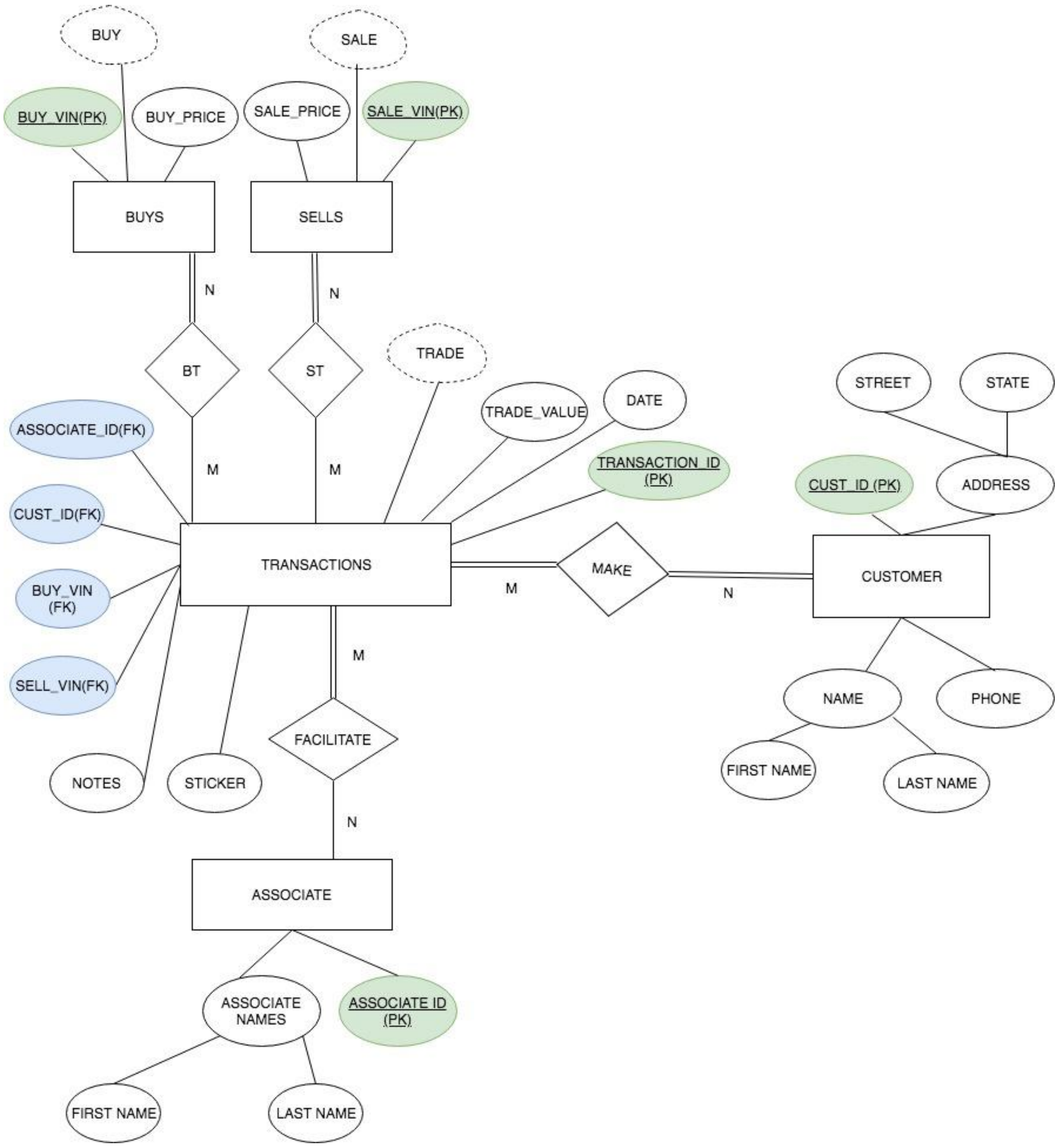
ER Diagram for Pre-owned dealer database (Assignment 3)



In order to draw the Entity-relationship for the Pre-owned dealer database, I first chose Transactions as the primary entity. There are several attributes such as buy/purchase price, date etc. and the primary key is transaction_id. On further investigation of the transactions table, it was seen that there can be additional sub-entities drawn under transactions and these are the “Buys” and “Sells” entity, since each of these have their own attributes to be defined. The buys entity has the BUY_VIN primary key that binds with the transactions entity. It also contains buy_price and another attribute called “Buy” which is a derived attribute. The “buy” attribute can be derived from buy_vin key as follows: If buy_vin is not null then buy attribute will take the value ‘yes’. Same logic holds good for the “Sells” entity. I have chosen to not create another entity for trade because, since there is no key such as trade_vin that is already present in the table, it might not be wise to create a new column by that name and make it an entity. We can still retain the 3NF without adding “trade” as another entity as there are no transitional dependencies. But for the other two entities- “Buys” and “Sells” a separate table was created in order to retain every original information present in the table including buy_vin and sell_vin.

Note: The primary and foreign keys are represented inside the box as PK and FK respectively. Further, a color coding of the box – Primary keys in green and foreign keys in blue is also provided for better distinguishing the keys from the rest of the attributes.

We then introduce another entity called the customer and link the two entities to a relationship.

USING STRUCTURED ENGLISH TO EXPLAIN THE ER DIAGRAM:

1) Entity name: TRANSACTIONS
   The Entity: This database records data about Transactions in a pre-owned dealer business. For each transaction, we record an Associate_id(created as a unique id for each associate), cust_id, buy_vin , sell_vin, notes, sticker, trade_value, date, transaction_id and trade.

   The attributes : For each TRANSACTION, there will be one and only one associate_id, cust_id, buy_vin, sell_vin, notes, sticker, trade_value. The value from these attributes will not be subdivided further.

   The Keys : For each transaction, there is an attribute – transaction_id – that will be unique enough to identify individual entities.

2) Entity name : BUYS
   The Entity : This database records data about cars bought in a pre-owned dealer business. For each car bought, we record buy_price, buy, buy_vin.

   The attributes : For each car bought, there will be one and only one buy_price and its value will not be subdivided.
   For each entity, there may exist attribute “buy” which can be derived from the database.

   The Keys : For each buy, there is an attribute – buy_vin – that will be unique enough to identify individual entities.

3) Entity name : SELLS
The Entity : This database records data about cars bought in a pre-owned dealer business. For each car sold, we record sell_price, sell, sell_vin.

The attributes : For each car sold, there will be one and only one buy_price and its value will not be subdivided.
For each entity, there may exist attribute “sell” which can be derived from the database.

The Keys : For each sell, there is an attribute – sell_vin – that will be unique enough to identify individual entities.

4) Entity name : CUSTOMER
The Entity : This database records data about customers. For each customer, we record name, address, cust_id, phone

The attributes : For each customer, there will be one and only one phone and its value will not be subdivided.
For each customer, there will be one and only one name and address. The value for name will be subdivided into first name and last name. The value for address is subdivided into street and state

The Keys : For each customer, there is an attribute – cust_id – that will be unique enough to identify individual entities.

At this stage no further integration of any attributes is required. We then establish a relationship between the entities as follows:

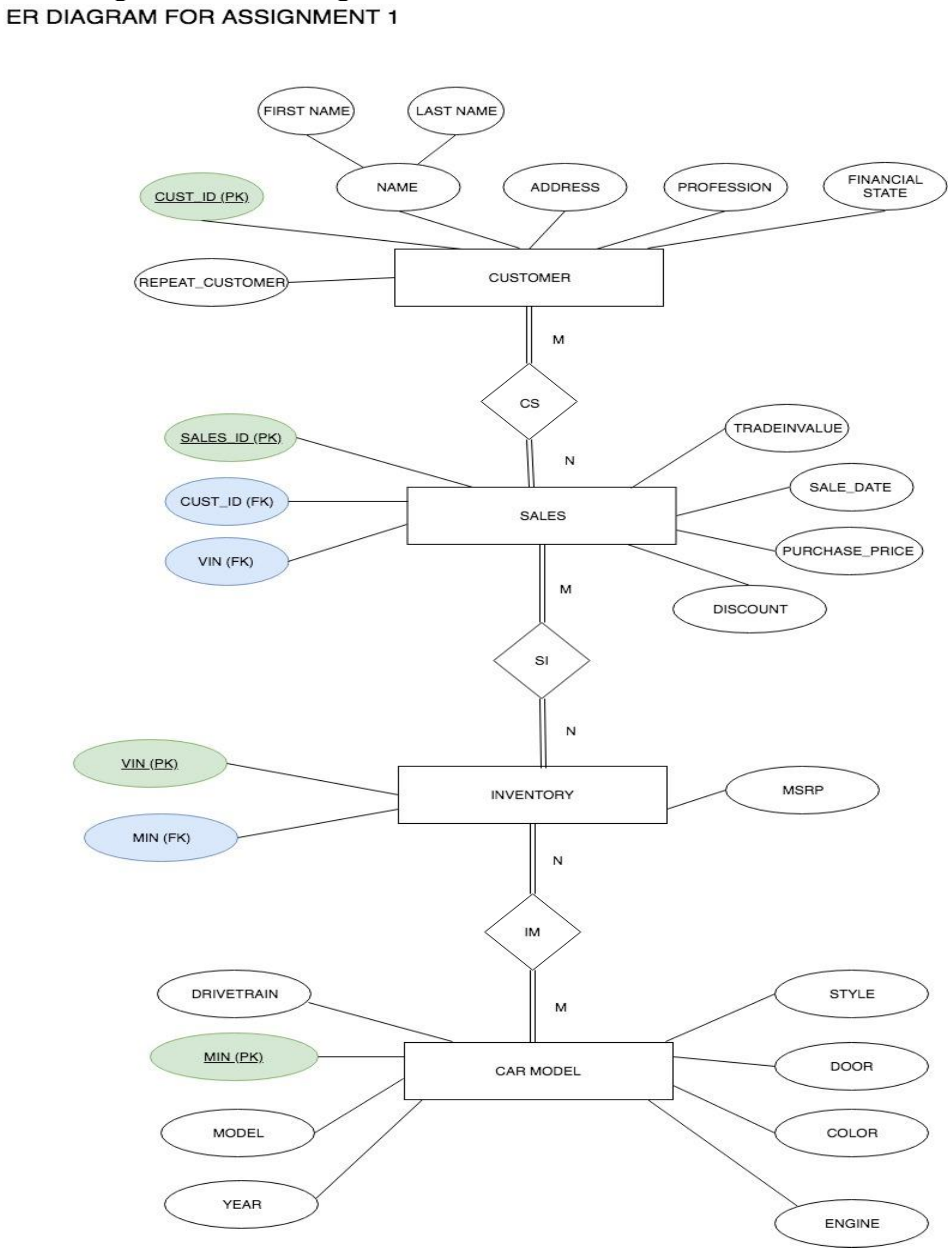
1) TRANSACTIONS entity and SELLS entity
“Sold” information that is recorded in the database must be related to one or more transactions.
Transactions, but not necessarily all transactions (which are recorded in the database) may be related to many (zero or more) cars sold.

2) TRANSACTIONS entity and BUYS entity
“Buys” information that is recorded in the database must be related to one or more transactions.
Transactions, but not necessarily all transactions (which are recorded in the database) may be related to many (zero or more) cars bought.

3) TRANSACTIONS entity and ASSOCIATES entity
Transactions that are recorded in the database must be related to one or more associates.
Associates, but not necessarily all associates in our dataset may be related to many transactions.

4) TRANSACTIONS entity and CUSTOMERS entity
Transactions in our database must record many(one or more) customer's information.
Customers must have one or more transactions to be in our database.

ER Diagram for Assignment 1



The ER diagram contains four entities, with the primary entity being sales table. Although originally the number of tables given were 3, I have normalised the inventory table further by splitting it into inventory and car model. The car-model table only consists of details about a car and these need not be updated every time the inventory table is refreshed, updated or deleted.

We will thus use four entities – customer, sales, inventory and car model.
Note : Since it is always required to use the combination of first and last names in the customer table to perform merging of tables, this has been avoided by the introducing a unique identifier called cust_id to the sales and customer relations table. Here every customer is given a unique cust_id.

USING STRUCTURED ENGLISH TO EXPLAIN THE ER DIAGRAM:

1) Entity name : SALES
   The Entity : This database records data about sales. For each transaction, we record sales_id, cust_id, VIN, tradeinvalue, sale_date, purchase_price, discount.

   The attributes : For each sale, there will be one and only one cust_id, VIN, tradeinvalue, sale_date, purchase_price and discount. The value from these attributes will not be subdivided further.

   The Keys : For each sale, there is an attribute – sales_id – that will be unique enough to identify individual entities.

2) Entity name : CUSTOMER
   The Entity : This database records data about customers. For each customer, we record cust_id, name, address, financial state, repeat_customer and profession.

   The attributes : For each customer, there will be one and only one address, financial state, repeat_customer and profession. The value from these attributes will not be subdivided further.
   For each customer, there will be one and only one name. The value for name will be subdivided into first name and last name.

   The Keys : For each customer, there is an attribute – cust_id – that will be unique enough to identify individual entities.

3) Entity name : INVENTORY
   The Entity : This database records data about inventory. For each inventory, we record VIN, MIN, MSRP.

The attributes : For each inventory, there will be one and only one MIN and MSRP and these attributes will not be subdivided.

The Keys : For each inventory, there is an attribute – VIN – that will be unique enough to identify individual entities.

4) Entity name : CAR_MODEL
The Entity : This database records data about car models. For each model, we record MIN, model, drivetrain, year, style, door, color, engine.

The attributes : For each inventory, there will be one and only one model, drivetrain, year, style, door, color, engine; and these attributes will not be subdivided.

The Keys : For each inventory, there is an attribute – MIN – that will be unique enough to identify individual entities.

At this stage no further integration of any attributes is required. We then establish a relationship between the entities as follows:

1) SALES entity and CUSTOMER entity
Sales/Transactions in our database must record many(one or more) customer's information.
Customers must have one or more transactions to be in our database.

2) SALES entity and INVENTORY entity
Transactions in our database must be associated with many inventories of cars.
Inventory (which is recorded in our database) must be associated with many (one or more) transactions.

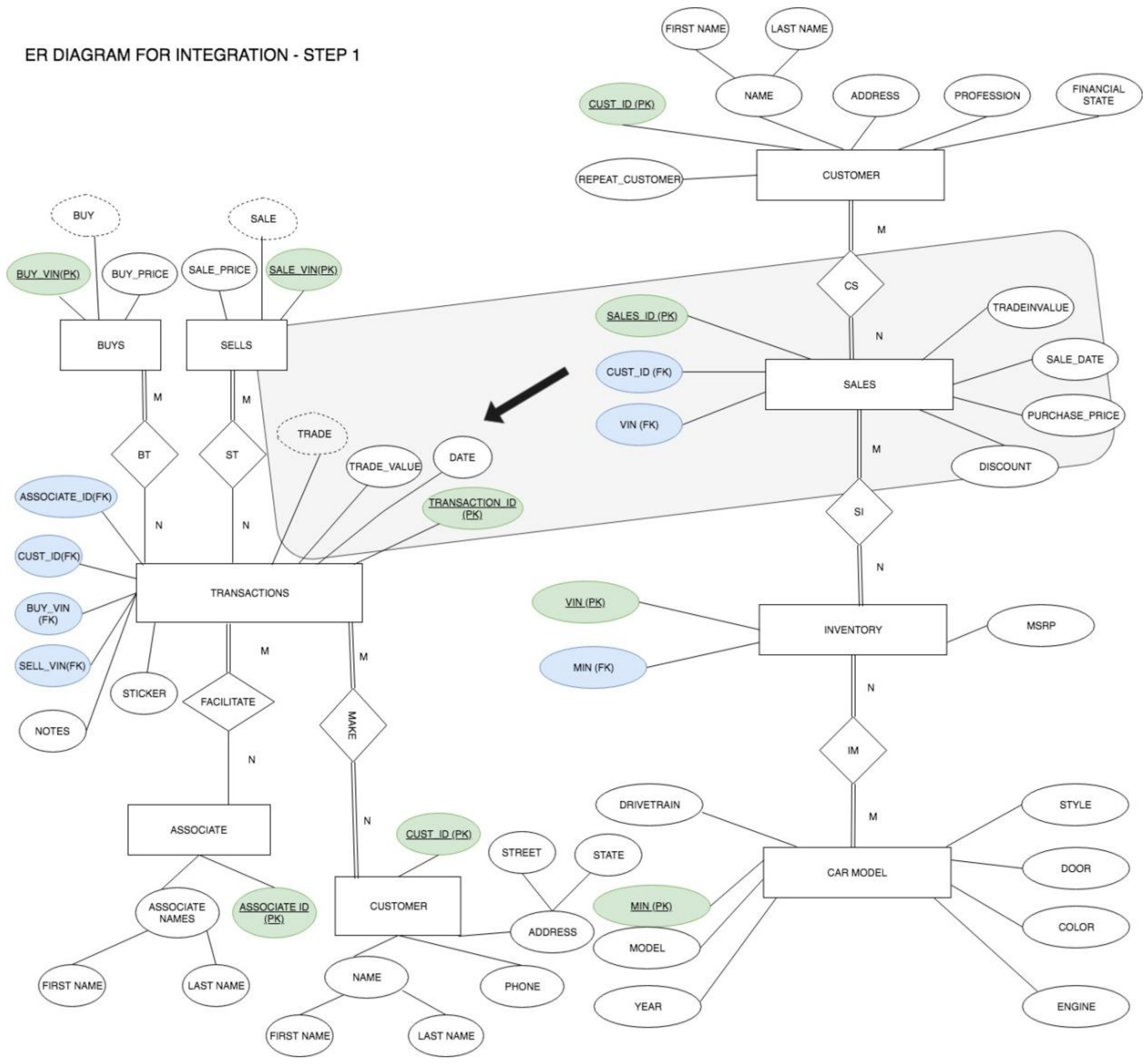
3) INVENTORY entity and CAR MODEL entity
Inventory in our database must contain one or more models of car.
Car models to be sold or traded must be present in atleast one inventory.

Integrating the two databases:

On closely looking at the ER diagrams of the two databases, we can see that there are two places where merging/integration can be achieved.

1) Firstly, the primary entities of both the databases which are transactions and sales. Both the entities mean the same and also contain similar attributes. Attributes from

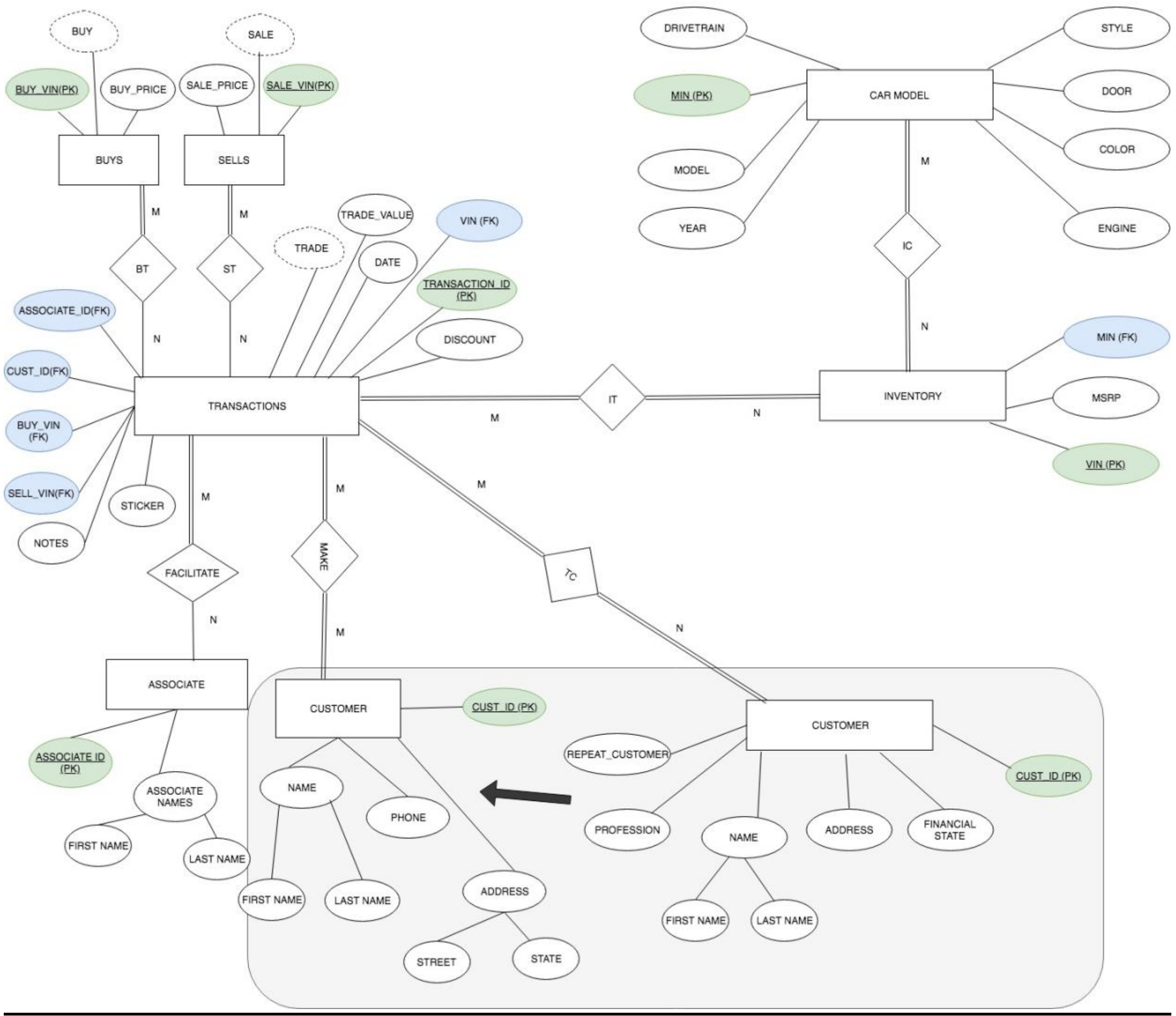
the transactions entities- trade, trade_value, cust_id and date are also found in the sales entity with almost similar names- cust_id, tradeinvalue, sale_date. We can consider transaction_id and sales_id to be the same attributes and do a merge based on them. We will then retain the rest of the attributes in the transactions table and add the additional attributes that are not present in transaction from the sales table in order to retain every single column from both the tables. Hence no information is lost during the process of integration.



<u>Step 2</u>

In the next step we merge customer entities from both the databases. Again, there are similar columns in both the datasets which enables efficient merging. The customer entities from both the databases have common attributes such as name and address. The two can be linked together by using their primary key - cust_id. By doing this not only can we merge the two similar entities as one, but also reduce a lot of redundant space; by preserving all the information from both the tables. Thus we migrate the extra attributes such as financial_state, profession and repeat_customer and retain every attribute.

ER DIAGRAM FOR INTEGRATION - STEP 2

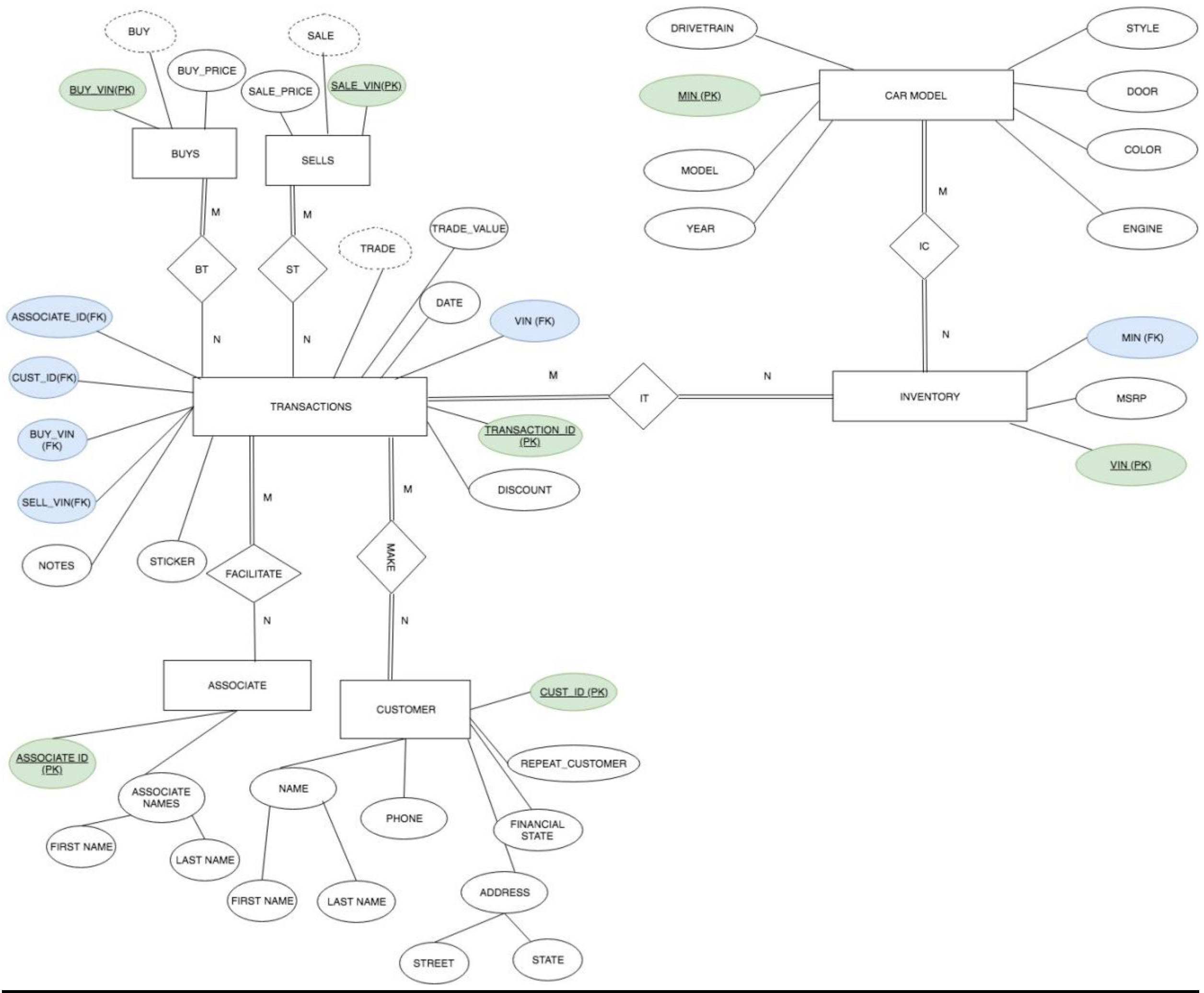


Integrated ER Diagram

Once the integration is achieved, we have 7 entities in total from both the databases. And they are:

1) Transactions
2) Associate
3) Customer
4) Inventory
5) Car model
6) Buys
7) Sell

ER DIAGRAM _ INTEGRATED MODEL



The above DTD design supports the goals of data curation effectively. The model is conceptual, in the sense that it goes above the notion of just abstraction and indirection to interact with data. The model in its original form allows better interpretation of the real world entities. This conceptual model can be further used to develop relational databases.
After building the ER diagram, a simple interpretation of it in English is provided which help even a layman to understand the logic of the model. Further, the role of each entity and attribute is documented and workflow of the representation is also explained. The only disadvantage in this type of model is that it is **catered to a specific database** and thus the cardinalities and relationships cannot be directly reused for another database application. However, if the model has to be implemented for another similar database, the documentation provides a good explanation of how and where the changes have to be made.