UNIVERSIDAD DE GRANADA

DECSAI

# Diseño y Desarrollo de Sistemas de Información

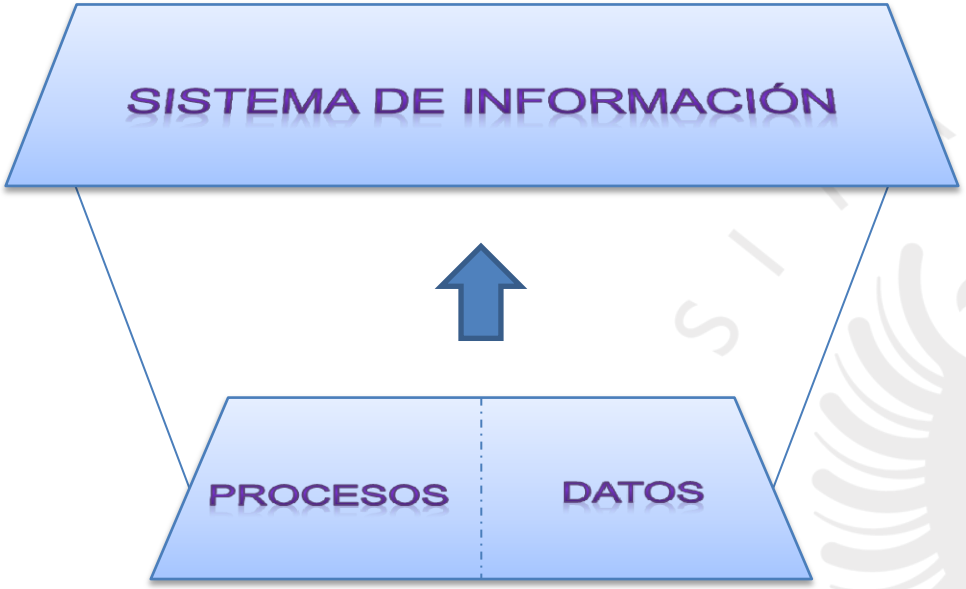
## Grado en Ingeniería Informática

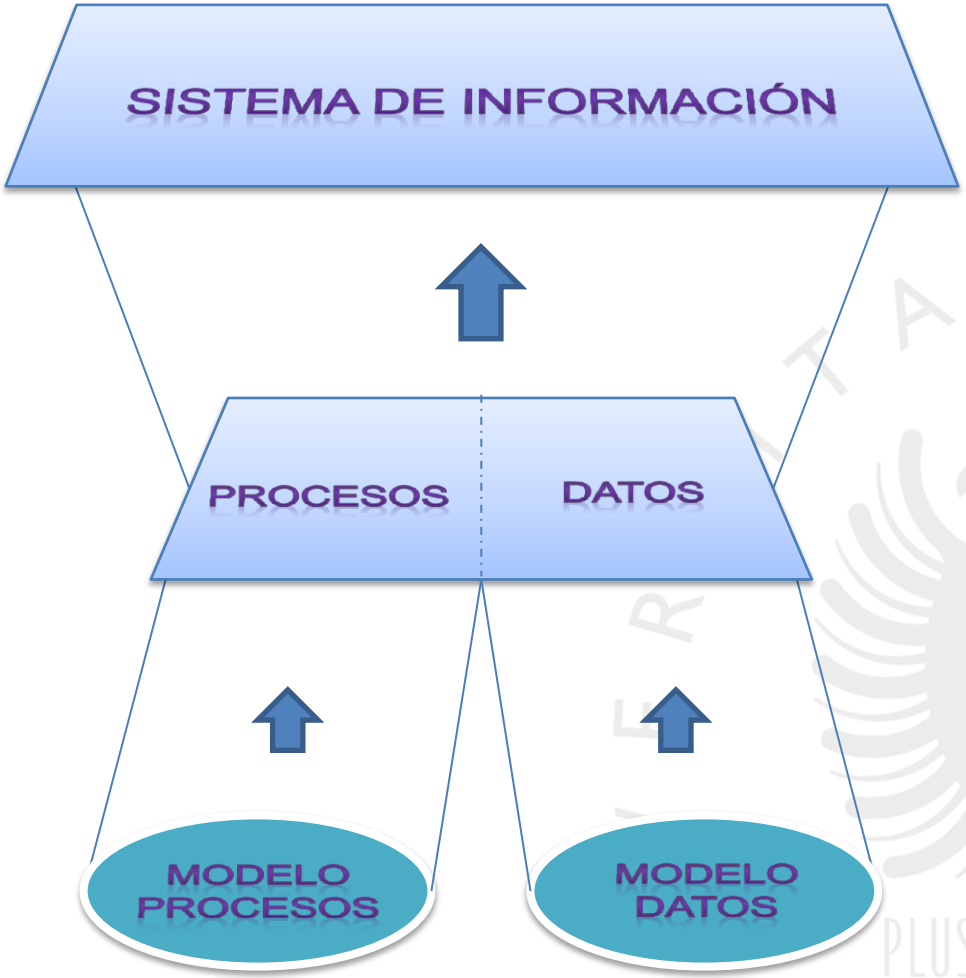
## Tema 3 – Modelos de Datos

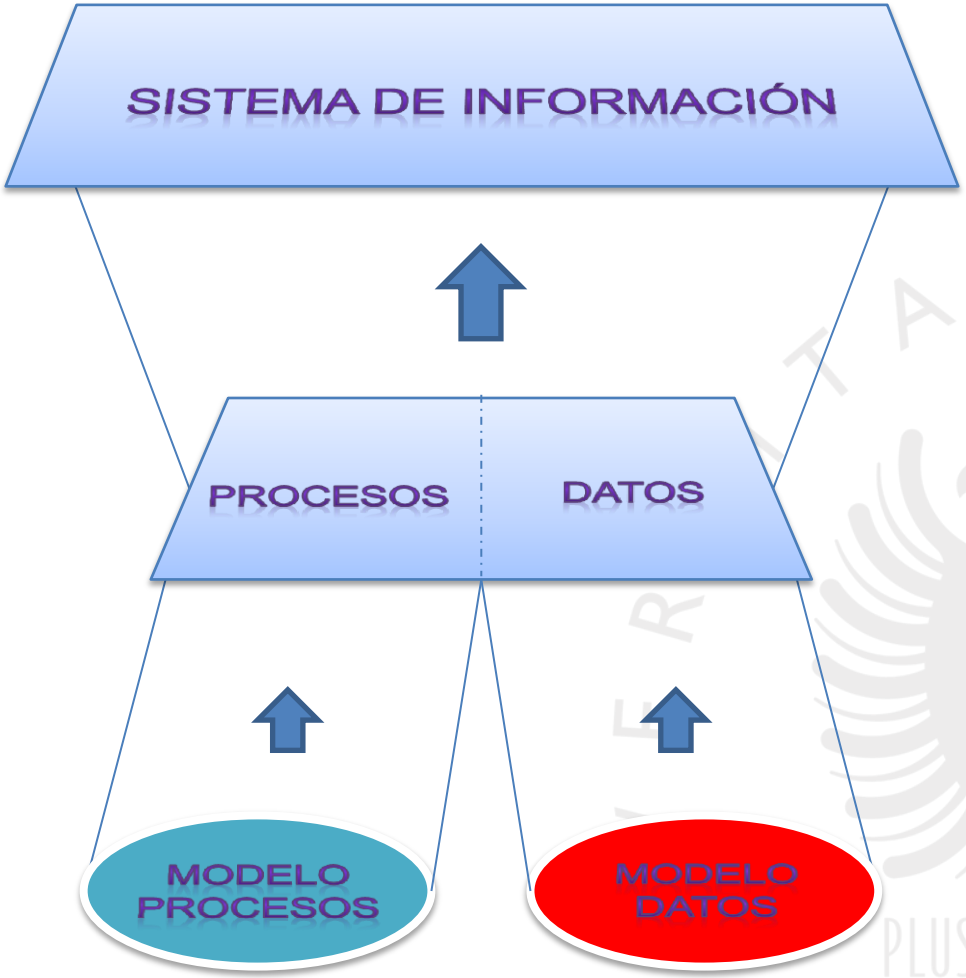
©I. J. Blanco, F. J. Cabrerizo, C. Cruz, M. J. Martín, D. Sánchez



Departamento de Ciencias de la Computación e Inteligencia Artificial
http://decsai.ugr.es


SISTEMA DE INFORMACIÓN
PROCESOS
DATOS


SISTEMA DE INFORMACIÓN
PROCESOS
DATOS
MODELO PROCESOS
MODELO DATOS


SISTEMA DE INFORMACIÓN
PROCESOS
DATOS
MODELO PROCESOS
MODELO DATOS

Representación, relativamente sencilla, orientada a la descripción de los datos del mundo real y que, usualmente describe:

- La estructura de los datos
- Las condiciones que deben cumplir los datos
- Cómo se manejan los datos

- Tienen asociados una serie de conceptos, que describen un conjunto de datos y operaciones para manipular los datos.

- Dichos conceptos tienen asociados una construcción lingüística y una gráfica.

- Modelo Conceptual que representa la realidad en un alto nivel de abstracción. Genera el Esquema Conceptual

- Modelo Lógico, o Modelo de Base de Datos, que describe las relaciones lógicas entre los datos y la base de datos. Genera el Esquema Lógico.

- Los elementos de un modelo representan Entidades genéricas.

- Los valores concretos se denominan instancias u ocurrencias de una entidad.

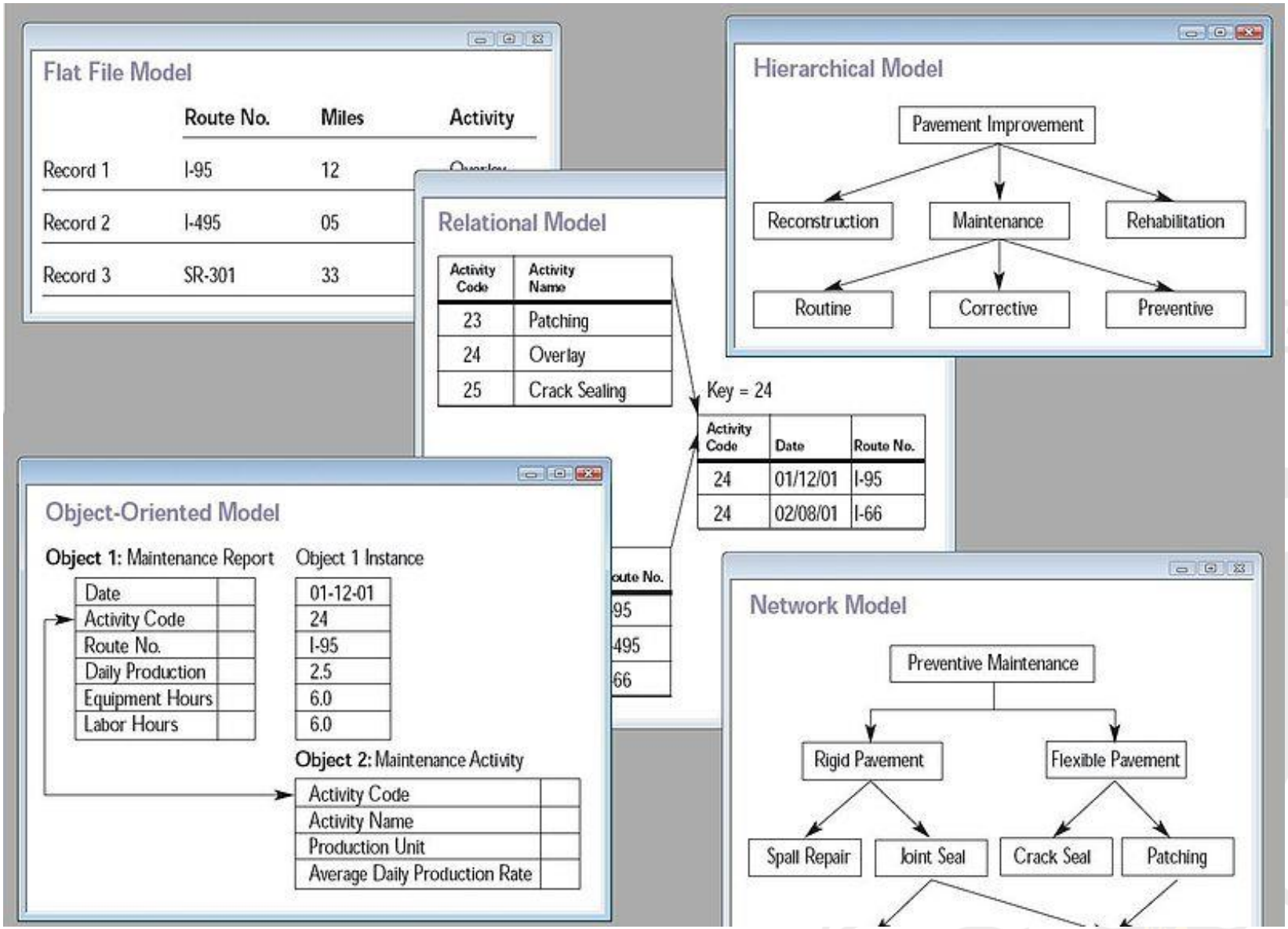
- Cada SGBD se asocia a un modelo de datos específico, aunque puede haber excepciones.

Respecto a los datos:
- Datos o Entidades.
- Propiedades de los Datos.
- Relaciones de los Datos.
- Restricciones de los Datos.

Se representa mediante el Lenguaje de Definición de Esquemas (LDE) del SGBD.

Respecto a las operaciones:
- Operaciones de los Datos
- Operaciones sobre Relaciones de los Datos.
- Relaciones entre Operaciones.

Se representa mediante el Lenguaje de Manipulación de Datos (LMD) del SGBD.


Flat File Model
Route No.
Miles
Activity
Record 1
I-95
12
Record 2
I-495
05
Record 3
SR-301
33
Hierarchical Model
Pavement Improvement
Reconstruction
Maintenance
Rehabilitation
Routine
Corrective
Preventive
Relational Model
Activity Code
Activity Name
23
Patching
24
Overlay
25
Crack Sealing
Key = 24
Activity Code
Date
Route No.
24
01/12/01
I-95
24
02/08/01
I-66
Object-Oriented Model
Object 1: Maintenance Report
Object 1 Instance
Date
Activity Code
Route No.
Daily Production
Equipment Hours
Labor Hours
01-12-01
24
I-95
2.5
6.0
6.0
Object 2: Maintenance Activity
Activity Code
Activity Name
Production Unit
Average Daily Production Rate
Network Model
Preventive Maintenance
Rigid Pavement
Flexible Pavement
Spall Repair
Joint Seal
Crack Seal
Patching

# Características:

Modelo mas simple
Matriz bidimensional de elementos, sin relaciones con otras matrices
Miembros en una columna tienen valores del mismo tipo
Los miembros de la misma fila están relacionados entre ellos

Flat File Model

| | Route No. | Miles | Activity |
|---|---|---|---|
| Record 1 | I-95 | 12 | Overlay |
| Record 2 | I-495 | 05 | Patching |
| Record 3 | SR-301 | 33 | Crack seal |

# Ejemplos de su uso

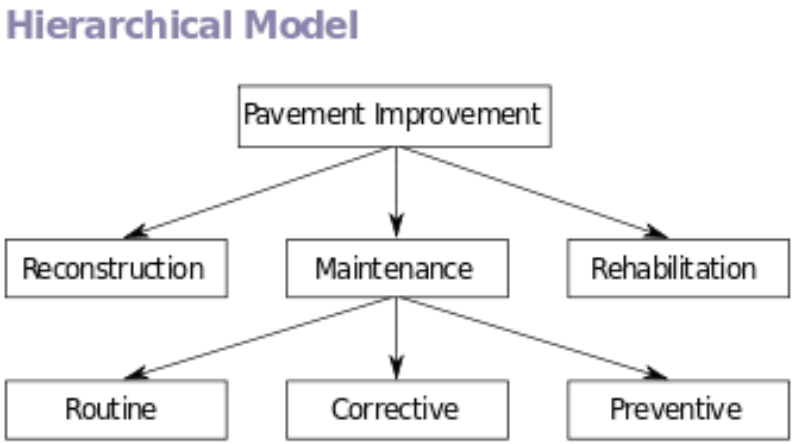
| | |
|---|---|
| **usuario1:** | Nombre de la cuenta (Login) |
| **FXWUuZ.vwXttg:** | Clave de acceso encriptada (password) |
| **500:** | UID de esta cuenta |
| **501:** | GID del grupo principal al que pertenece la cuenta |
| **usuario pepito:** | Nombre del usuario |
| **/home/usuario1:** | Directorio de trabajo de usuario1 |
| **/bin/bash:** | Interprete de comando (shell) de usuario pepito |

etc/passwd

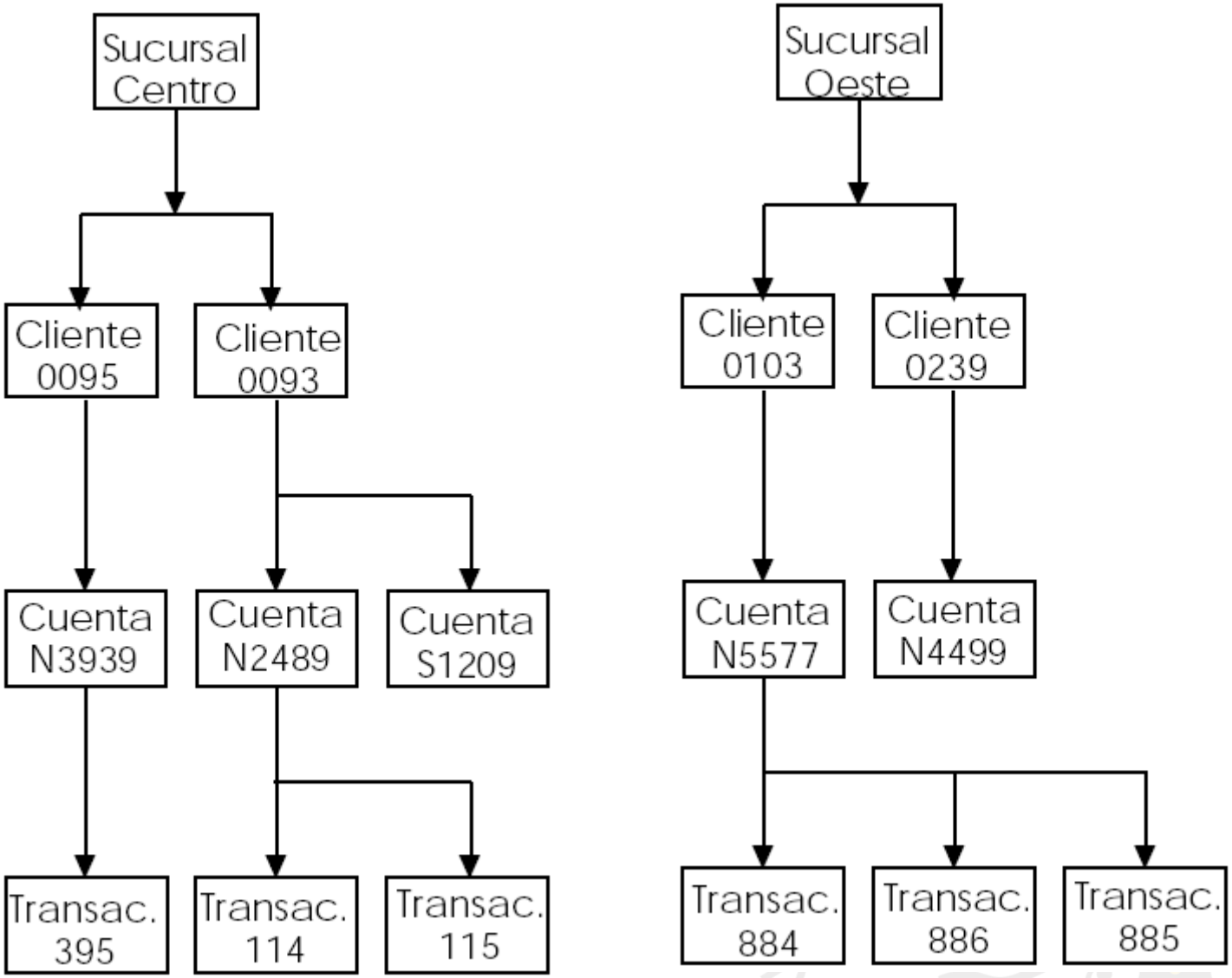
Tablas y plantillas Excel

## Características:

Modelo creado por IBM (Information Management System, IMS -> Programa Apolo) en los años 60
Datos organizados en una estructura arbórea
Relaciones 1:N entre los datos
Las relaciones entre datos se establecen siempre a nivel físico
Muy eficiente en relaciones de datos con estructura jerárquica

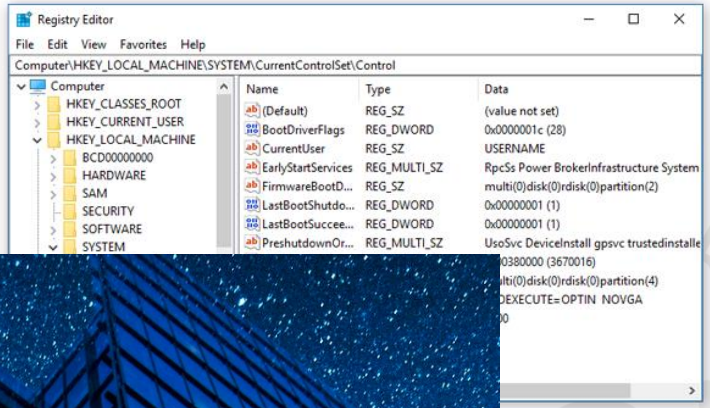


# Modelo de Datos Jerárquico



# Ejemplos de su uso

Sabre



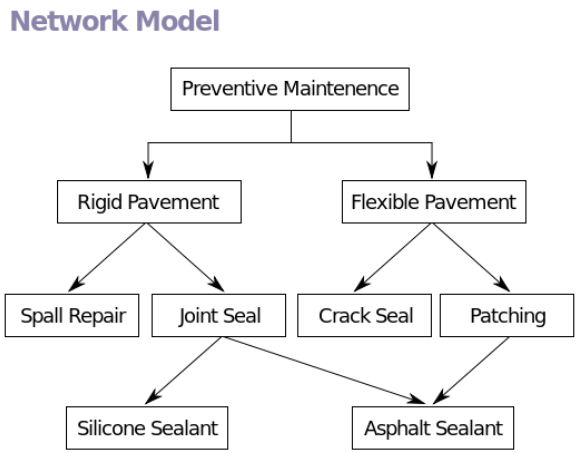


```
<?xml version="1.0"?>
<quiz>
 <qanda seq="1">
  <question>
   Who was the forty-second
   president of the U.S.A.?
  </question>
  <answer>
   William Jefferson Clinton
  </answer>
 </qanda>
 <!-- Note: We need to add
  more questions later.-->
</quiz>
```

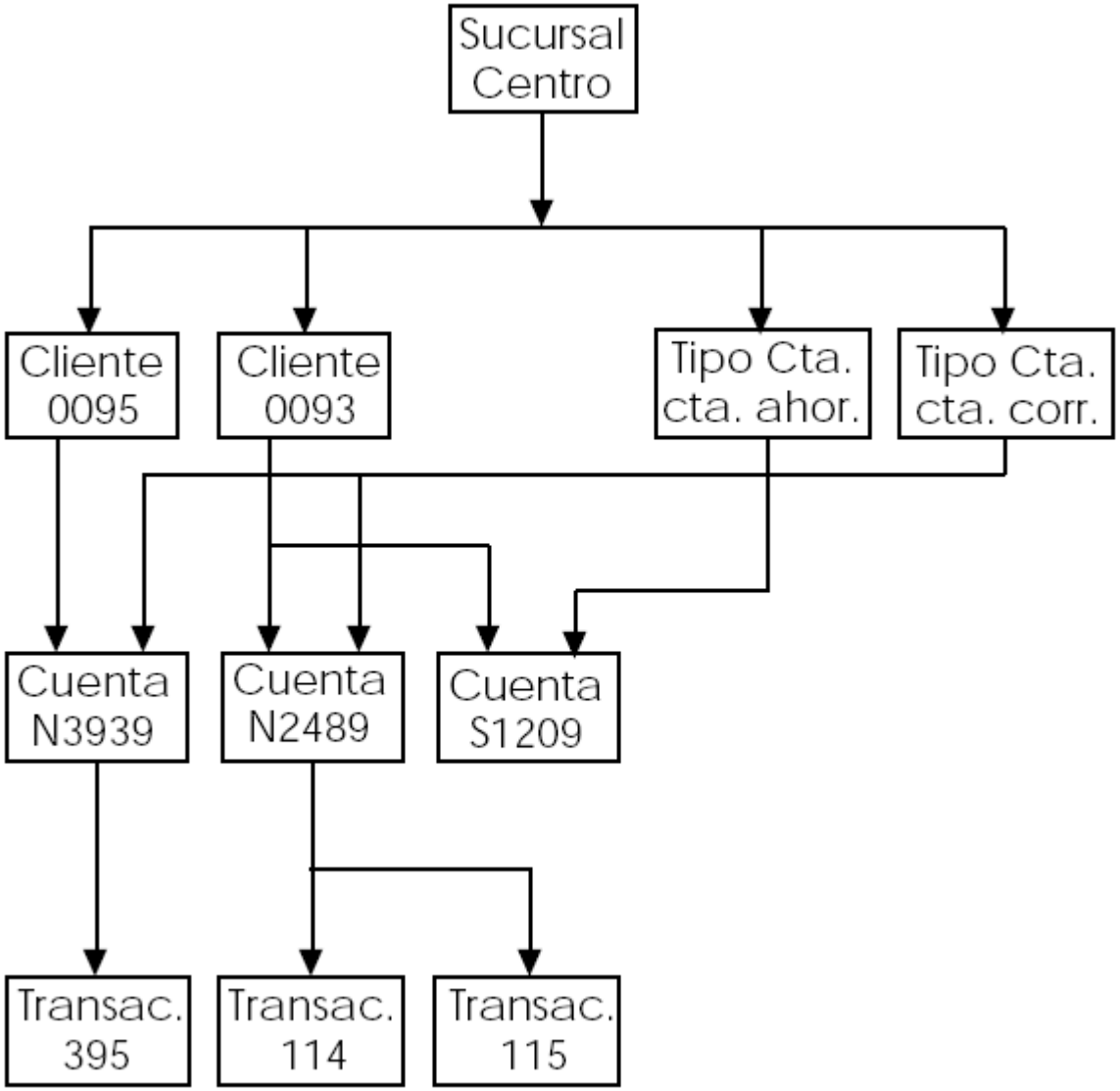
XML

## Características:

Modelo creado a partir de especificaciones del grupo CODASYL (Conference on Data Systems Languages)
Generalización del modelo jerárquico,
Permite relaciones N:N en una estructura tipo árbol que permite múltiples padres




De este modo se reducen las redundancias.
Desaparece la herencia de los campos.
La integridad de datos, asociada a los arcos padre-hijo, se mantiene.

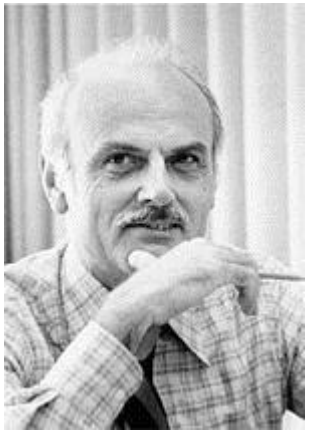

Sucursal Centro
Cliente 0095
Cliente 0093
Tipo Cta. cta. ahor.
Tipo Cta. cta. corr.
Cuenta N3939
Cuenta N2489
Cuenta S1209
Transac. 395
Transac. 114
Transac. 115

# Ejemplos de su uso

**IDMS (Integrated Database Management System)**



**British Telecom: 10 mil millones de transacciones por año**



**Edgar Frank "Ted" Codd**

*Information Retrieval* P. BAXENDALE, Editor

## A Relational Model of Data for Large Shared Data Banks

E. F. Codd
*IBM Research Laboratory, San Jose, California*

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted

**Relational Model**

| Activity Code | Activity Name |
|---|---|
| 23 | Patching |
| 24 | Overlay |
| 25 | Crack Sealing |

Key = 24

| Activity Code | Date | Route No. |
|---|---|---|
| 24 | 01/12/01 | I-95 |
| 24 | 02/08/01 | I-66 |

| Date | Activity Code | Route No. |
|---|---|---|
| 01/12/01 | 24 | I-95 |
| 01/15/01 | 23 | I-495 |
| 02/08/01 | 24 | I-66 |

Basado en el concepto de **Relación,** que formalmente puede verse como un par de conjuntos

$$(R, r)$$

donde,

$R$ se denomina **esquema** y

$r$ se denomina **instancia**

Esquema es un conjunto:

$$R = \{A_1 : D_1, A_2 : D_2, \ldots, A_n : D_n\}$$

donde,

$A_i$ son atributos, es decir, una representación de un elemento de información del "mundo" que es susceptible de tomar valores; y

$D_i$ son los dominios de los atributos, es decir, los posibles valores que toman los atributos

Instancia es la aplicación de un esquema a un conjunto finito de datos

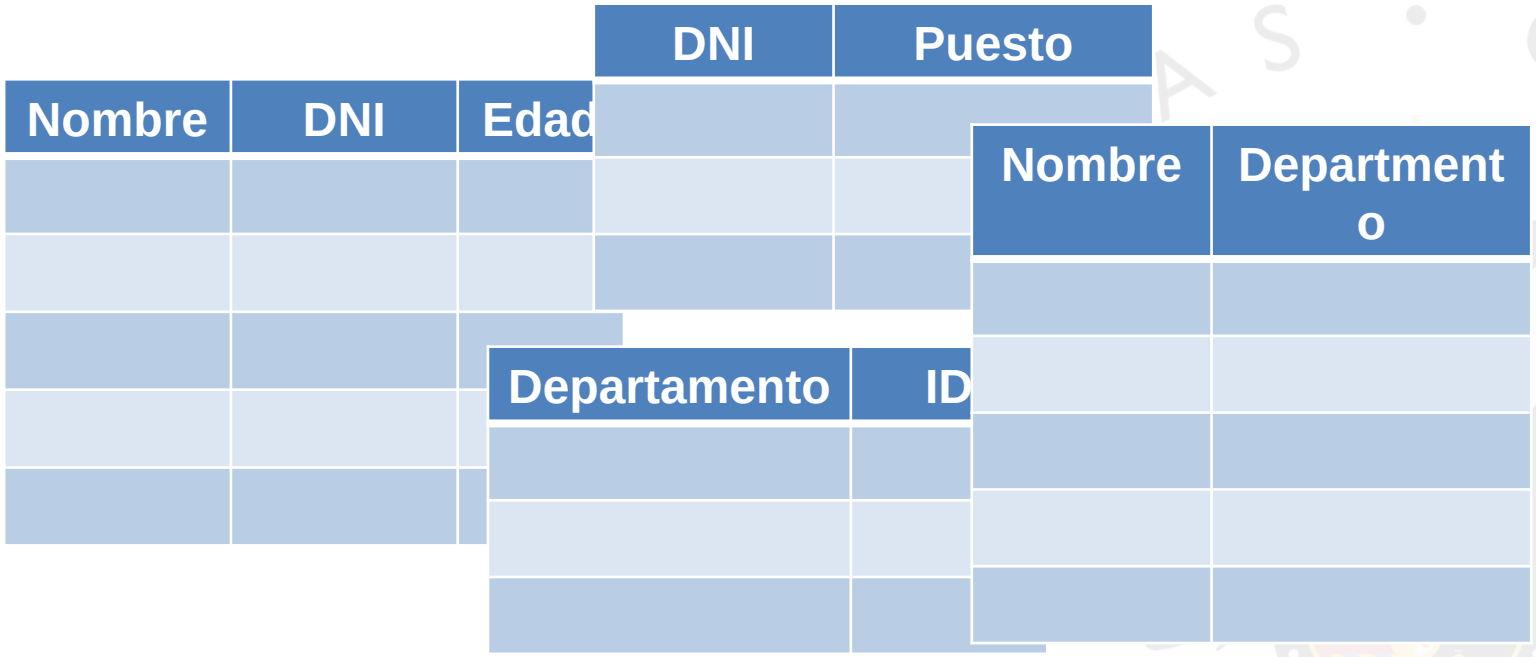
$$r \in D_1 \times D_2 \times \cdots \times D_n$$

(contenido de una tabla o parte de ella en un momento dado)

Visualmente, es una estructura bidimensional formada por columnas (**atributos**) y filas (**tuplas**)

| A1 | A2 | A3 | A4 | A5 |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Una **base de datos relacional** es un conjunto finito de relaciones



Una **instancia de la BBDD** es una colección de instancias de las relaciones

| Nombre | DNI | Edad |
|---|---|---|
| Luis | 21323431 | 25 |
| José | 34235571 | 12 |
| Ana | 55555111 | |
| Lucía | 12432139 | |
| María | 19392931 | 85 |

| DNI | Puesto |
|---|---|
| 55555111 | Ayudante |
| 12432139 | Prof. |

| Departamento | ID |
|---|---|
| DECSAI | 02 |
| LSI | 03 |
| ATC | 06 |

| Nombre | Departamento |
|---|---|
| Luis | DECSAI |
| María | LSI |
| Ana | Álgebra |
| Lucía | ATC |
| Ignacio | DECSAI |

Sin embargo, no todas las instancias de las relaciones de datos son válidas

| Profesor | DNI |
|---|---|
| G.N.G | 21323431 |
| M.P.C | 34235571 |
| K.B.A | 55555111 |
| L.E.Z | 12432139 |
| B.P.C | 19392931 |

| DNI | Departamento |
|---|---|
| 21323431 | DECSAI |
| 34235571 | DECSAI |
| 11111111 | LSI |

**No es el DNI de ningún profesor!!**

Puede no mantenerse la **corrección semántica**

Para mantener la semántica se deben respetar las **restricciones de integridad**, es decir, unas reglas que mantienen correcta la información que almacena

El **esquema de una base de datos** es una colección de **esquemas de relación** junto con una serie de **restricciones de integridad**

Una instancia de la base de datos es **válida** si respeta la reglas de integridad

Para mantener la semántica se deben respetar las restricciones de integridad

Restricciones específicas, son aquellas que provienen de la semántica del atributo y son propias de cada base de datos concreta. Ejemplos:

EDAD >= 0
NUM_ALUMNOS <= NUM_MAX_ALUMNOS
SUELDO >= IMPUESTOS+SEG_SOCIAL
…

Para mantener la semántica se deben respetar las restricciones de integridad

Restricciones genéricas, son aquellas que se aplican a los atributos en función del papel que desempeñan en la estructura de la BBDD
Son meta-reglas, normas genéricas que generan reglas
Giran en torno a los conceptos de clave primaria y externa

Transacciones ACID (Atomiciad, Consistencia, Isolation (aislamiento) y Durabilidad), que garantizan la consistencia y estabilidad de las operaciones pero requieren una gestión de bloqueo sofisticada:

**Atomicidad:** es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto, no puede quedar a medias ante un fallo del sistema. Por ejemplo, en el caso de una transacción bancaria o se ejecuta tanto el depósito como la deducción o ninguna acción es realizada.

**Consistencia:** *Integridad*. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido (de acuerdo con las restricciones y reglas establecidas en la BD).

**Aislamiento:** La ejecución concurrente de dos transacciones da como resultado un estado del sistema que sería el mismo que si se hubieran ejecutado secuencialmente.

**Durabilidad:** *Persistencia*. Es la propiedad que asegura que una vez establecida una transacción, ésta persistirá y no se perderán los cambios aunque falle el sistema.

Un conjunto *CC* de atributos en una relación se dicen que son **claves candidatas** si se verifica lo siguiente:

**Unicidad**. Si dos instancias de la relación coinciden en los atributos de la clave candidata, entonces coinciden en el resto de atributos de la relación

**Minimalidad**. La propiedad anterior no se verifica para cualquier subconjunto propio de *CC*

Se llama **superclave** si solo verifica la unicidad

Se llama **clave primaria** a una clave candidata elegida por el diseñador (sólo una por relación)

Dada una clave primaria ***CP*** de una relación ***R***, y un conjunto de atributos ***CE*** de otra relación ***S***, con ***CE*** contenido en ***CP***. Se dice ***CE*** es **clave externa** respecto a ***CP*** si el dominio activo de ***CE*** está contenido en el dominio activo de ***CP***

| Profesor | DNI |
|---|---|
| G.N.G | 21323431 |
| M.P.C | 34235571 |
| K.B.A | 55555111 |
| L.E.Z | 12432139 |
| B.P.C | 19392931 |

| DNI | Departamento |
|---|---|
| 21323431 | DECSAI |
| 34235571 | DECSAI |
| 55555111 | LSI |

**Es necesario que los DNI sean de profesores!!**

Propiedades de las relaciones:

No hay orden en las tuplas
No hay orden en los atributos
No hay tuplas duplicadas (es un conjunto!)
El esquema de toda relación tiene una clave primaria
Los valores de los atributos son atómicos (no tienen estructura)

Reglas de integridad:

**Integridad de entidad**. Los atributos de una clave primaria no pueden tomar valores nulos

**Integridad referencial**. El valor de una clave externa debe ser igual a un valor del dominio activo de la clave primaria, o nulo

¿Como mantener la integridad del sistema?

## Integridad de entidad

Comprobar que los atributos que forman parte de una clave primaria son no nulos y que el valor conjunto de ellos no está repetido en procesos de *Inserción* y *Actualización*

## Integridad referencial

*Inserción.* Comprobar que el valor de la clave externa es nulo o concuerda con un valor de la clave primaria
*Actualización*. Si se actualiza la clave externa, comprobar las condiciones de clave externa. Si se actualiza la clave primaria, actualizar en cadena la clave externa
*Borrado.* Si se borra la clave primaria, borrado en cadena o valor nulo en cadena de la clave externa

**SQL** es el lenguaje de consultas predominante

Basado en:

- Algebra Relacional (teoría de conjuntos)
  Obtiene resultado aplicando varios operadores (8)

- Cálculo Relacional (lógica de primer orden)
  Orientado a tuplas
  Orientado a dominios

SQL presenta dos sublenguajes:

**DDL (*Data Description Language*)** que permite definir y manejar esquemas de estructuras relacionales (relaciones, vistas, ...)

**DML (*Data Management Language*)** que permite manipular instancias de estructuras relacionales (tuplas)


ORACLE
DATABASE
Microsoft
SQL Server
IBM
DB2
SYBASE
An SAP Company
Microsoft
Access 2016
FileMaker
An Apple Subsidiary
dBase™
NexusDB Database
Informix
SOFTWARE


## Object-Oriented Model

**Object 1:** Maintenance Report

| | |
|---|---|
| Date | |
| Activity Code | |
| Route No. | |
| Daily Production | |
| Equipment Hours | |
| Labor Hours | |

Object 1 Instance

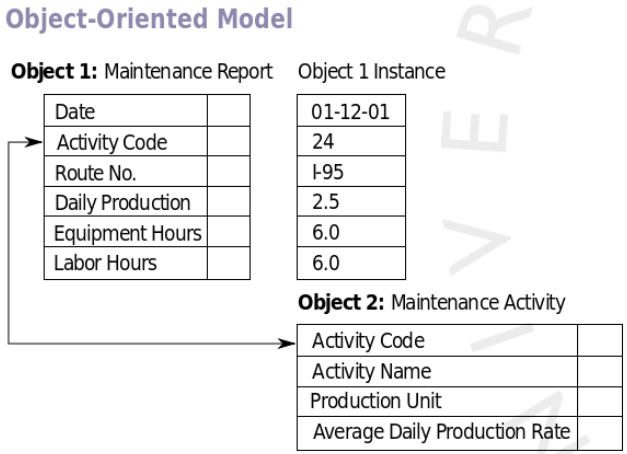
| |
|---|
| 01-12-01 |
| 24 |
| I-95 |
| 2.5 |
| 6.0 |
| 6.0 |

**Object 2:** Maintenance Activity

| | |
|---|---|
| Activity Code | |
| Activity Name | |
| Production Unit | |
| Average Daily Production Rate | |

# Motivación

Necesidad de los Lenguajes OO
Limitaciones de Bases de Datos Relacionales



# Motivación

## Necesidad de los Lenguajes OO

- Los modelos de datos y las estructuras de datos de los LPOO están desacoplados
- Persistencia de Objetos (más allá de los programas)
- Almacenamiento mas eficiente y gestión de datos en memoria secundaria
- Independencia de los datos respecto de los programas
- Lenguaje de consulta eficiente y de alto nivel (independiente de la estructura física)
- Gestión de transacciones que permita: acceso concurrente, integridad, seguridad y recuperación ante fallos
- Control de integridad

# Motivación

## Limitaciones de Bases de Datos Relacionales

- Presentan estructuras simples (ej: imposición de 1FN).
- Poca riqueza semántica.
- No soportan tipos definidos por el usuario (sólo dominios).
- No soportan recursividad.
- Falta de procedimientos/disparadores.
- No admite herencia.

Por todo lo anterior, las bases de datos relacionales no se consideran apropiadas para aplicaciones que manejen estructuras de datos complejas

# Motivación

## Limitaciones de Bases de Datos Relacionales

¿Qué ocurre si convertimos objetos y relaciones al modelo relacional?.

La traducción de objetos a tablas implica:

- Mayor tiempo de desarrollo. El tiempo empleado en generar el código para la traducción de objetos a tablas y viceversa.
- Errores debidos precisamente a esa traducción.
- Inconsistencias debidas a que el ensamblaje / desensamblaje puede realizarse de forma diferente en las distintas aplicaciones.
- Mayor tiempo de ejecución empleado para el ensamblaje / desensamblaje.

Manifiesto de los sistemas de base de datos orientados (1989)

- Obligatorias: imprescindible satisfacerlas para ser calificadas como OO.
- Opcionales: pueden añadirse para mejorar el sistema.
- Abiertas: posibilidades adicionales aceptables, a aplicar a juicio del diseñador

# Manifiesto de los sistemas de base de datos orientados
## Características obligatorias

Al ser un SGBD

- Persistencia.
- Gestión del almacenamiento secundario.
- Concurrencia.
- Recuperación ante fallos.
- Lenguajes ad-hoc para manipulación.

Al ser OO

- Objetos complejos.
- Identidad del objeto.
- Encapsulamiento.
- Tipos o clases.
- Herencia.
- Polimorfismo, sobrecarga y vinculación dinámica.
- Extensibilidad.
- Completitud de cálculos (lenguaje de propósito general).

# Manifiesto de los sistemas de base de datos orientados
# Características obligatorias

Características opcionales

- Herencia múltiple.
- Verificación e inferencia del tipo.
- Distribución.
- Transacciones de diseño.

Opciones abiertas

- Paradigma de programación.
- Sistema de representación (tipos
- atómicos y constructores).
- Sistema de tipos.
- Uniformidad (¿todo objetos?).

# Productos y estándares. SGBDOO puros

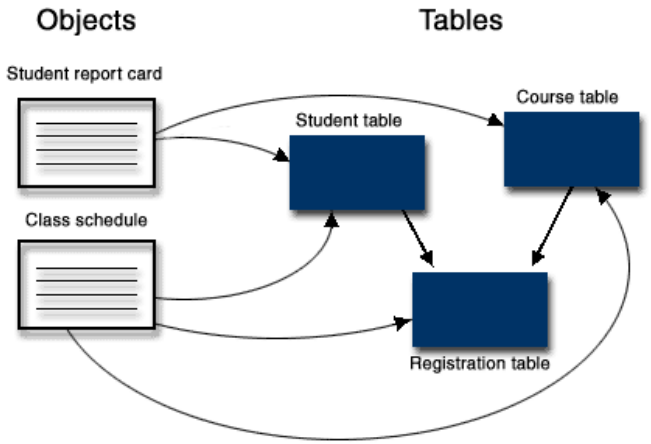
Estándares
ODMG-93, Cattell(1994), Cattell(1995), ODMG V.2.0 Cattell(1997), ODMG V.3.0 Cattell(2000).

Productos

- ObjectStore de Object Design. Persistencia de objetos en C++, Java.
- O2 de O2, Leeluse et al. (1988). Lenguajes: C++, lenguajes de consulta (O2SQL) y programación (O2C) propios. Java.
- Gemstone de Servi Logic, Meier y Stone (1987) Persistencia de objetos en Samalltalk. Soporta también C++ y Java.
- POET de Poet Corporation. Persistencia de objetos C++, Java.
- db4o, Matisse...

Sistema de gestión de bases de datos, similar a una base de datos relacional, pero con un modelo de base de datos orientada a objetos.

Soporta objetos, clases y herencia directamente en los esquemas de bases de datos y en el lenguaje de consulta.



# Reglas generales para el modelado objeto/relacional

- Cada clase persistente tiene una tabla de base de datos correspondiente.
- Campos de objetos con tipos de datos primitivos (enteros, caracteres, cadenas, etc.) se asignan a columnas en la tabla de base de datos asociada.
- Cada fila de una tabla de base de datos corresponde a una instancia de su clase persistente asociada.
- Cada relación de objeto de muchos a muchos requiere una tabla de join al igual que las entidades de base de datos con muchos-a-muchos requieren tabla de joins.
- La herencia es modelada a través de una relación uno-a uno entre las dos tablas que corresponden a la clase y subclase.

# Manifiesto de los SGBD de 3a Generación

Además de los servicios tradicionales de gestión de datos, los SGBD-3G proporcionarán gestión de objetos y reglas más ricas:

- Un SGBD-3G debe tener un sistema de tipos rico.
- La herencia es una buena idea.
- Las funciones (procedimientos y métodos) son una buena idea.
- Los IDOs para los registros deberían asignarse por el SGBD sólo si no se dispone de una clave primaria.
- Las reglas (disparadores, restricciones) se convertirán en una característica primordial de los sistemas futuros.

## Estándares

SQL: 1999, Melton (1999). SQL: 2003, Melton (2003).

## Productos

- POSTGRES (Miró/Illustra), Stonebrakeret al. (1992) Combina capacidades de BD OO y activas con BD relacionales.
- ORACLE V8, de Oracle (1997) Extiende el modelo relacional del SQL92 con capacidades de objetos y actividad.
- Universal Server de Informix, ...

SGBD "Revolucionarios"
SGBD OO "Puros"
SGB DE OBJETOS

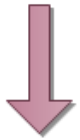
**OBJECTSTORE, O2, ONTOS, VERSANT, POET, GEMSTONE, ...**

**ODMG 3.0**

Implica una ruptura con los modelos anteriores, adaptación directa de principio de programación OO

SGBDR "Extendidos"
SGBD "Evolutivos"
TERCERA GENERACIÓN
OBJETO-RELACIONAL

**ORACLE, IBM, MICROSOFT, INFORMIX, SYBASE, CA, ...**

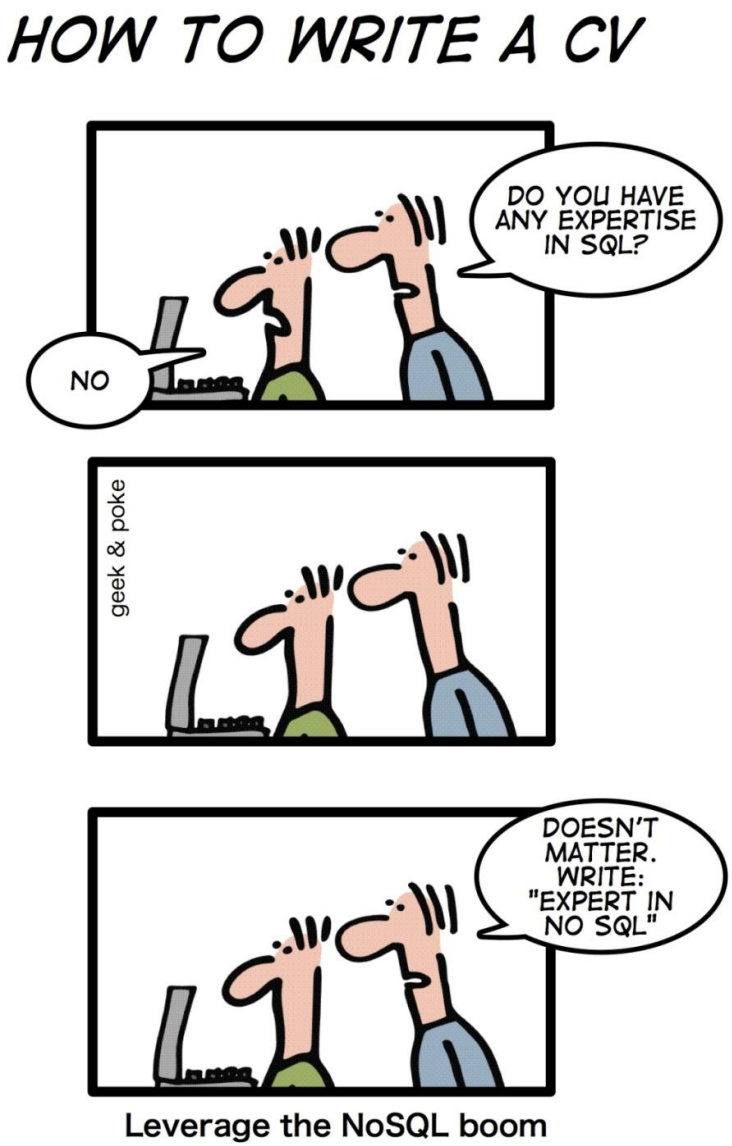


**SQL:2003**

Continuidad de modelos anteriores, aprovechando inversiones previas y subsanando carencias

- La BD no escala con el tráfico a un coste aceptable.
- El tamaño del esquema de datos crece desproporcionadamente.
- El sistema de información genera muchos datos temporales que no corresponden al almacén de datos principal (carritos de compra, personalización de portales).
- Tener que desnormalizar la BD por razones de rendimiento o por conveniencia para utilizar los datos en una aplicación.
- La BD contiene grandes cantidades de texto o imágenes en columnas como BLOBs.(Binary Large Objects)
- Se ejecutan consultas sobre los datos que implican relaciones jerárquicas complejas; recomendaciones o consultas de inteligencia de negocio.
- Se usan transacciones locales que no necesitan ser muy durables.

.


HOW TO WRITE A CV
DO YOU HAVE ANY EXPERTISE IN SQL?
NO
geek & poke
DOESN'T MATTER. WRITE: "EXPERT IN NO SQL"
Leverage the NoSQL boom

**NoSQL -** "not only SQL" – categoría general de sistemas de gestión de bases de datos que difiere de los RDBMS en diferentes modos:

- No usan SQL como el principal lenguaje de consultas.
- Los datos almacenados no requieren estructuras fijas como tablas
- Normalmente no soportan operaciones JOIN,
- Ni garantizan completamente ACID (atomicidad, consistencia, aislamiento y durabilidad),
- Habitualmente escalan bien horizontalmente.

**BA**sic availability: Siempre se obtiene una respuesta del sistema a una petición de datos aunque esta sea un fallo o que sean inconsistentes o estén en fase de cambio.

**S**oft-state: el estado del sistema cambia constantemente a lo largo del tiempo, incluso aunque no hayan entradas de datos en ese periodo, debido a la consistencia eventual.

**E**ventual consistency: eventualmente, el sistema se volverá consistente a partir de que deje de recibir datos. Los datos se propagarán pero el sistema seguirá recibiendo datos sin evaluar la consistencia de los datos para cada transacción antes de avanzar a la siguiente.

Se han diseñado para potenciar aspectos como:

**Flexibilidad**: las bases de datos NoSQL generalmente ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo. El modelo de datos flexible hace que las bases de datos NoSQL sean ideales para datos semiestructurados y no estructurados.

**Escalabilidad**: las bases de datos NoSQL generalmente están diseñadas para escalar usando clústeres distribuidos de hardware en lugar de escalar añadiendo servidores caros y sólidos. Algunos proveedores de la nube manejan estas operaciones fuera del alcance, como un servicio completamente administrado.

**Alto rendimiento:** la base de datos NoSQL está optimizada para modelos de datos específicos (como documentos, clave-valor y gráficos) y patrones de acceso que permiten un mayor rendimiento que el intento de lograr una funcionalidad similar con bases de datos relacionales.

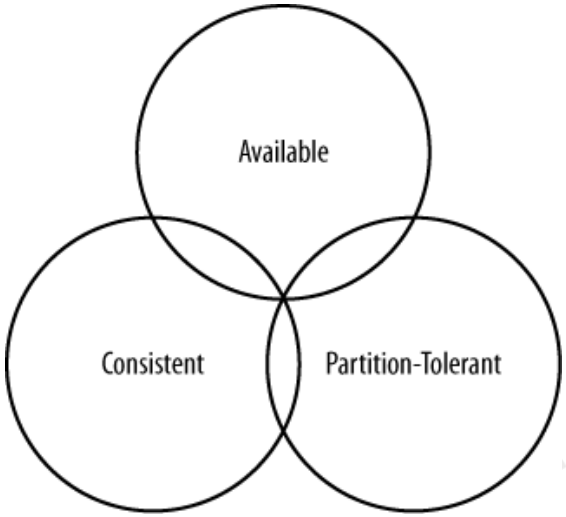
**Altamente funcional:** las bases de datos NoSQL proporcionan API altamente funcionales y tipos de datos que están diseñados específicamente para cada uno de sus respectivos modelos de datos.

"Es imposible para un sistema computacional distribuido ofrecer simultáneamente las siguientes tres garantías":

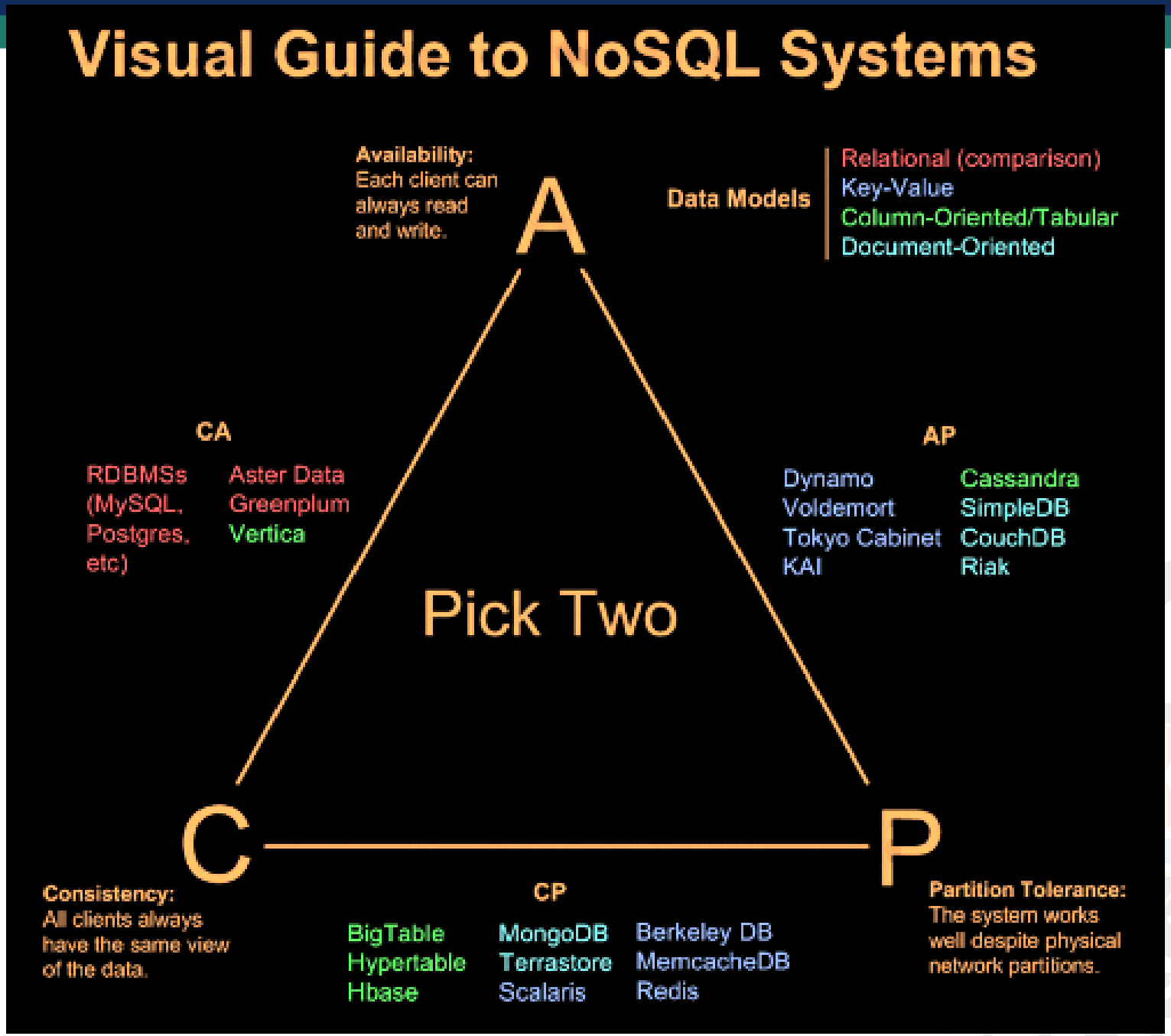
- Consistency : todos los nodos ven los mismos datos al mismo tiempo.

- Availability: garantiza que cada petición recibe una respuesta acerca de si tuvo éxito o no. Cada cliente siempre puede leer y escribir.

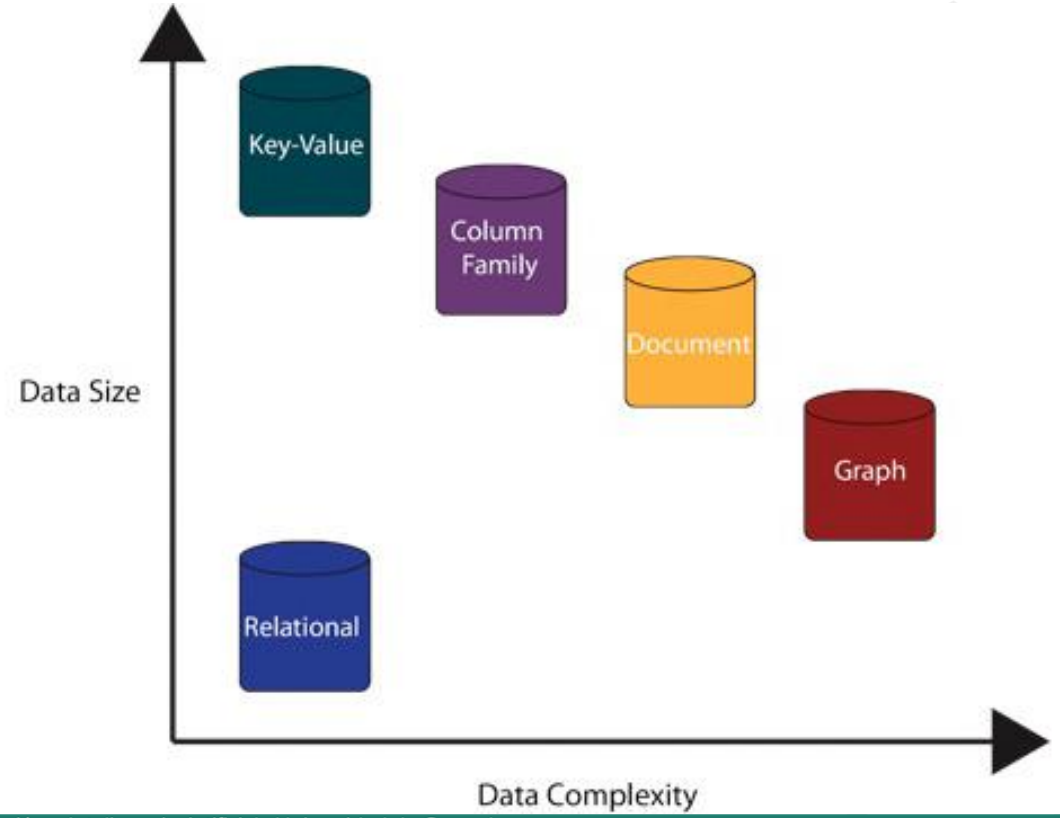
- Partition tolerance: el sistema tiene que seguir funcionando aunque existan fallos o caídas parciales en alguna de las partes en que se divida el sistema.



# Visual Guide to NoSQL Systems

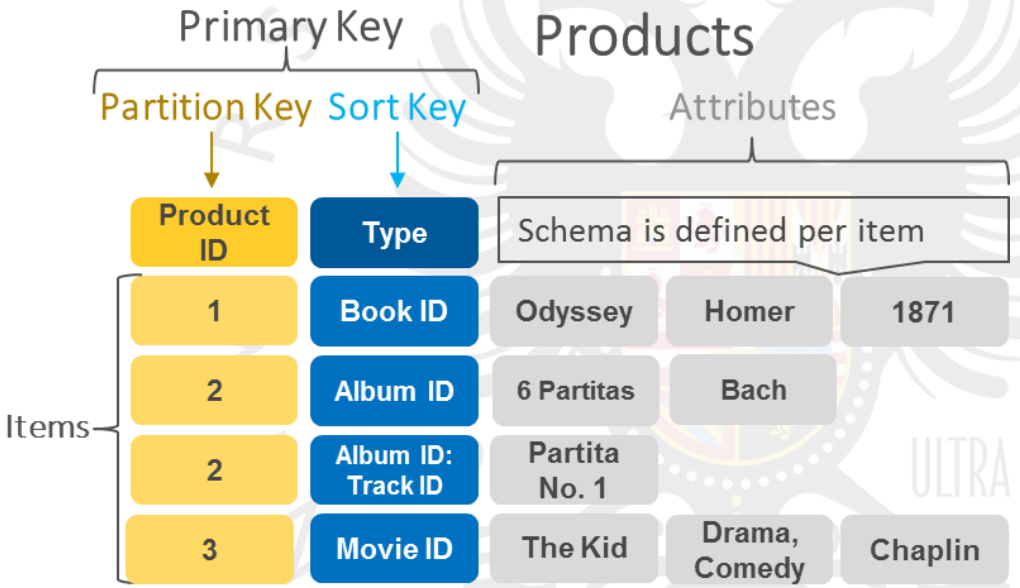


- Bases de datos de Clave-Valor
- Bases de datos de Familia de Columnas
- Bases de datos de Documentos
- Bases de datos de Grafos



Su precursor fue Amazon Dynamo
Basadas en DHT (Distributed Hash Tables).

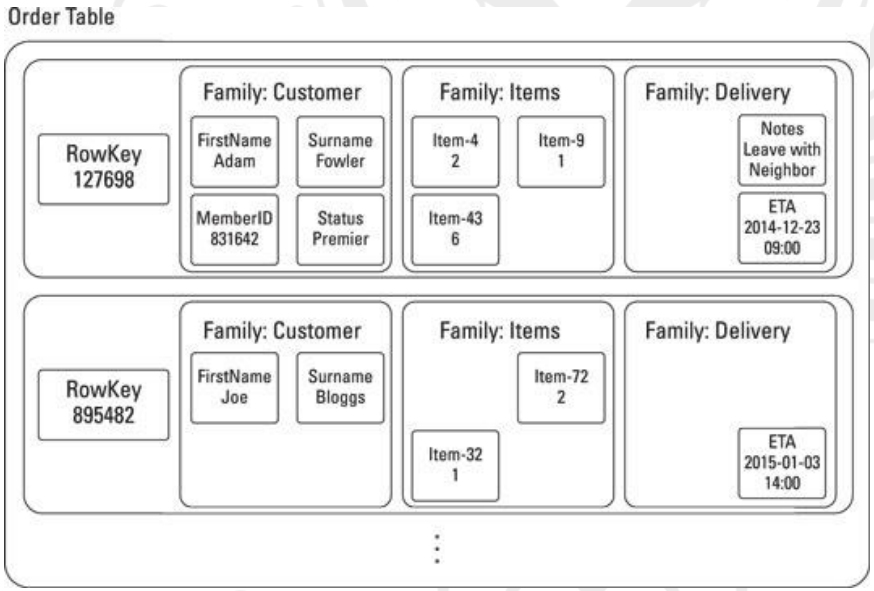
Ejemplos:
Cassandra, de Apache
BigTable, de Google
DynamoDB, de Amazon
Project Voldemort, de LinkedIn
Riak
Redis
Oracle NoSQL



Su precursor fue Google BigTable
Modelo de datos: familia de columnas, esto es, un modelo tabular donde cada fila puede tener una configuración diferente de columnas
Guardan datos por columna en vez de las BBDD tradicionales que lo hacen por filas

Ejemplos: HBase, Hypertable, Apache Accumulo, Riak

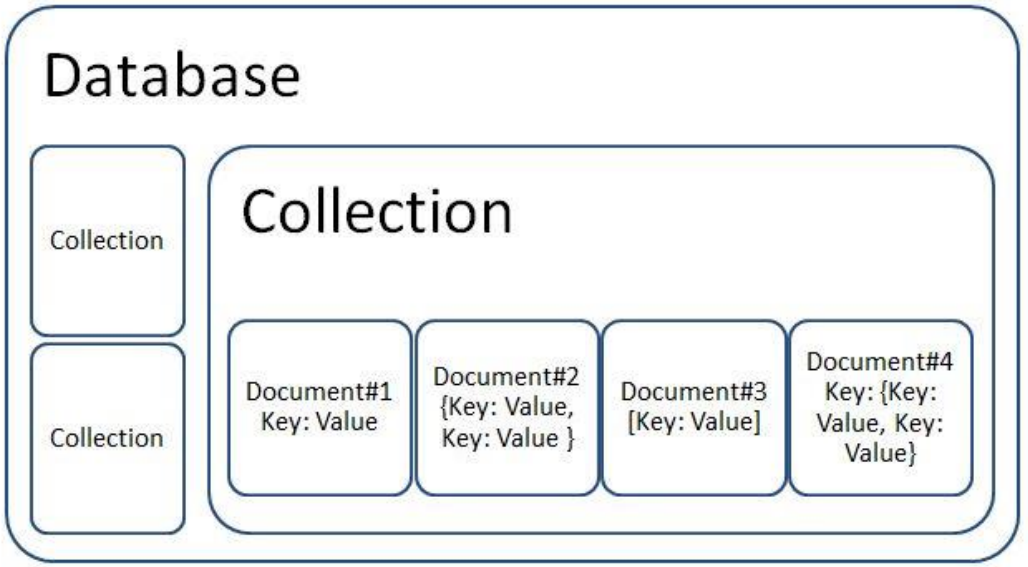
Buenas en:
Gestión de tamaño
Cargas de escrituras masivas orientadas al stream
Alta disponibilidad
MapReduce



La precursora fue Lotus Notes
Modelo de datos: colecciones de documentos (JSON, XML, BSON) que contienen colecciones de claves-valor

Ejemplos: CouchDB, MongoDB

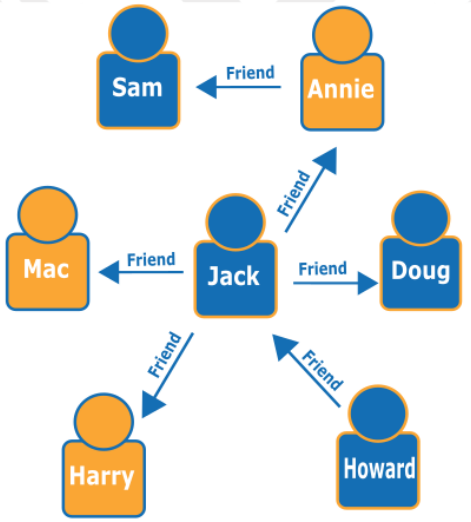
Buenas en:
Modelado de datos natural
Amigables al programador
Desarrollo rápido
Orientas a la web: CRUD



Inspiradas por Euler y la teoría de grafos
Modelo de datos: nodos, relaciones con pares clave valor en ambos

Ejemplos: AllegroGraph, VertexBD, Neo4j

Buenas en:
Modelar directamente un dominio en forma de grafo, una manera común de representar y entender datasets
Excelente rendimiento cuando los datos están interconectados y no tabulares
Realizar operaciones transaccionales que exploten la entidades



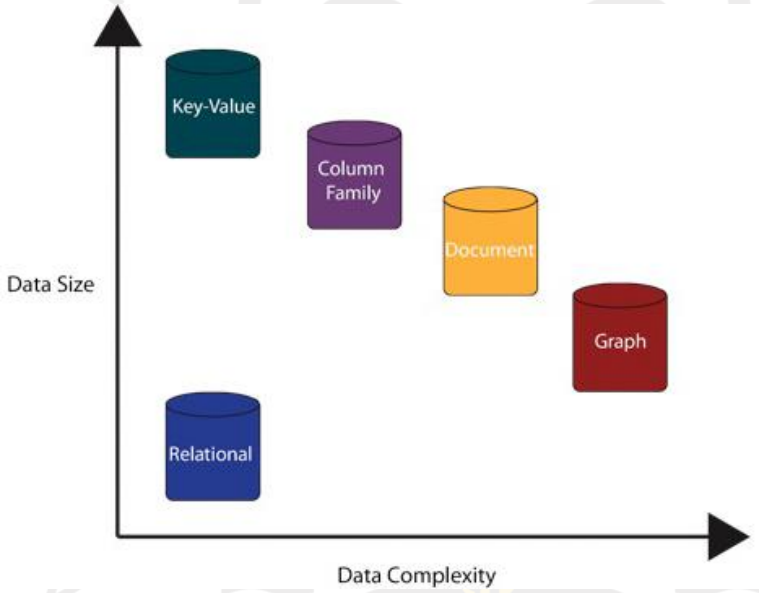
Algunas orientaciones y comparaciones pueden encontrarse en:

[35+ Use Cases For Choosing Your Next NoSQL Database]

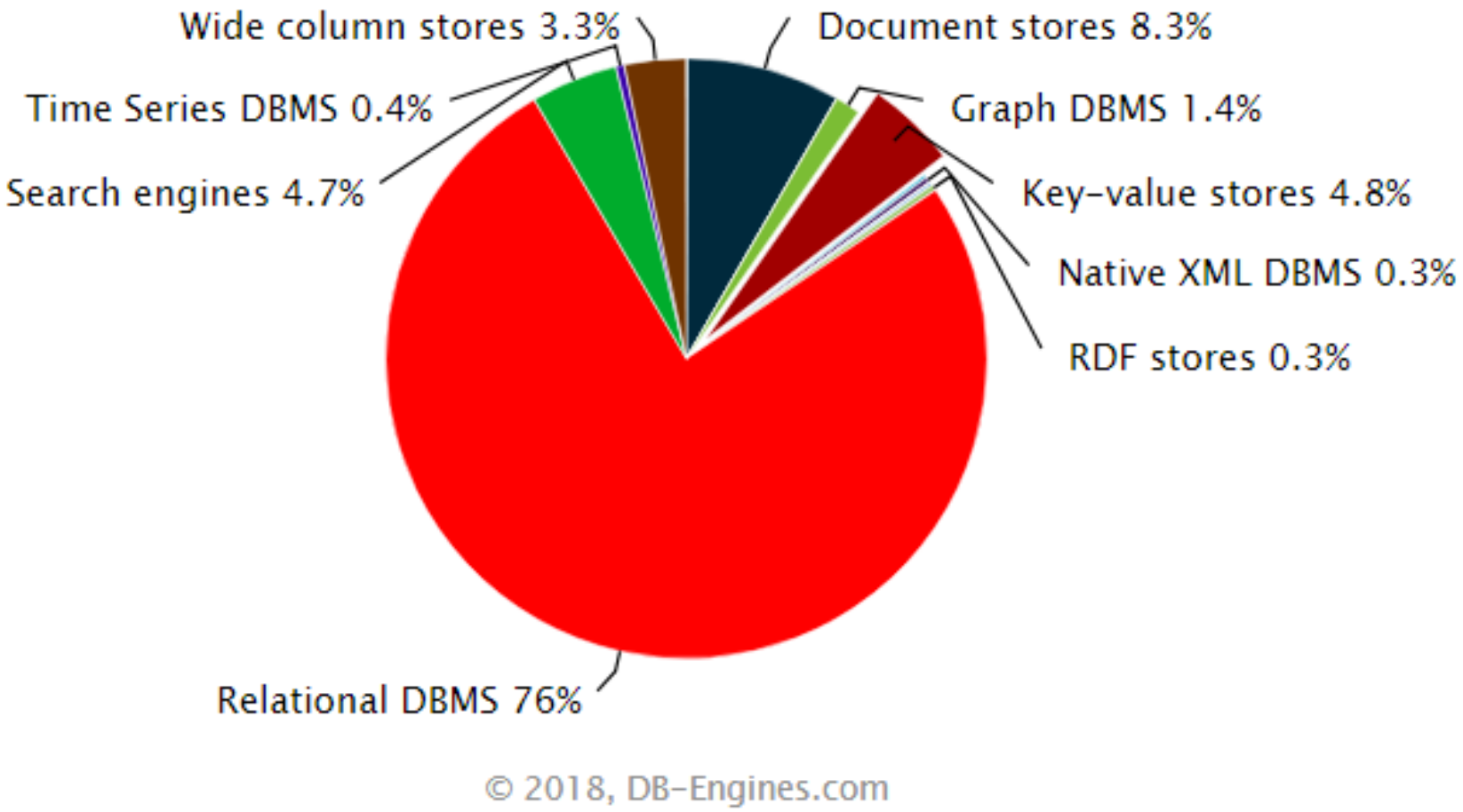
[Five Reasons to Use NoSQL]

[Which freaking database should I use?]

[Comparativa de BBDD NoSQL]



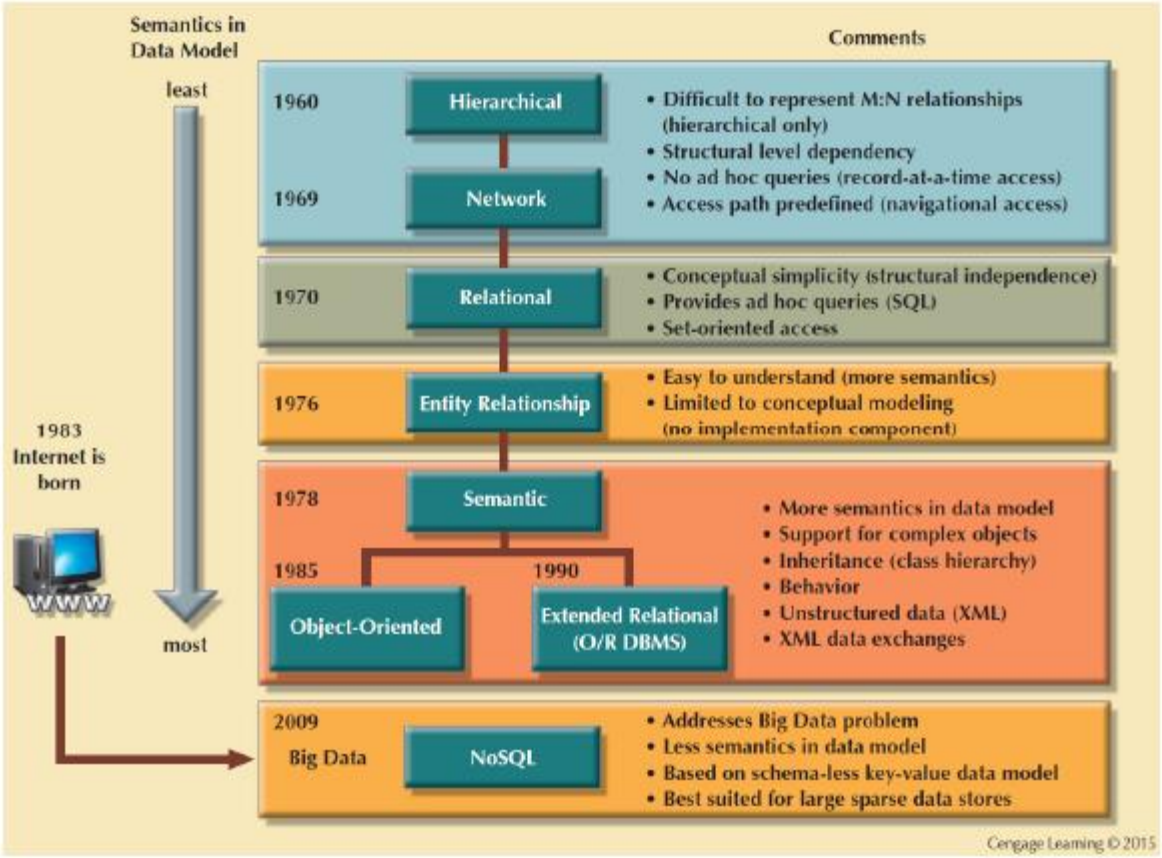
# Ranking por categoría



# Ranking por sistema gestor de bases de datos

| Rank Oct 2018 | Rank Sep 2018 | Rank Oct 2017 | DBMS | Database Model | Score Oct 2018 | Score Sep 2018 | Score Oct 2017 |
|---|---|---|---|---|---|---|---|
| 1. | 1. | 1. | Oracle | Relational DBMS | 1319.27 | +10.15 | -29.54 |
| 2. | 2. | 2. | MySQL | Relational DBMS | 1178.12 | -2.36 | -120.71 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational DBMS | 1058.33 | +7.05 | -151.99 |
| 4. | 4. | 4. | PostgreSQL | Relational DBMS | 419.39 | +12.97 | +46.12 |
| 5. | 5. | 5. | MongoDB | Document store | 363.19 | +4.39 | +33.79 |
| 6. | 6. | 6. | DB2 | Relational DBMS | 179.69 | -1.38 | -14.90 |
| 7. | ↑ 8. | ↑ 9. | Redis | Key-value store | 145.29 | +4.35 | +23.24 |
| 8. | ↓ 7. | ↑ 10. | Elasticsearch | Search engine | 142.33 | -0.28 | +22.09 |
| 9. | 9. | ↓ 7. | Microsoft Access | Relational DBMS | 136.80 | +3.41 | +7.35 |
| 10. | 10. | ↓ 8. | Cassandra | Wide column store | 123.39 | +3.83 | -1.40 |

DB-Engines Ranking



Los principales problemas de NoSQL son:
Su complejidad:
Instalación
Consultas (comprender bien MapReduce)
Los modelos de datos usados
Su falta de madurez

¿Dónde usarlas?
Datos sociales
Procesado de datos (Hadoop)
Búsqueda (Lucene)
Caching (Memcache)
Data Warehousing

¿Qué problema quieres resolver?
Transacciones
Grandes volúmenes de datos (Exabytes)
Estructura de los datos

- En NoSQL, generalmente los datos son recuperados de manera mucho más rápida que en un RDBMS, sin embargo las consultas que se pueden hacer son más limitadas y requieren trasladar complejidad a la aplicación

- RDBMS para escribir usan locks y redos para garantizar ACID, pero NoSQL no soporta a menudo Atomicy, Consistency o Durability
  - Si quieres soporte transaccional integral debes seguir usando RDBMS
  - Aplicaciones que generan informes emplean consultas complejas para las que NoSQL no es muy adecuado

- Aplicando MapReduce, las bases de datos NoSQL pueden paralelizar operaciones complejas como agregaciones estadísticas, filtros, agrupaciones o ordenación.

- Desde un punto de vista de sistemas deberíamos considerar la combinación de SQL y NoSQL:
  - LinkedIn comenzó sólo con un RDBMS, pero desarrolló su propia BBDD NoSQL (Voldemort)
  - Facebook tienen una arquitectura híbrida con Memcached y MySQL junto a un OLTP (envío de mensajes al Wall), y Cassandra/HBase para la búsqueda en la bandeja de entrada


SCALEBASE
Tokutek
dbShards
ScaleArc
AKIBAN
ScaleDB
SCHOONER
NewSQL
JustOneDB
VoltDB
NUODB
SQLFire
TRANSLATTICE
H-Store
xeround
GENIEDB
Clustrix

**Nuevas arquitecturas**
Google Spanner, CockroachDB, Altibase, Apache Ignite, Microsoft Cosmos DB, GridGain, TiDB[13], Clustrix, VoltDB, MemSQL, NuoDB, HarperDB[14] and Trafodion[

**Motores SQL: optimizada como motor de almacenamiento de SQL**
MySQL Cluster, Infobright, TokuDB, MyRocks, SQL Server (with ColumnStore and InMemory features), and MariaDB Columnstore.

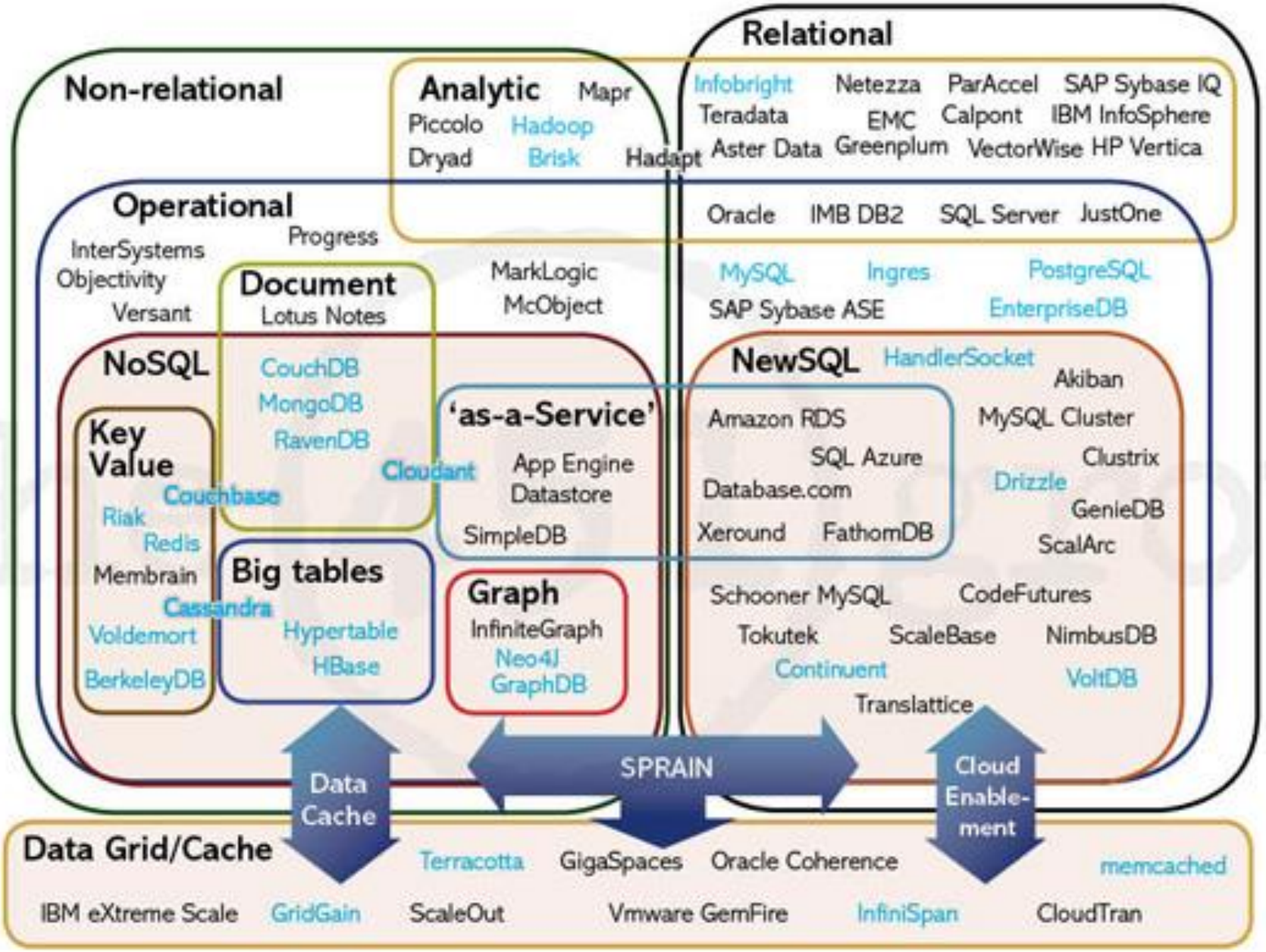
**Protección transparente**
Estos sistemas proporcionan una capa de middleware de fragmentación para dividir automáticamente las bases de datos en varios nodos dbShards, Scalearc, Scalebase y MySQL Cluster.

Las BBDD NoSQL son una clara alternativa a los RDBMS
Sobre todo para algunas aplicaciones sociales y web que requieren elevada escalabilidad

No son idóneas para todo, de hecho en la mayoría de los casos las RDBMS deberían seguir siendo la primera opción:
La capacidad de hacer JOIN y las garantías ACID son muy importantes para muchas aplicaciones

Las RDBMS actuales evolucionan para incorporar capacidades de NoSQL -> NewSQL

Lo que se va a estilar es ¡¡¡"**persistencia políglota**"!!!!


Relational
Non-relational
Analytic
Mapr
Infobright
Netezza
ParAccel
SAP Sybase IQ
Teradata
EMC
Calpont
IBM InfoSphere
Piccolo
Hadoop
Aster Data
Greenplum
VectorWise
HP Vertica
Dryad
Brisk
Hadapt
Operational
Oracle
IMB DB2
SQL Server
JustOne
Progress
InterSystems
Objectivity
Versant
Document
Lotus Notes
MarkLogic
McObject
MySQL
Ingres
PostgreSQL
SAP Sybase ASE
EnterpriseDB
NoSQL
CouchDB
MongoDB
RavenDB
Cloudant
'as-a-Service'
App Engine
Datastore
SimpleDB
NewSQL
HandlerSocket
Akiban
Amazon RDS
SQL Azure
Database.com
Xeround
FathomDB
MySQL Cluster
Clustrix
Drizzle
GenieDB
ScalArc
Key Value
Couchbase
Riak
Redis
Membrain
Cassandra
Voldemort
BerkeleyDB
Big tables
Hypertable
HBase
Graph
InfiniteGraph
Neo4J
GraphDB
Schooner MySQL
CodeFutures
Tokutek
ScaleBase
NimbusDB
Continuent
VoltDB
Translattice
Data Cache
SPRAIN
Cloud Enable-ment
Data Grid/Cache
Terracotta
GigaSpaces
Oracle Coherence
memcached
IBM eXtreme Scale
GridGain
ScaleOut
Vmware GemFire
InfiniSpan
CloudTran