










# ASSINGMENT 2

jackson piper  
RMIT S3893796

## Task 1

### Task 1.1

Query		History
<pre>1 Select DeptName, Count(deptname) AS DeptNameCount 2 From Department 3 WHERE deptName IS NOT NULL 4 GROUP BY DeptName 5 ORDER BY DeptNameCount DESC, DeptName 6 ; 7 8</pre>		
Grid view		Form view
        		Total rows loaded: 31
	DeptName	DeptNameCount
1	Computer Science	11
2	Department of Computer Science	3
3	School of Information Technology	3
4	CRC for Distributed Systems Technology	2
5	Applied Computing and Mathematics	1
6	Basser Department of Computer Science	1
7	Centre for Information Technology Research	1
8	Computer Science Department	1
9	Computer Science Laboratory	1
10	Computer Science and Computer Engineering	1
11	Computing	1
12	Computing Science	1
13	Department Mathematics and Computing	1
14	Department of Computing	1

## Task 1.2

research

Query History

```
1 Select t1.Title, t1.Givenname, t1.Famname, Count(t1.ACNUM) As PaperCount
2 From Academic t1 join Interest t2 on t1.ACNUM = t2.ACNUM
3 join author t4 on t1.ACNUM = t4.ACNUM
4 join paper t3 on t4.PANUM = t3.PANUM
5 Where t1.title = 'Dr'
6 AND t2.DESCRIP like '%database%'
7 AND t3.TITLE like '%database%'
8 GROUP BY t1.ACNUM
9 HAVING Count(t1.ACNUM) >1
10 Order By PaperCount Desc, Famname
11 ;
12
```

Grid view Form view

Total rows loaded: 4

	Title	Givenname	Famname	PaperCount
1	Dr	Gerard	Shetty	6
2	Dr	Ron	Das	3
3	Dr	Lee	Reichelt	2
4	Dr	Xuming	Rooney	2

## Task 2

### Task 2.1

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons for query execution, formatting, and saving. Below the toolbar, there are two tabs: "Query" and "History". The "Query" tab is active, displaying a SQL query:

```
1 Select Deptnum, Instname, Deptname, State
2 From Department
3
4 Where Deptnum not IN (Select deptnum from Academic)
5 And State IS NOT NULL
6 ;
```

Below the query editor, there are two tabs: "Grid view" and "Form view". The "Grid view" tab is active, displaying a table of results. The table has four columns: "Deptnum", "Instname", "Deptname", and "State". There are two rows of data:

	Deptnum	Instname	Deptname	State
1	131	University of Queensland	CRC for Distributed Systems Technology	Qld
2	143	Monash University	Gippsland School of Computing and I.T.	VIC

Below the table, there is a status bar that says "Total rows loaded: 2".

## Task 2.2

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons for execution, saving, and editing. Below the toolbar, there are tabs for 'Query' and 'History'. The 'Query' tab is active, displaying a SQL query:

```
1 Select d.Deptnum, d.Instname, d.Deptname, d.State
2 From Department d
3
4 Where NOT EXISTS (
5 Select A.Deptnum
6 from Academic a
7 Where a.DEPTNUM = d.DEPTNUM)
8 AND State IS NOT NULL
9 ;
```

Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, showing a table of results. The table has four columns: 'Deptnum', 'Instname', 'Deptname', and 'State'. There are two rows of data. The first row is highlighted with a blue background. To the right of the table, it says 'Total rows loaded: 2'.

	Deptnum	Instname	Deptname	State
1	131	University of Queensland	CRC for Distributed Systems Technology	Qld
2	143	Monash University	Gippsland School of Computing and I.T.	VIC

## Task 2.3

research

Query History

```
1 Select DISTINCT ACNUM, Givename, Famname
2 from ACADEMIC t1 join author t2
3 on t1.acnum = t2.acnum
4 where t2.PANUM in (
5     Select t3.PANUM
6     from Author t3
7     where t3.ACNUM = 202
8 )
9 AND t1.Acnum != 202
10
11 ORDER BY t1.FAMNAME, t1.GIVENAME
12 ;
```

Grid view Form view

Total rows loaded: 11

	ACNUM	Givename	Famname
1	290	Jonathan	Garber
2	232	Pete	Grohol
3	264	Eduardo	Harding
4	200	Thomas	Hedges
5	199	Svante	Karr
6	201	Hans	Kellner
7	282	Roger	Khadye
8	203	Iain	Mitchell
9	206	Edward	Robinson
10	204	Rob	Schaeffer
11	205	Will	Zakel

## Task 2.4

The screenshot shows a database query tool interface. At the top, there is a toolbar with various icons for execution, saving, and editing. Below the toolbar, there are tabs for 'Query' and 'History'. The 'Query' tab is active, displaying a SQL query:

```
1 SELECT Deptname, MAX(Amount)
2 from ( Select Deptname, COUNT(DEPTNAME) AS Amount
3       from department
4       Group by Deptname
5 )
6 ;|
```

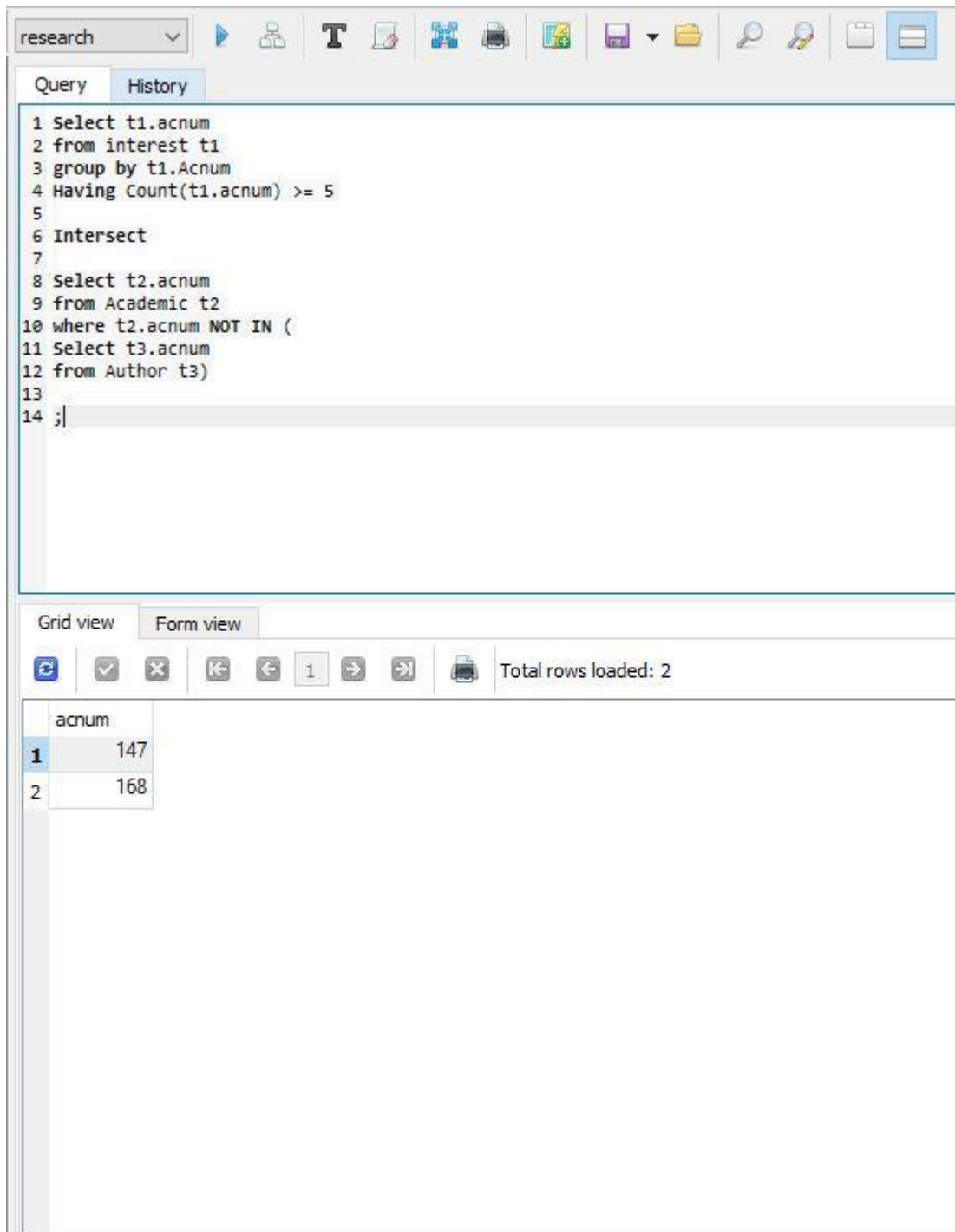
Below the query editor, there are tabs for 'Grid view' and 'Form view'. The 'Grid view' tab is active, showing a table with two columns: 'Deptname' and 'MAX(Amount)'. The table contains one row of data:

	Deptname	MAX(Amount)
1	Computer Science	11

Below the table, there is a toolbar with navigation icons and a status bar that reads 'Total rows loaded: 1'.

## Task 3

### Task 3.1



The screenshot displays a database query tool interface. At the top, there is a toolbar with various icons for navigation and editing. Below the toolbar, the 'Query' tab is active, showing the following SQL query:

```
1 Select t1.acnum
2 from interest t1
3 group by t1.Acnum
4 Having Count(t1.acnum) >= 5
5
6 Intersect
7
8 Select t2.acnum
9 from Academic t2
10 where t2.acnum NOT IN (
11 Select t3.acnum
12 from Author t3)
13
14 ;
```

Below the query editor, the 'Grid view' tab is active, showing the results of the query. The results are displayed in a table with two columns: 'acnum' and a row number. The table contains two rows of data:

	acnum
1	147
2	168

At the bottom right of the grid view, it indicates 'Total rows loaded: 2'.



## Task 3.2

research

Query History

```
1 Select DISTINCT Acnum
2 from Interest
3 where fieldNum In(
4 Select fieldnum
5 from interest
6 where acnum=114
7 )
8
9 except
10
11 Select Acnum
12 from interest
13 where acnum = 114
14 ;
```

Grid view Form view

Total rows loaded: 39

	Acnum
1	101
2	103
3	108
4	117
5	121
6	129
7	133
8	140
9	166
10	169
11	184
12	188
13	242
14	251

## Task 4

### Task 4.1

#### Course

CCode  $\rightarrow$  Name

#### CourseOffering

CCode, OCode  $\rightarrow$  Start Date, Weeks, Has Break

#### Contract

CNumber  $\rightarrow$  Start Date, End Date, Salary, Is Full Time, Is Casual

Start Date  $\rightarrow$  SNo, Staff Name

#### Staff

SNo  $\rightarrow$  Staff Name, Academic Level

#### Lecture

CCode, OCode, SNo  $\rightarrow$  (They are all determinants in this table)

#### Tutor

CCode, OCode, SNo  $\rightarrow$  Hours, Rate

#### Coordinate

CCode, OCode, SNo  $\rightarrow$  Hours

## Task 4.2

Using the methods provided to us in Week 4 'Mapping the ER Model to the Relational Model' and then figuring out from there the functional dependencies within those tables. While for the most part they are the same as the above functional dependencies there are some key differences.

The following functional dependencies are different. Any not shown should be assumed the same as the previous task

Course Offering

CCode, OCode → Start Date, Weeks, Has Breaks

CCode, OCode → SNo

The difference for Course Offering is that SNo is included as There is a 1: N relationship between the Staff Table and the Course Offering table. This also makes sense as a staff member must be assigned to the Course to Offer It.

Contract

CNumber → Start Date, End Date, Salary, Is Full Time, Is Casual

StartDate → SNo

The main change here is to remove staff name as following the methods provided, staff name is not required in the Contract Table as it is neither a Partial Key, Foreign Key or Primary Key in either table. Again, SNo is included due the nature of the relationship between the tables.

So, the New Functional Dependencies based on the Entity Relationship Map provided are as follows:

Course

CCode → Name

Course Offering

CCode, OCode → Start Date, Weeks, Has Breaks

CCode, OCode → SNo

Contract

CNumber → Start Date, End Date, Salary, Is Full Time, Is Casual

StartDate → SNo

Staff

SNo → Staff Name, Academic Level

Lecture

CCode, OCode, SNo → (They are all determinants in this table)

Tutor

CCode, OCode, SNo  $\rightarrow$  Hours, Rate

Coordinate

CCode, OCode, SNo  $\rightarrow$  Hours

### Task 4.3

#### Course

CCode → Name

3NF

#### Course Offering

CCode, OCode → Start Date, Weeks, Has Breaks

CCode, OCode → SNo

3NF

#### Contract

CNumber → Start Date, End Date, Salary, Is Full Time, Is Casual

StartDate → SNo

2NF as there is a transitive dependency between CNumber → Start Date → SNo. The easiest way to achieve 3NF would be to create a table in between the Contract and Staff Table, Staff Contract(SNo\*, CNumber\*, Start Date, End Date) and to reduce data redundancy remove Start Date and End Date from the contract table

CNumber → Start Date, End Date, Salary, Is Full Time, Is Casual

Which would be 3NF

#### Staff

SNo → Staff Name, Academic Level

You might argue that the staff name holds multiple values, but I am assuming it is just the staff member's last name. If it had multiple values it would need to be broken down as follows

SNo → Staff First Name, Staff Last Name, Academic Level.

But for the remainder of the assignment and as it is not indicated in the ERM that it is storing multiple values I will continue the rest of the tasks under the assumption that it is a single value store.

3NF

#### Lecture

CCode, OCode, SNo → (They are all determinants in this table)

3NF

#### Tutor

CCode, OCode, SNo → Hours, Rate

3NF

#### Coordinate

CCode, OCode, SNo → Hours

3NF

#### Task 4.4

Course(CCode, Name)

CourseOffering(CCode\*, OCode, Start Date, Weeks, Has Break, SNo\*)

Contract(CNumber, Salary, Is Full Time, Is Casual)

StaffContract(CNumber\*, SNo\*, Start Date, End Date)

Staff(SNo, Staff Name, Academic Level)

Lecture(SNo\*, CCode\*, OCode\*)

Tutor(SNo\*, CCode\*, OCode\*, Hours, Rate)

Coordinate(SNo\*, CCode\*, OCode\*, Hours)

## Task 5

The Department of Health is exploring an online vaccination system that will allow Australians to perform three main tasks via the internet. The three main tasks are booking a vaccine, reviewing their immunisation history and seeing the recommendations for future vaccines based upon their vaccine history.

The objective of this report is to consider the two commercially available database platforms either a traditional relational database system or a No-SQL database system that would be best suited as the backend database solution for this new system. The report will look at the advantages and disadvantages of each, Touching on scope, scalability, performance issue and data security among other points.

It will also compare two real world examples of how a Traditional Relational Database System and a Np-SQL database system have performed and compare them to this proposed system. Both of these platforms could provide the support for this new system as they can both scale to the amount of data needed while differing in performance and data consistency. However, given the track record of Traditional Relation Database Systems and the amount of data that will be handled in addition to the level of data consistency required. This report would recommend that a traditional database system would be the best choice for this system.

Data stored in a traditional database system is stored in rows and columns to make up different tables when collected. The information stored in each table is one snippet of data relevant to the whole system. While some tables may not interact with others in the system they all are imported to keep the data organised and consistent when additional data is added to the system. The difference between normal files being saved and stored on your computer and using a database is the relationships that the tables (or 'files') have with each other and interact with each other. It also allows developers to use Structured Query Language (SQL) to write queries that allow data points from multiple tables ('files') to be analysed in ways that are not available through conventional data storage options. This often leads to traditional relational databases being referred to as SQL databases. Traditional Database systems come with a variety of advantages and disadvantages for proposed projects including scalability, performance and cost among others. To properly integrate a database system a Relational Database Management System is needed, this is the software that represents the data in tables with rows and columns. Two popular Relational Database management systems are Oracle which was established in 1977, and SQL Server which was established in 1989 by Microsoft.

While SQL databases were the first set of Database management systems designed to work with large amounts of data that is very structured and rigid in its design. Non-SQL Database systems are a newer way to handle data of varying sizes without the need to have it structured as rigidly as a traditional relational database. The main reason it is referred to as a Non-SQL database is because it doesn't make use of SQL. Instead using graph databases, key-value pairs or document-oriented storage. MongoDB is a document-based database. The key difference here is that the data is not stored in tables with rows and columns, each having a relationship between each other, so they all fit into the strict schema of a SQL database. Instead, data is stored on multiple tables such as Employee information, Contact information, Department etc. it is stored as one object with multiple field-value pairs. These documents are then collected into Collections with other documents with similar fields. This is the biggest difference as the documents forming a collection don't have to conform to any particular schema although some document databases do offer schema

validation. The advantages for flexibility of data being stored is a clear example of an advantage of Non-SQL database systems.

One of the advantages that both databases have been their scalability both horizontally and vertically. Scaling often refers to adding or sometimes removing large portions of data. For example, a business that is scaling up means its employing more. However, the business can't upscale if it doesn't have the room or can't afford to. While MongoDB allows for it to scale vertically well, other non-SQL databases lack the same customisation. However traditional relational databases scale well both vertically by adding more CPU, RAM or SSD, or horizontally through sharding and indexing. The reliability on traditional databases to have the functionality to scale as needed with the project, especially with the amount of data that may eventually be stored. Is evident as to why it is more suitable for this project.

The flexibility of data storage is a key aspect to be considered for this project. While Non-SQL databases offer unmatched flexibility in organizing data, this also creates potential discrepancies between different sets of documents in a collection. On the other hand, traditional SQL-based databases offer a structured approach to storing data, reducing the potential for inconsistencies. With the type of data being collected for this project, and the consistency needed. Makes the flexibility of Non-SQL databases redundant, as strict schemas help led to a greater data consistency. Data consistency is important as it helps to ensure the data is accurate and as vaccinations can be potentially life threatening depending on allergies and reactions to previous vaccines. It is important that the data is up to date.

One disadvantage that a traditional relational database has opposed to a Non-SQL database is the potential performance issues. Non-SQL databases such as MongoDB and optimised around dealing with processing high volume and velocity data. This is ideal for applications that generate large amounts of data as the project has the potential too with the large number of potential users in Australia. Traditional databases struggle as the amount and different types of data being input may require multiple updates of the schema which could lead to degradation of the performance. However, because of the strict nature of the data being generated by this project, this data degradation and need for schema updates is unlikely. However, if more features were to be added to the project in the future, then this risk increases greatly. Meaning that if the project is not going to change in the future a traditional relational database would perform adequately unless additional changes were made to the scope of the project.

Data security is also something to consider for this project as much of the information being stored in the system is personal and also medical information for potentially millions of users in Australia. While Non-SQL databases systems like MongoDB do have security features built into them including mechanisms for secure data management, however when comparing them to traditional relation databases they are still new and are not as tested. Traditional databases systems also have a large community of driven developers who have been working on improving security measures for a lot longer. There is also a large market for third party security software unlike with Non-SQL systems.

Netflix is an example of a company who started off with a small number of features and low user load but has now ballooned to a company that offers a variety of products across a range of products. While having originally started with an Oracle a traditional relationship database, they have now moved to a Non-SQL based database. The reasoning behind their move was to reduce downtime occurring with schema updates and managing accessibility to the cloud. They are currently using amazon-based web services. This move also helped to save costs as system and database administrators were not required to be hired to build their own datacentre. Similar to Netflix if this current project reaches a certain scope, it may be beneficial to start with one style and swap to another.



However, the guardian is an example of a company that evaluated their business and decided that encryption was more important to them than the flexibility that a Non-SQL database. While deciding to switch to a cloud-based system similar to Netflix they also made their decision to opt out of the obvious cloud-based choice DynamoDB, because it did not offer encryption at rest (at the time of their change). While the platform that they did choose to go with was PostgreSQL. Similarly, to the proposed project, the guardian had large data that needed to be safely encrypted and Non-SQL databases just couldn't provide that at the time.

Something to note with both of these examples is they moved to a system reliant on Amazon Web Services (AWS). More and more companies are moving to AWS to help reduce the operating cost and reduced hardware and management overhead. This cloud-based solution may work for most systems. However, the legal requirements for Government run applications may require it to be stored in a government facility.

In summary a traditional relational database system is more appropriate for this project. It will provide a much greater ability to scale vertically with additional users signing up for the project. While there are some performance risks, they are outweighed by the data security and data consistency benefits. It is this report's recommendation to use a traditional relational database system.

## References

- D.D.A. (2023) *Why the Guardian switched from mongodb to postgresql*, Medium. Better Programming. Available at: <https://betterprogramming.pub/why-the-guardian-switched-from-mongodb-to-postgresql-861b6cf01e1f> (Accessed: February 5, 2023).
- Document database - nosql* (no date) MongoDB. Available at: <https://www.mongodb.com/document-databases> (Accessed: February 5, 2023).
- IT, B. (2017) *What is encryption at rest, and why is it important for your business?*, Brightline Technologies. Available at: <https://brightlineit.com/encryption-at-rest-important-business/> (Accessed: February 5, 2023).
- Kanaracus, C. (2010) *Netflix turns from Oracle, IBM to Amazon to save cash*, PCWorld. Available at: [https://www.pcworld.com/article/498983/netflix\\_turns\\_from\\_oracle\\_ibm\\_to\\_amazon\\_to\\_save\\_cash.html](https://www.pcworld.com/article/498983/netflix_turns_from_oracle_ibm_to_amazon_to_save_cash.html) (Accessed: February 5, 2023).
- Lampitt, T.B.D.B.A. and Lampitt, A. (2013) *Big movies, big data: Netflix embraces NoSQL in the cloud*, InfoWorld. InfoWorld. Available at: <https://www.infoworld.com/article/2614318/big-movies--big-data--netflix-embraces-nosql-in-the-cloud.html> (Accessed: February 5, 2023).
- Solutions* (1991) Amazon. National Council on Vocational Education. Available at: <https://aws.amazon.com/solutions/case-studies/amazon-database-migration/> (Accessed: February 5, 2023).
- SQL vs. NoSQL databases: What's the difference?* (no date) IBM. Available at: <https://www.ibm.com/cloud/blog/sql-vs-nosql> (Accessed: February 5, 2023).