Degrees of Isolation

Every transaction has three characteristics: **diagnostics_size**, **access_mode**, and **isolation_level**.

Diagnostics_size:

The diagnostics_size determines the number of error condition that can be recorded for the transaction.

Access_mode:

There are two access_modes: READ ONLY and READ WRITE.

If the access_mode is READ ONLY, the transaction is not allowed to modify the database. Thus INSERT, DELETE, UPDATE and CREATE statements cannot be executed. For transactions with READ ONLY access_mode, only shared locks need to be obtained, thereby increasing concurrency.

If we have to execute one of commands INSERT, DELETE, UPDATE or CREATE, the access_mode should be set to READ WRITE.

Isolation_levels

Most systems do not provide automatically serializability!!

- Implementors did not understand the issues
- Implementors make a compromise between correctness and performance and provide options called ***levels of isolation*** (or degrees of isolation)

The isolation_level controls the extent to which a given transaction is exposed to the actions of other transactions executing concurrently. By choosing one of four possible isolation_level settings, a user can obtain greater concurrency at the cost of increasing the transaction's exposure to other transaction's uncommitted changes.

In SQL-92 the isolation levels are:

- READ UNCOMMITTED
- READ COMMITTED
- REPEATABLE READ
- SERIALIZABLE

Isolation_levels

- SERIALIZABLE – this isolation level ensures that T reads only the changes made by committed transactions, that no value read or writtten by T is changed by any other transaction until t is complete.

  In terms of lock-based implementation, a SERIALIZABLE means that the lock algorithm is two-phase and well formed.

- REPEATABLE READ - this isolation level ensures that T reads only the changes made by committed transactions, that no value read or writtten by T is changed by any other transaction until t is complete. However, T could experience the phantom phenomenon.

  In terms of lock-based implementation, a REPEATABLE READ means that the lock algorithm is two-phase and well formed.
  A REPEATABLE READ uses the same locking protocol as a SERIALIZABLE transaction except that it does not do index locking – it locks only individual objects - not sets of objects.

Isolation levels

- READ COMMITTED (*cursor stability*) - this isolation level ensures that T reads only the changes made by committed transactions, that no value writtten by T is changed by any other transaction until t is complete. However, a value read by T may well be modified by another transaction while T is in progress, and T is exposed to the phantom phenomenon.

  In terms of lock-based implementation, a READ COMMITTED means that the lock algorithm is two-phase with respect to write locks and well formed with respect to reads. In other words, all shared locks obtained by T are released immediately.

- READ UNCOMMITTED (browse)- T can read changes made to an object by an ongoing transaction. Moreover, the object can be changed further while T is in progress, and T is exposed to the phantom phenomenon.

  In terms of lock-based implementation, a READ UNCOMMITTED means a transaction T obtains write locks before writing data items, and holds these locks until the end, but does not obtain shared locks before reading data items.

  READ UNCOMMITTED is allowed only for read-only transactions – a transaction is required to have an access mode of READ ONLY.

Isolation levels

Isolation Level	**Dirty Read**	**Unrepeatable Read**	**Phantom**
READ UNCOMMITTED	maybe	maybe	maybe
READ COMMITTED	no	maybe	maybe
REPEATABLE READ	no	no	maybe
SERIALIZABLE	no	no	no

Why READ COMMITTED is called sometimes *Cursor Stability*?

```
exec sql select balance
        into :balance
        from account
        where account_id=:id;

balance=balance+10;

exec sql update account
        set balance=:balance
        where account_id=:id;
```

```
exec sql declare cursor c for
        select balance
        from account
        where account_id=:id;

exec sql open cursor c
exec sql fetch c into :balance
balance=balance+10;

exec sql update account
        set balance=:balance
        where current of cusor c;
exec sql close c;
```

- Most SQL-systems keep a shared lock on the record currently addressed by a cursor.

The isolation and access-mode can be set using the SET TRANSACTION command. The following command declares the current to be SERIALIZABLE and READ ONLY:

```
SET TRANSACTION ISOLATION LEVEL
SERIALIZABLE READ ONLY
```