SCC201 2021/22 Coursework Assignment

Agent: Alex Chalakov

Jan 2022

To all people of planet Earth: NB the following section is "written in Hero's language", an equivalent version for people can be found in the second section. NB a FAQ will be maintained for this assignment which will found on the module in moodle. This document shall be referred to as the System Requirements Companion (SRC) in any future communication. Statements like 'please refer to section 2 of the SRC' may be used.

1 INCOMING MESSAGE : CONFIDENTIAL, TOP SECRET

IMPORTANT COMMUNICATION:

THE DESIGNS FOR THE INTERGALACTIC SUPER HERO DATABASE HAVE BEEN PARTIALLY RECOVERED!

Agent Indigo retrieved access to a Schrödinger file containing a Zylon Information Package (zip file) **from** a portal **on** the planet Pylong. **As** you **are** probably aware, we **are** never certain about the state of the zip file until the Schrödinger file has been opened. **In** the current **case**, the embedded *cat* command outputs the contents of the Schrödinger file **and** obfuscates the document **when** it opens. The resulting zap (Zylon Assessment Package zip file renamed so that it can be delivered **by** email) file contains the design of the database with the entity **and** attribute **names** shortened. The zp file fortunately contains **some** snippets of data for entities which should help you build a real database **from** the design (NOTE: data may be incomplete **and** may have become corrupted... the file **names in** the ENTITIES folder may **not match** the entities **in** the er diagram ... . but the attributes do **match** ...).

You have been chosen for this assignment based **on** your **unique** scc.201 training which allows you to work **on** database designs **using** abstract symbols.

Agent Indigo has intermittent access to the data portal (based **on** the allignment of galactic objects (aka galactic events aka courswork submissions)). Access to the portal allows us to populate the relation tables with additional data which is the final part to being able to **create and** interrogate the database. Agent Indigo requires coderunner credits **in order** to be able to access the relations portal during these galactic events to **get** data to populate the relation tables(**where** appropriate). Coderunner contains the private **key** for Schrödinger file which will allow the Triate to **add** queries (**using** a top secret babel module) which map the entities back to the un-obfuscated form. Once that has been done, we will have everything we need. You will therefore have to **create** access points **in** a Just Another Virtual Application which can consume the new data. Agent Pink has obtained examples of the data structures which **are** included **in** the Zylon Information Package.

WARNING: The database will be very large once fully populated with every known planet **and** human **in** the galaxy. Running queries against the

database may take a long time. It is imperative that the database is optimised for performance AND size. Agent Green was unfortunately dehydrated because they did not have the SCC.201 indexing skills needed to optimise the database. (Initial attempts to rehydrate with beer were not successful.. a clear head is needed for this task). The coderunner instance will self destruct if either the queries take too long, or the database takes up too much file space. Agent Indigo can only keep the portal to the relations' data open for a short period of time.

Agent Purple suggests that we focus on entities 'D' and 'S'. Entity 'B' is also important to understanding communication channels being used. Entity 'F' should help to determine how to best utilise each super hero.

ESSENTIAL: You must use the names of the entities and attributes provided. Not using the correct symbols will result in an F for FAILURE.

The current galactic event is starting to fade and access to coderunner and the portal is likely to end soon. You should submit TWO files (solution.sql and Populator.java) using the secure Moons Of Oburon Data Logging Environment (MOODLE).

DO NOT transmit a Random Angry Rant file (rar), because they have been known to be completely ignored by agent coderunner! Zylong Information Package files are also useless because the cat (if still alive) has been known to chew them up!

Hopefully I don't have to remind you that you are covered by the official secrets act (aka THE REGULATIONS: https://www.lancaster.ac.uk/student-and-education-services/exams-and-assessment/regulations/). Divulging your work to anyone apart from SCC.201 staff will be dealt with appropriately.

Best of luck in your valiant efforts to understanding the designs and in building a working copy of the database. The UNIVERSITY is depending on you.

COMMANDER MOBOBO

PS, we fear another COVID attack is imminent and need to find the probable birth place and the current location of the super hero who is immune from all corona viruses.

== Signal is br.ak.ng up. . ..need m.re bee. ==

2 The problem in normal language

You basically have to submit two things via moodle:

1. A sql file called *solution.sql* which has the commands needed to create and populate a sqlite database based on the ERD provided (see fig 7)

2. A MODIFIED version of the *Populator.java* file.

The coursework for this year is to take an ER diagram (using Chen notation (Chen 1976)) and create a database which will hold appropriate data. You will also have to optimise the database and extend a very small application which manages the data in the database.

The diagram (fig 7) does not use words for entities and attributes. Having worked through the module, you should now be able to apply advanced database techniques to any solution. We are not asking you to understand a real world problem, rather we are asking you to use the diagram as the starting point for creating a database where the requirements have already been decided. You will however have to look at the data provided to determine any properties of the tables you create and choose appropriate sql data types.

Example queries are also provided which show how the database is likely to be used. These queries should guide you in deciding how to optimise the database structure and indexes.

Example files are included which contain data which will need to be added to the database. The names of the csv files are NOT the names of the entities which they have the data for. You can rename the csv files to match the correct entities if you wish. You will need to complete the *methods* in *Populator.java* which will take a file as an input and populate your database appropriately. The file which was emailed to you has a working visual code setup. If you cd into the unziped folder and type *code .* as visual studio code window will be opened if you are on a machine with visual studio code installed.

You will need to provide DDL and DML queries using standard SQL (*W3 Schools: sql quickref* 2022).

It is suggested that you focus on entities 'D' and 'S'. Entity 'B' and relationship G are also important in understanding communication channels being used. Entity 'F' should help to determine how to best utilise each super hero.

The database created from your *solution.sql* file needs to be optimised so that when large amounts of data are added, moodle will not time out when testing some of the functionality.

ESSENTIAL: You must use the names of the entities and attributes provided. Not using the correct symbols will result in an F for FAILURE.

Please refer to https://www.lancaster.ac.uk/student-and-education-services/exams-and-assessment/regulations/ for the regulations relating to plagirism and collusion. Using someone elses solution is not allowed. The work should be your own. Any advice should be obtained from the SCC.201 staff.

3 Milestones

Here we suggest a series of milestones your SQL and program should reach. You do not have to develop your code in this way. (For example, we suggest you deal with INSERT statements after CREATE TABLE statements. But you could do this the other way around if you prefer). However, your work will be marked according to these milestones and how well you achieved the required outcome.

a) A single text file which contains at the start the CREATE TABLE statements that create the tables for the ENTITIES that your ER Diagram defines. (But without the primary and foreign keys being indicated).

```
1 CREATE TABLE S (r,g,a,j);
```

b) A single text file as in (a) above, but the CREATE TABLE statements include indicators of primary keys and the variable types.

```
1 CREATE TABLE S (r,g VARCHAR(64),a Date,j integer,PRIMARY KEY r);
```

c) A single text file as in (b) above, but including foreign keys.

```
CREATE TABLE S (r,g,a,j,c,PRIMARY KEY r,FOREIGN KEY(c) REFERENCES D(c));
```

d) If the database structure is modified after creation it is possible for tables to be returned in an order that would break key constraints – i.e. a table depends on a yet to be created table. CREATE statements need sorting based on foreign keys.

As with (**h**)) but ensure the CREATE TABLE statements are in the 'correct' order.

e) A single text file containing all the correct INSERT..INTO statements for data which involves **TWO** entities. e.g.
DATA FILE HERO.CSV to be inserted into S and R:

```
r,g,a,q,p
101,'Eric␣Noel','2001-12-25','Eric␣Noel', 'Santa␣Clause'
```

```
PRAGMA foreign_keys=ON;
BEGIN TRANSACTION;
-- NB might already exist --
INSERT OR IGNORE INTO S(r,g,a) VALUES( 101, 'Eric␣Noel', '2001-012-25' )
    ;
INSERT INTO R(r,q,p) VALUES (101,'Eric␣Noel','Santa␣Clause');
COMMIT;
```

To test this, we will provide you (on the Moodle page) with the data for all instances of S and R.

f) A single text file containing all the correct INSERT..INTO statements for data which involves TWO entities and a many to many relationship. e.g.
DATA FILE SPEAKS.CSV to be inserted into S,B and G:

```
r,g,a,l,h
101,'Eric␣Noel','2001-12-25','En', 'English'
101,'Eric␣Noel','2001-12-25','Hh', 'Ho␣ho␣ho'
```

```
PRAGMA foreign_keys=ON;
BEGIN TRANSACTION;
-- NB might already exist --
INSERT OR IGNORE INTO S(r,g,a) VALUES( 101, 'Eric␣Noel', '2001-02-25' );
INSERT OR IGNORE INTO B(l,h) VALUES ('En','English');
INSERT OR IGNORE INTO B(l,h) VALUES ('Hh','Ho␣ho␣ho');
INSERT INTO G(r,l) VALUES (101,'En');
INSERT INTO G(r,l) VALUES (101,'Hh');
COMMIT;
```

To test this, we will provide you (on the Moodle page) with the data for all instances of S and R.

g) A single text file containing all the INSERT..INTO statements required for the language.csv file:

```
INSERT INTO S(r,g,a) VALUES( 101, 'Eric␣Noel', '2001-012-25' );
```

NB the text field values are quoted in primes'.

h) The database also needs indexes in the database to speed up possible queries involving the combination of entities and attribute constraints in table 1. Add code to establish what indexes are present and include CREATE INDEX statements in your DDL to create these. You should have a set of statements of the form (what follows is just an example):

```
CREATE INDEX planets_id ON D (c);
```

Indexes can be specified as `ASC` (ascending) or `DESC` (descending).

i) Create views needed to implements the relationships between entities as described in table 1

j) Complete the code in *Populator.java* which will read all csv files in the ENTITIES folder and populate the **appropriate** entities. NB it may not be possible to insert all of the CSV data because it is a random sample and some foreign key dependencies may not be met! The purpose of the sample CSV files is to give you an idea of the types of attributes. The code should also be able to make relevant updates and retrieve key information. Relevant pieces of code are marked with a *// TODO*.

NB you must not add additional java libraries (jar files). Coderunner will not have the additional libraries and tests will fail if your code depends on them. There are THREE jar files included in the folder sent to you. These allow a connection to a sqlite database, a connection to csv files and the ability to run the JUnit tests. The folder provide can be edited most easily using VSCode. Navigate to the folder and type *code .*

Your solution will be tested against different sets of data. The expectation is that your solution is complete/ correct enough, at the stage you have reached, to work with any data. Marks will be awarded for how far you got with your solution, how well it addresses the issues for each stage. Some additional marks may be available for additional features – a simple example might be inclusion of additional flags like *NOT NULL* on some attributes (there are a few possible), including comments, etc. Marks may be awarded for structure, efficiency, commenting, etc.

NB VIEWS should have a DDL which creates a table with the correct columns. No entity or relationship has the name M or N on the er diagram.

3.1 Submission

If you get past milestone (**c**)), you must modify the *Populator.java* file and complete the *//TODO* sections. A *PopulatorTest.java* and associated *jar* files are included to guide you through the milestones.

3.2 Submission (to Moodle)

Checklist (for the milestone you have reached)

1. Your version of the *Populator.java* file. If you have developed your code in an IDE, do not submit the entire project. Just the Java file you modified. (You should make sure it compiles and run outside of the IDE).

2. Your sql file named: *solution.sql*

3. FINALLY and not least: Paste the contents of *solution.sql* into Question 1 of the coderunner quiz. For Question 2, paste the java code for *Populator.java* into the text box and attach a file solution.sql which contains the text from Question 1 of the coderunner quiz.

To gain marks you must submit to Moodle *solution.sql* and *Populator.java* both to the quiz and upload individual files to the assignment submission page on moodle. DO NOT SUBMIT *rar* or *zip* files.

You can paste the code into the coderunner quiz as many times as you want without penalty.

4 Example

Here are some examples:

```
1 CREATE TABLE 'S' ('r' integer,'g' varchar(64) NOT NULL,'a' date,'j' integer,
      primary key ('c'));
2 CREATE INDEX index_age ON 'S' ('j');
3 -- update the ages  -- yuck!!! sqlite3 specific
4 UPDATE 'S' SET 'j'= (strftime('%Y', 'now') - strftime('%Y', 'a')) - (
     strftime('%m-%d', 'now') < strftime('%m-%d', 'a')) ;
5 -- query which may be made by coderunner
6 CREATE VIEW 'JuniorHERO' as SELECT *,'q' as secretIdentity from 'S' natural
     join 'R' where 'S'.'j'<21;
```

NB attribute 'j' is an age which is computed from attribute 'a' which is a dob.

view name	entities	relation	conditions (if needed)
hero	S, R	ISA	
speaks	S, B	G	
person_job	S,D,T	H	
hero_mission	R, L	E	
JuniorHERO	'S' , 'R'	ISA	'S'.'j'<21

Table 1: Views which need to be created

```
c,d,e
11,Hyrakius,\N
2,Bgztl,8
17,Rimbor,30
13,Krypton,0
6,Colu,30
```

Figure 1: Partial data for an entity

You will need the views in table 1 which link entities using the respective relation:

For example the speaks view would look like:

```
CREATE VIEW 'speaks' as SELECT * from 'S' natural join 'G' natural join 'R';
```

5 Samples of data

Samples of data can be found in figures: 1,2,3,4,5 and 6. CSV data files are also found in the ENTITIES folder of the zip file provided. These files are unique to your version of the assignment.

6 ERD

References

Chen, P. P.-S. (1976), 'The entity-relationship model—toward a unified view of data', *ACM transactions on database systems (TODS)* **1**(1), 9–36.

W3 Schools: sql quickref (2022).
URL: *https://www.w3schools.com/sql/sql_quickref.asp*

```
r,b,f
14,49,Telescopic/Microscopic Vision
33,102,Superhuman strength
36,108,Magnetism manipulation
19,72,X-Ray Vision
13,39,Eidetic memory
```

Figure 2: Partial data for an entity

```
i,l
En,English
Kl,Klingon
O2,Obfiscation
Ob,Obduron
Fr,French
```

Figure 3: Partial data for an entity

```
c,h
Zorgorn,"Mission␣on␣non-existant␣planet"
Earth,"Earth␣War"
Daxam,"Planet␣Kidnap"
Apocalypse,Darkseid
```

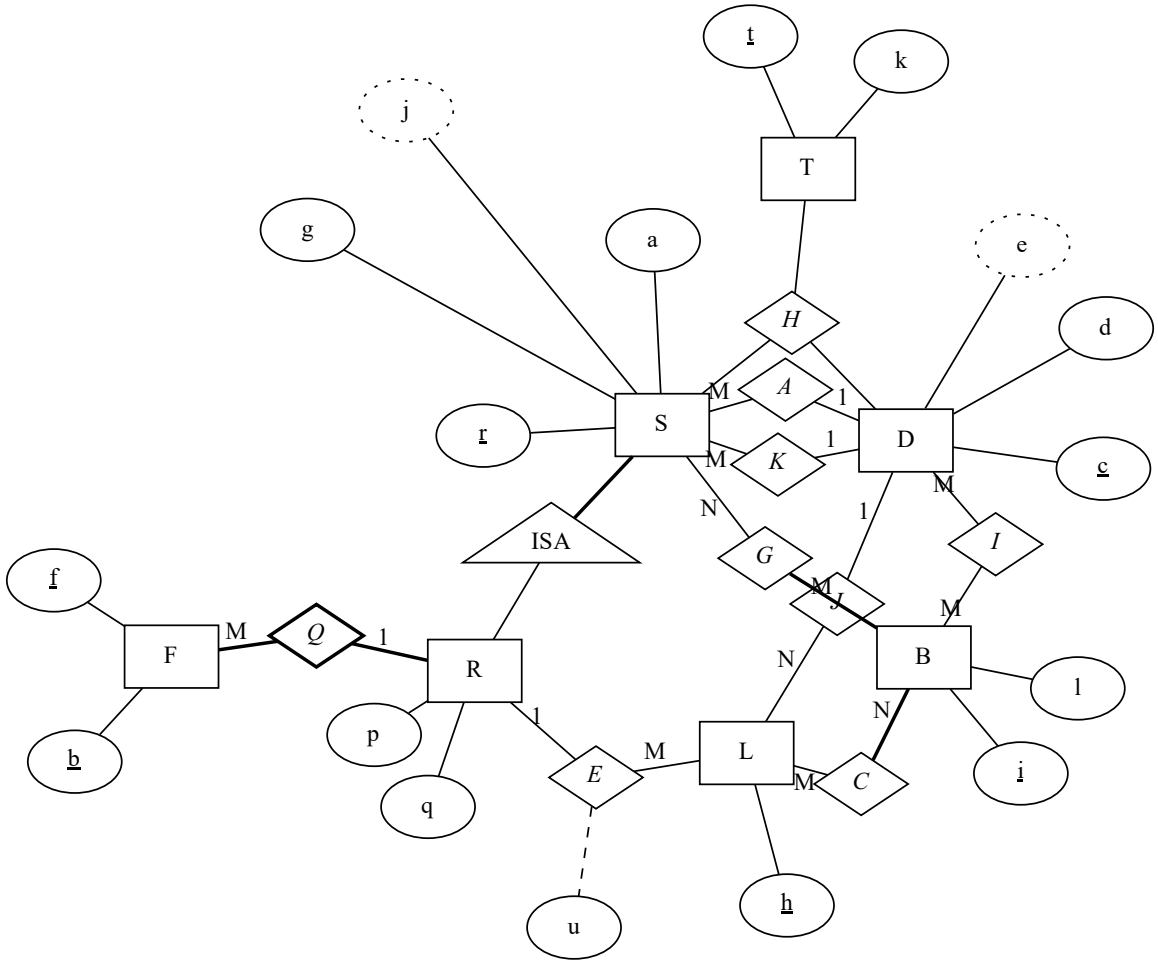
Figure 4: Partial data for an entity

```
r,q,p,c
27,"Tasmia␣Mallor␣","Shadow␣Lass␣",19
19,"Lar␣Gand␣",Mon-El,7
4,"Luornu␣Durgo␣","Triplicate␣Girl",5
24,"Andrew␣Nolan␣","Ferro␣Lad␣",10
36,"Pol␣Krinn␣","Magnetic␣Kid␣",4
```

Figure 5: Partial data for an entity

```
j,r,a,g
22,14,2000-01-01,""
22,29,2000-01-01,"Brin␣Londo␣"
22,35,2000-01-01,"Mysa␣Nal␣"
22,26,2000-01-01,"Projectra␣Wind'zzor␣"
22,19,2000-01-01,"Lar␣Gand␣"
```

Figure 6: Partial data for an entity



SCC.201 Entity Relationship Diagram
for agent:
Alex Chalakov

Figure 7: This is your own personal er diagram for the coursework.