

INFS 740
DB Programming
for the Web

Prof. Alex Brodsky

GMU

Review: The ACID properties

- **A**tomicity: All actions in the Xact happen, or none happen.
- **C**onsistency: If each Xact is consistent, and the DB starts consistent, it ends up consistent.
- **I**solation: Execution of one Xact is isolated from that of other Xacts.
- **D**urability: If a Xact commits, its effects persist.
- The **Recovery Manager** guarantees Atomicity & Durability.

Transactions in JDBC

```
con.setAutoCommit( false );
...
bError = false;
try
{
    for( ... )
    { // validate data, set bError true if error
        if( bError )
        { break; }
        stmt.executeUpdate( ... );
    }
    if( bError )
    { con.rollback(); }
    else
    { con.commit(); }
```

```
} // end try
catch ( SQLException SQLe)
{
    con.rollback();
    ...
} // end catch
catch ( Exception e)
{
    con.rollback();
    ...
} // end catch
```

Isolation Levels

- `getTransactionIsolation()` and `setTransactionIsolation (int level)`
 - `TRANSACTION_NONE`
 - `TRANSACTION_READ_COMMITTED`
 - `TRANSACTION_READ_UNCOMMITTED`
 - `TRANSACTION_REPEATABLE_READ`
 - `TRANSACTION_SERIALIZABLE`

Properties for Isolation Levels

	Dirty	Non Repeatable	Phantom Read
Read Uncommitted	Y	Y	Y
Read Committed	N	Y	Y
Repeatable Read-	N	N	Y
Serializable	N	N	N

Phantom Read: roughly – some inserted data (by another transaction) that should have been read (due to query condition) but not ‘seen’.

```

import java.sql.*;
public class TransactionPairs {
    public static void main(String args[]) {
        String url = "jdbc:mysql:myDataSource";
        Connection con = null;
        Statement stmt;
        PreparedStatement updateSales;
        PreparedStatement updateTotal;
        String updateString = "update COFFEES " +
            "set SALES = ? where COF_NAME like ?";
        String updateStatement = "update COFFEES " +
            "set TOTAL = TOTAL + ? where COF_NAME like ?";
        String query = "select COF_NAME, SALES, TOTAL from COFFEES";
        try {
            Class.forName("myDriver.ClassName");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }
    }
}

```

```

try {
    con = DriverManager.getConnection(url,
                                     "myLogin", "myPassword");
    updateSales = con.prepareStatement(updateString);
    updateTotal = con.prepareStatement(updateStatement);
    int [] salesForWeek = {175, 150, 60, 155, 90};
    String [] coffees = {"Colombian", "French_Roast",
                        "Espresso", "Colombian_Decaf",
                        "French_Roast_Decaf"};

    int len = coffees.length;
    con.setAutoCommit(false);
    for (int i = 0; i < len; i++) {
        updateSales.setInt(1, salesForWeek[i]);
        updateSales.setString(2, coffees[i]);
        updateSales.executeUpdate();
        updateTotal.setInt(1, salesForWeek[i]);
        updateTotal.setString(2, coffees[i]);
        updateTotal.executeUpdate();
        con.commit();
    }
}

```


Distributed Transaction: Two-Phase Commit (2PC)

- Site at which Xact originates is **coordinator**; other sites at which it executes are **subordinates**.
- When a transaction wants to commit:
 - ★ Coordinator sends **prepare** msg to each subordinate.
 - ★ Subordinate force-writes an **abort** or **prepare** log record and then sends a **no** or **yes** msg to coordinator.
 - ✱ If coordinator gets unanimous yes votes, force-writes a **commit** log record and sends **commit** msg to all subs. Else, force-writes **abort** log rec, and sends **abort** msg.
 - ✱ Subordinates force-write **abort/commit** log rec based on msg they get, then send **ack** msg to coordinator.
 - ⊞ Coordinator writes **end** log rec after getting all acks.

Comments on 2PC

- Two rounds of communication: first, **voting**; then, **termination**. Both initiated by coordinator.
- Any site can decide to abort an Xact.
- Every msg reflects a decision by the sender; to ensure that this decision survives failures, it is first recorded in the local log.
- All commit protocol log recs for an Xact contain Xactid and Coordinatorid. The coordinator's abort/commit record also includes ids of all subordinates.

Restart After a Failure at a Site

- If we have a **commit** or **abort** log rec for Xact T, but not an end rec, must redo/undo T.
 - If this site is the coordinator for T, keep sending **commit/abort** msgs to subs until **acks** received.
- If we have a **prepare** log rec for Xact T, but not **commit/abort**, this site is a subordinate for T.
 - Repeatedly contact the coordinator to find status of T, then write **commit/abort** log rec; redo/undo T; and write **end** log rec.
- If we don't have even a **prepare** log rec for T, unilaterally abort and undo T.
 - This site may be coordinator! If so, subs may send msgs.