University of British Columbia, Vancouver
Department of Computer Science

Deliverables

Project Summary

We are modeling the relationship between the increasing usage of satellite technology, and the amount of space pollution around the Earth. Our database is designed to store information about launching and maintaining satellites, as well as the debris this creates.

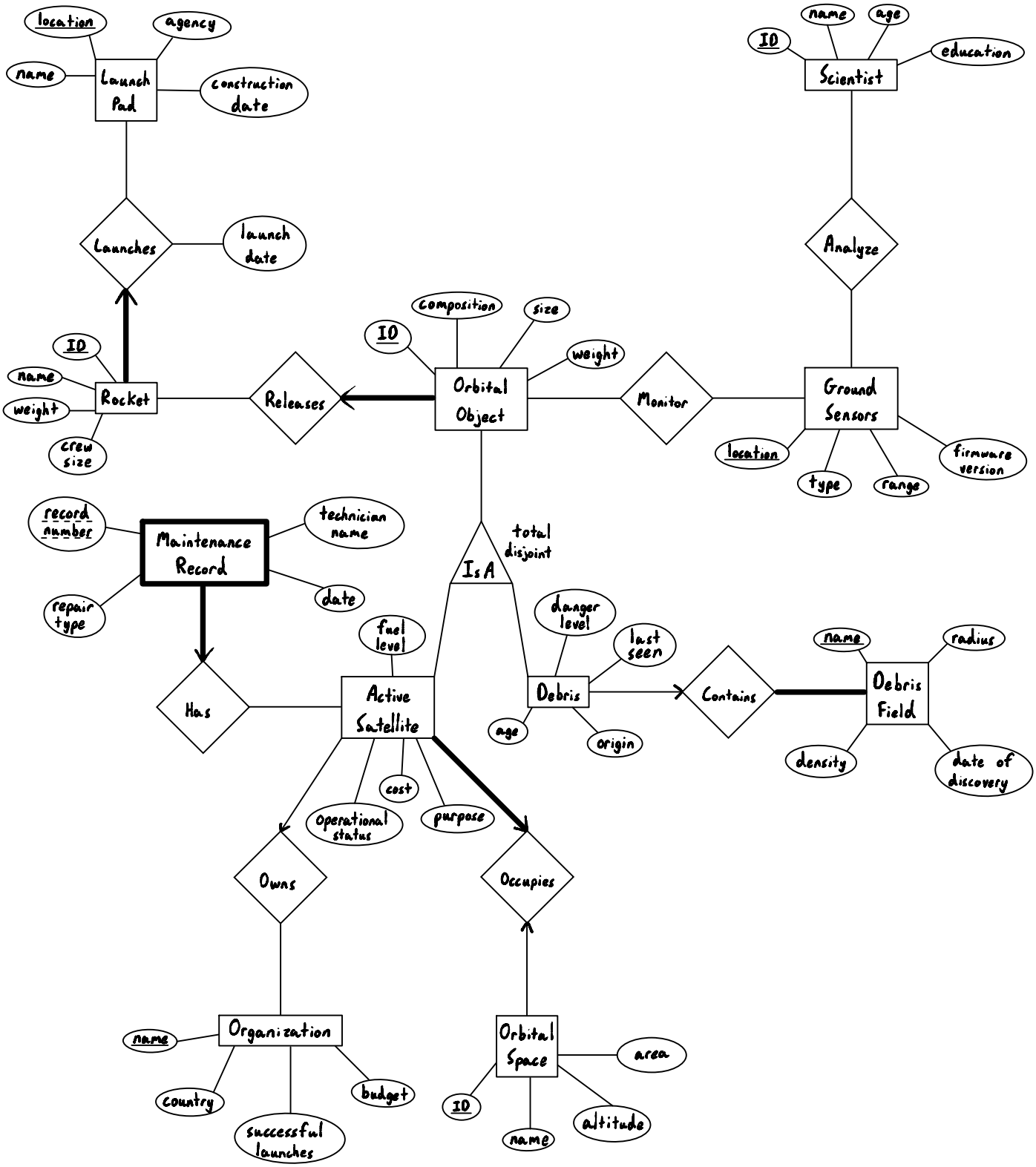
Tech Stack

We will be using Javascript, React, and the department provided Oracle database.

ER Diagram

See next page

Space Debris



University of British Columbia, Vancouver
Department of Computer Science

Database Schema

Independent Entities:

LaunchPad (location: varchar, agency: varchar,name: varchar, constructionDate: date)
- CK: location, name
- PK: location

Rocket (ID: int, name: varchar, weight: float, crewSize: int, launchPadLocation: varchar, launchDate: date)
- CK: ID, name
- PK: ID
- FK: launchPadLocation (NOT NULL)

Scientist (ID: INT, name: varchar, age: varchar, education: varchar)
- CK: ID
- PK: ID

Organization (name: varchar, country: varchar, sucessfulLaunches: int, budget: float)
- CK: name
- PK: name

OrbitalSpace (ID: int, name, varchar, altitude: float, area: float)
- CK: ID, name
- PK: ID

Note: Total participation constraint cannot be fully enforced for now (for activeSatellite.ID)

DebrisField (name: varchar, radius: varchar, density: varchar, discoveredDate: date)
- CK: name
- PK: name

Note: Total participation constraint cannot be fully enforced for now (for Debris.ID)

Note: dateOfDiscovery has been renamed to discoveredDate for readability

ActiveSatellite (ID: int, composition: varchar, size: int, weight: float, fuelLevel: int, operationalStatus: varchar, cost: float, purpose: varchar, orbitalSpaceID: int, organizationName: varchar, rocketID: int)

- CK: ID, purpose
- PK: ID
- FK: orbitalSpaceID references OrbitalSpace.ID (NOT NULL and UNIQUE)
- FK: organizationName references Organization.name
- FK: rocketID references Rocket.ID (NOT NULL)

Note: Total participation constraint cannot be fully enforced for now (for OrbitalSpaceID)
Note: Total participation constraint cannot be fully enforced for now (for RocketID)

Debris (ID: int, composition: varchar, size: float, weight: float, age: int, origin: varchar, dangerLevel: varchar, lastSeen: date, debrisFieldName: varchar, rocketID: int)

- CK: ID, origin
- PK: ID
- FK: debrisFieldName references DebrisField.name (NOT NULL)
- FK: rocketID references Rocket.ID (NOT NULL)

Note: Total participation constraint cannot be fully enforced for now (for DebrisFieldName)
Note: Total participation constraint cannot be fully enforced for now (for RocketID)

GroundSensors (location: varchar, type: varchar, range: float, firmwareVersion: varchar)

Weak Entity:
MaintenanceRecord (recordNumber: int, activeSatelliteID: int, repairType: varchar, technicianName: varchar, repairDate: date)

- PK: recordNumber and activeSatelliteID
- FK: activeSatelliteID references activeSatellite.ID (NOT NULL)

Note: We changed the attribute date to repairDate to avoid confusion with the SQL type DATE.

Relationships:

MonitorSatellite (groundSensorLocation: varchar, activeSatelliteID: int)

- PK: groundSensorLocation & activeSatelliteID
- FK: groundSensorLocation references groundSensor.location
- FK: activeSatelliteID references activeSatellite.ID

MonitorDebrisgroundSensorLocation: varchar, debrisID: int)

- PK: groundSensorLocation & debrisID
- FK: groundSensorLocation references groundSensor.location
- FK: debrisID references debris.ID

Analyze (groundSensorLocation: varchar, scientistID: int)

- PK: groundSensorLocation & scientistID
- FK: groundSensorLocation references groundSensor.location
- FK: scientistID references scientist.ID

Functional Dependencies:

Independent Entities:

- **LaunchPad.location** -> LaunchPad.agency, LaunchPad. name, LaunchPad.constructionDate
- **Rocket.ID** -> Rocket.name, Rocket.weight, Rocket.crewSize
- **Scientist.ID** -> Scientist.name, Scientist.age, Scientist.education
- **Organization.name** -> Organization.country, Organization.successfulLaunches, Organization.budget
- **ObitialSpace.ID** -> ObitialSpace.name, ObitialSpace.altitude, ObitialSpace.area
- **DebrisField.name** -> DebrisField.raduis, DebrisField.density, DebrisField.discoveryDate
- **ActiveSatellite.ID** -> ActiveSatellite.composition, ActiveSatellite.size, ActiveSatellite.weight, ActiveSatellite.fuelLevel, ActiveSatellite.operationalStatus, ActiveSatellite.purpose, ActiveSatellite.cost, Organization.name, OrbitalSpace.ID, Rocket.ID
- **Debris.ID** -> Debris.composition, Debris.size, Debris.weight, Debris.dangerLevel, Debris.lastSeen, Debris.age, Debris.origin, DebrisField.name, Rocket.ID

- **GroundSensors.location** -> GroundSensors.type, GroundSensors.range, GroundSensors.firmwareVersion

Weak Entity:

- **MaintenanceRecord.recordNumber, ActiveSatellite.ID** -> MaintenanceRecord.technicianName, MaintenanceRecord.repairType, MaintenanceRecord.date

Relationships:

- **Rocket.ID** -> LaunchPad.location, Launches.launchDate
- **OrbitalSpace.ID** -> ActiveSatellite.ID

Extra Functional Dependencies:

- **Rocket.name** -> Rocket.crewSize
- **Organization.country** -> Organization.budget
- **DebrisField.radius** -> DebrisField.density

Normalization

Decompose to BCNF:

1. **Rocket.name** -> Rocket.crewSize

Consider the relation Rocket (ID, name, weight, crewSize, launchPadLocation, launchDate):

- **Rocket.name** -> Rocket.crewSize
- **Rocket.ID** -> Rocket.name, Rocket.weight, Rocket.crewSize, launchPadLocation, launchDate

Closures:

- Rocket.ID+ = {Rocket.ID, Rocket.name, Rocket.weight, Rocket.crewSize, LaunchPad.location, Launches.launchDate}
- Rocket.name+ = {Rocket.name, Rocket.crewSize}
- Rocket.weight+ = {Rocket.weight}
- Rocket.crewSize+ = {Rocket.crewSize}
- Rocket.launchPadLocation+ = {Rocket.launchPadLocation}
- Rocket.launchDate+ = {Rocket.launchDate}

Rocket.name -> Rocket.crewSize violates BCNF because although Rocket.name uniquely determines a rocket's crewSize, it is not listed as a primary key. Hence, it must be decomposed.

Decomposition:

Decompose **Rocket.name** -> Rocket.crewSize

R1(Rocket.name, Rocket.crewSize)
R2(Rocket.ID, Rocket.name, Rocket.weight, Rocket.launchPadLocation, Rocket.launchDate)

Hence, the final relations are: R1(Rocket.name, Rocket.crewSize)
R2(Rocket.ID, Rocket.name, Rocket.weight, Rocket.launchPadLocation, Rocket.launchDate)

2. **Organization.country** -> Organization.budget

Consider the relation Organization(name, country, successfulLaunches, budget):

- Organization.name -> Organization.country, Organization.successfulLaunches, Organization.budget
- Organization.country->Organization.budget

Closures:

- Organization.name+ = {Organization.name, Organization.country, Organization.successfulLaunches, Organization.budget}
- Organization.country+ = {Organization.country, Organization.budget}
- Organization.successfulLaunches += {Organization.successfulLaunches}
- Organization.budget+ = {Organization.budget}

Organization.country->Organization.budget violates BCNFsince it is not a primary key while also being able to define another attribute. Hence it must be decomposed.

Decomposition:

Decompose **Organization.country**->Organization.budget

R1(Organization.country, Organization.budget)

R2(Organization.name, Organization.country, Organization.successfulLaunches)

Hence, the final relations are: R1(Organization.country, Organization.budget), R2(Organization.name, Organization.country, Organization.successfulLaunches)

3. **DebrisField.radius** -> DebrisField.density

Consider the relation DebrisField(name, radius, density, discoveredDate):

- **DebrisField.radius** -> DebrisField.density
- **DebrisField.name** -> DebrisField.raduis, DebrisField.density, DebrisField.discoveryDate

Closures:

- DebrisField.name+ = {DebrisField.name, DebrisField.radius, DebrisField.density, DebrisField.density, DebrisField.discoveryDate}
- DebrisField.radius+ = {DebrisField.radius, DebrisField.density}
- DebrisField.density+ = {DebrisField.density}
- DebrisField.discoveryDate+ = {DebrisField.discoveryDate}

We determine that **DebrisField.radius** -> DebrisField.density violates BCNF since DebrisField.radius is not a primary key nor does it uniquely determine a Debris Field's density. Hence, it must be decomposed.

Decomposition:

Decompose **DebrisField.radius** -> DebrisField.density

R1(DebrisField.radius, DebrisField.density)
R2(DebrisField.name, DebrisField.raduis, DebrisField.discoveryDate)

Hence, the final relations are: R1(DebrisField.radius, DebrisField.density) R2(DebrisField.name, DebrisField.raduis, DebrisField.discoveryDate)

SQL/DDL

Citation: We found ON DELETE RESTRICT on a stack overflow question about explaining SQL delete rules. Link to question: https://dba.stackexchange.com/questions/44956/good-explanation-of-cascade-on-delete-update-behavior

---

— Entity Tables —

```
CREATE TABLE LaunchPad (
        location VARCHAR(20) PRIMARY KEY,
        agency VARCHAR(20),
        name VARCHAR(20) UNIQUE,
        constructionDate DATE
)

CREATE TABLE Organization (
        name VARCHAR(20) PRIMARY KEY,
        country VARCHAR(20),
        successfulLaunches INTEGER,
        budget FLOAT
)

CREATE TABLE OrbitalSpace (
        ID INTEGER PRIMARY KEY,
        name VARCHAR,(20)
        altitude FLOAT,
        area FLOAT
)

CREATE TABLE DebrisField (
        name VARCHAR(20) PRIMARY KEY
        radius FLOAT,
        density FLOAT,
        discoveryDate DATE
)

CREATE TABLE Scientist (
        ID INTEGER PRIMARY KEY,
        name VARCHAR(20),
        age INTEGER,
        education VARCHAR
)

CREATE TABLE GroundSensors (
        location VARCHAR(20) PRIMARY KEY,
```

```
    type VARCHAR(10),
    range FLOAT,
    firmwareVersion VARCHAR(20),
)

CREATE TABLE Rocket (
    ID INT PRIMARY KEY,
    name VARCHAR(20) UNIQUE,
    weight FLOAT,
    crewSize INTEGER
    launchPadLocation VARCHAR(20) NOT NULL,
    launchDate DATE,
    FOREIGN KEY (launchPadLocation) REFERENCES LaunchPad ON
    DELETE RESTRICT ON UPDATE CASCADE
)
```

Deletion Explanation: For references to LaunchPad we will restrict deletion if any Rockets still reference it to avoid a case where deleting a LaunchPad deletes the majority of the database.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

We choose to include the Launches relationship as part of the Rocket table because of the participation constraint on Rocket in our ER diagram. Due to the participation constraint having a separate table for Launches

```
CREATE TABLE ActiveSatellite (
    ID INTEGER PRIMARY KEY,
    composition VARCHAR(20),
    size INTEGER,
    weight FLOAT,
    fuelLevel INTEGER,
    operationalStatus VARCHAR(20),
    purpose VARCHAR,
    cost FLOAT,
    rocketID INTEGER NOT NULL,
```

```
    organizationName VARCHAR(20),
    orbitalSpaceID INTEGER NOT NULL UNIQUE,
    FOREIGN KEY (rocketID) REFERENCES Rocket ON DELETE
    RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (organizationName) REFERENCES Organization ON DELETE
    SET NULL ON UPDATE CASCADE,
    FOREIGN KEY (orbitalSpaceID) REFERENCES OrbitalSpace ON
    DELETE CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: For references to Rocket we will restrict deletion if any OrbitalObject superclasses still reference it to avoid a case where deleting a Rocket deletes the majority of the database.

For references to Organization we can set the relationship to NULL as our ER diagram allows ActiveSatellites to be owned by zero or one Organizations.

For references to OrbitalSpace we want the deletion to cascade as our ER diagram does not allow ActiveSatellites to exist without an OrbitalSpace

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

```
CREATE TABLE Debris (
    ID INTEGER PRIMARY KEY,
    composition VARCHAR(20),
    size INTEGER,
    weight FLOAT,
    dangerLevel VARCHAR(20),
    lastSeen DATE,
    age INTEGER,
    origin VARCHAR(20),
    rocketID INTEGER NOT NULL,
    debrisFieldName VARCHAR(20),
    FOREIGN KEY (rocketID) REFERENCES Rocket ON DELETE
    RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (debrisFieldName) REFERENCES DebrisField ON
```

```
        DELETE SET NULL ON UPDATE CASCADE
)
```

Deletion Explanation: For references to Rocket we will restrict deletion if any OrbitalObject superclasses still reference it to avoid a case where deleting a Rocket deletes the majority of the database.

For references to Debris we can set the relationship to NULL as our ER diagram allows debris to exist outside of a DebrisField

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

— Weak Entity Table —

```
CREATE TABLE MaintenanceRecord (
        technicianName VARCHAR(20),
        repairType VARCHAR(20),
        repairDate DATE,
        activeSatelliteID INTEGER,
        recordNumber INTEGER,
        PRIMARY KEY (recordNumber, activeSatelliteID),
        FOREIGN KEY (activeSatelliteID) REFERENCES ActiveSatellite ON
        DELETE CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: This is a weak entity, so based on the slides when the owner entity is deleted all of its weak entities must be deleted as well.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to the owner entity to be reflected in the weak entity.

— Relationship Tables —

```
CREATE TABLE Analyze (
        scientistID INTEGER,
        groundSensorsLocation VARCHAR(20),
        PRIMARY KEY (scientistID, groundSensorsID),
```

```
        FOREIGN KEY (scientistID) REFERENCES Scientist ON DELETE
        CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (groundSensorsLocation) REFERENCES
        GroundSensors ON DELETE CASCADE ON UPDATE CASCADE,
)
```

Deletion Explanation: For references to Scientist and GroundSensors will will simply delete the tuple in the Analyze relationship, as they are in a many-to-many relationship with no participation constraints.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

```
CREATE TABLE MonitorActiveSatellite (
        groundSensorsLocation VARCHAR(20),
        activeSatelliteID INTEGER,
        PRIMARY KEY (groundSensorsLocation, activeSatelliteID),
        FOREIGN KEY (groundSensorsLocation) REFERENCES
        GroundSensors ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (activeSatelliteID) REFERENCES ActiveSatellite ON
        DELETE CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: For references to ActiveSatellite and GroundSensors will simply delete the tuple in the Monitors relationship, as they are in a many-to-many relationship with no participation constraints.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

```
CREATE TABLE MonitorDebris (
        groundSensorsLocation VARCHAR(20),
        debrisID INTEGER,
        PRIMARY KEY (groundSensorsLocations, debrisID),
```

```
        FOREIGN KEY (groundSensorsLocation) REFERENCES
        GroundSensors ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (debrisID) REFERENCES Debris ON DELETE
        CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: For references to Debris and GroundSensors will will simply delete the tuple in the Monitors relationship, as they are in a many-to-many relationship with no participation constraints.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

— Normalized Tables —

```
CREATE TABLE RocketTeam (
        name VARCHAR(20) PRIMARY KEY,
        size INTEGER
)

CREATE TABLE Rocket (
        ID INTEGER PRIMARY KEY,
        weight FLOAT,
        name VARCHAR(20),
        launchPadLocation VARCHAR(20),
        FOREIGN KEY (name) REFERENCES RocketTeam ON DELETE CASCADE ON
        UPDATE CASCADE,
        FOREIGN KEY (launchPadLocation) REFERENCES LaunchPad ON DELETE
        RESTRICT ON UPDATE CASCADE
)
```

Deletion Explanation: Since we normalized Rocket into two tables, to maintain the original entity deletions of RocketTeam should cascade onto the normalized Rocket.

For references to LaunchPad we follow the same logic as the non-normalized table.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

```
CREATE TABLE OrganizationFunding (
        country VARCHAR(20) PRIMARY KEY,
        budget FLOAT
)

CREATE TABLE Organization (
        name VARCHAR(20) PRIMARY KEY,
        country VARCHAR(20),
        sucessfulLaunches INTEGER,
        FOREIGN KEY (country) REFERENCES OrganizationFunding ON DELETE
        CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: Since we normalized Organization into two tables, to maintain the original entity deletions of OrganizationFunding should cascade onto the normalized Organization.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

```
CREATE TABLE DebrisFieldRelativeSize (
        radius FLOAT PRIMARY KEY
        density FLOAT
)

CREATE TABLE DebrisField (
        name VARCHAR(20) PRIMARY KEY,
        radius FLOAT,
        discoveryDate DATE,
        FOREIGN KEY (radius) REFERENCES DebrisFieldRelativeSize ON DELETE
        CASCADE ON UPDATE CASCADE
)
```

Deletion Explanation: Since we normalized DebrisField into two tables, to maintain the original entity deletions of DebrisFieldRelativeSize should cascade onto the normalized DebrisField.

Note: We understand that Oracle does not support ON UPDATE CASCADE, but we have included it here as we would like updates to any of the referenced tables to be reflected in this table as well.

Generative AI Acknowledgement

Generative AI and AI tools were not used to create, edit, or otherwise produce any of the content in this document.