We used the following optimization techniques in parsing:

1. Improved Stored Procedure:
Using select count (*) to find the maxid of movie id and star id is proven to be expensive. By creating another table containing maxid for stars, movies and genres, each time doing insertion, the stored procedure will select the maxid stored in the max table. And update the max value.

2. Use of batch insertion. The corresponding sentences are:

```
state.addBatch();
        if (i%1000==0 || i>0) {
        state.executeBatch();
        state.clearBatch();
        i=0;
        }
```

In this case, we batch the insertion in pack of 1000.

3. We disabled auto-commit. The corresponding sentences are:

```
Connection conn = (Connection) java.sql.DriverManager.getConnection(URL, user, password);
conn.setAutoCommit(false);
```

Originally, on our local machines, without the improving techniques, the insertion of a single file main243.xml took 10 seconds, that is about 12 minutes, so we took the testing to the AWS Instance. The following results are done on AWS.

Without Optimization:
mains243.xml: 10s.
actor63.xml: 12s.

With Batch Insertion:
mains243.xml: 2s.
actor63.xml: 3s.

With Batch Insertion and Disabled Auto-Commit:
mains243.xml: 1s.
actor63.xml: 900ms.