University of Kentucky
Creation of Relational Schema and SQL Database for Student Recognition Awards and Prizes System

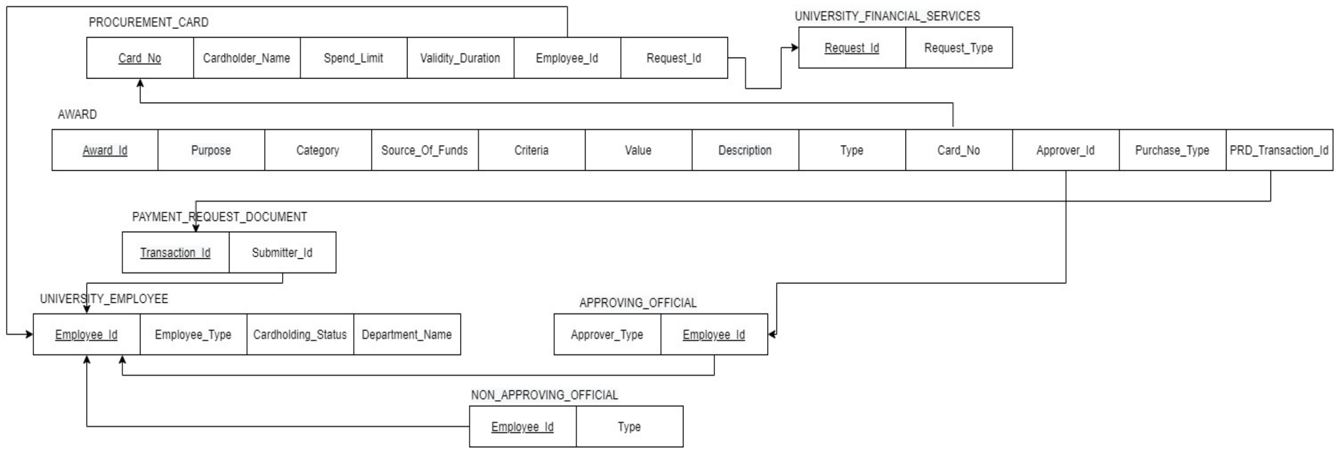
To: Ajay Sharma, Assistant Deputy Director of Financial Services at the University of Kentucky
From: Amol Shyamsunder Govekar, Hrishikesh Potdar, Jun Liu
Date: 17th October 2021
Subject: Revision of EER diagram, creation of relational schema, and SQL database for Student Recognition Awards and Prizes System.

We have revised the EER diagram for the system based on the comments given for the milestone-1 submission. Based on the revised EER diagram we have created a relational schema and the SQL database for it.

Revisions to EER diagram:
We have removed the multilevel specialization for University Employee that included Approving and Non-Approving Official, which further specialized into President, Provost, Dean and Cardholder, Business Officer and Supervisor. So, in the new diagram, we have restricted the specialization level only up to Approving and Non-Approving Official. We have given “Approver_Type” attribute to APPROVING_OFFICIAL and “Type” attribute to NON_APPROVING_OFFICAL to convey the various employee types that were previously represented through multilevel specialization. This also resolves the problem of entities not having attributes. Also, the diagram has consistent formatting across the board. We reduced the complexity of the system by removing unnecessary entities like OSFA and PBS Form.

Relational Schema and its justification:

Diagram of Relational Schema:



In the relational schema, we have included seven tables. The tables are UNIVERSITY_EMPLOYEE, AWARD PROCUREMENT_CARD, APPROVING_OFFICAL, NON_APPROVING_OFFICIAL, PAYMENT_REQUEST_DOCUMENT, UNIVERSITY_FINANCIAL_SERVICES.

There are seven entities in our EER diagram. So we have mapped these to the seven tables in our Relational schema. We did not have an M:N cardinality among entities in our EER diagram thus we do not have any table for relationships. We have multiple entities with one-to-many relationships(1:N cardinalities), to address this, we have put foreign key constraints on entities that are part of "Many" for example - So there is a one-to-many relationship between PROCUREMENT_CARD and AWARD. So there is a foreign key in the AWARD table i.e. Card_No which maps to the Card_No(primary key) in the PROCUREMNT_CARD table.

To address 1:1 cardinalities among entities we have used participation as criteria. The entities that have total participation in the relationships have foreign key constraints on them. For example, the is a one-to-one relationship between PROCUREMENT_CARD and UNIVERSITY_EMPLOYEE but PROCUREMENT_CARD has total participation while UNIVERSITY_EMPLOYEE has partial participation. So there is a foreign key in the PROCUREMENT_CARD table i.e. Employee_Id which maps to the Employee_Id (primary key) in the UNIVERSITY_EMPLOYEE table.

1) For UNIVERSTY_EMPLOYEE the Employee_Id is a primary key and has not null and unique constraints as multiple employees cannot have the same employee id. There are no foreign key constraints on the table.
2) The APPROVING_OFFICAL and NON_APPROVING_OFFICAL are specializations of UNIVERSITY_EMPLOYEE and both have Employee_Id as the primary key as well the foreign key. The foreign key constraint is used to link child tables to the parent table.
3) AWARD table has Award_Id as the primary key and has a unique and not null constraints on it as every award will have a unique id. It has three foreign key constraints. They are Approver_Id, Card_No, and PRD_Transaction_Id. Approver_Id links the table to Approving_Official. Card_No will have a non-null value if the award was purchased using a card and PRD_Transaction_Id will have a non-null value if the award was purchased using a payment request document.
4) PROCUREMENT_CARD has Card_No as the primary key which has not null and unique constraints. It has two foreign keys Employee_Id and Request_Id which connect it to UNIVERSITY_EMPLOYEE table and UNIVERSITY_FINANCIAL_SERVICES table respectively.
5) PAYMENT_REQUEST_DOCUMENT has Transaction_Id as the primary key and it has not null and unique constraints on it. It has Award_Id and Submitter_Id as foreign keys which links it to AWARD and UNIVERSITY_EMPLOYEE.
6) UNIVERSITY_FINANCIAL_SERVICES has Request_Id as the primary key and it has not null and unique constraints on it. It has no foreign key constraints on it.

SQL Queries and their Explanation:

Questions that client may have and the SQL queries for retrieving them

1) Client wants Card Number, Cardholder Name, and approver type of employees where cardholding status is true.

SQL query –

```
SELECT Card_No, Cardholder_Name, t2.Approver_Type
FROM PROCUREMENT_CARD pc
JOIN
(SELECT u.Employee_Id, a.Approver_Type FROM UNIVERSITY_EMPLOYEE u
INNER JOIN APPROVING_OFFICIAL a ON u.Employee_Id = a.Employee_Id
WHERE u.Cardholding_Status = "True") t2
ON pc.Employee_Id = t2.Employee_Id
```

Explanation: In the above query, there are two joins. The first join is in inner query where we are joining APPROVING_OFFICIAL and UNIVERSITY EMPLOYEE based on attribute Employee_Id. We are taking the result of this query as t2 and joining it with table PROCUREMENT_CARD on Employee_Id. Then we are selecting Card_No, Cardholder_Name from PROCUREMENT_CARD, and Approver_Type from t2.

2) Client wants to get the cardholder's name for the card that has the highest expenditure ratio i.e. Award Value/Spend Limit.

SQL query –

```
Select pc.Cardholder_Name, a.Card_No, pc.Employee_Id, (SUM(a.Value)*100/pc.Spend_Limit) as
Percentage_Expenditure
FROM AWARD a
INNER JOIN PROCUREMENT_CARD pc
ON a.Card_No = pc.Card_No AND a.Purchase_type = "Card"
GROUP BY a.Card_No
ORDER BY Percentage_Expenditure DESC
```

```
LIMIT 1
```

Explanation: We are joining AWARD and PROCUREMENT_CARD tables on Card_No in the AWARD table and Card_No in the PROCUREMENT_CARD table and then we are selecting only those awards where the Purchase_Type is “Card”. We are grouping them by Card_No from the AWARD table and ordering them by Percentage_Expenditure in descending order. Then we are using aggregate function SUM to get total award value, dividing it by spend limit we are obtaining percentage expenditure.

3) Client wants award ids of those awards (purchased by card) whose value is greater than average award value in database.

SQL Query:
```
SELECT a.Award_Id, a.Value, a.Approver_Id, ao.Approver_Type
FROM AWARD a
JOIN APPROVING_OFFICIAL ao
ON a.Approver_Id = ao.Employee_Id
WHERE a.Value > (
   SELECT AVG(a.Value)
   FROM AWARD a
          WHERE a.Purchase_Type = "Card"
          )
```

Explanation: We are joining the AWARD table with the APPROVING OFFICIAL table to get the columns - Award_Id, Value, Approver_Id, and Approver_Type. After that, we are using a nested query to filter only those rows where the Award value is greater than the Average award value in the database. We are finding the average of award values for only those rows where Purchase_Type is “card”. Thus, we are returning the required result to the client.

Important Controversies in Design and how we addressed it:

1. A controversy arose on how to update the specializations for the EER diagram on basis of the Milestone 1 comments, regarding whether to completely remove multilevel specialization or add attributes to entities like Provost, Dean, President, etc. We came to a consensus that simplification would be a better choice. Thus, we removed the multilevel specialization and replaced it with attribute “Approver_Type” for entity APPROVING_OFFCIAL and attribute “Type” for NON_APPROVING_OFFCIAL.

2. Another controversy arose on whether to use ON DELETE CASCADE or ON DELETE UPDATE rules on foreign keys. We decided to proceed with ON DELETE CASCADE as after deletion in the parent table there was no purpose for a tuple to exist in the child table (For example – After deletion of Employee_Id from UNIVERSITY EMPLOYEE there is no need for a corresponding card to exist in PROCUREMENT_CARD table as a card cannot exist without a cardholder).

Contributions of each group member:
During our group meetings, all of us collectively analyzed the feedback and then we collaboratively revised the diagram in the draw.io tool, discussed the design of the database, designed, and executed queries, and wrote the memo.

Amol S Govekar: Amol worked on designing the relational schema and updating the tables.
Hrishikesh Potdar: Hrishikesh inserted realistic data in the database and wrote sample queries.
Jun Liu: Jun worked on creating the database.