UT3/4. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

1. Introducción.-

En el presente tema se estudia y analiza los aspectos fundamentales relacionados con los lenguajes de definición y manipulación de datos en los sistemas de bases de datos relacionales. Se trata de un tema de suma importancia dentro del campo de estudio de las aplicaciones ya que la mayoría de las aplicaciones almacenan sus datos de forma persistente en una base de datos relacional, por lo que la forma en que éstos se definan y manipulen va a condicionar el mantenimiento, rendimiento y velocidad de desarrollo de la aplicación.

2. Sistemas de información.-

Un sistema de información es un conjunto de elementos que gestiona la información de una determinada organización. Sus componentes son:

- **Datos:** Información relevante que almacena y gestiona el sistema de información
- **Hardware**: Equipamiento físico que se utiliza para gestionar datos
- **Software**: Aplicaciones que permiten el funcionamiento adecuado del sistema
- **Recursos humanos**: Personal que maneja el sistema de información.

Existen dos tipos fundamentales de sistemas de información:

- **Orientados al proceso:** En estos sistemas de información se crean diversas aplicaciones(software) para gestionar diferentes aspectos del sistema. Cada aplicación realiza unas determinadas operaciones y almacena y utiliza sus propios datos.
- **Orientados a los datos:** En estos sistemas los datos se almacenan en una única estructura lógica que es utilizable para las aplicaciones. A través de esta estructura se accede a los datos que son comunes a todas las aplicaciones.

UT3/4. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

2.1.-Sistemas de ficheros.-

Los sistemas de ficheros son sistemas de información orientados al proceso que tienen las siguientes características:

- Los ficheros se diseñan para una determinada aplicación
- Los datos se encuentran almacenados en soportes de almacenamiento secundario, mientras su descripción está separada de los mismos formando parte de los programas
- No hay control sobre el acceso y manipulación de los datos más allá de lo impuesto por los programas de aplicación.

Los inconvenientes que se plantean son los siguientes:

- Hay una ocupación inútil de memoria secundaria.
- Suele haber cierto grado de inconsistencia y duplicación de información.
- Falta de flexibilidad del sistema de ficheros para adaptarse a las nuevas necesidades.
- Existe cierta dificultad para compartir información.

2.2.- Bases de datos.-

Una base de datos es un conjunto estructurado de datos relacionadas entre sí que reside en soportes de almacenamiento secundario.

Las características que los diferencian de los sistemas de ficheros son:

- Además de los datos, se almacenan las relaciones entre ellos y sus restricciones semánticas.
- No debe existir redundancia lógica, aunque se admite redundancia física por eficiencia.
- Las bases de datos han de atender a múltiples usuarios y diferentes aplicaciones.
- Existe independencia tanto física como lógica entre datos y tratamientos.
- La definición y la descripción de los datos están integradas en los mismos datos.
- Incorporan procedimientos de actualización y recuperación que mantienen la integridad, seguridad y confidencialidad de los datos.

UT3/4. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

2.3.- Modelos de datos. Esquemas. Diseño de bases de datos.

Un **modelo de datos** es una colección de conceptos para la descripción de los datos, las relaciones entre ellos y las restricciones que deben cumplir. Un **esquema** es una descripción de una base de datos mediante un modelo de datos. **Los modelos de datos** se clasifican en:

Modelos conceptuales: Describen los datos con un alto nivel de abstracción, utilizando entidades(concepto del mundo real), atributos(propiedades de interés de una entidad) y las relaciones(interacción entre dos o más entidades). Son independientes de la base de datos a utilizar. Por ejemplo, el modelo entidad/relación y el modelo orientado a objetos.

Modelo lógico: Representan los datos valiéndose de estructuras de registros de varios tipos, formados por un número determinado de campos. Son dependientes de la base de datos a utilizar. Por ejemplo: el modelo relacional, modelo de red y el modelo jerárquico.

Modelos físicos: los modelos físicos describen cómo se almacenan los datos en cuanto al formato de los registros, la estructura de los ficheros y los métodos de acceso utilizados.

El diseño de la base de datos se estructura en tres pasos:

Diseño conceptual: Recibe como entrada la especificación de requerimientos y su resultado es el esquema conceptual, que es una descripción de alto nivel de la estructura de la base de datos mediante un modelo conceptual y que es independiente del SGBD que se utilice.

Diseño lógico: Recibe como entrada el diseño conceptual y da como resultado el esquema lógico, que es la descripción de la estructura de la base de datos mediante un modelo lógico, y que puede ser procesado por el SGBD que se utilice.

Diseño físico: Recibe como entrada el esquema lógico y da como resultado un esquema físico, que es una descripción mediante un modelo físico de las estructuras de almacenamiento y de los métodos usados para tener un acceso efectivo a los datos.

UT3/4. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

3. Lenguajes para la definición y manipulación de datos en sistemas de base de datos relacionales.-

Podemos definir una base de datos como un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna organización. Para trabajar con dicho conjunto de datos, la arquitectura ANSI/X3/SPARC diferencia dos tipos de lenguajes:

- ✔ **Lenguaje de definición de datos**
- ✔ **Lenguaje de manipulación de datos**

3.1.-Lenguajes para la definición de datos.-

El lenguaje de descripción o definición de datos (DDL Data Definition Language) propio de cada SGBD, permite al desarrollador de la base de datos especificar los elementos de datos que integran su estructura y las relaciones que existen entre ellos, las reglas de integridad semántica, las características de tipo físico y las vistas lógicas de los usuarios.

3.2.- Lenguajes para la manipulación de datos.-

Una vez descrita la estructura de la base de datos, los usuarios utilizarán un lenguaje de manipulación de datos (DML Data Manipulation Language) para realizar las siguientes operaciones sobre dicha estructura:

- ✚ **Recuperar información:** Consultar la base de datos
- ✚ **Actualizar información:** La actualización supondrá tres tipos de operaciones.
  - ◆ Inserción.
  - ◆ Borrado.
  - ◆ Modificación de los datos.

A su vez, dentro de los DML podemos destacar los siguientes ejemplos comerciales:

SQL: (Structured Query Language)

QBE: Query By Example. Basado en el cálculo relacional orientado a dominios. QBE de microsoft (Access)..

QUEL (Query Language) basado en cálculo relacional orientado a tuplas.

3.3. Características:

- ✚ **Embebido/autocontenido:** Un lenguaje autocontenido no necesita de otro, mientras que un lenguaje embebido se inserta en el seno de un programa escrito en otro lenguaje, denominado anfitrión.
- ✚ **Procedimental:** Un lenguaje de manipulación de datos es más procedimental en cuanto con más detalles es preciso especificar el

procedimiento necesario para acceder a la base de datos.

- **Diferido/Conversacional:** Algunos DML se utilizan en modo diferido, esto es, en tratamiento por lotes, pero en la actualidad la mayoría permiten su uso en modo conversacional, es decir, interactivo, desde un terminal o interfaz gráfico.
- **Navegacional/especificación**: Existen DML llamados navegacionales que recuperan o actualizan los datos registro a registro, debiéndose indicar el camino que se ha de recorrer hasta llegar al registro buscado. Otros actúan sobre el conjunto de registros como el caso de SQL.
- **Compilación/interpretación:** Algunos lenguajes necesitan de un compilador el cual se encarga de generar el código máquina a partir de un código fuente. Por el contrario un lenguaje interpretado es el que es ejecutado paso a paso, sin ser necesaria una traducción previa a la ejecución.

4. Lenguaje SQL.-

El SQL (Structured Query Language), actúa como interfaz entre un usuario y el sistema. El SQL es un lenguaje con un conjunto de instrucciones para definir, manejar y controlar los datos de una BD relacional. Es un lenguaje con construcciones sencillas en un inglés natural. Puede actuar como un lenguaje auto-contenido (de consulta interactiva) y se puede incorporar a otros lenguajes convencionales (C, Java, PHP, etc.).

El SQL es un lenguaje declarativo, no procedimental, es decir, los usuarios especifican el QUE, no el COMO, es decir, se especifican que datos se requieren sin especificar un procedimiento para obtenerlos.

4.1.- El Lenguaje de definición de datos DDL

Proporciona órdenes para definir, eliminar y modificar tablas, así como para crear índices y vistas. Las órdenes DDL más importantes del DDL son:

Creación de una tabla:

CREATE sirve para crear objetos de la bases de datos, entre estos objetos tenemos las tablas, vistas, etc.

```
CREATE [TEMPORARY] TABLE IF NOT EXISTS nombre_tabla
( nombre_coll tipo_coll
…..........................…
[CONSTRAINT nombre_restricción]
[not null]
 [PRIMARY KEY]
```

```
[UNIQUE]
[DEFAULT valor]
[check <condición>]
[REFERENCES nombre_tabla_ref(colref)[ON DELETE CASCADE]],
…
[RESTRICCIONES DE LA TABLA]
)
[tablespace nombre-tablespace];
```

Donde:

Nombre_tabla: Es el nombre de la nueva tabla. Debe ser único dentro de la Base de datos y además debe identificar su contenido, el nombre de la tabla puede ser una cadena de 1 a 30 caracteres alfanuméricos (0-9,a-z,subrayado, $,#) empezando siempre por un carácter alfabético.

nombre_coll: es el nombre de una columna de la tabla. Los nombre de columna deben cumplir las reglas de los identificadores y deben ser únicos en la tabla.

Tipo_coll:especifica el tipo de datos de columna. Se permiten el tipo de datos del sistema o definidos por el usuario.

Cuando almacenamos datos en una tabla se ajustan a una serie de restricciones predefinidas, por ejemplo, que una columna no pueda tomar valores negativos o que una columna tenga que empezar en mayúsculas.

La integridad hace referencia a que los datos de la base de datos han de ajustarse a restricciones antes de almacenarse en la BD y una restricción de integridad será una regla que restringe el rango de valores de una o más columnas de la tabla. Existe otro tipo de integridad que es la integridad referencial que garantiza que los valores de una o varias columnas de una tabla dependan de los valores de otro u otras columnas de otra tabla.

Las restricciones se definen dentro de la orden **CREATE TABLE** y para ello se utiliza la cláusula **CONSTRAINT**. Una vez creadas las restricciones se pueden añadir, modificar o borrar a través de la orden **ALTER TABLE**.

Una cláusula **CONSTRAINT** puede restringir una sola columna, se habla entonces de restricción de columna o puede restringir un grupo de columnas de una tabla, en este caso se llama de restricción de tabla.

- **CONSTRAINT**: palabra clave opcional que indica le principio de una definición de una restricción para la columna o tabla: **PRIMARY KEY**, **NOT NULL**, **UNIQUE**, **FOREIGN KEY** o **CHECK**.

- NOMBRE_RESTRICCIÓN: Los nombres de restricción deben ser únicos en la base de datos
- **NULL** O **NOT NULL** especifica que la columna admite o no admite valores nulos, es decir, si un campo puede quedar sin valor.
- **DEFAULT**: indica el valor por defecto que toma el campo si no es especificado de forma explícita. Por ejemplo, si se declara la siguiente columna:

  ```
  CodigoPostal numeric(5,0) NOT NULL DEFAULT 28941
  ```

  Indica que el campos CodigoPostal es un número de 5 dígitos enteros, que no admite valores nulos y que en caso de no especificarlo por ejemplo en una inserción el valor que tomará por defecto será 28941.
- **UNIQUE KEY**: especifica la creación de un índice, esto es, una estructura de datos que permite un acceso rápido a la información de una tabla y además, controla que no haya ningún valor repetido en el campo. Se producirá un error si se intenta insertar un elemento que ya coincida con un valor anterior. A efectos prácticos la diferencia con una clave primaria es básicamente que un campo UNIQUE puede admitir valores NULL, mientras que un campo que se define como clave primaria no puede admitir valores nulos (debe ser not null).
- **AUTO_INCREMENT** (campos de tipo entero) Indica que ese campo es auto-incrementado después de cada inserción, debe ser numérico, permite indicar el valor inicial para campos de este tipo.
- **COMMENT**: comentario para una tabla, hasta 60 caracteres, también se pueden crear comentarios para cada columna mediante el token COMMENT.
- **ENGINE**: El almacenamiento físico de una tabla en MySQL está controlada por un software especial denominado Motor de almacenamiento. Mediante la opción Engine=nombre_motor, se indica el motor de almacenamiento para la tabla, puede ser entre otros: innodb (son tablas transaccionales con bloqueo de registro y claves foráneas), myIsam por defecto usado en MySQL y que genera tablas operadas a gran velocidad pero sin control referencial y Memory que genera tablas que están almacenadas en memoria en lugar de un archivo físico.
- **PRIMARY KEY** Indica que la columna definida es clave primaria. Una tabla tan sólo puede tener una clave primaria. Si se especifica a nivel de columna la clave sólo puede tener un campo, si la clave se especifica a nivel de tabla, la clave puede ser multicolumna.
- **CHECK** (establece una condición de validación para el campo).
- **IF NOT EXISTS**: también se puede indicar a nivel de tabla, si ya existe

no se crea nuevamente.

- **MAX_ROWS**: número máximo de registros.
- **MIN_ROWS**: número mínimo de registros.
- **TEMPORARY**: el token opcional temporary se utiliza para crear tablas temporales, es decir, se borrará en el momento en que el usuario se desconecte.

- Tipo de datos:

  - Numéricos: **INT** , **DECIMAL** o **DEC**(parte entera, decimales)
  - Carácter **CHAR(n)**: Cadena de longitud fija. Se completa con espacios
  - **VARCHAR(n)**: Cadenas de longitud variables. Se ajusta al contenido
  - **TEXT** y **BLOB**: grandes cantidades de texto.
  - Fecha y Hora:DATE Y **DATETIME**

OPCIONES: especifica que hacer con las reglas de integridad referencial

1. ON DELETE CASCADE
2. ON UPDATE CASCADE
3. …

Las opciones **ON DELETE** y **ON UPDATE** establecen el comportamiento del gestor en caso de que las filas de la tabla padre (es decir, la tabla referenciada) se borren o actualicen. Los comportamientos pueden ser **CASCADE, SET NULL** Y **NO ACTION**.

Si se usa un **NO ACTION**, y se intenta un borrado o actualización sobre la tabla padre, la operación se impide, rechazando el borrado o la actualización

Si se especifica **CASCADE**, la operación se propaga a la tabla hija, es decir, si se actualiza la tabla padre, se actualizan los registros relacionados con la tabla hija, y si se borra un registro de la tabla padre, se borran aquellos registros de la tabla hija que estén referenciando al registro borrado.

Si se indica **SET NULL**, se establece a null la clave foránea afectada por un borrado o modificación de la tabla padre.

EJEMPLOS:

Crear una tabla:

```
create table if not exists pedido(
        codigo int auto_increment primary key,
        fecha datetime,
        estado enum('pendiente','entregado','rechazado')
        comment='tabla de pedido a proveedores'
        auto_increment=10000
        max_rows=1000000
        engine=innodb;
```

Crear una tabla padre e hija(con referencias)

```
create table clientes(
dni varchar(8) PRIMARY KEY
nombre varchar(50),
direccion varchar(60)
};
create table mascotas(
codigo integer PRIMARY KEY,
nombre varchar(50),
raza varchar(50),
cliente varchar(9) REFERENCES clientes (dni)
ON DELETE CASCADE ON UPDATE SET NULL";
```

Si no se especifica ON DELETE u ON UPDATE por defecto se actúa como NO ACTION.

Borrado de una tabla (Estructura y datos):

```
DROP table nombre_tabla [CASCADE CONSTRAINT];
```

Al borrar una tabla se borra tanto su contenido como sus datos, sus índices asociados y los privilegios concedidos sobre estas también se borran, las vistas creadas directa o indirectamente sobre esta tabla. Cada usuario puede borrar sus propias tablas pero no las de otro usuario al menos que tenga concedido un permiso adecuado. Si se hace referencia a la clave primaria de esta tabla mediante restricciones ***Foreign Key o References***, la cláusula CASCADE CONSTRAINT permite suprimir estas restricciones de integridad referencial en las tablas descendientes.

Borrado de los registros de una tabla.-

Con la orden TRUNCATE se eliminan todas las filas de una tabla y se puede liberar espacio utilizado por esa tabla.

```
TRUNCATE table nombre_table [{DROP|REUSE} STORAGE]:
```

Con Drop Storage se desasigna todo el espacio, con drop reuse mantendrá reservado el espacio para nuevas filas.

Modificar la estructura de una tabla.-

Para modificar la estructura de una tabla se utiliza el comando ALTER TABLE.

```
ALTER TABLE nombre_table
{[ADD (columna tipodato [restriccion columna][..])]
[MODIFY (columna tipodato[restriccion columna][..])]
[ADD CONSTRAINTS restricción]
[DROP CONSTRAINTS restricción]};
```

Añadir o modificar columnas:

```
ALTER TABLE nombretabla {ADD|MODIFY} (columna tipo[restriccion..]);
```

Eliminación de columnas:

```
ALTER TABLE nombretabla DROP COLUMN nombre_columna;
```

Añadir restricción de tabla:

```
ALTER TABLE nombretabla ADD CONSTRAINT nombrerestriccion restriccion:
```

Eliminar una restricción de una tabla:

```
alter table nombre table DROP CONSTRAINT nombre_restricción;
```

Activación y desactivación de una restricción:

```
ALTER TABLE <nombretable> [enable validate|enable novalidate|disable] <nombrerestricción>;
```

Donde:

enable validate: activa la restricción si el conjunto cumple la restricción

enable novalidate: activa la restricción para las siguientes instrucciones

disable: desactiva la restricción

Renombrado de tablas

```
RENAME TABLE nombre_tabla TO nombre_tabla_nuevo
```

Consultar la estructura de una tabla

- `DESCRIBE [ESQUEMA] nombre_tabla;`
- `SHOW CREATE TABLE nombre_tabla.`

da un listado de la tabla mostrando sus columnas, y aportando información sobre los tipos de datos, restricciones, etc.

VISTAS.-

Las vistas son tablas virtuales que contienen el resultado de una consulta ***select***, tienen la misma estructura que una tabla cualquiera, es decir, están organizadas en filas y columnas. Una de las principales ventajas de utilizar vistas procede del hecho de que la vista no almacena los datos, sino que hace referencia a una o varias tablas de origen mediante una consulta ***select***, consulta que se ejecuta cada vez que se hace referencia a la vista. Cualquier modificación que se haga sobre los datos de las tablas origen es inmediatamente visible en la vista, cuando ésta vuelva a ejecutarse. Su sintaxis es:

```
CREATE [or REPLACE] VIEW nombre_vista
[lista de columnas]
as SELECT [..]
```

La opción REPLACE reemplaza la vista en el caso de que ésta ya exista. Se pueden realizar consultas sobre las vistas, también sentencias DML sobre las vistas, las modificaciones, borrados e inserciones están restringidas a vistas que estén definidas sobre una única tabla.

Borrado de una vista

```
DROP VIEW <nombre_vista>
```

Creación de un índice.-

Los índices sirven para mejorar el rendimiento de las consultas.

```
CREATE [unique] INDEX <nombre_indice>
```

```
ON nombre_table [columnas [{asc|desc}][..])
```

Borrado de un índice.-

Cuando se borra una tabla automáticamente se borran los índices asociados a ella. Los índices ocupan espacio dentro de la BD como si de una tabla se tratara y por esa razón se aconseja tener solo como índices aquellas columnas por las cuales se realizan consultas de forma periódica. Para borrar un índice:

```
DROP INDEX <nombre_indice>;
```

4.2.- Lenguaje de manipulación de datos.-

Instrucciones de actualización: son aquellas que no devuelven ningún registro sino que son las encargadas de acciones como añadir, borrar y modificar registros.

INSERT, DELETE, UPDATE

Inserción de registros.-

La sentencia INSERT nos permite insertar nuevas filas en una tabla de la base de datos. Su sintaxis más sencilla:

```
INSERT INTO <table> (<lista_campos>) VALUES (valores);
```

Si no se especifica la lista de campos, el orden de los valores debe ser el de la tabla.

Borrado de registros.-

```
DELETE FROM table WHERE <condición>;
```

La principal diferencia entre DELETE y TRUNCATE, es que con DELETE se puede borrar un solo registro o más mientras que truncate los borra todos. Con DELETE se puede disparar un trigger asociado a la tabla mientras que con TRUNCATE no.

Modificación de registros.-

Permite modificar filas de una tabla de una base de datos.

```
UPDATE TABLE SET columna1=valor1,[columna2=valor2..] … [where <condición>];
```

Se modifican las columnas indicadas en el apartado set con los valores indicados, la cláusula where permite especificar que campos son modificados.
Otro tipo de sintaxis:

```
UPDATE TABLE SET columna1=[SELECT ...WHERE <condición>]
```

Este tipo de actualizaciones sólo son válidas si la sentencia SELECT devuelve un único valor, que además debe ser compatible con la columna que se actualiza.

UT3/4. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS EN SISTEMAS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

Consulta de datos: cláusula SELECT

Su función principal es la de recuperar filas de la tabla o tablas, además es capaz de:

- Obtener datos para la creación de una tabla.
- Realizar operaciones estadísticas.
- Definir cursores.
- Realizar operaciones «totalizadoras» sobre los grupos de valores que tienen los mismos en ciertas columnas.
- Se puede utilizar como «sub-consulta» para formar parte de una condición.

Sintaxis básica:

```
SELECT lista
From nombre_tabla
Where condición
Group by expresiones
Having condición
Order by expresion [asc|desc]
```

Select: especifica las columnas que serán devueltas
From: indica la tabla o tablas a las que se accederá
Group by: agrupa las filas seleccionadas por la cláusula where
Having: especifica una condición de selección para un grupo
Order by: ordena las filas seleccionadas