UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
Departamento de Ciências de Computação - SCC
SCC0640 - Bases de Dados

Projeto de Base de Dados - Exploração de Planetas

Sistema de Missões em Exploração da Galáxia

Professor(a):
Elaine Parros Machado de Sousa

Alunos:
Aleixo Damas Neto - nUSP 10310975
Antônio Pedro Amado de Menezes Medrado - nUSP 10748389
Lucas Henrique Sant'Anna - nUSP 10748521
Mateus Santos Messias - nUSP 12548000

São Carlos
2023

Parte I: Descrição do Sistema

1. Descrição do Problema

1.1. Introdução

O projeto idealizado pelo grupo descreve um sistema que ilustra um cenário fictício, onde a exploração humana agora tem um escopo galáctico. Nesse cenário, o sistema assume a existência de diversos governos, que se utilizam dos serviços de distintas corporações interplanetárias para expandir sua influência sobre a galáxia. As corporações realizam diversos contratos gerados pelos governos, executando missões de naturezas variadas. A base também inclui informações dos sistemas estelares - e os corpos celestes pertencentes a eles - relevantes para as missões realizadas.

O usuário final do sistema seria um fiscal ou administrador de um órgão intergovernamental (como a ONU), tendo foco em analisar não as questões geopolíticas, mas os dados dos contratos realizados e as missões envolvidas. Esse administrador controla as informações sobre os sistemas estelares e seu conteúdo, além dos governos e corporações que fazem parte desse programa de exploração. Os governos e corporações também são usuários, podendo inserir informações sobre suas atividades.

É claro que tal cenário envolveria um número de acordos entre governos para colonizar, explorar e pesquisar os corpos celestes, contudo essas informações são dispensáveis ao foco da aplicação, que busca administrar a relação comercial entre governos e corporações interplanetárias.

1.2. Visão Geral de Problema

De forma mais detalhada, cada **Governo** cadastrado na aplicação é identificado apenas pelo seu *Nome* (*string*) contando também a informação de *Tipo de Governo* (*string*), que descreve o sistema político ou modelo de governança implementado. Os **Contratos** são criados pelos **Governos**, que podem ser firmados com as **Corporações** para atuar em um **Corpo Planetário**.

As **Corporações** possuem um *Nome* (*string*) e são identificadas pelo seu *Código Corporativo* (*string*) que é um cadastro único com significado externo, similar a um CNPJ.

Os **Contratos** são identificados pelo seu *Título de Contrato* (*string*) e devem armazenar as informações de *Data Início* (data), *Data Limite* (data), *Encerrado* (booleano) e *Expirado* (booleano que é atualizado de acordo com a *Data Limite*). Quando uma **Corporação** começa a realizar um contrato, **Missões** são geradas para diferentes **Corpos Planetários**, em que cada uma pode ser identificada conjuntamente pelo seu *Título da missão* (*string*) e *Data de Limite* (data) apenas. Cada **Missão** deve conter os dados de *Data Limite* (data), *Status* (*string* que pode assumir opções entre "A Iniciar", "Ativa", "Finalizada" ou "Cancelada"), *Tipo de Missão* (*string*) e Expirada (booleano que é atualizado utilizando a *Data de Limite*).

As **Corporações** que firmam **Missões** possuem diferentes **Naves**. As **Naves** devem ser de posse de uma **Corporação**, e apenas uma, podendo ser identificadas pelo *Nome da Embarcação* (*string*) ou pelo seu *Nome de Batismo* (*string*). Devem ser armazenados também o *Modelo* (*string*) e *Ano* (numérico). Cada **Nave** pode, então, ser delegada para diferentes **Missões**, sendo que cada missão pode ter várias **Naves** encarregadas por ela. Uma **Nave** só pode atuar em uma **Missão** se ela pertencer à **Corporação** que firmou o **Contrato** da **Missão.**

Diferentes **Corpos Planetários** possuem também diferentes *Quantidades* (numérico) de diferentes **Recursos**, podendo um **Recurso** se repetir em diferentes **Corpos Planetários** . Um recurso existente já descoberto no universo pode ser identificado unicamente pelo seu *Nome* (*string*) e tem as informações relevantes de *Valor* (numérico) por quantidade extraída e *Tipo* (*string*) (se é um recurso mineral, vegetal, etc).

Uma **Missão** no sistema deve ser, necessária mas não unicamente, de quatro tipos distintos:

- Cada **Missão** de **Construção de Estação** é focada em construir as diferentes instalações nesse novo ambiente para o **Governo**, devendo armazenar então o *Tipo de Estação* (*string*).
- Cada **Missão** de **Exploração e Reconhecimento** busca dar os primeiros passos para a exploração do novo **Corpo Planetário**. Ela deverá armazenar então os *Tipos de Dados Coletados* (*string*) por uma **Missão** dessa categoria (se a missão está coletando dados atmosféricos, geológicos, etc).
- Cada **Missão** de **Pesquisa Científica** deverá armazenar quais *Pesquisas* (*string*) estão sendo feitas, listando as distintas pesquisas em realização.

- Cada **Missão** de **Extração de Recurso** deverá armazenar a sua *Escala de Operação* (*string* com valores a serem escolhidos entre “Pequena”, “Médio”, “Grande” e “Continental”) que prevê o tamanho daquele empreendimento de extração. Uma **Missão** de **Extração de Recurso** será responsável por extrair um **Recurso** do corpo planetário, devendo-se armazenar as informações de *Quantidade* (numérico) e *Impacto Ambiental* (*string* com valores a serem escolhidos entre “Baixíssimo”, “Baixo”, “Alto” ou “Altíssimo”).

Os **Corpos Planetários**, por sua vez, são um tipo de **Corpo Celeste**. Um **Corpo Celeste** qualquer é único em seu *Nome* (*string*), e deve armazenar as suas informações de *Raio Médio* (numérico), *Massa* (numérico) e *Tipo* (*string* podendo assumir os valores de “Planetário” se for um **Corpo Planetário** ou “Estrela”). Um **Corpo Celeste** deve ser obrigatoriamente um **Corpos Planetários** ou uma **Estrela**.

- Os **Corpos Planetários**, especificamente, devem armazenar as suas informações de tempo de *Período Orbital* (numérico) e de *Rotação* (numérico), se é *Respirável* (booleano) e o seu *Tipo* (*string* podendo assumir os valores de “Asteróide”, “Planeta” ou “Satélite”).
- **Estrelas** devem armazenar apenas qual a sua *Categoria* (caractere que pode assumir um valor de O, B, A, F, G, K ou M de acordo com a sua temperatura).

As **Estrelas** pertencem necessariamente a um **Sistema**, que por sua vez tem um *Nome* (*string*) único, e também uma localização composta pelas *Coordenada X* (numérico) e *Coordenada Y* (numérico) única no universo.

Corpos Planetários são classificados em 3 tipos:

- Cada **Asteroide** deve orbitar uma **Estrela** e armazenar o valor médio da *Distância da Estrela* (numérico). Eles não possuem atmosfera respirável.
- Cada **Planeta** deve orbitar uma **Estrela** e armazenar o valor médio da *Distância da Estrela* (numérico). Cada um também deve armazenar sua *Categoria* (*string* que descreve o planeta como gasoso, rochoso, planeta anão, etc).
- Cada **Satélite** deve orbitar um **Planeta** e armazenar o valor médio da *Distância do Planeta* (numérico). Cada um também deve armazenar sua *Categoria* (*string* que descreve o planeta como gasoso, rochoso, planeta anão, etc).

O Diagrama Entidade-Relacionamento desse sistema encontra-se no fim do documento.

2. Principais Operações

2.1. Administrador

- Inserir, editar e remover **Sistemas** estelares;
- Inserir, editar e remover **Corpos Celestes**, em todas as suas formas específicas;
- Inserir, editar e remover **Recursos**;
- Associar **Recursos** a **Corpos Celestes**;
- Inserir, editar e remover **Governos** e **Corporações**;
- Consultar **Recursos**, com filtros relacionados a **Corpos Planetários**, **Governos**, **Corporação**, *Quantidade* (extraída), *Nome*, *Impacto Ambiental*, *Tipo*, *Valor*, *Quantidade* (existente num **Corpo Planetário**) e/ou **Sistema;**
- Consultar **Governos**, com filtros relacionados a **Corporações**, **Recursos**, **Corpos Planetários**, **Contratos** (quantidade de contratos) e/ou **Sistema**;
- Consultar **Corporações**, com filtros relacionados **Governos**, **Missões**, **Contratos** (quantidade de contratos), *Tipo de Missão*, **Corpo Planetário** e/ou **Sistema**;
- Consultar **Sistemas**, com filtros relacionados a **Corpos Planetários**, *Categorias* de **Estrelas**, Posição do Sistema (*Coordenadas X e Y*), e/ou **Missões** realizadas nos mesmos.

2.2. Governo

- Inserir, editar e remover **Contratos**;
- Consultar **Corporações** com filtros relacionados aos *Tipos de Missão* que elas já realizaram;
- Consultar **Sistemas** e **Corpos Planetários**, com filtros relacionados aos **Recursos** que possuem.

2.3. Corporação

- Inserir, editar e remover **Naves**;

- Consultar **Sistemas** e **Corpos Planetários**, com filtros relacionados aos **Recursos** que possuem.
- Consultar **Contratos**, com filtros relacionados a seu **Governo** criador;
- Realizar e cancelar **Contratos**, criando, editando e removendo **Missões** especificadas para seus contratos;
- Atribuir **Naves** a **Missões**.

3. Restrições de Integridade e Observações

3.1. O ciclo Nave - Corporação - Missão

Há um ciclo entre as entidades **Nave**, **Corporação** e **Missão**, que pode ocasionar a situação em que uma **Nave** possuída por uma **Corporação** está participando de uma **Missão** que não é realizada pela mesma **Corporação**. Essa é uma situação errônea que deve ser conferida em nível de aplicação, não permitindo que uma **Corporação** tenha **Missões** nas quais participam **Naves** de outras **Corporações**.

3.2. O ciclo Recurso - Extração de Recursos - Corpo Planetário

Há um ciclo entre as entidades **Recurso**, **Extração de Recursos** e **Corpo Planetário**, que pode ocasionar a situação em que uma missão do tipo **Extração de Recursos**, explorando um dado **Corpo Planetário**, extraia **Recursos** que não estão presentes no mesmo. Essa é uma situação errônea que deve ser conferida em nível de aplicação, não permitindo que uma missão de tipo **Extração de Recursos** extraia **Recursos** que não estão presentes no **Corpo Planetário** em que a missão ocorre.

3.3. Sobre a participação de uma nave em missões

É necessário garantir que uma mesma **Nave** não esteja participando de múltiplas **Missões** ao mesmo tempo - ou seja, ela só pode participar de uma próxima **Missão** se a *Data Limite* da anterior não ultrapassar a *Data Início* da mesma.

3.4. Sobre a quantidade de recursos extraída em missões

A *Quantidade* de um **Recurso** extraída numa missão do tipo **Extração de Recursos** não pode ser maior do que a *Quantidade* daquele **Recurso** presente no **Corpo Planetário** no qual a missão ocorre.

3.5. Sobre a modelagem de Corpos Celestes

A maneira com o qual **Corpos Celestes** foram modelados reflete uma necessidade de evitar problemas de consistência no caso de uma relação direta entre **Corpos Celestes** e **Sistemas** - em seu lugar separando-os por especialização em **Estrelas** pertencentes aos **Sistemas** e **Corpos Planetários** que as orbitam.

3.6. Sobre as distâncias de órbita e posições de Sistemas

Os atributos *Distância do Planeta* em **Satélites** e *Distância da Estrela* em **Planetas** e **Asteroides** devem ser maiores que zero ao se cadastrar um desses corpos em órbita - afinal, eles não podem estar ocupando o mesmo espaço que o corpo que orbitam. Isso deve ser feito em nível de aplicação.

Similarmente, não faz sentido que dois **Sistemas** se encontrem na mesma posição no espaço. Para garantir que isso não ocorra, decidiu-se por tornar as *Coordenadas X* e *Y* a chave primária (composta) de **Sistema**, ao invés de seu *Nome*.

3.7. Sobre um Corpo Planetário ser respirável ou não

O atributo *Respirável* de um **Corpo Planetário** especificado como **Asteroide** deve ser falso (zero); afinal, asteroides não podem possuir atmosferas. Isso é um cuidado que deve ser garantido a nível de aplicação.

3.8. Sobre qual Estrela orbita um Planeta ou Satélite

Devido à natureza da simplificação com que são modelados sistemas estelares na presente aplicação, um dado **Sistema** pode conter mais de uma **Estrela** (sem informações sobre sua configuração orbital, como um sistema binário ou outro), e cada uma dessas

Estrelas pode conter diferentes **Planetas** e **Asteroides** orbitando-a, mas não orbitando a(s) outra(s) **Estrela(s)** presente(s) no **Sistema**.

Tal situação pode causar confusões em buscas sobre as informações completas sobre os **Corpos Planetários** presentes num **Sistema** estelar. Para garantir que as informações em sua totalidade sejam retornadas, cuidados devem ser tomados para associar a busca de **Corpos Planetários** não apenas à **Estrela** que orbitam, mas ao **Sistema** que contém essa **Estrela**.

3.9. Sobre os atributos de expiração

Por serem atributos simples e baratos de calcular a partir dos outros atributos das entidades relacionadas, os atributos *Expirado* em **Contrato** e *Expirada* em **Missão** serão calculados a cada consulta em nível de aplicação. Eles poderiam ser incluídos nas tabelas de suas respectivas entidades para evitar esse cálculo repetido - que torna uma consulta um pouco mais cara. Contudo, considerou-se que o benefício trazido por essa abordagem seria maior, garantindo menos espaço ocupado por atributos que podem ser calculados facilmente, e que o valor do atributo estará sempre correto e atualizado.

Parte 2: Projeto Lógico

4. Discussões sobre os mapeamentos propostos para o Modelo Relacional

A seguir estão listados as justificativas para adotar as opções para o mapeamento das diferentes do elementos do esquema criando no Modelo Entidade-Relacionamento para o Modelo Relacional. Este encontra-se no fim desse documento, após o Diagrama Entidade-Relacionamento.

4.1. Relacionamentos de cardinalidade 1:N

No esquema do MER há seis relacionamentos de cardinalidade 1:N - porém todos eles têm uma entidade, aquela que pode participar de N relacionamentos, com participação total no relacionamento. Esses relacionamentos são: Relacionamento **Pertence a** entre **Sistema** e **Estrela**, relacionamento **Orbita** entre **Estrela** e **Asteroide**, relacionamento **Orbita** entre **Estrela** e **Planeta**, relacionamento **Orbita** entre **Planeta** e **Satélite**, relacionamento **Possui** entre **Corporação** e **Nave** e o relacionamento **Gera** entre **Governo** e **Contrato**. Neste último **Contrato** é Entidade Fraca do relacionamento.

Dado o tipo de relacionamento que todos possuem, esses relacionamentos foram mapeados seguindo a forma como se mapeia participação total (exceto no caso do relacionamento **Gera**) - isto é, a entidade fraca terá uma chave estrangeira apontando para a entidade forte, sendo a chave estrangeira *not null*.

4.2. A agregação Missão e suas especializações

O relacionamento **Realiza**, entre **Corporação**, **Contrato** e **Planeta**, exige uma avaliação complexa por conter 3 pontos de atenção: um relacionamento ternário de cardinalidade 1:N:M, uma entidade agregada gerada por esse relacionamento, e um relacionamento de generalização para essa entidade agregada.

Decidiu-se por adotar um mapeamento condizente com o contexto de uma agregação na qual cada instância do CR gera múltiplas entidades agregadas - afinal, um **Contrato** aceito por uma **Corporação** pode gerar múltiplas **Missões**. Visto que elas podem ser identificadas por seus próprios atributos - *Tipo de Missão* e *Data Início* -, optou-se por uma relação cuja

chave primária é composta apenas por esses atributos-chave de **Missão**. Os demais atributos-chave das entidades geradoras foram deixados apenas como *not null*, mas não são parte da chave composta nessa relação.

Para respeitar o contexto de um relacionamento de generalização tipo O, adaptamos esse mapeamento para seguir o Procedimento Padrão 9 estudado: há uma tabela para a entidade geral (**Missão**), contendo apenas a chave composta de missão e o atributo discriminante *TipoMissão* - que faz parte da chave composta para essa relação, seguindo o contexto de uma generalização tipo O. As entidades específicas para cada tipo de missão contêm todos os demais atributos, que não foram listados na relação da entidade geral, já que esta serve primariamente para casos em que é necessário buscar rapidamente qual ou quais são os tipos de uma **Missão**. Essa é a principal vantagem desse mapeamento, buscando garantir participação total e o contexto de *overlap* da generalização. O fato de ser possível identificar uma **Missão** por seus atributos também evita a replicação das chaves das entidades específicas na tabela referente à entidade geral, e a escolha por ter todos os atributos para cada tipo de missão em suas respectivas tabelas evita maiores custos de junção quando se quer informações para uma ou mais missões de um tipo previamente conhecido.

Contudo, alguns pontos devem ser garantidos em aplicação: é necessário evitar que sejam criadas entradas na tabela geral de **Missões** sem que existam seus devidos correspondentes nas tabelas específicas.

4.3. As especializações para CorpoCeleste e CorpoPlanetário

Na especialização das entidades **Corpo Celeste** e **Corpo Planetário** foi feita a escolha de utilizar um dos procedimentos padrões mostrados em aula, o Procedimento Padrão 8. Onde cada Conjunto Entidade Específicos é representado em sua própria relação e há uma tabela geral que tem apenas os atributos que fazem parte da chave principal e o atributo critério. Assim temos a relação **CorpoCeleste**, com a chave principal *Nome* e o atributo *Tipo* (*not null* pois precisamos saber qual será sua especificação), e as relações **CorpoPlanetário** e **Estrela**, que nascem de sua especificação. **Estrela** tem como chave primária *Nome*, apontando para **CorpoCeleste** por definição, e os outros atributos como *RaioMédio*, *Massa*, *Categoria*, *CoordenadaX* e *CoordenadaY*. Os últimos dois sendo chave estrangeira composta. A relação **CorpoPlanetário** não tem os mesmos atributos por ser ele também um fruto de uma Conjunto Entidade Generalização. Assim, os atributos *RaioMédio*, *Massa*,

PeríodoOrbital, *Rotação* e *Respirável* não constarão em sua relação, apenas nas relações **Asteroide**, **Planeta** e **Satélite**.

Apesar de ser o procedimento padrão para realizar o mapeamento nos dois casos de especialização total exclusiva, a solução não garante a exclusividade total, devendo esse ponto ser garantido em aplicação. Além disso, pode gerar vulnerabilidade de consistência do *Tipo* de **CorpoCeleste**, que seus subtipos relacionam-se entre si. Nesse caso, também é necessário tratar possíveis inconsistências de tipo na aplicação. Contudo, como já dito, o modo escolhido é o mais adequado para especialização total e exclusão mútua.

4.4. A entidade Recurso e seus relacionamentos N:N

A entidade **Recurso** participa de dois relacionamentos no sistema: um com **Corpo Planetário** e um com o tipo de missão **Extração de Recursos**. Nos dois casos, como não é necessário garantir participação total, optou-se por mapear cada um desses relacionamentos N:N criando uma nova tabela, cuja chave é composta pelas chaves de **Recurso** e da outra entidade envolvida. Cada tabela também contém os atributos contidos no relacionamento listado, como *Quantidade* e *Impacto Ambiental* para o relacionamento entre **Extração de Recursos** e **Recurso**.

4.5. Atributos multivalorados

Os atributos multivalorados ocorrem em duas especializações de **Missão**: *Tipos de Dados Coletados* de **Exploração e Reconhecimento** e *Pesquisas* de **Pesquisa Científica**. Os dois casos foram mapeados utilizando a solução padrão, criando para os atributos multivalorados as tabelas **DadosColetados** e **PesqCientifica**, respectivamente. As tabelas podem possuir várias instâncias para uma mesma **Missão**, sendo única a composição das chaves primárias da **Missão** e o Atributo específico para cada caso, já que não se sabe a quantidade de entradas envolvida para cada caso. O custo dessa implementação é a criação de uma nova tabela para essa função, o que dificulta buscas que exijam a junção de tabelas.

A outra forma de mapeamento, que consiste em decompor cada atributo multivalorado em diversos atributos da relação mapeada, não faria muito sentido nos dois contextos aqui expressos tendo em vista que não temos o especifico número de valores que teríamos em atributo multivalorado.

4.6. Atributos derivados

Como mencionado na Parte 1, os atributos derivados *Expirada* em **Contrato** e **Missão** podem ser calculados em nível de aplicação com pouco custo. Assim, optou-se por não incluí-los nas tabelas do Modelo Relacional mapeado.

5. Mudanças na Parte 1 para a Segunda Entrega

5.1. Correções ao Modelo Entidade-Relacionamento proposto

Como apontado na correção da primeira entrega, a modelagem proposta para as entidades relacionadas a sistemas planetários, como **Planeta**, **Estrela**, **Asteroide** etc., não estava de acordo com os padrões de um MER. Em vista disso, foi transformado o relacionamento de um **Corpo Celeste** com a entidade que ele orbita (ou o sistema do qual faz parte). Ao invés de descrever esses corpos como entidades fracas - o que não pode ocorrer, visto que eles fazem parte de relacionamentos de especialização -, suas órbitas foram modeladas como relacionamentos com participação total pela entidade orbitante.

Havia uma incoerência entre o relatório e o diagrama no tocante à participação de **Corpo Celeste** em seu relacionamento de especialização. Isso foi corrigido - agora se trata positivamente de uma participação total.

Também em se tratando de participação total em relacionamentos, no relacionamento **Corporação** possui **Nave**, foi modelada participação total por parte de nave, visando excluir a ocorrência de casos de **Naves** que não são parte da frota de nenhuma **Corporação**.

Notou-se uma incoerência com a semântica na representação do relacionamento entre **Corporação**, **Contrato** e **Planeta** (gerador de **Missão**) - um **Contrato** para um **Planeta** não pode ser realizado por mais de uma **Corporação**. Visando corrigir esse erro, foi alterada a cardinalidade do relacionamento para apenas 1 com **Corporação**.

Referências a chaves secundárias erroneamente incluídas no DER foram removidas. Agora elas são mencionadas no Modelo Relacional apenas.

O modelo também foi tratado a fim de remover ocorrências de ID sintético em chaves. Essas foram substituídas por elementos descritivos, como *Título*, *Data*, etc. Um exemplo mantido da primeira entrega é *Código Corporativo*, descrito como uma espécie de CNPJ - ou seja, tendo um significado externo.

5.2. Revisões ao relatório

Visando corrigir a apresentação das informações de nosso sistema, unimos a antiga seção 3 à seção 1.2. Agora, há uma seção única descrevendo todo o sistema, suas entidades, relacionamentos e respectivos atributos.

Parte 3: Implantação da base de dados e implementação do Sistema.

6. Aplicação

Para implementar o projeto, optamos por fazer uma aplicação em *Python* 3.10.5 de interface com o usuário simples por meio do terminal (**Figura 1**), por onde são permitidas inserções e consultas pré-estabelecidas. Já para o banco de dados de dados, optamos por utilizar o *PostgreSQL* 16. Foi utilizada a biblioteca *psycopg2* para a comunicação entre o código da aplicação e o servidor Postgres. Essa extensão deve ser instalada antes que a aplicação possa ser executada.

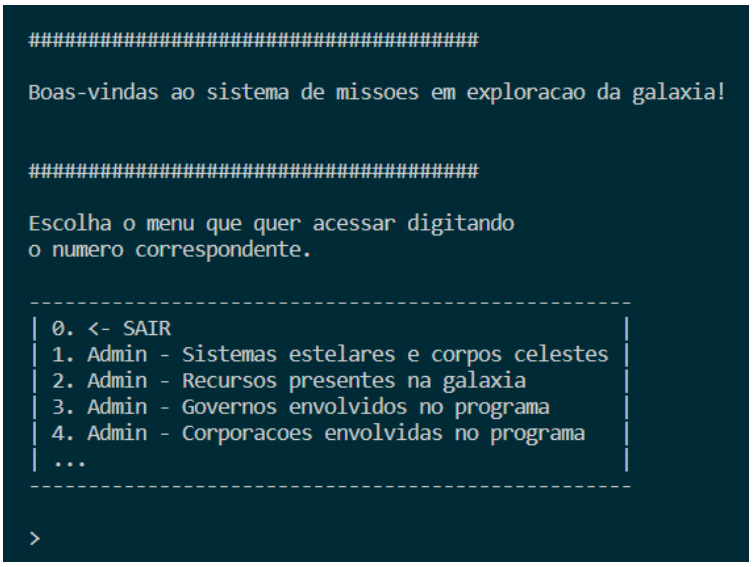


Figura 1: "Menu Principal" da aplicação em CLI.

O projeto SQL está dividido no seguintes arquivos de scripts:

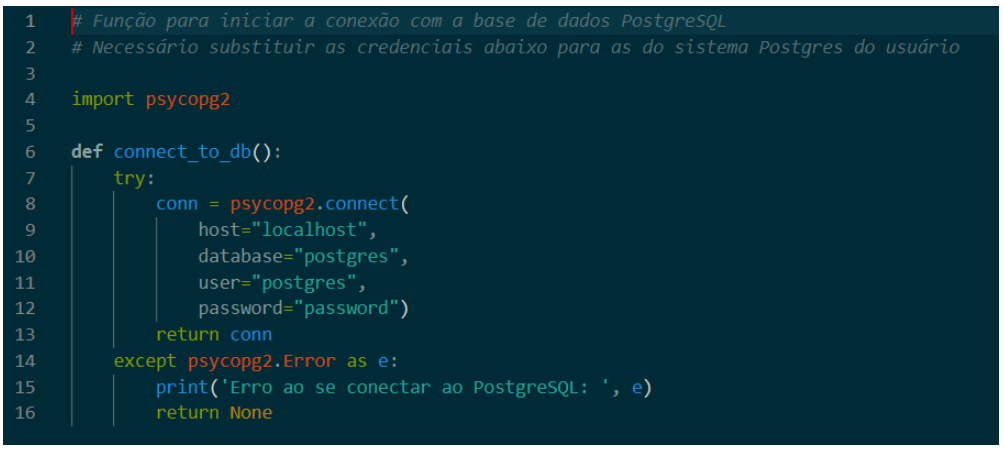
- `esquema.pgsql`, que contém as criações das tabelas em SQL;
- `dados.pgsql`, que contém os dados iniciais a serem inseridos no banco de dados;
- `consultas.pgsql`, que contém as consultas pré-estabelecidas a serem feitas pela aplicação quando solicitadas pelo usuário;
- `drop_all.pgsql`, que contém comandos de `drop table` para todas as tabelas para limpar o banco de dados.

Existem também os arquivos da aplicação, que para ser iniciada deve-se executar o *script* do arquivo `main.py`.

- `main.py`, arquivo que deve ser executado para iniciar a aplicação;
- `application.py`, que contém classes e métodos para o funcionamento da aplicação;

- `aux_functions.py`, que contém funções auxiliares às funções da aplicação, como validações e funções de impressão;
- `interface.py`, que agrega muitas das strings que compõem a interface em terminal da aplicação, separadas em sua própria classe para fins de organização;
- `database.py`, que contém o método de conexão à base de dados e as credenciais.

Após instalar a extensão requerida, o usuário deve acessar o arquivo `database.py` e preencher as credenciais de conexão com o servidor local Postgres, como exemplificado na imagem a seguir (**Figura 2**). Para fins de apresentação deste projeto, também foi adicionada uma funcionalidade no arquivo `main.py` para limpar o conteúdo do banco de dados, restaurando-o a um estado inicial preenchido com os dados incluídos em `dados.pgsql`.



```python
# Função para iniciar a conexão com a base de dados PostgreSQL
# Necessário substituir as credenciais abaixo para as do sistema Postgres do usuário

import psycopg2

def connect_to_db():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="postgres",
            user="postgres",
            password="password")
        return conn
    except psycopg2.Error as e:
        print('Erro ao se conectar ao PostgreSQL: ', e)
        return None
```

Figura 2: Informações que devem ser preenchidas para conexão com servidor PostgreSQL.

Tomou-se o cuidado, por meio de commits e o isolamento das execuções do cursor, em respeitar as transações ao longo da execução da aplicação.

6.1. Exemplo de cadastro de dados

Para a funcionalidade de cadastro, temos como exemplos a inserção de um novo **Sistema** como também de uma nova **Estrela** - estas ações podem ser escolhidas separadamente pelo menu de sistemas estelares e corpos celestes. Destes, o exemplo de cadastro de **Sistema** está ilustrado a seguir. É possível observar o uso dos commits, assim como validação do input do usuário, para estar de acordo com os padrões necessários para o banco.

```
87
88      def insert_sistema(self):
89          insert = input(self.i.admin_ins_sistema)
90          try:
91              x, y, nome = insert.split(',')
92              x, y = float(x.strip()), float(y.strip())
93          except ValueError:
94              print('\nErro no input. Por favor, tente novamente.\n')
95              _ = input('Pressione Enter para retornar.')
96              return
97          if not aux.validate_numeric(x, 8, 2) or not aux.validate_numeric(y, 8, 2):
98              print('\nCoordenadas excedem precisao (8) ou escala (2) permitida. Tente novamente.\n')
99              _ = input('Pressione Enter para retornar.')
100             return
101         nome = nome.strip()
102         with conn.cursor() as cursor:
103             cursor.execute('''INSERT INTO SISTEMA (COORDENADA_X, COORDENADA_Y, NOME)
104                               VALUES (%s, %s, %s);''', (x, y, nome))
105             conn.commit()
106
107         print('\nSistema inserido com sucesso!\n')
108         _ = input('Pressione Enter para retornar.')
```

Figura 3: Exemplo de cadastro de sistemas no banco de dados

6.2. Exemplo de consulta ao banco

Para a funcionalidade de consulta, temos similarmente uma busca por informações de **Sistemas** ou **Estrelas** no banco de dados. Ela é feita com base em dois possíveis campos: *Nome* (da **Estrela** ou **Sistema**) ou *Coordenadas X e Y* (do **Sistema** na galáxia e de **Estrelas** pertencentes a um **Sistema** com essas coordenadas). Em ambas as opções, é feita a consulta por resultados nas duas tabelas, imprimindo-se resultados de ambos - ou um aviso caso nada tenha sido encontrado. Para buscas por nomes de **Estrelas**, para fins de conveniência ao usuário da aplicação, é feita uma consulta sobre o **Sistema** ao qual a **Estrela** pertence, cujas informações são incluídas no resultado da consulta.

Similarmente ao exemplo de cadastro, observa-se tratamento com erro de inputs do usuário - em particular, se este errou na digitação das coordenadas para pesquisa.

```python
def search_sistema_estrela(self):
    search = input(self.i.admin_bus_sistema)
    if ',' in search: # Busca por coordenadas
        try:
            x, y = search.split(',')
            x, y = float(x.strip()), float(y.strip())
        except ValueError:
            print('\nErro no input. Por favor, tente novamente.\n')
            _ = input('Pressione Enter para retornar.')
            return
        with conn.cursor() as cursor:
            cursor.execute('''SELECT * FROM SISTEMA WHERE COORDENADA_X = %s AND COORDENADA_Y = %s;''', (x, y))
            sistemas = cursor.fetchall()
        with conn.cursor() as cursor:
            cursor.execute('''SELECT * FROM ESTRELA WHERE COORDENADA_X = %s AND COORDENADA_Y = %s;''', (x, y))
            estrelas = cursor.fetchall()
    else: # Busca por nome
        nome = search.strip()
        with conn.cursor() as cursor:
            cursor.execute('''SELECT * FROM SISTEMA WHERE NOME = %s;''', (nome,))
            sistemas = cursor.fetchall()
        if len(sistemas) > 0:
            with conn.cursor() as cursor:
                cursor.execute('''SELECT * FROM ESTRELA WHERE COORDENADA_X = %s AND COORDENADA_Y = %s;''',
                               (sistemas[0][0], sistemas[0][1]))
                estrelas = cursor.fetchall()
        else:
            with conn.cursor() as cursor:
                cursor.execute('''SELECT * FROM ESTRELA WHERE NOME = %s;''', (nome,))
                estrelas = cursor.fetchall()
    if len(estrelas) > 0:
        with conn.cursor() as cursor:
            cursor.execute('''SELECT * FROM SISTEMA WHERE COORDENADA_X = %s
                           AND COORDENADA_Y = %s;''', (estrelas[0][4], estrelas[0][5]))
            sistemas = cursor.fetchall()
    # Imprimir resultados
    print('\n[SISTEMAS ENCONTRADOS]\n')
    if len(sistemas) > 0:
        for sistema in sistemas: aux.print_sistema(sistema)
    else: print('Nao foi encontrado sistema com esses dados.\n')
    print('\n[ESTRELAS ENCONTRADAS]\n')
    if len(estrelas) > 0:
        for estrela in estrelas: aux.print_estrela(estrela, sistemas[0])
    else: print('Nao foi encontrada estrela com esses dados.\n')
    _ = input('Pressione Enter para retornar.')
```

Figura 4: Exemplo de consulta de sistemas e estrelas no banco de dados

7. Consultas

Conforme requisitado na especificação do projeto, foram construídas cinco consultas de maior complexidade, buscando refletir os casos de uso da presente aplicação. Essas foram:

7.1. Consultando as Naves que não estão participando de Missões num determinado momento

Dentro do *'mundo exemplo'* dentro do projeto (instâncias e esquemas) definimos com momento atual a data *01/01/2567* Utilizamos, então, essa data para realizar a consulta das

Naves que não estão participando de nenhuma **Missão** neste exato (ou uma pequena alteração na condição permitiria consultar para qualquer momento específico). Foi realizado junção de tabelas Nave, Participa e das cinco tabelas de missão (a Generalizada e as Especializadas) por meio de LEFT JOIN, selecionando todas as naves que estão associadas a Missões com intervalo de atividade onde a data atual não está contida. Foi então possível subtrair os resultados dessa consulta de uma busca geral na tabela de Naves.

```
SELECT *
FROM
    nave
WHERE nave.nome_embarcacao NOT IN
    (SELECT
        nave.nome_embarcacao
    FROM nave
    JOIN participa ON participa.nave = nave.nome_embarcacao
    JOIN missao ON
        participa.titulo_missao = missao.titulo_missao
        AND participa.data_inicio = missao.data_inicio
        AND participa.tipo_missao = missao.tipo_missao
    LEFT JOIN construcao_estacao ON
        construcao_estacao.titulo_missao = missao.titulo_missao
        AND construcao_estacao.data_inicio = missao.data_inicio
        AND construcao_estacao.tipo_missao = missao.tipo_missao
    LEFT JOIN exploracao_reconhecimento ON
        exploracao_reconhecimento.titulo_missao = missao.titulo_missao
        AND exploracao_reconhecimento.data_inicio = missao.data_inicio
        AND exploracao_reconhecimento.tipo_missao = missao.tipo_missao
    LEFT JOIN pesq_cientifica ON
        pesq_cientifica.titulo_missao = missao.titulo_missao
        AND pesq_cientifica.data_inicio = missao.data_inicio
        AND pesq_cientifica.tipo_missao = missao.tipo_missao
    LEFT JOIN extracao_recurso ON
        extracao_recurso.titulo_missao = missao.titulo_missao
        AND extracao_recurso.data_inicio = missao.data_inicio
        AND extracao_recurso.tipo_missao = missao.tipo_missao
    WHERE
        COALESCE(construcao_estacao.data_limite,
exploracao_reconhecimento.data_limite, pesq_cientifica.data_limite,
extracao_recurso.data_limite) >= '2567-01-01'
        AND missao.data_inicio <= '2567-01-01');
```

7.2. Consultando qual o tipo de Missão que uma Corporação mais realizou

Esta consulta permite que um usuário seja capaz de visualizar, para cada **Corporação**, qual dos quatro tipos de **Missão** - dentre **Construção de Estação**, **Exploração e Reconhecimento**, **Extração de Recursos**, ou **Pesquisa Científica -** ela mais realizou. Para contabilizar os tipos de missões, foram realizadas junções externas e em seguida uma contagem dos atributos dos diferentes tipos. Para cada **Corporação**, é possível comparar as quantidades de cada tipo de **Missão** realizado por ela no resultado.

```
SELECT DISTINCT missao.titulo_missao, missao.data_inicio FROM missao;
SELECT
    corporacao AS Corporacao,
    COUNT(CASE WHEN tipo_construcao_estacao IS NOT NULL THEN 1 END) AS
Qtd_Construcao_Estacao,
    COUNT(CASE WHEN tipo_exploracao_reconhecimento IS NOT NULL THEN 1 END) AS
Qtd_Exploracao_Reconhecimento,
    COUNT(CASE WHEN tipo_pesquisa_cientifica IS NOT NULL THEN 1 END) AS
Qtd_Pesquisa_Cientifica,
    COUNT(CASE WHEN tipo_extracao_recurso IS NOT NULL THEN 1 END) AS
Qtd_Extracao_Recurso
FROM (
    SELECT
        missao.titulo_missao,
        missao.data_inicio,
        MAX(construcao_estacao.tipo_missao) AS tipo_construcao_estacao,
        MAX(exploracao_reconhecimento.tipo_missao) AS
tipo_exploracao_reconhecimento,
        MAX(pesq_cientifica.tipo_missao) AS tipo_pesquisa_cientifica,
        MAX(extracao_recurso.tipo_missao) AS tipo_extracao_recurso,
        MAX(contrato.titulo_contrato) AS contrato_titulo,
        MAX(contrato.data_inicio) AS contrato_data_inicio,
        MAX(contrato.governo) AS contrato_governo,
        MAX(corporacao.nome) AS corporacao
    FROM missao
    LEFT JOIN construcao_estacao ON
        construcao_estacao.titulo_missao = missao.titulo_missao
        AND construcao_estacao.data_inicio = missao.data_inicio
        AND construcao_estacao.tipo_missao = missao.tipo_missao
    LEFT JOIN exploracao_reconhecimento ON
        exploracao_reconhecimento.titulo_missao = missao.titulo_missao
        AND exploracao_reconhecimento.data_inicio = missao.data_inicio
        AND exploracao_reconhecimento.tipo_missao = missao.tipo_missao
    LEFT JOIN pesq_cientifica ON
        pesq_cientifica.titulo_missao = missao.titulo_missao
        AND pesq_cientifica.data_inicio = missao.data_inicio
```

```
        AND pesq_cientifica.tipo_missao = missao.tipo_missao
    LEFT JOIN extracao_recurso ON
        extracao_recurso.titulo_missao = missao.titulo_missao
        AND extracao_recurso.data_inicio = missao.data_inicio
        AND extracao_recurso.tipo_missao = missao.tipo_missao
    JOIN contrato ON
        (
            (
                contrato.titulo_contrato = construcao_estacao.titulo_contrato
                AND contrato.data_inicio =
construcao_estacao.data_inicio_contrato
                AND contrato.governo = construcao_estacao.governo
            ) OR (
                contrato.titulo_contrato =
exploracao_reconhecimento.titulo_contrato
                AND contrato.data_inicio =
exploracao_reconhecimento.data_inicio_contrato
                AND contrato.governo = exploracao_reconhecimento.governo
            ) OR (
                contrato.titulo_contrato = pesq_cientifica.titulo_contrato
                AND contrato.data_inicio = pesq_cientifica.data_inicio_contrato
                AND contrato.governo = pesq_cientifica.governo
            ) OR (
                contrato.titulo_contrato = extracao_recurso.titulo_contrato
                AND contrato.data_inicio = extracao_recurso.data_inicio_contrato
                AND contrato.governo = extracao_recurso.governo
            )
        )
    JOIN corporacao ON corporacao.codigo_corporativo = contrato.corporacao
    GROUP BY missao.titulo_missao, missao.data_inicio
    ORDER BY missao.data_inicio, missao.titulo_missao
) AS subconsulta
GROUP BY corporacao
ORDER BY corporacao;
```

Abaixo um exemplo de resultado da consulta mostrando as quantidades de tipos de missão por corporação.

	corporacao text	qtd_construcao_estacao bigint	qtd_exploracao_reconhecimento bigint	qtd_pesquisa_cientifica bigint	qtd_extracao_recurso bigint
1	Companhia de Extração	8	3	2	8
2	Empresa Bet Aposte Aqui	4	0	0	7
3	Moster AnyJobs LTD	1	1	1	0
4	Taylor Swift Corporation	2	2	1	1
5	Uber and Sons Express	0	0	3	2

7.3. Consultando os Recursos mais extraídos por cada Governo

A consulta abaixo tem como objetivo mostrar o **Recurso** mais extraído por cada **Governo**, retornando um conjunto de tuplas com o nome do governo, o recurso mais extraído e a quantidade extraída. A quantidade total leva em conta todas as **Missões** (**de Extração**) em todos os **Corpos Planetários**. Na subconsulta seleciona-se o **Governo**, **Recurso**, a soma da quantidade do recurso (operação **SUM** da quantidade sobre cada **Recurso** de cada **Governo**) e aplica a função ROW_NUMBER() sobre as linhas particionando-as pelo Governo e em ordem decrescente pela somatória da quantidade de um determinado Recurso, o resultado aparece como um atributo Rank. A tabela de onde será selecionado os atributos é uma junção entre Governo, Contrato, Extracao_Recurso e Extrai, de forma a pegar todas as informações necessárias espalhadas por essas tabelas. Ao final seleciona o Rank = 1 para selecionar apenas a primeira linha dos recursos mais extraídos por cada Governo.

```
SELECT
    governo,
    recurso,
    total_extraido
FROM
    (SELECT
        governo.nome AS governo,
        recurso.nome AS recurso,
        SUM(extrai.quantidade) AS total_extraido,
        ROW_NUMBER() OVER (PARTITION BY governo.nome ORDER BY
SUM(extrai.quantidade) DESC) as rank
    FROM governo
    JOIN contrato ON contrato.governo = governo.nome
    JOIN extracao_recurso ON
        extracao_recurso.governo = contrato.governo
        AND extracao_recurso.titulo_contrato = contrato.titulo_contrato
        AND extracao_recurso.data_inicio_contrato = contrato.data_inicio
    JOIN extrai ON
        extrai.titulo_missao = extracao_recurso.titulo_missao
        AND extrai.data_inicio = extracao_recurso.data_inicio
        AND extrai.tipo_missAO = extracao_recurso.tipo_missao
    JOIN recurso ON recurso.nome = extrai.recurso
    GROUP BY governo.nome, recurso.nome
    ORDER BY governo.nome, SUM(extrai.quantidade) DESC) AS ranked
WHERE rank = 1;
```

Abaixo está um exemplo de resultado para a consulta.

	governo character varying	recurso character varying	total_extraido numeric
1	Fundação X	Silício	5021787500.000
2	Império de Neo Pompeia	Chifre das Baleias Barcelona	4100.000
3	República Federativa StarBet	Silício	4000.000

7.4. Consultando os Corpos Planetários que estão há mais tempo sem receber uma Missão

Esta consulta permite que um usuário seja capaz de selecionar, dentre os **CorposPlanetarios** que já receberam uma **Missão** mas que não estão recebendo uma no momento, aqueles que estão há mais tempo sem receber uma **Missão**, de qualquer tipo. Realizamos uma subconsulta para pegar o TituloMissao, DataInicio de Missão e DataLimite e CorpoPlanetario de alguns dos esquemas especializados de Missão (Construcao_Estacao, Exploracao_Reconhecimento, Pesq_Cientifica e Extracao_Recurso). Para isso foi necessário realizar a junção entre os Missão e suas especializações por meio de uma sucessão de JOIN LEFT, usando como condição a chave primária de todas elas. A principal condição da para tabela resultante foi a de selecionar apenas as tuplas onde o Status fosse 'FINALIZADA', indicando uma missão finalizada. A partir daqui temos todas as missões (independente do tipo) que já acabaram, realizando-se um LEFT JOIN com a tabela das missões com Status 'ATIVA', usando CorpoPlanetario como condição de junção, e uma condição adicional `MA.CORPO_PLANETARIO IS NULL`, temos todos os corpos planetários que não estão sendo visitados no momento. Agrupando por CorpoPlanetario é possível aplicar a função de agrupamento `MAX()` para se obter a data mais recente da última visita por CorpoPlanetario.

```
SELECT MF.CORPO_PLANETARIO, MAX(MF.DATA_LIMITE) AS ULTIMA_VISITA
FROM (
    SELECT
    M.TITULO_MISSAO,
    M.DATA_INICIO,
    COALESCE(MAX(C.DATA_LIMITE), MAX(E.DATA_LIMITE), MAX(P.DATA_LIMITE),
MAX(R.DATA_LIMITE)) AS DATA_LIMITE,
    COALESCE(MAX(C.CORPO_PLANETARIO), MAX(E.CORPO_PLANETARIO),
MAX(P.CORPO_PLANETARIO), MAX(R.CORPO_PLANETARIO)) AS CORPO_PLANETARIO
    FROM
        MISSAO M
    LEFT JOIN CONSTRUCAO_ESTACAO C
```

```
        ON M.TITULO_MISSAO = C.TITULO_MISSAO
        AND M.DATA_INICIO = C.DATA_INICIO
        AND M.TIPO_MISSAO = C.TIPO_MISSAO
    LEFT JOIN EXPLORACAO_RECONHECIMENTO E
        ON M.TITULO_MISSAO = E.TITULO_MISSAO
        AND M.DATA_INICIO = E.DATA_INICIO
        AND M.TIPO_MISSAO = E.TIPO_MISSAO
    LEFT JOIN PESQ_CIENTIFICA P
        ON M.TITULO_MISSAO = P.TITULO_MISSAO
        AND M.DATA_INICIO = P.DATA_INICIO
        AND M.TIPO_MISSAO = P.TIPO_MISSAO
    LEFT JOIN EXTRACAO_RECURSO R
        ON M.TITULO_MISSAO = R.TITULO_MISSAO
        AND M.DATA_INICIO = R.DATA_INICIO
        AND M.TIPO_MISSAO = R.TIPO_MISSAO
    WHERE
        COALESCE(
            C.STATUS,
            E.STATUS,
            P.STATUS,
            R.STATUS
        ) = 'FINALIZADA'
    GROUP BY M.TITULO_MISSAO, M.DATA_INICIO
) AS MF
LEFT JOIN (
    SELECT
        M.TITULO_MISSAO,
        M.DATA_INICIO,
        COALESCE(MAX(C.CORPO_PLANETARIO), MAX(E.CORPO_PLANETARIO),
MAX(P.CORPO_PLANETARIO), MAX(R.CORPO_PLANETARIO)) AS CORPO_PLANETARIO
    FROM
        MISSAO M
    LEFT JOIN CONSTRUCAO_ESTACAO C
        ON M.TITULO_MISSAO = C.TITULO_MISSAO
        AND M.DATA_INICIO = C.DATA_INICIO
        AND M.TIPO_MISSAO = C.TIPO_MISSAO
    LEFT JOIN EXPLORACAO_RECONHECIMENTO E
        ON M.TITULO_MISSAO = E.TITULO_MISSAO
        AND M.DATA_INICIO = E.DATA_INICIO
        AND M.TIPO_MISSAO = E.TIPO_MISSAO
    LEFT JOIN PESQ_CIENTIFICA P
        ON M.TITULO_MISSAO = P.TITULO_MISSAO
        AND M.DATA_INICIO = P.DATA_INICIO
        AND M.TIPO_MISSAO = P.TIPO_MISSAO
    LEFT JOIN EXTRACAO_RECURSO R
        ON M.TITULO_MISSAO = R.TITULO_MISSAO
        AND M.DATA_INICIO = R.DATA_INICIO
        AND M.TIPO_MISSAO = R.TIPO_MISSAO
    WHERE
        COALESCE(
            C.STATUS,
```

```
            E.STATUS,
            P.STATUS,
            R.STATUS
        ) = 'ATIVA'
    GROUP BY
        M.TITULO_MISSAO, M.DATA_INICIO
) AS MA
ON MF.CORPO_PLANETARIO = MA.CORPO_PLANETARIO
WHERE MA.CORPO_PLANETARIO IS NULL
GROUP BY MF.CORPO_PLANETARIO
ORDER BY ULTIMA_VISITA;
```

Abaixo pode-se ter um exemplo do resultado da consulta.

	corpo_planetario text	ultima_visita date
1	Barcelona	2426-03-29
2	Calculo	2495-07-30
3	USP	2505-07-10
4	Camisus	2566-06-25
5	Shortus	2566-06-25
6	Chinelus	2566-08-25
7	Tenius	2566-08-25
8	Bolsus	2566-08-25

7.5. Consultando as Corporações que pegaram todos os contratos de um Governo

A consulta em questão poderia ter sido realizada de uma forma mais específica para um **Governo**, o primeiro *Script* mostra com tal consulta poderia ser realizada. Nela observa-se uma junção interna entre os esquemas **Corporação** e **Contrato** por meio do Código Corporativo, selecionando apenas a tuplas onde **Governo** fosse ‘Fundação X’ (um Governo específico). Nesse ponto já teríamos as **Corporações** que tem algum **Contrato** daquele **Governo**, mas há uma forma bem simples de descobrir se a **Corporação** realiza a totalidade dos **Contratos** liberados pelo **Governo**, basta verificar se o número de tuplas de **Contrato**, onde **Corporação** tem ligação com ‘Fundação X’ é a igual ao número total de tuplas em **Contratos** onde **Governo** é igual a ‘Fundação X’.

Podemos deixar isso mais generalizado consultando todas as **Corporações** que realizam todos os **Contratos** de lançados por uma determinado **Governo** (para fins de esclarecimento é frisar que **Corporações** podem estar em todos os **Contratos** de um **Governo** e mesmo assim estar em **Contratos** com outros **Governos**). A consulta teria grande utilidade para verificar se algum **Governo** faz contratos apenas com uma **Corporação**.

```
SELECT CORP.CODIGO_CORPORATIVO, CORP.NOME
FROM CORPORACAO CORP JOIN CONTRATO C
ON CORP.codigo_corporativo = C.CORPORACAO
WHERE C.GOVERNO = 'Fundação X'
GROUP BY CORP.CODIGO_CORPORATIVO, CORP.NOME
HAVING COUNT(*) = (
    SELECT COUNT(*)
    FROM CONTRATO CONT
    WHERE CONT.GOVERNO = 'Fundação X'
);

SELECT
    CORP.CODIGO_CORPORATIVO,
    CORP.NOME,
    GOV.NOME AS NOME_GOVERNO,
    COUNT(*) AS NUMERO_CONTRATOS
FROM
    CORPORACAO CORP
JOIN
    CONTRATO C ON CORP.CODIGO_CORPORATIVO = C.CORPORACAO
JOIN
    GOVERNO GOV ON C.GOVERNO = GOV.NOME
GROUP BY
    CORP.CODIGO_CORPORATIVO,
    CORP.NOME,
    GOV.NOME
HAVING COUNT(*) = (
    SELECT COUNT(*)
    FROM CONTRATO CONT
    WHERE CONT.GOVERNO = GOV.NOME
);
```

Abaixo está um exemplo de resultado para os dois scripts acima, respectivamente.

	codigo_corporativo character varying	nome character varying
1	CIA13145	Companhia de Extração

	codigo_corporativo character varying	nome character varying	nome_governo character varying	numero_contratos bigint
1	CIA13145	Companhia de Extração	Fundação X	2
2	BET12345	Empresa Bet Aposte Aqui	Unidade de Ursa Menor	1

8. Mudanças na Parte 2 para a Terceira Entrega

8.1. Correções ao Modelo Lógico proposto

No Modelo Lógico entregue na segunda entrega do projeto existiam alguns problemas. São eles:

- A relação dos diferentes **tipos de missão** com **Corporação**, o que permitia a inconsistência de uma missão pode apontar para um contrato de uma corporação diferente à apontada pela própria missão. Esse problema foi corrigido retirando-se o relacionamento direto dos tipos de missões com corporação, uma vez que essa informação já estava no contrato da missão.
- O mapeamento como participação total de **Corporação** em **Contrato**, quando o mesmo não existia no Modelo Entidade-Relacionamento. Foi resolvido retirando o *not null* do atributo corporação presente em contrato.
- A chave estrangeira de **Missão** é feita de maneira incompleta, não carregando a informação de TipoMissão. Foi resolvido adicionando-se o atributo à chave estrangeira às tabelas que referenciam a tabela Missão.
- A ausência da nota **not null*, que foi adicionada como solução.

8.2. Revisões ao relatório e ao Modelo Entidade-Relacionamento

Na segunda entrega, o relatório trazia a explicação da fragilidade que o erro do primeiro ponto do tópico anterior implicava. Foi necessário então corrigir o texto para explicar o modelo sem as relações entre os tipos de Missão e Corporação e retirar a explicação da fragilidade.

Foi também corrigido o atributo *TipoMissão* da tabela **Missão** no Modelo Entidade-Relacionamento, que deve ser multivalorado.

9. Conclusão

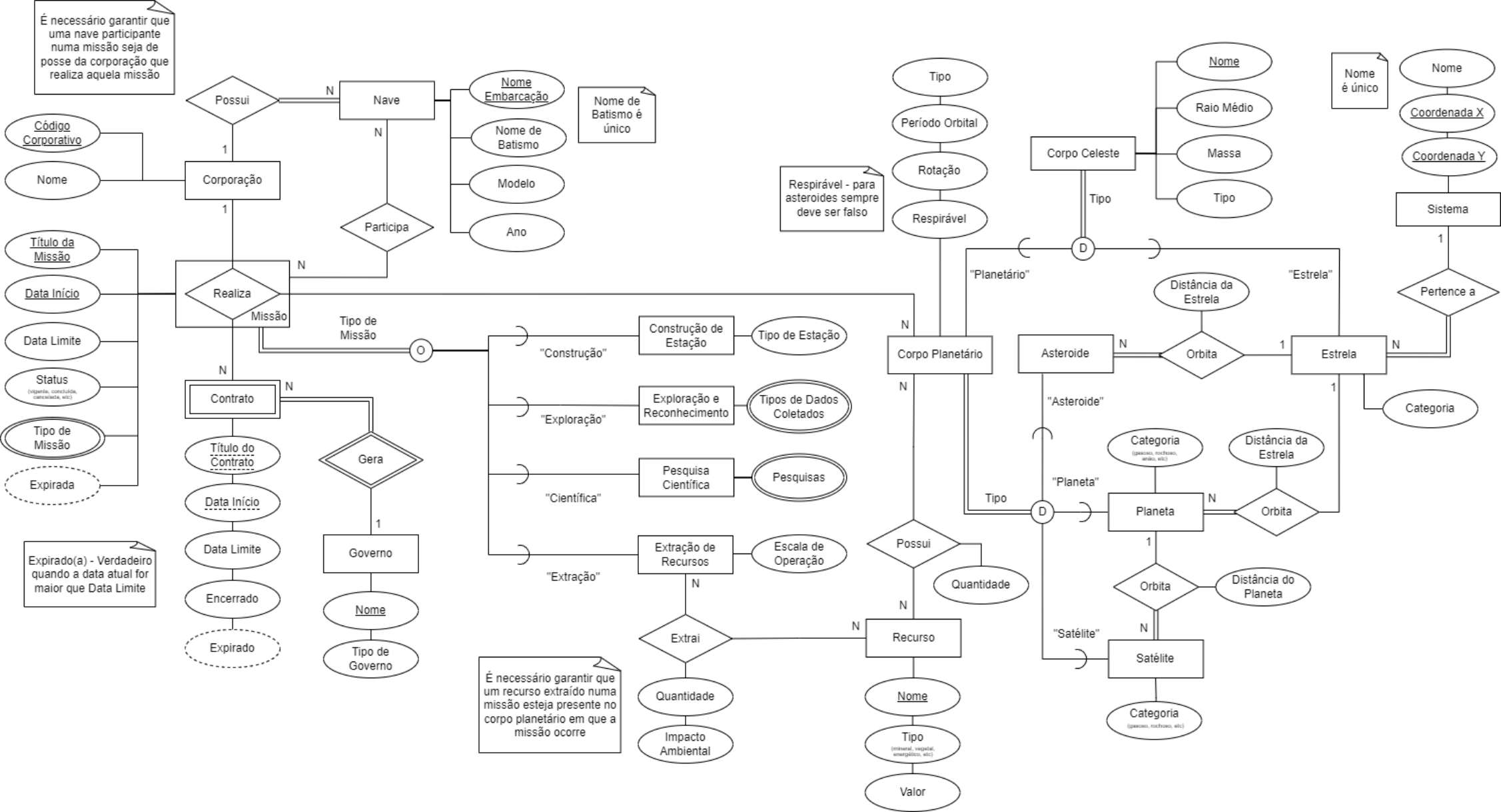
O desenvolvimento do projeto foi bastante desafiador e instigante, desde conceber as primeiras ideias e como passá-las para um modelo entidade-relacionamento, até construir a aplicação e criar as consultas. Entender as fragilidades que a arquitetura do banco favorecia foi um desafio em particular, mas que com os feedbacks, agregou muito ao conhecimento de todos da equipe.

A primeira parte do projeto se mostrou desafiadora pois ela exige uma visão de projeto que não tínhamos no início da disciplina. Como o MER é um modelo que não tem uma única resposta possível, fazê-lo como uma solução adequada para o problema exige experiência com a área.

Da segunda parte do projeto, montar o modelo lógico foi considerado bem leve. O mais trabalhoso foram as discussões que exigiram atenção para todas as outras modelagens possíveis para cada relacionamento e todas as nuances do banco, que poderia fazer com que houvesse soluções diferentes para um mesmo tipo de relacionamento.

A terceira parte foi a que o grupo mais se sentiu estimulado por ser uma etapa mais prática, ao mesmo tempo que foi uma etapa bem trabalhosa e desafiadora, principalmente na construção dos *scripts* das consultas e da aplicação em si.

Na disciplina, como um todo, consideramos que as atividades solicitadas ao longo da disciplina foram condizentes, tanto nas provas como no trabalho, mas que talvez ambas foram extensas considerando o tempo que seria necessário para fazê-las com qualidade.


É necessário garantir que uma nave participante numa missão seja de posse da corporação que realiza aquela missão
Possui
N
Nave
Nome Embarcação
Nome de Batismo
Modelo
Ano
Nome de Batismo é único
Código Corporativo
Nome
Corporação
1
N
Participa
Título da Missão
Data Início
Data Limite
Status
(vigente, concluída, cancelada, etc)
Tipo de Missão
Expirada
Realiza
N
Missão
Tipo de Missão
O
Contrato
N
Título do Contrato
Data Início
Data Limite
Encerrado
Expirado
Gera
1
Governo
Nome
Tipo de Governo
Expirado(a) - Verdadeiro quando a data atual for maior que Data Limite
"Construção"
Construção de Estação
Tipo de Estação
"Exploração"
Exploração e Reconhecimento
Tipos de Dados Coletados
"Científica"
Pesquisa Científica
Pesquisas
"Extração"
Extração de Recursos
Escala de Operação
N
Extrai
Quantidade
Impacto Ambiental
É necessário garantir que um recurso extraído numa missão esteja presente no corpo planetário em que a missão ocorre
Possui
N
Quantidade
N
Recurso
Nome
Tipo
(mineral, vegetal, energético, etc)
Valor
Tipo
Período Orbital
Rotação
Respirável
Respirável - para asteroides sempre deve ser falso
N
Corpo Planetário
N
Corpo Celeste
Nome
Raio Médio
Massa
Tipo
Tipo
D
"Planetário"
"Estrela"
"Asteroide"
"Planeta"
"Satélite"
Tipo
D
Asteroide
N
Orbita
1
Estrela
N
Distância da Estrela
Categoria
Planeta
N
Orbita
1
Distância da Estrela
Categoria
(gasoso, rochoso, anão, etc)
Orbita
N
Distância do Planeta
Satélite
Categoria
(gasoso, rochoso, etc)
Nome é único
Nome
Coordenada X
Coordenada Y
Sistema
1
Pertence a
N
1
Categoria


* = NOT NULL
Governo = {Nome, TipoGoverno}
Corporação = {CódigoCorporativo, Nome}
Contrato = {Governo, TítuloContrato, DataInício, Corporação, DataLimite, Encerrado}
Nave = {NomeEmbarcação, NomeBatismo, Modelo, Ano, Corporação}
Participa = {TítuloMissão, DataInício, TipoMissão, Nave}
Missão = {TítuloMissão, DataInício, TipoMissão}
ConstrucaoEstacao = {TítuloMissão, DataInício, TipoMissão, Governo, TítuloContrato, InícioContrato, CorpoPlanetário, DataLimite, Status, TipoEstacao}
ExploracaoReconhecimento = {TítuloMissão, DataInício, TipoMissão, Governo, TítuloContrato, InícioContrato, CorpoPlanetário, DataLimite, Status}
DadosColetados = {TítuloMissão, DataInício, TipoMissão, DadoColetado}
PesqCientifica = {TítuloMissão, DataInício, TipoMissão, Governo, TítuloContrato, InícioContrato, CorpoPlanetário, DataLimite, Status}
Pesquisas = {TítuloMissão, DataInício, TipoMissão, Pesquisa}
ExtracaoRecursos = {TítuloMissão, DataInício, TipoMissão, Governo, TítuloContrato, InícioContrato, CorpoPlanetário, DataLimite, Status, EscalaOperação}
Extrai = {TítuloMissão, DataInício, TipoMissão, Recurso, Quantidade, ImpactoAmbiental}
Sistema = {CoordenadaX, CoordenadaY, Nome}
Corpo Celeste = {Nome, Tipo}
Estrela = {Nome, RaioMédio, Massa, Categoria, CoordenadaX, CoordenadaY}
Corpo Planetário = {Nome, Tipo}
Asteroide = {Nome, RaioMédio, Massa, Estrela, DistânciaEstrela, PeríodoOrbital, Rotação, Respirável}
Recurso = {Nome, Tipo, Valor}
Planeta = {Nome, RaioMédio, Massa, Estrela, DistânciaEstrela, Categoria, PeríodoOrbital, Rotação, Respirável}
Satélite = {Nome, RaioMédio, Massa, Planeta, DistânciaPlaneta, Categoria, PeríodoOrbital, Rotação, Respirável}
Possui = {Recurso, CorpoPlanetario, Quantidade}