Table of Contents

Introduction

Aim

The project is to provide a prototype database application for asset integrity inspections. The application needs to demonstrate generating key functionalities used within the asset integrity assessment process.

It is hoped from demonstrating a database with functionalities capable from current excel worksheets that additional funding and approval will be provided for the next phase of the design, generating the flask application component.

Background

Our company handles asset integrity assessments of infrastructure, which entails obtaining past information of a physical buildings, logging defects obtained from a client or from our own inspections. These defects are logged and remediations are generated based on the defects recorded.

From there, the client approves remediations, which moves a project into construction works.

This application is being developed within gated rounds of fundings and needs to demonstrate development that replaces existing excel documents.

In this report, it is needed to demonstrate that the database can generate reports or tables used to check the progress of an asset integrity assessment.

By showing these functions can be generated from the database, the evaluators will give indication to whether this project is to progress to the next phase of the design, which is to generate the web-based aspect of the application.

Further details

Refer to project proposal for additional information surrounding the background, functionality, and assumptions used in the design of the database.

Query Demonstration

Within the project, I needed to demonstrate my capability to generate queries. Below are subsections that demonstrate the application of different types of queries in a practical application.

Each of the sections below are broken down into the following sections:

- Contextual importance
- Demonstration of the query
- Explanation of the query
- How it would be applied to the application

Join Query

Contextual importance

During asset integrity assessment, the dataset of the defects for a structure grows. Some of the defects logged have been generated from inspection drawings. Other defects have not.

These inspection drawings contain geometric information about a defect. Therefore, sometimes workers need to refer to the relevant drawing to gain contextual information about a defect.

This query demonstrated below is to generate list of all defects and their respective inspection drawings.

In doing this, the worker can have this list open on their desktop to locate relevant inspection drawings as needed.

Demonstration of the query

Below are images demonstrating the query and its output:

```
SELECT ID.ProjectId, ID.DrawingType, ID.DrawingNumber, ID.RevisionNumber, S.StructureName, D.StructureID, D.DefectID, D.DefectClassification, D.DefectSeverity
FROM inspection_drawing ID, inspection_drawing_highlights_defects IDHD, defect D, structure S
WHERE
IDHD.ProjectID = 611524 AND
ID.ProjectId = IDHD.ProjectId AND
ID.DrawingType = IDHD.DrawingType AND
ID.DrawingNumber = IDHD.DrawingNumber AND
ID.RevisionNumber = IDHD.RevisionNumber AND
IDHD.StructureID = D.StructureID AND
IDHD.DefectID = D.DefectID AND
IDHD.StructureID = S.StructureID
```

ProjectId	DrawingType	DrawingNumber	RevisionNumber	StructureName	StructureID	DefectID	DefectClassification	DefectSeverity
611524	INS	1	A	Silo 1	7	1	SS	1
611524	INS	2	A	S1 Foundations	8	1	SC	2
611524	INS	3	A	S1 Foundations	8	2	SC	3
611524	INS	4	A	S1 Walls	9	1	SC	3
611524	INS	5	A	S1 Walls	9	2	SC	3

Explanation to the query

The query joins the inspection drawing table to the defect drawing table through the relationship table inspection_drawing_highlights_defects.

To assist the user to filter to the relevant project data, the query provides an input to the projectID. This allows the user to visually see only the inspection drawings and defects relevant to the project they are working on.

The table returned to the user contains the information needed to identify the inspection drawing, the name of the structure, and basic information of the defect.

The defect classification and defect severity are provided to the worker as contextual reference to the type of defect logged in the system. This helps the worker locate the defect within an inspection drawing.

How it would be applied to the application

This query would be provided in the application by receiving the request to provide the data, and the server will serve the tabled results to the front end of the application.

Division Query

Contextual importance

A structure will eventually have several defects and remediations generated over an asset integrity assessment. Some defects will or will not be captured by remediations.

For a worker, it is important to be aware of what defects have not been captured by remediations. As such, workers need to review the list of defects and find which defects have been not captured by remediations.

Previously, this process was manually done through Excel spreadsheets. Below is a query that will automatically generate a list of defects that have not been captured by remediations.

Demonstration of the query

Below are images demonstrating the query and its output:

```
SELECT D.StructureID, D.DefectID
FROM defect D
WHERE D.StructureID = 9 AND NOT EXISTS (
  SELECT DCBR.StructureID, DCBR.DefectID
  FROM defect_covered_by_remediation DCBR
  WHERE D.StructureID = DCBR.StructureID AND D.DefectID = DCBR.DefectID
)
```

StructureID	DefectID
9	1
9	2

Explanation to the query

The query finds the division between defect_covered_by_remediation relationship table and defect table. For MySQL, divisions cannot not be simply specified. Nested queries are needed to undertake the same operation.

For this query, it finds the rows within the defect table that do not exist in the joined table of the two previously mentioned tables.

In addition, when querying, the user is given the ability to filter the defects by structureID.

Doing this, it provides the users with the ability to filter defects that is relevant to certain structure.

The returned table contains structureID and defectID. Both columns are used for the user to quickly capture scope that maybe missing from the current remediations provided.

How it would be applied to the application

This query would be provided in the application by receiving the request to provide the data, and the server will serve the tabled results to the front end of the application.

Aggregation Query

Contextual importance

Mangers may sometimes want to get a snapshot to the number of defects in the system that are above a certain defect severity.

As such, they may ask for the number of defects logged in the system that have a risk rating equal to or above 3. (A rating of 1 being low and 3 being typically high enough to warrant remediation efforts to be undertaken).

Currently, this metric is obtained through filter data in an excel spreadsheet.

Demonstration of the query

Below are images demonstrating the query and its output:

```
SELECT S.StructureName, S.StructureID, COUNT(D.StructureID) AS NumberOfDefectsLogged
FROM structure S, defect D
WHERE S.StructureID = D.StructureID AND D.StructureID = 8 AND D.DefectSeverity >= 3
```

StructureName	StructureID	NumberOfDefectsLogged
S1 Foundations	8	1

Explanation to the query

The query counts the number of defects that meet the condition specified in the "where" section of the query.

In the "where" section, the structure and defect tables are matched. As the defects are a weak entity, there will always be a structureID for the defect table to match to.

Additional parameters are provided to the user to filter the count. For workers, it is important to specify the structure and defect severity to filter by. These filter options are provided within the query.

From providing the "where" parameters and providing count in the select section, the user is able to see a summary statistic to the number of defects logged in the system for a given structure and minimum defect severity.

How it would be applied to the application

This query would be provided in the application by receiving the request to provide the data, and the server will serve the tabled results to the front end of the application.

Aggregation with Group-By Query

Contextual importance

Managers need to assess the total cost of remediations for a structure.

Typically, the manger would filter the remediations and then sum the total remediations for a given structure.

Currently, this is conducted through an excel document.

As such, this query is to provide automatic metrics for the manager of the project.

Demonstration of the query

Below are images demonstrating the query and its output:

```
SELECT S.StructureName, SUM(R.EstimateCost) AS CostToStructure, R.ApprovedStatus
FROM remediation R, defect_covered_by_remediation DCBR, structure S
WHERE S.StructureID = DCBR.StructureID AND DCBR.RemediationID = R.RemediationID
GROUP BY DCBR.StructureID, R.ApprovedStatus
```

StructureName	CostToStructure	ApprovedStatus
Dump Station 1	4000	Not Yet Approved
Dump Station 1	8500	Approved

Explanation to the query

The purpose of the query is to take the total cost of remediations for a given structure. This is done through the use of group-by and combining the remediation, defect_covered_by_remediation and structure tables.

The query first joins the data of all the previously mentioned tables by linking the keys with each other.

From there, it uses the group-by command to aggregate the data by both structureID and ApprovedStatus.

Using this and the Sum command in the “select” section, the query is able to provide a summary table showing structures names, total cost of both approved and not yet approved remediations.

How it would be applied to the application

This query would be provided in the application by receiving the request to provide the data, and the server will serve the tabled results to the front end of the application.

Update Operation

Contextual importance

There are scenarios during work, where workers will change their phone number.

In these circumstances, the worker needs to be able to update there contact details.

Demonstration of the query

Below are images demonstrating the query and its output:

```
1 UPDATE worker
2 SET PhoneNumber = '+61487912003'
3 Where WorkerID = 4;
4
5 SELECT *
6 FROM worker
7 Where WorkerID = 4
```

WorkerID	FirstName	LastName	PhoneNumber	Email
4	Zack	Smith	+61487912003	zack.smith@company.com.au

Explanation to the query

The query has two components: one, the update query; and second, the query to display results.

The update query uses the “where” section to specify the worker of interest to update information.

The “set” section in the update query allows to specify the attribute to update and the value to assign.

The second query displays all worker information from worker table. The “where” section allows the user to specify which workerID to display.

How it would be applied to the application

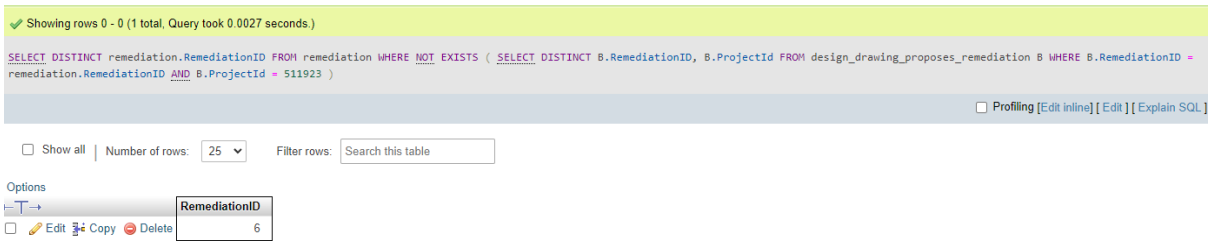
This query would be provided in the application by receiving the request to provide the data, and the server will serve the tabled results to the front end of the application.

Extra Query

From the project proposal, there was another completed query that provided the difference between the design drawing and remediations to find the outstanding scope.

Below is a snippet of the code and output.

```
SELECT DISTINCT remediation.RemediationID
FROM remediation
WHERE NOT EXISTS (
   SELECT DISTINCT B.RemediationID, B.ProjectId
   FROM design_drawing_proposes_remediation B
   WHERE B.RemediationID = remediation.RemediationID AND B.ProjectId = 511923
   )
```



Please refer to project proposal for explanation to how the query works and how it is valuable within the asset integrity assessment process.

Conclusion

To summarise the project, I had broken the conclusion into subsections.

Topics covered were time spent on the project, reflection on work, and advise for future students.

Time spent

Overall, the project took 60 hours. Below is a rough breakdown to the number of hours it took to complete the project:

- Entity Relational Model and Design of Tables 18 hrs.
- Coding Database with sample query 18 hrs
- Proposal writing 6hrs
- Developing queries 12hrs
- Final report writing 6hrs

Reflection on likes and dislikes of the project

I immensely enjoyed the practicality of this project. Before this course, I was designing Express JS and Flask web applications and I was always wondering how I was going to store the user information. From undertaking this assignment, I feel immensely happy knowing how I can solve these problems.

If I had to choose which part of the project, I enjoyed the most, it was the project proposal. I felt a sense of appreciation for the content when coding up the tables after making entity relationship diagrams, checking BCNF and reviewing lecture slides for best practices.

If I had to pick a dislike, it would have to be generating real practical sample data for the application. This process was cumbersome at times.

Even though I disliked generating data, I would not remove the experience as I believe it was a positive experience to getting a feeling to what it would be like working in the workforce.

Reflection on what helped learn best in the project

Overall, the project itself is a fantastic opportunity to practice implementing database design. This project has now given me a starting point to demonstrating my capabilities to work colleagues.

Within the project, learning database design was appreciated in different ways depending on the phase of the project.

At the start of the project, I had to generate entity and relation models. The lecture slides were a solid reference to revise important concepts. The tutorials gave me the starting point to approaching the design process. Together with reference material and an introduction to how to start design, I applied these experiences to the project.

In more detail, I found it helpful to have the lecture slide relevant to relational modelling up when undertaking the design. This assured me that I was approaching the design correctly at the start of the process and reducing the amount potential rework.

During the coding of the database tables and queries, the practicals introduced how to get the application started. To me, this was helpful as I did not know how to start the process.

After setting up, I found the SQL Blackboard learning content most effective to designing the database. While coding the queries, I found referring to the lecture slides and W3shools effective.

Suggestions for future students

I would suggest a beginner to immerse themself practically into SQL design and queries. I recommend the student to download a dataset and practice pulling out specific pieces of information using SQL queries. By doing this, the student will become comfortable in generating queries and will make the student think practically on how to retrieve information.

For the project, I highly recommend students to pick a database problem that they personally find interesting. For me, I really wanted to design a database for a Structural Engineering work scenario. Doing this, I found it easier conceptualise the business and user requirements of the application.