|  |
| --- |
| Unidad 1- Introducción a los sistemas de base de datos |

**Base de Datos**: Una base de datos es un conjunto de datos relacionados entre sí, persistentes en el tiempo que son utilizados por programas de aplicación de una organización.

**Propiedades Implícitas**:

* Es  un  conjunto  de  datos  lógicamente coherente, con cierto significado inherente. Por ejemplo una agenda de números telefónicos.
* Se diseña, construye y carga datos para un propósito específico. Está dirigida a un determinado grupo de usuarios y la utilizan aplicaciones específicas.

**Sistema de Base de Datos**: Un sistema de bases de datos es un sistema computarizado para guardar datos y permitir a los usuarios recuperarlos y actualizarlos cuando se necesite.

**Composición de un Sistema de Base de Datos**:

* **Datos**: pueden estar integrados y compartidos, en sistemas monousuario o multiusuario.
* **Hardware**: discos, memoria, procesador, dispositivos de E/S.
* **Usuarios**: 3 grupos, usuarios finales, programadores de aplicaciones y el administrador de bases de Datos.
* **Software**: entre la base de datos física y los usuarios existe una capa de software conocida como el administrador de base de datos (DBMS). Es el encargado de gestionar las solicitudes de acceso a la BD.

**Propósito de un Sistema de Base de Datos**:

* Compactación
* Velocidad.
* Menos trabajo laborioso.
* Actualidad.

**Beneficios de un Sistema de Base de Datos**:

* Los datos pueden compartirse.
* Es posible reducir la redundancia.
* Es posible evitar la inconsistencia.
* Es posible brindar un manejo de transacciones.
* Es posible mantener la integridad.
* Es posible hacer cumplir la seguridad.
* Es posible equilibrar los requerimientos en conflicto.
* Es posible hacer cumplir los estándares.

**Objetivo de un Sistema de Base de Datos**:

* El objetivo fundamental de un Sistema de Bases de Datos es lograr la Independencia de los Datos.

**Administrador de base de datos (DBMS)**: Es el software que maneja todo el acceso a la base de datos.

* Es el componente del sistema más importante.
* También existen otras herramientas de desarrollo y utilitarios, generadores de reportes, y el más importante el monitor de transacciones o PT.

**Conceptualmente**:

* Un usuario hace una petición de acceso, usando un lenguaje específico.
* El DBMS intercepta la solicitud y la analiza.
* El DBMS inspecciona el esquema externo para ese usuario y hace las transformaciones correspondientes
* Ejecuta las operaciones necesarias sobre la base de datos  almacenada.

**Componentes y Funciones**:

* Definición de datos.
* Manipulación de batos.
* Optimización y ejecución.
* Seguridad e integridad.
* Recuperación de datos y concurrencia.
* Diccionario de datos (Metadata)
* Rendimiento

**Comparación con un sistema de archivos**: El sistema de archivos:

* No conoce la estructura interna de los registros.
* No provee restricciones de seguridad e integridad.
* No provee mecanismos de recuperación y concurrencia.
* No existe el diccionario de datos.
* No proporciona una correcta independencia de los datos.
* No pueden compartirse e integrarse los datos de la misma manera que en una base de datos.

**Control Centralizado**:

* Este concepto surge de las bases de datos multiusuario.
* Se reducen los archivos dispersos y propios de cada aplicación.
* Un único punto de acceso a los datos, se centraliza la seguridad y políticas de acceso. Estas políticas las fija el administrador de datos.

**Independencia de Datos:** Es la inmunidad de las aplicaciones a cambios en la representación física y en la técnica de acceso

* Existe la dependencia física y lógica. Nos ocuparemos de la física.
* Los sistemas previos a los sistemas de bases de datos relacionales estaban más al tanto de la representación física y de las técnicas de acceso.

**Requisitos para logar la Independencia de Datos**:

* Representación de datos numéricos.
* Representación de datos de caracteres.
* Unidades para datos numéricos.
* Codificación de los datos.

|  |  |  |  |
| --- | --- | --- | --- |
| DNI | Nombre | FechaNacimiento | Edad |

* Materialización de los datos: Ej.: Donde Edad = Fecha del Sistema – FechaNacimiento expresado en años.

|  |  |  |  |
| --- | --- | --- | --- |
| DNI | Sexo |  | Foto |

* Estructura de los registros almacenados:

|  |  |  |
| --- | --- | --- |
| DNI | Sexo | Foto |

**Arquitectura de un Sistema de Bases de Datos**:

* Brinda un marco para definir los conceptos generales.
* Es genérico, no quiere decir que un sistema de base de datos en particular lo cumpla.
* Está basado en la arquitectura ANSI/SPARC

**Arquitectura ANSI/SPARC**: consta de 3 niveles.

* **Nivel Interno**:
* Es una representación de bajo nivel, consiste de todas las ocurrencias de registros almacenados, índices, etc.
* Esta representación dista igualmente de la forma en que realmente se almacenan los datos.
* Se escribe usando el DDL Interno.
* **Nivel Conceptual**:
  + Es una representación de todo el contenido de la base de datos.
  + a vista conceptual está comprendida por todas las ocurrencias de registros conceptuales.
  + No hay referencias a las técnicas de acceso, índices, apuntadores, etc.
  + Se escribe usando el DDL Conceptual.
  + Incluye aspectos de seguridad e integridad, reglas del negocio en general.
* **Nivel Externo**:
  + Es el nivel del usuario individual.
  + Se usa el lenguaje primario y el sublenguaje de datos (SLD), generalmente este es SQL.
  + El SLD consta de dos sublenguajes, el DDL y el DML.
  + Una vista externa es la visión particular que tiene un usuario de la base de datos. Es la ocurrencia de todos los registros externos.

**Transformaciones**:

* La arquitectura comprende también las transformaciones entre los diferentes modelos
* externa/conceptual y conceptual/interna
* Por ejemplo si se modifica la vista interna, también se deberá modificar la transformación conceptual/interna, para mantener invariable la vista conceptual.

**Dependencia de Datos**: Una aplicación es dependiente de los datos cuando es imposible modificar la representación física o la técnica de acceso sin afectar drásticamente la aplicación

**DBA**: Es la persona con conocimiento técnico, capaz de implementar las políticas de seguridad, de acceso, de optimización y respaldo de la base de datos en una organización.

**Responsabilidades**:

* Definir el esquema conceptual.
* Definir el esquema interno.
* Establecer un enlace con los usuarios.
* Definir las restricciones de seguridad e integridad.
* Definir las políticas de vaciado y carga.
* Supervisar el rendimiento.

**Otras definiciones**:

* **Campo almacenado**: es la unidad más pequeña  que se puede almacenar. Ej.: DNI, nombre
* **Registro almacenado**: es un conjunto de  campos almacenados. Ej.: registro almacenado  alumno.
* **Archivo almacenado**: es la colección de todas  las ocurrencias de un mismo tipo de registro  almacenado.

|  |
| --- |
| Unidad 2 - Modelo conceptual de datos |

**Diseño Conceptual de Base de Datos**:

* Sirve como auxiliar en el Diseño de Base de Datos.
* Es un Lenguaje de Alto Nivel.
* Permite Identificar Conceptos semánticos importantes.
* Independiente del DBMS

**Modelar**:

* Permite comunicar conceptos que de otra forma seria difícil.
* Es una abstracción de la realidad.
* Permite entender lo que estamos construyendo y guiar el proceso.
* Documenta las decisiones tomadas.

**Lenguaje Unificado de modelado (UML)**:

* Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema software.
* Brinda una forma standard de escribir conceptos, procesos de negocios, funciones del sistema, hasta cosas concretas como clases escritas en un lenguaje específico, esquemas de bases de datos y partes reutilizables de software.
* Es extensible, pudiéndose adaptar a necesidades específicas de una aplicación u organización.

**Diagramas de Clases**:

* El diagrama de clases es usado para hacer modelado estructural, pero el enfoque particular que vamos a darle es el Diseño Conceptual de Bases de Datos.
* Un diagrama de clases es una vista estática del diseño de un sistema.
* Es el diagrama más comúnmente encontrado en cualquier diseño OO.
* Muestra un conjunto de clases, interfaces, colaboraciones y sus relaciones.

**Diagrama Entidad Relación**:

* Es el más ampliamente difundido.
* Lo entienden la mayoría de los diseñadores de Base de Datos.
* Existen diversas herramientas CASE que permiten exportar e importar esquemas prácticamente sin intervención del usuario.

**Ventajas**:

* + Solo se enfoca en los datos.
  + El diagrama de Clases es un Superconjunto del Diagrama E/R.
  + Permite modelar comportamiento.
  + Permite utilizar el mismo lenguaje (UML) para modelar el sistema completo.

**Consecuencias**:

* + Facilita la comunicación entre analistas, programadores y diseñadores de base de datos.
  + Permite utilizar las mismas herramientas CASE.
  + Permite utilizar un standard actual de documentación y diseño.
  + Permite relacionar mejor el sistema en su conjunto y no tenerlo en diseños separados.

**Visualización**:

* Al ver un diagrama de clases generalmente se pueden apreciar a simple vista un conjunto de cajas unidas por líneas de diversas formas.
* Las cajas son representaciones de Clases o de Interfaces.
* Las líneas representan las Relaciones entre las clases, estas relaciones pueden ser de Asociación o Generalización.

**Estructura de una clase**:

* Se representa mediante una caja con tres divisiones.
* En la primera división se indica el nombre de la clase.
* La segunda división se especifican los atributos.
* La tercera división se especifican las operaciones, las cuales están fuera del alcance del contenido de la materia.
* Los prefijos en los atributos u operaciones + (Publico) – (Privado) # (Protegido) indican la visibilidad de los mismos.
* Luego del nombre del atributo o de la operación aparece el nombre de un tipo de dato. En el contexto de las bases de datos relacionales, estos tipos deberían ser tipos simples como Int, String, Float, Boolean.

**Relaciones**:

* Se representan mediante líneas de diversas formas y estilos.
* Agregan semántica al diagrama.
* Pueden ser relaciones de Asociación o de Generalización.
* Dentro de la asociación podemos encontrar la asociación propiamente dicha, la agregación, que a su vez puede ser agregación compartida o bien composición.
* En los extremos de las relaciones figuran símbolos como 0..1, 1, 1..\*, \*. Estos indican la multiplicidad, o sea con la cantidad de instancias del objeto que pueden relacionarse.
* La multiplicidad es un concepto importante al momento de transformar el diagrama al esquema de la base de datos. Esto define las claves foráneas.
* En la línea puede aparecer un comentario que sirve para darle significado a la relación.



**Propiedades Extendidas**:

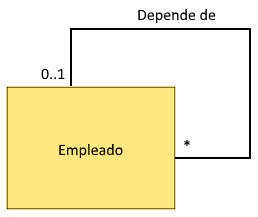
* UML es extensible y nos proporciona determinadas técnicas para documentar o extender el lenguaje.
* **Stereotipos**: mediante el uso de una simple anotación podemos documentar otras cosas. Generalmente el stereotipo se lo representa por un texto encerrado entre << >> El stereotipo que indica que una clase representa una estructura en una base de datos es Persistent. Si el stereotipo se encuentra en una relación se lo puede representar entre llaves, por ejemplo {ordered}
* **Propiedades**: permiten indicar propiedades adicionales a los atributos. Las propiedades se especifican entre llaves y se ponen al final de los atributos.
* El uso más común que podemos darle en el contexto del diseño conceptual de bases de datos es definir la clave primaria. Por convención se utiliza {OID}
* También se podríamos llegar a indicar si un atributo puede tomar valores nulos {nullable}, o la clave alternativa {alternate OID}

**Consideraciones**:

* Nombrar las clases en singular.
* Nombrar todas las asociaciones n a n. Facilita la transformación.
* Evitar las relaciones 1 a 1 y las n-arias.
* Evitar tener grandes jerarquías de clases, es complicada la implementación.
* Las clases especializadas sin atributos tienen solamente significado semántico. Al hacer la transformación no se obtiene una relación como resultado.

**Relaciones n-arias**:

* Existen otro tipo de relaciones más complejas donde la aridad de la relación es mayor a 2.
* Es bastante frecuente encontrar relaciones ternarias.
* Las relaciones de aridad mayores a 3 son muy poco frecuentes.
* Las relaciones n-arias resuelven problemas que no se pueden resolver con relaciones binarias.
* La forma de representar una relación n-aria es con un rombo.

**Reflexiva**: