



# **Preparador Informática**

[www.preparadorinformatica.com](http://www.preparadorinformatica.com)

## **TEMA 41. INFORMÁTICA**

**UTILIDADES DE LOS SISTEMAS  
GESTORES DE BASE DE DATOS PARA  
EL DESARROLLO DE APLICACIONES.  
TIPOS. CARACTERÍSTICAS.**

## **TEMA 41 INF: UTILIDADES DE LOS SISTEMAS GESTORES DE BASE DE DATOS PARA EL DESARROLLO DE APLICACIONES. TIPOS. CARACTERÍSTICAS.**

1. INTRODUCCIÓN
2. CONCEPTOS BÁSICOS
3. TIPOS Y CARACTERÍSTICAS DE LAS UTILIDADES DE LOS SISTEMAS GESTORES DE BASES DE DATOS
  - 3.1. ACCESO Y PRESENTACIÓN DE LOS DATOS
    - 3.1.1. MACROS
    - 3.1.2. INTERFACES
  - 3.2. HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES
    - 3.2.1. LENGUAJES DE PROGRAMACIÓN ANFITRIONES
    - 3.2.2. LENGUAJES DE CUARTA GENERACIÓN
    - 3.2.3. LENGUAJES DE PROGRAMACIÓN EMPOTRADOS
    - 3.2.4. GENERADOR DE APLICACIONES
  - 3.3. RECUPERACIÓN DE LA BASE DE DATOS EN CASO DE CAÍDAS O FALLOS DEL SISTEMA
  - 3.4. SEGURIDAD DEL SISTEMA
4. CONCLUSIÓN
5. BIBLIOGRAFÍA

## 1. INTRODUCCIÓN

Antes de aparecer los sistemas gestores de bases de datos (SGBD), la información se trataba y se gestionaba utilizando los típicos sistemas de gestión de archivos que iban soportados sobre un sistema operativo. Estos consistían en un conjunto de programas que definían y trabajaban sus propios datos. Los datos se almacenaban en archivos y los programas manejaban esos archivos para obtener la información. Si la estructura de los datos de los archivos cambiaba, todos los programas que los manejaban debían modificarse.

Esto suponía un gran inconveniente a la hora de tratar grandes volúmenes de información. Surgió así la idea de separar los datos contenidos en los archivos de los programas que los manipulan, es decir, que se pueda modificar la estructura de los datos de los archivos sin que por ello se tengan que modificar los programas con los que trabajan. Es decir, estructurar y organizar los datos de forma que se pueda acceder a ellos con independencia de los programas que los gestionan. Y es así como surgen las bases de datos, que son gestionadas por los SGBD. A su vez estos SGBD tienen utilidades para poder desarrollar aplicaciones que nos ayudarán en nuestro trabajo diario con las bases de datos.

Lo expuesto anteriormente justifica la importancia del tema en el cual vamos a empezar definiendo y diferenciando los conceptos de bases de datos y SGBD para posteriormente adentrarnos en las utilidades de los sistemas gestores de bases de datos para el desarrollo de aplicaciones y sus tipos y características.

## 2. CONCEPTOS BÁSICOS

Podemos definir una **base de datos** como un conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. Los datos están interrelacionados y estructurados.

Un **sistema gestor de base de datos o SGBD** es una colección de programas de aplicación que permite a los usuarios la creación y el mantenimiento de una base de datos, facilitando la definición, construcción y manipulación de la información contenida en ésta.

En la actualidad debido al gran volumen de datos que manejamos se requiere de sistemas gestores de bases de datos robustos, que nos permitan acceder y

gestionar los datos de un modo eficaz y eficiente. Los sistemas gestores de bases de datos (SGBD) más extendidos son los relacionales.

### **3. TIPOS Y CARACTERÍSTICAS DE LAS UTILIDADES DE LOS SISTEMAS GESTORES DE BASES DE DATOS.**

El conjunto de herramientas que nos permiten realizar las operaciones típicas de una base de datos e interactuar con la misma de manera más fácil y rápida es lo que consideramos las utilidades de un SGBD. Las podemos clasificar dependiendo de la función que realizan:

- Acceso y presentación de los datos.
- Herramientas para el desarrollo de aplicaciones.
- Recuperación de la base de datos en caso de caídas o fallos del sistema.
- Seguridad del sistema.

Hablaremos de los cuatro tipos centrándonos en las herramientas para el desarrollo de aplicaciones que es el eje central del tema y de los otros tipos haremos un pequeño comentario resumen sobre ellos.

#### **3.1. ACCESO Y PRESENTACIÓN DE LOS DATOS.**

Los sistemas de base de datos proporcionan gran variedad de herramientas que permiten generar interfaces especializadas para tareas específicas como generación de informes, consultas, formularios...

##### **3.1.1. MACROS**

Una macro es una serie de instrucciones que se almacenan para que se puedan ejecutar de manera secuencial mediante una sola llamada u orden de ejecución. Dicho de otra manera, es una instrucción compleja formada por otras instrucciones más sencillas. Esto permite la automatización de tareas repetitivas.

Las macros tienden a almacenarse en el ámbito del propio programa que las utiliza y se ejecutan pulsando una combinación especial de teclas o un botón especialmente creado y asignado para tal efecto. Las macros se componen de una serie de acciones predeterminadas por el programa, que realizan operaciones de gestión tales como abrir o cerrar formularios e informes, ejecutar comandos de menú o examinar los registros de una tabla.

### 3.1.2. INTERFACES

Las interfaces se usan para acceder y presentar datos. Hay varias categorías:

- **Interfaces orientadas a consultas:** Las consultas permiten formular preguntas sobre los datos almacenados en las tablas. La forma en que se diseñe la consulta indicará con exactitud al SGBD qué datos debe recuperar. Se trata de una de las partes más importantes de una base de datos. La estructura de una consulta la componen:
  - Elección de los campos.
  - Criterios de consulta.
  - Campos calculados.
  - Valores únicos.
  - Enlace de varias tablas.
  - Totales.
- **Interfaces para formularios:** Los formularios son herramientas para la manipulación de la información contenida en una base de datos. Van a ser el elemento que el usuario final de la aplicación que desarrollemos va a utilizar, por tanto, en su diseño, tendremos en cuenta que sean formularios de fácil uso, útiles para:
  - Presentar los datos de forma atractiva.
  - Ofrecer una apariencia similar a los formularios de papel.
  - Calcular totales.
  - Mostrar gráficos.
  - Mostrar datos de varias tablas.
- **Informes:** Los informes son el medio que ofrece el SGBD para presentar la información de la base de datos en un formato adecuado para su salida por impresora o cualquier dispositivo de salida.
- **Interfaces para el acceso remoto a bases de datos:** Las bases de datos incorporan utilidades para poder crear de una manera fácil un acceso desde ordenadores externos a bases de datos de cualquier tipo. Por ejemplo, ODBD y ODAP.
- **Interfaces Web:** Son una serie de herramientas para permitir la correcta comunicación entre Internet e Intranets y las bases de datos.

## 3.2. HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES.

Una **aplicación** es un programa informático que realiza una tarea.

Una **aplicación de Base de Datos** es un programa que utiliza los datos de un sistema de gestión de bases datos.

La mayoría de las aplicaciones de base de datos presentan datos de forma útil o facilitan la introducción o actualización de datos en la Base de Datos. Las aplicaciones de bases de datos trabajarán interactuando con el SGBD.

Hay varias formas de desarrollar aplicaciones de BD, desde lenguajes de alto nivel hasta herramientas específicas para este propósito. Veremos a continuación las distintas formas en los siguientes apartados.

### 3.2.1. LENGUAJES DE PROGRAMACIÓN ANFITRIONES.

Los programas de aplicación se escriben normalmente en un lenguaje de alto nivel (Cobol, C, C++, Python, Java, etc...), que denominaremos lenguaje anfitrión. Los lenguajes de este tipo son procedimentales, es decir, que es necesario especificar con detalle el procedimiento necesario para acceder a la base de datos y para conseguir este acceso, las instrucciones DML (Lenguaje de Manipulación de Datos) necesitan ser ejecutadas desde el lenguaje anfitrión. Hay dos maneras de conseguir esto:

- Mediante una API (Librería de procedimientos) que permita enviar instrucciones DML y DDL a la base de datos, así como recuperar los resultados. Un ejemplo de ese tipo de API es ODBC (Open Data Base Connectivity), definida para permitir el acceso a la base de datos desde C ó JDBC (Java Data Base Connectivity) es similar para Java.
- Extendiendo la sintaxis del lenguaje anfitrión para incorporar llamadas DML dentro del programa del lenguaje anfitrión. Normalmente se implementa con un preprocesador. Si se utiliza esta técnica se dice que el lenguaje de consulta está embebido en el lenguaje anfitrión.

### 3.2.2. LENGUAJES DE CUARTA GENERACIÓN

Los lenguajes de cuarta generación son entornos de desarrollo de aplicaciones constituidos por un conjunto de herramientas integradas entre las que se

encuentran editores, compiladores, sistemas para el acceso a bases de datos, generadores de informes, generadores de pantallas (modo carácter, interfaces gráficas) ...

Los lenguajes que incorporan los 4GL suelen ser mezcla de lenguajes procedimentales y no procedimentales. La parte procedimental se manifiesta en la definición de tipos de constantes, tipos de datos elementales, visibilidad de las variables (locales o globales), sentencias de control de flujo, definición de funciones y procedimientos, etc., mientras que la parte no procedimental suele estar basada en el lenguaje SQL o, como mínimo, en lenguajes de consulta de bases de datos relacionales.

Con los 4GL se consigue un aumento de productividad gracias a:

- La utilización de funciones preprogramadas.
- El entorno de desarrollo que facilita la realización de determinadas tareas como diseño de pantallas o informes.

La mayoría de los gestores de bases de datos cuentan con un lenguaje de cuarta generación. Son lenguajes propietarios, es decir, que sirven únicamente para acceder a esa base de datos en particular. El aprovechamiento de recursos del gestor es muy alto. Aunque también existen Lenguajes que son independientes del gestor de base de datos con capacidad para acceder a diferentes bases de datos, generalmente aquellas que soportan un estándar común.

Los lenguajes de cuarta generación suelen utilizarse conjuntamente con las denominadas Herramientas CASE que son herramientas para el desarrollo rápido de aplicaciones.

Los principales componentes de un lenguaje de cuarta generación son:

- **Editor:** Donde se escriben las sentencias del lenguaje de programación.
- **Compilador:** Traduce las sentencias del lenguaje fuente a código binario o a un lenguaje intermedio.
- **Módulo de acceso a base de datos:** Facilita toda la comunicación con la base de datos, desde el diseño de las tablas hasta la construcción de sentencias para recuperar información. La mayoría de los 4GLs soporta el lenguaje SQL estándar como lenguaje de acceso a base de datos relacionales, lo que garantiza la portabilidad.

- **Módulo de ayuda a las pruebas:** Permiten una ejecución controlada del código para poder aislar un error.
- **Generador de informes y pantallas:** Incorporan módulos para la construcción rápida de pantallas, ya sea en modo carácter o en modo gráfico. Asimismo, algunos cuentan con un módulo de generación de informes a través de consultas a la base de datos.
- **Diccionario de datos:** cuentan con un diccionario en el que almacenan la información referente a los objetos de la aplicación.
- **Gestor de librerías:** Permite la distribución de los objetos por librerías y su rápida localización.
- **Módulo de control de versiones:** Incorpora facilidades para el control de versiones de la aplicación.

### 3.2.3. LENGUAJES DE PROGRAMACIÓN EMPOTRADOS.

Los lenguajes definidos para el modelo relacional de datos son lenguajes orientados a la manipulación de conjuntos de tuplas. En ellos los operadores seleccionan tuplas con la información requerida en una consulta, o bien actualizan grupos de tuplas de la forma especificada en un requerimiento de actualización.

Esta característica introduce un problema a la hora de integrar estos lenguajes con los lenguajes de programación clásicos, ya que estos últimos no contemplan la manipulación de estructuras de datos complejas.

La dificultad principal que se plantea en el acceso a una base de datos relacional desde un programa reside en la incompatibilidad entre los tipos de estructuras del modelo de datos y del lenguaje de programación. Para superar esta dificultad existen dos opciones: restringir las consultas a aquellas que devuelvan una única tupla o bien introducir algún mecanismo que permita seleccionar y manipular las tuplas de una relación individualmente, este último mecanismo viene representado por los cursores. En esencia un cursor es un puntero destinado a apuntar a las tuplas de una relación. El cursor puede moverse a través de las tuplas de la relación, pudiéndose en cualquier momento seleccionar o actualizar la tupla apuntada.

El acceso a una base de datos relacional desde un programa se plantea necesario cuando se piensa en el desarrollo de aplicaciones complejas que



necesitan: manipulación individualizada de las tuplas, estructuras de programación tradicionales (selección y repetición), interacción con el usuario y gestión de errores. Veamos un ejemplo de lenguaje de programación empotrado:

### SQL empotrado

El SQL empotrado es un conjunto de instrucciones que permite que SQL sea utilizado con lenguajes de programación tradicionales como COBOL, C y Pascal (que se denominan lenguajes anfitrión). Estas instrucciones notifican al preprocesador que lo que sigue se debe reemplazar con llamadas a las rutinas de SGBD. Estas instrucciones se ejecutan cuando se ejecuta el programa de acuerdo a su lógica interna. En este contexto, el intercambio de información con el SGBD se realiza a través de variables del lenguaje, a las que se denomina *variables huéspedes*; por ejemplo, el resultado de las consultas es asignado a variables del programa declaradas con ese fin. La estructura típica de un programa de manipulación de base de datos es la siguiente:

Programa

Declaración de variables

Declaración de variables huéspedes para acceso a la BD

Fin declaración de variables huéspedes para acceso a la BD

Fin declaración de variables

Comienzo del código del programa

*instrucciones propias del lenguaje*

Conexión a la base de datos

*instrucciones de SQL*

...

Desconexión de la base de datos

*instrucciones propias del lenguaje*

Fin del código de programa

Algunas consideraciones generales de sintaxis para SQL empotrado son:

1. Todas las instrucciones del SQL embebido van precedidas de las palabras reservadas EXEC SQL, de forma que son fácilmente identificables por el precompilador, y finalizan con un símbolo especial. Por ejemplo, en el lenguaje C este símbolo es el *punto y coma (;)*.

2. Las variables del lenguaje que son utilizadas en instrucciones SQL (variables huéspedes) deben ser declaradas en una sección especial encabezada y terminada de la siguiente forma:

```
EXEC SQL BEGIN DECLARE SECTION;  
...  
EXEC SQL END DECLARE SECTION;
```

Las variables deben tener un tipo apropiado al uso que se va a hacer de ellas; sintácticamente pueden aparecer en una instrucción SQL en cualquier lugar en el que puede aparecer un literal y deben ir precedidas del símbolo dos puntos (:).

Todo programa con SQL embebido debe incluir la declaración de una variable SQLCODE o una variable SQLSTATE o ambas. Después de la ejecución de cada instrucción SQL, el SGBD devuelve al programa en estas variables información sobre la ejecución de la instrucción.

3. En un programa, cualquier instrucción SQL debería estar seguida por código que controlase los valores de las variables SQLCODE y SQLSTATE relativos a la ejecución de la instrucción; para simplificar esta tarea el lenguaje proporciona una sentencia para el control general de errores, ésta es la sentencia **WHENEVER**:

```
EXEC SQL WHENEVER condición acción  
condición ::= {SQLERROR | NOT FOUND}  
acción ::= {CONTINUE | GO TO etiqueta}
```

La sentencia **WHENEVER** no es una instrucción ejecutable, es más bien una directriz al procesador del lenguaje SQL:

- *WHENEVER condición GO TO etiqueta* significa que el procesador de SQL incluirá detrás de cualquier sentencia SQL del programa la sentencia *IF condición THEN GO TO etiqueta*.
  - *WHENEVER condición CONTINUE* significa que el procesador de SQL no tomará ninguna acción siendo responsabilidad del programador controlar el flujo del programa.
4. La operación de consulta se realiza con la siguiente variante de la sentencia **SELECT** del lenguaje SQL interactivo:

```
SELECT [ALL | DISTINCT] lista_elemento_selección  
INTO lista_variable  
FROM lista_relación  
[WHERE condición_búsqueda]
```

```
[GROUP BY lista_atributo]  
[HAVING condición_búsqueda]
```

En este contexto, la sentencia SELECT debe devolver una única tupla, sino se producirá un error. Como se puede observar la única variante de la sentencia consiste en la cláusula INTO que especifica la lista de variables del programa que serán utilizadas para recibir los datos seleccionados. Los elementos de la lista de selección y de la lista de variables se corresponden uno a uno en el orden en que aparecen, por ello ambas listas deben tener el mismo número de elementos y ser de tipos de datos compatibles.

Las sentencias de actualización: UPDATE, INSERT y DELETE, tienen la misma sintaxis que en el SQL interactivo con la única diferencia de que pueden incluir variables huéspedes.

### Ejemplo de programa de SQL empotrado en C

Veamos un sencillo ejemplo de programa de SQL empotrado en C que se conectará a la base de datos, hará una consulta sencilla y muestra el resultado por pantalla:

```
#include <iostream>  
#include <string.h>  
  
main()  
{  
EXEC SQL BEGIN DECLARE SECTION  
    VARCHAR nom_usuario[30], contraseña[10];  
    VARCHAR vnombre[20], vapellido[30];  
    float vsalario;  
EXEC SQL END DECLARE SECTION;  
  
strcpy((char *)nom_usuario.arr, "scott");  
nom_usuario.len = strlen(nom_usuario.arr);  
  
strcpy((char *)contrasenia.arr, "tiger");  
contrasenia.len= strlen(contrasenia.arr);  
  
EXEC SQL WHENEVER SQLERROR DO sql_error();  
  
EXEC SQL CONNECT :nom_usuario IDENTIFIED BY :contrasenia  
  
EXEC SQL SELECT nombre, apellido, salario INTO :vnombre,  
:vapellido, :vsalario FROM EMPLEADO WHERE salario =  
(select max(salario) from empleado);  
  
vnombre.arr[vnombre.len]='\0';  
vapellido.arr[vapellido.len]='\0';
```

Declaración de variables

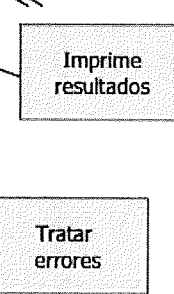
Poner valores variables 'anfitrionas'

Sentencia control de errores

Conexión base de datos

Realizar consulta

```
cout << "Nombre, apellidos, salario:\n";  
cout << vnombre.arr << ' ' << vapellido.arr << ' ' <<  
    vsalario;  
}  
  
int sql_error()  
{  
    EXEC_SQL WHENEVER SQLERROR CONTINUE;  
    cout << "Error detectado" << endl;  
}
```



Imprime resultados

Tratar errores

### 3.2.4. GENERADOR DE APLICACIONES

El generador de aplicaciones es una herramienta de diseño que sirve para crear un sistema central de menús que llame a los programas de utilidad del SGBD. Por tanto, se trata de un sistema conducido por menús para desarrollar aplicaciones. Permite enlazar los distintos componentes del SGBD permitiendo definir imágenes visuales de objetos (por ejemplo, un menú) en la superficie de trabajo del Generador de Aplicaciones. Una vez que están definidos estos objetos se les puede asignar una acción. Una acción es un procedimiento que se va a realizar cuando se seleccione ese ítem.

Para crear una aplicación a partir del Generador de Aplicaciones hacemos:

- **Definición del entorno implícito:** El generador de aplicaciones presenta, al iniciarse, un formulario para definir el entorno de la aplicación. Se utiliza el formulario de definición de aplicaciones para definir el entorno implícito de su aplicación. Es aquí donde se nombra la aplicación, se define su menú principal y se especifica una base de datos o vista que se empleará posteriormente. El SGBD utiliza esta información para crear dos programas que definen su aplicación. Uno describe el sistema de menús y el otro describe las acciones que han asignado a esos menús.
- **Definición de objetos:** El generador de aplicaciones permite diseñar objetos de distintos tipos, mediante una interfaz visual. Una vez que se definen estos objetos, se les pueden asignar acciones. Utiliza los objetos que se definen y las acciones que especifica para generar el código de la aplicación. Permite definir menús, listas y procesos por lotes.

- **Menús:** Permiten especificar tres tipos de menús (de barras horizontales, desplegables y menús de aparición autónomos).  
Proporcionan una forma cómoda y coherente de agrupar los comandos de modo que el usuario pueda acceder fácilmente a ellos.  
Los menús desplegables van ligados normalmente a una de las opciones de un menú de barra horizontal, de forma que cuando se seleccione una opción se desplegará automáticamente el menú desplegable asociado. También puede utilizarse un menú de aparición como submenú para otros menús de aparición.
- **Listas:** Las cajas de listas contienen los nombres de los ficheros, campos de la base de datos o contenidos de campos especificados de una base de datos. Hay tres tipos de cajas de listas (listas de ficheros que contienen nombres de fichero, listas de estructuras que contienen nombre de campo y listas de valores que contienen contenidos de campos de base de datos).
- **Procesos por lotes:** Permiten especificar una lista de operaciones que se van a ejecutar en secuencia cuando se escoge una opción de un menú específico. De forma diferente a los menús y las cajas de listas, el usuario nunca ve procesos por lotes, y no lo necesita. Se puede utilizar un proceso por lotes para hacer una copia de seguridad de una base de datos.
- **Asignación de acciones:** Una vez que se hayan definido los menús y sus ítems correspondientes, pueden asignarse acciones a cada ítem de cada menú. Ejemplos de acciones que podemos asignar son añadir un registro, utilizar un formulario confeccionado o imprimir un informe.
- **Generación del código:** Cuando esté finalizando el sistema de menús y asignadas las acciones o procesos por lotes a los ítems del menú, se puede generar el código de la aplicación. La aplicación no correrá a menos que se haya generado en el lenguaje del SGBD el código del programa necesario.

### 3.3. RECUPERACIÓN DE LA BASE DE DATOS EN CASO DE CAÍDAS O FALLOS DEL SISTEMA.

Esta utilidad es imprescindible para nuestra base de datos ya que cualquier ordenador está sujeto a fallos que pueden hacer que se produzca pérdida de información con graves perjuicios para nuestra base de datos. Por esto el sistema de recuperación de nuestro SGBD tiene la misión de recuperar y

restaurar la base de datos al estado consistente, es decir, el estado en el que estaba antes de que se produjera el fallo.

Para poder actuar debemos primero identificar los tipos de fallos que pueden suceder (fallos en transacciones, fallos en el sistema, fallos del medio).

### **3.4. SEGURIDAD DEL SISTEMA.**

El problema de la seguridad consiste en lograr que los recursos de un sistema sean, bajo toda circunstancia, utilizados para los fines previstos. Los datos de la base de datos deben estar protegidos contra los accesos no autorizados, de la destrucción o alteración malintencionada y de la introducción de inconsistencias. Las medidas de seguridad pueden ser físicas (p.ej, mediante tarjetas de acceso), personales (acceso sólo de personal autorizado con identificación directa de personal) y gracias al uso de las utilidades o herramientas que nos proporciona nuestro SGBD (perfiles de usuario, vistas, restricciones de uso de vistas...).

### **4. CONCLUSIÓN**

El modo en que se accede a la información ha variado mucho en las últimas décadas. El nivel de accesibilidad y la facilidad de uso se han ido incrementando y en la actualidad se accede a los datos mediante aplicaciones con interfaces amigables, las cuales han sido desarrolladas bien en algún lenguaje de programación o bien mediante las utilidades que los propios gestores de bases de datos ofrecen.

En este tema se ha presentado una visión global sobre las utilidades que los sistemas gestores de bases de datos nos ofrecen para el desarrollo de aplicaciones, con el objetivo de conocer cómo se crean las aplicaciones para que las empresas y organizaciones gestionen los datos de una manera eficaz y segura.

### **5. BIBLIOGRAFÍA**

- Date D.J.: **Introducción a los sistemas de bases de datos**. Editorial Addison-Wesley
- De Miguel A,y Piattini M:**Fundamentos y modelos de BBDD**. Edit. Ra-Ma
- Korth H. y Silberschatz: **Fundamentos de bases de datos**. Editorial McGraw-Hill