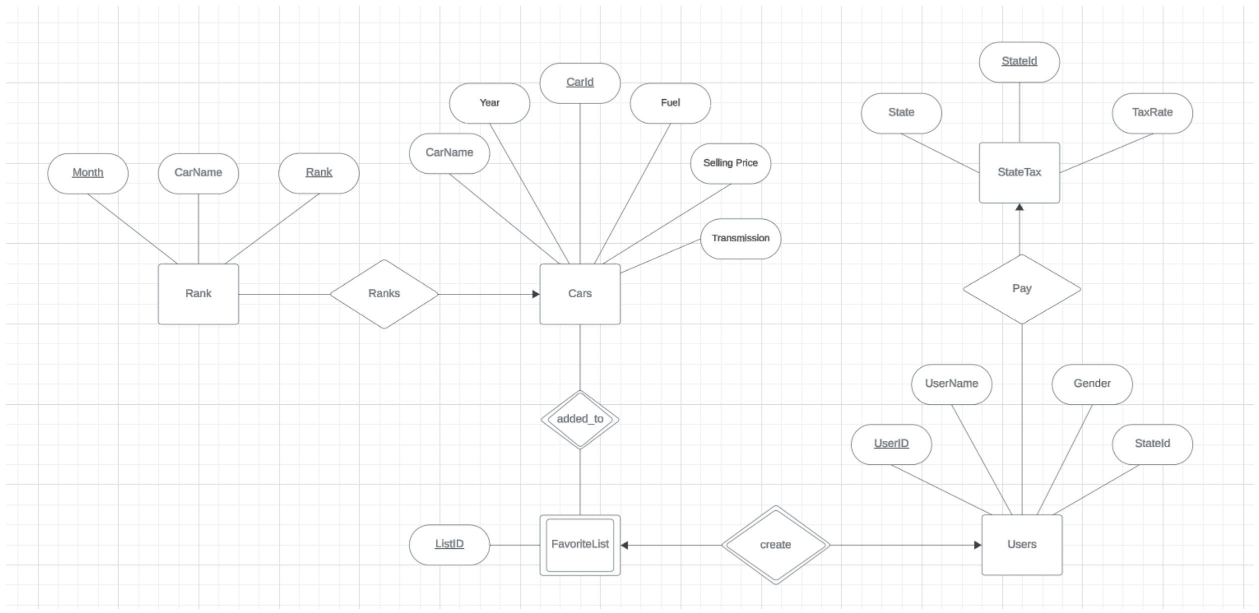
ER Diagram



3NF Explanation

Rank Table:

Candidate Key: CarId, Month

Functional Dependencies: CarId, Month->CarName

This table appears to be in 3NF since there are no transitive dependencies.

Cars Table:

Candidate Key: CarId

Functional Dependencies: CarId->CarName, Year, Fuel, Selling Price, Transmission

This table appears to be in 3NF since there are no transitive dependencies.

FavoriteList Table:

Candidate Key: ListId, CarId, UserId

Functional Dependencies: ListId, CarId, UserId -> No other attributes

This table is in 3NF since the only attributes are part of the composite candidate key

User Table:

Candidate Key: UserId

Functional Dependencies: UserId -> UserName, Gender

This table appears to be in 3NF since there are no transitive dependencies.

StateTax Table:

Candidate Key: StateId, State, TaxRate

Functional Dependencies: StateId->StateId, TaxRate

This table is in 3NF for the same reason.

Added_to Table:

Candidate Key: StateId, UserId

Functional Dependencies: StateId, UserId -> No other attributes

This table is in 3NF since the only attributes are part of the composite candidate key

3NF: A database is in 3NF if it is in 2NF and all the attributes that are not part of a candidate key are non-transitively dependent on every candidate key in the table.

Relational Schema

Rank(

Rank: INT[PK, FK to Ranks.Rank],

Month: INT[PK, FK to Ranks.Month],

CarName: VARCHAR(255),

)

Ranks(

Rank: INT[PK, FK to Rank.Rank],

Month: INT[PK, FK to Rank.Month],

CarId: INT [PK, FK to Cars.CarId],

)

Cars(

CarId: INT [PK, FK to Ranks.CarId, FK to added_to.CarId],

CarName: VARCHAR(255),

Year: INT,

SellingPrice: INT,

Fuel: VARCHAR(100),

Transmission: VARCHAR(100)

)

Added_to)Favorite(

CarId: INT [PK, FK to Cars.CarId],

ListID: INT [PK, FK to ListID.ListID]

Create_Favorite(

ListID: INT [PK, FK to FavoriteList.ListID]

UserId: INT [PK, FK to UsersTable.UserId],

)

Users(

UserId: INT [PK, FK to Create.UserId],

UserName: CarName: VARCHAR(255),

Gender: VARCHAR(100),

StateId: INT [FK to Pay.StateId]

)

Pay(

StateId: INT [FK to StateTaxTable.StateId]

UserId: INT [PK, FK to Users.UserId],

)

StateTax(

StateId: INT [FK to Pay.StateId],

State: VARCHAR(100),

TaxRate: Decimal

)

Entity Assumptions:

CarsTable:

Assumption: Each car has a unique identification number (CarId) that contains attributes such as its name, price, year, fuel, and transmission.

Reason: As a transaction object, a car has multiple attributes that are directly related to the car, so the car is modeled as an entity.

Users Table:

Assumption: Each user has a unique identifier (UserId) and contains attributes such as username, state, and gender.

Reason: The user has an important and central role in the interaction, with identifiable attributes and behaviors, so the user is modeled as an entity.

StateTax Table:

Assumption: Each state has a unique tax rate, identified by StateId.

Reason: Tax rates are important financial information separated by state, each state may have different tax rates, so the tax rates are modeled as a separate entity. When the user tries to calculate the final price including different state tax will get different results.

Rank Table:

Assumption: Each vehicle will have a different sales ranking in different months based on specific criteria.

Reason: Ranking data is time-dependent, so vehicle changes over time need to be tracked. We designed the ranking table as a stand-alone entity so that users can clearly distinguish which vehicle performed best in different months. Since ranking depends on time and vehicle, it contains multiple attributes, such as ranking, month, and CarName, which define a ranked record together. The combined nature of these attributes means that ranking is not suitable as a single field for vehicle or month attributes, so we model it as a separate entity containing multiple attributes.

FavoriteList Table:

Assumption: Each user can create a list of multiple favorite cars and each car can be added to many favorite lists of users.

Reason: User preferences are different and may change over time. Then, this preference should not be directly used as an attribute of the user or car, but represented by an independent association table, allowing the user to have multiple favorite cars, and also allowing a car to be collected by multiple users.

Added_to Table:

Assumption: Each list can have multiple favorite cars and each car can be added to multiple lists.

Reason: Each list can add new cars in the list or delete cars from the list, and each car can be added or removed from lists. Thus, this relationship should be an associative entity due to a many-to-many relationship between cars and favorite lists.

Relational Assumptions and Cardinality:

Cars Table and Rank Table:

Relational Assumption: A car may have different rankings in different months.

Cardinality: one-to-many relationships (one car corresponds to multiple ranking records).

Users Table and FavoriteList Table:

Relational Assumption: A user can create at most one list, and one list can be created by at most one user.

Cardinality: one-to-one relationships (one user corresponds to one favorite list).

Cars Table and FavoriteList Table:

Relational Assumption: One car can be added to multiple favorite lists, and one favorite list can have multiple cars.

Cardinality: many-to-many relationships (many cars corresponds to many favorite lists)

Users Table and State Tax Table:

Relational Assumption: Each user belongs to only one state, and each state can have many users.

Cardinality: one-to-many relationships (one state corresponds to multiple users).