Implementing an Information System for Procurement Card Usage

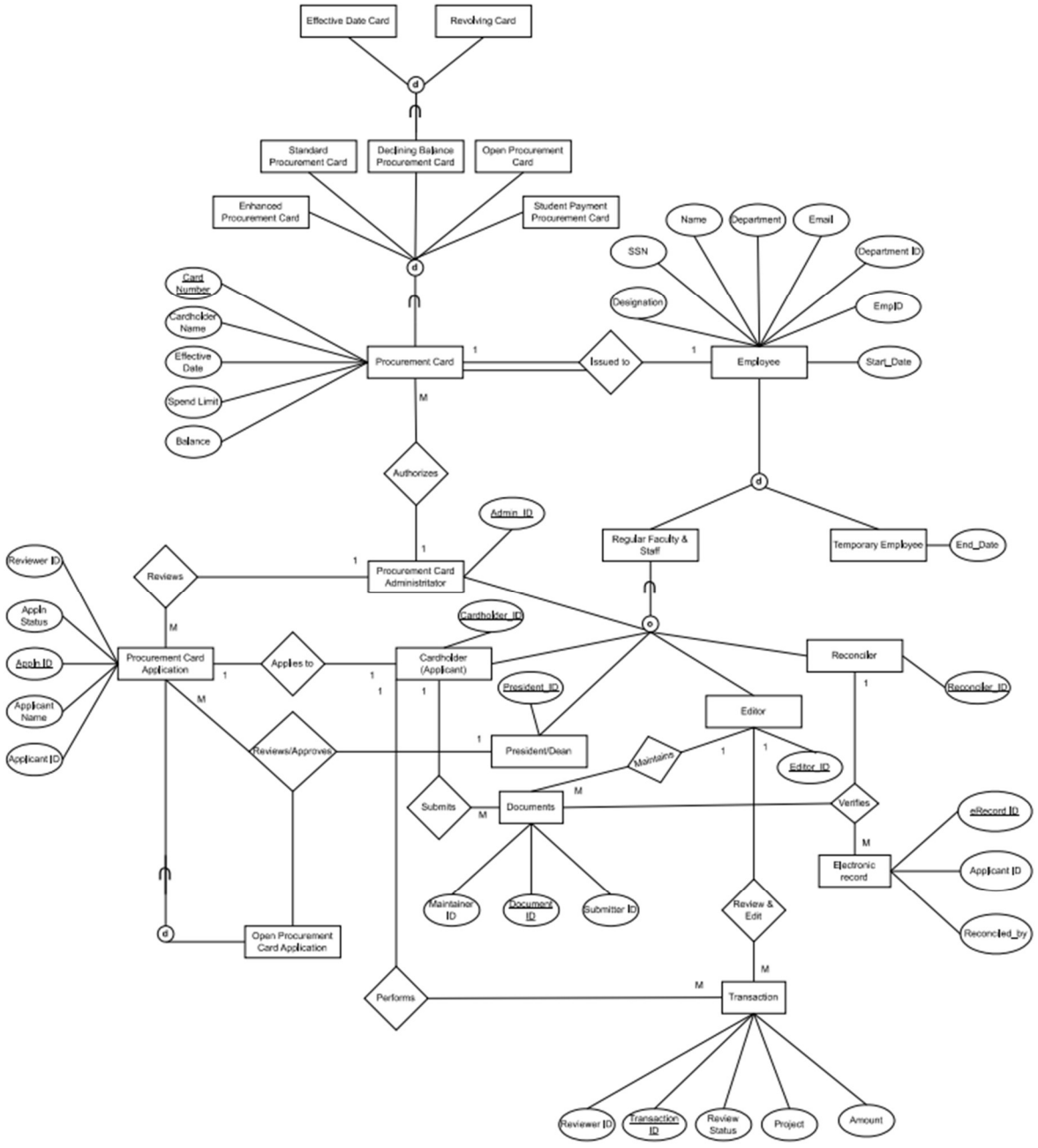
Contents

1. Abstract:

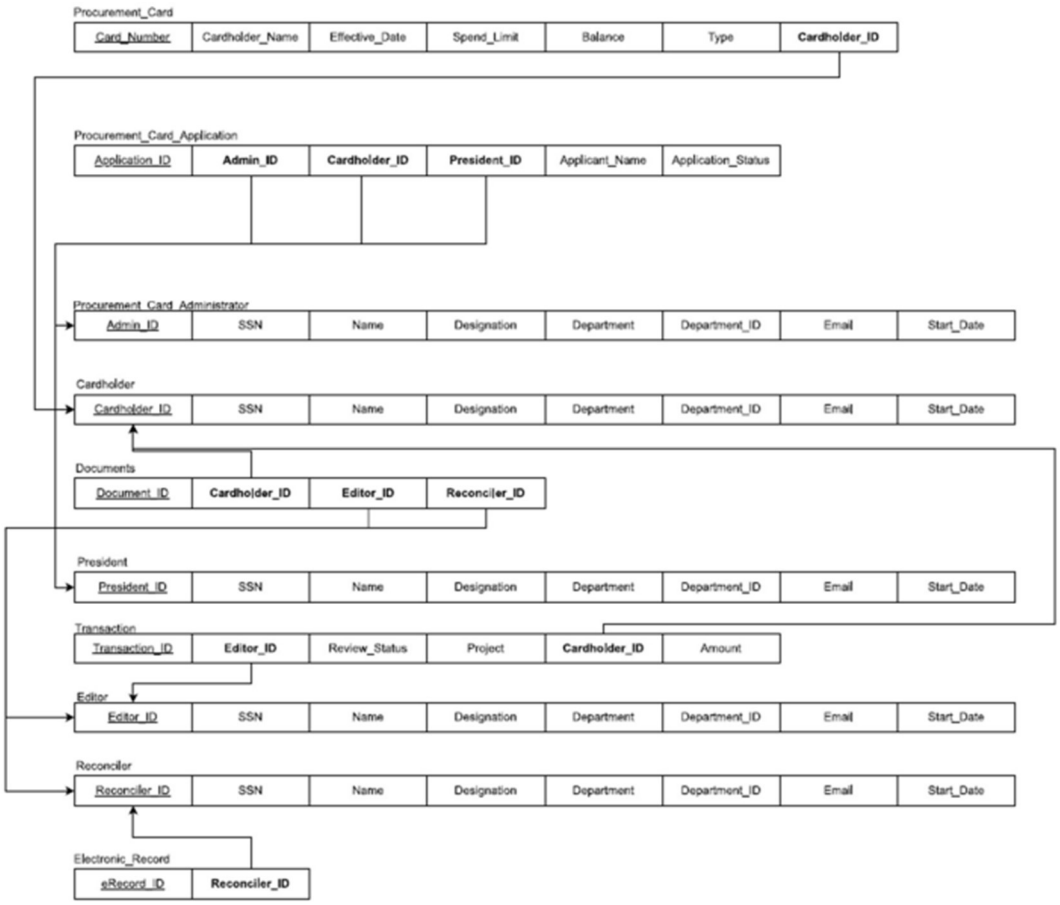
In our report, several approaches and tools used in the design, development, and management of an information system for the employee procurement card usage process at Kentucky university are compared and contrasted. The study demonstrates every step of the process, including creation of entity-relationship diagrams, analyzing SQL queries, and implementing graphical databases.

2. Introduction:

This report explains the process of creating an Entity Relationship Diagram (EERD) to model the procurement card usage for Kentucky university's employees. The article discusses five types of procurement cards: Standard, Enhanced, Declining Balance, Open, and Student Payment. Each card contains the card number, cardholder name, effective date, spend limit, and remaining balance. Employees are issued these cards and can be regular or temporary. The regular employees can have different roles like Procurement Card Administrator, Cardholder, President/Dean, Editor, and Reconciler. The Procurement Card Administrator reviews and authorizes the card, while the President/Dean approves the application. The Cardholder applies for the card and submits documents to the Editor. The Editor reviews and edits transactions, and the Reconciler verifies the document and electronic record.



The report also covered the SQL queries used for data analysis and the database schema and limitations. The schema included entities like "Procurement_Card", " Procurement_Card_Application", " Procurement_Card_Administrator", "Cardholder", "Documents", "President", “Transaction”, “Editor”, “Reconciler”, and "Electronic_Record," each with unique identifiers and related attributes. Business inquiries regarding the acceptance/rejection of procurement cards and the average expenditure ceiling within an organization/department were addressed using SQL queries.



The database was eventually transformed into CSV files and put into a Neo4j database as the final step. The previously described business questions were answered using cypher queries. The group overcame difficulties with node creation and importation while drawing conclusions from the data at hand.

Group members worked on a variety of activities during the project, including comprehending the case study, drawing the EER diagram, writing SQL queries, importing the database into Neo4j, and composing the final report.

3. Methodology:

3.1 Entity-Relationship Diagramming (EERD)

3.1.1 Summary:

The fundamental phase, which involved building an EERD model—a visual representation of data linkages and attributes—was put into practice in the first clause. Our illustration demonstrates how employees use the Kentucky University procurement card as well as the approval/rejection process that this card goes through with various stakeholders.

3.1.2 Explanation:

When we talk about ***Procurement Cards*** there are five primary types of ***procurement cards*** that are available to us namely - ***Standard Procurement Card, Enhanced Procurement Card, Declining Balance Procurement Cards, Open Procurement Card,*** and ***Student Payment Procurement Card***. The Declining Balance Procurement Cards can be further classified as ***Effective Date Card*** and ***Revolving Card***. Each of these procurement cards will contain the Procurement ***Card Number, the Cardholder Name, the Effective Date, the Spend limit,*** and lastly the remaining ***Balance*** in the card.

Each of these ***Procurement Cards*** is *issued to* an **Employee**. Each ***Employee*** has an ***Employee ID, Name, SSN, Post, Start Date, Department,*** corresponding ***Department ID*****,** and lastly an ***Email***. ***Employees*** can be of two types – the ***Regular Faculty and Staff*** and ***Temporary Employees*** who are hired as consultants for a short duration of time. We also have the ***End Dates*** for the ***Temporary Employees***.

The ***Regular Faculty and Staff*** comprises of people who could hold overlapping roles in the University. These roles include ***Procurement Card Administrator, Cardholder (Applicant), President/Dean, Editor,*** and ***Reconciler***.

The ***Procurement Card Administrator*** *reviews* the ***Procurement Card Application***. This ***Procurement Card Application*** consists of the following fields: ***Reviewer ID, Application Status, Application ID, Applicant Name,*** and ***Applicant ID***. The ***Procurement Card Administrator*** also has the power to *authorize* the ***Procurement Card.***

The ***President/Dean*** *reviews and approves* this ***Procurement Card Application*** and if things don't add up then the ***President/Dean*** can ***open the procurement card application*** as well.

The ***Cardholder (Applicant)*** is the one who *applies* for the ***Procurement Card Application.*** The ***Cardholder (Applicant)*** is also required to *submit* proper ***documents*** to the appropriate ***Editor*** in a timely manner. Each ***Document*** has a ***Maintainer ID, Document ID,*** and ***Submitter ID***.

The ***Editor*** maintains the ***Document*** and *reviews and edits* all ***Transactions***. Each ***Transaction*** contains the ***Reviewer ID, Transaction ID, Reviewer Status, Project, Sender ID,*** and ***Amount***.

Lastly, the ***Reconciler*** is the one who verifies the above-mentioned ***Document*** and the ***Electronic Record*** for the same. This ***Electronic Record*** must contain ***the eRecord ID, the Applicant ID,*** and the name of the person it was ***reconciled by***.

3.1.3 Advantages of Entity-Relationship Diagramming (EERD):

It is simpler to convert to database design because to EERD's comprehensive understanding of the system's entities and relationships.

Additionally, it makes it possible to identify important entities, attributes, and relationships, which helps to reduce data duplication and enhance data integrity.

3.1.4 Disadvantages of Entity-Relationship Diagramming (EERD):

EERD does have certain drawbacks, though. For instance, our diagram becomes more complex as it becomes bigger, making it difficult to read and understand.

Additionally, it can be difficult and time-consuming to determine the best connections between entities and attributes.

3.2 SQL Query Analysis*:*

3.2.1 Summary:

Developing a relational database schema and writing SQL queries for data analysis were the next steps.

3.2.2 Explanation:

After finishing the EERD diagram, we went into the second phrase. Now we have raised multiple business questions. The questions were as follows:

a. How many procurement cards were distributed to the employees in the last 6 months per department?
b. Provide details for cardholders whose procurement card applications were rejected
c. Departments which have an average spend limit higher than the average spend limit for the organization

To answer these business question, we used SQL query analysis method. Below are the queries we have written to answer the above questions.

3.2.3 Business Questions of SQL Query:

a. The first query uses the "cardholder" table and selects the "Department", "datename(month, start_date)", and "count(cardholder_ID)" columns. The "where" clause filters out any rows where the department is null. The "group by" clause groups the results by department and the month in which the card was issued. The "order by" clause orders the results by department and the number of cards in descending order.

```
SELECT
Department, datename(month, start_date) Card_Issued_Month, count(cardholder_ID)
Number_Of_Cards
FROM cardholder
where department is not null group by Department, datename(month, start_date)
order by Department, count(cardholder_ID) desc
```

OUTPUT:

	Department	Card_Issued_Month	Number_Of_Cards
1	Architecture	December	3
2	Architecture	February	2
3	Architecture	January	2
4	Architecture	November	1
5	Business	January	3
6	Business	February	2
7	Business	October	2
8	Business	November	1
9	Chemistry	February	3
10	Chemistry	December	2
11	Chemistry	November	2
12	Chemistry	October	1
13	English	November	2
13	English	November	2
14	English	January	2
15	English	December	2
16	English	February	2
17	English	October	1
18	History	October	3
19	History	December	2
20	History	November	2
21	History	January	1
22	History	February	1
23	Physics	December	4
24	Physics	October	2
25	Physics	February	1
26	Physics	November	1

b. The second query uses the "cardholder" and "procurement_card_application" tables, joining them on the "cardholder_ID" column. The "where" clause filters out any rows where the application status is not "Rejected". The "group by" clause groups the results by department and application status.

SELECT
Department, Application_Status, COUNT(application_ID) as NumberOfPeople
FROM cardholder ch
INNER JOIN procurement_card_application pca ON ch.cardholder_ID = pca.cardholder_ID
WHERE application_status = 'Rejected'
GROUP BY Department, Application_Status

OUTPUT:

	Department	Application_Status	NumberOfPeople
1	Architecture	Rejected	2
2	Business	Rejected	2
3	Chemistry	Rejected	4
4	English	Rejected	2
5	History	Rejected	3
6	Physics	Rejected	6

c. The third query uses the "cardholder" and "procurement_card" tables, joining them on the "cardholder_ID" column. The "group by" clause groups the results by department and calculates the average spend limit for each department. The "having" clause filters out departments whose average spend limit is less than or equal to the overall average spend limit.

SELECT department, AVG(spend_limit) as Dep_Avg_Spend_Limit
FROM cardholder c INNER JOIN procurement_card pc
ON c.cardholder_ID = pc.cardholder_ID
GROUP BY department
HAVING AVG(spend_limit) > (SELECT AVG(spend_limit) FROM procurement_card)

OUTPUT:

	department	Dep_Avg_Spend_Limit
1	Architecture	6121
2	History	6993

3.2.4 Advantage of SQL Query:

After conducting our research, we concluded that SQL is an effective tool for manipulating and retrieving data. Large-scale data analysis is made possible by it. It also provides versatility when it comes to combining tables and aggregating data.

3.2.5 Disadvantage of SQL Query:

To construct effective SQL queries, one must have a thorough understanding of the database's structure. In other ways, it is also incredibly boring. In conclusion, using SQL to do computations across numerous tables might be challenging.

3.3 *Graph Database (Neo4j):*

3.3.1 Summary:

The previously generated database was broken down into csv files for each table and then uploaded into the Neo4j database's IMPORT folder. The files were then loaded once more in order to build nodes and respond to business queries with Cypher.

3.3.2 Business Questions of Neo4j:

Business Question: The same previous business questions discussed where answered through neo4j this time.

a. How many procurement cards were distributed to the employees in the last 6 months per department?

```
MATCH (c:Cardholder)
WHERE c.Department IS NOT NULL
WITH c, date(datetime(c.Start_Date)) AS date
RETURN c.Department AS Department,
    date.month AS Card_Issued_Month,
    count(c.Cardholder_ID) AS Number_Of_Cards
ORDER BY Department, Number_Of_Cards DESC
```

This Cypher query, like the SQL query's WHERE clause, first filters out any Cardholder nodes that do not have a department property set. The date () method is then used to transform the start_date property to a Date type, and the month property is used to extract the month of the date. Finally, it groups the results by Department and Card_Issued_Month and sorts them in decreasing order by Department and Number_Of_Cards.

Because the date() method in Cypher demands a string in the format yyyy-mm-dd, you may need to change the format of the start_date field in your Cardholder nodes.

b. Provide details for cardholders whose procurement card applications were rejected

```
MATCH (ch:Cardholder)-
[:HAS_PROCUREMENT_CARD]->(pca:procurement_card_application)
WHERE pca.Application_Status = 'Rejected'
WITH ch.Department AS Department, pca.Application_Status AS Application_Status,
COUNT(pca.Application_ID) AS NumberOfPeople
RETURN Department, Application_Status, NumberOfPeople
ORDER BY Department, NumberOfPeople DESC
```

This Cypher query first searches all Cardholder nodes that are attached to a procurement_card_application node with a HAS_PROCUREMENT_CARD relationship and filters out those procurement_card_application nodes whose Application_Status property is equal to 'Rejected'. The remaining procurement_card_application nodes are then grouped by Department and Application_Status, and the number of Application_ID values in each group is counted using the COUNT() method. Finally, it returns the Department, Application_Status, and NumberOfPeople values for each group and arranges the results in descending order by Department and NumberOfPeople.

c. Departments which have an average spend limit higher than the average spend limit for the organization

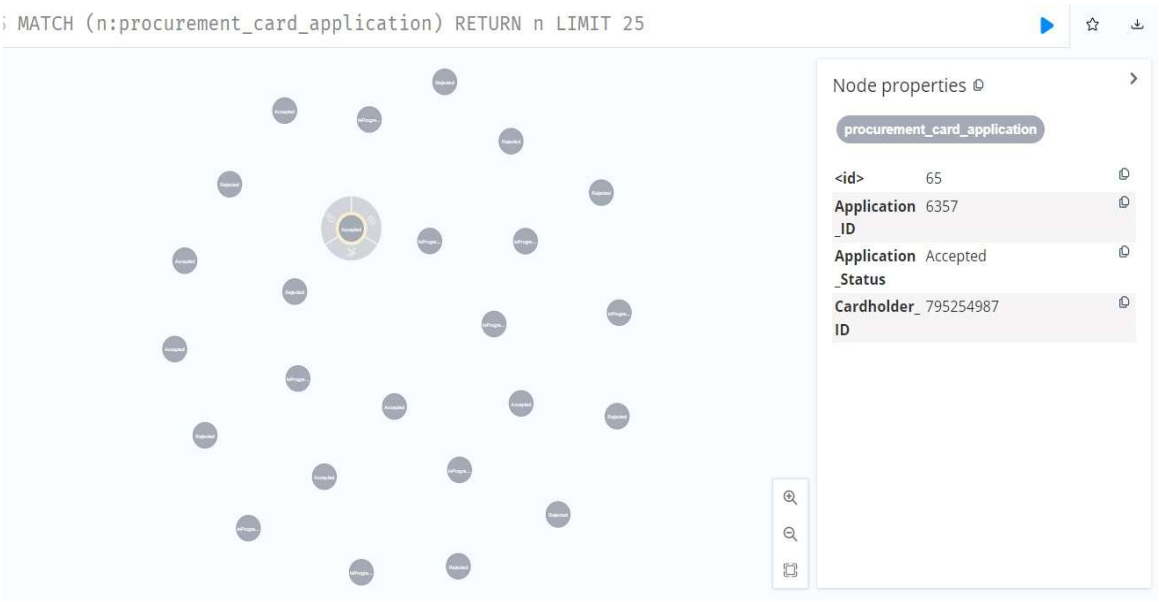
```
MATCH (c:Cardholder)-[:HAS_PROCUREMENT_CARD]->(pc:procurement_card)
WITH c.Department AS department, AVG(pc.Spend_Limit) AS Dep_Avg_Spend_Limit,
AVG(pc.Spend_Limit) > AVG(pc.Spend_Limit) OVER () AS exceed_avg
WHERE exceed_avg = true
RETURN department, Dep_Avg_Spend_Limit
ORDER BY Dep_Avg_Spend_Limit DESC
```

This Cypher query first matches all Cardholder nodes that are connected to a procurement_card node with an HAS_PROCUREMENT_CARD relationship. It then groups the results by Department using the GROUP BY clause and calculates the average Spend_Limit for each group using the AVG() function. It also includes a calculated boolean value exceed_avg which is set to true if the average Spend_Limit of the current department is greater than the overall average Spend_Limit of all procurement cards. Finally, it filters the results based on exceed_avg = true using the WHERE clause, and returns the Department and Dep_Avg_Spend_Limit values for each group, ordered by Dep_Avg_Spend_Limit in descending order.

Note that the OVER() function is used with AVG(Spend_Limit) to calculate the average spend limit for all procurement cards, and is used to compare it with the average spend limit for each department.

OUTPUTS:

	Department	Application_Status	NumberOfPeople
1	"Architecture"	"Rejected"	152
2	"Business"	"Rejected"	152
3	"Chemistry"	"Rejected"	152
4	"English"	"Rejected"	171
5	"History"	"Rejected"	171
6	"Physics"	"Rejected"	152



3.3.3 Advantage of Neo4j:

The example's preceding output makes it very evident that graph databases are exceptional at displaying the relationships between data points, which makes them perfect for studying relational data.

In general, the Cypher query language for Neo4j is expressive and effective, enabling complex querying and data manipulation.

3.3.4 Disadvantage of Neo4j:

The database was transformed into CSV files and put into a Neo4j graph database in order to prepare the work for analysis. Therefore, we assume that adding nodes and importing data may be complicated processes that call for additional configurations.

Second, when dealing with activities that aren't fundamentally graph-based, graph databases could not perform as well as relational databases.

4. Conclusion:

A thorough overview of the tools and processes used in the development and maintenance of the information system was provided to us during the Information Modeling lecture.

After conducting our case study, we discovered that each instrument has advantages and disadvantages and is appropriate for various situations. These aspects should be carefully addressed in the context of the organization's unique needs and capabilities for an organization to make an informed decision on the implementation of an information system.

We personally were more inclined towards relational database as opposed to graph database as it conveyed more information and we felt it was easier for us to work on RDBMS.