Sovereignty Database

The game’s database

Anthony Cali

12/2/2013

Table of Contents

pg 3. Executive Summary

pg 4. Entity Relation Diagram

pg 5. Create Statements and Functional Dependencies

pg12. Views

pg14. Can be added

pg15. Improvements

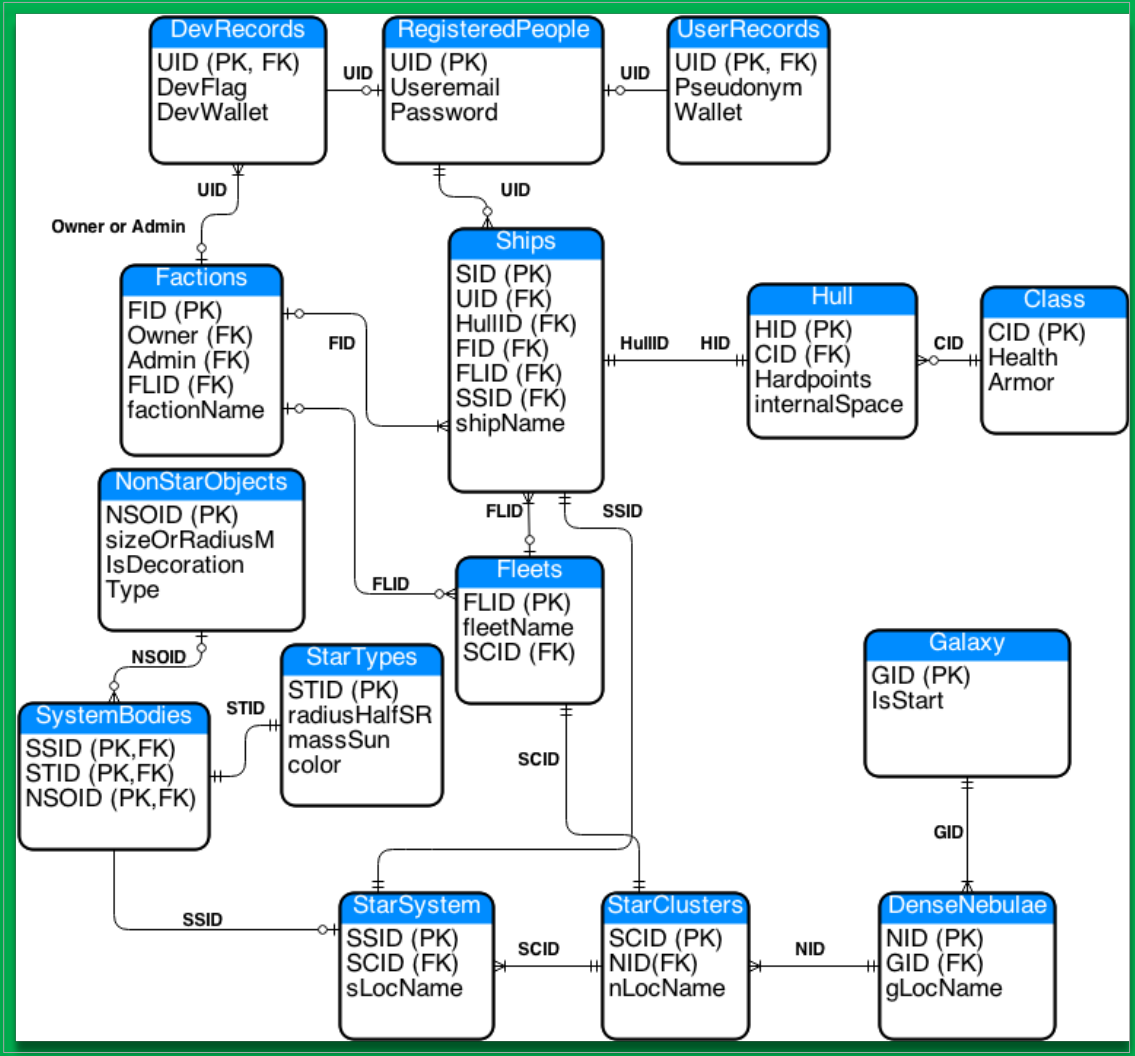
Executive Summary

This database is built for the purpose of streamlining the implementation of the game Sovereignty. The game idea is a massively-online-multiplayer real-time-strategy. To prevent server overloading due to the game's scale and to provide a resilient backend a database must be deployed. This implementation of Sovereignty is an early build and as a result only contains the most minimal game elements. The actual game will be built on the database. This presentation is the intended server-side backend. The database coordinates the following information:

- Registered users
- Developers and players
- Ships, fleets, and factions
- The game universe

This is all implemented in a PostgreSQL v 9.2 server.

Entity-Relation Diagram



Create Statements and Functional Dependencies

```
drop table if exists RegisteredUsers;
create table if not exists RegisteredUsers (
  UID serial not null primary key,
  Useremail varchar(20) not null,
  Password varchar(20) not null
);
```

Functional Dependencies: UID -> Useremail, Password

```
drop table if exists DevRecords;
create table if not exists DevRecords (
  UID int not null primary key references RegisteredUsers(UID),
  DevFlag boolean default(true),
  DevWallet decimal default(10000000.00)
);
```

Functional Dependencies: UID -> DevFlag, DevWallet

```
drop table if exists UserRecords;
```

```
create table if not exists UserRecords (
  UID int not null primary key references RegisteredUsers(UID),
  Pseudonym  varchar(50) not null,
  Wallet decimal default(100000.00)
);
```

Functional Dependencies: UID -> Pseudonym, Wallet

```
drop table if exists Class;
create table if not exists Class (
  CID serial not null primary key,
  Health int not null,
  Armor int not null
);
```

Functional Dependencies: CID -> Health, Armor

```
drop table if exists Hull;
create table if not exists Hull (
  HID serial not null primary key,
```

```
  CID int not null references Class(CID),
  Hardpoints int not null,
  internalSpace int not null
);
```

Functional Dependencies: HID -> Hardpoints, internalSpace

```
drop table if exists Galaxy;
create table if not exists Galaxy (
  GID serial not null primary key,
  isStart boolean default(false)
);
```

Functional Dependencies: GID -> isStart

```
drop table if exists DenseNebulae;
create table if not exists DenseNebulae (
  NID serial not null primary key,
  GID int not null references Galaxy(GID),
  gLocName text default ('Unknown')
```

```
);
```
Functional Dependencies: NID ->GID, gLocName
```
drop table if exists StarClusters;
create table if not exists StarClusters (
  SCID serial not null primary key,
  NID int not null references DenseNebulae(NID),
  nLocName text default ('Unknown Cluster')
);
```
Functional Dependencies: SCID -> NID, nLocName
```
drop table if exists StarSystem;
create table if not exists StarSystem (
  SSID serial not null primary key,
  SCID int not null references StarClusters(SCID),
  sLocName text default ('Unknown System')
);
```
Functional Dependencies: SSID -> SCID, sLocName

```
drop table if exists Fleets;
create table if not exists Fleets (
  FLID serial not null primary key,
  fleetName text not null,
  SCID int references StarClusters(SCID)
);
```

Functional Dependencies: FLID -> fleetName, SCID

```
drop table if exists Factions;
create table if not exists Factions (
  FID serial not null primary key,
  Owner int not null references DevRecords(UID),
  Admin int references DevRecords(UID),
  FLID int references Fleets(FLID),
  factionName text not null
);
```

Functional Dependencies: FID ->Owner, Admin, FLID, factionName

```
drop table if exists Ships;
create table if not exists Ships (
  SID serial not null primary key,
  UID int not null references RegisteredUsers(UID),
  HullID int not null references Hull(HID),
  FID int references Factions(FID),
  FLID int references Fleets(FLID),
  SSID int not null references StarSystem(SSID),
  shipName varchar(50) not null
);
```

Functional Dependencies: SID ->UID, HullID, FID, FLID, SSID, shipName

```
drop table if exists NonStarObjects;
create table if not exists NonStarObjects (
  NSOID serial not null primary key,
  sizeOrRadiusM numeric not null,
  IsDecoration boolean default(false),
```

```
  Type text check (Type = 'planet' or Type = 'asteroid belt'
                   or Type = 'space debris' or Type = 'moon'
                   or Type = 'station')
);
```

Functional Dependencies: NSOID->sizeOrRadiusM, IsDecoration, Type

```
drop table if exists StarTypes;
create table if not exists StarTypes (
  STID serial not null primary key,
  radiusHalfSR numeric not null default(1),
  massSun numeric not null default(1),
  color text check (color = 'white' or color = 'blue'
                    or color = 'red' or color = 'yellow'
                    or color = 'orange')
);
```

Functional Dependencies: STID->radiusHalfSR, massSun, color

```
drop table if exists SystemBodies;
```

```
create table if not exists SystemBodies(
  SSID int references StarSystem(SSID),
  STID int references StarTypes(STID),
  NSOID int references NonStarObjects(NSOID),
  primary key(SSID, STID, NSOID)
);
```

Functional Dependencies: (STID, STID, NSOID)->

Views

Here are a few view definitions:

```
create view “player ships” as
select u. Pseudonym, s.shipName,h.Hardpoints, h.internalSpace, c.Health, c.Armor
  from UserRecords u, Ships s, Hull h, Class c
where u.UID = s.UID
   and s.HullID = h.HID
   and h.CID = c.cid
group by s.SID asc


create view “ships in fleet” as
select s.shipName, f.fleetName
  from Fleets f, Ships s
where f.flid = s.flid
```

Can be added

- Stored procedures
- Security
- More views
- A better grasp on the data
- Sample data

Improvements

- Security
- Weapons
- Modules (engines, quarters etc)
- A better location system
- Client-side database