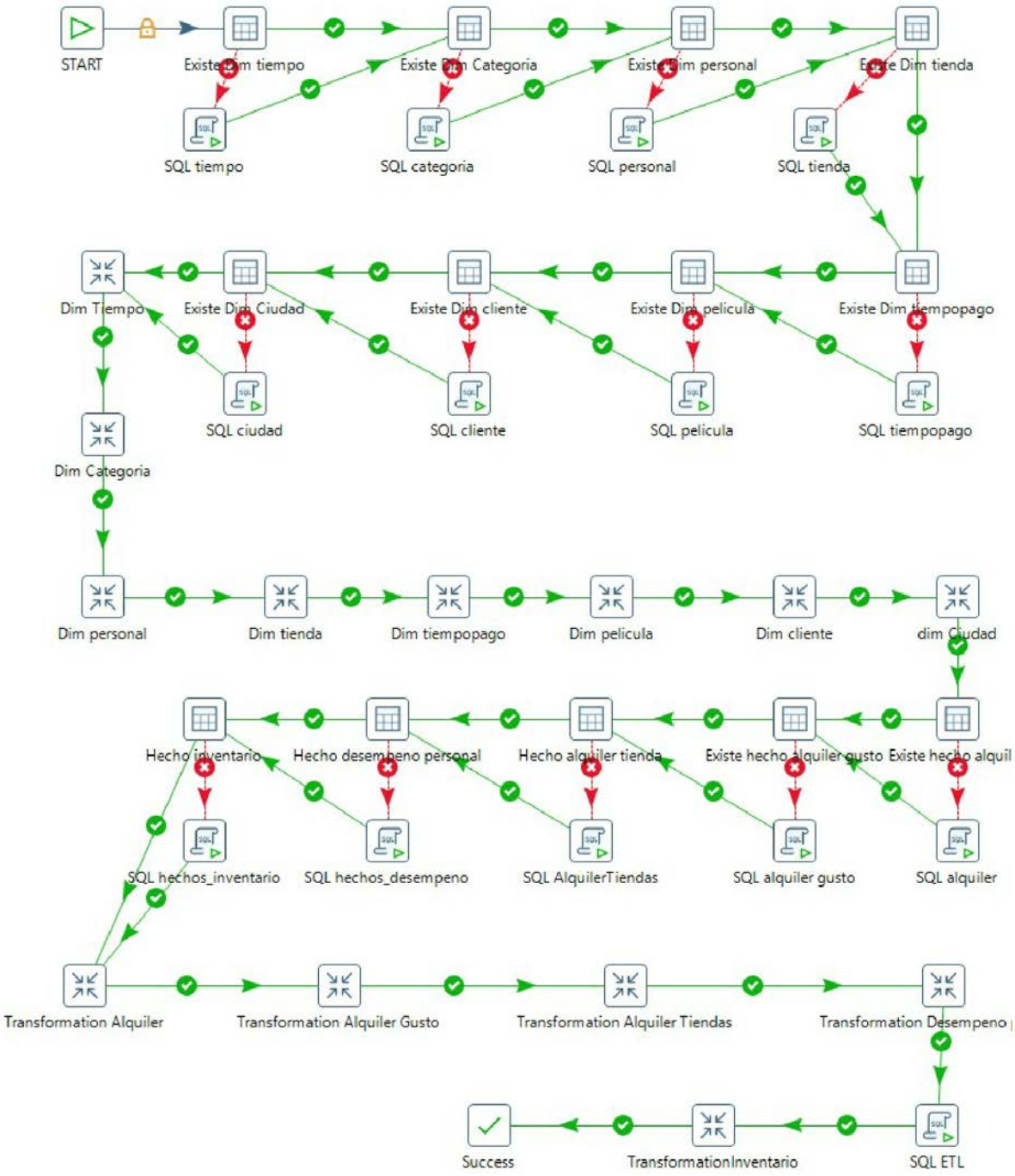
```
SELECT dp.idpersonal as personal_id,
    dt.idtiempopago as tiempopago_id,
    sum((CASE WHEN (desempenno.pago = 0 and desempenno.renta<>-1)
              THEN 1 ELSE 0 END)) as cantidad_alquiler,
    sum(desempenno.pago)::double precision as monto_alquiler
FROM (SELECT rental_date as fecha,
       staff_id as staff,
       0 as pago,
       rental_id as renta
      FROM rental UNION (SELECT payment_date, staff_id, amount, -1
                         FROM payment)) as desempenno
JOIN dwh.personal dp on(desempenno.staff=dp.original_personal_id)
JOIN dwh.tiempopago dt on(desempenno.fecha::date=dt.fecha)
GROUP BY dp.idpersonal, dt.idtiempopago;
```



- Actualización de la tabla hechos Inventario: Se procederá a comparar los nuevos datos con los antiguos. Haciendo una consulta a la tabla hechos Inventario y otra al OLTP, cada consulta se ordenada por las claves primarias y luego se procederá a unir los datos antiguos con los nuevos.



Proceso completo de Carga y Actualización:



Reporte usando Report Design de Pentaho

Se elaboró un reporte utilizando el DataMart "AlquilerGustos" para mostrar los Gustos de los clientes (categorías de películas que alquilan) por zona.

La elaboración y presentación de dichos reportes se resume en el video que se encuentra en la siguiente dirección: https://youtu.be/3BzNywqBfGw

A continuación, se presentan algunas capturas de los reportes realizados:

*"Tendencia de los gustos por **Clientes** en la **ciudad** de Kansas City por la **categoría** Classics"*

*"Tendencia de gustos de **clientes** en el **año** 2005 en el **país** Venezuela por la **categoría** Family"*