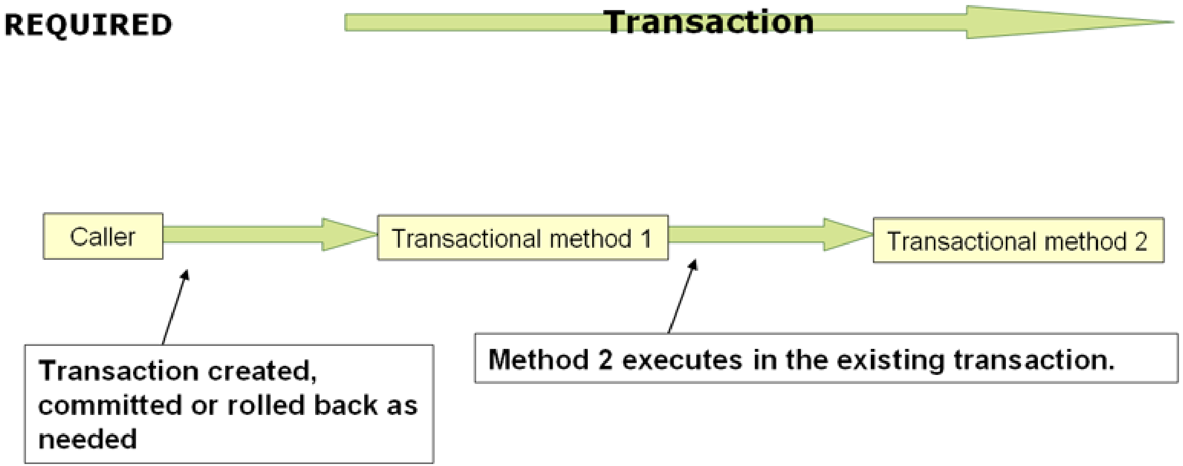
# SPRING TRANSACTIONS

ADVANCED CONCEPTS

MARCH, 2015

# Transaction propagation



REQUIRED

Transaction

Caller

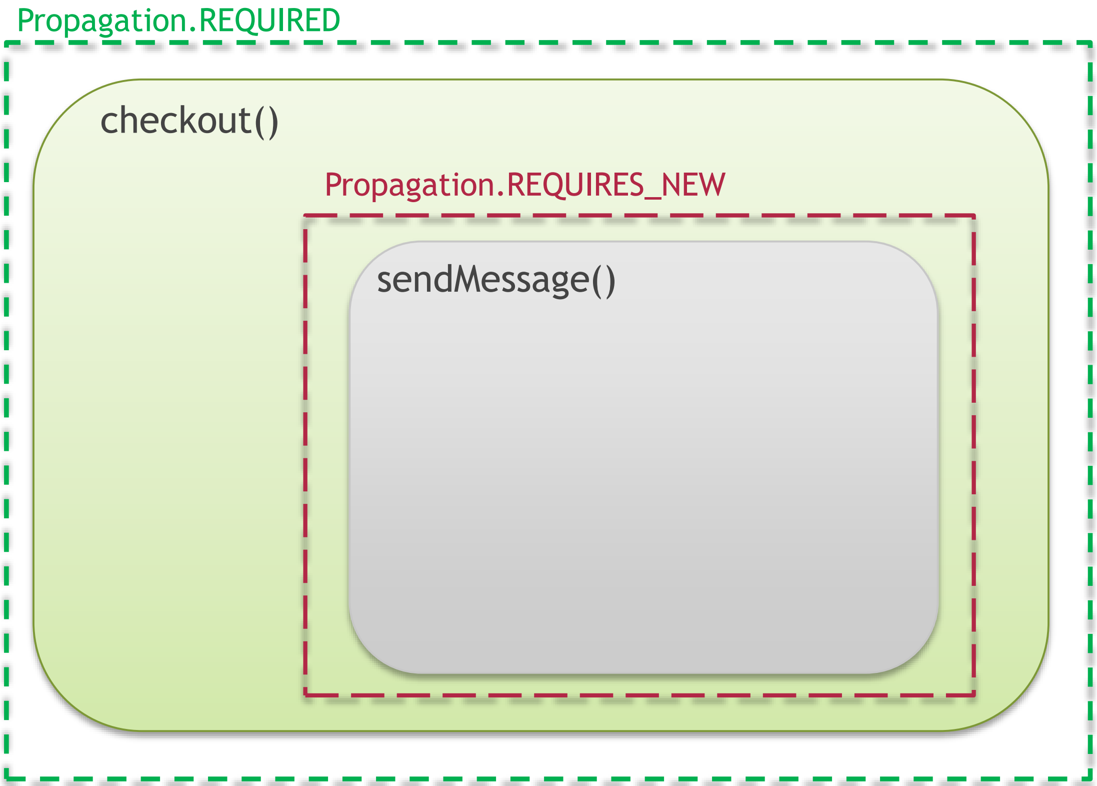
Transactional method 1

Transactional method 2

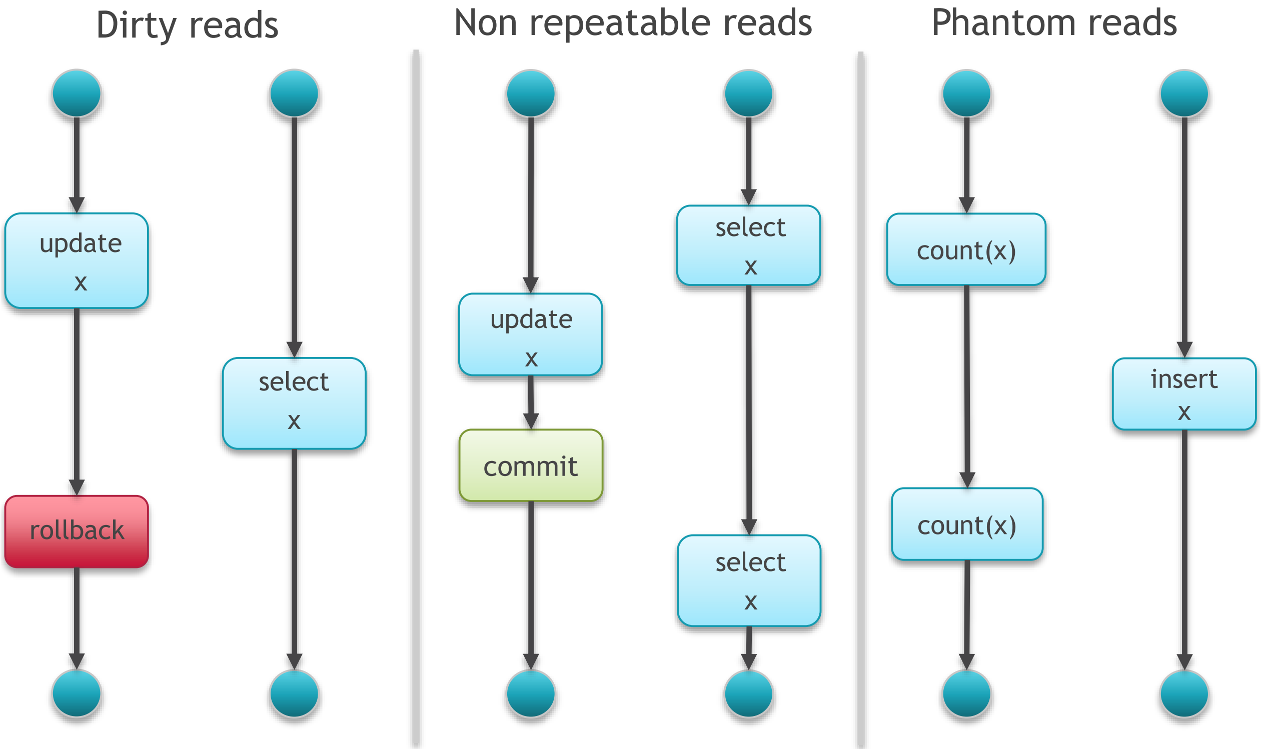
Transaction created, committed or rolled back as needed

Method 2 executes in the existing transaction.

# Transaction propagation issues



# Transaction isolation

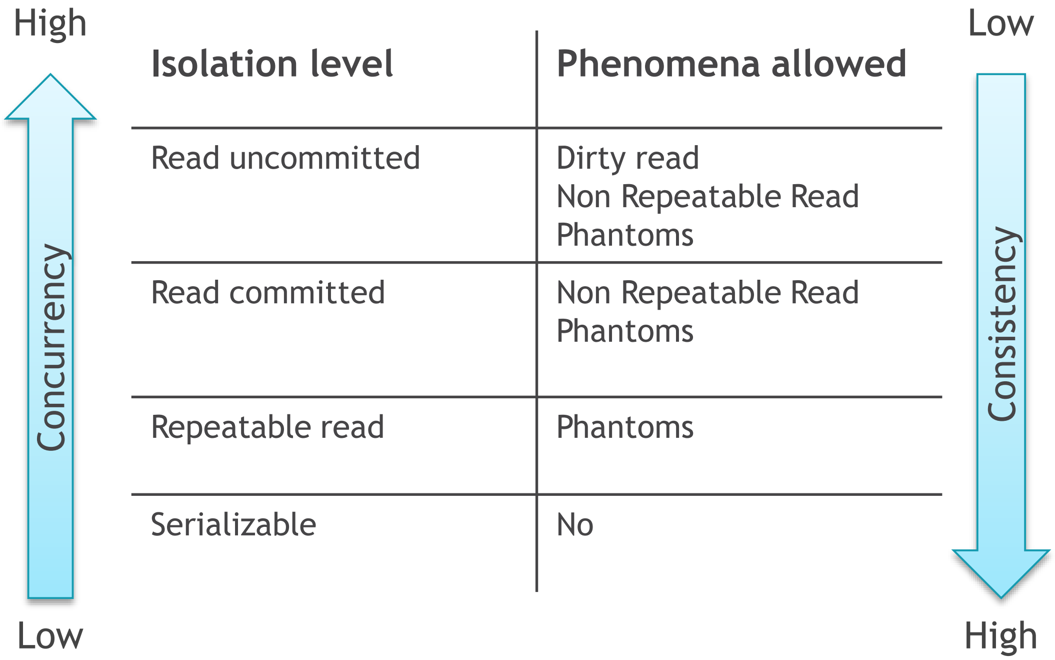


Dirty reads

Non repeatable reads

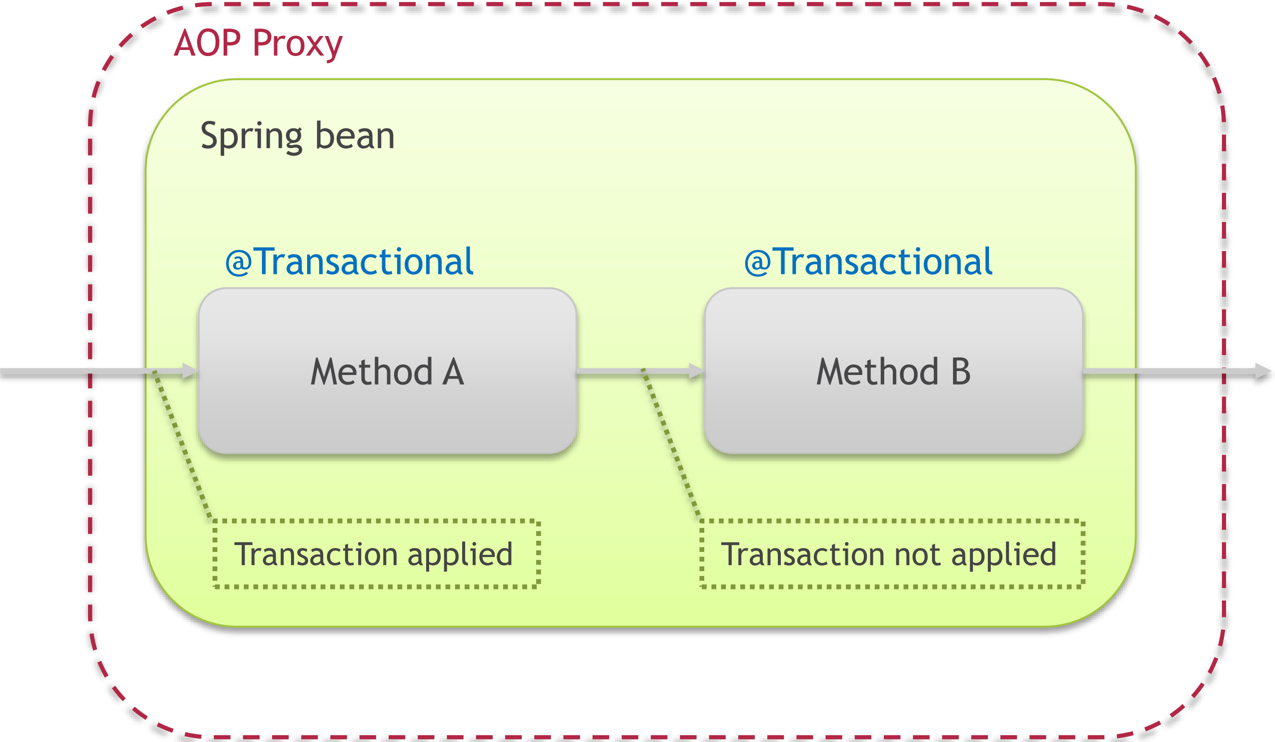
Phantom reads

# Consistency vs Concurrency



| Isolation level | Phenomena allowed |
|---|---|
| Read uncommitted | Dirty read<br>Non Repeatable Read<br>Phantoms |
| Read committed | Non Repeatable Read<br>Phantoms |
| Repeatable read | Phantoms |
| Serializable | No |

# Local transactional calls



# Transactional methods visibility

```
// this one will work
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
  public void updateFoo(Foo foo) { // do something }

// this one will be ignored
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
  protected void updateFoo2(Foo foo) { // do something  }

// this one will be ignored
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
  void updateFoo3(Foo foo) { //do something }

// this one will be ignored
@Transactional(readOnly = false, propagation = Propagation.REQUIRED)
  private void updateFoo4(Foo foo) { //do something }
```

# THANK YOU!

**MAKSYM_GOVORISCHEV@EPAM.COM**

MARCH, 2015