NoSQL

Introducción

¿DE DONDE PARTIMOS Y CÓMO LLEGAMOS HASTA NOSQL?

Comencemos con algo de historia...

Tres Revoluciones de las Bases de Datos

TRES REVOLUCIONES

Primera Revolución

1951: Magnetic Tape 1955: Magnetic Disk

1965: ISAM

1965: Modelo Jerárquico (IBM IMS)

1969: Modelo de Red (IDMS Mainframes)

Segunda Revolución

1970: Codd's Paper 1974: System R

1978: Oracle 1981: Informix 1984: DB2

1987: Sysbase 1989: SQL Server

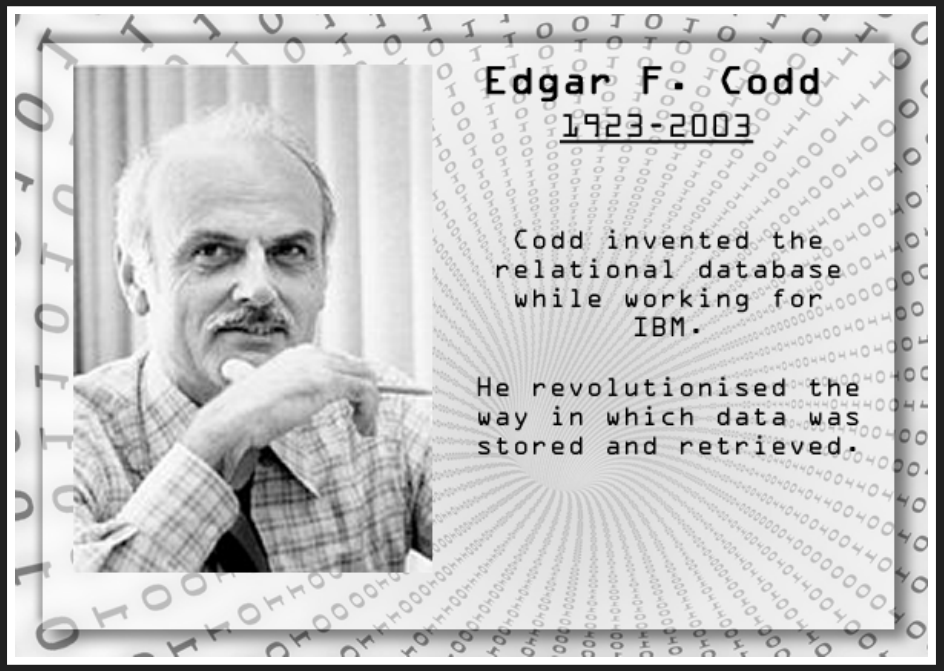
1990: OODBMS (Non Relational) 1995: MySQL

1995: PostgreSQL

SEGUNDA REVOLUCIÓN: EL MODELO RELACIONAL

¿Qué generó su aparición?

EDGARD COOD



- Paper Fundacional: A Relational Model for Large Shared Data Banks
- Trabajaba en IBM cuando escribió y publico su trabajo sobre el modelo relacional. ¿Saben que BD tiene IBM en el mercado?
- IBM no quizo implementar el modelo de Cood.
- ¿Saben cual es la BD con más éxito en el mercado?
- Larry Ellison diseño e implementó su Base de Datos basándose en las ideas de Codd. ¿Cual es la empresa de Larry?

EL MODELO RELACIONAL

¿En qué se diferencia de los modelos jerárquico y de Red?

El Modelo Relacional describe como los datos deben ser presentados a los usuarios, en lugar de como deben ser persistidos.

EL MODELO RELACIONAL

Fortalezas, principales características

- Lenguaje de Consulta SQL (estandar)
- Las formas normales definen el buen diseño (no existe en el diseño de objetos).
- Transacciones ACID (Agregado por Jim Gray (IBM) al trabajo de Cood).
- Consistencia: Transacciones, Claves Forañas, Restricciones. Todas herramientas para garantizar la consistencia.

TERCERA REVOLUCIÓN (2005)

NoSQL

¿Qué generó su aparición?

NoSQL

¿Qué generó su aparición?

Entre 1995 y 2005 Internet fue transformándose... partiendo desde conexiones dial-up, utilizadas mas que nada por “curiosos” hacia el sistema de comunicación mas importante de nuestra civilización.

NoSQL

¿Qué generó su aparición?

- Hoy Google procesa **40.000 búsquedas por segundo** en promedio, que se traduce en 3.5 billones por día.
- En 1999 le llevó un mes a Google obtener (crawl) e indexar 50 millones de páginas web.
- En 2012 le llevo menos de 1 minuto la misma tarea.

NoSQL

¿Qué generó su aparición?

- **Amazon** necesitaba procesar 35 ordenes de compra por segundo.
- **Facebook**, **Twitter** y otras redes sociales tuvieron que lidear también con un número **altísimo** de demanda.

APLICACIONES "PLANET SIZE"

- **Google**, **Amazon**, **Facebook**, **Twitter**, y recibieron el nombre de aplicaciones "Planet Size". O también se las denomina "Web Scale".

> *Su demanda es Tan alta... pero **TAN ALTA**... que no funcionaban en un único servidor de base de datos, por mas grande que éste fuera.*

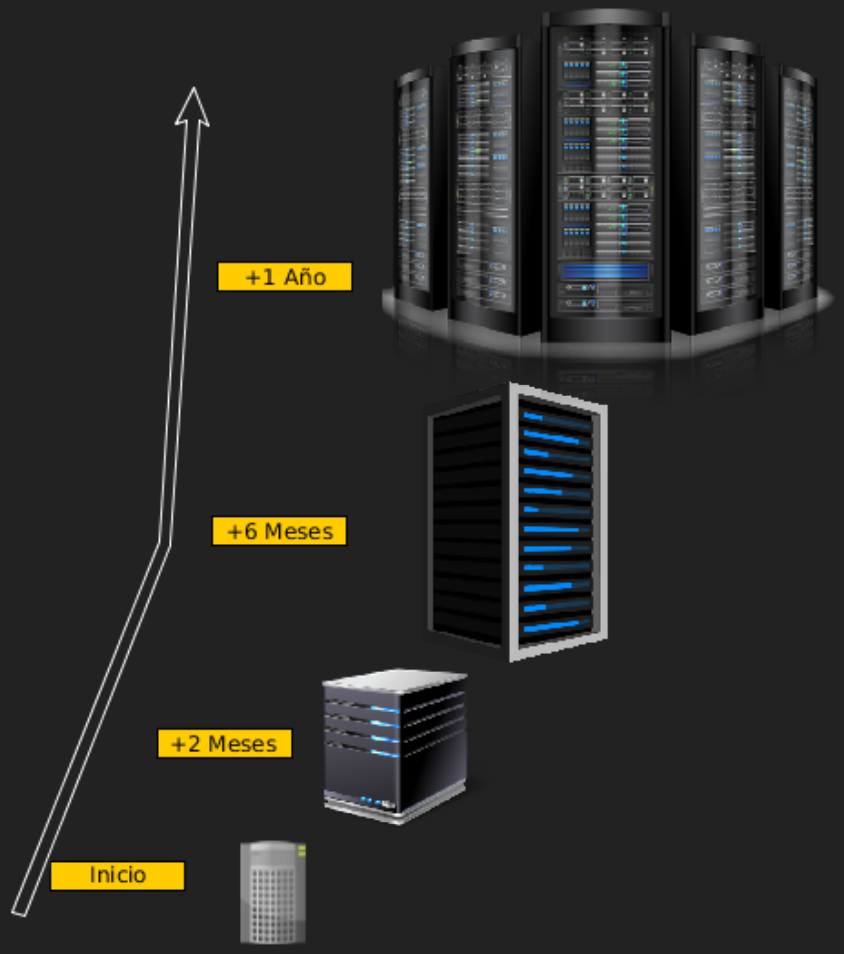
ESCALANDO BASES DE DATOS

DOS FORMAS DE ESCALAR

Escalar Vertical (scale-up)

Escalar Horizontal (scale-out)

ESCALAR VERTICAL



- Scale-up
- Un server mas grande cada vez que notemos degradarse la performance.

ESCALAR HORIZONTAL

Scale-out: Agregamos mas servers simples al cluster.

La única forma para soportar "Planet Size" Apps (google, facebook, twitter, amazon, etc).

APLICACIONES "PLANET SIZE"

¿entonces? Un único server no caminaba... ¿que hicieron?

- Distribuyeron la carga en varios servers (segun la demanda cerca de 100).
- **Cluster**: Varias máquinas conectadas entre sí que funcionan como si fueran solo una.

TEOREMA DE CAP

Al distribuir una base de datos... te encontras con el Teorema de CAP = Consistency, Availability, Partition Tolerance

¿Sabemos que es Consistencia? ¿Sabemos que es Disponibilidad? ¿Tolerante a Fallos de red?

TEOREMA DE CAP

Solo puedo ser CP o AP

- Supongamos dos nodos conetados vía red...
- Si quiero ser consistente, al escribir un dato en un nodo, no puedo devolver el control al usuario hasta que ese dato se replique en todos los nodos.
- Supongamos que hay una corte en la red...
- Si los mensajes no pueden viajar de un nodo al otro, en el escenario donde un usuario escribe en un nodo, entonces tengo dos opciones: le devuelvo al usuario el control sin terminar de replicar en el resto de los nodos, no soy consistente pero si tengo disponibilidad o hago esperar al usuario hasta que el dato nuevo pueda llegar, entonces soy consistente pero no tengo disponibilidad.

DISTRIBUIR UNA BD RELACIONAL

Las BDs relacionales NO fueron diseñadas para distribuirse.

Todas sus características fuertes van en contra de distribuir los datos en más de un único server.

Veamos algunos ejemplos.

DISTRIBUIR UNA BD RELACIONAL

Pensemos que estamos en un cluster con *n* número de máquinas.

- Consistencia: Todos los servers deben estar idénticos, si escribo en uno debo replicarlo en el otro y no puedo devolver el control al cliente hasta que todos los servers en el cluster esten iguales.
- Transacciones: imagínense arrancando una transacción y teniendo que bloquear tablas que estan en otro server.
- La red del cluster y su latencia hace que esto sea impracticable. Muy lento sería.

DISTRIBUIR UNA BD RELACIONAL

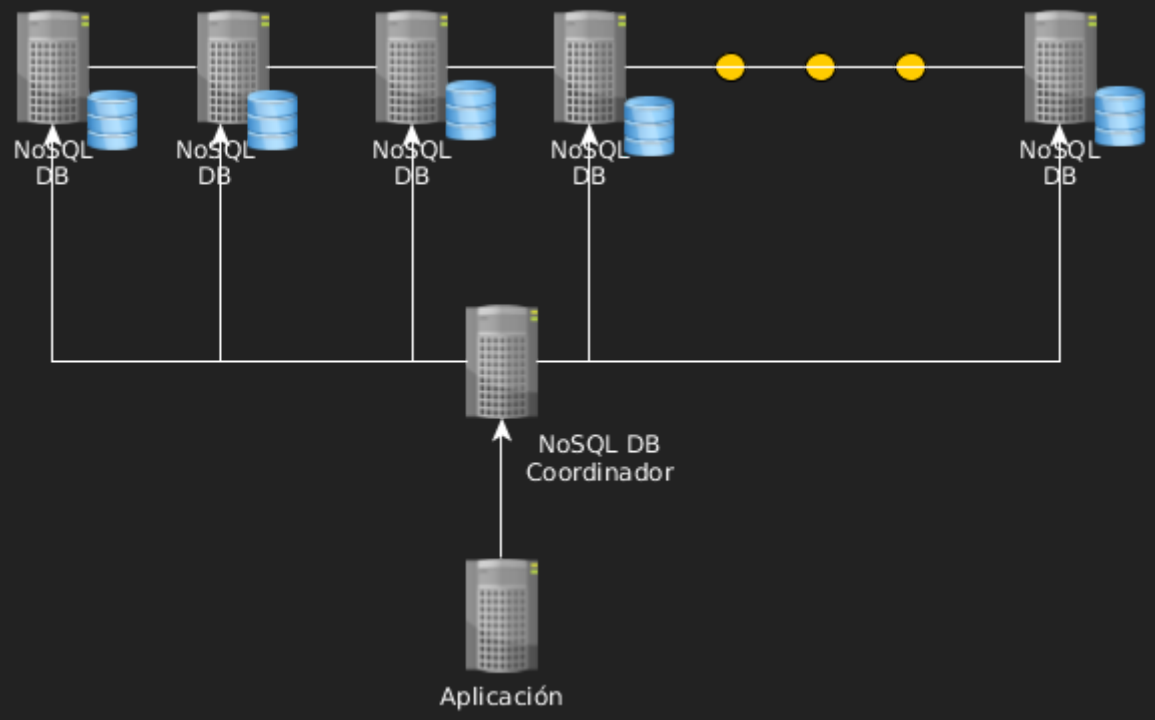
Todas estas cuestiones hacen que las BDs relacionales solo puedan hostearse en un único server.

¿Queremos soportar aplicaciones Planet Size?

Entonces no tenemos opción, tenemos que relajar estas cuestiones (sobre todo la consistencia) para poder distribuír.

NoSQL

Diseñadas para escalar en forma horizontal



- Puede haber un coordinador o los nodos realizan esa tarea.

NoSQL

Hoy es una movida MUY grande... y se valen de otras características también:

Esquema Flexible: No tienen un esquema rígido como el relacional, es flexible (json en general).

Big Data: Al distribuir soportan Big Data (por esto Google puede indexar la web completa...).

Open Source: Y sobre HW más económico al aprovechar el cluster.

NoSQL BASE vs Relacional ACID

- **B**asically **A**vailable: Siempre habra respuesta
- **S**oft-State: Los datos pueden cambiar a raiz de cambios en otros nodos (eventual consistency)
- **E**ventual Consistency: La cosistencia será eventual, o sea, cuando sea posible replicar un cambio en todos los nodos del cluster.

Consistencia vs Disponibilidad

- Las BDs relacionales favorecen la **consistencia** por sobre otro atributo.
- Las BDs NoSQL favorecen la **disponibilidad** por sobre la consistencia.
- Las BDs NoSQL ofrecen **Consistencia Eventual**.

NoSQL: ¿Y que significa?

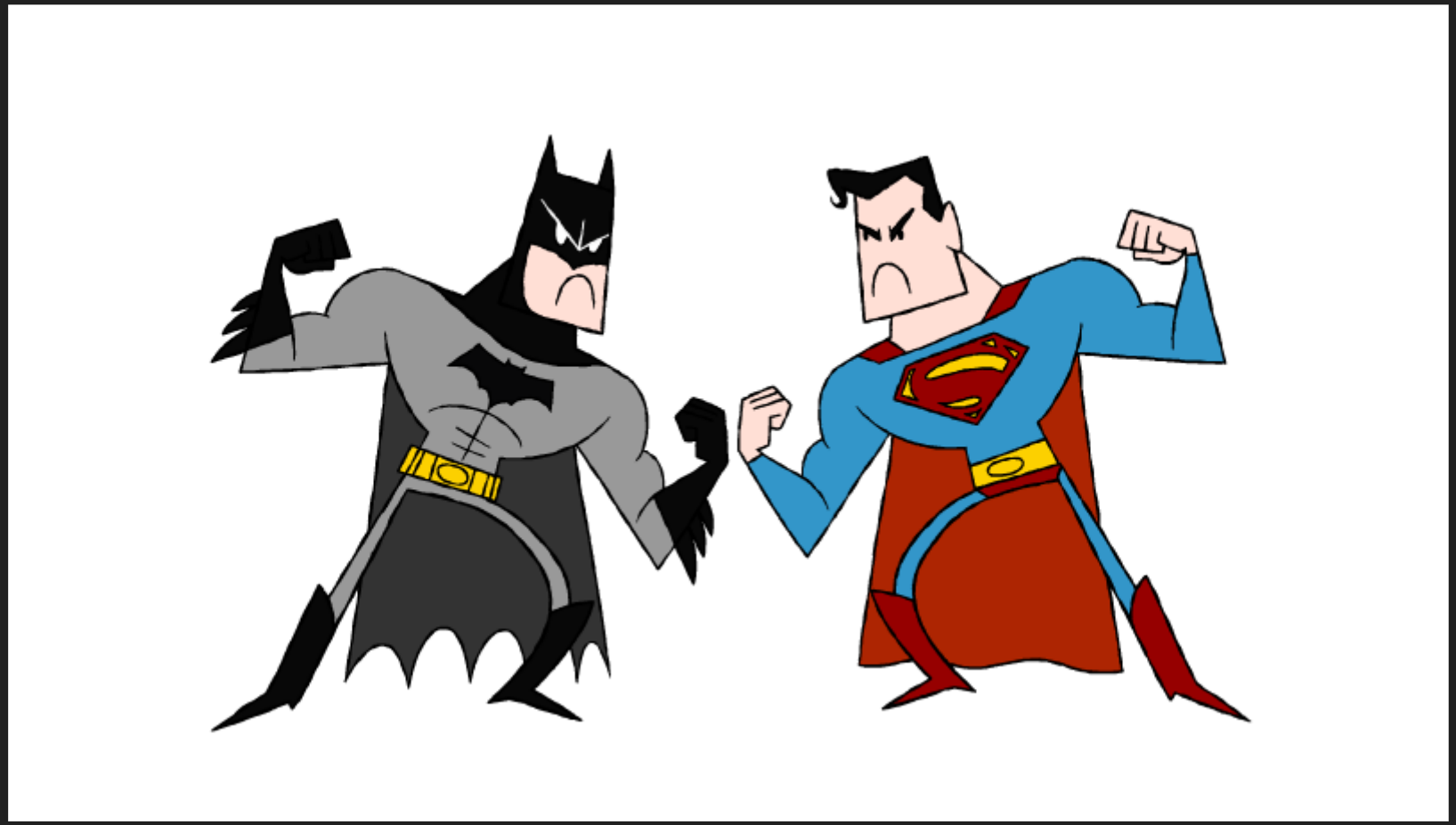
NoSQL: Not Only SQL

nosql-database.org: Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally scalable.

Noten que quienes las inventaron son Google y Amazon, simplemente porque sino su negocio no prosperaba.

Esto no es todo...

En la competencia entre SQL/Relacional vs NoSQL...



Tambien tenemos:

NewSQL


WHAT ?
HAPPY-WISHE
UT I HAVE NEVER HEARD OF TH

NewSQL

- Extrañamos el Lenguaje SQL
- Extrañamos ser Consistentes, teniendo alta disponibilidad.
- Y queremos distribuir, escalar en forma horizontal

Google Spanner, VoltDB, YugaByte

NewSQL
GOOGLE SPANNER

Spanner claims to be consistent and available. Despite being a global distributed system, Spanner claims to be consistent and highly available, which implies there are no partitions and thus many are skeptical. Does this mean that Spanner is a CA system as defined by CAP? The short answer is “no” technically, but “yes” in effect and its users can and do assume CA.

The purist answer is “no” because partitions can happen and in fact have happened at Google, and during (some) partitions, Spanner chooses C and forfeits A. It is technically a CP system. We explore the impact of partitions below.