Checklist review today:

* Volatile (fields) // update checklist
  + Visibility
    - guarantee
    - changes made by one thread, data are visible to other threads
  + Atomicity
    - NOT guarantee
    - + CAS (Compare and Swap ->thread safety)
      * Lock free
      * AtomicReference
        + May occur ABA problem
        + AtomicStampedReference can solve
        + compare not only content but also version # can solve.
      * unsafe
      * while loop (for compare)
  + Instruction reordering
    - guarantee
    - may increase parallel execution (by JVM)

---

* Concurrency
  + dirty read: read uncommitted data from another transaction
  + non-repeatable read: read committed data from an update query from another transaction
  + phantom read: read committed data from an insert or delete query from another
  + Isolation level

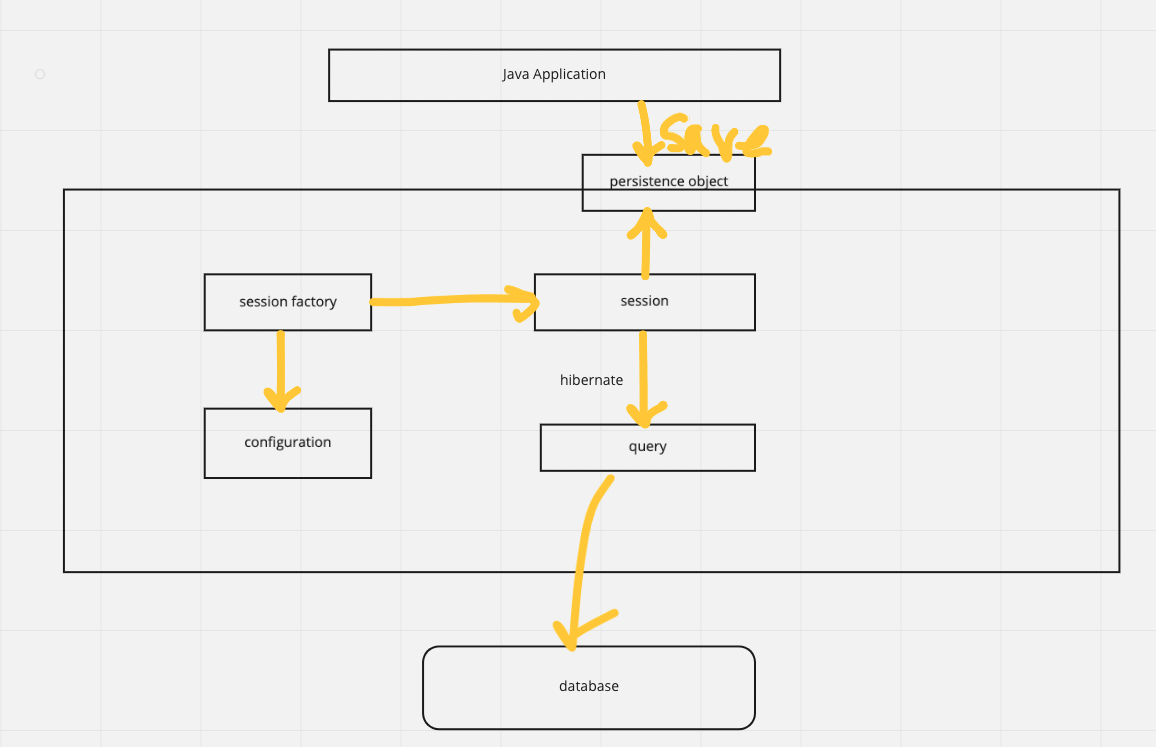
一張含有 桌 的圖片

自動產生的描述

* default in MySQL - Repeatable read
* implements by: Lock + Log (store order of transactions’’ steps)
* Lock
* manually
  + Read Lock: shared lock
    - for example: read same record of transactions simultaneously
  + Write Lock: exclusive lock
    - lock whole table
* handle by MySQL itself
* Intension Lock (IS or IX)
* DeadLock
  + might happened waiting for Lock finish with different order in transactions
  + solve: using top or ps command in Linux sever find key 3306 (default) of CPU time or resources or RAM. check Locks

New things learned today:

* Hibernate
* object–relational mapping (ORM) tool for the Java.
* It enables us to write queries without doing raw SQL as well as gives us some object-oriented control over the queries
* ORM:
  + convert data (sql record) to List <Object>
* Entity (class store table column)
  + @Entity
* Primary key
  + @Id
* Mapping fields to columns
  + @Column(name = “”);
* Session Factory
  + like Connection in JDBC
* Session
  + like Statement in JDBC
* Architecture



* Fetch type
  + eager loading
    - load all as soon as it is executed
    - data too big, we consider lazy loading
  + lazy loading
    - delaying load or initialization of resources until they’re actually needed
* Mapping
  + One to One
  + One to Many
  + Many to One
  + Many to Many
* Cascade
  + When we perform some action on the target entity, the same action will be applied to the associated entity
  + persist
    - save() or persist() operations cascade to related entities.
  + merge
    - related entities are merged when the owning entity is merged.
  + refresh
    - does the same thing for the refresh() operation
  + remove detach
    - remove - removes all related entities association with this setting when the owning entity is deleted
    - detach - detaches all related entities if a “manual detach” occurs
  + all
    - propagates all operations: including Hibernate-specific ones from a parent to a child entity.
* Cache
  + Session - first level cache (open by default)
  + Session factory - second level cache (close by default)
* HQL
  + Hibernate Query Language - similar in appearance to SQL
  + fully object-oriented and understands notions like inheritance, polymorphism and association.
  + Case Sensitivity
* Criteria Queries
  + build up a criteria query object programmatically, where we can apply different kinds of filtration rules and logical conditions
  + Hibernate Criteria API is deprecated since Hibernate 5.2, now focused on the JPA Criteria API
* Native SQL
  + SQL that the data source uses. e.g. Oracle SQL…
  + create a native SQL query from the session with the createSQLQuery() method

Learned Plan tomorrow:

* Review Hibernate
  + default
  + steps
  + fetch type, cascade, mapping, cache…
* Review Keywords
  + every keywords
  + final
    - immutable class
  + static
  + volatile