School of Computing Technologies

ISYS1055/ISYS3412 (Practical) Database Concepts

Assessment 2: SQL Programming and Normalisation

Assessment type: PDF Word limit: N/A

Draft Due Date: Sunday, 7 May 2023 at 23:59 (Melbourne time) – Week 9 (otherwise 2 mark deduction)

Final Due Date: Sunday, 21 May 2023 at 23:59 (Melbourne time) – Week 11

Weighting: 30% (30 marks)

Overview

The objective of this assignment is to reinforce what you have learned in the lectures and tute/ lab sessions. Specifically, it covers the advanced concepts in the relational database design, using SQL for querying a relational database and analyse different database models for different applications.

Assessment criteria

This assessment will measure the below aspects:

- Appreciate a good database design
- Apply good database design guidelines, such as normalisation, to build a well-designed database schema
- Write efficient SQL statements for retrieving data for specific user requirements
- Analyse different database models for different applications
- Write a technical report suitable for a non-technical audience, presenting your findings and recommendations.

Course learning outcomes

This assessment is relevant to the following course learning outcomes:

CLO1	describe the underlying theoretical basis of the relational database model and apply the theories into practice;
CLO3	develop a sound database design;
CLO4	develop a database based on a sound database design;
CLO5	Apply SQL as a programming language to define database schemas, update database contents and to extract data from databases for specific users' information needs.

Assessment details

Questions are worth different marks for ISYS1055 and ISYS3412. Respective marking rubrics are available in the Assignment 2 section on the Canvas shell for each course.

Part A: Relational Database Design (12 marks for ISYS1055; 14 Marks for ISYS3412)

Consider the below business rules of a consultancy company that has many departments.

- Each employee belongs to only one department. Each employee has a unique email address.
- Each department has one manager and many employees.
- Each project is owned by one department.
- An employee can work on many projects with different roles, but has one role for each project.
- A project can have many employees and they can be from different departments.
- The progress of projects is evaluated once a week by department managers. The evaluation of a project on a particular date has a rating, which is an integer between 0 (very bad) to 5 (excellent).

Consider the below relational database schema of five relations for managing projects of the company. Possibly due to an ER model not well designed or incorrect mapping from the ER to relational model, the database schema is not fully in 3NF and contains data redundancy. Some example tuple is given to explain the meaning of attributes.

The Project Management Database	
Department(deptID, deptName, manager, empID)	<2, ‘Production’, ‘E5’, ‘E4’>
Employee(empID, empName, deptID, email)	<’E5’, ‘Ann’, 4, ‘abc@usa.com’>
Project(projID, startYear, deptID)	<’P3’, 2000, 2>
EmpProj(empID, projID, role)	<’E2’, ‘P6’, ‘Designer’>
Evaluation(projID, manager, evalDate, rating)	<’P3’, ‘E5’, ’11-01-2020’, 5>

Answer following questions:

1. For each given relation, write down all functional dependencies according to the business rules. If there are no functional dependencies among attributes, you must state so. Do not write redundant or trivial functional dependencies (e.g. Email → Email).
2. For each given relation, indicate the primary key (underlined) and explain if there are any other candidate keys. Indicate any foreign keys with an asterisk (*). For each relation, write down its highest normal form and state the reasons why it doesn’t meet the requirements for the next higher level normal form. This is not required if the relation is in 3NF.
3. For each given relation not in 3NF, decompose it into relations in 3NF.
4. Following results in Question 3, replace relations after decomposition with the newly created relations (relations without decomposition remain unchanged). Merge relations with a common primary key and remove any redundant subset relations. Check if the merged relations are in 3NF, and apply further decomposition if needed until all final relations are in 3NF. Write down the final relational database schema and indicate the primary key (underlined) and foreign key(s) (with an asterisk*) in each relation.

Important: No marks are awarded to the final schema in Question 4 if you do not show the workings of FDs (Question 1) and decompositions (Questions 3).

Part B: SQL (13 marks for ISYS1055; 16 Marks for ISYS3412)

LibraryDB is a database system that keeps track of information concerning the books and their circulation in an imaginary library.

Disclaimer: The data that populates the database are artificially constructed and by no means correspond to actual real-world data.

The schema for the LibraryDB database is given below.

borrow(transactionID, personID*, borrowdate, duedate, returndate)

author(authorID, firstname, middlename, lastname)

book_copy(bookID, bookdescID*)

book(bookdescID, title, subtitle, edition, voltitle, volnumber, language, place, year, isbn, dewey, subjectID*)

borrow_copy(transactionID*, bookID*)

person(personID, firstname, middlename, lastname, address, city, postcode, phonenumber, emailaddress, studentno, idcardno)

publisher(publisherID, publisherfullname)

written_by(bookdescID*, authorID*, role)

published_by(bookdescID*, publisherID*, role)

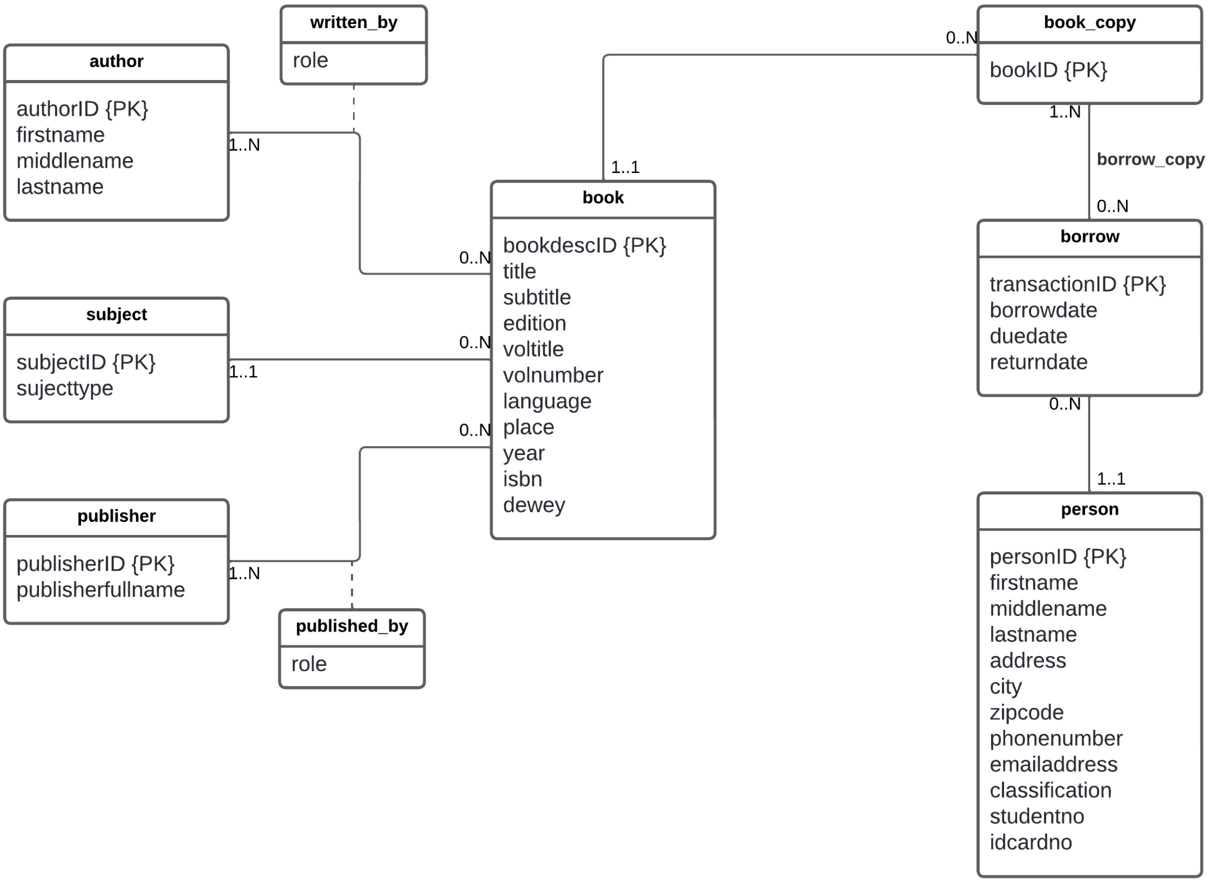
subject(subjectID, subjecttype)

The primary keys are underlined. The foreign keys are denoted by asterisks (*).

Description of the schema

- **person** -- keeps track of the people who borrow books from the library. The attributes contain personal and contact information.
- **author** -- keeps track of personal information about authors.
- **publisher** -- keeps track of the publisher information. To make simple, most of the attributes have been truncated in the sample database.
- **subject** -- this relation keeps information about the subjects on which the library collection have books (such as Mathematics, Database, etc)
- **book** -- contains information about the books that are available in the library. Every book can have one or more physical copies in the collection. Each book can have one or more authors and it is published by one or more publishers.
- **book_copy** -- keeps track of the physical copies of the books in the library collection.
- **borrow** -- keeps track of the check-ins and check-outs of the books. Every transaction is done by one person, however may involve with one or more book copies. If there is no return date, it means the book has been checked out but not returned.
- **written_by** -- associates books with authors. A book may be associated with several authors and an author may be associated with several books. There is also an attribute 'role' that specifies the role of the author for the book (author/ editor/ translator/ etc).
- **published_by** -- associates publishers with books. There is an attribute 'role' here too.
- **borrow_copy** -- associates physical copies of books with a transaction. Members are allowed to borrow several books in a single transaction.

A conceptual data model (shown as an entity-relationship diagram) which represents these data is given below.



You need to install SQLite Studio to complete questions in this part. The instructions for installing, configuring and using SQLite Studio is provided in the Week 3 Labsheet. Also included is the pre-built Library database in SQLite format (**Library.db**), available for downloading at the following address (depending on your course):

(ISYS1055) https://rmit.instructure.com/courses/107515/modules/items/4713808
(ISYS3412) https://rmit.instructure.com/courses/107459/modules/items/4686203

Write ONE SQL query for each question. Note the below points:

- Your query must use only SQLite syntax.
- The output of your queries should not include duplicates, but you should use DISTINCT only if necessary.
- A hint of the expected number of results/rows for each query is given to help you determine if your understanding/solution is completely wrong. However it is important to note that a query that returns the correct number of rows (or even the correct rows for the current instance) does not necessarily mean that the query is logically correct (or will always return the correct rows for different data).
- It is important that a query is logically correct with respect to the changing contents of the database. A query that returns the correct output for the current database content can still be logically incorrect. A query that returns nil output for the current database content can still be logically correct.

- You are advised to develop your query iteratively – start by selecting everything from the correct tables (including any appropriate JOIN conditions), then develop any GROUPing, the conditions for the GROUPing (HAVING) and WHERE conditions, the SELECT clause, and finally result ORDERing – this way you can verify that your query logic and result is correct so far before developing it further.
- While you can (and are encouraged to) include additional attributes in your select statement while debugging/developing, your final submitted answer must only return the requested attributes.
- Make use of the LOWER function to ensure your answer will work regardless of any variation in case.
- Avoid using LEFT/RIGHT joins and sub select queries unless absolutely necessary or requested.
- Do not use NATURAL joins or implicit joins unless the question explicitly requests it.
- Ensure that your queries are properly structured and formatted – Keywords in UPPERCASE, attribute and table names in lowercase, appropriate indentation and each clause on separate lines. eg:

```
SELECT ROUND(AVG(year), 0) AS "average year published"
FROM ((book b JOIN written_by wb
        ON b.bookdescid = wb.bookdescid)
             JOIN author a
                  ON wb.authorid = a.authorid)
WHERE subjectid IN (SELECT subjectid
                    FROM subject
                    WHERE subjecttype NOT LIKE '%computer%'
                    AND   subjecttype NOT LIKE '%science%')
ORDER BY "average year published" DESC;
```

If you run out of space horizontally you can reduce indentation for nested queries as follows:

```
SELECT ROUND(AVG(year), 0) AS "average year published"
FROM ((book b JOIN written_by wb
        ON b.bookdescid = wb.bookdescid)
             JOIN author a
                  ON wb.authorid = a.authorid)
WHERE subjectid IN (
        SELECT subjectid
        FROM subject
        WHERE subjecttype NOT LIKE '%computer%'
        AND   subjecttype NOT LIKE '%science%')
ORDER BY "average year published" DESC;
```

1. Display the name, and address of persons who live in the city of Portland. The output should have column headings "Name" and "Address", and respectively display first name and last name separated by a space, and address and city separated by comma and space. (104 rows)

2. Compute the total number of books for each subject. Output the subjectid together with its total number of books, in decreasing order of total number of books. (16 rows)

3. Display the first name, last name and city of persons who borrowed any book. There should not be any duplicate records in your result. (128 rows)

    a. Write your query without using the JOIN operator keyword or any sub-query.

    b. Write your query using the JOIN operator.

    c. Write your query using an IN sub query.

4. Are there books that belong to on the subject 'Databases' that are written by more than 2 authors (in whatever role)? Display the bookdescid and title of these books. (0 rows)

5. The dates in relation "borrow" are stored as REAL type data in Julian Days. You can use the built-in date() function to convert the real value into date format. For example, to find out a duedate as YYYY-MM-DD format you can use date(duedate). Find out the borrowers who returned the books after the due dates. Display book title, "borrower name" (concatenated first name and last name), date of return, due date, and how many days were delayed from the due date. Display the dates in YYYY-MM-DD format. (33 rows)

6. Find out the books that have never been borrowed. Display the book bookdescid, titles along with the year of publication, ordered alphabetically by title and then from newest to oldest. (295 rows)

7. Find out the authors (including co-authors) who have written (in the "Author" role) more than one book. Display the first name, last name of authors and their role along with the book title, ordered alphabetically by the last name and then first name of authors. (151 rows)

8. A person is reading books about Networks (title containing the string 'network' in any case). They want to find books on the topic co-written (in whatever role) by ONLY 'Tim Miller' and 'Jason Noel' (no other authors). Display the titles of these books. (0 rows)

9. Find all OTHER books written by the author of the book with title 'COMPUTER SCIENCE'. Display the title, "Author Name" (first name and last name), and year of publication of OTHER books by the same author(s). (1 row)

10. Find out the persons who borrowed books on the subject of "Image Processing". Display borrower name (first and last names, separated by a space) under the column heading of "Borrower", book title, book subject, borrow date, and return date. Borrow and return dates should be in the YYYY-MM-DD format. (5 rows)

Part C: Research questions. (5 marks for ISYS1055 only; not required for ISYS3412)

As a database expert, you are invited to write a report of maximal one page on the design and functionality of the Library database. Your report must address each of the following questions:

1. The Dewey Decimal Classification (DDC) system is used by libraries around the world to organise books for storage according to subject classes. Here is the Wikipedia entry on the list of DDC classes. The DDC is structured around ten main subject classes covering the entire world of knowledge; each main class is further structured into ten hierarchical divisions, each having ten divisions of increasing specificity. For example 000-099 is for the subject class of “Computer science, information and general works”, and 005 is specifically for “Computer programming, programs and data”. A book on SQL programming may have a Dewey call number like 005.789, where 005 is for the main subject class and 789 is for a specific subclass. Discuss if the DDC implies any data integrity constraint on the Dewey call number and the subject class of books, and if the given Library database schema enforces such constraint. Further explain if any such constraint can be enforced using the data integrity mechanism in the relational data model.

2. With the current Library database ER model, Person (as a library user) and Author are separate entity types and there are not any relationships between them. Discuss any weakness of this design choice, and give your recommendations to improve the ER model and the corresponding relational database schema.

3. Using artificial primary keys for entity types (and tables) is commonly used for ER modelling (and relational database design). Read this blog post “Natural vs. Artificial Primary Keys” on this topic. Artificial primary keys are often necessary for modelling complex real world applications, but artificial primary keys can also lead to data fragmentation – data about one data object are kept in multiple tables, which means complex join queries on many tables. In the given Library database ER diagram, “transactionID” is the artificial primary key for the “borrow” entity type and the “borrow” table. Discuss the real-world scenario of “users borrow books” modelled in the current Library database. Further discuss how the real-world scenario of “users borrow books” would be different if without the artificial primary key “transactionID”.

Submission format

You should submit one PDF document with all answers together. (You should also submit a text /sql file for your queries in Part B questions 1-10). You may use SQLiteStudio to work on your assignment. You must not submit result sets from SQL queries, only the SQL queries are to be submitted. You may use Word or any other word processor to compile your submission, by collating everything into one document. At the end, convert it into PDF format. Do not submit Word files. if that option is not available on your system there are free pdf converters online you can utilise. e.g. http://convertonlinefree.com/

Academic integrity and plagiarism

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarised, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct.

Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source
- Copyright material from the internet or databases
- Collusion between students
- Utilising AI tools to generate content without referencing (or where AI tools are specifically prohibited)

For further information on policies and procedures, please refer to the University website.

Penalties for late submissions

Assignments received late and without prior extension approval or special consideration will be penalised by a deduction of 10% of the total score possible per calendar day late for that assessment.

Assessment declaration

When you submit work electronically, you agree to the assessment declaration.