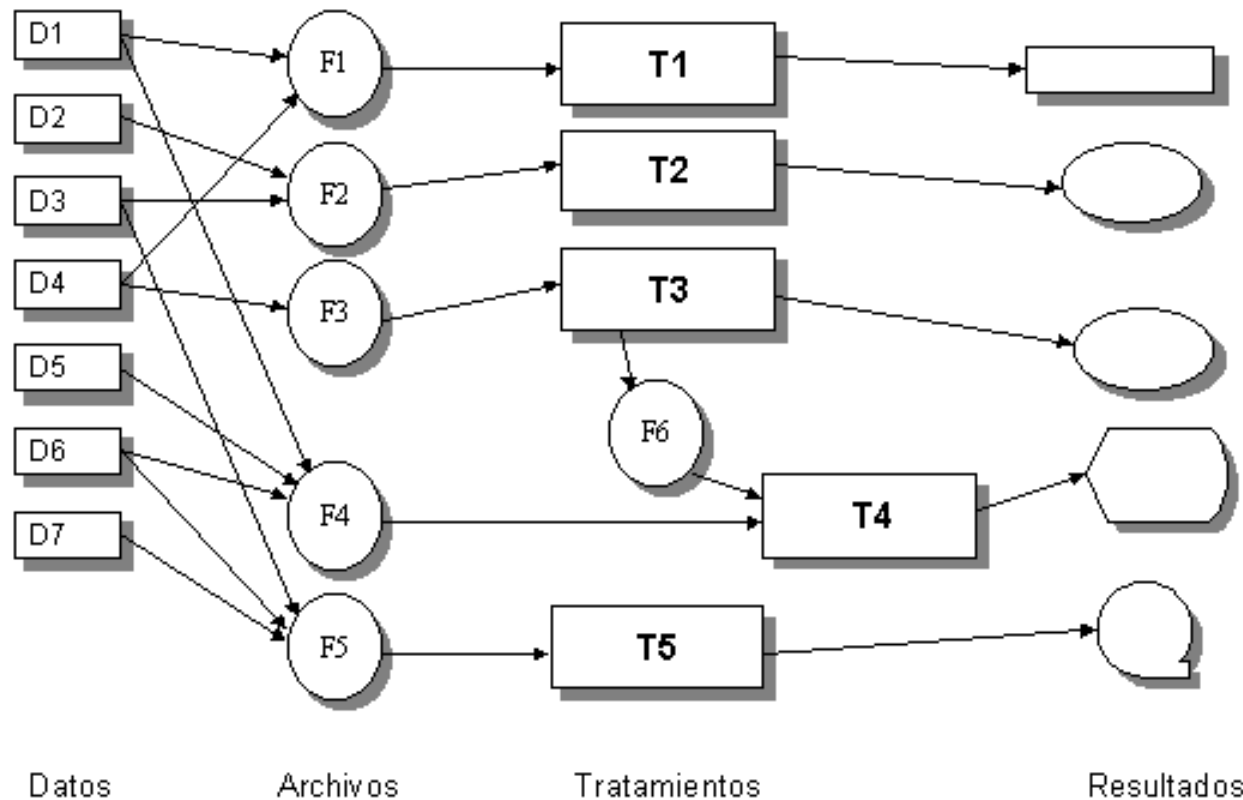


Introducción

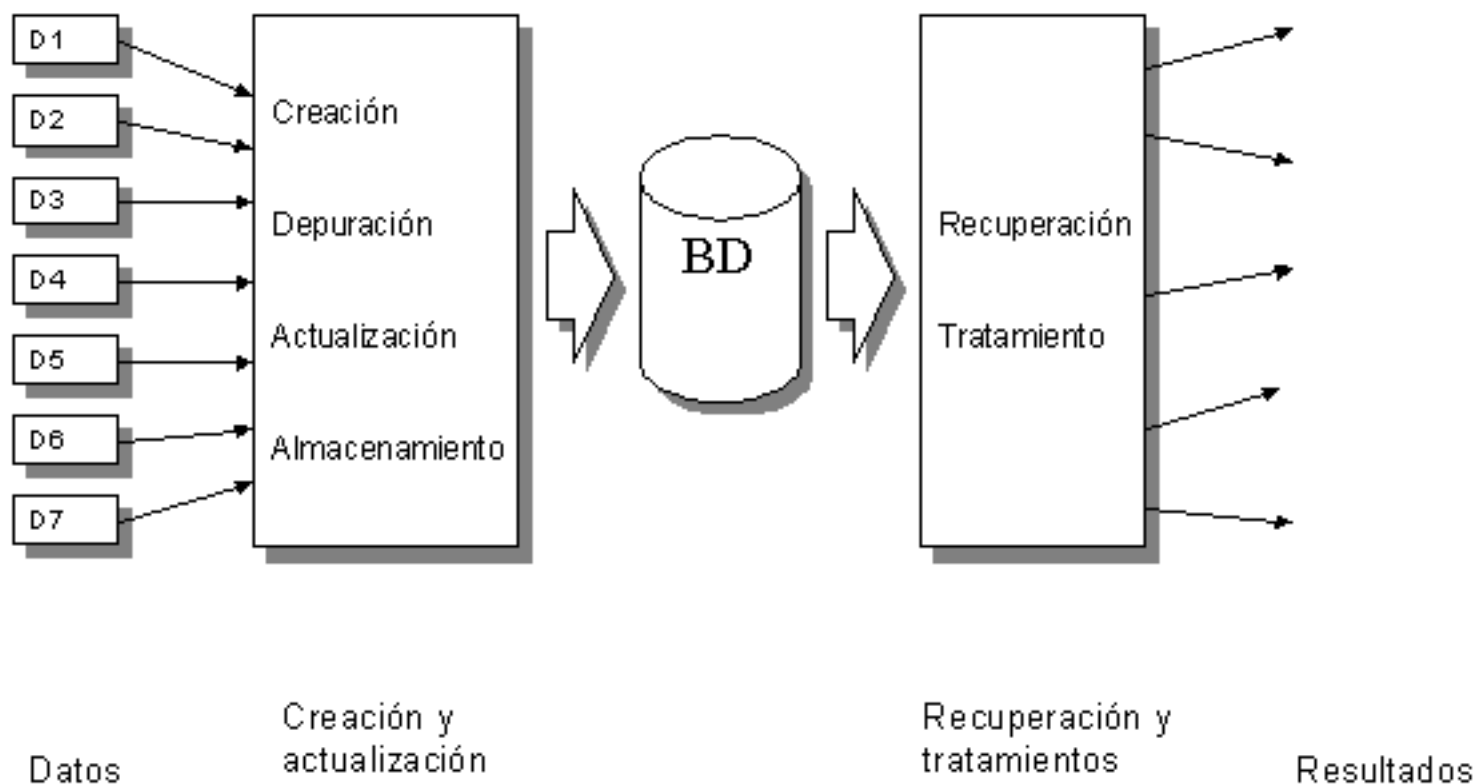
“Los datos no son de una aplicación sino de una organización entera que los va a utilizar”

Estructura en el manejo de datos a través de archivos



Introducción

Estructura en el manejo de datos a través de bases de datos



Definiciones

Definición 1: "Colección de datos interrelacionados almacenados en conjunto **sin redundancias perjudiciales o innecesarias**; su finalidad es servir a una o más aplicaciones de la mejor forma posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados". **J. Martin**

Definición 2: "Colección integrada y generalizada de datos, estructurada atendiendo a las relaciones naturales de modo que suministre todos los caminos de acceso necesarios a cada unidad de datos con objeto de poder atender todas las necesidades de los diferentes usuarios". **Deen.**

Definición 3: "Colección de datos integrados, con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real; los datos, que han de ser compartidos por diferentes usuarios y aplicaciones, deben mantenerse independientes de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de actualización y recuperación, comunes y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de los datos". **A. de Miguel.**

Definición 4: "Una base de datos consiste en alguna colección de datos persistentes e independientes usados por una organización determinada." **J.Date.**

Características elementales

Disminuir la redundancia de un conjunto de datos determina dos características fundamentales para **Sistema de Bases de Datos**:

Integrada:

Podemos considerar que una BD es una integración de varios archivos de datos donde se elimina total o parcialmente cualquier tipo de redundancia de datos.

Compartida:

Cada parte individual de la BD puede ser accedida por un conjunto de usuarios.

Esquema en el diseño de BD

El diseño de una base de datos se realiza en tres fases:

Diseño conceptual

Representación total y abstracta de los datos que componen la Base. ***Esquema conceptual***. (DER)

Diseño lógico

Transformar el diseño conceptual al modelo de datos del **SGBD**.

Diseño físico

Implementar de forma eficiente el diseño lógico. Es completamente dependiente del SGBD y el equipo en donde se implemente.

Ventajas de las DB

Cuadro resumen de las ventajas de las bases de datos	
Referidas a	Ventajas
Los datos	<ul style="list-style-type: none">• Independencia de estos respecto de los tratamientos y viceversa• Mejor disponibilidad de los mismos• Mayor eficiencia en la recuperación, codificación y entrada
Los resultados	<ul style="list-style-type: none">• Mayor coherencia• Mayor valor informativo• Mejor y más normalizada documentación de la información
Los usuarios	<ul style="list-style-type: none">• Acceso más rápido y sencillo de los usuarios finales• Más facilidades para compartir los datos por el conjunto de los usuarios• Mayor flexibilidad para atender a demandas cambiantes.

Desventajas de las DB

Cuadro resumen de las desventajas de las bases de datos	
Relativas a	Desventajas
La implantación	<ul style="list-style-type: none">• Costosa en equipos (lógico y físico)• Ausencia de estándares• Larga y difícil puesta en marcha• Rentabilidad a mediano plazo
Los usuarios	<ul style="list-style-type: none">• Personal especializado• Desfase entre teoría y práctica

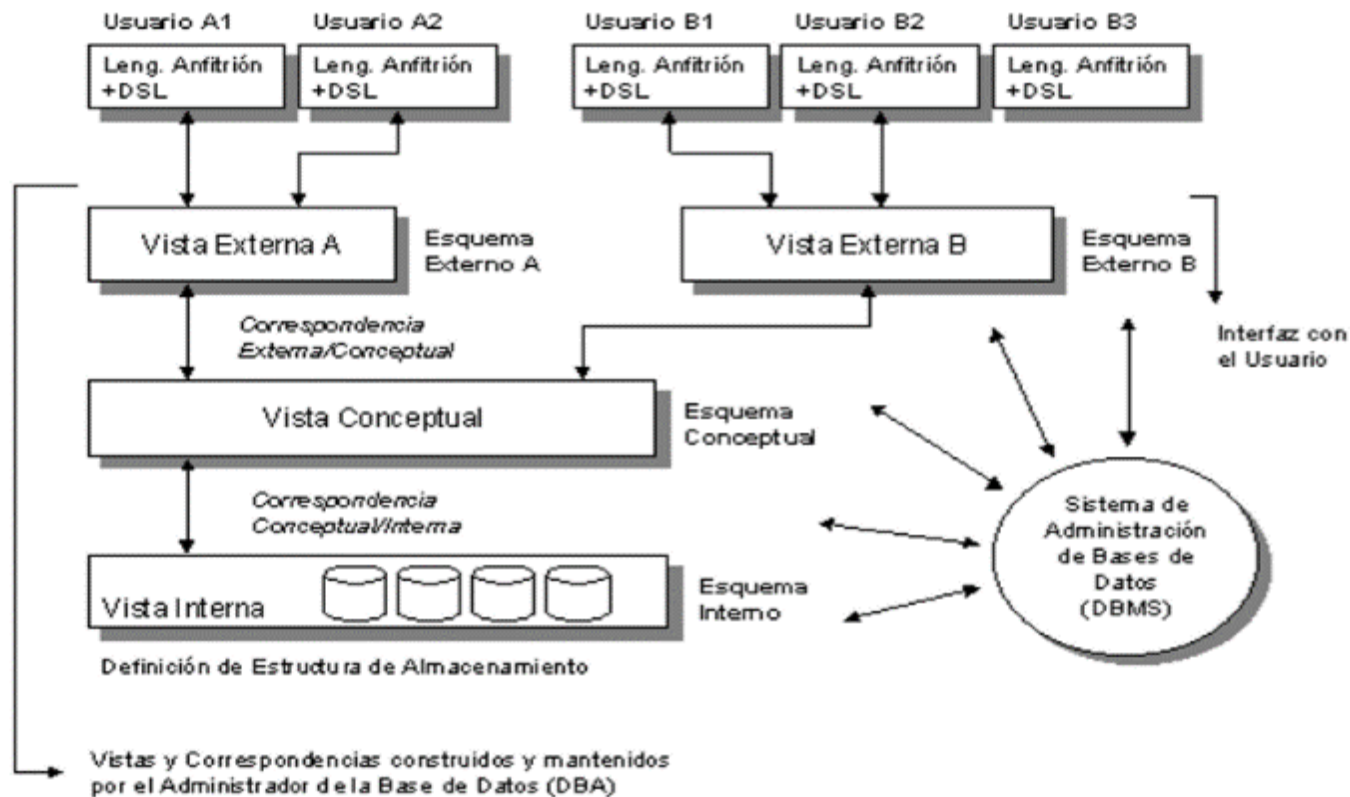
Componentes de los Sistemas de BD

Un sistema de bases de datos contempla los siguientes componentes:

- La base de datos
- El sistema de Gestión de Bases de Datos (SGBD o DBMS) o motor de bases de datos: SQL Server, Oracle, SyBase, MySQL, Interbase, DBase.
- Programas de Aplicación
- Conjunto de usuarios (finales, DBA, programadores)
- Hardware
- Programas utilitarios (Generadores de informes, backup, herramientas de desarrollo, etc.)

Componentes de los Sistemas de BD

Esquema general de los componentes de un Sistema de Bases de Datos



DSL: Sub Lenguaje de Datos – DDL: Lenguaje de Definición de Datos – DML: Lenguaje de Manipulación de Datos.

Arquitectura general

Vista Externa:

Es una visión particular de un usuario o grupo de usuarios de la Base de Datos. El ***esquema externo*** representa una forma de definición o formalización de una vista externa.

Vista Conceptual:

Pretende ser la representación total y abstracta de los datos que componen la base, la formalización de esta se logra mediante el ***esquema conceptual***.

Vista Interna:

Es el nivel más bajo de la arquitectura y corresponde al almacenamiento físico de los datos de la base.

Funciones del DBA

El **Administrador de Bases de Datos** corresponde a la persona o grupo de personas encargada del control general del sistema. Sus funciones generales son:

- Decidir el contenido de la Base de Datos: la identificación de entidades de interés para la organización y los datos a registrar de éstas entidades. Luego se define el contenido de la Base de Datos generando un Modelo Conceptual.
- Decidir la estructura de almacenamiento y la estrategia de acceso: decidir como deben representarse los datos en forma interna.
- Vincularse con los usuarios: garantizar la existencia de los distintos esquemas externos.
- Definir los controles de autorización y procedimientos de validación: Seguridad.
- Definir una estrategia de respaldo y recuperación.
- Controles de desempeño y responder a los cambios de requerimiento: la idea aquí es lograr un desempeño aceptable, según expectativas, del Sistema mediante mecanismos de control.

El Sistema de Gestión de Bases de Datos

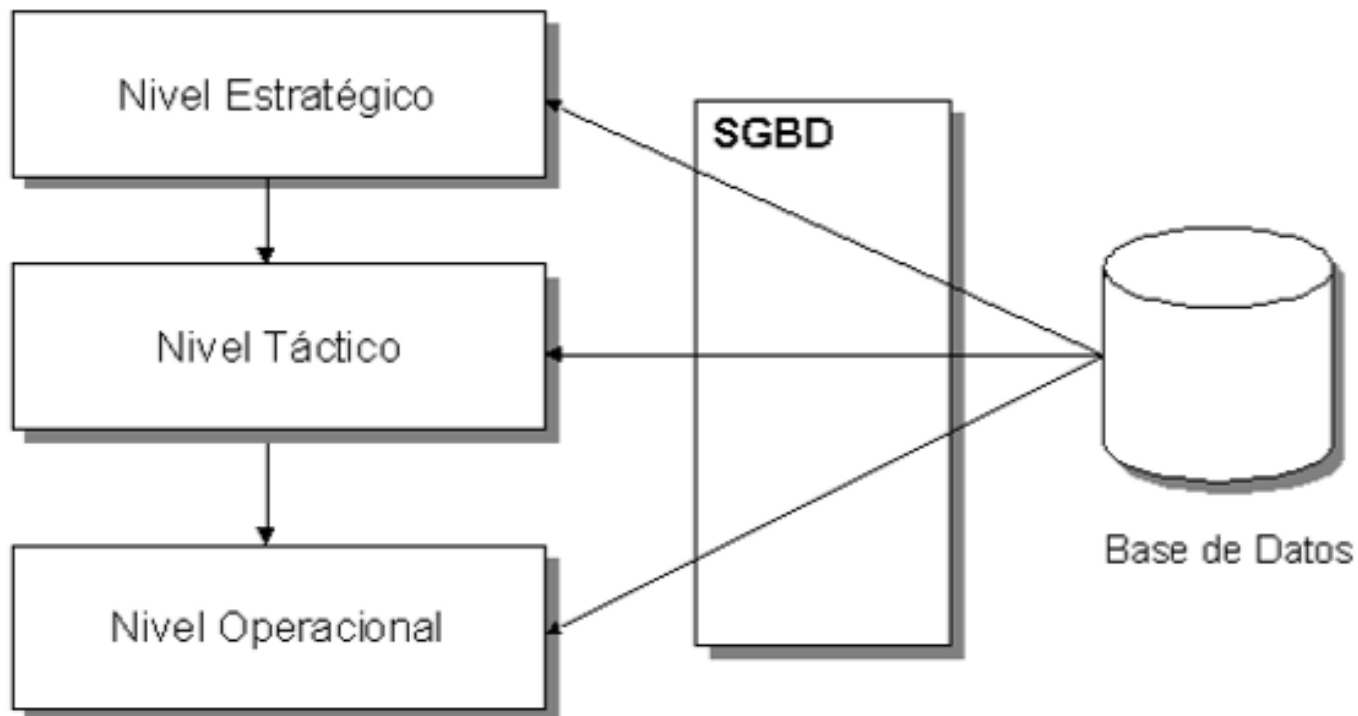
Se trata de una colección de datos interrelacionados y un conjunto de programas para acceder a esos datos.

El objetivo principal de un **SGBD** es proporcionar un entorno que sea a la vez conveniente y eficiente para ser utilizado al extraer y almacenar información en la base de datos. Manipulación. Provee lenguaje, herramientas y funciones específicas para llevar adelante estas tareas.

Motor de Base de datos

Definición del SGBD

El **SGBD** es un conjunto coordinado de programas, procedimientos y lenguajes que suministra, tanto a usuarios no informáticos como a los analistas, programadores o al administrador (DBA), los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad



Funciones del SGBD

De Descripción o Definición

Debe proveer al DBA herramientas que le permitan especificar la información diseñada en el ***esquema conceptual***.

Debe proveer un lenguaje de definición de datos (LDD - DDL), propio del SGBD, y debe ser capaz de definir las estructuras de datos a los tres niveles (nivel externo, nivel lógico global o conceptual y nivel interno).

Funciones del SGBD (De Descripción o Definición)

A nivel interno se define:

- Espacio reservado para la base (volúmenes, cilindros y pistas)
- Longitud de los campos
- Modo de representación de los datos (binario, decimal, alfanumérico, etc.)
- Caminos de acceso como punteros e índices.

Funciones del SGBD (De Descripción o Definición)

A nivel externo y conceptual

La función de descripción proporciona los instrumentos para la definición de entidades, su identificación, atributos, interrelaciones entre ellas, autorizaciones de acceso, restricciones de integridad, esquemas, etc.

El **SGBD**, además de describir, debe permitir la correspondencia o mapping entre estos niveles.

Funciones del SGBD

De Manipulación

Permite a los usuarios de la base (todos) buscar, eliminar o modificar los datos de la base, de acuerdo a las especificaciones y normas de seguridad dadas por el administrador.

Esto se realiza mediante el lenguaje de manipulación de datos (LMD - DML), mediante un conjunto de instrucciones (lenguaje huésped) que son admitidas por un lenguaje de programación (lenguaje anfitrión), o bien, mediante un lenguaje autocontenido, que posee todas las instrucciones necesarias para llevar a cabo estas tareas.

Funciones del SGBD

De Utilización

Reúne todas las interfaces que necesitan los diferentes tipos de usuarios para comunicarse con la base y proporciona un conjunto de procedimientos para el administrador.

Algunas de estas funciones de servicio son:

- cambiar capacidades de los archivos
- obtener estadísticas de utilización
- Respaldos
- cargar y descarga de la base
- seguridad, etc.

Lenguajes de los SGBD

El SGBD debe proveer diferentes tipos de lenguajes según las operaciones que deba realizar.

Por tipo de función, tendremos lenguajes de definición y de manipulación.

Tanto el DSL como el **DDL** (data definition language o LDD) son lenguajes internos del SGBD, dedicado a la definición de la base de datos. Como palabras reservadas se encuentran: *CREATE*, *ALTER*, *DROP*.

El **DML** (data manipulation language) es el más significativo a nivel usuario ya que permite interactuar directamente con la manipulación de los datos. Como palabras reservadas se encuentran: *INSERT*, *UPDATE*, *DELETE* y *SELECT*.

Lenguajes de los SGBD - DML

Para cumplir los objetivos asignados a la función de manipulación, se ha de contar con lenguajes que den a los usuarios la posibilidad de referirse a determinados conjuntos de datos que cumplan ciertas condiciones (criterio de selección).

El SQL (Structured Query Language) como lenguaje de manipulación de datos tiene la propiedad dual, es decir, puede actuar como huésped o autocontenido.

A través de sentencias específicas se indica al SGDB la tarea a realizar, ya sea de consulta como de actualización de datos.

Diseño conceptual

Mediante técnicas de relevamiento, se establece conceptualmente cuál es la estructura que debería tener la DB.

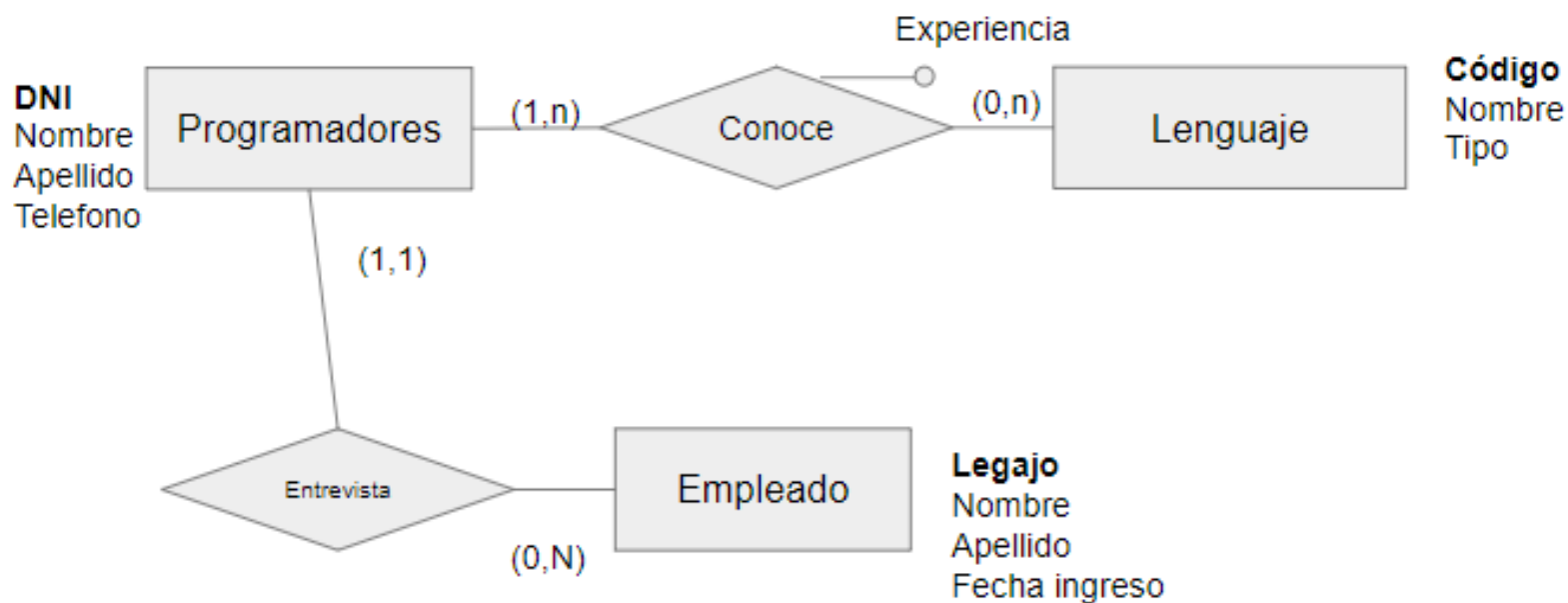
- Idea intuitiva: Obtener la información necesaria para saber que se desea que represente la BD.
- Relevamiento: Suele realizarse mediante consulta con empleados, etc.
- Ejemplo de una biblioteca:
 - Fichas con los atributos de los libros: signatura topográfica, título, autor, género, etc.
 - Fichas con los datos de los lectores: nombre y apellido, documento, domicilio, etc.
 - Fichas de pedidos: Fecha del pedido, código del libro, fecha devolución, código de lector.
 - Información adicional:
 - De cada libro pueden existir varios ejemplares.
 - Se desea información del idioma de un libro.
 - Se desea reflejar los temas que trata un libro.
 - Se desea conocer el nombre de los autores.

Diseño conceptual

Para definir un diseño conceptual de una base de datos utilizamos el DER:

- Cuadrados para las entidades: elementos que existen realmente, tanto físicos (autores, libros) como lógicos (idiomas, temas).
- Rombos para las relaciones entre entidades: La cardinalidad indica las posibilidades de relación entre entidades:
 - 1:1 Una entidad A con una entidad B y viceversa.
 - 1:N Una entidad A con N entidades B, pero solo una entidad B con una entidad A.
 - N:M N entidades A con M entidades B y viceversa.

Diseño conceptual



Diseño lógico

La conversión del diseño conceptual al lógico se basa en tres reglas básicas:

- Toda entidad se convierte en tabla.
- Toda relación 1:N se convierte en una propagación de clave (primaria o foránea).
 - Excepcionalmente se crea una tabla intermedia.
- Toda relación N:M se convierte en una tabla intermedia.

Esquema interno – Nivel Interno

Este esquema es dependiente del SGBD.

Sin embargo, existen elementos comunes que son:

- Estrategia de almacenamiento.
- Camino de acceso: claves primarias, secundarias, índices.
- Técnicas de compresión de datos.
- Técnicas de criptografía.
- Correspondencia conceptual/interna: Representación de los registros.
- Técnicas de Tuning y optimización.
- Dispositivos de memoria.
- Organizaciones físicas.
- Control de acceso.

Esquema interno – Ejemplo

Del DER al SQL Server



Microsoft
SQL Server Management Studio

v18.12.1

© 2022 Microsoft.
All rights reserved.

Esquema interno – Ejemplo

Numéricos exactos - Enteros

Bigint, numeric, bit, smallint, Int, tinyint, integer

Numéricos aproximados - Reales

Float, decimal, smallmoney, money, real, Fecha y hora, date, datetimeoffset, datetime2, smalldatetime, datetime, time

Cadenas de caracteres

Char(longitud), varchar(, text

Cadenas de caracteres Unicode

Nchar, nvarchar, ntext

Cadenas binarias

Binario, varbinary, imagen

Para más info:

https://www.w3schools.com/sql/sql_datatypes.asp

Otros tipos de datos

Cursor, rowversion, hierarchyid, uniqueidentifier, sql_variant, xml, Tipos de geometría espacial, Tipos de geografía espacial, table

Teoría de Normalización

Antes del método de Codd, año 1970 (creador del modelo relacional), las bases de datos se almacenaban en un solo archivo.

- En un diseño de una BD. se obtienen unas tablas, pero no podemos estar seguros de que no presenten problemas:
 - Incapacidad de almacenar ciertos hechos.
 - Redundancias.
 - Ambigüedades.
 - Pérdida de información.
- Las reglas formales que forman la teoría de la normalización permiten detectar y corregir esos errores.
- Existen 6 FN, aunque lo normal es aplicar las 3 primeras FN.

Teoría de Normalización

La normalización es el proceso mediante el cual se transforman datos complejos a un conjunto de estructuras de datos más pequeñas, que además de ser más simples y más estables, son más fáciles de mantener.

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos.

Objetivos de la Normalización

- Minimizar la redundancia.
- Minimizar el mantenimiento de datos.
- Minimizar el impacto de futuros cambios (anomalías de actualización y anomalías de borrado) de datos, e ingreso de información (anomalías de inserción).

Ventajas de la Normalización

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.
- Facilidad de uso.
- Flexibilidad.
- Precisión.
- Seguridad. Facilidad de implementación.
- Independencia de datos.
- **Claridad.**
- Facilidad de gestión.
- Mínima redundancia.
- Máximo rendimiento de las aplicaciones.

Teoría de Normalización

Existen 5 Formas Normales y una variante de la 4ta:

- Primera Forma Normal (1FN)
- Segunda Forma Normal (2FN)
- Tercera Forma Normal (3FN)
- Forma Normal de Boyce Codd (FNBC)
- Cuarta Forma Normal (4FN)
- Quinta Forma Normal (5FN)

Teoría de Normalización

Tomaremos el siguiente ejemplo para ver las formas normales y reconocer el funcionamiento de cada una de ellas:

FACTURACIÓN

Codigo_factura

Codigo_cliente
Nombre_cliente
Direccion_cliente
Localidad_cliente
Fecha_factura
Forma_pago
Codigo_articulo_1
Descripcion_1
Cantidad_1
Importe_1
Tipo_IVA_1
...
Codigo_articulo_N
Descripcion_N
Cantidad_N
Importe_N
Tipo_IVA_N

1FN (Primera Forma Normal)

- Una base de datos esta en 1FN si:
 - Cada atributo de una tabla contiene un valor atómico (simple).

FACTURA

★ <u>Codigo_factura</u>
Codigo_cliente
Nombre_cliente
Direccion_cliente
Localidad_cliente
Fecha_factura
Forma_pago

Clave primaria

DETALLE_FACTURA

★ <u>Codigo_factura</u>
★ <u>Codigo_articulo</u>
Descripcion
Cantidad
Importe
Tipo_IVA

Definición formal: “ una relación está en 1FN, si cumple la propiedad de que sus dominios no tienen elementos que, a su vez, sean conjuntos.”

2FN (Segunda Forma Normal)

Una base de datos esta en 2FN si:

- Esta en 1FN.
- Cada atributo **no clave** depende de la clave completa y no de parte de ella.
 - Toda tabla con clave, formada por un solo atributo cumple con esta propiedad.

FACTURA

Codigo_factura
Codigo_cliente
Nombre_cliente
Direccion_cliente
Localidad_cliente
Fecha_factura
Forma_pago
Tipo_IVA

DETALLE_FACTURA

Codigo_factura
Codigo_articulo
Cantidad
Importe

ARTICULO

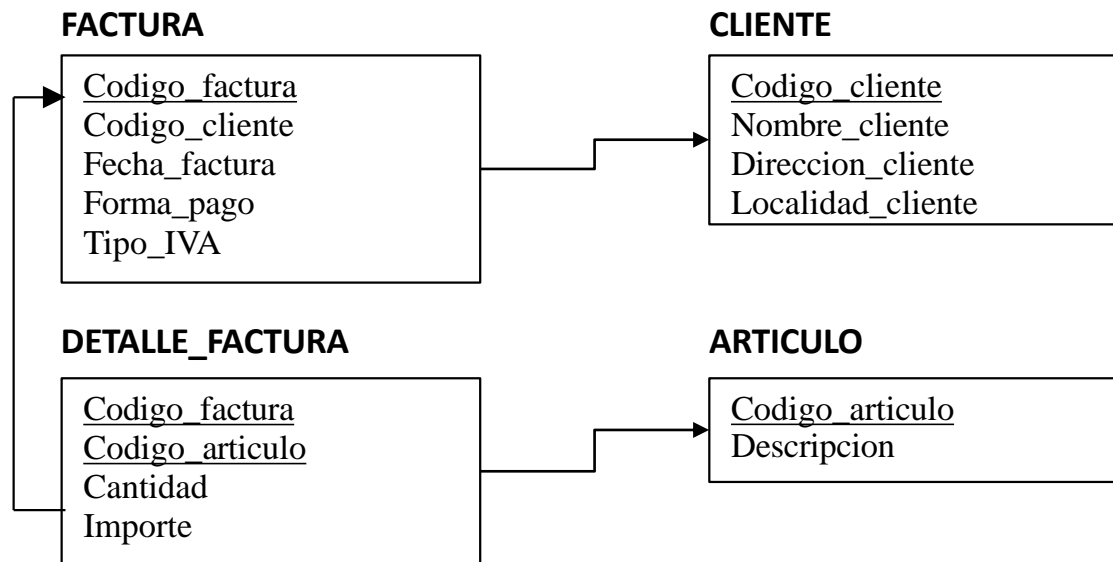
Codigo_articulo
Descripcion

La definición formal es: “se dice que una relación R está en 2FN, si además de estar en 1FN, cualquiera de sus atributos no-primarios tienen una dependencia funcional plena con cada una de las claves candidatas de R”.

3FN (Tercera Forma Normal)

Una base de datos esta en 3FN si:

- Esta en 2FN.
- Todos los atributos que no son claves son independientes entre si.



La definición formal es: “una relación R está en 3FN si esta en 2FN y además ninguno de sus campos no-primarios tiene dependencias transitivas respecto de las claves candidatas de R.”

Otras formas normales

Investigar desde la teoría del aula virtual

- *Archivo: 05 - Teoria_de_Normalizacion.pdf*
- Forma Normal de Boyce Codd (FNBC)
- Cuarta Forma Normal (4FN)
- Quinta Forma Normal (5FN)

Consideraciones Finales

Si se realiza un diseño conceptual y lógico:

- Aplicar las formas normales nos comprueba que es correcto.
- Ayuda a corregir los errores existentes.

Sin embargo, las FN implican descomponer tablas en otras más pequeñas.

- Problema de integridad de la base de datos.
- Disminución del rendimiento del sistema.

Por tanto es necesario llegar a un punto intermedio, que garantice la “normalización de los datos” a un costo operativo aceptable.

Claves en una BD

Una clave en una base de datos es un atributo o conjunto de atributos que identifican a un registro.

Clave Primaria

La clave primaria de una relación es un atributo o conjunto de atributos que identifican unívocamente a una tupla. Es decir, que todos los elementos del conjunto son únicos por cada registro.

Cientes
* Código
Nombre y Apellido
CUIT
Domicilio

Items de factura
* Codigo_factura
* Item_factura
Cantidad
Precio

Claves en una BD


Clave Candidata

La clave candidata es aquella columna que cumple todos los requisitos de una clave principal. En otras palabras, tiene potencial para ser una clave principal.

Cientes
* Codigo
Nombre y Apellido
CUIT
Domicilio



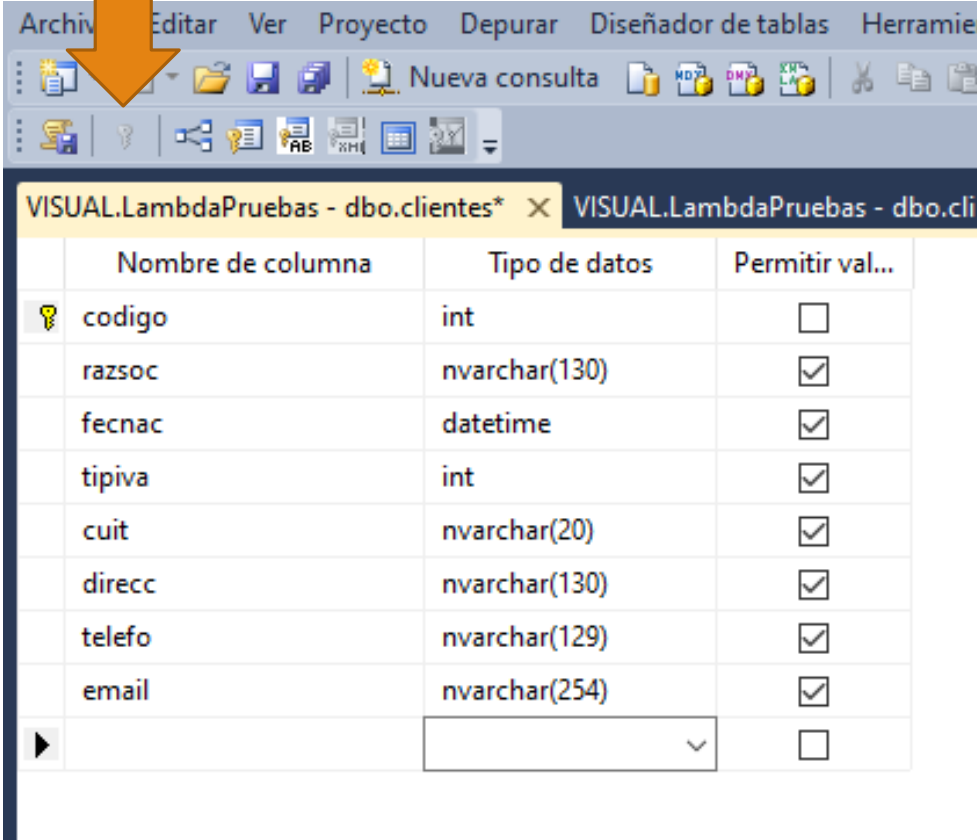
Tener en cuenta cada implementación. Puede que la CUIT cambie para un mismo cliente. En este caso dejaría de ser útil como clave primaria



En este ejemplo, CUIT podría reemplazar a **Codigo** como clave primaria ya que *podría* identificar al registro unívocamente.

Claves en una BD

Creación de una clave primaria en SQL Server



The screenshot shows the SQL Server Enterprise Designer interface. A large orange arrow points to the 'Diseñador de tablas' (Table Designer) tab in the menu bar. Another large orange arrow points to the 'clientes' table in the table list on the left. The table structure is displayed in the main area, showing columns and their data types. The 'codigo' column is marked with a primary key icon (a yellow key).

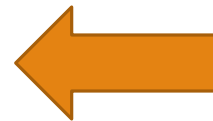
	Nombre de columna	Tipo de datos	Permitir val...
🔑	codigo	int	<input type="checkbox"/>
	razsoc	nvarchar(130)	<input checked="" type="checkbox"/>
	fecnac	datetime	<input checked="" type="checkbox"/>
	tipiva	int	<input checked="" type="checkbox"/>
	cuit	nvarchar(20)	<input checked="" type="checkbox"/>
	direcc	nvarchar(130)	<input checked="" type="checkbox"/>
	telefono	nvarchar(129)	<input checked="" type="checkbox"/>
	email	nvarchar(254)	<input checked="" type="checkbox"/>
▶			<input type="checkbox"/>

Claves en una BD

Creación de una clave primaria a través del lenguaje de manipulación:

```
CREATE TABLE [usuario].[clientes] (  
    [codigo] [int] NOT NULL,  
    [razsoc] [nvarchar](130) NULL,  
    [fecnac] [datetime] NULL,  
    [tipiva] [int] NULL,  
    [cuit] [nvarchar](20) NULL,  
    [direcc] [nvarchar](130) NULL,  
    [telefono] [nvarchar](129) NULL,  
    [email] [nvarchar](254) NULL,  
    CONSTRAINT [PK_clientes] PRIMARY KEY CLUSTERED  
    (  
        [codigo] ASC  
    )  
)
```

Es el nombre de usuario de la tabla. Por defecto toma el usuario logeado



Claves en una BD

Una clave primaria, como se mencionó anteriormente, debe identificar unívocamente a un registro, por lo que de ser necesario, esta puede estar compuesta por más de un registro.

```
CREATE TABLE [usuario].[items_factura](  
    [codigo_factura] [int] NOT NULL,  
    [item_factura] [int] NOT NULL,  
    [cantidad] [float] NULL,  
    [codigo_articulo] [int] NOT NULL,  
    [precio_unicatio] [money] NULL,  
CONSTRAINT [PK_items_factura] PRIMARY KEY CLUSTERED  
(  
    [codigo_factura] ASC, [item_factura] ASC)  
)
```



Claves en una BD

Clave secundaria o índice secundario

Al igual que se estudió en archivos indexados, un índice es una estructura que mantiene ordenada según un criterio a la tabla.

De esta manera, pueden existir claves secundarias, únicas o no, que permitan al SGBD ordenar una tabla por otro criterio que no sea el establecido por la propia clave primaria.

En el ejemplo de la derecha, **Código** es clave primaria e identifica unívocamente a la factura, pero el subconjunto {Tipo_comprobante, Letra, Sucursal, Número}, como superclave, también es única para la tabla.

Cabecera_Factura

* **Código**

Tipo_Comprobante

Letra

Sucursal

Número



Fecha_Emision

Cod_Cliente

Domicilio_Entrega

Claves en una BD

```
CREATE TABLE [dbo].[Cabecera_factura] (  
    [codigo] [int] NOT NULL,  
    [tipo_Comprobante] [char](2) not NULL,  
    [letra] [char](1) NULL,  
    [sucursal] [int] NULL,  
    [numero] [int] NULL,  
    [Fecha_Emission] [DateTime] NULL,  
    [Cod_cliente] [int] NOT NULL,  
    [Domicilio_Entrega] [nvarchar](254) NULL,  
  
    CONSTRAINT [PK_CabeceraFactura] PRIMARY KEY CLUSTERED  
    ( [codigo] ASC)  
)  
  
CREATE UNIQUE INDEX ix_numeroFactura  
    ON Cabecera_Factura (tipo_comprobante, letra,  
                        sucursal, numero)
```



Cabecera_Factura

*** Codigo**

Tipo_Comprobante

Letra

Sucursal

Número

Fecha_Emission

Cod_Cliente

Domicilio_Entrega

Claves en una BD

Por otro lado, cómo se mencionó anteriormente, un índice no necesariamente es único, sino que puede ayudarnos con la performance de una consulta.

Si el sistema cuenta con un listado tipo hoja de ruta en donde siempre se ordena por Domicilio de Entrega, entonces podremos definir un índice para que cree un árbol ordenado por este campo:

```
CREATE INDEX ix_Domicilio_Entrega  
ON Cabecera_Factura (Domicilio_Entrega)
```



Cabecera_Factura

*** Código**

Tipo_Comprobante

Letra

Sucursal

Número

Fecha_Emision

Cod_Cliente

Domicilio_Entrega

Claves en una BD

Claves foráneas o ajenas

Una clave de este tipo es un atributo dentro de la tabla que apunta a un registro único en otra tabla.

En nuestro ejemplo, **Cod_Cliente** hace referencia a la clave primaria de la tabla ***Clientes***. En este caso se puede definir una clave foránea que nos ayudará a mantener la consistencia entre los datos.

```
ALTER TABLE [Cabecera_factura]
ADD CONSTRAINT FK_Cod_Cliente FOREIGN KEY (Cod_cliente)
REFERENCES Clientes (Codigo)
ON DELETE CASCADE
ON UPDATE CASCADE
;
```



Cabecera_Factura

* **Codigo**

Tipo_Comprobante

Letra

Sucursal

Número

Fecha_Emision

Cod_Cliente

Domicilio_Entrega

Claves en una BD – Claves foráneas

```
ALTER TABLE [Cabecera_factura]
ADD CONSTRAINT FK_Cod_Cliente FOREIGN KEY (Cod_cliente)
REFERENCES Clientes (Codigo)
ON DELETE CASCADE
ON UPDATE CASCADE
;
```

Cómo se muestra en la sintaxis SQL, existen dos atributos importantes (Restricciones) en la creación de una clave foránea que indican qué debe hacer el SGBD ante un caso de actualización o eliminación del atributo: ***ON DELETE***, ***ON UPDATE***.

Especificando el atributo **NO ACTION**, se indica que si intentamos eliminar o actualizar un valor de la clave primaria de la tabla referenciada (Clientes) que tengan referencia en la tabla principal (En nuestro ejemplo Cabecera_Factura), se genere un error y la acción no se realice; es la opción predeterminada.

Especificando **CASCADE**, se indica que si eliminamos o actualizamos un valor de la clave primaria en la tabla referenciada (Clientes), los registros coincidentes en la tabla principal (Cabecera_Factura), también se eliminan o modifiquen; es decir, si eliminamos o modificamos un valor de campo definido con una restricción "primary key" o "unique", dicho cambio se extiende al valor de clave externa de la otra tabla (integridad referencial en cascada).