V. 1.2

Diseño de Sistemas

Unidad 6: Modelado de Datos - parte 3

Martín Agüero
aguero.martin@gmail.com

UTN.BA
DPTO. INGENIERÍA EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DISEÑO DE SISTEMAS

Agenda

Unidad 6:

- Serialización. Precedencia.
- Tipos de Bases de Datos: Relacional, Orientada a Objetos, Multidimensionales, NoSQL: Clave Valor, Orientada a Columnas, Orientada a Documentos.
- Mapeo Objeto-Relacional.

Modelado de Datos

Serialización

Modelado de Datos

Serialización

El objetivo de un protocolo de control de concurrencia es programar transacciones de manera tal que no ocurra ninguna interferencia entre sí. Una solución consiste en permitir que sólo una transacción pueda ser ejecutada a la vez, es decir, una transacción es comprometida (commited) antes de la siguiente. No obstante, el objetivo de un DBMS multiusuario es maximizar el grado de concurrencia o paralelismo de un sistema, De modo tal que las transacciones se puedan ejecutar sin interferir con otra que se ejecuta en paralelo.

Serialización: es la técnica que permite identificar las transacciones que garantizan el aseguramiento de la consistencia (Papadimitriou, 1979).

Modelado de Datos

Serialización

Lost Update (Actualización perdida)

Time	T_1	T_2	bal_x
t_1		begin_transaction	100
t_2	begin_transaction	read(bal_x)	100
t_3	read(bal_x)	$bal_x = bal_x + 100$	100
t_4	$bal_x = bal_x - 10$	write(bal_x)	200
t_5	write(bal_x)	commit	90
t_6	commit		90

La pérdida de las actualizaciones es uno de los potenciales problemas causados por la concurrencia. ***Control de concurrencia*** es el proceso de gestionar operaciones simultáneas en la base de datos sin interferir una con la otra.

Modelado de Datos

Serialización

La programación serializada (serial schedule), asegura que las transacciones sean ejecutadas consecutivamente. La **serialización** nunca deja a la base de datos en un estado inconsistente y su objetivo es: Encontrar programas no-seriales que permitan a las transacciones ser ejecutadas de forma concurrente sin interferir una con la otra.

Por ejemplo: si un conjunto de transacciones se ejecuta de manera concurrente, se dice que la programación no-en-serie es correcta si produce los mismos resultados que la ejecución en serie. Esa programación se la denomina serialización.

Modelado de Datos

Serialización

En serialización el órden las operaciones de lectura y escritura es crítico, por ejemplo:

- ➔ Si dos transacciones sólo leen un ítem de datos, el órden no entra el conflicto.
- ➔ Si dos transacciones leen y escriben dos ítems de datos completamente distintos, tampoco entran en conflicto.
- ➔ En cambio, si una transacción lee un ítem de datos y otra lee y escribe el mismo ítem, el órden de la ejecución sí es crítico.

Modelado de Datos

Serialización

Time	T_7	T_8
t_1	begin_transaction	
t_2	read(bal_x)	
t_3	write(bal_x)	
t_4		begin_transaction
t_5		read(bal_x)
t_6		write(bal_x)
t_7	read(bal_y)	
t_8	write(bal_y)	
t_9	commit	
t_{10}		read(bal_y)
t_{11}		write(bal_y)
t_{12}		commit

(a)

T_7	T_8
begin_transaction	
read(bal_x)	
write(bal_x)	
	begin_transaction
	read(bal_x)
read(bal_y)	
	write(bal_x)
write(bal_y)	
commit	
	read(bal_y)
	write(bal_y)
	commit

(b)

T_7	T_8
begin_transaction	
read(bal_x)	
write(bal_x)	
read(bal_y)	
write(bal_y)	
commit	
	begin_transaction
	read(bal_x)
	write(bal_x)
	read(bal_y)
	write(bal_y)
	commit

(c)

Las transacciones T_7 y T_8 en (a) y (b) no han sido serializadas. Cambiando el órden de las operaciones no conflictivas, se consigue una equivalente programación serializada (c).

Modelado de Datos

Serialización

Precedencia

Bajo **reglas de restricción de escritura** (cuando una transacción actualiza un ítem previamente leído), se puede crear una visualización mediante un **grafo de precedencia** para evaluar si existe algún conflicto de serialización.

Un grafo de precedencia es un grafo dirigido G = (V,E) donde V son los vértices y E vértices dirigidos. Se construye de la siguiente manera:

- ➔ Se crea un nodo por cada transacción
- ➔ Se crea un vértice dirigido $T_i \rightarrow T_j$, si T_j lee el valor de un ítem escrito por T_i
- ➔ Se crea un vértice dirigido $T_i \rightarrow T_j$, si T_j escribe un valor en un ítem luego de haber sido leído por T_i
- ➔ Se crea un vértice dirigido $T_i \rightarrow T_j$, si T_j escribe un valor en un ítem luego de haber sido escrito por T_i

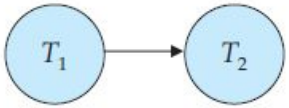
Serialización

Precedencia

Ejemplos

programa 1

T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit

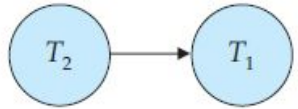


(a)

Todas las instrucciones de T_1 son ejecutadas antes de que la primer instrucción de T_2 sea ejecutada.

programa 2

T_1	T_2
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
	$B := B + temp$
	write(B)
	commit
read(A)	
$A := A - 50$	
write(A)	
read(B)	
$B := B + 50$	
write(B)	
commit	



(b)

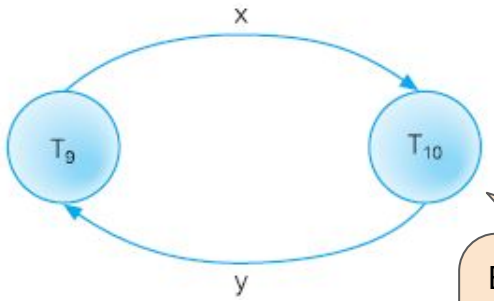
Todas las instrucciones de T_2 son ejecutadas antes de que la primer instrucción de T_1 sea ejecutada.

Serialización

Precedencia

Time	T_9	T_{10}
t_1	begin_transaction	
t_2	read(bal_x)	
t_3	$bal_x = bal_x + 100$	
t_4	write(bal_x)	begin_transaction
t_5		read(bal_x)
t_6		$bal_x = bal_x * 1.1$
t_7		write(bal_x)
t_8		read(bal_y)
t_9		$bal_y = bal_y * 1.1$
t_{10}		write(bal_y)
t_{11}	read(bal_y)	commit
t_{12}	$bal_y = bal_y - 100$	
t_{13}	write(bal_y)	
t_{14}	commit	

Ejemplo 3: Considerando dos transacciones T_9 y T_{10} donde T_9 transfiere \$100 de una cuenta con balance bal_x a otra cuenta con balance bal_y, mientras T_{10} está incrementando el balance de esas dos cuentas en un 10%.



En este caso el grafo de precedencia posee un ciclo, entonces no es serializable.

Serialización

Comportamiento Serializado

En la práctica es casi imposible requerir que las operaciones se ejecuten de forma serializada, por lo que se requiere contar con cierto nivel de paralelismo. Los DBMS adoptan un mecanismo de aseguramiento de la serialización, por más que la ejecución no sea en serie, el resultado es mostrado a los usuarios como si las operaciones fueran ejecutadas de esa forma.

Un enfoque común para los DBMS es bloquear (lock) los elementos de la base de datos, de modo tal que dos funciones no puedan acceder simultáneamente al mismo dato.

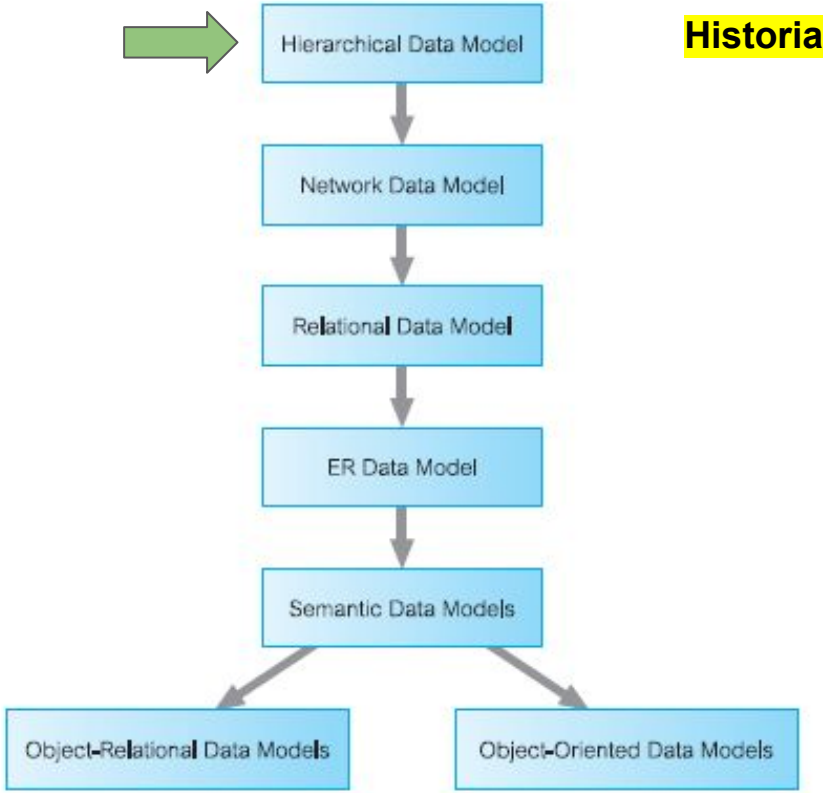
Modelado de Datos

Tipos de Bases de Datos

Modelado de Datos

Tipos de Bases de Datos

El primer enfoque estuvo basado en el modelo de datos jerárquico, denominado IMS (Information Management System) por IBM, en respuesta al enorme volumen de registro de datos generado por el programa espacial Apollo.

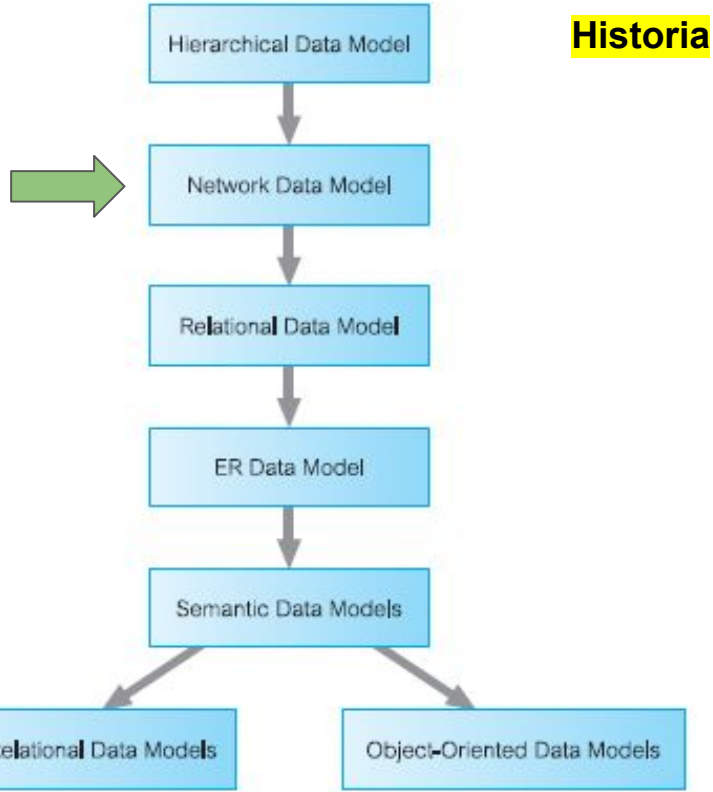


Modelado de Datos

Tipos de Bases de Datos

El segundo enfoque se basó en el modelo de datos de red, que intentó generar un estándar y resolver algunas dificultades del modelo jerárquico, como incapacidad para representar relaciones complejas de manera efectiva. Junto con el jerárquico, representaron la 1era generación de los DBMS.

Historia

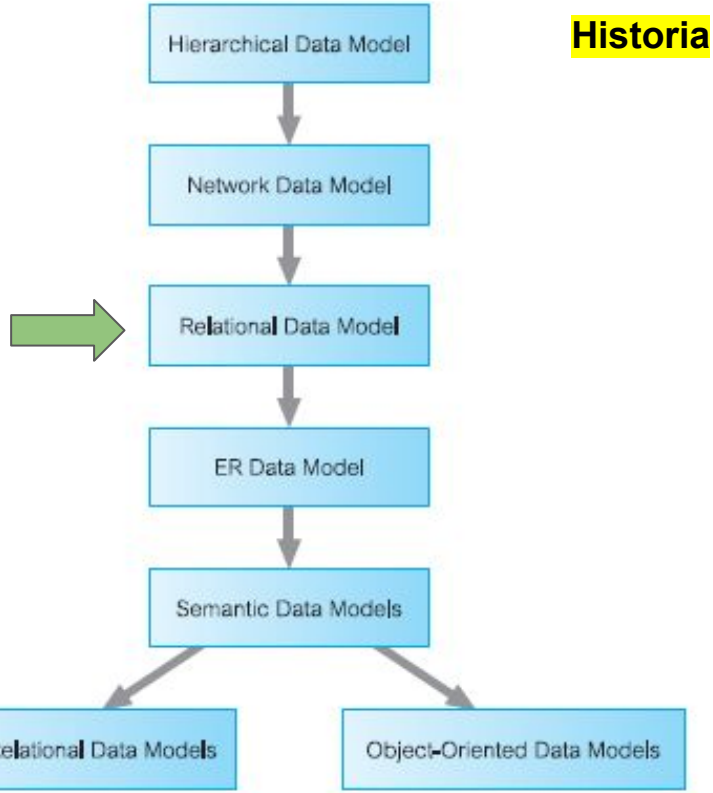


Modelado de Datos

Tipos de Bases de Datos

Historia

En 1970 Codd presentó en un paper el modelo de datos relacional. Luego de varias implementaciones experimentales, en los finales de los '70 y principios de los '80 aparecieron los primeros productos comerciales de DBMS relacionales. Estos se refieren como la 2da generación de sistemas de gestión de bases de datos.

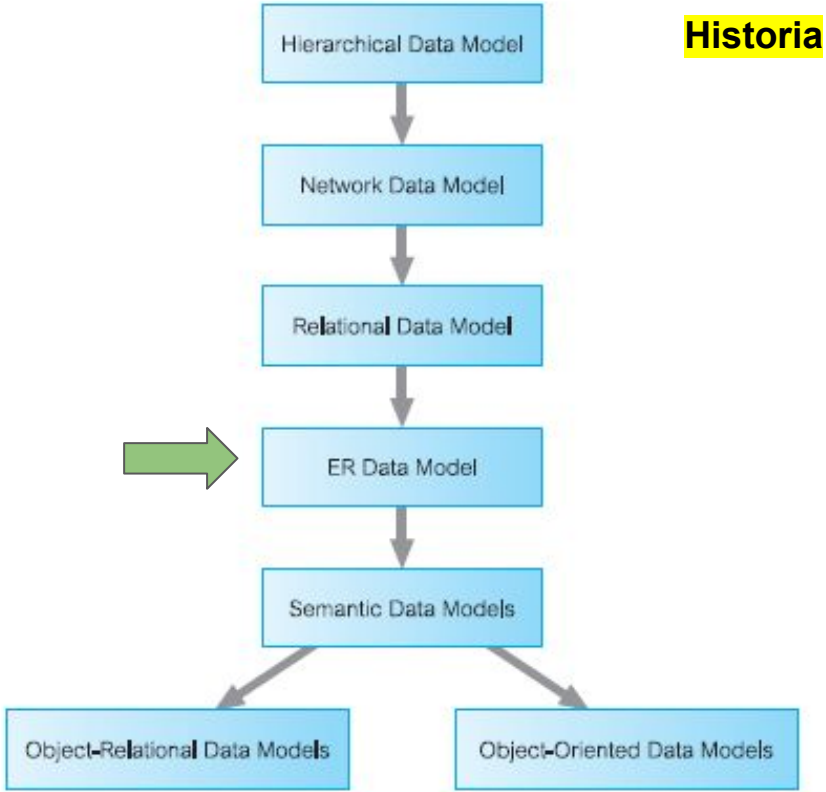


Modelado de Datos

Tipos de Bases de Datos

Historia

En 1976 Chen presentó el modelo Entidad-Relación que actualmente es una técnica muy extendida para diseñar de bases de datos.



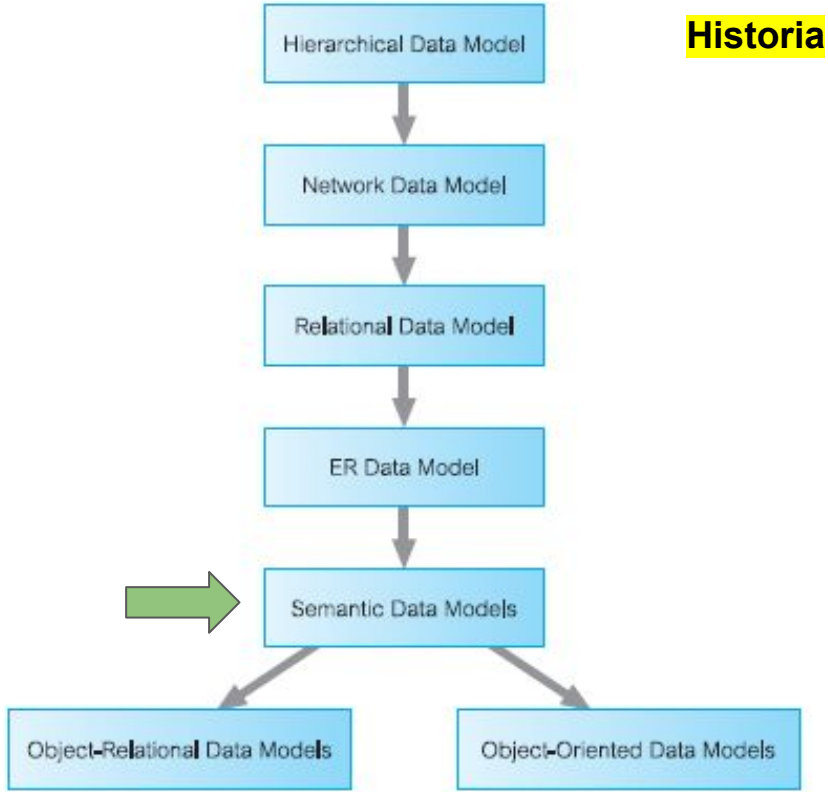
Modelado de Datos

Tipos de Bases de Datos

Historia

Los intentos de proveer un modelo de datos que represente de forma más aproximada el "mundo real" derivó en el modelo de datos semántico. Algunos de los modelos más famosos son:

- Modelo de Datos Semántico (Hammer and McLeod, 1981)
- Modelo de Datos Funcional (Shipman, 1981)
- Modelo de Asociación Semántica (Su, 1983)

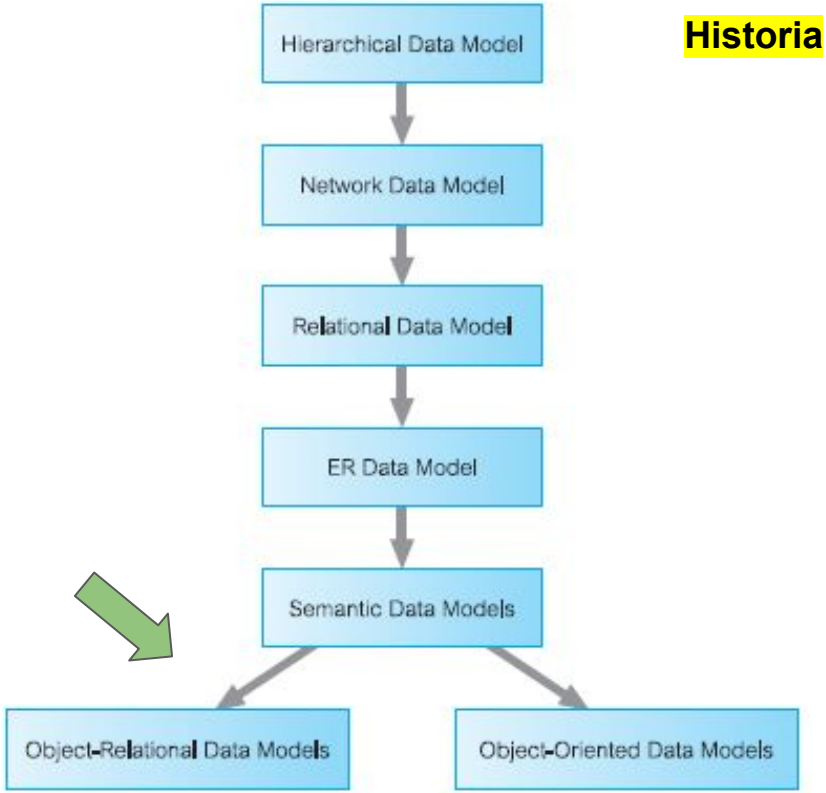


Modelado de Datos

Tipos de Bases de Datos

Historia

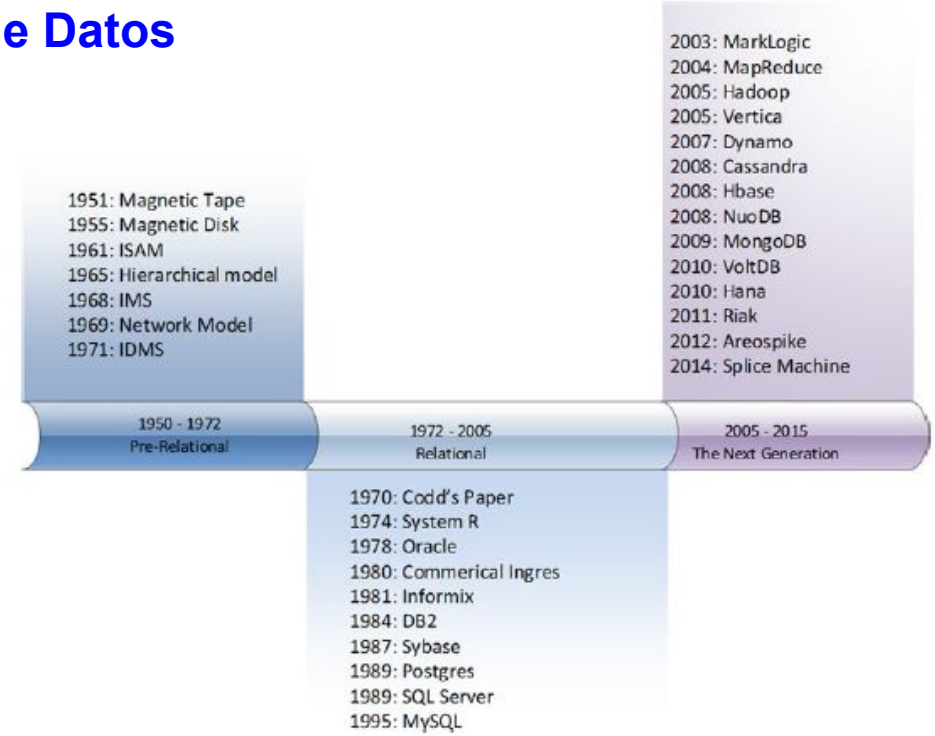
En respuesta a la creciente complejidad de las aplicaciones de bases de datos, surgieron dos "nuevos" modelos de datos:
el Modelo de Datos Orientado a Objetos (OODM) y el Modelo de Datos Objeto-Relacional (ERDM). Esta evolución representa la 3era generación de los DBMS.



Modelado de Datos

Tipos de Bases de Datos

Historia



Modelado de Datos

Tipos de Bases de Datos

Modelos

Numerosas áreas de aplicación de las bases de datos están limitadas por las restricciones que presenta el **modelo relacional**. Como resultado, investigadores han desarrollado otros modelos de datos (en muchos basos basados en el enfoque de orientación a objetos) para dar soporte a estos dominios.

El **modelo objeto-relacional** combina características del relacional y el OO. Provee una amplia variedad de tipos y lo combina con los fundamentos de persistencia del modelo relacional. También contempla características como herencia.

Por otro lado, también existe el **modelo semiestructurado**, soportado por el lenguaje XML, que originalmente diseñado como un modo de agregar metadatos a los documentos de texto. XML provee una forma de representar los datos mediante una estructura anidada y permite dar gran flexibilidad a datos estructurados.

Modelado de Datos

Tipos de Bases de Datos

Relacional

El modelo relacional, es actualmente el modelo de datos más utilizado para el procesamiento de datos de aplicaciones. Conserva esta posición debido a su simplicidad dado que básicamente consiste en colecciones de tablas con columnas y filas. Además de la simplicidad su principales fortalezas son: idoneidad para **OLTP** (On Line Transaction Processing), soporte para la independencia de los datos y un fuerte fundamento teórico. Sin embargo, los RDBMS son inadecuados para aplicaciones como por ejemplo:

- Diseño asistido por computadora (CAD)
- Manufactura asistida por computadora (CAM)
- Ingeniería de software asistido por computadora (CASE)
- Sistemas de gestión de redes (NMS)
- Sistemas de oficina (OIS) y multimedia.
- Publicación digital
- Sistemas de información geográfica (GIS)
- Sitios web dinámicos e interativos

Modelado de Datos

Tipos de Bases de Datos

Relacional

Como debilidades de los RDBMS se pueden destacar:

- ➔ Pobre representación de las entidades del "mundo real"
- ➔ Sobrecarga de semántica
- ➔ Débil soporte a la integridad o restricciones generales
- ➔ Estructura de datos homogénea
- ➔ Operaciones limitadas
- ➔ Dificultad para manejar solicitudes recursivas
- ➔ Falta de concordancia (impedance mismatch)

Modelado de Datos

Tipos de Bases de Datos

Orientada a Objetos

Una base de datos orientada a objetos (OO) puede ser definida como una colección de objetos, a la cual se le agregan características de persistencia. En otras palabras, es una colección a la que se debe acceder protegiendo en todo momento las propiedades A.C.I.D. (al igual que el modelo relacional).

Atomicidad (Atomicity): La atomicidad es la propiedad que establece que se ejecuta todo como una unidad indivisible o no se ejecuta nada.

Consistencia (Consistency): Esta propiedad se refiere a mantener la coherencia de la información, asegurando que la información nunca pase a un estado incoherente.

Aislamiento (Isolation): Cada conjunto de operaciones se debe ejecutar en forma aislada respecto de los otros conjuntos de operaciones. Una vez que todo el conjunto de operaciones se haya ejecutado en forma completa y satisfactoria, recién en ese momento los cambios serán visibles para los demás usuarios.

Durabilidad (Durability): Esta propiedad establece que un cambio realizado deberá ser mantenido por el repositorio soportando cualquier situación anormal, por ejemplo fallas. A partir de la ejecución satisfactoria de un cambio, el nuevo estado de la información se debe mantener como base para cualquier otro cambio que se realice en el futuro.

Modelado de Datos

Tipos de Bases de Datos

Orientada a Objetos

Identificador de Objeto (OID)

En un sistema orientado a objetos, a cada objeto le es asignado un Identificador de Objeto (OID), cuyas características son:

- ➔ Creado por el sistema
- ➔ Unico (unique)
- ➔ Invariante, es decir, no puede ser alterado durante la existencia del objeto.
- ➔ Independiente del valor de sus atributos
- ➔ Invisible al usuario

El OID asegura que un objeto pueda ser identificado unívocamente.

Modelado de Datos

Tipos de Bases de Datos

Orientada a Objetos

Características (OID)

- ***Son eficientes:*** Un OID requiere de un mínimo de espacio más allá de que el objeto sea complejo. Generalmente son más pequeñas que los nombres de texto, claves foráneas u otras referencias semánticas.
- ***Son rápidos:*** Los OID son referencias a objetos. Esto significa que los objetos pueden ser ubicados rápidamente ya sea en memoria local o en disco.
- ***No pueden ser modificados por el usuario:*** Si son generados por el sistema o de sólo lectura, el sistema puede asegurar integridad referencial.
- ***Son independientes del contenido:*** No dependen de los datos contenidos por el objeto.

Modelado de Datos

Tipos de Bases de Datos

Orientada a Objetos

Ejemplos de uso en ambientes productivos:

SLAC (Stanford Linear Accelerator Center) es una instalación científica que estudia la estructura de la materia. Para llevar esto adelante suelen realizar experimentos en los cuales se aceleran partículas básicas para hacerlas chocar luego y estudiar las nuevas partículas que se generan en dicha colisión. Para dar una idea de la magnitud de estos experimentos, la base de datos tiene una tasa de persistencia de 1 Tera por día, llegando a un tamaño aproximado de 900 TB de información comprimida (Objectivity/DB).

P8A – Multi-Mission Maritime Aircraft es un proyecto de Boeing que utiliza bases de datos orientadas a objetos embebidas como repositorio de la información requerida por sus naves de vigilancia y detección de submarinos (DB4O).

Modelado de Datos

Tipos de Bases de Datos

Orientada a Objetos

Productos disponibles (algunos comerciales, otros open-source)

Versant	MongoDB (antes 10Gen)	Objetivity/DB
Prevayler	VOSS	Orion
ODE	ObjectStore	GBase
VBase	Exodus	Jasmine
Matisse	Ontos	O2
FastObjects	Jade	Caché
Perst	Ozone	Zope
EyeDB	Gemstone	ObjectDB

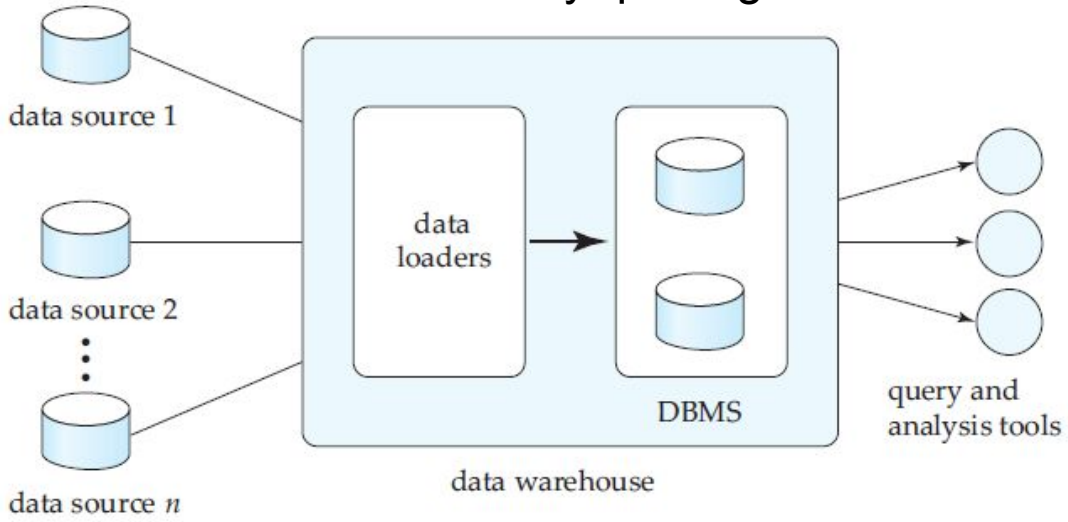
Modelado de Datos

Tipos de Bases de Datos

Multidimensionales

Las compañías de gran dimensión generan grandes volúmenes de datos. Los OLTP son grandes productores de datos que pueden ser transformados en información para la toma de decisiones. Un **datawarehouse** es un repositorio (o archivo) de información obtenida desde diferentes fuentes y que registra los datos bajo un esquema unificado.

Proveen de un única interfaz consolidada para acceder a los datos, facilitando el desarrollo de consultas para la toma de decisiones.



Modelado de Datos

Tipos de Bases de Datos

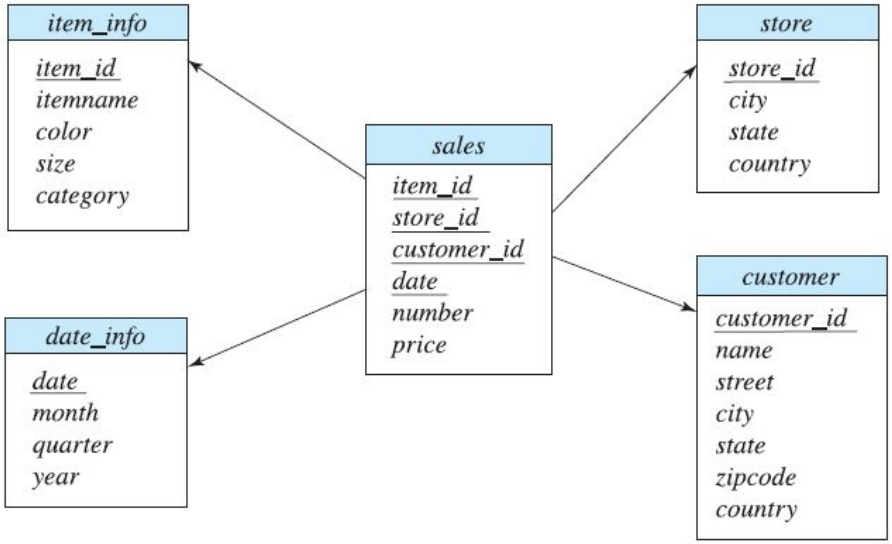
Multidimensionales

Los **datawarehouse** típicamente poseen esquemas que son diseñados para el análisis de datos empleando herramientas OLAP (On Line Analytical Processing). A este esquema se lo denomina **esquema estrella**, donde los datos son multidimensionales, con una dimensión de atributos y una medida de atributos. Las tablas que contienen datos multidimensionales se denominan tablas de hechos (fact tables) y habitualmente son de grandes dimensiones.

Modelado de Datos

Tipos de Bases de Datos

Multidimensionales



Una tabla que registra información de ventas (***sales***), con una tupla por cada ítem vendido. La dimensión de ventas (***sales)*** puede incluir: el ítem (el código de barras), la fecha de ventas, la ubicación (***store)***, el cliente (***customer)*** y otros. Otros atributos para la medición pueden ser la cantidad y el precio.

Para minimizar los requerimientos de almacenamiento, los atributos de dimensiones generalmente son identificadores que son claves foráneas a otras tablas denominadas tablas de dimensiones (dimension tables).

Modelado de Datos

Tipos de Bases de Datos

Multidimensionales

Diseños más complejos de data warehouses pueden requerir múltiples niveles de tablas de dimensiones, por ejemplo: **item_info** con un atributo **manufacturer_id** que sea clave foránea a otra tabla con detalles de fabricantes. Esos esquemas se denominan esquemas de copo de nieve (snowflake schemas).

Data Warehouses complejos también pueden poseer más de una tabla de hechos.

Modelado de Datos

Tipos de Bases de Datos

NoSQL

El término NoSQL apareció por primera vez a finales de los '90 como el nombre de una base de datos relacional open-source. Esta base de datos almacenaba sus tablas en archivos ASCII, donde cada tupla representa una línea con campos separados por tabulaciones. El nombre surgió del hecho que no usara SQL para las consultas, su manipulación era a través de scripts de consola. Las bases de datos NoSQL no utilizan SQL, pero muchas poseen un lenguaje de consulta similar al SQL. Por ejemplo Cassandra CQL es casi igual a SQL. Otra importante característica de las bases NoSQL es que generalmente son proyectos open-source.

Modelado de Datos

Tipos de Bases de Datos

NoSQL

La mayoría de las bases NoSQL se ejecutan en clusters. Esto se deriva de su modelo de datos al igual que su enfoque respecto de la consistencia. Las bases de datos relacionales utilizan el concepto ACID para asegurar la consistencia de las transacciones. Por el contrario, las bases NoSQL y por su organización en clusters, ofrecen un amplio rango de opciones para gestionar consistencia y distribución. Sus principales características son:

- ➔ No utilizan el modelo relacional
- ➔ Funcionan bien en clusters
- ➔ Son open-source
- ➔ Desarrolladas para la web del siglo 21
- ➔ No poseen esquema

Modelado de Datos

Tipos de Bases de Datos

NoSQL

Uno de los más importantes cambios que ofrece NoSQL es que no utiliza el modelo relacional. Cada solución de NoSQL posee un diferente modelo de datos. Los más utilizados son: Clave-Valor (key-value), Orientado a Documento, Orientado a Columnas y Grafo. Los tres primeros comparten el modelo de datos que puede denominarse Orientado a la Agregación. Entendiendo a agregación como una colección de objetos relacionados que se desea tratar como una unidad. Viendo a la unidad como una unidad de manipulación de datos y gestión de consistencia.

Modelado de Datos

Tipos de Bases de Datos

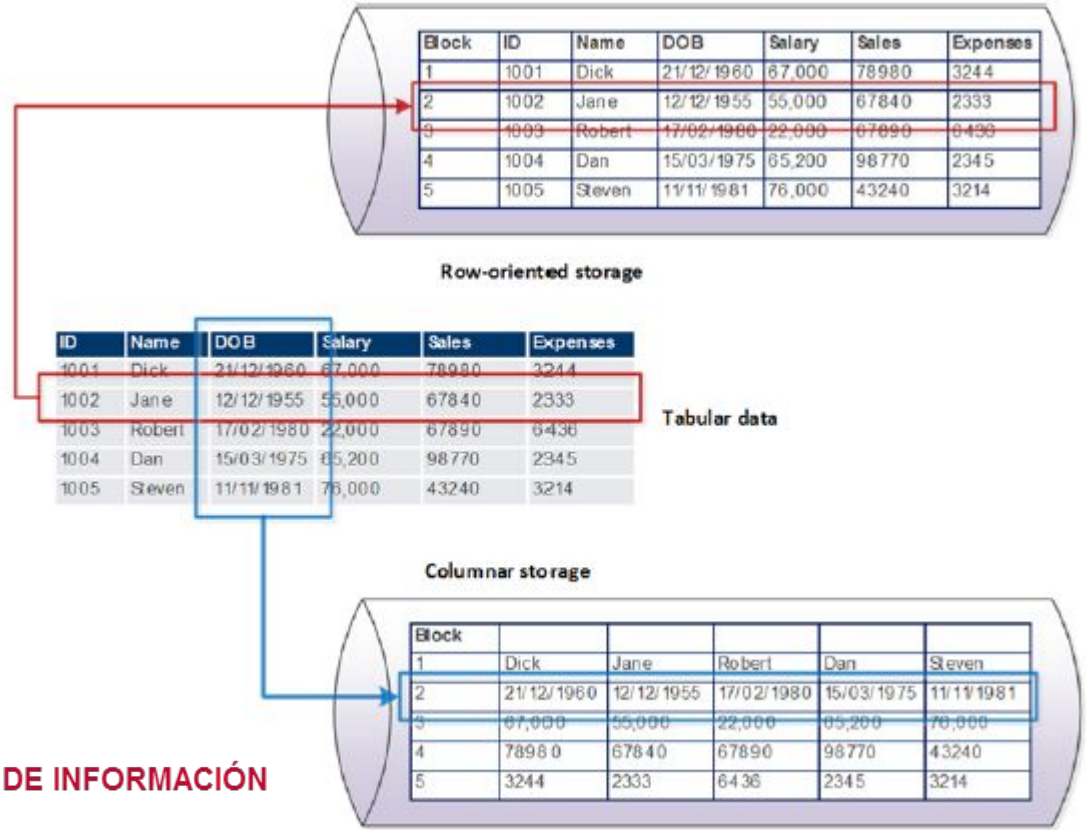
NoSQL

Las bases de datos **clave-valor** típicamente aceptan cualquier valor binario como clave y es capaz de almacenar cualquier dato binario como valor. En este tipo de almacenamiento la única forma de ubicar un objeto es a través de la clave. Implementaciones como Riak incluyen a Solr como motor de búsqueda de textos. En el caso de JSON y XML las búsquedas pueden estar restringidas a ciertos campos específicos del documento.

Modelado de Datos

Tipos de Bases de Datos

En **orientación a columnas**, los datos para columnas son almacenados de forma agrupada en disco. En estas bases de datos los valores para una columna específica son ubicados en los mismos bloques de disco, mientras que en el modelo orientado a filas todas las columnas para una fila son co-ubicados.



Modelado de Datos

Tipos de Bases de Datos

NoSQL

Las dos principales ventajas de esta arquitectura son:

- ➔ Optimización para consultas que buscan agregar valores de una columna.
- ➔ Se consiguen tasas de compresión extremadamente altas con un overhead computacional muy bajo.

La desventaja de esta arquitectura es que son muy poco apropiadas para bases de datos OLTP. Para operaciones de recuperación de filas, estas bases de datos, requieren el reensamble de cada fila por cada columna que se almacena para esa tabla.

Modelado de Datos

Tipos de Bases de Datos

NoSQL

Una base de datos **orientada a documentos** es una base no relacional que almacena datos como documentos estructurados, por lo general en formatos XML o JSON. Estas bases de datos no implican nada particular respecto al manejo de las transacciones u otras características propias de los RDBMS. Permiten que los datos sean descriptos sin la rigidez de un esquema de base de datos relacional ni la falta de esquema de los almacenamientos clave-valor.
Como están muy alineadas a las prácticas del desarrollo web, ha resultado que bases de datos de este tipo, como MongoDB, se hayan convertido en la elección favorita de los desarrolladores web.

Mapeo Objeto-Relacional

Modelado de Datos

Mapeo Objeto-Relacional

La técnica de crear un "puente" entre el modelo de objetos y el modelo relacional se denomina Mapeo Objeto-Relacional (ORM). Donde se busca:

- ➔ **Objetos:** Las aplicaciones deberían ser escritas en los términos del modelo de dominio, no ligadas al modelo relacional.
- ➔ **Conveniencia:** Las herramientas de mapeo deberían ser empleadas por quien está familiarizado con la tecnología relacional.
- ➔ **No intrusivo:** La solución de persistencia no debería ser un intruso en el modelo de dominio.
- ➔ **Datos legados** (legacy)**:** Soporte a esquemas legados es uno de los más relevantes casos de uso.

Modelado de Datos

Mapeo Objeto-Relacional

JDBC (Java Database Connectivity)

Desde la versión 1.1 de Java esta tecnología está disponible como una abstracción simple y portátil que permite interactuar con bases de datos relacionales a través de un SQL no propietario. Ejemplo:

```
public static void viewTable(Connection con, String dbName) {
      String query = "select COF_NAME, SUP_ID, PRICE, " + "SALES, TOTAL " + "from " + dbName + ".COFFEES";
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery(query);
            while (rs.next()) {
                  String coffeeName = rs.getString("COF_NAME");
                  int supplierID = rs.getInt("SUP_ID");
                  float price = rs.getFloat("PRICE");
                  int sales = rs.getInt("SALES");
                  int total = rs.getInt("TOTAL");
                  System.out.println(coffeeName + "\t" + supplierID + "\t" + price + "\t" + sales + "\t" + total);
            }
stmt.close();
}
```

Fuente: https://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html

Modelado de Datos

Mapeo Objeto-Relacional

Soluciones propietarias de ORM:

Originalmente las dos APIs más populares de persistencia fueron TopLink para el ámbito comercial e Hibernate en la comunidad open-source. Estos dos productos y otros, podían ser fácilmente integrados a la mayoría de los servidores de aplicaciones, proveyendo de forma exitosa todas la características de persistencia que las aplicaciones necesitaban. Como los objetos de persistencia eran objetos Java convencionales, este modelo de persistencia comenzó a ser conocido como persistencia POJO (Plain Old Java Object).

Estas soluciones de terceras partes cumplian de manera satisfactoria con los requisitos de los desarrolladores.

Modelado de Datos

Mapeo Objeto-Relacional

¿Por qué un estándar?

Un estándar va más allá de un producto, un solo producto no es una especificación. La intención de una especificación es que pueda ser implementada por diferentes fabricantes, con una oferta de diferentes productos que ofrezcan interfaces y semántica estándar que puedan ser empleados por aplicaciones sin acoplarla a un producto en particular. Además, las tecnologías de software representan una gran inversión para las empresas y el riesgo debe ser medido y controlado. Utilizando tecnologías estándar, se reduce sustancialmente el riesgo y permite a la empresa cambiar de fabricante (vendor) si la elección original no cumplió con los requerimientos. Más allá de la portabilidad, el valor de una tecnología estandarizada, también se manifiesta en el respaldo que representa el grupo de expertos que la desarrolla y mantiene.

Modelado de Datos

Mapeo Objeto-Relacional

JPA

Java Persistence API (JPA)

Es un marco de trabajo (framework) de tipo liviano, basado en POJOs para persistencia Java. Más allá del mapeo objeto-relacional, JPA brinda soluciones de persistencia escalables para aplicaciones empresariales. Sus principales características son:

- ➔ **Persistencia POJO:** Sólo intervienen objetos Java convencionales.
- ➔ **No intrusivo:** Las clases de objetos de persistencia no requieren ser modificados por la API.
- ➔ **Objetos de consulta:** Las consultas pueden ser expresadas con JPQL (Java Persistence Query Language) o a través de la API Criteria sin acoplar a un esquema en particular.

Modelado de Datos

Mapeo Objeto-Relacional

JPA

Java Persistence API (JPA)

- ➔ **Entidades móviles:** Los objetos que dejan la capa de persistencia se denominan "desprendidos" (detached) y permite que se realicen cambios fuera de línea para luego ser reincorporados a la capa de persistencia manteniendo consistencia y concurrencia.
- ➔ **Configuración simple:** Todo se configura a través de anotaciones y archivos XML.
- ➔ **Integración y pruebas:** Permite que las aplicaciones se ejecuten tanto en un servidor de aplicaciones como en un ambiente de desarrollo.

Mapeo Objeto-Relacional

JPA

Proveedores JPA

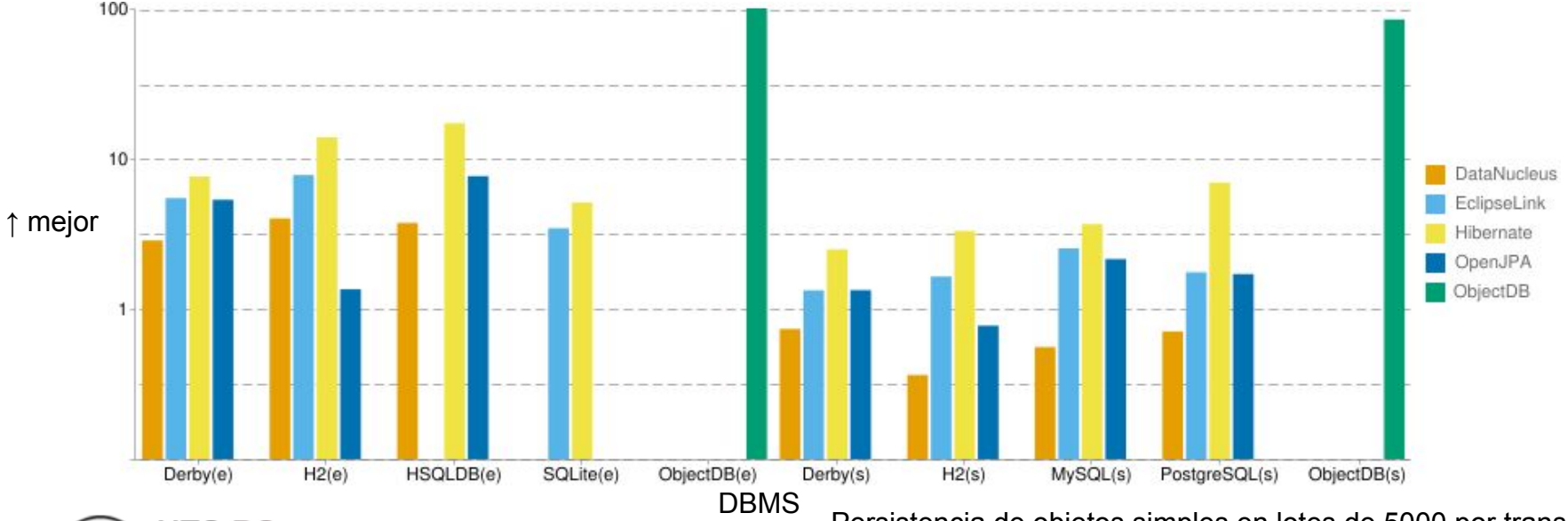
- Hibernate
- EclipseLink
- DataNucleus
- OpenJPA
- ObjectDB

Modelado de Datos

Mapeo Objeto-Relacional

JPA:Medición de desempeño

Proveedores JPA



Persistencia de objetos simples en lotes de 5000 por transacción

Fuente: https://dzone.com/articles/new-jpa-performance-benchmark

Referencias

Silberschatz, A., Database System Concepts.
McGraw-Hill. 2006.

Connolly, T., Database Systems.
Addison-Wesley. 2005.

Ullman, J., Database Systems.
Pearson. 2009.

Bazzocco, J., Persistencia Orientada a Objetos.
Edulp. 2011.

Sadalage, P., NoSQL Distilled.
Addison-Wesley. 2012.

Harrison, G., Next Generation Databases.
Apress. 2015.

Keith, M., Pro JPA 2.
Apress. 2009.