# ACID-compliant Deletion

Completed by Pradeep Prakash and Jonathan Koller

## Quick overview of ACIDity in DB operations

Atomicity

Consistency

Isolation

Durability

An understanding of how these are handled in MySQL informs the deletion in code.

[1]

## Point of Reference: InnoDB

default storage engine

variety of means available to user to maintain ACID principles in design and maintenance of code

[6]

## Atomicity

all or no transactions are committed

Connection.setAutoCommit(*boolean*);  // set to false for "manual"

-called after connection has been opened

Connection.commit();

If needed, Connection.rollback(); is conveniently available

[2][6]

## Consistency

transaction completes or previous state is returned

InnoDB doublewrite buffer [3] is lean; doesn't cost twice the amount of overhead

InnoDB crash recovery [4] utilizes data stored in buffer

## Isolation

transactions are isolated from each other

JDBC:
conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);

SET ISOLATION LEVEL statement

Also covered by Autocommit settings

[2][6]

## Durability

completed transaction is saved securely, typically through hardware configuration with regards to use of memory, processing, and storage

Default settings rely on doublewrite buffer

Pass the durability to the user for performance:

1) Adjust the following variable in config.ini:

```
NoOfReplicas=1 // default is 2
```

2) Then toggle logging:

```
SET ndb_table_no_logging=1;  // This is not durable because data changes do not persist to hard disk

SET ndb_table_no_logging=0;  // This is the default setting and will asynchronously update the hard disk as it occurs in memory
```

[5]

# ACID-compliant DELETION:

Line 25: conn.setAutoCommit(false); **Atomicity is enforced as no partial commits can occur without expressly-written code**

Line 28: conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);  **Isolation of transactions ensures that everything is staged synchronously (and then committed below)**

Line 131: stmt.executeUpdate("DELETE FROM Product WHERE prod = 'p1'");

Line 132: stmt.executeUpdate("DELETE FROM Stock WHERE prod = 'p1'");

**Serialization promotes Isolation in these two consecutive statements. This is ensured because of Line 28.**

Lines 128-160: *series of loops and statements for printing* **assure the user of Consistent data**

Line 174: conn.commit(); **<-- Durability ensured with manual commit**

## Informal Reference

[1] Introduction  to InnoDB. Retrieved from dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html

[2] JDBC (Scharff) Retrieved from docs.google.com/presentation/d/1acP9ZYOo6SC003xVGLtmRGoLUMwxc7F6BJzUtA6da4g/edit#slide=id.p1

[3] MySQL Glossary entry. Retrieved from dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_doublewrite_buffer

[4] MySQL Glossary entry. Retrieved from dev.mysql.com/doc/refman/5.6/en/glossary.html#glos_crash_recovery

[5] How can a database be in-memory and durable at the same time? (Morgan, 2010). Retrieved from http://www.clusterdb.com/mysql-cluster/how-can-a-database-be-in-memory-and-durable-at-the-same-time

[6] InnoDB and the ACID Model. Retrieved from dev.mysql.com/doc/refman/5.7/en/mysql-acid.html