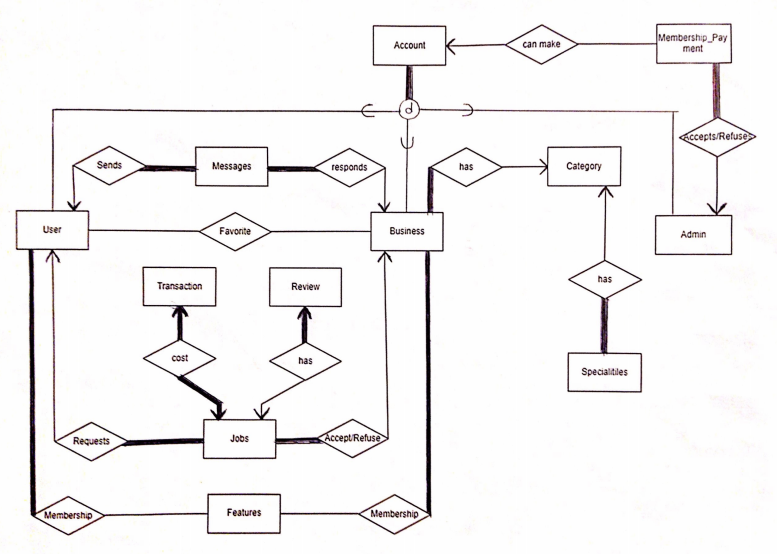
**Requirement Gathering**

* Our team went over the Project Overview and specifications provided to us by our Professor.
* After carefully comprehending the project description, we highlighted the keywords, listed our potential entities and their attributes.
* We went over several similar websites online that offered similar services, to gather and analyze the required information. Some websites are listed as follows.
  + <https://www.angieslist.com/>
  + <https://porch.com/>
  + <https://www.thumbtack.com/>
  + <https://www.improvenet.com/>
  + <https://www.yelp.com/>
* We had a meeting with the client who explained as a primary stakeholder his expectations from the product, his potential clients, business transaction model and other features.
* Following the primary stakeholder’s descriptions about the product from our meeting we were able to get a rough idea about the product and its potential requirements.
* We further studied the project specifications and presentation slides that our client provided and were able to draft a list of major entities and their attributes for our project.
* After carefully analyzing the potential clients for the product as per the main stakeholder’s perspective and descriptions as well as our own considerations; we categorized the stakeholders into three groups: The Users, Businesses and Admins.
* After several meetings discussing with our group we drafted an initial draft of our requirements.
* We decided to further clarify our remaining questions and polish our requirements for the project on the second meeting with our primary stakeholder.
* During the second meeting we were able to further clarify the Stakeholder requirements regarding payment features, user sign up and clear out potential redundancies and errors for our ER diagram as well.

**Business Rules/Constraints**

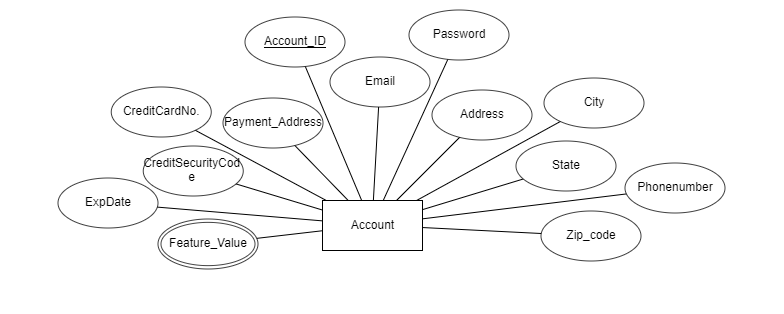
* There is one website named “**WhoDoesThat**”.
* Each Stakeholder can have an **Account**.
* The **Account** will be divided into 3 groups: **User, Business and Admin**
* Each **Account** will be unique with an “Account\_ID” and can belong to only one of the three groups.
* Each **Account** can have payment information stored in it.
* The **User** and **Business** can upgrade their membership to various “**Plan**” (**Basic, Standard and Premium**) through **Admin** by paying the necessary amount using their **Accounts.**
* Each **Account** can make a payment to **Admin** using payment information stored in their **Account**.
* The **Admin** can upgrade/modify the membership of each **User** and **Business** and enable membership features for respective Plan (basic, standard, premium).
* Each **User** can send many messages to a **Business** and a **Business** can also send many messages to a User.
* Each message can only be sent from one **User** to a **Business** and each message can be sent from only one **Business** to a **User**.
* The **User** and **Business** can see a history of their past messages.
* A **User** can request many Jobs from a Business. A **Business** can choose to accept or refuse any **Job** from any **User**.
* Each **Job** will have a certain cost associated with it that the user can pay for it using their payment information stored in the app.
* Each **Job** will also have a **Review** that the **User** can provide when the **Job** is complete.
* The User cannot post a review until the job is complete.
* The **Reviews** made by various **Users** on the jobs that the **Business** performed for them will be available for all **Users** of the app to see. Access to the number of reviews will be based on Plan (basic, standard, premium).
* Each **User** can see a history of their transactions.
* Each **Business** can see a history of their transactions.
* Each **User** can have many favorite **Businesses** that they can see whenever they want.
* Each **Business** can have many favorite **Users**.
* Each **B**usiness must have one of three membership plans to access the Features associated with each **Plan** (basic, standard, premium).
* Each **Transaction** must have a **Job**.
* Each **Review** must have a **Job**.
* Each **Business** must have a **Category** that describes what kind of work the business does.
* Each of the **Specialties** must have a specific **Category** that it belongs to.

**E-R Diagram Overview**

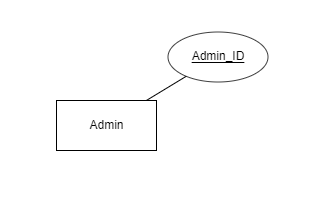


**The ER Diagram for each Entity**

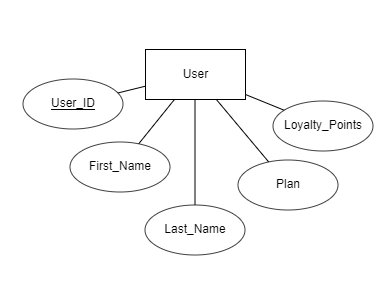
1. Account



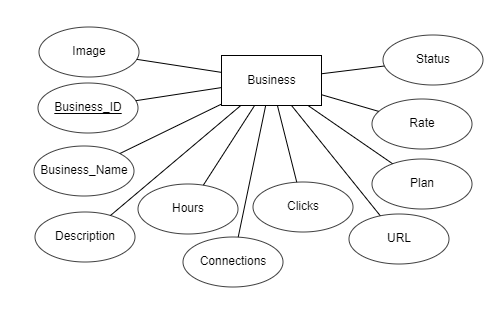
1. Admin



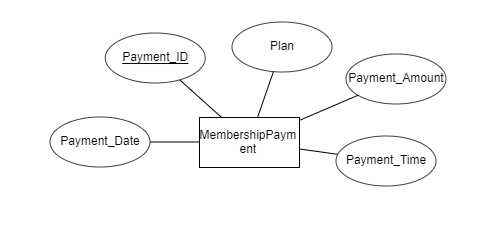
1. User



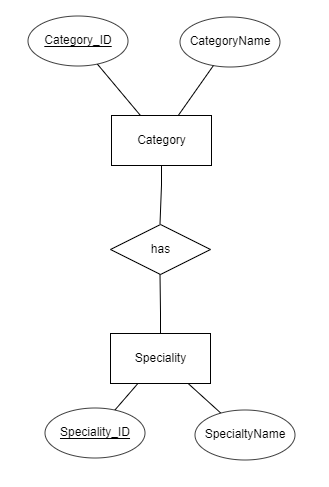
1. Business



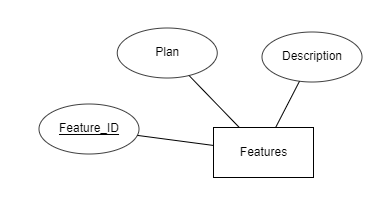
1. Membership\_Payment



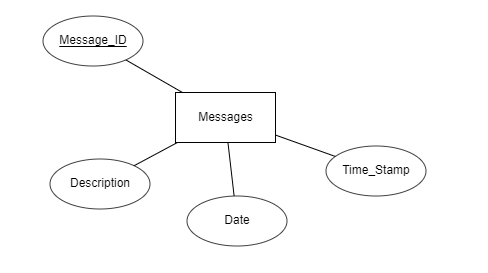
1. Category-Specialty



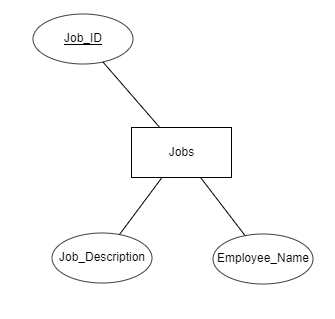
1. Features



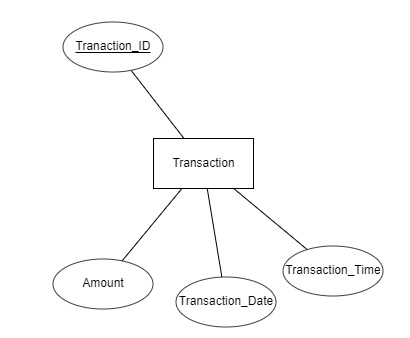
1. Messages



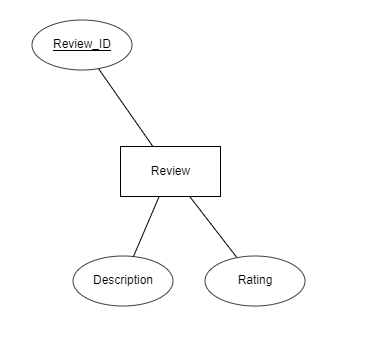
1. Jobs



1. Transactions



1. Reviews



**Relational Schema Diagram**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Account** | Account\_ID | Phonenumber | address | CreditSecurityCode | CreditCardNo. | Payment\_Address | Password | Email | City | ExpDate | State | Zip\_code | Feature\_Value |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **User** | User\_ID | First\_Name | Last\_Name | Plan | Loyalty\_Points |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Buisness** | Buisness\_ID | Image | Business\_Name | Description | Hours | Connections | Clicks | URL | Plan | Rate | Status |

|  |  |
| --- | --- |
| **Admin** | Admin\_ID |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **MembershipPayment** | PaymentID | Plan | Account\_ID | Payment\_Time | Payment\_Amount | Payment\_Date |

|  |  |  |
| --- | --- | --- |
| **Category** | Category\_ID | CategoryName |

|  |  |  |
| --- | --- | --- |
| **Speciality** | Speciality\_ID | SpecialtyName |

|  |  |  |  |
| --- | --- | --- | --- |
| **Features** | Feature\_Id | Plan | Description |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Jobs** | Job\_ID | Job\_Description | Employee\_Name | User\_ID | Business\_ID |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Transaction** | Transaction\_ID | Transaction\_Date | Amount | Transaction\_Time | Job\_ID |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Review** | Review\_ID | Desription | Rating | Job\_ID |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Messages** | Message\_ID | Description | Date | Time\_Stamp | User\_ID | Business\_ID |

**Domain of All Attributes**

**Admin Types:**

int Admin\_ID

**Membership\_Payment Types:**

int Payment\_ID

String Plan

double PaymentAmount

String PaymentDate

String PaymentTime

int Account\_ID //foreign key from Account entity

**Account Types:**

int Account\_ID

String Address

String City

String State

int Zip\_Code

String PhoneNumber

String Email

String Password

int CreditCardNo.

String payment\_Address

String Feature\_Value

int CreditSecurityCode

String ExpDate

**Business Types:**

int Business\_ID

String Business\_Name

String Description

String Hours

String Image

String URL

String Status

int Connections

int Clicks

String Plan //membership plan for business (basic standard premium)

double Rate

**Category Types:**

int Category\_ID

String CategoryName

**Specialty Types:**

int Speciality\_ID

String SpecialityName

**Users Types:**

int User\_ID

String First\_Name

String Last\_Name

int Loyalty\_Points

String Plan //membership plan for user (basic standard premium)

**Jobs Types:**

int Job\_ID

String Job\_Description

String Employee\_Name

int User\_ID //foreign key from user entity

int Business\_ID //foreign key from business entity

**Features Types:**

int Feature\_ID

String Description

String Plan // connects features to plans that user accounts have

**Messages Types:**

int Message\_ID

String Description

String Date

String Time\_Stamp

int User\_ID //foreign key from user entity

int Business\_ID //foreign key from business entity

**Transaction Types:**

int Transaction\_ID

double Amount

String Transaction\_Date

int Transaction\_Time

int Job\_ID //foreign key from Jobs entity

**Review Types:**

int Review\_ID

int Job\_ID //foreign key from Jobs entity

int Rating

String Description

**Favorite Types:**

String UserFirstName

String UserLastName

String BusinessName

int User\_ID // Foreign Key from User Entity Table

int Business\_ID // Foreign Key from Business Entity Table

**Design Decision and Domain of All Attributes**

|  |  |
| --- | --- |
| **Entity: Account [users, business, admin]** | |
| **Primary Key Attribute: Account\_ID** | |
| String Email  [ < 50chars] | Account Email address |
| String Password  [ < 50chars] | Password to access specific Account |
| String Address  [ < 150chars] | Address of the account holder. |
| String City  [ < 150chars] | the city of Account holder is currently situated. |
| String State  [ < 50chars] | The State where the user lives |
| int Zipcode | Account Zip code |
| String Phonenumber  [ < 15chars] | Phone number associated to Account |
| String Feature\_Value  [ < 150chars] | Multivalued Attribute for connecting user to specific features |
| int CreditCardNo. | Credit Card Number associated to Account |
| int CreditSecurityCode | Credit Card security code associated to Account |
| String ExpDate  [ < 15chars] | Expiration Date for credit card associated to Account |
| String PaymentAddress  [ < 150chars] | Payment Address associated to Account Credit Card |
| The Account Entity is the root entity (parent) that holds the core information attributes of all 3 groups (users, business, admin) (subclasses) and provides them with a Unique ID. (Account\_ID).  Account required by User, Admin and Business (one to one).  Account is subclasses are **disjoint** and hence may only belong to one group (user, admin, business). So, no overlapping of sub groups for any Account.  An Account can make many payments to an Admin. Each payment information to Admin can be stored in a table called MembershipPayment. So, an Account can make many payments. [one to many] and many payments can belong to one Account [many to one]. An Account can choose to not make a payment or make a payment anytime Hence, The Account is in **Partial Participation** with MembershipPayments.  An Account must belong to any one of the three **disjoint** subclass groups: User, Business or Admin.  Hence, An Account is in **Total Participation** with User, Business and Admin. | |

|  |  |
| --- | --- |
| **Entity: Admin** | |
| **Primary Key Attribute: Account\_ID** | |
| int Account\_ID | Provide all stakeholders: users, business, admin with a unique ID. |
| Admin can receive payment from many Accounts. Admin can make changes to all accounts.  Admin can receive many payments [many to one].  Admin may or may not have payments made to them. Hence an Admin is in **Partial Participation** with membership payments. | |

|  |  |
| --- | --- |
| **Entity: User** | |
| **Primary Key Attribute: User\_ID** | |
| String First\_Name  [ < 50chars] | The first name of the user |
| String Last\_Name  [ < 50chars] | The last name of the user |
| String Plan  [ < 100chars] | The name of the membership level that user has |
| int Loyaty\_Point | The sum of loyalty points that user gain by his activities |
| **Jobs:**  A User can have many Jobs [many to one]. A Job can belong to only one User [many to one].  A User can choose to request a job, so a User is in **Partial Participation** with Job.  A Job must have User to request it, so a job is in **Total Participation** with User.  **Messages:**  A User may send many Messages. [one to many]  A User can choose to send a message or not, so a User is in **Partial Participation** with  A Message must have a sender (user or business) and a receiver (user or business), Hence  Message is in **Total Participation** with both User and Business.  **Features:**  A User may have many Features in their account. [one to many.]  A User must have Features, Hence the User is in **Total Participation** with Features.  Features may exist without User or Business.  Hence the Features is **Partial Participation** to both Business and User.  **Favorite:**  A user may have many favorite Businesses [many to many]  A business may have many users who made them their Favorite. [many to many] | |

|  |  |
| --- | --- |
| **Entity: Business** | |
| **Primary Key Attribute: Business\_ID** | |
| String Description  [ < 150chars] | The Business name |
| String Date  [ < 50chars] | The description about business services |
| int Connection | The number of connections that business has with users |
| int Click | The number of clicks that business gains |
| String Plan  [ < 50chars] | The name of the membership level that business has |
| double Rate | The business rate based on user rate |
| String Status  [ < 50chars] | Status determines if the business is approved by admin or not |
| String Hours  [ < 50chars] | The business working hours |
| String Image  [ < 150chars] | The business logo picture |
| String URL  [ < 150chars] | The business website URL |
| **Jobs:**  A Business can have many Jobs [many to one].  A Job can belong to only one Business [many to one].  A Business can choose to accept/refuse a job, so a business is in **Partial Participation** with Job.  A Job must have a Business to accept/refuse it, so a job is in **Total Participation** with Business.  **Messages:**  A Business may send many Messages. [one to many]  A Business can choose to send/reply a message or not, so a Business is in **Partial Participation** with Job.  A Message must have a sender (user or business) and a receiver (user or business), Hence  Message is in **Total Participation** with both User and Business.  **Features:**  A Business may have many Features in their account. [one to many.]  A Business must have Features, Hence the Business is in **Total Participation** with Features.  Features may exist without User or Business.  Hence the Features is in **Partial Participation** to both Business and User.  **Favorites:**  A user may have many favorite Businesses [many to many]  A business may have many users who made them their Favorite. [many to many] | |

|  |  |
| --- | --- |
| **Entity: Message** | |
| **Primary Key Attribute: Message\_ID** | |
| String Description  [ < 150chars] | The message content |
| String Date  [ < 50chars] | The date that a message is sent |
| String Time\_Stamp  [ < 150chars] | The time that a message is sent |
| A Message must have a sender (user or business) and a receiver (user or business), Hence  Message is in **Total Participation** with both User and Business.  Many messages can belong to one user (many to one).  Many messages can belong to one business (many to one). | |

**Membership\_Payment:**

|  |  |
| --- | --- |
| Entity: Membership Payment | |
| Primary Key Attribute: Payment\_ID | |
| String Plan  [ < 150chars] | The type of membership and it’s features that an account wishes to access. (Standard, Basic and Premium). |
| double PaymentAmount | Payment amount for a specific membership Plan. |
| String PaymentDate  [ < 50chars] | The Date when the payment for membership plan was made. |
| String PaymentTime  [ < 50chars] | The Time when the payment for membership plan was made. |
| The Membership Payment Entity can store a history of information about all payments made to an admin from various Accounts for different membership plans.  Many Membership Payments can be made to one Admin [many to one]  All Membership Payments must be made to an Admin Hence MembershipPayments is in **Total Participation** with Admin. | |

|  |  |
| --- | --- |
| **Entity: Category** | |
| **Primary Key Attribute: Category\_ID** | |
| String CategoryName  [ < 50chars] | Name of Business Category |
| A Category may have many Specialty. [one to many].  A Category may have many Businesses associated to it. [one to many].  A Category may or may not have a Specialty. Hence a category is in **Partial Participation** with Specialty. | |

|  |  |
| --- | --- |
| **Entity: Specialty** | |
| **Primary Key Attribute: Specialty\_ID** | |
| String SpecialtyName  [ < 50chars] | Name of Specialty in Category |
| A specialty must always belong to a category. Hence, A specialty is in **Total Participation** to Category. Many Specialty may belong to one Category. Hence Specialty is in many to one relationship with Category [many to one]. | |

|  |  |
| --- | --- |
| **Entity: Features** | |
| **Primary Key Attribute: Feature\_ID** | |
| String Plan  [ < 50chars] | The type of Plan Name (Name of Feature) in Feature table. |
| String Description  [ < 150chars] | The type of value the Plan will store |
| The Features may exist on their own without being associated to a Business or a User.    Hence the Features is **Partial Participation** to both Business and User.  Many Features may belong to many users [many to many] | |

|  |  |
| --- | --- |
| **Entity: Jobs** | |
| **Primary Key Attribute: Job\_ID** | |
| String Job\_Description  [ < 150chars] | The details and specifications about the Job Offered by the user and Accepted by the business. |
| String Employee\_Name  [ < 50chars] | The name of the Employee that the business assigned to the job. |
| Every Job will need a Transaction and may or may not have a review.  Each Job will have one Transaction [one to one].  Each Job will have a Review [one to one].  A Job may not necessarily need a review, so a Job is in **Partial Participation** with a review.  A Job must have a Transaction associated to it and hence is in **Total Participation** with Transaction.  A job can belong to only one User [many to one].  A User can have many Jobs [many to one]  A business can have many Jobs [many to one]  A job can belong to only one Business [many to one].  A job must have User to request it, so a job is in **Total Participation** with User.  A job must have a Business to accept/refuse it, so a job is in **Total Participation** with Business. | |

|  |  |
| --- | --- |
| **Entity: Transaction** | |
| **Primary Key Attribute: Transaction\_ID** | |
| double Amount | The Amount of Money paid by the User for a specific job done by the Business |
| String Transaction\_Date  [ < 50chars] | The Date when the Transaction between the user and the business took place |
| int Transaction\_Time  [ < 50chars] | The time when the Transaction between the user and the business took place |
| Each Transaction will belong to one Job [one to one].  A Transaction requires a Job and so is in **Total Participation** with Job. | |

|  |  |
| --- | --- |
| **Entity: Review** | |
| **Primary Key Attribute: Review\_ID** | |
| int Rating | The rating that the user gave to the business for the job done. |
| String Description  [ < 500chars] | The Information and User feedback regarding a specific job done by the Business that the user gave the Job to. |
| The Review can only be made after the job is done and is optional. The user may or may not provide a review.  Each Review will belong to one job. [one to one]  Each Review must have a Job. Hence a Review is in **Total Participation** with Job**.** | |

|  |  |
| --- | --- |
| **Entity: Favorite** | |
| **Primary Key Attribute: (User\_ID, Business\_ID)** | |
| String UserFirstName  [ < 50chars] | First Name of User |
| String UserLastName  [ < 50chars] | Last Name of User |
| String BusinessName  [ < 150chars] | Name of Business |
| Favorite relation stores a list of businesses favorited by users.  Many Users can like many businesses and many businesses can be liked by many users[many to many]. | |

**Description of mapping ER diagram to Relational Schema**

**How did you handle N:M relationships?**

* The “Favorite” relationship between User entity and Business entity is handled by adding a relation called “Favorite”. In this relation, User\_ID, and Business\_ID are included as foreign keys.
* The “Membership” relationship between User entity and Features entity is handled by adding a relation called “Membership”. In this relation, User\_ID, and Feature\_ID are included as foreign keys.
* The “Membership” relationship between Business entity and Features entity is handled by adding a relation called “Membership”. In this relation, Business\_ID, and Feature\_ID are included as foreign keys.
* We had an N:M relationship called messages, but instead opted to change messages into an entity, which created a bridge entity. This turned our N:M relationship into a pair of 1:N relationships.

**How did you handle Ternary Relationships?**

We do not have Ternary Relationships.

**How did you handle Recursive Relationships?**

We do not have Recursive Relationships.

**How did you handle Inheritance?**

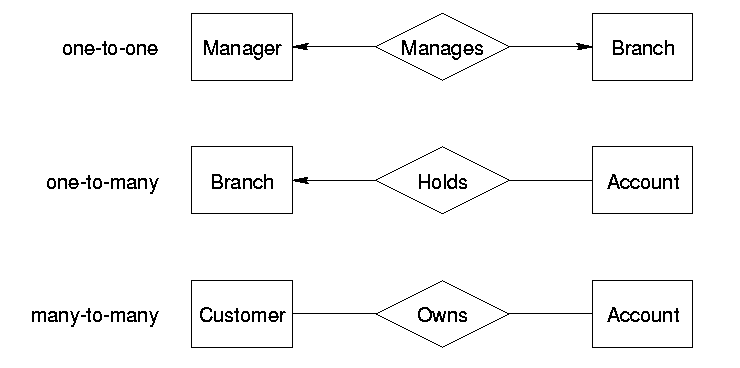
For Inheritance we have “Account” Entity as our parent class. The children are the **User, Admin and Business** entities. The inheritance is disjoint and so an account can only belong to one group.

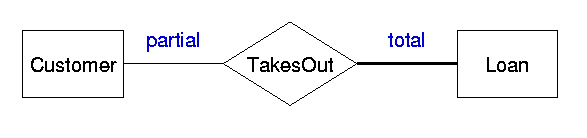
**Problems Faced during ER diagram and its Mapping**

* We initially had structured our ER diagram with features entity divided into 3 separate entities each representing a membership plan (basic, standard and premium). The challenge with this design was the implementation of adding additional attributes (columns) into our tables:
  + **Solution/Decision**: We decided to remove the three tables and replace it with a Features table after further discussions in class we decided to go ahead with this design plan. Now we can add additional features to our membership plans as they are now rows in the Features table.
* Originally the credit card information was stored in a separate table for user and business as the table would only be needed for transaction when the user and business chose to store that data into the system. This severely increased the complexity of our design with 2 additional tables.
  + **Decision:** User credit card information was shifted to Account table for easier payment transaction between Account and Admin. Thus, removing the two unnecessary tables.
* Finding online tools that could help in building an ER diagram was very challenging as many tools offered very different functionalities and features. ERDplus was still best but had many limitations especially since it does not have ternary relations feature nor does it support inheritance.
  + **Solution:** we decided to print out the ERD diagram that we built in ERDplus and draw the (Inheritance) subclasses relations on top of paper then scan and upload the image on word.exe.

**Design Conventions Used for ERD Relations**

*(Pictures below taken from CS5532 Database Theory and Design lecture Slides, Dr. Anne Hee Hiong Ngu, Department of Computer Science, Texas State University)*





**Project Log**

|  |  |  |  |
| --- | --- | --- | --- |
| Feb 8th | 1:00 - 2:00 PM | Nishant Gurung, Kyle Chile, Mahya Saeednejad | Discussed about Project Specifications, Requirements Gathering, Business Constraints Rules, ER Diagram, Log History. **Nishant:** Established google docs drive for sharing project documents. Proposed structure of user entity table with membership constraints, will keep log of all meetings. **Kyle:** Proposed usage of flags for different members rank and made further changes to ER diagram of User table with additional business rules edits. **Mahya:** prepared the first scratch of ER diagram entities, attributes, relations and business constraints, research potential similar market sites and shared final ER diagram from initial draft discussion with team. |
| Feb 12th | 3:00 - 5:00 pm | Nishant Gurung, Kyle Chile, Mahya Saeednejad | **Nishant:** Set up the meeting, Discussed the structure and relation between the entities with team**.** Established additional business rules for newer entities and attributes for ERD. **Kyle:** Negotiated business rules constraints for project and potential questions to ask client for changes in relationship to entities based on client response. **Mahya:** Discussed on relation between each of the additional entities and their relationship with the current entities and shared edits on ER diagram from meeting with the team on what’s app group. |
| Feb 18th | 2:00 - 4:00 pm | Nishant Gurung, Kyle Chile, Mahya Saeednejad | Agenda: Discussion: 1.) Polishing our remaining entities and the relationship 2.) Completing the Requirement Gathering, Business rules constraints 3.) Creating a rough initial draft diagrams (Individual Entities- attributes & Rough Overall ER Diagram) 4.) Converting Diagram to relational Schema 5.) Setting up questions for ER diagram to ask Client at next meeting. -- (**Nishant, Kyle, Mahya**). |
| Feb 19th | 3:00 - 5:00 pm | Nishant Gurung, Kyle Chile, Mahya Saeednejad | **Mahya:** adding relation and constraints to ER diagram, preparing for the presentation. **Kyle:** Proposed new structure for user and business features, helped add cardinality to ER Diagram, added new attributes needed for jobs and messages. **Nishant**: Made the slides for class presentation of ERD to Client. Added new business rules/constraints and modified previous constraints/rules as per stakeholder feedback and team discussions. Wrote the Domain of all attributes Section. Completed the problems faced during development section. |
| Feb 22th | 11:30- 2 pm | Nishant Gurung, Kyle Chile, Mahya Saeednejad | **Mahya:** review final ER diagram and relations, constraints, total and partial relations, each entities' attributes, edit log file, Added User, business and messages tables in design decision and domain. **Kyle:** After discussion completed Relational Schema Diagram, reviewed and finalized decisions for attributes, finalized use of foreign keys within certain entities. **Nishant:** Built the Final ERD (overview) on ERDplus.com software after several design iterations/changes to implement the final one. Made small editing/drawing to the ER diagrams arrows for ease of understanding. Created the individual entities and their attributes for ERD and added them on word.exe. Wrote the design decision and domain of all attributes. (For Account, Admin, MembershipPayment, Transaction,  Review, Jobs, Category, Specialty, Favorites tables). Added Design Conventions used for ERD. Edited log file. |
| Feb 25th | 12:00 - 2 pm | Nishant Gurung, Kyle Chile, Mahya Saeednejad | This meeting most of our work is combined effort to complete and Revise the final ERD Document and make the final touches to the document. Completed/Modified the business constraints/rules as per the final ERD Specs. Finished the Domains with foreign keys added. Checked attribute names with domain names and relational schema. Completed the LOG in Excel and added the Log to Word ERD. Added design relational description to domain. Celebrated Completion. –(**Nishant, Kyle, Mahya**). |