

Understanding isolation levels

08/12/2019 • 2 minutes to read •  +2

In this article

[Remarks](#)

[See also](#)



[Download JDBC Driver](#)

Transactions specify an isolation level that defines the degree to which one transaction must be isolated from resource or data modifications made by other transactions. Isolation levels are described in terms of which concurrency side effects, such as dirty reads or phantom reads, are allowed.

Transaction isolation levels control the following:

- Whether locks are taken when data is read, and what type of locks are requested.
- How long the read locks are held.
- Whether a read operation referencing rows modified by another transaction:
 - Block until the exclusive lock on the row is freed.
 - Retrieve the committed version of the row that existed at the time the statement or transaction started.
 - Read the uncommitted data modification.

Choosing a transaction isolation level doesn't affect the locks that are acquired to protect data modifications. A transaction always gets an exclusive lock on any data it modifies and holds that lock until the transaction completes, regardless of the isolation level set for that transaction. For read operations, transaction isolation levels primarily define the level of protection from the effects of modifications made by other transactions.

A lower isolation level increases the ability of many users to access data at the same time, but increases the number of concurrency effects, such as dirty reads or lost updates, that users might encounter. Conversely, a higher isolation level reduces the types of concurrency effects that users might encounter, but requires more system resources and increases the chances that one transaction will block another. Choosing the appropriate isolation level depends on balancing the data integrity requirements of the application against the overhead of each isolation level. The highest isolation level, serializable, guarantees that a transaction will retrieve exactly the same data every time it repeats a read operation, but it does this by performing a level of locking that is likely to impact other users in multi-user systems. The lowest isolation level, read uncommitted, can retrieve data that has been modified but not committed by other transactions. All concurrency side effects can happen in read uncommitted, but there's no read locking or versioning, so overhead is minimized.

Remarks


The following table shows the concurrency side effects allowed by the different isolation levels.

Isolation Level	Dirty Read	Non Repeatable Read	Phantom
Read uncommitted	Yes	Yes	Yes
Read committed	No	Yes	Yes
Repeatable read	No	No	Yes
Snapshot	No	No	No


Isolation Level	Dirty Read	Non Repeatable Read	Phantom
Serializable	No	No	No

Transactions must be run at an isolation level of at least repeatable read to prevent lost updates that can occur when two transactions each retrieve the same row, and then later update the row based on the originally retrieved values. If the two transactions update rows using a single UPDATE statement and don't base the update on the previously retrieved values, lost updates can't occur at the default isolation level of read committed.


To set the isolation level for a transaction, you can use the [setTransactionIsolation](#) method of the [SQLServerConnection](#) class. This method accepts an **int** value as its argument, which is based on one of the connection constants as in the following:

Java	 Copy
<pre>con.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED);</pre>	

To use the new snapshot isolation level of SQL Server, you can use one of the `SQLServerConnection` constants:

Java	 Copy
<pre>con.setTransactionIsolation(SQLServerConnection.TRANSACTION_SNAPSHOT);</pre>	

or you can use:

Java	 Copy
<pre>con.setTransactionIsolation(Connection.TRANSACTION_READ_COMMITTED + 4094);</pre>	

For more information about SQL Server isolation levels, see "Isolation Levels in the Database Engine" in SQL Server Books Online.

See also

[Performing transactions with the JDBC driver](#)

Is this page helpful?

 Yes  No

Recommended content

[SET TRANSACTION ISOLATION LEVEL \(Transact-SQL\) - SQL Server](#)

SET TRANSACTION ISOLATION LEVEL (Transact-SQL)

[Transaction Locking and Row Versioning Guide - SQL Server](#)

Transaction Locking and Row Versioning Guide

[Display an Actual Execution Plan - SQL Server](#)

Learn how to generate actual graphical execution plans by using SQL Server Management Studio. An actual graphic...

[Transaction Isolation Levels \(ODBC\) - SQL Server](#)

Transaction Isolation Levels (ODBC)

Show more 