**Youtube:** [**https://www.youtube.com/watch?v=1fQtFALX80w&list=PLzS3AYzXBoj-H1SJxp2RuMMS4xUWrPV\_3&ab\_channel=KKJavaTutorials**](https://www.youtube.com/watch?v=1fQtFALX80w&list=PLzS3AYzXBoj-H1SJxp2RuMMS4xUWrPV_3&ab_channel=KKJavaTutorials)

1. **Vd1**

ACID:

* Automic: all or nothing is done
* Consistent: no foreign key constraint is violated (one or no table is updated)
* Isolated
* Durable

1. **Isolation level**

Isolation define how changes from 1 transaction affect other transaction

* Define isolation level for a transactional method means define its read properties

3 problems with non-transactional method when reading data:

* Dirty reading : trans A write x, Trans B read x, Trans A rollback => Trans B read dirty
* Non-repeatable reading: trans A reads x, Trans B writes x, trans A reads x => trans A read non-repeatable data
* Phantom reading: a bizard data appear in the next reading because other trans inserted new data

Isolation properties inside @Transactional to avoid those problem:

* READ\_UNCOMMITED => no problems solved
* READ\_COMMITED => prevent dirty read
* READ\_REPEATABLE => prevent dirty + non\_repeatable
* SERIALIZABLE:=> prevent all

Using strict isolation level like SERIALIZABLE may affect performance because it block other concurrent transactions from happening ( the update, insert transaction will not executed), READ\_REPEATABLE prevent update transaction,

Default is REPEATABLE

1. **Propergation**

If both nested method and outer method use:

* REQUIRED => both use the same trans, if nested throw UNCHECKED exceptions then outer rollback
* REQUIRED\_NEW:

Inner is independent of outer