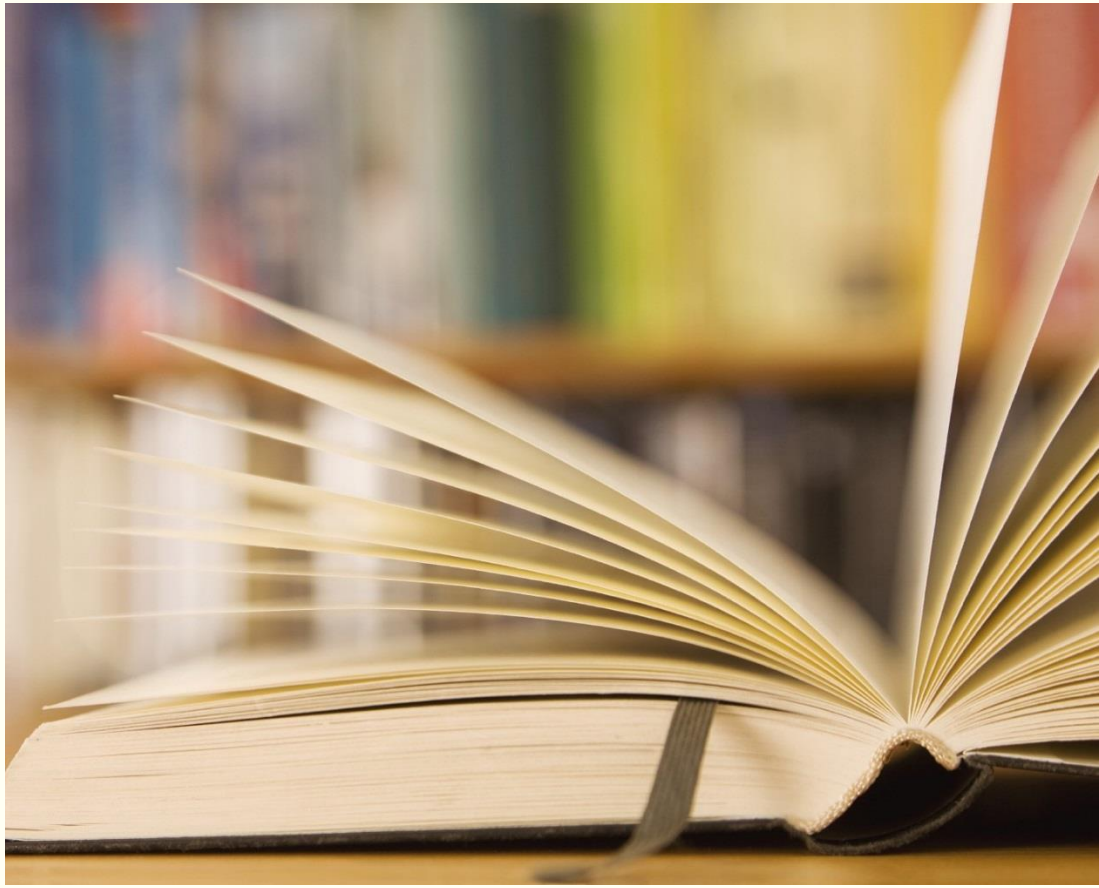
ENTERPRISE ARCHITECTURE

Najeeb Najeeb, PhD



LESSON 04 CONCURRENCY

The problem

- Enterprise application have large number of users. High volume of users results in high chances of collisions. What problems can result from this?
- Data race; incorrect data; incomplete operations.
- Performance issues.
- The solution may also suffer performance limitations?

Entity and em

- Managed entity should be in one em, in one persistence context.
- Merging an entity into two different persistence contexts could produce undefined results. The specs do not address this.
- em.refresh() to synchronize the managed entity with DB.
- Enterprise implies concurrency.
- The longer an entity is in memory the higher chance of change in DB by another process.
- Race conditions
- Locking

OPTIMISTIC LOCKING

Optimistic Locking

- Premise there is a good chance that the transaction is the only one.
- Not acquire a lock on entity until the change is made.
- Locking at the end of the transaction.
- Flushing of transaction checks for data change and possibly throws OptimisticLockException.
- Consequence: re-perform the intended operation. Better than race condition.

Versioning in Optimistic Locking

- Maintain Entity versioning.
- Dedicated field to store version.
- Version stored in DB.
- If entity version > DB version update.
- If entity version <= DB version entity was modified since last read.
- When updating an entity update the version on the entity and DB.
- Best performance.
- @Version

```
@Entity
public class Student {
    @Id private int id;
    @Version private int version;
    private String name;
    private float gpa;

    //  Getters & Setters
}

//  Can be int, short, long, or
java.sql.Timestamp
//  Try to treat it like id.  Do not modify, it
is only read.
```

Recovering from Optimistic Failure

- Maintain a copy of the data and try applying the change.
- Most of the time it may not be possible.
- Inform the user of the data change and request re-entry.
- In a container-based environment most exceptions will be wrapped.
- Treat all transaction exceptions the same and retry the transaction from the beginning or indicate to the user they must restart and retry.

Main Point

- With the increase of users, the chances of data collision increase. When the probability of collisions is low optimistic locking is more attractive. Optimistic locking has a low overhead and is the default supported locking technique. On the other hand, recovery from collisions is expensive.