Bases de Datos I
14/12/2020
Apuntes de la clase 14/12
IS-501
José Inestroza

Definición de Base de datos: Las bases de datos son colecciones de datos relacionados, no necesariamente debe vivir en un sistema relacionado. Las bases de datos se manejan por sistemas de información llamado sistema de gestión de bases de datos, por ejemplo **MySQL, Oracle, SQL Server, etc.** Permite la protección de información, y ayuda a los administradores de las bases de datos para tener control de la base de datos (DBA Database Administrator).

Comandos Dentro de MariaDB [(none)]
1. **Mostrar base de datos :** show databases.
2. **Usar base de datos :** use nombreBaseDatos.
3. **Mostrar tablas de una base de datos :** show tables.
4. **Pedir info a la base de datos :** select 1; (muestra el titulo y el valor).
    —> select 1 as number, “cadena” as string;
    —> select now( );

La construcción de la base de datos: Es el proceso de almacenar los datos en algunos medio de almacenamiento controlado por el DBMS.

Consulta y transacción: Una consulta normalmente hace que se recuperen algunos datos; una transacción puede hacer que se lean algunos datos y que se escriban algunos datos en la base de datos.

Las principales características del enfoque de base de datos frente a El enfoque de procesamiento de archivos es el siguiente:
■ Naturaleza autodescriptiva de un sistema de base de datos
■ Aislamiento entre programas y datos, y abstracción de datos
■ Soporte de múltiples vistas de los datos
■ Intercambio de datos y procesamiento de transacciones multiusuario
Describimos cada una de estas características en una sección separada.

Vista: Una vista puede ser un subconjunto de la base de datos o puede contienen datos virtuales que se derivan de los archivos de la base de datos pero que no se almacenan explícitamente.

Ventajas de utilizar el enfoque DBMS
- Controlar la redundancia
- Restringir el acceso no autorizado
- Proporcionar almacenamiento persistente para objetos de programa

- Proporcionar estructuras de almacenamiento y búsqueda Técnicas para un procesamiento de consultas eficiente
- Proporcionar respaldo y recuperación
- Proporcionar múltiples interfaces de usuario
- Representar relaciones complejas entre datos
- Hacer cumplir las restricciones de integridad
- Permitir inferencias y acciones
- Usar reglas y activadores

Cuándo no usar un DBMS

A pesar de las ventajas de usar un DBMS, hay algunas situaciones en las que un
El DBMS puede implicar costos generales innecesarios en los que no se incurriría en
procesamiento de archivos tradicional. Los costos generales de usar un DBMS se deben a la siguiendo:

- Alta inversión inicial en hardware, software y capacitación.
- La generalidad que proporciona un DBMS para definir y procesar datos.
- Gastos generales para proporcionar funciones de seguridad, control de simultaneidad, recuperación e integridad.

Por lo tanto, puede ser más conveniente desarrollar aplicaciones de bases de datos personalizadas en las siguientes circunstancias:

- Aplicaciones de base de datos simples y bien definidas que no se espera que cambien en absoluto.
- Requisitos estrictos en tiempo real para algunos programas de aplicación que pueden no se cumple debido a la sobrecarga de DBMS.
- Sistemas integrados con capacidad de almacenamiento limitada, donde un propósito general DBMS no encajaría.
- Sin acceso de múltiples usuarios a los datos.

Ver los usuarios creados en la base de datos

Mostrar Bases: **$ SHOW DATABASES;**
Usar la base mysql: **$ USE mysql;**
Mostrar Tablas: **$ SHOW TABLES;**
Ver registros o campos de la tabla usuario: **$ DESCRIBE user**
Ver usuarios que hay en la tabla y desde que dominio se puede conectar a dicho usuario: **$ SELECT Host, User FROM user**
Mostrar la contraseña de los usuarios: **$ SELECT Host, User, Password FROM user**

Comando SELECT
Lo que hace SELECT es pedir un campo de la base de datos al cual a ese campo se le puede poner una etiqueta, luego se usa la palabra clave FROM para saber desde donde se pide dicha consulta, se pueden pedir consultas de tablas indicando solo la tabla o indicando a la base de datos (distinta a la que esta accedido actualmente) y luego a la tabla, Ejemplo: **$ SELECT count(*) FROM**

Generar flotantes aleatorios →
SELECT rand(); se obtiene un valor aleatorio flotante.

Generar un entero a partir de un flotante aleatorio →
SELECT CAST (rand() AS INT) ; de esta forma se obtiene siempre cero dado que los flotantes están entre 0 y 1, sirve solo para castear.
SELECT CAST (rand() * (max – min) + min AS INT) ;

Concatenar →

```
SELECT CONCAT( “Hola” , “ “, “Mundo”) AS “Cadena de Hola Mundo”;
```

Generación de SQL mediante SQL →

```
SELECT CONCAT(“SELECT ” , “Campo1,”,
“Campo2,” ,”Campo3,”,”FROM DATABASE.TABLENAME”) as “Query
de SQL”;
```

El modelo entidad relación: Debe tener un diagrama entidad relación y por último se genera o traslada a SQL. El diagrama ayuda a pensar en todas las características, analizar y describir la funcionalidad de cada entidad del programa, su funcionalidad es establecer esas entidades como si fueran objetos donde cada elemento cada ente tiene una cantidad de características y una relación con el resto de elementos que se encuentran en el diagrama , el modelo entidad relación da una idea de que tablas, campos y tipos de datos se haran en esa representación, demás que ya se podrán traducir cada relación a una tabla y por ultimo llegar a SQL para aplicar data manipulation y data definition.

Data Definition Statements
Instrucciones SQL con las cuales se puede crear la base de datos, las tablas, definir las restricciones que tienen que ver con estructura o integridad.

ON UPDATE NOW() permite actualizar un campo al actualizar la tabla que lo contiene, considerado un trigger o disparador –

Normalización: Distribución de datos únicos en diferentes tablas para que no exista información duplicada sobre elementos de la base de datos.

Join → Unir una tabla ante otra y unir campos.

Subconsulta: Es una consulta SQL dentro de otra consulta SQL, una subconsulta no debería generar una tabla sino solamente un dato.

El estado de una relación: Si la relación es la definición de la tabla, el estado corresponde con el contenido de esa relación en ese momento dado por tuplas; los atributos que tiene cada tupla son elementos que están relacionados y por lo tanto se le da su componente relacional.

Las Restricciones de integridad: Tiene que prohibir que se puedan eliminar el primer registro de distintas tablas, es decir llaves foráneas que se utilizan en otras tablas. Si la integridad no existe se producen agujeros de información en la base de datos, por lo tanto se debe tener alguna forma para restringir que información se puede eliminar o modificar haciendo uso de las restricciones de integridad. La integridad de la información no se debe quebrantar eliminando datos de la base de datos.

Sentencias de manipulación de datos en SQL
Aliasing en tablas: Se aplica mediante la palabra reservada AS es atributos o relaciones para generar un alias temporal para mostrar un dato con un nombre distinto o para aplicar nomenclatura especifica.

Limite: Se aplica mediante la palabra reservada **LIMIT**, si solo se necesita un numero especifico de filas de un conjunto de resultados, utilizar una clausula LIMIT en la consulta, en lugar de recuperar todo el conjunto de resultados. Mediante el LIMIT también es posible generar la visualización de resultados mediante "paginas de datos", limitando la cantidad de filas y la cantidad de filas que se muestran en pantalla.

El **BETWEEN** Incluye a los extremos. El **LIMIT** se usa al final de la instrucción que limita la cantidad de registros a mostrar. El **LIMIT** se puede delimitar por **donde_empieza, cuantos_mostrara** también conocida como paginación.

Los operadores: =, <, >, <>, <=, =>, <<, >>, <=>, AND, OR o LIKE se pueden usar en expresiones en la lista de columnas de salida(a la izquierda de FROM) en instrucciones SELECT. También pueden aplicarse en los condicionales WHERE.

LIKE
El operador LIKE se usa en la cláusula WHERE para buscar un patrón específico sobre un atributo. Los patrones en SQL se aplican con algo llamado Comodines.
Hay dos comodines que se utilizan con el operador LIKE, pueden tener diferentes combinaciones y esas combinaciones son las que permiten identificar patrones; los comodines siguientes pueden variar según SGBD.

- % Representa 0, 1 o varíos caracteres es decir puede haber cualquier carácter cualquier cantidad de veces (En expresión regular " .* ").
- _ Representa un solo carácter cualquier carácter pero solo una repetición.

LIKE En la cláusula WHERE

- Atributo LIKE ' z___% ': Buscar cualquier valor que comience con "z" que tenga al menos 3 caracteres de longitud.
- Atributo LIKE '_z% ': Cualquier valor que tenga una z en la segunda posición.
- Atributo LIKE '%z': Buscar cualquier valor que termine en una z.
- Atributo LIKE '%hn%': Buscar cualquier valor que tenga hn en cualquier posición del campo.

IN

Abreviatura de multiples condiciones OR permite especificar varios valores en una clausula WHERE, Ejemplo:
Comando: **$ WHERE atributo IN(value1, value2, ...);**
Comando: **$ WHERE atributo IN(SELECT STATEMENT ...);**

GROUP BY, MAX, MIN, AVG, SUM

La función COUNT() devuelve el numero de filas que coincide con un criterio especifico, se ha usado para la verificación de existencia de cantidad de tareas para un usuario, etc.

- La función MIN() devuelve el valor más pequeño de la columna seleccionada.
- La función MAX() devuelve el valor más grande de la columna seleccionada.
- La función AVG() devuelve el valor promedio de una columna numérica.
- La función SUM() devuelve la suma total de una columna numérica.

ORDER BY

Se utiliza para ordenar el conjunto de resultados en orden ascendente (ASC) o descendente (DESC).

HAVING

Dicha cláusula se usa en SQL para cuando existen condicionales que no pueden aplicarse en el WHERE. Ejemplo:
Comando: **$ GROUP BY atributo(s) HAVING condicional**

JOIN

- **JOIN** → Retorna los registros que tienen una condición de igualdad ambas tablas

- **LEFT JOIN → Devuelve todos los registros de la tabla izquierda (la que está en el FROM) y los registros coincidentes de la tabla derecha (la que está en el JOIN), el resultado es NULL desde el lado derecho sino hay coincidencia.**
- **RIGTH JOIN → Devuelve todos los registros de la tabla derecha.**

SUBSTRING_INDEX(tex_name, ´´, 1) → Permite conocer la cadena que esta antes de un condicional de texto; INSTRUCCIÓN ANTERIOR → Implica que se obtendrá el texto que haya antes del primer espacio.