DML

(Data Manipulation Lenguaje). Lenguaje proporcionado por el gestor de bases de datos. Permite al usuario realizar tareas de consulta y manipulación de datos.

Se basa en 4 comandos base sobre los que se pueden realizar ciertas acciones:

-**SELECT**: Con dicho comando podemos elegir (seleccionar) columnas de una tabla (FROM) para mostrar un contenido que ya estaba metido en una base de datos. Con SELECT hacemos consultas, y hoy en día se considera mas propio considerarle parte del lenguaje DQL. No entraré en detalles sobre este comando pues lo tengo explicado en los primeros apuntes.

-**INSERT**: Con INSERT podemos añadir filas a una tabla, luego en un ejemplo veremos varios ejemplos de estructura. Las declaraciones con INSERT no pueden tener nombres duplicados en la lista de columnas de destino. Sus valores deben agregarse en el mismo orden que las columnas especificadas. La cantidad de valores debe de coincidir con el numero de columnas especificadas. Y los valores deben ser de un tipo torelable por la columna a la que se insertan. Si omites los nombres de columna, se agregarán de forma ascendente.

-**UPDATE**: Con dicho comando puedes cambiar el valor de los datos en una o más columnas. Con un ejemplo veremos mejor su estructura, pero esta debe contener dicho comando, la cláusula SET y la palabra clave WHERE seguida de una condición. Para actualizar todas las filas de la tabla usa “WHERE true”.

-**DELETE**: Usado para quitar filas de una tabla. Se basa en especificar a que tabla te refieres y a establecer una condición con WHERE que determina que columna quieres borrar. Posteriormente, ejemplos.

<u>INSERT EJEMPLOS:</u>

1. Con valores explícitos: Se especifica a la tabla que va dirigida y a las columnas, de tal manera que los datos introducidos primero en cada sentencia se dirigen a “product” y los otros a “quantity”. De tener más columnas estas quedarían NULL al no especificarse nada.

```
INSERT dataset.Inventory (product, quantity)
VALUES('top load washer', 10),
      ('front load washer', 20),
      ('dryer', 30),
      ('refrigerator', 10),
      ('microwave', 20),
      ('dishwasher', 30),
      ('oven', 5)
```

2. Con values + subconsulta: El concepto básicamente es insertar una fila de una tabla en otra gracias al uso de subconsultas.

```
INSERT dataset.DetailedInventory (product, quantity)
VALUES('countertop microwave',
  (SELECT quantity FROM dataset.DetailedInventory
   WHERE product = 'microwave'))
```

3. Sin nombres de columna: Añadirá dichos valores debajo de lo que ya estaban establecidos en la tabla

```
INSERT dataset.Warehouse VALUES('warehouse #4', 'WA'), ('warehouse #5', 'NY')
```

DELETE EJEMPLOS:

1. Con clausula WHERE: Borras de la tabla especificada los valores que cumplan la condición del WHERE.

```
DELETE dataset.Inventory
WHERE quantity = 0
```

2. Con subconsulta: Borras de la tabla especificada aquellos valores que no están presentes en la tabla especificada en la subconsulta

```
DELETE dataset.Inventory i
WHERE i.product NOT IN (SELECT product from dataset.NewArrivals)
```

UPDATE EJEMPLOS:

1. Con clausula WHERE: Se actualiza Inventory reduciendo quantity en 10 para todos los valores que contienen “washer”

```
UPDATE dataset.Inventory
SET quantity = quantity - 10
WHERE product like '%washer%'
```

2. Campos anidados: El concepto es el mismo, pero con más de una modificación

```
UPDATE dataset.DetailedInventory
SET specifications.color = 'white',
    specifications.warranty = '1 year'
WHERE product like '%washer%'
```

3. Registros repetidos: Se agrega una entrada a un registro repetido en la columna comments para productos que contienen “washer”

```
UPDATE dataset.DetailedInventory
SET comments = ARRAY(
  SELECT comment FROM UNNEST(comments) AS comment
  UNION ALL
  SELECT (CAST('2016-01-01' AS DATE), 'comment1')
)
WHERE product like '%washer%'
```

4. Uniendo tres tablas: Se establece “supply_constrained” en true para los productos de NewArrivals en los que la ubicación del depósito esta en “WA”

```
UPDATE dataset.DetailedInventory
SET supply_constrained = true
FROM dataset.NewArrivals, dataset.Warehouse
WHERE DetailedInventory.product = NewArrivals.product AND
      NewArrivals.warehouse = Warehouse.warehouse AND
      Warehouse.state = 'WA'
```

Restricciones de integridad referencial:

Integridad referencial significa que cuando un registro en una tabla haga referencia a un registro en otra tabla, el registro correspondiente debe existir. Es decir, al establecer una relación entre tablas, esta queda determinada por el paso de la clave primaria de una como clave foránea a la otra.

La expresión completa de una restricción de clave foránea es la siguiente:

```
[CONSTRAINT símbolo] FOREIGN KEY (nombre_columna, ...)
                  REFERENCES nombre_tabla (nombre_columna, ...)
                  [ON DELETE {CASCADE | SET NULL | NO ACTION
                              | RESTRICT}]
                  [ON UPDATE {CASCADE | SET NULL | NO ACTION
                              | RESTRICT}]
```

1. **CONSTRAIT**: Puede colocar restricciones para limitar el tipo de dato que puede ingresarse en una tabla. Dichas restricciones pueden especificarse cuando la tabla se crea por primera vez a través de la instrucción CREATE TABLE, o una vez creada con ALTER TABLE.
2. **ON DELETE**: Especifica que acción sucede con las filas en la tabla que se altera siempre que esas filas tienen una relación referencial y la fila referenciada se elimina de la tabla primaria.
    a. **NO ACTION**: Cuando borras algo en la tabla padre no afecta a las hijas que contengan información derivada de dicha tabla.
    b. **CASCADE**: Sería lo contrario a NO ACTION. Borro los registros de la tabla dependiente cuando se borra el registro de la tabla principal.
    c. **SET NULL**: Establece “NULL” como valor de la clave secundaria que se ve afectada por algo de la principal (modificación o eliminación). Si especifica una acción SET NULL, asegurate de que no has declarado las columnas de la tabla secundaria como NOT NULL. También existe SET DEFAULT, por la cual eres tu quien da un valor por defecto.
    d. **RESTRICT**: Es el comportamiento por defecto, que impide realizar modificaciones que atentan contra la integridad referencial.
3. **ON UPDATE**: Especifica qué acción sucede con las filas de la tabla alteradas cuando esas filas tienen una relación referencial y la fila referenciada se actualiza en la tabla principal.
    a. **NO ACTION**: Cuando modificas algo en la tabla padre no afecta a las hijas que contengan información derivada de dicha tabla.
    b. **CASCADE**: Sería lo contrario a NO ACTION. Modifico los registros de la tabla dependiente cuando se borra el registro de la tabla principal.
    c. **SET NULL**: Establece “NULL” como valor de la clave secundaria que se ve afectada por algo de la principal (modificación o eliminación). Si especifica una acción SET NULL, asegurate de que no has declarado las columnas de la tabla secundaria como NOT NULL. También existe SET DEFAULT, por la cual eres tu quien da un valor por defecto.
    d. **RESTRICT**: Es el comportamiento por defecto, que impide realizar modificaciones que atentan contra la integridad referencial.

4. **MATCH FULL**: No permitirá que una columna de una clave foránea de varias columnas sea NULL a menos que todas las columnas de clave foránea sean nulas. Si todos son nulos, no es necesario que la fila tenga una coincidencia en la tabla referenciada.
5. **MATCH SIMPLE**: Permite que cualquiera de las columnas de clave foránea sea NULL. Si alguno de ellos es NULL, no se requiere que la fila tenga una coincidencia con la tabla referenciada.
6. **MATCH PARTIAL**: si todas las columnas de referencia son nulas, la fila de la tabla de referencia pasa la verificación de restricción. Si al menos una columna de referencia no es nula, entonces la fila pasa la verificación de restricción si y solo si hay una fila de la tabla referenciada que coincida con todas las columnas de referencia no nula. (Preguntar porque no acabo de entender)
7. Otras restricciones de comprobación en Constraints son: NOT DEFERRABLE, INITIALLY DEFERRED, INITIALLY INMEDIATE Y DEFERRABLE.
8. **NOT DEFERRABLE**: La restricción debe ser comprobada después de la ejecución de cada sentencia SQL. Este es el valor predeterminado.
9. **DEFERRABLE**: La verificación del cumplimiento de la restricción puede ser aplazado, pero debe cumplirse antes del fin de la transacción actual.
10. **INITIALLY INMEDIATE**: Desde el comienzo de la transacción la restricción debe ser comprobada después de la ejecución de cada sentencia SQL.
11. **INITIALLY DEFERRED**: como se está al principio de la transacción, la comprobación de la restricción puede ser aplazada hasta más tarde, pero no más tarde que en el final de la actual transacción. O sea, que la restricción puede ser incumplida por alguna sentencia SQL en un punto intermedio de la transacción, pero no al final de la misma.

Restricciones de comprobación

Limitan los posibles valores que se añadan a partir de unas restricciones que exigen el cumplimiento de ciertas comprobaciones. Distinguimos básicamente entre CHECK y UNIQUE.

1. **CHECK**: Se asegura de que todos los valores de la columna cumplan ciertas condiciones. Estas condiciones pueden escribirse de una manera lógica e intuitiva similar a las limitaciones que podemos usar al hacer un if en java. Se puede aplicar más de una restricción CHECK a una misma columna
2. **UNIQUE**: Se asegura de que todos los valores de una columna sean distintos. Podría decirse que la clave primaria ya contiene de por sí esta característica por cuestiones lógicas. Por lo que UNIQUE se utiliza para limitar aquellos parámetros que queramos que no tengan valores duplicados, pero que a su vez NO SEAN claves principales. UNIQUE permite valores NULL (y la clave primaria obviamente no) aunque por lógica, solo podrá haber un NULL en dicha columna.

ISOMORFISMO CURRY HOWARD LAMBEK

Basicamente es la relación directa entre el ámbito de la programación y el ámbito de la lógica. Estableciendo símiles entre los conceptos lógicos y los conceptos de programación.

DDL

(Data Definition Lenguaje) Permite generar las estructuras en las que almacenaremos los datos de nuestra base de datos.

Se basa en 4 comandos que son los siguientes:

-**CREATE**: Permite crear nuevas bases de datos y nuevas tablas.

-**ALTER**: Permite modificar los datos de dichas tablas.

-**DROP**: Permite eliminar bases de datos y tablas.

-**TRUNCATE**: Borrar todas las filas de una tabla sin borrar dicha tabla.

Tipos de datos más recurrentes en columnas:

-**INT**: Para valores enteros

-**FLOAT**: Para valores con decimales

-**DOUBLE**: Similar a FLOAT

-**VARCHAR**: Tipo texto con longitud variable

-**CHAR**: Tipo texto con longitud fija

-**DATE**: Tipo fecha (Similar a LocalDate en java para entendernos)

-**TEXT**: Tipo texto con longitud ilimitada

-**BOOLEAN**: Datos de tipo booleano para establecer true / false

CREATE EJEMPLOS:

1. Sintaxis para crear una base de datos: De esta forma se crea una base de datos con nombre “coches” en el servidor vacía.

   CREATE DATABASE “DBnombre”
   CREATE Coches
   Database name = Coches

2. Sintaxis para crear una tabla: Creas una tabla aporandole un nombre y columnas que permitirán un tipo de valor que tu elijas.

   CREATE TABLE Nombredetabla
   (
   Columna1 tipodato,
   Col2 tipodato …)

3. Sintaxis para crear una tabla a partir de otra ya existente: Creas una tabla nueva con todas las columnas y valores establecidos en la tabla “Coches”

   CREATE TABLE nombreTabla AS
   SELECT * FROM Coches

ALTER EJEMPLOS:

1. Añadir columna nueva:

```
ALTER TABLE table_name ADD(
    column_name datatype);
```

2. Añadir más de una columna nueva: NO TE OLVIDES DE LAS COMAS NUNCA

```
ALTER TABLE table_name ADD(
    column_name1 datatype1,
    column-name2 datatype2,
    column-name3 datatype3);
```

3. Añadir columnas con un valor por defecto:

```
ALTER TABLE table_name ADD(
    column-name1 datatype1 DEFAULT some_value
);
```

4. Modificar una columna existente:

```
ALTER TABLE table_name modify(
    column_name datatype
);
```

5. Renombrar una columna:

```
ALTER TABLE table_name RENAME
    old_column_name TO new_column_name;
```

6. Borrar una columna:

```
ALTER TABLE table_name DROP(
    column_name);
```

DROP EJEMPLOS:

1. Borrar una tabla:

```
DROP TABLE table_name
```

2. Borrar una base de datos:

```
DROP DATABASE Test;
```

TRUNCATE EJEMPLOS:

1. Borrar columnas de una tabla:

```
TRUNCATE TABLE table_name
```