Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение

высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе № 4

по дисциплине «Базы данных»

Выполнили: Борисов Георгий

Граник Артем

Факультет: ПИиКТ

Группа: P33212

Цель работы.

Для выполнения лабораторной работы №4 необходимо:

- Реализовать разработанную в рамках лабораторной работы №3 даталогическую модель в реляционной СУБД PostgreSQL.
- Заполнить созданные таблицы данными.
- Обеспечить целостность данных при помощи средств языка DDL.
- В рамках лабораторной работы должны быть разработаны скрипты для создания/удаления требуемых объектов базы данных, заполнения/удаления содержимого созданных таблиц.

Ход работы:

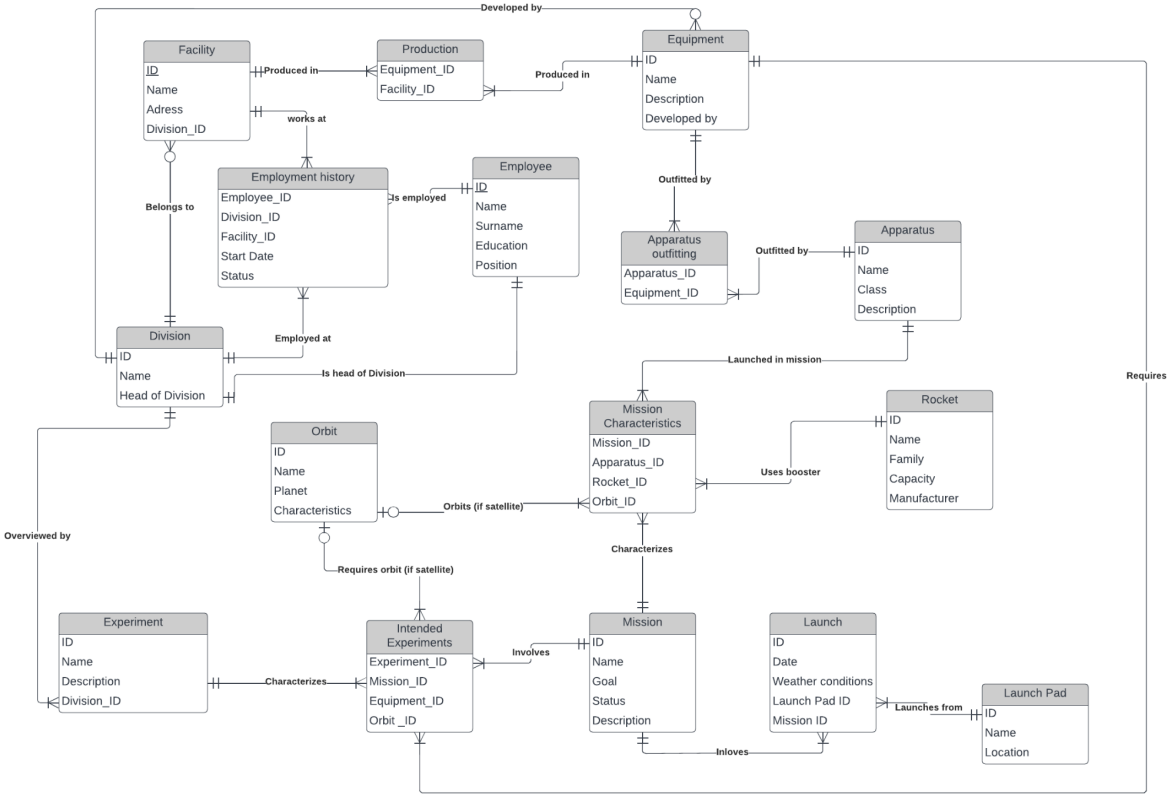
Предметная область: управление программой по исследованию Марса (на основе Mars Exploration Program, реализуемой NASA).

Описание предметной области: исследование Марса является одной из самых актуальных и передовых задач современной науки. В рамках программ по исследованию Марса задействованы колоссальные объемы материальных и людских ресурсов, задачи, решаемые в рамках этой области, крайне сложны и многочисленны, что делает использование баз данных в данной области необходимым.

Список основных сущностей:

1. Сотрудник (Employee)
2. Департамент (Division)
3. Миссия (Mission)
4. Оборудование (Equipment)
5. Производственный объект (Manufacturing facility)
6. Стартовая площадка (Launch pad)
7. Ракета-носитель (Rocket)
8. Исследовательский аппарат (Exploration apparatus)
9. Запуск (Launch)
10. Орбита (Orbit)
11. Научный эксперимент (Experiment)

ER-диаграмма



Даталогическая модель

Все поля таблиц должны быть not NULL
За исключением поля Orbit.orbit_id и Launch.weather_conditions

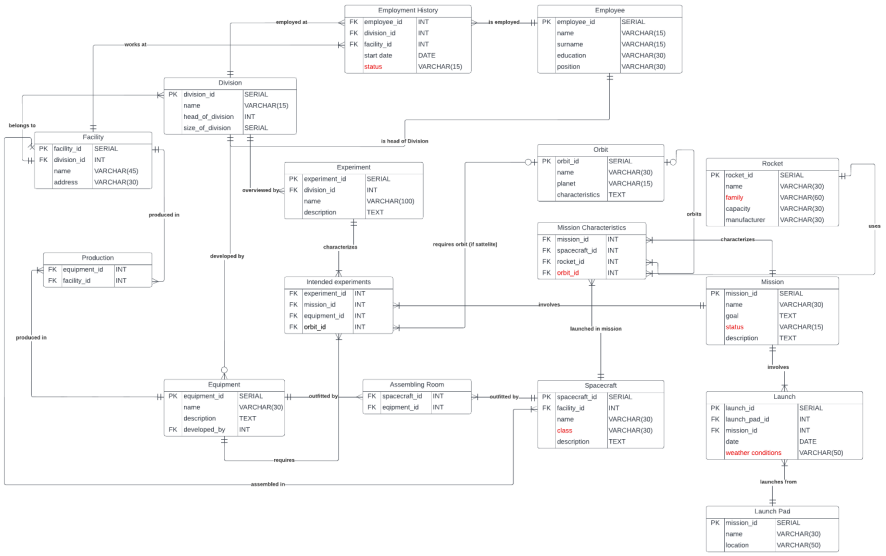
Ограничения уникальности

Ограничения:
1. Для аттрибута spacecraft_id в таблице Mission Characteristics предусмотрено ограничение, что в миссии участвует
2. Изначально значение аттрибута Production.status "prep"

Ссылочная целостность

Ссылочная целостность достигается за счет:
1.Наличию внешних ключей, помеченных в таблицах как FK(Foreign Key)
2. За счет триггеров
-EMPLOYMENT_PROCESS: данный триггер запускается при запросе, когда добавляется новый сотрудник (Employee), он меняет аттрибут status таблицы Employment History на "hired" и инкрементирует значение аттрибута size_of_division
-DISHARGE_PROCESS: данный триггер запускается при запросе, когда увольняется сотрудник (Employee), он меняет аттрибут status таблицы Employment History на "fired" и декрементирует значение аттрибута size_of_division
-PRODUCTION: данный триггер переводит аттрибут status в таблице Production в значение "in development"
-LAUNCH_PROCESS: данный триггер задает значение аттрибуту LaunchPad.status "go for launch" когда в таблицу Launch попадает запрос на добавление данных
-ABORT_LAUNCH: данный триггер запускается когда значение аттрибута Launch.weather_condition " thundestorm" OR "tornados" OR "hurricane" OR "severe conditions"

Пользовательская целостность:
Для достижения целостности, необходимой пользователю, на все аттрибуты, выделенные на диаграмме красным, накладываются ограничения, реализованные в виде CHECK'ов. Для отдельных полей они представляют из себя ограничение возможных значений конкретным диапазоном (family в таблице Rocket, class в таблице Spacecraft и т. д.), для orbit_id в тех таблицах, где этот аттрибут является внешним ключем, реализована проверка на то, что данный аппарат может иметь орбиту (если это не спутник, то значение может быть только NULL).



Скрипты создания, очистки и удаления таблиц

```
CREATE TABLE s295964.Employee(employee_id serial PRIMARY KEY,name VARCHAR(30) NOT NULL,surname VARCHAR(30) NOT NULL, education VARCHAR(30) NOT NULL,position VARCHAR(60) NOT NULL);
CREATE TABLE s295964.Division(division_id serial PRIMARY KEY, name VARCHAR(15) NOT NULL, head_of_division INT NOT NULL references s295964.Employee, size_of_division INT);
CREATE TABLE s295964.Facility(facility_id serial PRIMARY KEY,division_id serial references s295964.Division(division_id), name VARCHAR(45) NOT NULL, address VARCHAR(45) NOT NULL);
CREATE TABLE s295964.Experiment(experiment_id serial PRIMARY KEY, division_id SERIAL references s295964.Division(division_id),name VARCHAR(150) NOT NULL, description TEXT NOT NULL);
CREATE TABLE s295964.Equipment(equipment_id serial PRIMARY KEY, name VARCHAR(60) NOT NULL, description TEXT NOT NULL, developed_by serial references s295964.Division(division_id));
CREATE TABLE s295964.Production(equipment_id serial references s295964.Equipment(equipment_id),facility_id serial references s295964.Facility(facility_id));
CREATE TABLE s295964.EmploymentHistory(employee_id serial references s295964.Employee(employee_id),division_id serial references s295964.Division(division_id), facility_id serial references s295964.Facili
CREATE TABLE s295964.Mission(mission_id serial PRIMARY KEY, name VARCHAR(70) NOT NULL,goal TEXT NOT NULL,status VARCHAR(35) CHECK (status='LAUNCH PHASE' OR status= 'CRUISE PHASE' OR status= 'ENCOUTER PHA
CREATE TABLE s295964.Spacecraft(spacecraft_id serial PRIMARY KEY,facility_id INT NOT NULL references s295964.Facility(facility_id),name VARCHAR(30) NOT NULL, class VARCHAR(30) NOT NULL, description TEXT N
CREATE TABLE s295964.AssemblingRoom(spacecraft_id INT NOT NULL references s295964.Spacecraft(spacecraft_id), equipment_id INT NOT NULL references s295964.Equipment(equipment_id));
CREATE TABLE s295964.Orbit(orbit_id serial PRIMARY KEY, name VARCHAR(30) NOT NULL, planet VARCHAR(15) NOT NULL,characteristics TEXT NOT NULL);
CREATE TABLE s295964.Rocket(rocket_id serial PRIMARY KEY, name VARCHAR(30) NOT NULL, family VARCHAR(60) NOT NULL CHECK(family='Small-lift launch vehicle' OR family='Medium-lift launch vehicle' OR family='
CREATE TABLE s295964.LaunchPad(launch_pad_id serial PRIMARY KEY, name VARCHAR(30) NOT NULL, location VARCHAR(50) NOT NULL);
CREATE TABLE s295964.Launch(launch_id serial PRIMARY KEY,launch_pad_id INT NOT NULL references s295964.LaunchPad(launch_pad_id),mission_id INT NOT NULL references s295964.Mission(mission_id),date DATE NOT
CREATE FUNCTION hasOrbit(id integer) RETURNS boolean AS $$
    SELECT class = 'satellite' FROM Spacecraft WHERE spacecraft_id = hasOrbit.id;
$$ LANGUAGE SQL;
CREATE TABLE s295964.IntendedExperiments(experiment_id INT NOT NULL references s295964.Experiment, mission_id INT NOT NULL references s295964.Mission, equipment_id INT NOT NULL references s295964.Equipmen
CREATE TABLE s295964.MissionCharacteristics(mission_id INT NOT NULL references s295964.Mission, spacecraft_id INT NOT NULL references s295964.Spacecraft, rocket_id INT NOT NULL references s295964.Rocket,
```

```
TRUNCATE TABLE s295964.Division CASCADE;
TRUNCATE TABLE s295964.Facility CASCADE;
TRUNCATE TABLE s295964.Experiment CASCADE;
TRUNCATE TABLE s295964.Equipment CASCADE;
TRUNCATE TABLE s295964.Production CASCADE;
TRUNCATE TABLE s295964.Employee CASCADE;
TRUNCATE TABLE s295964.EmploymentHistory CASCADE;
TRUNCATE TABLE s295964.Mission CASCADE;
TRUNCATE TABLE s295964.Spacecraft CASCADE;
TRUNCATE TABLE s295964.AssemblingRoom CASCADE;
TRUNCATE TABLE s295964.Orbit CASCADE;
TRUNCATE TABLE s295964.Rocket CASCADE;
TRUNCATE TABLE s295964.LaunchPad CASCADE;
TRUNCATE TABLE s295964.Launch CASCADE;
TRUNCATE TABLE s295964.IntendedExperiments CASCADE;
TRUNCATE TABLE s295964.MissionCharacteristics CASCADE;
```

```
DROP TABLE IF EXISTS s295964.Division CASCADE;
DROP TABLE IF EXISTS s295964.Facility CASCADE;
DROP TABLE IF EXISTS s295964.Experiment CASCADE;
DROP TABLE IF EXISTS s295964.Equipment CASCADE;
DROP TABLE IF EXISTS s295964.Production CASCADE;
DROP TABLE IF EXISTS s295964.Employee CASCADE;
DROP TABLE IF EXISTS s295964.EmploymentHistory CASCADE;
DROP TABLE IF EXISTS s295964.Mission CASCADE;
DROP TABLE IF EXISTS s295964.Spacecraft CASCADE;
DROP TABLE IF EXISTS s295964.AssemblingRoom CASCADE;
DROP TABLE IF EXISTS s295964.Orbit CASCADE;
DROP TABLE IF EXISTS s295964.Rocket CASCADE;
DROP TABLE IF EXISTS s295964.LaunchPad CASCADE;
DROP TABLE IF EXISTS s295964.Launch CASCADE;
DROP TABLE IF EXISTS s295964.IntendedExperiments CASCADE;
DROP TABLE IF EXISTS s295964.MissionCharacteristics CASCADE;
```

Код на Java, используемый для генерации скрипта заполнения таблиц большими объемами данных

```
    FileWriter writer = new FileWriter( fileName: "lab4_fill_tables_script_s295964.txt");
    Random rnd = new Random();
    writer.write(header(schema, employeeSignature));
    for (int i = 0; i < 10000; i++) {...}
    writer.write(header(schema, divisionSignature));
    for (int i = 0; i < divisions.length; i++) {...}
    writer.write(header(schema, facilitySignature));
    for (int i = 0; i < facilities.length; i++) {...}
    writer.write(header(schema, experimentSignature));
    for (int i = 0; i < surfaceExperiments.length; i++) {...}
    for (int i = 0; i < orbitExperiments.length; i++) {...}
    writer.write(header(schema, equipmentSignature));
    for (int i = 0; i < equipment.length * 100; i++) {...}
    writer.write(header(schema, productionSignature));
    for (int i = 0; i < 10000; i++) {...}
    writer.write(header(schema, employmentHistorySignature));
    for (int i = 0; i < 10000; i++) {...}
    for (int i = 0; i < 5000; i++) {...}
    writer.write(header(schema,missionSignature));
    for (int i = 0; i < 10000 * missions.length; i++) {...}
    writer.write(header(schema, spacecraftSignature));
    for (int i = 0; i < spacecrafts.length; i++) {...}
    writer.write(header(schema, assemblingRoomSignature));
    for (int i = 0; i < 100; i++) {...}
    writer.write(header(schema, orbitSignature));
    for (int i = 0; i < orbits.length * 10; i++) {...}
    writer.write(header(schema, rocketSignature));
    for (int i = 0; i < rockets.length; i++) {...}
    writer.write(header(schema,launchPadSignature));
    for (int i = 0; i < launchPads.length; i++) {...}
    writer.write(header(schema,launchSignature));
    for (int i = 0; i < 100; i++) {...}
    writer.write(header(schema, intendedExperimentsSignature));
    for (int i = 0; i < 10000; i++) {...}
    writer.write(header(schema, missionCharacteristicsSignature));
    for (int i = 0; i < 10000; i++) {...}
    writer.close();
}
```

```java
for (int i = 0; i < 10000; i++) {
    writer.write(String.format(employeeValue,
            names[rnd.nextInt(names.length)],
            surnames[rnd.nextInt(surnames.length)],
            educations[rnd.nextInt(educations.length)],
            positions[rnd.nextInt(positions.length)]));
    if(i == 9999){
        writer.write( str: ";\n");
    }else{
        writer.write( c: ',');
    }
}
```

Выводы

При выполнении лабораторной работы мы реализовали разработанную для нашей предметной области ранее даталогическую модель в СУБД PostgreSQL. Для определения таблиц были написаны скрипты, позволяющие быстро отлаживать выполнение и исправлять ошибки. Кроме того, при определении таблиц, в соответствии с даталогической моделью, реализованы различные ограничения целостности. Для заполнения созданных таблиц крупными объемами данных, необходимых при выполнении следующей лабораторной, написана специальная программа, генерирующая файл со скриптом.