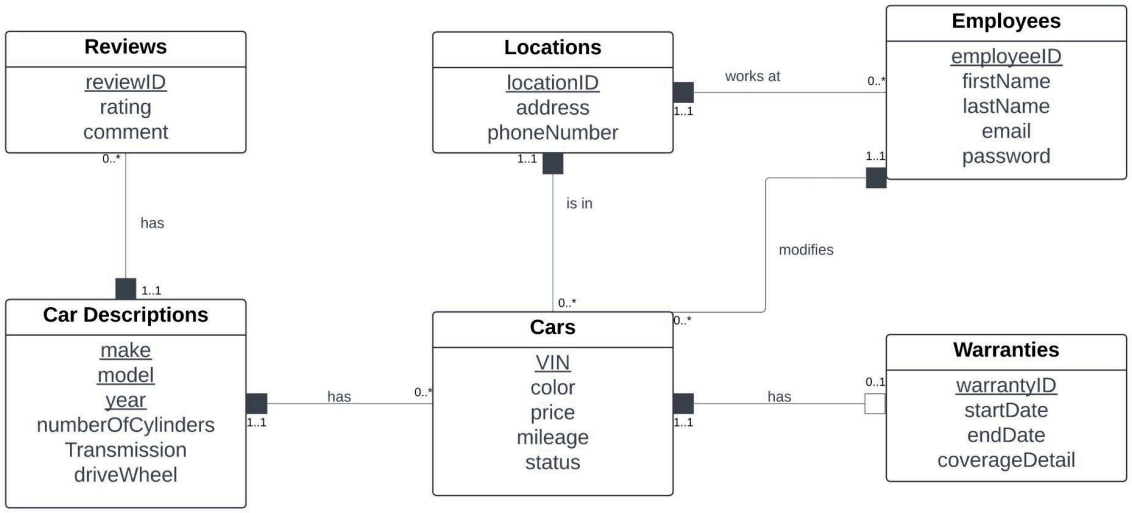
UML



Description of Relationship

Reviews are their own entity instead of an attribute of Cars or Car Descriptions because they represent reviews that can change overtime and are unique. For this entity, every review is connected to one specific Car Description, and multiple reviews can exist for the Car Description. So, it is a 1-to-many for Car Description and Reviews. Making Reviews its own entity means our database system can store multiple reviews for the same car model without having duplicate information. Reviews are important for our system because we want to be able to do data analysis on customer reviews to see which car models are the most desirable.

The Locations entity represents each dealership's locations. For this entity, each VIN (car unique id) from Cars is linked to one locationID, but one locationID can be linked to many VIN's. It is a 1-to-many relationship between Locations and Cars. Location is not an attribute of Car because it would be difficult to keep track of multiple cars at one location if stored in the same table as other car attributes. This entity is important for our system because keeping track of car location is important for dealerships with multiple locations to keep track of their inventory.

The Employees entity represents the employees that the dealership has and is linked to Cars and Locations. For this entity, one employee can modify information of multiple cars in Cars, and an employee can only work at one dealership location. It is a 1-to-many relationship between Employees and Cars, and 1-to-many between Employees and Locations. Having this entity is important so that dealerships can keep track of which employees interact with which cars.

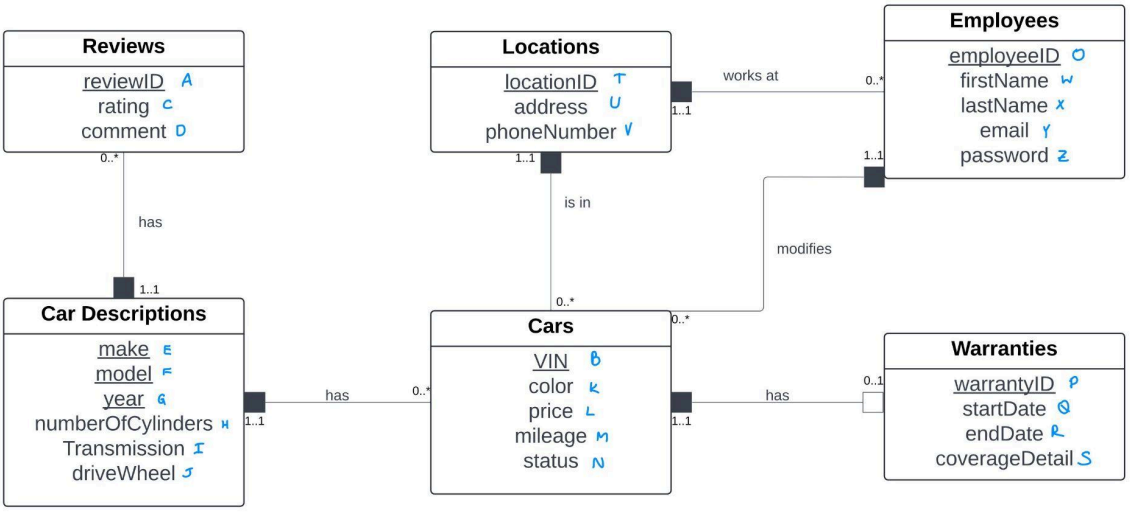
The Car Descriptions entity represents car model information like make, model, and year, instead of having them as attributes of the Cars entity. This is because multiple cars can have the same description, so having the description as a separate entity would help reduce redundancy since many cars can be linked to just 1 description. There is a 1-to-many

relationship between Car Descriptions and Cars. This entity is important for our system because dealerships might have many of the same car in stock, and having to repeat the description for each car within the Tables car would not be efficient.

The Warranties entity represents an individual car's warranty information, which is different for each car because not all of them will have one or will have varying coverage and start/end dates. The relationship between Cars and Warranties is a 1-to-1 relationship because every warranty needs to be linked to a car. This entity is important because the system can help the dealership track what cars have warranties as well as update any details without directly changing the Cars table.

In general, the design of our entities is meant to represent the business needs of a dealership inventory management system. The system will help make data management more efficient and scalable.

Normalization to 3NF



According to the UML (with the assigned letters A-Z), we know the follows are all functional dependencies:

A → CD
EFG → HIJ
B → KLMN
P → QRS
T → UV
O → WXYZ

A → EFG
B → EFG
B → T
B → O
B → P
P → B
O → T

We have to decompose it into 3NF

1) Find the candidate key(s)

Left	Middle	Right	None
A	B	C	
	E	D	
	F	H	

	G	I	
	O	J	
	P	K	
	T	L	
		M	
		N	
		Q	
		R	
		S	
		U	
		V	
		W	
		X	
		Y	
		Z	

We have $\{AP\}^+ \rightarrow$ ABCDEFGHIJKLMNOPQRSTUVWXYZ
Therefore, the candidate keys are AB, AP.

2) <u>Find the minimal basis</u>
   2.1) Break the functional dependencies so that only singleton exist in the RHS

   A → C
   A → D
   EFG → H
   EFG → I
   EFG → J
   B → K
   B → L
   B → M
   B → N
   P → Q
   P → R
   P → S

$T \rightarrow U$
$T \rightarrow V$
$O \rightarrow W$
$O \rightarrow X$
$O \rightarrow Y$
$O \rightarrow Z$
$A \rightarrow E$
$A \rightarrow F$
$A \rightarrow G$
$B \rightarrow E$
$B \rightarrow F$
$B \rightarrow G$
$B \rightarrow T$
$B \rightarrow O$
$B \rightarrow P$
$P \rightarrow B$
$O \rightarrow T$

2.2) Remove unnecessary attributes from LHS

Nothing can be removed.

2.3) Remove functional dependencies that can be inferred from the rest

$B \rightarrow T$ needs to be removed because without $B \rightarrow T$, $\{B\}^+ \rightarrow BOT$.

3) Write the relations from the minimal basis

(1) $R_1(A,C,D,E,F,G)$
(2) $R_2(B,K,L,M,N,E,F,G,O,P)$
(3) $R_3(E,F,G,H,I,J)$
(4) $R_4(P,Q,R,S,B)$
(5) $R_5(T,U,V)$
(6) $R_6(O,W,X,Y,Z,T)$

4) Add another relation whose schema is a key

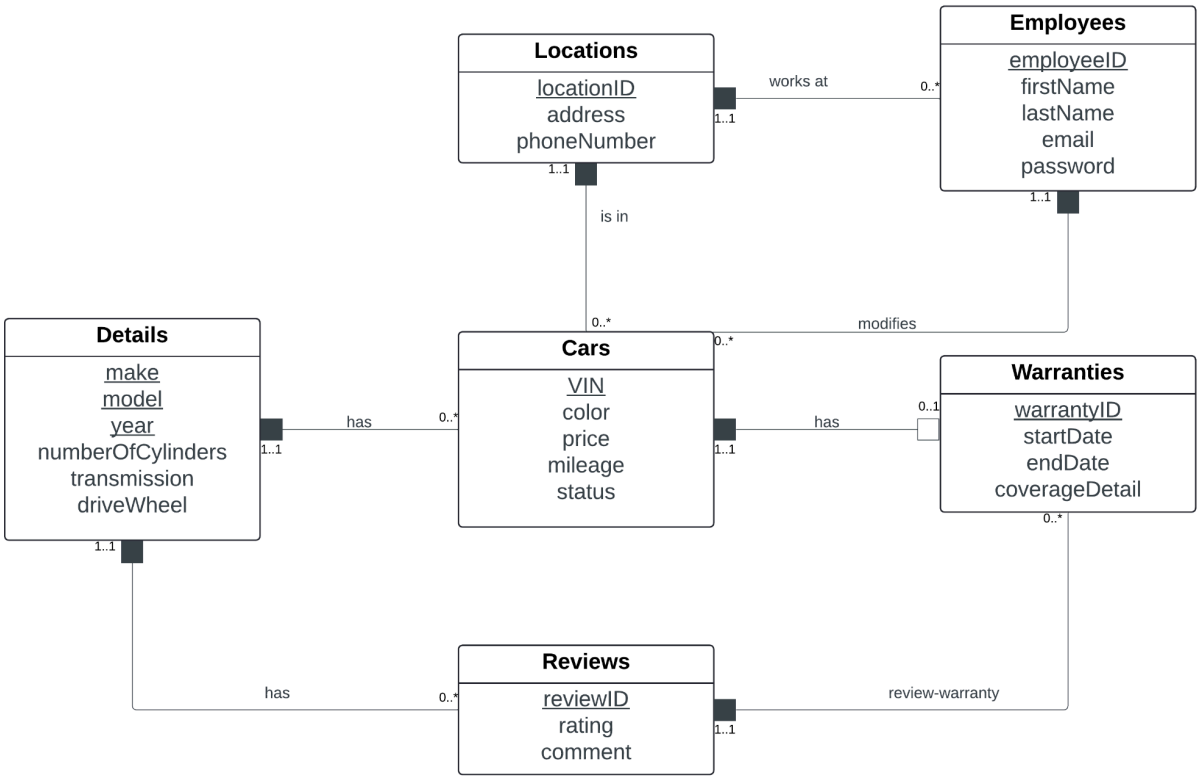
We need to add $R_7(A,B)$ or $R_7(A,P)$ since there is no candidate key in any relations from R_1 to R_7 (in this case, we chose to add the key AP).

Thus, the relations in 3NF are as follows:

(1) $R_1(A,C,D,E,F,G)$
(2) $R_2(B,K,L,M,N,E,F,G,O,P)$
(3) $R_3(E,F,G,H,I,J)$
(4) $R_4(P,Q,R,S,B)$
(5) $R_5(T,U,V)$
(6) $R_6(O,W,X,Y,Z,T)$
(7) $R_7(A,P)$

And the final UML is then:

Why did we choose 3NF?



We chose 3NF because we want to ensure that schema is organized to be free from issues during operations like insert, update, delete. By making sure all attributes are only dependent on the candidate key, we eliminate partial and transitive dependencies and reduce anomalies. By using 3NF, we can add new data without causing incomplete or redundant entries, avoid inconsistencies due to duplicate data, and avoid wrong deletions. We chose it over BCNF because it reduces redundancy while also being more practical because if we went with using BCNF we might have over complicated the UML design. Using 3NF makes our database more efficient, minimizes redundancy, and enhances scalability.

Logical Design (Relational Schema)

Reviews(
    reviewID: INT [PK]
    rating: INT
    comment: VARCHAR(1000)
    make: VARCHAR(20) [FK to Details.make]
    model: VARCHAR(20) [FK to Details.model]
    year: INT [FK to Details.year]
    warrantyID: INT [FK to Warranties.warrantyID]
)

Cars(
    VIN: VARCHAR(17) [PK]
    color: VARCHAR(20)
    price: REAL
    mileage: REAL
    status: VARCHAR(20)
    make: VARCHAR(20) [FK to Details.make]
    model: VARCHAR(20) [FK to Details.model]
    year: INT [FK to Details.year]
    lastModifiedBy: INT [FK to Employees.employeeID]
    warrantyID: INT [FK to Warranties.warrantyID]
)

Details(
    make: VARCHAR(20) [PK]
    model: VARCHAR(20) [PK]
    year: INT [PK]
    numberOfCylinders: INT
    transmission: VARCHAR(15)
    driveWheel: VARCHAR(15)
)

Warranties(
    warrantyID: INT [PK]
    startDate: DATE
    endDate: DATE
    coverageDetail: VARCHAR(200)
    VIN: VARCHAR(17) [FK to Cars.VIN]
)

Locations(
    locationID: INT [PK]
    address: VARCHAR(100)
    phoneNumber: INT
)

Employees(
    employeeID: INT [PK]
    firstName: VARCHAR(20)
    lastName: VARCHAR(20)
    email: VARCHAR(25)
    password: VARCHAR(20)
    locationID: INT [FK to Locations.locationID]
)