Department of Purchasing Division
University of Kentucky
Memorandum

To: **Michelle Zhang, Director of Purchasing Division**
From: **Bo Jia, Chuyi Wang, Jiewen Wei, Wei Jing, Yueran Zeng**
Date: **Dec 14, 2020**
Subject: **Procurement Card Usage**

Introduction

The University of Kentucky issued procurement cards to its employees and students to increase the efficiency of the purchasing and payment process. University Staff and students who want to get the procurement card must complete the application, and these applications will be reviewed and signed by appropriate deans or administrative officers. Besides, the transaction record would be reviewed by editors and reconcilers, and posted on the system.

In order to analyze and design the whole procurement card system, our team provides four methods, EER diagram, SQL, Datalog, and Protege. In the memo, we will discuss each method's advantages and disadvantages, and finally come out with recommendations to help our client, the department of purchasing division at the University of Kentucky, to make decisions.

Conceptual Modeling

We are first tasked with an Enhanced Entity-Relational (EER) diagram of the procurement card system which involves multiple relevant parties of the school, such as the users, administrators, as well as officers in different departments, levels, and all the corresponding interactivities within the system. We identified the entity classes and relationships from the given domain document and it serves as the basis for further techniques including SQL, Datalog, and protégé.

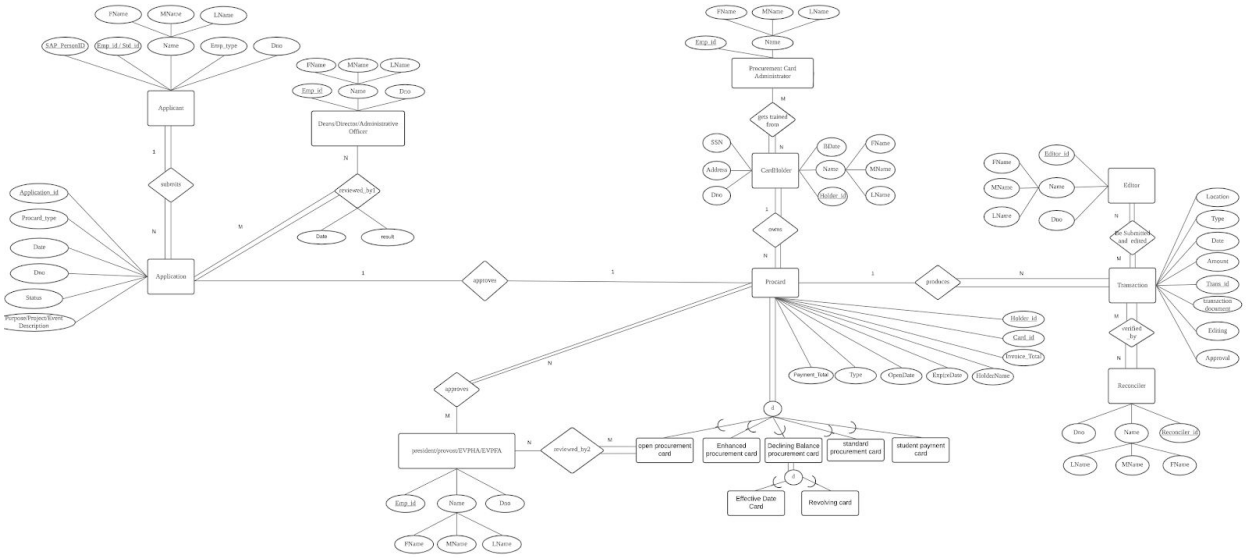

Fig 1. EER diagram

Fig 1 is our EER diagram and it can be viewed as two parts: application (Fig 2) and transaction (Fig 3).

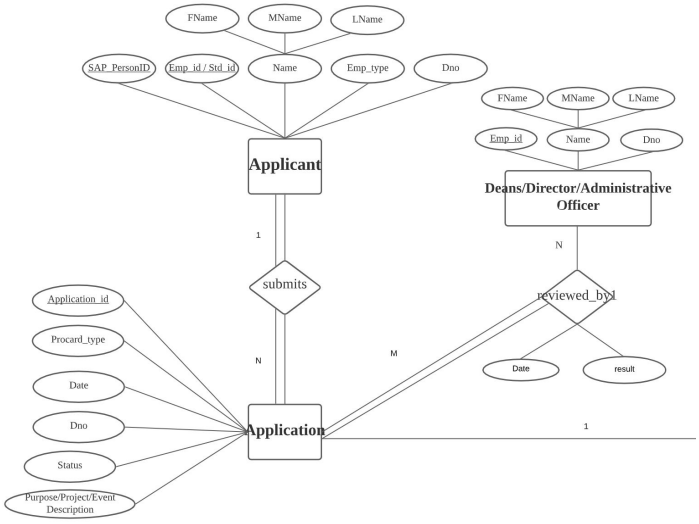

Fig 2. EER diagram application part

When an applicant wants to make a purchase for their department, he/she should submit an application that contains personal information and purpose descriptions. And it will be reviewed by the deans, director, or administrative officers.

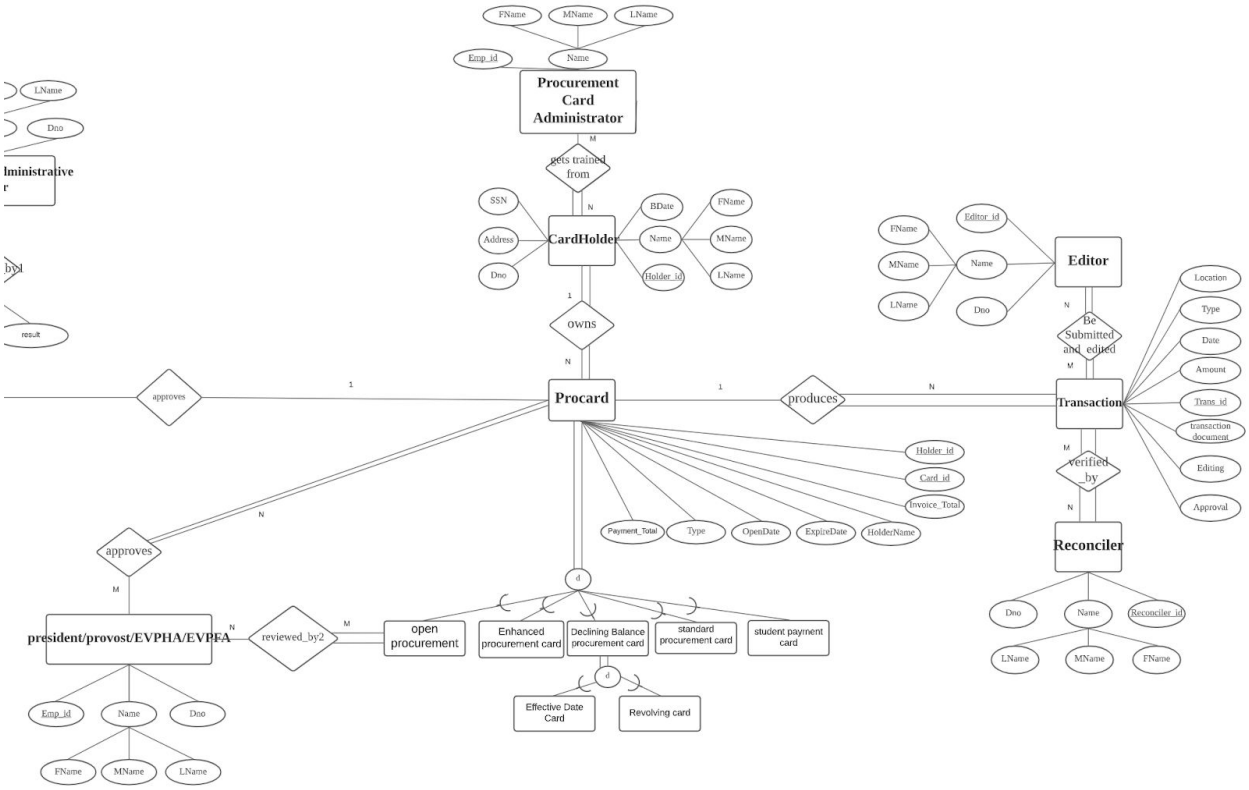

Fig 3. EER diagram transaction part

Once the application gets approved, the applicant will become a cardholder and get trained by the procard administrator. Procard has several subtypes which is a class specialization representation. One of the subtypes (Open Procurement Card) needs to be additionally reviewed and approved by the president, provost, EVPHA, or EVPFA. A cardholder can own one or more procards, and there are several subtypes of the procard. Once a cardholder makes a purchase using the procard, a transaction will be automatically generated and be submitted to the editor to check all aspects of the transaction following the college's terms and policies. A reconciler will also audit and verify the transaction process.

Strength and weakness of EER

The advantage of an EER diagram is that it shows a good visual representation of the whole process. It is a diagrammatic representation of any logical structure of a database. By seeing EER diagrams, we can easily understand the relationships among entities, and thus it could be an

effective communication tool. Also, it allows easy conversion to relational models and other data models such as hierarchical data model and network data model.
However, an EER diagram has limited relationship representation compared with other data models like the relational model. It is hard to show any data manipulation and some content in an EER model is lost or hidden. Besides, there is no industry standard for the notation which may make it hard to understand for others.

Relational Database Design and Queries

In Milestone 2, we are designing a relational model, generating the corresponding database schema with SQL, and implementing SQL queries for question answering. We use MySQL to create the database and populate it with instance-level data.

The following is the whole process of relational database design and queries:
First, modifying the EER Diagram above. For example, what's the logical relationship between “Editor” and “Reconciler”, and make sure the parallel order between them. After reviewing all the entities and relationships, we have improved our EER diagram again. Besides, generating Relational-Database Schema. Because our database is a complex system, multiple fact tables share dimensions during the entire process of using Procurement Card (from application to transactions). Besides, inserting instance-level data. We should make sure of some things in advance. For instance, Whether the column you’re going to add the date in is constrained by some foreign key relationship. If not, we can generate data by ourselves, otherwise, we have to get the data from the corresponding column in another table.
The next step is to design SQL queries: According to our database, we design three SQL queries, which are as follows: How many applicants who are teachers get approved under the purpose of “ buying machines”? Show the id and name of cardholders with total payment over 1000 and sort them in descending order. Show the personal information of cardholders whose balance is more than 2000. The following image is the second query.

```
185    # List cardholder's Holder_id, Firstname, and Lastname, whose total payment is more than 1000, and sort them in descending order by the holder_id
186
187 •  select c.Holder_id, c.FName,c.LName
188    from
189    PROCUREMENT.cardholder as c
190    join
191    PROCUREMENT.Procard as p
192    on
193    p.Holder_id = c.Holder_id
194    where
195    p.payment_total>1000
196    order by p.holder_id desc;
197
198    # List personal information (first name, middle name, last name, cardid ,holder id) , whose Procard's balance is more than 2000
199
```

6:190 | 2 errors found

Result Grid | Filter Rows: Search | Export:

Holder_id	FName	LName
98235	jing	wei
78934	bo	jia
78345	joseph	Tom

Fig 4. SQL query and result

Finally, to export SQL script: we export our SQL script from the MySQL Workbench platform as our relational database coding file.

Strength and Weakness of Relational Database

Strengths:

1. Support complete and straightforward data type: The type of data and the whole database is simple and complete, that's why it is named "structured database". Because of that, it supports other functions and operations as follows.
2. Avoid data inconsistency and redundancy: Due to the relationship constraints of different tables, all the data and information related to one object are stored together in certain tables, which avoids redundancy. Furthermore, whenever someone wants to create or update the data, he has to make sure the related "parent data" have already been updated, which is in order to maintain the structure and relationship, which avoids inconsistency.
3. Support "CRUD" operation of the database: We all know that relational databases are popular in many companies all over the world, the databases such as Oracle, MySQL, SQLite, SQL Server all support "Create, Read, Update, Delete" operation with SQL. So relational databases are very general and have strong extraction and analysis functions.

Weaknesses:

1. The entities and relationships are not well represented. Compared with other types of systems, such as EER, Datalog, Protégé. We cannot intuitively grasp the entities and relationships. Not to mention the subclass, overlapping, disjoint relationships. So it isn't able to represent the structure of the database.
2. Relational database is difficult to store and represent complex data types. To maintain the complete and structured, some other data types besides Varchar, String, Integer are not easily stored, such as the image. Therefore the information being kept is limited.
3. The graphical interface is not clear enough. Most companies and researchers use a relational database like MySQL in the terminal of a computer, which is a platform without a graphical page. Even though we can use some GUI such as "MySQL Workbench", "Navicat", the user interface is still not clear enough.
4. Relational databases have technical requirements for operators. Compared with other database systems, which we can easily add/create data. In relational databases, the operator has to master the basic grammar of SQL, and also know some theory of databases, such as the scrollback, privacy, etc. Accordingly, Therefore, relational databases have certain technical requirements for users and are relatively unfriendly to novices.

Datalog

Like what has been discussed earlier based on ER and EER diagrams, there are basically two core events in our database system, that is the application for procurement cards and transactions of the cards. And the entities in ER diagram are parallel to the name of facts in Datalog. For application-centric, the facts include applicant, dean/director/administrative officer, president/provost/EVPHA/EVPFA. And for transaction-centric, the facts are Procard, procurement card administrator, transaction, editor, reconciler.

Unlike ER diagrams, since it is based on tuples instead of columns, the facts are always followed by a tuple of the specific values in the attributes. As such it is more instance-based (rows like but no column names), and for each tuple, the class name should be pointed out.

In terms of queries in Datalog, it is also unique. It basically could be divided into the following categories: CQ (conjunctive queries); UCQ (union of conjunctive queries); UCQ negs (with negation); FO –First-Order Logic as a Query Language; Fixpoint extensions(Datalog variants).

Compared to the ER diagram, it is more expressive in integrity constraints (ICS). The integrity constraints being able to be expressed in Datalog are basically two types: Functional Dependencies FD (key constraints), which means that one or more columns provide a unique key; and Inclusion Dependencies ID (referential ICs), similar with foreign key. In our database, the applicantion_id in Procard is the primary key in (application) as foreign key.
Technically speaking, Datalog belongs to SQL while the former is based on first-order logic and the latter is based on an extended relational algebra. Datalog outperforms SQL in two aspects. The first is the cleaning semantics, which allows better reason about problem specifications. With more compact formulations, it performs notably better when using recursive predicates, and helps to be better understood and programming maintained.

Besides, as mentioned earlier, it is capable of more expressivities without linear limitation. Most conception models can express things including functional dependencies (key constraints), inclusion dependencies (foreign key constraints), cardinality constraints, hierarchical constraints (class hierarchy), etc. Datalog based on semantics expression could allow us to impose more rules to be expressed, such as general arithmetic constraints, application-specific constraints, and knowledge, purpose, goals, other general requirements, etc.

For example, in a conceptual tree graph of the employees in our data model, we could find all relationships between two random employees represented by the nodes in the graph. And there must be some non-linear relationship and spiral relationship, such as organizational hierarchical relationships between the two employees. And below is a specific use case achieved by our project.

> *% List the qualified transactions and the corresponding cards where for any approved transaction, the amount must be less than the card payment limit, that is the invoice total.*
> transaction_under_invoce(T,C) :-

transaction(T, C,_, _, _, P, _,_, Z),
procard(C,_,_, I,_,_,_,_,_),
P=<I,
Z='Y'.

Txy: y is a transaction of card x.
Ixz: z is the total invoice amount of card x.
Syz: y is smaller than z.
Ay: y is approved.
$\forall x \forall y \forall x((Pxy \wedge Ixz \wedge Ay) \rightarrow Syz)$

Propositional Logic Semantics: For all approved transactions of a card, the amount of every single transaction is no larger than the invoice total of the card.

Protégé

Protégé is a strong supportive community ontology editor. By using Protégé, we create an OWL ontology that models the domain entities and relationships, enforcing domain, range, and disjointness constraints. Class equivalences are added to identify some inferences. We are building the OWL ontology based on the EER diagram, but we have changed some logical relationships to have better performance on Protégé. After we have created a complete and effective database, Protégé can efficiently identify and determine the type of data by identifying the inferences. We could see a complete list of instances of a certain class when we click in the classes. In the image, we selected the approced_transaction. After running the reasoner, the system will automatically list the instances that are in this class, in our case “transaction001”. The image(Fig 5) on the right shows all individuals in the system. When we run any individual with the reasoner, the software would automatically tell us its specific type, shaded in yellow. In this case, it’s showing “approved_transaction”. We can see other properties as well.

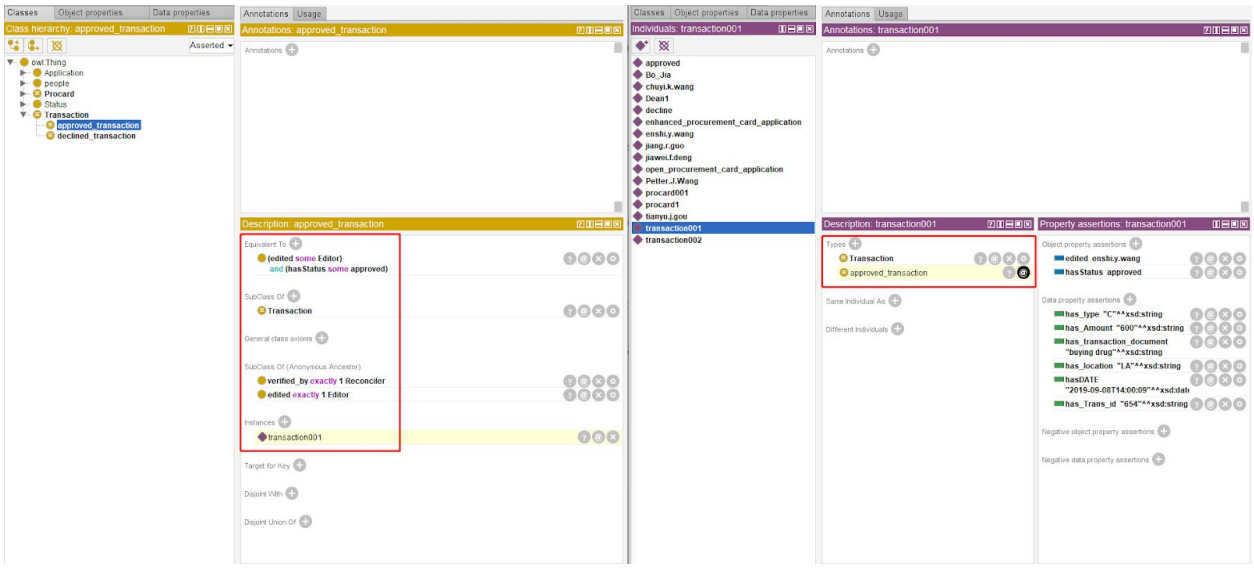

Fig 5. Transaction001

Furthermore, Protégé has a valuable advantage which has an extensible open-source environment. The product does not cost a lot and has a supportive community. Under such circumstances, using Protégé can save a lot of unnecessary capital expenditure, which is very valuable for users like academic institutions. Secondly, the user interface is very clean and straightforward. It does not take a lot of time to navigate through the interfaces and the functions are all rather self-explanatory, which leads to the third advantage. Since the software is easy to use, it does not have a lot of learning cost for new users, which is crucial to our dataset. Because not all staff have good skills in using data, lower learning costs reduce the difficulty of using software, which makes it easier for users to master the use of databases. Aside from its strength, it also has some weaknesses. Even though it would be easy for those to learn how to operate on an existing project, it does require relative high-level skills to create the OWL in the first place and it takes a lot of time. Another weakness is that it is not suitable for massive data. As we can see, if the size of data increases, it becomes very hard to manage. Protégé does not have an efficient way to add or modify large chunks of data because users can usually only work with one individual at a time. Besides, the software is very hard to debug. The debugging message is very vague and does not have clear indicators of what is wrong with the system, and we can see

from this example below (Fig 6).

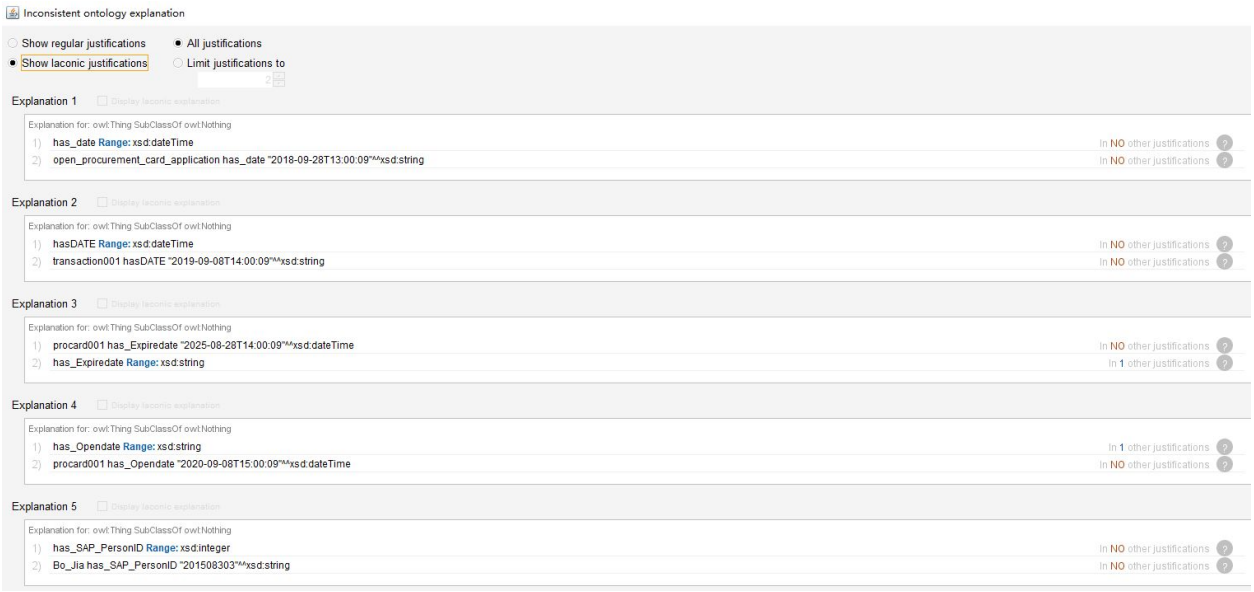

Fig 6.debugging message

Overall, Protégé is suitable for our dataset since Protégé is budget-friendly and our dataset is not very large. Moreover, the low learning cost allows employees who do not have a strong database background to use it easily.

Summary

After analyzing each technique tool and comparing their upsides and downsides, we recommend our client, the department of purchasing division at the University of Kentucky, to choose Protege. Firstly, the interface of Protege is very straightforward, it can clearly show each status and type of each entity, for example, application status, employees' position, and entities' object properties, etc. Secondly, considering the procurement card system is not very complicated and the data volume of a university is usually not large, Protege is an open-source platform, so its future maintenance cost will not be very high. Finally, staff working at the University usually don't have strong data and coding background, compared with SQL and Datalog, they don't have to input hundreds even thousands of queries to search what they want by using Protege, so Protege is very user-friendly. Therefore, among these four technology tools, Protege is better than others, we do recommend our client to choose it.