DBMS Models and Implementation
Instructor: Sharma Chakravarthy
Project 3: Map/Reduce Programming Exercise and Analysis

Made available on:	**11/1/2025**
Submit by:	**12/8/2025 (11:59 PM) No extension for this project as the semester ends.**
Finish demo by:	**12/9/2025 (grades need to be submitted soon after that)**
Submit to:	**Canvas (1 zipped folder containing all the files/sub-folders) https://uta.instructure.com/**
Weight:	**15% of total**
Total Points:	**100**

One of the advantages of cloud computing is its ability to deal with appropriate amounts of resources to process very large data sets to manage response time. Typically, the map/reduce paradigm is used for these types of problems in contrast to the RDBMS approach used for storing, managing, and manipulation of data over long periods of time. An immediate one-time analysis of a large data can benefit from not designing a schema and loading the data set into an RDBMS. Ability to handle large data sets and deploy as many processors as needed for reducing response time is also an important aspect. Scalability can also be achieved using this approach or architecture. Hadoop is a widely used open-source map/reduce platform. Hadoop Map/Reduce is a software framework for writing applications which process vast amounts of data in parallel on large clusters. In this project, you will use the IMDB (International Movies) dataset that you used for project Ia (loaded on Oracle) and develop programs to get interesting insights into the dataset using Hadoop map/reduce paradigm. Please use the following links for a better understanding of Hadoop and Map/Reduce (https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html). This project will also help you understand the differences between an RDBMS and map/Reduce for solving the **same problem**.

1.

i. **SDSC Expanse M/R system**
You will be using the SDSC Expanse system for your project. Separately, you will be given instructions for creating accounts (or getting a user id), using the system for this project, etc. This is a facility supported by NSF for educational purpose. There is a limit on the resources available. Keep that in mind while you do the project. No similar capability is provided by UTA for educational purpose.
You can install Hadoop on your laptop/desktop for developing and testing the code before you run it on Expanse. This is for your convenience. For that you may have to install WSL 1 or 2 if you are using a windows machine. We may not be able provide instructions or lots of support for that. Please look up and install Hadoop on your laptop if you plan to do that.

ii. **Local Hadoop system**
If you want to install on your laptop/desktop, you are welcome to do so for developing code and testing purposes before you run it on the Expanse system. Submitted code MUST run as expanse mentioned in i. above. Local code execution will NOT be acceptable for this

project. Expanse allows you to increase the number of mappers and reducers which may not be possible on your laptop depending on its hardware configuration.

2. IMDb Dataset

IMDb is a data set that contains information about movies (including international) and TV episodes from their beginnings. The information includes movie titles, directors, actors, genre, and year produced/started. Some rating information is also included. The same is true for TV episodes and includes number of seasons in terms of start and end years as well as episodes in each season. It is quite large and additional information to understand the data set can be looked up on Google. The data that you will use is given below. Here are the 2 files you will be using for this project. You can download them by clicking on the link (zip files)

1. imdb00.title.basics.zip	(tab separated file .tsv),	742MB,	9,099,385 lines
2. imdb00.title.actors.zip	(comma separated file .csv),	666 MB,	20,114,478 lines

i. imdb00.title.basics.tsv

This dataset contains the information about the various **IMDb titles** (movies, tv episodes, documentaries etc.), produced across the world. Each field in this dataset is separated by a tab. A random sample from this file is shown below.

```
tt0000091	short	The House of the Devil		1896	Horror,Short
tt0468569	movie	The Dark Knight	2008	Action,Crime,Thriller
tt0088610	tvSeries		Small Wonder	1985	Comedy,Family,Sci-Fi
```

The description of the various fields has been given below.

Field Number	Field Name	Field Description
1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*
2	TITLE TYPE	Every IMDB title in the file is categorized as one of the following TITLETYPEs • *tvSpecial* • *tvMovie* • *tvShort* • *short* • *tvEpisode* • *videogame* • *movie* • *tvSeries* • *tvMiniSeries* • *video*
3	TITLE NAME	The name of the IMDB Title, example: *The Dark Knight*
4	YEAR	The year of release, example: *2008*

5	GENRE LIST	Multiple genres can be attached to a particular title, and they are separated by **comma,** example: *Action,Crime,Thriller*

ii. imdb00.title.actors.csv

This dataset contains the information about the **actors from each IMDB title**. Each field in this dataset is separated by a comma. A random sample from this file has been shown below.

```
tt1410063,nm0000288,Christian Bale
tt1429751,nm0004266,Anne Hathaway
tt1872194,nm0000375,Robert Downey Jr.
```

The description of the various fields has been given below.

Field Number	**Field Name**	**Field Description**
1	TITLE ID	The 9-digit unique IMDB title identifier attached to every entry, example: *tt0000091*
2	ACTOR ID	The 9-digit unique actor identifier, example: *nm0000288*
3	ACTOR NAME	The name of the actor, example: *Christian Bale*

3. **Project 3 Problem Specification:** You need to compute the following for the given data using the map/reduce paradigm. Try to compare and understand how you would do it using RDBMS if these files were stored as relations. This may help you understand when map/reduce is meaningful and when to use a RDBMS. You must implement Q3 as given in Project 1a (repeated below.)

[PROJECT 3 – 100 points] Choose your favorite actor/actress and find the number of movies done by that individual for each year in the entire data set. **Write a Map/Reduce program to output the actor's name, year, and the count of movies they have done for that year. All datasets given are needed for this.** Note that the titletype considered should only be movie; do not include tvMovie or others.

You need to do this with multiple partitions (shards) of input (at least 4 for each input) even if the total number of mappers used is less than the number of partitions (or shards). You will need to write two chained Map/Reduce jobs, where the output from Job 1 will be used as input for Job 2. You must write two mapper classes (one for each input file) and one reducer class for Job 1, and one mapper class and one reducer class for Job 2. You need to analyze the response time for each and compare with that of the alternatives.

Hint: This problem corresponds to a typical SQL query which has joins and group by clauses with count as the aggregate function. The purpose of this problem is to understand how some of the relational computations can be performed using the map/reduce paradigm. You can also write an SQL for this and run it on the Omega Oracle IMDb database that has been setup.

You will need use at least two configs to solve this problem:

a. *At least 2 mappers (one for each input) and 1 reducer for job 1, and 1 mapper and 1 reducer for job 2*

   You can also increase the number of partitions to see the effect for the above configuration.

b. *At least 2 mappers for each input, 2 reducers for job 1, and 2 mappers and 2 reducers for job 2*
   You can also increase the number of partitions and/or mappers and reducers.

c. *if you are interested, you can also try with number of mappers equal to the number of partitions to get an understanding of how increasing parallel computation reduces total response time. This option could be used for one of the above or in addition to the above.*

Understand the difference between mappers and mapper tasks! They are not the same.

4. **Project Report:** Please include (at least) the following sections in a **REPORT.docx** file that you will turn in with your code:

   i. **Overall Status**
   Give a *brief* overview of how you implemented the major components. If you were unable to finish any portion of the project, please give details about what is completed and your understanding of what is not. (This information is useful when determining partial credit.)

   ii. **Approach:** for a) logic of mapper code, combiner code (if present), and reducer code for each job. Why they work and produce what you are looking for. For b) how and why you chose the number of mappers/reducers you did, justification behind them, and what you expected to see and what you observed.

   iii. **Analysis:**
   Time taken for a. and what you learnt from that. Compare the performance (response time) of the effect of increasing partitions. For b) compare the response times with a. and discuss. Analysis of different numbers of mappers and reducers.

   iv. **File Descriptions**
   List the files you have created and *briefly* explain their major functions and/or data structures.

   v. **Division of Labor**
   Describe how you divided the work, i.e. which group member did what. Please also include how much time each of you spent on this project. (This has no impact on your grade whatsoever; we will only use this as feedback in planning future projects -- so be honest!)

   vi. **M/R configuration details for multiple inputs and other details**:
   What libraries and packages you have used for dealing with multiple inputs.

5. **What to submit:**

   - After you are satisfied that your code does exactly what the project requires, you may turn it in for grading. Please submit your project report with your project.

- You will turn in one zipped file containing **a) source code, b) outputs from the M/R code for each configuration, log files, and c) raw analysis results (spreadsheets etc.) as well as the d) report. This is for each configuration.**
- All of the above files should be placed in a single zipped folder named as - 'Fall_2025_**proj3_team_<*TEAM_NO*>**'. **Only one zipped folder should be uploaded using canvas.**
- You can submit your zip file at most 3 times. The latest one (based on timestamp) will be used for grading. So, be careful in what you turn in and when!
- **Only one person per group should turn in the zip file!**
- **Late Submissions not allowed for this project.**

6. Coding style:

Be sure to observe the following standard Java naming conventions and style. These will be used across all projects for this course; hence it is necessary that you understand and follow them correctly. You can look this up on the web. Remember the following:

i. Class names begin with an upper-case letter, as do any subsequent words in the class name.
ii. Method names begin with a lower-case letter, and any subsequent words in the method name begin with an upper-case letter.
iii. Class, instance and local variables begin with a lower-case letter, and any subsequent words in the name of that variable begin with an upper-case letter.
iv. No hardwiring of constants. Constants should be declared using all upper-case identifiers with _ as separators.
v. All user prompts (if any) must be clear and understandable.
vi. Give meaningful names for classes, methods, and variables even if they seem to be long. The point is that the names should be easy to understand for a new person looking at your code.
vii. Your program is properly indented to make it understandable. Proper matching of if … then … else and other control structures is important and should be easily understandable.
viii. Do not put multiple statements in a single line.

In addition, ensure that your code is properly documented in terms of comments and other forms of documentation for generating meaningful Javadoc.

7. Grading rubrics:

The **PROBLEM** will be graded **out of 100** points using the following scheme:

a.	Correctness of the Code (job 1, job 2 and combiner if used)	**30**
b.	Correctness of the Output for various partitions and configs	**20**
c.	Report	**20**
	i. Analysis of 2 given configurations	
	ii. Analysis of their response time	
	iii. Any additional configuration or analysis you have done	
d.	Answering questions during demo	**30**

8. **Project 3 Demonstrations:** Mandatory Team-wise demos will be scheduled after the last day of classes. Submission must be done prior to demo. A sign-up sheet for the demo Schedule will be provided. If you finish early, you are welcome to demonstrate early by sending the TA an email.
9. **Additional information about registration and sample program can be downloaded from**
   F25-supplementary-materials.zip (1 zip file and 2 .docx files)
10. **Please enter your ACCESS username into the google spread sheet before Nov 4th, 2025.**
    https://docs.google.com/spreadsheets/d/1vWcf2TIO6Bus95DGF9AVukziFipzS-ehhrrQQaPEUcA/edit?usp=sharing