Practical 2: Database management with SQLite

IS5102 – Database Management Systems

Due date: Monday 13th November 2023 (week 10), 21:00
50% of the coursework component (30% of the module)
(MMS is the definitive source for deadline and credit details)

In this assignment, you are asked to convert an E–R model into an SQLite database, populate it with test data, and then run a number of SQL queries.

You are expected to have read and understood all the information in this specification at least a week before the deadline. You must contact the lecturer regarding any queries well in advance of the deadline.

Summary

In lectures and exercise classes, we have spent a fair amount of time on SQL, and its use in database creation, data insertion, and database querying. This assignment will help you to develop and practise skills in all of these areas:

- converting E–R data models into SQL;
- understanding the relationship between conceptual and logical data models;
- enforcing database integrity;
- confidently translating business queries to and from SQL.

This is an **individual assignment**. You are expected to work on your own to complete the tasks in this assignment.

E–R Model

We will work with the E–R Model for a bus company scenario. Bus services are identified by a number used by customers and drivers (like 42, 99A, or X59). Each service has a starting bus station (origin) and an ending bus station (destination). Bus stations have a name and town (like Seagate in Dundee or Buchanan in Glasgow). Each service runs multiple times a day and passes a number of stops. For each run of the service, it is necessary to record its arrival time to each stop on the route, as well as the fare from the origin to each stop.

The bus company employs bus drivers, who work on hourly wages. Bus drivers are assigned to one or more bus services. To calculate their pay, it is necessary to record their hourly wage, and how many hours they have worked on each service. The bus company also employs station managers, who are paid an annual salary.

The rest of the details must be inferred from the E–R diagram on Figure 1. **In case of any ambiguities or unspecified details in the scenario above, your implementation should always match the model described by this E–R diagram**.

✱ many-to-one & total participation → no table

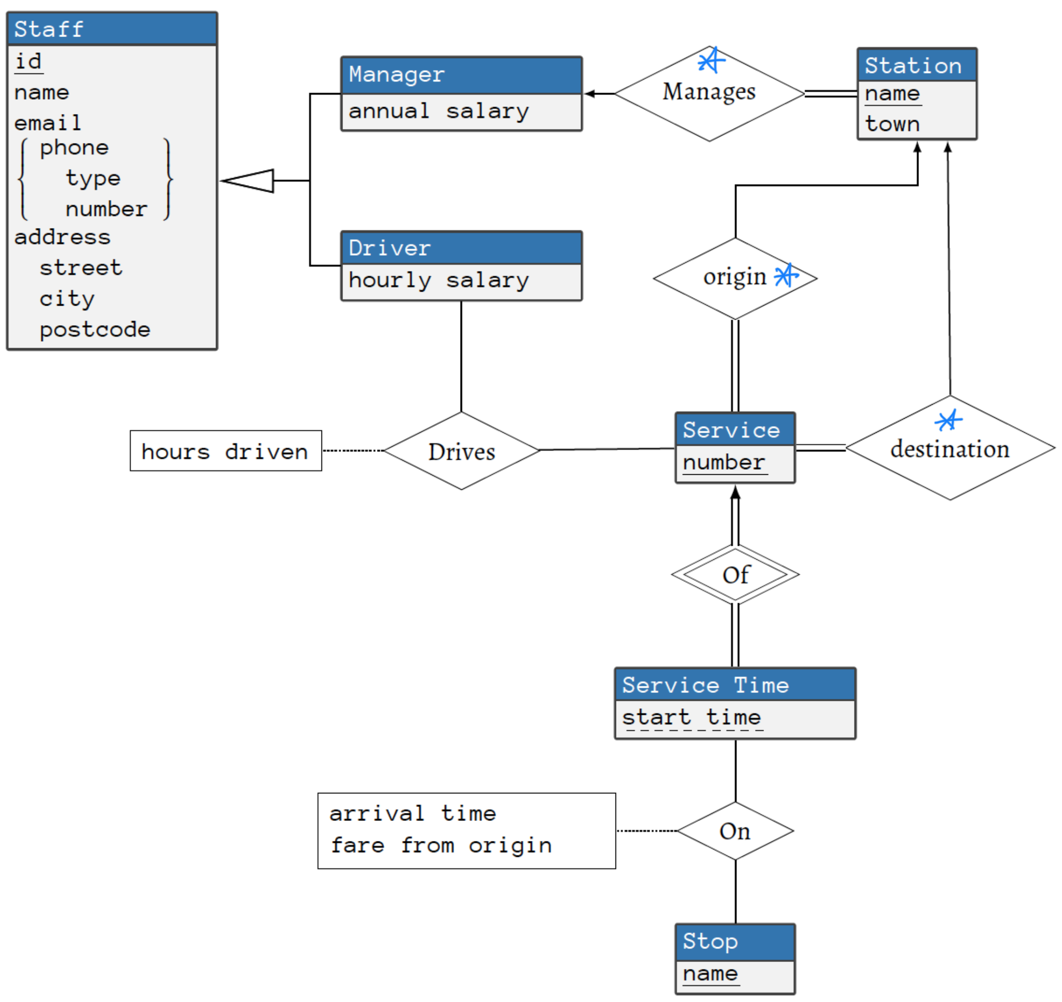

Figure 1: E–R Diagram for a bus company scenario

Notation:

- *Primary key* — underlined
- *Discriminator* — underlined with dashes
- *Defining relationship* — doubled diamond
- *Cardinality of one* — black arrow
- *Generalisation* — white arrow
- *Total participation* — doubled line
- *Multivalued attributes* — in curly braces
- *Relationship attributes* — connected to the relation with a dashed line

Tasks

Task 1: Translation

Translate the E–R model into the corresponding database schema, i.e. into the collection of relation schemas. In your report, present the relation schemas and provide a brief rationale for any design choices you make. Be sure to identify appropriate primary keys, foreign keys, and attribute types.

Task 2: SQL Data Definition

Using the Command Line Shell for SQLite (`https://sqlite.org/cli.html`) or DBeaver (`https://dbeaver.io/`), create an SQLite database, which corresponds to the database schema from Task 1.

To complete this task, you should create a plain text file called `buses.sql`, containing SQL code to define tables for each relational schema from Task 1, and then insert data into your tables.

You need to create about 10 rows per each table, and make sure that you insert rows which make each query from Task 3 to return at least 2 results (you may revisit this task later, and add extra data while working on Task 3, if necessary).

Make sure that integrity constraints are enforced. In SQLite, you have to put the line

```
PRAGMA foreign_keys = TRUE;
```

in the the file `buses.sql` in the beginning of the script, **above** the commands used to create and populate the tables. In addition to specifying primary and foreign keys, and attribute types, you are encouraged to use additional SQL features to enforce database integrity, e.g. cascading actions, default values, etc.

Populating the database with the test data, you may find it useful to invest some efforts into making the values of attributes minimally meaningful and recognisable. This will help you to check that the queries developed in Task 3 are returning correct results.

You have to ensure that the `buses.sql` script is runnable in a clean new session in the Command Line Shell for SQLite or in DBeaver, and, starting from an empty database, will create tables and populate them with data, without any error messages.

In your report, give the high-level overview of this process, describing your approach to developing and testing the SQL code to ensure the integrity of your data. If necessary, you may include fragments of the code from `buses.sql` to illustrate your statements (but you do not have to repeat the whole content of `buses.sql`, since this file will be submitted separately).

Task 3: SQL Data Manipulation

Extend the `buses.sql` script created in Task 2, adding to it (below the statements to create table and insert date), **SQL code to perform the following queries**:

1. List all services which have Seagate Bus Station in Dundee as their origin;
2. Calculate an average monthly salary of a bus station manager;
3. List the names of all drivers of services which have Edinburgh Bus Station in Edinburgh as their origin or destination, in increasing order of the amount to be paid to them for the hours driven;
4. List the manager of the most connected station, measured by the number of services which have that station as their origin or destination.
5. For the bus stop "Buchanan Gardens, St Andrews" list in the chronological order arrival times at this stop, origins, destinations, and service numbers of all bus services passing this stop between 10 am and 2 pm.

The queries above mention only the minimally necessary attributes that should be displayed. When appropriate, you should add additional attributes to make the output more informative and verifiable.

Next, **formulate at least 4 new queries and at least 3 appropriate views**, stating them as precisely as you can in plain English. **Implement them in SQL**, and include the SQL code for all the queries and views, and their output, in the report.

Writing SQL code, you should adhere to some consistent coding style (for example, to the "SQL Style Guide" by Simon Holywell, `https://www.sqlstyle.guide/` or any other consistent convention), and use sensible names of attributes, capitalisation, indentation, blank lines and comments to make your code readable and understandable.

Include all descriptions in plain English, corresponding SQL commands and their outputs in the report, either as copied and pasted text (recommended), or as screenshots.

When including SQL code and input/output from the Command Line Shell for SQLite, pay attention to its formatting - in particular, use monospace font to preserve code indentation and tables layout. It is acceptable to use screenshots instead, provided that they are of high quality. To improve screenshots readability, you can resize windows, change font size, and change settings to use dark text on a light background; note however that screenshots with SQL code are not useful since the text on them can not be searched and copied.

Task 4: Reflection

Describe briefly your experience of the process of converting the E–R model into SQL and working with data, linking back to the work done in Tasks 1–3. What did you feel you did well? What did you find challenging? What problems did you encounter, if any, and how did you try to resolve those? What would you do differently, if anything? This should be a short reflective section of your report. You can say as much or as little as you want, but as guidance, between half a page to a page should be enough.

Submission

Prepare a **zip** archive, containing two files:

- the **report** in **PDF format**, with sections per task as above;
- the `buses.sql` script in **plain text format**.

If you have correctly followed the requirements of Tasks 2 and 3, the `buses.sql` should contain complete SQL code needed to reproduce your work. It is recommended to check before submission that your code works without errors on school systems by running it in the terminal with

```
sqlite3 --init buses.sql
```

Submit the archive as specified above via MMS by the deadline, checking to make sure that the version you submit is the one you mean to submit.

Marking

The basic requirements for this practical are:

- A correct translation of the given E–R model into corresponding relational schema, together with primary and foreign keys;
- Correct SQL implementing the schema (should work in SQLite);
- SQL corresponding to the given queries, and to the self-defined queries (should work in SQLite);
- Correct and appropriate view definitions (should work in SQLite);
- Runnable without error messages, properly formatted and appropriately commented SQL code in the `buses.sql` script.

A competent work addressing all these requirements will get marks up to 16. Marks of 17 and above will be awarded for work addressing all the basic requirements and going further in terms of (one or more of):

- Evidence of an in-depth consideration of data types and constraints;
- Relevant and interesting new queries and views;
- Good style in using SQL, perhaps by the use of joins or views;
- Report demonstrating excellent understanding of SQL, of the relationship between conceptual and logical data models and their SQL implementations;
- Any other work which goes out of the basic specifications given to demonstrate excellence in understanding and/or technique, for example the use of triggers to enforce the consistency of the database.

Policies and Guidelines

Marking

See also the standard mark descriptors in the School Student Handbook, which apply as usual:
`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/feedback.html#Mark_Descriptors`

Lateness

The standard penalty for late submission applies (Scheme B: 1 mark per 8 hour period, or part thereof):
`https://info.cs.st-andrews.ac.uk/student-handbook/learning-teaching/assessment.html#Lateness_Penalties`

Good Academic Practice

The University policy on Good Academic Practice applies:
`https://www.st-andrews.ac.uk/students/rules/academicpractice/`