Bienvenidos a la primera presentación sobre modelos de agregación. Los modelos de agregación agrupan una serie de modelos de datos que subyacen en diversas bases de datos NoSQL.

Modelos de agregación

- Introducción
- Ejemplo motivador

EIMT.UOC.EDU

Esta presentación se divide en dos secciones.

En la primera introduciremos algunos términos de referencia, mientras que en la segunda sección presentaremos un ejemplo para ilustrar los principales beneficios e inconvenientes asociados a los modelos de agregación, tomando como marco de referencia para la comparación el modelo relacional.

Modelos de agregación

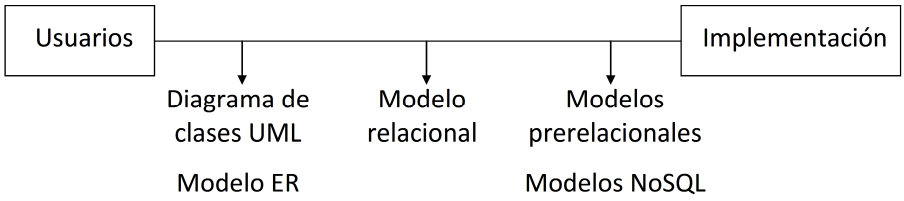
- Introducción
- Ejemplo motivador

EIMT.UOC.EDU

En primer lugar vamos a definir qué se entiende por modelo desde la perspectiva de la ingeniería informática.

Modelo de datos

- Un modelo conceptual de datos es una descripción de la parte del mundo real que nos interesa conceptualizar. Incluye clases de objetos (o entidades), junto con sus propiedades (o atributos) y relaciones entre clases (por ejemplo, relaciones de herencia, asociaciones etc.).
- 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, e incluye los siguientes elementos:
  - Estructuras de datos para construir la base de datos
  - Operaciones para manipular y consultar los datos
  - Reglas de integridad que deben cumplir los datos



EIMT.UOC.EDU

Un modelo es un esquema que nos permite analizar, conceptualizar y representar una parte de la realidad que nos interesa comprender con el objetivo de extraer conocimiento. Los modelos se utilizan en todas las disciplinas científicas y en ingeniería, y por lo tanto, también en informática.

Desde una perspectiva de ingeniería del software, un modelo conceptual de datos (o simplemente, un modelo conceptual) constituye una descripción de una parte del mundo real (o dominio de aplicación) que nos interesa analizar y conceptualizar.

Para construir modelos conceptuales se suelen usar modelos que se basan en lenguajes gráficos, dado que se orientan a personas. Su construcción surge como consecuencia de un proceso de abstracción a partir de la observación del mundo real que nos interesa modelar. Como ejemplos de modelos que nos permiten realizar esta tarea, tenemos los diagramas de clases de UML y el modelo *Entity-Relationship* (o modelo entidad-interrelación, también conocido simplemente como modelo ER). UML se basa en el paradigma de orientación a objetos y es más expresivo que el modelo ER. Con los diagramas de clases de UML podemos representar clases de objetos del mundo real que tienen propiedades comunes, y diferentes tipos de relaciones entre clases (por ejemplo, relaciones de herencia, asociaciones y diversos tipos de relaciones parte-todo).

Una característica fundamental de los modelos conceptuales es que se aíslan de posibles tecnologías de representación. A pesar de ello, un modelo conceptual tiene que ser finalmente implementado, con el objetivo de que pueda ser utilizado por nuestros programas de aplicación.

Cuando la tecnología de implementación es una base de datos, el modelo conceptual se tiene que transformar de acuerdo al modelo de datos en el que se basa la base de datos elegida.

Desde la perspectiva de bases de datos, un modelo de datos constituye el conjunto de elementos que nos proporciona el sistema gestor de la base de datos para implementar (o representar) nuestro modelo conceptual, e incluye (al menos desde una perspectiva clásica) los siguientes elementos: 1) Estructuras de datos; 2) Operaciones para consultar y modificar los datos; y 3) Mecanismos para definir restricciones de integridad que los datos deben cumplir.

A modo de ejemplo, el modelo relacional (que debido a su simplicidad es relativamente fácil de ser comprendido por las personas) incluye la relación como elemento de estructuración, lenguajes como el álgebra relacional y SQL para consultar y modificar los datos, y la posibilidad de definir restricciones de integridad (por ejemplo, claves primarias y foráneas).

El modelo relacional no es el único modelo de datos, también existen los modelos prerrelacionales (basados en los modelos jerárquicos y en red) y los modelos de datos NoSQL. Una característica común a todos estos modelos es que están más próximos (con respecto al modelo relacional) al mundo de las representaciones informáticas.

Modelo de datos

- Bajo el paraguas asociado a las tecnologías NoSQL subyacen 4 modelos de datos principales:
  - Modelo clave-valor
  - Modelo documental (u orientados a documentos)
  - Modelo orientado a columnas

  } Modelos de agregación
  - Modelo de grafos (u orientados a grafos)
- Los modelos de agregación se basan en la noción de agregado.
- Un agregado es una colección de objetos (o entidades) relacionados que deseamos tratar como una unidad independiente (desde un punto de vista semántico) a efectos de:
  - Acceso y manipulación
  - Consistencia y control de concurrencia
  - Distribución de datos

EIMT.UOC.EDU

Bajo el paraguas NoSQL subyacen principalmente 2 familias de modelos de datos, el modelo de grafos y los modelos de agregación. El modelo de grafos se orienta a representar dominios de aplicación donde se establecen múltiples y complejas interrelaciones entre los objetos del mundo real que deseamos representar. Por su parte, los modelos de agregación incluyen 3 modelos: el clave-valor, el documental, y el orientado a columnas.

En consecuencia, tenemos 4 modelos de datos que se acostumbran a usar para clasificar las diferentes implementaciones de bases de datos NoSQL disponibles en el mercado.

Los modelos de agregación se basan en el concepto de agregado. Un agregado es una colección de objetos del mundo real relacionados que deseamos tratar como una unidad independiente (o atómica) desde un punto de vista de significado a efectos de:

1) Acceso y manipulación: la unidad mínima de intercambio de datos entre nuestros programas y el sistema gestor de la base de datos (o simplemente la base de datos) es el agregado.
2) Consistencia y control de concurrencia: asegurar la definitividad de los cambios sobre el agregado y el mantenimiento de su integridad se circunscriben al ámbito del agregado. Si es necesario mantener restricciones de integridad entre diferentes agregados, la responsabilidad de asegurar la consistencia será de los programas y no de la base de datos.
3) Si la base de datos está distribuida, el agregado se usa como unidad de distribución de los datos.

Si consideramos los tres elementos que desde un punto de vista teórico todo modelo de datos tiene que proveer, es importante destacar que cada modelo de agregación ofrece estructuras (en algunos casos, mínimas) para almacenar los datos. En relación a las operaciones para manipular y consultar los datos, cada fabricante ofrece, en general, su propio lenguaje (recordemos la falta de estandarización de las tecnologías NoSQL). Finalmente, y tal como acabamos de comentar, los modelos de agregación delegan el mantenimiento de restricciones de integridad principalmente en los programas que acceden a la base de datos.

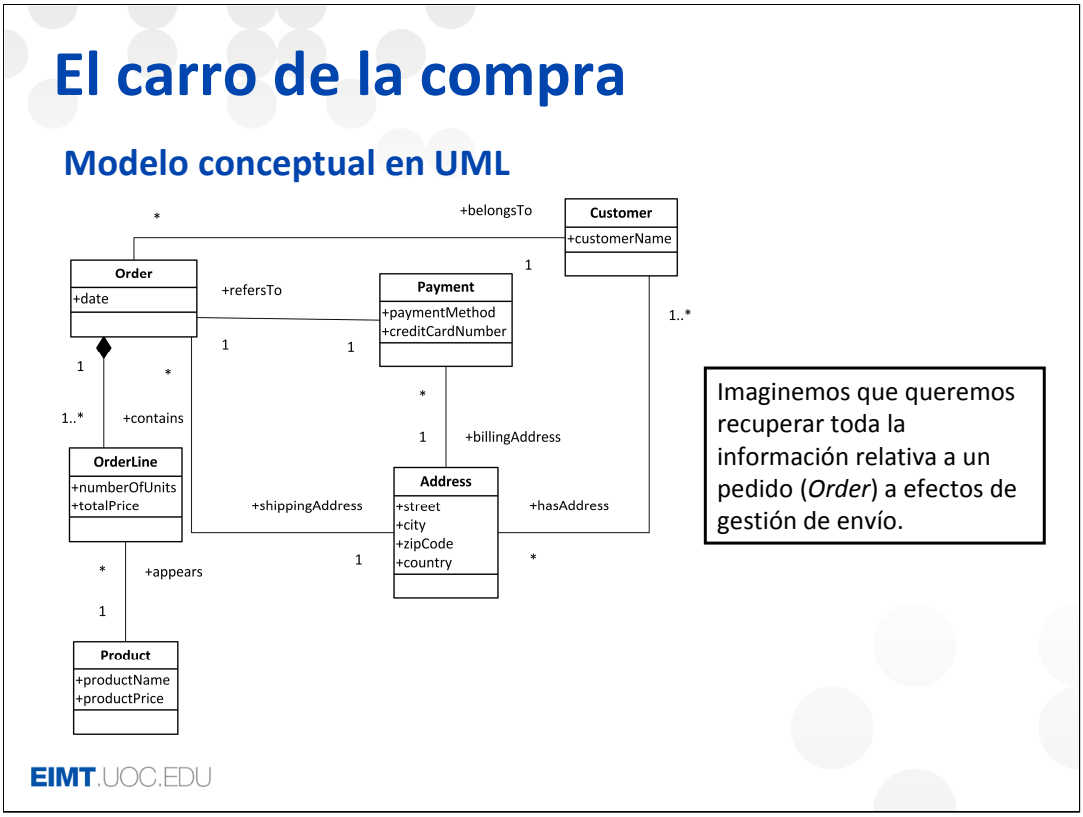
Modelos de agregación

- Introducción
- Ejemplo motivador

EIMT.UOC.EDU

En esta sección mostraremos un ejemplo que nos servirá como base para ilustrar los beneficios e inconvenientes asociados a los modelos de agregación, tomando como marco de referencia para la comparación el modelo relacional.

Es importante destacar que la discusión toma como base el modelo documental.



El ejemplo que os proponemos constituye una simplificación del carrito de la compra que utilizamos en aplicaciones de *e-commerce*.

El modelo conceptual incluye datos de algunas de las clases de objetos que intervienen en el dominio a modelar, sus propiedades y también las relaciones que existen entre dichas clases.

En primer lugar tenemos clientes (*Customer*). De los clientes podemos tener registradas entre 0 y múltiples direcciones postales (*Address*). Los clientes también pueden haber efectuado entre 0 y múltiples pedidos (*Order*). De todos los clientes guardamos su nombre. Que un cliente no tenga pedidos (o dirección postal) se puede relacionar, por ejemplo, con el hecho de que el cliente alguna vez se ha registrado en el sistema pero que, por los motivos que sea, no ha completado ningún proceso de compra. Las direcciones quedan caracterizadas por la calle, ciudad, código postal y país. No se guardan direcciones que no pertenezcan a ningún cliente, y podemos tener clientes que comparten domicilio.

Un pedido se realiza en una fecha. Los pedidos tienen una dirección postal de envío y datos asociados para efectuar el pago (*Payment*). A su vez, cada pago tiene una dirección asociada para el envío de la factura y datos sobre cómo se efectuará el pago (tipo y número de tarjeta).

Cada pedido es un objeto compuesto que incluye, al menos, una línea de pedido (*OrderLine*). El hecho que sea un objeto compuesto se indica mediante el diamante negro que en UML representa una relación de composición. Ésta es un tipo de relación parte-todo, cuya semántica implica que las líneas del pedido no tienen sentido sin el objeto en el que se agregan (el pedido). En otras palabras, cada línea de pedido sólo puede formar parte de un pedido, y si el pedido se elimina, también se eliminarán sus líneas. Cada línea de pedido hace referencia a un producto (*Product*). También incluye cuántas unidades se adquieren del producto y el precio total. De los productos se guarda su nombre y precio unitario. Hay productos que no aparecen en ninguna línea de pedido (y en consecuencia tampoco en ningún pedido), por ejemplo, porque o no se han sido adquiridos por nadie o porque todavía no se han puesto en oferta.

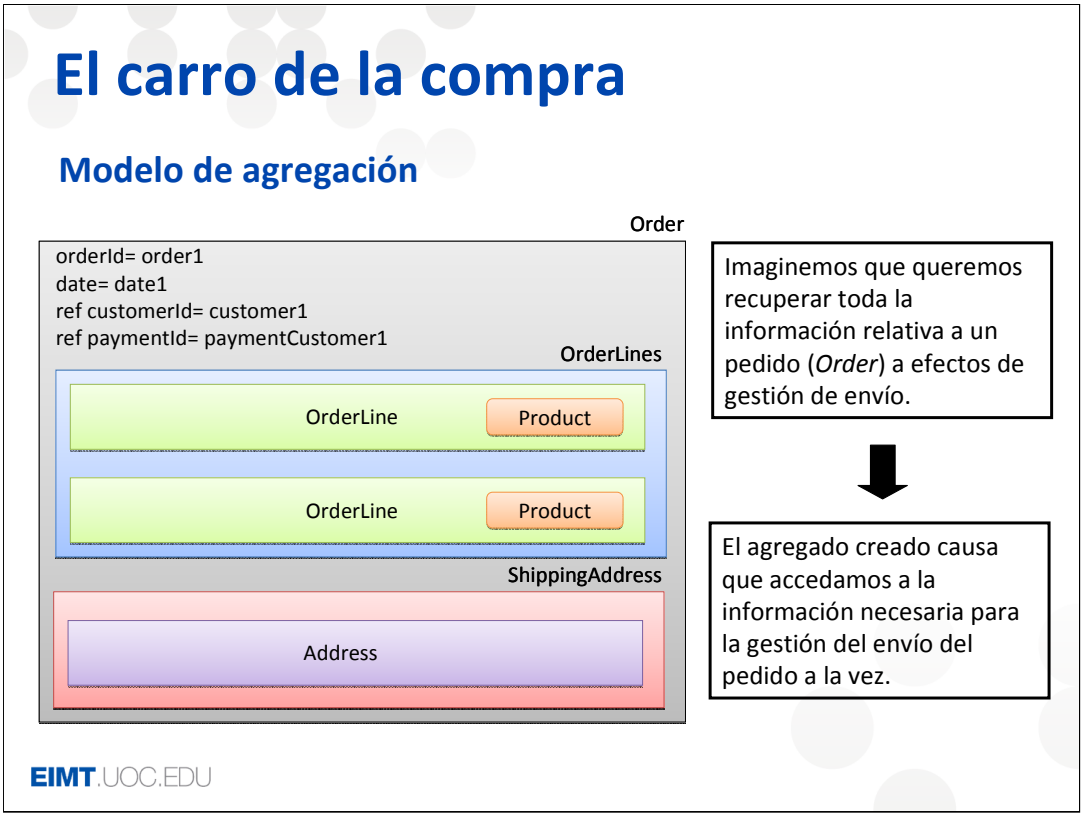
Imaginemos que sobre este modelo conceptual queremos conocer la información relativa a un pedido concreto a efectos de gestionar su envío. Esta información será más fácil o difícil de recuperar en función del tipo de base de datos elegida para implementar este modelo conceptual.

Supongamos que la base de datos elegida se basa en el modelo relacional. En este caso, la base de datos incluirá diversas relaciones, tal y como se muestra en esta transparencia.

Las claves primarias de cada relación están subrayadas, y también se indican (si existen) las claves foráneas y alternativas. Para las claves primarias se han añadido atributos en las relaciones que no existían en modelo conceptual. Fijémonos que la relación que representa los pedidos (*Order*) no se diferencia de otras relaciones, a pesar de que se trata de un objeto compuesto. Esto es una muestra de la falta de expresividad achacada al modelo relacional. A pesar de ello, mediante la definición de efectos de borrado en cascada para la clave foránea entre pedidos y líneas de pedido, sería posible eliminar automáticamente las líneas de pedido en el caso que el pedido al que estuviesen asociadas se borrase.

Para obtener la información de un pedido concreto a efectos de gestionar su envío, será necesario realizar la combinación de las relaciones pedido, línea de pedido, producto y dirección. Esto es así porque el modelo relacional se basa en la normalización de los datos, es decir, se basa en el principio de que cada relación tiene que representar un único concepto del mundo real con el fin de evitar redundancias en los datos.

Es importante destacar que las operaciones de combinación pueden resultar costosas y más en el caso de que la base de datos se encuentre distribuida.



Imaginemos ahora que elegimos una base de datos basada en un modelo de agregación (en concreto un modelo documental).

En este caso, para recuperar la información requerida, sería posible diseñar un agregado (o documento en términos del modelo documental) que incorpore todos los datos necesarios.

Este agregado (que representa el objeto compuesto pedido) incluye los atributos propios del pedido (su identificador y fecha en la que se realiza) y las referencias a los agregados que representan los datos de pago y del cliente que efectúa el pedido (dichas referencias nos permitirían navegar a dichos agregados). Adicionalmente, cada pedido agrega toda la información sobre las líneas de pedido (y los productos a los que hacen referencia), así como la dirección postal de envío.

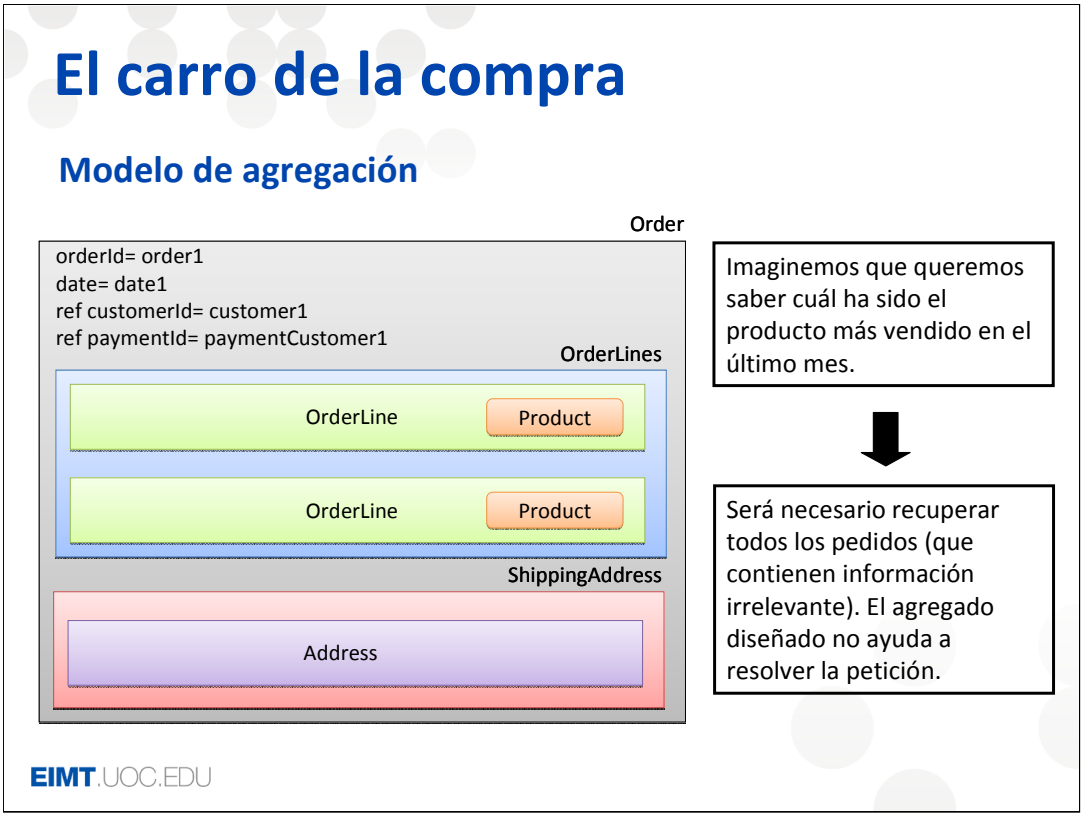
Dado que el agregado es la unidad mínima de acceso y manipulación, obtendríamos toda la información deseada de forma eficiente mediante una sola operación. Con respecto a una base de datos relacional, nos estamos ahorrando las operaciones de combinación. Esto es debido al hecho que un modelo de agregación se basa en la aplicación de técnicas de desnormalización.

Si la base de datos estuviese distribuida, el agregado se habría utilizado como unidad de distribución de los datos, garantizando de esta manera que estaría almacenado de manera conjunta en alguna de las bases de datos que conforman el sistema distribuido.

Supongamos ahora que deseamos saber cuál ha sido el producto más vendido en el último mes.

Asumiendo que nuestra base de datos es relacional, para recuperar la información deseada, sería necesario realizar la combinación de las relaciones pedido y línea de pedido. Dependiendo de los datos que se deseen recuperar de los productos, podría ser necesario efectuar una operación de combinación adicional sobre la relación de productos.

Con respecto a la eficiencia en la recuperación de los datos se aplicarían los mismos razonamientos explicados sobre la anterior consulta de ejemplo.

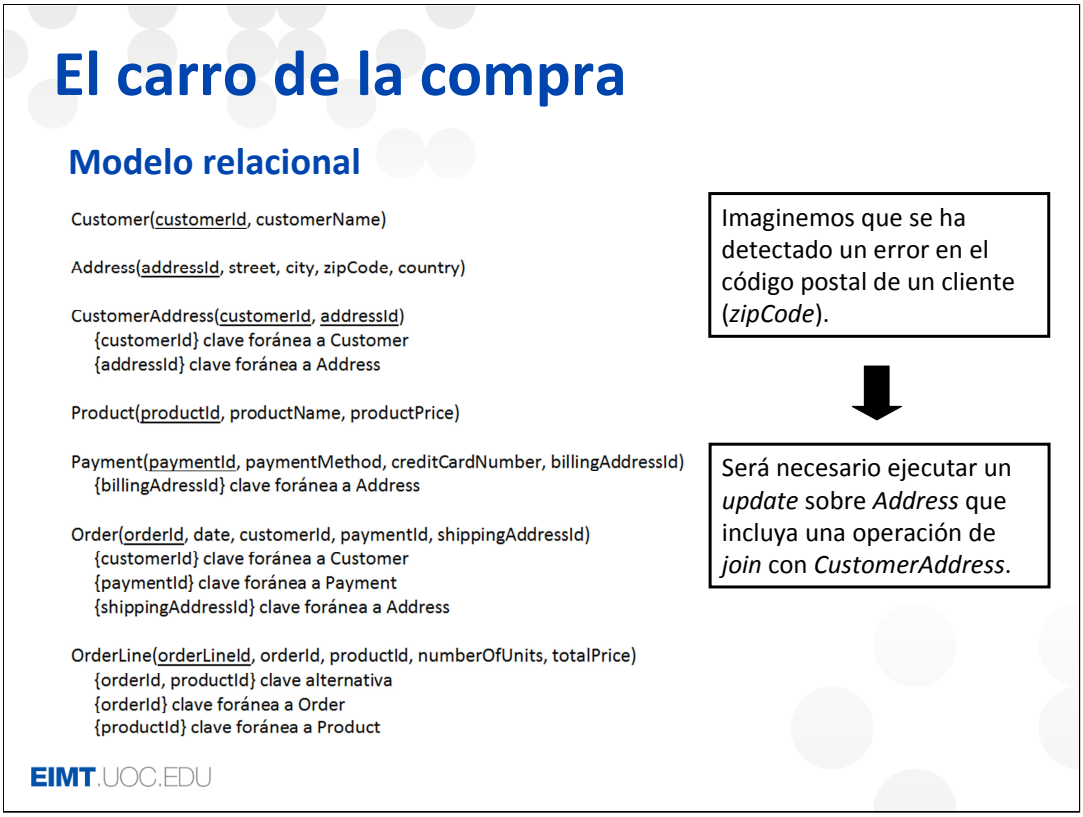


En el caso que usemos una base de datos basada en un modelo documental, asumiendo que únicamente hemos diseñado un agregado como el descrito anteriormente, podemos ver que para recuperar la información deseada será necesario acceder a todos los pedidos.

Los pedidos recuperados, además, contienen información irrelevante para responder a la nueva petición, y deberán ser tratados uno a uno por nuestra aplicación. Adicionalmente, si la base de datos está distribuida, es posible que los diferentes pedidos estén almacenados en diferentes nodos del sistema distribuido, penalizando el tiempo de respuesta.

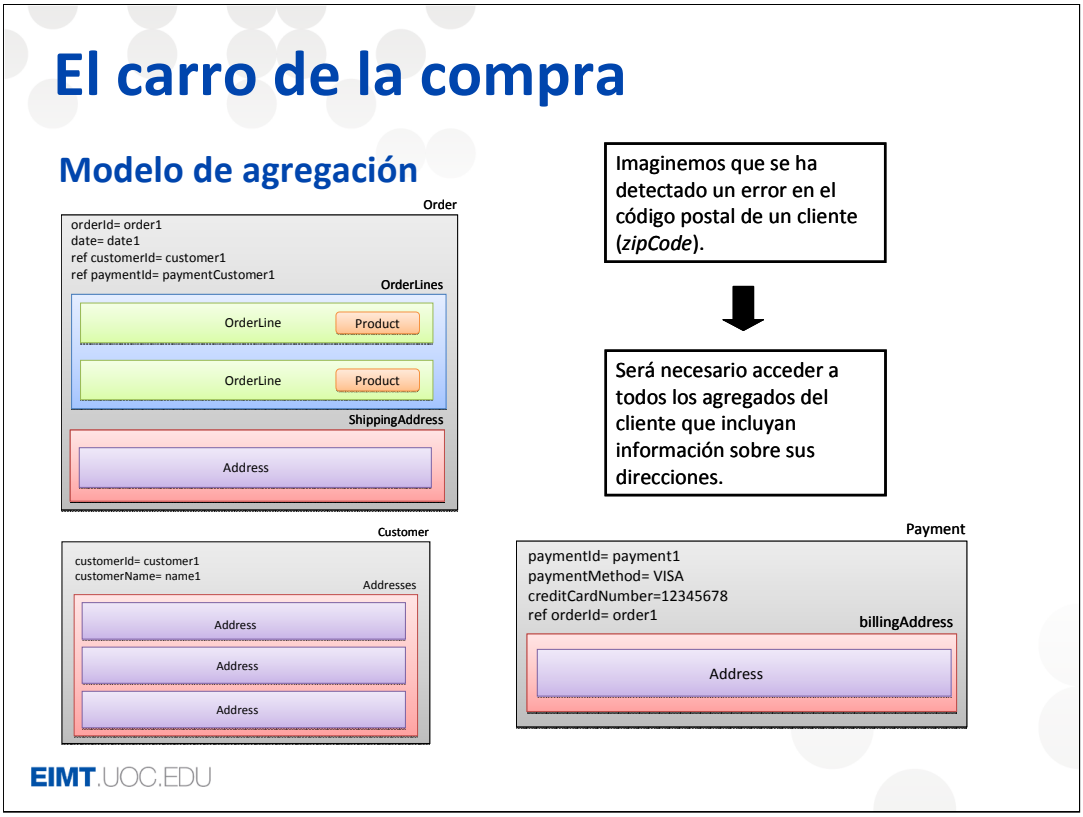
Por lo tanto una conclusión importante es que el diseño de agregados no da soporte a cualquier necesidad de información.

Esto no es así en el caso de una base de datos relacional, dado que ésta se basa en unos principios de diseño que garantizan una visión global de la realidad a representar. Esta visión deber ser lo suficientemente flexible para que se pueda adaptar a las necesidades de las diferentes aplicaciones y grupos de usuarios.



Para finalizar, supongamos que se ha detectado un error en la introducción de datos. En concreto, el código postal de un cliente es incorrecto.

En el caso que la base de datos fuese relacional, para subsanar el error, sería necesario modificar el código postal en cuestión en la relación de direcciones. La petición requerirá una operación de combinación con la relación que guarda cuáles son las direcciones asociadas a cada cliente (esta relación es *CustomerAddress*).



Imaginemos ahora que utilizamos una base de datos basada en un modelo documental. Supongamos también que hemos diseñado diferentes agregados. Tres de estos agregados (pedidos, clientes y pagos) incluyen datos relativos a direcciones postales de clientes.

En este caso, para hacer efectivo el cambio de código postal, será necesario acceder a todos los agregados del cliente que incluyan información sobre sus direcciones. Adicionalmente la aplicación deberá controlar que cada cambio en cada agregado afectado se ha realizado con éxito en la base de base de datos (en definitiva, la aplicación deberá estar atenta a que no se produzca ningún error de operación). Esto es consecuencia de que el agregado es la unidad a efectos de consistencia y control de concurrencia. En otras palabras, la base de datos no se responsabilizará de garantizar que todos los cambios en todos los agregados afectados se realicen de forma atómica.

Fijémonos que tenemos un problema derivado de la redundancia de los datos, es decir, de repeticiones que, desde el prisma del modelo relacional, son evitables. Si bien la existencia de redundancias favorece la consulta de los datos, es importante destacar que hace más compleja su modificación.

Para finalizar, comentar que la problemática descrita se podría haber solventado de otras maneras. Por ejemplo, modificando únicamente los datos en el agregado de clientes, o añadiendo una nueva dirección en ese mismo agregado a utilizar en futuros pedidos. En definitiva, es perfectamente posible que nuestro dominio de interés (el carrito de la compra) pueda renunciar a garantizar la coherencia de unos mismos datos.

Modelos de agregación

- Introducción
- Ejemplo motivador

EIMT.UOC.EDU

Hasta aquí esta introducción a los modelos de agregación donde hemos ilustrado, a través de un ejemplo motivador, los principales beneficios e inconvenientes de este modelo en comparación al modelo relacional.

Referencias

R. Camps (2011). Los datos: conceptos introductorios. Material docente UOC, asignatura Uso de bases de datos.

R. Camps (2011). Introducción a las bases de datos. Material docente UOC, asignatura Uso de bases de datos.

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

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.

EIMT.UOC.EDU

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

Que tengáis un buen día.