



Código edición: MAN\_EU\_2303

Reservados todos los derechos.  
© 2013 | EUROINNOVA EDITORIAL

Queda prohibida, salvo excepción prevista en la ley, cualquier forma de reproducción, distribución, comunicación pública y transformación de esta obra sin contar con autorización de los titulares de la propiedad intelectual. La infracción de los derechos mencionados puede ser constitutiva de delito contra la propiedad intelectual (arts. 270 y ss. del Código Penal). El Centro Español de Derechos Reprográficos vela por el respecto de los citados derechos.

Editado por: Euroinnova Editorial  
E-mail: [info@euroinnovaeditorial.com](mailto:info@euroinnovaeditorial.com)  
Web: [www.euroinnovaeditorial.com](http://www.euroinnovaeditorial.com)

Diseño de cubierta: Euroinnova Editorial.  
Impresión: Euroinnova Editorial (Granada).  
Impreso en España.

## **Presentación del manual**

La cualificación profesional es el “conjunto de competencias profesionales con significación en el empleo que pueden ser adquiridas mediante formación modular u otros tipos de formación, así como a través de la experiencia laboral” (Ley 5/2002 de las Cualificaciones y de la Formación Profesional).

Cada cualificación se organiza en unidades de competencia, siendo la unidad de competencia el agregado mínimo de competencias profesionales, susceptible de reconocimiento y acreditación parcial.

Así mismo, cada unidad de competencia lleva asociado un módulo formativo, donde se describe la formación necesaria para adquirir esa unidad de competencia.

Siguiendo esta secuencia, el presente manual “Desarrollo de componente software en sistemas ERP-CRM”, está basado en los contenidos del Módulo formativo: “Creación y mantenimiento de componentes software en sistemas de planificación de recursos empresariales y de gestión de relaciones con clientes” asociado a la Unidad de Competencia: “Realizar y mantener componentes software en un sistema de planificación de recursos empresariales y de gestión de relaciones con clientes”, según el Real Decreto correspondiente.

### **MÓDULO FORMATIVO: Creación y mantenimiento de componentes software en sistemas de planificación de recursos empresariales y de gestión de relaciones con clientes**

Nivel: 3

Código: MF1215\_3

Asociado a la UC: UC1215\_3 Realizar y mantener componentes software en un sistema de planificación de recursos empresariales y de gestión de relaciones con clientes.

Horas: 210

### **UNIDAD FORMATIVA: Desarrollo de componente software en sistemas ERP-CRM**

Nivel: 3

Código: UF1889

Horas: 90



***UF1889***  
***DESARROLLO DE***  
***COMPONENTE***  
***SOFTWARE EN***  
***SISTEMAS ERP-CRM***



# ÍNDICE

|  |           |
|--|-----------|
| <b>UNIDAD DIDÁCTICA 1. TÉCNICAS Y ESTÁNDARES PARA EL DESARROLLO DE COMPONENTES .....</b>     | <b>9</b>  |
| 1. Especificaciones funcionales para el desarrollo de componentes .....                      | 11        |
| 2. Técnicas de optimización de consultas y acceso a grandes volúmenes de información.....    | 21        |
| RECUERDA.....  | 35        |
| Preguntas de Autoevaluación.....   | 37        |
| <b>UNIDAD DIDÁCTICA 2. EL LENGUAJE PROPORCIONADO POR LOS SISTEMAS ERP-CRM .....</b>          | <b>39</b> |
| 1. Características y sintaxis del lenguaje.....  | 41        |
| 1.1. Características del lenguaje y su clasificación.....                                    | 41        |
| 1.2. Sintaxis del lenguaje.....  | 43        |
| 2. Declaración de datos. Estructuras de programación.....                                    | 45        |
| 2.1. Declaración de datos.....   | 45        |
| 2.2. Estructuras de programación .....   | 49        |
| RECUERDA.....  | 51        |
| Preguntas de Autoevaluación.....   | 53        |
| <b>UNIDAD DIDÁCTICA 3. DEFINICIÓN DE LOS LENGUAJES DE PROGRAMACIÓN ..</b>                    | <b>55</b> |
| 1. Sentencias del lenguaje .....   | 57        |
| 2. Entornos de desarrollo y herramientas de desarrollo en sistemas ERP y CRM .....           | 58        |
| RECUERDA.....  | 65        |
| Preguntas de Autoevaluación.....   | 67        |
| <b>UNIDAD DIDÁCTICA 4. DEFINICIÓN DE LA BASE DE DATOS.....</b>                               | <b>69</b> |
| 1. Definición de la base de datos y estructura de tablas de un sistema ERP .....             | 71        |
| RECUERDA.....  | 81        |
| Preguntas de Autoevaluación.....   | 83        |
| <b>UNIDAD DIDÁCTICA 5. ANÁLISIS FUNCIONAL .....</b>  | <b>85</b> |
| 1. División de las actividades del ERP en módulo .....                                       | 87        |
| 2. Trazabilidad entre los módulos.....   | 92        |
| RECUERDA.....  | 95        |
| Preguntas de Autoevaluación.....   | 97        |
| <b>UNIDAD DIDÁCTICA 6. PROGRAMACIÓN EN SISTEMAS ERP Y CRM .....</b>                          | <b>99</b> |
| 1. Generación de programas de extracción de datos entre sistemas (batch inputs) .....        | 101       |
| 1.1. Herramientas para la carga y extracción de datos de sistemas de almacén de datos... 101 |           |

|  |            |
|--|------------|
| 1.2. Creación de extractores de datos .....  | 109        |
| 2. Extracciones de informaciones contenidas en sistemas ERP-CRM, procesamiento de datos .. | 118        |
| RECUERDA.....  | 125        |
| Preguntas de Autoevaluación.....   | 127        |
| <b>UNIDAD DIDÁCTICA 7. BIBLIOTECA DE FUNCIONES BÁSICAS .....</b>                           | <b>129</b> |
| 1. Definición de funciones .....   | 131        |
| 2. Definición de librerías de funciones (API) .....  | 136        |
| RECUERDA.....  | 139        |
| Preguntas de Autoevaluación.....   | 141        |
| <b>UNIDAD DIDÁCTICA 8. DOCUMENTACIÓN .....</b>   | <b>143</b> |
| 1. Documentación del análisis funcional .....  | 145        |
| 2. Documentación de las librerías y funciones.....   | 149        |
| RECUERDA.....  | 153        |
| Preguntas de Autoevaluación.....   | 155        |
| <b>UNIDAD DIDÁCTICA 9. PRUEBAS Y DEPURACIÓN DE UN PROGRAMA .....</b>                       | <b>157</b> |
| 1. Validación de programas .....   | 159        |
| 2. Manejo de errores .....   | 168        |
| RECUERDA.....  | 177        |
| Preguntas de Autoevaluación.....   | 179        |
| <b>ACTIVIDADES PRÁCTICAS .....</b>   | <b>181</b> |
| Actividad Práctica RP2 .....   | 183        |
| <b>RESPUESTAS A LAS PREGUNTAS DE AUTOEVALUACIÓN .....</b>                                  | <b>185</b> |

# UD1 Técnicas y estándares para el desarrollo de componentes



**UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM**



# 1. Especificaciones funcionales para el desarrollo de componentes

Los principales componentes de un sistema de almacén de datos son los siguientes:

- **Sistema ETL (Extraction, Transformation, Load):** realiza las funciones de extracción de las fuentes de datos, transformación y carga del AD, realizando:
  - Extracción de los datos.
  - Filtrado de los datos: limpieza, consolidación, etc.
  - Carga inicial del almacén: ordenación, agregaciones, etc.
  - Refresco del almacén: operación periódica que propaga los cambios de las fuentes externas al almacén de datos.
- **Repositorio propio de datos:** información relevante, metadatos.
- **Interfaces y gestores de consulta:** permiten acceder a los datos y sobre ello se conectan herramientas más sofisticadas, por ejemplo OLAP.
- **Sistemas de integridad y seguridad:** se encargan del mantenimiento global, y copias de seguridad.

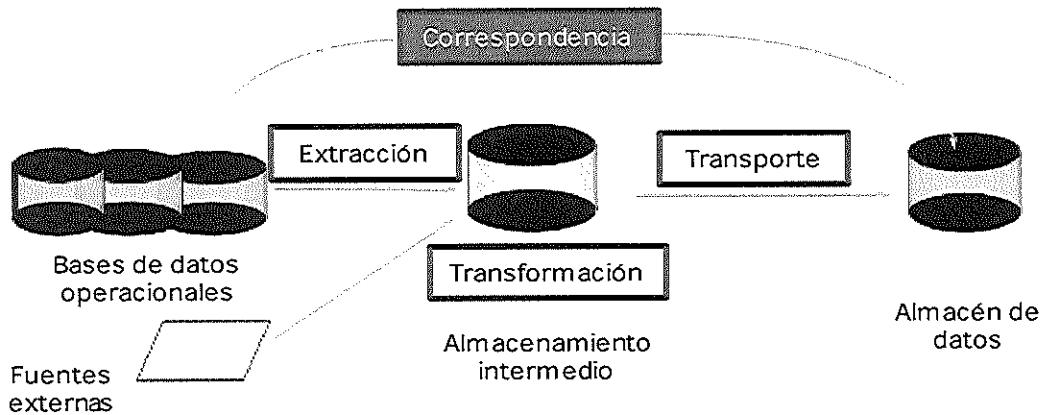
El sistema ETL en español se conoce con las siglas ETT (extracción, Transformación y transporte). Este es el sistema encargado del mantenimiento y almacén de datos, por lo que:

- La construcción del sistema E.T.T. es responsabilidad del equipo de desarrollo del almacén de datos.
- El sistema E.T.T. es construido específicamente para cada almacén de datos. Aproximadamente el 50% del esfuerzo.
- En la construcción del E.T.T. se pueden utilizar herramientas del mercado o programas diseñados específicamente.

Entre las funciones que cumple este sistema cabe destacar:

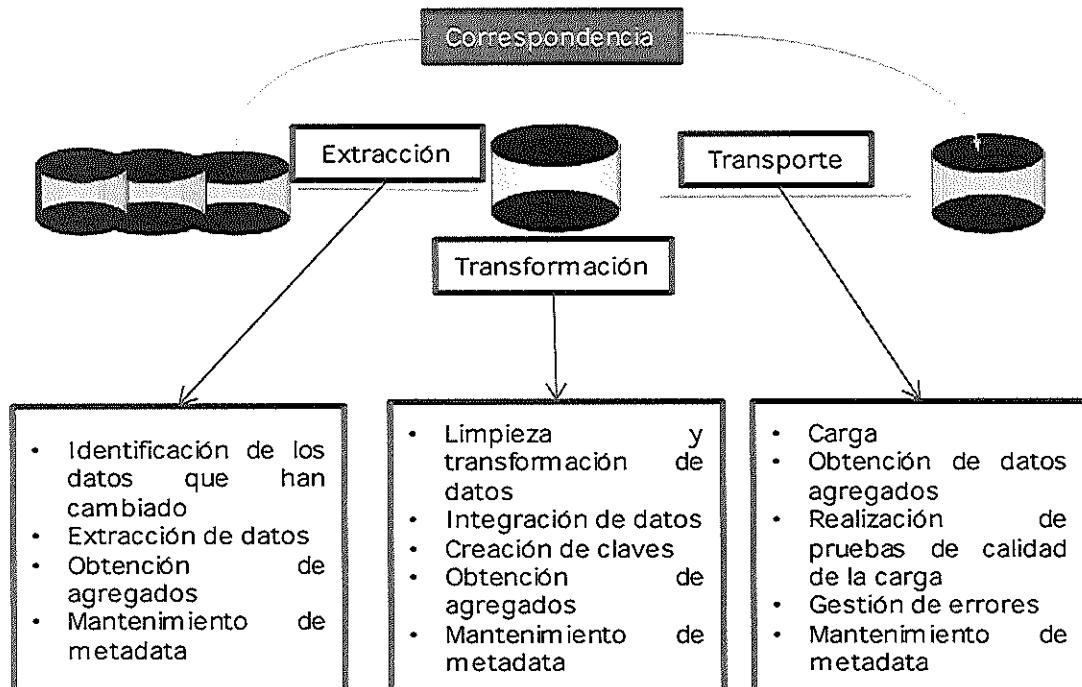
- **Carga inicial:** initial load.

- **Mantenimiento o refresco periódico (refreshment):**  
inmediato, diario, semanal, mensual etc.



El almacenamiento intermedio permite:

- Realizar transformaciones sin paralizar las bases de datos operacionales y el almacén de datos.
- Almacenar metadatos.
- Facilitar la integración de fuentes externas.

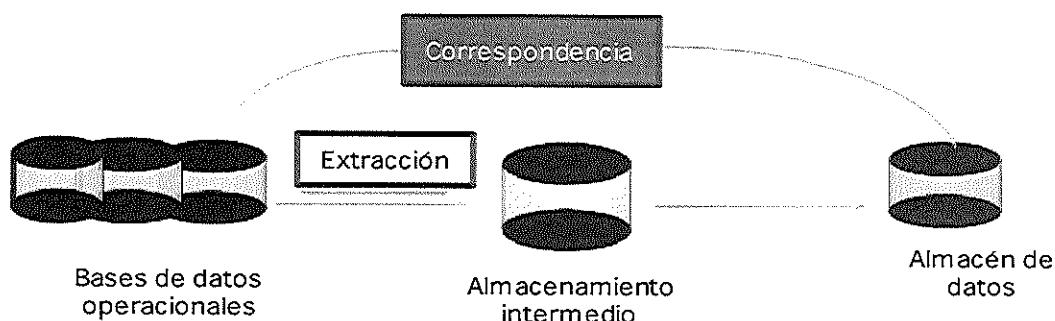


La calidad de los datos es la clave del éxito de un almacén de datos.

Para definir una estrategia de calidad:

- Actuación sobre los sistemas operacionales: modificar las reglas de integridad, los disparadores y las aplicaciones de los sistemas operacionales.
- Documentación de las fuentes de datos.
- Definición de un proceso de transformación.
- Nombramiento de un responsable de calidad del sistema.

## **Extracción**



En la extracción se produce la lectura de datos del sistema operacional, ya sea durante la carga inicial o durante la etapa de mantenimiento del almacén de datos.

La ejecución de la extracción:

- Cuando los datos operaciones están mantenidos en un SGBDR, la extracción de datos se puede reducir a consultas en SQL o rutinas programadas.
- Cuando los datos operacionales están en un sistema propietario o en fuentes externas textuales, la extracción puede ser más complicada y puede tener que realizarse a partir de informes o volcados de datos proporcionados por los propietarios que deberán ser procesados posteriormente.

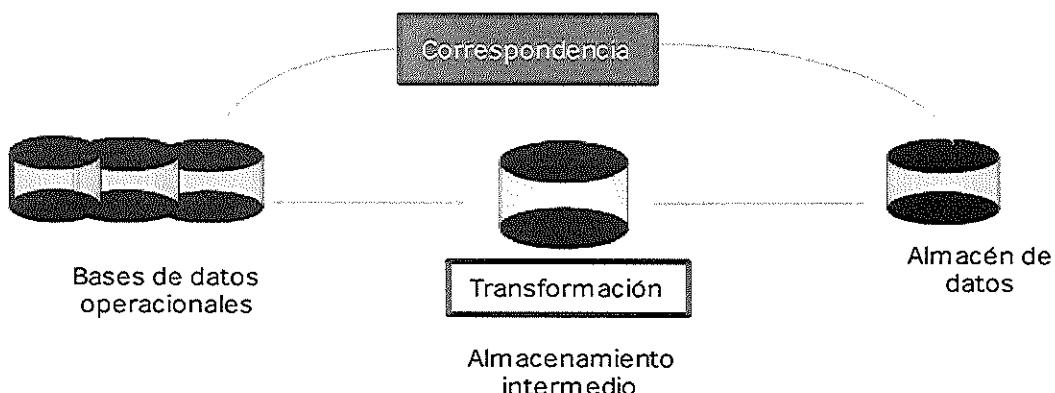
Durante el mantenimiento del almacén de datos se requiere precisión en la extracción de los datos, y para ello es oportuno realizar la identificación de los cambios.

La **identificación de los cambios** se realiza al detectar aquellos datos operacionales, o relevantes, que han podido sufrir una modificación desde la fecha del último mantenimiento.

Para ello existen diferentes métodos:

- Carga total: cada vez que empieza de cero.
- Comparación de instancias de la base de datos operacional.
- Uso de marcas de tiempo en los registros del sistema operacional.
- Uso de disparadores en el sistema operacional.
- Uso del fichero de gestión de transacciones del sistema operacional.
- Uso de técnicas mixtas.

### Transformación



La transformación se refiere al cambio de los datos extraídos de las fuentes operacionales (limpieza, estandarización, etc.) y al proceso de cálculo de los datos derivados al aplicar las leyes de derivación.

En los datos operacionales pueden existir anomalías, desde desarrollos independientes a lo largo del tiempo, fuentes heterogéneas, etc.

Es necesario e importante eliminar dichas anomalías, mediante alguno de los siguientes procesos:

- **Limpieza de datos:** eliminar datos, corregir, completar datos, eliminar duplicados.
- **Estandarización:** codificación, formatos, unidades de medida, etc.

## Transporte

La fase de transporte o de carga consiste en mover los datos desde las fuentes operacionales o el almacenamiento intermedio hasta el almacén de datos y cargar los datos en la correspondiente estructura de datos.

El proceso de carga puede consumir mucho tiempo, en la carga inicial del almacén de datos se mueven grandes volúmenes de datos. En los mantenimientos que se realizan de forma periódica del almacén de datos se mueven pequeños volúmenes de datos.

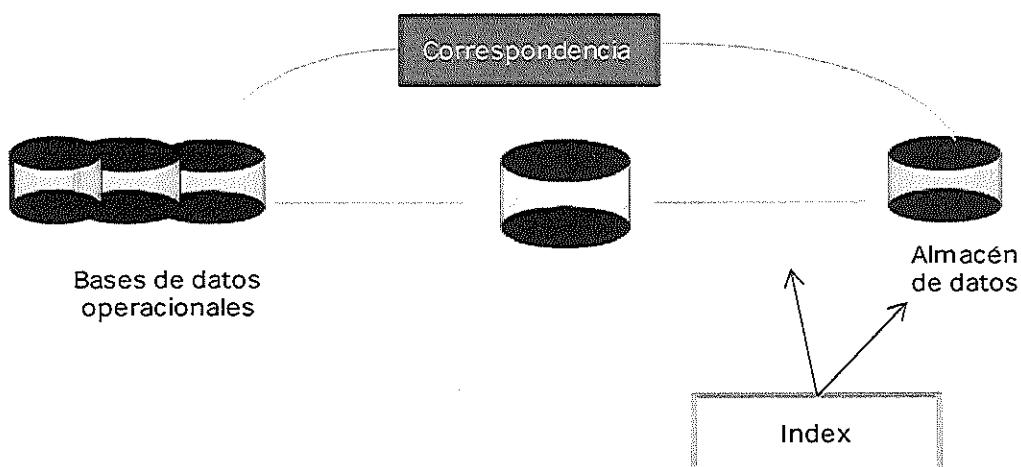
La frecuencia del mantenimiento periódico está determinada por el gránulo del almacén de datos y los diferentes requisitos de los usuarios.

El primer paso es la creación y mantenimiento de la base de datos, posteriormente se definen los intervalos fijos de tiempo a los que añadir cambios al almacén de datos. Se deben determinar las “ventanas de carga” más convenientes para no saturar la base de datos operacional.

Ocasionalmente se pueden archivar o eliminar datos obsoletos que ya no interesa para el análisis.

Posteriormente, tras la carga viene el proceso de indexación.

- **Durante la carga:** la carga con el índice habilitado y el proceso de tupla a tupla, es un proceso lento.
- **Después de la carga:** se produce la carga con el índice deshabilitado y se produce la creación del índice de forma total o parcial.

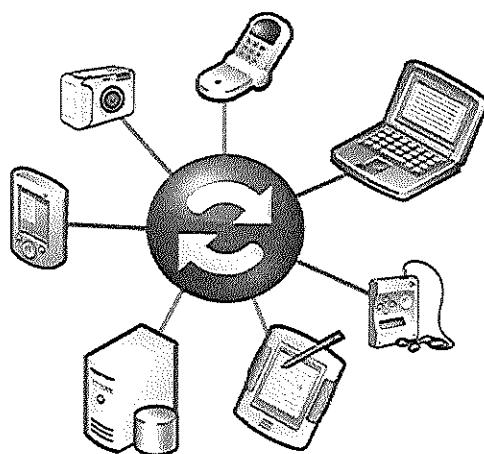


Finalmente se realiza la obtención de agregados.

- Durante la extracción.
- Despues de la carga o transporte.

## **Identificación de orígenes de datos para la carga de datos**

Un sistema de almacén de datos puede verse como una jerarquía de datos, que tiene su origen en los datos almacenados en los sistemas operacionales y que termina con los datos almacenados en el almacén de datos.



El repositorio de datos operacionales es la fuente donde se encuentran los datos primitivos, actuales e integrados, por lo tanto es el encargado de suministrar datos al sistema, estos datos operacionales pueden ser:

- Procedentes de sistemas mainframe.
- Datos de estaciones de trabajo o servidores privados.
- Sistemas externos como las bases de datos comerciales, de proveedores o clientes, o incluso de Internet.
- Datos departamentales almacenados en sistemas propietarios.

El gestor de carga es el encargado de la extracción y la carga de los diferentes datos que se encuentran en el repositorio de datos, además de realizar algunas transformaciones simples a los datos con el fin de que estén adaptados a las necesidades del almacén de datos.

La **carpeta de origen** es la carpeta del servidor en la que se encuentran los archivos de las bases de datos que se van a transferir. Estos datos como se explicó con anterioridad pueden tener diferentes estructuras, y ello marcará en consecuencia la forma de trabajar con ello.

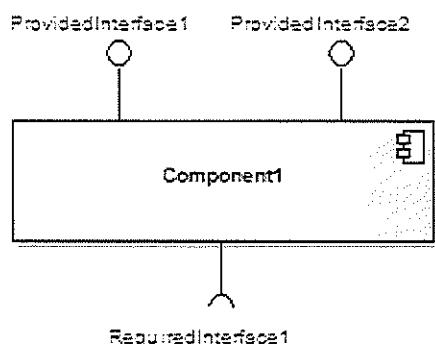
## **Componentes de software para extraer información de un sistema de almacén de datos**

Un **componente** es un elemento de un sistema software que ofrece un conjunto de servicios o funcionalidades, a través de interfaces definidas.

Estos componentes deben cumplir con una serie de características:

- Ser reutilizables.
- Ser intercambiables.
- Poseer interfaces definidas.
- Ser cohesivos.

Estos componentes son la piedra angular de diferentes paradigmas de programación. Esto ha dado lugar a la aparición en el mercado de gran variedad de especificaciones que plantean la forma de construir, utilizar y distribuir componentes.



La **extracción** de información es un tipo de recuperación de la información cuyo objetivo es extraer automáticamente información estructurada o semiestructurada desde documentos legibles del ordenador.

Los principales objetivos del desarrollo de componentes de software ya sea concretamente en sistemas de almacenamiento de datos, o en cualquier otro tipo de sistemas, es en general, reducir el tiempo de trabajo, el esfuerzo que requiere

implementar la aplicación, y los costos del propio proyecto, y de esta forma, incrementar el nivel de productividad de los grupos desarrolladores y minimizar los riesgos globales.

El desarrollo de la descripción de la secuencia de actividades que debe seguirse en un equipo para poder generar un conjunto coherente de productos, en este caso de información es el objetivo del desarrollo de software. Con ello se pretende hacer predecible el trabajo, tanto el costo, como mantener un nivel de calidad.

No existe un único proceso de desarrollo universal. Se ha de configurar en función de la naturaleza del producto y de la experiencia de la empresa. Existen diferentes tipos de aplicaciones:

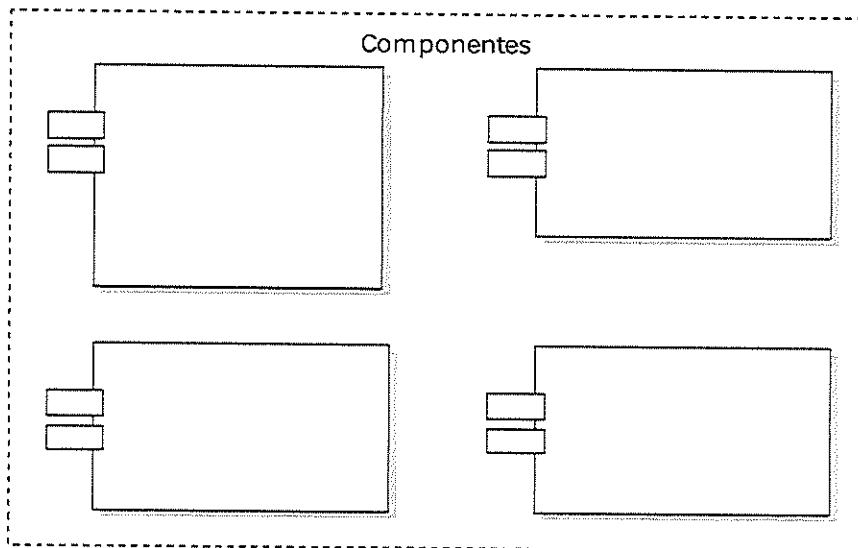
- **Aplicación monoprocesadora:** se realiza en un solo ordenador, no se produce comunicación con otras aplicaciones.
- **Aplicaciones embebidas:** se ejecuta en un entorno automático especial.
- **Aplicaciones de tiempo real:** tiene entre sus especificaciones requerimientos temporales.
- **Aplicaciones distribuidas:** se ejecutan en varios procesadores, por lo que requiere de intercomunicación a través de la red.

### **Modelo de componentes**

Este modelo ilustra los componentes de software que se usarán para construir el sistema. Se pueden construir a partir del modelo de clases y escribir desde cero para el nuevo sistema o se puede importar de otros proyectos y de productos de terceros.

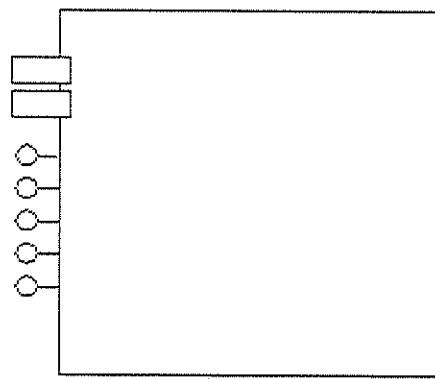
Los componentes son agregaciones de alto nivel de las piezas de software más pequeñas y proveen un enfoque de construcción de bloques de "caja negra" para la elaboración del software.

Los componentes se suelen representar gráficamente como a continuación se detalla, pudiendo ser estos desde controles hasta interfaz de usuario o como servidores de reglas de negocio.

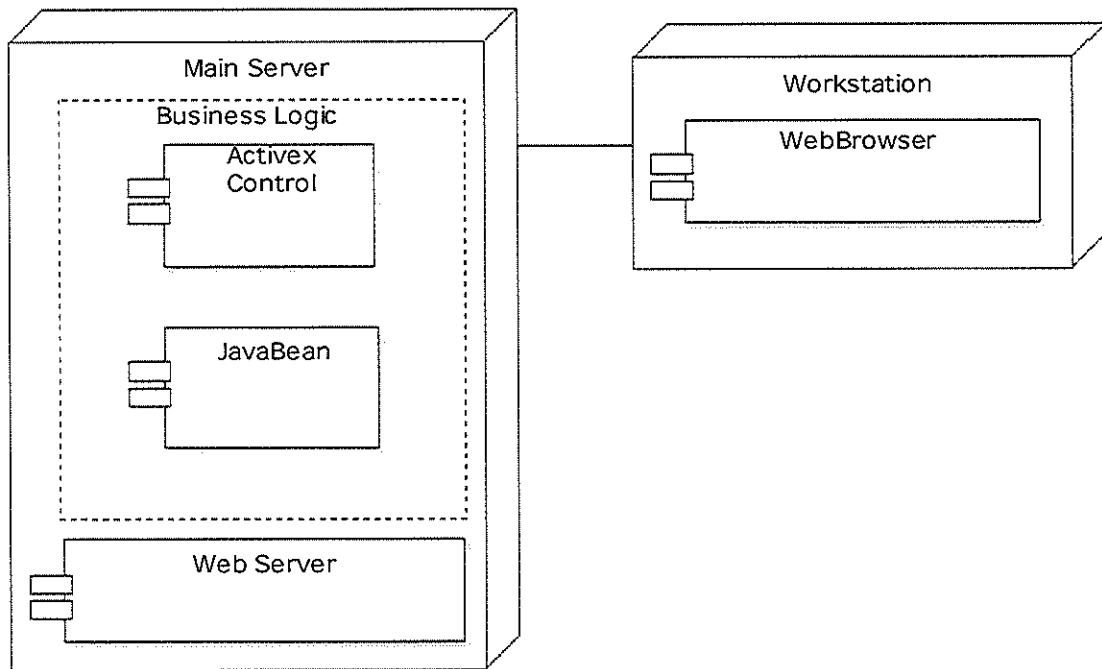


La relación entre los diferentes componentes y el software se puede observar en el diagrama de componentes, sus dependencias, su comunicación así como sus condiciones y la ubicación-

No obstante los componentes pueden exponer las interfaces. Estas son todos los puntos visibles de entrada o los servicios que un componente ofrece y deja disponible a otros componentes de software y clases. Típicamente un componente está compuesto por numerosas clases y paquetes de clases internos. También se puede crear a partir de la colección de componentes más pequeños.



Es en el diagrama de despliegue en el que se puede observar el despliegue físico del sistema en un ambiente de producción o de prueba. Se observa dónde se ubican los diferentes componentes, en qué servidores, máquinas o hardware. Puede representar los enlaces de redes, el ancho de banda de LAN, etc.



Los componentes pueden ir acompañados de ciertos requisitos para indicar sus obligaciones contractuales, es decir, qué servicios son los que proveen en el modelo. Los requisitos suponen una ayuda para poder documentar el comportamiento funcional de los elementos del software.

Además de los requisitos, también pueden verse incluidas ciertas restricciones que indican el entorno o lugar en el que pueden operar. Las pre-condiciones especifican lo que debe ser verdadero antes de que un componente pueda realizar alguna función; las post-condiciones indican lo que debe ser verdadero durante la vida útil del componente.

Las descripciones textuales y procedimentales de las acciones de un objeto a lo largo del tiempo, son los escenarios, y describen la forma en la que un componente trabaja. Se pueden crear múltiples escenarios para describir tanto el camino básico como las excepciones, errores y otras condiciones.

Un componente puede implementar otro elemento del modelo o un componente puede ser implementado por otro elemento. Al emplear las relaciones de realización desde y hacia los componentes, se pueden seguir las dependencias entre los elementos del modelo y la trazabilidad desde los requisitos iniciales hasta la implementación final.

## 2. Técnicas de optimización de consultas y acceso a grandes volúmenes de información

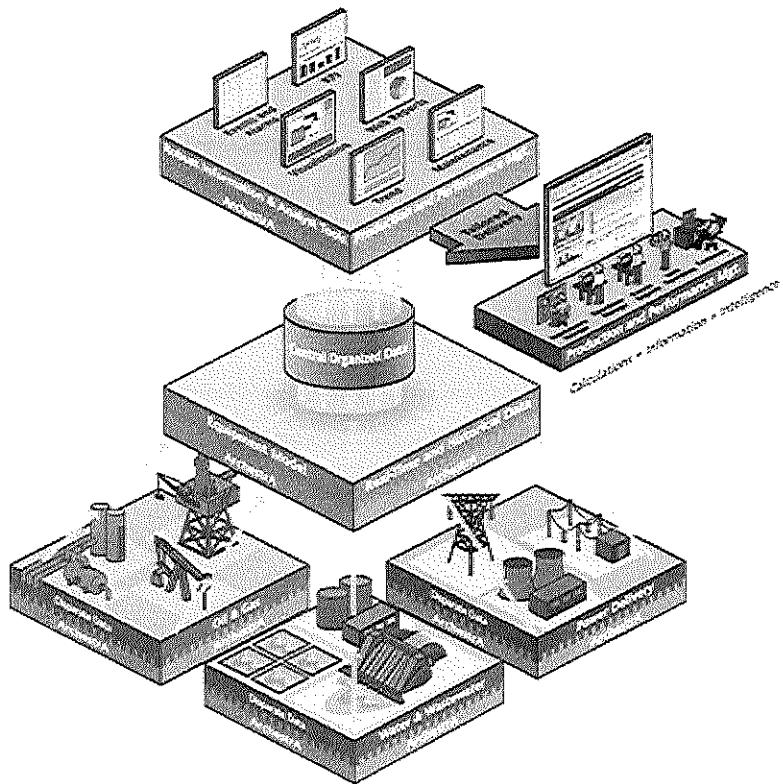
Los almacenes de datos son también conocidos como **Data Warehouse**. Son una colección de datos con las siguientes características:

- **Organizado en torno a temas:** la información se clasifica en base a los aspectos que son de interés para la empresa.
- **Integrado:** es el aspecto más importante. La integración de datos consiste en convenciones de nombres, codificaciones consistentes, medida uniforme de variables, etc.
- **Dependiente del tiempo:** esta dependencia aparece de tres formas:
  - La información representa los datos sobre un horizonte largo de tiempo.
  - Cada estructura clave contiene (implícita o explícitamente) un elemento de tiempo (día, semana, mes, etc.).
  - La información, una vez registrada correctamente, no puede ser actualizada.
- **No volátil:** el Almacén de Datos sólo permite cargar nuevos datos y acceder a los ya almacenados, pero no permite ni borrar ni modificar los datos.

Los almacenes de datos suponen una ayuda en la toma de decisión por parte de la empresa o la organización. Este tipo de almacenes son sobre todo un expediente de una empresa que va más allá de la información transaccional y operacional, almacenado en una base de datos diseñada para favorecer el análisis y la divulgación eficiente de datos.

La función principal que se le otorga a un almacén es que debe entregar la información correcta a la gente indicada en el momento oportuno en el formato deseado. El almacén de datos ofrece respuesta a las diferentes necesidades del usuario, utilizando para ello sistemas de ayuda en la decisión (DSS), sistemas de información Ejecutiva (EIS) o bien herramientas para realizar consulta o informes. Los usuarios finales fácilmente pueden hacer consultas sobre sus almacenes de datos sin tocar o afectar la operación del sistema.

## Estructura



Los cubos de información también son conocidos como **DataMart**, siguen una lógica de los datos en bruto, de los datos provistos por su sistema de operaciones/finanzas hacia el almacén de datos con la adición de nuevas dimensiones o información calculada.

Se les denominada DataMart porque son la representación de un conjunto de datos relacionados con un tema en particular como por ejemplo:

- Ventas.
- Operaciones.
- Recursos humanos.

Todo ello a disposición de los clientes a quienes les pueda interesar.

Esta información puede ser organizada en tablas mediante el uso de tablas dinámicas de MS-Excel o programas personalizados. Las **tablas dinámicas** permiten manipular las vistas de la información con relativa facilidad. Los cubos de información se producen con bastante rapidez. A ellos se les aplican las reglas de seguridad de accesos necesarios.

La información se puede clasificar en:

- Estática.
- Dinámica.

El análisis está basado en las dimensiones y por lo tanto se denomina **análisis multidimensional**.

Por tanto relacionando todo esto con el almacén de memoria, se deduce que un data warehouse es una colección de datos que está formado por dimensiones y variables, entendiendo como dimensiones a aquellos elementos que participan en el análisis y variables a los valores que se desean analizar.

Generalmente, las variables son representadas por valores detallados y numéricos para cada instancia del objeto o evento medido.

Por el contrario, las dimensiones son atributos relativos a las variables, y son utilizadas para ordenar, agrupar o abreviar los valores de las mismas. Las dimensiones poseen una granularidad menor y toman como valores un conjunto de elementos menor que el de las variables.

Las **dimensiones** son atributos relativos a las variables, son las perspectivas de análisis de las variables. Forman parte de las tablas de dimensiones.

Las **variables** son conocidas como indicadores de gestión; son datos que están siendo analizados. Forman parte de la tabla de hecho. Más formalmente, las variables representan algún aspecto cuantificable o medible de los objetos o eventos a analizar.

La **estructura lógica** de un almacén de Datos está compuesta por los siguientes niveles:

- **Metadatos:** describen la estructura de los datos contenidos en el almacén. Están en una dimensión distinta al resto de niveles.
- **Datos detallados actuales:** obtenidos directamente del procesado de los datos. Forman el nivel más bajo de detalle. Ocupan mucho espacio. Se almacenan en disco, para facilitar el acceso.

- **Datos detallados históricos:** igual que los anteriores, pero con datos correspondientes al pasado. Se suelen almacenar en un medio externo, ya que su acceso es poco frecuente.
- **Datos ligeramente resumidos:** primer nivel de agregación de los datos detallados actuales. Corresponden a consultas habituales. Se almacenan en disco.
- **Datos muy resumidos:** son el nivel más alto de agregación. Corresponden a consultas que se realizan muy a menudo y que se deben obtener muy rápidamente. Suelen estar separados del Almacén de datos, formando Supermercados de Datos (Data Marts).

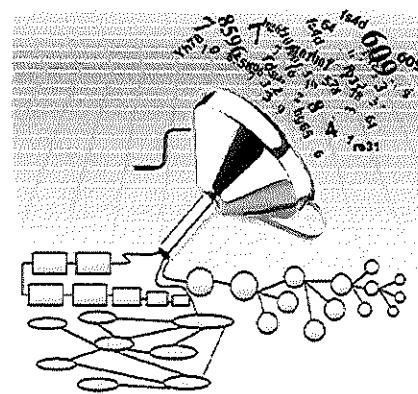
La **estructura física** puede presentar cualquiera de las siguientes configuraciones:

- **Arquitectura centralizada:** todo el almacén de datos se encuentra en un único servidor.
- **Arquitectura distribuida:** los datos del almacén se reparten entre varios servidores. Asignando cada servidor a uno o varios temas lógicos.
- **Arquitectura distribuida por niveles:** refleja la estructura lógica del Almacén, asignando los servidores en función del nivel de agregación de los datos que contienen. Un servidor está dedicado para los datos de detalle, otro para los resumidos y otro para los muy resumidos.

Cuando los datos muy resumidos se duplican en varios servidores para agilizar el acceso se habla de Supermercados de datos (Data Marts).

## **Tipos de estructuras de información y sus relaciones para almacenar información**

La información está constituida por mensajes, es decir, es un conjunto de datos que representan ideas mediante las cuales se incrementa nuestra conciencia, inteligencia o conocimiento. Los mensajes pueden adoptar diferentes manifestaciones físicas. En líneas generales se definen los mensajes como manifestaciones físicas de la información.



Se pueden definir dos tipos generales de información:

- **Continua:** se caracteriza porque sus datos pueden adoptar un número infinito de valores.
- **Discreta:** se caracteriza porque sus datos pueden adoptar sólo un número finito de valores.

Una necesidad básica de la vida moderna es el intercambio de información. La satisfacción de esta necesidad se obtiene con el transporte o envío de la información.

El problema reside en la transmisión de información, el cual se ha resuelto gracias al empleo de sistemas eléctricos de comunicaciones, que representan grandes ventajas sobre otros posibles.

Los sistemas eléctricos de comunicación son aquellos que utilizan dispositivos eléctricos, electromagnéticos u ópticos, o la combinación de éstos, para transmitir información desde donde se produce hasta donde se utiliza.

Para que la información se pueda transmitir a través de un sistema eléctrico de comunicación se debe convertir de su forma física original a energía eléctrica. En esta forma de energía la información se conoce como señal, y el proceso de conversión recibe el nombre de transducción.

Los sistemas de transmisión de datos constituyen un apoyo para los sistemas de cómputo para el transporte de la información que manejan. Sin estos sistemas no se podría haber desarrollado las redes avanzadas de cómputo de procesamiento distribuido, en las que se comparte y transfiere la información de datos entre ordenadores que se encuentran separados a más de 20 metros, incluso en zonas geográficas diferentes.

Las redes de transmisión de datos e información pueden ser sencillas, como puede ser el caso de un ordenador y la conexión con sus periféricos, o pueden ser mucho más complejas pasando por la conexión de punto a punto de larga distancia que se satisface con la utilización de módems, o redes ligeramente más complejas que conectan varias terminales de cómputo de edificios lejanos con el ordenador principal de un centro especializado de datos.

La comunicación de datos presupone mayores requisitos en su red básica que el servicio de señal analógica o de voz para conseguir la transferencia correcta de datos.

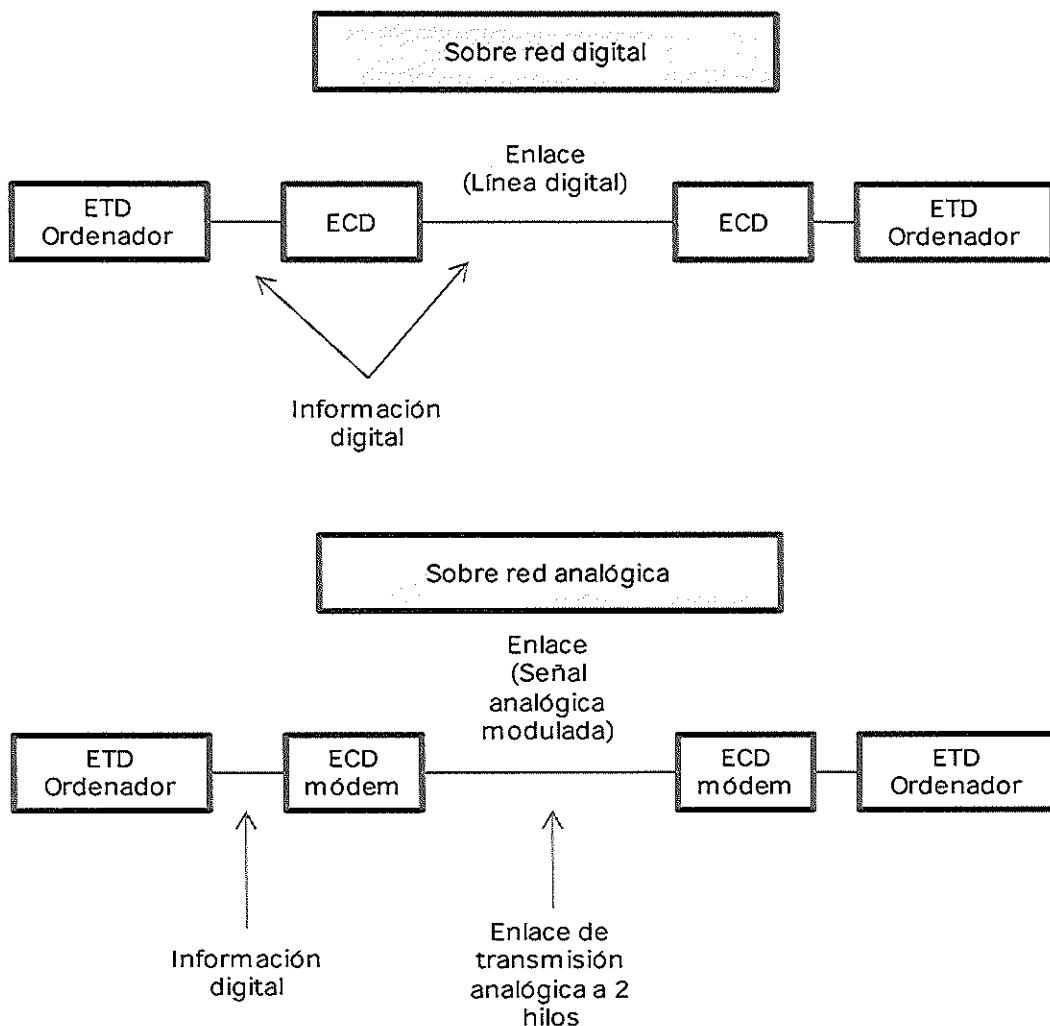
Los ordenadores son considerados inteligentes, pero pese a ello no son seres humanos capaces de realizar juicios, entablar una comunicación organizada con la información apropiada y ordenadamente para representar una sesión coherente y con significado.

La señal de los datos es bastante estricta con el contenido de su información, de tal modo que cualquier error, por pequeño que éste sea puede ser desastroso para la comunicación, es decir, la transmisión de datos debe ser más confiable.

De este modo, dos de las principales medidas adicionales que se deben tomar para satisfacer los mayores requisitos de la transmisión de datos son el control adecuado del flujo de datos durante la transmisión y la codificación para la detección y corrección de errores.

Estos aspectos están incluidos en lo que se conoce como protocolo de comunicación, cuya función es asegurar la comunicación correcta, completa y entendible para los ordenadores.

Las condiciones existentes en la actualidad de las redes de datos, implican la existencia de redes analógicas trabajando al lado de redes digitales, la forma más sencilla de la comunicación de datos entre dos ordenadores se pueden basar en alguna de las siguientes configuraciones que muestra la imagen.



Los ordenadores de ambos extremos se conocen como equipo terminal de datos (ETD) y los equipos de transmisión como equipo terminal de circuito de datos (ECD).

En el caso de la red analógica la transmisión sobre el enlace es digital en tanto que la información con estructura digital del ordenador se debe convertir a la forma apropiada para su transmisión a través de una red analógica. El ECD analógico en este sistema se encarga de esta conversión, es el famoso módem que transmite los datos binarios digitales imponiéndolos sobre una señal portadora de audiofrecuencia.

En el caso de la red digital, el ECD es digital, y por supuesto, con funciones diferentes a las del ECD analógico:

- **En transmisión:** regenerar y convertir la señal del ETD a un formato, nivel y código de línea apropiados para su transmisión sobre la línea digital.

- **En recepción:** establecer el voltaje de referencia que emplea el ETD y reconvertir la señal de línea a la forma apropiada para su aplicación al ETD.

Los ECD digitales pueden suministrar diferentes velocidades digitales de bits que van de 2.4kbits/s, pasando por el canal estándar de 64kbits/s, sistemas de orden superior como 1.544 Mbit/s, 2.048Mbit/s o 45Mbit/s.

El ECD también es conocido como:

- Unidad terminal de red.
- Unidad de servicio de canal.
- Unidad de servicio de datos.
- Unidad terminal de línea.

Otra diferencia importante entre los ECD digitales y analógicos consiste en que, como en la red analógica la tasa de bits de la señal que se recibe no se establece con precisión, en el caso de transmisión analógica el ECD no puede confiar en la red para obtener información precisa de reloj.

Es por esta razón que se necesita un reloj interno del ECD para poder mantener la tasa de bits de transmisión precisa.

En el caso de una red de señal digital, la señal de reloj se recibe desde la red y se deriva de un reloj maestro altamente preciso que funciona como patrón para toda la red de la empresa pública de telecomunicaciones (EPT).

Una **estructura de datos**, o un tipo de datos estructurado, es un tipo de dato construido a partir de otros. Un dato de tipo estructurado está compuesto por una serie de datos de tipos elementales y alguna relación existente entre ellos. Normalmente, la relación suele ser de orden aunque puede ser de cualquier otro tipo.

Se dice que una estructura de datos que es homogénea cuando todos los datos elementales que la forman son del mismo tipo. En caso contrario, se dice que la estructura es heterogénea. Por ejemplo, el tipo de datos complejo es una estructura homogénea, tanto la parte real como la imaginaria se representan con datos reales.

Siempre que se utilice un dato en un programa debe estar determinado su tipo, para que el traductor sepa como debe tratarlo y almacenarlo. En el caso de datos de tipos elementales, el tipo de dato determina el espacio que se utiliza en memoria. Esto, puede no ocurrir si el dato es de un tipo estructurado.

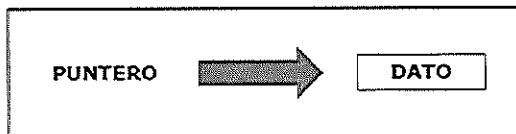
Una estructura de datos que siempre ocupa el mismo espacio en memoria, se dice que es **estática**. Por el contrario, si la memoria asignada a una determinada estructura de datos va variando durante la ejecución del programa, es decir, se realiza una asignación dinámica de memoria, se dice que es una estructura de datos **dinámica**.

### **Estructura de datos estática**

Dentro de los tipos de estructuras de datos que ocupan siempre el mismo espacio en memoria tenemos los punteros, las cadenas y los arrays.

#### *Punteros*

Un puntero es un dato que indica la posición de otro dato. Su utilidad se pone de manifiesto en la construcción de estructuras de datos, ya que son ellos los que proporcionan los lazos de unión entre los elementos que constituyen las estructuras. Son importantes los punteros al principio y al final de la estructura.



Si, ocasionalmente, se necesita que un puntero no señale a ningún dato, se dice que el puntero tiene un valor nulo (puntero nulo).

#### *Cadenas*

Una cadena es una secuencia de caracteres que se interpretan como un dato único. Las cadenas pueden tener longitud fija o variable. La longitud de la cadena se indica tanto por el número de caracteres que contiene ésta, indicado al principio de la misma, como por un carácter especial denominado fin-de-cadena. Sobre datos de tipo cadena se pueden realizar las siguientes operaciones:

- **Concatenación:** consiste en formar una cadena a partir de dos ya existentes, juntando los caracteres de ambas.

- **Extracción de subcadena:** permite formar una cadena a partir de otra ya existente. La subcadena se forma tomando un tramo consecutivo de la cadena inicial.
- **Comparación de cadenas:** es posible comparar dos cadenas. Se considera menor aquella en que el primer carácter en que difieren ambas es menor.
- **Obtención de la longitud:** la longitud de una cadena es un dato de tipo entero, cuyo valor es el número de caracteres que contiene ésta.

### Arrays

El *array* (también llamado formación o matriz), es la estructura de datos más usual. Existe en todos los lenguajes de programación y en algunos es de las pocas estructuras de datos existentes (BASIC y FORTRAN).

Un array es una estructura de datos formada por una cantidad fija de datos del mismo tipo, cada uno de los cuales tiene asociado uno, o más índices, que determinan de forma única la posición del dato en el array.

Podemos imaginar un array como una estructura de celdas donde se pueden almacenar valores. En la figura podemos ver una matriz de un sólo índice que toma valores de 1 a 7.

| ELEMENTO 4 |   |   |                      |   |   |   |
|------------|---|---|----------------------|---|---|---|
| 1          | 2 | 3 | 4                    | 5 | 6 | 7 |
|            |   |   | ****<br>****<br>**** |   |   |   |

En el array de la figura siguiente utilizamos dos índices con valores entre 1 y 3, el primero, y entre 1 y 5, el segundo. Cada elemento de esta matriz está representado por un par ordenado de números, el valor de los dos índices.

| ELEMENTO 1,4 |   |   |       |   |
|--------------|---|---|-------|---|
| 1            | 2 | 3 | 4     | 5 |
| 1            |   |   | ***** |   |
| 2            |   |   |       |   |
| 3            |   |   |       |   |

En general, al número de índices del array se le denomina número de dimensiones del array. La dimensión de la formación está dada por los valores máximos de los índices y, el número total de elementos es el producto de estos valores máximos. En los dos ejemplos anteriores, el número de dimensiones de los arrays son 1 y 2, las dimensiones son (7) y (3; 5) y los números totales de elementos son 7 y 15, respectivamente.

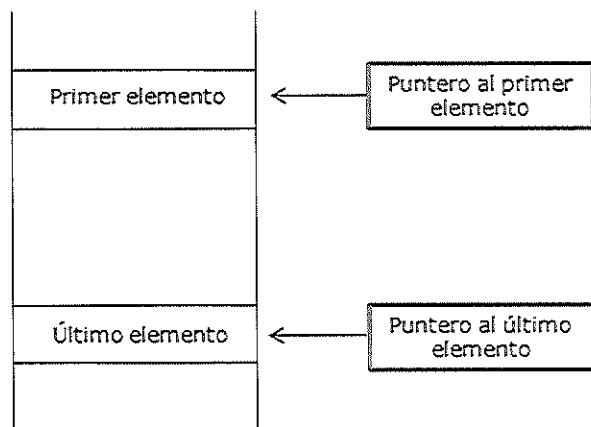
La principal operación que se puede realizar con los arrays es la selección, que consiste en especificar un elemento determinado del array. Esta operación se efectúa dando un valor para todos y cada uno de los índices del array. Con el elemento seleccionado se pueden realizar las operaciones propias de su tipo. Así, con cada elemento de un array real, una vez seleccionado, se pueden realizar las operaciones definidas para datos de tipo real (operaciones aritméticas).

### Estructura de datos dinámica

Como ya se ha dicho, este tipo de estructuras ocupan un espacio en memoria que va evolucionando según el tamaño que dicha estructura vaya adquiriendo. Las estructuras de dinámicas que vamos a estudiar son: colas, pilas, listas encadenadas y árboles.

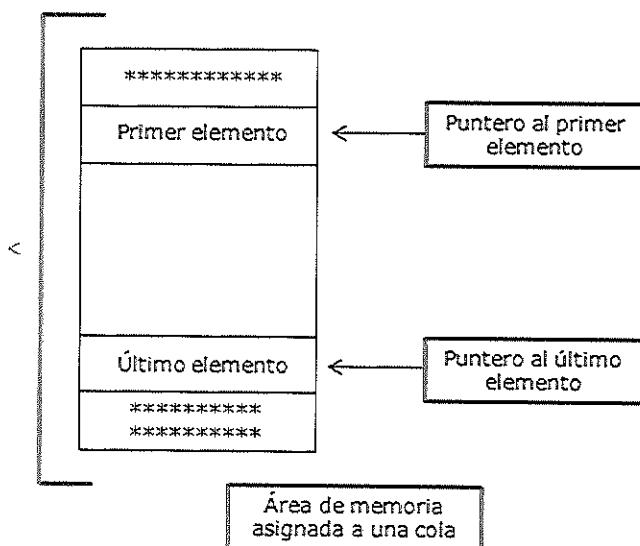
#### *Colas (FIFO)*

Una cola es una estructura de datos en la que el primer dato en entrar es el primer dato en salir. Es decir, es una estructura FIFO (*First In First Out*). Todo el mundo conocemos como funciona una cola, los nuevos se ponen al final, los servicios se prestan al principio y no está permitido "colarse". Las mismas reglas se aplican a las colas de datos almacenadas en la memoria de un computador.



Hay varias formas de implementar una cola en la memoria de un computador. Una forma simple consiste en almacenar los datos en posiciones de memoria adyacentes y utilizar punteros para el principio y el fin de la cola. Cuando un elemento se añade a la cola, el puntero de la parte posterior se ajusta para que señale al nuevo elemento. De manera similar, cuando un elemento se elimina de la cola, se ajusta el puntero delantero para que señale al nuevo primer elemento.

El problema de este método para implementar las colas es que las posiciones de memoria que ocupan, varían a medida que se añaden y eliminan elementos de la misma. La solución habitual consiste en asignar un área fija para almacenar la cola y permitir que se mueva en este área de manera circular. Un área de almacenamiento de esta forma se denomina buffer circular, y puede apreciarse en la figura siguiente.



Entre las aplicaciones que tienen las colas se encuentran el almacenamiento de datos en camino, entre un procesador y un periférico, o actuar como punto intermedio en las redes de comunicación de datos.

### *Pilas (LIFO)*

Una pila es una colección ordenada de datos a los que sólo se puede acceder por un extremo, denominado tope o cima de la pila. La pila es una estructura en la que el último elemento en entrar será el primero en salir, es decir, es lo que se denomina estructura LIFO (*Last In First Out*).

Podemos comparar esta estructura con una pila de platos colocada sobre un muelle. Cuando se añade un nuevo plato en lo alto de la pila, los demás bajan, cuando se retira un plato de la pila, los demás suben.

Igual que en el caso de las colas, van a existir dos punteros, uno que indica la posición tope de la pila, denominado puntero de pila, y otro que señala su base, denominado base de pila, y que mantiene el mismo valor mientras existe la pila. Cuando la pila está vacía el puntero de pila tiene el mismo valor que la base de pila.

La pila es una de las estructuras más importantes en computación. Se usa en cálculos, para pasar de un lenguaje de computador a otro y, para transferir el control de una parte del programa a otra.

Las operaciones que se pueden realizar tanto con las colas como con las pilas, son las siguientes:

- **Añadir o eliminar un elemento:** si es una cola podremos añadirlo o eliminarlo al final de la misma, y si es una pila al principio.
- **Acceder al primer elemento:** normalmente es el único al que se va a poder acceder directamente.
- **Acceder al elemento siguiente del último procesado:** este es el mecanismo normal de acceso tanto a colas como a pilas.
- **Saber si está vacía:** están vacías si no contienen ningún elemento.

#### *Listas encadenadas*

Una lista es un conjunto ordenado de datos. Los elementos de la lista pueden insertarse o eliminarse en cualquier punto de la misma, por lo que es menos restrictiva que una pila o una cola.

La forma más sencilla de implementar una lista es hacer uso de un puntero que señale desde un dato al siguiente. También hay un puntero que señala al primer elemento de la lista, mientras que para el último se emplea un puntero nulo.

Una estructura de este tipo se denomina lista encadenada. Cada elemento de la lista consiste en una parte de datos y un puntero.

Una variación sobre la idea de una lista es el caso en el que el puntero del final de la lista señale al primer elemento. Esto crea lo que se denomina una lista circular.

Si los elementos de la lista están en orden alfabético o numérico, dichas listas se conocen como listas ordenadas.

### *Árboles*

Un árbol es una estructura que implica una jerarquía, en la que cada elemento está unido a otros bajo él. Cada dato en un árbol es un nodo de dicho árbol. El nodo más alto se denomina **raíz**. Cada nodo puede estar conectado a uno o más subárboles, que también responden a la estructura de un árbol. Un nodo, en la parte inferior, del que no cuelgue ningún subárbol se denomina nodo terminal u hoja.

Un tipo especial de árboles muy usados en computación son los árboles binarios. En ellos, de cada nodo pueden colgar, a lo más, dos subárboles, denominados subárbol derecho y subárbol izquierdo, que también son árboles binarios.

La forma usual de representar los árboles supone el uso de punteros. En un árbol binario cada nodo está constituido por una parte de datos y dos punteros. Uno, o ambos punteros, pueden tener un valor nulo si del nodo no cuelgan subárboles.

Son muy utilizados en informática. Las partes de muchos programas se enlazan como si se tratara de árboles. Los árboles se utilizan para representar operaciones aritméticas, y en búsquedas y ordenaciones.

## RECUERDA

- Los principales componentes de un sistema de almacén de datos son los siguientes:
  - Sistema ETL (Extraction, Transformation, Load).
  - Repositorio propio de datos.
  - Interfaces y gestores de consulta.
  - Sistemas de integridad y seguridad.
- Entre las funciones que cumple el sistema ETL cabe destacar:
  - Carga inicial.
  - Mantenimiento o refresco periódico.
  - Extracción: En la extracción se produce la lectura de datos del sistema operacional, ya sea durante la carga inicial o durante la etapa de mantenimiento del almacén de datos.
  - Transformación: cambio de los datos extraídos de las fuentes operacionales (limpieza, estandarización, etc.) y al proceso de cálculo de los datos derivados al aplicar las leyes de derivación.
  - Transporte: consiste en mover los datos desde las fuentes operacionales o el almacenamiento intermedio hasta el almacén de datos y cargar los datos en la correspondiente estructura de datos.
- Un componente es un elemento de un sistema software que ofrece un conjunto de servicios o funcionalidades, a través de interfaces definidas.
- Los almacenes de datos son también conocidos como Data Warehouse. Son una colección de datos con las siguientes características:
  - Organizado en torno a temas.
  - Integrado: convenciones de nombres, codificaciones consistentes, medida uniforme de variables, etc.
  - Dependiente del tiempo.
  - No volátil.
- La función principal que se le otorga a un almacén de debe entregar la información correcta a la gente indicada en el momento oportuno en el

formato deseado.

- Las dimensiones son atributos relativos a las variables, son las perspectivas de análisis de las variables. Forman parte de las tablas de dimensiones.
- Las variables son conocidas como indicadores de gestión; son datos que están siendo analizados.
- La estructura lógica de un almacén de Datos está compuesta por los siguientes niveles:
  - Metadatos.
  - Datos detallados actuales.
  - Datos detallados históricos.
  - Datos ligeramente resumidos.
  - Datos muy resumidos.
- La estructura física puede presentar cualquiera de las siguientes configuraciones:
  - Arquitectura centralizada.
  - Arquitectura distribuida.
  - Arquitectura distribuida por niveles.
- Una estructura de datos es un tipo de dato construido a partir de otros.
- Una estructura de datos que siempre ocupa el mismo espacio en memoria, se dice que es estática. Por el contrario, si la memoria asignada a una determinada estructura de datos va variando durante la ejecución del programa, es decir, se realiza una asignación dinámica de memoria, se dice que es una estructura de datos dinámica.

## Preguntas de Autoevaluación

### 1. Completa el espacio en blanco del siguiente enunciado:

"Los sistemas de \_\_\_\_\_ y \_\_\_\_\_, se encargan del mantenimiento global y copias de seguridad".

- a) Integridad y seguridad.
- b) Extracción y transporte.
- c) Seguridad y extracción.

### 2. En la extracción...:

- a) Se detectan los datos operacionales.
- b) Se produce la lectura de datos del sistema operacional.
- c) Se produce el cambio de datos extraídos de las fuentes operacionales.

### 3. El repositorio de datos operacionales es la fuente donde se encuentran los datos primitivos, actuales e integrados, por lo tanto es el encargado de suministrar datos al sistema. Estos datos operacionales pueden ser: **(Respuesta múltiple)**

- a) Datos de servidores públicos.
- b) Datos de staciones de trabajo o servidores privados.
- c) Procedentes de sistemas mainframe.

**4. Los Data Warehouse se caracterizan por: (Respuesta múltiple)**

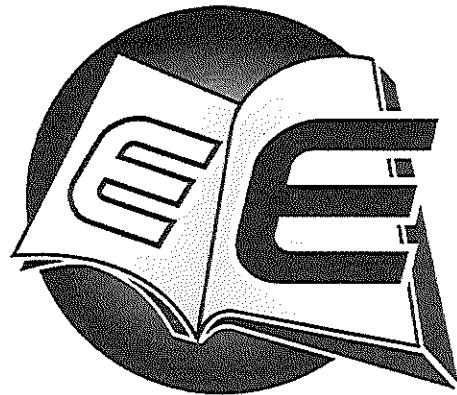
- a) Ser volátil.
- b) Estar organizado en torno a temas.
- c) No permitir borrar ni modificar datos.

**5. Indica si es verdadero o falso el siguiente enunciado:**

*"Se pueden definir dos tipos generales de información: continua y discontinua".*

- a) Verdadero.
- b) Falso.

# UD2 El lenguaje proporcionado por los sistemas ERP- CRM



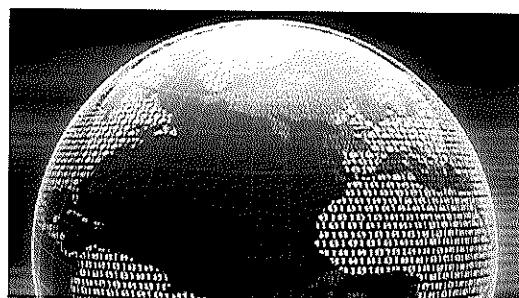
**UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM**



# 1. Características y sintaxis del lenguaje

## 1.1. Características del lenguaje y su clasificación

Tanto el software de sistemas como el de aplicaciones están escritos en esquemas codificados y se les conoce como **lenguajes de programación**.



Un lenguaje de programación es una notación o conjunto de símbolos y caracteres combinados entre si de acuerdo con una sintaxis ya definida que posibilita la transmisión de instrucciones a la CPU.

La **principal función** de un lenguaje de programación es proporcionar instrucciones al sistema de la computadora para que pueda realizar una actividad de procesamiento.

Profesionales especializados en sistemas informáticos trabajan con el lenguaje de programación, el cual consiste en grupos de símbolos y reglas utilizados para escribir códigos de programas. La programación incluye la traducción de lo que quiere lograr un usuario a un código que pueda comprender y ejecutar la computadora. Al igual que al escribir un informe o un ensayo en español, la escritura de un programa de computación en un lenguaje de programación requiere que el programador siga un grupo de reglas. Cada lenguaje de programación utiliza un grupo de símbolos que tiene un significado específico. Cada lenguaje también tiene su propio grupo de reglas, o la sintaxis del lenguaje. Dicha sintaxis establece cómo deben combinarse los símbolos en enunciados capaces de hacer llegar instrucciones significativas al CPU.

En la actualidad podemos encontrar diferentes tipos de lenguajes de programación con los que trabajar (Java, PHP, C++, C#, etc.); por lo tanto, las

funcionalidades van a depender del tipo de lenguaje de programación al que estemos haciendo referencia.

## Clasificación de los lenguajes de programación

Los lenguajes de programación son clasificados de formas muy diversas, como se describe a continuación.

### Clasificación según nivel de abstracción

- **Lenguajes de bajo nivel.** Son aquellos que por sus características se encuentran más próximos a la arquitectura de la máquina, englobándose en este grupo el lenguaje máquina y el lenguaje ensamblador.
- **Lenguajes de medio nivel.** Hay lenguajes de programación que son considerados por algunos expertos como lenguajes de medio nivel (como es el caso del lenguaje C) al tener ciertas características que los acercan a los lenguajes de bajo nivel pero teniendo, al mismo tiempo, ciertas cualidades que lo hacen un lenguaje más cercano al humano y, por tanto, de alto nivel.
- **Lenguajes de alto nivel.** Son aquellos lenguajes que por sus características se encuentran más próximos al usuario o programador y se consideran como tales el resto de los lenguajes de programación como por ejemplo BASIC, COBOL, Pascal, etc. Una de las características más importantes de estos lenguajes es que, a diferencia de los lenguajes de alto nivel, son independientes de la arquitectura del ordenador utilizado como soporte, lo que implica que los programas desarrollados en lenguajes de alto nivel pueden ser ejecutados sobre ordenadores con distinto microprocesador. Por otro lado, cabe destacar una mayor facilidad en el desarrollo, depuración y mantenimiento de los programas frente a los desarrollados con lenguajes de bajo nivel.

### Clasificación según su paradigma

- **Paradigma imperativo.** Describe la programación como una secuencia instrucciones o comandos que cambian el estado de un

programa. El código máquina en general está basado en el paradigma imperativo. Su contrario es el paradigma declarativo. En este paradigma se incluye el paradigma procedimental (procedural) entre otros.

- **Paradigma declarativo.** No se basa en el cómo se hace algo (cómo se logra un objetivo paso a paso), sino que describe (declara) cómo es algo. En otras palabras, se enfoca en describir las propiedades de la solución buscada, dejando indeterminado el algoritmo (conjunto de instrucciones) usado para encontrar esa solución. Es más complicado de implementar que el paradigma imperativo, tiene desventajas en la eficiencia, pero ventajas en la solución de determinados problemas.

### **Clasificación según la forma de ejecución**

- **Lenguajes compilados.** Un programa traductor traduce el código del programa (código fuente) en código máquina (código objeto). Otro programa, el enlazador, unirá los ficheros de código objeto del programa principal con los de las librerías para producir el programa ejecutable. Ejemplo: C.
- **Lenguajes interpretados.** Un programa (intérprete), ejecuta las instrucciones del programa de manera directa. Ejemplo: Lisp.

## **1.2. Sintaxis del lenguaje**

A la forma visible de un lenguaje de programación se le conoce como **sintaxis**. La mayoría de los lenguajes de programación son puramente textuales, es decir, utilizan secuencias de texto que incluyen palabras, números y puntuación, de manera similar a los lenguajes naturales escritos. Por otra parte, hay algunos lenguajes de programación que son más gráficos en su naturaleza, utilizando relaciones visuales entre símbolos para especificar un programa.

La sintaxis de un lenguaje de programación describe las combinaciones posibles de los símbolos que forman un programa sintácticamente correcto. El significado que se le da a una combinación de símbolos es manejado por su semántica (ya sea formal o como parte del código duro de la referencia de implementación).

La sintaxis de los lenguajes de programación es definida generalmente utilizando una combinación de expresiones regulares (para la estructura léxica) y la Notación de Backus-Naur (para la estructura gramática). Este es un ejemplo de una gramática simple, tomada de Lisp:

```

expresión ::= átomo | lista
átomo      ::= número | símbolo
número     ::= [+ -] ? ['0'-'9']+ 
símbolo    ::= ['A'-'Z'] ['a'-'z'] *
lista       ::= '(' expresión ')'

```

Con esta gramática se especifica lo siguiente:

- Una expresión puede ser un átomo o una lista.
- Un átomo puede ser un número o un símbolo.
- Un número es una secuencia continua de uno o más dígitos decimales, precedido opcionalmente por un signo más o un signo menos.
- Un símbolo es una letra seguida de cero o más caracteres (excluyendo espacios).
- Una lista es un par de paréntesis que abren y cierran, con cero o más expresiones en medio.

Algunos ejemplos de secuencias bien formadas de acuerdo a esta gramática:

'12345', '()', '(a b c232 (1))'

No todos los programas sintácticamente correctos son semánticamente correctos. Muchos programas sintácticamente correctos tienen inconsistencias con las reglas del lenguaje; y pueden (dependiendo de la especificación del lenguaje y la solidez de la implementación) resultar en un error de traducción o ejecución. En algunos casos, tales programas pueden exhibir un comportamiento indefinido. Además, incluso cuando un programa está bien definido dentro de un lenguaje, todavía puede tener un significado que no es el que la persona que lo escribió estaba tratando de construir.

El siguiente fragmento en el lenguaje C es sintácticamente correcto, pero ejecuta una operación que no está definida semánticamente (dado que p es un apuntador nulo, las operaciones p->real y p->im no tienen ningún significado):

```

complex *p = NULL;
complex abs_p = sqrt (p->real * p->real + p->im * p->im);

```

Si la declaración de tipo de la primera línea fuera omitida, el programa dispararía un error de compilación, pues la variable "p" no estaría definida. Pero el programa sería sintácticamente correcto todavía, dado que las declaraciones de tipo proveen información semántica solamente.

La gramática necesaria para especificar un lenguaje de programación puede ser clasificada por su posición en la **Jerarquía de Chomsky**. La sintaxis de la mayoría de los lenguajes de programación puede ser especificada utilizando una gramática Tipo-2, es decir, son gramáticas libres de contexto. Algunos lenguajes, incluyendo a Perl y a Lisp, contienen construcciones que permiten la ejecución durante la fase de análisis. Los lenguajes que permiten construcciones que permiten al programador alterar el comportamiento de un analizador hacen del análisis de la sintaxis un problema sin decisión única, y generalmente oscurecen la separación entre análisis y ejecución. En contraste con el sistema de macros de Lisp y los bloques BEGIN de Perl, que pueden tener cálculos generales, las macros de C son meros reemplazos de cadenas, y no requieren ejecución de código.

## 2. Declaración de datos. Estructuras de programación

### 2.1. Declaración de datos

Se llama "dato" a la información que procesan las sentencias de un programa. En programación estructurada, todos los datos que maneje nuestro programa deben especificarse explícitamente indicando la clase de información que va a almacenar.

A la especificación de los atributos de un dato en un programa, siguiendo la sintaxis concreta de un lenguaje, se la denomina **declaración del dato** en el programa. A continuación se describen los atributos que se especifican de un dato cuando realizamos una declaración.

Hay que aclarar que no todos los atributos son especificados explícitamente por el programador sino que algunos van implícitos en la declaración. Señalaremos con un (\*) los atributos que especifica formalmente el programador.

### **Nombre (\*)**

Es una etiqueta formada por caracteres alfanuméricos que identifica al dato. La referencia al dato en el programa se hace usualmente a través del nombre. El nombre no siempre es suficiente para referenciar una variable. Por ejemplo, en lenguajes como C, C++ o Pascal podemos llamar con el mismo nombre a variables en distintos subprogramas y entonces es el ámbito de la variable el que las distingue. También veremos que, si una función es recursiva, por cada llamada a ésta se crean nuevas variables locales de la función que se referencian por el mismo nombre y es la pila del programa la que determina las variables instanciadas que están activas en cada momento.

Existen restricciones sobre el tipo de cadenas de caracteres que pueden constituir el nombre de una variable. Estas restricciones dependen del lenguaje utilizado. Por ejemplo, en C y C++ los nombres deben comenzar por un carácter alfabético o por el símbolo de subrayado. Está prohibido que los nombres comiencen por un carácter numérico o por un símbolo de operación. Así, los nombres 'x', 'x1', 'suma' son permitidos, en cambio '1x', '+suma', no lo están. Además estos lenguajes diferencian entre mayúsculas y minúsculas, de tal manera que los nombres 'mi\_variable' y 'Mi\_variable' son válidos y distintos. Es aconsejable poner nombres con significado a las variables, de tal manera que estén relacionados con el dato que contienen.

### **Espacio de memoria**

Es el lugar que se le asigna en la memoria RAM para contener al valor de la variable en el momento de la declaración. El tamaño de este espacio viene determinado por el tipo de dato y se le llama normalmente celda. El espacio de memoria es único y diferente para cada dato y es la característica que lo identifica únicamente.

### **Valor (\*)**

Normalmente el valor de un dato suele asignarse en la declaración del mismo. A este acto se le llama **inicialización del dato**. Es normalmente incorrecto

utilizar en el programa un dato no inicializado ya que contiene como valor algo desconocido. Algunos lenguajes, como Pascal se preocupan de ¡metizar automáticamente a unos valores por defecto, otros como C y C++ no lo hacen, con lo que dentro del espacio de memoria nos encontraremos la configuración de bits que existía en la RAM cuando se reservó ese espacio como almacenamiento de una variable.

El tamaño finito de espacio de almacenamiento que se asigna a un dato, restringe el rango de valores que pueden asignarse al mismo.

### **Tipo (\*)**

Este concepto surge de un proceso de abstracción de la noción de dato. La naturaleza de los datos que manejamos en un computador es matemática, lógica o simbólica; de este modo, hay datos numéricos, lógicos y caracteres que tienen un significado en un lenguaje. Los datos numéricos se caracterizan por su forma de escribirlos (escribimos un número entero de la forma habitual y un número real con un punto para indicar los decimales) y por las operaciones que podemos hacer sobre ellos (sumar, restar,...). Abstrayendo esta idea, realmente cualquier dato tiene una representación y unas operaciones que pueden hacerse con él. Estas dos cualidades son las que usaremos para especificar un modelo de datos en un lenguaje de programación que nos permita "hacer como si" utilizáramos realmente estos conceptos matemáticos.

Definimos, pues, tipo de dato como la especificación del conjunto de valores permitidos para ese dato y el conjunto de operaciones permitidas sobre el mismo. Esta definición nos lleva a dos puntualizaciones:

- El conjunto de valores permitido lo determina el número de bytes que cada compilador asigna para almacenar un dato de un tipo concreto. Esto permite una abstracción en la representación en memoria del dato, de tal manera que si compilamos el mismo programa con diferentes compiladores de un mismo lenguaje, el rango de valores con que podemos jugar para un mismo tipo de dato puede ser diferente.
- La operación definida sobre un tipo es una ley de composición interna; es decir, da como resultado un dato del mismo tipo. Así está prohibido realizar una operación "raíz cuadrada" sobre un dato de tipo entero, ya que no es una operación propia de este tipo. Es

posible que el compilador la acepte, pero internamente transformará el tipo de datos a uno que permita esta operación, generalmente un double).

Para los lenguajes imperativos como C y C++, hay que especificar el tipo en el momento de la declaración de una variable. Esto es debido a que el tipo es el indicador, en el momento de la declaración, del número de bytes que ocupa en memoria dicha variable.

### **Ámbito**

Es la región del código de un programa sobre la cual un dato está definido. Los datos cuyo ámbito sea todo el código de un programa se denominan **datos globales**. Su declaración se realiza en la cabecera del programa. Los datos con una región de validez menor se denominan datos locales. El ámbito de una variable local es el código del subprograma (función o procedimiento) donde está definida, y su declaración como variable está en la cabecera del código correspondiente al subprograma. El ámbito de un dato implementa la idea de encapsulación de los datos dentro de los módulos. Según esta filosofía, la descomposición de un problema en módulos implica que cada módulo es una unidad funcional que realiza una tarea y toda la información que maneja de manera exclusiva para realizar esta tarea debe ser inaccesible para el resto de módulos (también se dice que la información está encapsulada en el módulo). Como los módulos acaban siendo procedimientos o funciones, las variables locales a esos módulos están encapsuladas en ellos y no son accesibles por los restantes procedimientos o funciones del programa.

### **Dirección**

Todo dato de tipo simple tiene una dirección que es la dirección del primer byte que forma la celda de memoria preparada para contener el valor del dato. Esto es cierto al menos mientras la sentencia que se ejecuta pertenece al conjunto de sentencias de las estructuras de selección en programación ámbito de dicha variable.

### **Persistencia**

Es el tiempo que una variable está presente en la memoria respecto del tiempo de ejecución del programa. Como veremos más adelante, las variables

tienen diferente persistencia dependiendo del tipo que sean. Así hay variables que sólo existen mientras está en ejecución la función que las declaró y otras como las globales o las locales estáticas que existen mientras el programa está en ejecución. Aunque esta propiedad parece equivalente al concepto de ámbito, hay detalles que la diferencian.

## 2.2. Estructuras de programación

Las instrucciones dentro de un programa se ejecutan una a una en el orden en que están escritas. A esto se le llama **Ejecución Secuencial**.

Las estructuras de programación son las siguientes:

- **Estructuras de selección.** En una estructura de selección/decisión, el algoritmo al ser ejecutado toma una decisión, ejecutar o no ciertas instrucciones si se cumplen o no ciertas condiciones. Las condiciones devuelven un valor, verdadero o falso, determinando así la secuencia a seguir. Por lo general los lenguajes de programación disponen de dos estructuras de este tipo: estructura de decisión simple (if), y estructura de decisión múltiple (CASE, SWITCH).
- **Estructuras de repetición.** La estructura de repetición o bucle hace posible la ejecución repetida de una o más instrucciones. Las estructuras de repetición nos permiten ejecutar varias veces unas mismas líneas de código. Estas estructuras describen procesos que se repiten varias veces en la solución del problema. El conjunto de acciones que se repiten conforman el cuerpo del bucle y cada ejecución del cuerpo del bucle se denomina iteración.



## RECUERDA

- Un lenguaje de programación es una notación o conjunto de símbolos y caracteres combinados entre si de acuerdo con una sintaxis ya definida que posibilita la transmisión de instrucciones a la CPU.
- La principal función de un lenguaje de programación es proporcionar instrucciones al sistema de la computadora para que pueda realizar una actividad de procesamiento.
- Los lenguajes de programación se clasifican según:
  - El nivel de abstracción.
  - El paradigma.
  - La forma de ejecución.
- A la forma visible de un lenguaje de programación se le conoce como sintaxis.
- La sintaxis de un lenguaje de programación describe las combinaciones posibles de los símbolos que forman un programa sintácticamente correcto. El significado que se le da a una combinación de símbolos es manejado por su semántica.
- La sintaxis de los lenguajes de programación es definida generalmente utilizando una combinación de expresiones regulares (para la estructura léxica) y la Notación de Backus-Naur (para la estructura gramática).
- No todos los programas sintácticamente correctos son semánticamente correctos. Muchos programas sintácticamente correctos tienen inconsistencias con las reglas del lenguaje; y pueden resultar en un error de traducción o ejecución.
- La gramática necesaria para especificar un lenguaje de programación puede ser clasificada por su posición en la Jerarquía de Chomsky. La sintaxis de la mayoría de los lenguajes de programación puede ser especificada utilizando una gramática Tipo-2, es decir, son gramáticas libres de contexto. Algunos lenguajes, incluyendo a Perl y a Lisp, contienen construcciones que permiten la ejecución durante la fase de análisis.
- Se llama "dato" a la información que procesan las sentencias de un

programa. En programación estructurada, todos los datos que maneja nuestro programa deben especificarse explícitamente indicando la clase de información que va a almacenar.

- A la especificación de los atributos de un dato en un programa, siguiendo la sintaxis concreta de un lenguaje, se la denomina declaración del dato en el programa. A continuación se describen los atributos que se especifican de un dato cuando realizamos una declaración.
- No todos los atributos son especificados explícitamente por el programador sino que algunos van implícitos en la declaración.
  - Los atributos que especifica formalmente el programador son:
    - Nombre.
    - Valor.
    - Tipo.
  - Los atributos que no especifica formalmente el programador son:
    - Espacio de memoria.
    - Ámbito.
    - Dirección.
    - Persistencia.
- Las instrucciones dentro de un programa se ejecutan una a una en el orden en que están escritas. A esto se le llama Ejecución Secuencial.
- Las estructuras de programación son las siguientes:
  - Estructuras de selección.
  - Estructuras de repetición.

## Preguntas de Autoevaluación

### 1. Completa el espacio en blanco del siguiente enunciado:

"En la actualidad podemos encontrar diferentes tipos de lenguajes de programación con los que trabajar, como son Java, \_\_\_\_\_, C++, y C#".

- a) PHP
- b) C\*\*
- c) HPH

### 2. La clasificación del lenguaje según su forma de ejecución engloba a...: (Respuesta múltiple)

- a) Lenguajes codificados.
- b) Lenguajes compilados.
- c) Lenguajes interpretados.

### 3. ¿Qué es la Jerarquía Chomsky? (Respuesta múltiple)

- a) Un lenguaje de programación.
- b) La gramática necesaria para especificar un lenguaje de programación.
- c) Un tipo de clasificación según su posición jerárquica.

### 4. Indica si es verdadero o falso el siguiente enunciado:

"Se llama 'dato' a la información que procesan las sentencias de un programa.".

- a) Verdadero.
- b) Falso.

**5. La clasificación de las estructuras de programación son: (Respuesta múltiple)**

- a) Estructuras de selección.
- b) Estructuras de repetición.
- c) Estructuras de ejecución.

# UD3 Definición de los lenguajes de programación



**UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM**



## 1. Sentencias del lenguaje

En programación, una **sentencia** es una línea de código en algún lenguaje de programación. Un programa está constituido por múltiples sentencias de programación, lo que es llamado **código fuente**.



Un algoritmo de programación está constituido por una o más sentencias de programación.

Las sentencias de programación suelen tener algún carácter que determina su final, por lo general es un punto y coma (;) o un punto final (.), y algunas están separadas simplemente por enteros (retorno de carro). Esto depende del lenguaje de programación que se esté empleando. Existen lenguajes que no necesitan un carácter que determine el final de una sentencia.

En algunos lenguajes de programación las sentencias están numeradas, de esta manera, si existe algún error de sintaxis (o alguna advertencia), el compilador entrega un mensaje con el número de sentencia donde fue encontrado.

Una sentencia de programación tiene una **sintaxis** y una **semántica**. La sintaxis está determinada por el lenguaje de programación y, si existe algún error, el compilador suele detectarlo. También una sentencia tiene una semántica, si existe algún error de semántica, en general, no puede ser descubierto por el compilador. Esto es porque la semántica está asociada al "significado" o "función" de esa sentencia (o un grupo de sentencias), pero el compilador no puede determinar qué quiere hacer el programador.

Un **snippet** es un segmento de una o más sentencias de programación, y permite reutilizar códigos, hacer códigos más eficientes o facilitar el trabajo al programador.

## Clasificación de las sentencias del lenguaje

Dada la definición anterior, resulta evidente que pueden existir infinitas sentencias distintas, e innumerables criterios para su clasificación. Una de las posibles, que además coincide con la clasificación del Estándar, distingue las siguientes clases de sentencia:

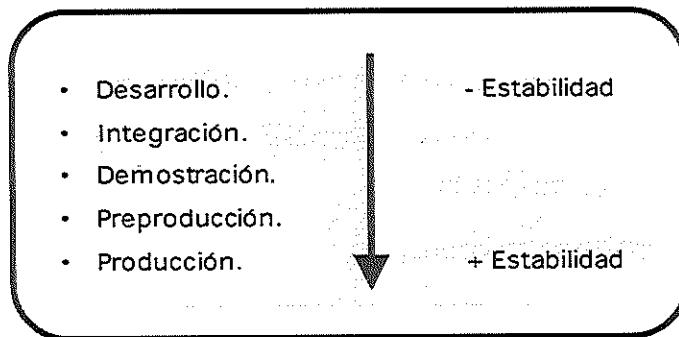
- **De etiqueta.** Existen tres clases de sentencias etiquetadas: las etiquetas directas, las sentencias case y las default, estas últimas se utilizan en conjunción con las sentencias switch.
- **De expresión.** Podríamos decir que son las que no pertenecen a ninguno de los otros grupos y que, en la práctica, son las más abundantes. Generalmente son asignaciones o invocaciones de funciones.
- **Compuestas.** También denominadas bloques se utilizan en aquellas situaciones en que la sintaxis espera una sentencia pero se necesita usar varias.
- **De selección o de control de flujo.** Pueden decidir entre varios cursos de acción distintos en función de ciertos valores.
- **De iteración.** permiten repetir un conjunto de sentencias ejecutando un bucle.
- **De salto.** Permiten transferir el control del programa de forma incondicional.
- **De declaración.** Este tipo de sentencias introducen uno o más identificadores en un bloque.
- **Bloques de intento.** Estas sentencias deben estar seguidas de una sentencia match.

## 2. Entornos de desarrollo y herramientas de desarrollo en sistemas ERP y CRM

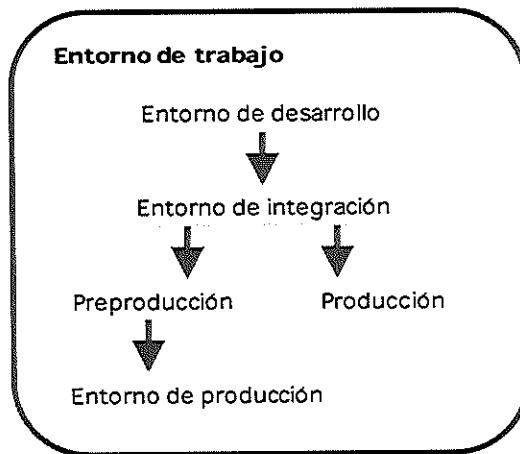
Las empresas disponen habitualmente de varios entornos donde se desarrolla, prueba y utiliza un sistema de información.

Podemos definir un **entorno**, dentro de nuestro marco de trabajo, como la infraestructura necesaria para acometer las tareas específicas requeridas por el producto software, en función del estado en el que se encuentra.

Un producto software puede encontrarse en alguno de estos estados, en un momento dado, en su evolución:



Cada estado define un nivel de estabilidad del software, y la secuencia de estados representa la evolución del producto software desde su fase más temprana (toma de requerimientos) hasta su puesta en funcionamiento (deployment o despliegue en producción).



Estos cinco estados nos definirán los entornos de trabajo que vamos a describir para el desarrollo de nuestras aplicaciones, y cada entorno se centrará en un subconjunto específico de las tareas de creación y mantenimiento de la aplicación.

### **Entornos de desarrollo**

Un entorno de desarrollo integrado, llamado también IDE (sigla en inglés de integrated development environment), es una aplicación visual que sirve para la

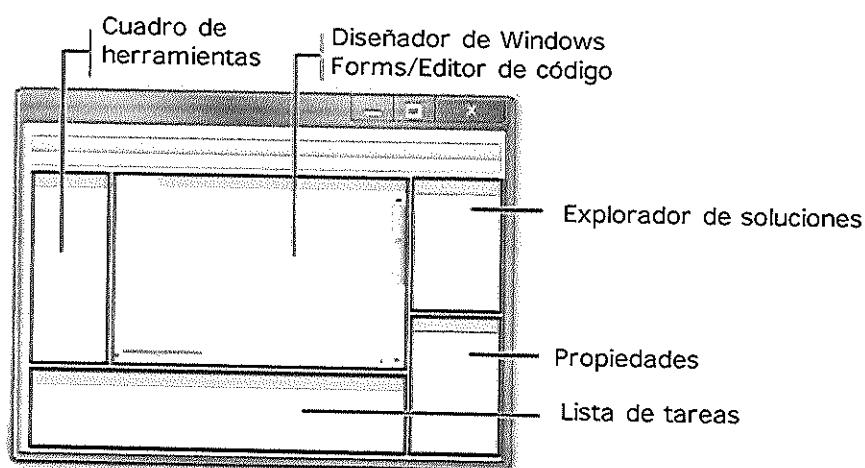
construcción de aplicaciones a partir de componentes. Por lo general todas ellas cuentan con los siguientes elementos:

- Una o más "paletas" para mostrar como iconos los componentes disponibles.
- Un "lienzo" o "contenedor" en el cual se colocan los componentes y se interconectan entre sí.
- Editores específicos para configurar y especializar los componentes.
- Ojeadores (bromen) para localizar componentes de acuerdo a ciertos criterios de búsqueda.
- Directorios de componentes.
- Acceso a editores, interpretes, compiladores y depuradores para desarrollar nuevos componentes.
- Acceso a algunas herramientas de control y gestión de proyectos y CSCW, esenciales para grandes proyectos software.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes.

Ejemplos de IDEs son Visual Studio de Microsoft, VisualAge de IBM o VisualCafe de Symantec, complementados con lenguajes de configuración como VBScript y JavaScript.

Como ejemplo, en la siguiente imagen vemos la representación del entorno integrado (IDE) de visual C#:



## Entorno de pruebas

El entorno de pruebas se divide a su vez en el entorno de integración y entorno de preproducción. Vemos cada uno de ellos.

### *Entorno de integración*

En el entorno de integración se lleva a cabo las siguientes tareas:

- Integración de los distintos módulos que componen la aplicación.
- Pruebas de integración.

En el proceso de integración tiene especial importancia el sistema de control de versiones. Debemos obtener los fuentes etiquetados con la versión estable que deseamos integrar. Cuando los fuentes son obtenidos se compilan y se genera la aplicación con todos los módulos integrados.

Llegados a este punto, debemos poner en funcionamiento la aplicación para someterla a las pruebas de integración. Desplegamos la aplicación basándonos en los documentos generados durante el proyecto. Una vez integrada y activada la aplicación, llevaremos a cabo la secuencia de pruebas. El objetivo de estas pruebas es comprobar el funcionamiento de la aplicación como un todo. En ellas, se trata de probar las funcionalidades que debe cumplir el producto, aunque también debe ser probadas las anteriores funcionalidades para evitar que los cambios introducidos alteren inadvertidamente el comportamiento de las mismas.

Si la aplicación satisface las pruebas de integración, entonces está lista para su paso al entorno de preproducción (para realizar las pruebas de calidad), y al entorno de demostración (para permitir el acceso al cliente).

En el caso en el que se produzca un error en estas pruebas debemos generar una incidencia y emitirla al encargado del módulo en el cual se produjo.

### *Entorno de preproducción*

En el entorno de preproducción se lleva a cabo las pruebas finales de la aplicación antes de su paso final al entorno de producción, donde se pondrá en funcionamiento en un escenario real.

Para ello, en primer lugar se transfiere la aplicación una vez ha sido validada en el entorno de integración, a ser posible de una forma automatizada. El equipo de

calidad, someterá la aplicación a un conjunto exhaustivo de pruebas, de diversos tipos:

- Funcionales y estructurales.
- De rendimiento.
- De tolerancia a fallos.
- De seguridad.

Si estas pruebas, conocidas como de aceptación, resultan satisfactorias, entonces la aplicación está ya lista para su paso al entorno de producción.

### **Entorno de explotación**

El entorno de explotación o también conocido con el nombre de entorno de producción contiene en todo momento la versión activa de la aplicación. Los usuarios finales tienen acceso a la aplicación implantada en este entorno, de modo que resulta indispensable planificar y adoptar las medidas de seguridad oportunas, en consonancia con la importancia de la información que maneja el sistema.

Por otra parte, este entorno también contiene los datos reales, información que es preciso salvaguardar frente a posibles pérdidas mediante la aplicación sistemática de una política de copias de seguridad. También debemos proteger los datos frente a exposición o usos fraudulentos de los mismos, restringiendo su acceso exclusivamente al personal de confianza que administra el sistema.

La aplicación se despliega en el entorno de producción procedente de la versión existente en el entorno de preproducción. La subida debería producirse de forma automática para evitar, en la medida de lo posible, la introducción de errores.

### **Herramientas de desarrollo en sistemas ERP-CRM**

Como en todos los entornos de desarrollos e implantación, es necesario realizar las pruebas del sistema. Ésta es la Fase previa a la puesta en marcha dónde se procede a la integración y aceptación del sistema, comprobando que los procesos definidos funcionan correctamente y que cumplen con lo establecido en las fases previas, antes de dejar de utilizar definitivamente el sistema antiguo en la fase siguiente.

Una vez superada esta fase, pasaremos a la fase final de la implantación. En esta etapa se hace la migración definitiva de los datos, se realiza la formación final de usuarios. Se realiza el arranque en real de la nueva solución.

Es muy importante que cuando empieza a funcionar el ERP, se asegure la posibilidad del funcionamiento del grueso del sistema independientemente del fallo, parada y revisión de alguno de los módulos, aunque sea introduciendo manualmente los datos que precisen el resto de módulos.

El entorno de desarrollo conlleva el abanico más amplio de tareas, que abarca desde el comienzo del ciclo de vida del software (toma de requerimientos) hasta la obtención de una versión mínimamente estable de la aplicación, o de un subconjunto de la misma (módulos).

En este entorno se llevan a cabo las siguientes tareas:

- **Toma de requerimientos:** una vez hemos concretado las necesidades del cliente y hemos establecido varias reuniones con este fin, es aconsejable documentarlo, de manera que todo el equipo pueda tener una visión global del proyecto – requerimientos funcionales. Este documento se podrá ir completando en posteriores entrevistas puesto que nunca quedan resueltas desde un principio las necesidades funcionales del sistema. En esta fase suele ser necesario usar herramientas de documentación y toma de requerimientos.
- **Análisis de arquitectura y Diseño técnico:** estas son dos de las tareas más importantes del desarrollo software y que pueden determinar el éxito o fracaso de un proyecto. Se deben realizar las siguientes funciones: acuerdo inequívoco de los requerimientos especificados, elección de la tecnología que pueda abarcar las necesidades del proyecto con una adecuada arquitectura y el diseño de los componentes, localización de las fases más críticas, aportando soluciones de contención ante posibles problemas que pudieran aparecer. Además necesitaremos una serie de herramientas que nos permitan realizar estas funciones de forma productiva.
- **Implementación:** indica el comienzo de la implementación con la tecnología adecuada elegida en la fase de análisis. Si se ha realizado un buen análisis y modelado técnico se reduce considerablemente la complejidad de la implementación.

- **Pruebas de unidad y módulos:** estas pruebas se deben realizar sobre los módulos y componentes que cada desarrollador vaya finalizando. Se realizan en un entorno local para comprobar su correcto funcionamiento y poder integrarlas para obtener el producto final.

De las anteriores tareas podemos deducir las necesidades tanto software como hardware del entorno de desarrollo, teniendo en cuenta la política de trabajo en equipo que decidamos seguir.

## RECUERDA

- En programación, una sentencia es una línea de código en algún lenguaje de programación. Un programa está constituido por múltiples sentencias de programación, lo que es llamado código fuente.
- Una sentencia de programación tiene una sintaxis y una semántica
- Un snippet es un segmento de una o más sentencias de programación, y permite reutilizar códigos, hacer códigos más eficientes o facilitar el trabajo al programador.
- Las sentencias pueden clasificarse en: de etiqueta, de expresión, compuestas, de selección o de control de flujo, de iteración, de salto, de declaración y bloques de intento.
- Un entorno se define como la infraestructura necesaria para acometer las tareas específicas requeridas por el producto software, en función del estado en el que se encuentra.
- Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación; es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).
- El entorno de pruebas se divide a su vez en el entorno de integración y entorno de preproducción:
  - Entorno de integración: lleva a cabo las tareas de integración de los distintos módulos que componen la aplicación y las pruebas de integración.
  - Entorno de reproducción: se lleva a cabo las pruebas finales de la aplicación antes de su paso final al entorno de producción.
- El entorno de explotación o también conocido con el nombre de entorno de producción contiene en todo momento la versión activa de la aplicación.
- El entorno de desarrollo conlleva el abanico más amplio de tareas, que abarca desde el comienzo del ciclo de vida del software (toma de requerimientos) hasta la obtención de una versión mínimamente estable de la aplicación, o de un subconjunto de la misma (módulos). Dichas tareas son:

- o Toma de requerimientos.
- o Análisis de arquitectura y Diseño técnico.
- o Implementación.
- o Pruebas de unidad y módulos.

## Preguntas de Autoevaluación

### 1. Completa el espacio en blanco del siguiente enunciado:

"En programación, una \_\_\_\_\_ es una línea de código en algún lenguaje de programación.".

- a) Sentencia.
- b) Sintaxis.
- c) Entorno.

### 2. Una sentencia de programación: (Respuesta múltiple)

- a) Tiene una sintaxis y una semántica.
- b) Se compone de snippet.
- c) Es una herramienta útil en el almacenamiento de datos semiestructurados.

### 3. Algunas de las posibles maneras de clasificar las sentencias son: (Respuesta múltiple)

- a) Sintaxis y semántica.
- b) De etiqueta, de expresión, compuestas.
- c) De interacción, de salto y de declaración.

### 4. Un entorno de desarrollo integrado...

- a) Se divide en entorno de integración y entorno de producción.
- b) Contiene en todo momento la versión activa de la aplicación.
- c) Es una aplicación visual que sirve para la construcción de aplicaciones a partir de componentes.

**5. Indica si es verdadero o falso el siguiente enunciado:**

*"Las sentencias de programación suelen tener algún carácter que determina su final, por lo general es un punto y coma (;) o un punto final (.), y algunas están separadas simplemente por enteros".*

- a) Falso.
- b) Verdadero.

# UD4 Definición de la base de datos



UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM



## 1. Definición de la base de datos y estructura de tablas de un sistema ERP

El conjunto unificado de información, resultante de nuestro proyecto informático y, que será compartida por los diferentes usuarios de la organización, va a conformar la denominada **Base de Datos**.

La función básica de una base de datos es permitir el almacenamiento y la recuperación de la información necesaria, para que las personas de la organización puedan tomar decisiones. Es así que las Bases de Datos se tornan esenciales para la supervivencia de cualquier organización; pues los datos estructurados constituyen un recurso básico para todas las organizaciones.

Dependiendo de la capacidad de almacenamiento y procesamiento del hardware, la organización puede contar con una única Base de Datos, o con múltiples Bases de Datos.

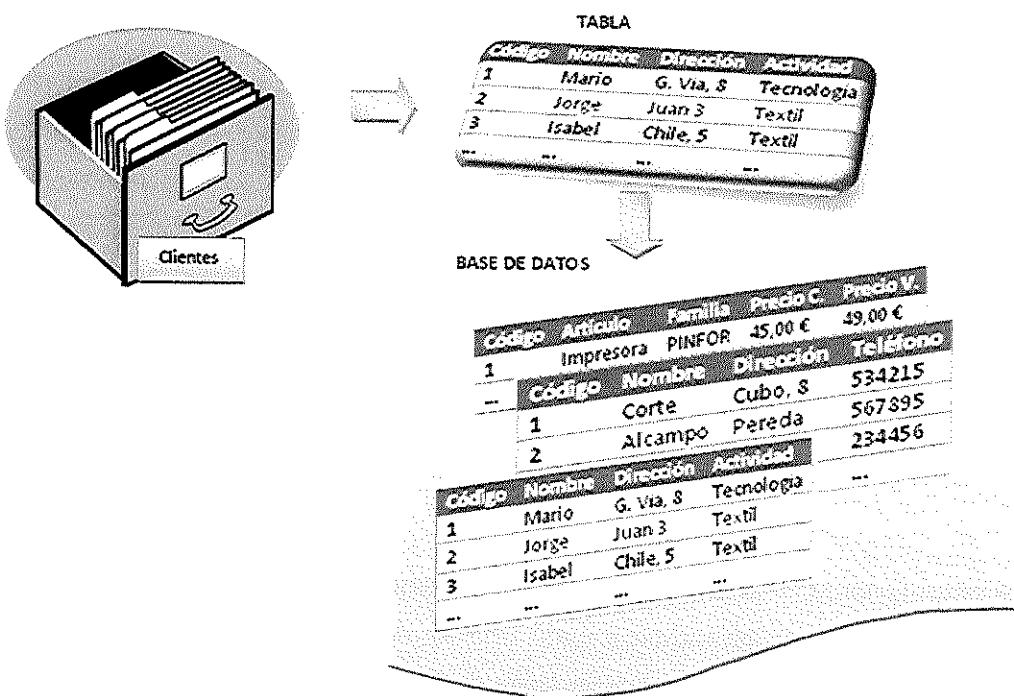
Es común que en las pequeñas y medianas empresas se cuente con microcomputadoras, y por ello tengan que distribuir su información en un conjunto de Bases de Datos; asignándole a cada una de ellas, información sobre cada área específica de la empresa. Un ejemplo sería el de contar con una base de datos para el almacenamiento de la información correspondiente al área financiera, otra para el área de personal, una más para el área de ventas o el área de producción.

Mientras tanto las grandes organizaciones poseen computadoras de gran porte, y es así que pueden almacenar toda la información necesaria, integrada, consistente y consolidada, en una única base de datos.

Independientemente de la Base de Datos que será implementada, ésta necesita de un **Sistema de Gestión de Base de Datos (SGBD o DBMS)**. Los sistemas de Gestión de Base de datos, son programas de software para la administración de las Bases de Datos; y en particular, para: almacenar, manipular y recuperar datos en una computadora. El SGBD también se encargará de la comunicación entre el usuario y la base de datos, proporcionándole al usuario, los medios necesarios para poder obtener información, introducir nuevos datos y actualizar los ya existentes.

## Estructura de una base de datos

Una Base de Datos está compuesta por un conjunto de **tablas** o **archivos**. Tabla en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de hoja de cálculo.



Para una mayor comprensión podemos ejemplificar la siguiente Base de Datos de compras.

### ARCHIVO DE PRODUCTOS

| Código artículo | Descripción del material             | Unidad          | Cantidad |
|-----------------|--------------------------------------|-----------------|----------|
| 1.0.01          | CD-ROM RW IDE                        | Unidad          | 10       |
| 1.01.02         | Disco rígido ATA 66                  | Unidad          | 20       |
| 1.02.01         | Disco Flexible de 3 1/2" 1,44 Mbytes | Caja de 10      | 20       |
| 2.01.01         | Sonido de 16 bit                     | Unidad          | 5        |
| 3.01.01         | Papel carta para impresora.          | Resma 100 hojas | 25       |
| 4.01.01         | Pentium II 200Mhz                    | Unidad          | 7        |
| 4.01.02         | Pentium III 500Mhz                   | Unidad          | 8        |
| 4.01.03         | Pentium III 800Mhz                   | Unidad          | 9        |

## ARCHIVO DE PROVEEDORES

| Código proveedor | Nombre del proveedor | Teléfono del proveedor | Dirección del proveedor |
|------------------|----------------------|------------------------|-------------------------|
| 001              | Inca Tel             | 4923-4803              | Av. La Plata 365        |
| 002              | Infocad              | 4633-2520              | Doblas 1578             |
| 003              | Herrera Compusistem  | 4232-7711              | Av. Rivadavia 3558      |

## ARCHIVO DE ORIGEN DE LOS PRODUCTOS

| Código proveedor | Código del artículo | Precio |
|------------------|---------------------|--------|
| 001              | 11.01.01            | 70,00  |
| 002              | 1.01.01             | 80,00  |
| 003              | 1.01.01             | 75,00  |
| 002              | 2.01.01             | 50     |
| 001              | 4.01.03             | 450    |

Como podemos ver esta base de datos cuenta con tres entidades:

- Datos sobre productos (Entidad producto), almacenados en el archivo de **PRODUCTOS**.
- Datos sobre proveedores (Entidad proveedores), almacenados en el archivo **PROVEEDORES**.
- Datos sobre el origen de los productos (Entidad origen del producto), o sea, los productos son provistos por cada proveedor y viceversa, almacenados en el archivo de **ORIGEN DEL PRODUCTO**.

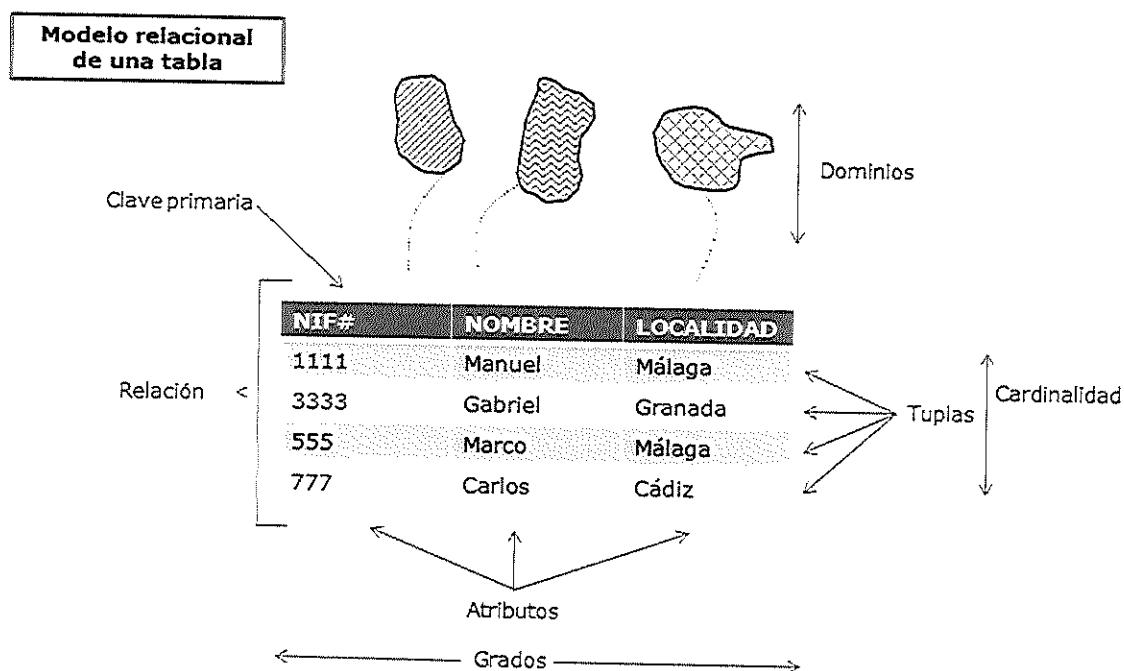
La información almacenada en cada uno de estos archivos se conoce con el nombre de **Entidad**. Por lo tanto una entidad es cualquier persona, cosa o evento, real o imaginario, de interés para la organización y acerca del cual se capturan, almacenan o procesan datos.

Además, cada uno de estos archivos está formado por un conjunto de registros que describe, a través de los atributos o datos (columna), cada entidad en

el almacenado. Un **atributo** es pues, cualquier detalle que sirve para identificar, clasificar, cuantificar o expresar el estado de una entidad.

Todos los registros de un archivo, identificados por las filas de cada tabla, poseen el mismo formato, o sea tienen el mismo conjunto de datos o atributos, identificados por las columnas, que describen a las entidades.

En otras palabras los **registros** están formados por un conjunto de datos almacenados en los campos de cada atributo; y cada registro debe contener el conjunto de atributos necesarios, para describir completamente cada entidad sobre la cual una organización necesita almacenar y obtener información.



Una **tupla** corresponde a una fila de la tabla. Representa cada una de las ocurrencias de la relación (equivale a lo que conocemos como ocurrencia de un registro, en ficheros clásicos). El número de tuplas se denomina **cardinalidad**, y esta varía con el tiempo.

### Tipos de archivos

Los archivos pueden clasificarse en cuatro tipos básicos; que son: los **archivos maestros**, los **archivos de transacciones**, los **archivos de control** y los **archivos de planeamiento**. Esta clasificación dependerá de la relación lógica que tengan que tener los datos, para dar apoyo a la actividad de la organización.

#### *Archivo maestro*

Un archivo maestro es un conjunto de registros que se refieren a algún aspecto importante de las actividades de una organización, como por ejemplo el archivo de VENDEDORES. Un archivo maestro también puede reflejar la historia de los eventos que afectan a una entidad determinada, como es en el caso de un archivo HISTÓRICO DE VENTAS. Otros ejemplos son los archivos maestros de: PLAN DE CUENTAS; BANCOS, NÓMINA DEL PERSONAL, CLIENTES, VENDEDORES, PRODUCTOS, PROVEEDORES, COMPETIDORES.

#### *Archivo de transacciones*

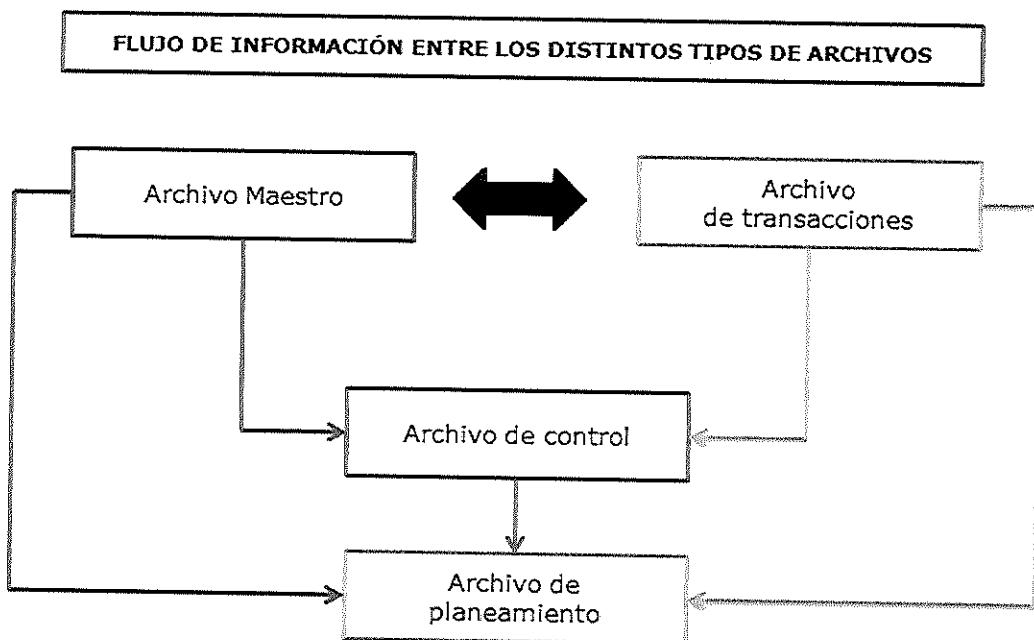
Un archivo de transacciones es un archivo temporal que persigue básicamente dos propósitos; uno es el de acumular datos de eventos en el momento que ocurran, y el segundo propósito es el de actualizar los archivos maestros para reflejar los resultados de las transacciones actuales. En otras palabras, guardan información sobre los eventos que afectan a la organización y sobre los cuales se calculan datos; como es en el caso de los archivos de VENTAS, ORDENES DE PRODUCCIÓN o PAGO DE SALARIOS. Otros ejemplos de archivos de transacciones son los archivos de: REGISTROS CONTABLES, COSTOS, FACTURAS, PAGOS A RECIBIR, PROCESOS DE EXPORTACIÓN, CONSULTA DE CLIENTES, PEDIDOS DE CLIENTES Y PEDIDOS A PROVEEDORES.

#### *Archivos de control*

Los archivos de control contienen datos de los archivos maestros y de transacciones, para permitir el análisis del desempeño de la organización. Estos archivos generan medidas de control de los negocios, como ser el VOLUMEN DE VENTA POR PRODUCTO, VOLUMEN DE VENTA POR VENDEDOR, VOLUMEN DE VENTA POR CLIENTE, COMPRAS POR PROVEEDOR, COSTO DE REPOSICIÓN.

#### *Archivo de planeamiento*

Los archivos de planeamiento, contienen datos referentes a los niveles esperados de los datos existentes en los archivos maestros y de transacciones; como por ejemplo: PROGRAMA DE VENTAS, PROGRAMA DE COMPRAS, PROGRAMA DE PRODUCCIÓN; PRESUPUESTO FINANCIERO. Por lo tanto los datos existentes en un archivo de planeamiento provienen de los archivos maestros, de transacciones, y de control.



### **Clave primaria o identificadora**

Cada instancia de una entidad debe ser únicamente identificable, de manera tal que cada registro de la entidad debe estar separado y ser únicamente identificable del resto de los registros de esa misma entidad; y quien permite esta identificación es la **clave primaria**. La clave primaria, que generalmente se identifica por medio de la letra @, puede ser un atributo o una combinación de atributos.

En consecuencia en cada archivo solo podrá existir un único registro que posea un valor determinado para su llave primaria. En otras palabras no puede existir en un archivo un registro que cuente con el mismo valor de otro registro en el campo de la llave primaria; la llave primaria no puede tener valores repetidos para distintos registros.

La clave primaria debe permitirle a un Sistema de Gestión de Base de Datos (SGBD), correctamente proyectado, generar un error si un usuario intenta incluir un nuevo registro cuya llave primaria coincide con la de otro registro ya existente en el archivo.

Una clave principal correcta debe tener varias características:

- Identifica inequívocamente cada fila.

- Nunca debe estar vacía ni ser nula (siempre debe contener un valor).
- Los valores que contiene no suelen cambiar (lo ideal es que no cambien).

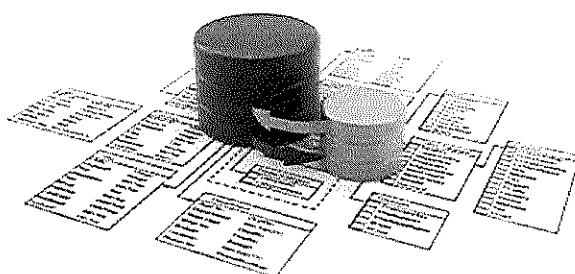
## Índices de acceso

Un índice de acceso es un archivo auxiliar utilizado internamente por el SGDB para acceder directamente a cada registro del archivo de datos. La operación de indexación, creada por el SGDB, ordena a los registros de un archivo de datos de acuerdo con los campos utilizados como llave primaria e, incrementa sensiblemente la velocidad de ejecución de algunas operaciones sobre el archivo de datos. Normalmente para cada archivo de datos debe existir un índice cuya llave de indexación sea idéntica a su llave primaria. Este índice es llamado **índice primario**.

También es posible crear índices para un archivo de datos utilizando atributos (campos), o conjunto de atributos, diferentes de los de la llave primaria. Este tipo de índice, llamado **índice secundario**, es utilizado para reducir el tiempo de localización de una determinada información dentro de un archivo o para clasificar los registros del archivo de acuerdo con el orden necesario para la obtención de la información deseada.

## Relaciones entre las tablas

Una **relación** se establece haciendo coincidir los datos de las columnas de clave, normalmente las columnas con el mismo nombre en ambas tablas. En la mayor parte de los casos, la relación hace coincidir la clave principal de una tabla, que proporciona un identificador único para cada fila, con una entrada de la clave externa de la otra tabla. Por ejemplo, las ventas pueden asociarse a los títulos concretos vendidos creando una relación entre las columnas de identificador de título de la tabla Títulos (la clave principal) y de la tabla Ventas (la clave externa).



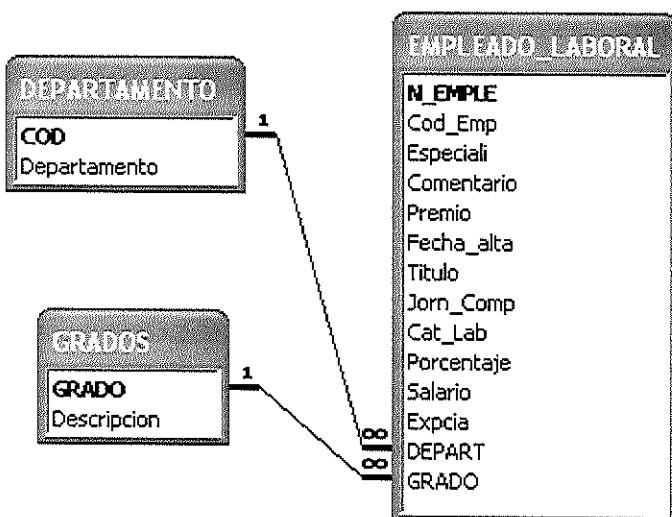
La manera en que se relacionan las tablas entre sí da lugar a comportamientos diferentes. Podemos definir tres tipos de relaciones:

- **Uno a muchos.** Cuando un registro de una tabla (tabla secundaria) sólo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la otra tabla (tabla principal) puede tener más de un registro relacionado en la primera tabla (tabla secundaria).
- **Muchos a muchos.** Cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa.
- **Uno a uno.** Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa.

De todas ellas la más utilizada y recomendable en la mayoría de los casos será el modelo Uno a muchos. Nosotros vamos a ver como funciona los tres tipos de relaciones.

#### *Relación uno a muchos*

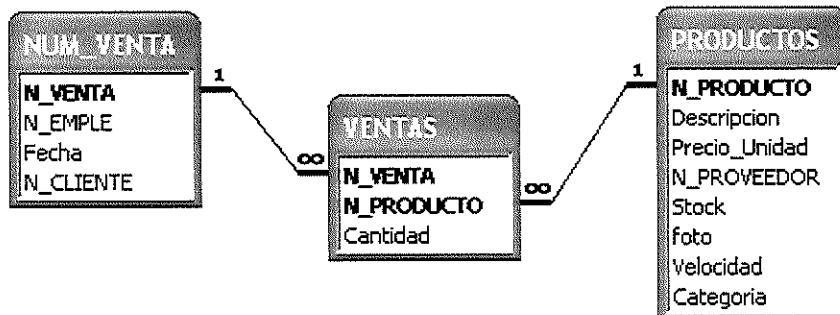
Por cada registro de la tabla principal (tabla de la clave principal o lado uno de la relación) pueden existir muchos (infinitos) registros en la tabla relacionada (tabla de la clave externa o lado infinito de la relación). La tabla relacionada no puede contener un registro que no esté relacionado con uno de la tabla principal, pero pueden haber muchos registros que estén relacionados con el mismo registro de la tabla principal: varios (infinitos) empleados de la tabla EMPLEADO\_LABORAL, pueden estar en el mismo departamento de la tabla DEPARTAMENTO...



### *Relación muchos a muchos*

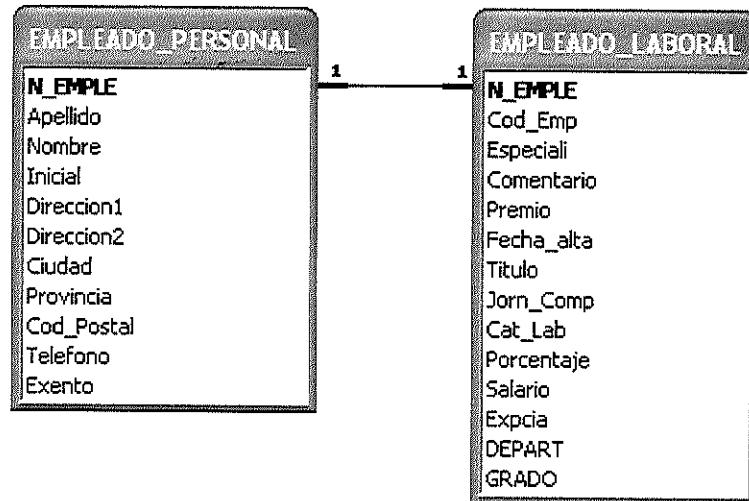
Debes tener claro las llaves primarias y una tabla de unión para que se produzca la relación varios a varios entre PRODUCTOS y NUM\_VENTA. Un registro de la tabla NUM\_VENTA puede estar relacionado con varios registros de la tabla PRODUCTOS y un registro de la tabla PRODUCTOS puede estar relacionado con varios registros de la tabla NUM\_VENTA. Explicación: cuando realizamos una venta le asignamos un número (N\_VENTA) y lo almacenamos en la tabla NUM\_VENTA; esa venta puede estar formada por uno o varios productos identificados con un numero de producto (N\_PRODUCTO) de la tabla PRODUCTOS.

Pero, ¿cómo podemos relacionar varios registros de la tabla NUM\_VENTA con varios de PRODUCTOS, y viceversa? En realidad esta relación está formada por dos relaciones de uno a muchos. Una tabla intermedia (tabla de unión VENTAS) contiene la clave principal múltiple que se forma con la combinación de dos (o más) claves externas: N\_VENTA y N\_PRODUCTO. La combinación de estos dos campos forma un campo que no se repite. Por ejemplo: la venta 200 (N\_VENTA vale 200) se realizó con los productos 12, 14 y 36 (N\_PRODUCTO). Si formamos con N\_VENTA y N\_PRODUCTO una clave principal obtenemos 20012, 20014 y 20036, valores no repetidos. Con otros valores de N\_VENTA se procedería igual.



### *Relación uno a uno*

Por cada registro de la tabla principal (tabla que contiene la clave principal) puede existir un sólo registro en la tabla relacionada (tabla que contiene la clave externa). La tabla relacionada no puede contener un registro que no esté relacionado con uno de la tabla principal: no puede existir un registro con FECHA\_ALTA, SALARIO, etc., si no hay un empleado con el que se relacione. Esta relación se utiliza para simplificar y organizar las tablas con muchos campos. Ver el ejemplo siguiente:



## RECUERDA

- El conjunto unificado de información, resultante de nuestro proyecto informático y, que será compartida por los diferentes usuarios de la organización, va a conformar la denominada Base de Datos.
- La función básica de una base de datos es permitir el almacenamiento y la recuperación de la información necesaria.
- Dependiendo de la capacidad de almacenamiento y procesamiento del hardware, la organización puede contar con una única Base de Datos, o con múltiples Bases de Datos.
- Independientemente de la Base de Datos que será implementada, ésta necesita de un Sistema de Gestión de Base de Datos (SGBD o DBMS). Los sistemas de Gestión de Base de datos, son programas de software para la administración de las Bases de Datos; y en particular, para: almacenar, manipular y recuperar datos en una computadora.
- Una Base de Datos está compuesta por un conjunto de tablas o archivos. Tabla en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de hoja de cálculo.
- Una entidad es cualquier persona, cosa o evento, real o imaginario, de interés para la organización y acerca del cual se capturan, almacenan o procesan datos.
- Un atributo es pues, cualquier detalle que sirve para identificar, clasificar, cuantificar o expresar el estado de una entidad.
- Los registros están formados por un conjunto de datos almacenados en los campos de cada atributo; y cada registro debe contener el conjunto de atributos necesarios, para describir completamente cada entidad sobre la cual una organización necesita almacenar y obtener información.
- Los archivos pueden clasificarse en cuatro tipos básicos; que son:
  - Los archivos maestros.
  - Los archivos de transacciones.

- Los archivos de control.
- Los archivos de planeamiento.
- Esta clasificación dependerá de la relación lógica que tengan que tener los datos, para dar apoyo a la actividad de la organización.
- Cada instancia de una entidad debe ser únicamente identificable, de manera tal que cada registro de la entidad debe estar separado y ser únicamente identificable del resto de los registros de esa misma entidad; y quien permite esta identificación es la clave primaria.
- Un índice de acceso es un archivo auxiliar utilizado internamente por el SGDB para acceder directamente a cada registro del archivo de datos.
- Existen dos tipos de índice de acceso:
  - Índice primario.
  - Índice secundario.
- Una relación se establece haciendo coincidir los datos de las columnas de clave, normalmente las columnas con el mismo nombre en ambas tablas.
- La manera en que se relacionan las tablas entre sí da lugar a comportamientos diferentes. Podemos definir tres tipos de relaciones:
  - Uno a muchos.
  - Muchos a muchos.
  - Uno a uno.

## Preguntas de Autoevaluación

### 1. Completa el espacio en blanco del siguiente enunciado:

"Los registros están formados por un conjunto de datos almacenados en los campos de cada \_\_\_\_\_".

- a) Atributo.
- b) Base de datos.
- c) Tabla.

### 2. Una tupla: (Respuesta múltiple)

- a) Corresponde a una fila de la tabla.
- b) El número de tuplas se denomina cardinal.
- c) Es invariable en el tiempo.

### 3. Seleccione cuáles de los siguientes archivos son maestro: (Respuesta múltiple)

- a) Procesos de exportación.
- b) Nómina de personal.
- c) Plan de cuentas.

### 4. ¿Cuáles son las características de una clave principal correcta? (Respuesta múltiple)

- a) Puede estar vacía o ser nula.
- b) Identifica inequívocamente cada fila.
- c) Los valores que contiene no suelen cambiar.

**5. Indica si es verdadero o falso el siguiente enunciado:**

*"El llamado índice secundario, es utilizado para ampliar el tiempo de localización de una determinada información dentro de un archivo o para clasificar los registros del archivo de acuerdo con el orden necesario para la obtención de la información deseada".*

a) Verdadero.

b) Falso.

# UD5 Análisis funcional



**UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM**



## 1. División de las actividades del ERP en módulo

Los módulos de un sistema ERP varían dependiendo de las características de la empresa, pues son muy diferentes los requerimientos en organizaciones en las que, por ejemplo, su principal negocio es la producción, la distribución o bien los servicios.

La mayoría de los ERPs adoptan una estructura modular que soporta los diferentes procesos de una empresa: el modulo de gestión financiera, el módulo de compras, el módulo de gestión de ventas, el módulo de recursos humanos, etc.

Todos estos módulos están interconectados y comparten una base de datos común, garantizando de este modo la coherencia e integración de los datos generados.

El hecho de que estos productos sean modulares posibilita la implantación del sistema por etapas, reduciendo el impacto global en la organización al facilitar la transición desde los sistemas anteriores. Normalmente, el primer módulo que se pone en marcha es el financiero y, posteriormente, se van integrando los restantes, dependiendo de las características particulares de cada empresa

El sistema básico del ERP está formado por las aplicaciones técnicas y la arquitectura necesaria para servir de plataforma al resto de los módulos.

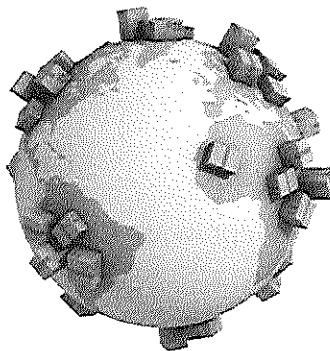
Proporciona herramientas de administración para controlar tanto el sistema en sí (rendimiento, comunicación con otras aplicaciones y otros sistemas, etc.), como la base de datos que constituye el núcleo del producto.

Las principales plataformas de servidores son Microsoft y UNIX, mientras que las bases de datos más utilizadas son Oracle, Microsoft SQL

Server e IBM. Asimismo, la mayor parte de los sistemas ERP disponen de lenguajes de programación propietarios de cuarta generación, que facilitan el desarrollo y adaptación de aplicaciones a la medida de cada cliente.

## Módulos de Gestión de Compras

El proceso de compras en una empresa comprende la gestión de materiales y la relación con los proveedores.



En el apartado de gestión de materiales el sistema debe dar soporte a la definición de los datos necesarios para el tratamiento de los materiales a lo largo de toda la cadena logística, así como las transacciones realizadas con ellos, facilitando el control de los stocks, la generación de nuevos pedidos, la valoración de inventarios de acuerdo con distintos criterios, etc.

En lo que se refiere al apoyo a la relación de la empresa con los proveedores, el sistema debe proporcionar toda la información sobre precios y condiciones de entrega, historial de compras, disponibilidad, etc., facilitando de este modo el proceso de toma de decisiones de compra.

Asimismo, mediante distintas opciones de análisis, el sistema puede realizar una valoración de los proveedores: cumplimiento de plazos de entrega, estado de los materiales, fiabilidad, etc.

Este módulo se apoya en dos bases de datos fundamentales:

- La base de datos de materiales, que permite registrar para cada referencia su código, descripción, peso, dimensiones, calidad, cantidad en stock, etc.
- La base de datos de proveedores, que almacena los datos sobre cada uno de los proveedores seleccionados: nombre, personas de contacto, dirección de pedido, datos fiscales para facturación, etc., así como precios y condiciones de entrega de los productos que ofrece.

El módulo de gestión de compras facilita la planificación de los pedidos a proveedores a partir de las necesidades de compra de la empresa, que pueden venir determinadas por la demanda de productos terminados o por el control de unos stocks mínimos de producción.

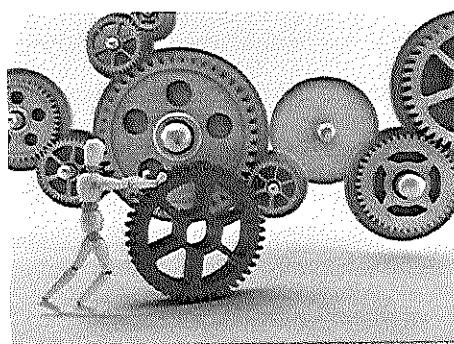
Además, este módulo puede ofrecer la posibilidad de consultar el historial de los proveedores y de los movimientos de materiales que se han realizado.

En definitiva, el módulo de gestión de compras deberá dar soporte a todos los procesos de compra, desde la gestión de proveedores y tarifas hasta el control de los procesos de pedidos, conciliación de facturas y otras fases implicadas en el gestión de compras, tanto de productos como de materias primas, bienes de inversión o servicios, así como la gestión de contratos de suministro.

## Módulo de producción

El módulo de producción se encarga de gestionar los materiales y servicios empleados en la cadena de producción de una empresa, así como los recursos (máquinas, utillaje, personal) utilizados en ésta.

Este módulo facilita la planificación de los materiales y de las capacidades de los recursos, lanzando las órdenes de montaje o de fabricación y adaptándose a las características específicas de los distintos sistemas de fabricación: fabricación contra stock, fabricación a medida contra pedido (build to order) o montaje (únicamente se realiza el ensamblaje final de las distintas piezas que componen el producto).



Para contribuir a una adecuada gestión de los stocks de materiales, este módulo debe estar totalmente integrado con el módulo de gestión de compras. Además, este módulo puede incorporar diferentes funcionalidades adicionales como

la planificación a capacidad finita, la captura de datos en planta, la gestión de subcontrataciones, etc.

## Módulos de Ventas

El módulo de ventas se ocupa de la relación de la empresa con los clientes, dando soporte a todas las actividades comerciales preventa y postventa.

Asimismo, facilita la gestión y configuración de los pedidos, la logística de distribución, la preparación de entregas, la expedición y el transporte.



Para un correcto funcionamiento, el módulo de ventas deberá estar integrado con los módulos de almacén, logística, módulo financiero, etc.

Asimismo, cada vez se exige un mayor nivel de integración entre ventas y compras, reflejo de una progresiva orientación a una operativo "bajo pedido".

## Módulo de Finanzas

El módulo de finanzas se encarga de la contabilidad y de la gestión financiera de la empresa. Se trata de un módulo esencial dentro del sistema ERP, ya que va a estar totalmente integrado con los restantes módulo. Por este motivo, resulta fundamental para la correcta implantación del ERP.



Este módulo proporciona herramientas flexibles y aplicaciones orientadas tanto a la contabilidad financiera, como a la contabilidad analítica o de costes.

Entre sus múltiples funciones relacionadas con la operativa financiera y contable podemos destacar las siguientes:

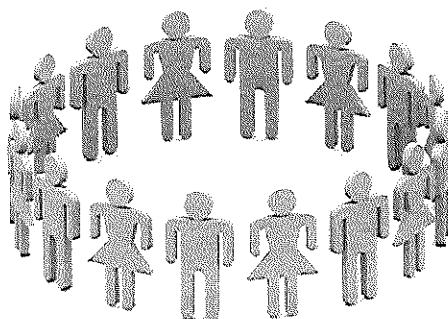
- Contabilización de las operaciones de la empresa (generación de asientos contables).
- Elaboración de los balances y de la cuenta de resultados.
- Elaboración de presupuestos, generación de informes y análisis de desviaciones.
- Gestión de la tesorería (control de flujos de cobros y pagos, gestión de cuentas corrientes, etc.).
- Gestión de activos.

Asimismo, este módulo proporciona funciones específicas para el departamento de administración de una empresa:

- Facturación.
- Liquidación.
- Gestión de cobros y reclamación de impagados.

En general todos los sistemas ERP disponen de un gran número de informes financieros y contables estándar e incorporan herramientas de diseño a medida para facilitarles la generación de informes adaptados a las necesidades de cada cliente, como en el caso de la liquidación de impuestos de cada país.

## **Módulo de Recursos Humanos**



El módulo de recursos humanos de un ERP permite gestionar la información relacionada con los empleados de una organización (datos personales, formación recibida, experiencia, ocupación, etc.). Entre las múltiples funciones que facilita podemos destacar las siguientes:

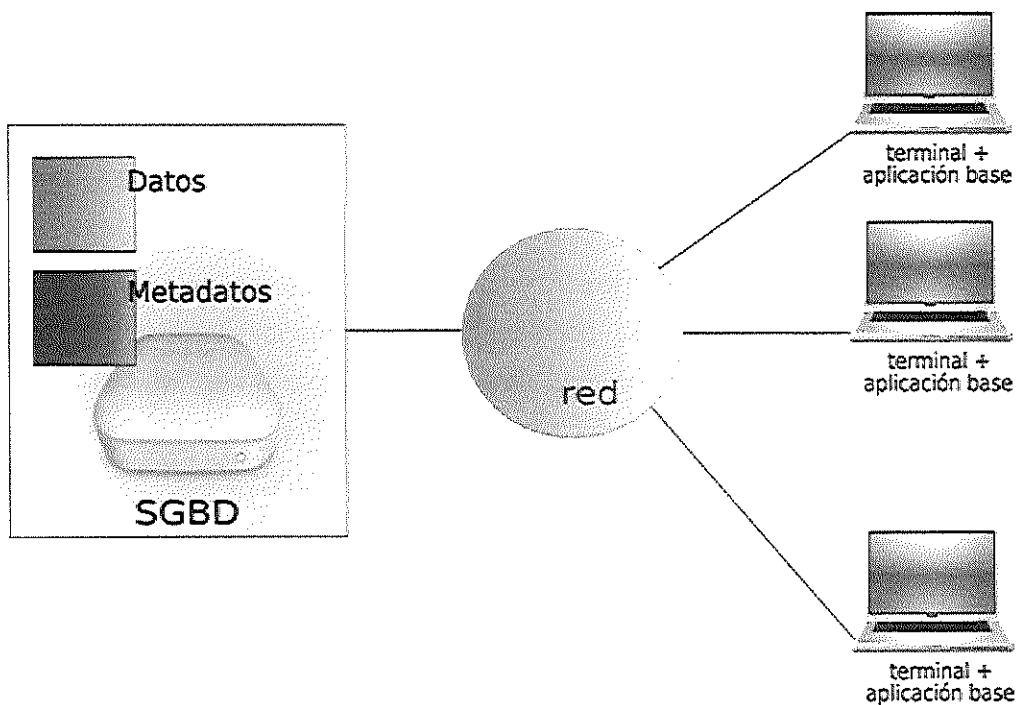
- Definición de estructuras organizativas.
- Planificación de las necesidades de personal.
- Soporte al proceso de evaluación y selección de personal.
- Control de presencia, relacionado generalmente con el módulo de producción.
- Soporte a la contratación de personal.
- Gestión de las acciones formativas.
- Registro de gastos de representación y de dietas por desplazamientos.
- Soporte a la generación de nóminas.

### **Módulo de gestión de medios técnicos y mantenimiento**

Este módulo facilita el control de los recursos materiales y técnicos de la empresa, maquinaria, elementos de transporte y repuestos, integrando las funciones empresariales de compras y mantenimiento para asegurar la disponibilidad de estos recursos en las operaciones empresariales.

## **2. Trazabilidad entre los módulos**

La gran mayoría de sistemas ERP ofrecen la posibilidad de instalación tanto en un entorno monopuesto o en red (cliente/servidor). Indudablemente, la instalación en un entorno cliente/servidor, necesitará un sistema gestor de base de datos centralizado y con los protocolos de seguridad y acceso necesarios para asegurar la integridad de los datos.



- Clientes: Los clientes son las máquinas que se encuentran conectadas directamente al SGBD (sistema gestor de base de datos) pudiendo acceder a la base de datos. En cada terminal o máquina cliente se ejecuta el software que denominamos aplicación base.
- SGBD: El gestor de base de datos se encarga de almacenar y mantener dos tipos de información (y aquí está la clave):
  - Datos: en esta zona de la base de datos se almacenan los datos concretos que la aplicación maneja y que tienen sentido para el usuario (datos de clientes, facturas, etc.). Son los datos "tradicionales".
  - Módulos de metadatos: los módulos contienen la información necesaria para implementar las aplicaciones de usuario: formularios, definiciones de tablas y campos, código de los scripts que realizan los procesos, formato y definición de los informes.
  - Los metadatos: residen en la base de datos, pero previamente deben ser cargados desde el directorio en disco en el que han sido alojados tras su descarga.

La interconexión entre cada uno de los módulos pertenecientes al sistema, proporciona un mejor rendimiento empresarial y mayores réditos.

Esta facilidad permite que no sólo la aplicación de la Planificación de Recursos Empresariales forme parte de toda la compañía en general, sino que también puede que esté aplicado solamente en un sector determinado de cada compañía, dependiendo de la gestión y de cómo se busquen procesar las distintas actividades de la firma.

Una de las formas de aplicación más comunes está ligada a la utilización de una Base de Datos Central que permite tener un mayor control de todas las actividades y los Procesos Empresariales, teniendo para ello que realizar una gestión que permita obtener la información de forma Precisa y Eficiente, sumado además a otorgar distintos permisos para los Usuarios que tendrán la aptitud para compartir los datos aportados y la posibilidad de acceder a ellos, o una parte de los mismos, en forma constante.

Pero siempre tenemos que tener en cuenta que la diferencia fundamental del Enterprise Resource Planning con respecto a otro tipo de aplicaciones de gestión empresarial es justamente la integración, funcionando cada parte en un todo, ofreciendo un Sistema Íntegro que tiene a todas sus aplicaciones trabajando en conjunto.

Esta integración se da por el conjunto de programas que permiten distinguir una División Interna que se logra diferenciar en Módulos determinados, que se pueden incluir en el sistema o quitar a requerimiento de cada compañía (Cliente), permitiendo una muy alta adaptabilidad del Software ERP a cada una de las planificaciones.

## RECUERDA

- Los módulos de un sistema ERP varían dependiendo de las características de la empresa, la mayoría de los ERPs adoptan una estructura modular que soporta los diferentes procesos de una empresa.
- El proceso de compras en una empresa comprende la gestión de materiales y la relación con los proveedores. Por lo tanto, se apoya en dos bases de datos fundamentales: la base de datos de materiales y la base de datos de proveedores.
- El módulo de gestión de compras facilita la planificación de los pedidos a proveedores a partir de las necesidades de compra de la empresa.
- El módulo de producción se encarga de gestionar los materiales y servicios empleados en la cadena de producción de una empresa, así como los recursos (máquinas, utilaje, personal) utilizados en ésta
- El módulo de ventas se ocupa de la relación de la empresa con los clientes, dando soporte a todas las actividades comerciales preventa y postventa
- El módulo de finanzas se encarga de la contabilidad y de la gestión financiera de la empresa. Se trata de un módulo esencial dentro del sistema ERP, ya que va a estar totalmente integrado con los restantes módulo.
- El módulo de recursos humanos de un ERP permite gestionar la información relacionada con los empleados de una organización (datos personales, formación recibida, experiencia, ocupación, etc.).
- El módulo de gestión de medios técnicos y mantenimiento facilita el control de los recursos materiales y técnicos de la empresa, maquinaria, elementos de transporte y repuestos, integrando las funciones empresariales de compras y mantenimiento para asegurar la disponibilidad de estos recursos en las operaciones empresariales.
- La interconexión entre cada uno de los módulos pertenecientes al sistema, proporciona un mejor rendimiento empresarial y mayores réditos.



## Preguntas de Autoevaluación

### 1. Completa el espacio en blanco del siguiente enunciado:

"El proceso de compras en una empresa comprende la \_\_\_\_\_ y la \_\_\_\_\_".

- a) Gestión de materiales y la relación con los proveedores.
- b) Gestión de materiales y la relación con los clientes.
- c) La relación con los proveedores y la gestión de stock.

### 2. El módulo de compras se apoya en dos bases de datos fundamentales:

- a) Base de datos de materiales y de proveedores.
- b) Base de datos Microsoft y Oracle.
- c) Base de datos de proveedores y base de datos microsoft.

### 3. El módulo que se encarga de gestionar los materiales y servicios empleados en la cadena de producción de una empresa, así como los recursos (máquinas, utilaje, personal) utilizados en ésta, es:

- a) Módulo de compras.
- b) Módulo de producción.
- c) Módulo de finanzas.

**4. Para contribuir a una adecuada gestión de los stocks de materiales, el módulo de producción debe estar totalmente integrado con el módulo de:**

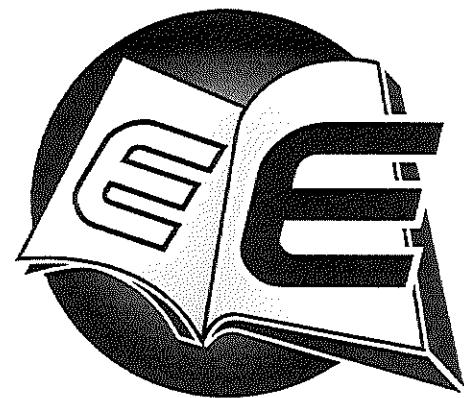
- a) Finanzas.
- b) Ventas.
- c) Gestión de compras.

**5. Indica si es verdadero o falso el siguiente enunciado:**

*"La interconexión entre cada uno de los módulos pertenecientes al sistema, proporciona un mejor rendimiento empresarial y mayores réditos".*

- a) Verdadero.
- b) Falso.

# UD6 Programación en sistemas ERP y CRM



**UF1889 Desarrollo de  
Componente Software  
en Sistemas ERP-CRM**



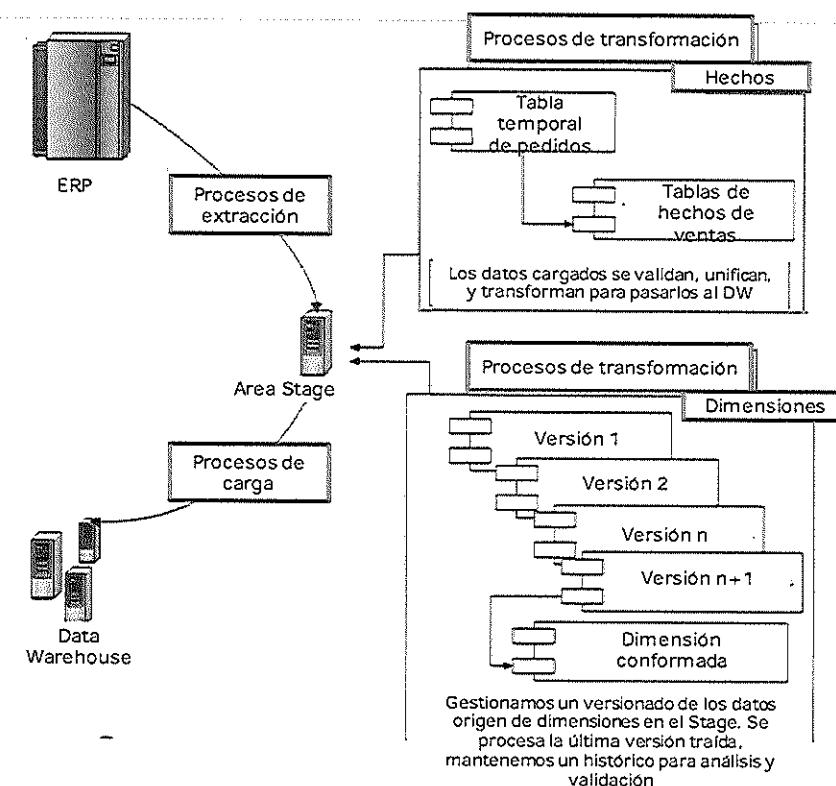
# 1. Generación de programas de extracción de datos entre sistemas (batch inputs)

## 1.1. Herramientas para la carga y extracción de datos de sistemas de almacén de datos

El proceso de extracción, transformación y carga, conocido como ETL, es importante ya que es la forma en que los datos se guardan en un almacén de datos.

Implican las siguientes operaciones:

- **Extracción:** acción de obtener la información deseada a partir de los datos almacenados en fuentes externas.
- **Transformación:** cualquier operación realizada sobre los datos para que puedan ser cargados en el data warehouse o se puedan migrar de éste a otra base de datos.
- **Carga:** consiste en almacenar los datos en la base de datos final.

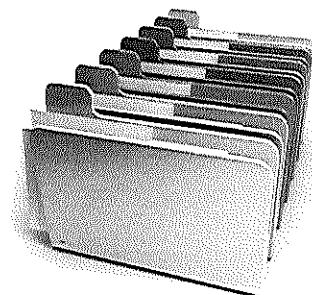


### 1.1.1. Mecanismos que se utilizan para la extracción de datos

La primera parte del proceso ETL consiste en **extraer** los datos desde los sistemas de origen. La mayoría de los proyectos de almacenamiento de datos fusionan datos provenientes de diferentes sistemas de origen. Cada sistema separado puede usar una organización diferente de los datos o formatos distintos.

Los formatos de las fuentes normalmente se encuentran en bases de datos relacionales o ficheros planos, pero pueden incluir bases de datos relacionales u otras estructuras diferentes. La extracción convierte los datos a un formato preparado para iniciar el proceso de transformación.

Una parte intrínseca del proceso de extracción es la de analizar los datos extraídos, de lo que resulta un chequeo que verifica si los datos cumplen la pauta o estructura que se esperaba. De no ser así los datos son rechazados.



Un requerimiento importante que se debe exigir a la tarea de extracción es que ésta cause un impacto mínimo en el sistema origen. Si los datos a extraer son muchos, el sistema de origen se podría ralentizar e incluso colapsar, provocando que éste no pueda utilizarse con normalidad para su uso cotidiano. Por esta razón, en sistemas grandes las operaciones de extracción suelen programarse en horarios o días donde este impacto sea nulo o mínimo.

La extracción de datos se define como una estrategia o proceso central que trata de ensamblar datos desde sistemas y fuentes dispares, enriqueciendo estos datos de manera que se produzca información valiosa y reutilizable.

Hay ciertos aspectos a tener en cuenta en este proceso:

- Extraer los datos eficiente y eficazmente desde los sistemas fuentes.
- Identificar y documentar el nivel de servicio de acuerdo con los sistemas fuentes.

- Brindar una guía sobre el mecanismo de transferencia de datos, escalando los procedimientos en caso de fallos durante la transmisión de los datos.
- Suplementar/enriquecer los datos desde los sistemas fuentes para una fácil actualización y cambio que sea aplicable en el ambiente de data warehouse.
- Proveer un marco para permitir una estrategia de reconciliación favorable de acuerdo a la fuente y el destino final.

En esta fase lo más adecuado es utilizar una herramienta ETT (en inglés ETL), que permita manipular una amplia variedad de datos del sistema fuente pudiendo conseguir que todos los datos tengan un formato común.

Hay que tener en cuenta que la herramienta de ETT que se seleccione ha de permitir lograr los resultados deseados en un tiempo relativamente menor que la forma tradicional de codificar y mantener los objetos del data warehouse.

En la selección del ETT se debe tener en cuenta:

- Fácil de usar y comprensible desde el punto de vista del mantenimiento y el desarrollo de la perspectiva.
- El proceso ETL debe integrarse con el proceso de negocio.
- Debe soportar el procesamiento de grandes volúmenes de datos.
- Poder extraer datos desde distintas fuentes heterogéneas.
- Puede ser necesario que la herramienta ETL soporte procesamiento paralelo.
- Debe poseer un amplio espectro de conectividad y la habilidad de estandarizar los datos tomados desde diversas fuentes, que pueden estar incluso almacenadas en bases de datos soportadas sobre una plataforma diferente.

Una posible metodología para la extracción de conocimiento a partir de los datos sería la que aparece a continuación descrita en la siguiente tabla.

| Fases       | Objetivo  | Técnica/Herramienta   |
|-------------|---|---|
| Identificar | Seleccionar las bases de datos que puedan aportar la información necesaria para obtener Experiencia de los expertos. el conocimiento. |   |
| Extraer     | Ensamblar datos desde fuentes dispares, enriqueciéndolos de manera que cree información valiosa.                                      | Herramientas ETL/ Data Warehouse.                             |
| Procesar    | Construir por medio de algoritmos de Minería de Datos, modelos de comportamiento.   | Minería de Datos  |
| Almacenar   | Validar, seleccionar y mantener los modelos de comportamiento.  | Minería de Datos/ Experiencia del Ingeniero del Conocimiento. |
| Compartir   | Poner a disposición de la Organización el conocimiento descubierto.   | Portal del Conocimiento.                                      |

Una vez extraídos los datos, se integran y almacenan en un data warehouse. La meta de un data warehouse es integrar aplicaciones a nivel de datos. El dato extraído de los sistemas operacionales se procesa, se transforma y ubica de acuerdo a un esquema similar a un modelo entidad/relación. La noción del data warehouse debe ser extendida para incluir no sólo datos orientados a transacciones, sino también aquellos datos creados por los ingenieros del conocimiento.

### **OLAP On-Line Analytical Processing**

Los sistemas OLAP son bases de datos orientadas al procesamiento analítico. Este análisis implica la lectura de grandes cantidades de datos para llegar a extraer algún tipo de información útil. Este sistema es típico de los datamart.

- El acceso a los datos suele ser de sólo lectura. La acción más común es la consulta, con muy pocas inserciones, actualizaciones o eliminaciones.
- Los datos se estructuran según las áreas de negocio, y los formatos de los datos están integrados de manera uniforme en toda la organización.
- El historial de datos es a largo plazo, por lo general implica entre dos y cinco años.