# Transaction

Collection of **one or more operations** on **one or more databases** which reflects a **single real world transaction**

## Transaction Properties ACID

* Atomic
* Consistent
* Isolated
* Durable

|  |  |
| --- | --- |
| **COMMIT** | **ROLLBACK** |
| * When you want to store changes made by the transaction * **Request**, meaning system might reject it with reason | * When you want to **cancel/abort transaction process** and **revert** tot eh **most recent stable state before transaction** beginning   + User’s change of mind   + Explicit program calls for **error handling**   + Integrity constraints   + System crash |

**Auto-commit**

* When commit is done after EACH SQL COMMAND

**Python DB-API** is **explicit mode** meaning you need to call **commit()** to store and complete transactions

Transaction is considered **consistent if**

* All **Static Integrity** is satisfied
* No **Dynamic constraints** have been violated
* **New State** satisfies specification of **transaction**

## Transaction Checking

|  |  |
| --- | --- |
| Automatically | Manually |
| * CHECK * ASSERTION * TRIGGER * Not always desirable since unnecessary checking might result in slower processing time | * Perform checks in application code only when it is needed * Difficult to maintain as transactions are **modified/added** |

## Deferrable Integrity constraints

* When you can withhold the Integrity constraint checking until after transaction is complete

## Incomplete Transaction, System crash recovery

* When System crash happens **during transaction** the process should clean up and call **rollback** responsively to revert transaction.
* Each item holds the value of the last committed transaction for this **rollback functionality**

## Concurrency

* To allow transactions to be **MULTI-THREAD SAFE**

### Serializability

* + Process of finding elements that do not require **changes to internal state of database** and only locking the database to those processes **that updates**