**CS 631 Project**

**Team:** Sneha Bhut,

Manuel Bravo,

Lorenzo Campanella

**Deliverable 2**

1. **GOALS:**
   1. To map from the EER diagram to a relational schema using the EER to relational algorithm.
   2. To illustrate in detail how we did the translation.
   3. To correctly identify primary keys, foreign keys, other keys, and other constraints beyond referential integrity constraints
   4. To incorporate changes made on data requirements for the payment tracking component and email and phone number verification.
2. **EER TO RELATIONAL MAPPING:**

Step 1: For each regular entity type, we created a relation including all the simple attributes of the entity. We also chose key attributes. This included:

USER\_ACCOUNT {SSN, Name, Balance, PBankID, PBANumber}

BANK\_ACCOUNT {BankID, BANumber, Verified}

REQUEST\_TRANSACTION {RTId, Amount, Date, Memo, ISSN}

SEND\_TRANSACTION {STId, Amount, Date, Memo, Cancelled, ISSN, To-Identifier}

ELECTRONIC\_ADDRESS {Identifier}

Step 2: Our EER diagram has no weak entity types

Step 3: Out EER diagram has no binary 1:1 relationship types

Step 4: We mapped all binary 1:N relationships by choosing the participating entity type at the N-side of the relationship type, and including in it the primary key of the other relation as a foreign key, and the simple attributes of the relationship type. We did this for:

USER\_ACCOUNT  N<HAS\_PRIMARY>1  BANK\_ACCOUNT

(Add FK { PBankID, PBANumber } to relation USER\_ACCOUNT)

USER\_ACCOUNT **mandatory participation**

BANK\_ACCOUNT **partial participation**

USER\_ACCOUNT  1<INITIATES>N REQUEST\_TRANSACTION

(Add FK { ISSN } to relation REQUEST\_TRANSACTION)

REQUEST\_TRANSACTION **mandatory participation**

USER\_ACCOUNT **partial participation**

USER\_ACCOUNT  1<ISSUES>N SEND\_TRANSACTION

(Add FK { ISSN } to relation SEND\_TRANSACTION)

SEND\_TRANSACTION **mandatory participation**

USER\_ACCOUNT **partial participation**

SEND\_TRANSACTION  N<TO>1 ELECTRONIC\_ADDRESS

(Add FK { To-Identifier } to relation SEND\_TRANSACTION)

SEND\_TRANSACTION **mandatory participation**

ELECTRONIC\_ADDRESS **partial participation**

USER\_ACCOUNT  1<HAS>N EMAIL\_ADDRESS

(Add FK { USSN } to relation EMAIL\_ADDRESS)

USER\_ACCOUNT **mandatory participation**

EMAIL\_ADDRESS **partial participation**

USER\_ACCOUNT  1<HAS>N PHONE

(Add FK { USSN } to relation PHONE)

USER\_ACCOUNT **mandatory participation**

PHONE **partial participation**

Step 5: We mapped all binary M:N relationships by creating a new relation and including in it the primary keys of the participating entity types as foreign keys, and also any simple attributes of the relationship type. We did this for:

USER\_ACCOUNT  M<HAS\_ADDITIONAL>N  BANK\_ACCOUNT

(Create relation HAS\_ADDITIONAL with

FK { UBankID, UBANumber } referencing BANK\_ACCOUNT

FK { USSN } referencing USER\_ACCOUNT

)

USER\_ACCOUNT **partial participation**

BANK\_ACCOUNT **partial participation**

REQUEST\_TRANSACTION  M<FROM>N ELECTRONIC\_ADDRESS

(Create relation FROM with

FK { RRTid } referencing REQUEST\_TRANSACTION

FK { EIdentifier } referencing ELECTRONIC\_ADDRESS

)

REQUEST\_TRANSACTION **mandatory participation**

ELECTRONIC\_ADDRESS **partial participation**

Step 6: We had no multivalued attributes

Step 7: We had no n-ary relationship types

Step 8: We mapped all superclass/subclass relationships using Option A, that is, to create a new relation for the superclass entity type including all its simple attributes and choosing a primary key, and to create a new relation for each subclass relationship type, and for each subclass relation, to include all its attributes plus the primary key of its superclass relation as a foreign key and as its primary key. We did this for:

Superclass ELECTRONIC\_ADDRESS

{ Identifier }

Subclass EMAIL\_ADDRESS  ⊂ ELECTRONIC\_ADDRESS

{ Identifier, Verified }

Subclass PHONE ⊂  ELECTRONIC\_ADDRESS

{ Identifier, Verified }

ELECTRONIC\_ADDRESS constraints: **disjoint** and **total participation**

Our relational database schema has no other keys

**CONSTRAINTS**

USER\_ACCOUNT { SSN, Name, Balance, PBankID, PBANumber }

* Each SSN must be unique
* Each value in the PBankID and PBANumber column must have a matching tuple in the BANK ACCOUNT relation
* Cascade on delete

BANK\_ACCOUNT { BankID, BANumber, Verified }

* BankID and BANumber must be unique
* Cascade on delete

REQUEST\_TRANSACTION { RTId, Amount, Date, Memo , ISSN}

* RTId must be unique
* Each value in ISSN column must have a matching tuple in the USER\_ACCOUNT relation

SEND\_TRANSACTION { STId, Amount, Date, Memo, ISSN, Cancelled , ToIdentifier}

* STId must be unique
* Each value in ISSN column must have a matching tuple in the USER\_ACCOUNT relation
* Each value in ToIdentifier must have a matching tuple in the ELECTRONIC\_ADDRESS relation

ELECTRONIC\_ADDRESS { Identifier }

* Identifier must be unique
* Restriction: can be updated but not deleted

HAS\_ADDITIONAL(UBankID, UBANumber, USSN}

* Each value pair columns UBANID and UBANumber has a matching tuple in the BANK\_ACCOUNT relation
* Each value in USSN column has a matching tuple in the USER\_ACCOUNT relation

EMAIL\_ADDRESS(Verified, Identifier, USSN)

* Identifier must be unique
* Each column in Identifier has a matching tuple in the ELECTRONIC\_ADDRESS column
* Each column in USSN has a matching tuple in the USER\_ACCOUNT relation PHONE(Verified, Identifier, USSN)

PHONE(Verified, Identifier, USSN)

* Identifier must be unique
* Each value in Identifier column has a matching tuple in the ELECTRONIC\_ADDRESS column
* Each value in USSN column has a matching tuple in the USER\_ACCOUNT relation

FROM{RRTid, EIdentifier)

* RRTid and Identifier must be unique in every tuple
* Each value in RRTid column must have a matching tuple in the REQUEST\_TRANSACTION tuple
* Each value in EIdentifier column must have a matching tuple in the ELECTRONIC\_ADDRESS relation

1. **DIFFICULTIES** 
   1. Foreign keys can be multiple attributes
   2. Deciding how to represent the electronic address superclass/subclass
2. **DIAGRAM**

****