


Spring @Transaction Propagation Isolation



The developer has the ability to decide how the business methods should be encapsulated in both logical or physical transactions.

the following are the different propagation behaviours provided by Spring.



TRANSACTION *levels*

REQUIRED:

Code will always run in a transaction. Create a new transaction or reuse one if available.

REQUIRES_NEW:

Code will always run in a new transaction. Suspend current transaction if one exist.

NESTED:

The **NESTED** makes nested Spring transactions to use the same physical transaction but sets savepoints between nested invocations so inner transactions may also rollback independently of outer transactions. This may be familiar to JDBC aware developers as the savepoints are achieved with JDBC savepoints, so this behaviour should only be used with Spring JDBC managed transactions



MANDATORY:

The **MANDATORY** behaviour states that an existing opened transaction must already exist.
If not an exception will be thrown by the container.

NEVER:

The **NEVER** behaviour states that an existing opened transaction must **not** already exist.
If a transaction exists an exception will be thrown by the container.



NOT_SUPPORTED:

The **NOT_SUPPORTED** behaviour will execute outside of the scope of any transaction. If an opened transaction already exists it will be paused.

SUPPORTS:

The **SUPPORTS** behaviour will execute in the scope of a transaction if an opened transaction already exists. If there isn't an already opened transaction the method will execute anyway but in a non-transactional way.



ISOLATION *levels*

Dirty reads:

occur when transaction reads an uncommitted data.

Non-repeatable Reads:

occurs when a transaction reads the same data multiple times but gets different results each time.

Phantom Reads:

occur when two transactions work on a same row where one updates and other reads. The reading transaction get new data.

ISOLATION attributes

ISOLATION_DEFAULT

- default isolation specific to the data source.

ISOLATION_READ_UNCOMMITTED

- Read changes that are uncommitted. Leads to dirty reads, Phantom reads and non repeatable reads.

ISOLATION_READ_COMMITTED:

- Reads only committed data. Dirty read is prevented but repeatable and non repeatable reads are possible.
-



ISOLATION_REPEATABLE_READ:

Multiple reads of same field yield same results unless modified by same transaction. Dirty and non repeatable reads are prevented but phantom reads are possible as other transactions may edit the fields.

ISOLATION_SERIALIZABLE:

Dirty, phantom and non repeatable reads are prevented. But hampers the performance of application.

Dirty reads:

A *dirty read* occurs when a transaction is allowed to read data from a row that has been modified by another running transaction and not yet committed.

Transaction 1

```
/* Query 1 */ SELECT age FROM users  
WHERE id = 1; /* will read 20 */
```

```
/* Query 1 */ SELECT age FROM users  
WHERE id = 1; /* will read 21 */
```

Transaction 2

```
/* Query 2 */ UPDATE users SET age =  
21 WHERE id = 1; /* No commit here */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

Non-repeatable reads:

A *non-repeatable read* occurs, when during the course of a transaction, a row is retrieved twice and the values within the row differ between reads.

Transaction 1

```
/* Query 1 */ SELECT * FROM  
users WHERE id = 1;
```

```
/* Query 1 */ SELECT * FROM  
users WHERE id = 1; COMMIT; /*  
lock-based REPEATABLE READ */
```

Transaction 2

```
/* Query 2 */ UPDATE users SET age  
= 21 WHERE id = 1; COMMIT; /*  
in multiversion concurrency control,  
or lock-based READ COMMITTED */
```


Phantom reads:

A *phantom read* occurs when, in the course of a transaction, new rows are added by another transaction to the records being read.

Transaction 1

```
/* Query 1 */ SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;
```

```
/* Query 1 */ SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;  
COMMIT;
```

Transaction 2

```
/* Query 2 */ INSERT INTO  
users(id,name,age) VALUES ( 3, 'Bob', 27  
) COMMIT;
```

Isolation Levels, read phenomena, and locks

Isolation level	Dirty reads	Non-repeatable reads	Phantoms
Read Uncommitted	may occur	may occur	may occur
Read Committed	don't occur	may occur	may occur
Repeatable Read	don't occur	don't occur	may occur
Serializable	don't occur	don't occur	don't occur