Programación web con PHP

Lenguaje SQL: Consultas

Temario

Obtener datos de las tablas

Contar filas

Para contar las filas en una tabla utilizamos el comando COUNT(*)

```
SELECT COUNT(*) FROM estudiantes;
```

Obtener todos los datos de una tabla

Para obtener todas las columnas de una tabla utilizamos SELECT seguido de un asterisco, FROM y el nombre de la tabla.

```
SELECT * FROM estudiantes;
```

Limite

La clausula LIMIT permite limitar el numero de registros que mostrara la sentencia SELECT.

```
SELECT * FROM estudiantes LIMIT 3;
```

Seleccionar columnas

Para seleccionar solo algunas columnas de una tabla las listamos después del SELECT separadas por comas.

```
SELECT nombre, apellido FROM estudiantes;
```

Ordenar

Para ordenar los resultados devueltos por la clausula SELECT utilizamos ORDER BY seguido de la columna por la cual se desea ordenar. Pueden ordenarse columnas de cualquier tipo.

```
SELECT * FROM estudiantes ORDER BY edad;
```

Al ordenar por una columna varchar se ordena de forma alfabético.

```
SELECT * FROM estudiantes ORDER BY apellido;
```

Para revertir el orden utilizamos DESC

```
SELECT * FROM estudiantes ORDER BY edad DESC;
```

Si se quiere ordenar por los valores de mas de una columna separamos las columnas por comas

```
SELECT * FROM estudiantes ORDER BY apellido,edad;
```

Esto significa que se ordenaran los registros por apellido y en el caso que el apellido coincida se ordenaran por edad.

Filtrar

En oraciones en necesario solo mostrar filas de la tabla que cumplan cierta condición, para hacerlo utilizamos el comando WHERE seguido por la condición a cumplir.

```
SELECT * FROM estudiantes WHERE  edad >25;

SELECT * FROM estudiantes WHERE  nombre =”Pedro”;
```

Comparación

Cuando necesitamos comparar un carácter o una parte de una cadena utilizamos LIKE y NOT LIKE. Para hacerlo utilizamos patrones, los patrones pueden ser muy complejos y comprenden una gran cantidad de símbolos, los mas utilizados son:

%: Cualquier cantidad de cualquier letra

_: Una letra cualquiera

Si quisiéramos mostrar todos los nombres que comiencen con p:

```
SELECT * FROM estudiantes WHERE  nombre LIKE 'p%';
```

Esto seria todas las cadenas que comiencen con p y tengas cualquier cantidad de cualquier letra después

Similar caso seria encontrar nombres que terminen con a

```
SELECT * FROM estudiantes WHERE  nombre LIKE '%a';
```

Si quisiéramos encontrar los nombres que contengan una letra, por ejemplo k, deberíamos usar en sigo % en ambos lados.

```
SELECT * FROM estudiantes WHERE  nombre LIKE '%k%';
```

Si lo que queremos es encontrar los nombres con una cantidad especifica de letras utilizamos _ para denotar cuantos caracteres necesitamos que tenga la cadena.

```
SELECT * FROM estudiantes WHERE  nombre LIKE '_____';
```

En este caso traerá todas los nombres con exactamente 5 letras.

Agrupar

Para mostrar los registros de forma agrupada utilizamos la sentencia GROUP BY seguido de la columna por la cual se quiere agrupar.

```
SELECT apellido, count(apellido) FROM estudiantes GROUP BY apellido
```

Esta sentencia agrupará los registros cuyos apellidos sean iguales, estamos mostrando el apellido y la función count() aplicada a el campo apellido, esto nos mostrará cuantos registros hay en cada grupo, por lo tanto cuantos alumnos comparten ese apellido.

Si en vez de la función count utilizamos la función sum() esta nos devolverá la suma de el campo de todos los registros de ese grupo.

```
SELECT apellido, sum(cuota) FROM estudiantes GROUP BY apellido
```

Transacciones

Una transacción es un conjunto de instrucciones SQL que se ejecutan como una unidad, todas tienen que ejecutarse, de manera contraria no tienen efecto. Solo cuando todas las instrucciones hayan sido realizadas con éxito los cambios serán guardados en la base de datos. Si una de ellas contiene un error ningún cambio se guardara.

El ejemplo mas sencillo de una transacción es una trasferencia bancaria entre dos cuentas, debemos restar el saldo en una cuenta y añadirlo en otra. Si la segunda instrucción no pudo completarse el dinero se habrá perdido ya que no estará en ninguna de las dos cuentas.

Para poder implementar transacciones en MySQL debemos utilizar el motor InnoDB.

Una transacción tiene dos finales posibles:

COMMIT: todas las instrucciones se han ejecutada y se guardan los datos.

ROLLBACK: se ha producido un error y no se guardaran los datos.

MySQL trae por defecto activado el modo autocommit por lo tanto cada instrucción se confirma automáticamente.

Podemos verlo ejecutando la instrucción :

```
SHOW GLOBAL VARIABLES
```

y podemos desactivarlo con la instrucción:

```
SET AUTOCOMMIT = 0
```

Para indicar que comienza una transacción se utiliza el comando START TRANSACTION, luego se listan todas las instrucciones requeridas y al finalizar se utiliza el comando COMMIT.

```
START TRANSACTION;
SELECT *
FROM estudiantes
WHERE codigo =11;
INSERT INTO estudiantes( codigo, nombre, apellido )
VALUES ( 11, 'Pablo', 'Gomez' );
COMMIT;
```

Consultas anidadas

Una consulta anidada o subconsulta es un comando SELECT dentro de otro comando.

```
SELECT * FROM materias WHERE profesorId = (SELECT profesorId FROM
profesores WHERE nombre=”Pablo” && apellido=”Sosa”);
```

También podemos utilizarlo para otros comandos como DELETE

```
DELETE FROM materias
WHERE profesorId =
(SELECT profesorId FROM profesores WHERE nombre=”Pablo” &&
apellido=”Sosa”)
```

Si la subconsulta puede devolver mas de un registro entonces utilizamos el comando IN

```
SELECT * FROM materias WHERE profesorId IN (SELECT profesorId FROM
profesores WHERE nombre LIKE “P%”);
```

Esta subconsulta puede contener cualquier palabra clave, clausula o comando de un SELECT pero el comando exterior puede ser SELECT, INSERT, UPDATE o DELETE.

Uniones

Una unión o join es una combinación de dos o mas tablas de una base de datos relacional. Con ella podemos leer datos de diferentes tablas en una única consulta.

INNER JOIN

Este tipo de unión une los resultados que coincidan en ambas tablas, no devolverá un registro de una tabla que no coincida con otro de otra.

```
SELECT *
FROM estudiantes
INNER JOIN carreras
WHERE estudiantes.carreraId=carreras.carreraId
```

En este caso devolverá todos los datos de ambas tablas pero solo en el caso de que un estudiante tenga una carrera asociada, mostrará por ejemplo nombre, apellido y edad del estudiante y nombre, año y grado de la carrera, todo en un mismo registro.

Pero solo traerá aquellos alumnos que tengan asociada una carrera y las carreras que tienen asociado algún alumno

LEFT JOIN

Devuelve los registros de la primera tabla aunque no coincida con ninguno de la otra, si esto sucede simplemente devolverá la segunda tabla con todos los campos nulos.

```
SELECT *
FROM estudiantes
LEFT JOIN carreras
WHERE estudiantes.carreraId=carreras.carreraId
```

A diferencia de INNER JOIN en este caso traerá todos los registros de alumnos sin importar si tienen una carrera asociada, en el caso de no tener ninguna todos los campos solicitados de la tabla carreras estarán nulos.

Las datos de las carreras que no tienen alumnos asociados no se mostrarán.

RIGHT JOIN

Similar a LEFT JOIN excepto que devolverá los datos de la segunda tabla en vez de la primera.

```
SELECT *
FROM estudiantes
```

```
RIGHT JOIN carreras
WHERE estudiantes.carreraId=carreras.carreraId
```

En este caso se traerán todos los registros de la carrera sin importar si tienen alumnos asociados, de no tenerlos los campos de la tabla alumnos serán nulos.

Los datos de los alumnos que no tienen una carrera asociada no se mostrarán.