Transactions

A transaction is a change in the state of the database that usually combines multiple database operations. These are units or sequences of work accomplished in a logical order, whether in a manual fashion by a user or automatically by some sort of a database program.

Transactions can be used to ensure that the change of state is *atomic* and *valid* in nature. All transactions must follow four standard properties which are referred to as **A.C.I.D**.

A.C.I.D

- **A**tomicity: ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **C**onsistency: ensures that the database properly changes states upon a successfully committed transaction.
- **I**solated: enables transactions to operate independently of and transparent to each other.
- **D**urability: ensures that the result or effect of a committed transaction persists in case of a system failure.

Transaction Control Language (TCL)

The following commands are used to control transactions.

- `BEGIN` - Begins a transaction
- `COMMIT` - Commits all changes to permanency
- `SAVEPOINT` - Creates a savepoint in the transaction
- `ROLLBACK` - Rollback to the beginning of the transaction or to a savepoint.
- `RELEASE` - Release a savepoint from memory
- `SET TRANSACTION` – Places a name on a transaction.

TCL and transactions affect data, they do *not* affect database objects (e.g schema, table, etc.)

TCL Example:

Use the table that we created in the previous example in the normalization demo.

```
BEGIN;
        INSERT INTO store (name) VALUES ('first store');
        INSERT INTO store (name) VALUES ('second store');
        SAVEPOINT mysavepoint;
        INSERT INTO store (name) VALUES ('third store');
        ROLLBACK TO SAVEPOINT mysavepoint;
        INSERT INTO store (name) VALUES ('fourth store');
COMMIT;
```

```
SELECT * FROM store;
```