III UNIDAD

Practica de Laboratorio: Creación de Tablas y Manipulación de Registros en SQL

SQL Server es un sistema administrador para bases de datos relacionales basadas en la arquitectura cliente/servidor. Transact-SQL es el lenguaje que emplea para mandar peticiones entre el cliente y el servidor. Es un lenguaje exclusivo de SQL Server, pero basado en el lenguaje SQL estándar, utilizado por casi todos los tipos de bases de datos relacionales que existen. SQL Server otorga a los administradores una herramienta potencialmente robusta, provista de las herramientas suficientes que le permiten mantener un óptimo nivel de seguridad en la utilización de los recursos del sistema y de la base de datos, que en este camino van cogidos de la mano La finalidad de SQL Server 2000 es analizar y administrar datos, dar mayor escalabilidad, disponibilidad y seguridad a las aplicaciones de análisis y los datos empresariales y potenciar las aplicaciones dando una mayor preestabilizadad

SQL (Standar Query Lenguaje) es un lenguaje estandarizado de base de datos, el cual nos permite realizar tablas y obtener datos de ella de manera muy sencilla. Para exponer mas claramente los conceptos se realizaran ejemplo sobre relaciones que se crearan aquí para entender mejor como funciona SQL.

Cuando aquí nos refiramos a relación estamos hablando mas concretamente a la tabla de datos en si, y sus atributos serán los campos de la tabla.

Practica 3.1 Cree una Base de Datos llamada Practica, y cree la siguiente tabla (relación) la llamaremos persona y sus atributos (campos) son nombre, apellido Y DNI.

PERSONA	NOMBRE	APELLIDO	DNI
1	MARTIN	MARQUESI	26125988
2	PABLO	MARQUESI	25485699
3	ROBERTO	SANCHEZ	20566401
4	ESTEFANIA	GUISSINI	27128064
5	RUBEN	ALEGRATO	24238975
6	SANDRA	BRITTE	25483669
7	MELISA	ARDUL	27456224
8	SOLEDAD	MICHELLI	29889656

9	BETANIA	MUSACHEGUI	27128765
10	JUAN	SERRAT	28978845

Respuesta 3.1

```
Create database Practica
Use Practica

Create table Persona
(nombre varchar(80),
  apellido varchar(80),
  dni int)

INSERT INTO Persona (nombre,apellido,dni) VALUES ('MARTIN','MARQUESI',26125988)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('PABLO','MARQUESI',25485699)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('ROBERTO','SANCHEZ',20566401)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('ESTEFANIA','GUISSINI',27128064)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('RUBEN','ALEGRATO',24238975)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('SANDRA','BRITTE',25483669)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('MELISA','ARDUL',27456224)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('SOLEDAD','MICHELLI',29889656)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('BETANIA','MUSACHEGUI',27128765)
INSERT INTO Persona (nombre,apellido,dni) VALUES ('JUAN','SERRAT',28978845)

Select * from Persona
```

SQL es un lenguaje que consta de varias partes

- Lenguaje de definición de datos (DDL): Proporciona ordenes para definir esquemas de relación, eliminar relaciones, crear índices y modificar esquemas de relación.
- Lenguaje de manipulación de datos interactivos (DML): incluye un leguaje de consultas que permite rescatar datos de las relaciones. También incluye ordenes para insertar, suprimir y modificar tuplas.
- Lenguaje de manipulación de datos inmerso (DML): La forma inmersa de SQL esta diseñada para usar dentro de los lenguajes de programación de lenguaje general.
- Definición de vistas (DDL): incluye órdenes para definir vistas.

Estructura básica

La estructura básica de una expresión para consulta SQL consta de tres cláusulas:

- SELECT
- FROM
- WHERE

La cláusula SELECT se usa para listar los atributos que se desean en el resultado de una consulta.

La cláusula FROM lista las relaciones que se van a examinar en la evaluación de la expresión

La cláusula WHERE costa de un predicado que implica atributos de las relaciones que aparecen en la cláusula FROM.

Una consulta básica en SQL tiene la forma:

SELECT A1,A2,...,An
FROM r1,r2,...,rn
WHERE P
Donde Ai = atributo (Campo de la tabla)
ri = relación (Tabla)
P = predicado (condición)

Nota: *Todas las consultas analícelas y córralas en el Analizador de Consultas de SQL.*

Practica 3.2: Seleccione todos los nombres de las personas que tengan el apellido MARQUESI de la tabla persona

Respuesta 3.2:
```
SELECT nombre
FROM persona
WHERE apellido = 'MARQUESI'
```

ANSWER	NOMBRE
1	MARTIN
2	PABLO

El resultado de una consulta es por supuesto otra relación. Si se omite la cláusula WHERE, el predicado P es verdadero. La lista A1, A2,..., An puede sustituirse por un asterisco (*) para seleccionar todos los atributos de todas las relaciones que aparecen en la cláusula FROM, aunque no es conveniente elegir esta ultima opción salvo que sea necesario pues desperdiciamos mucho tiempo en obtenerlo

Alias

Es posible renombrar los atributos y las relaciones, a veces por conveniencia y otras veces por ser necesario, para esto usamos la cláusula AS como en el siguiente ejemplo.

Practica 3.3: Ejecute la siguiente consulta

```
SELECT nombre AS [PRIMER NOMBRE]
FROM  persona
WHERE apellido = 'MARQUESI'
```

ANSWER	PRIMER NOMBRE
1	MARTIN
2	PABLO

Los corchetes los usamos cuando usamos espacios en blancos o el carácter (–) en el nombre de atributo o alias.

Usar alias en los atributos nos permite cambiar el nombre de los atributos de la respuesta a la consulta.

Cuando asociamos un alias con una relación decimos que creamos una variable de tupla. Estas variables de tuplas se definen en la cláusula FROM después del nombre de la relación.

En las consultas que contienen sub consultas, se aplica una regla de ámbito a las variables de tupla. En una sub consulta está permitido usar solo variables de tupla definidas en la misma sub consulta o en cualquier consulta que tenga la sub consulta.

Predicados y conectores

Los conectores lógicos en SQL son:

- AND
- OR
- NOT

La lógica de estos conectores es igual que en cualquier lenguaje de programación y sirven para unir predicados.

Las operaciones aritméticas en SQL son:

- + (Suma)
- - (Resta)
- * (Multiplicación)
- / (División)

También incluye el operador de comparación BETWEEN, que se utiliza para valores comprendidos

Practica 3.4: Encontrar todos los nombres y DNI de las personas cuyos DNI sea mayor que 26 millones y menor a 28 millones.

Respuesta 3.4:
```
SELECT nombre, dni
FROM persona
WHERE dni BETWEEN 26000000 and 28000000
```

ANSWER	NOMBRE	DNI
1	MARTIN	26125988
2	ESTEFANIA	27128064
3	MELISA	27456224
4	BETANIA	27128765

Análogamente podemos usar el operador de comparación NOT BETWEEN. SQL también incluye un operador de selección para comparaciones de cadena de caracteres. Los modelos se describen usando los caracteres especiales:

- El carácter (%) es igual a cualquier sub cadena
- El operador (_) es igual a cualquier carácter

Estos modelos se expresan usando el operador de comparación LIKE. Un error muy frecuente es tratar de utilizar los modelos mediante el operador de igualdad (=) lo cual es un error de sintaxis.

Practica 3.5: Encontrar los nombres que comiencen con la letra p o el nombre tenga exactamente 6 caracteres de la relación persona.

Respuesta 3.5:
```
SELECT nombre
FROM persona
WHERE (nombre LIKE 'P% ') OR (nombre LIKE '_____ ')
```

ANSWER	NOMBRE
1	MARTIN
2	PABLO
3	MELISA
4	SANDRA

Análogamente podemos buscar desigualdades usando el operador de comparación NOT LIKE.

Tuplas duplicadas

Los lenguajes de consulta formales se basan en la noción matemática de relación como un conjunto. Por ello nunca aparecen tuplas duplicadas en las relaciones. En la práctica la eliminación de duplicados lleva bastante tiempo. Por lo tanto, SQL permite duplicados en las relaciones. Entonces, en las consultas se listarán todas las tuplas inclusive las repetidas.

En aquellos casos en los que queremos forzar la eliminación de duplicados insertamos la palabra clave DISTINCT después de la cláusula SELECT

Practica 3.6: Listar todos los apellidos no repetidos de la relación persona.

Respuesta 3.6

```
SELECT DISTINCT apellido
 FROM persona
```

ASWER	APELLIDO
1	MARQUESI
2	SANCHEZ
3	GUISSINI
4	ALEGRATO
5	BRITTE
6	ARDUL
7	MICHELLI
8	MUSACHEGUI
9	SERRAT

Si observamos la tabla original de la relación persona veremos que el apellido MARQUESI aparecía dos veces, pero debido al uso de DISTINCT en la consulta la relación respuesta solo lista un solo MARQUESI.

Operaciones de conjunto.

SQL incluye las operaciones de conjuntos UNION, INTERSECT, MINUS, que operan sobre relaciones y corresponden a las operaciones del álgebra unión, intersección y resta de conjuntos respectivamente. Para realizar esta operación de conjuntos debemos tener sumo cuidado que las relaciones tengan las mismas estructuras.

Practica 3.7: Incorporemos ahora una nueva tabla, llamada jugadores que representa las personas que juegan al fútbol, sus atributos serán DNI, puesto y nro_camiseta. Supongamos que esta nueva tabla está conformada de la siguiente manera

JUGADORES	DNI	PUESTO	NRO_CAMISETA
1	26125988	DELANTERO	9
2	25485699	MEDIO	5
3	28978845	ARQUERO	1
4	29789854	DEFENSOR	3

Respuesta 3.7

```
Create table jugadores
(dni int,
Puesto varchar(15),
nro_camiseta int)

INSERT INTO jugadores (dni,puesto,nro_camiseta) VALUES (26125988,'DELANTERO',9)
INSERT INTO jugadores (dni,puesto,nro_camiseta) VALUES (25485699,'MEDIO',5)
INSERT INTO jugadores (dni,puesto,nro_camiseta) VALUES (28978845,'ARQUERO',1)
INSERT INTO jugadores (dni,puesto,nro_camiseta) VALUES (29789854,'DEFENSOR',3)
```

Practica 3.8: Obtener todos los nombres y apellidos de la relación persona cuyos apellidos sean Marquesi o Serrat.

Respuesta 3.8

```
SELECT nombre, apellido
FROM PERSONA
WHERE apellido = 'MARQUESI'
UNION
SELECT nombre, apellido
FROM PERSONA
WHERE apellido = 'SERRAT'
```

ANSWER	NOMBRE	APELLIDO
1	MARTIN	MARQUESI
2	PABLO	MARQUESI
3	JUAN	SERRAT

Por omisión, la operación de unión elimina las tuplas duplicadas. Para retener duplicados se debe escribir UNION ALL en lugar de UNION.

Practica 3.9: Obtener todos los DNI de los que juegan al fútbol y, además, que estén en la lista de la relación persona.

Nota: Utilizar EXISTS y NOT EXISTS para buscar intersecciones y diferencias

Las sub consultas presentadas con EXISTS y NOT EXISTS se pueden usar en dos operaciones de la teoría de conjuntos: intersección y diferencia. La intersección de dos conjuntos contiene los elementos que pertenecen a los dos conjuntos originales. La diferencia contiene los elementos que pertenecen sólo al primero de los dos conjuntos.

Respuesta 3.9

```
SELECT DISTINCT dni
FROM persona
WHERE EXISTS
  (SELECT *
   FROM jugadores
   WHERE persona.dni = jugadores.dni)
```

ANSWER	DNI
1	26125988
2	25485699
3	28978845

Por supuesto, también podría haberse escrito como una simple combinación.

```
SELECT DISTINCT persona.dni
FROM persona INNER JOIN jugadores
ON persona.dni = jugadores.dni
```

Pertenencia a un conjunto

El conector IN prueba si se es miembro de un conjunto, donde el conjunto es una colección de valores producidos en lo general por una cláusula SELECT. Análogamente el conector NOT IN prueba la no pertenencia al conjunto

Practica 3.10: Encontrar los nombres de las personas que juegan al fútbol y, además, se encuentran en la relación persona.

Respuesta 3.10:

```
SELECT nombre, apellido
 FROM persona  WHERE dni
IN
(SELECT dni FROM jugadores)
```

ANSWER	NOMBRE	APELLIDO
1	MARTIN	MARQUESI
2	PABLO	MARQUESI
3	JUAN	SERRAT

Comparación de conjuntos

En conjuntos la frase << mayor que algún >> se representa en SQL por (>SOME), también podría entenderse esto como << mayor que el menor de >>, su sintaxis es igual que la del conector IN. SQL también permite las comparaciones (>SOME),(=SOME) (>=SOME), (<=SOME) y (<>SOME).

También existe la construcción (>ALL), que corresponde a la frase << mayor que todos >>. Al igual que el operador SOME, puede escribirse (>ALL),(=ALL) (>=ALL), (<=ALL) y (<>ALL).

En ocasiones podríamos querer comparar conjuntos para determinar si un conjunto contiene los miembros de algún otro conjunto. Tales comparaciones se hacen usando las construcciones CONTAINS y NOT CONTAINS

Pruebas para relaciones vacías

La construcción EXISTS devuelve el valor TRUE si la subconsulta del argumento no esta vacía, y la construcción NOT EXISTS devuelve TRUE si la consulta es vacía.

Practica 3.11: Encontrar todos los nombre y apellidos de la relación persona si es que en la relación jugadores existe un jugador con el número de dni 27128055.

Respuesta 3.11

```
SELECT nombre, apellido
FROM persona
WHERE EXISTS
   (SELECT dni
    FROM jugadores
    WHERE dni = 27128055 )
```

ANSWER	NOMBRE	APELLIDO

Como el dni = 27128055 no existe en la relación jugadores, la condición es FALSE y por lo tanto la respuesta es vacía, en la consulta anterior cambie el dni 27128055 por 26125988, ¿Cuál es la respuesta y por que?

Ordenación de la presentación de tuplas

SQL ofrece al usuario cierto control sobre el orden en el que se va a presentar las tuplas en una relación. La cláusula ORDER BY hace que las tupla en el resultado dé una consulta en un orden específico.

Por omisión SQL lista los elementos en orden ascendente. Para especificar el tipo de ordenación, podemos especificar DESC para orden descendente o ASC para orden ascendente.

También es posible ordenar los resultados por más de una atributo

Practica 3.12: Encontrar todos los nombres y apellido de la relación persona y ordenar los resultados por apellido y nombre en forma descendente.

Respuesta 3.12

```
SELECT apellido, nombre
FROM persona
ORDER BY apellido DESC, nombre DESC
```

ANSWER	APELLIDO	NOMBRE
1	SERRAT	JUAN
2	SANCHEZ	ROBERTO
3	MUSACHEGUI	BETANIA
4	MICHELLI	SOLEDAD
5	MARQUESI	PABLO
6	MARQUESI	MARTIN
7	GUISSINI	ESTEFANIA
8	BRITTE	SANDRA
9	ARDUL	MELISA
10	ALEGRATO	RUBEN

Funciones de agregación

SQL ofrece la posibilidad de calcular funciones en grupos de tuplas usando la cláusula GROUP BY, también incluye funciones para calcular

- Promedios AVG
- Mínimo MIN
- Máximo MAX
- Total SUM
- Contar COUNT

Practica 3.13: Para los próximos ejemplos incorporar una nueva tabla llamada PRO que representara los jugadores profesionales de fútbol, sus atributos serán dni, años_pro, club, valor_actual. Y los valores son los siguientes:

PRO	DNI	AÑOS_PRO	CLUB	VALOR_ACTUAL
1	26125988	5	ALL BOYS	1000
2	25485699	2	ALL BOYS	2500
3	27126045	3	LANUS	12000
4	26958644	4	LANUS	6500

5	29120791	1	LANUS	450

-- Respuesta 3.13

```
Create table pro
(dni int,
años_pro int,
club varchar(20),
valor_actual int)

INSERT INTO pro (dni,años_pro,club,valor_actual) VALUES (26125988,5,'ALL BOYS',1000)
INSERT INTO pro (dni,años_pro,club,valor_actual) VALUES (25485699,2,'ALL BOYS',2500)
INSERT INTO pro (dni,años_pro,club,valor_actual) VALUES (27126045,3,'LANUS',12000)
INSERT INTO pro (dni,años_pro,club,valor_actual) VALUES (26958644,4,'LANUS',6500)
INSERT INTO pro (dni,años_pro,club,valor_actual) VALUES (29120791,1,'LANUS',450)
```

Practica 3.14: Determinar el valor total en jugadores, así como la cantidad de jugadores de cada club en la relación pro.

Respuesta 3.14

```
SELECT club, SUM(valor_actual) AS VALOR_TOTAL,
COUNT(club) AS NRO_JUGADORES
FROM pro
GROUP BY CLUB
```

ANSWER	CLUB	VALOR_TOTAL	NRO_JUGADORES
1	ALL BOYS	3.500,00	2
2	LANUS	18.950,00	3

Practica 3.15: Determinar por cada club cual es el valor actual del jugador más caro de la relación pro.

Respuesta 3.15

```
SELECT club, MAX(valor_actual) AS JUG_MAS_CARO
FROM pro
GROUP BY CLUB
```

ANSWER	CLUB	JUG_MAS_CARO
1	ALL BOYS	2500
2	LANUS	12000

Hay ocasiones en la que los duplicados deben eliminarse antes de calcular una agregación. Cuando queremos eliminar los duplicados del calculo usamos la palabra clave DISTINCT antepuesto al atributo de agregación que queremos calcular, como por ejemplo COUNT (DISTINCT club).

Hay ocasiones en las que es útil declara condiciones que se aplican a los grupos mas que a las tuplas. Para esto usamos la cláusula HAVING de SQL.

Práctica 3.16: Determinar por cada club cual es el valor actual del jugador más caro, pero con la condición de que este sea mayor a 10000 de la relación pro.

Respuesta 3.16

```
SELECT club, MAX(valor_actual) AS JUG_MAS_CARO
FROM pro
GROUP BY CLUB
HAVING MAX(valor_actual) > 10000
```

ANSWER	CLUB	JUG_MAS_CARO
1	LANUS	12000

Si en la misma consulta aparece una cláusula WHERE y una cláusula HAVING, primero se aplica el predicado de la cláusula WHERE, las tuplas que satisfacen el predicado WHERE son colocadas en grupos por la cláusula GROUP BY. Después se aplica la cláusula HAVING a cada grupo.

Modificación de la base de datos

Eliminación

Una solicitud de eliminación se expresa casi de igual forma que una consulta. Podemos suprimir solamente tuplas completas, no podemos suprimir valores solo de atributos.

```
DELETE FROM r
WHERE P
```

Donde P presenta un predicado y r representa una relación. Las tuplas t en r para las cuales P(t) es verdadero, son eliminadas de r.

Si omitimos la cláusula WHERE se eliminan todas las tuplas de la relación r (un buen sistema debería buscar confirmación del usuario antes de ejecutar una acción tan devastadora)

Práctica 3.17: Eliminar todas las tuplas de la relación persona en donde apellido sea igual a "BRITTE".

Respuesta 3.17

```
DELETE FROM persona
WHERE apellido = 'BRITTE'
```

Deleted	NOMBRE	APELLIDO	DNI
1	SANDRA	BRITTE	25483669

Inserción

Para insertar datos en una relación, especificamos una tupla que se va a insertar o escribimos una consulta cuyo resultado es un conjunto de tuplas que se van a insertar. La inserción de tuplas la realizamos mediante las sentencias

```
INSERT INTO r1
VALUES (v1,v2,...,v)
```

Práctica 3.18: Insertar una tupla con los mismos valores de la tupla eliminada en el ejemplo anterior en la relación persona.

Respuesta 3.18

```
INSERT INTO Persona  (Nombre, Apellido, DNI)
VALUES     ('SANDRA', 'BRITTE', '25483669')
```

inserted	NOMBRE	APELLIDO	DNI
1	SANDRA	BRITTE	25483669

En este ejemplo, los valores se especifican en el orden en que se listan los atributos correspondientes en el esquema de relación. Para poder ingresar los datos en un orden diferente podríamos haber escrito

```
INSERT INTO Persona (DNI, NOMBRE, APELLIDO)
VALUES (25483669,'SANDRA','BRITTE')
```

Actualizaciones

En ciertas ocasiones podemos desear cambiar los valores de una tupla sin cambiar todos los valores en dicha tupla. Para este propósito usamos la sentencia

```
UPDATE r1
SET A1 = V1, A2 = V2,...,An = Vn
WHERE P
```

Donde r1 es la relación Ai el atributo a modificar Vi el valor que se le asignara a Ai y P es el predicado.

Práctica 3.19: En la relación jugadores actualizar la posición de los jugadores que posean la camiseta número 5 y asignarles la camiseta numero 7.

Respuesta 3.19

```
UPDATE jugadores
SET nro_camiseta = 7
WHERE nro_camiseta = 5
```

updated	DNI	PUESTO	NRO_CAMISETA
1	25485699	MEDIO	5

Valores nulos

Es posible que para las tuplas insertadas se den valores únicamente a algunos atributos del esquema. El resto de los atributos son asignados a valores nulos representados por NULL. Para esto colocamos la palabra reservada NULL como valor del atributo.

Práctica 3.20: Insertar en la relación jugadores un jugador con dni = 26356312, puesto = defensor, y al cual aun no le han asignado un nro camiseta.

Respuesta 3.20

```
INSERT INTO jugadores
VALUES (26356312,'DEFENSOR', NULL)
```

inserted	DNI	PUESTO	NRO_CAMISETA
1	26356312	DEFENSOR	

Definición de datos
Creación

Una relación en SQL se define usando la orden

CREATE TABLE r(A1 D1, A2 D3,...,An Dn)

Donde r es el nombre de la relación, cada Ai es el nombre de un atributo del esquema de la relación r y Di es el tipo de dato de Ai. Una relación recién creada esta vacía. La orden INSERT puede usarse para cargar la relación

Práctica 3.21: Crear la relación lesionado con los atributos nombre, apellido ambos de tipo char y tiempo_inhabilit de tipo entero.

Respuesta 3.21

```
CREATE TABLE lesionado (NOMBRE CHAR(20), APELLIDO CHAR(20),TIEMPO_INHABILT INTEGER)
```

lesionado	NOMBRE	APELLIDO	TIEMPO_INHABILT

Eliminación

Para eliminar una relación usamos la orden DROP TABLE r, esta orden elimina toda la información sobre la relación sacada de la base de datos, esta orden es mas fuerte que DELET FROM r ya que esta ultima elimina todas las tuplas pero no destruye la relación, mientras que la primera si.

Práctica 3.22: Eliminar la relación persona.

Respuesta 3.22

```
DROP TABLE persona
```

Actualización

La orden ALTER TABLE se usa para añadir atributos a una relación existente. A todas las tuplas en la relación se les asigna NULL como valor de atributo. La sintaxis de ALTER TABLE es la siguiente:

ALTER TABLE r1 ADD A1 D1

Práctica 3.23: Agregar los atributos de tipo char nombre y apellido a la relación jugadores.

Respuesta 3.23

```
ALTER TABLE jugadores ADD NOMBRE CHAR(20)
ALTER TABLE jugadores ADD APELLIDO CHAR(20)
```

- ***No te quedes con los síntomas de los problemas: intenta averiguar las causas.***
- ***No te apresures a sacar conclusiones: busca la verdad oculta que todo encierra.***
- ***Si diriges gente sabe que la dirección es el arte de conseguir resultados a través de los demás.***
- ***El mejor organizador es el ejecutante.***
    - ***El usuario es el único que sabe lo que quiere.***
- ***La documentación es una importante ayuda para el análisis, el diseño, el control del proyecto, completar lagunas y para la comunicación.***
- ***La documentación es un puente entre el presente y el pasado, y también entre el presente y el futuro.***
- ***Sin documentación, el mantenimiento se torna casi imposible.***
- ***Los problemas de organización son problemas abiertos y no cerrados, no te quedes con la primera solución: creatividad.***