# 

# GESTIÓN DE DATOS

Año 2021 - 1° Cuatrimestre

Curso: K3013

Grupo: LAWE

Integrantes:

* Lazarte, Emmanuel - 168-926.5
* Saba Lagos, Alesio - 167-211.3
* Saba Lagos, Leonardo - 167-519.9
* Ramirez Lazo, Willian Eduardo - 167-477.8

Fechas de entrega:

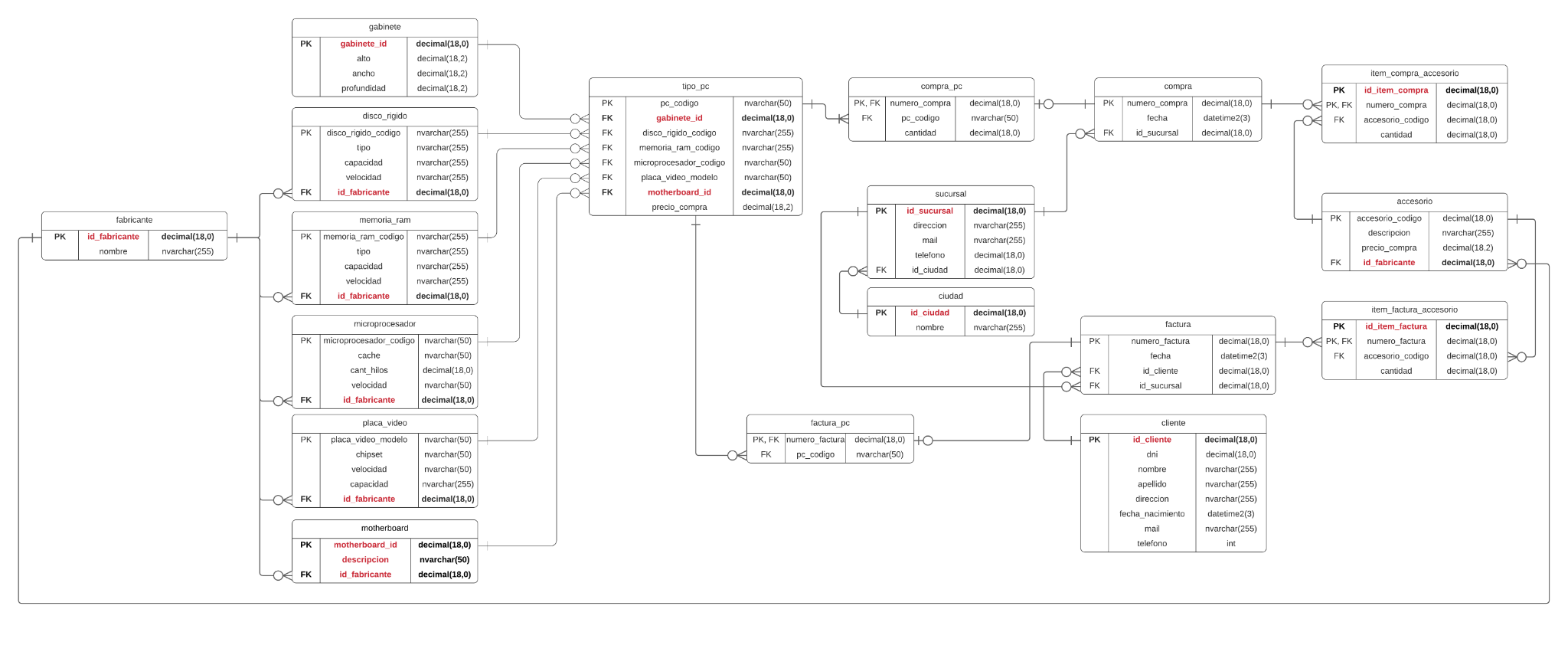
* Entrega N°1 - DER - 04/05/2021
* Entrega N°2 - Modelo de datos y Migración - 26/05/2021
* Entrega N°3 - Modelo Business Intelligence - 30/06/2021

Índice

* 1. Diagrama de Entidad – Relación (Pág. 3)
* 2. Migración - Aspectos Generales (Pág. 10)
  + 2.1 Eliminación de cualquier objeto existente (Pág. 10)
  + 2.2 Creación de objetos necesarios para la migración (Pág. 10)
  + 2.3 Ejecución de Stored Procedures para la migración de datos (Pág. 20)
* 3. Vistas (Pág. 22)
  + 3.1 Vista - Compras PC (Pág. 22)
  + 3.2 Vista - Compra de Accesorios (Pág. 23)
  + 3.3 Vista - Compras Accesorios (Pág. 24)
  + 3.4 Vista - Facturas PC (Pág. 25)
  + 3.5 Vista - Facturas Accesorios (Pág. 26)
  + 3.6 Vista - Ventas de Accesorios (Pág. 27)
  + 3.7 Vista – Clientes (Pág. 28)
  + 3.8 Vista - Tipos de Computadoras existentes (Pág. 28)
* 4. Modelo de Inteligencia de Negocios (BI) (Pág. 29)
  + 4.1 Borrado Previo (Pág. 30)
  + 4.2 Modelo Estrella (Pág. 31)
  + 4.3 Tablas de hecho (Pág. 31)
  + 4.4 Migración hacia el modelo Business Intelligence (Pág. 32)
  + 4.5 Tablas confeccionadas para el modelo de Business Intelligence (Pág. 34)
  + 4.6 Proceso de migración hacia el modelo de Business Intelligence (Pág. 34)
  + 4.7 Tablas de hechos (Pág. 34)
* 5. Migración hacia el Modelo BI (Pág. 36)
  + 5.1 Funciones (Pág. 36)
  + 5.2 Stored Procedures (Pág. 38)
  + 5.3 Creación de índices (Pág. 47)
* 6. Vistas Modelo BI (Pág. 47)
  + 6.1 Vista - PC Compradas / Vendidas por mes y sucursal (Pág. 47)
  + 6.2 Vista - PC - Ganancias por sucursal y mes (Pág. 48)
  + 6.3 Vista - PC – Precio promedio vendido y comprado (Pág. 49)
  + 6.4 Vista - PC – Tiempo en stock promedio (Pág. 49)
  + 6.5 Vista - Accesorio - Máxima cantidad de Stock anual por cada sucursal (Pág. 49)
  + 6.6 Vista - Accesorio - Máxima cantidad de Stock anual por sucursal (Pág. 50)
  + 6.7 Vista - Accesorio - Ganancias anual por sucursal por mes (Pág. 50)
  + 6.8 Vista - Accesorio – Precio promedio vendido y comprado (Pág. 51)
  + 6.9 Vista - Accesorio – Tiempo stock promedio vendido y comprado (Pág. 51)

# **Diagrama de Entidad - Relación**

El siguiente diagrama de entidad - relación representa la estructura del modelo de datos a través del cual se organizan y normalizan los datos de la única tabla provista por la cátedra. Adjuntamos una copia de este en su resolución original en la entrega cuyo nombre es “DER.png”.

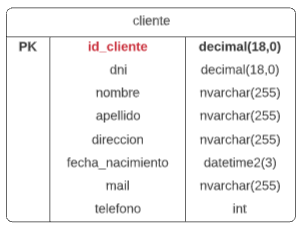


A continuación, procedemos a justificar la creación de las distintas tablas utilizadas en el modelo relacional para la migración de datos de la tabla maestra.

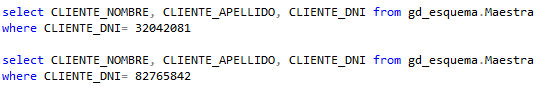
### 

### **Tabla cliente**

Para evitar la redundancia de los datos de un cliente relacionados a las facturas decidimos extraer esos datos y definir y crear una única tabla CLIENTE, compuesta por los siguientes campos:



Decidimos no usar el DNI de una persona como PRIMARY KEY debido a que detectamos incoherencias como DNI duplicados asociados a personas con distintos nombre y apellido o personas con mismos datos, pero distinto DNI. Detallamos un ejemplo de la incoherencia:





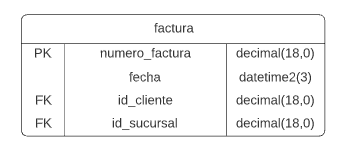


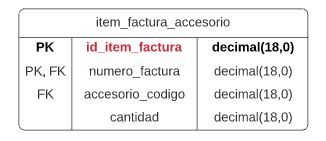
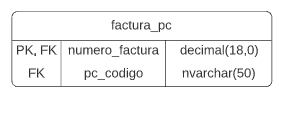
### 

Por eso se decide establecer un **id\_cliente** como **PRIMARY KEY** que será generado por un **IDENTITY.**

### **Tabla factura**

Decidimos estructurar las facturas en forma de Tipo - Subtipo, aquellos atributos que son inherentes a cualquier factura como fecha, cliente, sucursal formarán parte de tabla factura. Luego aquellos atributos que sean inherentes a la facturade una pc o a la factura de un accesorio formarán parte de las tablas **factura\_pc** y **item\_factura\_accesorio** respectivamente.





La tabla factura tendrá como **PRIMARY KEY** a **numero\_factura** y servirá como **FOREIGN KEY** en la tabla **item\_factura\_accesorio** y la tabla **factura\_pc**, para poder referenciar a dicha factura.

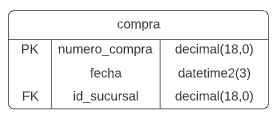
Decidimos para la tabla **item\_factura\_accesorio** agregar la **PRIMARY KEY id\_item\_factura** debido a que se considera posible que un cliente compre varios accesorios y queden reflejados en una sola factura, así podremos identificar a cada ítem que pertenece a una factura.

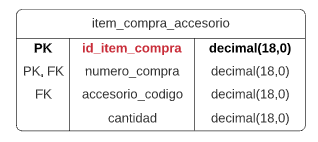
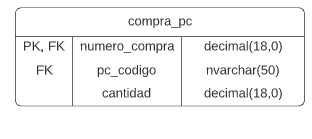
#### **Explicación de las Foreign key**

* **numero\_factura** nos permitirá referenciar la factura correspondiente a la venta realizada.
* **id\_cliente** nos permitirá referenciar al cliente que realiza la compra.
* **id\_sucursal** nos permitirá referenciar a la sucursal donde se realiza la venta.
* **pc\_codigo** nos permitirá referenciar a la pc que se está vendiendo.
* **accesorio\_codigo** nos permitirá referenciar al accesorio que se está vendiendo.

### **Tabla compra**

Decidimos estructurar las compras en forma de Tipo - Subtipo, aquellos atributos que son inherentes a cualquier compra como fecha y sucursal formarán parte de la tabla **compra**. Luego aquellos atributos que sean inherente a la compra de una pc o a la compra de accesorios formarán parte de las tablas **compra\_pc** o **item\_compra\_accesorio** respectivamente.





La tabla compra tendrá como **PRIMARY KEY** a **numero\_compra** y servirá como **FOREIGN KEY** en la tabla **item\_compra\_accesorio** y la tabla **compra\_pc**, para poder referenciar a dicha compra.

Decidimos para la tabla **item\_compra\_accesorio** agregar la **PRIMARY KEY id\_item\_compra** debido a que se considera posible que una sucursal compre varios accesorios en una sola compra, así podremos identificar a cada ítem que pertenece a una compra.

#### **Explicación de las Foreign key**

* **numero\_compra** nos permitirá referenciar la compra correspondiente a la compra realizada por parte de la sucursal.
* **id\_sucursal** nos permitirá referenciar a la sucursal que realizó la compra.
* **pc\_codigo** nos permitirá referenciar a la pc que se está comprando.
* **accesorio\_codigo** nos permitirá referenciar al accesorio que se está comprando.

### **Tabla fabricante**

Decidimos normalizar el atributo **nombre** de un fabricante, que formaba parte de los distintos componentes de una pc y de los accesorios, decidimos extraerlo y armar una tabla **fabricante** para ello.

Esto nos permite que ante un eventual cambio en el nombre será más sencillo realizar las modificaciones sobre la tabla fabricante en vez de tener que recorrer las tablas de componentes y accesorios para realizar el cambio.

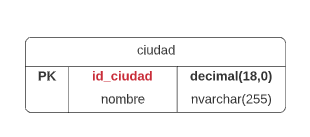
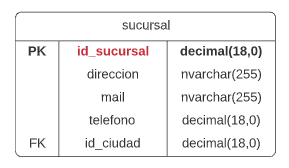


A través del atributo **id\_fabricante** se podrá referenciar al fabricante de un determinado componente o accesorio.

### **Tabla sucursal**

Para evitar la redundancia de los datos de una sucursal correspondientes a una compra o una factura decidimos extraer todos esos datos (dirección, email, teléfono, ciudad) y crear la tabla **sucursal.**

Como un paso adicional decidimos normalizar el dato de ciudad porque creemos que a futuro podríamos tener más de una sucursal en una misma ciudad, y el atributo de ciudad termina siendo redundante.

****

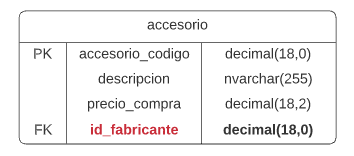
Para la tabla ciudad establecemos la **PRIMARY KEY id\_ciudad**, y servirá como **FOREIGN KEY** dentro de la tabla sucursal para poder referencia a la tabla ciudad.

### **Tabla accesorio**

Para evitar la redundancia de datos de un accesorio correspondientes a una compra o factura decidimos extraer todos esos datos como la descripción y su código para definir y crear la tabla **accesorio**

Además, agregamos el campo **precio\_compra** ya que creímos importante que cada accesorio refleje su precio unitario y evitar el problema que ante un aumento o disminución del precio, sea una tarea difícil.

También decidimos agregar el atributo fabricante, a pesar de que en la base de datos no existe, en la consigna del tp se menciona que en una compra de un accesorio se indica su fabricante.



Para la tabla accesorio establecemos como **PRIMARY KEY** el campo **accesorio\_código**, debido a que es un valor que logra identificar unívocamente a un **accesorio** y no se repite.

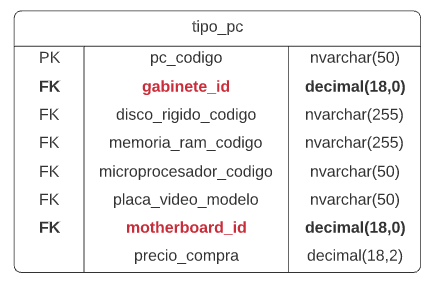
### **Tabla tipo\_pc**

Para evitar la redundancia de datos que generaría los datos de una pc en una compra o factura, decidimos definir y crear la tabla **tipo\_pc** la cual contendrá un conjunto de componentes y su precio total.

Un tipo\_pc contendrá un gabinete, una memoria ram, un microprocesador, una placa de video, una motherboard y el precio de compra de este modelo de pc.

Decidimos generar la abstracción **gabinete** como una tabla aparte donde se detallarán las medidas de una pc.

También decidimos agregar el atributo **motherboard\_id**, a pesar de que en la base de datos no existía, se menciona en la consigna, por eso lo agregamos como una **FOREIGN KEY** que más adelante podría hacer referencia a la tabla de motherboard.



Para la tabla **accesorio** establecemos como **PRIMARY KEY** el campo **pc\_codigo**, debido a que es un valor que logra identificar unívocamente a un **tipo\_pc** y no se repite.

#### **Explicación de las Foreign key**

* **gabinete\_id** nos permite referenciar al gabinete de una pc, que contendrá sus medidas.
* **disco\_rigido\_codigo** nos permite referenciar al disco rígido de una pc.
* **memoria\_ram\_codigo** nos permite referenciar a la memoria ram de una pc.
* **microprocesador\_codigo** nos permite referenciar al microprocesador de una pc.
* **placa\_video\_modelo** nos permite referenciar al placa video de una pc.
* **motherboard\_id** nos permite referenciar al motherboard de una pc.

.

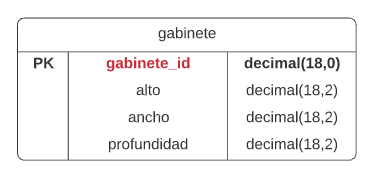
### **Componentes**

Para evitar la redundancia de datos de los distintos componentes que pertenecen a una pc, definimos y creamos las correspondientes tablas para cada uno de ellos.

Cada uno de los componentes a excepción del gabinete, tendrá como **FOREIGN KEY** a **id\_fabricante** el cual nos permitirá referenciar al nombre del fabricante del componente

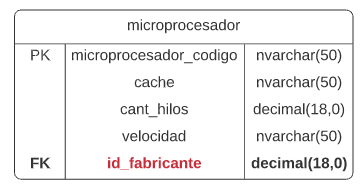
#### **Tabla gabinete**

Es una abstracción de las medidas de una pc, se detallarán el alto, ancho y profundidad de una pc.



#### **Tabla microprocesador**

Tabla que contiene todos los atributos propios de un microprocesador, como su caché, cantidad de hilos y velocidad.



#### **Tabla disco\_rigido**

Tabla que contiene todos los atributos propios de un microprocesador, como su tipo, capacidad y velocidad.



#### **Tabla placa\_video**

Tabla que contiene todos los atributos propios de una placa de video, como su chipset, velocidad y capacidad.



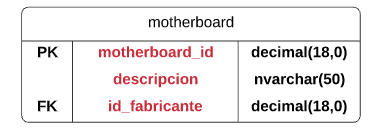
#### **Tabla memoria\_ram**

Tabla que contiene todos los atributos propios de una memoria ram, como su tipo, capacidad y velocidad.



#### **Tabla motherboard**

Tabla que decidimos agregar a pesar de que en la base de datos no existían datos relacionados a este tipo de componente, se menciona en la consigna que existen las motherboards.



# Migración - Aspectos Generales

En este apartado, se detallarán las decisiones tomadas en el desarrollo de la migración de datos en orden tal como se encuentra en el script de creación inicial.

## 2.1 Eliminación de cualquier objeto existente

Con el objeto de poder automatizar la creación de las distintas abstracciones que implementamos mediante estructuras varias, y repetir la operación de creación de dichas estructuras procedemos a eliminar toda aparición de objeto que será generado por el script. Dicha tarea engloba tanto a las tablas, funciones, vistas, procedimientos e índices. Teniendo en cuenta que el último objeto por borrar será el esquema.

## **Secuencia de Eliminación de tablas**

En el caso de las tablas tuvimos en cuenta que la secuencia de borrado de estas era fundamental para evitar tener inconvenientes con la **regla de integridad referencial**, donde entablamos las relaciones mediante las claves foráneas respetando el dominio del problema en cuestión.

## 2.2 Creación de objetos necesarios para la migración

**Creación del esquema**

Dado que el enunciado pide que cada objeto sea creado en el esquema cuyo nombre debe corresponder con el del grupo procedemos a crear un **esquema** llamado **“LAWE”.** Cada objeto de los creados a continuación será creado dentro este esquema.

**Creación de las tablas**

## En esta sección procedemos a crear las tablas que corresponden al diagrama de entidad – relación detallado anteriormente las cuales contendrán los datos migrados de la tabla maestra.

Tuvimos que establecer un orden para la creación de las tablas ya que algunos atributos de estas referencian a través de una **FOREING KEY** a otras tablas, por lo tanto, primero deben ser creadas las tablas que serán referenciadas y luego las que hacen referencia.

En esta sección se establecen las **PRIMARY KEY** de cada tabla cuya conformación se encuentra justificada en el apartado de diagrama entidad – relación.

En la creación de las tablas establecimos los **CONSTRAINTS** necesarios de los atributos como **NOT NULL** y **IDENTITY**.

Consideramos que ningún atributo de las tablas podía tener valores nulos y debido a esto se aplica el constraint **NOT NULL**, excepto los atributos de la tabla **motherboard** yel atributo **id\_fabricante** de la tabla **accesorio** ya que no existían datos en la tabla maestra sobre estos.

Utilizamos el constraint **IDENTITY** para establecer que los atributos que conforman **PRIMARY KEYS** se incrementen en una unidad su valor cada vez que se inserte un registro en una tabla a la hora de realizar la migración de datos.

**Creación de índices**

Durante el proceso de migración, detectamos que los tiempos de ejecución y respuesta eran prolongados ante ciertas consultas tales como el obtener los datos de los clientes. Para evitar este exceso de procesamiento, recurrimos a la creación de un índice sobre la tabla **cliente**.

Por lo tanto, creamos el índice **ix\_cliente** sobre la tabla **cliente** según los atributos **dni** y **apellido**, para así mejorar el tiempo de búsqueda de un cliente. Concluimos que era necesario ya que mejoró notablemente la performance al momento de realizar la migración de datos.

**Creación de funciones auxiliares**

A modo de evitar disponer de operaciones repetitivas que podían ser representadas mediante subqueries extensas y en algunos casos quizás hasta complejas, recurrimos a implementar funciones propias que nos permite desarrollar el motor. La utilización de estas funciones auxiliares mejora la performance de la migración.

A continuación, detallamos las funciones auxiliares creadas:

* **OBTENER\_ID\_FABRICANTE**, esta función nos permite obtener el número de id de un fabricante a partir de su nombre. Utilizamos esta función para la migración de datos de los distintos componentes de computadora
* **OBTENER\_ID\_GABINETE**, esta función nos permite obtener el número de id de un gabinete a partir de su alto, ancho y profundidad, y nos permite la migración de datos a la tabla **tipo\_pc**
* **OBTENER\_ID\_CIUDAD**, esta función nos permite obtener el número de id de una ciudad a partir de su nombre. Utilizamos esta función para la migración de datos a la tabla **sucursal**
* **OBTENER\_ID\_SUCURSAL** nos brinda el número de id de una sucursal a partir de su dirección, facilitó la migración de datos a las tablas **compra** y **factura**
* **OBTENER\_ID\_CLIENTE**, esta otra función nos permite obtener el número de id de un cliente a partir de sus datos, facilitando la migración de datos a la tabla **factura**
* **OBTENER\_NOMBRE\_FABRICANTE,** esta función nos permite obtener el nombre de un fabricante a partir de su número de id. Utilizamos esta función en la vista **modelos\_de\_PC** y evitamos múltiples joins innecesarios.

**Creación de Stored Procedures**

Detallamos los procedimientos almacenados que creamos para luego ser utilizados para la migración de datos de la tabla maestra al modelo relacional.

**Migración de fabricantes**

Se obtienen los distintos nombres de fabricantes de los componentes de computadoras de la tabla maestra, utilizamos la sentencia UNION porque los fabricantes de cada tipo de componente se encuentran en distintos atributos de la tabla maestra y también evitamos migrar fabricantes repetidos en caso de que estos fabriquen más de un tipo de componente.

Los datos obtenidos son insertados en la tabla **fabricante,** el atributo **id\_fabricante** posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada fabricante insertado se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_fabricante AS

BEGIN

  INSERT INTO LAWE.fabricante (nombre)

    -- La sentencia UNION realiza un ORDER BY por el campo 'nombre'

    SELECT DISTINCT MEMORIA\_RAM\_FABRICANTE FROM gd\_esquema.Maestra WHERE MEMORIA\_RAM\_FABRICANTE IS NOT NULL

        UNION

    SELECT DISTINCT DISCO\_RIGIDO\_FABRICANTE FROM gd\_esquema.Maestra WHERE DISCO\_RIGIDO\_FABRICANTE IS NOT NULL

        UNION

    SELECT DISTINCT MICROPROCESADOR\_FABRICANTE FROM gd\_esquema.Maestra WHERE MICROPROCESADOR\_FABRICANTE IS NOT NULL

        UNION

    SELECT DISTINCT PLACA\_VIDEO\_FABRICANTE FROM gd\_esquema.Maestra WHERE PLACA\_VIDEO\_FABRICANTE IS NOT NULL

END

GO

**Migración de gabinetes**

Se obtienen las medidas (alto, ancho y profundidad) de los distintos modelos de gabinete diferenciando estos por el código de las pc y son insertados en la tabla temporal **#temp\_gabinete.**

Luego los datos obtenidos excepto el código de la pc son insertados en la tabla **gabinete,** el atributo **id\_gabinete** posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada modelo de gabinete insertado se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_gabinete AS

BEGIN

  SELECT DISTINCT PC\_CODIGO, PC\_ALTO, PC\_ANCHO, PC\_PROFUNDIDAD INTO #temp\_gabinete

    FROM gd\_esquema.Maestra WHERE PC\_CODIGO IS NOT NULL

  INSERT INTO LAWE.gabinete (alto,ancho,profundidad)

    SELECT PC\_ALTO, PC\_ANCHO, PC\_PROFUNDIDAD FROM #temp\_gabinete

  DROP TABLE #temp\_gabinete

END

GO

**Migración de componentes de las computadoras**

Los procedimientos almacenados utilizados para la migración de datos de los distintos tipos de componentes de computadora tienen una estructura muy similar. Se obtienen las características de cada modelo de componente diferenciados por su código o modelo, y el **id\_fabricante** mediante la función **OBTENER\_ID\_FABRICANTE** a partir del nombre del fabricante del componente.

Luego se insertan los datos obtenidos en las correspondientes tablas de componentes con sus respectivas características insertando el código o modelo del componente según corresponda, estos representan las **PRIMARY KEYS** de estas tablas.

CREATE PROCEDURE LAWE.migrar\_disco\_rigido AS

BEGIN

  INSERT INTO LAWE.disco\_rigido (disco\_rigido\_codigo,tipo,capacidad,velocidad,id\_fabricante)

    SELECT

        DISTINCT DISCO\_RIGIDO\_CODIGO,

        DISCO\_RIGIDO\_TIPO,

        DISCO\_RIGIDO\_CAPACIDAD,

        DISCO\_RIGIDO\_VELOCIDAD,

        LAWE.OBTENER\_ID\_FABRICANTE(DISCO\_RIGIDO\_FABRICANTE)

    FROM gd\_esquema.Maestra

    WHERE DISCO\_RIGIDO\_CODIGO IS NOT NULL

END

CREATE PROCEDURE LAWE.migrar\_memoria\_ram AS

BEGIN

  INSERT INTO LAWE.memoria\_ram (memoria\_ram\_codigo,tipo,capacidad,velocidad,id\_fabricante)

    SELECT

        DISTINCT MEMORIA\_RAM\_CODIGO,

        MEMORIA\_RAM\_TIPO,

        MEMORIA\_RAM\_CAPACIDAD,

        MEMORIA\_RAM\_VELOCIDAD,

        LAWE.OBTENER\_ID\_FABRICANTE(MEMORIA\_RAM\_FABRICANTE)

    FROM gd\_esquema.Maestra

    WHERE MEMORIA\_RAM\_CODIGO IS NOT NULL

END

CREATE PROCEDURE LAWE.migrar\_microprocesador AS

BEGIN

  INSERT INTO LAWE.microprocesador (microprocesador\_codigo,cache,cant\_hilos,velocidad,id\_fabricante)

    SELECT

        DISTINCT MICROPROCESADOR\_CODIGO,

        MICROPROCESADOR\_CACHE,

        MICROPROCESADOR\_CANT\_HILOS,

        MICROPROCESADOR\_VELOCIDAD,

        LAWE.OBTENER\_ID\_FABRICANTE(MICROPROCESADOR\_FABRICANTE)

    FROM gd\_esquema.Maestra

    WHERE MICROPROCESADOR\_CODIGO IS NOT NULL

END

CREATE PROCEDURE LAWE.migrar\_placa\_video AS

BEGIN

  INSERT INTO LAWE.placa\_video (placa\_video\_modelo,chipset,velocidad,capacidad,id\_fabricante)

    -- con la función REPLACE eliminamos la palabra 'Modelo' de cada registro de placa de video

    SELECT

        DISTINCT REPLACE(PLACA\_VIDEO\_MODELO,'Modelo ',''),

        PLACA\_VIDEO\_CHIPSET,

        PLACA\_VIDEO\_VELOCIDAD,

        PLACA\_VIDEO\_CAPACIDAD,

        LAWE.OBTENER\_ID\_FABRICANTE(PLACA\_VIDEO\_FABRICANTE)

    FROM gd\_esquema.Maestra

    WHERE PLACA\_VIDEO\_MODELO IS NOT NULL

END

**Migración de modelos de computadoras**

Se obtienen los datos correspondientes a cada modelo de computadora distinto, el **id\_gabinete** se obtiene a través de la función **OBTENER\_ID\_GABINETE** a partir del ancho, alto y profundidad del mismo. El campo COMPRA\_PRECIO representa el precio al cual se compró una unidad de un determinado modelo de computadora.

Luego se insertan los datos obtenidos en la tabla **tipo\_pc** con su respectivo código de pc el cual representa la **PRIMARY KEY** de la tabla. En el atributo **motherboard\_id** se inserta un valor **NULL** porque no existen los datos de este componente, pero la consigna menciona que una computadora tiene asignado una motherboard.

CREATE PROCEDURE LAWE.migrar\_tipo\_pc AS

BEGIN

  INSERT INTO LAWE.tipo\_pc (pc\_codigo, gabinete\_id, disco\_rigido\_codigo, memoria\_ram\_codigo,  microprocesador\_codigo, placa\_video\_modelo, motherboard\_id, precio\_compra)

    SELECT

DISTINCT PC\_CODIGO,

LAWE.OBTENER\_ID\_GABINETE(PC\_ANCHO, PC\_ALTO, PC\_PROFUNDIDAD),

DISCO\_RIGIDO\_CODIGO,

MEMORIA\_RAM\_CODIGO,

MICROPROCESADOR\_CODIGO,

-- con la función REPLACE eliminamos la palabra 'Modelo' de cada registro de placa de video

REPLACE(PLACA\_VIDEO\_MODELO,'Modelo ',''),

NULL,

COMPRA\_PRECIO

    FROM gd\_esquema.Maestra

    WHERE PC\_CODIGO IS NOT NULL AND COMPRA\_NUMERO IS NOT NULL

END

**Migración de ciudades**

Se obtienen los nombres de las distintas ciudades existentes en la tabla maestra. Los datos obtenidos son insertados en la tabla **ciudad.** El atributo **id\_ciudad** posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada ciudad insertada se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_ciudad AS

BEGIN

  INSERT INTO LAWE.ciudad(nombre)

    SELECT DISTINCT CIUDAD FROM gd\_esquema.Maestra

END

**Migración de sucursales**

Se obtienen los datos que corresponden a cada sucursal distinta, diferenciamos las sucursales por su dirección ya que no existe más de una sucursal con la misma dirección. El atributo **id\_ciudad** se obtiene a través de la función **OBTENER\_ID\_CIUDAD** a partir del nombre de la ciudad.

Los datos obtenidos son insertados en la tabla **sucursal.** El atributo **id\_sucursal** posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada sucursal insertada se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_sucursal AS

BEGIN

  INSERT INTO LAWE.sucursal(direccion,mail,telefono,id\_ciudad)

    SELECT

DISTINCT SUCURSAL\_DIR,

SUCURSAL\_MAIL,

SUCURSAL\_TEL,

LAWE.OBTENER\_ID\_CIUDAD(CIUDAD)

FROM gd\_esquema.Maestra

END

**Migración de compras**

Se obtienen aquellos datos que son inherentes a cualquier compra (numero de la compra, fecha de la compra y sucursal que realizó la compra) indistintamente de si estos corresponden a la compra de una pc o a la compra de accesorios. El atributo **id\_sucursal** se obtiene a través de la función **OBTENER\_ID\_SUCURSAL** a partir de la dirección de la sucursal a la cual corresponde la compra**.**

Los datos obtenidos son insertados en la tabla **compra,** insertando el atributo **COMPRA\_NUMERO** de la tabla maestra en el atributo **numero\_compra** de la tabla **compra** ya que estos representan la **PRIMARY KEY** de esta tabla.

CREATE PROCEDURE LAWE.migrar\_compra AS

BEGIN

  INSERT INTO LAWE.compra (numero\_compra, fecha, id\_sucursal)

    SELECT

DISTINCT COMPRA\_NUMERO,

COMPRA\_FECHA,

LAWE.OBTENER\_ID\_SUCURSAL(SUCURSAL\_DIR)

    FROM gd\_esquema.Maestra

    WHERE COMPRA\_NUMERO IS NOT NULL

END

**Migración de compras de computadoras**

Se obtienen los datos relacionados a compras de computadoras. Los datos obtenidos son insertados en la tabla **compra\_pc,** insertando el atributo **COMPRA\_NUMERO** de la tabla maestra en el atributo **numero\_compra** de la tabla **compra\_pc** ya que estos representan la **PRIMARY KEY** de esta tabla.

CREATE PROCEDURE LAWE.migrar\_compra\_pc AS

BEGIN

  INSERT INTO LAWE.compra\_pc(numero\_compra, pc\_codigo, cantidad)

    SELECT COMPRA\_NUMERO,PC\_CODIGO, COMPRA\_CANTIDAD

    FROM gd\_esquema.Maestra

    WHERE COMPRA\_NUMERO IS NOT NULL AND PC\_CODIGO IS NOT NULL

END

**Migración de accesorios**

Se obtienen los datos correspondiente a cada accesorio distinto existente. Los datos obtenidos son insertados en la tabla **accesorio.**

Se inserta el atributo **ACCESORIO\_CODIGO** de la tabla maestra en el atributo **accesorio\_codigo** de la tabla **accesorio** ya que estos representan la **PRIMARY KEY** de esta tabla.

En el atributo **id\_fabricante** de cada accesorio se inserta el valor **NULL** ya que no existen los datos de los fabricantes de los accesorios, pero la consigna menciona que deben tener asignado un fabricante.

CREATE PROCEDURE LAWE.migrar\_accesorio AS

BEGIN

  INSERT INTO LAWE.accesorio(accesorio\_codigo, descripcion, precio\_compra, id\_fabricante)

-- insertamos NULL en el campo "id\_fabricante"

-- porque los datos sobre los fabricantes de los accesorios son inexistentes

    SELECT DISTINCT ACCESORIO\_CODIGO, AC\_DESCRIPCION, COMPRA\_PRECIO, NULL

    FROM gd\_esquema.Maestra

    WHERE COMPRA\_NUMERO IS NOT NULL AND ACCESORIO\_CODIGO IS NOT NULL

END

**Migración de compras de accesorios**

Se obtienen los datos correspondientes a cada ítem de las compras de accesorios ya que es posible que una sucursal compre más de un tipo de accesorio por compra. Los datos obtenidos son insertados en la tabla **item\_compra\_accesorio**.

La **PRIMARY KEY** de la tabla **item\_compra\_accesorio** está compuesta por el atributo **numero\_compra** (el cual corresponde al valor del atributo **COMPRA\_NUMERO**)yporel atributo **id\_item\_compra,** este último posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada ítem insertado se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_item\_compra\_accesorio AS

BEGIN

  INSERT INTO LAWE.item\_compra\_accesorio(numero\_compra, accesorio\_codigo, cantidad)

    SELECT COMPRA\_NUMERO, ACCESORIO\_CODIGO, COMPRA\_CANTIDAD

    FROM gd\_esquema.Maestra

    WHERE COMPRA\_NUMERO IS NOT NULL AND ACCESORIO\_CODIGO IS NOT NULL

END

**Migración de clientes**

Se obtienen los datos de los clientes, pero ya que estos no pueden ser diferenciados por ningún atributo existente en la tabla maestra deben ser agrupados por sus datos para poder asegurar la unicidad de los clientes en la tabla. Los datos son insertados en la tabla **cliente**. El atributo **id\_cliente** posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada cliente insertado se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_cliente AS

BEGIN

  INSERT INTO

 LAWE.cliente(dni, nombre, apellido, direccion, fecha\_nacimiento, mail, telefono)

    SELECT

CLIENTE\_DNI,

CLIENTE\_NOMBRE,

CLIENTE\_APELLIDO,

CLIENTE\_DIRECCION,

CLIENTE\_FECHA\_NACIMIENTO,

CLIENTE\_MAIL,

CLIENTE\_TELEFONO

    FROM gd\_esquema.Maestra

    WHERE CLIENTE\_DNI IS NOT NULL

    GROUP BY CLIENTE\_DNI, CLIENTE\_NOMBRE, CLIENTE\_APELLIDO, CLIENTE\_DIRECCION,

 CLIENTE\_FECHA\_NACIMIENTO, CLIENTE\_MAIL, CLIENTE\_TELEFONO

END

**Migración de facturas**

Se obtienen aquellos datos que son inherentes a cualquier factura (numero de la factura, fecha de la factura, el cliente involucrado en la venta y la sucursal que realizó la venta) indistintamente de si estos corresponden a la venta de una pc o a la venta de accesorios.

El atributo **id\_cliente** se obtiene a través de la función **OBTENER\_ID\_CLIENTE** a partir de los datos del cliente involucrado en la venta.El atributo **id\_sucursal** se obtiene a través de la función **OBTENER\_ID\_SUCURSAL** a partir de la dirección de la sucursal a la cual corresponde la venta**.**

Los datos obtenidos son insertados en la tabla **factura,** insertando el atributo **FACTURA\_NUMERO** de la tabla maestra en el atributo **numero\_factura** de la tabla **factura** ya que estos representan la **PRIMARY KEY** de esta tabla.

CREATE PROCEDURE LAWE.migrar\_factura AS

BEGIN

  INSERT INTO LAWE.factura(numero\_factura, fecha, id\_cliente, id\_sucursal)

    SELECT DISTINCT FACTURA\_NUMERO,

        FACTURA\_FECHA,

        LAWE.OBTENER\_ID\_CLIENTE(CLIENTE\_DNI,CLIENTE\_NOMBRE,CLIENTE\_APELLIDO,

CLIENTE\_DIRECCION,CLIENTE\_FECHA\_NACIMIENTO,CLIENTE\_MAIL,CLIENTE\_TELEFONO),

        LAWE.OBTENER\_ID\_SUCURSAL(SUCURSAL\_DIR)

    FROM gd\_esquema.Maestra

    WHERE FACTURA\_NUMERO IS NOT NULL

END

**Migración de facturas de computadoras**

Se obtienen los datos relacionados a facturas de computadoras. Los datos obtenidos son insertados en la tabla **factura\_pc,** insertando el atributo **FACTURA\_NUMERO** de la tabla maestra en el atributo **numero\_factura** de la tabla **factura\_pc** ya que estos representan la **PRIMARY KEY** de esta tabla.

CREATE PROCEDURE LAWE.migrar\_factura\_pc AS

BEGIN

  INSERT INTO LAWE.factura\_pc(numero\_factura, pc\_codigo)

    SELECT FACTURA\_NUMERO, PC\_CODIGO

    FROM gd\_esquema.Maestra

    WHERE FACTURA\_NUMERO IS NOT NULL AND PC\_CODIGO IS NOT NULL

END

**Migración de facturas de accesorios**

Se obtienen los datos correspondientes a cada ítem de las facturas de accesorios ya que es posible que una sucursal venda más de un tipo de accesorio en la misma venta. Los datos obtenidos son insertados en la tabla **item\_factura\_accesorio**. Ya que existía más de un registro por tipo de accesorio para la misma factura decidimos agruparlos por los atributos **FACTURA\_NUMERO** y **ACCESORIO\_CODIGO**, de esta manera se puede calcular que cantidad de cada accesorio se vendió en una determinada venta.

La **PRIMARY KEY** de la tabla **item\_factura\_accesorio** está compuesta por el atributo **numero\_factura** (el cual corresponde al valor del atributo **FACTURA\_NUMERO**)yporel atributo **id\_item\_factura,** este último posee el constraint **IDENTITY(1,1)** con la finalidad de que a cada ítem insertado se le asigne un id único.

CREATE PROCEDURE LAWE.migrar\_item\_factura\_accesorio AS

BEGIN

  INSERT INTO LAWE.item\_factura\_accesorio(numero\_factura, accesorio\_codigo, cantidad)

    SELECT

        FACTURA\_NUMERO,

        ACCESORIO\_CODIGO,

-- representa la cantidad de accesorios de ese codigo que se vendieron en la factura

        COUNT(\*)

    FROM gd\_esquema.Maestra

    WHERE FACTURA\_NUMERO IS NOT NULL AND ACCESORIO\_CODIGO IS NOT NULL

    GROUP BY FACTURA\_NUMERO, ACCESORIO\_CODIGO

END

# Ejecución de Stored Procedures para la migración de datos

Para realizar la migración de datos de la tabla maestra deben ser ejecutados los stored procedures creados anteriormente respetando el siguiente orden:

EXEC LAWE.migrar\_fabricante

EXEC LAWE.migrar\_gabinete

EXEC LAWE.migrar\_disco\_rigido

EXEC LAWE.migrar\_memoria\_ram

EXEC LAWE.migrar\_microprocesador

EXEC LAWE.migrar\_placa\_video

EXEC LAWE.migrar\_tipo\_pc

EXEC LAWE.migrar\_ciudad

EXEC LAWE.migrar\_sucursal

EXEC LAWE.migrar\_compra

EXEC LAWE.migrar\_compra\_pc

EXEC LAWE.migrar\_accesorio

EXEC LAWE.migrar\_item\_compra\_accesorio

EXEC LAWE.migrar\_cliente

EXEC LAWE.migrar\_factura

EXEC LAWE.migrar\_factura\_pc

EXEC LAWE.migrar\_item\_factura\_accesorio

# Vistas

Se crearon las siguientes vistas para de modo que se pueda acceder a la información de ellas de manera inmediata, evitando tener que realizar reiteradas consultas.

## **3.1 Vista - Compras PC**

El nombre de la vista creada es **v\_computadoras\_compradas.** En esta vista obtenemos las compras de las computadoras, el precio de la compra, como también los datos de la sucursal donde se realizó la compra. Esta vista informa los siguientes datos sobre una compra de computadora:

* Numero de compra
* Fecha
* Código de la Computadora
* Cantidad
* Precio de Compra
* Dirección de la Sucursal
* Ciudad de la Sucursal
* Mail de la Sucursal
* Teléfono de la Sucursal

Para obtener dicha información recurrimos a la operación **join** a modo de relacionar las siguientes tablas:

* compra\_pc
* tipo\_pc
* sucursal
* ciudad

CREATE VIEW LAWE.v\_computadoras\_compradas AS

          SELECT

                c.numero\_compra AS Numero\_de\_compra,

                c.fecha AS Fecha,

                cpc.pc\_codigo AS Codigo\_de\_PC,

                cpc.cantidad AS Cantidad,

                tpc.precio\_compra AS Precio\_Unidad\_Comprada,

                s.direccion AS Direccion\_Sucursal,

                ciu.nombre AS Ciudad\_Sucursal,

                s.mail AS Mail\_Sucursal,

                s.telefono AS Telefono\_Sucursal

            FROM LAWE.compra c

                  JOIN LAWE.compra\_pc cpc ON cpc.numero\_compra = c.numero\_compra

                  JOIN LAWE.tipo\_pc tpc ON tpc.pc\_codigo = cpc.pc\_codigo

                  JOIN LAWE.sucursal s ON c.id\_sucursal = s.id\_sucursal

                  JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

## **3.2 Vista - Compra de Accesorios**

El nombre de la vista creada es **v\_accesorios\_comprados.** En esta vista obtenemos cada uno de los accesorios, su cantidad, el precio al que fue comprado. También se agregaron los datos de la sucursal que realizó la compra. Esta vista informa los siguientes datos sobre un item de una compra de accesorios:

* Número de Compra
* Fecha
* Código del Accesorio
* Cantidad
* Precio de unidad comprada
* Descripción
* Dirección de la Sucursal
* Ciudad de la Sucursal
* Mail de la Sucursal
* Teléfono de la Sucursal

Para obtener dicha información relacionamos las siguientes tablas:

* item\_compra\_accesorio
* accesorio
* sucursal
* ciudad

CREATE VIEW LAWE.v\_accesorios\_comprados AS

        SELECT

            c.numero\_compra AS Numero\_de\_compra,

            c.fecha AS Fecha,

            ica.accesorio\_codigo AS Codigo\_del\_Accesorio,

            ica.cantidad AS Cantidad,

            a.precio\_compra AS Precio\_Unidad\_Comprada,

            a.descripcion AS Descripcion,

            -- No se obtienen los datos de los fabricantes de los accesorios porque son inexistentes

            s.direccion AS Direccion\_Sucursal,

            ciu.nombre AS Ciudad\_Sucursal,

            s.mail AS Mail\_Sucursal,

            s.telefono AS Telefono\_Sucursal

        FROM LAWE.compra c

            JOIN LAWE.item\_compra\_accesorio ica ON c.numero\_compra = ica.numero\_compra

            JOIN LAWE.accesorio a ON ica.accesorio\_codigo = a.accesorio\_codigo

            JOIN LAWE.sucursal s ON c.id\_sucursal = s.id\_sucursal

            JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

## **3.3 Vista - Compras Accesorios**

El nombre de la vista creada es **v\_compras\_de\_accesorios**. En esta vista observamos las compras de accesorios realizadas con su importe total, los datos de la sucursal que la realizó al igual que la vista anterior que representaba los items. Esta vista informa los siguientes datos sobre una compra de accesorios:

* Número de Compra
* Fecha
* Importe Total
* Dirección de la Sucursal
* Ciudad de la Sucursal
* Mail de la Sucursal
* Teléfono de la Sucursal

Obtuvimos los datos a partir de las siguientes tablas, en donde las relacionamos a partir de sus claves foráneas

* item\_compra\_accesorio
* accesorio
* sucursal
* ciudad

CREATE VIEW LAWE.v\_compras\_de\_accesorios AS

        SELECT

            c.numero\_compra AS Numero\_de\_compra,

            c.fecha AS Fecha,

            SUM(ica.cantidad \* a.precio\_compra) AS Importe\_Total, -- calculamos el importe total, sumando el total por cada item

            s.direccion AS Direccion\_Sucursal,

            ciu.nombre AS Ciudad\_Sucursal,

            s.mail AS Mail\_Sucursal,

            s.telefono AS Telefono\_Sucursal

        FROM LAWE.compra c

            JOIN LAWE.item\_compra\_accesorio ica ON c.numero\_compra = ica.numero\_compra

            JOIN LAWE.accesorio a ON ica.accesorio\_codigo = a.accesorio\_codigo

            JOIN LAWE.sucursal s ON c.id\_sucursal = s.id\_sucursal

            JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

        GROUP BY c.numero\_compra,c.fecha,s.direccion,ciu.nombre,s.mail,s.telefono

## **3.4 Vista - Facturas PC**

El nombre de la vista representada es **v\_ventas\_de\_computadoras**. Nos enfocamos en obtener las ventas realizadas de computadoras, de las cuales figura la información detallada sobre el cliente involucrado en la venta, la sucursal que realizó la venta y datos de la computadora vendida. Además, como otros datos relevantes aparecen el precio al que se factura y el precio al que fue comprada por la sucursal. Esta vista informa los siguientes datos sobre una venta de computadora:

* Número de la Factura
* Fecha
* DNI, Apellido, y Nombre del Cliente (cada una de manera individual)
* Codigo de PC
* Precio de Compra
* Precio facturado
* Dirección, Ciudad, Mail y Teléfono de la Sucursal (cada una de manera individual)

El precio de facturación de la PC se calcula en un 20% aproximadamente del precio en el cual se realizó la compra dicha computadora.

Las tablas que usamos de referencia para obtener los datos fueron

* cliente
* sucursal
* ciudad
* factura\_pc
* tipo\_pc

CREATE VIEW LAWE.v\_ventas\_de\_computadoras AS

        SELECT

            f.numero\_factura AS Numero\_de\_Factura,

            f.fecha AS Fecha,

            c.apellido AS Apellido\_Cliente,

            c.nombre AS Nombre\_Cliente,

            c.dni AS DNI\_Cliente,

            fpc.pc\_codigo AS Codigo\_PC,

            tpc.precio\_compra AS Precio\_de\_Compra,

            tpc.precio\_compra \* 1.2 AS Precio\_Facturado, -- El precio de la facturación de la PC se calcula en un 20% aproximadamente del precio en el cual se realizó la compra del mismo.

            s.direccion AS Direccion\_Sucursal,

            ciu.nombre AS Ciudad\_Sucursal,

            s.mail AS Mail\_Sucursal,

            s.telefono AS Telefono\_Sucursal

        FROM LAWE.factura f

            JOIN LAWE.cliente c ON c.id\_cliente = f.id\_cliente

            JOIN LAWE.sucursal s ON f.id\_sucursal = s.id\_sucursal

            JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

            JOIN LAWE.factura\_pc fpc ON f.numero\_factura = fpc.numero\_factura

            JOIN LAWE.tipo\_pc tpc ON fpc.pc\_codigo = tpc.pc\_codigo

## **3.5 Vista - Facturas Accesorios**

El nombre de la vista es **v\_accesorios\_vendidos**. Podemos observar las ventas realizadas de accesorios, en donde se detallan los datos particulares del cliente involucrado en la venta, la sucursal que realizó la venta y datos de los accesorios vendidos. Esta vista informa los siguientes datos sobre los ítems de las ventas de accesorios:

* Número de Factura
* Fecha
* DNI, Apellido, Nombre del Cliente (cada uno en una columna independiente)
* Código del Accesorio
* Descripción del accesorio
* Cantidad
* Precio por Unidad
* Total por accesorio
* Dirección, Ciudad, Mail y Teléfono de la Sucursal (cada una de manera individual)

Entre dichas columnas no aparecen los datos del fabricante, porque los datos de este no existen. Reutilizamos la información de las siguientes tablas:

* cliente
* sucursal
* ciudad
* item\_factura\_accesorio
* accesorio

CREATE VIEW LAWE.v\_accesorios\_vendidos AS

        SELECT

            f.numero\_factura AS Numero\_de\_Factura,

            f.fecha AS Fecha,

            c.apellido AS Apellido\_Cliente,

            c.nombre AS Nombre\_Cliente,

            c.dni AS DNI\_Cliente,

            acc.accesorio\_codigo AS Codigo\_Accesorio,

            acc.descripcion AS Descripcion,

            -- No se obtienen los datos de los fabricantes de los accesorios porque son inexistentes

            ifa.cantidad AS Cantidad,

            acc.precio\_compra AS Precio\_por\_Unidad,

            acc.precio\_compra \* ifa.cantidad AS Total\_por\_Accesorio,

            s.direccion AS Direccion\_Sucursal,

            ciu.nombre AS Ciudad\_Sucursal,

            s.mail AS Mail\_Sucursal,

            s.telefono AS Telefono\_Sucursal

        FROM LAWE.factura f

            JOIN LAWE.cliente c ON c.id\_cliente = f.id\_cliente

            JOIN LAWE.sucursal s ON f.id\_sucursal = s.id\_sucursal

            JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

            JOIN LAWE.item\_factura\_accesorio ifa ON f.numero\_factura = ifa.numero\_factura

            JOIN LAWE.accesorio acc ON ifa.accesorio\_codigo = acc.accesorio\_codigo

## **3.6 Vista - Ventas de Accesorios**

El nombre que elegimos como representativo es **v\_ventas\_de\_accesorios**. La información que figura en esta vista es sobre las ventas de accesorios realizadas con su respectivo importe total facturado. También los datos de la sucursal que realizó la venta, y por último los datos del cliente involucrado en dicha venta. Esta vista informa los siguientes datos sobre las ventas de accesorios:

* Número de Factura
* Fecha
* Importe total
* Cliente
  + DNI
  + Apellido
  + Nombre
* Sucursal
  + Dirección
  + Ciudad
  + Mail
  + Teléfono

El importe total se calculó sumando el total por cada item. Agrupamos los datos con las columnas anteriormente mencionadas, y reutilizamos la información de las siguientes tablas:

* cliente
* sucursal
* ciudad
* item\_factura\_accesorio
* accesorio

CREATE VIEW LAWE.v\_ventas\_de\_accesorios AS

        SELECT

            f.numero\_factura AS Numero\_de\_Factura,

            f.fecha AS Fecha,

            SUM(ifa.cantidad \* acc.precio\_compra) AS Importe\_Total, -- calculamos el importe total, sumando el total por cada item

            c.apellido AS Apellido\_Cliente,

            c.nombre AS Nombre\_Cliente,

            c.dni AS DNI\_Cliente,

            s.direccion AS Direccion\_Sucursal,

            ciu.nombre AS Ciudad\_Sucursal,

            s.mail AS Mail\_Sucursal,

            s.telefono AS Telefono\_Sucursal

        FROM LAWE.factura f

            JOIN LAWE.cliente c ON c.id\_cliente = f.id\_cliente

            JOIN LAWE.sucursal s ON f.id\_sucursal = s.id\_sucursal

            JOIN LAWE.ciudad ciu ON s.id\_ciudad = ciu.id\_ciudad

            JOIN LAWE.item\_factura\_accesorio ifa ON f.numero\_factura = ifa.numero\_factura

            JOIN LAWE.accesorio acc ON ifa.accesorio\_codigo = acc.accesorio\_codigo

GROUP BY  f.numero\_factura,f.fecha,c.apellido,c.nombre,

c.dni,s.direccion,ciu.nombre,s.mail,s.telefono

## **Vista - Clientes**

En esta vista, nos encontramos con todos los clientes existentes, con su información pertinente. Usamos **v\_cliente** como nombre representativo. Esta vista informa los siguientes datos sobre los clientes:

* Apellido
* Nombre
* DNI
* Dirección
* Correo Electrónico
* Teléfono
* Fecha de Nacimiento

No vimos necesario tener que obtener información de varias tablas, si no que únicamente utilizamos la tabla **cliente.**

CREATE VIEW LAWE.v\_clientes AS

        SELECT

            apellido AS Apellido,

            nombre AS Nombre,

            dni AS DNI,

            direccion AS Direccion,

            mail AS Correo\_Electronico,

            telefono AS Telefono,

            fecha\_nacimiento AS Fecha\_de\_Nacimiento

        FROM LAWE.cliente

## **3.8 Vista - Tipos de Computadoras existentes**

Utilizamos **v\_modelos\_de\_pc** como nombre de vista. En esta aparecen los distintos modelos de computadoras junto con la información de sus componentes. Esta vista informa los siguientes datos sobre los modelos de computadoras:

* Codigo de PC
* Precio Compra
* Gabinete
  + Alto
  + Ancho
  + Profundidad
* Disco Rigido
  + Tipo
  + Capacidad
  + Nombre del Fabricante
* Memoria RAM
  + Tipo
  + Capacidad
  + Nombre del Fabricante
* Microprocesador
  + Velocidad
  + Nombre del Fabricante
* Placa de Video
  + Modelo
  + Nombre del Fabricante
  + Chipset

Marcamos como relevante que no incluimos información sobre la **motherboard** que no había datos de ellas.

Para obtener dicha información de todos los componentes, utilizamos las siguientes tablas

* gabinete
* disco\_rigido
* memoria\_ram
* microprocesador
* placa\_video

    CREATE VIEW LAWE.v\_modelos\_de\_pc AS

        SELECT

            tpc.pc\_codigo AS Codigo\_de\_PC,

            tpc.precio\_compra AS Precio\_Compra\_PC,

            -- datos del gabinete

            g.alto AS Alto\_Gabinete,

            g.ancho AS Ancho\_Gabinete,

            g.profundidad AS Profundidad\_Gabinete,

            -- datos del disco rigido

            dr.tipo AS Tipo\_Disco,

            dr.capacidad AS Capacidad\_Disco,

            LAWE.OBTENER\_NOMBRE\_FABRICANTE(dr.id\_fabricante) AS Fabricante\_Disco,

            -- datos de la memoria ram

            mem.tipo AS Tipo\_RAM,

            mem.capacidad AS Capacidad\_RAM,

            LAWE.OBTENER\_NOMBRE\_FABRICANTE(mem.id\_fabricante) AS Fabricante\_RAM,

            -- datos del microprocesador

            mic.velocidad AS Velocidad\_Micro,

            LAWE.OBTENER\_NOMBRE\_FABRICANTE(mic.id\_fabricante) AS Fabricante\_Micro,

            -- datos de la placa de video

            pv.placa\_video\_modelo AS Modelo\_Placa\_de\_Video,

            LAWE.OBTENER\_NOMBRE\_FABRICANTE(pv.id\_fabricante) AS Fabricante\_Placa\_de\_Video,

            pv.chipset AS Chipset\_Placa\_de\_Video

            -- no incluimos datos del motherboard porque son inexistentes

        FROM LAWE.tipo\_pc tpc

  JOIN LAWE.memoria\_ram mem ON tpc.memoria\_ram\_codigo = mem.memoria\_ram\_codigo

  JOIN LAWE.microprocesador mic ON tpc.microprocesador\_codigo = mic.microprocesador\_codigo

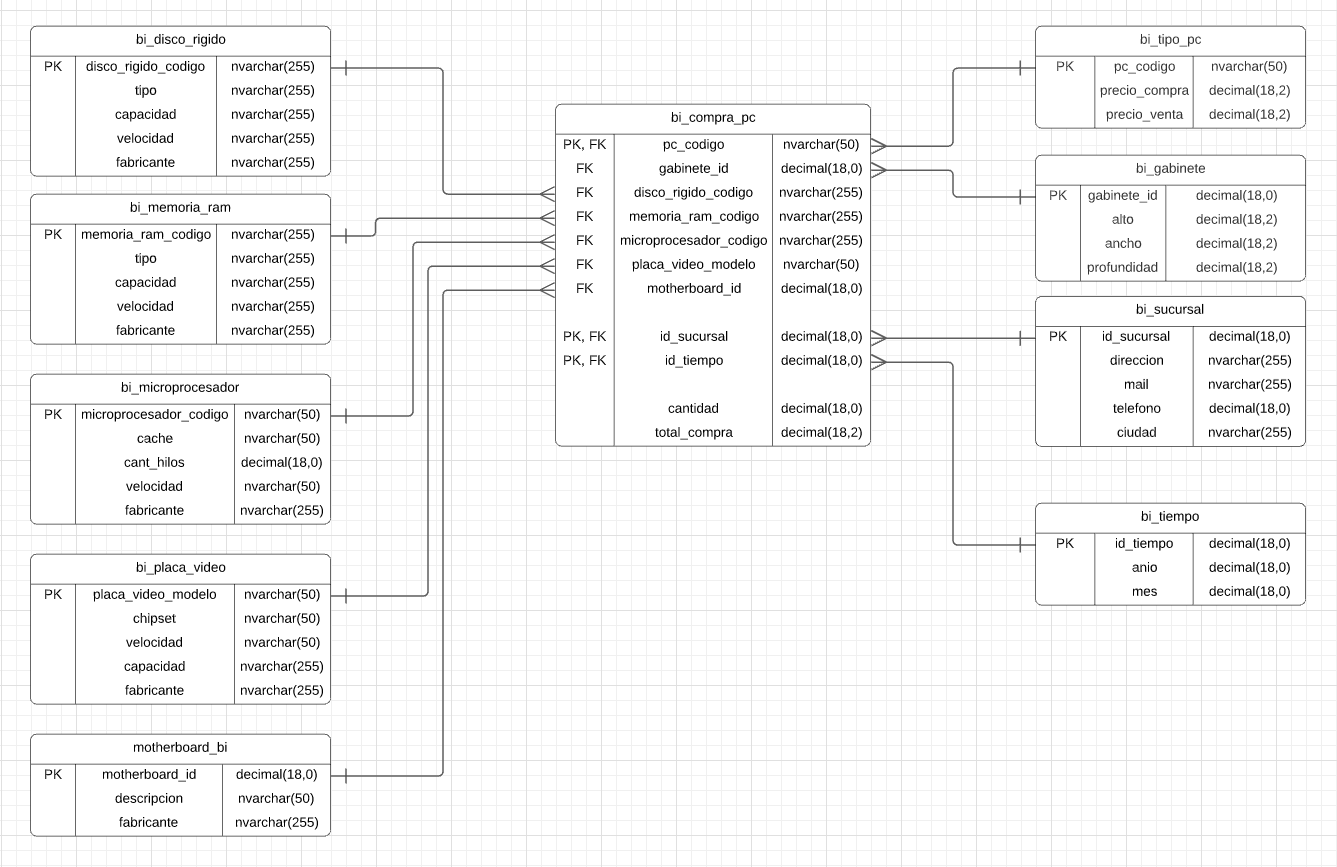
  JOIN LAWE.placa\_video pv ON tpc.placa\_video\_modelo = pv.placa\_video\_modelo

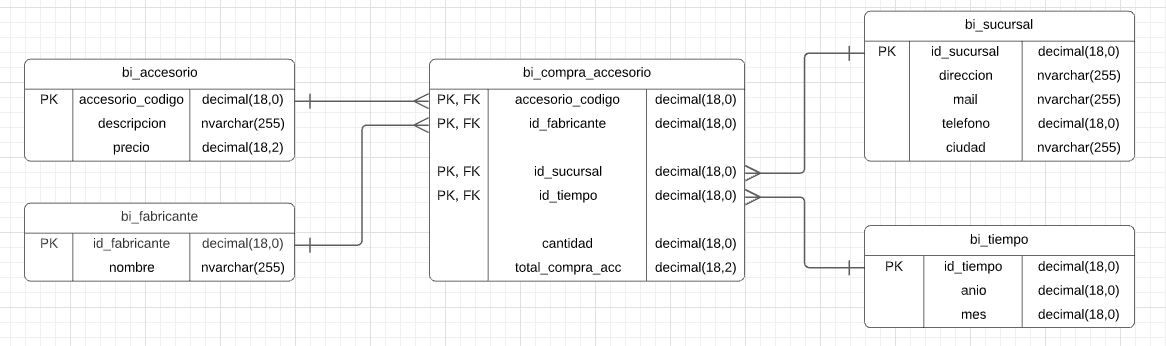
JOIN LAWE.gabinete g ON tpc.gabinete\_id = g.gabinete\_id

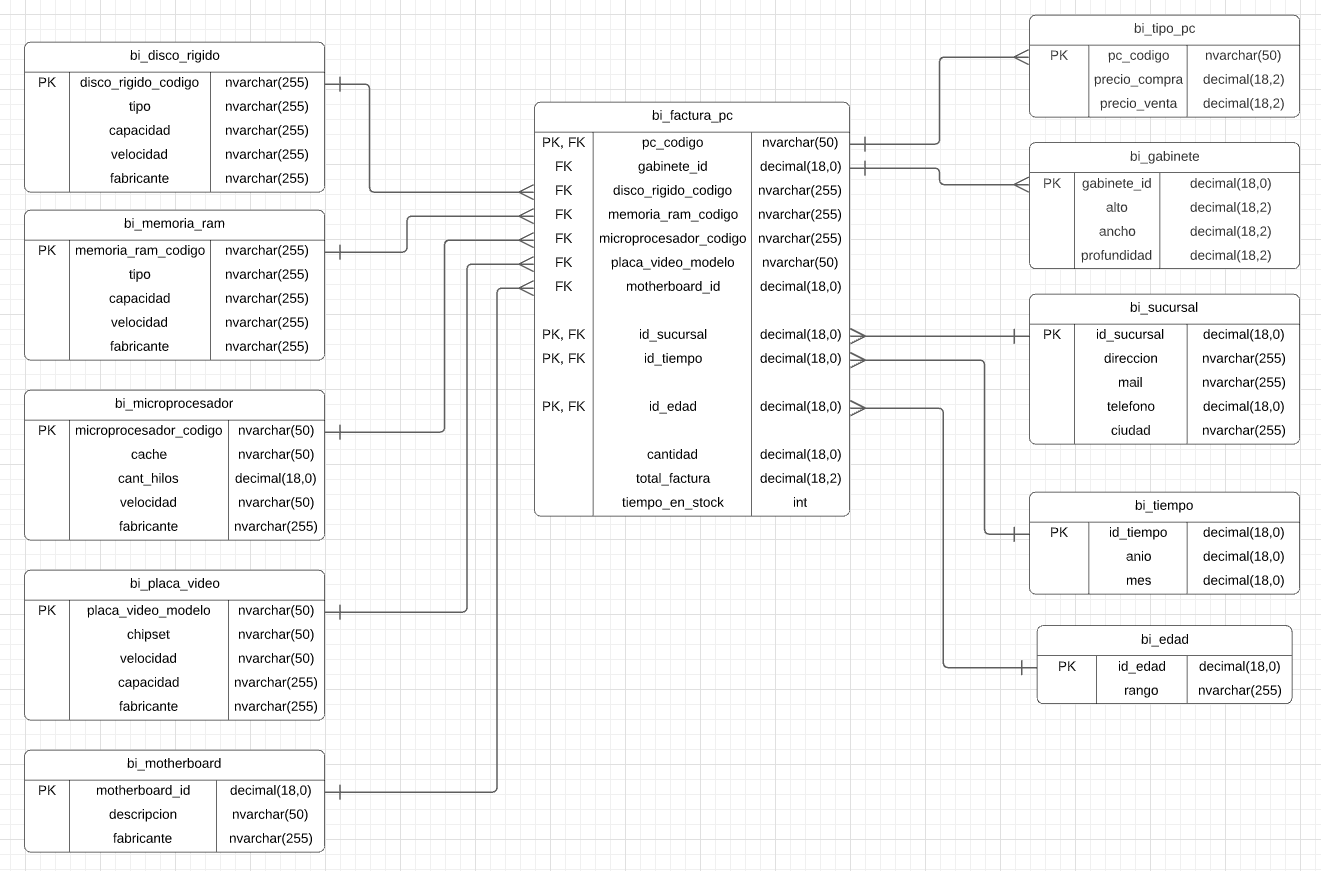
  JOIN LAWE.disco\_rigido dr ON tpc.disco\_rigido\_codigo = dr.disco\_rigido\_codigo

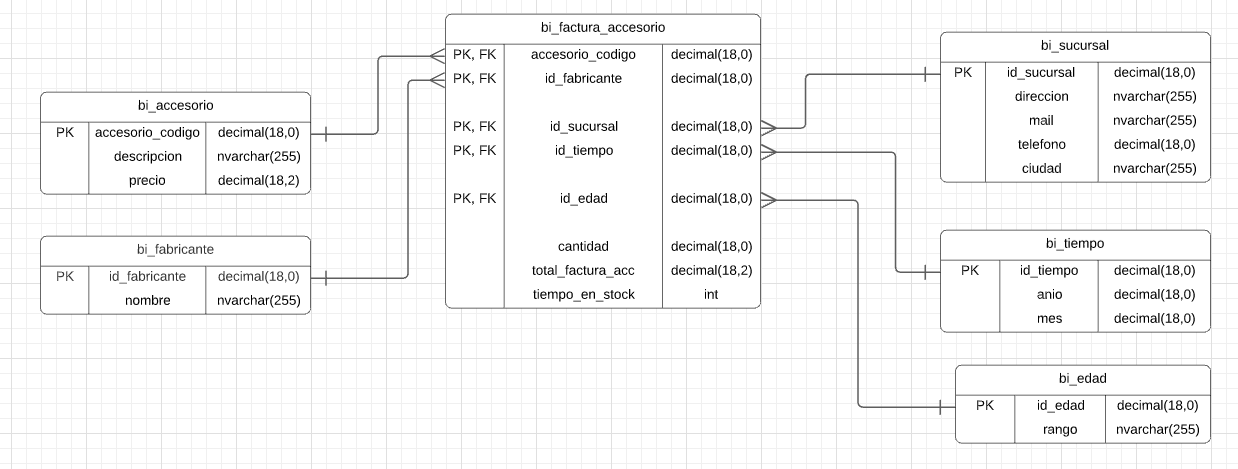
# Modelo de Inteligencia de Negocios (BI)

En este apartado se detalla el procedimiento utilizado para el desarrollo del modelo de Business Intelligence. A continuación, se presenta el DER (separado por partes) respectivo al Modelo BI:









## **4.1 Borrado Previo**

Igual que en la sección anterior, antes de realizar la migración se realiza un borrado de objetos de base de datos que hayan sido creados por este script. La modalidad es la misma y tiene como objetivo evitar conflictos al ejecutar el script en reiteradas ocasiones. El script no funcionará si anteriormente no se ejecuta script\_creación\_inicial.sql.

## **4.2 Modelo Estrella**

Tal y como se vio en clase, se decide utilizar el modelo estrella para confeccionar el modelo de *Business Intelligence.*

## **4.3 Tablas de hechos (Fact tables)**

Para confeccionar el modelo estrella se han definido las siguientes tablas de hechos:

* bi\_compra\_pc: En esta tabla se detalla las compras de pc
* bi\_compra\_accesorio: En esta tabla se detallan las compras de accesorio
* bi\_factura\_pc: En esta tabla se detallan las facturas de pc
* bi\_factura\_accesorio: En esta tabla se detalla las facturas de accesorio

Para confeccionar las tablas descriptas anteriormente se han tenido en cuenta las siguientes dimensiones

**bi\_compra\_pc**

* codigo pc
* gabinete
* disco rígido
* memoria ram
* microprocesador
* placa video
* motherboard
* sucursal
* tiempo
* cantidad
* total compra

**bi\_factura\_pc**

* codigo pc
* gabinete
* disco rígido
* memoria ram
* microprocesador
* placa video
* motherboard
* sucursal
* tiempo
* edad
* cantidad
* total factura
* tiempo en stock

**bi\_compra\_accesorio**

* codigo accesorio
* fabricante
* sucursal
* tiempo
* cantidad
* total de la compra

**bi\_factura\_accesorio**

* codigo accesorio
* fabricante
* sucursal
* tiempo
* edad
* cantidad
* total factura
* tiempo en stock

Se decide separar en distintas tablas de hechos (distintos Data Marts) debido a que, si fuese una sola tabla de hechos, habría que lidiar con claves nulas, así como también atributos nulos, lo cual complica las operaciones sobre las tablas que son necesarias para confeccionar el modelo.

De la misma manera, si bien tiene sentido agregar datos como la edad de los clientes y otras dimensiones en la parte del modelo que se encarga de las facturas, no se ve necesario para, por ejemplo, el cálculo del stock. Podría ser necesario en caso de implementar técnicas de Data Mining, pero se escapa completamente del objetivo del TP y complejiza altamente la confección del modelo.

## **4.4 Migración hacia el modelo de Business Intelligence**

Para poder confeccionar el modelo se decide migrar las tablas necesarias que pasaran a ser las dimensiones del modelo.

Tablas migradas:

* bi\_disco\_rigido
* bi\_memoria\_ram
* bi\_microprocesador
* bi\_placa\_video
* bi\_motherboard
* bi\_tipo\_pc
* bi\_gabinete
* bi\_sucursal
* bi\_fabricante
* bi\_accesorio

Diagrama

Descripción generada automáticamente

## **4.5 Tablas confeccionadas para el modelo de Business Intelligence**

Para facilitar la confección del modelo, se han creado las siguientes tablas:

Diagrama

Descripción generada automáticamente

## **4.6 Proceso de migración hacia el modelo Business Intelligence**

Para migrar los datos del modelo transaccional hacia el modelo de Business Intelligence, se han utilizado, de la misma manera que la migración de la entrega anterior, Stored Procedures que serán detallados más adelante.

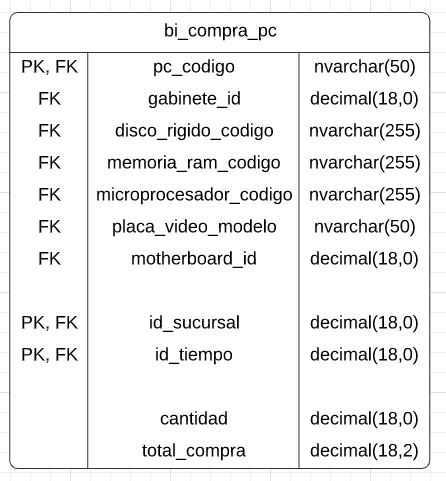
A continuación, detallaremos aquellas decisiones tomadas acerca de las migraciones

a) **Cliente**: No hay forma de calcular u obtener el sexo de cada cliente, con lo cual, se decide dejar el atributo en null.

b) **Tiempo de stock promedio de pc y accesorios:** Usamos un enfoque FIFO para el cálculo del tiempo en stock tanto de las pc como de los accesorios, intentamos hacer una vinculación entre la primera compra que se realizó de cierto producto con la primera factura que se realizó sobre el mismo producto y en base a eso obtenemos el tiempo que estuvo en stock

## **4.7 Tablas de hechos**

Se han creado las siguientes tablas de hechos tal y como se mencionó anteriormente:



Tabla

Descripción generada automáticamente con confianza baja

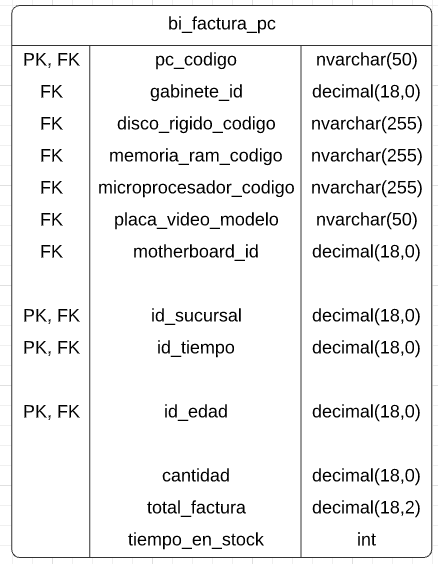


Imagen que contiene Tabla

Descripción generada automáticamente

# Migración hacia el Modelo BI

# **5.1 Funciones**

## **Implementación de Funciones Adicionales**

Desarrollamos las siguientes funciones, que nos facilitan acceder a la información desde distintas

consultas. Destacamos cuáles fueron las más importantes con una breve explicación sobre la problemática que soluciona.

* **OBTENER\_NOMBRE\_CIUDAD**

Esta función nos permite obtener el nombre de ciudad a partir de su número de id, y permite la migración de datos de **sucursales** al modelo BI

CREATE FUNCTION LAWE.OBTENER\_NOMBRE\_CIUDAD(@id\_ciudad DECIMAL(18,0))

RETURNS NVARCHAR(255) AS

    BEGIN

        DECLARE @ciudad NVARCHAR(255);

        SELECT @ciudad = nombre FROM LAWE.ciudad WHERE id\_ciudad = @id\_ciudad

        RETURN @ciudad;

END

* **OBTENER\_NOMBRE\_FABRICANTE**

Dicha función nos permite obtener el nombre de fabricante a partir de su número de id, facilitando la migración de los distintos componentes a las dimensiones del modelo BI

CREATE FUNCTION LAWE.OBTENER\_NOMBRE\_FABRICANTE(@id\_fabricante DECIMAL(18,0))

RETURNS NVARCHAR(255) AS

    BEGIN

        DECLARE @fabricante NVARCHAR(255);

        SELECT @fabricante = nombre FROM LAWE.fabricante

WHERE id\_fabricante = @id\_fabricante

        RETURN @fabricante;

    END

* **OBTENER\_ID\_TIEMPO**

Esta función nos brinda el id de un “tiempo” (mes y año) específico de una determinada fecha

CREATE FUNCTION LAWE.OBTENER\_ID\_TIEMPO(@fecha DATE) RETURNS DECIMAL(18) AS

BEGIN

    DECLARE @anio DECIMAL(18),

            @mes DECIMAL(18),

            @id\_tiempo DECIMAL(18)

    SET @anio = DATEPART(YEAR, @fecha)

    SET @mes = DATEPART(MONTH, @fecha)

    SELECT @id\_tiempo = id\_tiempo FROM LAWE.bi\_tiempo

WHERE anio = @anio AND mes = @mes

    RETURN @id\_tiempo

END

* **OBTENER\_ID\_EDAD**

Esta función nos permite obtener el id del rango de la edad de un cliente

CREATE FUNCTION LAWE.OBTENER\_ID\_EDAD(@FECHA\_NACIMIENTO DATE) RETURNS DECIMAL(18) AS

BEGIN

    DECLARE @id\_edad DECIMAL(18);

    DECLARE @HOY DATE;

    DECLARE @EDAD INT;

    SET @HOY = GETDATE();

    SET @EDAD = (DATEDIFF(DAY, @FECHA\_NACIMIENTO, @HOY) / 365)

    IF @EDAD BETWEEN 18 AND 30

        SELECT @id\_edad = id\_edad FROM LAWE.bi\_edad WHERE rango = '18 - 30 años'

    ELSE IF @EDAD BETWEEN 31 AND 50

        SELECT @id\_edad = id\_edad FROM LAWE.bi\_edad WHERE rango = '31 - 50 años'

    ELSE

        SELECT @id\_edad = id\_edad FROM LAWE.bi\_edad WHERE rango = '> 50 años'

    RETURN @id\_edad;

END

## **5.2 Stored Procedures**

Para poder confeccionar el modelo se decide migrar las tablas necesarias que pasarán a ser las dimensiones del modelo. Realizamos la migración a través de Stored Procedures los cuales cargan los datos en las tablas dimensión y tablas de hechos a partir de las tablas existentes del Modelo Relacional.

Tablas migradas:

* bi\_tipo\_pc

CREATE PROCEDURE LAWE.migrar\_bi\_tipo\_pc AS

BEGIN

  INSERT INTO LAWE.bi\_tipo\_pc (pc\_codigo, precio\_compra, precio\_venta)

    SELECT pc\_codigo, precio\_compra, 1.2 \* precio\_compra  FROM LAWE.tipo\_pc

END

* bi\_gabinete

CREATE PROCEDURE LAWE.migrar\_bi\_gabinete AS

BEGIN

  INSERT INTO LAWE.bi\_gabinete (gabinete\_id, alto,ancho,profundidad)

    SELECT gabinete\_id, alto,ancho,profundidad FROM LAWE.gabinete

END

* bi\_sucursal

CREATE PROCEDURE LAWE.migrar\_bi\_sucursal AS

BEGIN

  INSERT INTO LAWE.bi\_sucursal (id\_sucursal, direccion, mail, telefono, ciudad)

    SELECT id\_sucursal, direccion, mail, telefono, LAWE.OBTENER\_NOMBRE\_CIUDAD(id\_ciudad)

    FROM LAWE.sucursal

END

* bi\_disco\_rigido

CREATE PROCEDURE LAWE.migrar\_bi\_disco\_rigido AS

BEGIN

  INSERT INTO

LAWE.bi\_disco\_rigido (disco\_rigido\_codigo, tipo, capacidad, velocidad, fabricante)

    SELECT disco\_rigido\_codigo, tipo, capacidad, velocidad,

LAWE.OBTENER\_NOMBRE\_FABRICANTE(id\_fabricante)

    FROM LAWE.disco\_rigido

END

* bi\_memoria\_ram

CREATE PROCEDURE LAWE.migrar\_bi\_memoria\_ram AS

BEGIN

  INSERT INTO

LAWE.bi\_memoria\_ram (memoria\_ram\_codigo, tipo, capacidad, velocidad, fabricante)

    SELECT memoria\_ram\_codigo, tipo, capacidad, velocidad,

LAWE.OBTENER\_NOMBRE\_FABRICANTE(id\_fabricante)

    FROM LAWE.memoria\_ram

END

* bi\_microprocesador

CREATE PROCEDURE LAWE.migrar\_bi\_microprocesador AS

BEGIN

  INSERT INTO

LAWE.bi\_microprocesador (microprocesador\_codigo, cache, cant\_hilos, velocidad, fabricante)

    SELECT microprocesador\_codigo, cache, cant\_hilos, velocidad,

LAWE.OBTENER\_NOMBRE\_FABRICANTE(id\_fabricante)

    FROM LAWE.microprocesador

END

* bi\_placa\_video

CREATE PROCEDURE LAWE.migrar\_bi\_placa\_video AS

BEGIN

  INSERT INTO

LAWE.bi\_placa\_video (placa\_video\_modelo, chipset, velocidad, capacidad, fabricante)

    SELECT placa\_video\_modelo, chipset, velocidad, capacidad,

 LAWE.OBTENER\_NOMBRE\_FABRICANTE(id\_fabricante)

    FROM LAWE.placa\_video

END

* bi\_motherboard

CREATE PROCEDURE LAWE.migrar\_bi\_motherboard AS

BEGIN

  INSERT INTO LAWE.bi\_motherboard (motherboard\_id, descripcion, fabricante)

    SELECT motherboard\_id, descripcion, LAWE.OBTENER\_NOMBRE\_FABRICANTE(id\_fabricante)

    FROM LAWE.motherboard

END

* bi\_accesorio

CREATE PROCEDURE LAWE.migrar\_bi\_accesorio AS

BEGIN

  INSERT INTO LAWE.bi\_accesorio (accesorio\_codigo, descripcion, precio)

    SELECT accesorio\_codigo, descripcion, precio\_compra

    FROM LAWE.accesorio

END

* bi\_fabricante

CREATE PROCEDURE LAWE.migrar\_bi\_fabricante AS

BEGIN

  INSERT INTO LAWE.bi\_fabricante (id\_fabricante, nombre)

    SELECT id\_fabricante, nombre

    FROM LAWE.fabricante

END

Tablas confeccionadas para el modelo de Business Intelligence. Para facilitar la confección del modelo, se han creado las siguientes tablas:

* bi\_tiempo
  + id\_tiempo
  + mes
  + año

CREATE PROCEDURE LAWE.cargar\_bi\_tiempo AS

BEGIN

    INSERT INTO LAWE.bi\_tiempo (anio, mes)

        SELECT DISTINCT YEAR(fecha), MONTH(fecha)

        FROM LAWE.compra

        UNION

        SELECT DISTINCT YEAR(fecha), MONTH(fecha)

        FROM LAWE.factura

END

* bi\_edad
  + id\_edad
  + rango

CREATE PROCEDURE LAWE.cargar\_bi\_edad AS

BEGIN

    INSERT INTO LAWE.bi\_edad (rango)

        VALUES  ('18 - 30 años'),

                ('31 - 50 años'),

                ('> 50 años')

END

Tablas de Hechos:

* bi\_compra\_pc

BEGIN

    INSERT INTO

LAWE.bi\_compra\_pc (pc\_codigo, gabinete\_id, disco\_rigido\_codigo, memoria\_ram\_codigo,

microprocesador\_codigo, placa\_video\_modelo, motherboard\_id, id\_sucursal, id\_tiempo,

 cantidad, total\_compra)

    SELECT

        t\_pc.pc\_codigo,

        gabinete\_id,

        disco\_rigido\_codigo,

        memoria\_ram\_codigo,

        microprocesador\_codigo,

        placa\_video\_modelo,

        motherboard\_id,

        id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        COUNT(cantidad),

        SUM(cantidad \* t\_pc.precio\_compra)

    FROM LAWE.compra\_pc c\_pc

        JOIN LAWE.compra c ON c.numero\_compra = c\_pc.numero\_compra

        JOIN LAWE.tipo\_pc t\_pc ON c\_pc.pc\_codigo = t\_pc.pc\_codigo

    GROUP BY

        t\_pc.pc\_codigo,

        gabinete\_id,

        disco\_rigido\_codigo,

        memoria\_ram\_codigo,

        microprocesador\_codigo,

        placa\_video\_modelo,

        motherboard\_id,

        id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha)

END

* bi\_factura\_pc

CREATE PROCEDURE LAWE.migrar\_bi\_factura\_pc AS

BEGIN

    INSERT INTO LAWE.bi\_factura\_pc (pc\_codigo, gabinete\_id, disco\_rigido\_codigo, memoria\_ram\_codigo,microprocesador\_codigo, placa\_video\_modelo, motherboard\_id, id\_sucursal, id\_tiempo, id\_edad, cantidad, total\_factura, tiempo\_en\_stock)

    SELECT

        t\_pc.pc\_codigo,

        gabinete\_id,

        disco\_rigido\_codigo,

        memoria\_ram\_codigo,

        microprocesador\_codigo,

        placa\_video\_modelo,

        motherboard\_id,

        id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        LAWE.OBTENER\_ID\_EDAD(c.fecha\_nacimiento),

        COUNT(\*),

        SUM(t\_pc.precio\_compra \* 1.2),

        SUM(t.tiempo\_en\_stock) / COUNT(\*) AS tiempo\_stock\_promedio

    FROM LAWE.factura\_pc f\_pc

        JOIN LAWE.factura f ON f.numero\_factura = f\_pc.numero\_factura

        JOIN LAWE.tipo\_pc t\_pc ON f\_pc.pc\_codigo = t\_pc.pc\_codigo

        JOIN LAWE.cliente c ON f.id\_cliente = c.id\_cliente

        JOIN #tiempo\_en\_stock\_pc t ON f.numero\_factura = t.numero\_factura

    GROUP BY

        t\_pc.pc\_codigo,

        gabinete\_id,

        disco\_rigido\_codigo,

        memoria\_ram\_codigo,

        microprocesador\_codigo,

        placa\_video\_modelo,

        motherboard\_id,

        id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        LAWE.OBTENER\_ID\_EDAD(c.fecha\_nacimiento)

    DROP TABLE #tiempo\_en\_stock\_pc

END

Para calcular el tiempo de stock de las computadoras vendidas utilizamos una tabla temporal llamada #tiempo\_en\_stock\_pc, en la cual asociamos una factura con su respectiva compra para conocer la cantidad de días que permaneció en stock la computadora vendida.

CREATE TABLE #tiempo\_en\_stock\_pc (

        numero\_compra DECIMAL(18),

        fecha\_compra DATETIME2(3),

        pc\_codigo\_compra NVARCHAR(50),

        numero\_factura DECIMAL(18),

        fecha\_factura DATETIME2(3),

        pc\_codigo\_factura NVARCHAR(50),

        tiempo\_en\_stock DECIMAL(18))

Este Stored Procedure calcula el tiempo que estuvo en stock una PC comparando la fecha de su compra con la fecha de su correspondiente factura, basándonos en el concepto FIFO (La primer compra de PC corresponde a la primer factura de PC, y así sucesivamente). Para realizar la asociación correctamente decidimos utilizar dos cursores de manera simultánea en el Stored Procedure los cuales recorren uno a uno los registros de las tablas de compras y facturas de PC.

CREATE PROCEDURE LAWE.calcular\_tiempo\_stock\_pc AS

BEGIN

    DECLARE cursor\_compras\_pc CURSOR

    FOR

        SELECT c.numero\_compra, fecha, c\_pc.pc\_codigo

        FROM LAWE.compra\_pc c\_pc

            JOIN LAWE.compra c ON c.numero\_compra = c\_pc.numero\_compra

        DECLARE @numero\_compra DECIMAL(18), @fecha\_compra DATETIME2(3),

 @pc\_codigo\_compra NVARCHAR(50)

    DECLARE cursor\_facturas\_pc CURSOR

    FOR

        SELECT f.numero\_factura, fecha, f\_pc.pc\_codigo

        FROM LAWE.factura\_pc f\_pc

            JOIN LAWE.factura f ON f.numero\_factura = f\_pc.numero\_factura

        DECLARE @numero\_factura DECIMAL(18), @fecha\_factura DATETIME2(3),

@pc\_codigo\_factura NVARCHAR(50)

        OPEN cursor\_compras\_pc

        OPEN cursor\_facturas\_pc

        FETCH cursor\_compras\_pc INTO @numero\_compra, @fecha\_compra, @pc\_codigo\_compra

        FETCH cursor\_facturas\_pc INTO @numero\_factura, @fecha\_factura, @pc\_codigo\_factura

        WHILE (@@FETCH\_STATUS=0)

        BEGIN

            INSERT INTO #tiempo\_en\_stock\_pc

            VALUES(

                @numero\_compra,

                @fecha\_compra,

                @pc\_codigo\_compra,

                @numero\_factura,

                @fecha\_factura,

                @pc\_codigo\_factura,

                DATEDIFF(DAY,@fecha\_compra,@fecha\_factura))

        FETCH cursor\_compras\_pc INTO @numero\_compra, @fecha\_compra, @pc\_codigo\_compra

        FETCH cursor\_facturas\_pc INTO @numero\_factura, @fecha\_factura, @pc\_codigo\_factura

END

        CLOSE cursor\_compras\_pc

        CLOSE cursor\_facturas\_pc

DEALLOCATE cursor\_compras\_pc

        DEALLOCATE cursor\_facturas\_pc

END

* bi\_compra\_accesorio

CREATE PROCEDURE LAWE.migrar\_bi\_compra\_accesorio AS

BEGIN

    INSERT INTO  LAWE.bi\_compra\_accesorio (

        accesorio\_codigo,

        id\_fabricante,

        id\_sucursal,

        id\_tiempo,

        cantidad,

        total\_compra\_acc)

    SELECT

        i.accesorio\_codigo,

        a.id\_fabricante,

        f.id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        COUNT(\*),

        SUM(i.cantidad \* a.precio\_compra)

    FROM LAWE.item\_factura\_accesorio i

        JOIN LAWE.factura f ON i.numero\_factura = f.numero\_factura

        JOIN LAWE.accesorio a ON i.accesorio\_codigo = a.accesorio\_codigo

    GROUP BY

        i.accesorio\_codigo,

        a.id\_fabricante,

        f.id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha)

END

* bi\_factura\_accesorio

CREATE PROCEDURE LAWE.migrar\_bi\_factura\_accesorio AS

BEGIN

    INSERT INTO  LAWE.bi\_factura\_accesorio (

        accesorio\_codigo,

        id\_fabricante,

        id\_sucursal,

        id\_tiempo,

        id\_edad,

        cantidad,

        total\_factura\_acc,

        tiempo\_en\_stock)

    SELECT

        i.accesorio\_codigo,

        a.id\_fabricante,

        f.id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        LAWE.OBTENER\_ID\_EDAD(c.fecha\_nacimiento),

        COUNT(\*),

        SUM(i.cantidad \* a.precio\_compra \* 1.2),

        SUM(t.tiempo\_en\_stock) / COUNT(\*) AS tiempo\_stock\_promedio

  FROM LAWE.item\_factura\_accesorio i

        JOIN LAWE.factura f ON i.numero\_factura = f.numero\_factura

        JOIN LAWE.accesorio a ON i.accesorio\_codigo = a.accesorio\_codigo

        JOIN LAWE.cliente c ON f.id\_cliente = c.id\_cliente

        JOIN #tiempo\_en\_stock\_accesorios t ON f.numero\_factura = t.numero\_factura

    GROUP BY

        i.accesorio\_codigo,

        a.id\_fabricante,

        f.id\_sucursal,

        LAWE.OBTENER\_ID\_TIEMPO(fecha),

        LAWE.OBTENER\_ID\_EDAD(c.fecha\_nacimiento)

    DROP TABLE #tiempo\_en\_stock\_accesorios

END

Para calcular el tiempo de stock de los accesorios vendidos utilizamos una tabla temporal llamada #tiempo\_en\_stock\_accesorios, en la cual asociamos una factura con su respectiva compra para conocer la cantidad de días que permanecieron en stock los accesorios vendidos.

CREATE TABLE #tiempo\_en\_stock\_accesorios (

        numero\_compra DECIMAL(18),

        fecha\_compra DATETIME2(3),

        accesorio\_codigo\_compra NVARCHAR(50),

        numero\_factura DECIMAL(18),

        fecha\_factura DATETIME2(3),

        accesorio\_codigo\_factura NVARCHAR(50),

        tiempo\_en\_stock DECIMAL(18)

    )

Este Stored Procedure calcula el tiempo que estuvieron en stock un conjunto de accesorios comparando la fecha de su compra con la fecha de su correspondiente factura, basándonos en el concepto FIFO (La primer compra de accesorios corresponde a la primer factura de accesorios, y así sucesivamente).

Para realizar la asociación correctamente decidimos utilizar dos cursores de manera simultánea en el Stored Procedure los cuales recorren uno a uno los registros de las tablas de compras y facturas de accesorios.

CREATE PROCEDURE LAWE.calcular\_tiempo\_stock\_accesorios AS

BEGIN

    DECLARE cursor\_compras\_accesorios CURSOR

    FOR

        SELECT c.numero\_compra, fecha, i\_c\_acc.accesorio\_codigo

        FROM LAWE.item\_compra\_accesorio i\_c\_acc

            JOIN LAWE.compra c ON i\_c\_acc.numero\_compra = c.numero\_compra

        DECLARE @numero\_compra DECIMAL(18), @fecha\_compra DATETIME2(3), @accesorio\_codigo\_compra NVARCHAR(50)

    DECLARE cursor\_facturas\_accesorios CURSOR

    FOR

        SELECT f.numero\_factura, fecha, i\_f\_acc.accesorio\_codigo

        FROM LAWE.item\_factura\_accesorio i\_f\_acc

            JOIN LAWE.factura f ON f.numero\_factura = i\_f\_acc.numero\_factura

        DECLARE @numero\_factura DECIMAL(18), @fecha\_factura DATETIME2(3), @accesorio\_codigo\_factura NVARCHAR(50)

        OPEN cursor\_compras\_accesorios

        OPEN cursor\_facturas\_accesorios

        FETCH cursor\_compras\_accesorios INTO @numero\_compra, @fecha\_compra, @accesorio\_codigo\_compra

        FETCH cursor\_facturas\_accesorios INTO @numero\_factura, @fecha\_factura, @accesorio\_codigo\_factura

        WHILE (@@FETCH\_STATUS=0)

        BEGIN

            INSERT INTO #tiempo\_en\_stock\_accesorios

            VALUES(

                @numero\_compra,

                @fecha\_compra,

                @accesorio\_codigo\_compra,

                @numero\_factura,

                @fecha\_factura,

                @accesorio\_codigo\_factura,

                DATEDIFF(DAY,@fecha\_compra,@fecha\_factura))

        FETCH cursor\_compras\_accesorios

INTO @numero\_compra, @fecha\_compra, @accesorio\_codigo\_compra

FETCH cursor\_facturas\_accesorios

INTO @numero\_factura, @fecha\_factura, @accesorio\_codigo\_factura

        END

        CLOSE cursor\_compras\_accesorios

        CLOSE cursor\_facturas\_accesorios

        DEALLOCATE cursor\_compras\_accesorios

        DEALLOCATE cursor\_facturas\_accesorios

END

## **5.3 Creación de Índices**

Durante la migración nos percatamos que para acelerar el proceso de ejecución de consultas a las tablas compra\_pc y facturas\_pc era necesario el uso de índices ya que el tiempo de respuesta era muy largo.

Por lo cual creamos los siguientes índices **ix\_compra\_pc** y **ix\_factura\_pc** para mejorar el tiempo de búsqueda de compras y ventas en su respectivas tablas. Determinamos que hubo una notable diferencia en la performance en la migración de los datos al nuevo modelo.

CREATE INDEX ix\_compra\_pc ON LAWE.compra\_pc (numero\_compra);

CREATE INDEX ix\_factura\_pc ON LAWE.factura\_pc (numero\_factura);

# Vistas

Se desarrollaron las vistas correspondientes para cumplir con los requerimientos solicitados en el enunciado.

# **6.1 Vista - PC Compradas / Vendidas por mes y sucursal**

El nombre de la vista creada es **v\_cantidad\_pc\_comprada\_sucursal\_mes**

Esta vista nos permite visualizar la cantidad de computadoras compradas por sucursal y mes.

Las columnas que agregamos para mostrar dicha información

* ID de la sucursal
* Número de Mes
* Cantidad de computadoras compradas

CREATE VIEW LAWE.v\_cantidad\_pc\_comprada\_sucursal\_mes AS

    SELECT

        c.id\_sucursal,

        mes,

        SUM(c.cantidad) cantidad\_pc\_compradas

        FROM LAWE.bi\_compra\_pc c

            JOIN LAWE.bi\_tiempo t ON c.id\_tiempo = t.id\_tiempo

    GROUP BY c.id\_sucursal,mes

El nombre de la vista creada es **v\_cantidad\_pc\_vendida\_sucursal\_mes**

Esta vista nos permite visualizar la cantidad de computadoras vendidas por sucursal y mes.

Las columnas que agregamos para mostrar dicha información

* ID de la sucursal
* Número de Mes
* Cantidad de computadoras vendidas

CREATE VIEW LAWE.v\_cantidad\_pc\_vendida\_sucursal\_mes AS

    SELECT

        f.id\_sucursal,

        mes,

        SUM(f.cantidad) cantidad\_pc\_vendidas

        FROM LAWE.bi\_factura\_pc f

            JOIN LAWE.bi\_tiempo t ON f.id\_tiempo = t.id\_tiempo

    GROUP BY f.id\_sucursal,mes

## **6.2 Vista - PC - Ganancias por sucursal y mes**

El nombre de la vista creada es **v\_pc\_ganancias\_sucursal\_mes**

Dicha vista mostrará la suma total de ganancias por sucursal y mes, esto segundo fue el criterio que utilizamos para agrupar los datos.

Tuvimos en cuenta que el cálculo del mismo resulta de la diferencia entre el precio de venta y el de compra.

Las columnas que agregamos para mostrar dicha información

* ID de la sucursal
* Número de Mes
* Total de ganancias

CREATE VIEW LAWE.v\_pc\_ganancias\_sucursal\_mes AS

    SELECT

        f.id\_sucursal,

        mes,

        SUM(total\_factura - 0.8 \* total\_factura) ganancias

    FROM LAWE.bi\_factura\_pc f

        JOIN LAWE.bi\_tiempo t ON t.id\_tiempo = f.id\_tiempo

    GROUP BY id\_sucursal, mes

## **6.3 Vista - PC – Precio promedio vendido y comprado**

El nombre de la vista creada es **v\_pc\_precio\_promedio\_compra**

Dicha vista mostrará el promedio total vendido y comprado de computadoras.

Las columnas que agregamos para mostrar dicha información

* Código de PC
* Promedio de Precio de Compra
* Promedio de Precio de Venta

CREATE VIEW LAWE.v\_pc\_precio\_promedio\_compra AS

    SELECT

        pc.pc\_codigo,

        AVG(pc.precio\_compra) precio\_promedio\_compra,

        AVG(pc.precio\_venta) precio\_promedio\_venta

    FROM LAWE.bi\_compra\_pc c

        JOIN LAWE.bi\_tipo\_pc pc ON pc.pc\_codigo = c.pc\_codigo

        JOIN LAWE.bi\_factura\_pc f ON f.pc\_codigo = f.pc\_codigo

    GROUP BY pc.pc\_codigo

## **6.4 Vista - PC – Tiempo en stock promedio**

El nombre de la vista creada es **v\_pc\_tiempo\_promedio\_stock**

Esta vista permite visualizar el tiempo en stock promedio de cada modelo de pc.

Las columnas que agregamos para mostrar dicha información

* Código de PC
* Tiempo de stock promedio

CREATE VIEW LAWE.v\_pc\_tiempo\_promedio\_stock AS

    SELECT

        fpc.pc\_codigo,

        AVG(tiempo\_en\_stock) tiempo\_stock\_promedio

    FROM LAWE.bi\_factura\_pc fpc

    GROUP BY fpc.pc\_codigo

## **6.5 Vista - Accesorio - Máxima cantidad de Stock anual por cada sucursal**

El nombre de la vista creada es **v\_accesorio\_max\_cant\_stock.**

En esta otra vista calculamos la cantidad máxima de stock de cada accesorio en cada año por sucursal.

Las columnas a mostrar son

* ID de sucursal
* Código de accesorio
* Año
* Cantidad total de stock

CREATE VIEW LAWE.v\_accesorio\_max\_cant\_stock AS

    SELECT bc.id\_sucursal, bt.anio, bc.accesorio\_codigo, SUM(bc.cantidad) cantidad\_stock

    FROM LAWE.bi\_compra\_accesorio bc

        join LAWE.bi\_tiempo bt ON bc.id\_tiempo = bt.id\_tiempo

    GROUP BY bc.id\_sucursal, bt.anio,  bc.accesorio\_codigo

## **6.6 Vista - Accesorio - Máxima cantidad de Stock anual por sucursal**

El nombre de la vista creada es **v\_accesorio\_max\_cant\_stock2,** creamos esta vista ya que no estábamos seguros si la vista 6.5 cubre el requerimiento pedido en el enunciado.

En esta otra vista calculamos la cantidad máxima de stock de accesorios en cada año por sucursal.

Las columnas a mostrar son

* ID de sucursal
* Año
* Cantidad total de stock

CREATE VIEW LAWE.v\_accesorio\_max\_cant\_stock2 AS

    SELECT bc.id\_sucursal, bt.anio, SUM(bc.cantidad) cantidad\_stock

    FROM LAWE.bi\_compra\_accesorio bc

        JOIN LAWE.bi\_tiempo bt ON bc.id\_tiempo = bt.id\_tiempo

    GROUP BY bc.id\_sucursal, bt.anio

## **6.7 Vista - Accesorio - Ganancias anual por sucursal por mes**

El nombre de la vista creada es **v\_accesorio\_ganancias\_sucursal\_mes.**

Dicha vista mostrará la suma total de ganancias por sucursal y mes, esto segundo fue el criterio que utilizamos para agrupar los datos. Tuvimos en cuenta que el cálculo del mismo resulta de la diferencia entre el precio facturado y el de compra.

Las columnas a mostrar son

* ID de sucursal
* Número de mes
* Total de ganancias

CREATE VIEW LAWE.v\_accesorio\_ganancias\_sucursal\_mes AS

    SELECT

        fa.id\_sucursal,

        mes,

        SUM(total\_factura\_acc - 0.8 \* total\_factura\_acc) ganancias

    FROM LAWE.bi\_factura\_accesorio fa

        JOIN LAWE.bi\_tiempo t ON t.id\_tiempo = fa.id\_tiempo

    GROUP BY id\_sucursal, mes

## **6.8 Vista - Accesorio – Precio promedio vendido y comprado**

El nombre de la vista creada es **v\_accesorio\_precio\_promedio\_compra.**

En esta vista se muestra el promedio de tiempo en stock por accesorio.

Las columnas que agregamos para mostrar dicha información

* Código de Accesorio
* Promedio de Precio Facturado
* Promedio de Precio de Compra

CREATE VIEW LAWE.v\_accesorio\_precio\_promedio\_compra AS

    SELECT

        ca.accesorio\_codigo,

        AVG(a.precio) precio\_promedio\_compra,

        AVG(a.precio \* 1.2) precio\_promedio\_venta

    FROM LAWE.bi\_compra\_accesorio ca

        JOIN LAWE.bi\_accesorio a ON ca.accesorio\_codigo = a.accesorio\_codigo

        JOIN LAWE.bi\_factura\_accesorio fa ON fa.accesorio\_codigo = a.accesorio\_codigo

    GROUP BY ca.accesorio\_codigo

## **6.9 Vista - Accesorio – Tiempo stock promedio vendido y comprado**

El nombre de la vista creada es **v\_accesorio\_tiempo\_promedio\_stock.**

En esta vista se muestra el promedio de tiempo en stock de cada accesorio.

Las columnas que agregamos para mostrar dicha información

* Código de Accesorio
* Promedio de tiempo en Stock

CREATE VIEW LAWE.v\_accesorio\_tiempo\_promedio\_stock AS

    SELECT

        fa.accesorio\_codigo,

        AVG(tiempo\_en\_stock) tiempo\_stock\_promedio

    FROM LAWE.bi\_factura\_accesorio fa

    GROUP BY fa.accesorio\_codigo