



MODELOS DE DATOS

Para introducirnos en el tema, empezaremos definiendo que es un modelo a partir de lo expuesto por varios autores.

“Un modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia.” [Silberschatz et al., 2003].

“Un modelo de datos es una notación para describir información acerca de los datos.” [García-Molina et al., 2009].

“Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos.” [Mercedes Marqués., 2011].

En otras palabras, un modelo de datos es una representación visual que describe las estructuras de datos. Los modelos de datos conceptuales presentan ideas que son fácilmente comprensibles para los usuarios, mientras que los modelos físicos detallan la manera en que los datos se almacenan en un sistema computacional.

Los modelos de datos se dividen en tres grupos:

- ❖ Lógico basado en objetos.
- ❖ Lógico basado en registros.
- ❖ Físicos de datos.

MÓDELO LÓGICO BASADO EN OBJETOS


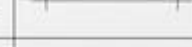
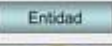

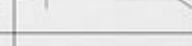


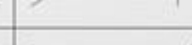



Se usan para describir datos en los niveles conceptuales, es decir, con este modelo representamos los datos de tal forma como nosotros los captamos en el mundo real, tienen una capacidad de estructuración bastante flexible y permiten especificar restricciones de datos explícitamente. Existen diferentes modelos de este tipo, pero nosotros utilizaremos el *MODELO ENTIDAD-RELACIÓN*.

Los modelos más conocidos son:

- Modelo entidad-relación.
- Modelo orientado a objetos.
- Modelo binario.
- Modelo semántico de datos
- Modelo funcional de datos.

El **modelo entidad-relación** (denominado por sus siglas DER) es ampliamente utilizado para el diseño conceptual de bases de datos. Peter Chen lo introdujo en 1976. Este modelo se compone de un conjunto de conceptos que permiten describir la realidad a través de representaciones gráficas y lingüísticas. Representa a la realidad a través de **entidades** que son objetos que existen y que se distinguen de otros por sus características.

Utiliza las simbologías descriptas a continuación.

Cardinalidad	Chen		Elemento	Símbolo
Uno a uno (1:1)	1  1		Entidad	
Uno a muchos (1:N)	1  N		Atributo	
Muchos a uno (N:1)	N  1		relación	
Muchos a muchos (M:N)	M  N			

En la antigüedad el modelo entidad-relación solo contenía las entidades, atributos y relaciones. Con el paso del tiempo, el diagrama fue mutando acorde a las necesidades actuales hasta llegar al denominado hoy día como “*modelo entidad-relación extendido*” que integra atributos compuestos y jerarquías de generalización.

MÓDELO LÓGICO BASADO EN REGISTROS

Se utilizan para describir datos en los niveles conceptual y físico. Estos modelos utilizan registros e instancias para representar la realidad, así como las relaciones que existen entre estos registros. A diferencia de los modelos de datos basados en objetos, se usan para especificar la estructura lógica global de la base de datos y para proporcionar una descripción a nivel más alto de la implementación.

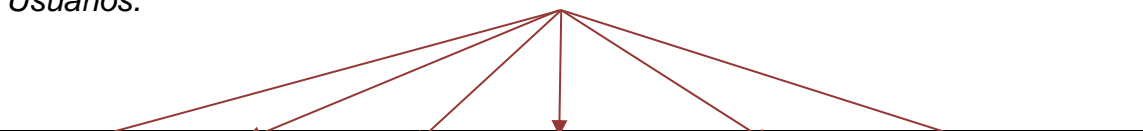
Los tres modelos más utilizados son los siguientes:

- Modelo relacional.
- Modelo de red.
- Modelo jerárquico.

El *modelo relacional* representa los datos y las relaciones entre estos a través de una colección de tablas, en las cuales los renglones (tuplas) equivalen a cada uno de los registros que contendrá la base de datos y las columnas corresponden a las características (atributos) de cada registro localizado en la tupla.

Supongamos que tenemos una tabla de *Usuarios*:

Cada una de las columnas representa a los atributos de la entidad *Usuarios*.



id	username	name	email	password	twoFactor
1	rotnic	Nicolás	nico@nico.com
2	palfer	Ferpa	ferpa@ferpa.com

Cada una de las filas contiene la información de cada uno de los registros.

Por otro lado, tenemos la entidad de *Productos*:

id	nombre	precioVenta	codigoProducto	updatedAt
1	SAPHIRUS AEROSOL	\$1000	321456A	01/05/2024
2	SAPHIRUS DIFUSOR	\$2400	321456D	01/05/2024

Ahora nos preguntamos... cómo relacionamos una tabla con la otra?

Para ello, debemos en primer lugar definir lo que es una **clave primaria**: es un atributo el cual definimos como atributo principal, es una *forma única de identificar a un registro en una entidad*. En nuestro caso, el campo codigoProducto es único para cada registro.

Existen 2 maneras; primero veremos la correcta para representar la relación en estos 2 modelos, suponiendo que un usuario puede comprar muchos productos y un producto puede ser comprado por varios usuarios.

1. Realizamos una tabla intermedia que contenga cada una de las claves primarias de las entidades involucradas en la relación.

Entidad Producto_Usuario	
productoid	usuarioid
321456A	1
321456D	1

Supongamos ahora que un usuario solo puede adquirir solo un producto, algo ilógico pero ficticio para demostrar la ocasión.

2. Incluimos en la entidad correspondiente la **clave primaria** de la tabla con la que se relaciona.



id	username	name	email	pass	twoFactor	productoid
1	rotnic	Nicolás	nico@nico.com	321456A
2	palfer	Ferpa	ferpa@ferpa.com	321456D

El *modelo de red* representa los datos mediante colecciones de registros y sus relaciones se representan por medio de “ligas” o “enlaces”, los cuales pueden verse como punteros. Los registros se organizan en un conjunto de gráficas arbitrarias.

1	rotnic	Nicolás	321456A		1	321456A
2	palfer	Ferpa	321456A		2	321456A

Por último, el *modelo jerárquico*, es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros y sus enlaces. La diferencia radica en que están organizados por conjuntos de árboles en lugar de gráficas arbitrarias.

Nodo raíz	1	rotnic	Nicolás	321456A		1	321456A
	2	palfer	Ferpa	321456A		2	321456A

INSTANCIAS Y ESQUEMAS

A medida que la base de datos transcurre en el tiempo, va aumentando su volumen en cuanto a registros como así también, eliminando los inutilizables lo cual ocasiona que esta cambie constantemente.

Denominamos **instancia** al estado que presenta una base de datos en un tiempo dado. Sería como un “snapshot” o “fotografía” que le tomamos a la base de datos en un tiempo N; transcurrido este tiempo N la base de datos ya no es la misma.

Por otro lado, llamamos **esquema** a la descripción lógica de la base de datos, que proporciona los nombres de las entidades y sus atributos especificando las relaciones que existen entre ellos.

Considerando el ejemplo anterior del usuario y los productos, veamos como quedaría:

- **Esquema:**
 - { Usuarios: id, username, nombre, email, password, twoFactor }
 - { Productos: id, nombre, precioVenta, codigoProducto, updatedAt }
- **Instancia:**

1	rotnic	Nicolás	@...		1	SAPH...	\$1000	321456A
---	--------	---------	------	--	---	---------	--------	---------



Como podemos observar el esquema nos muestra la estructura en el cual se almacenarán los datos, en este caso en registros cuyos nombres de campos son: por parte del Usuario (id, username, nombre, email, password, twoFactor) y por el producto (id, nombre, precioVenta, codigoProducto, updatedAt). La instancia representa a una serie de datos almacenados en los registros establecidos por el esquema, estos datos varían, no permanecen fijos en el tiempo.

INDEPENDENCIA DE LOS DATOS

Se refiere a la protección contra los programas de aplicación que puedan originar modificaciones cuando se altera la organización física o lógica de la base de datos.

Existen 2 niveles de independencia de datos:

1. Independencia física de datos: La independencia física de datos se refiere a la capacidad de alterar el esquema físico de la base de datos —es decir, la estructura de almacenamiento físico de los datos— sin que sea necesario modificar los programas de aplicación que acceden a estos datos. Esto significa que los cambios en cómo se almacenan los datos (como cambiar de sistema de archivos, modificar índices o particiones) no afectarán a las aplicaciones que ejecutan consultas sobre esos datos. Los ajustes en el nivel físico están generalmente orientados a optimizar el rendimiento y la eficiencia del almacenamiento sin comprometer la integridad y la accesibilidad de los datos.
2. Independencia lógica de datos: La independencia lógica de datos es la capacidad de modificar el esquema conceptual —el diseño y la estructura lógica de la base de datos como las tablas, las relaciones entre ellas y los campos— sin necesidad de reescribir los programas de aplicación. Esto permite que las modificaciones necesarias para reflejar cambios en el dominio del problema, como agregar nuevas relaciones o atributos, se realicen sin afectar las aplicaciones existentes. Este nivel de independencia asegura que las aplicaciones no necesiten ser actualizadas cada vez que se realizan cambios en la estructura lógica de los datos, lo cual es crucial para mantener la flexibilidad y escalabilidad de los sistemas de información.

Ambos tipos de independencia son fundamentales para mantener la agilidad de los sistemas de bases de datos en organizaciones grandes, permitiendo que los ajustes necesarios debido a cambios en los requisitos del negocio o mejoras tecnológicas se implementen con mínimo impacto en las operaciones cotidianas.



SQL se puede dividir en dos lenguajes DDL (Data Definition Language) y DML (Data Manipulation Language). Cada parte tiene unos comandos para realizar operaciones de distinta naturaleza.

Un comando es una “orden” que le indicará a la base de datos lo que esperamos que haga.

- **DDL**: es el lenguaje de definición de datos que permite crear nuevas bases de datos, modificarlas y eliminarlas. En este nivel, es donde trabaja el administrador de base de datos. Estos comandos afectan a tablas, campos e índices.
 - CREATE (crea nuevas tablas, campos e índices)
 - ALTER (modifica las tablas)
 - DROP (elimina tablas e índices)
- **DML**: el lenguaje de manipulación de datos nos permite, tal cual indica su nombre, manipular (consultar, ordenar, filtrar, etc) los datos existentes en un almacén de datos. En este nivel es donde trabajan los usuarios.
 - SELECT (consulta filas en la DB)
 - UPDATE (modifica valores de las filas)
 - INSERT (inserta una nueva fila)
 - DELETE (elimina filas)

Existen 2 tipos de lenguajes de manipulación de datos:

- **Procedimentales**: los DML requieren que el usuario especifique que datos se necesitan y cómo obtenerlos.
- **No procedimentales**: los DML requieren que el usuario especifique que datos se necesitan y sin especificar cómo obtenerlos.

DATABASE MANAGEMENT SYSTEM (DBMS)

El DBMS o SGDB (Sistema de gestión de base de datos), es la porción más importante del software de un sistema de base de datos. Un DBMS es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

El hecho fundamental que justifica la utilización de un DBMS es que, si antes los programas llamaban directamente al sistema operativo para manejar sus ficheros, ahora es el DBMS es el encargado de facilitar los servicios de acceso de las aplicaciones a los datos.

Los programas piden al SGBD ciertos datos de la BD y éste, con la información y herramientas de que dispone, las convierte en operaciones de acceso a los distintos ficheros de la BD, operaciones que son ejecutadas por los métodos de acceso.



La figura muestra el DBMS como interface entre la base de datos física y las peticiones del usuario. El DBMS interpreta las peticiones de entrada/salida del usuario y las manda al sistema operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal.

Las características que definen a un SGBD, [Celma et al., 2003] son las siguientes:

a) *Integración de toda la información de una organización*: la base de datos se crea para dar servicio a toda o a una parte importante de la organización y no para unos usuarios particulares; de esta forma se evita la redundancia de datos dentro del sistema de información y los problemas de inconsistencia derivados de ella.

b) *Persistencia de los datos*: los datos deben estar disponibles en todo momento, lo que significa que la base de datos debe almacenarse en un dispositivo de memoria secundaria.

c) *Accesibilidad simultánea para distintos usuarios*: debido al carácter integrador que tiene la base de datos, ésta tendrá que ser compartida por distintos grupos de usuarios, lo que significa que estos podrán acceder simultáneamente a los datos.

d) *Independencia de los programas respecto a la representación física de los datos*: las aplicaciones que se desarrollen para manipular los datos deben ser independientes de la implementación elegida para las estructuras de la base de datos. A esta característica se le conoce como independencia de datos.

e) *Definición de vistas parciales de los datos para distintos usuarios*: debido también al carácter integrador de la base de datos, en ésta se recogen los datos que interesan a cada grupo de usuarios de la organización, con lo que se incrementa su tamaño y complejidad. Se debe permitir definir vistas parciales de la base de datos que contengan sólo aquellos datos que son relevantes para cada uno de los grupos.

f) *Mecanismos para controlar la integridad y la seguridad de los datos*: para que la base de datos refleje fielmente la realidad de la cual es una representación, el DBMS debe asegurar en todo momento la calidad de la información almacenada (integridad) evitando que ésta se deteriore por un uso incorrecto (actualizaciones que no son válidas, accesos concurrentes no controlados, etc.). Así mismo, debe asegurar que a la información almacenada sólo acceden las personas autorizadas y en la forma autorizada (seguridad).