Bienvenidos a la presentación sobre los tipos de modelos de agregación. En esta presentación hablaremos sobre los distintos modelos de agregación en los que se basan buena parte de las bases de datos NoSQL.

Modelos de agregación

- Modelo de agregación clave-valor
- Modelo de agregación documental
- Modelo de agregación de columnas

EIMT.UOC.EDU

Una vez vistas las características generales asociadas a los modelos de agregación, es el momento de analizar las características específicas de cada tipo de modelo de agregación.

Empezaremos hablando del modelo clave-valor, que ve al agregado como una caja negra. Este modelo es el menos expresivo de todos los modelos de agregación.

Posteriormente profundizaremos en el modelo de agregación documental, donde los agregados son definidos mediante documentos, generalmente en formato XML, JSON o similar. En este modelo, los datos se pueden estructurar mínimamente en el contexto de cada documento, haciendo que partes del agregado sean accesibles individualmente.

Finalmente trataremos con el modelo agregado de columnas. Este modelo organiza los datos en primera instancia por filas y luego en columnas, permitiendo una visión matricial de los datos, donde la clave permite identificar los agregados y las características (o propiedades) del agregado se representan mediante columnas y/o agrupaciones de columnas.

Modelos de agregación

- Modelo de agregación clave-valor
- Modelo de agregación documental
- Modelo de agregación de columnas

EIMT.UOC.EDU

Comenzaremos hablando del modelo de agregación clave-valor.

Modelo de agregación clave-valor

- El modelo clave-valor tiene su origen en Berkeley DB (1986).
- Algunas bases de datos NoSQL que se basan en el modelo clave-valor son DynamoDB, Redis, Riak, Aerospike, Oracle Berkeley DB, entre otras.
- El agregado consiste en un par *(clave, valor)*.
- La clave se puede extraer del dominio de aplicación (p.e. nombres de usuario, direcciones de correo electrónico, números de la seguridad social, coordenadas cartesianas etc.)
- La base de datos ignora la estructura asociada al contenido del agregado (*valor*), es decir, el esquema está implícito y su interpretación pasa a ser responsabilidad de la aplicación.
- A pesar de ello, algunas bases de datos clave-valor (p.e. Riak) permiten estructurar un mínimo el contenido del agregado.

EIMT.UOC.EDU

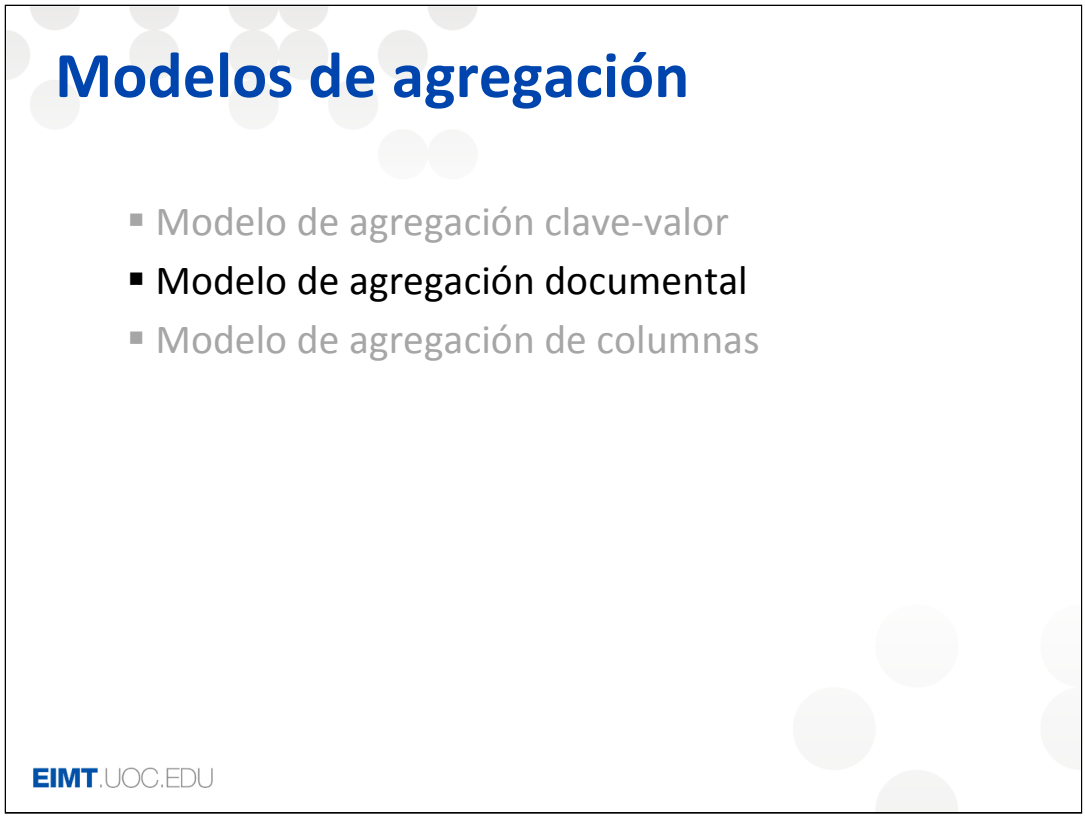
El modelo clave-valor tiene su origen en la base de datos BerkeleyDB que a día de hoy es propiedad de Oracle. Otros ejemplos de bases de datos NoSQL basadas en este modelo son DynamoDB (creado por Amazon), Redis y Riak.

Se trata del modelo más simple dentro de los modelos de agregación, dado que el agregado constituye un par *(clave, valor)*.

La clave de cada clase de agregado puede tener un significado dentro del dominio de interés a modelar. Éste sería el caso, por ejemplo, de claves como nombres de usuario, direcciones de *email*, coordenadas cartesianas para el almacenamiento de datos de geolocalización etc.

El sistema gestor de la base de datos desconoce la estructura interna asociada al agregado (el elemento *valor* del par *(clave, valor)*). En definitiva, el valor asociado al agregado es almacenado como un *blog* (*binary large object*) en la base de datos. Esto no significa necesariamente que el agregado no tenga estructura, sino que ésta sólo será comprendida por los programas que manipulan los agregados.

A pesar de ello, existen bases de datos clave-valor, como es el caso de Riak, que permiten definir una estructuración mínima del agregado y relaciones entre agregados. Relacionado con esto cabe destacar que, en NoSQL, las bases de datos suelen ser aproximaciones a los modelos de datos. Esto significa que tanto pueden no implementarlos en su totalidad, como pueden añadir funcionalidades extra. Este hecho puede provocar que algunas bases de datos NoSQL se puedan clasificar de diferentes maneras, según la fuente de información consultada.



El siguiente modelo de agregación es el denominado modelo documental u orientado a documentos. Este modelo es el que hemos usado en nuestro ejemplo motivador.

Modelo de agregación documental

- Internamente el modelo documental se puede ver como una extensión del modelo clave-valor.
- En el modelo documental los agregados tienen una estructura interna que recibe el nombre de documento.
- Esta estructura interna simplifica el desarrollo de aplicaciones, pero reduce la flexibilidad del modelo clave-valor.
- Los documentos se acceden mediante una clave única, o mediante atributos de los mismos. Esto permite, por ejemplo:
  - La recuperación de una parte del documento
  - La definición de índices

EIMT.UOC.EDU

Este modelo se considera un caso particular del modelo clave-valor.

A diferencia del modelo clave-valor, en el modelo documental los agregados (que reciben el nombre de documento) tienen una estructura interna.
Esta estructura interna simplifica el desarrollo de aplicaciones, pero reduce la flexibilidad del modelo clave-valor.

La estructuración interna puede ser aprovechada por el sistema gestor de la base de datos y por los lenguajes que ofrecen estas bases de datos. Así, por ejemplo, los documentos se pueden recuperar mediante su clave o mediante el valor que toman sus atributos. También es posible acceder a partes del documento y crear índices que ayuden a recuperar eficientemente los documentos almacenados en la base de datos. Los documentos se pueden agrupar en colecciones.

Aunque los documentos puedan tener una estructura interna, a diferencia del modelo relacional, no es necesario definirla de antemano, sino que será implícita y dependerá de cómo están estructurados los datos en los documentos. En consecuencia, distintos documentos que representen el mismo concepto del mundo real (por ejemplo, los datos de dos personas) pueden tener estructuras totalmente distintas o variaciones de la misma estructura.

Modelo de agregación documental

- La distribución de documentos entre los nodos del sistema distribuido se puede realizar en función del valor que toman atributos contenidos en la estructura interna del documento.
- La mayoría de bases de datos basadas en el modelo documental se caracterizan por almacenar documentos en formato JSON, pero también en XML, entre otros formatos.

EIMT.UOC.EDU

Como el resto de los modelos de agregación, el modelo documental es ampliamente utilizado en sistemas altamente distribuidos. El almacenamiento distribuido de documentos entre los diferentes nodos del sistema se puede realizar de dos formas:

1) A través de funciones de *hash*, tal y como se hace en los modelos clave-valor.

O bien,

1) En función de valor que toman uno o más atributos de la estructura interna del documento.

La mayoría de bases de datos basadas en el modelo documental se caracterizan por almacenar documentos en formato JSON, pero también se admiten otros formatos, como sería el caso de XML.

Modelo de agregación documental

- Algunas de las bases de datos NoSQL basadas en el modelo documental son MongoDB, CouchDB, Marklogic y RethinkDB.
- De entre las especializadas en XML podemos encontrar a BaseX, eXist y Apache Sedna, entre otras.
- Algunas bases de datos relacionales también permiten almacenar y manipular documentos tales como PostgreSQL, DB2, MS SQL Server u Oracle.

EIMT.UOC.EDU

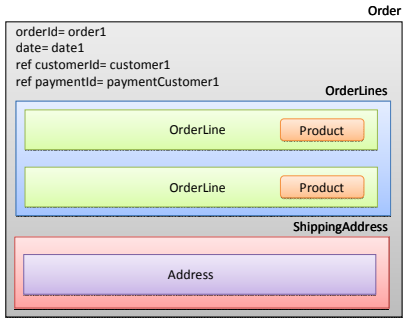
Algunas de las bases de datos NoSQL más conocidas que implementan el modelo documental son MongoDB, CouchDB, Marklogic y RethinkDB.

También encontramos un conjunto de bases de datos especializadas en el formato XML como BaseX, eXist o Apache Sedna.

Finalmente, también existen bases de datos relacionales que mediante extensiones permiten trabajar con documentos, en general, definidos en formato XML. Entre dichas bases de datos encontramos PostgreSQL, DB2, SQL Server y Oracle. No obstante, cabe tener en cuenta que estas bases de datos continúan siendo relacionales, con todas las ventajas e inconvenientes que eso puede implicar.

Modelo de agregación documental

El carro de la compra: ejemplo de documento




```
// order
{
   "orderId": 100,
   "date": "01/03/2014",
   "customerId: 1000,
   "paymentId": 10,
   "orderLine":[
      {
         "productId": 27,
         "productName": "Le pere Goriot",
         "numberOfUnits": 1,
         "price": 18.50
      }
   ],
   "shippingAddress":[
      {
         "street": "Champs Élysées 156",
         "city": "Paris",
         "zipCode": "75008",
         "country": "France"
      }
   ]
}
```

EIMT.UOC.EDU

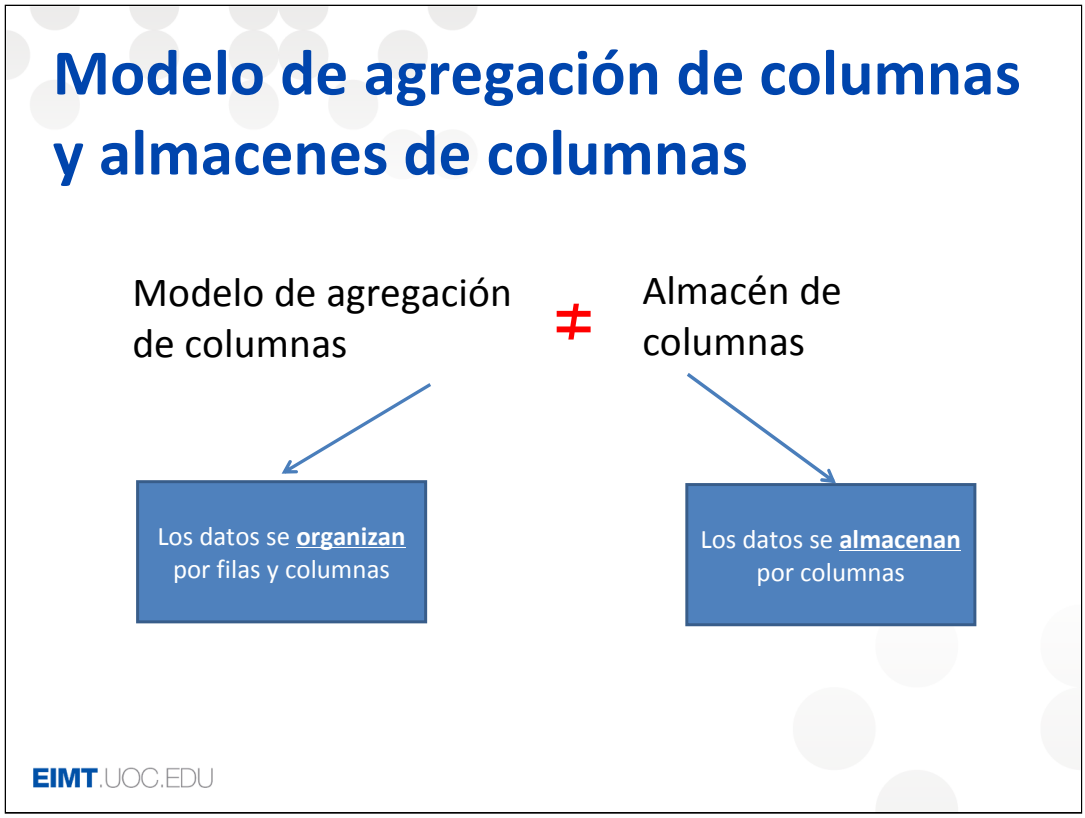
Por último, y volviendo a nuestro ejemplo del carrito de la compra, en la parte derecha de esta transparencia se muestra un documento en formato JSON que se corresponde con un pedido concreto. En dicho pedido, podemos apreciar que existen dos agregaciones de datos. Una para las líneas de pedido (o conjunto de productos adquiridos), y otra para los datos correspondientes a la dirección de envío del pedido.

Modelos de agregación

- Modelo de agregación clave-valor
- Modelo de agregación documental
- Modelo de agregación de columnas

EIMT.UOC.EDU

Una vez vistos los modelos clave-valor y documental vamos a introducir el modelo de agregación de columnas.



No empezaremos hablando de qué son los modelos de agregación de columnas, sino de lo qué no son.

En las bases de datos relacionales clásicas los datos se organizan y se almacenan por filas. Decimos que se organizan por filas porque los datos se agrupan en forma de filas para representar, para cada fila, un objeto del mundo real. Decimos que se almacenan por filas porque cuando se guardan estos datos en disco, los datos de cada fila se guardan secuencialmente. Ello implica que recuperar un conjunto de filas pueda ser muy eficiente, pero recuperar los datos de un atributo de múltiples filas pueda ser muy costoso.

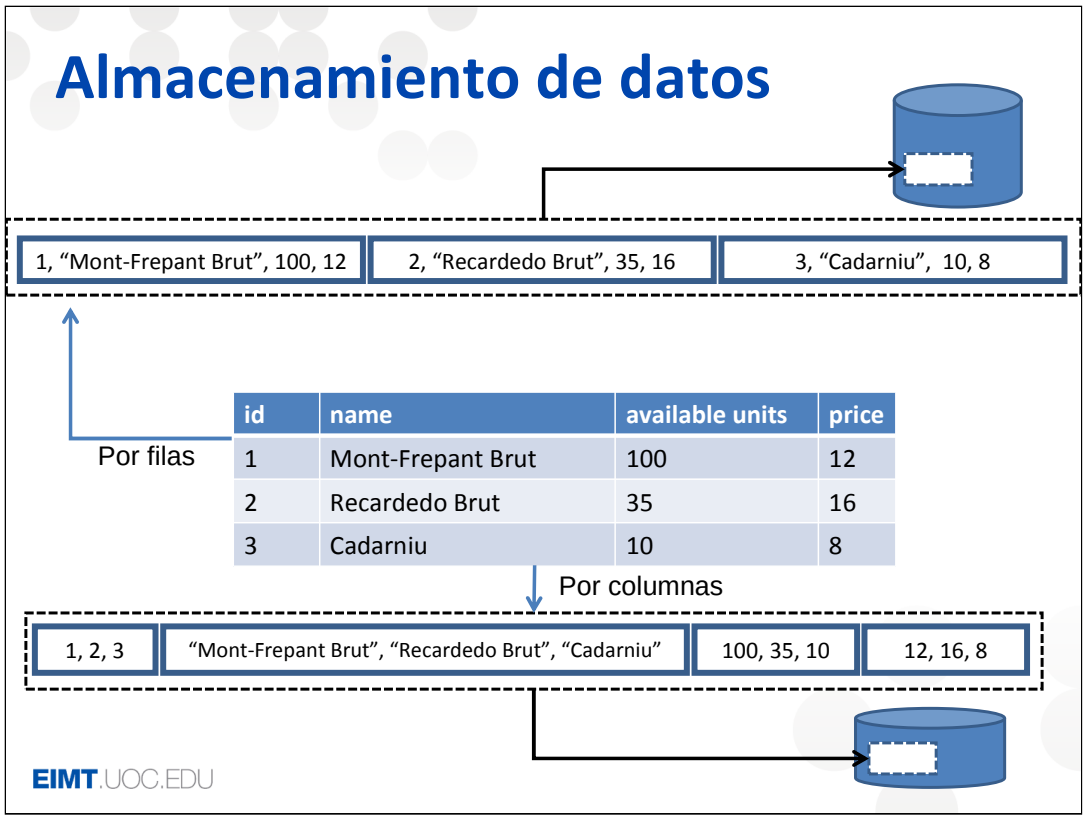
Con los modelos de agregación de columnas puede existir una confusión terminológica importante. Las siguientes explicaciones están orientadas facilitar la distinción de las bases de datos NoSQL de columnas con los denominados almacenes de columnas.

Existe un tipo de bases de datos relacionales que almacenan los datos por columnas. Estos tipos de base de datos se suelen denominar almacenes de columnas (o *column stores* en inglés). Estas bases de datos no siguen el modelo agregado de columnas y aún menos pueden considerarse bases de datos NoSQL.

Por otro lado, las bases de datos NoSQL que siguen el modelo de agregación de columnas organizan los datos por filas que se guardan en tablas. Cada fila constituye un agregado e incluye columnas y/o conjuntos de columnas, donde cada conjunto de columnas tiene un significado concreto. Estas bases de datos NoSQL pueden también almacenar los datos por columnas, aunque no es su característica principal. Por lo tanto, pueden diferir de la forma en que se almacenan los datos en los almacenes de columnas.

Por todo ello, es importante destacar las diferencias entre los modelos de columnas NoSQL y los almacenes de columnas.

Los almacenes de columnas tienen unas características que los hacen especialmente útiles para un *data warehouse*. Por este motivo, en las siguientes transparencias, entraremos en detalle en ambos modelos, con el objetivo de facilitar al oyente su comprensión y sus oportunidades de aplicación.



Antes de entrar en detalle en los almacenes de columnas, es importante mostrar la diferencia entre almacenar los datos por filas o por columnas. Vamos a verlo mediante un ejemplo, simplificando mucho el proceso de almacenaje para facilitar su comprensión. El almacenamiento real de los datos difiere ligeramente de lo explicado aquí.

Supongamos que tenemos una base de datos relacional con datos de un almacén de vinos. La relación mostrada contiene información de los cavas que hay en el almacén, en particular de su código, nombre, unidades disponibles y precio.

En un sistema relacional clásico los datos se almacenarían en disco por filas. Es decir, los datos de cada fila se escribirían de forma atómica en el disco y las filas se almacenarían consecutivamente. Esta forma de almacenamiento puede ser muy conveniente si las consultas a la base de datos implican a todos los atributos de la relación. Por otro lado, añadir nuevas filas en la relación es relativamente sencillo, ya que se pueden añadir de forma consecutiva a las que ya existen. Sin embargo, añadir nuevas columnas es bastante más complejo.

En un sistema de almacenamiento orientado a columnas, la filosofía es totalmente distinta. Los datos se almacenan de forma consecutiva por columnas. Es decir, en el ejemplo, primero se almacenarían todos los valores de la columna *id*. Secuencialmente a estos, se almacenarían todos los valores del atributo *name*, y así sucesivamente.

Hay diversas ventajas que este sistema de almacenamiento puede ofrecer:

- Por un lado, se puede optimizar el espacio utilizado para almacenar los datos. Los datos de cada columna se almacenan juntos. Como los datos de cada columna tienen el mismo tipo, su entropía es muy baja y permiten un factor de compresión elevado. Además, con respecto un almacenamiento orientado a filas, existe un ahorro en términos datos de control. Este sería el caso de aquellos que se usan para saber, para cada fila almacenada, dónde comienza cada columna.
- Las consultas sobre una selección de columnas (o una columna) de la tabla (o relación), para diversas filas, serán más eficientes, ya que con una sola lectura podríamos recuperar todos los valores de una columna.

Como principal problema, tenemos que las inserciones de datos son más costosas que en el almacenamiento por filas.

Teniendo en cuenta las ventajas e inconvenientes de este tipo de almacenaje, es razonable pensar que puede ser muy conveniente en *data warehouses*. Esto es así porque las consultas sobre los mismos, normalmente, implican un gran número de filas, se consultan datos de pocas columnas y el número de operaciones de inserción y modificación son reducidas y están controladas.

Almacenes de columnas

- El almacenamiento de datos por filas puede ser poco eficiente en algunos casos.
- El almacenamiento de datos por columnas es conveniente cuando:
  - Hay que almacenar gran cantidad de datos
  - Las operaciones se realizan a nivel de columna en vez de a nivel de fila.
  - *Data warehouses*
- Este tipo de bases de datos **NO** tienen nada que ver con NoSQL.
- Algunos ejemplos son MonetDB, LucidDB e Infobright.

EIMT.UOC.EDU

Resumiendo, el almacenamiento por filas puede no ser lo más adecuado en algunos casos. Almacenar los datos por columnas permite reducir el espacio de almacenaje y optimizar consultas sobre gran cantidad de filas a nivel de columna o columnas. Por estos motivos, los almacenes de columnas son muy recomendables en *data warehouses*. De hecho, diversos estudios indican mejoras de optimización por encima de 1:10 en este tipo de sistemas.

Este tipo de bases de datos NO tienen nada que ver con NoSQL, aunque las bases de datos de columnas de NoSQL puedan utilizar sistemas de almacenamiento parecido.

Hay una gran oferta de sistemas de este tipo en el mercado. Algunos ejemplos de almacenes de columnas son MonetDB, LucidDB e Infobright.

Modelo de agregación de columnas

- Su precursor es el modelo de datos BigTable de Google.
- Algunas bases de datos NoSQL basadas en este modelo son Cassandra, Hbase y Amazon SimpleDB.
- El modelo de datos puede verse como una matriz:
  - Las filas representan agregaciones de datos y se acceden mediante su clave.
  - Las columnas representan atributos de la agregación y se representan mediante una tripleta:

    <nombre, valor, *timestamp*>
  - Las columnas pueden agruparse en familias de columnas.
- Los datos se almacenan agrupados por columnas o por combinaciones de ellas (familias de columnas).

EIMT.UOC.EDU

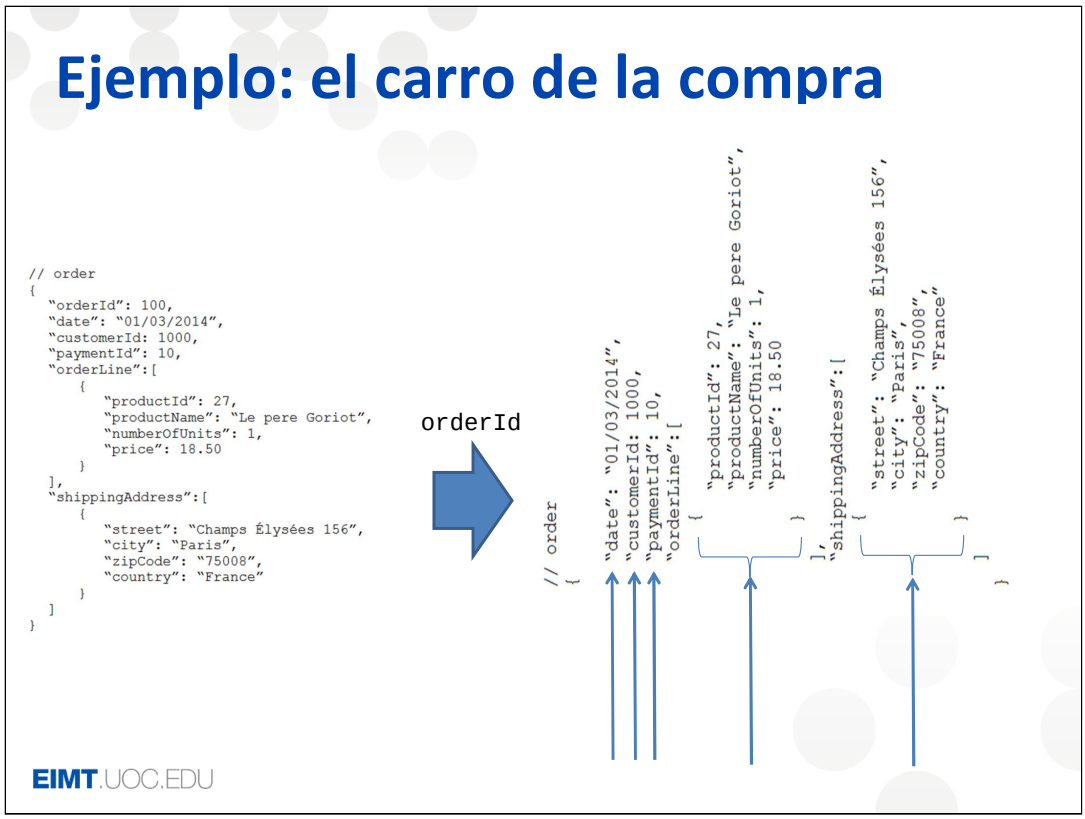
Los modelos de agregación de columnas están basados en el modelo de datos BigTable de Google. A pesar de ello, hoy en día, algunas bases de datos NoSQL adheridas a este modelo han evolucionado y pueden presentar diferencias significativas con el modelo de datos motivador. Algunas de dichas bases de datos son Cassandra, HBase y Amazon SimpleDB. Estas bases de datos también reciben el nombre de almacenes para grandes datos (*big data stores*) y almacenes de registros extensibles (*extensible record stores*).

Conceptualmente, podemos ver este modelo como un modelo bidimensional (una matriz), donde cada fila de la tabla representa un agregado y es accesible a partir de una clave. Hasta ahora no hay ninguna novedad respecto a los modelos de agregación vistos anteriormente. No obstante, en este modelo, los datos de los agregados (es decir, cada una de las filas) se organizan en columnas. Un agregado es un conjunto de columnas, donde cada columna está formada por una tripleta compuesta por el nombre de la columna, el valor de la columna y una marca de tiempo (*timestamp*) que indica cuándo se añadió la columna en la base de datos.

Un conjunto de columnas puede agruparse en una nueva estructura, llamada normalmente familia de columnas. Una familia de columnas tiene un nombre y una semántica muy definida. Habitualmente, dentro de una agregación, una familia de columnas representa un concepto de la agregación. Por ejemplo, entendiendo un estudiante como un agregado (fila), tres de sus familias de columnas podrían ser, respectivamente, sus atributos personales, su domicilio y los estudios previos realizados por el estudiante.

A partir de aquí, distintas bases de datos NoSQL utilizan diferentes variantes de esta representación.

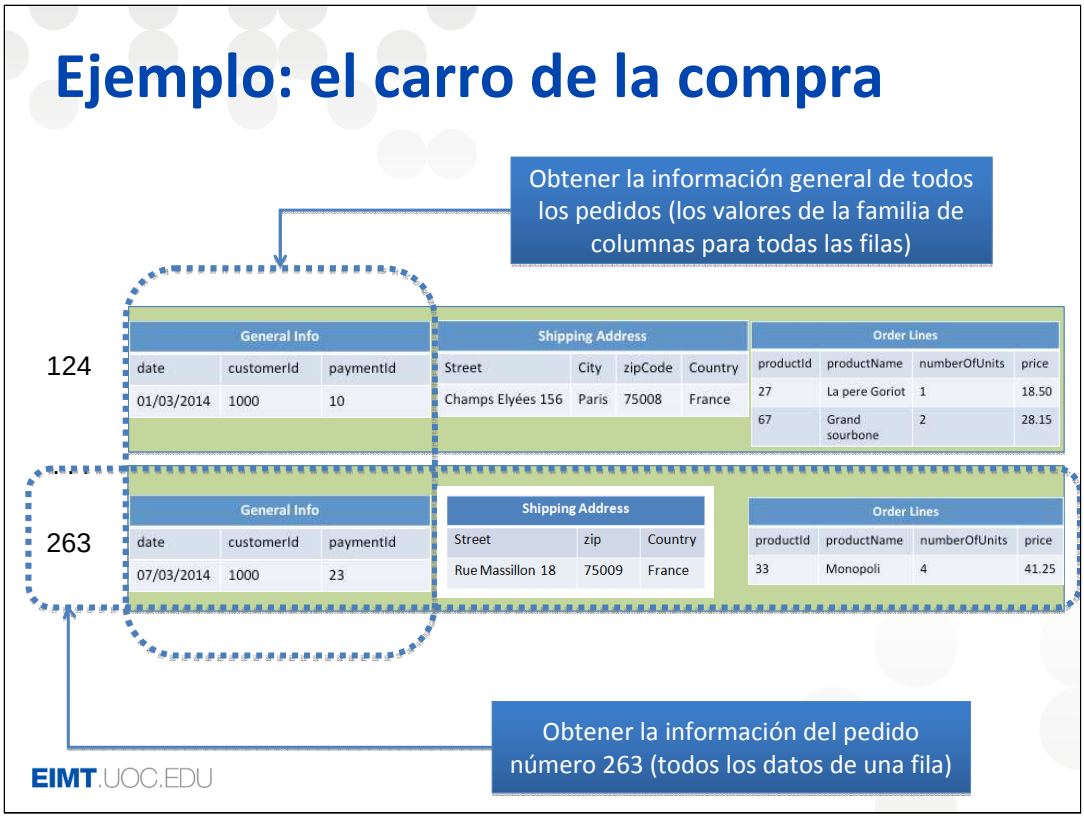
Un resultado relevante de utilizar este modelo es la facilidad para acceder a un subconjunto de los atributos de un agregado, recordemos que los datos están organizados por columnas. Esto permite, por ejemplo, recuperar datos eficientemente, sólo con los atributos relevantes en cada momento. Otra característica de este modelo es que, en comparación con el modelo relacional, no todas la filas de una misma tabla deben tener el mismo conjunto de columnas. El modelo por columnas puede ser un poco confuso al principio, pero es un modelo muy versátil y con muchas posibilidades.



Vamos a recuperar el ejemplo del carro de la compra que hemos utilizado en el modelo de agregación de documentos, y vamos a ver cómo podríamos almacenar los pedidos usando una base de datos NoSQL que siguiese un modelo de agregación de columnas.

Una manera fácil de entender un agregado de un modelo documental como un conjunto de columnas es girar el agregado 90 grados. Una vez hecho esto, podemos ver la clave del agregado como el identificador de fila (*orderId* en nuestro caso) y los otros atributos como columnas. Las agregaciones de atributos, en el ejemplo *orderLine* y *shippingAdress*, pueden verse como familias de columnas.

Suponiendo que creamos una nueva familia de columnas para almacenar la información general de los pedidos, vamos a ver cómo se organizarían los datos del carrito de la compra en un modelo de agregación de columnas.



Como podemos ver, los agregados representan objetos del mundo real a almacenar, en este caso, pedidos concretos. En el ejemplo hay dos agregados, que responden a dos pedidos de un mismo cliente. Los agregados se identifican mediante el identificador de pedido (*orderId*), que es la clave del agregado.

En los agregados, las columnas representan atributos del agregado, es decir, atributos del pedido, como pueden ser la fecha, el cliente, o el domicilio donde se ha enviado. Podemos ver que el número de columnas de los agregados puede ser diferente. El primer agregado tiene una columna ciudad (*city*), mientras que el segundo agregado no la tiene. Otra particularidad es que las columnas pueden tener distinto número de valores en distintos agregados, tal y como podemos en el primer agregado, que tiene 2 valores para la columna *productId*, mientras que el segundo agregado con sólo tiene una.

Las familias de columnas agrupan distintas columnas para representar una parte del agregado con semántica propia. Cada familia de columnas tiene un nombre que la identifica. En el ejemplo, podemos ver como hay tres familias de columnas, una para representar la información general del pedido (compuesta por las columnas fecha, identificador de cliente e identificador de pago), otra para representar la dirección de entrega (compuesta por las columnas calle, ciudad, código postal y país) y la última para representar las líneas de pedido, compuestas por los identificadores de producto, los nombres de productos, el número de unidades y el precio final. Notad también que algunas familias de columnas tienen sólo un valor (información general y dirección de entrega), mientras que otras pueden almacenar múltiples valores (líneas de producto).

En un modelo de agregación de columnas se puede acceder a una fila (o agregado) concreto a partir de su clave. Es decir, sobre nuestro ejemplo, sería posible obtener la información de un pedido concreto (el agregado) a partir del identificador del pedido (que es la clave del agregado). Por ejemplo, se podría acceder el agregado del segundo pedido a partir de su identificador, con valor 263.

Por otro lado, también es posible obtener información sobre una familia de columnas para todos los agregados. Por ejemplo, podríamos obtener la información general de todos los pedidos. Es decir, los valores de la familia de columnas *General Info* para todas las filas (recordad que cada fila es un agregado).

Evidentemente, también puede obtenerse información mediante la combinación de las dos operaciones anteriores.

Conceptualmente, también podríamos ver este tipo de organización como un modelo donde cada agregado está compuesto por un conjunto de tablas relacionales con estructura variable.

Modelos de agregación

- Modelo de agregación clave-valor
- Modelo de agregación documental
- Modelo de agregación de columnas

EIMT.UOC.EDU

Aquí finaliza la última presentación que hemos dedicado a los modelos de agregación.

En esta presentación hemos introducido los principales tipos de modelos de agregación usados en NoSQL: el clave-valor, el documental y el de columnas. Asimismo hemos examinado algunos aspectos a tener en cuenta en cada uno de estos modelos, hemos enumerado algunas bases de datos NoSQL que los implementan y hemos indicado en qué casos puede ser conveniente su uso.

Referencias

R. Catell (2010). "Scalable SQL and NoSQL Data Stores". *SIGMOD Record* 39(4), pp 12-27. (http://dl.acm.org/citation.cfm?id=1978919).

F. Chang et al. (2006). Bigtable: A Distributed Storage System for Structured Data, "Proceedings of the 7th Symposium on Operating Systems Design and Implementation". (http://static.googleusercontent.com/media/research.google.com/es//archive/bigtable-osdi06.pdf)

C. Coronel, S. Morris & P. Rob (2013). *Database Systems: Design, Implementation and Management 10e*. Course Technology, Cengage Learning.

G. DeCandia et al. (2007). Dynamo: Amazon's Highly Available Key-value Store, "Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles", pp 205-220. (http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf).

E. Redmond, J Wilson (2012). *Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement*, The Pragmatic Bookshelf.

I. Robinson, J. Webber & E. Eifren (2013). *Graph Databases,* O'Reilly.

P.J. Sadalage & M. Fowler. (2013). *NoSQL Distilled. A brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education. (http://bit.ly/1koKhBZ).

EIMT.UOC.EDU

Esperamos que hayáis disfrutado y aprendido con este vídeo. A continuación encontraréis algunas referencias que os permitirán profundizar más en los temas tratados.

Que tengáis un buen día.