Madeline Atwood

Database Management

November 19, 2020

Lab 9- Normalization Three

Functional Dependencies Per Table:

1. People

   pid→ firstName, lastName, birthDate

2. Engineers

   pid→ degree, favVideoGame

3. Astronauts

   pid→ startedFlying, goldHandicap, spouseFN, spouseLN

4. Flight Control

   pid→ chairPref, drinkPref, hangoverCure

5. Crew

   composite

6. Space Craft

   SID→ name, tailNum, weightTons, fuel, crewCap, Parts, SystemID

7. Catalog

   composite

8. Suppliers

   supplierID→ name, addressID, paymentTerms

9. Parts

   partID→ name, description, costUSD, supplierID

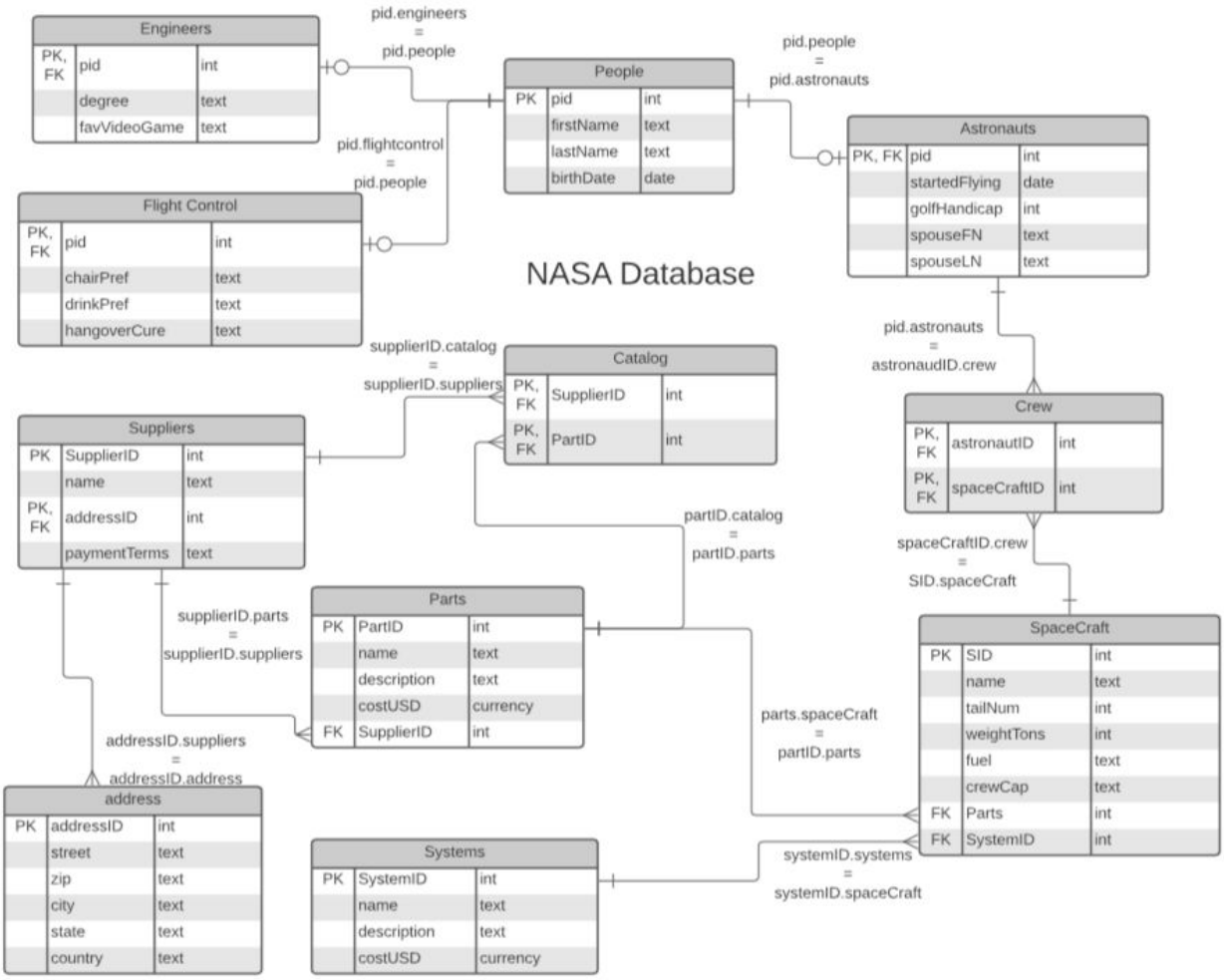
10. Address

    addressID→ street, zip, city, state, country

11. Systems

    SystemID→ name, description, costUSD

Beautiful E/ R Diagram:



Why this is in Boyce-Codd Normal Form:

This beautiful database above is in 1st, 2nd, 3rd and Boyce-Codd NF for many reasons. The first being that it passes 1NF since every value/ piece of data from the table is atomic-

meaning it cannot be broken down anymore. I have even broken down address into its own table so that it passes 1NF- although every variable in the table is categorized as text (zip code being text because a zip code can occasionally have dashes in it and we don't want zip code to be accidentally subtracted) it still breaks up address to make the reading of the data more user friendly and also saves room by adding an address key to maintain continuity if for some reason the key must be used multiple times (if places share an address). This is also in 2NF and 3NF because there are no multiple key dependencies- meaning there can be no partial key dependencies. To aid in this form I have created composite keys to allow for a wide range of data types to be entered into this system and also because of functional dependencies. There is also no many to many relations between the tables- to ensure this there are associate entity tables. Finally, this is in Boyce Codd NF because the transitive dependencies are among the non-key attributes of the table. There is one primary key per table that allows for proper relations and each other depency is between foreign keys.