Pregunta: Suponiendo que la tabla Pedidos tiene muchos registros, que consideraciones se deberían tener en cuenta para realizar consultas a la base de datos de manera eficiente?

Respuesta

Algunas consideraciones que tendría en cuenta para realizar consultas a la base de datos sobre una tabla de muchos registros son:

- Evaluar la utilización de un sistema de búsqueda distribuida con servidores indexados. Donde cada servidor de búsqueda contenga una parte de la indexación del contenido general (almacenado en memoria), lo cual permite un acceso a la información de forma muy rápida. Para esto se necesita un sistema que centralice las búsquedas y devuelva la unión de los resultados que recolectó de los distintos servidores. Esto también tiene como beneficio no centralizar las búsquedas en un solo servidor y en el caso de que un servidor baje su performance puede reemplazarse rápidamente por otro servidor con el mismo índice
- Utilizar índices sobre tablas/queries y verificar cual es la mejor estrategia para implementar esos índices.
- Optimización de queries: verificar que las queries que van a la base de datos estén optimizadas.
- Selección del engine correcto: los engines de las bases de datos utilizan distintos métodos de lockeo y buffer cuando se accede a los datos. Se debería seleccionar el engine más adecuado según las operaciones del sistema.
- Utilizar cache en memoria: Hoy en día existen varios frameworks muy potentes para implementar en un backend como pueden ser Guava de google (https://github.com/google/guava) o EhCache (http://www.ehcache.org/). Los mismos aparte de su funcionalidad ofrecen estadísticas que pueden ser muy útiles.
- Si estoy utilizando un ORM, como puede ser hibernate, verificaría que las configuraciones del mismo sean correctas. Analizaría cómo se comportan los distintos niveles de caché que el framework propone y verificaría si hay que hacer algún ajuste sobre los mismos. A veces utilizar un ORM es práctico para el desarrollo pero para determinados sistemas que tienen gran flujo de datos puede ser contraproducente si no están bien configurados. Cuando se utiliza un ORM tratar de hacer operaciones cortas y no operaciones largas. En el caso de accesos masivos a la base es conveniente hacer operaciones e ir persistiendo en la Base de datos las mismas ya que, dependiendo de la configuración del ORM, estas operaciones se guardan en la caché del framework y seguramente bajará la performance de la aplicación.
- Utilizar en la base de datos tablas e índices particionados para descomponerlos en subtablas y/o subíndices.
- Utilizar el query optimizer de la base de datos para mejorar las queries.
- Evaluar si es necesario utilizar algún store procedure (sólo si es necesario y de determinada forma). Evitar lógica de negocio en los mismos ya que tendríamos la lógica en código y en la base y eso sería inmantenible con el tiempo.