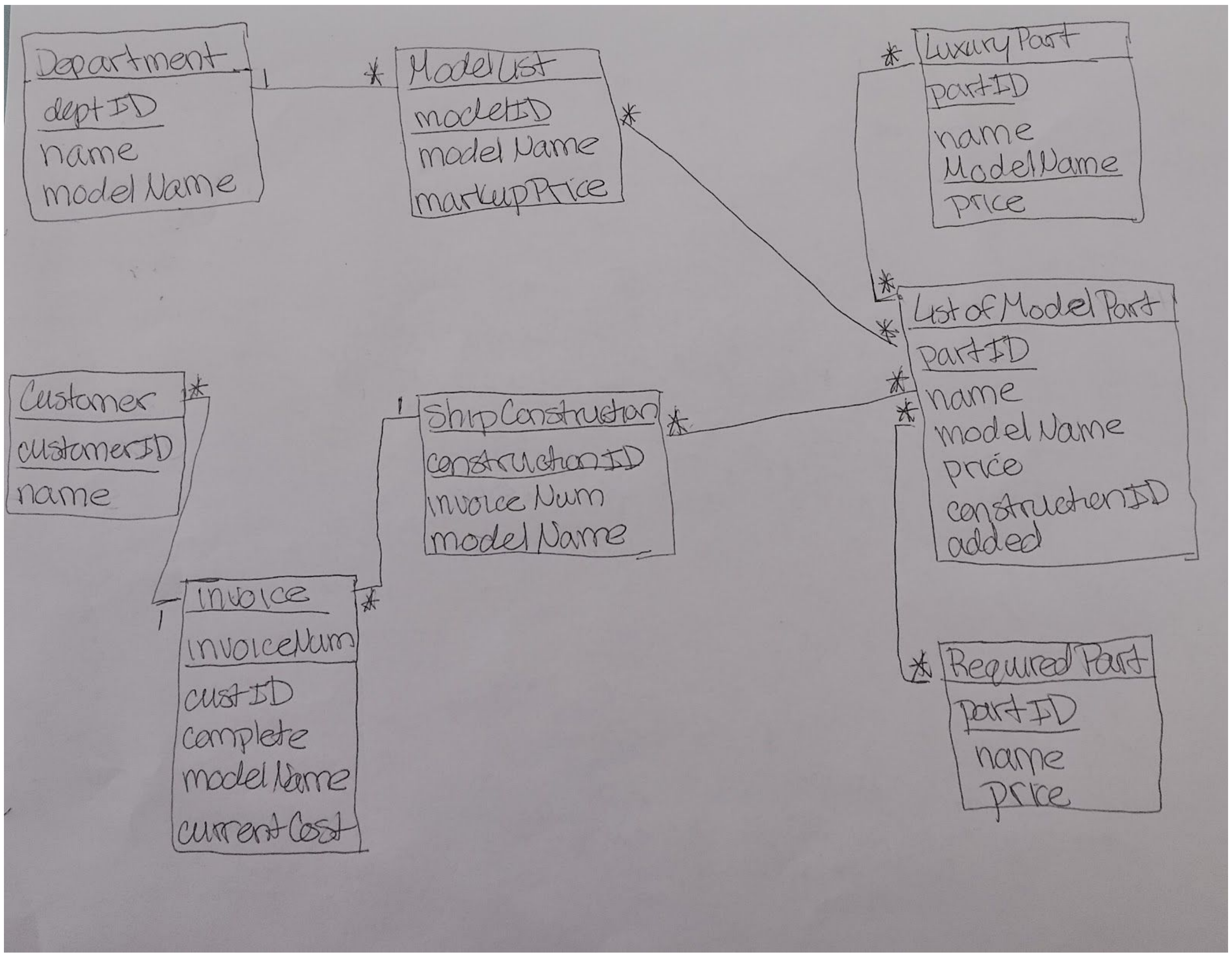
CSc 460 Final Project Write Up
Aakash Rathore
Ana Huff
Briana Liosatos
Tanner Bernth

Conceptual Database Design:

ListOfModelParts changes when a user wants to create a new ship (by creating a new invoice). The ListOfModelParts table then adds all of the parts (luxury and required) needed to complete the ship. The added field will change from 0 to 1 when the user adds a part to that specific ship.

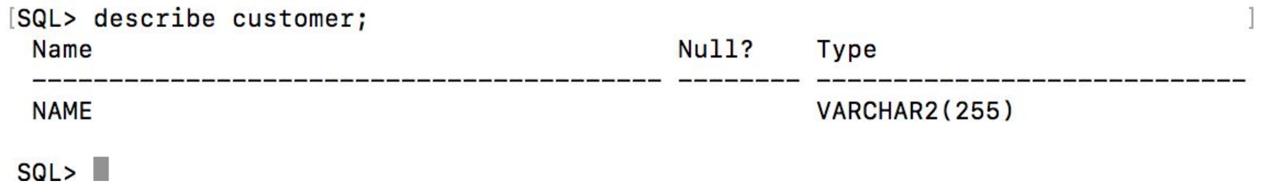
The user will not be able to add two or more of the same parts to the same ship (Ship A cannot have two steering wheels, two dance floors, etc.).

Below is the ER Diagram for the final design of the database.



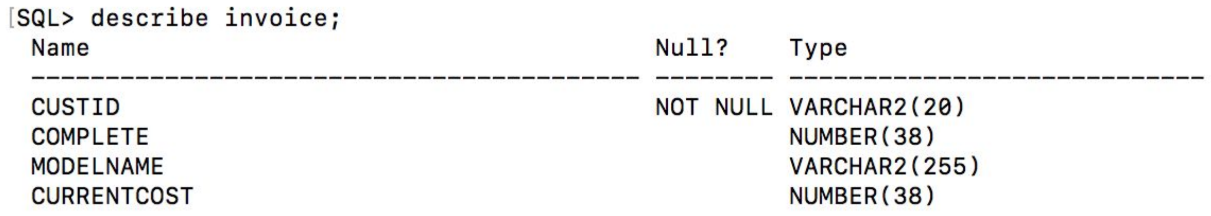
Logical Database Design:
For the tables listed, we have used ROWID as one of the columns for some table. Oracle doesn't provide an auto_increment feature, so we decided to use this because the ROWIDs are unique and are generated automatically by the database.

- Customer Table: **custID** INTEGER PRIMARY KEY NOT NULL (ROWID):



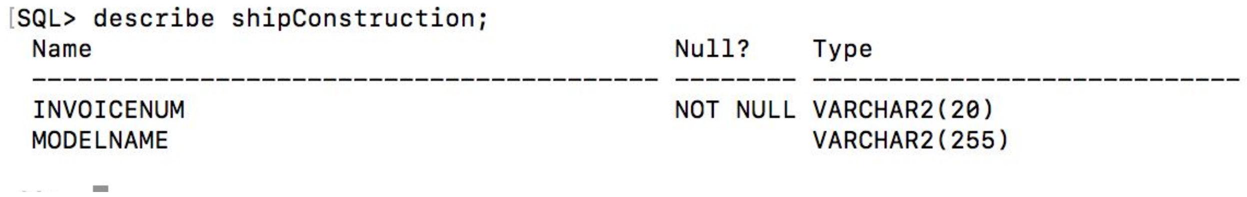
```
[SQL> describe customer;                                                         ]
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(255)

SQL>
```

- Invoice Table: **invoiceNum** integer PRIMARY KEY NOT NULL (ROWID)



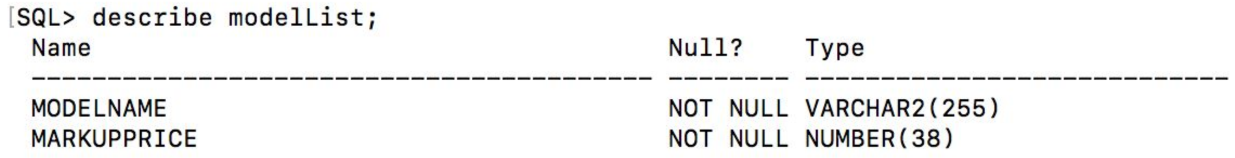
```
[SQL> describe invoice;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 CUSTID                                    NOT NULL VARCHAR2(20)
 COMPLETE                                           NUMBER(38)
 MODELNAME                                          VARCHAR2(255)
 CURRENTCOST                                        NUMBER(38)
```

- shipConstruction: **constructionID** PRIMARY KEY NOT NULL (ROWID)



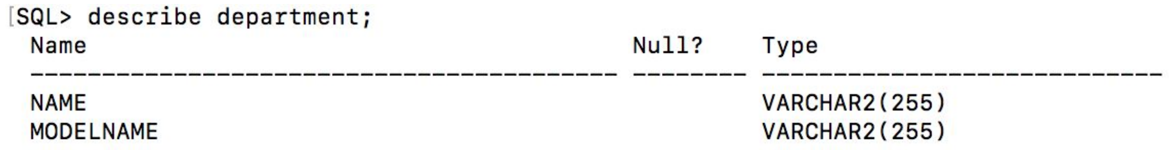
```
[SQL> describe shipConstruction;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INVOICENUM                                NOT NULL VARCHAR2(20)
 MODELNAME                                          VARCHAR2(255)
```

- modeList: **modelID** integer PRIMARY KEY NOT NULL (ROWID)



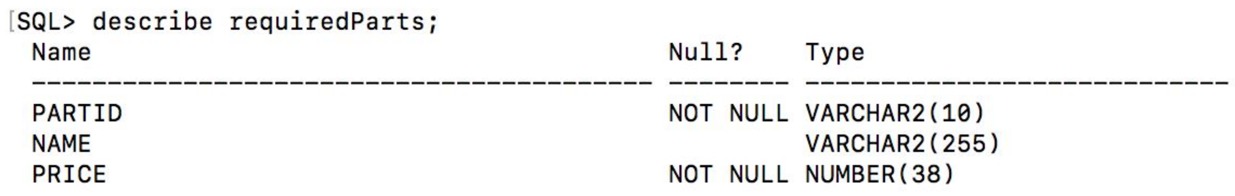
```
[SQL> describe modelList;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MODELNAME                                 NOT NULL VARCHAR2(255)
 MARKUPPRICE                               NOT NULL NUMBER(38)
```

- Department: **deptID** integer PRIMARY KEY NOT NULL AUTO_INCREMENT



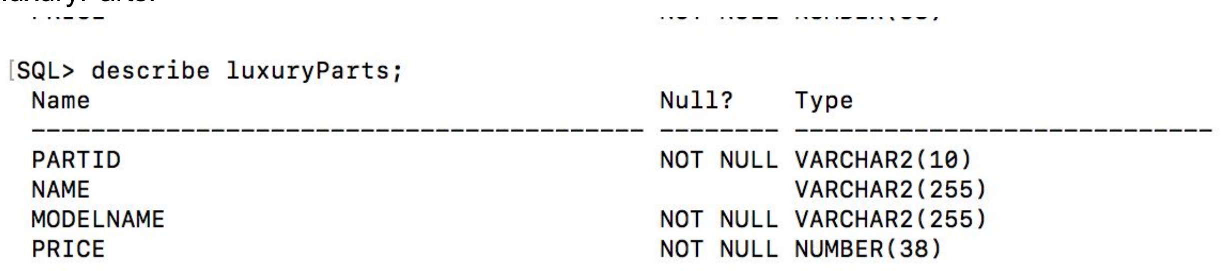
```
[SQL> describe department;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                               VARCHAR2(255)
 MODELNAME                                          VARCHAR2(255)
```

- requiredParts:



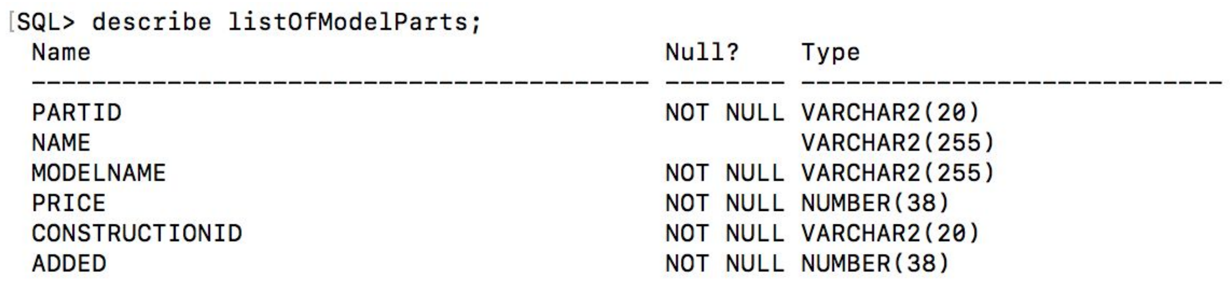
```
[SQL> describe requiredParts;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PARTID                                    NOT NULL VARCHAR2(10)
 NAME                                               VARCHAR2(255)
 PRICE                                     NOT NULL NUMBER(38)
```

- luxuryParts:



```
[SQL> describe luxuryParts;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PARTID                                    NOT NULL VARCHAR2(10)
 NAME                                               VARCHAR2(255)
 MODELNAME                                 NOT NULL VARCHAR2(255)
 PRICE                                     NOT NULL NUMBER(38)
```

- listOfModelParts:



```
[SQL> describe listOfModelParts;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 PARTID                                    NOT NULL VARCHAR2(20)
 NAME                                               VARCHAR2(255)
 MODELNAME                                 NOT NULL VARCHAR2(255)
 PRICE                                     NOT NULL NUMBER(38)
 CONSTRUCTIONID                            NOT NULL VARCHAR2(20)
 ADDED                                     NOT NULL NUMBER(38)
```

<u>Normalization Analysis:</u>

FDs:

Customer-

custID → name

Invoice-

invoiceNum → custID
invoiceNum → complete
invoiceNum → modelName
invoiceNum → currentCost

Ship Construction-

constructionID → invoiceNum
constructionID → modelName

Model List-

modelID → modelName
modelID → markupPrice

Department-

deptID → name
deptID → modelName

Required Parts-

partID → name
partID → price

Luxury Parts-

(partID, modelName) → name
(partID, modelName) → price

List Of Model Parts-

partID → name
partID → price
constructionID → modelName
(partID, constructionID) → added
(partID, constructionID) → name
(partID, constructionID) → price
(partID, constructionID) → modelName

We found that all of our relations were in 3NF. To show this, we will list the normal form progression from 1NF through 3NF. Evey form follows the requirements for the previous form and also answers the additional requirements of that form.

All of the relations are in **1NF** because none of the attributes in each table are set valued. This can be clearly seen in all of the relations.

All of the relations are in **2NF**. All of the relations each have full functional dependencies. If you remove anything from the CK, the dependency will be destroyed. In the luxury parts table, the modelName is needed to identify the luxury parts that are in each individual ship since many different ships can have the same luxury part. In the Customer relation, name is functionally dependent on customerID. If you remove customerID, the functional dependency is destroyed making Customer 2NF. In the Invoice relation, all of the non-prime attributes (custID, complete, modelName, currentCost) are functionally dependent on the invoiceNum. If you remove invoiceNum, it destroys all of those functional dependencies which means the relation is in 2NF. For the ship construction relation, invoiceNum and modelName are functionally dependent on constructionID. If constructionID is removed, the functional dependencies are destroyed as in the previous relations. In the model list relation, modelName and markupPrice are functionally dependent on modelID. The relation is in 2NF because if you remove modelID, it would destroy the functional dependency. Again, the same can be said for the department relation. The modelName and name are both functionally dependent on the deptID and if the deptID is removed, the functional dependencies are destroyed. The required parts relation is also in 2NF due to the same reason. The name and price of the part are both functionally dependent on the partID. Remove partID and destroy the functional dependency. In the luxury parts relation, the CK (partID, modelName) has functional dependencies with name and price. If partID or modelName is removed, the functional dependency will be destroyed. Lastly for 2NF is the list of model parts relation. The CK (partID, constructionID) has functional dependencies with added, name, price, and modelName. If constructionID is removed, the correct modelName will not be found. If partID is removed, the correct part and price will not be found. Removing one or both of these CKs will destroy the functional dependencies which means it is in 2NF.

All but one of the relations are in **3NF**. For all non-trivial functional dependencies in the relations, there is no transitive functional dependency of non-prime attributes on any super key. The customer relation is in 3NF due to X being a superkey, or option (a) in the 3NF definition. The X in this functional dependency is custID, which is also the CK, making it a superkey. The invoice functional dependency is also in 3NF due to the first option (a) in the defintion. The invoiceNum is the CK and a superkey for all of the functional dependencies in that relation. Ship construction is in 3NF for the same reason. It adheres to 3NF through option (a) due to constructionID being a CK and a superkey for the invoiceNum and modelName functional dependencies. The model list relation is 3NF by option (a) in the definition. The modelID is the CK and superkey for all of the functional dependencies. The department relation is 3NF by the (a) option in the definition. The deptID is the CK and superkey for all functional dependencies making the relation 3NF. Again, the required parts relation is 3NF because it has the partID as a CK and a superkey for all of the functional dependencies. This satisfies the (a) option in the definition once again. The luxury parts relation holds under 3NF through option (a) by (partID, modelName) being the CK and superkey for the functional dependencies with name and price. The last relation, list of model parts is not in 3NF. We could not find a way to keep track of the parts that had been added to a ship and have the relation be in 3NF. It is not in 3NF due to the functional dependencies that are with partID and constructionID alone. The partID having functional dependencies with name and price breaks 3NF since X is not a superkey and A is not a prime attribute as the definition for 3NF specifies. The same can be said for the constructionID functional dependency with modelName. If the CK (partID, constructionID) is used, it counts as a superkey and the relation would be in 3NF. Since there is this redundancy in our last relation, it would not be considered 3NF and is only 2NF.

Query Description:

Query 1: SELECT SUM(currentCost) as cost from invoice GROUP BY custID having custID='someID here';
Question: What is the amount of money spent so far by a user both completed and not completed including the markup for the completed ones.
Input: CustomerID

Significance: It lets the users know how much money they have spent so far, in case they needs to know if they should or should not order a new ship.

Query 2: Select ROWID as invoiceNum, custID, modelName, currentCost from invoice WHERE complete=1 AND where custID='someID here'';
Question: List all the ships a user has completed or paid for.
Input: CustomerID
Significance: A simple, yet a useful query. It tells the user about the ships that they have already built and bought.