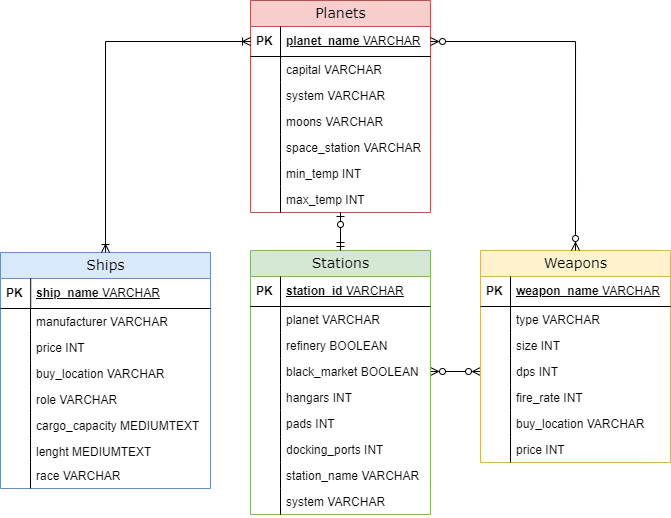
# Programming Assignment 2 Report

|  |  |
| --- | --- |
| Student(s): | Erik Borgström – [eb22](mailto:ilir.jusufi@lnu.se)3fe@student.lnu.se  Robin Bergvall – rb222vv@student.lnu.se |

## Project Idea

In this assignment we have designed a database that handles data regarding the game Star Citizen. It's a space game with emphasis on detail and creates a great opportunity for a database with many different ships, weapons, planets, and space stations. Our goal was for the user to be able to search for specific ships, planets and more, creating a tool to aid you when playing the game. Some data were taken from excel files found on the internet but most of the data was inputted manually from the Star Citizen wiki. https://starcitizen.tools/.

## StarCitizien design for schema



Started by creating the table planets with the planet\_name as a primary key seeing as no planet has the same name making it unique. Planets is basically the main table that has connection to all the other tables (Ships, Stations & Weapons).

Weapons holds weapon\_name as primary key as the weapon names is unique.

Stations holds station\_id as primary key seeing as the station id is unique to the stations.

Ships holds ship\_name as primary key seeing as the ship names are unique.

Planets is connected to Ships with a one or many relations on both directions seeing as:

*One planet will always sell at least one ship, but it can sell more.*

*One ship can be bought on one planet but also on many different planets.*

These are connected using buy\_location in Ships and planet\_name in Planets.

Planets is connected to Stations with a one and only one relation however stations are connected to planets with a zero or one relation seeing as:

*One planet will always have one and only one station.*

*One station can belong to a planet but does not always do so.*

These are connected using station\_id in Stations and space\_station in Planets.   
But also, planet in Stations and planet\_name in Planets.

Planets is connected to Weapons with a zero to many relations in both directions seeing as:

*One planet does not always sell a weapon, but it can sell many.*

*One weapon is not always sold by a planet, but it can be sold by many different planets.*

These are connected using buy\_location in Weapons and capital in Planets.

Weapons and Stations are connected using zero or many relations on both directions seeing as:

*One station does not always sell a weapon, but it can sell many.*

*One weapon is not always sold by a station, but it can be sold by many different stations.*

These are connected using buy\_location in Weapons and station\_id in Stations.

## Description of our SQL Queries

Below you will see our five queries and a description on how they were implemented.

Q1: **List full description on all the planets/ships/species/space stations/weapons.**

Our first option in the main menu was split into five choices in a sub menu. It gives the opportunity to list a full description on all the five tables depending on what you choose. Below shows a query to show a full description of all the planets.

CREATE VIEW planetsdesc

AS SELECT \*

FROM Planets

Q2: **List all weapons with a specific maximum cost.**

First an input takes a integer to set the maximum price of the weapons. Then a view is created that selects the weapon name, type and price in other words the three columns to show in the output. Then an aggregation is done to filter out the weapons with a greater price than entered.

choice = int(input("What’s the maximum price of the weapons to be shown? "))

CREATE VIEW selected\_weapons

AS SELECT weapon\_name, type, price

FROM Weapons

WHERE price < {choice}

Q3: **Show what ships you can buy at a specific planet.**

This is one of our more complex queries that's also a multi-relational query. It joins planets and ships with the condition that the ships buy location must be a partial string match to the planet’s capital. The other condition is that the planet name must be the same as the choice. When we do this, we connect the planet with the ships buy location through the planet’s capital. The partial string match is done since buy location can contain multiple locations for example: Area18, New Babbage.

choice = input("Type the planet name: ")

CREATE VIEW planetsview

AS SELECT planets.capital, ships.ship\_name, ships.price

FROM Planets

JOIN Ships

ON Ships.buy\_location LIKE concat(‘%’ + planets.capital, + ‘%’)

WHERE planets.planet\_name = ‘{choice}’

Q4: **Show the average price of the different weapon sizes.**

In the game the weapons are categorized in different sizes ranging from 1-6. This query takes the average price on the weapons and groups them based on size, making use of aggregation and grouping.

CREATE VIEW avg\_price

AS SELECT ROUND(AVG(price), 2), size, ships.price

FROM Weapons

GROUP BY size

Q5: **Show what weapons you can buy at a specific planet or station.**

The fifth query is separated into two queries depending on if you input a planet or a station. The two queries are much alike, and both creates different views that's saved to the cursor. This makes it so that the display part must be written only once. The query's uses buy location of the weapons and the planets/stations name to join the two tables. A partial string match is once again used since the column buy\_location can contain multiple values. The partial string match is done with the key word LIKE and the % signs after and before the station id meaning that the string could contain signs both before and after the station\_id.

choice = input("Type the Planet/Stations name: ")

CREATE VIEW weapons\_stations AS

SELECT stations.station\_name, weapons.weapon\_name, weapon.type, weapons.price

FROM Stations

INNER JOIN Weapons

ON weapons.buy\_location LIKE concat(‘%’ + station.station\_id, + ‘%’)

WHERE stations.station\_id = ‘{choice}’

## Discussion and Resources

The one main problem we ran into was using multiple strings inside a column like the following example:

En bild som visar text

Automatiskt genererad beskrivning

Here buy\_location has several locations where you can buy the weapon, we needed this to be connected to the Stations table and Planets table.

However just using normal queries only selected weapons where one station or planet was the sole seller of that weapon. Meaning it would list a weapon like this:



But not weapons like the first example.

The options we had was to try and fix this or make the weapons and ships be only sold at one location making it “incorrect” data.

In the end we managed to find the solution using LIKE concat query to be able to do a partial string match.

The project uses the following libraries:

* Mysql.connector to connect to the database
* Pandas to create and populate the table in a simple way but also to output good looking diagrams.
* OS to wait for an input to continue the program like assignment 1.

Please check README.md for installation details regarding the packages.

Source code: [Source Code](https://github.com/TheRobban02/Database-Mini-Project)

Video demonstration: [youtube/vimeo/... link]

# Changelog

|  |  |  |
| --- | --- | --- |
| Person | Task | Date |
| Robin | Planned project and set up git repository. | 2022-03-02 |
| Erik | Planned project and created csv files. | 2022-03-02 |
| Erik | Created schema. | 2022-03-03 |
| Robin | Made CreateDataBase(). | 2022-03-03 |
| Erik | Made PopulateTables(). | 2022-03-03 |
| Robin | Base implementation of mainMenu(). | 2022-03-03 |
| Erik | Base implementation of subMenu(). | 2022-03-07 |
| Robin | Created option 2 in mainMenu() query. | 2022-03-07 |
| Erik | Created all of the basic queries in the subMenu(). | 2022-03-07 |
| Robin | Created option 4 in mainMenu() query. | 2022-03-07 |
| Erik | Created option 3 in mainMenu() query. | 2022-03-07 |
| Robin | Created option 5 in mainMenu() query. | 2022-03-08 |
| Erik | Went through the code with flake8 linter. | 2022-03-08 |
| Robin | Wrote chapter 1, 3 & a little in 4 in report and README.md | 2022-03-08 |
| Erik | Wrote chapter 2 and discussion in 4 in report and made some fixes to the repository. | 2022-03-08 |