Transactions and Isolation Levels

.NET

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

https://docs.microsoft.com/en-us/sql/connect/jdbc/understanding-isolation-levels?view=sql-server-ver15

Isolation Level and Read Errors

https://www.geeksforgeeks.org/transaction-isolation-levels-dbms/#:~:text=

Isolation levels define the degree to which a transaction must be isolated from other data modifications made by any other transaction. A ***transaction isolation level*** is determined by its permissiveness of the following.

- ***Dirty Read*** – When a transaction reads data that is new and has not yet been committed by another transaction.
- ***Non-Repeatable read*** – When a transaction reads the same row twice and gets a different value each time.
- ***Phantom Read*** – When two identical queries are executed but the rows retrieved by the two are different.

Isolation Levels

https://docs.microsoft.com/en-us/sql/connect/jdbc/understanding-isolation-levels?view=sql-server-ver15

Transaction isolation levels control the following:

- If ***locks***, and what type, are enacted when data is read.
- How long read-locks are held.
- If a read operation references 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.

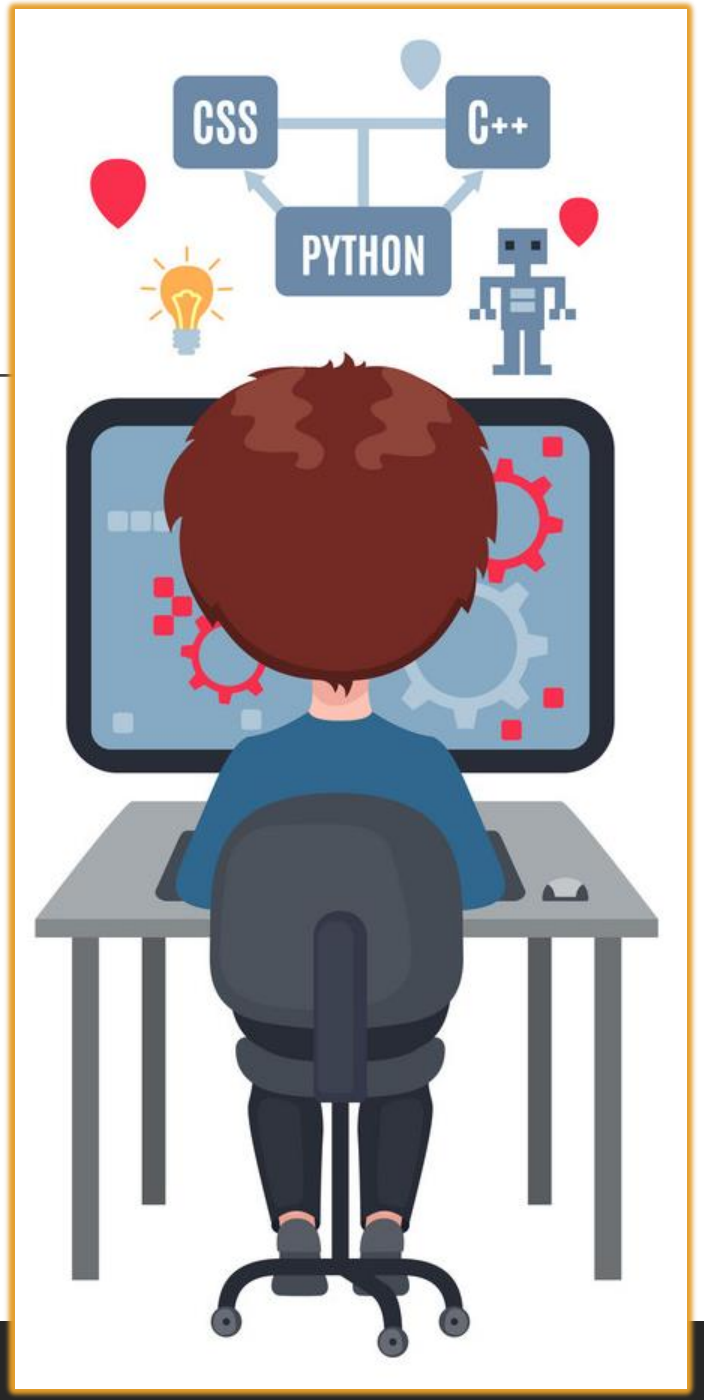
Choosing a transaction isolation level doesn't affect the locks that are acquired to protect data modifications. A transaction always gets an exclusive lock on any data it modifies and holds that lock until the transaction completes, regardless of its isolation level.

Isolation Levels – Serializable

https://sqlperformance.com/2015/04/t-sql-queries/the-read-uncommitted-isolation-level

The only transaction isolation level that provides complete isolation from concurrency effects is ***serializable***. The SQL Server implementation of the serializable isolation level means a transaction will see the latest committed data. The set of data encountered under ***serializable*** isolation is guaranteed not to change before the transaction ends.

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

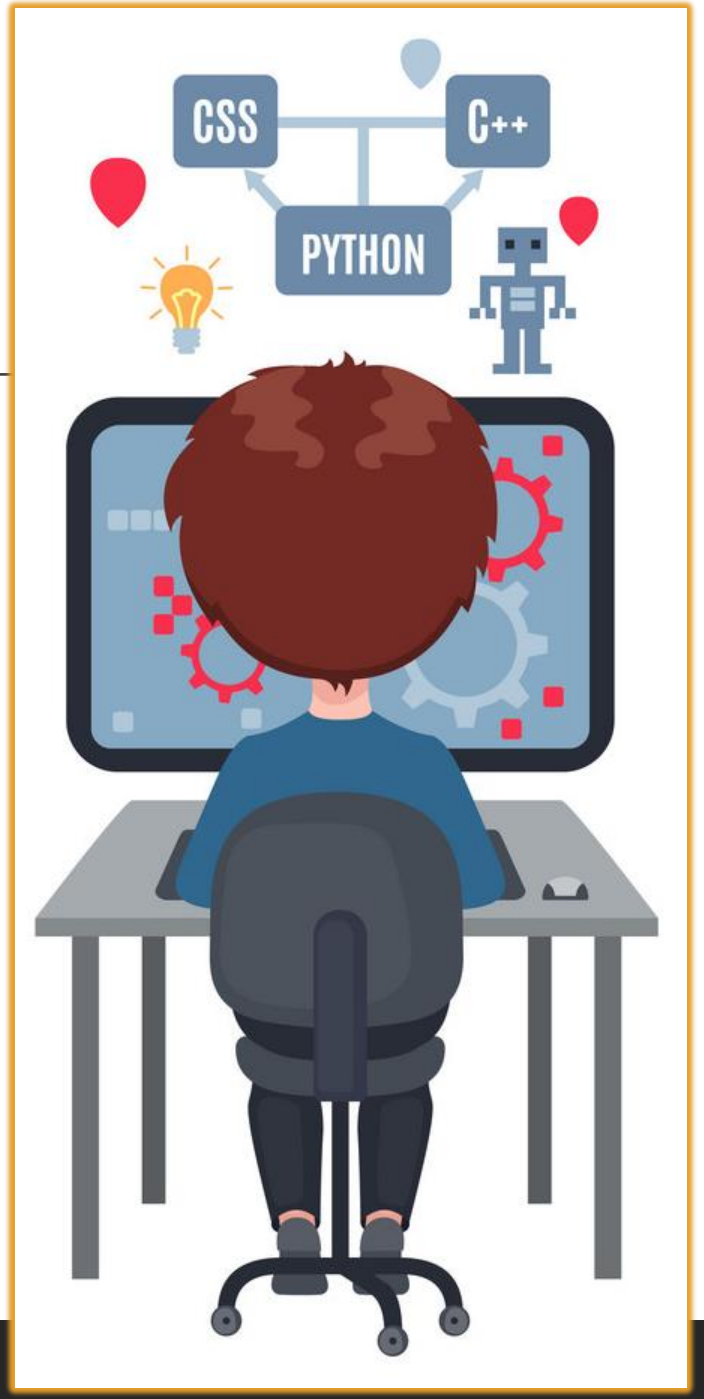


Isolation Levels – Repeatable-Read

https://sqlperformance.com/2015/04/t-sql-queries/the-read-uncommitted-isolation-level

The ***Repeatable Read*** isolation level is guaranteed to read only committed data. If two identical queries are enacted simultaneously, they may return different results.

Isolation Level	Dirty Read	Non Repeatable Read	Phantom
Repeatable read	No	No	Yes

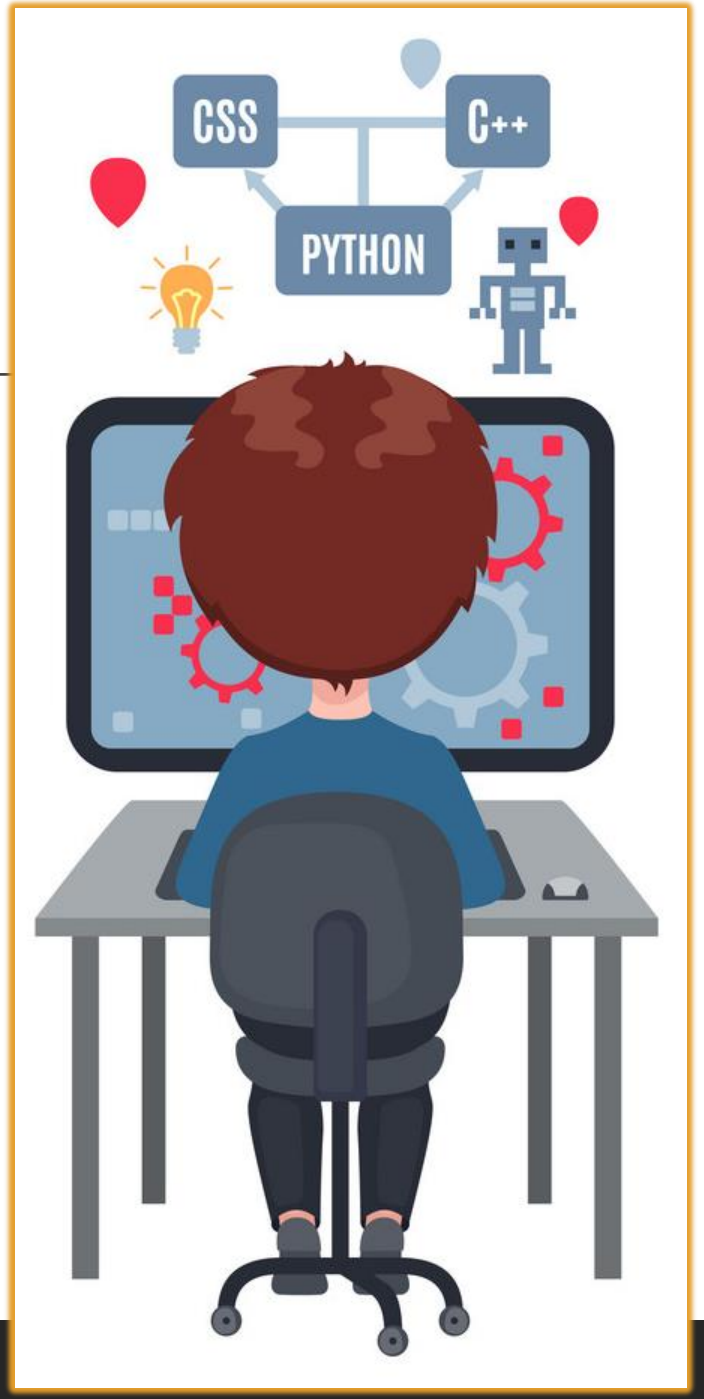


Isolation Levels – Read-Committed

https://sqlperformance.com/2015/04/t-sql-queries/the-read-uncommitted-isolation-level

Read committed can see committed data from different points in time – even for a single row if, for example, the query plan uses techniques like index intersection.

Isolation Level	Dirty Read	Non Repeatable Read	Phantom
Read committed	No	Yes	Yes

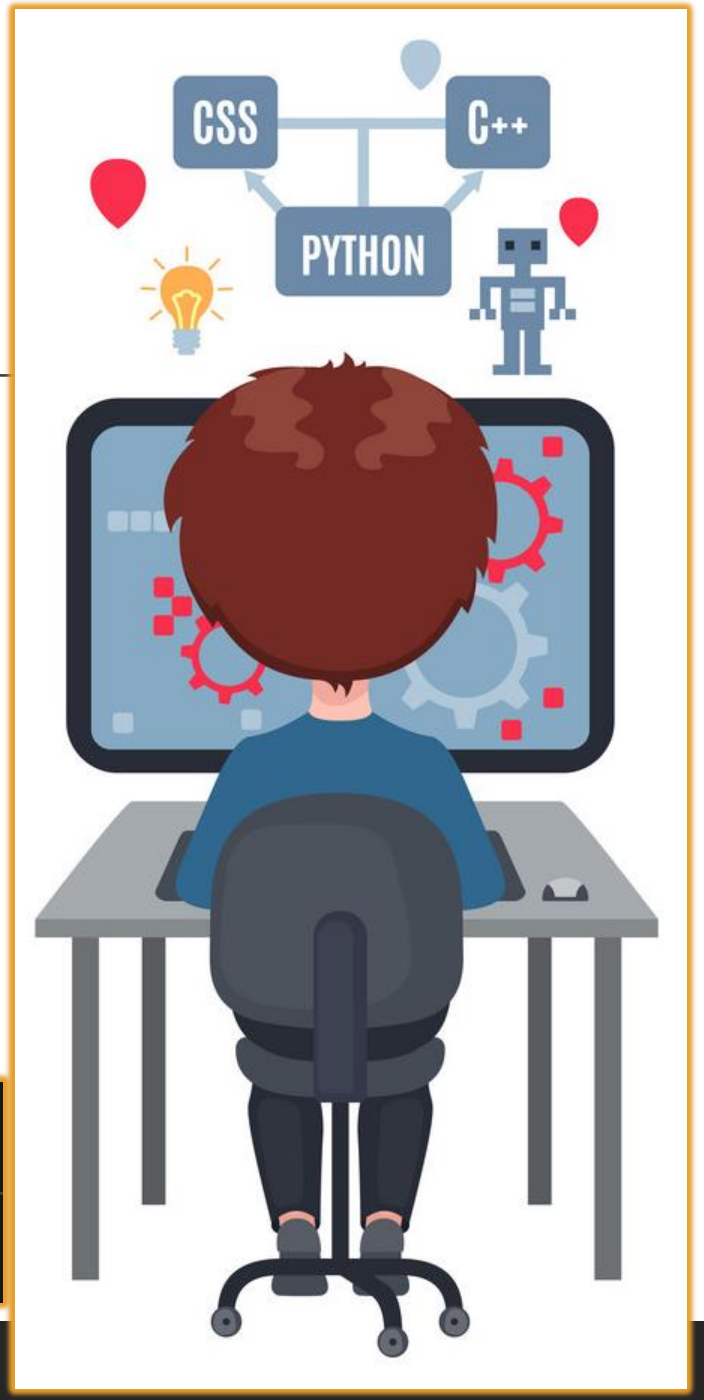


Isolation Levels – Read-Uncommitted

https://sqlperformance.com/2015/04/t-sql-queries/the-read-uncommitted-isolation-level

Read uncommitted is the weakest of the four transaction isolation levels. It allows all three "concurrency phenomena"; ***dirty reads***, ***non-repeatable reads***, and ***phantom reads***.

Isolation Level	Dirty Read	Non Repeatable Read	Phantom
Read uncommitted	Yes	Yes	Yes



Isolation Levels – In context

https://docs.microsoft.com/en-us/sql/connect/jdbc/understanding-isolation-levels?view=sql-server-ver15#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
Serializable	No	No	No

Transactions – Commit/Rollback/Savepoint

https://www.techopedia.com/definition/16/commit#:~:text=

- ***Commit*** - Refers to the saving of data permanently after a set of tentative changes. A commit ends a transaction within a Relational Database and allows all other users to see the changes.
- ***Rollback*** - In database technologies, a rollback is an operation which returns the database to some previous state. Rollbacks are important for database integrity, because they mean that the database can be restored to a clean copy even after erroneous operations are performed.
- ***Savepoint*** - A way of implementing nested transactions within a Relational Database Management System by indicating a point within a transaction that can be "rolled back to" without affecting any work done in the transaction before the ***Savepoint*** was created. Multiple ***Savepoints*** can exist within a single transaction.