Database Concepts ISYS1055

Assignment 2

Name: Syed Hariz
Student ID: 3701799
s3701799@student.rmit.edu.au

Table of Contents

Part A: Relational Database Design

Author

Functional Dependencies from ER Diagram

FD1: Email → Name, Address

Since Telephone is a multi-valued attribute, it does not depend on Email. Therefore, there is no functional dependency exist between Email and Telephone attributes.

Determine the correct Primary key based on relational schema

Author (Email, Name, Address, Telephone1, Telephone2, Telephone3)

Based on the relation given and comparing to FD1 stated above, Email is the correct primary key because it can determine both Name and Address attributes. For an attribute to be a primary key, it must be able to determine all attributes and does not depend on any other attribute to be determined by.

Highest Normal Form

Author (Email, Name, Address, Telephone1, Telephone2, Telephone3)

This relation is a result of an incorrect attempt to address multi-valued attribute. On a quick glance, you may consider this relation is in 1NF because no multi-valued attributes are visible. However, Telephone 1, Telephone2 and Telephone3 is created to store the same value. This is not an acceptable solution, so we have to normalize it.

Normalization:

Author1 (Email, Name, Address, {Telephone})

The curly-braces denote that Telephone is a multi-valued attribute. After normalization, the highest normal form of this relation is 0NF or un-normalized form.

Decomposition

From Author1, to address this problem, we have to remove the offending attribute (Telephone) along with the primary key (Email) and form a separate relation.

Final relation schema:

Author2 (Email, Name, Address)

Author3 (Email*, Telephone)

After decomposition, both Author2 and Author3 relations are in 3NF because there are no transitive dependencies exist in both relation.

Publisher

Functional Dependencies from ER Diagram

FD1: Name → Address, URL, ABN

Determine the correct Primary key based on relational schema

Publisher (Name, Address, URL, ABN)

Based on the relation schema given, Name is the correct primary key because if we look into the Functional dependency, FD1 above, Name attribute can determine all other attributes from this relation.

Highest Normal Form

Publisher (Name, Address, URL, ABN)

From this relation, we can safely assume that highest normal form for Publisher relation is in 3NF because there are no transitive dependencies in this relation.

Decomposition

No decomposition is needed because the relation has met the 3NF requirement.

Final relation schema:

Publisher (Name, Address, URL, ABN)

WrittenBy

Functional Dependencies from ER Diagram

Based on the ER Diagram provided, WrittenBy is a Many to Many relation between Book and Author. Therefore, it inherits both primary keys from Book and Author and makes them as its primary and foreign key written as below;

WrittenBy (Email*, ISBN*)

Since, WrittenBy does not hold any attributes in its relation, there are no functional dependencies exist as its primary key cannot determine any attributes.

Determine the correct Primary key based on relational schema

WrittenBy (Email*, ISBN*, Title*)

Reflecting from the explanation that was stated above, WrittenBy inherits both primary keys from Book and Author entity and makes them as its own primary and foreign key. Therefore, the primary key stated in this relation is correct.

However, a functional dependency exist between Title and ISBN.

FD1: ISBN → Title

Since Title is a part of Book entity attributes and ISBN is the primary key of Book entity, Title is functional dependant on ISBN as showed in FD1 and since WrittenBy does not hold any attributes, this would make Title as a redundant attribute in this relation. The removal of this redundancy will be processed through next step.

Highest Normal Form

From the relation schema that was provided, we can write its functional dependency as follow:

FD2: Email, ISBN → Title

The highest normal form of this relation is in 1NF because it does not fulfil the 2NF requirement since there is a partial dependency in the relation. Based on FD1, Title can only be determined by ISBN, it does not depend on Email which makes FD2 a proof of partial dependency existence.

Decomposition

Final relation schema:

WrittenBy1 (Email*, ISBN*)

WrittenBy2 (ISBN, Title)

Both WrittenBy1 and WrittenBy2 relation meets 2NF and 3NF requirement because no partial dependency and transitive dependency exist in both relation.

Book

Functional Dependencies from ER Diagram

FD1: ISBN → Title, Edition, year, ListPrice

Determine the correct Primary key based on relational schema

Book (ISBN, Title, Edition, Year, ListPrice, PublisherName*)

Referring to FD1, ISBN can determine all other attributes of Book entity. Therefore, the primary key that was selected in this relation is incorrect because it cannot determine PublisherName.

However, PublisherName is a primary key for Publisher entity and since Book and Publisher entity is a Many to Many relationship, Book relation does not inherit Publisher's primary key as a foreign key in its relation. Decomposition process is needed for this relation as there is a redundant attribute.

Highest Normal Form

Book1 (ISBN, Title, Edition, Year, ListPrice, PublisherName*)

The highest normal form of this relation is 0NF because PublisherName cannot be determined by any other attributes in Book relation.

Decomposition

From Book1, to address this problem, we have to remove the offending attribute (PublisherName) along with the primary key (ISBN) and form a separate relation.

Final relation schema:

Book1 (ISBN, Title, Edition, Year, ListPrice)

Book2 (ISBN*, PublisherName*)

After decomposition, both Book1 and Book2 relations are in 3NF because there are no transitive dependencies exist in both relation.

Warehouse

Functional Dependencies from ER Diagram

FD1: Code → Address

Determine the correct Primary key based on relational schema

Warehouse (Code, Address)

Comparing both FD1 and the relation schema that is provided, the primary key that was stated in Warehouse relation is correct since Code can determine the Address attribute of this relation and it cannot be determined by any other attributes.

Highest Normal Form

Warehouse (Code, Address)

The highest normal form of this relation is in 3NF because it meets the requirement of 3NF which is no transitive dependency must exist.

Decomposition

No decomposition is needed because the relation has met the 3NF requirement.

Final relation schema:

Warehouse (Code, Address)

StockedAt

Functional Dependencies from ER Diagram

FD1: ISBN, Code → StockQty

Determine the correct Primary key based on relational schema

StockedAt (ISBN*, Code*, StockQty)

Since StockedAt is a Many to Many relation between Book and Warehouse entity, it inherits both primary key from both entities and makes it as its primary and foreign key. Therefore, the primary key that was selected in this relation is correct.

Highest Normal Form

We can safely assume that this relation is in 3NF because StockQty depends on both primary keys, ISBN and Code. Logically thinking, ISBN is the primary key of Book entity which can determine the other attributes of a Book, while a Book is being stored in a warehouse, they all have different codes which makes the same Book copies unique. Therefore, StockQty must depend on both of these information to get its value.

Decomposition

No decomposition is needed as the relation is considered as 3NF.

Final relational schema:

StockedAt (ISBN*, Code*, StockQty)

ShoppingCart

Functional Dependencies from ER Diagram

FD1: CartID → TimeStamp

FD2: CartID → Email

FD3: CartID, ISBN → BuyPrice, Qty

Based on the ER Diagram and FD2, the relationship between ShoppingCart and Customer exists a functional dependency between CartID and Email. Since the relation between 2 entities is 1 to Many and ShoppingCart is the M-side of the relation, ShoppingCart will inherit Customer's primary key (Email) in its relation.

FD3 shows that, the relation between ShoppingCart and Book is a Many to Many relationship. And between those two entities, the relation holds BuyPrice and Qty attributes. Therefore, both of attributes must rely on ShoppingCart's and Book's primary key as their dependency.

Determine the correct Primary key based on relational schema

ShoppingCart (CartID, TimeStamp, ISBN, BuyPrice, Qty)

Comparing all of the FDs stated above and ShoppingCart relation that is provided, the primary key that was selected is incorrect since CartID cannot determine other attributes of ShoppingCart relation except TimeStamp.

There are non-trivial FD found in this relation. Such data redundancy like ISBN, BuyPrice and Qty. Therefore decomposition is needed.

Highest Normal Form

ShoppingCart1 (CartID, TimeStamp, ISBN, BuyPrice, Qty)

The highest normal form of this relation is 2NF because BuyPrice and Qty is dependent on a non-primary attribute which is ISBN.

Decomposition

By using FD1, FD2 and FD3 we can decompose ShoppingCart1 to form a separate relation.

ShoppingCart2 (CartID, TimeStamp)

ShoppingCart3 (CartID*, ISBN*, BuyPrice, Qty)

ShoppingCart4 (CartID, Email*)

Since ShoppingCart2, ShoppingCart3 and ShoppingCart4 does not violate the 3NF requirement, all 3 relations are successfully decomposed.

However, since ShoppingCart4 and ShoppingCart2 shares the same primary key we can join them together as 1 relation and discard both of them.

ShoppingCart5 (CartID, TimeStamp, Email*)

Final relation schema:

ShoppingCart3 (CartID*, ISBN*, BuyPrice, Qty)

ShoppingCart5 (CartID, TimeStamp, Email*)

Customer

Functional Dependencies from ER Diagram

FD1: Email → Name, Address

FD2: CartID → Email

Determine the correct Primary key based on relational schema

Customer (Email, Name, Address, CartID*)

The primary key that was stated in this relation is incorrect since FD1 proves that Email can determine both Name and Address attribute of Customer Entity but not CartID.

Highest Normal Form

Customer1 (Email, Name, Address, CartID*)

Based on ER Diagram that is provided Customer and ShoppingCart is a 1 to Many relation. Therefore, the primary key of ShoppingCart (CartID) will not be inherited as a foreign key in Customer relation because Customer is at the 1-side of the relationship.

By comparing both FDs and this relation. The highest normal form of this relation is 2NF since there is transitive dependency proven at FD1 and FD2.

Decomposition

Customer2 (Email, Name, Address)

Customer3 (CartID, Email*)

Post-process

Joining relation process:

Author2 (Email, Name, Address)

Author3 (Email*, Telephone)

Publisher (Name, Address, URL, ABN)

WrittenBy1 (Email*, ISBN*)

WrittenBy2 (ISBN, Title)

Book1 (ISBN, Title, Edition, Year, ListPrice)

Book2 (ISBN*, PublisherName*)

Warehouse (Code, Address)

StockedAt (ISBN*, Code*, StockQty)

ShoppingCart3 (CartID*, ISBN*, BuyPrice, Qty)

ShoppingCart5 (CartID, TimeStamp, Email*)

Customer2 (Email, Name, Address)

Customer3 (CartID, Email*)

Based on the final relational schema, we can join WrittenBy2 and Book1 as a single relation since they both share the same primary key. We can also join Customer3 and ShoppingCart5 together since they share the same primary key.

After joining process:

Author2 (Email, Name, Address)

Author3 (Email*, Telephone)

Publisher (Name, Address, URL, ABN)

WrittenBy1 (Email*, ISBN*)

Book1 (ISBN, Title, Edition, Year, ListPrice)

Book2 (ISBN*, PublisherName*)

Warehouse (Code, Address)

StockedAt (ISBN*, Code*, StockQty)

ShoppingCart3 (CartID*, ISBN*, BuyPrice, Qty)

ShoppingCart5 (CartID, TimeStamp, Email*)

Customer2 (Email, Name, Address)

Renaming the relations with meaningful name and final relational schema:

Author (Email, Name, Address)

Telephone (Email*, Telephone)

Publisher (Name, Address, URL, ABN)

WrittenBy (Email*, ISBN*)

Book (ISBN, Title, Edition, Year, ListPrice)

PublishedBy (ISBN*, Name*)

Warehouse (Code, Address)

StockedAt (ISBN*, Code*, StockQty)

AddedTo (CartID*, ISBN*, BuyPrice, Qty)

ShoppingCart (CartID, TimeStamp, Email*)

Customer (Email, Name, Address)

Part B: SQL

1a)
```
select FIRSTNAME, LASTNAME from author
where AUTHORID in ( select AUTHORID from written_by where BOOKDESCID in (select
BOOKDESCID from book where subjectid = 1));
```

1b)
```
select au.FIRSTNAME, au.LASTNAME from
(author au join written_by wb on au.AUTHORID = wb.AUTHORID)
join book bk on wb.BOOKDESCID = bk.BOOKDESCID and bk.SUBJECTID=1;
```

2)
```
select au.FIRSTNAME, au.MIDDLENAME, au.LASTNAME
from book b join written_by wb on wb.BOOKDESCID = b.BOOKDESCID
join author au on wb.AUTHORID = au.AUTHORID where wb.ROLE = 'Translator'
and b.TITLE = upper('American Electrician''s Handbook');
```

3)
```
select b.TITLE from book b
join book_copy bc on bc.BOOKDESCID = b.BOOKDESCID
where bc.BOOKID not in ( select bocp.BOOKID from borrow_copy bocp
join borrow bo on bocp.TRANSACTIONID = bo.TRANSACTIONID) group by b.TITLE;
```

4a)
```
select b.TITLE
from book b join book_copy bc on bc.BOOKDESCID = b.BOOKDESCID
where b.TITLE like '%DATABASE%' and bc.BOOKID not in (
select bocp.BOOKID from borrow_copy bocp join borrow bo on bocp.TRANSACTIONID =
bo.TRANSACTIONID);
```

4b)
```
select * from book b
join written_by wb on wb.BOOKDESCID = b.BOOKDESCID
join author a on a.AUTHORID = wb.AUTHORID
join book_copy bc on bc.BOOKDESCID = b.BOOKDESCID
where b.BOOKDESCID in (
select BOOKDESCID from written_by where AUTHORID in (
select AUTHORID from written_by
where BOOKDESCID in (select BOOKDESCID from book where TITLE = 'PRINCIPLES AND
PRACTICE OF DATABASE SYSTEMS' )))
and bc.BOOKID not in (select bocp.BOOKID from borrow_copy bocp join borrow bo on
bocp.TRANSACTIONID = bo.TRANSACTIONID);
```

5)
```
select pu.PUBLISHERFULLNAME, b.TITLE from publisher pu
join published_by puby on pu.PUBLISHERID = puby.PUBLISHERID
join book b on b.BOOKDESCID = puby.BOOKDESCID
where b.SUBJECTID in ( select SUBJECTID from subject where SUBJECTTYPE = 'DataBases' );
```

6a)
```
select b.TITLE
from book b left outer join book_copy bc on bc.BOOKDESCID = b.BOOKDESCID
where bc.BOOKID not in ( select bocp.BOOKID from borrow_copy bocp
join borrow bo on bocp.TRANSACTIONID = bo.TRANSACTIONID) group by b.TITLE;
```

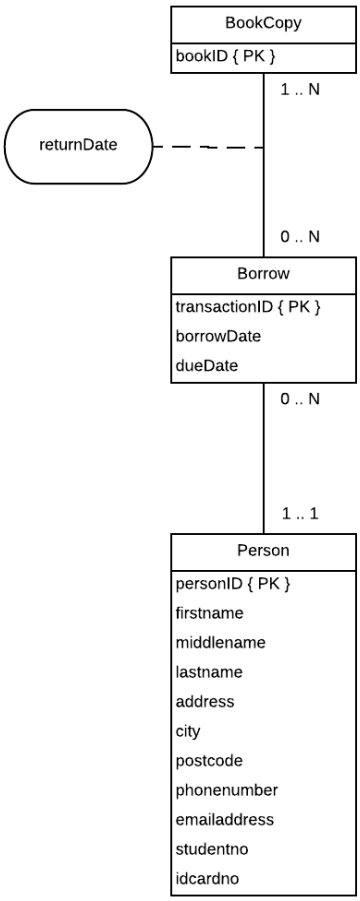
6b)
```
select b.TITLE from book b
join book_copy bc on bc.BOOKDESCID = b.BOOKDESCID
where bc.BOOKID not in ( select bocp.BOOKID from borrow_copy bocp
join borrow bo on bocp.TRANSACTIONID = bo.TRANSACTIONID) group by b.TITLE;
```

7)
```
select PUBLISHERFULLNAME from publisher pu where
EXISTS (
select a.AUTHORID, wb.BOOKDESCID, p.PUBLISHERID, p.PUBLISHERFULLNAME from
author a join written_by wb on a.AUTHORID = wb.AUTHORID
join published_by pb on pb.BOOKDESCID = wb.BOOKDESCID
join publisher p on p.PUBLISHERID = pb.PUBLISHERID
where a.FIRSTNAME = 'ALFRED'
and a.LASTNAME = 'AHO'
and pu.PUBLISHERID = p.PUBLISHERID);
```

8)
```
select a.FIRSTNAME, a.LASTNAME, count(*) as NoBook
from author a
join written_by wb on wb.AUTHORID = a.AUTHORID
join book b on b.BOOKDESCID = wb.BOOKDESCID
where wb.ROLE = 'Author'
group by wb.AUTHORID
having count(*) > 3;
```

9)
```
select bc.BOOKDESCID, count(*) as sum, b.TITLE
from book_copy bc join book b on bc.BOOKDESCID = b.BOOKDESCID
group by bc.BOOKDESCID
having count(*) = (
select max(sum) as max from (
select BOOKDESCID, count(*) as sum from book_copy
group by BOOKDESCID));
```

10)

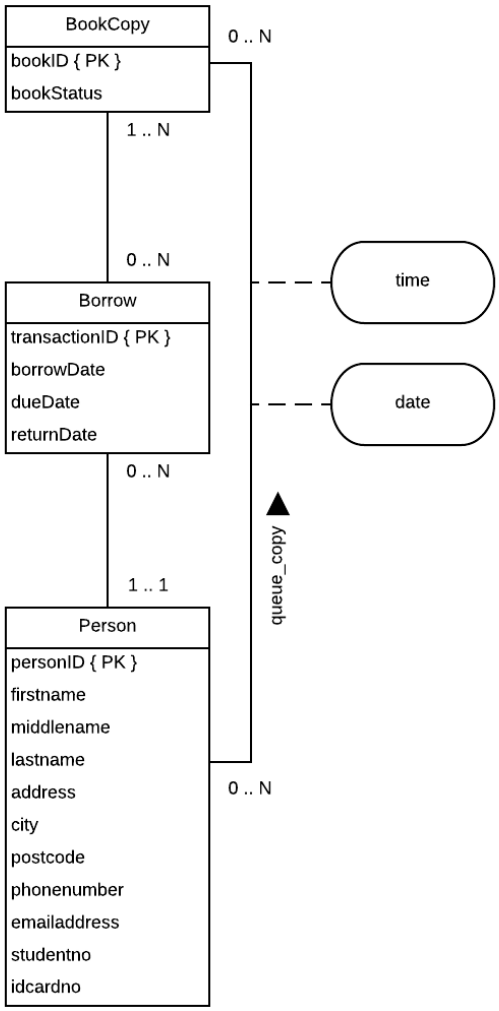


Since each transaction can have Many book stored, the solution to it is to take returnDate attribute out from Borrow entity and set it as a relation attribute between Borrow entity and BookCopy entity. Since Borrow and BookCopy is a Many to Many relationship a new relation scheme can be form written as follow;

borrow_copy (transactionID*, bookID*, returnDate)

Now, borrow_copy can keep track on having different returnDate on each transaction.

11)



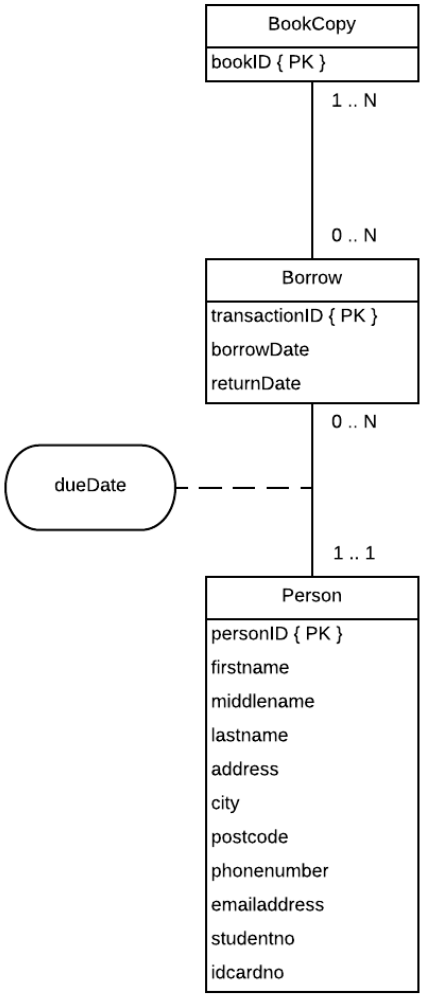
A book copy can be queued by 0 to Many person and a person can queue 0 to Many books. Since it is a Many to Many relation, we can derive a relation scheme as follows;

queue_copy (personID*, bookdescID*, date, time)

The primary key of this relation is both BookCopy's and Person's primary keys. Therefore, queue_copy attributes (date & time) are dependent on them.

A bookStatus attribute is also added in BookCopy entity to indicate the availability of the book. In this way, we are able to record information on holding books, which book is on hold, which person is queuing for it and what time and date it was queued for.

12 a)



YES.

By removing the duedate attribute from borrow entity and make it as an attribute between Person and Borrow entity. This would mean, the dueDate can be extended and both primary keys of Person and Borrow can determine it.

12 b)

NO.

Since relationship of two entities are 1 to 1, 1 to Many and Many to Many, it is impossible to have a restriction of exactly 2 times interaction between 2 entities.

Part C: Research Question

Topic: The ideal database system between Relational Database and No-SQL Database for PTV application

Introduction

The purpose of this report is to discuss the difference between Relational Database Systems and No-SQL Database Systems. Today, database storage have played a pivotal role for many companies to organize collection of various forms of data. It is also known as a structured set of data that can be accessed, updated and managed easily through a computer program. Database storage is a platform or a system that enables its users to manage large chunks of data, security of data and data integrity.

When it comes to choosing a database system, one of the biggest decision is picking a traditional relational database systems (eg. Oracle and SQL Server) or non-relational No-SQL (eg. MongoDB) data structure. While both systems remains a viable option, there are certain key differences between the two which separates them between a better or less likely approach. (1) Therefore, it is important that users recognize their capabilities when making a decision on using the best database system.

This report summarizes findings between Relational Database Systems and No-SQL Database Systems. The report also discusses and analyse the advantages and disadvantages of both systems, which will be used as a reference to reflect final recommendations and conclusion on which database system offer better benefits in terms of its' approach and structure for PTV application.

Findings

Relational Database (SQL)	Non-relational Database (NoSQL)
MySQL	MongoDB
Oracle	CouchDB
SQL Server	BigTable

Diagram 1: Types of Database system

A relational database system such as SQL Server uses structured query language (SQL) for defining and manipulation data. SQL has been the traditional and optimum approach because SQL is one of the most versatile and extremely powerful options available in the market, making it the safer choice which especially works great for complex queries. (2)

On the other hand, A NoSQL database has dynamic schema for unstructured data and its data can be stored in many ways, it can be document-oriented, column-oriented, graph based or organized as KeyValue store. (3) This flexibility means that, you can create documents without having to first define their structure, each document can have its own unique structure and you can add fields as you go.

The difference between them are established in the way they are designed, which data types they support and most importantly how they store them.

Advantages

Relational Database System

Maturity

MySQL is an extremely established database system, meaning that there's a huge community within it, extensive testing for users to refer and quite stable system.

Security

Data in tables within a relational database management systems can limit user accessibility to only particular users.

Replicable

The MySQL database can be replicated across multiple nodes, meaning that workload can be reduced.

Accuracy

Data is being stored just once which will eliminate data duplication in the system. Matching entries can be located with ease by using database queries.

Non-Relational Database System

Storing large volumes of data without structure

A NoSQL database system does not limit storable data types. Users can also add new types as business needs change.

Speed

It is well known for its high performance for simple queries.

Flexibility

You can add new columns or fields on MongoDB without affecting existing rows or application performance.

Manageability

The database does not require a database administrator. Since it is highly user-friendly, it can be used by both developers and administrators.

Disadvantages

Relational Database System

Structured Limits

Relational databases impose limits on field lengths. Therefore, when users design a database, they have to specify the amount of data that can be fit into the field. Since some of the search queries are shorter than the actual, this may lead to data loss.

Costly

The expense of maintaining and setting up a database up a relational database system is relatively high. (4) In order to set up a relational database system, users must purchase specialized software. Large companies would need more robust database. Therefore, large companies would need to hire a programmer to create a relational database using Structured Query Language (SQL) and a database administrator to maintain the system once it is built.

Performance

A major constraint and one of the disadvantage of using relational database system is machine performance. (4) If the number of tables between which relationships to be established are large, the tables themselves effect the performance in responding to the SQL queries. When using relational database system, data is usually being normalized meaning lots of joins are produced which affects the performance speed.

Non-Relational Database System

Community is not well defined

While the continuing rapid growth of Non-relational Database System, the NoSQL community is relatively new and lacks the maturity of having large user base comparing to MySQL. Even though NoSQL is growing rapidly, MySQL is hard to bear for its large network of highly experienced end users.

Lack of reporting tools

A major problem with NoSQL database systems is the lack of reporting tools for performance testing and analysis. (5) Unlike MySQL, you can find a wide collection of reporting tools to help you validate your application.

Lack of standardization

In order for NoSQL to grow, it needs a standard and preferred language like in SQL. NoSQL's lack of standardization can cause a problem during business migration. Besides this, standardization is one of the most important aspect for a database industry.

Conclusion

In conclusion, since PTV is a fast growing and popular application, my recommendation is to use a Relational Database system as its database system. The reason to back this recommendation is because Relational Database is a really established platform with a large amount community. Therefore, there are many extensive testing that were made for users to refer when facing a problem. This would help the people of the company to tackle a problem quickly. Since Relational Database system has a large community within it, this would amount a large number of experienced and professional users using this approach. In return, PTV can easily hire database professionals as part of their employee or for consultations. Even though it is costly to require a database administrator to maintain the database system, with the large amount of PTV users, budget of having a great database system should not be a problem.

In terms of security, Relational Database system is proved to be secured since the common practice of it is restricting accessibility to particular users to manage and handle the data. Therefore, personal data of PTV's users are not being exposed to a major vast of crowd. However, NoSQL system uses cloud to store its data this would put PTV's user data at risk since it depends on a 3rd party to store their data. The best way of protecting a customer's personal data is to have it stored within the company itself. Hence, the information only flows from the customer to the company without involving any other parties which in return will give great reliability and security towards the company. Since security is a pivotal quality that companies should practice, NoSQL can be a liability since it is lack of security and mainly because their designer focuses on other purposes than security. Low amount of reporting tool for NoSQL would make it harder for companies to validate their application. Generally, NoSQL databases solution is still fresh in the market and it does not reach its full maturity since many vulnerabilities in it can be found. Another benefit PTV can gain from using Relational Database system is the structured approach they have in its system. Relational database system is best for companies with structured data, since PTV has a fixed timetable attributes it will be easier to relocate such information from the database table.

To conclude, Relational Database system is still the best approach for PTV that is why it has been the optimum choice in many companies. Even though NoSQL is growing rapidly today, it will face huge growth and improvement in the future once it solves its security problems and system flaws. (6)

Word count: 1297 words

References

1. Brody A. panoply. [Online].; 2017. Available from: https://blog.panoply.io/sql-or-nosql-that-is-the-question.

2. Supriya S. Pore SBP. Comparative Study of SQL & NoSQL Databases. ; Volume 4(Issue 5).

3. Xplenty. Medium. [Online].; 2017. Available from: https://medium.com/xplenty-blog/the-sql-vs-nosql-difference-mysql-vs-mongodb-32c9980e67b2.

4. K J. acodez. [Online].; 2018. Available from: https://acodez.in/advantages-disadvantages-rdbms/#Disadvantages.

5. Gajani A. Monitis. [Online].; 2017. Available from: http://www.monitis.com/blog/cc-in-review-the-key-differences-between-sql-and-nosql-dbs/.

6. Mohamed A. Mohamed OGAMOI. Relational vs. NoSQL Databases: A Survey. 2014 May; Volume 3(Issue 3).