***Practical Database Concepts – A2***

***Keewoo Moon (s3831370)***

|  |  |  |
| --- | --- | --- |
| ***Part A: Relational Database Design*** | | |
| ***Q1.*** | ***1. Write down all functional dependencies***   1. ***Branch*** *(BranchNo, B\_Street, B\_suburb, B\_Postcode, StartDate, Monthly\_Bonus, StaffNo\*)*  * *FD1****:*** *BranchoNo*🡪*B\_Street, B\_suburb, B\_Postcode* * *FD2****:*** *BranchoNo*🡪*StartDate, Monthly\_Bonus* * *FD3****:*** *BranchoNo*🡪*StaffNo*  1. ***Staff*** *(StaffNo, S\_Name, S\_Address, Position, Salary, Supervisor\_No\*, BranchNo\*)*  * *FD4****:*** *StaffNo* 🡪 *S\_Name, S\_Address, Position, Salary* * *FD5****:*** *StaffNo* 🡪 *Supervisor\_No* * *FD6****:*** *StaffNo* 🡪 *BranchNo*  1. ***Property*** *(PropNo, P\_Street, P\_Suburb, P\_Postcode, Type, No\_of\_Rooms, Weekly\_Rent, Available\_for\_Rent, Ad\_on\_Other\_Website, StaffNo\*)*  * *FD7: PropNo* 🡪 *P\_Street, P\_Suburb, P\_Postcode, Type, No\_of\_Rooms, Weekly\_Rent, Available\_for\_Rent, Ad\_on\_Other\_Website* * *FD8: PropNo* 🡪 *StaffNo* | |
| ***Q2.*** | ***2. Write down the highest normal form each of these relations***   1. ***Branch***   *Telephone is a multi-valued attribute. Therefore, for branch, is in unnormalized (0NF) form.*   1. ***Staff***   *No repeating group 🡪 In 1NF*  *Primary key is a single-valued, means automatically in 2NF*  *No transitive dependencies, since there is no specific assumption being related to Position and Salary 🡪 3NF*  *The highest normal form is 3NF.*   1. ***Property***   *No repeating group 🡪 In 1NF*  *Primary key is a single-valued, means automatically in 2NF*  *No transitive dependencies, since there is no specific assumption being related to Property\_Suburb and Property\_Postcode. Also, for Ad\_on\_Other\_Website, there is no specific assumption about whether advertisements are posted on the domain website as list.*  *Therefore, it is indicated by Boolean as yes or no 🡪 3NF*  *The highest normal form is 3NF.* | |
| ***Q3.*** | ***3. If they are not in 3NF, decompose them into 3NF relations***   1. ***Branch***   *Decompose it to eliminate multi-valued Telephone attribute.*  ***Branch*** *(BranchNo, B\_Street, B\_suburb, B\_Postcode, StartDate, Monthly\_Bonus, StaffNo\*)*  ***BranchContact*** *(BranchNo\*, Telephone)*  ***BranchContact*** *does not have any non-primary key attribute. Therefore, by default, it is in 3NF.*   1. ***Staff***   *It is already in 3NF. No decomposition required.*   1. ***Property***   *It is already in 3NF. No decomposition required.* | |
| ***Q4.*** | ***4. Where possible, combine the relations resulting from Part3.***  ***R1: Branch*** *(BranchNo, B\_Street, B\_suburb, B\_Postcode, StartDate, Monthly\_Bonus, StaffNo\*)*  ***R2: BranchContact*** *(BranchNo\*, Telephone)*  ***R3: Staff*** *(StaffNo, S\_Name, S\_Address, Position, Salary, Supervisor\_No\*, BranchNo\*)*  ***R4: Property*** *(PropNo, P\_Street, P\_Suburb, P\_Postcode, Type, No\_of\_Rooms, Weekly\_Rent, Available\_for\_Rent, Ad\_on\_Other\_Website, StaffNo\*)* | |
| ***Q5.*** | ***5. Final relational schema.***  ***R1: Branch*** *(BranchNo, B\_Street, B\_suburb, B\_Postcode, StartDate, Monthly\_Bonus, StaffNo\*)*  ***R2: BranchContact*** *(BranchNo\*, Telephone)*  ***R3: Staff*** *(StaffNo, S\_Name, S\_Address, Position, Salary, Supervisor\_No\*, BranchNo\*)*  ***R4: Property*** *(PropNo, P\_Street, P\_Suburb, P\_Postcode, Type, No\_of\_Rooms, Weekly\_Rent, Available\_for\_Rent, Ad\_on\_Other\_Website, StaffNo\*)* | |
| ***(Reference List)***   * *Shaahin, M 2021, ‘Week9 Course Lectorial’, lecture notes, ISYS3412, RMIT University, viewed 05 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-9-lectorial-and-practical?module\_item\_id=3204881>* * *Shaahin, M 2021, ‘Week10 Course Lectorial’, lecture notes, ISYS3412, RMIT University, viewed 05 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-10-lecture-exercises-and-practical?module\_item\_id=3204889>* * *Santha, S 2021, ‘Week 10: Relational Database Design’, lecture notes, ISYS3412, RMIT University, viewed 05 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-10-lecture-exercises-and-practical?module\_item\_id=3204889>* * *Santha, S 2021, ‘Week 11: Relational Database Design’, lecture notes, ISYS3412, RMIT University, viewed 05 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-11-lectures?module\_item\_id=3204897>* | | |
| ***Part B: SQL*** | | |
| ***Q1.*** | | *SELECT \**  *FROM SUBJECT S*  *JOIN*  *BOOK B ON S.subjectID = B.subjectID*  *WHERE subjecttype = "DataBases";* |
| ***Q2. a*** | | *SELECT person.firstname,*  *person.lastname*  *FROM person*  *WHERE person.personID IN (*  *SELECT borrow.personID*  *FROM subject*  *JOIN*  *book ON subject.subjectID = book.subjectID*  *JOIN*  *book\_copy ON book.bookdescID = book\_copy.bookdescID*  *JOIN*  *borrow\_copy ON book\_copy.bookID = borrow\_copy.bookID*  *JOIN*  *borrow ON borrow\_copy.transactionID = borrow.transactionID*  *WHERE subject.subjecttype = "DataBases"*  *);* |
| ***Q2. b*** | | *SELECT person.firstname,*  *person.lastname*  *FROM subject*  *JOIN*  *book ON subject.subjectID = book.subjectID*  *JOIN*  *book\_copy ON book.bookdescID = book\_copy.bookdescID*  *JOIN*  *borrow\_copy ON book\_copy.bookID = borrow\_copy.bookID*  *JOIN*  *borrow ON borrow\_copy.transactionID = borrow.transactionID*  *JOIN*  *person ON borrow.personID = person.personID*  *WHERE subjecttype = "DataBases";* |
| ***Q3.*** | | *SELECT person.firstname,*  *person.middlename,*  *person.lastname*  *FROM book*  *JOIN*  *book\_copy ON book.bookdescID = book\_copy.bookdescID*  *JOIN*  *borrow\_copy ON book\_copy.bookID = borrow\_copy.bookID*  *JOIN*  *borrow ON borrow\_copy.transactionID = borrow.transactionID*  *JOIN*  *person ON borrow.personID = person.personID*  *WHERE title = "COMPUTER SCIENCE";* |
| ***Q4.*** | | *SELECT book.title*  *FROM book*  *JOIN*  *book\_copy ON book.bookdescID = book\_copy.bookdescID*  *JOIN*  *borrow\_copy ON book\_copy.bookID = borrow\_copy.bookID*  *JOIN*  *borrow ON borrow\_copy.transactionID = borrow.transactionID*  *WHERE returndate IS NULL;* |
| ***Q5. a*** | | *SELECT \**  *FROM book*  *WHERE title LIKE '%DATA%'*  *EXCEPT*  *SELECT \**  *FROM book*  *WHERE title = "DATA PROCESSING CONCEPTS";* |
| ***Q5. b*** | | *SELECT book.title*  *FROM book*  *JOIN*  *written\_by ON book.bookdescID = written\_by.bookdescID*  *JOIN*  *author ON written\_by.authorID = author.authorID*  *WHERE author.authorID = (*  *SELECT author.authorID*  *FROM book*  *JOIN*  *written\_by ON book.bookdescID = written\_by.bookdescID*  *JOIN*  *author ON written\_by.authorID = author.authorID*  *WHERE book.title = "DATA PROCESSING CONCEPTS"*  *)*  *AND*  *title LIKE '%DATA%';* |
| ***Q6.*** | | *SELECT publisher.publisherfullname*  *FROM publisher*  *JOIN*  *published\_by ON publisher.publisherID = published\_by.publisherID*  *JOIN*  *book ON published\_by.bookdescID = book.bookdescID*  *JOIN*  *subject ON book.subjectID = subject.subjectID*  *WHERE subject.SUBJECTTYPE <> 'DataBases';* |
| ***Q7. a*** | | *SELECT person.FIRSTNAME,*  *person.middlename,*  *person.lastname*  *FROM person*  *LEFT OUTER JOIN*  *borrow ON person.personID = borrow.personID*  *WHERE transactionID IS NULL;* |
| ***Q7. b*** | | *SELECT person.firstname,*  *person.middlename,*  *person.lastname*  *FROM person*  *WHERE person.personID NOT IN (*  *SELECT borrow.personID*  *FROM borrow*  *);* |
| ***Q8.*** | | *SELECT publisher.publisherfullname*  *FROM publisher*  *JOIN*  *published\_by ON publisher.publisherID = published\_by.publisherID*  *JOIN*  *book ON published\_by.bookdescID = book.bookdescID*  *JOIN*  *written\_by ON book.bookdescID = written\_by.bookdescID*  *WHERE written\_By.authorID NOT IN (*  *SELECT author.authorID*  *FROM author*  *WHERE author.firstname = "ALFRED" AND*  *author.lastname = "AHO"*  *);* |
| ***Q9.*** | | *SELECT person.firstname,*  *person.lastname,*  *count (\*) AS NoOfBook*  *FROM borrow*  *JOIN*  *person ON borrow.personID = person.personID*  *GROUP BY person.firstname,*  *person.lastname*  *HAVING count (\*) > 3;* |
| ***Q10.*** | | *SELECT person.firstname,*  *person.lastname*  *FROM borrow*  *JOIN*  *person ON borrow.personID = person.personID*  *GROUP BY person.firstname,*  *person.lastname*  *HAVING count (\*) = (*  *SELECT max(borrowCount)*  *FROM (*  *SELECT count (\*) AS borrowCount*  *FROM borrow*  *GROUP BY personID*  *)*  *);* |
| ***Q11.*** | | *<Updated relational database schema>*  ***borrow*** *(transactionID, personID\*, borrowdate, 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\*, duedate)*  ***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)*  *<A portion of the updated ER model>* |
| ***Q12.*** | | *<Updated relational database schema>*  ***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)*  ***hold*** *(holdID, holdstartdate, holdendddate, bookID\*, personID\*)*  *<A portion of the updated ER model>* |
| ***Q13.*** | | *The original database schema doesn’t support that operation so that updated the original schema and original ER model to corresponding that operation.*  *<Updated relational database schema>*  ***borrow*** *(transactionID, personID\*, borrowdate, duedate)*  ***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\*, returndate)*  ***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)*  *<A portion of the updated ER model>* |
| ***(Reference List)***   * *Emily, Z/Shaahin, M 2021, ‘Lectorial 6: SQL Continued’, lecture notes, ISYS3412, RMIT University, viewed 03 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-6-learning-materials-and-activities?module\_item\_id=3204856>* * *Santha, S 2021, ‘Week 7: Tute/Lab – SQL Programming’, lecture notes, ISYS3412, RMIT University, viewed 03 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-7-lectorial-and-practical?module\_item\_id=3204865>* * *Emily, Z/Shaahin, M 2021, ‘Lectorial 7: SQL Continued’, lecture notes, ISYS3412, RMIT University, viewed 03 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-7-learning-materials-and-activities?module\_item\_id=3204863>* * *Santha, S 2021, ‘Week 8: Tute/Lab – SQL Programming’, lecture notes, ISYS3412, RMIT University, viewed 03 October 2021, <https://rmit.instructure.com/courses/79700/pages/week-8-lectorial-and-practical?module\_item\_id=3204873>* | | |
| ***Part C: Research Question*** | | |
| For the requirement in this scenario, we need a database that can enter information referred to the COVID-19 vaccine online booking system. For this vaccine online booking application, there are going to be multiple data points recorded including a list of 28 mandatory attributes which are needed for the vaccination booking process. Therefore, we can consider two kinds of database methods as a candidate for this booking application. The first is the traditional relational database or RDBMS that allows for data to be stored in tables and connected via relationships between tables. The second option is to use a NoSQL database that stores data in documents which can be a JSON object with a hierarchy containing all the related information into a single document. To identify more specific features, we are now going to compare the characteristics of each type of database and the suitability for this COVID-19 vaccine online booking application.    NoSQL database is a non-relational DBMS designed differently from traditional relational database management systems (RDBMS). One of the characteristics is that it has horizontal scalability that can be expanded on a large scale, so enormous data can be flexibly processed. In addition, the main characteristic that compares with a relational database is that without a schema such as a table column, key values for simple search and additional operations are optimized in a distributed environment, and latency and throughput are excellent. Because of these features, it provides a mechanism for storing and retrieving data using a relatively less restrictive consistency model than RDBMS. Motivation for this approach can be said to include simplification of design, horizontal scalability, and detailed control. Therefore, the NoSQL database is excellently recognized for its ease of development, functionality, and performance in expansion as it uses various data models, including documents, graphs, memory, and searches. Currently, NoSQL is widely used for commercial use of big data and real-time web applications, and MongoDB is representative.  The relational database is defined simply as database types that allow you to store and access data points related to each other. Normally a relational database is based on a relational model, an intuitive and simple way to display data in a table, and is a collection of predefined data items. These items consist of a set of tables with columns and rows. Tables are used to hold information about objects to be displayed in the database. Each column in the table stores a specific type of data, and the field stores the actual value of the attribute. Rows in the table represent a collection of related values for an object or entity. Each row in a table can be represented by a unique identifier called the primary key, and rows between multiple tables can be related using a foreign key.  Based on the requirements and what can be expected from a booking system, we will need to record the following things. Personal details of the individual, information on multiple vaccines, eligibility for each vaccine, availability of times for vaccination, venues, a record of shots, and potentially more tables for administration and record keeping. Due to the potential of high demand, we need to consider traffic volume issues in the system and need to be consistent without resulting in duplicate appointments for the same vaccination booking spot.  If the RDBMS options is chosen, all necessary information will be recorded into tables after the necessary data model is identified after normalisation. This may end up with tables such as Person, Vaccination, Venue, Available Appointments and more. Data can be retrieved via joins between these tables when necessary and each table will contain consistent information about the entity it represents.  If the NoSQL database is chosen all the information that will be required needs to be modelled carefully to avoid the need for any joins as NoSQL databases do not allow joins across multiple collections. A Document Store like MongoDB will make it easier to store detailed information about the person or the vaccine without needing to create multiple tables like in RDBMS's. But because NoSQL cannot guarantee ACID properties, we need to create an application that may need to deal with duplicate bookings against a single vaccination slot or the opposite of cancellations that may have not been recorded and handle these exceptions in the application layer. If this booking system is to be used at a very large scale (State level with millions of people), databases like MongoDB might struggle with consistency as it can only be eventually consistent on reading replicas.  In conclusion, according to this scenario, for the COVID-19 vaccine online booking system, we would recommend a traditional relational database system such as Oracle and MySQL as a database solution instead of a no-SQL database system such as Mongo DB. As shown in the scenario, the data structure is clear and there is less need for changes that a NoSQL database can provide. It also requires a system that uses systematic and well-defined data, ensures data integrity, and frequent changes to the data. Most social networking platforms, such as Facebook and Twitter, use these databases a lot to store details such as user information and messages. These databases have a semi-definite structure, but schemas are bound to change over time and require highly scalable systems and immediate consistency is not a must and eventual consistency is sufficient. The COVID-19 vaccine online booking system, on the other hand, requires a high level of consistency to ensure all users of the booking system are never double booked to find themselves at an appointment without an available dose or oppositely a dose to be wasted if a cancellation is not recorded due to the health risks in this global pandemic. Therefore, the recommendation is to use a Relational Database such as Oracle, MSQL or MySQL where consistency can be guaranteed, is ACID compliant and can scale to accommodate a large number of users while providing consistency. | | |
| ***(Reference List)*** *Software Testing Help 2021, ‘SQL Vs NoSQL Exact Differences and Know When to Use NoSQL and SQL’, Software Testing Help, viewed 07 October 2021, <https://www.softwaretestinghelp.com/sql-vs-nosql/#SQL\_vs\_NoSQL\_Security>**Software Testing Help 2021, ‘MONGODB Vs MySQL – Key Differences Explained’, Software Testing Help, viewed 07 October 2021, <https://www.softwaretestinghelp.com/mongodb-vs-mysql/>**Stack Exchange 2021, ‘Concurrency in NoSQL’, Stack Exchange, viewed 07 October 2021, <https://dba.stackexchange.com/questions/171011/concurrency-in-nosql>**Study tonight 2021, ‘Mongo DB vs SQL Databases’, Study tonight, viewed 07 October 2021, <https://www.studytonight.com/mongodb/mongodb-vs-rdbms>**Study tonight 2021, ‘Introduction to NoSQL’, Study tonight, viewed 07 October 2021, <https://www.studytonight.com/mongodb/what-is-nosql>**Study tonight 2021, ‘Introduction to Mongo DB’, Study tonight, viewed 07 October 2021, <https://www.studytonight.com/mongodb/introduction-to-mongodb>**Mongo DB 2021, ‘Comparing MongoDB vs MySQL’, Mongo DB, viewed 07 October 2021, <https://www.mongodb.com/en-us/compare/mongodb-mysql>**W3spoint 2021, ‘MONGODB Vs MySQL – Key Differences Explained’, W3spoint, viewed 08 October 2021, <https://www.w3spoint.com/mysql-advantages-disadvantages>**Makble 2021, ‘The advantage and disadvantages of MySQL’, Makble, viewed 07 October 2021, <http://makble.com/the-advantages-and-disadvantages-of-mysql>**Nitya, R 2021, ‘What are the advantages and disadvantages of using MySQL stored procedures?’, turotialspoint, viewed 08 October 2021, <https://www.tutorialspoint.com/What-are-the-advantages-and-disadvantages-of-using-MySQL-stored-procedures>* | | |