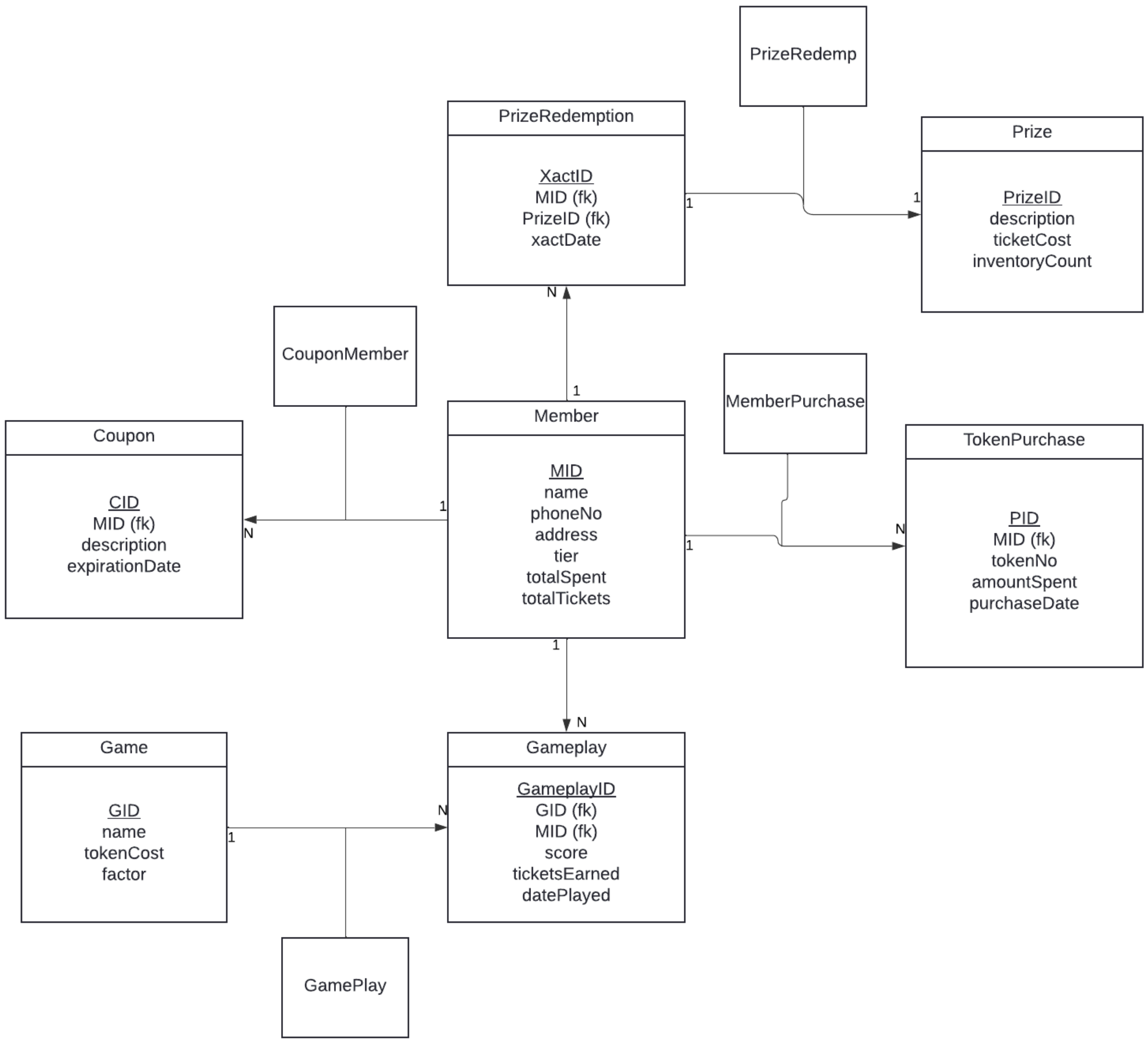
Conceptual Database Design



Based on the application domain, this is the conceptual design we created. There were multiple things that the database should be keeping track of based on the description, such as members, games, prizes, and coupons. We also needed to keep track of transactions made by members, which is why we have relations TokenPurchase and PrizeRedemption to keep track of transactions associated with the purchase of tokens as well as the redemption of tickets for prizes. Finally, we have a table called GamePlay that keeps track of all the instances of games played by members. There are also certain details of the application domain that we included in attributes of certain relations. For example, we needed a way to calculate how many tickets to give a member if they get a certain score. We decided to do this in our Game relation, where we have an attribute called “factor”, which determines how many tickets to give an individual based on their score. Each unique game has their own factor, which allows different games to give different amounts of tickets based on the score. The tickets earned can be calculated by multiplying the factor by the score, which would be stored in the ticketsEarned attribute of GamePlay. We also decided for each relation to have its own primary key to help with normalization later on.

Logical Database Design

Member:

<u>MID</u>	name	phoneNo	address	tier	totalSpent	totalTickets

PrizeRedemption:

<u>XactID</u>	MID (fk to MID in Member)	PrizeID (fk to PrizeID in Prize)	xactDate

Prize:

<u>PrizeID</u>	description	ticketCost	inventoryCount

TokenPurchase:

<u>PID</u>	MID (fk to MID in Member)	tokenNo	amountSpent	purchaseDate

Coupon:

<u>CID</u>	MID (fk to MID in Member)	description	expirationDate

Gameplay:

<u>GameplayID</u>	GID (fk to GID in Game)	MID (fk to MID in Member)	score	ticketsEarned	datePlayed

Game:

<u>GID</u>	name	tokenCost	factor

<u>Normalization Analysis</u>

<u>*Member:*</u>

MID -> name
MID -> phoneNo
MID -> address
MID -> tier
MID -> totalSpent
MID -> totalTickets

<u>*PrizeRedemption:*</u>

XactID -> MID
XactID -> PrizeID
XactID -> xactDate

<u>*Prize:*</u>

PrizeID -> description
PrizeID -> ticketCost
PrizeID -> inventoryCount

<u>*GamePlay:*</u>

GamePlayId ->GID
GamePlayID -> MID
GamePlayID -> score
GamePlayID -> ticketsEarned
GamePlayID -> datePlayed

<u>*Game:*</u>

GID -> name
GID -> tokenCost
GID -> factor

Coupon:

CID -> MID
CID -> description
CID -> expirationDate

TokenPurchase:

PID -> MID
PID -> tokenNo
PID -> amountSpent
PID -> purchaseDate

For all of our tables, we have a primary key which is an attribute that can functionally determine all other attributes in their respective relation. Thus, this makes each primary key of each relation a candidate key of that relation. There is also no other attribute X in any of our tables that can functionally determine another attribute Y if X is not the primary key of the relation.

According to the definition of third normal form, “A relation R is in third normal form if, for every non-trivial functional dependency X->A that holds in R, either (a) X is a superkey of R or (b) A is a prime attribute of R.” A more restrictive normal form, called Boyce-Codd normal form, must have the (a) constraint fulfilled (not either (a) or (b), only (a)). Also, “a superkey is a set of attributes that includes a candidate key”. In our relations, we do not have any trivial FDs, so each FD has to follow either of the constraints in order for the relation to be considered in third normal form. As previously mentioned, since the primary keys in each relation are the only candidate keys, the primary key would be considered to be a superkey. Since this superkey functionally determines all other attributes in the relation for all of our relations, we can say that all of our relations are in third normal form. Moreover, since all of our relations satisfy the (a) constraint of third normal form, we can say that all of our relations are in Boyce-Codd normal form as well.

Query Description

Our self-designed query allows members to see which game they have the highest score in. This is a useful capability to have because it keeps the customers more engaged in the arcade. Knowing which game they have the highest score in may make members play that game more often as they will compete with each other, which keeps the customers playing more games and making more of a profit for the pizzeria and the arcade 🤑.