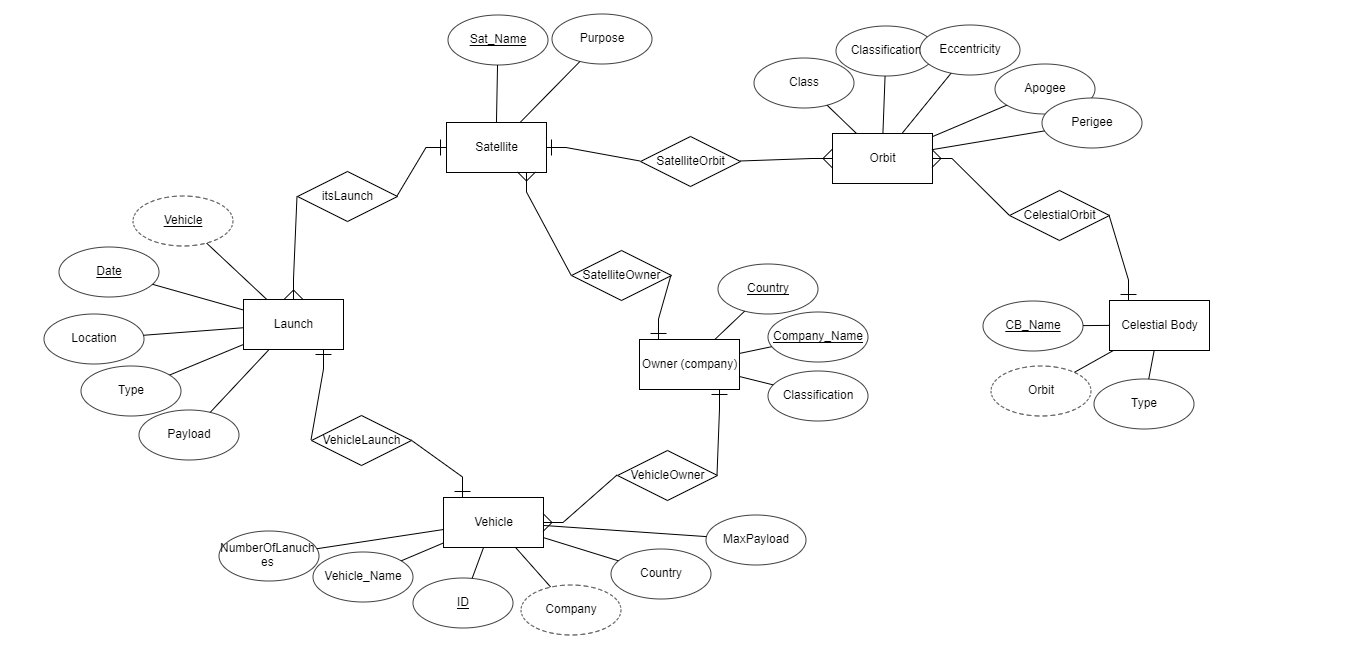
Current orbiting Satellites by Jacques Benzakein, Yakov Yeger, Yosef Berger

Part 1: Creating ERD and defining the Schemas to be used to create our tables

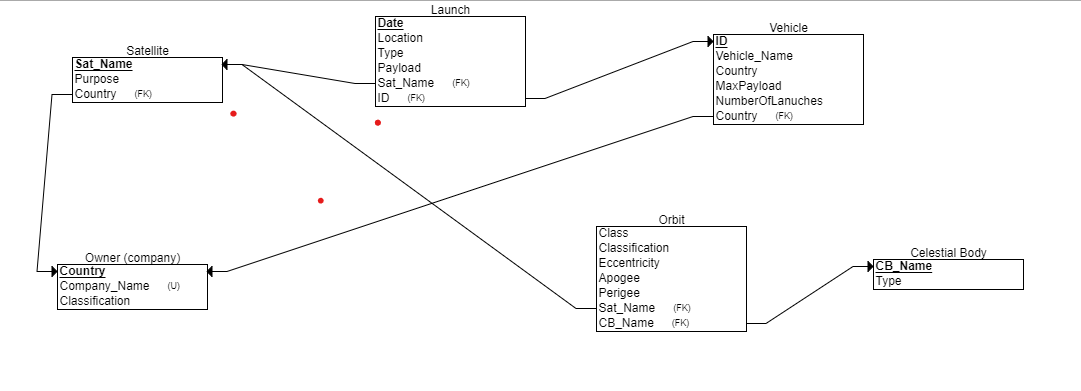
The data for this project is all coming from the site <https://www.ucsusa.org/resources/satellite-database> where there is a myriad of detailed information on the satellites orbiting earth. This data includes pertinent information to the company/countries operating the satellites, their names and their purpose. It also has information pertaining to their launches, vehicles used for the launch as well as specific details regarding the path of their orbit. There are a little over 2600 entries for satellites around the world. The file size is 2341KB which is a relatively small file.

We created an ERD of the data to help with the creation of the schema as seen here:



There is a circular logic to the entities vehicle, launch, owner, and satellite as they all connect to each other in the same way- namely the satellite. We can then see the satellite has its orbit and every orbit is centered around some celestial body so we included that as well.

The following is the automatically generated relational schema:



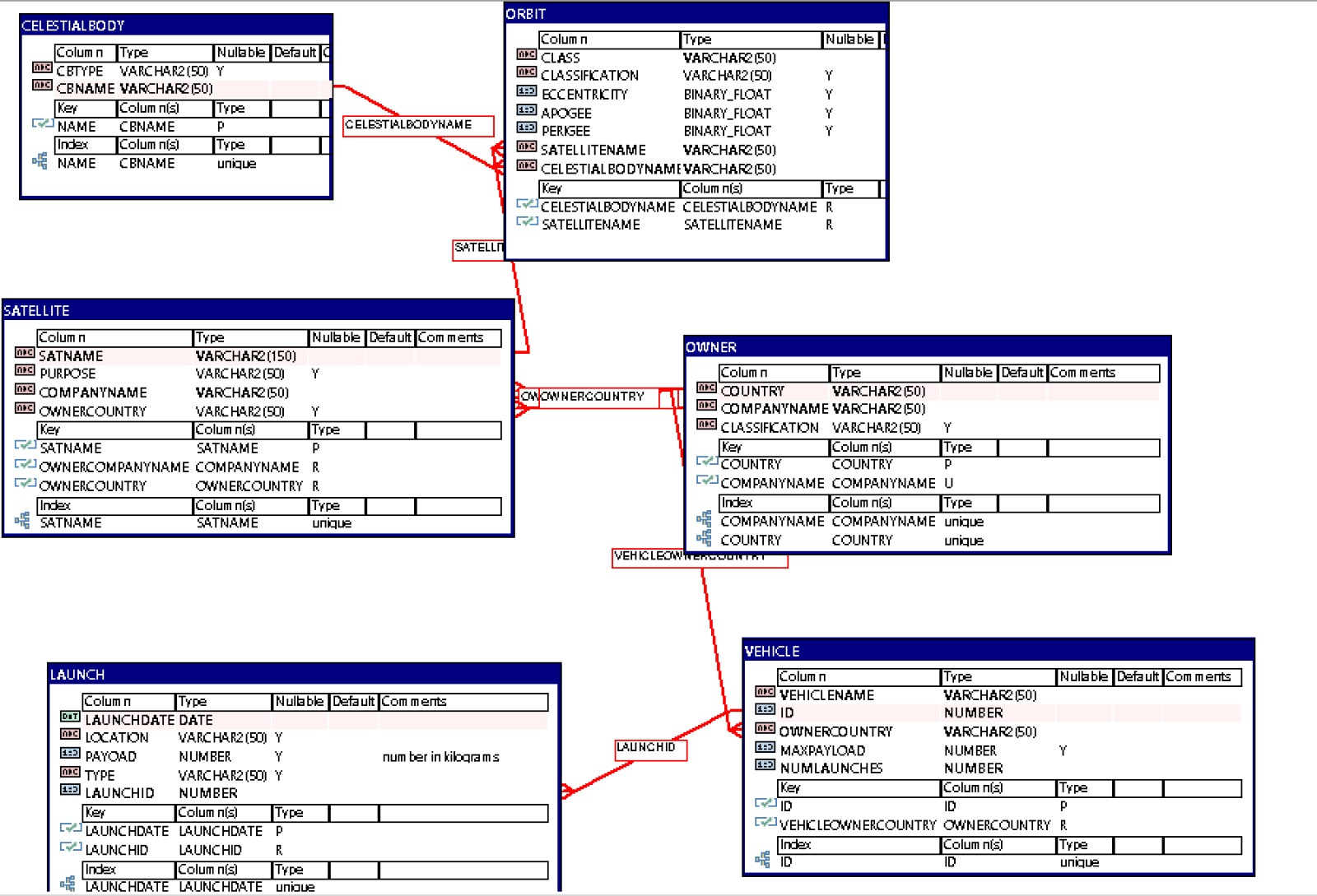
Our table setup in SQL varied slightly from the automatically generated schema with regard to Launch- we removed the foreign key to satellite as each launch could have dozens of satellites attached to it and thus having that relation proved incorrect.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Our next step was to create the tables outlined in the relational schema. We did so using the PL/SQL program using the following commands:

1. CREATE TABLE OWNER
2. (
3. country VARCHAR2(50) NOT NULL,
4. companyname VARCHAR2(50) NOT NULL,
5. classification VARCHAR2(50)
6. )
7. tablespace USERS
8. PCTFREE 10
9. initrans 1
10. maxtrans 255;
11. *-- Create/Recreate primary, unique and foreign key constraints*
12. ALTER TABLE OWNER
13. add constraint COUNTRY primary key (COUNTRY)
14. using INDEX
15. tablespace USERS
16. PCTFREE 10
17. initrans 2
18. maxtrans 255;
19. ALTER TABLE OWNER
20. add constraint COMPANYNAME UNIQUE (COMPANYNAME)
21. using INDEX
22. tablespace USERS
23. PCTFREE 10
24. initrans 2
25. maxtrans 255;
26. *-- Create table*
27. CREATE TABLE celestialBody
28. (
29. TYPE VARCHAR2(50),
30. name VARCHAR2(50) NOT NULL
31. )
32. ;
33. *-- Create/Recreate primary, unique and foreign key constraints*
34. ALTER TABLE celestialBody
35. add constraint name primary key (NAME);
37. *-- Create table*
38. CREATE TABLE SATELLITE
39. (
40. satname VARCHAR2(150) NOT NULL,
41. purpose VARCHAR2(50),
42. companyname VARCHAR2(50) NOT NULL,
43. ownercountry VARCHAR2(50)
44. )
45. *-- Create table*
46. CREATE TABLE VEHICLE
47. (
48. vehiclename VARCHAR2(50) NOT NULL,
49. id NUMBER NOT NULL,
50. ownercountry VARCHAR2(50) NOT NULL,
51. maxpayload NUMBER,
52. numlaunches NUMBER NOT NULL
53. )
54. tablespace USERS
55. PCTFREE 10
56. initrans 1
57. maxtrans 255;
58. *-- Create/Recreate primary, unique and foreign key constraints*
59. ALTER TABLE VEHICLE
60. add constraint ID primary key (ID)
61. using INDEX
62. tablespace USERS
63. PCTFREE 10
64. initrans 2
65. maxtrans 255;
66. ALTER TABLE VEHICLE
67. add constraint VEHICLEOWNERCOUNTRY foreign key (OWNERCOUNTRY)
68. references OWNER (COUNTRY);
69. *-- Create table*
70. CREATE TABLE ORBIT
71. (
72. class VARCHAR2(50) NOT NULL,
73. classification VARCHAR2(50),
74. eccentricity BINARY\_FLOAT,
75. apogee BINARY\_FLOAT,
76. perigee BINARY\_FLOAT,
77. satellitename VARCHAR2(50) NOT NULL,
78. celestialbodyname VARCHAR2(50) NOT NULL
79. )
80. tablespace USERS
81. PCTFREE 10
82. initrans 1
83. maxtrans 255;
84. *-- Create/Recreate primary, unique and foreign key constraints*
85. ALTER TABLE ORBIT
86. add constraint CELESTIALBODYNAME foreign key (CELESTIALBODYNAME)
87. references CELESTIALBODY (CBNAME);
88. ALTER TABLE ORBIT
89. add constraint SATELLITENAME foreign key (SATELLITENAME)
90. references SATELLITE (SATNAME);
91. *-- Create table*
92. CREATE TABLE LAUNCH
93. (
94. launchdate DATE NOT NULL,
95. location VARCHAR2(50),
96. TYPE VARCHAR2(50),
97. launchid NUMBER NOT NULL
98. )
99. tablespace USERS
100. PCTFREE 10
101. initrans 1
102. maxtrans 255;
103. *-- Add comments to the columns*
104. COMMENT ON column LAUNCH.payload
105. IS 'number in kilograms';
106. *-- Create/Recreate primary, unique and foreign key constraints*
107. ALTER TABLE LAUNCH
108. add constraint LAUNCHDATE primary key (LAUNCHDATE)
109. using INDEX
110. tablespace USERS
111. PCTFREE 10
112. initrans 2
113. maxtrans 255;
114. ALTER TABLE LAUNCH
115. add constraint LAUNCHID foreign key (LAUNCHID)
116. references VEHICLE (ID);

The PLSQL Schema:



We then ran some insertion commands to and tested the validity of the scripts:

**Test Script: Must insert to owner first as satellite uses that information**

INSERT INTO OWNER

VALUES('USA', 'SpaceX', 'Commercial');

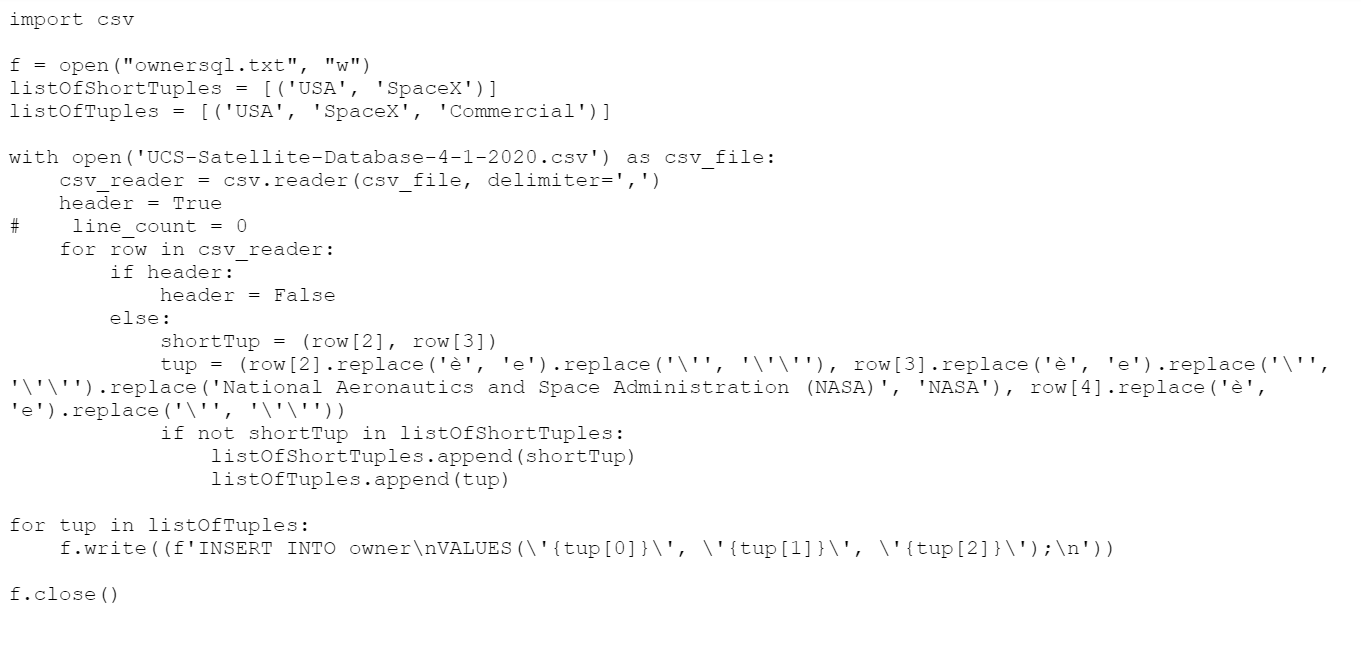
INSERT INTO SATELLITE

VALUES('Starlink-1007', 'Communication', 'SpaceX', 'USA');

SELECT \* FROM SATELLITE;

(Attached is the table report generated by PLSQL)

We imported the data for the owner table using a python script (attached) to convert the excel file into insert commands file which is also attached.



We then used Oracles built in system for importing data from an excel sheet:   
UCS-Satellite-Database-4-1-2020.xls

We used the following scripts to ensure the data had been imported properly and that the tables functions as expected:  
SELECT \* FROM vehicle WHERE ownercountry LIKE '%Israel%';

SELECT \* FROM vehicle WHERE ownercountry LIKE '%USA%';

SELECT country

FROM owner

WHERE classification = 'Military';

Following this, we ran some tests to ensure the backup was working using the following drop table commands:

DROP TABLE celestialbody;

DROP TABLE launch;

DROP TABLE orbit;

DROP TABLE owner;

DROP TABLE satellite;

DROP TABLE vehicle;

We opted not to use the data generator at this stage of the project but will be using it to test performance of the database down the line.