Práctica 1: Modelado en Cassandra

viu Universidad Internacional de Valencia

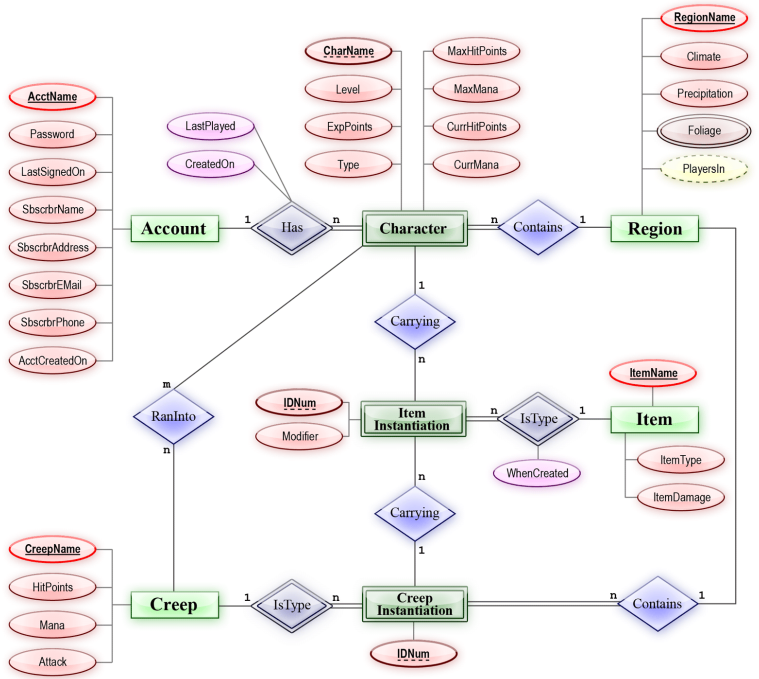
24.10.23

¿Cómo diseñar una BBDD?

- Evitar el diseño directo.
- Seguir metodología.
- Identificar todos los niveles:
  - Conceptual
  - Lógico
  - Físico
- Estos 3 modelos serán usados para modelar en varias BBDD, desde relacionales hasta NoSQL

Modelo conceptual

- Representan el dominio de la aplicación a un alto nivel, utilizando términos y conceptos familiares para los usuarios de la aplicación, ignorando los aspectos de nivel lógico y físico
- Formados por 3 elementos principales:
  - Entidad
  - Atributo
  - Relación



Obtención del modelo conceptual

- Entender modelo a representar, es decir, el problema.
- Identificar entidades, atributes y relaciones entre atributos.

- Ejemplo de un caso de estudio:
  - Tenemos que representar un modelo conceptual de una empresa que tiene que enviar productos a sus clientes en pedidos

1. Cada pedido tiene un número, y se almacenará el cliente que lo solicita y la fecha en la que se realiza
2. De cada producto se quiere registrar un código de producto, su descripción, su precio y sus existencias
3. Para los clientes se quiere registrar su nombre, su DNI y su dirección
4. Un pedido incluirá una serie de productos cada cual con su cantidad

Entidades del problema

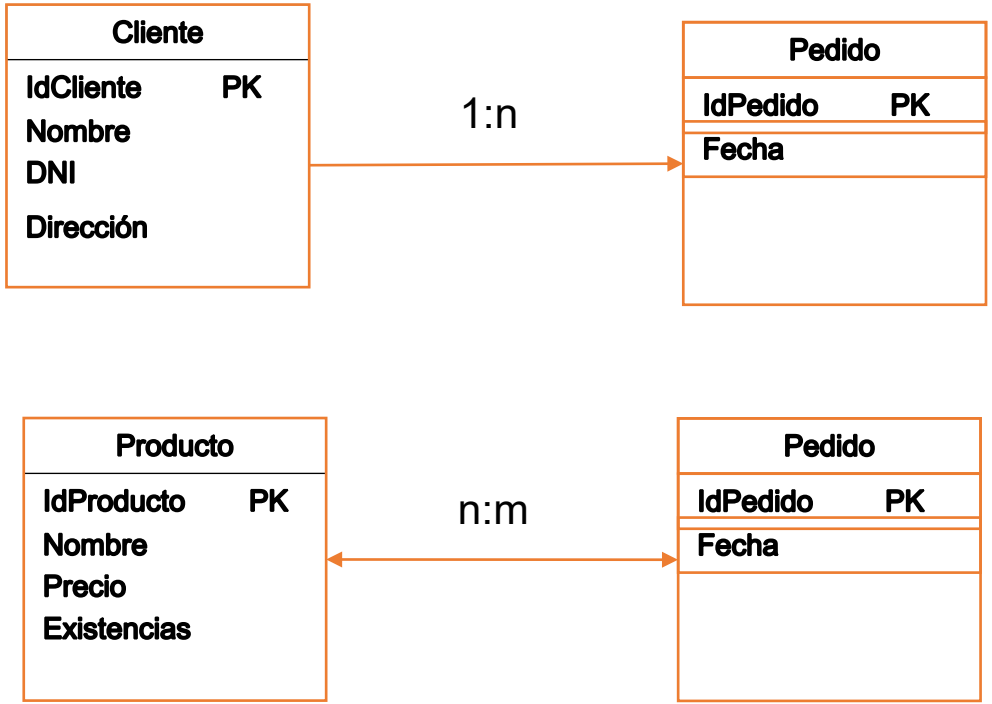
Producto	
IdProducto	PK
Nombre	
Precio	
Existencias	

Pedido	
IdPedido	PK
Fecha	

Cliente	
IdCliente	PK
Nombre	
DNI	
Dirección	

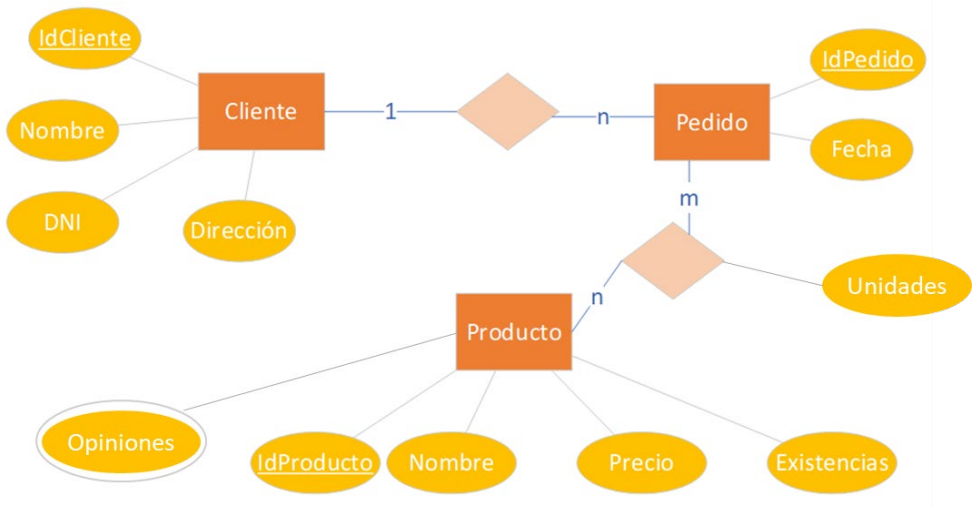
Relaciones

- Tres tipos de relaciones:
  - 1:1
  - 1:n
  - n:m
- “Un cliente puede realizar muchos pedidos, pero un pedido sólo está asociado a un cliente” -> Relación 1:n
- “un producto puede aparecer en muchos pedidos, y un pedido puede contener muchos productos”-> Relación n:m



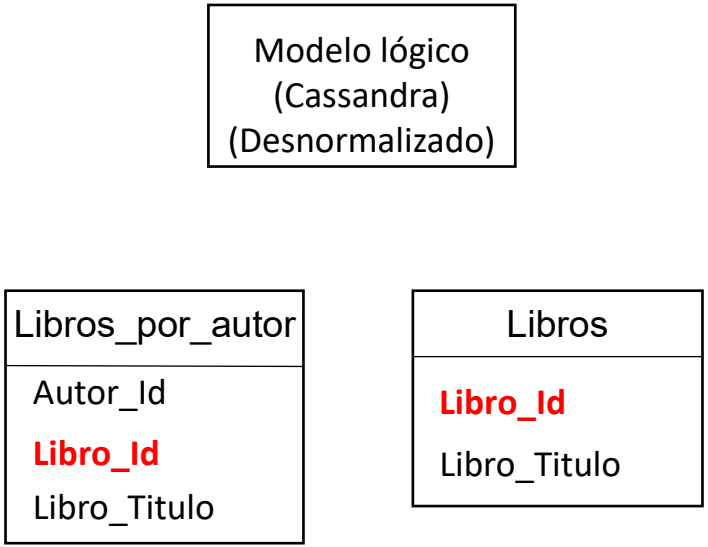
Representación

- **Rectángulos representan entidades.**
- **Elipses representan atributos.**
- **Rombos representan relaciones.**
- **Líneas representan relaciones entre los atributos y las entidades o entre las relaciones y las entidades**
- **Las claves primarias se representan con subrayado.**
- **Doble elipse representa un atributo de tipo conjunto.**



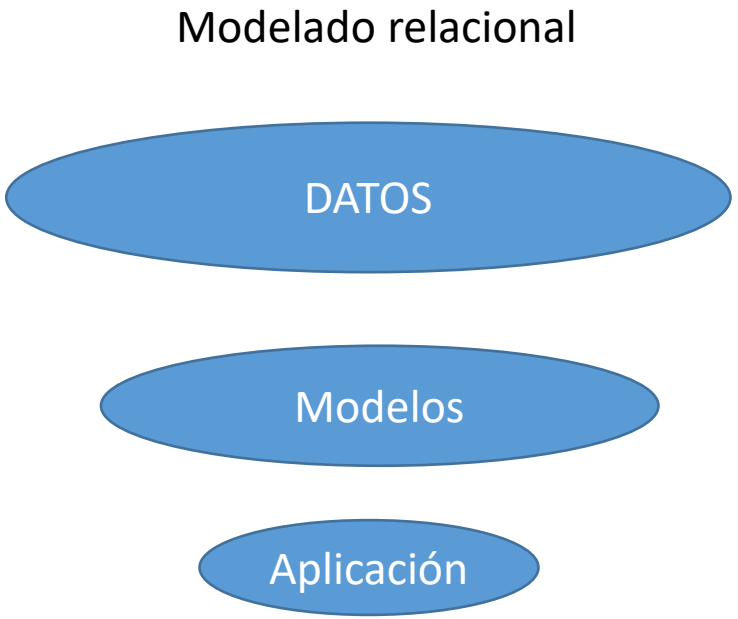
Apache Cassandra

- **Modelo de datos basado en las necesidades de la aplicación que usaremos para trabajar con Cassandra.**
- **Estas necesidades serán las consultas (queries en inglés) que ejecutamos, siendo por lo general una tabla por consulta.**
- **Esto nos va a llevar a una probable duplicación de datos en nuestra base de datos ya que una consulta puede consultar el mismo dato varias veces.**

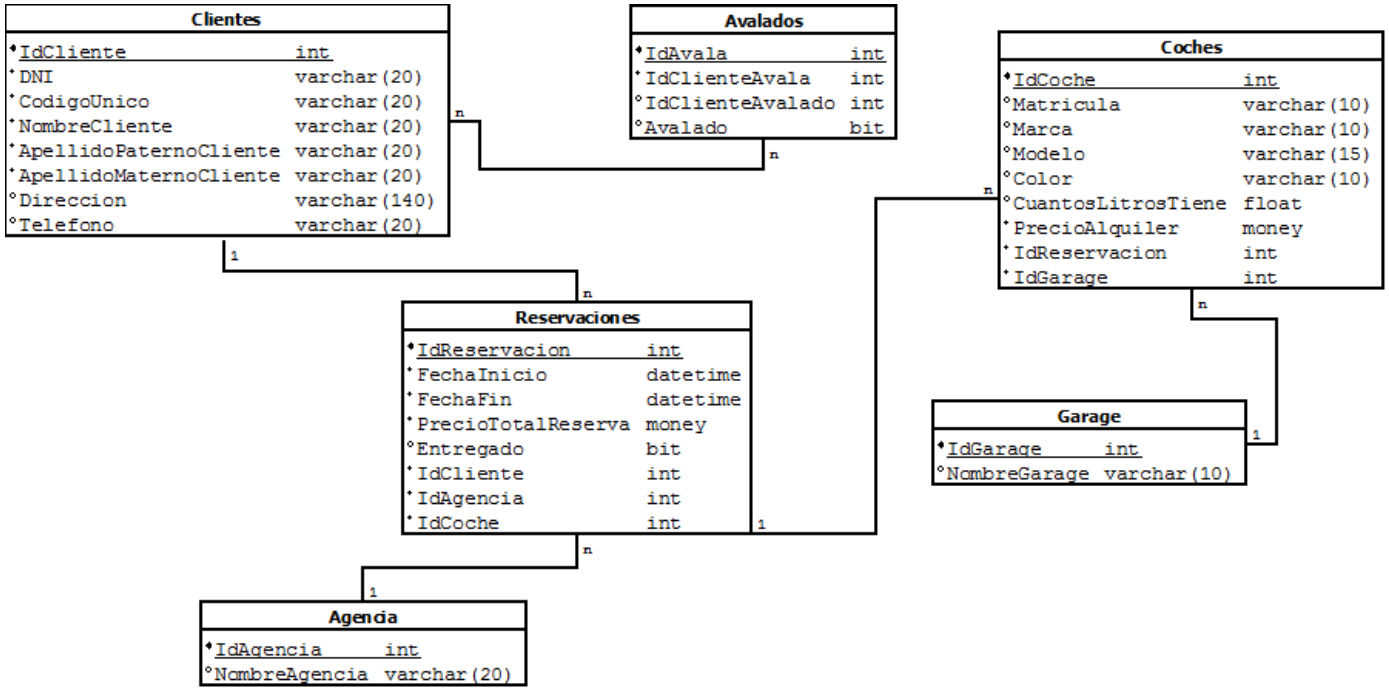


Modelado en una BBDD relacional

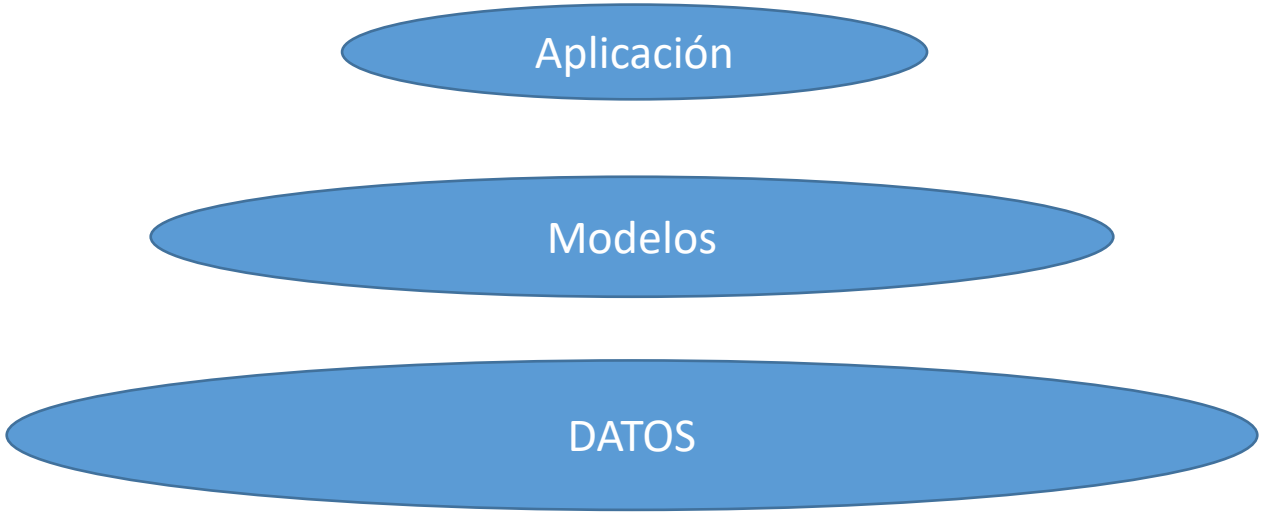
- **En una base de datos relacional modelamos nuestros datos basándonos en los datos.**
- **Lo más común es tener una base de datos normalizada y en cada una de las tablas la única duplicidad de datos serán las claves ajenas que nos servirán para realizar JOINs entre tablas.**
- **Esto en Cassandra no tendrá ningún sentido, pues no podemos realizar consultas a través de JOINS.**



Ejemplo de base de datos relacional



Modelado en Cassandra



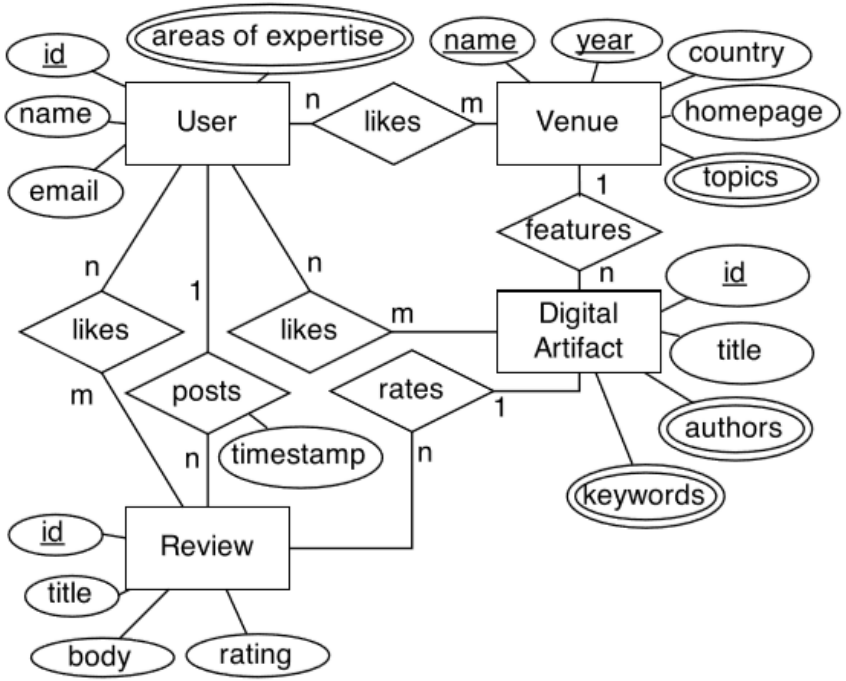
Apache Cassandra

- **Como necesitamos modelar a través de la aplicación necesitar seguir una metodología para modelar nuestra base de datos de forma adecuada.**
- **Una opción sería crear las tablas de forma directa, basándonos en qué queremos extraer. Sin embargo con esto corremos el riesgo de crear tablas que hagan inconsistente nuestra base de datos.**
- **Por lo tanto seguiremos una metodología específica propuesta por Artem Chebotko, de Dataxtax Inc.**

Modelo de datos

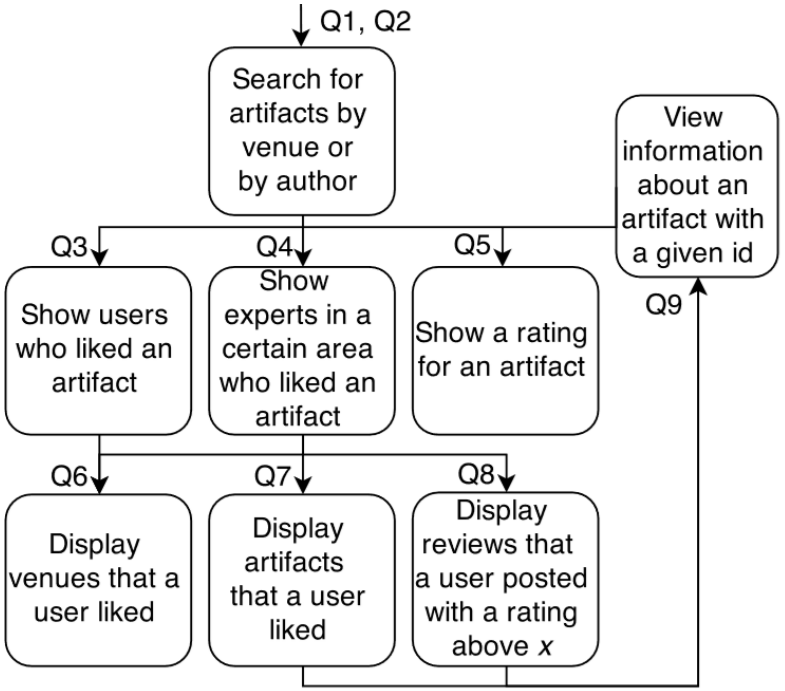
- Los 4 pasos que seguiremos para construir nuestro modelo de datos son:
  1. Construir el flujo de trabajo de la aplicación
  2. Modelar las consultas
  3. Diseñar las tablas
  4. Elegir de forma correcta la clave primaria
- Seguiremos estos 4 pasos a través del ejemplo definido por Chebotko, Kashlev y Lu en el artículo "A Big Data Modeling Methodology for Apache Cassandra".
- Este trabajo propone la automatización de modelos lógicos basándonos en un modelo conceptual y en consultas.

Ejemplo DataLibraryPortal



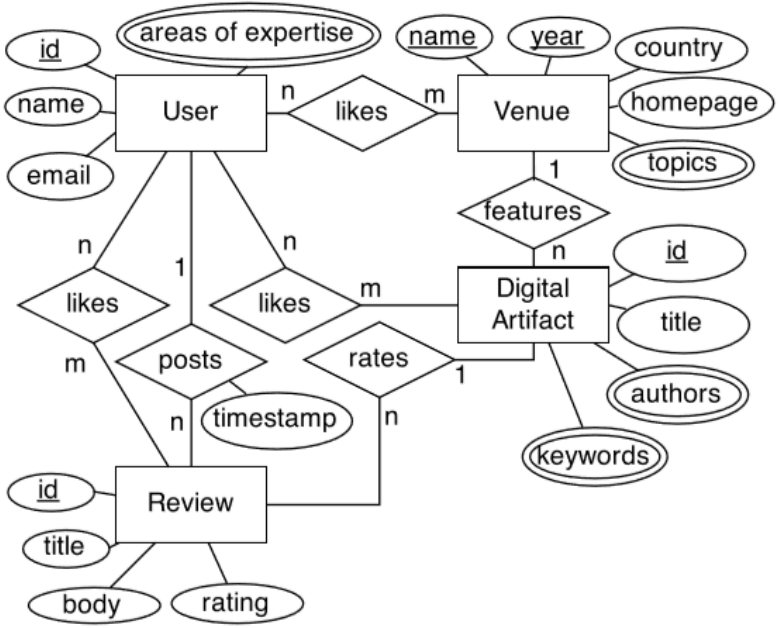
Flujo de trabajo de la aplicación

- **Una vez establecido qué hace nuestra aplicación y qué necesitamos, tenemos un conocimiento específico de qué tablas tenemos que crear.**
- **Aunque en nuestros ejercicios por simplicidad haremos una tabla por necesitad, podría darse el caso de crear una tabla que pueda satisfacer a dos necesidades diferentes.**



Modelar las consultas

- **Para poder modular nuestra consultas de una forma ordenada primero crearemos un modelo conceptual con entidades y relaciones.**
- **Este paso es muy importante, pues sin un conocimiento de como es nuestra aplicación de forma conceptual podemos crear tablas que impidan obtener un estado consistente en la base de datos.**
- **P. ej: Creamos tablas de Usuarios en los que solo ponemos columnas correspondientes al nombre y áreas de las que es experto. ¿Y si hay dos usuarios con los mismos valores en ambos campos?**



Modelar las consultas

- **Una vez establecido el modelo conceptual y las necesidades de nuestra aplicación podemos empezar a crear nuestra consultas que implementaremos en la aplicación.**
- **Transformaremos las necesidades de nuestra aplicación a consultas especificas sobre el modelo conceptual.**

Buscar artifacts por autor o sede	Buscar id de artifact por el autor o por las PK de Venue (RELACION)
Mostrar usuarios a los les gusto un artifact	Buscar id de usuarios por el id del artifact. (RELACION)
Mostrar sedes que les gustaron a un usuario	Buscar id de venue por el id del usuario. (RELACION)
Mostrar expertos a los les gustó un artifact	Buscar id de usuarios a través del id del artifact y un tipo de área de expertos (RELACIÓN)
Mostrar artifacts gustados por un usuarios	Buscar id de artifacts a través de id de usuario (RELACIÓN)
Mostrar revisiones de artifacts que superen una nota	Buscar id de revisiones a través de la nota de la revisión y el id del artifact (RELACIÓN)

Modelar las consultas

Ver la información de un artifact

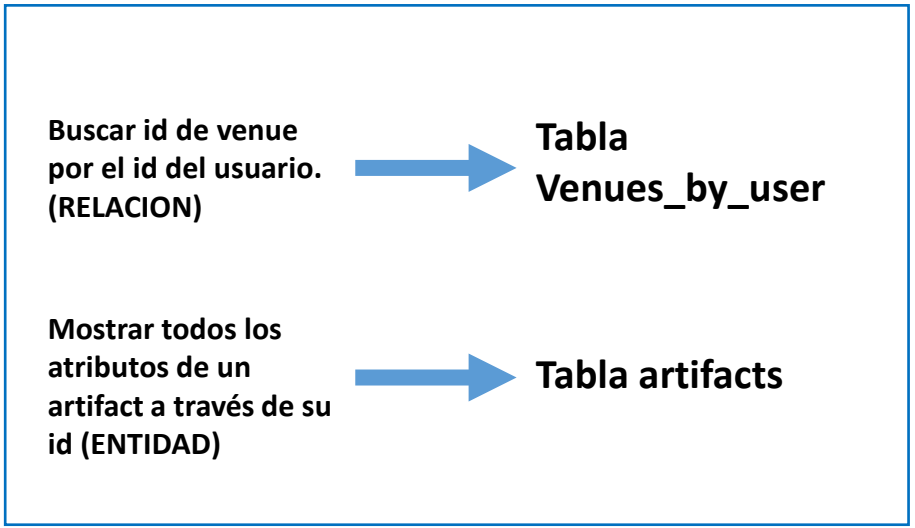
Mostrar todos los atributos de un artifact a través de su id (ENTIDAD)

Mostrar valoraciones para un artifact

Mostrar notas de un artifact a través de id de Artifact e id de review (RELACION)

Creación de tablas

- Utilizaremos las consultas que hemos determinado previamente para crear las tablas.
- Lo primero que haremos será darles nombres a las tablas. Puede parecer algo trivial, pero es muy importante darles nombres relevantes con los datos almacenados en ellas. (No poner Tabla1, Tabla2… Esto solo será válido en situaciones en las que se indique expresamente)
- Para ello miraremos que consultamos en cada consulta y a través de qué
- Si una tabla solo contiene datos de una entidad y la clave primaria es la de la entidad, podemos llamarla como esta entidad.

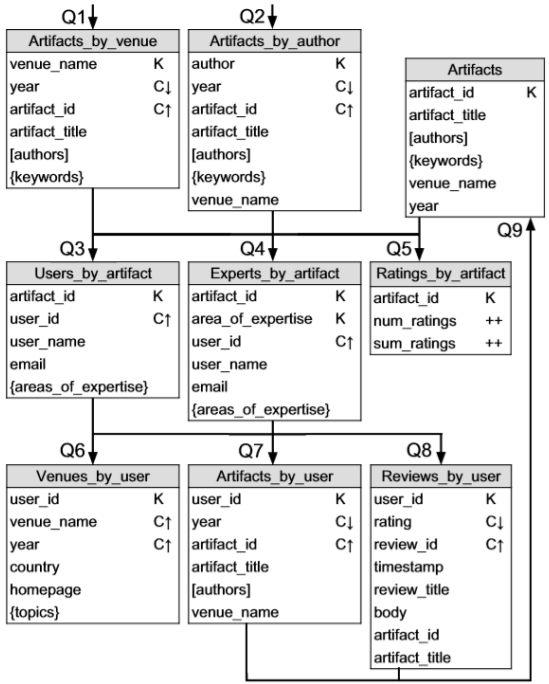


Creación de tablas

- **Al crear las tablas debemos tener especial cuidado con el resto de columnas. Por ejemplo, si buscamos por un atributo no clave, tendremos que añadir en la clave de la tabla una columna que identifique de forma única a dicho atributo.**
- **Por ejemplo, la primera necesidad es “buscar artifacts por sede o autor”. Si miramos al modelo lógico comprobamos que la sede (‘Venue’) es una entidad, por lo que esa necesidad la podemos implementar como una tabla en la que la clave primaria sean columnas asociadas a las claves primarias de ‘Venue’ en el modelo conceptual: año y nombre.**
- **Sin embargo, el autor es un atributo no clave, por lo que ya no será tan sencillo.**
- **En este caso debemos crear una tabla en la que podamos buscar por el autor pero en la que también esté en la clave primaria el id del propio Artifact. De otra manera podríamos tener una base de datos inconsistente debido a un mal modelo.**

Creación de tablas

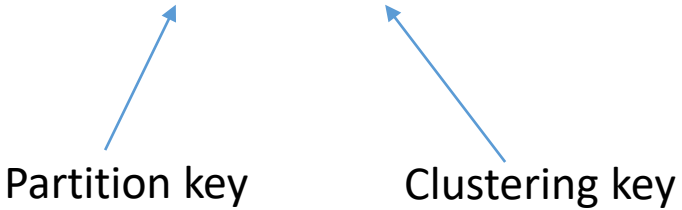
- **Los nombres del atributo es aconsejable que den información de a que entidad pertenecen.**
- **Si el nombre del atributo es único en todo el modelo conceptual, puedes omitir la entidad en tu tabla Cassandra, aunque no es aconsejable.**



Elegir columnas clave

- **En Cassandra existen dos tipos de columnas clave:**
  - **Partition key: El valor de dicha tabla indicará en que nodo se almacenará dicha fila de datos.**
  - **Clustering key: Dentro de una misma tabla permitirá ordenar los datos según estos datos**
- **Estas columnas son muy importantes tanto para rendimiento de nuestra base de datos como para tener el modelo que queremos. En Cassandra solo podemos usar columnas clave en el criterio de búsqueda, siendo la partition obligatoria si existe un criterio.**

```
CREATE TABLE
Users_by_artifact
(artifact_id uuid, user_id
uuid,  user_name text, email
text, areas_of_expertise
set<text>, PRIMARY KEY
(artifact_id, user_id)
```

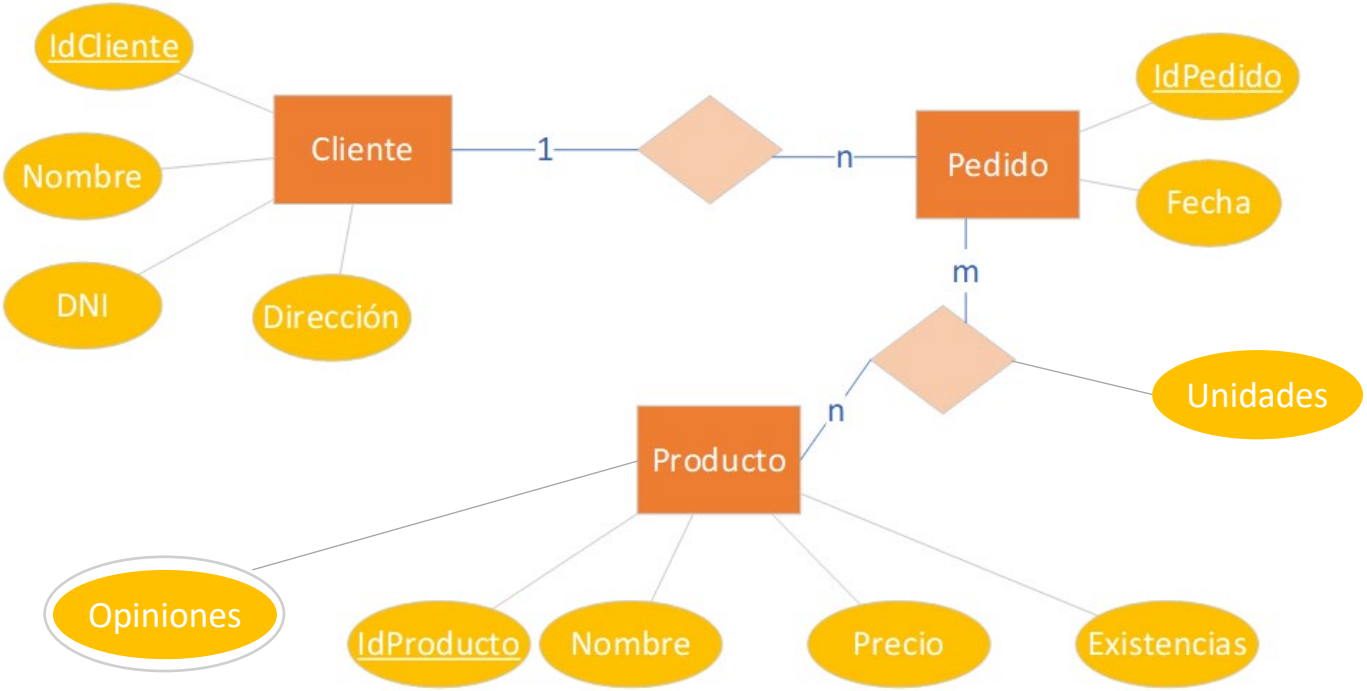


Elegir columnas clave

- **Dependiendo de las características de la tabla, podemos orientarla a que la partition key sea una columna con valores que se puedan repetir (un atributo normal) y que la clustering key sea la que tiene los valores que son distintos (la primary key del modelo conceptual)**
- **Esta decisión podría ayudarnos a mejorar el rendimiento de nuestra base de datos a través de la ejecución de consultas que retornen cientos de miles de filas rápidamente.**

IMPORTANTE: Se debe elegir una clave que garantice la unicidad de los datos.

Modelo conceptual ejemplo a usar



Comparación para consulta **“queremos consultar los productos almacenados en el sistema en base a su precio”.**

Tabla Productos

Pérdida de información

Precio (K)	Id_Producto	Nombre	Existencias

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5
2	2	Kiwy	10

Precio (K)	Id_Producto	Nombre	Existencias
1	3	Mandarina	4
2	2	Kiwy	10

Comparación para consulta **"queremos consultar los productos almacenados en el sistema en base a su precio".**

Tabla Productos

Pérdida de información

Precio (K)	Id_Producto	Nombre	Existencias

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5
2	2	Kiwy	10

Precio (K)	Id_Producto	Nombre	Existencias
1	3	Mandarina	4
2	2	Kiwy	10

Los datos se han sobrescrito

Comparación para consulta **"queremos consultar los productos almacenados en el sistema en base a su precio".**

Tabla Productos

Pérdida de información

Precio (K)	Id_Producto	Nombre	Existencias

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5

Precio (K)	Id_Producto	Nombre	Existencias
1	1	Manzana	5
2	2	Kiwy	10

Precio (K)	Id_Producto	Nombre	Existencias
1	3	Mandarina	4
2	2	Kiwy	10

Sin pérdida de información

Precio (K)	Id_Producto (C)	Nombre	Existencias

Precio (K)	Id_Producto (C)	Nombre	Existencias
1	1	Manzana	5

Precio (K)	Id_Producto (C)	Nombre	Existencias
1	1	Manzana	5
2	2	Kiwy	10

Precio (K)	Id_Producto (C)	Nombre	Existencias
1	1	Manzana	5
2	2	Kiwy	10
1	3	Mandarina	4

En Cassandra solo las columnas clave pueden ser parte del WHERE

Tabla Productos

Precio (K)	Id_Producto (C)	Nombre	Existencias

Queremos consultar por el precio:

SELECT * FROM Productos WHERE Precio = 1;

Operación correcta porque el Precio forma parte de la clave

Queremos consultar por el nombre:

SELECT * FROM Productos WHERE Nombre = “Manzana”;

Operación no válida porque Nombre no es parte de la clave primaria

Otra razón: Las columnas de la partition key deben de tener valores que se repitan para particionar mejor

Casos de modelado

Tabla que almacena una entidad

- Las claves primarias de la entidad siempre deben tener columnas mapeadas que pertenezcan a la clave primaria de la tabla.
- El resto de columnas deberán ser las que se consultarán en la consulta:
  - Si se busca por un atributo (equivalente al WHERE), la columna mapeada a dicho atributo será K.
  - Si se quiere encontrar información de un atributo (equivalente al SELECT) las columnas mapeadas a los atributos deberán ser no clave, excepto si el atributo a buscar es en sí K de la entidad, en cuyo caso la columna será clave debido al primer punto.

Tabla que almacena una o más relaciones

- Las tablas pueden almacenar múltiples relaciones a la vez, similar a diversos JOINs.
- Para garantizar la unicidad será obligatorio que se almacenen las claves primarias de la entidades de menor nivel en las relaciones.
- Por ejemplo, si existen tres entidades A, B y C todas relacionadas con relaciones 1:n encadenadas, solo necesitaríamos la clave primaria de C para garantizar unicidad.
- Opcionalmente, se pueden incluir las claves del resto de entidades si se justifica correctamente.
- En el caso de relaciones n:m es necesario incluir las claves primarias de ambas entidades.

Casos especiales

Agregaciones

Contar relaciones

- Sirven para almacenar información de con cuantas instancias de una entidad B está relacionada una entidad A.
- Ejemplo: Cuantos pedidos ha comprado un cliente

Num_pedidos	
IdCliente	PK
NumPedidos	+

Calculo numérico atributos entidad

- Sirven para almacenar cálculos numéricos con información propia de la entidad. Normalmente no tienen como clave primaria a la clave primaria de la entidad, siendo este el único caso.
- Ejemplo: Cuantos productos que cuestan un determinado precio

MismoPrecio	
Precio	PK
NumProductos	+

Sumas de valores

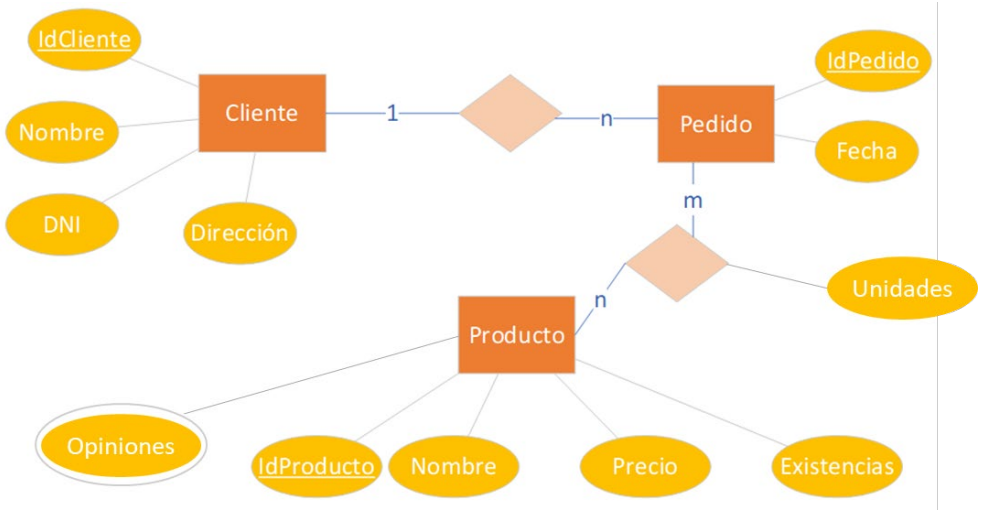
- En algunas ocasiones se requiere obtener una suma de valores que sean números
- Ejemplo: Cual es la cuantía total de un pedido sumando el precio de los productos solicitades

TotalPedido	
IdPedido	PK
CuantíaTotal	+

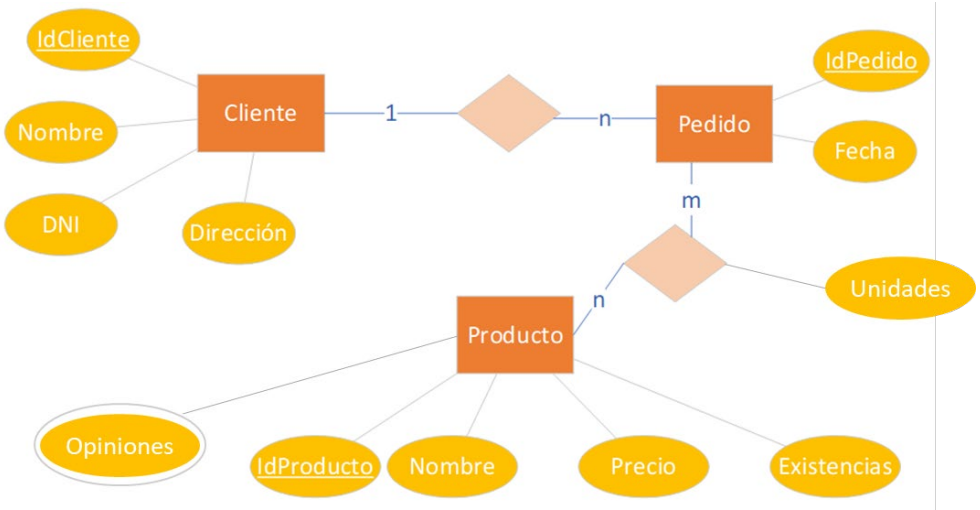
Símbolo ‘+’ representa columnas agregada

Atributos de conjuntos

- En algunos escenarios se requiere buscar algo por un conjunto como puede ser “opiniones” de producto
- En Cassandra un conjunto como los tipo ‘set’ no pueden ser clave primaria.
- En estos casos se debe implementar una columna que no sea un conjunto como clave primaria.
- El tratamiento posterior de los datos se hará a nivel de aplicación. Por cada elemento del conjunto, una nueva fila en la tabla.
- **Con respecto a la creación del esquema, no necesitamos realizar ninguna tarea extra a las que tendríamos que hacer para atributos que no representan conjuntos. Solo será relevante a la hora de manipular datos.**



Atributos de conjuntos



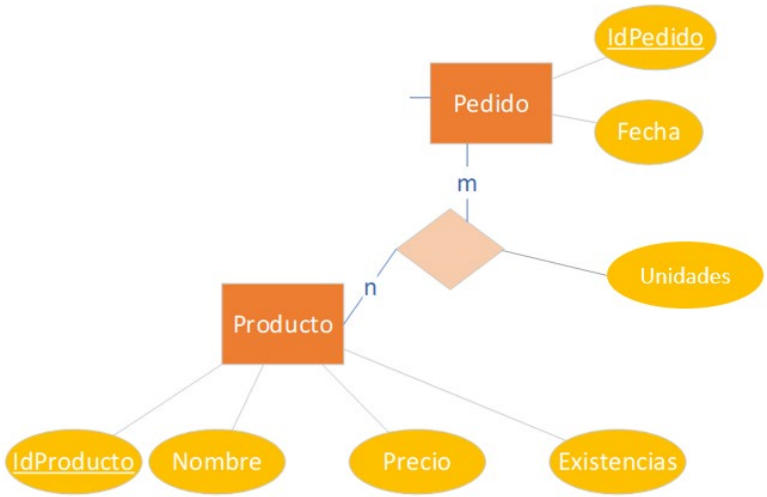
Consulta: Productos por opinión

Opiniones	
Opinion	K
IdProducto	C
Opiniones	

Opinión solo almacenará valores individuales, no de conjunto

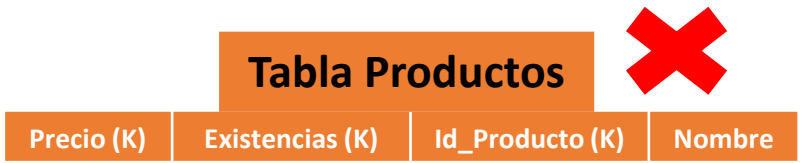
Casos especiales

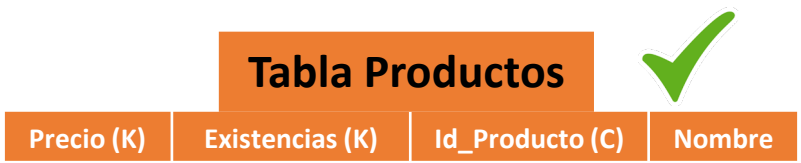
- Las relaciones pueden tener atributos asignados.
- En ese caso cuando se haga una tabla que incluya la relación este atributo será incluido en la tabla.
- Pueden ser clave primaria si son requeridas por la consulta como criterio de búsqueda.
- Para garantizar la unicidad de la tabla creada:
  - 1:n: Se requiere la PK de la entidad detalle (cardinalidad n)
  - n:m: Se requiere la PK de ambas entidades relacionadas



Sobre la importancia de la Partition key

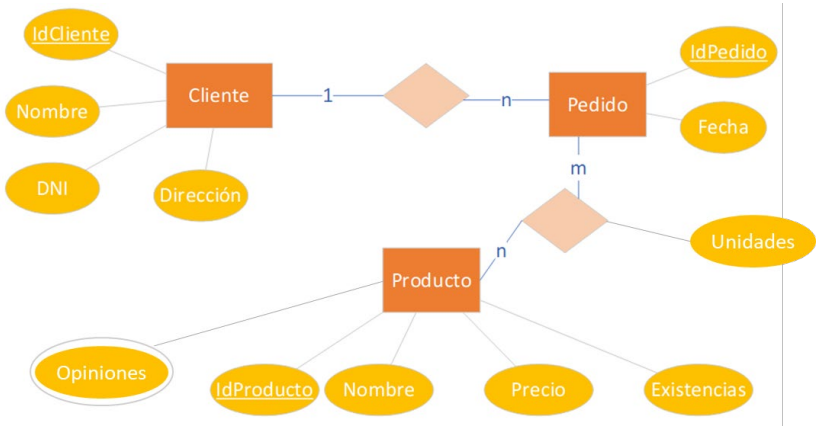
- Hasta ahora hemos comentado que es recomendable que la partition key sea una columna que tenga valores que se repitan.
- Se pueden hacer diseños más refinados estudiando los datos que vamos a almacenar. Por ejemplo imaginemos un escenario basándonos en la tabla Productos que hemos creado en la que existe una cantidad mayoritaria de productos que tienen el mismo precio.
- En ese escenario la tabla que planteamos podría presentar problemas a la hora de distribuir los datos al asignar todos los de ese precio a solo una partición.
- Posibilidad: Crear una **composite partition key** en la que incluyamos las existencias aunque no fuese requerido en la consulta.
- Mal diseño: Incluir el Id_Producto como partition key. Esto es debido a que perderíamos la propiedad de agrupar los datos en particiones.





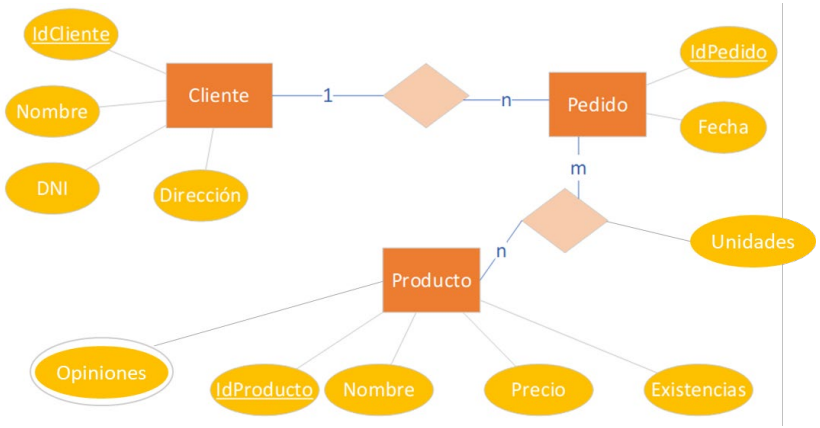
Diseñar Tablas Cassandra

1. **Crear una tabla que satisfaga la siguiente consulta: "saber todos los clientes que han realizado pedidos en una fecha determinada".**
2. **Crea una tabla que satisfaga la siguiente consulta: "Cuantos pedidos se han realizado en una determinada fecha"**
3. **Crea una tabla que satisfaga la siguiente consulta: "Quiero saber todos los productos comprados por un determinado cliente identificando también el pedido. El cliente se buscará por nombre y DNI".**
4. **Crea una tabla que satisfaga la siguiente consulta: "Quiero saber según el número de unidades las combinaciones de productos y pedidos que hay con esas unidades del producto"**



Diseñar Tablas Cassandra

5. **En el caso de que tengamos muchos productos con el mismo nombre, proponga un diseño para la siguiente consulta "Según el nombre de un producto quiero obtener la información de los mismos" para que no exista una situación en la que un nodo tenga que almacenar muchos datos.**



Crear una tabla que satisfaga la siguiente consulta: “saber todos los clientes que han realizado pedidos en una fecha determinada”.

Column	Primary Key
Fecha	Partition key
IdPedidos	Clustering Key
Nombre	
DNI	
Dirección	
IdCliente	

Crea una tabla que satisfaga la siguiente consulta: “Cuantos pedidos se han realizado en una determinada fecha”

Column	Primary Key
Fecha	Partition key
NumPedidos	+

Crea una tabla que satisfaga la siguiente consulta: “Quiero saber todos los productos comprados por un determinado cliente, el cual se buscará por nombre y DNI”.

Column	Primary Key
Nombre	Partition key
DNI	Clustering Key
IdProducto	Clustering Key
IdPedido	Clustering Key
NombreProducto	
Precio	
Existencia	

Crea una tabla que satisfaga la siguiente consulta: “Quiero saber según el número de unidades, cuantas combinaciones productos y pedidos hay con esas unidades del producto”

Column	Primary Key
Unidades	PK
IdProducto	Clustering Key
IdPedido	Clustering Key
NombreProducto	
Precio	
Existencia	
Fecha	

En el caso de que tengamos muchos productos con el mismo nombre, proponga un diseño para la siguiente consulta “Según el nombre de un producto quiero obtener la información de los mismos” para que no exista una situación en la que un nodo tenga que almacenar muchos datos.

Column	Primary Key
NombreProducto	Partition Key
Precio	Partition Key
Id	Clustering Key
Precio	
Existencia	
Opiniones	