Hibernate Performance

Tipos de Fetch

Se utilizan para cuando y como se traen las relaciones

- Eager (default) - Siempre al pedir el objeto.
- Lazy - Solamente al hacer referencia a la variable.

Fetch Mode

- SELECT (Default)
  - Se realiza una segunda query después de obtener el resultado de la primera, esta segunda utiliza los ids
  - En el caso de que sea EAGER la segunda query se realiza inmediatamente
  - En el caso de LAZY la segunda query se realiza al pedir por la variable
- BatchSize
  - Realiza una query para buscar N elementos dentro de esta coleccion, trae de a rafagas de N elementos.
- JOIN
  - Solo realiza una sola query con un join en ella, en el caso de que se encuentre mapeado con LAZY no lo será.
- SUBSELECT
  - Realiza una segunda query donde el where contiene un IN con la primera query.

Cache LVL 1

- Esta cache viene por defecto en Hibernate, y solamente se encuentra en la Session. Esto quiere decir que si realizo N veces la misma query dentro de la Session solo se traerán los datos 1 sola vez.

Cache lvl 2

Para utilizarla se tienen que agregar en el hibernate.cfg.xml

```
<property name="hibernate.cache.use_second_level_cache">true</property>
<property name="hibernate.cache.region.factory_class">
org.hibernate.cache.ehcache.EhCacheRegionFactory</property>
```

READ_ONLY

Es la estrategia de concurrencia menos estricta. Ideal para datos que nunca cambian.

```
@Entity

@Cache(usage = CacheConcurrencyStrategy.READ_ONLY)

public class Item{

...

}
```

NONSTRICT_READ_WRITE

No ofrece ninguna garantía de consistencia entre la caché y la base de datos. Para sincronizar los objetos de la cache con la base de datos se utilizan timeouts, de modo que cuando caduca el timeout se recargan los datos.

```
@Entity

@Cache(usage = CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)

public class Item{

...

}
```

TRANSACTIONAL

Garantiza un nivel de aislamiento.
Se recomienda su uso cuando no podamos permitirnos datos que queden desfasados. Esta estrategia sólo se puede utilizar en clusters, es decir, con cachés distribuidas.

```
@Entity
@Cache(usage = CacheConcurrencyStrategy.TRANSACTIONAL)
public class Item{
...
}
```

READ_WRITE

Mantiene un aislamiento hasta el nivel de commited (realizar un soft lock a la cache), utilizando un sistema de marcas de tiempo (timestamps). El caso de uso recomendable es el mismo que para la estrategia transactional pero con la diferencia de que esta estrategia no se puede utilizar en clusters.

```
@Entity
@Cache(usage = CacheConcurrencyStrategy.READ_WRITE)

public class Item{
...
}
```

Indices

Un índice contiene claves generadas a partir de una o varias columnas de la tabla o la vista. Dichas claves están almacenadas en una estructura (árbol b) que permite que SQL Server busque de forma rápida y eficiente la fila o filas asociadas a los valores de cada clave.

```
@Entity
@Table(indexes = { @Index(columnList="peso", name="item_peso_idx") })
public class Item {
        private int peso;
...
}
```

Tp 3

La Base fue creada por Fusheng Wang and Carlo Zaniolo de siemens

- La base tiene 300.000 empleados con 2.8 millones de salarios.

Objetivo:

- Reducir el tiempo de los test.
- Traer lo mínimo e indispensable para cumplir con el requerimiento.
- Evaluar si es conveniente pre-calcular datos
- Devolver estructuras nuevas