Bases de Datos No Convencionales

Máster en Business Analytics y Big Data

Contenidos de la sesión

- Introducción NoSQL
- Persistencia políglota
- Teorema CAP
- Introducción a los modelos NoSQL

Introdución NoSQL

Introducción

- Origen del término: En 1998, Carlo Strozzi lo usa para nombrar su SGBD, que no usaba el lenguaje SQL.
  - ¡Pero no que tiene que ver con las BBDD NoSQL actuales!
- Origen del concepto: En 2009 se usa para hacer referencia a una nueva generación de bases de datos no relacionales y altamente distribuidas
  - Estas BBDD empiezan a hacer aparición en la primera década del 2000:

    BigTable (Google) en 2003, Marklogic en 2005.

Introducción

- ¿Por qué surge el concepto?
  - Auge de Internet:

    ⇧ servicios on line + ⇧ usuarios conectados = ⇧ ⇧ volumen de datos
  - El tratamiento centralizado de los datos deja de ser eficiente:
    - Necesidad de garantizar el servicio a un gran número de usuarios (potencialmente desconocido) en cualquier momento.
    - Necesidad de buenos tiempos de respuesta independiente del número de usuarios conectados y su ubicación.

Introducción

- Teorema CAP:
  - Eric Brewer (2000): Las propiedades de un sistema distribuido son la Consistencia, la Disponibilidad y la Tolerancia a particiones, pero en un sistema distribuido sólo se puede garantizar 2 propiedades simultáneamente.
  - S. Gilbert y N. A. Lynch (2002): Prueban formalmente la validez del teorema CAP para ciertos modelos distribuidos.

Introducción

- Características de la base de datos NoSQL:
  - No ofrecen SQL como lenguaje estándar.
  - Esquema flexible (*schemaless*)
  - No garantizan las propiedades ACID (*Atomicidad, Consistencia, Aislamiento y Definitividad*) al completo para mejorar el rendimiento y aumentar la disponibilidad.
  - Favorecen la escalabilidad, especialmente horizontal.
  - Ofrecen solución a los inconvenientes y rigidez del modelo relacional.
  - En general, son:
    - Distribuidas
    - De código abierto

Introducción

- Característica de la aplicación que usa bases de datos NoSQL:
  - Datos con estructura variable.
  - Múltiples orígenes de datos, con diferente formato.
    - Comparadores online
  - Altamente relacionados.
    - Facebook
  - Los datos fluyen en tiempo real.
    - Twitter
  - Introducción masiva de BI y *data warehouse*
  - Grandes volúmenes de datos (PB).
  - Flexibilidad y rendimiento
    - Tiendas online
  - El proceso de los datos se debe hacer en "tiempo real".
    - Aplicaciones de bolsa

Introducción

BBDD Relacionales	BBDD NoSQL
Un único modelo de datos	Varios modelos de datos
Esquema estricto	Esquema flexible (*schemaless*)
Estándar SQL	Sin estándares

Persistencia políglota

Persistencia políglota

- La persistencia políglota consiste en el uso de diferentes tecnologías de almacenamiento en un mismo proyecto o empresa para dar respuesta a las diferentes necesidades.
  - La persistencia políglota no consiste en substituir una tecnología de almacenamiento de datos por otra.
  - Se habilita la comunicación entre las tecnologías y su acceso por una aplicación externa.

Persistencia políglota

- ¿Por qué no siempre funciona la estrategia one size fits all?
  - La información en determinados dominios ha cambiado:
    - Datos semi-estructurados
    - Datos sin estructura
    - Flujos de datos
    - Datos provenientes de sensores
    - Información incierta
    - Información en forma de grafos
    - Estructuras de datos compleja

Persistencia políglota

- ¿Por qué no siempre funciona la estrategia one size fits all?
  - Hay necesidad de almacenamiento de datos de forma distribuida
  - Buena parte de las aplicaciones actuales necesitan realizar operaciones simples sobre modelos de datos complejos.
  - Limitaciones de SQL para realizar ciertos tipo de consultas
  - La dificultad en la modificación del esquema de una base de datos relacional
  - Las propiedades ACID no son siempre necesarias y pueden ralentizar el sistema.
    - En motores de búsqueda (devuelven resultados diferentes a dos usuarios simultáneamente)
    - En sitios web como Amazon (devuelven diferentes *reviews* de un mismo producto a dos usuario)

Persistencia políglota

- Dominios en los que no funciona un SGBDR:
  - Data warehouse y BI
  - Flujos de datos financieros en tiempo real
  - Redes sociales y motores de búsqueda
  - Sistemas de recomendación
  - Plataformas de juegos en línea
  - Geolocalización móvil y sensores

Persistencia políglota

- Inconvenientes de la persistencia políglota:
  - Falta de madurez de los productos.
  - Escasez de profesionales.
  - Ausencia de estándares.
  - Administración compleja.
  - Menor soporte.
  - Complejidad de esquemas.

Teorema CAP y Modelo BASE

Teorema CAP y Modelo BASE

- Desde los 80 impera el modelo ACID para transacciones:
  - Atomicidad: el conjunto de operaciones que constituyen la transacción es la unidad indivisible de ejecución.
  - Consistencia: la ejecución de la transacción tiene que preservar la consistencia de la BD.
  - Aislamiento: una transacción no puede ver interferida su ejecución por otras transacciones que se estén ejecutando de forma concurrente con ella.
  - Definitividad/Durabilidad: los resultados producidos por una transacción que confirma tienen que ser definitivos en la BD.

Teorema CAP y Modelo BASE

- Para mantener las propiedades de consistencia y aislamiento el SGBD implementa:
  - Protocolos de control de concurrencia
  - Protocolos para la gestión de replicas (BD distribuidas)
- Para mantener las propiedades de atomicidad y definitividad el SGBD implementa:
  - Protocolos de recuperación: requieren *logs* y *backups*.
  - Protocolo de confirmación en dos fases (BD distribuidas)

Teorema CAP y Modelo BASE

- Los SGBDR trabajan con transacciones que verifican las propiedades ACID.
- Es posible relajar el nivel de aislamiento de las transacciones, lo que implica que:
  - Se pueden producir interferencias
  - Se mejora el rendimiento
  - Disminuye la sobrecarga del SGBD

Teorema CAP y Modelo BASE

- Soportar un modelo de transacciones ACID en BD que almacenan grandes volúmenes de datos, que están distribuidas y con replicación de datos es complejo y puede causar problemas de rendimiento.
  - Se han propuesto modelos transaccionales alternativos (modelo BASE)
  - Se evitan transacciones que incluyan diversas operaciones (en modelos de agregación)
- Las BD NoSQL de grafos, en general, se ajustan a modelos de transacciones ACID.

Teorema CAP y Modelo BASE

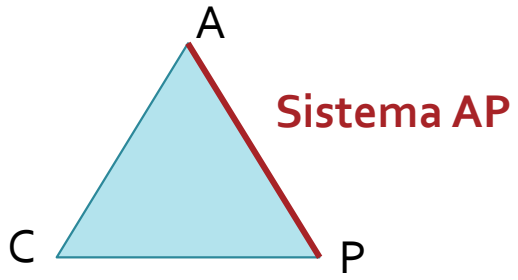
- El teorema CAP enuncia las propiedades deseables de un sistema distribuido, y cómo éstas se pueden ver comprometidas ante situaciones de avería.
- El teorema del CAP captura la idea de que la consistencia puede no ser compatible con la gestión de datos altamente distribuidos.
- NoSQL aboga sobre su base por la relajación de las propiedades ACID.

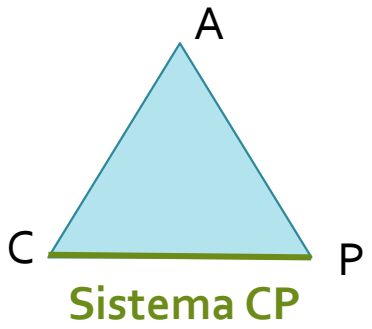
Teorema CAP y Modelo BASE

- El teorema CAP enuncia que es imposible garantizar simultáneamente las tres características.
  - Consistencia (*consistency*): los usuarios del sistema tienen que poder recuperar siempre los mismos datos en un mismo instante de tiempo.
    - Consistencia en las réplicas.
  - Disponibilidad *(availability):* las peticiones de servicio enviadas por los usuarios a un nodo que está disponible deben obtener su debida respuesta.
  - Tolerancia a particiones (*tolerance to network partitions)*: el sistema debe proporcionar servicio a los usuarios a pesar de que se puedan producir situaciones de avería que causen que el sistema quede particionado en diferentes componentes aislados.

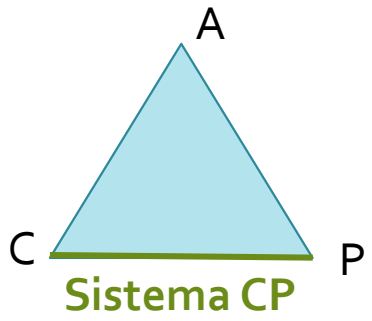
Teorema CAP y Modelo BASE

- En sistemas altamente distribuidos son factibles situaciones de avería que particionen el sistema.
  - Bajo estas condiciones, el sistema debe garantizar seguir dando servicio: Se garantiza *P*
- Hay que decidir qué priorizar:
  - Consistencia (sistema CP): El sistema siempre puede mostrar una visión consistente de los datos, aunque no esté totalmente disponible en presencia de particiones.
  - Disponibilidad (sistema es AP): El sistema siempre está disponible, aunque temporalmente puede mostrar datos inconsistentes cuando existen particiones.





Teorema CAP y Modelo BASE



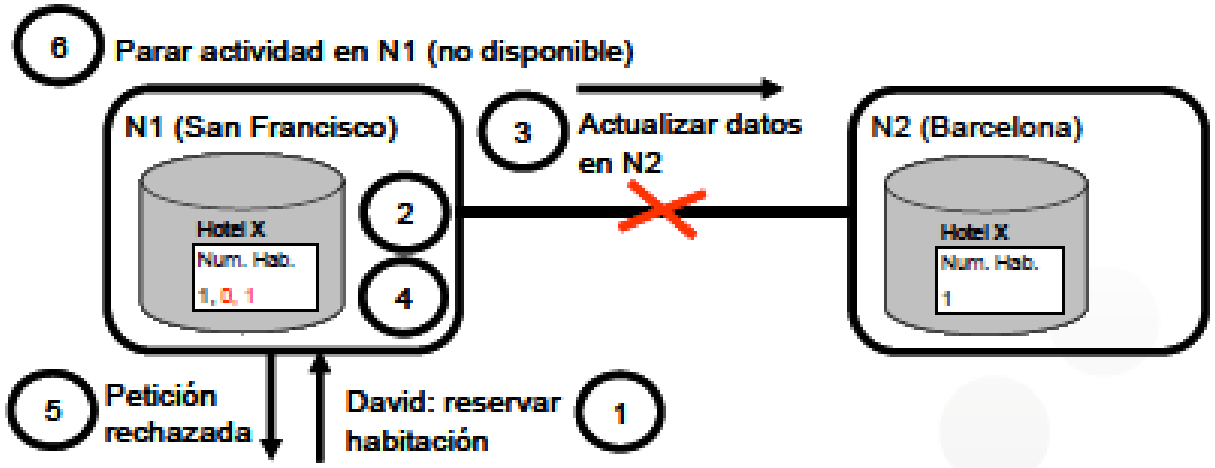
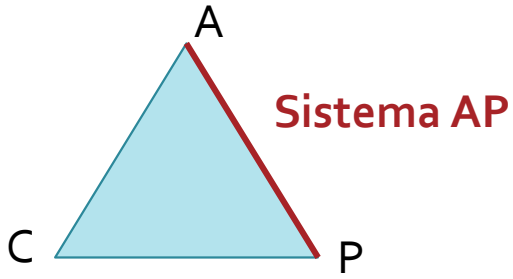
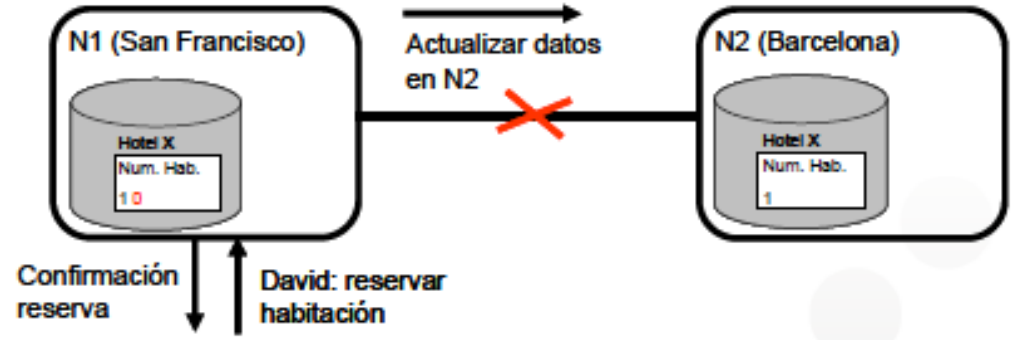
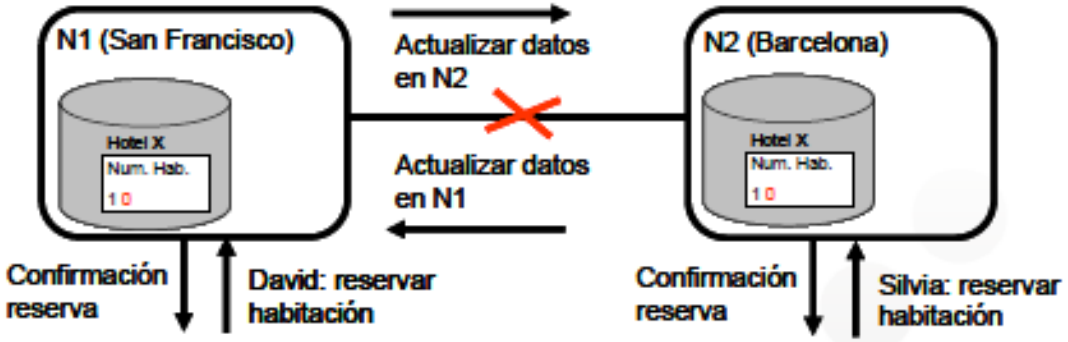


Imagen extraída de documentación de la UOC. Autores: E. Rodríguez y J. Conesa

Teorema CAP y Modelo BASE







Imágenes extraída de documentación de la UOC. Autores: E. Rodríguez y J. Conesa

CIFF
Universidad de Alcalá

Teorema CAP y Modelo BASE

- Modelo BASE: Modelo de transacciones en los que se basan los sistemas AP
  - Disponibilidad limitada (*Basic Availability*): que una parte del sistema no esté disponible, no significa que todo el sistema deje de funcionar.
  - Estado flexible *(Soft-state):* los datos puedan estar temporalmente en un estado no consistente.
    - Se acepta la perdida de datos cuando no pueden resolverse los conflictos.
  - Consistencia final en el tiempo (*Eventual consistency*): a la larga todas las réplicas acaban convergiendo.

Modelos NoSQL

Modelos NoSQL

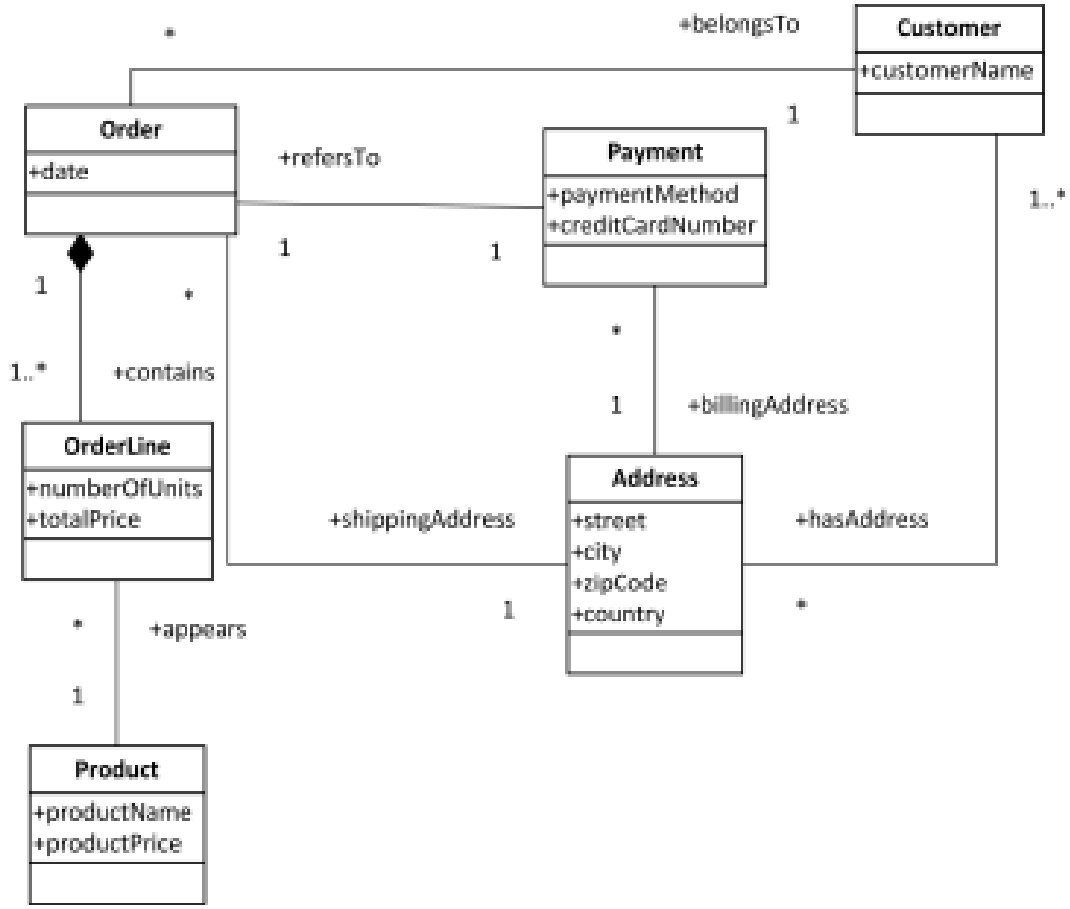
- Un modelo de datos (o modelo de base de datos) constituye el conjunto de componentes o herramientas que proporciona el sistema gestor de la base de datos para estructurar y manipular los datos.
- En bases de datos NoSQL hay dos familias de modelos:
  - Modelo de agregación:
    - Modelo clave-valor
    - Modelo documental
    - Modelo de columnas
  - Modelo de grafos

Modelos NoSQL

- Modelos de agregación:
  - Los modelos de agregación se basan en la noción de agregado.
  - Agregado: colección de objetos relacionados que deseamos tratar como una unidad independiente (desde un punto de vista semántico) a efectos de:
    - Acceso y manipulación: la unidad mínima de intercambio entre las aplicaciones y el SGBD es el agregado
    - Consistencia y control de concurrencia: la definitividad de los cambios se asegura sobre el agregado y el mantenimiento de su integridad se circunscriben al ámbito del agregado (desde la aplicación)
    - Distribución de datos

Modelos NoSQL

- R1: Recuperar toda la información relativa a un pedido (*Order*) a efectos de gestión de envío.



Modelos NoSQL

- Modelo relacional:
  - R1: combinación (*join*) entre pedido (*Order*), línea de pedido (*OrderLine*), producto (*Product*) y dirección (*Address*).

Customer(customerId, customerName)

Address(addressId, street, city, zipCode, country)

CustomerAddress(customerId, addressId)
{customerId} clave foránea a Customer
{addressId} clave foránea a Address

Product(productId, productName, productPrice)

Payment(paymentId, paymentMethod, creditCardNumber, billingAddressId)
{billingAdressId} clave foránea a Address

Order(orderId, date, customerId, paymentId, shippingAddressId)
{customerId} clave foránea a Customer
{paymentId} clave foránea a Payment
{shippingAddressId} clave foránea a Address

OrderLine(orderLineId, orderId, productId, numberOfUnits, totalPrice)
{orderId, productId} clave alternativa
{orderId} clave foránea a Order
{productId} clave foránea a Product

Modelos NoSQL

- Modelo agregación:
  - R1: Accedamos a la información necesaria para la gestión del envío del pedido a la vez.

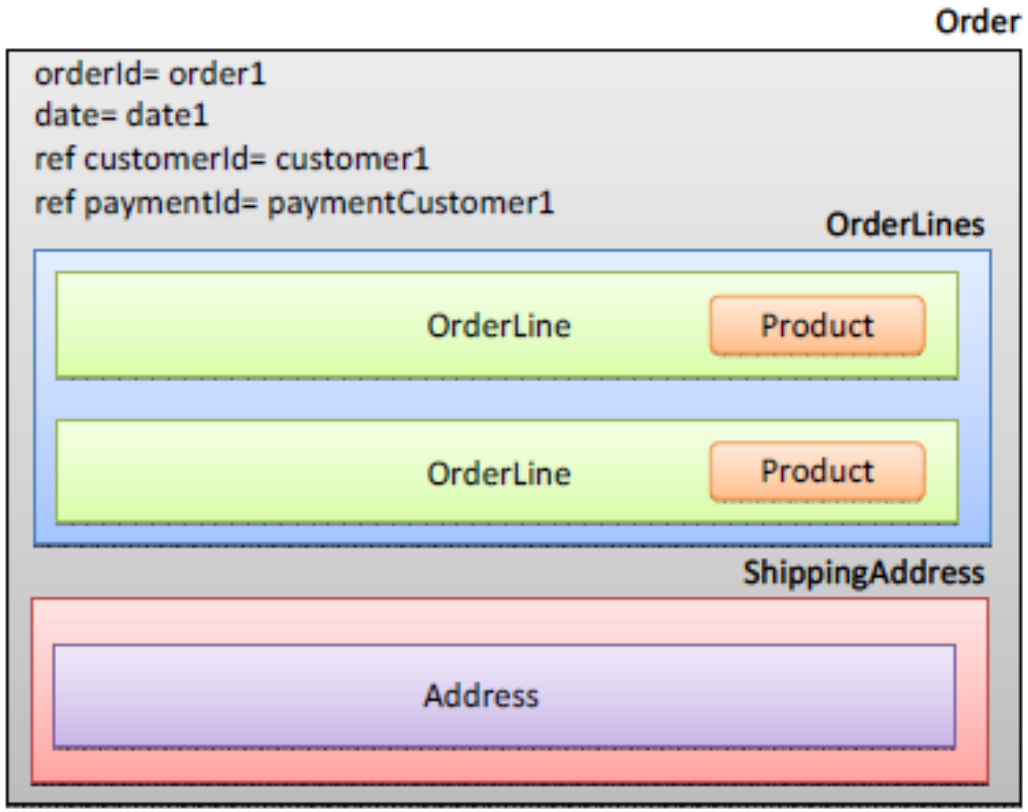


Imagen extraída de documentación de la UOC. Autores: E. Rodriguez, J. Conesa y P. Urbón

Modelos NoSQL

- Características de los modelos de agregación:
  - El agregado es la unidad de acceso y manipulación.
  - El agregado puede ayudar a reducir los accesos necesarios para recuperar los datos.
  - La consistencia se mantiene a nivel de agregado.
    - Así, por ejemplo, la BD garantiza la definitividad de los cambios producidos a nivel de cada agregado.

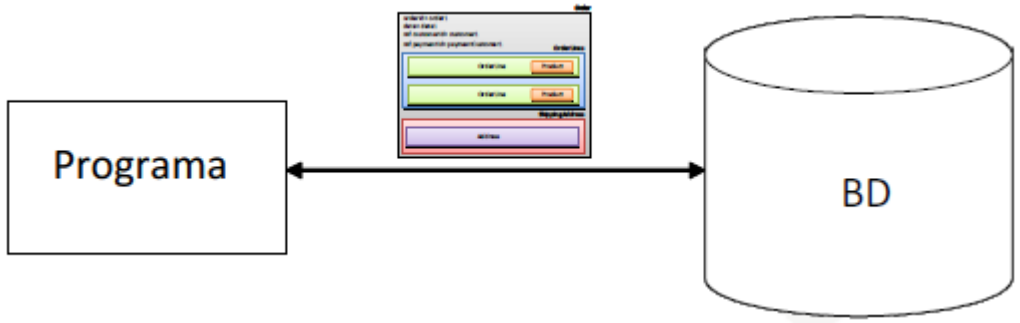


Imagen extraída de documentación de la UOC. Autores: E. Rodriguez, J. Conesa y P. Urbón

Modelos NoSQL

- Características de los modelos de agregación:
  - Dependiendo del diseño de agregados efectuado, hay objetos que pueden quedar repartidos en diferentes agregados.
    - La responsabilidad del mantenimiento de la consistencia en objetos repartidos en múltiples agregados es de la aplicación.

Imagen extraída de documentación de la UOC. Autores: E. Rodriguez, J. Conesa y P. Urbón

Modelos NoSQL

- Características de los modelos de agregación:
  - El agregado constituye la unidad a efectos de distribución de los datos, en el caso de que la base de datos sea distribuida.
    - Para la distribución de agregados se usan principalmente técnicas de *hash*
    - Para aumentar la disponibilidad se usa replicación

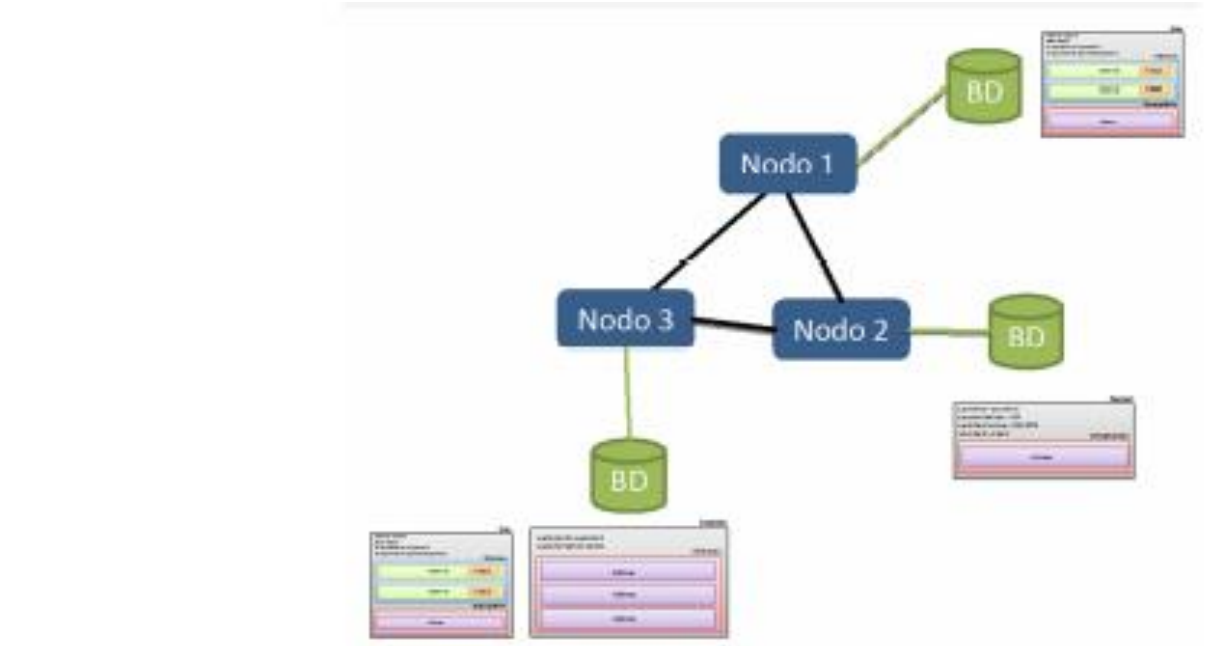


Imagen extraída de documentación de la UOC. Autores: E. Rodriguez, J. Conesa y P. Urbón

Modelos NoSQL

- Características de los modelos de agregación:
  - Los modelos de agregación presentan un esquema flexible (*schemaless*).
    - El esquema está definido de forma implícita en función de la estructura de los datos insertados o el esquema queda implícito en los programas que acceden a la base de datos.
    - La estructuración de agregados de un mismo tipo puede variar

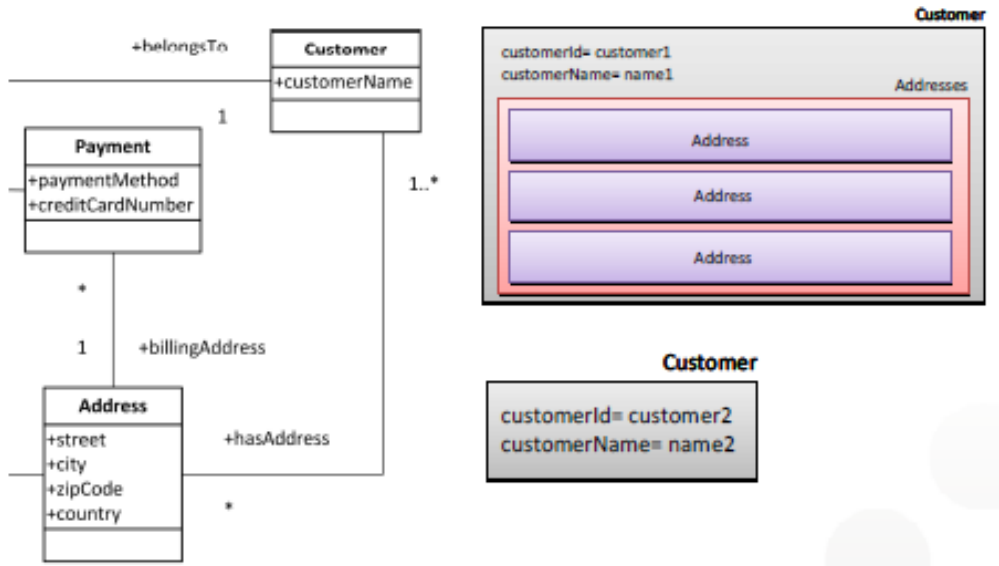


Imagen extraída de documentación de la UOC. Autores: E. Rodriguez, J. Conesa y P. Urbón

Modelos NoSQL

- Características de los modelos de agregación:
  - Los modelos de agregación presentan un esquema flexible (*schemaless*).
    - El esquema está definido de forma implícita en función de la estructura de los datos insertados o el esquema queda implícito en los programas que acceden a la base de datos.
    - La estructuración de agregados de un mismo tipo puede variar

Modelos NoSQL

- El diseño de agregados está guiado por las funcionalidades.
- Los modelos de agregación son una buena solución cuando:
  - Las funcionalidades que tiene que proveer la aplicación están claramente fijadas a priori y no se esperan que vayan a variar significativamente.
  - Las funcionalidades presentan poco solapamientos en cuanto a necesidades de datos.
  - No existen interrelaciones complejas en el dominio de aplicación que se desea representar.
  - Los datos están sujetos a pocos cambios, especialmente en aquellos datos que aparecen repetidos en diferentes agregados.

Modelos NoSQL

- Casos de uso:
  - Almacenamiento de datos de sesiones de usuarios
  - Perfiles de usuario
  - Aplicaciones de e-*commerce*
  - Gestión de contenidos
  - Web *analytics* y analíticas en tiempo real
  - Almacenamiento de series temporales

Lecturas

- A. Popescu. *NoSQL and Polyglot Persistence*. http://bit.ly/1ggTT4k
- P.J. Sadalage & M. Fowler (2013). *NoSQL Distilled. A brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education. http://bit.ly/1koKhBZ
- P. Atzeni, C.S. Jensen, G. Orsin & S.Ram (2013). "The relational model is dead, SQL is dead, and I don't feel so good myself", *SIGMOD Record* 42(2), pp. 64-68. http://www.sigmod.org/publications/sigmodrecord/1306/pdfs/11.reports.atzeni.pdf
- M. Stonebraker & U. Çetintemel (2005). "One Size fits all: An Idea whose time has come and gone", *Proceedings of the International Conference on Data Engineering*, pp 2-11. http://cs.brown.edu/people/ugur/fits_all.pdf
- Joe Celko's (2013). *Complete Guide to NoSQL*. Elsevier. http://www.sciencedirect.com/science/book/9780124071926
- E. A. Brewer (2012). "CAP Twelve Years Later: How the "Rules" Have Changed", *Computer* 45(2), pp 23-29. (http://www.infoq.com/articles/cap-twelve-years-laterhow- the-rules-have-changed)
- E. Redmond, J Wilson (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*. The Pragmatic Bookshelf.
- P.J. Sadalage & M. Fowler. (2013). *NoSQL Distilled. A brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education.