Gestión de Datos

Trabajo Práctico
1° Cuatrimestre 2020
FRBA - Viajes

Integrantes:

Peduto, Francisco
Rodriguez Cary, Hernán Ezequiel
Cervantes Ortiz, Marisol Carolina
Morsella, Leonel Lucas

Grupo 45:
SELECT_QUANTUM_LIBRARY

Información de sus Integrantes:

Integrante	Legajo	E-mail
Peduto, Francisco	164.210-8	franchupedu@outlook.com
Rodriguez Cary, Hernán Ezequiel	163.160-1	hernan_rodriguez_2013@hotmail.com
Cervantes Ortiz, Marisol Carolina	163.920-1	mar.y.sol.240@gmail.com
Morsella, Leonel Lucas	160.689-0	leonelmorsella@hotmail.com

ÍNDICE

DER **3**

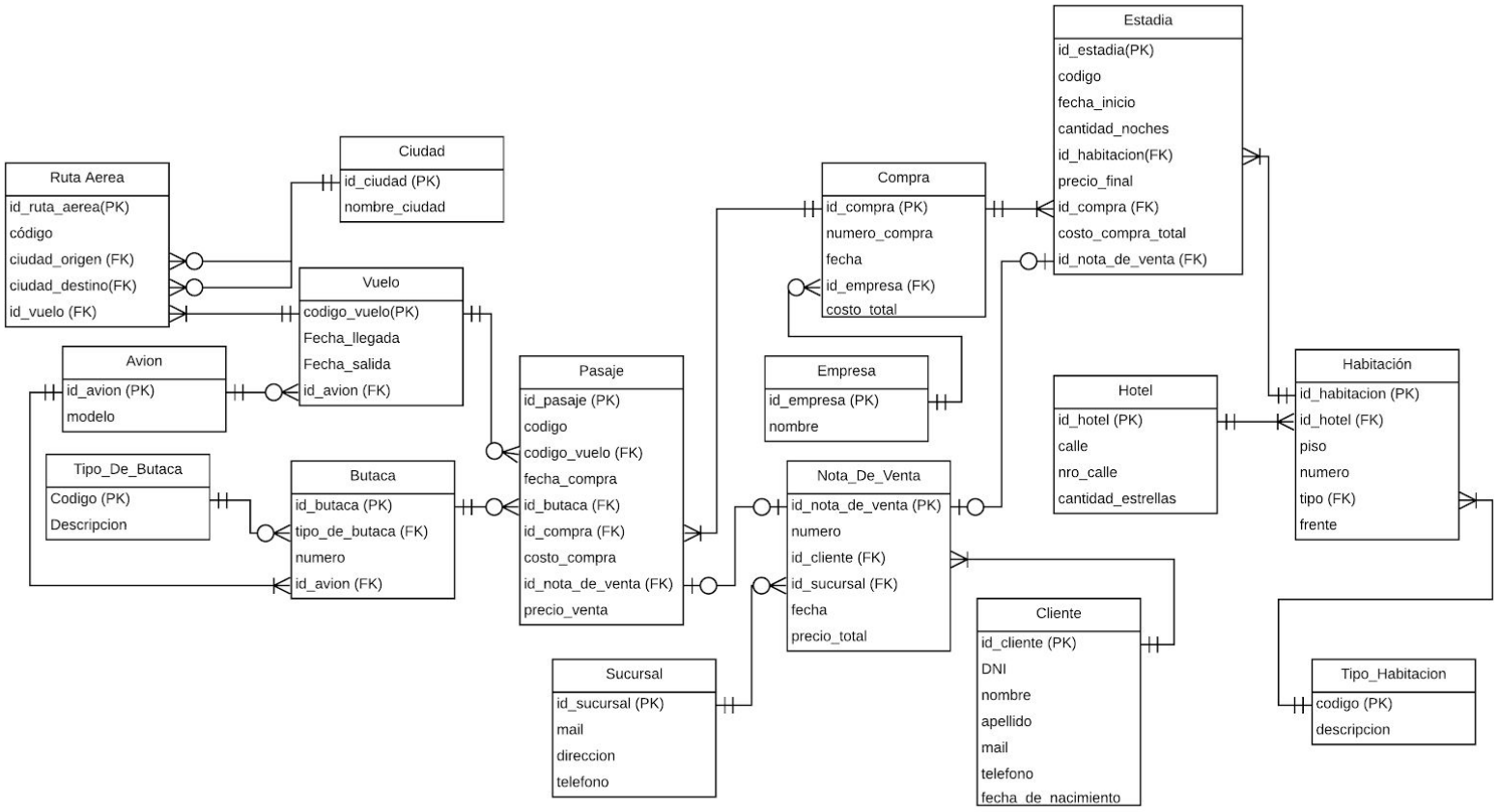
Justificación de Entidades **3**

Ciudad 3
Ruta Aérea 4
Avión 4
Tipo de Butaca 4
Butaca 4
Vuelo 5
Pasaje 5
Sucursal 5
Nota De Venta 6
Cliente 6
Compra 6
Empresa 6
Estadía 7
Hotel 7
Habitación 7
Tipo de Habitación 8

Explicación algoritmos usados **8**

Creates 8
Índices 10
Inserts 10
Updates 12

DER



Justificación de Entidades

Ciudad

Esta entidad refleja a los destinos y orígenes de los vuelos que realiza la empresa identificadas por una primary key llamada id_ciudad y tiene un campo que contiene el nombre de la ciudad llamada nombre_ciudad. En cuanto a sus relaciones, tiene una relación de uno a muchos con la tabla de rutas aéreas en su campo ciudad_origen, ya que conoce cuál es la ciudad de la que partió ese vuelo y esa ciudad puede estar en muchas rutas aéreas, y tiene una segunda relación de uno a muchos con la tabla de rutas aéreas en su campo ciudad_destino, ya que conoce la ciudad a la que llegó el vuelo y esa ciudad puede estar en muchas rutas.

Ruta Aérea

Esta tabla registra las rutas que van trazando los diferentes vuelos a lo largo del tiempo, diferenciando entre vuelos de diferentes escalas. Se identifica por una primary key llamada id_ruta_aerea y tiene los campos código, que contiene el código de la ruta aérea; la ciudad_origen, que es una foreign key al dato de la tabla de ciudades que contiene el dato de la ciudad de donde partió el vuelo; la ciudad_destino, que al igual que la ciudad_origen contiene una foreign key al dato de la tabla de ciudades que contiene la ciudad en donde finaliza el vuelo, y el id_vuelo, que es una foreign key a la tabla vuelo que apunta al registro del vuelo que recorrió esa ruta aérea, las rutas que son escalas del mismo vuelo tienen la misma foreign key en este último dato. La entidad tiene dos relaciones de uno a uno con ciudad, porque necesariamente hay una ciudad orígen y otra destino en cada ruta aérea. También hay una relación de uno a uno con vuelo, porque cada ruta aérea corresponde únicamente a un vuelo en particular.

Avión

Esta entidad refleja los aviones existentes para los cuales se los identifica cada uno inequívocamente con un campo id_avion como Primary Key y el modelo especifico del avión. La relación que posee a través de su PK es de muchos o ninguno con vuelo, dado que un avión puede ser usado en varios vuelos o en ninguno.

Tipo de Butaca

Esta entidad refleja los tipos de butaca existentes en un avión, identificando cada tipo existente con una Primary Key codigo y luego la descripción específica del tipo de butaca existente. En cuanto a las relaciones, tiene una de uno a muchos a la tabla de butacas, ya que una butaca solo tiene un tipo pero hay muchas butacas del mismo tipo.

Butaca

Esta entidad refleja las butacas existentes en cada avión, cada butaca posee una Primary Key id_butaca para identificar todas, se tiene ademas el codigo del tipo de butaca que es guardado como una Foreign Key llamada tipo_de_butaca, se tiene el numero de butaca del avión y finalmente el id_avion como una Foreign Key que identifica a qué avión pertenece dicha butaca.
Una butaca puede pertenecer a varios pasajes distintos, emitidos en diferentes momentos o a ninguno. Por esto se observa la relación de muchos o ninguno por parte de butaca.
Es de un solo tipo y es obligatorio que pertenezca a uno. Lo mismo se da con los aviones, una butaca en particular pertenece obligatoriamente a un solo vuelo.

Vuelo

Esta entidad refleja los vuelos programados, cada vuelo posee un código_vuelo como Primary Key que identifica cada vuelo programado existente, teniendo además su fecha de llegada y fecha de salida, Finalmente el id_avion como una Foreign key que identifica el avión involucrado en dicho vuelo. En cuanto a sus relaciones, tiene una relación de uno a muchos con la tabla de rutas aéreas, ya que cada vuelo puede tener una ó varias rutas aéreas por las escalas y las rutas aéreas se relacionan con un solo vuelo; tiene una relación de muchos a uno con la tabla de aviones, ya que cada avión puede participar de más de un vuelo, pero cada vuelo tiene solo un avión, y tiene una relación de uno a muchos con la tabla de pasajes, ya que se venden varios pasajes para un mismo vuelo, pero cada pasaje es solo para un vuelo en específico.

Pasaje

Esta entidad refleja cada pasaje emitido para un vuelo específico, cada pasaje tiene una Primary Key llamada id_pasaje que identifica cada pasaje existente, el pasaje posee además un código, el codigo_vuelo como Foreign Key del vuelo al que pertenece, la fecha en que se compró dicho pasaje, el id_butaca e id_compra como Foreigns Key de sus respectivas tablas identificando así la butaca asignada en el pasaje y el id de la compra, a su vez se tiene el costo por la compra del pasaje, un id_nota_de_venta como Foreign Key que muestra la Nota De venta asociada a el pasaje y finalmente el precio de venta por el pasaje. En cuanto a sus relaciones, tiene una relación de muchos a uno con la tabla de vuelos, ya que cada pasaje pertenece a un vuelo y por cada vuelo se venden varios pasajes; tiene una relación de muchos a uno con la tabla de butacas, ya que a cada pasaje le corresponde una butaca y una butaca puede relacionarse con varios pasajes a lo largo del tiempo en el registro histórico; tiene una relación de muchos a uno con la tabla de compras, ya que en una compra se pueden comprar varios pasajes, pero cada pasaje es comprado en una compra, y tiene una relación de uno a uno con la tabla de notas de venta, ya que al igual que con la compra a cada pasaje le corresponde una nota de venta y los pasajes se venden de a uno por cliente.

Sucursal

Esta entidad refleja las sucursales existentes para la venta de estadías o pasajes, cada sucursal es identificada por un id_sucursal como Primary Key, a su vez se posee el mail de la sucursal en específico, la dirección y teléfono. Cuenta con una relación de muchos o ninguno con nota de venta, ya que en una misma sucursal se pueden emitir varias de ellas o ninguna.

Nota De Venta

Esta entidad representa el evento de una compra la cual puede ser tanto de un pasaje como de una estadía ó ambos. La misma está identificada por una primary key llamada id_nota_de_venta y tiene los campos de número, que contiene el número de la venta; id_cliente, que contiene una foreign key que apunta al registro que contiene el cliente que realizó la compra en la tabla de clientes; id_sucursal, que contiene una foreign key que apunta al registro de la tabla de sucursales que contiene la sucursal e la que se realizó esta venta; fecha, que contiene la fecha en la que se realizó a compra, y precio_total, que contiene la sumatoria de los precios de todos los ítems vendidos en esta venta.
Una nota de venta pertenece exclusivamente a un cliente y es obligatorio que se especifique uno.
Como es posible emitir notas de venta para la venta de estadías o de pasajes, esta entidad tiene una relación de cero o uno con estas.

Cliente

Esta entidad refleja los clientes existentes en la base permitiendo identificarlos según un id_cliente como Primary Key, no se tomó el DNI como una Primary Key debido a que existen casos donde el mismo se repita, por lo que luego del id_cliente se guarda el DNI, nombre,apellido,mail, teléfono y fecha de nacimiento del cliente, conteniendo así todos los datos básicos del Cliente. En cuanto a sus relaciones, tiene una relación de uno a muchos con la tabla de notas de venta, ya que un cliente puede comprar varias veces a lo largo del tiempo y una venta es a un cliente.

Compra

Esta entidad posee un id como clave primaria, la fecha en que se realizó la misma, el id de la empresa de la que se compró y el monto total de la misma. El número también identifica la compra.
Una misma compra puede tener varios pasajes o varias estadías, siendo obligatorio al menos uno de ellos.
Estas compras siempre se realizan a una misma empresa, por lo que si se compran pasajes, todos pertenecen a la misma aerolínea, y si se compran estadías, todas son de un mismo hotel.

Empresa

Esta entidad refleja las empresas que realizan las compras de las estadías, para esta entidad solo es necesario un id_empresa como Primary Key que identifique inequívocamente cada empresa y el nombre de la misma. En cuanto a sus relaciones,

tiene una de uno o muchos con la tabla de compras, ya que nuestra empresa de viajes puede hacer varias compras a la misma empresa o ninguna.

Estadía

Esta entidad refleja un conjunto de noches alquiladas para 1 habitacion en 1 determinado hotel, siendo estas adquiridas por una empresa para su reventa. Su Primary Key es id_estadia, una FK a la habitación de dicha estadía, una FK id_compra a la compra realizada por parte de la empresa, y una FK id_nota_de_venta en caso de que la estadía ya haya sido revendida a un cliente. En cuanto a sus relaciones, tiene una de muchos a uno con la tabla de compras, ya que cada estadía puede ser comprada una vez, pero una compra puede contener varias estadías; tiene una de uno a uno con la tabla de notas de venta, ya que en cada nota de venta solo se puede comprar una estadía a la vez y cada estadía solo puede ser vendida una vez, y tiene una de muchos a uno con la tabla de habitaciones, ya que cada estadía está acompañada por una habitación a la vez y las habitaciones pueden ser utilizadas en varias estadías a lo largo del timepo en el registro histórico.

Hotel

Esta entidad refleja cada hotel existente en la base de datos, poseyendo un id_hotel como Primary Key que identifica cada hotel inequívocamente, luego se tienen los campos calle, nro_calle que reflejan los datos de dirección del hotel y finalmente cantidad_estrellas que reflejan el nivel de servicio del hotel. En cuanto a sus relaciones, tiene una de uno a muchos con la tabla de habitaciones, ya que cada habitación está en un hotel, pero cada hotel tiene varias habitaciones.

Habitación

Esta entidad refleja cada habitación existente en un hotel en específico pudiendo ser identificada las habitaciones de los hoteles con su Primary Key id_habitacion. Ahora bien, dichas habitaciones también están sujetas a un id_hotel siendo una Foreign Key que identifica el hotel al que pertenece la habitación en particular, luego con los campos piso y número identifica el número que posee la habitación en particular y el piso donde se encuentra en el hotel, se tiene además la Foreign Key tipo que identifica el tipo de habitación que es.
Finalmente se tiene un campo frente el cual identifica si la habitación se encuentra en el frente del hotel o no. En cuanto a sus relaciones, tiene una de uno a muchos con la tabla de estadías, ya que cada estadía es en una habitación y cada habitación puede ser usada en varias estadías a lo largo del tiempo en el registro histórico; tiene una de muchos a uno con la tabla de hoteles, ya que cada habitación está en un hotel y cada hotel tiene varias habitaciones, y tiene una de muchos a uno con la tabla de tipos de habitación, ya que cada habitación tiene un tipo y hay varias habitaciones del mismo tipo.

Tipo de Habitación

Esta entidad refleja los distintos tipos existentes de habitaciones que posee un hotel en particular, contando así con una Primary Key codigo que refleja el código que identifica al tipo de habitación y también a su vez cuenta con su descripción en donde se enuncia el tipo que posee dicha habitación.
Se relaciona con habitación debido a que muchas habitaciones pueden ser del mismo tipo. Además, en nuestro negocio, ofrecemos un tipo de habitación si al menos tenemos una.

Explicación algoritmos usados

Para el archivo de migración se usaron distintos tipos de sentencias, empezando con un create que se usa tanto para la creación del esquema como para las distintas tablas. En la creación de estas últimas se usaron constraints para marcar lo que se detalla a continuación:

Creates

Se decidió usar el código que ya venía en “tipo de habitación” en la tabla maestra ya que eran pocos los tipos de habitaciones y con el código que ya venía incluido no habría conflicto con la identificación unívoca de estos como podría suceder en otras tablas (véase create de Cliente por ej). Las constraints de NOT NULL son para justamente asegurarnos de no meter registros que no tuvieran un dato relevante.
El create de Avion, Vuelo y Pasaje son análogos a lo expuesto.

```
CREATE TABLE SELECT_QUANTUM_LIBRARY.Tipo_Habitacion (
codigo INT NOT NULL,
descripcion NVARCHAR (255) NOT NULL,
PRIMARY KEY (codigo)
);
```

En este caso no teníamos un dato para diferenciar el tipo de butaca por lo que recurrimos a un identity para crear su PK.
Los create de Butaca, Sucursal, Cliente, Hotel, Habitacion, Ciudad, Ruta_Aerea, Empresa, Compra, Nota_De_Venta y Estadia son análogos a lo expuesto.

```
CREATE TABLE SELECT_QUANTUM_LIBRARY.Tipo_Butaca (
codigo INT IDENTITY (1,1),
descripcion NVARCHAR (255) NOT NULL,
PRIMARY KEY (codigo)
);
```

Algo para destacar del create en la tabla Cliente, es que en primera instancia se iba a usar el DNI como PK para diferenciar unívocamente a cada uno, pero ya que en la tabla maestra notamos que se repetía para dos clientes diferentes terminamos por usar un identity.

```
CREATE TABLE SELECT_QUANTUM_LIBRARY.Cliente (
id_cliente INT IDENTITY (1,1),
DNI INT NOT NULL,
nombre NVARCHAR (255) NOT NULL,
apellido NVARCHAR (255) NOT NULL,
mail NVARCHAR (255) NOT NULL,
telefono INT NOT NULL,
fecha_de_nacimiento DATE NOT NULL
PRIMARY KEY (id_cliente)
);
```

Índices

Luego de terminados los creates se procedió a la creación de dos índices particulares: El DNI y el Mail ambos de la tabla del cliente. Esto se realizó para agilizar la migración a esta tabla ya que fue la que más registros logró alcanzar y por tanto la que más tardaba.

```
CREATE INDEX IX_Cliente_DNI ON SELECT_QUANTUM_LIBRARY.Cliente (DNI) WHERE DNI IS NOT NULL

CREATE INDEX IX_Cliente_Mail ON SELECT_QUANTUM_LIBRARY.Cliente (mail) WHERE mail IS NOT NULL

GO
```

Inserts

Luego se continuó insertando los datos en particular en cada tabla dichos insert se decidió implementarlos dentro de un Stored Procedure que realice la Migración.

```
CREATE PROCEDURE SELECT_QUANTUM_LIBRARY.Migracion AS
```

Dentro del Stored Procedure se prosiguió a crear todos los insert necesarios.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Ciudad SELECT DISTINCT
RUTA_AEREA_CIU_ORIG
FROM gd_esquema.Maestra
WHERE RUTA_AEREA_CIU_ORIG IS NOT NULL
```

para este caso como el id_ciudad es autogenerado al crear las tablas en particular únicamente fue necesario el traer una única vez la ciudades de origen que se mostraban en la tabla maestra.
para el caso de la tabla Tipo_Butaca se realizó el mismo proceso de insert en relación del id el cambio que se observa es que en dicha tabla se trae el campo Butaca_Tipo existente en la tabla.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Tipo_Habitacion SELECT DISTINCT
TIPO_HABITACION_CODIGO,
TIPO_HABITACION_DESC
FROM gd_esquema.Maestra
WHERE TIPO_HABITACION_CODIGO IS NOT NULL
```

Para el caso de los tipo de habitación el valor de la PK siendo el código inequívoco que identifica al tipo de habitación el id del tipo de habitación es insertado desde la tabla maestra, luego únicamente se inserta la descripción de la misma.
Esta estructura de insert se repite en Avión ya que la Primary Key es traída también desde la tabla maestra.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Vuelo SELECT
VUELO_CODIGO,
VUELO_FECHA_SALUDA,
VUELO_FECHA_LLEGADA,
AVION_IDENTIFICADOR
FROM gd_esquema.Maestra
WHERE VUELO_FECHA_SALUDA IS NOT NULL
GROUP BY VUELO_CODIGO, VUELO_FECHA_SALUDA, VUELO_FECHA_LLEGADA, AVION_IDENTIFICADOR
```

Para el caso del insert de el Vuelo fue necesario luego de insertar todos los datos referentes al vuelo un Group By de todos sus campos porque el código de vuelo se repetía en la tabla maestra para distintas rutas aéreas lo que terminaba generando duplicados de un mismo vuelo en la migración.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Habitacion SELECT DISTINCT
(SELECT id_hotel FROM SELECT_QUANTUM_LIBRARY.Hotel WHERE calle = HOTEL_CALLE AND nro_calle = HOTEL_NRO_CALLE),
HABITACION_PISO,
HABITACION_NUMERO,
(SELECT codigo FROM SELECT_QUANTUM_LIBRARY.Tipo_Habitacion WHERE codigo = TIPO_HABITACION_CODIGO),
HABITACION_FRENTE
FROM gd_esquema.Maestra
WHERE HABITACION_NUMERO IS NOT NULL
```

Para el caso del insert de Habitación fue necesario utilizar subselects en los campos id_hotel y código de tipo de habitación para poder conseguir los códigos específicos relacionados a la habitación en la tabla maestra.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Nota_De_Venta SELECT DISTINCT
FACTURA_NRO,
(SELECT id_cliente FROM SELECT_QUANTUM_LIBRARY.Cliente WHERE CLIENTE_DNI = DNI AND mail = CLIENTE_MAIL)
(SELECT id_sucursal FROM SELECT_QUANTUM_LIBRARY.Sucursal WHERE direccion = SUCURSAL_DIR),
FACTURA_FECHA,
ISNULL (PASAJE_PRECIO, 0) + ISNULL (HABITACION_PRECIO * ESTADIA_CANTIDAD_NOCHES, 0)
FROM gd_esquema.Maestra
WHERE FACTURA_NRO IS NOT NULL
```

Para el caso del insert de la Nota de venta no solo fueron necesarios los subselect para traer los datos relacionados desde la tabla maestra sino también el validar que los precios referentes en la nota de venta no estuviesen en Null, para la resolución de esta situación se optó por un ISNULL en donde en caso de ser null el valor será 0 permitiendo así realizar la suma necesaria.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Estadia SELECT DISTINCT
ESTADIA_CODIGO,
ESTADIA_FECHA_INI,
ESTADIA_CANTIDAD_NOCHES,
(SELECT id_habitacion FROM SELECT_QUANTUM_LIBRARY.Habitacion Ha JOIN SELECT_QUANTUM_LIBRARY.Hotel Ho ON Ho.id_hotel = Ha.id_hotel
WHERE Ho.calle = HOTEL_CALLE AND Ho.nro_calle = HOTEL_NRO_CALLE AND Ha.numero = HABITACION_NUMERO),
ESTADIA_CANTIDAD_NOCHES * HABITACION_PRECIO,
(SELECT id_compra FROM SELECT_QUANTUM_LIBRARY.Compra WHERE COMPRA_NUMERO = numero_compra),
ESTADIA_CANTIDAD_NOCHES * HABITACION_COSTO,
NULL
FROM gd_esquema.Maestra
WHERE ESTADIA_CODIGO IS NOT NULL
```

Para el insert referente a las estadías además de subselects para mantener la relación existente de la tabla maestra, a su vez también fue necesario para el costo total realizar el cálculo de la cantidad de noches por el costo unitario de la habitación por noche.

Updates

```
UPDATE SELECT_QUANTUM_LIBRARY.Pasaje SET id_nota_de_venta =
(SELECT id_nota_de_venta FROM SELECT_QUANTUM_LIBRARY.Nota_De_Venta WHERE numero_venta = M.FACTURA_NRO)
FROM SELECT_QUANTUM_LIBRARY.Pasaje P JOIN gd_esquema.Maestra M ON M.PASAJE_CODIGO = P.codigo_pasaje
WHERE (SELECT id_nota_de_venta FROM SELECT_QUANTUM_LIBRARY.Nota_De_Venta WHERE numero_venta = M.FACTURA_NRO) IS NOT NULL
```

```
UPDATE SELECT_QUANTUM_LIBRARY.Estadia SET id_nota_de_venta =
(SELECT id_nota_de_venta FROM SELECT_QUANTUM_LIBRARY.Nota_De_Venta WHERE numero_venta = M.FACTURA_NRO)
FROM SELECT_QUANTUM_LIBRARY.Estadia E JOIN gd_esquema.Maestra M ON M.ESTADIA_CODIGO = E.codigo
WHERE (SELECT id_nota_de_venta FROM SELECT_QUANTUM_LIBRARY.Nota_De_Venta WHERE numero_venta = M.FACTURA_NRO) IS NOT NULL
```

Tanto el update de pasaje como el de estadía, fueron necesarios ya que en la tabla maestra estas dos entidades podían o no tener su referencia a una nota de venta como también en null, ya sea porque fue vendida o porque no lo fue respectivamente. Para esto primero se realizó un insert en estas tablas de los registros donde el id nota de venta fuera null (los que no fueron vendidos) y luego con este update los que sí se vendieron.

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Pasaje SELECT DISTINCT
PASAJE_CODIGO,
VUELO_CODIGO,
COMPRA_FECHA,
(SELECT id_butaca FROM SELECT_QUANTUM_LIBRARY.Butaca B JOIN SELECT_QUANTUM_LIBRARY.Tipo_Butaca TB
(SELECT id_compra FROM SELECT_QUANTUM_LIBRARY.Compra WHERE COMPRA_NUMERO = numero_compra),
PASAJE_COSTO,
NULL,
PASAJE_PRECIO
FROM gd_esquema.Maestra
WHERE PASAJE_CODIGO IS NOT NULL
```

```
INSERT INTO SELECT_QUANTUM_LIBRARY.Estadia SELECT DISTINCT
ESTADIA_CODIGO,
ESTADIA_FECHA_INI,
ESTADIA_CANTIDAD_NOCHES,
(SELECT id_habitacion FROM SELECT_QUANTUM_LIBRARY.Habitacion Ha JOIN SELECT_QUANTUM_LIBRARY.Hot
ESTADIA_CANTIDAD_NOCHES * HABITACION_PRECIO,
(SELECT id_compra FROM SELECT_QUANTUM_LIBRARY.Compra WHERE COMPRA_NUMERO = numero_compra),
ESTADIA_CANTIDAD_NOCHES * HABITACION_COSTO,
NULL
FROM gd_esquema.Maestra
WHERE ESTADIA_CODIGO IS NOT NULL
```

Finalmente se procede a la ejecución de este procedure.

```
EXECUTE SELECT_QUANTUM_LIBRARY.Migracion;
```