

Relatório:

BASE DE DADOS 2023

PROJETO FINAL

Diogo Machado Marto N° Mec: 108298 50%
Tiago Pereira N° Mec: 108546 50%

Contents

Contents.....	2
Análise de requisitos	3
DER.....	5
Esquema Relacional da BD	6
.....	6
SQL DDL.....	7
SQL DML.....	8
Normalização	8
Indices.....	9
Caso 1: Procurar veiculos por owner.....	9
Caso 2: Procurar Launch por Launch site	10
Caso 3: Procurar Satellite por norad id.....	10
Triggers	11
Stored Procedures	12
UDF	15
Views.....	16
Cursorres.....	17
Transações.....	18
Conclusão	20

Análise de requisitos

Exploração espacial é apontada por muitos como a indústria do futuro. Para isso, temos de começar a preparar-nos já, no presente. Faremos uma modelação dum sistema de suporte a uma Companhia Espacial, por motivos óbvios, simplista face àquele que é o modelo real.

Assim, tomamos os seguintes requisitos:

Uma Companhia Espacial, caracterizada por nome, ID, país, tem supervisão sobre vários programas espaciais.

Companhia Espacial Privada- Uma Companhia Espacial com um CEO.

CEO- pessoa que tem um ou varias companhias espaciais

Companhia Espacial Pública- Uma Companhia Espacial sob alçada de um Governo.

Um programa está associado a uma única companhia espacial e tem várias missões.

Cada missão espacial tem uma descrição, um ID, um orçamento, data de início, data da sua conclusão, status e vários eventos. Uma missão pode pertencer a mais do que uma companhia e ter envolvidos mais do que uma spacecraft .

Eventos tem um nome, data / hora e status.

Launch- Missão de Lançamento com um launch site e um launch vehicle.

Launch site- uma localização associada a uma companhia espacial.

Launch vehicle – Vehicle caracterizado por Cost per launch, Development cost, load, fuel type, range.

Vehicle- Name, OWNER, Size, Mass, Manufacturer, Country.

Um supervisor é um astronauta responsável por comandar a missão.

Astronaut é identificado por ID, primeiro e último nome, data de nascimento, nacionalidade, número de missions que participou e pode possuir várias especialidades.

Crew é um aglomerado de astronautas com um supervisor.

Especialidade é identificada por nome e descrição.

Spacecraft- Vehicle caracterizado por COSPAR ID/Int'l Cod, launch date, launch site, Launch vehicle, Propulsion system, Missions que foi usado, Purpose, Status, Description.

Space Station - Spacecraft caraterizada por lista de módulos e NORAD/SATCAT, Orbit type, Perigee, Apogee, Inclination, Periodo, latitude, longitude , altitude , velocidade , Min capacity e Max capacity.

Módulos- tipo, descrição e status.

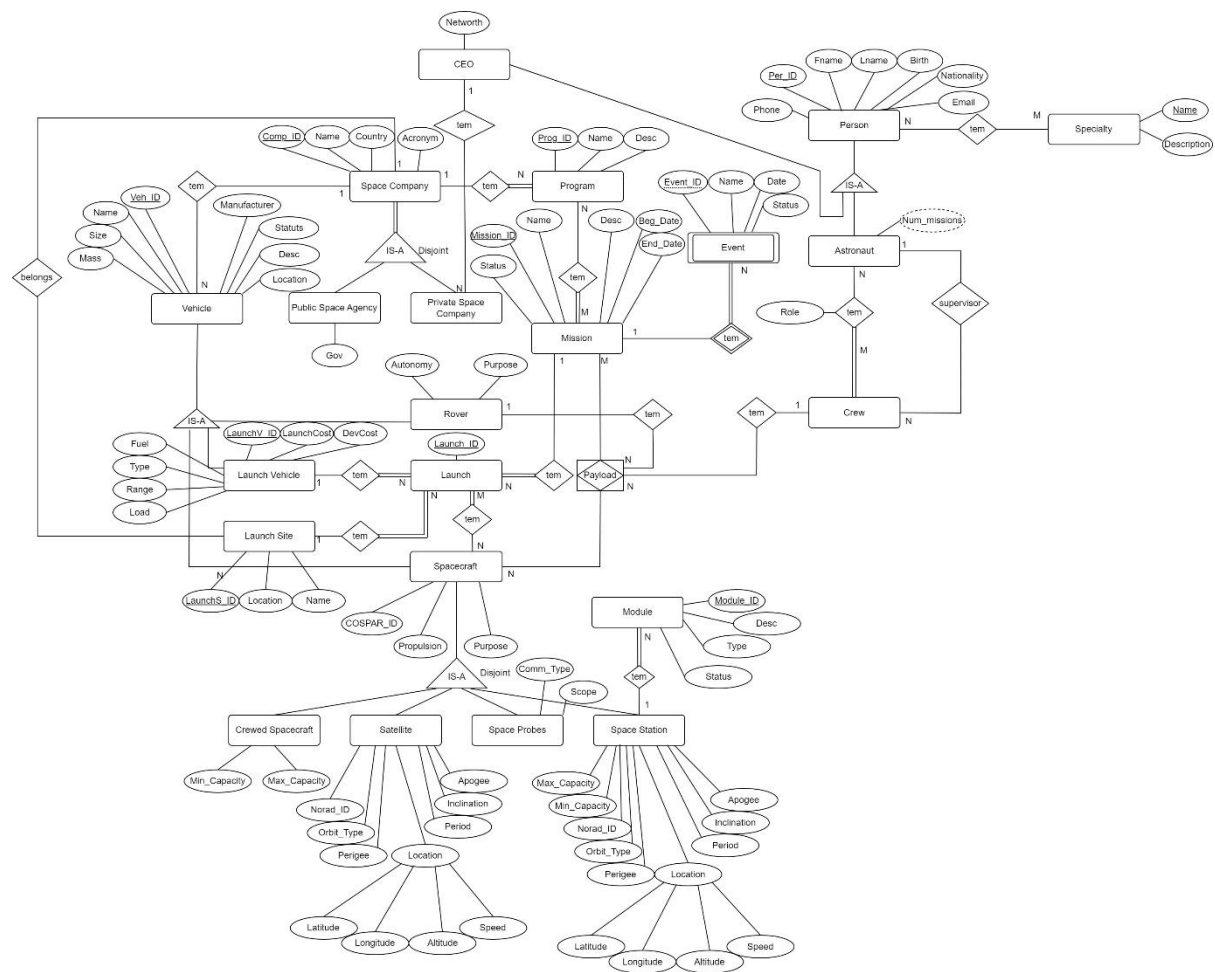
Rover- Spacecraft caracterizado por Location, autonomia

Space Probe- Spacecraft caracterizado por Location, autonomia, Communication Type(Communication).

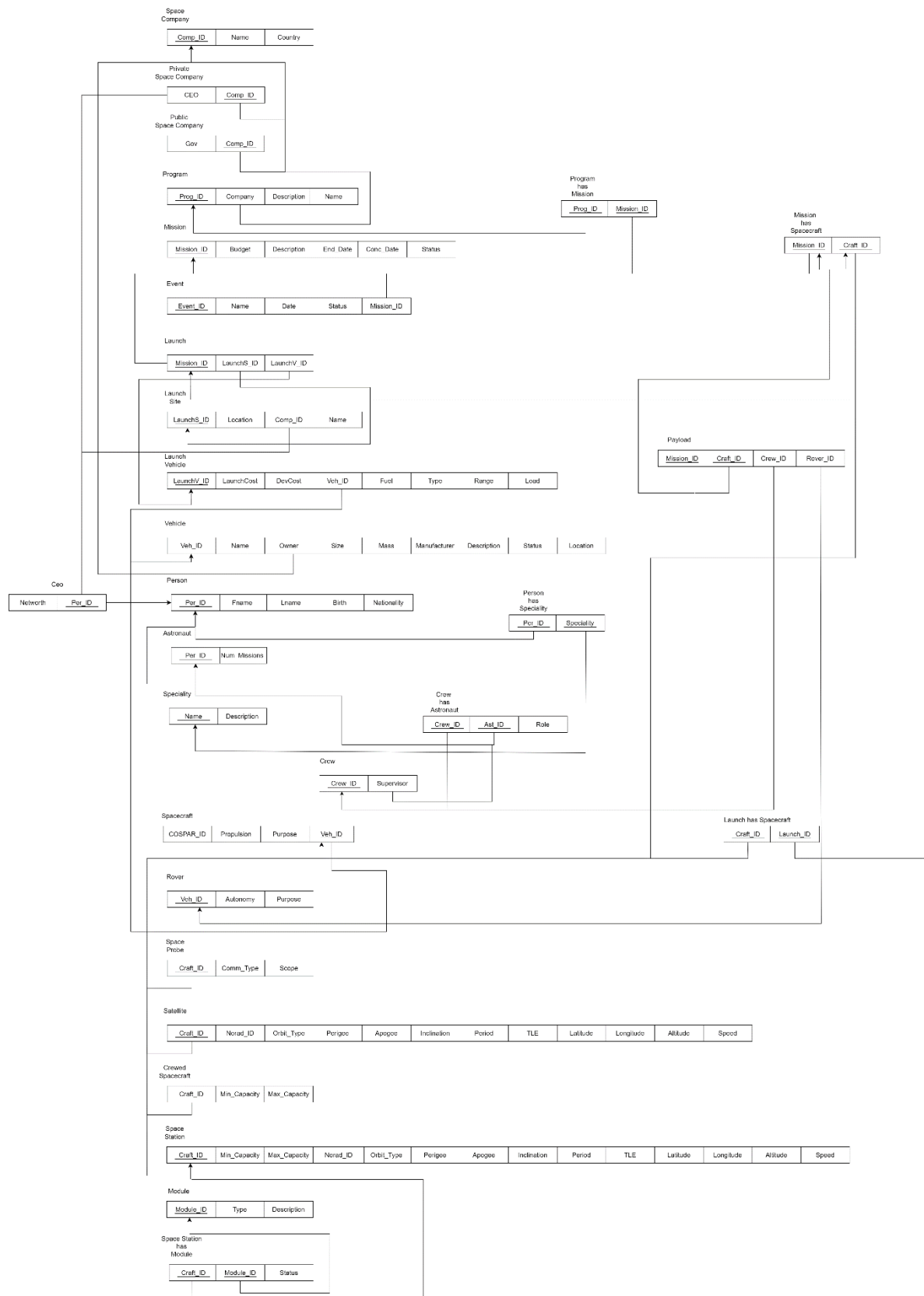
Satellite – spacecraft com os atributos NORAD/SATCAT, Orbit type, Perigee, Apogee, Inclination, Periodo, latitude, longitude , altitude , velocidade.

Crewed Spacecraft- spacecraft que pode ter tripulação. Caracterizada por Min capacity, Max capacity.

DER



Esquema Relacional da BD



SQL DDL

Resultante do ER feito na seccção anterior , criamos 29 tabelas e definimos restrições de integridades para cada uma delas como restrições foreign key , primary key , unique , not null e checks que juntas consituem restrições de integridade de entidade, integridade referencial e integridade de domínio.

Ex da tabela Satellite:

```
CREATE TABLE [Satelite] (
    [Craft_ID] INTEGER NOT NULL FOREIGN KEY REFERENCES SpaceCraft(Veh_ID),
    [Norad_ID] INTEGER NULL CHECK ([Norad_ID]>0),
    [Orbit_Type] VARCHAR(8) NULL,
    [Perigee] INTEGER NULL CHECK ([Perigee]>0),
    [Apogee] INTEGER NULL CHECK ([Apogee]>0),
    [Inclination] DECIMAL(8,5) NULL CHECK ([Inclination]>=0 AND
[Inclination]<=180),
    [Period] TIME NULL,
    [Latitude] DECIMAL(8,5) NULL CHECK ([Latitude]>=0 AND [Latitude]<=90),
    [Longitude] DECIMAL(8,5) NULL CHECK ([Longitude]>=0 AND [Longitude]<=180),
    [Altitude] DECIMAL(8,2) NULL CHECK ([Altitude]>0),
    [Speed] DECIMAL(16,4) NULL,
    PRIMARY KEY([Craft_ID]),
    UNIQUE(Norad_ID),
    UNIQUE([Latitude],[Longitude],[Altitude]) --This is the position if this was
not unique things would be in the same location which would be very bad
);
GO
```

Também em varias tabelas utilizamos identity para atribuir automaticamente um id as certas colunas

Ex da tabela SpaceCompany:

```
CREATE TABLE SpaceCompany(
    Comp_ID INT NOT NULL PRIMARY KEY identity(1,1),
    [Name] varchar(100) NOT NULL,
    Acronym varchar(10),
    Country varchar(100) NOT NULL
);
GO
```

Outro caso interessante é a coluna num_missions na tabela Astronaut que é um atributo derivado e é calculado através de uma call a uma udf que devolve um escalar.

```
ALTER TABLE Astronaut ADD Num_Mission AS dbo.getNumMissions(Per_ID);
```

SQL DML

Como principio base queremos evitar que a parte da interface manipule diretamente a base dados, por isso, as intruções de insert , update e delete nunca são executadas pela a Interface e são substituidas por stored procedures e udf's. Assim, criamos um camada de abstração para interagir como a base dados.

Também criamos intruções para inserirmos data para popular a base de dados após a criação da mesma, da qual a parte foi generada pelo o site <https://generatedata.com/>. Por exemplo inserirmos esta space companies:

```
INSERT INTO SpaceCompany ([Name],Acronym,Country)
VALUES
    ('European Space Agency','ESA','France'),
    ('National Aeronautics Space Administration','NASA','EUA'),
    ('Heróis do Espaço','HE','Portugal'),
    ('Samsung Voyager','SV','South Korea'),
    ('Outer Vladimir','Vlad','Russia');
```

Normalização

Após uma análise extensiva do ER, conseguimos concluir de o nosso esquema relacional encontra-se pelo menos na 3º forma normal. Uma decisão importante que fizemos foi ignorar depedências fisicas , por exemplo , alitude , longitude , latitude e periodo podem definir velocidade. Fizemos esta decisão porque queremos guardar todos os valores mesmo que dependam do valor de outros sem ter que complicar muito o esquema relacional.

Indices

Para encontrar indices para as nossas tabelas usamos estratégias usadas nas aulas práticas nomeadamente Live Query Statistics e o SQL Server Profiler. Executamos todas as query que desenvolvemos como Live Query Statistics e não encontramos nenhuma recomendação. Como não temos data de uso da base de dados e não sabemos as operações mais comuns que queremos otimizar, decidimos definir alguns casos e analisámos cada caso com o SQL Server Profiler.

Caso 1: Procurar veiculos por owner

Query:

```
select * from Vehicle where [Owner] < 3
```

Index:

```
create index VehicleOwner ON Vehicle ( [Owner] )
```

Antes do index:

select * from Vehicle where [Owner...	Microsoft SQ...	Kikom	DESKTO...	0	113	0	109
---------------------------------------	-----------------	-------	-----------	---	-----	---	-----

Depois do index:

select * from Vehicle where [Owner...	Microsoft SQ...	Kikom	DESKTO...	0	28	0	240
---------------------------------------	-----------------	-------	-----------	---	----	---	-----

Podemos observar uma redução de 113 reads para 28.

Caso 2: Procurar Launch por Launch site

Query:

```
select * from Launch where LaunchS_ID = 1
```

Index:

```
create index LaunchLaunchSite ON Launch ( [LaunchS_ID] )
```

Antes do index:

select * from Launch where LaunchS...	Microsoft SQ...	Kikom	DESKTO...	16	112	0	46
---------------------------------------	-----------------	-------	-----------	----	-----	---	----

Depois do index:

select * from Launch where LaunchS...	Microsoft SQ...	Kikom	DESKTO...	0	26	0	41
---------------------------------------	-----------------	-------	-----------	---	----	---	----

Podemos observar uma redução de 112 reads para 26.

Caso 3: Procurar Satellite por norad id

Query

```
select * from Satellite where Norad_ID < 50
```

Index:

```
create unique index SatelliteNorad ON Satellite ( [NORAD_ID] )
```

Antes do index:

select * from Satellite where Norad...	Microsoft SQ...	Kikom	DESKTO...	0	251	0	106
--	-----------------	-------	-----------	---	-----	---	-----

Depois do index:

select * from Satellite where Norad...	Microsoft SQ...	Kikom	DESKTO...	0	36	0	109
--	-----------------	-------	-----------	---	----	---	-----

Podemos observar uma redução de 251 reads para 36.

Triggers

Usamos os triggers para impor restrições na base de dados que não conseguimos utilizando o DDL.

 EmailOrPhone.sql

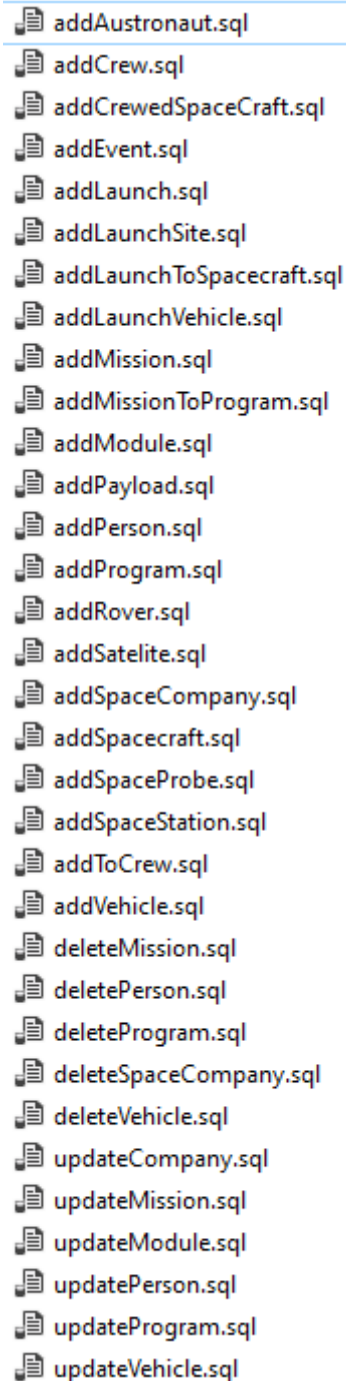
 NoradNullPositionNullS.sql

 NoradNullPositionNullSS.sql

- EmailOrPhone – Uma Person tem de ter um email or um phone se ambos tiverem a null o trigger é ativado e não é feito. [Instead of insert , update]
- NoradNullPositionNullS – Um Satellite pode ter um NORAD ID que lhe é atribuido quando entra orbita , assim quando um satellite tem NORAD ID a null este não pode estar em orbita , ou seja , não pode ter atributos relacionados com a orbita. [Instead of insert , update]
- NoradNullPositionNullSS – Uma SpaceStation pode ter um NORAD ID que lhe é atribuido quando entra orbita , assim quando um satellite tem NORAD ID a null este não pode estar em orbita , ou seja , não pode ter atributos relacionados com a orbita. [Instead of insert , update]

Stored Procedures

Utilizamos os stored procedures para encapsular um conjunto de instruções que são chamadas a partir da interface , assim criando uma camada de abstração que também protege a base de dados contra manipulação direta dos dados. Estas stored procedures encontra em 3 categorias add (inserts) , delete e updates.



A screenshot of a file explorer window displaying a list of SQL stored procedure files. The files are organized into three categories: 'add' (inserts), 'delete', and 'update'. Each file name is preceded by a small icon representing a document. The list is as follows:

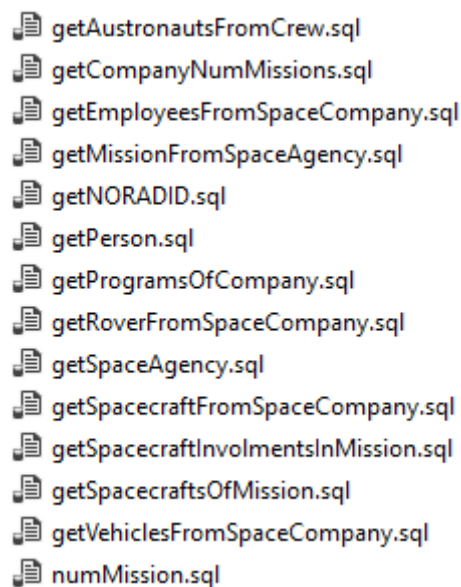
- addAstronaut.sql
- addCrew.sql
- addCrewedSpaceCraft.sql
- addEvent.sql
- addLaunch.sql
- addLaunchSite.sql
- addLaunchToSpacecraft.sql
- addLaunchVehicle.sql
- addMission.sql
- addMissionToProgram.sql
- addModule.sql
- addPayload.sql
- addPerson.sql
- addProgram.sql
- addRover.sql
- addSatelite.sql
- addSpaceCompany.sql
- addSpacecraft.sql
- addSpaceProbe.sql
- addSpaceStation.sql
- addToCrew.sql
- addVehicle.sql
- deleteMission.sql
- deletePerson.sql
- deleteProgram.sql
- deleteSpaceCompany.sql
- deleteVehicle.sql
- updateCompany.sql
- updateMission.sql
- updateModule.sql
- updatePerson.sql
- updateProgram.sql
- updateVehicle.sql

- addAstronaut adiciona uma Person á tabela Astronaut.
- addCrew cria uma Crew a partir de um Astronauta supervisor e devolve um id
- addCrewedSpaceCraft torna a Spacecraft passada como argumento de entrada numa CrewedSpacraft como atributos especificados também na entrada.
- addEvent adiciona um Event a uma missão
- addLaunch cria um Launch e devolve o id correspondente.
- addLaunchSite cria um LaunchSite e devolve o id correspondente.
- addLaunchToSpaceCraft adiciona SpaceCraft a um Launch e certifica-se que SpaceCraft está envolvida na Mission correspondente ao Launch.
- addLaunchVehicle torna o Vehicle passado como argumento de entrada num LaunchVehicle como atributos especificados também na entrada.
- addMission cria uma Missão e devolve o id correspondente.
- addMissionToProgram adiciona uma Mission a um Program.
- addModule cria um Module.
- addPayload adiciona uma Payload a uma Misson and Spacecraft.
- addPerson cria uma Person.
- addProgram cria um Programa de um SpaceCompany
- addRover torna o Vehicle passado como argumento de entrada num Rover como atributos especificados também na entrada.
- addSatelite torna a Spacecraft passada como argumento de entrada num Satelite como atributos especificados também na entrada.
- addSpaceCompany cria uma SpaceCompany privada ou publica com base nos argumentos de entrada.
- addSpacecraft torna o Vehicle passado como argumento de entrada numa Spacecraft como atributos especificados também na entrada.
- addSpaceProbe torna a Spacecraft passada como argumento de entrada numa SpaceProbe como atributos especificados também na entrada.
- addSpaceStation torna a Spacecraft passada como argumento de entrada num SpaceStation como atributos especificados também na entrada.
- addToCrew adiciona um Astronaut a uma Crew.
- addVehicle cria um Vehicle.

- deleteMission faz delete de uma Mission, as suas Payloads , tira a dos Programas onde pertencia , delete dos seus Eventos e os Launch da Mission ficam sem nenhuma Mission associada.
- deletePerson faz delete de uma Person , se for dono de uma PrivateSpaceCompany esta fica sem dono , delete de Astronaut e CEO.
- deleteProgram faz delete de um Program e tira-se das Mission
- deleteSpaceCompany faz delete de uma SpaceCompany publica ou private e todos os seus programas.
- deleteVehicle faz delete de uma veiculo qualquer , as suas payload , tira-se dos Launch que pertencia por ser LaunchVehicle ou por ser Spacecraft , se for SpaceStation os seus Modules ficam sem SpaceStation.
- updateCompany dá update a um SpaceCompany e pode mudar se é privada ou publica.
- updateModule dá update a um Module.
- updateVehicle dá update a um Vehicle.
- updateProgram dá update a um Program.
- updateMission dá update a uma Mission.
- updatePerson dá update a uma Person.

UDF

Usamos udf's principalmente para termos fontes de dados que aceitam parametros de entrada , mas temos outros casos como numMission que é usado para definir um atributo derivado de Austronaut. As udf's também são usadas diretamente pela a interface.



A screenshot of a file explorer window showing a list of 15 SQL files, each with a document icon. The files are listed in a single column and are as follows:




- getAustronautsFromCrew.sql
- getCompanyNumMissions.sql
- getEmployeesFromSpaceCompany.sql
- getMissionFromSpaceAgency.sql
- getNORADID.sql
- getPerson.sql
- getProgramsOfCompany.sql
- getRoverFromSpaceCompany.sql
- getSpaceAgency.sql
- getSpacecraftFromSpaceCompany.sql
- getSpacecraftInvolmentsInMission.sql
- getSpacecraftsOfMission.sql
- getVehiclesFromSpaceCompany.sql
- numMission.sql

- getAustronautsFromCrew devolve uma lista de Austronauts que pertecem a uma Crew.
- getCompanyNumMissions devolve um escalar que tem o numero de Mission que pertencem a um Program dessa SpaceCompany.
- getEmployeesFromSpaceCompany devolve todas uma lista de Person envolvidas com a SpaceCompany ou porque é CEO ou poque é um Austronaut que está numa Mission da SpaceCompany.
- getMissionFromSpaceAgency devolve uma lista de missões de uma SpaceCompany.
- getNORADID devolve o primeiro NORAD ID disponivel.
- getPerson devolve uma row que corresponde a uma Person.
- getProgramsOfCompany devolve uma lista de Program de uma SpaceCompany

- getSpaceAgency devolve uma lista de SpaceCompany mediante os parametros de entrada.
- getSpacecraftFromSpaceCompany devolve uma lista de Spacecrafts que pertecem a uma certa SpaceCompany.
- getSpacecraftInvolmentsInMission devolve para uma Mission e Spacecraft uma row que contem a Payload e se essa Mission tiver o Launch dessa Spacecraft devolve também informações sobre o Launch. Se passar Craft_ID como -1 lista a row anterior para cada Spacecraft dessa Mission.
- getSpacecraftOfMission devolve uma lista de Spacecraft que fazem parte de uma Mission.
- getVehiclesFromSpaceCompany lista Vehicles que pertecem a uma SpaceCompany.
- numMissions devolve um escalar com o numero de Mission de um Astronaut.

Views

Usamos views para definir novas fontes de dados que funcionam quase como tabelas e são usadas dentro de algumas queries mais complexas.

 AstronautView.sql
 CeoView.sql
 SpaceCraftView.sql

- AstronautView é o join da tabela Person com Astronaut.
- CeoView é o join da tabela Person com Ceo.
- Spacecraftview é o Nome da Spacecraft , Owner da Spacecraft , Status da Spacecraft , Propusion da Spacecraft , nome do Launch Vehicle da SpaceCraft e nome do LaunchSite da Spacecraft

Cursores

Usámos cursores em algumas stored procedures com o proposito de iterar sobre todos os elementos de uma tabela e realizar certas operações sobre eles. Ex:

```
DECLARE @pid as int, @comp as int;

DECLARE C CURSOR FAST_FORWARD
FOR select Prog_ID,Company from Program

OPEN C

FETCH C into @pid, @comp

WHILE @@FETCH_STATUS = 0
BEGIN
    if @comp = @id --@id is the id of space company
        EXEC deleteProgram @pid;
    FETCH C into @pid , @comp;
END

CLOSE C;
DEALLOCATE C;
```

Este pedaço de código é usado no deleteSpaceCompany e tem como função invocar a deleteProgram dos Program de uma SpaceCompany.

Transações

Usamos transações sempre que iríamos alterar multiplas tabelas devido a insert , updates ou deletes ou quando o valor de uma variavel dependia se uma alteração de dados numa tabela era válida. EX:

```
CREATE PROCEDURE updateCompany
@ID INTEGER,
@Name varchar(100) ,
@Country varchar(100) ,
@Acronym varchar(10) ,
@Type varchar(16) ,
@Owner varchar(100)
AS
BEGIN
-- SET NOCOUNT ON added to prevent extra result sets from
-- interfering with SELECT statements.
SET NOCOUNT ON;

BEGIN TRY
BEGIN TRANSACTION

UPDATE SpaceCompany
SET [Name] = @Name , [Country] = @Country , Acronym = @Acronym
WHERE Comp_ID = @ID

IF @Type like 'Public'
BEGIN
IF NOT EXISTS ( Select * from PublicSpaceCompany where Comp_ID = @ID)
BEGIN
INSERT INTO PublicSpaceCompany( Comp_ID ,Gov )
VALUES
(@ID,@Owner)
DELETE from PrivateSpaceCompany where Comp_ID = @ID
END
END
ELSE IF @Type like 'Private'
BEGIN
IF NOT EXISTS (select * from PrivateSpaceCompany where Comp_ID = @ID)
BEGIN
IF EXISTS ( SELECT * from CEO where Per_ID = @Owner )
INSERT INTO PrivateSpaceCompany( Comp_ID ,CEO )
VALUES
(@ID,@Owner);
ELSE
RAISERROR('No such ceo',16,1);
DELETE from PublicSpaceCompany where Comp_ID = @ID
END
END
ELSE
RAISERROR('Not a valid Agency type.', 16, 1);
COMMIT
END TRY
BEGIN CATCH
IF (@@TRANCOUNT > 0)
BEGIN
```

```
ROLLBACK TRANSACTION
PRINT 'Error detected, all changes reversed'
END
END CATCH
END
GO
```

Este exemplo demonstra o uso de uma transação para certificarmos nos que ocorrem problemas de se uma das operações de insert , delete ou update falharem.

Conclusão

Em conclusão este trabalho abordou vários topicos sobre base de dados e fez-nos pensar e considerar diversos aspectos na criação deste tipos de sistemas. Devido ao numero elevado de tabelas que tinhamos na parte da interface não conseguimos fazer tudo o que queriamos , mas tentámos abordar uma grande variedade de tópicos que fizemos nas aulas práticas de modo a reforçar os conhecimentos lessionadas nesta cadeira.