CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

SPRINT -2 DOCUMENT

Group-3
Jay Alpeshkumar Patel (B00969013)
Manish Shankar Jadhav (B00969328)

GitLab Link: https://git.cs.dal.ca/patel38/csci_5408_s24_3/-/tree/main?ref_type=heads

Table of Contents

Architecture Diagram

1) Entity Relation Generator

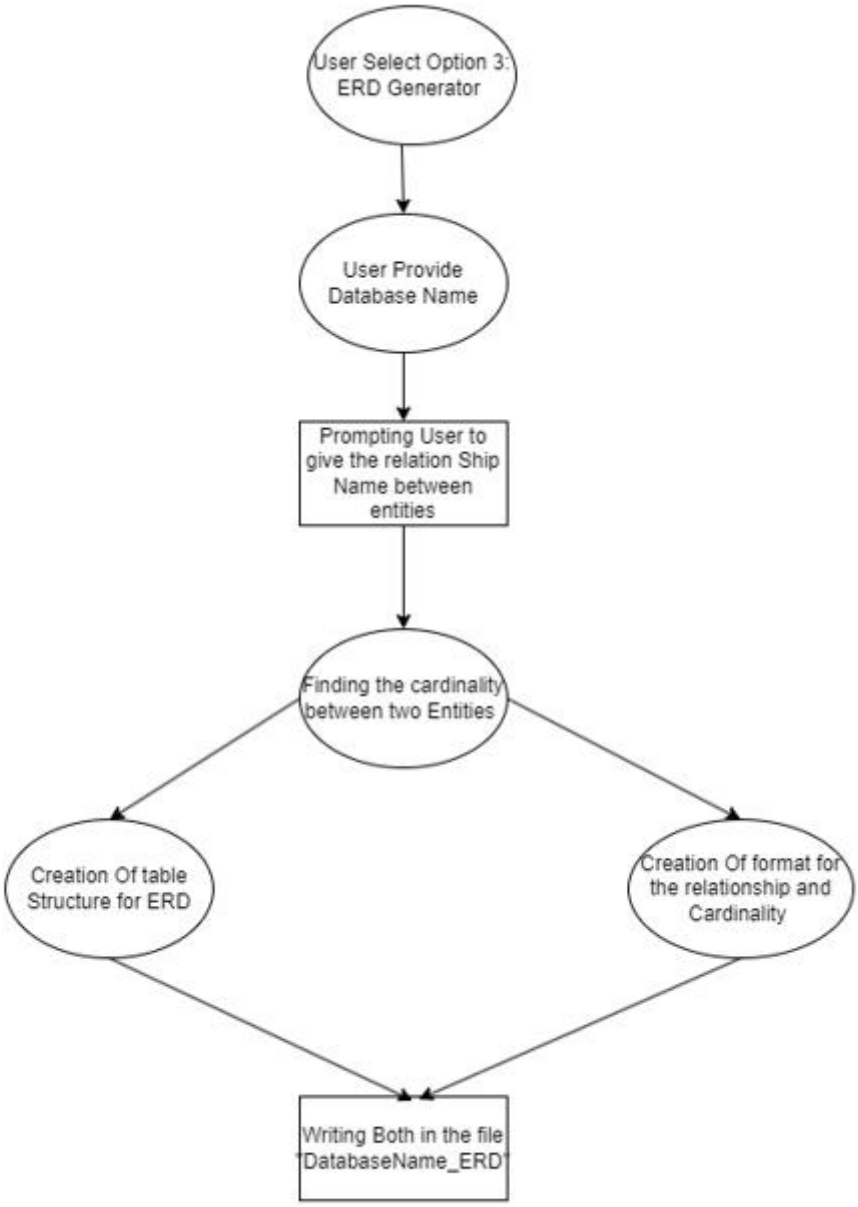

Figure 1 ERD generator

2) Schema Generator

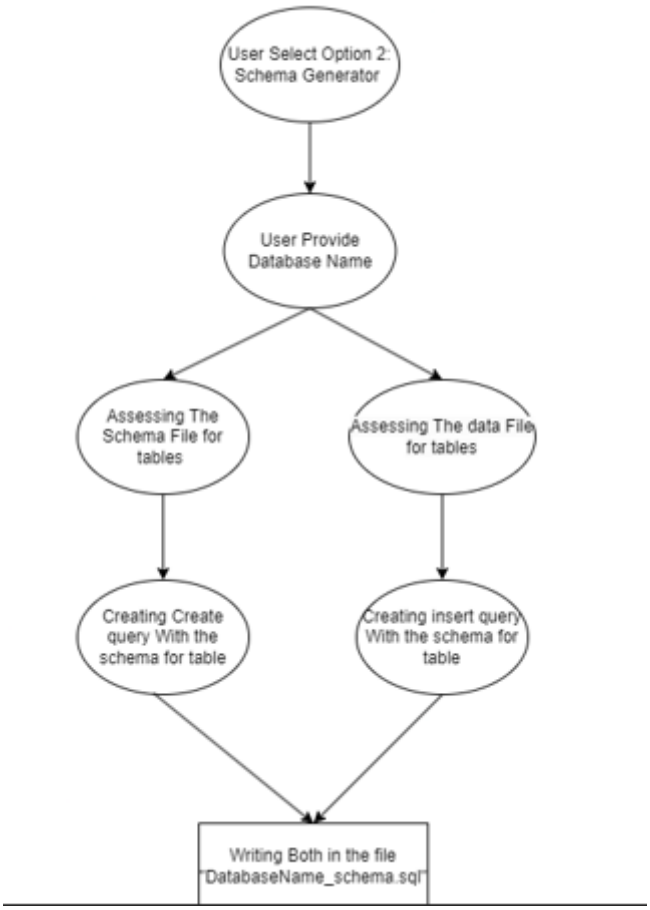

Figure 2 ERD generator

Pseudocode

Authentication:

FUNCTION displayAuthPage()

Display welcome message and options to user

```
Loop until user chooses to exit
Get user's choice (sign up, sign in, or exit)
Handle sign up process if chosen
Handle sign in process if chosen
Exit program if chosen
```

FUNCTION signUpUser()

```
Get username, password, security question, and security answer from user
Check if user already exists
If not, hash password and write user info to file
```

FUNCTION loginUser()

```
Get username and password from user
Check if user exists and password is correct
If correct, get security answer from user and check if correct
Return true if login successful, false otherwise
```

FUNCTION writeAuthFile()

Write user info to file in format "username||hashed password||security question||security answer"

FUNCTION checkIfUserExists()

```
Read user profiles from file
Check if username exists in file
Return true if exists, false otherwise
```

FUNCTION hashPassword()

Hash password using MD5 algorithm
Return hashed password

FUNCTION checkIfPasswordCorrect()

Read user profiles from file
Check if password matches hashed password for given username
Return true if correct, false otherwise

FUNCTION checkIfSecurityAnswerCorrect()

Read user profiles from the file
Check if the security answer matches for given username
Return true if correct, false otherwise

Entity Relationship Diagram

Function main:

```
Display menu and prompt user to select option 3
Prompt user for Database Name
Scan schema file to get table schemas
Call displayERDiagram with extracted schemas
```

Function displayERDiagram(entities, copyOfTables, outputPath):

```
Open file writer for outputPath
Compute max lengths for entity names and attributes
For each entity in entities:
  Write entity name and attributes with proper formatting
  Write primary key and foreign key details
  If entity has foreign key:
    Prompt user for relation between current and foreign table
    Determine and write foreign key relationship type
Write legend for relationships
Close file writer
```

Function createLine(writer, content, maxLineLength):

```
Calculate padding for content
Write content with padding to writer
```

```
Function checkElements(list1, list2):
  Count occurrences of elements in list2
  For each element in list1:
    If element count in list2 is not exactly 1:
      Return "------<>"
  Return "--------"

Function countOccurrences(list):
  Create counter map
  For each element in list:
    Increment counter for element
  Return counter map
```

Schema Generation Diagram

```
Function main:
  Prompt user to select option 2 and enter Database Name
  Call processFolder with folder path
  Call writeSQLToFile with tables and output file path

Function processFolder(folderPath):
  Initialize empty list of tables
  For each schema file in folder ending with "_schema":
    Find matching data file using base name
    If data file exists and both files are not empty:
      Read schema and data lines from files
      Generate and add Table object to tables list
  Return tables list

Function readLinesFromFile(filePath):
  Read and return all lines from filePath

Function readDataFromFile(filePath):
  Read and split lines by "|" from filePath, return as list of arrays

Function generateCreateTableQuery(tableName, schemaLines):
  Extract column definitions, primary key, and foreign key from schemaLines
  Construct and return create table query string

Function generateInsertQueries(tableName, dataLines):
  For each data line starting from second line:
    Construct and return insert queries list


Function writeSQLToFile(tables, outputFilePath):
  Write create table and insert queries for each table to outputFilePath
```

Test Cases and Evidence of Testing

AUTHENTICATION TESTING

1) Signup user

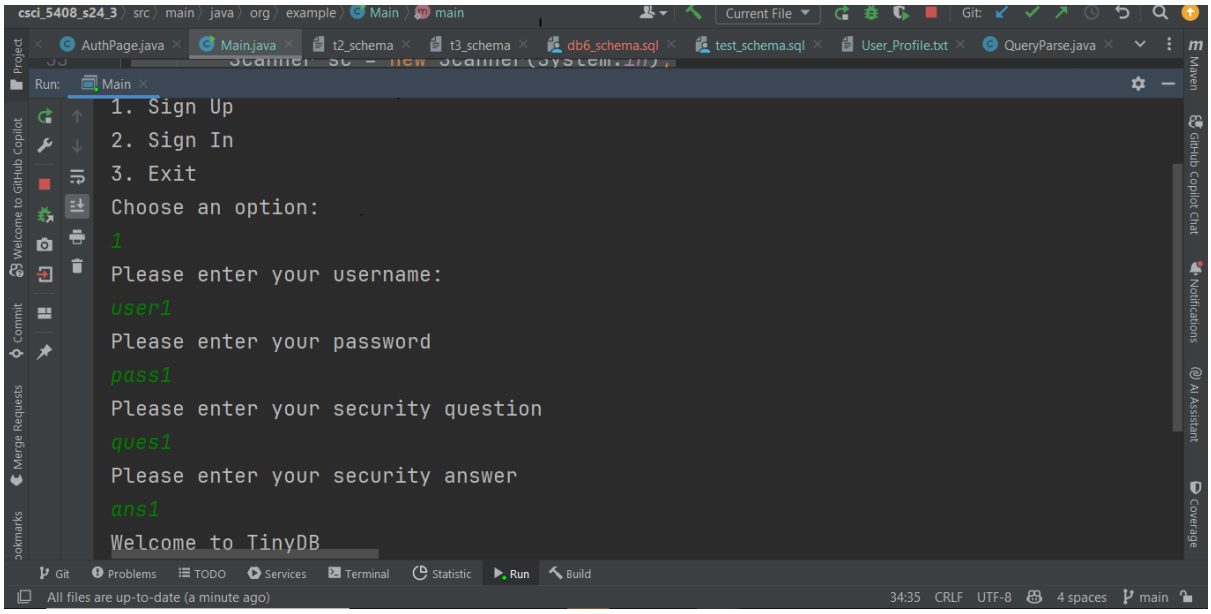

Figure 3: Signup user function.

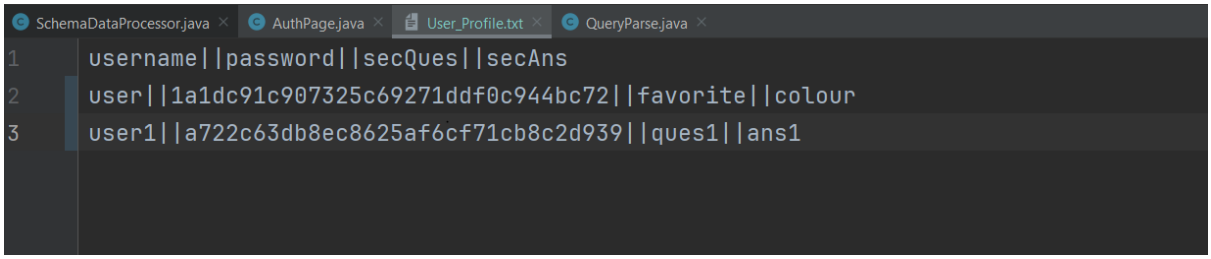

Figure 4: User values get entered into User_Profile file.

2) Signup user (User already exists)

```
1. Sign Up
2. Sign In
3. Exit
Choose an option:
1
Please enter your username:
user1
Please enter your password
pass1
User already exists
Welcome to TinyDB
----------------
1. Sign Up
2. Sign In
3. Exit
Choose an option:
```

Figure 5: Signup user for user already exists.

3) Login user (Correct)

```
Main
2. Sign In
3. Exit
Choose an option:
2
Please enter your username:
user1
Please enter your password
pass1
User_Profile.txt
Please enter your security answer
ans1
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
```

Figure 6: Login user correct password and answer.

4) Login user (Password is wrong)

```
Welcome to TinyDB
-----------------
1. Sign Up
2. Sign In
3. Exit
Choose an option:
2
Please enter your username:
user1
Please enter your password
pass123
User_Profile.txt
User Password is wrong
```

Figure 7: User password wrong.

5) Login user (Security answer is wrong)

```
Welcome to TinyDB
-----------------
1. Sign Up
2. Sign In
3. Exit
Choose an option:
2
Please enter your username:
user1
Please enter your password
pass1
User_Profile.txt
Please enter your security answer
abc
Security answer is wrong
```

Figure 8: Security answer wrong.

SCHEMA GENERATING SCHEMA:

```
SQL> select * from subject;
subjectid   subjectname studentid
SQL> select * from student;
studentid   studentname
```

Figure 9: Subject and student table created with no values.

```
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
2
Enter Database Name :
school
Database Schema Created Successfully

Process finished with exit code 0
```

Figure 10: Exporting Data and structure for school database

It appears that only the create queries are available since there is no data present in the table.

```
DiagramHandeler.java   Main.java   QueryParse.java   SchemaDataProcessor.java   QueriesToTest.txt   school_schema.sql   t2_schema   t2   t3   t3_schema
No data sources are configured to run this SQL and provide advanced code assistance.   Configure data sour
SQL dialect is not configured.   Change dialect to
-- SQL Dump for student
CREATE TABLE student (studentid int, studentname varchar(20), PRIMARY KEY (studentid));


-- SQL Dump for subject
CREATE TABLE subject (subjectid int, subjectname varchar(20), studentid int, PRIMARY KEY (subjectid), FOREIGN KEY (studentid) REFERENCES student(studentid));
```

Figure 11: SQL schema of School Database

During ERD (Entity-Relationship Diagram) creation, the user specifies the relation name between two tables as "learns".

```
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
3
Name of the Database
school
Please give the relation between subject and student
learns
```

Figure 12: Creation of school database when tables are empty

When tables are empty and contain no values, but have primary and foreign key relationships, the cardinality by default is 1:1, typically represented by "--------".

Figure 13: ERD for database school

```
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit

1
SQL> use school;
Entered into database
SQL> insert into student values(1,"manish jadhav")
Syntax Errorinsert into student values(1,"manish jadhav")
SQL> insert into student values(1,"manish jadhav");
Data Inserted Successfully
SQL> select * from student;
studentid   studentname
1   manish jadhav
SQL> insert into student values(2,"jay patel");
Data Inserted Successfully
SQL> select * from student;
studentid   studentname
1   manish jadhav
2   jay patel
```

Figure 14: Inserting Data into student table

```
SQL> insert into subject values(1,"DWMT",1);
Data Inserted Successfully
SQL> insert into subject values(2,"Cloud",2);
Data Inserted Successfully
SQL> select * from subject;
subjectid   subjectname studentid
1   dwmt    1
2   cloud   2
```

Figure 15: Inserting Data into subject table

Now that data has been inserted into the table, the SCHEMA.sql file will show both the create table queries and the insert values.

```
No data sources are configured to run this SQL and provide advanced code assistance.        Configure data source
SQL dialect is not configured.                                                              Change dialect to...

-- SQL Dump for student
CREATE TABLE student (studentid int, studentname varchar(20), PRIMARY KEY (studentid));

INSERT INTO student VALUES ('1', 'manish jadhav');
INSERT INTO student VALUES ('2', 'jay patel');

-- SQL Dump for subject
CREATE TABLE subject (subjectid int, subjectname varchar(20), studentid int, PRIMARY KEY (subjectid), FOREIGN KEY (studentid) REFERENCES student

INSERT INTO subject VALUES ('1', 'dwmt', '1');
INSERT INTO subject VALUES ('2', 'cloud', '2');
```

Figure 16: SQL schema of School Database

When you execute the SQL file, all the tables will be created, and the data will be inserted.

```
4  CREATE TABLE student (studentid int, studentname varchar(20), PRIMARY KEY (studentid));
5
6  INSERT INTO student VALUES ('1', 'manish jadhav');
7  INSERT INTO student VALUES ('2', 'jay patel');
8
9  -- SQL Dump for subject
10 CREATE TABLE subject (subjectid int, subjectname varchar(20), studentid int, PRIMARY KEY (subjectid), FOREIGN KEY (studentid) REFERENCES student(studentid));
11
12 INSERT INTO subject VALUES ('1', 'dwmt', '1');
13 INSERT INTO subject VALUES ('2', 'cloud', '2');
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

studentid	studentname
1	manish jadhav
2	jay patel
NULL	NULL

student 1

Output

Action Output

#	Time	Action	Message
1	22:29:49	Could not connect, server may not be running.	Unable to connect to localhost
2	22:31:21	use testing	0 row(s) affected
3	22:32:13	CREATE TABLE student (studentid int, studentname varchar(20), PRIMARY KEY (studentid))	0 row(s) affected
4	22:32:13	INSERT INTO student VALUES ('1', 'manish jadhav')	1 row(s) affected
5	22:32:13	INSERT INTO student VALUES ('2', 'jay patel')	1 row(s) affected
6	22:32:13	CREATE TABLE subject (subjectid int, subjectname varchar(20), studentid int, PRIMARY KEY (subjectid), FOREIGN KEY (studentid) REF...	0 row(s) affected
7	22:32:13	INSERT INTO subject VALUES ('1', 'dwmt', '1')	1 row(s) affected
8	22:32:13	INSERT INTO subject VALUES ('2', 'cloud', '2')	1 row(s) affected
9	22:32:28	select * from student LIMIT 0, 1000	2 row(s) returned

Figure 17: SQL Workbench

ERD Testing:

```
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
3
Name of the Database
school
Please give the relation between subject and student
learns

Process finished with exit code 0
```

Figure 18: creation of school database when cardinality between student and subject is 1:1

If we insert two IDs (1 and 2) into the student table, and these IDs serve as foreign keys in the subject table, where each ID from the student table relates to only one row in the subject table, this represents a 1:1 relationship, typically represented by "-------".

```
+----------------------------------------+
|                  student               |
+----------------------------------------+
|        studentname (varchar(20))       |
|          primaryKey = studentid        |
+----------------------------------------+

+----------------------------------------+
|                  subject               |
+----------------------------------------+
|        subjectname (varchar(20))       |
|             studentid (int)            |
|          primaryKey = subjectid        |
|          foreignKey = studentid        |
|       references student.studentid     |
+----------------------------------------+


Legend:
<>------: Foreign Key Relationship (1:N)
--------: Foreign Key Relationship (1:1)
         learns
student--------subject
```

Figure 19: ERD for database school where cardinality is 1:1 relationship

Inserting the data in subject table for making the cardinality between student and suject becomes 1: N

```
Welcome ! Choose an option
1. Write queries
2. Export Data and Structure
3. ERD
4. Exit
1
SQL> use school;
Entered into database
SQL> insert into subject values(3,"ASDC",1);
Data Inserted Successfully
SQL> select * from subject;
subjectid   subjectname studentid
1   dwmt    1
2   cloud   2
3   asdc    1
SQL>
```

Figure 20: Inserting Data into subject table

Now, as we insert data where one row in the Student table is related to multiple rows in the Subject table, the cardinality changes to 1
. This relationship is typically represented in an ERD between entities as "<>---------".

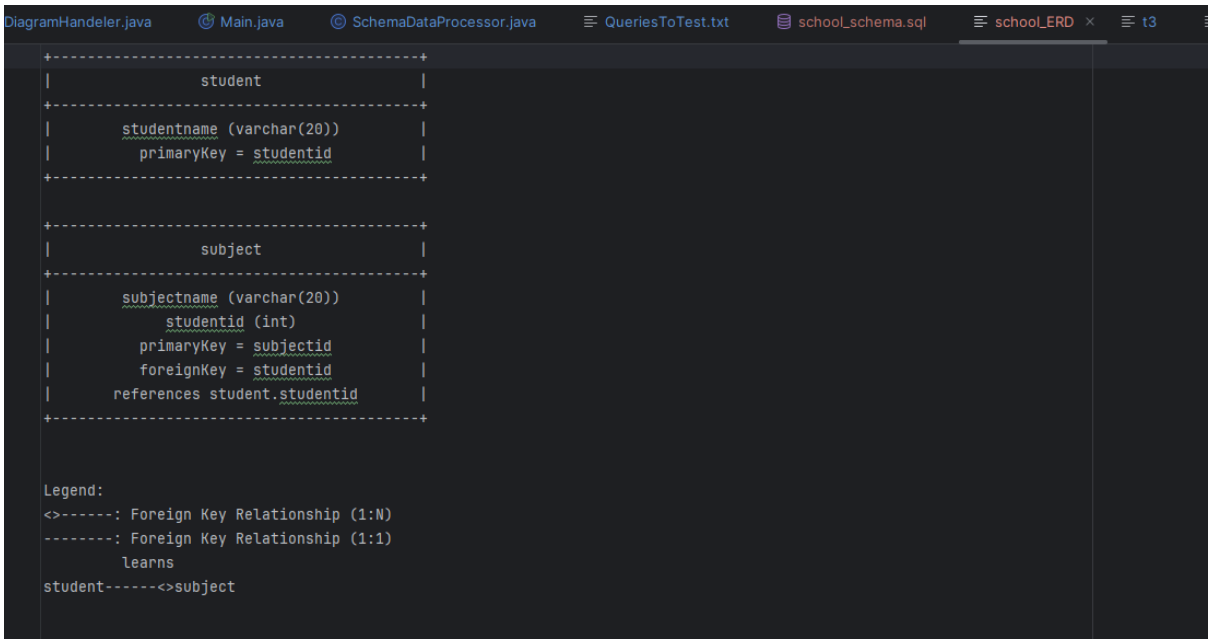

Figure 21: ERD for database school where cardinality is 1: N relationship

Meeting Logs

Date	Time	Attendees	Agenda	Meeting Type	Meeting Recording Link
01-07-2024	3:30-4:00	Manish, Jay	Going through remaining modules and deciding which ones to implement	Online	https://bit.ly/3Y1Xca9
06-07-2024	5:30-6:00	Manish, Jay	Integrating modules and testing the project	Online	Meeting in _General_-20240712_224301-Meeting Recording.mp4

References

[1] P. M. G. L. Slavica Kordić, "A Generator of SQL Schema Specifications," [Online]. Available: https://www.researchgate.net/publication/220117777_A_Generator_of_SQL_Schema_Specifications. [Accessed 08 07 2024].