El problema

¿Por qué tarda tanto?

Home

Pseudocodigo:

```
select datos de categorias
from categorias
where nivel in (l1, l2)
```

Suponiendo el siguiente pseudo modelo de datos:

```
id
titulo
nivel
categoria_padre_id
```

Página de detalle L2

Suponiendo el siguiente pseudo modelo de datos:

```
select datos de producto
from producto
where categoria_id = 4897
```

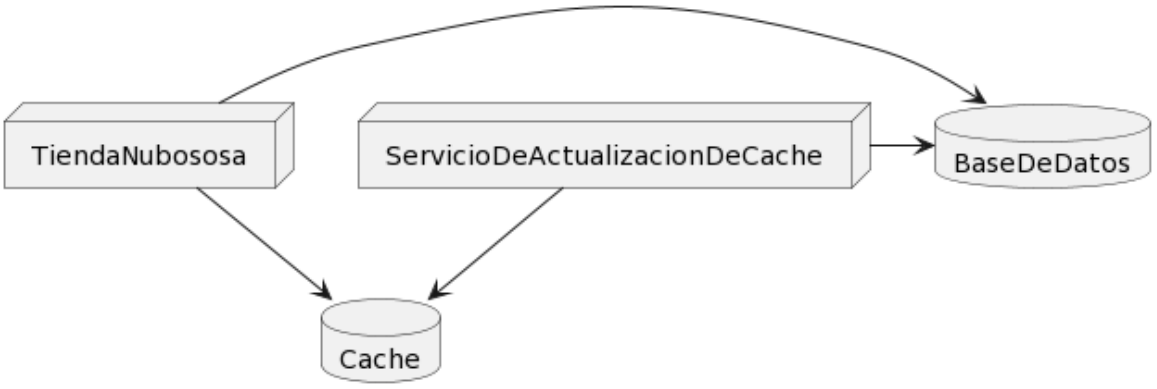
Pero ojo que además hay que:

- *joinear* con estadísticas de ventas.
- obtener el top 8
- contemplar que el es un árbol de categorías y que cada productos podría estar en cualquier nodo u hoja del árbol

Propuestas

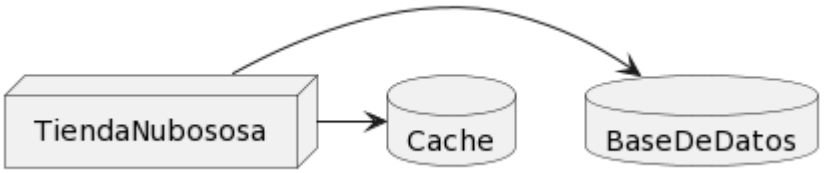
- Denormalizar la tabla de `productos`. Ok, pero hay un problema con la cantidad de normalización
    - `id`
    - `titulo`
    - `categoria_id`

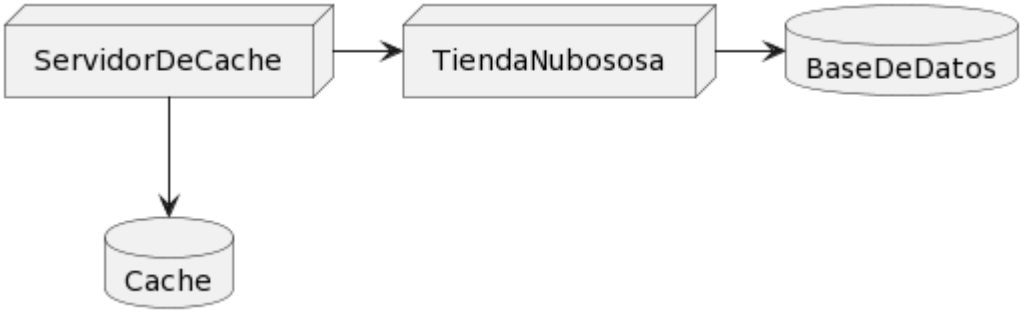
    - **`categoria_l2_id`**
    - `categoria_l3_id`
    - `categoria_l4_id`
    - etc…
    - Es una opción interesante, pero mucho cuidado porque la actualización del árbol de categorías ahora NO es trivial.
    - No resuelve completamente el problema, porque no da una respuesta al problema del join y mantenimiento de estadísticas
        - Se podría también desnormalizar la información de estadísticas y tener una tabla (o incorporar a una ya existente) la información de la cantidad de ventas en la última hora. Esto podría funcionar pero es complejo de implementar
        - Ojo que esto que no necesariamente resuelve el problema: ¿***estamos queriendo los datos de la última hora*** o de los últimos 60 minutos?
- **Caché offline**: Correr una tarea (cron) que periódicamente (por ejemplo, una vez por hora) realice la consulta que necesito (tarde lo que tarde) y luego guardarlos en un caché. Luego, cuando hagamos una consulta a través de la web, se va a consultar el caché directamente.
    - Desventaja: puede que hagamos computo innecesario si nadie lo requiere
    - Desventaja: no funciona si la caché está en memoria del proceso ni con proxies inversos.



- **Caché online**: Variante de lo anterior: también usar una caché pero el proceso de actualización no lo dispara un proceso independiente y cronometrado, sino que lo hace la propia petición de une usuarie.
    - Desventaja: la primera persona en acceder es penalizada con un tiempo de espera significativo. Si la consulta real en un contexto sin carga puede ejecutarse a tiempo, puede ser una estrategia razonable.
        - Variante: si no está en la caché, devolver aún el dato anterior, pero dejar un proceso en segundo plano que lo recompute.
    - Tanto para este caso como para el anterior, existen bases de datos especializadas que tienen formato clave-valor, viven en memoria, tienen arquitectura cliente/servidor y algunas incluso pueden ser distribuidas para aumentar su capacidad incrementando la cantidad de nodos.
        - Redis

        - Memcached
    - Ventaja: funciona con los dos primeros tipos de cachés comentados en la clase
    - Puede ser el propio servidor de TiendaNubosa quien se conecte a la caché, o bien puede ser otro que se coloque delante de TiendaNubosa:





- **Proxy Inverso** (variante de la caché online)