CitiusTech

Hibernate Caching

Session 7

Version 1.2

September 2018

Agenda

- **What is Caching?**
- Type of Cache
- Cache Scopes
- Second Level Cache Implementations
- Caching Strategies

What is Caching?

- Used for optimizing database applications
- A cache is designed to reduce traffic between your application and the database by conserving data already loaded from the database
  - Database access is necessary only when retrieving data that is not currently available in the cache
  - The application may need to empty (invalidate) the cache from time to time if the database is updated or modified in some way, because it has no way of knowing whether the cache is up to date

Agenda

- What is Caching?
- **Type of Cache**
- Cache Scopes
- Second Level Cache Implementations
- Caching Strategies

Types of Cache (1/2)

First Level Cache

- Associated with the Session object
- Hibernate uses first-level cache on a per transaction basis (within a single transaction boundary)
    - Used mainly to reduce the number of SQL queries it needs to generate within a given transaction
    - For example, if an object is modified several times within the same transaction, Hibernate will generate only one SQL UPDATE statement at the end of the transaction, containing all the modifications

Types of Cache (2/2)

Second level Cache

- Associated with the **SessionFactory** object
- Second-level cache keeps loaded objects at the Session Factory level across transaction boundaries
- The objects in the second-level cache are available to the whole application, not just to the user running the query
  - This way, each time a query returns an object that is already loaded in the cache, one or more database transactions potentially are avoided

Query Level Cache

- Use it when you need to cache actual query results, rather than just persistent objects

Agenda

- What is Caching?
- Type of Cache
- **Cache Scopes**
- Second Level Cache Implementations
- Caching Strategies

Cache Scopes

- **Transcation scope**
  - Attached to the current unit of work, which may be an actual database transaction or an application transaction
  - It's valid and used as long as the unit of work runs
  - Every unit of work has its own cache
  - First level cache i.e. Session level cache is transaction scoped
- **Process scope**
  - Shared among many (possibly concurrent) units of work or transactions
  - This means that data in the process scope cache is accessed by concurrently running transactions, obviously with implications on transaction isolation
  - A process scope cache might store the persistent instances themselves in the cache, or it might store just their persistent state in a disassembled format
- **Cluster scope**
  - Shared among multiple processes on the same machine or among multiple machines in a cluster
  - Requires some kind of remote process communication to maintain consistency
  - Second-level cache is process scoped or cluster scoped

Agenda

- What is Caching?
- Type of Cache
- Cache Scopes
- **Second Level Cache Implementations**
- Caching Strategies

Second Level Cache Implementations (1/2)

- Hibernate supports these open-source cache implementations out of the box
  - EHCache (org.hibernate.cache.EhCacheProvider)
  - OSCache (org.hibernate.cache.OSCacheProvider)
  - SwarmCache (org.hibernate.cache.SwarmCacheProvider)
  - JBoss TreeCache(org.hibernate.cache.TreeCacheProvider)

Second Level Cache Implementations (2/2)

EHCache

- Fast, lightweight, and easy-to-use in-process cache
- Supports read-only and read/write caching, memory and disk-based caching
- However, it does not support clustering

OSCache

- Part of a larger package, which also provides caching functionalities for JSP pages or arbitrary objects
- It is a powerful and flexible package, which, like EHCache, supports read-only and read/write caching, and memory- and disk-based caching
- It also provides basic support for clustering via either JavaGroups or JMS

SwarmCache

- Is a simple cluster-based caching solution based on JavaGroups
- Supports read-only or nonstrict read/write caching (the next section explains this term)
- This type of cache is appropriate for applications that typically have many more read operations than write operations

JBossTreeCache

- Is a powerful replicated (synchronous or asynchronous) and transactional cache
- Use this solution if you really need a true transaction-capable caching architecture

Agenda

- What is Caching?
- Type of Cache
- Cache Scopes
- Second Level Cache Implementations
- **Caching Strategies**

Caching Strategies

- Read-only
- Read/write
- Nonstrict read/write
- Transactional

Caching Strategy: Read/Write

- Appropriate if your data needs to be updated
- They carry more overhead than read-only caches
- In non-JTA environments, each transaction should be completed when `Session.close()` or `Session.disconnect()` is called
- To use this strategy in a cluster, you should ensure that the underlying cache implementation supports locking
- Built-in cache providers do not support locking

```
<class name="eg.Cat" .... >
    <cache usage="read-write"/>
    ....
    <set name="kittens" ... >
        <cache usage="read-write"/>
        ....
    </set>
</class>
```

Caching Strategy: Nonstrict Read/Write

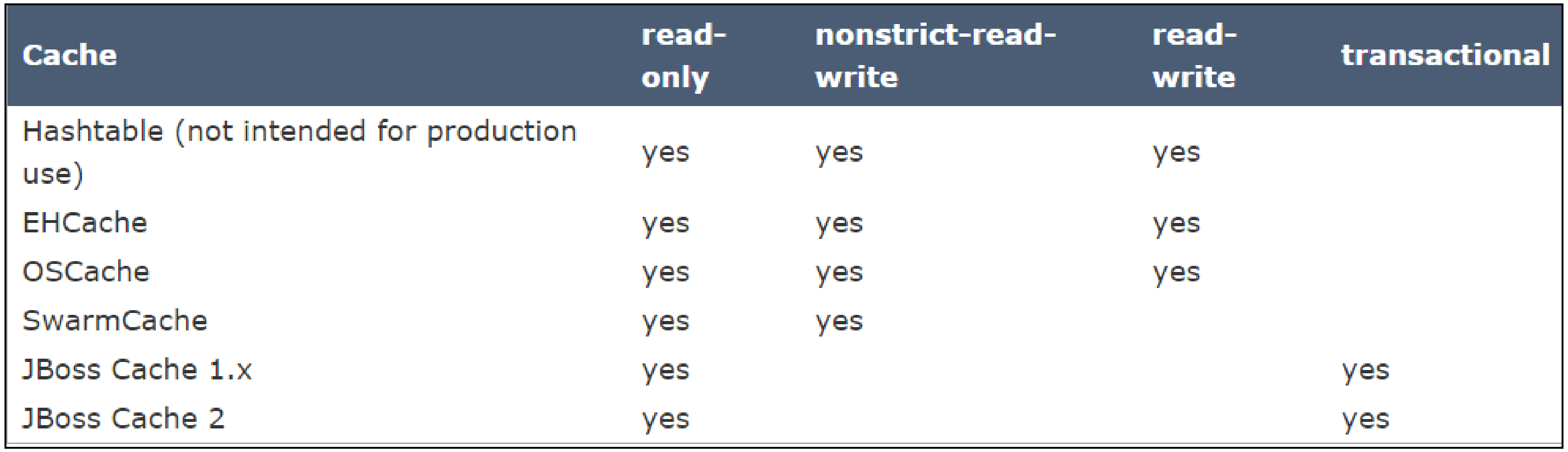
- Does not guarantee that two transactions won't simultaneously modify the same data
- Therefore, it may be most appropriate for data that is read often but only occasionally modified
- Strict transaction isolation is not required, a nonstrict-read-write cache might be appropriate
- In non-JTA environments, you should ensure that the transaction is completed when Session.close() or Session.disconnect() is called

Cache Strategy: Transactional

- Provides support for fully transactional cache providers such as **JBoss TreeCache**
- Cache can only be used in a JTA environment and you must specify **hibernate.transaction.manager_lookup_class**

Cache Concurrency Strategy Support

- Following table shows which providers are compatible with which concurrency strategies:



Cache	read-only	nonstrict-read-write	read-write	transactional
Hashtable (not intended for production use)	yes	yes	yes	
EHCache	yes	yes	yes	
OSCache	yes	yes	yes	
SwarmCache	yes	yes		
JBoss Cache 1.x	yes			yes
JBoss Cache 2	yes			yes

CitiusTech

https://docs.jboss.org/hibernate/orm/3.3/reference/en/html/performance.html

Query Cache

- Query result sets can also be cached. This is only useful for queries that are run frequently with the same parameters
- First need to enable the query cache:

```
Hibernate.cache.use_query_cache true
```

- This setting creates two new cache regions: one holding cached query result sets (**org.hibernate.cache.StandardQueryCache**)
- The other holding timestamps of the most recent updates to queryable tables (**org.hibernate.cache.UpdateTimestampsCache**)
- Caches only identifier values and results of value type
- Query cache does not cache the state of the actual entities in the result set
- Query cache should always be used in conjunction with the second-level cache

Thank You

CitiusTech Markets

CitiusTech Services

CitiusTech Platforms

Accelerating Innovation

CitiusTech Contacts

Email univerct@citiustech.com

www.citiustech.com