Employee

R={Fname, MI, Lname, Phone_num, Work_id,Title}

F={(Work_id->Fname, MI, Lname, Phone_num, Title),(Phone_num->Fname, MI, Lname, Work_id, Title}

Candidate Keys=Phone_num, Work_id

There are two functional dependencies in this and each of them has a candidate key on the left side. The determinate of each function dependency is a candidate key and the candidate keys are not overlapping. Also, all attributes are dependent on a candidate key. This relation is in BCNF.

Side note-Phone_num is a candidate key because an individuals phone number in the United States is unique. If the phone number is international then it is not necessarily unique unless you include country code which we are not. However, because this is for NASA I am assuming that all phone numbers are from the U.S.

Mission

R={Name, Mission_ID, Mleader_id, Ongoing}

F={(Mission_ID->Name, Mleader_id, Ongoing),(Mleader_id->Name,Mission_id,Ongoing)}

Candidate Keys=Mission_ID, Mleader_id

There are two functional dependencies in this and each of them has a candidate key on the left side. The determinate of each function dependency is a candidate key and the candidate keys are not overlapping. Also, all attributes are dependent on a candidate key. This relation is in BCNF.

Project

R={Name, Proj_id, Mission_id, Pleader_id}

F={(Proj_id->Name, Mission_id, Pleader_id),(Pleader_id->Name, Proj_ID, Mission_id)}

Candidate Keys=Proj_id, Pleader_id

There are two functional dependencies in this and each of them has a candidate key on the left side. The determinate of each function dependency is a candidate key and the candidate keys are not overlapping. Also, all attributes are dependent on a candidate key. This relation is in BCNF.

Side note-Mission_id is not on the left side of a functional dependency because Mission_id cannot uniquely determine any project because multiple projects can be in a mission.

Contractor
R={Name, Contractor_id}
F={(Contractor_id->Name)}

Candidate Keys=Contractor_id

There is one functional dependency because a contractor name cannot necessarily determine an ID. Two contractors could possibly have the same name especially if we only include part of the company name. This relation is in BCNF.

Works_on
R={Employee_id, Proj_id}
F={(Employee_id->Proj_id)}
Candidate Keys=Employee_id

There is only one functional dependency. Multiple employees can work for a single project so Proj_id cannot determine an employee id. Employee ID can determine Proj_id because an Employee cannot work on multiple projects so it is unique. This relation is in BCNF

Supplies

R={Proj_id, Contractor_id, Cost}

F={(Proj_id->Contractor_id, Cost)}

Candidate Keys=Proj_id

There is only one functional dependency based on the candidate key Proj_id. Contractor_id and Cost cannot be or be part of a candidate key because contractors can work on multiple projects. Projects, however, only have one contractor. The cost is also not unique because projects could cost the same amount. {Contractor_id, Cost} is not a candidate key because there could be multiple projects that cost the same and have the same contractor. This is in BCNF because all attributes are dependent on a candidate key and the candidate key is not overlapping.