Optimization for parsing the XML files

Introduction

For the project3, we need to parse 3 xml files and insert more data into the database. In this project I used DOM to implement the xml parser.
We can easily think of the naïve way to parse these xml files. Considering the large amount of data in the given xml, we need to do some optimizations to make the parser more efficient.

Naïve implementation

Following the example given by the instructions, we can use DOM to parse xml files. In this case, every record becomes a node. We retrieve the node value which is needed in the database and every time we get a row of data, we do insert operation to insert this record into database. Before the insertions, I need to do some query SELECT to check duplicates for the data. But it will be time-consuming doing so many queries.

Hash map and Hash set

To check the duplicates, we need to take data from database. And every time we get a new record from xml file, it's very necessary to check if it's already in the database or is inserted. Thus the best data structure can be used to store the data is hash map. When parsing star file, we can use star name as key and birth year as key value. We can consider 2 records the same if they have same keys and key values. And the complexity of get operation of hash map is usually O(1). So it will be very efficient.

Batch Insert

To optimize the insertions, we could cancel the auto commit for the query and do all the insertion manually in on transaction. Transaction is very expensive so it will waste a lot of time if we auto commit every query. In my project, to check duplicates and inconsistencies, I retrieve all the existed data from database and put them in the **hash map** so that when I get a new record from xml files, I can use hash map to check if it already existed in the database. Then I store all the needed data for the queries in the **batch** and do a **commit** every 50 queries.
It increases the performance a lot. I will take casts.xml for example. Before optimization, it takes more than 20 minutes to insert all the records. With batch insert, it just takes about more than 20 seconds to finish it.

I optimized all three parsers using these 2 method and increased my performance a lot compared to the naïve version of parser.