UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

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

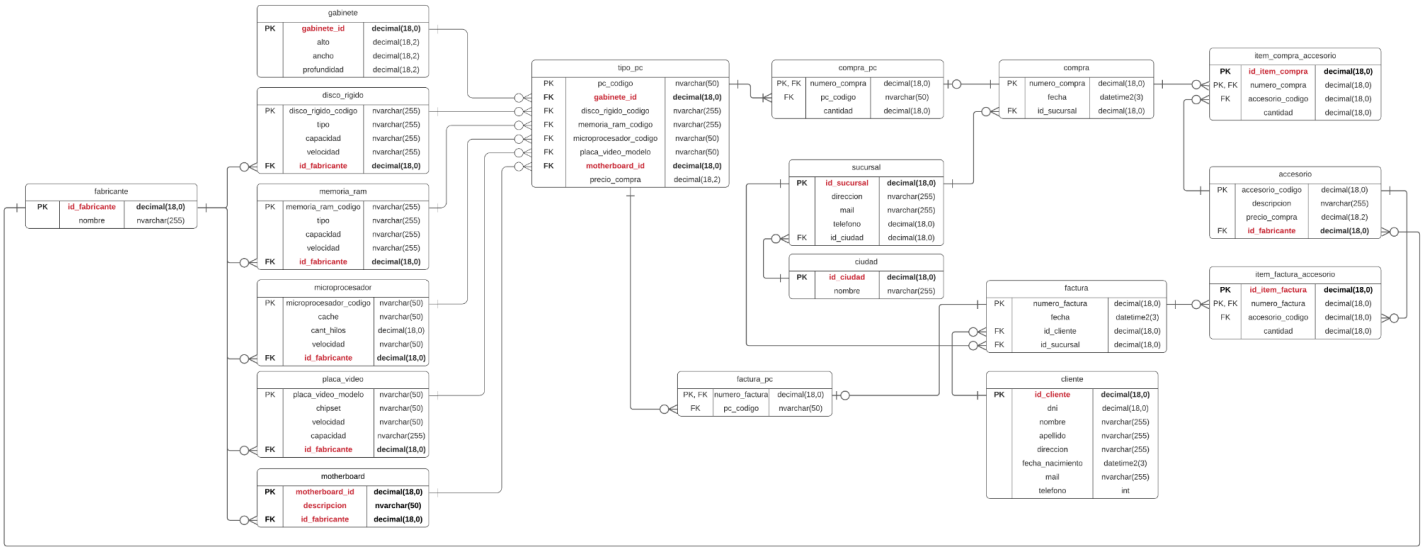
Fechas de entrega:

- 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

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:

cliente		
PK	id_cliente	decimal(18,0)
	dni	decimal(18,0)
	nombre	nvarchar(255)
	apellido	nvarchar(255)
	direccion	nvarchar(255)
	fecha_nacimiento	datetime2(3)
	mail	nvarchar(255)
	telefono	int

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:

```
select CLIENTE_NOMBRE, CLIENTE_APELLIDO, CLIENTE_DNI from gd_esquema.Maestra
where CLIENTE_DNI= 32042081

select CLIENTE_NOMBRE, CLIENTE_APELLIDO, CLIENTE_DNI from gd_esquema.Maestra
where CLIENTE_DNI= 82765842
```

	CLIENTE_NOMBRE	CLIENTE_APELLIDO	CLIENTE_DNI
1	VALERIA	González	32042081
2	ELIAN	Castro	32042081

	CLIENTE_NOMBRE	CLIENTE_APELLIDO	CLIENTE_DNI
1	CLAUS	Acosta	82765842
2	ULADIMIRO	Alarcón	82765842

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 factura de una pc o a la factura de un accesorio formarán parte de las tablas **factura_pc** y **item_factura_accesorio** respectivamente.

factura		
PK	numero_factura	decimal(18,0)
	fecha	datetime2(3)
FK	id_cliente	decimal(18,0)
FK	id_sucursal	decimal(18,0)

factura_pc		
PK, FK	numero_factura	decimal(18,0)
FK	pc_codigo	nvarchar(50)

item_factura_accesorio		
PK	**id_item_factura**	**decimal(18,0)**
PK, FK	numero_factura	decimal(18,0)
FK	accesorio_codigo	decimal(18,0)
	cantidad	decimal(18,0)

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.

compra		
PK	numero_compra	decimal(18,0)
	fecha	datetime2(3)
FK	id_sucursal	decimal(18,0)

compra_pc		
PK, FK	numero_compra	decimal(18,0)
FK	pc_codigo	nvarchar(50)
	cantidad	decimal(18,0)

item_compra_accesorio		
PK	id_item_compra	**decimal(18,0)**
PK, FK	numero_compra	decimal(18,0)
FK	accesorio_codigo	decimal(18,0)
	cantidad	decimal(18,0)

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.

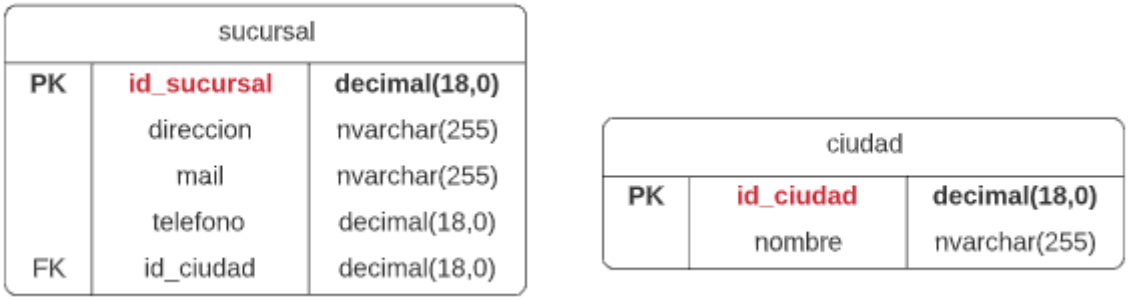
Esto nos permite que ante un eventual cambio en el nombre será más sencillo realizar las modificaciones sobre la tabla fabricante en vez de tener que recorrer las tablas de componentes y accesorios para realizar el cambio.

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.

accesorio		
PK	accesorio_codigo	decimal(18,0)
	descripcion	nvarchar(255)
	precio_compra	decimal(18,2)
FK	**id_fabricante**	**decimal(18,0)**

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.

tipo_pc		
PK	pc_codigo	nvarchar(50)
FK	**gabinete_id**	**decimal(18,0)**
FK	disco_rigido_codigo	nvarchar(255)
FK	memoria_ram_codigo	nvarchar(255)
FK	microprocesador_codigo	nvarchar(50)
FK	placa_video_modelo	nvarchar(50)
FK	**motherboard_id**	**decimal(18,0)**
	precio_compra	decimal(18,2)

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.

gabinete		
PK	**gabinete_id**	**decimal(18,0)**
	alto	decimal(18,2)
	ancho	decimal(18,2)
	profundidad	decimal(18,2)

Tabla microprocesador

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

microprocesador		
PK	microprocesador_codigo	nvarchar(50)
	cache	nvarchar(50)
	cant_hilos	decimal(18,0)
	velocidad	nvarchar(50)
FK	**id_fabricante**	**decimal(18,0)**

Tabla disco_rigido

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

disco_rigido		
PK	disco_rigido_codigo	nvarchar(255)
	tipo	nvarchar(255)
	capacidad	nvarchar(255)
	velocidad	nvarchar(255)
FK	**id_fabricante**	**decimal(18,0)**

Tabla placa_video

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

placa_video		
PK	placa_video_modelo	nvarchar(50)
	chipset	nvarchar(50)
	velocidad	nvarchar(50)
	capacidad	nvarchar(255)
FK	**id_fabricante**	**decimal(18,0)**

Tabla memoria_ram

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

memoria_ram		
PK	memoria_ram_codigo	nvarchar(255)
	tipo	nvarchar(255)
	capacidad	nvarchar(255)
	velocidad	nvarchar(255)
FK	**id_fabricante**	**decimal(18,0)**

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.

motherboard		
PK	**motherboard_id**	**decimal(18,0)**
	descripcion	**nvarchar(50)**
FK	**id_fabricante**	**decimal(18,0)**

2.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** y el 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**) y por el 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**) y por el 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
```

2.3 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
```

3. 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
```

3.7 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.

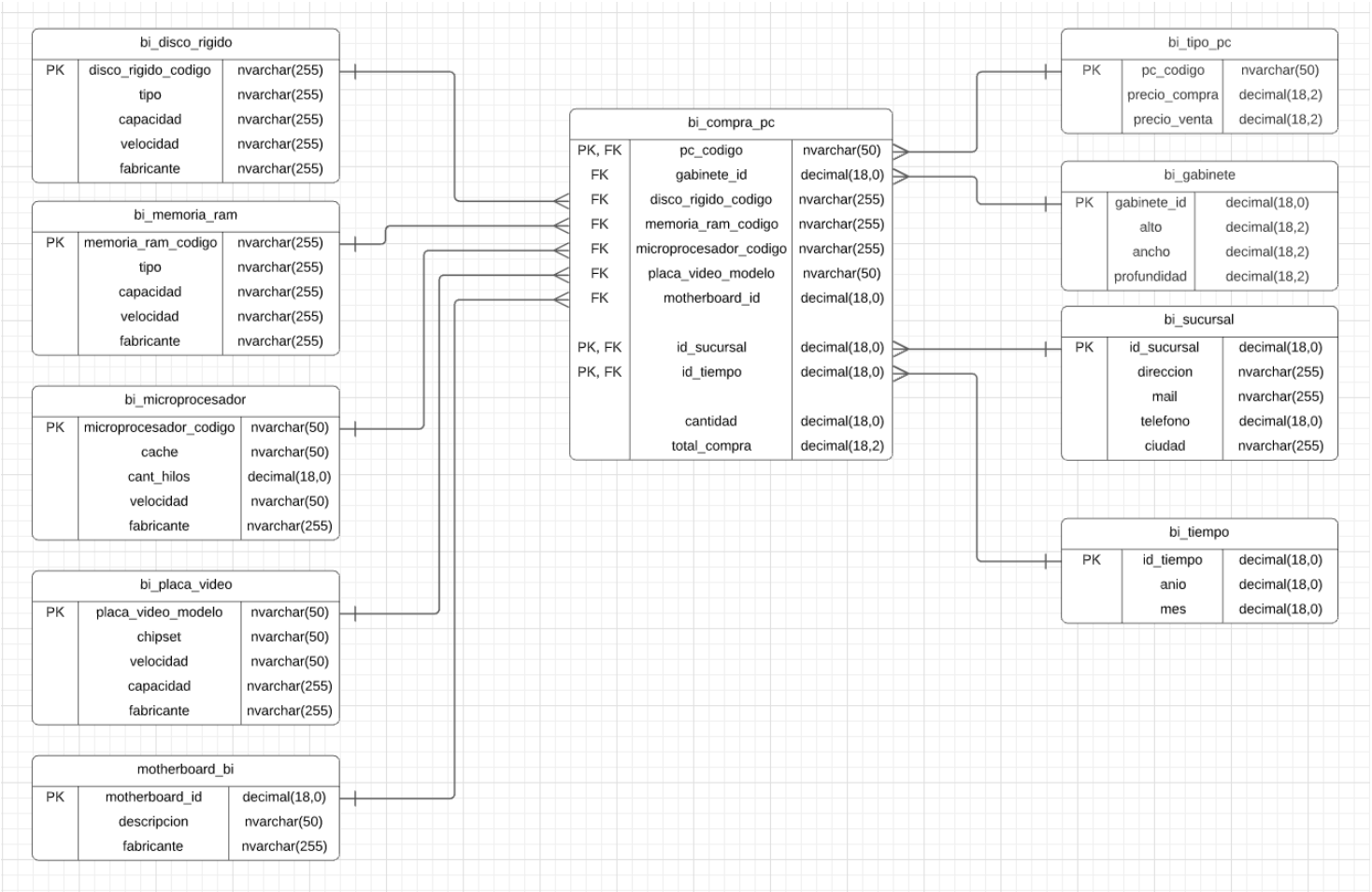
Para obtener dicha información de todos los componentes, utilizamos las siguientes tablas

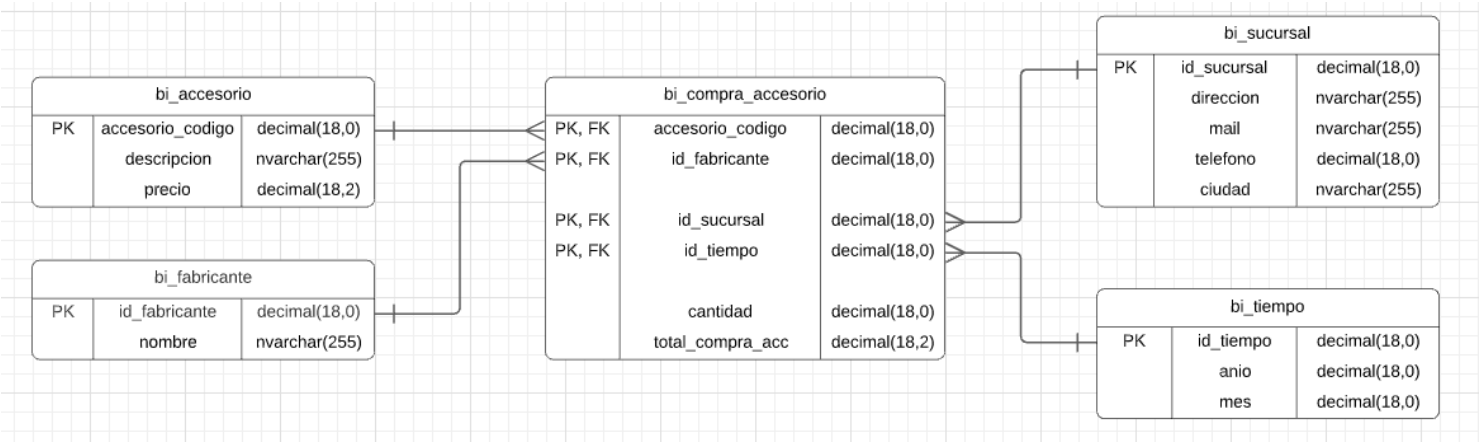
- gabinete
- disco_rigido
- memoria_ram
- microprocesador
- placa_video

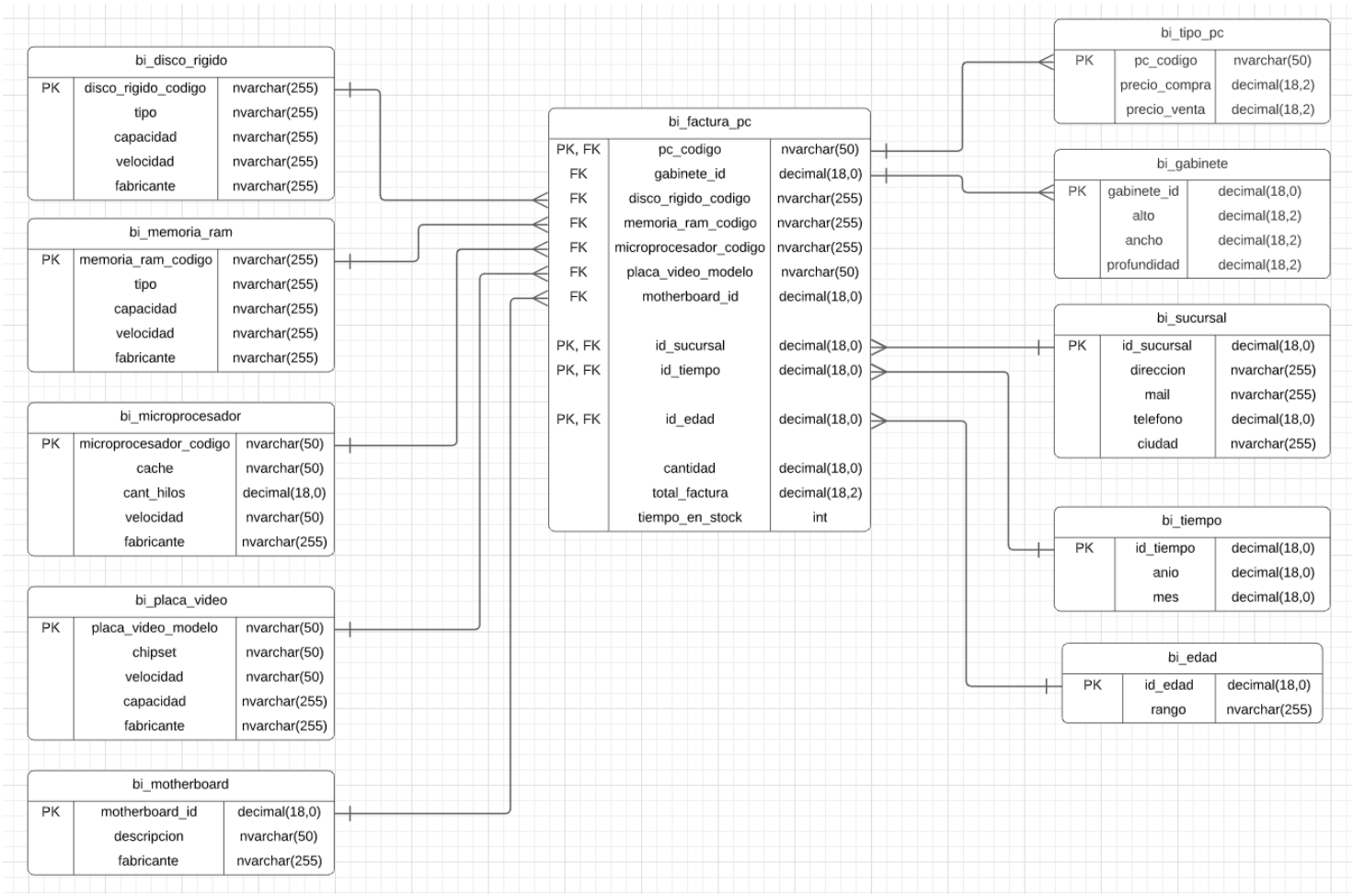
```
CREATE VIEW LAWE.v_modelos_de_pc AS
    SELECT
        tpc.pc_codigo AS Codigo_de_PC,
        tpc.precio_compra AS Precio_Compra_PC,
        -- datos del gabinete
        g.alto AS Alto_Gabinete,
        g.ancho AS Ancho_Gabinete,
        g.profundidad AS Profundidad_Gabinete,
        -- datos del 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
```

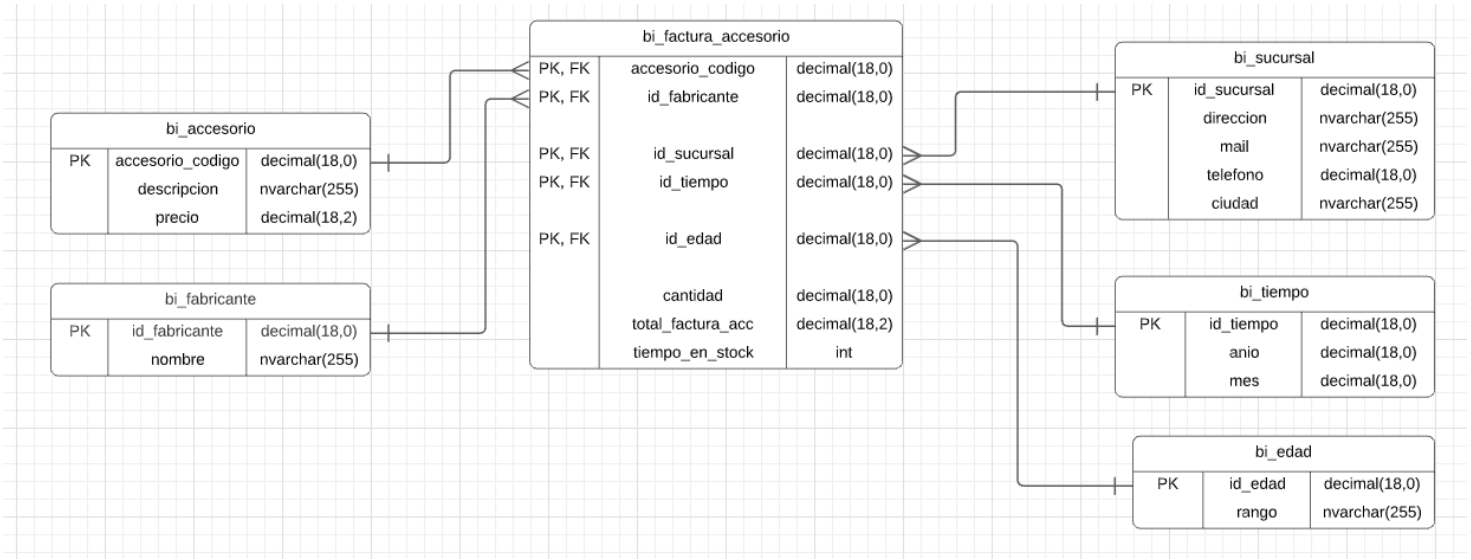
4. Modelo de Inteligencia de Negocios (BI)

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

bi_disco_rigido		
PK	disco_rigido_codigo	nvarchar(255)
	tipo	nvarchar(255)
	capacidad	nvarchar(255)
	velocidad	nvarchar(255)
	fabricante	nvarchar(255)

bi_memoria_ram		
PK	memoria_ram_codigo	nvarchar(255)
	tipo	nvarchar(255)
	capacidad	nvarchar(255)
	velocidad	nvarchar(255)
	fabricante	nvarchar(255)

bi_microprocesador		
PK	microprocesador_codigo	nvarchar(50)
	cache	nvarchar(50)
	cant_hilos	decimal(18,0)
	velocidad	nvarchar(50)
	fabricante	nvarchar(255)

bi_placa_video		
PK	placa_video_modelo	nvarchar(50)
	chipset	nvarchar(50)
	velocidad	nvarchar(50)
	capacidad	nvarchar(255)
	fabricante	nvarchar(255)

bi_motherboard		
PK	motherboard_id	decimal(18,0)
	descripcion	nvarchar(50)
	fabricante	nvarchar(255)

bi_tipo_pc		
PK	pc_codigo	nvarchar(50)
	precio_compra	decimal(18,2)
	precio_venta	decimal(18,2)

bi_gabinete		
PK	gabinete_id	decimal(18,0)
	alto	decimal(18,2)
	ancho	decimal(18,2)
	profundidad	decimal(18,2)

bi_sucursal		
PK	id_sucursal	decimal(18,0)
	direccion	nvarchar(255)
	mail	nvarchar(255)
	telefono	decimal(18,0)
	ciudad	nvarchar(255)

bi_accesorio		
PK	accesorio_codigo	decimal(18,0)
	descripcion	nvarchar(255)
	precio	decimal(18,2)

bi_fabricante		
PK	id_fabricante	decimal(18,0)
	nombre	nvarchar(255)

4.5 Tablas confeccionadas para el modelo de Business Intelligence

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

bi_tiempo		
PK	id_tiempo	decimal(18,0)
	anio	decimal(18,0)
	mes	decimal(18,0)

bi_edad		
PK	id_edad	decimal(18,0)
	rango	nvarchar(255)

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:

bi_compra_pc		
PK, FK	pc_codigo	nvarchar(50)
FK	gabinete_id	decimal(18,0)
FK	disco_rigido_codigo	nvarchar(255)
FK	memoria_ram_codigo	nvarchar(255)
FK	microprocesador_codigo	nvarchar(255)
FK	placa_video_modelo	nvarchar(50)
FK	motherboard_id	decimal(18,0)
PK, FK	id_sucursal	decimal(18,0)
PK, FK	id_tiempo	decimal(18,0)
	cantidad	decimal(18,0)
	total_compra	decimal(18,2)

bi_compra_accesorio		
PK, FK	accesorio_codigo	decimal(18,0)
PK, FK	id_fabricante	decimal(18,0)
PK, FK	id_sucursal	decimal(18,0)
PK, FK	id_tiempo	decimal(18,0)
	cantidad	decimal(18,0)
	total_compra_acc	decimal(18,2)

bi_factura_pc		
PK, FK	pc_codigo	nvarchar(50)
FK	gabinete_id	decimal(18,0)
FK	disco_rigido_codigo	nvarchar(255)
FK	memoria_ram_codigo	nvarchar(255)
FK	microprocesador_codigo	nvarchar(255)
FK	placa_video_modelo	nvarchar(50)
FK	motherboard_id	decimal(18,0)
PK, FK	id_sucursal	decimal(18,0)
PK, FK	id_tiempo	decimal(18,0)
PK, FK	id_edad	decimal(18,0)
	cantidad	decimal(18,0)
	total_factura	decimal(18,2)
	tiempo_en_stock	int

bi_factura_accesorio		
PK, FK	accesorio_codigo	decimal(18,0)
PK, FK	id_fabricante	decimal(18,0)
PK, FK	id_sucursal	decimal(18,0)
PK, FK	id_tiempo	decimal(18,0)
PK, FK	id_edad	decimal(18,0)
	cantidad	decimal(18,0)
	total_factura_acc	decimal(18,2)
	tiempo_en_stock	int

5. 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_r
am_codigo,microprocesador_codigo, placa_video_modelo, motherboard_id, id_sucursal, id_tiem
po, 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_codig
o_factura NVARCHAR(50)

        OPEN cursor_compras_accesorios
        OPEN cursor_facturas_accesorios

        FETCH cursor_compras_accesorios INTO @numero_compra, @fecha_compra, @accesorio_cod
igo_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);
```

6. 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

```sql
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

```sql
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

```sql
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
```