The benefit of a transaction is to perform complex changes, which may require multiple updates to different tables, and be assured that they will all succeed or else all be rolled back.

The term for this is *atomic*, i.e. the change can't be subdivided any smaller.

In fact, MySQL's default storage engine InnoDB uses transactions for everything whether you request it or not. But most people use a mode called *autocommit*, where each statement implicitly starts a transaction and implicitly commits as soon as the statement finishes. In autocommit mode, you have no opportunity to choose to roll back. Either the statement succeeds, or else it if it encounters an error it automatically rolls back.

If you start an explicit transaction, perform some updates, and then roll back, InnoDB restores the original state of data. It preserves the original data by storing it in an area of the database called the *rollback segment*. So if you roll back, it just re-copies those pages of data to replace the ones you changed.

This might take some time, so if you try to query data that was changed but rolled back, InnoDB automatically takes a detour to read the original data out of the rollback segment, until such time as it is re-merged into the tables.

Say for example you start a transaction, and UPDATE a billion rows. This copies many pages worth of the original rows to the rollback segment, and then fills the tables with changed data -- but the changed data is *uncommitted*. No one should be able to read uncommitted data, so anyone who queries the table will automatically get the original data from the rollback segment.

Then you rollback your transaction. Gradually over the next few minutes, InnoDB cleans up, and eventually it all comes back into sync. But anyone can continue to query the original data in the meantime.

If you had committed your transaction, then MySQL would just mark all the changed data as committed, and anyone subsequently reading the data wouldn't experience the slight overhead of reading from the rollback segment.