



Ildar92 August 4, 2020 at 09:32 AM

# Spring transaction management. Isolation and

Programming, Java

## Introduction

In my opinion transaction management is a really important topic for each backend developer while using Spring framework.

But I think, it is important to know how to use transactions properly. Because sometimes can be thrown inside your method, but transaction was not rolled back and it is not clear why? Or some

## @Transactional

In spring there is @Transactional annotation that can be used for wrapping a method in a transaction (lowest priority), classes or certain methods (highest priority).

@Transactional is applied only for public methods and method must be invoked from outside

@Transactional annotation has multiple parameters. I would like to focus on two of them: Isolation

**Isolation** is one of the main properties of transactions (Atomicity, Consistency, Isolation, Durability). It describes visibility of changes applied by concurrent transactions to each other.

In Spring it is possible to set one of the 5 isolation level values: DEFAULT, READ\_UNCOMMITTED, READ\_COMMITTED, REPEATABLE\_READ and SERIALIZABLE.

For example,

Ads



## Русский язык для всех

Научись анализировать правила орфографии, пунктуации и применять их на практике

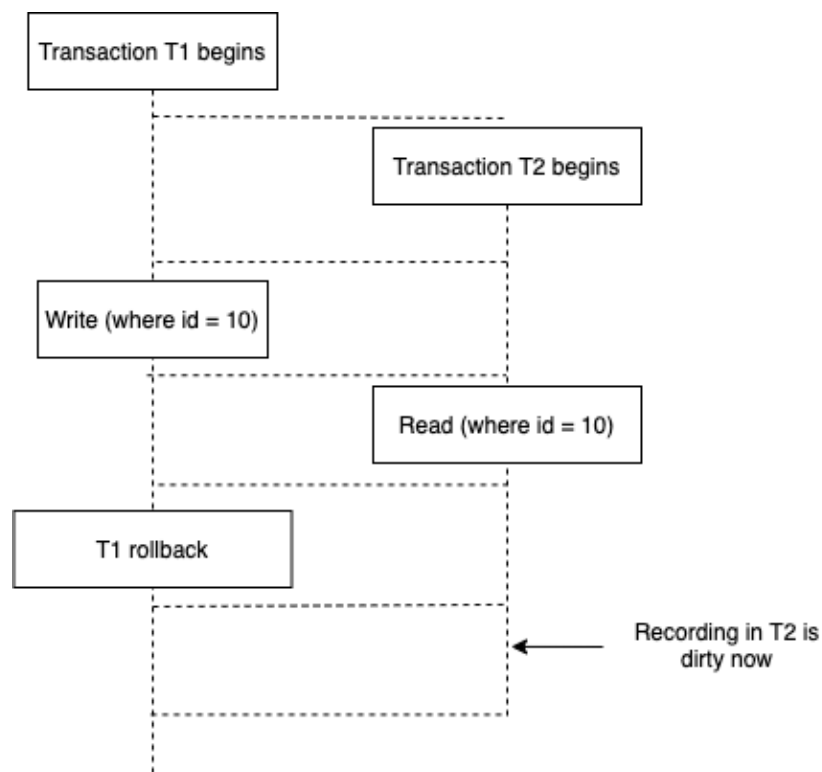
[Узнать больше](#)[fondfilm.ru](http://fondfilm.ru)

```
@Transactional (isolation=Isolation.READ_COMMITTED)
```

Each of these isolation levels may have or haven't different side effects: "dirty" read, non-repeatable read and phantom read. What each of them means?

**"Dirty" read** — one transaction can read changes of a concurrent transaction, that were not committed yet.

If you like more diagrams as I do:

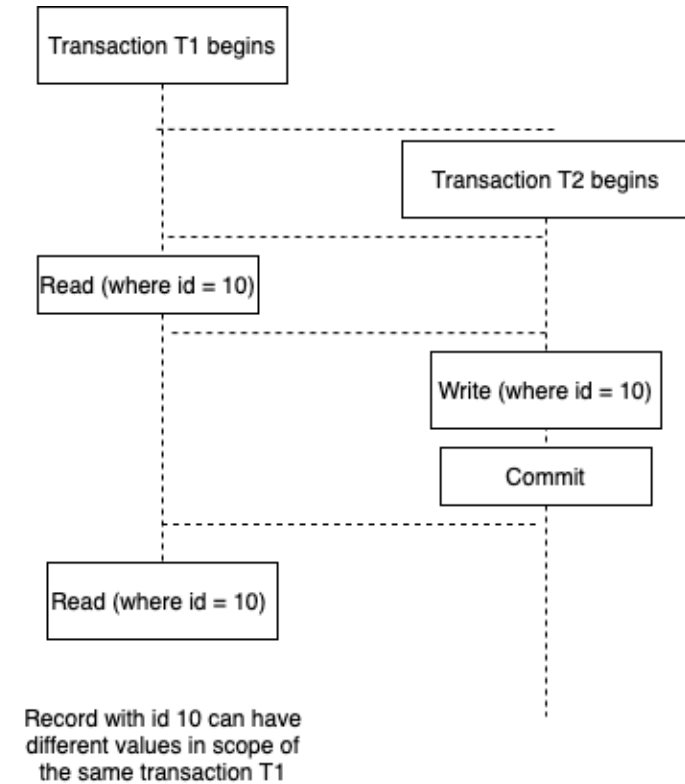


Transaction T1 begins first, then we start transaction T2.

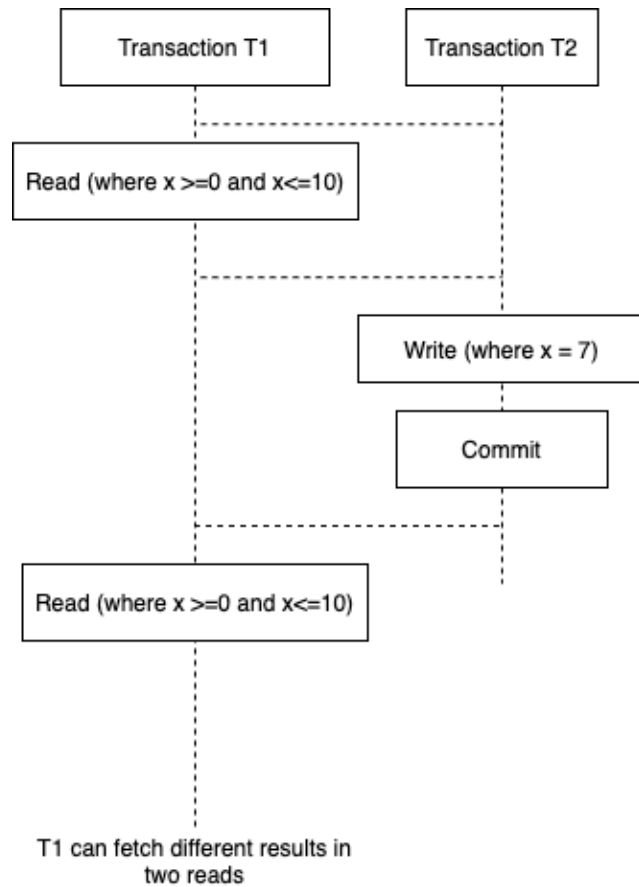
After that T1 changes row with id=10 in database and T2 reads it. Something wrong happens and T1 is rolled back. And recording in T2

is dirty now.

**Non-repeatable read** — one transaction reads the same row twice, but gets different values because between these reads the data was updated by the concurrent transaction.



**Phantom read** — one transaction runs the same query twice, but gets a different set of rows as result, because of the changes applied by another concurrent transaction.



Let's go back to the isolation levels and check their possible side effects.

1. DEFAULT value is used when we want to use default isolation level of our RDBMS. Default value for PostgreSQL, Oracle and SQL server is READ\_COMMITTED, for MySQL — REPEATABLE\_READ.
2. with READ\_UNCOMMITTED isolation level, we can have all three side effects
3. READ\_COMMITTED isolation level has one change in comparison with READ\_UNCOMMITTED — it prevents "dirty" reads.

4. REPEATABLE\_READ prevents "dirty" and non-repeatable reads.

5. SERIALIZABLE isolation level prevents all mentioned above side effects. It performs all transactions in sequence.



Isolation level	Phantom read	Non-repeatable read	"Dirty" read
READ_UNCOMMITTED	possible	possible	possible
READ_COMMITTED	possible	possible	not possible
REPEATABLE_READ	possible	not possible	not possible
SERIALIZABLE	not possible	not possible	not possible

There is also another important parameter of @Transactional: propagation.

Propagation can be set to REQUIRED, REQUIRES\_NEW, MANDATORY, SUPPORTS, NOT\_SUPPORTED, NESTED or NEVER.

Example:

All streams Development Administrating Design Management Marketing PopSci

Log in

Sign up

1. REQUIRED propagation level uses an existing transaction if there is one. Otherwise creates a new transaction.

2. REQUIRES\_NEW propagation level says to suspend outer transaction (if there is one) and create a new one.

3. MANDATORY propagation uses an existing transaction if there is one. Otherwise, an exception will be thrown.

4. SUPPORTS propagation level uses the current transaction if there is one. Otherwise, doesn't create a new one.
5. NOT\_SUPPORTED suspends current transaction if there is one.
6. NESTED creates nested transaction when there is an existing transaction already or works like REQUIRED if there is no transaction.
7. NEVER throws an exception if there is an active transaction.

Propagation	Calling method (outer)	Called method (inner)
REQUIRED	No	T2
REQUIRED	T1	T1
REQUIRES_NEW	No	T2
REQUIRES_NEW	T1	T2
MANDATORY	No	Exception
MANDATORY	T1	T1
NOT_SUPPORTED	No	No
NOT_SUPPORTED	T1	No
SUPPORTS	No	No

Propagation	Calling method (outer)	Called method (inner)
SUPPORTS	T1	T1
NEVER	No	No
NEVER	T1	Exception
NESTED	No	T2
NESTED	T1	T2

Also interesting is that Spring framework does automatic transaction rollback only for unchecked (Runtime) exceptions. To change it we can use `rollbackFor` parameter.

For example, we can put

```
@Transactional (rollbackFor=Exception.class)
```

## Conclusion

In the article, I tried to describe how to use **Isolation** and **Propagation** parameters of `@Transactional` in Spring.

I remember how several years ago I had some problems with the chain of transactional method and I hope that my article can help other developers to avoid them and have more clear understanding of the topic.

**Tags:** [java](#), [spring](#), [transactions](#)

**Hubs:** [Programming](#), [Java](#)

↑ +6 ↓ 7 6k Comment Share

**20.0**

Karma

**0.0**

Rating

**Ildar Nazmeev** @Ildar92

Пользователь

## SIMILAR POSTS

March 15, 2021 at 11:44 AM

### A tiny Rate Limiter Library for Spring MVC

↑ 3    👁 350    📖 0    💬 0

November 21, 2020 at 06:30 PM

### Spring Boot app with Apache Kafka in Docker container

↑ 1    👁 5.9k    📖 7    💬 1

August 26, 2020 at 07:09 PM

### Delayed queue in Java and Redis

↑ 1    👁 1.9k    📖 2    💬 0

Ads





No Design Skills Needed

 Doodly

[Open](#)

## Comments 0

Only users with [full accounts](#) can post comments. [Log in](#), please.

### TOP POSTS

Day

Week

Month

**Audio over Bluetooth: most detailed information about profiles, codecs, and devices**

↑ +22    👁 176k    📌 15    💬 9

**Java vs .Net: Who will Reign in the Future?**

↑ +3    👁 30.5k    📌 5    💬 3

Ugly API

↑ +1    👁 590    📌 0    💬 0

How to cook RTSP on your website in 2020, or why the boars will not have a chance to run away

↑ +4    👁 29.1k    📌 3    💬 2

Your account

[Log in](#)

[Sign up](#)

Sections

[Posts](#)

[Hubs](#)

[Companies](#)

[Users](#)

[Sandbox](#)

Info

[How it works](#)

[For Authors](#)

[For Companies](#)

[Documents](#)

[Agreement](#)

[Terms of service](#)

Services

[Ads](#)

[Subscription plans](#)

[Content](#)

[Seminars](#)

[Megaprojects](#)

