**VMT DEV**

**Trabajo Grupal**

# Ejecución de sentencias: Normalización, Creación de Constrains y Restricciones, y consultas agrupadas

**Integrantes:**

**Boris Ismael Suarez Contreras**

**Christell Nicole Baño Cordero**

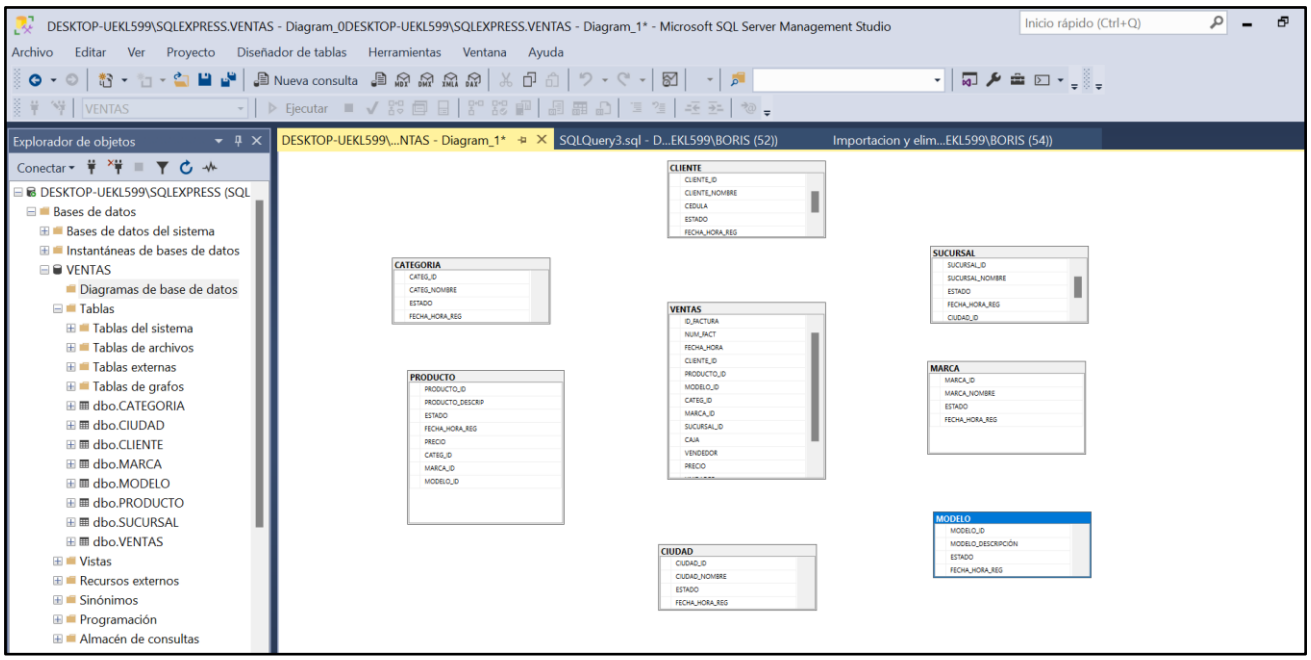
**Joshua Bryan Sarez Chiriguaya**

**Cristopher Juninho Vanegas Santi**

**Fecha: 17/06/2024**

## 1.Normalizacion de la base de datos VENTAS:

Tablas antes de la normalización




**Paso 1:** Crear una tabla llamada CAJA con 3 campos: CAJA_ID, CAJA_DESCRIPCION, ESTADO; y una tabla llamada VENDEDOR con 3 campos: VENDEDOR_ID, VENDEDOR_DESCRIPCION, ESTADO

```
USE VENTAS

CREATE TABLE CAJA
(CAJA_ID INT NOT NULL,
 CAJA_DESCRIPCION VARCHAR(50),
 ESTADO CHAR(1))

CREATE TABLE VENDEDOR
(VENDEDOR_ID INT NOT NULL,
 VENDEDOR_DESCRIPCION VARCHAR(50),
 ESTADO CHAR(1))
```

**Paso 2:** Cambiar los nombres de los campos de CAJA Y VENDEDOR de la tabla VENTAS a CAJA_ID y VENDEDOR_ID

```
EXEC SP_RENAME 'VENTAS.CAJA', 'CAJA_ID', 'COLUMN'
EXEC SP_RENAME 'VENTAS.VENDEDOR', 'VENDEDOR_ID', 'COLUMN'
```

**Paso 3:** Revisar los registros de VENTAS para saber si las CAJAS y VENDEDORES que se encuentren en la tabla

```
SELECT DISTINCT CAJA_ID FROM VENTAS
SELECT DISTINCT VENDEDOR_ID FROM VENTAS
```

Resultados | Mensajes

| | CAJA_ID |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

| | VENDEDOR_ID |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |

**Paso 4:** Crear registros en la tabla CAJA y en la tabla VENDEDORES

```
INSERT INTO CAJA (CAJA_ID, CAJA_DESCRIPCION, ESTADO ) VALUES (1, 'CAJA 001', 'A')
INSERT INTO CAJA (CAJA_ID, CAJA_DESCRIPCION, ESTADO ) VALUES (2, 'CAJA 002', 'A')
INSERT INTO CAJA (CAJA_ID, CAJA_DESCRIPCION, ESTADO ) VALUES (3, 'CAJA 003', 'A')

INSERT INTO VENDEDOR (VENDEDOR_ID, VENDEDOR_DESCRIPCION, ESTADO) VALUES (1,'MARCELO SUAREZ','A')
INSERT INTO VENDEDOR (VENDEDOR_ID, VENDEDOR_DESCRIPCION, ESTADO) VALUES (2,'FERNANDO SOLIS','A')
INSERT INTO VENDEDOR (VENDEDOR_ID, VENDEDOR_DESCRIPCION, ESTADO) VALUES (3,'ANITA RIOS','A')
```

**Paso 5:** Reemplazar los campos de CAJA_ID y VENDEDOR_ID con los códigos nuevos de las tablas CAJA y VENDEDOR para que queden códigos numéricos.

```
UPDATE VENTAS SET CAJA_ID = 1 WHERE CAJA_ID ='001-MARCELO PEREZ'
UPDATE VENTAS SET CAJA_ID = 2 WHERE CAJA_ID ='002-JUAN FERNANDO'
UPDATE VENTAS SET CAJA_ID = 3 WHERE CAJA_ID ='003-MARCELA RIOS'

UPDATE VENTAS SET VENDEDOR_ID = 1 WHERE VENDEDOR_ID ='V1-MARCELO SUAREZ'
UPDATE VENTAS SET VENDEDOR_ID = 2 WHERE VENDEDOR_ID ='V2-FERNANDO SOLIS'
UPDATE VENTAS SET VENDEDOR_ID = 3 WHERE VENDEDOR_ID ='V3-ANITA RIOS'

SELECT * FROM VENTAS
```

Resultados | Mensajes

| ACT | FECHA_HORA | CLIENTE_ID | PRODUCTO_ID | MODELO_ID | CATEG_ID | MARCA_ID | SUCURSAL_ID | CAJA_ID | VENDEDOR_ID |
|---|---|---|---|---|---|---|---|---|---|
| -4785 | 2016-12-28 22:11:00.000 | 3 | 20 | 1 | 1 | 1 | 5 | 1 | 2 |
| -4758 | 2018-11-25 16:40:00.000 | 4 | 10 | 1 | 2 | 3 | 3 | 1 | 2 |
| -4378 | 2018-12-06 18:06:00.000 | 7 | 3 | 1 | 2 | 3 | 3 | 2 | 3 |
| -4235 | 2017-12-17 11:02:00.000 | 3 | 4 | 1 | 1 | 2 | 5 | 1 | 3 |
| -4785 | 2016-10-10 07:03:00.000 | 4 | 8 | 1 | 2 | 2 | 2 | 2 | 1 |
| -4317 | 2018-06-13 18:36:00.000 | 6 | 4 | 1 | 1 | 2 | 4 | 3 | 3 |
| -4395 | 2018-04-18 07:54:00.000 | 2 | 8 | 1 | 2 | 2 | 5 | 3 | 1 |
| -4859 | 2016-05-12 18:59:00.000 | 4 | 2 | 1 | 2 | 3 | 1 | 1 | 3 |
| -4414 | 2017-01-29 22:18:00.000 | 3 | 8 | 1 | 2 | 2 | 3 | 1 | 3 |
| -4825 | 2017-05-05 09:08:00.000 | 3 | 20 | 1 | 1 | 1 | 4 | 1 | 2 |

**Paso 6:** Actualizar el campo ESTADO dejando 1 para REGISTRADA y 2 para ANULADA. Luego cambie la columna ESTADO de tipos de dato a entero.

```
UPDATE VENTAS SET ESTADO = '1' WHERE ESTADO = 'Registrada'
UPDATE VENTAS SET ESTADO = '2' WHERE ESTADO = 'Anulada'

ALTER TABLE VENTAS ALTER COLUMN ESTADO INT;
```

**2. Creación de CONSTRAINTS o RESTRICCIONES**

**Paso 1:** Crear un índice para el campo de fecha en la tabla VENTAS

```
--INDICE
CREATE INDEX INDICE1 ON VENTAS (FECHA_HORA)
```

**Paso 2:** Crear una clave primaria (PK) en la tabla principal (PRODUCTO), y una clave foránea (FK) en la tabla secundaria (VENTAS) con restricción para actualizar y eliminar en cascada. La PK no permite que 2 registros tengan la misma clave y la FK no permite que se le agreguen en PRODUCTO_ID y que no exista en PRODUCTO.

```
--a) CREE UNA CLAVE PRIMARIA (PK) EN LA TABLA PRINCIPAL (PRODUCTO). LA PK NO PERMITE DOS REGISTROS TENGAN LA MISMA CLAV
ALTER TABLE PRODUCTO ALTER COLUMN PRODUCTO_ID FLOAT NOT NULL --PARA QUITAR NULOS EN LAS TABLAS
ALTER TABLE PRODUCTO
ADD CONSTRAINT PK_PRODUCTO -- NOMBRE DE LA RELACIÓN QUE USTED DESEE PONER
PRIMARY KEY(PRODUCTO_ID) -- CAMPO PRODUCTO_ID DE LA TABLA PRODUCTO

--b) CREE UNA CLAVE FORÁNEA (FK) EN LA TABLA SECUNDARIA (VENTA) CON RESTRICCIÓN PARA ACTUALIZAR Y ELIMINAR EN CASCADA.
ALTER TABLE VENTAS WITH CHECK
ADD CONSTRAINT FK_PRODUCTO_VENTAS -- NOMBRE DE LA RELACIÓN QUE USTED DESEE PONER
FOREIGN KEY(PRODUCTO_ID) -- CAMPO PRODUCTO_ID DE LA TABLA VENTAS
REFERENCES PRODUCTO (PRODUCTO_ID) -- REALACIÓN A TABLA PRODUCTO CON CAMPO PRODUCTO_ID
ON UPDATE CASCADE -- ACTUALIZA EN CASCADA
ON DELETE CASCADE -- BORRA EN CASCADA
```

**Paso 3:** Realizar la prueba

```
--Realizar la prueba:
SELECT * FROM PRODUCTO WHERE PRODUCTO_ID = 20
SELECT * FROM VENTAS WHERE PRODUCTO_ID = 20

SELECT * FROM PRODUCTO WHERE PRODUCTO_ID = 1
SELECT * FROM VENTAS WHERE PRODUCTO_ID = 1

UPDATE PRODUCTO SET PRODUCTO_ID = 20 WHERE PRODUCTO_ID = 1
```

### 3. Prueba de TRANSACCIONES

Probar la ejecución de transacciones actualizando el precio de los registros de la tabla PRODUCTO.

```
--CON COMMIT
SELECT * FROM PRODUCTO;
BEGIN TRANSACTION
UPDATE PRODUCTO SET PRECIO = PRECIO* 1.10
SELECT * FROM PRODUCTO;
COMMIT

--CON ROLLBACK
SELECT * FROM PRODUCTO;
BEGIN TRANSACTION
UPDATE PRODUCTO SET PRECIO = PRECIO* 1.10
SELECT * FROM PRODUCTO;
ROLLBACK

--OPCIONAL: SI CAMBIA EL PRECIO Y DESEA DEJARLE EL VALOR ORIGINAL, UTILICE ESTE COMANDO
UPDATE PRODUCTO SET PRECIO = PRECIO / 1.10
```

### 4. Creación de consultas agrupadas

**Paso 1**: Ejecute con la función DATENAME

```
SELECT YEAR(FECHA_HORA) AS ANIO,
MONTH(FECHA_HORA) AS MES, DATENAME ( mm, FECHA_HORA) AS NOMBRE_MES,
S.SUCURSAL_NOMBRE,
C.CIUDAD_NOMBRE,
SUM(V.UNIDADES) AS UNIDADES, SUM(V.UNIDADES*V.PRECIO) AS USD, COUNT(*) AS CANT_REGISRTOS
FROM VENTAS V
INNER JOIN SUCURSAL S ON S.SUCURSAL_ID = V.SUCURSAL_ID
INNER JOIN CIUDAD C ON C.CIUDAD_ID = S.CIUDAD_ID
--JOIN MARCA M ON M.MARCA_ID= V.MARCA_ID
WHERE YEAR(FECHA_HORA) IN (2016, 2017) AND V.ESTADO = 1 AND CAJA_ID = 1
GROUP BY YEAR(FECHA_HORA), MONTH(FECHA_HORA), DATENAME (mm, FECHA_HORA), S.SUCURSAL_NOMBRE, C.CIUDAD_NOMBRE
HAVING SUM(V.UNIDADES*V.PRECIO) >= 2500 AND SUM(V.UNIDADES*V.PRECIO) <= 3000
ORDER BY YEAR(FECHA_HORA), MONTH(FECHA_HORA), DATENAME (mm, FECHA_HORA), S.SUCURSAL_NOMBRE, C.CIUDAD_NOMBRE
```

100 %

Resultados | Mensajes

| | ANIO | MES | NOMBRE_MES | SUCURSAL_NOMBRE | CIUDAD_NOMBRE | UNIDADES | USD | CANT_REGISRTOS |
|---|---|---|---|---|---|---|---|---|
| 1 | 2016 | 4 | April | EL DORADO | GUAYAQUIL | 5 | 2500 | 1 |
| 2 | 2017 | 3 | March | QUICENTRO | QUITO | 4 | 2800 | 1 |
| 3 | 2017 | 4 | April | MALL DEL SOL | GUAYAQUIL | 4 | 2800 | 2 |
| 4 | 2017 | 10 | October | QUICENTRO | QUITO | 5 | 2700 | 2 |

**Paso 2:** Crear otra consulta para el siguiente resultado

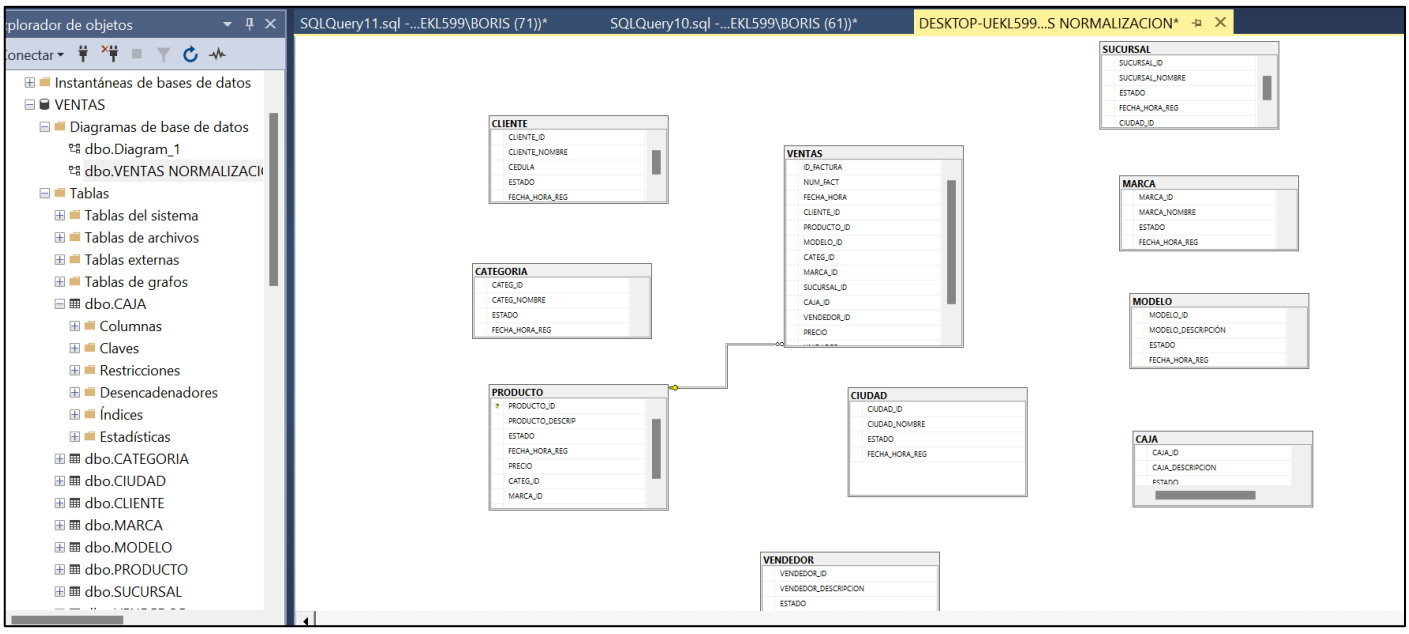
```
SELECT
S.SUCURSAL_NOMBRE,
P.PRODUCTO_DESCRIP,
SUM(V.UNIDADES) AS UNIDADES, SUM(V.UNIDADES*V.PRECIO) AS USD, COUNT(*) AS CANT_REGISRTOS

--BETWEEN O >= =< (PARA BUSCAR DATOS ENTRE FECHAS)
SELECT * FROM VENTAS WHERE FECHA_HORA BETWEEN '2016-06-01' AND '2016-07-15'
SELECT * FROM VENTAS WHERE FECHA_HORA >= '20160601' AND FECHA_HORA <= '20160715'
```

Resultados | Mensajes

| | ID_FACTURA | NUM_FACT | FECHA_HORA | CLIENTE_ID | PRODUCTO_ID | MODELO_ID | CATEG_ID | MARCA_ID | SUCURSAL_ID | CAJA_ID | VENDEDOR_ID | PRECIO |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2538138 | 001-001-4892 | 2016-06-17 13:27:00.000 | 7 | 4 | 1 | 1 | 2 | 5 | 2 | 1 | 500 |
| 2 | 3316565 | 001-005-4823 | 2016-06-14 10:26:00.000 | 2 | 6 | 1 | 1 | 2 | 3 | 2 | 3 | 120 |
| 3 | 4203749 | 002-004-4847 | 2016-07-08 19:49:00.000 | 4 | 8 | 1 | 2 | 2 | 2 | 2 | 1 | 1000 |
| 4 | 2285685 | 002-004-4515 | 2016-06-04 08:54:00.000 | 7 | 6 | 1 | 1 | 2 | 2 | 3 | 2 | 120 |
| 5 | 950049 | 001-001-4658 | 2016-06-25 09:19:00.000 | 2 | 9 | 1 | 2 | 3 | 2 | 3 | 2 | 180 |
| 6 | 3811892 | 001-002-4718 | 2016-06-23 00:57:00.000 | 5 | 10 | 1 | 2 | 3 | 5 | 1 | 3 | 800 |
| 7 | 3343772 | 001-005-4695 | 2016-06-16 18:31:00.000 | 6 | 20 | 1 | 1 | 1 | 1 | 3 | 2 | 100 |
| 8 | 2316148 | 003-003-4705 | 2016-07-03 10:29:00.000 | 4 | 9 | 1 | 2 | 3 | 5 | 3 | 1 | 180 |
| 9 | 4087606 | 001-004-4531 | 2016-07-04 00:37:00.000 | 2 | 20 | 1 | 1 | 1 | 5 | 1 | 3 | 100 |
| 10 | 2856071 | 003-003-4365 | 2016-06-18 21:06:00.000 | 3 | 5 | 1 | 1 | 2 | 4 | 1 | 3 | 900 |

**Paso 3:** Crear 2 sentencias con 2 o 3 niveles de agrupación para realizar consultas con las diversas tablas usando el comando WHERE y buscando por rango de fecha.

Tablas después de la normalización




## 5. Revise en internet el video y en máximo 4 líneas indique una de las funcionalidades de las funciones de texto CAST y CONVERT

La función CAST y CONVERT son importantes para la conversión de tipos de datos en SQL, es decir de un tipo a otro. Aunque CONVERT suele ofrecer características adicionales específicas de SQL Server que pueden ser ventajosas en ciertas situaciones como por ejemplo el formato para conversiones de fecha y hora.

**6. Revise en internet los tipos de datos que maneja SQL SERVER y determine que tipo de datos puede manejar una empresa que importa productos de China desde 0.0001 USD**

Para una empresa importadora, los tipos de datos que podría manejar tomando en cuenta el SQL Server serian tipos de datos numéricos exactos tales como:

- Decimal y numeric: Estos datos son importantes para poder manejar valores monetarios donde se necesita un alto grado de precisión y control.
- money y smallmoney: Estos datos son específicos para valores monetarios, pero tiene una precisión no para valores tan pequeños como 0.0001 USD.
- float y real: Estos datos son adecuados para valores numéricos que no requieren una precisión exacta fija.