FTL: FASTER THAN LIGHT

Kevin Callahan

DBMGT

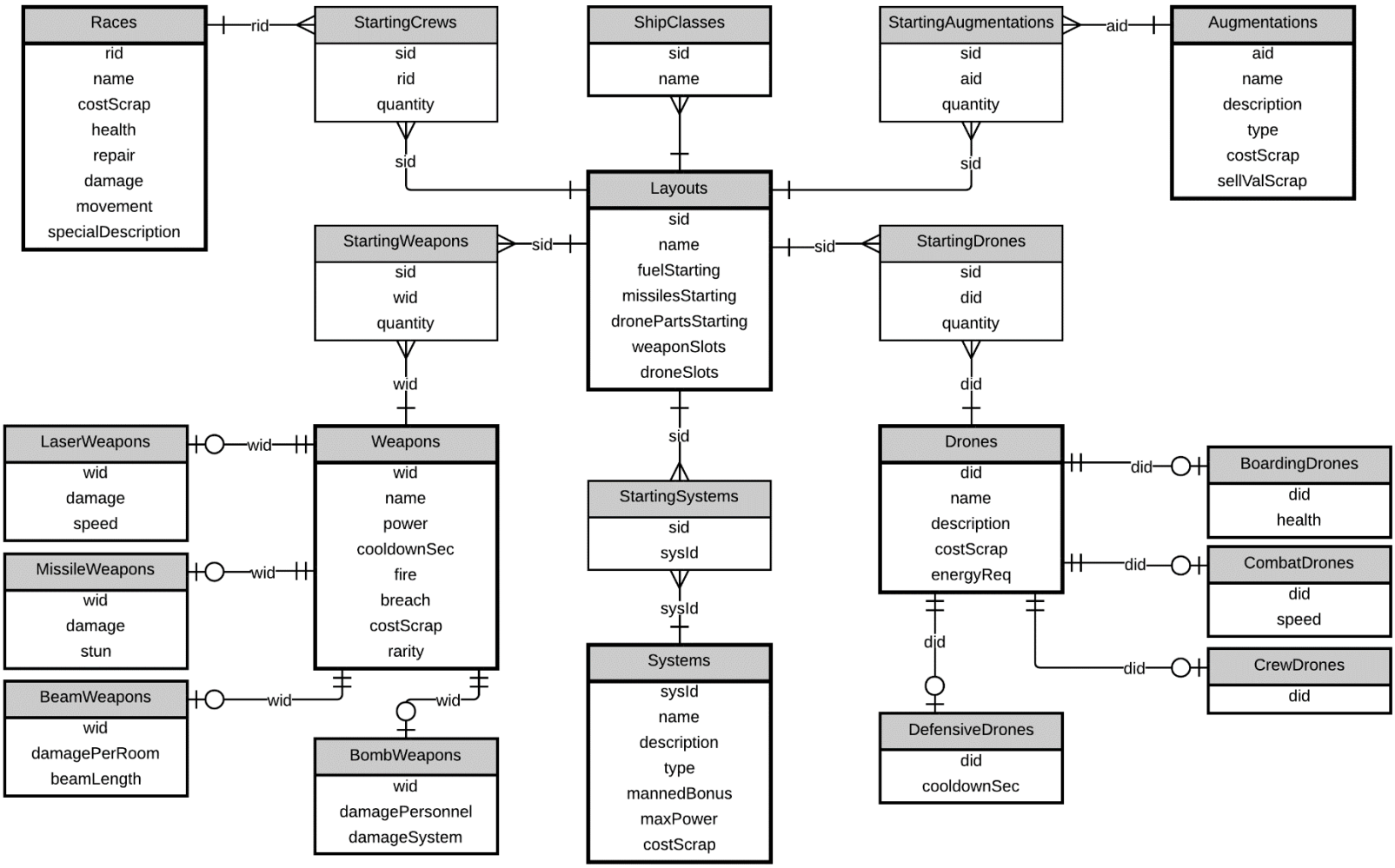
Contents

Executive Summary

This document describes the analysis, design and implementation of a database for an FTL: Faster Than Light information ship guide. This database is a way to see data on the weapons, drones, systems, etc. present in the game and the ship layouts that utilize these weapons. Possible use cases include gamers and those interested in FTL.

The following pages of this document illustrate the basic requirements need to implement an FTL guide. It will utilize an ER diagram to show the structure of the database. It will then detail all the tables, their functional dependencies, the SQL statements, and limited test data. Then the views, reports, and stored procedures for the database are shown. Finally security considerations and more information on the implementation and future enhancements that could be added to the database are discussed.

Entity Relationship Diagram



Tables

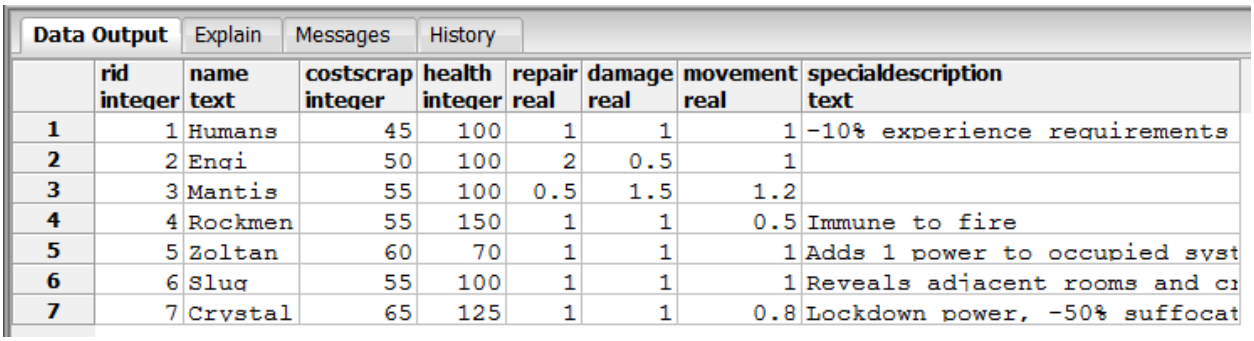
Races Table

The Races table holds all the available races a crew member can be in an FTL game.

```
CREATE TABLE races (
  rid                   integer not null,
  name                  text,
  costScrap             integer,
  health                integer,
  repair                real,
  damage                real,
  movement              real,
  specialDescription    text,
 primary key(rid)
);
```

Functional Dependencies: rid -> name, description, costScrap, health, repair, damage, movement, specialDescription

Sample Data:



Data Output | Explain | Messages | History

	rid integer	name text	costscrap integer	health integer	repair real	damage real	movement real	specialdescription text
1	1	Humans	45	100	1	1	1	-10% experience requirements
2	2	Engi	50	100	2	0.5	1	
3	3	Mantis	55	100	0.5	1.5	1.2	
4	4	Rockmen	55	150	1	1	0.5	Immune to fire
5	5	Zoltan	60	70	1	1	1	Adds 1 power to occupied syst
6	6	Slug	55	100	1	1	1	Reveals adjacent rooms and cr
7	7	Crystal	65	125	1	1	0.8	Lockdown power, -50% suffocat

Weapons Table

Contains a list of weapons and their stats and effects. Its attributes are inherited by its subtypes, laser, missile, beam, and bomb.

```
CREATE TABLE weapons (
  wid             integer not null,
  name            text,
  power           integer,
  cooldownSec     integer,
  fire            integer,
  breach          integer,
  costScrap       integer,
  rarity          integer,
 primary key(wid)
);
```

Functional Dependencies: wid -> name, power, cooldownSec, fire, breach, costScrap, rarity

Sample Data:

Data Output | Explain | Messages | History

	wid integer	name text	power integer	cooldownsec integer	fire integer	breach integer	costscrap integer	rarity integer
1	1	Basic Laser	1	10	1	0	20	0
2	2	Burst Laser II	2	12	1	0	80	4
3	3	Heavy Laser I	1	9	3	3	55	2
4	4	Heavy Ion	2	13	0	0	40	3
5	5	Ion Blast II	3	4	0	0	80	4
6	6	Artemis	1	11	1	1	38	0
7	7	Pike Beam	2	16	0	0	60	2
8	8	Healing Burst	1	18	0	0	40	3

LaserWeapons Table

Contains a list of laser weapons.

```
CREATE TABLE laserWeapons (
  wid           integer not null references weapons(wid),
  damage        text,
  speed         integer,
 primary key(wid)
);
```

Functional Dependencies: wid -> damage, speed

Sample Data:

Data Output | Explain | Message

	wid integer	damage text	speed integer
1	1	1	60
2	2	3x1	60
3	3	2	60
4	4	2 Ion	40
5	5	1 Ion	30

MissileWeapons Table

Contains a list of missile weapons.

```
CREATE TABLE missileWeapons (
  wid           integer not null references weapons(wid),
  damage        text,
  stun          integer,
 primary key(wid)
);
```

Functional Dependencies: wid -> damage, stun

Sample Data:

Data Output | Explain | Message

	wid integer	damage text	stun integer
1	6	2	0

BeamWeapons Table

Contains a list of beam weapons.

```
CREATE TABLE beamWeapons (
  wid            integer not null references weapons(wid),
  damagePerRoom  text,
  beamLength     integer,
 primary key(wid)
);
```

Functional Dependencies: wid -> damagePerRoom, beamLength

Sample Data:

Data Output | Explain | Messages | History

	wid integer	damageperroom text	beamlength integer
1	7	1	170

BombWeapons Table

Contains a list of bomb weapons.

```
CREATE TABLE bombWeapons (
  wid              integer not null references weapons(wid),
  damagePersonnel  integer,
  damageSystem     integer,
 primary key(wid)
);
```

Functional Dependencies: wid -> damagePersonnel , damageSystem

Sample Data:

Data Output | Explain | Messages | History

	wid integer	damagepersonnel integer	damagesystem integer
1	8	-150	0

Systems Table

Contains a list of systems available for ships.

```
CREATE TYPE sysType AS ENUM ('MainSystem', 'Subsystem');
CREATE TABLE systems (
  sysId           integer not null,
  name            text,
  description     text,
  type            sysType,
  mannedBonus     boolean,
  maxPower        integer,
  costScrap       integer,
 primary key(sysId)
);
```

Functional Dependencies: sysId -> name, description, type, mannedBonus, maxPower, costScrap

Sample Data:

Data Output | Explain | Messages | History

	sysid integer	name text	description text	type systype	mannedbonus boolean	maxpower integer	costscrap integer
1	1	Shields	Projects a protective ba	MainSystem	t	8	125
2	2	Medbay	Heals all crew members :	MainSystem	f	3	60
3	3	Sensors	Reveals the interior of	Subsystem	t	3	40

Drones Table

Contains a list of drone schematics available for ships. Its attributes are inherited by its subtypes, boarding, combat, crew, defensive.

```
CREATE TABLE drones (
  did             integer not null,
  name            text,
  description     text,
  costScrap       integer,
  energyReq       integer,
 primary key(did)
);
```

Functional Dependencies: did -> name, description, type, costScrap, energyReq

Sample Data:

Data Output | Explain | Messages | History

	did integer	name text	description text	costscrap integer	energyreq integer
1	1	Combat Drone Mark I	Continually attacks the	50	2
2	2	Anti-Personnel Drone	Attacks hostile boardin	60	2
3	3	System Repair Drone	Will seek out damaged s	30	1
4	4	Defense Drone Mark I	Shoots down incoming mi	50	2
5	5	Boarding Drone	Attacks crewmen on the	70	3

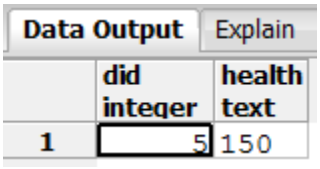
BoardingDrones Table

Contains a list of boarding drones.

```
CREATE TABLE boardingDrones (
  did          integer not null references drones(did),
  health       text,
 primary key(did)
);
```

Functional Dependencies: did -> health

Sample Data:



Data Output | Explain

	did integer	health text
1	5	150

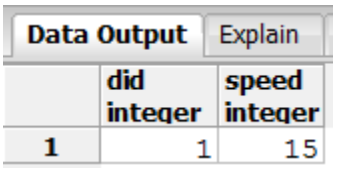
CombatDrones Table

Contains a list of combat drones.

```
CREATE TABLE combatDrones (
  did          integer not null references drones(did),
  speed        integer,
 primary key(did)
);
```

Functional Dependencies: did -> speed

Sample Data:



Data Output | Explain

	did integer	speed integer
1	1	15

CrewDrones Table

Contains a list of crew drones.

```
CREATE TABLE crewDrones (
  did          integer not null references drones(did),
 primary key(did)
);
```

Functional Dependencies: did ->

Sample Data:

Data Output

	did integer
1	2
2	3

DefensiveDrones Table

Contains a list of defensive drones.

```
CREATE TABLE defensiveDrones (
  did          integer not null references drones(did),
  cooldownSec  real,
 primary key(did)
);
```

Functional Dependencies: did -> cooldownSec

Sample Data:

Data Output | Explain | Mess

	did integer	cooldownsec real
1	4	1

Augmentations Table

Contains a list of augmentations available for ships.

```
CREATE TYPE augType AS ENUM ('Weapon', 'Defensive', 'FTL', 'Misc',
'Non-Purchasable');
CREATE TABLE augmentations (
  aid             integer not null,
  name            text,
  description     text,
  type            augType,
  costScrap       integer,
  sellValScrap    integer,
 primary key(aid)
);
```

Functional Dependencies: aid -> name, description, type, costScrap, sellValScrap

Sample Data:

Data Output | Explain | Messages | History

	aid integer	name text	description text	type augtype	costscrap integer	sellvalscrap integer
1	1	Engi Med-bot Dispersal	Heals the crew even whe	Non-Purchasable	0	30
2	2	Drone Reactor Booster	Onboard drones have the	Non-Purchasable	0	25

Layouts Table

Contains a list of available ships layouts.

```
CREATE TABLE layouts (
  sid                 integer not null,
  name                text not null unique,
  fuelStarting        integer,
  missileStarting     integer,
  dronePartsStarting  integer,
  weaponSlots         integer,
  droneSlots          integer,
 primary key(sid)
);
```

Functional Dependencies: sid -> name, fuelStarting, missileStarting, dronePartsStarting, weaponSlots, droneSlots

Sample Data:

Data Output | Explain | Messages | History

	sid integer	name text	fuelstarting integer	missilestarting integer	dronepartsstarting integer	weaponslots integer	droneslots integer
1	1	The Kestrel	16	8	0	4	2
2	2	Red-Tail	16	5	0	4	2
3	3	The Torus	16	0	15	3	3
4	4	The Vortex	16	0	6	3	3

ShipClasses Table

Contains a list of ship classes and their associated layouts.

```
CREATE TABLE shipClasses (
  sid           integer not null references layouts(sid),
  name          text not null,
 primary key(sid)
);
```

Functional Dependencies: sid -> name

Sample Data:

Data Output | Explain | Messages

	sid integer	name text
1	1	Kestrel Cruiser
2	2	Kestrel Cruiser
3	3	Engi Cruiser
4	4	Engi Cruiser

StartingWeapons Table

Contains a list of starting weapons for each layout.

```
CREATE TABLE startingWeapons (
  sid       integer not null references layouts(sid),
  wid       integer not null references weapons(wid),
  quantity  integer not null,
 primary key(sid, wid)
);
```

Functional Dependencies: sid, wid -> quantity

Sample Data:

Data Output | Explain | Message

	sid integer	wid integer	quantity integer
1	1	2	1
2	1	6	1
3	2	1	4
4	3	5	1
5	4	4	1
6	4	3	1

StartingCrews Table

Contains a list of starting crew for each layout.

```
CREATE TABLE startingCrews (
  sid       integer not null references layouts(sid),
  rid       integer not null references races(rid),
  quantity  integer not null,
 primary key(sid, rid)
);
```

Functional Dependencies: sid, rid -> quantity

Sample Data:

Data Output | Explain | Messages

	sid integer	rid integer	quantity integer
1	1	1	3
2	2	1	2
3	2	3	1
4	2	5	1
5	3	2	2
6	3	1	1
7	4	2	1

StartingAugmentation Table

Contains a list of starting augmentations for each layout.

```
CREATE TABLE startingAugmentation (
  sid       integer not null references layouts(sid),
  aid       integer not null references augmentations(aid),
  quantity  integer not null,
 primary key(sid, aid)
);
```

Functional Dependencies: sid, aid -> quantity

Sample Data:

Data Output | Explain | Message

	sid integer	aid integer	quantity integer
1	2	1	1
2	2	2	1

StartingDrones Table

Contains a list of starting drone schematics for each layout.

```
CREATE TABLE startingDrones (
  sid       integer not null references layouts(sid),
  did       integer not null references drones(did),
  quantity  integer not null,
 primary key(sid, did)
);
```

Functional Dependencies: sid, did -> quantity

Sample Data:

Data Output | Explain | Message

	sid integer	did integer	quantity integer
1	2	1	1
2	2	2	1
3	2	3	2

StartingSystems Table

Contains a list of starting drone schematics for each layout.

```
CREATE TABLE startingSystems (
  sid   integer not null references layouts(sid),
  sysId   integer not null references systems(sysId),
 primary key(sid, sysId)
);
```

Functional Dependencies: sid, sysId ->

Sample Data:

Data Output | Explain

	sid integer	sysid integer
1	1	1
2	1	2
3	1	3
4	2	1
5	2	2
6	2	3
7	3	1
8	3	2
9	3	3
10	4	1
11	4	2
12	4	3

Views

Drones Complete

Displays the complete drone schematics information.

```
CREATE VIEW DronesComplete
AS
SELECT d.did, d.name, d.description, d.costScrap, d.energyReq,
bd.health, cd.speed, dd.cooldownSec
FROM drones d, boardingDrones bd, combatDrones cd, crewDrones crd,
defensiveDrones dd
WHERE d.did = bd.did OR d.did = cd.did OR d.did = crd.did Or d.did =
dd.did
ORDER BY d.name ;
```

Sample Data:

Data Output | Explain | Messages | History

	did integer	name text	description text	costscrap integer	energyreq integer	health text	speed integer	cooldownsec real
1	2	Anti-Personnel Drone	Attacks hostile boarding	60	2	150	15	1
2	5	Boarding Drone	Attacks crewmen on the ta	70	3	150	15	1
3	5	Boarding Drone	Attacks crewmen on the ta	70	3	150	15	1
4	1	Combat Drone Mark I	Continually attacks the e	50	2	150	15	1
5	1	Combat Drone Mark I	Continually attacks the e	50	2	150	15	1
6	4	Defense Drone Mark I	Shoots down incoming miss	50	2	150	15	1
7	4	Defense Drone Mark I	Shoots down incoming miss	50	2	150	15	1
8	3	System Repair Drone	Will seek out damaged syst	30	1	150	15	1

Weapons Complete

Displays the complete weapons information.

```
CREATE VIEW WeaponsComplete
AS
SELECT DISTINCT w.wid, w.name, w.power, w.cooldownSec, w.fire,
w.breach, w.costScrap, w.rarity, lw.damage, lw.speed, mw.stun,
bw.damagePerRoom, bw.beamLength, bow.damagePersonnel, bow.damageSystem
FROM weapons w, laserWeapons lw, missileWeapons mw, beamWeapons bw,
bombWeapons bow
WHERE w.wid = lw.wid OR w.wid = mw.wid OR w.wid = bw.wid OR w.wid =
bow.wid
ORDER BY w.wid ;
```

Sample Data:

Data Output | Explain | Messages | History

	wid integer	name text	power integer	cooldownsec integer	fire integer	breach integer	costscrap integer	rarity integer	damage text	speed integer	stun integer	damageperroom text	bea inte
1	1	Basic Laser	1	10	1	0	20	0	1	60	0	1	
2	2	Burst Laser II	2	12	1	0	80	4	3x1	60	0	1	
3	3	Heavy Laser I	1	9	3	3	55	2	2	60	0	1	
4	4	Heavy Ion	2	13	0	0	40	3	2 Ion	40	0	1	
5	5	Ion Blast II	3	4	0	0	80	4	1 Ion	30	0	1	
6	6	Artemis	1	11	1	1	38	0	1	60	0	1	

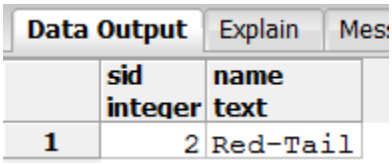
Queries

Layouts with Basic Laser

Displays the layouts with the basic laser.

```
SELECT l.sid, l.name
FROM layouts l, startingWeapons sw, weapons w
WHERE l.sid = sw.sid AND sw.wid = w.wid AND w.name = 'Basic Laser'
ORDER BY l.sid ;
```

Sample Data:



Data Output | Explain | Mes

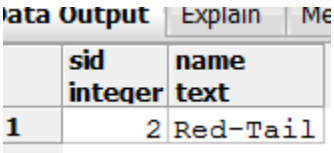
	sid integer	name text
1	2	Red-Tail

Layouts with Anti-Personnel Drone

Displays the layouts with the anti-personnel drone.

```
SELECT l.sid, l.name
FROM layouts l, startingDrones sd, drones d
WHERE l.sid = sd.sid AND sd.did = d.did AND d.name = 'Anti-Personnel
Drone'
ORDER BY l.sid ;
```

Sample Data:



ata Output | Explain | Me

	sid integer	name text
1	2	Red-Tail

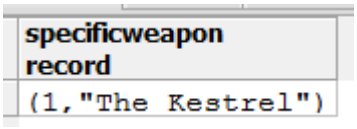
Stored Procedures

SpecificWeapon()

Displays the layouts with the given weapon.

```
CREATE OR REPLACE FUNCTION SpecificWeapon(weaponName text)
RETURNS TABLE("Layouts Id" integer, "Layouts Name" text)AS $$
BEGIN
RETURN QUERY SELECT l.sid AS "Layouts Id", l.name AS "Layouts Name"
FROM layouts l, startingWeapons sw, weapons w
WHERE w.name = weaponName AND l.sid = sw.sid AND sw.wid = w.wid
ORDER BY l.sid;
END;
$$ LANGUAGE plpgsql;
```

Sample Data: SELECT SpecificWeapon('Burst Laser II');



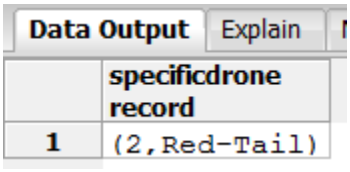
specificweapon record
(1,"The Kestrel")

SpecificDrone()

Displays the layouts with the given drone.

```
CREATE OR REPLACE FUNCTION SpecificDrone(droneName text)
RETURNS TABLE("Layouts Id" integer, "Layouts Name" text)AS $$
BEGIN
RETURN QUERY SELECT l.sid AS "Layouts Id", l.name AS "Layouts Name"
FROM layouts l, startingDrones sd, drones d
WHERE d.name = droneName AND l.sid = sd.sid AND sd.did = d.did
ORDER BY l.sid;
END;
$$ LANGUAGE plpgsql;
```

Sample Data: SELECT SpecificDrone('System Repair Drone');



Security

There are two types of users identified for this database.

1. The database admin who can change, update, and maintain the database.

```
CREATE ROLE db_admin
GRANT SELECT, INSERT, UPDATE, ALTER
ON ALL TABLES IN SCHEMA PUBLIC
TO db_admin
```

2. The public user who can see the database and perform queries on it.

```
CREATE ROLE public
GRANT SELECT
ON ALL TABLES IN SCHEMA PUBLIC
TO public
```

Implementation Notes / Known Problems / Future Enhancements

- In order for this system to be truly useful, it must first be populated with the complete data, by the admins. This become a problem as it would be a large undertaking and functionality would be limited until it is completed.
- The database is also limited to the standard release of FTL, new content added by an expansion would require addition tables to be add. An example of this can be seen in the recent update which introduced an addition weapon type.
- To further enhance the database with addition features, functionality could be added that would allow public users to add playthroughs or "runs," to the database. This could allow for stat tracking.