**Group 1**

Khoa Le

Thanh Le

Lam Nguyen

CS 430 | Database System

Dr. Jaiswal

Final Project Report

**Index**

1. Front end source code:

a) populate the databases,

b) measure the delay in search.

2. Graphs.

3. Graphs analysis.

4. Lessons learned.

5. Problems faced & adopted solutions.

6. Team member contributions.

**1. Front end source codes**

**Front end source codes to populate the database**

**mySQL:**

**GenerateData.java**

import java.io.\*;

import java.util.\*;

public class GenerateData {

public static void main(String[] args) {

PrintWriter out = new PrintWriter(System.out);

int mode = Integer.parseInt(args[0]);

if (mode == 0) {

printAuthor(out);

} else if (mode == 1) {

printPapers(out);

} else if (mode == 2) {

printComments(out);

} else {

printReviews(out);

}

out.close();

}

public static void printComments(PrintWriter out) {

String header = "Review\_id|Paper\_id|Rating\_score|Public\_comment\_info|Private\_comment\_info";

out.println(header);

int n = 1500000;

Random rand = new Random();

for (int i = 0; i < n; i++) {

int reviewId = rand.nextInt(n) + 1;

int paperId = i + 1;

int ratingScore = rand.nextInt(10);

String publicComment = generateRandomName();

String privateComment = generateRandomName();

out.println(reviewId + "|" + paperId + "|" + ratingScore + "|" + publicComment + "|" + privateComment);

}

}

public static void printAuthor(PrintWriter out) {

String header = "Author\_id|Paper\_id|Author\_name|Author\_email";

out.println(header);

int n = 1500000;

Random rand = new Random();

for (int i = 0; i < n; i++) {

int authorId = i + 1;

int paperId = rand.nextInt(n) + 1;

String name = generateRandomName();

String email = getEmailAddress(name);

out.println(authorId + "|" + paperId + "|" + name + "|" + email);

}

}

private static void printPapers(PrintWriter out) {

String header = "Paper\_id|Paper\_Author|Paper\_abstract|Paper\_year|Paper\_title|Paper\_filename";

out.println(header);

int n = 1500000;

Random rand = new Random();

for (int i = 0; i < n; i++) {

int paperId = i + 1;

String author = generateRandomName();

String paperAbstract = generateRandomName();

String paperTitle = generateRandomName();

String paperFilename = getFilename(paperTitle);

String year = getRandomYear(1990, 2017);

out.println(paperId + "|" + author + "|" + paperAbstract + "|" + year + "|" + paperTitle + "|" + paperFilename);

}

}

private static void printReviews(PrintWriter out) {

String header = "Review\_id|Review\_topic|Review\_email";

out.println(header);

int n = 1500000;

Random rand = new Random();

for (int i = 0; i < n; i++) {

int reviewId = i + 1;

String topic = generateRandomName();

String email = getEmailAddress(generateRandomName());

out.println(reviewId + "|" + topic + "|" + email);

}

}

private static String generateRandomName() {

Random rand = new Random();

int len = 5 + rand.nextInt(10);

String name = "";

for (int i = 0; i < len; i++) {

name += intToChar(rand.nextInt(26));

}

return name;

}

private static final String[] domains = {"@gmail.com", "@hotmail.com", "@truman.edu", "@havard.edu", "@yahoo.com"};

private static String getEmailAddress(String name) {

Random rand = new Random();

return name + domains[rand.nextInt(domains.length)];

}

private static final String[] extensions = {".txt", ".pdf", ".docx", ".pages"};

private static String getFilename(String filename) {

Random rand = new Random();

return filename + extensions[rand.nextInt(extensions.length)];

}

private static String getRandomYear(int low, int high) {

Random rand = new Random();

int year = rand.nextInt(high - low) + low;

return "" + year;

}

private static char intToChar(int val) {

val %= 26;

return (char) (val + 'a');

}

}

**Front end source codes to search the database in mySQL:**

**SqlFrontEnd.java**

import java.io.\*;

import java.util.\*;

import java.sql.\*;

public class SqlFrontEnd {

public static final String JDBC\_DRIVER = "com.mysql.jdbc.Driver";

public static final String DB\_URL = "jdbc:mysql://localhost:3306/CS430";

public static final String USER = "root";

public static final String PASS = "lendkhoa";

Connection connection = null;

public SqlFrontEnd(){

try{

Class.forName("com.mysql.jdbc.Driver");

//STEP 3: Open a connection

System.out.println("Connecting to database...");

connection = DriverManager.getConnection(DB\_URL,USER,PASS);

} catch(Exception e){

e.printStackTrace();

}

}

public void getQueryStatement(){

Scanner in = new Scanner(System.in);

StringTokenizer tk;

System.out.println();

System.out.println("Select option number and enter attribute");

System.out.println("Find best paper in year (1990 - 2017) - NUMBER 1");

System.out.println("Find paper with title - NUMBER 2");

System.out.println("Find paper with ID - NUMBER 3");

System.out.println("Find rating score with ID - NUMBER 4");

System.out.println("Find author email with paper title - NUMBER 5");

String input = in.nextLine();

System.out.println("====Result===");

tk = new StringTokenizer(input);

int option = Integer.parseInt(tk.nextToken());

if (option == 1){

findBestPaperIn(tk.nextToken());

}else if (option == 2){

findPaperWithTitle(tk.nextToken());

}else if (option == 3){

findPaperWithID(Integer.parseInt(tk.nextToken()));

}else if (option == 4){

findRatingScoreWithID(Integer.parseInt(tk.nextToken()));

}else if (option == 5){

findAuthorEmailWithTitle(tk.nextToken());

}

}

public void findBestPaperIn(String year){

try{

Statement statement = connection.createStatement();

String sql = "SELECT Paper\_title, Paper\_Author, Rating\_score FROM Papers INNER JOIN Comment ON Papers.Paper\_id = Comment.Paper\_id WHERE Paper\_year = '"+year+"' ORDER BY Rating\_score DESC LIMIT 1";

long start = System.currentTimeMillis();

ResultSet rs = statement.executeQuery(sql);

String res = "";

System.out.println("Best paper in year "+year+" :");

while (rs.next()){

String title = rs.getString("Paper\_title");

String authorName = rs.getString("Paper\_Author");

int rating = rs.getInt("Rating\_score");

res += "Paper title: " + title + " | Author name: " + authorName + " | Rating: " + rating;

System.out.println(res);

}

long end = System.currentTimeMillis();

System.out.println("Time elapsed: "+(end-start)+ " milliseconds");

rs.close();

statement.close();

} catch(Exception e){

e.printStackTrace();

}

}

public void findPaperWithTitle(String title) {

try{

Statement statement = connection.createStatement();

System.out.println("Paper with title: "+title);

String sql = "SELECT \* FROM papers WHERE Paper\_title='"+title+"'";

long start = System.currentTimeMillis();

ResultSet rs = statement.executeQuery(sql);

while (rs.next()){

int id = rs.getInt("Paper\_id");

String authorName = rs.getString("Paper\_Author");

System.out.println("id " + id + " " + authorName);

}

long end = System.currentTimeMillis();

System.out.println("Time elapsed: "+(end-start)+" milliseconds");

rs.close();

statement.close();

} catch(Exception e){

e.printStackTrace();

}

}

public void findPaperWithID(int id){

try{

Statement statement = connection.createStatement();

String sql = "SELECT Paper\_title, Paper\_Author, Review\_id, Rating\_score, Public\_comment\_info FROM papers INNER JOIN Comment ON papers.Paper\_id = Comment.Paper\_id WHERE papers.Paper\_id="+id;

String res = "";

long start = System.currentTimeMillis();

ResultSet rs = statement.executeQuery(sql);

System.out.println("Paper with id: "+id);

while (rs.next()){

String title = rs.getString("Paper\_title");

String authorName = rs.getString("Paper\_Author");

String reviewer = rs.getString("Review\_id");

int score = rs.getInt("Rating\_score");

String comment = rs.getString("Public\_comment\_info");

res += "Title: "+title+" | Author name: "+authorName+" | ReviewerID: "+reviewer+" | Rating: "+score+" | Comment: "+comment;

System.out.println(res);

}

long end = System.currentTimeMillis();

System.out.println("Time elapsed: "+(end-start)+" milliseconds");

rs.close();

statement.close();

} catch(Exception e){

e.printStackTrace();

}

}

public void findRatingScoreWithID(int id){

try{

Statement statement = connection.createStatement();

String sql = "SELECT Rating\_score FROM Comment WHERE Comment.Paper\_id="+id;

String res = "";

long start = System.currentTimeMillis();

ResultSet rs = statement.executeQuery(sql);

System.out.println("Paper with id: "+id);

while (rs.next()){

int score = rs.getInt("Rating\_score");

res += "Rating: "+score+" | Comment: ";

System.out.println(res);

}

long end = System.currentTimeMillis();

System.out.println("Time elapsed: "+(end-start)+" milliseconds");

rs.close();

statement.close();

} catch(Exception e){

e.printStackTrace();

}

}

public void findAuthorEmailWithTitle(String title){

try{

Statement statement = connection.createStatement();

String sql = "SELECT author\_email FROM authors INNER JOIN papers ON authors.paper\_id = papers.Paper\_id WHERE Paper\_title ='"+title+"'";

String res = "";

long start = System.currentTimeMillis();

ResultSet rs = statement.executeQuery(sql);

while (rs.next()){

String email = rs.getString("author\_email");

res += "Author Email: "+email;

System.out.println(res);

}

long end = System.currentTimeMillis();

System.out.println("Time elapsed: "+(end-start)+" milliseconds");

rs.close();

statement.close();

} catch(Exception e){

e.printStackTrace();

}

}

public void closeConnection(){

try{

connection.close();

} catch(Exception e){

e.printStackTrace();

}

}

public static void main(String[] args) {

SqlFrontEnd page = new SqlFrontEnd();

page.getQueryStatement();

page.closeConnection();

}

}

**Front end source codes to populate the database in Mongo:**

**populateData.js**

var firstname = ["immy", "enry", "onathan", "anilo", "onilo", "anila", "elsea", "away", "ohnson", "andrew"];

var lastname = ["immy", "enry", "onathan", "anilo", "onilo", "anila", "elsea", "away", "ohnson", "arry", "ueen"];

var abstract = ["MongoDb", "Recovery", "Paper is about mySQL", "Data Concurrency"];

var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

var title = ["A database ",

"is ",

"an",

" organized collection of data.",

"[1] It is the collection of schemas,",

" tables, queries, reports, views, and other objects.",

" The data ",

"are typically organized to model aspects ",

"of reality in a way that supports processes",

" requiring information, such as modelling the availability",

" of rooms in hotels in a way that supports finding a hotel ",

"with vacancies."];

var emailDomain=["@gmail.com", "@aol.com", "@yahoo.com", "@outlook.com"];

var comment = ["excellent", "in depth", "valuable", "informative"];

var number = 1500000;

var topics = ["Sport", "Politics", "Comedy", "Thriller", "Drama", "Tragedy", "Horror", "Romance", "Science", "Unix", "mySQL"]

function getRandomInt(range) {

return Math.floor(Math.random() \* (range));

}

function getTopics() {

var deck = topics;

for (var i = 0; i < 100; i++) {

var first = getRandomInt(topics.length);

var second = getRandomInt(topics.length);

var tmp = deck[first];

deck[first] = deck[second];

deck[second] = tmp;

}

var numberOfTopics = Math.floor(Math.random() \* 3) + 1;

var result = [];

for (var i = 0; i < numberOfTopics; i++) {

result.push(deck[i]);

}

return result;

}

var i = 0;

for (i == 0; i < number; i++){

db.ConferencePaper.insert(

{

Title: "The"+ title[Math.floor(Math.random()\*title.length)]+" volume "+Math.floor(Math.random()\*100),

Abstract: "Data abstraction, ways of life, no ideas "+ abstract[Math.floor(Math.random()\*abstract.length)],

Topics: getTopics(),

Year: Math.floor(Math.random()\* (2017 - 1990) + 1990),

Paper\_id: i,

Filename: Math.floor(Math.random()\*9999)+".pdf",

Author: [

{

Author\_id: i+possible[Math.floor(Math.random()\*possible.length)],

FirstName: possible[Math.floor(Math.random()\*possible.length)]+firstname[Math.floor(Math.random()\*firstname.length)],

LastName: possible[Math.floor(Math.random()\*possible.length)]+lastname[Math.floor(Math.random()\*lastname.length)],

Email: possible[Math.floor(Math.random()\*possible.length)]+Math.floor(Math.random()\*9999)+ emailDomain[Math.floor(Math.random()\*emailDomain.length)],

}

],

Rating: Math.floor(Math.random()\*10+1),

Reviewers:

[

{

Rating: [{

Recommendation: Math.floor(Math.random()\*10+1),

TechnicalMermit: Math.floor(Math.random()\*10+1),

Originality: Math.floor(Math.random()\*10+1),

Readability: Math.floor(Math.random()\*10+1),

Relavence: Math.floor(Math.random()\*10+1)

}],

Reviewer\_id: possible[Math.floor(Math.random()\*possible.length)]+i,

PrivateComments: comment[Math.floor(Math.random()\*comment.length)],

PublicComments: "Long live Arsene Wenger",

Email: possible[Math.floor(Math.random()\*possible.length)]+Math.floor(Math.random()\*9999)+ emailDomain[Math.floor(Math.random()\*emailDomain.length)],

Phone: Math.floor(Math.random()\*9999)+"-"+Math.floor(Math.random()\*9999)+"-"+Math.floor(Math.random()\*9999),

Name: possible[Math.floor(Math.random()\*possible.length)]+firstname[Math.floor(Math.random()\*firstname.length)]+" "+

possible[Math.floor(Math.random()\*possible.length)]+lastname[Math.floor(Math.random()\*lastname.length)],

Topic: ["Football", "UEFA", "Coaching"]

},

]

}

);

};

**Front end source codes to populate the database in Mongodb:**

**FrontPage.java**

import java.io.\*;

import java.util.\*;

import com.mongodb.MongoClient;

import com.mongodb.DB;

import com.mongodb.DBCollection;

import com.mongodb.DBObject;

import com.mongodb.BasicDBObject;

import com.mongodb.DBCursor;

public class FrontPage {

public static void main(String[] args) {

PrintWriter out = new PrintWriter(System.out);

FrontPage page = new FrontPage();

page.setup(args[0], args[1]);

out.println("Example document:");

out.println(page.getOneRecord() + "\n\n");

long start = System.currentTimeMillis();

out.println("Papers with rating more than 5:");

out.println(page.getRecordsWithRatingGreaterThan(5).count());

out.println();

out.println("Best paper in 1994:");

out.println(page.getBestPaperInYear(1994) + "\n\n");

out.println("Best paper in 1998:");

out.println(page.getBestPaperInYear(1998) + "\n\n");

out.println("Number of paper that has the topic Sport:");

out.println(page.getPapersInTopic("Sport").count());

out.println();

out.println("Count between 1994 and 1997 with rating gte 8.1: " + page.countPapersBetweenYearWithRatingGte(1994, 1997, 8.1));

out.println();

long end = System.currentTimeMillis();

out.println("Time elapsed: "+(end-start));

out.close();

}

public static void printDocuments(PrintWriter out, DBCursor cursor) {

// print the retrieved documents

while (cursor.hasNext()) {

out.println(cursor.next());

}

}

private DB db;

private DBCollection collection;

private MongoClient mongoClient;

public void setup(String dbName, String collectionName) {

// To connect to mongodb server

mongoClient = new MongoClient( "localhost" , 27017 );

db = mongoClient.getDB(dbName);

collection = db.getCollection(collectionName);

}

public DBObject getOneRecord() {

DBObject result = collection.findOne();

return result;

}

public DBCursor getRecordsWithRatingGreaterThan(double rate) {

// creating the filter query

DBObject filter = new BasicDBObject().append("$gte", rate);

DBObject query = new BasicDBObject().append("Rating", filter);

return collection.find(query);

}

public DBCursor getPapersInTopic(String topic) {

// create the filter query

DBObject query = new BasicDBObject().append("Topics", topic);

return collection.find(query);

}

public int countPapersBetweenYearWithRatingGte(int startYear, int endYear, double rate) {

DBCursor cursor = getCusorForPaperBetweenYear(startYear, endYear);

int count = 0;

while (cursor.hasNext()) {

DBObject object = cursor.next();

if ((double) object.get("Rating") >= rate) {

count++;

}

}

return count;

}

public DBObject getBestPaperInYear(int year) {

DBCursor papers = getCursorToPaperInYear(year);

return getBestPaper(papers);

}

public DBObject getBestPaperInTopic(String topic) {

DBCursor papers = getPapersInTopic(topic);

return getBestPaper(papers);

}

private DBObject getBestPaper(DBCursor cursor) {

// find the paper with highest rating

double max = Double.MIN\_VALUE;

DBObject result = null;

while (cursor.hasNext()) {

DBObject object = cursor.next();

if ((double) object.get("Rating") > max) {

result = object;

max = (double) object.get("Rating");

}

}

return result;

}

private DBCursor getCursorToPaperInYear(int year) {

// create the filter query

DBObject query = new BasicDBObject().append("Year", year);

return collection.find(query);

}

private DBCursor getCusorForPaperBetweenYear(int start, int end) {

// create the filter query

DBObject filter = new BasicDBObject().append("$gte", start).append("$lte", end);

DBObject query = new BasicDBObject().append("Year", filter);

return collection.find(query);

}

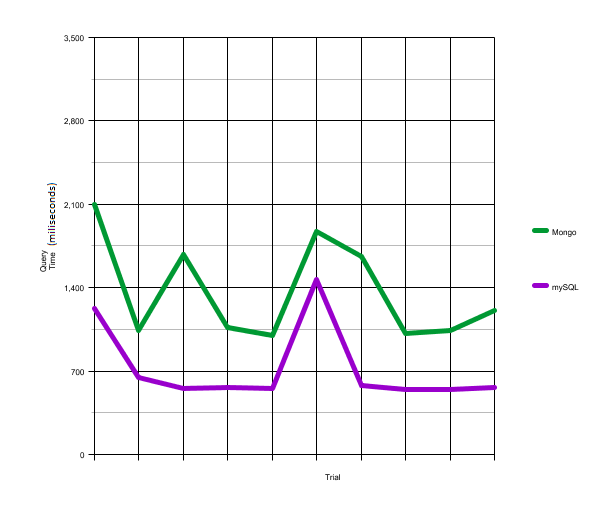
}

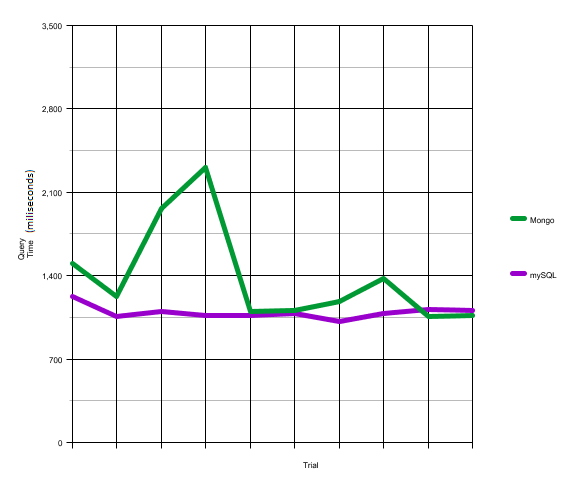
**MongoDB delay in search**

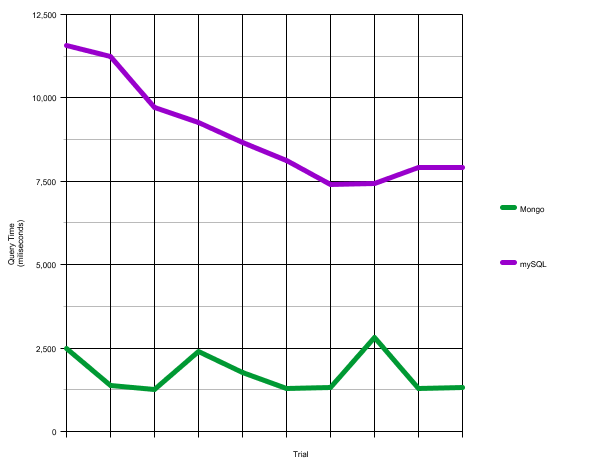
Sample query statement: db.ConferencePaper.find({**Rating**:8});

Through our observation, the delay in search of Mongo is quite consistent (more details below). They ranged from ~1000ms to ~2500ms

**2. Graphs of the given cases**

Case 1: mySQL vs Mongo. graph-noIndex ****

Case 2: mySQL vs Mongo. graph-oneIndex****

Case 3: mySQL vs Mongo. graph-manyIndex****

**3. Graph Analysis**

For each search case we run and time the query at least 10 times and observed the following patterns:

**Case 1: Using no index:**

- mySQL seems to run quite faster than Mongo does. The difference between the two sets of data is rather consistent.

- This result is a surprise when compare to our initial expectation that mySQL will actually run slower than Mongo with a big database.

**Case 2: One index:**

- mySQL is still faster than Mongo by a couple of hundreds of milliseconds, the speed is consistent throughout our testing.

- Mongo’s results fluctuate in the beginning but then it follows the trend of decreasing the time when the same query is run over and over again.

**Case 3: Multiple indexes:**

- In this case, Mongo runs a lot faster than mySQL does and the degree of fluctuation in time is not as large as in case 2.

- mySQL seems to get some improvement after running more queries.

**4. Lessons learned**

We have learned many valuable and practical lessons about database system through working on this project:

Fromthebeginning phase, we understood the importance of having a concise, and flexible design in building a well-structure database system. Therefore, we began with breaking the project case into smaller and simpler objects, then we created one single table and gradually built up the relationship between the entities. Doing so we were able to understand how the database system is created from scratch.

In the second phase, we learned how to implement data structure and create a suitable schema for our database. Moreover, we spent a lot of time in designing and choosing the most optimal solutions to accomplish the project’s requirements. Besides Mongodb and mySQL, we also acquired skills in working with Javascript, Java and Gradle to create the interface to connect to Mongo and mySQL.

**5. Problem faced and adopted solutions**

During the process of developing our database, we first faced some issues about table designing and how to implement the schema with our use cases. To solve that problem, we decided to be flexible in creating the fields in our tables, so it can adapt later situations. In the final phase of our project, we did have some issue about writing code to populate the database. It was mostly because of the ability to combine may tools that we have in order to make them work together. For example, we created some data files to feed the mySQL tables. However, by using Workbench, it almost took 2 days to fully import those files into our tables. To solve that, we had to find suitable query that run directly on the connection with the server. Another issue is the spike in difference of performances when running queries in different machines. We made sure that every tested query is run on a single system in order to get the most consistent data.

**6. Team member contributions**

**Thanh Le:** Writing front-end code to populate and query data for mySQL, ideas leading, quality testing

**Khoa Le:** Writing front-end code to populate and query data for Mongo, running queries for tests

**Lam Nguyen:** Finding suitable queries to use, debugging, gathering and reporting data