CMPT 308N Database Management

By: Brianna Lee

Table of Contents

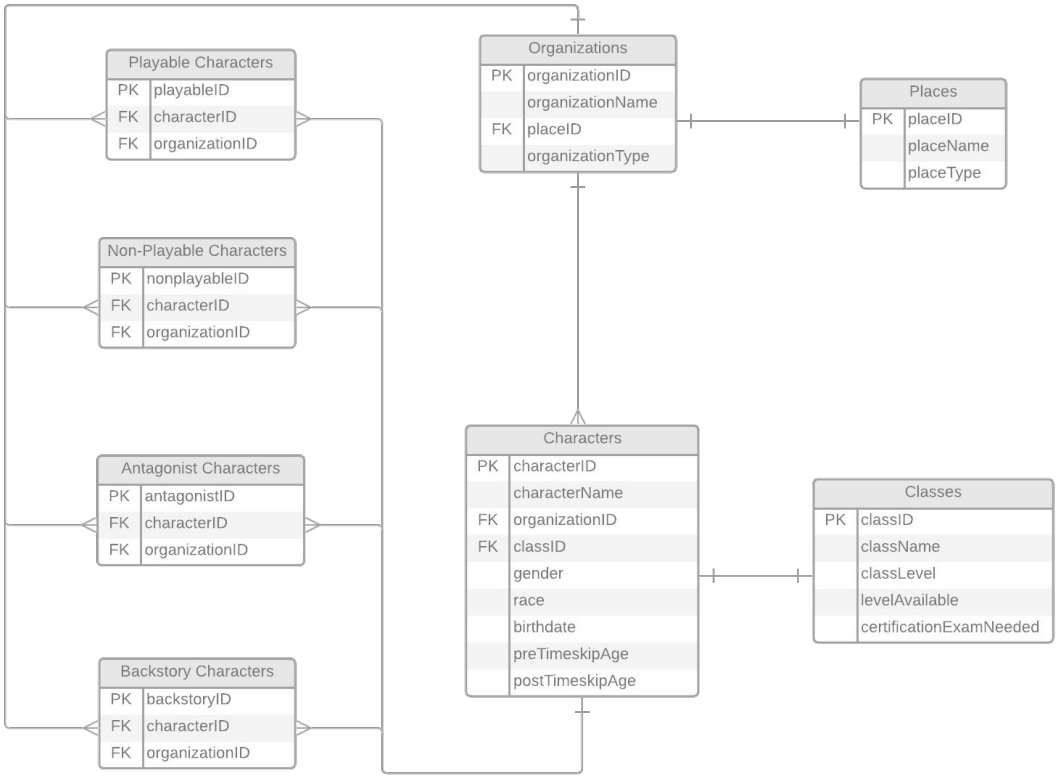
Executive Summary

Fire Emblem: Three Houses is a tactical role-playing game that is set on the continent Fóldan that is divided between three ruling nations. These nations are all connected to one another the Garreg Mach Monastery, which houses a church and an Officers' School for students from these nations. By taking the role of Byleth, who is a former mercenary with a strange past and the academy's newest professor, the player now has to choose a house to lead and guide the students through a series of battles.

This presentation outlines the design along with the implementation of a database that holds information about the places, organizations, classes, and characters in Fóldan. First, the ER diagram of the database will be presented, followed by all of the SQL create statements of SQL types, then a description of each table and the SQL create statements. The data for the table will also be provided. Afterward, the view tables, reports, and interesting queries, stored procedures, and triggers in the database are shown, along with the data output. Then the security privileges and the roles' descriptions are provided. Lastly, implementation notes, any known problems, and future ideas for improvements are discussed.

The database is intended for the player to learn more about the game on a closer level before they could start the game. This database will allow the player to easily see a list of the characters in the game as well as the possible classes that they could obtain. The player will also be able to see which characters they may be in a house with from the queries. The objective of the database is to provide the player with a functional database that lets them understand the game a bit better.

ER Diagram


Playable Characters
PK playableID
FK characterID
FK organizationID
Organizations
PK organizationID
organizationName
FK placeID
organizationType
Places
PK placeID
placeName
placeType
Non-Playable Characters
PK nonplayableID
FK characterID
FK organizationID
Antagonist Characters
PK antagonistID
FK characterID
FK organizationID
Characters
PK characterID
characterName
FK organizationID
FK classID
gender
race
birthdate
preTimeskipAge
postTimeskipAge
Classes
PK classID
className
classLevel
levelAvailable
certificationExamNeeded
Backstory Characters
PK backstoryID
FK characterID
FK organizationID

Types

locations: Places could be identified by four categories — A continent, a nation, an ancient Civilization, or a school

```
create type locations as enum('Continent', 'Nation', 'Ancient Civilization','School');
```

needed: Whether a Class needs a certification exam in order to advance to the next level — Yes or No

```
create type needed as enum('Yes', 'No');
```

levelOfClasses: Classes can be grouped into nine different levels — Unique, Beginner, Intermediate, Advanced, Special, Master, Non-Playable, Monster, and None

```
create type levelOfClasses as enum('Unique', 'Beginner', 'Intermediate', 'Advanced', 'Special', 'Master',
'Non-Playable', 'Monster', 'None');
```

groups: Organizations can be categorized into six categories, — A house, a religious group, a nation, a school, an antagonist group, or an ancient civilization

```
create type groups as enum('House', 'Religious Group', 'Nation', 'School', 'Antagonist Group', 'Ancient
Civilization');
```

genderIdentity: Characters can be two genders — Male or Female

```
create type genderIdentity as enum('Male', 'Female');
```

raceIdentity: There are seven different races in the universe — Humans, Dragons (Earth, Light, Sky, Water, and Wind), and Goddesses

```
create type raceIdentity as enum('Human', 'Earth Dragon', 'Light Dragon', 'Sky Dragon', 'Water Dragon', 'Wind
Dragon', 'Goddess');
```

Tables

Places

The Places Table includes all the places that are mentioned in the game.

```
CREATE TABLE Places (
  placeID      int not null,
  placeName    text,
  placeType    locations,
primary key(placeID)
);
```

Functional Dependencies:
placeID → placeName, placeType

	placeid [PK] integer	placename text	placetype locations
1	1	Foldan	Continent
2	2	Holy Kingdom of Faerghus	Nation
3	3	Leicester Alliance	Nation
4	4	Adrestian Empire	Nation
5	5	Agartha	Ancient Civilization
6	6	Nabatea	Ancient Civilization
7	7	Garreg Mach Monastery	School

Tables

Classes

The Classes Table includes all the possible classes that a character can achieve.

```
CREATE TABLE Classes (
   classID                  int not null,
   className                text,
   classLevel               text,
   levelAvailable           int,
   certificationExamNeeded  needed,
 primary key(classID)
);
```

Functional Dependencies:
classID → className, classLevel, levelAvailable, certificationExamNeeded

	classid [PK] integer	classname text	classlevel levelofclasses	levelavailable integer	certificationexamneeded needed
1	1	Commoner	Unique	0	No
2	2	Noble	Unique	0	No
3	3	Dancer	Unique	0	No
4	4	Myrmidon	Beginner	5	Yes
5	5	Soldier	Beginner	5	Yes
6	6	Fighter	Beginner	5	Yes
7	7	Monk	Beginner	5	Yes
8	8	Mage	Intermediate	10	Yes
9	9	Priest	Intermediate	10	Yes
10	10	Lord	Intermediate	10	Yes
11	11	Swordmaster	Advanced	20	Yes
12	12	Fortress Kni…	Advanced	20	Yes

Tables

Organizations

The Organizations Table includes all of the organizations that a character can be affiliated.

```
CREATE TABLE Organizations (
   organizationID             int not null,
   organizationName           text,
   placeID                    int,
   organizationType           groups,
 primary key(organizationID),
 foreign key(placeID) references Places(placeID)
);
```

Functional Dependencies:
organizationID → organizationName, placeID
organizationType

	organizationid [PK] integer	organizationname text	placeid integer	organizationtype groups
1	1	Black Eagles	4	House
2	2	Blue Lions	2	House
3	3	Golden Deer	3	House
4	4	Holy Kingdom of Faergh...	2	Nation
5	5	Adrestian Empire	4	Nation
6	6	Leicester Alliance	3	Nation
7	7	Garreg Mach Monastery	7	School
8	8	Church of Seiros	7	Religious Group
9	9	Those Who Slither in th...	5	Antagonist Group
10	10	Children of the Goddess	6	Ancient Civilization

Tables

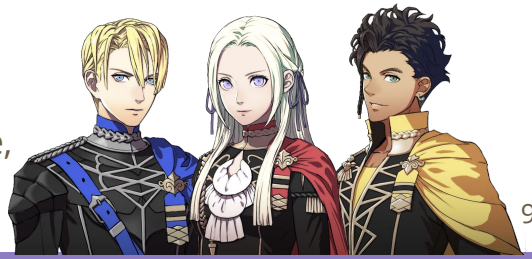
Characters

The Characters Table includes all of the characters with basic information about each character.

```
CREATE TABLE Characters (
   characterID         int not null,
   characterName       text,
   organizationID      int,
   classID             int,
   gender              genderIdentity,
   race                text,
   birthdate           date,
   preTimeskipAge      int,
   postTimeskipAge     int,
 primary key(characterID),
 foreign key(organizationID) references
      Organizations(organizationID),
 foreign key(classID) references Classes(classID)
);
```

	characterid [PK] integer	charactername text	organizationid integer	classid integer	gender genderidentity	race raceidentity	birthdate date	pretimeskipage integer	posttimeskipage integer
1	1	Byleth	8	11	Female	Human	1159-11-16	21	26
2	2	Edalgard	1	12	Female	Human	1162-06-22	18	24
3	3	Hubert	1	2	Male	Human	1159-04-17	21	26
4	4	Ferdinand	1	2	Male	Human	1162-04-30	18	24
5	5	Dimitri	2	13	Male	Human	1159-12-20	18	24
6	6	Felix	2	2	Male	Human	1162-02-20	18	23
7	7	Mercedes	2	1	Female	Human	1157-05-27	23	29
8	8	Claude	3	14	Male	Human	1162-07-24	18	24
9	9	Lysithea	3	8	Female	Human	1164-02-28	16	21
10	10	Hilda	3	15	Female	Human	1161-02-03	19	24
11	11	Seteh	8	14	Male	Earth Dragon	0086-12-27	1000	1000
12	12	Flayn	8	9	Female	Light Dragon	0086-07-12	1000	1000

Functional Dependencies:
characterID → characterName, organizationID, classID, gender, race, birthdate, preTimeskipAge, postTimeskipAge



Tables

Playable Characters

The Playable Characters Table is a subtype of Characters and contains all the characters that the player is able to play as, along with the organization the character is affiliated.

```
CREATE TABLE PlayableCharacters (
   playableID     int not null,
   characterID    int,
   organizationID int,
 primary key(playableID),
 foreign key(characterID) references Characters(characterID),
 foreign key(organizationID) references Organizations(organizationID)
);
```

Functional Dependencies:
playableID → characterID, organizationID

	playableid [PK] integer	characterid integer	organizationid integer
1	1	1	7
2	2	2	1
3	3	3	1
4	4	4	1
5	5	5	2
6	6	6	2
7	7	7	2
8	8	8	3
9	9	9	3
10	10	10	3
11	11	11	8
12	12	12	8

Tables

Non-Playable Characters

The Non-Playable Characters Table is a subtype of Characters and contains all the characters that the player cannot play as, along with the organization the character is affiliated.

```
CREATE TABLE NonplayableCharacters (
   nonplayableID    int not null,
   characterID      int,
   organizationID   int,
 primary key(nonplayableID),
 foreign key(characterID) references Characters(characterID),
 foreign key(organizationID) references
      Organizations(organizationID)
);
```

	nonplayableid [PK] integer	characterid integer	organizationid integer
1	1	13	5
2	2	14	5
3	3	15	7
4	4	31	8
5	5	32	4

Functional Dependencies:
nonplayableID → characterID, organizationID

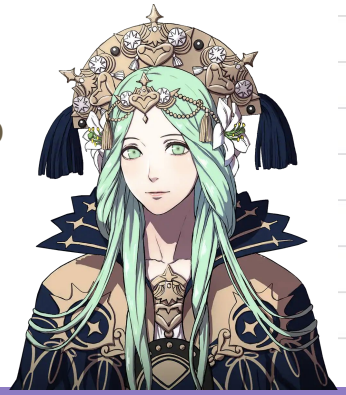
Tables

Antagonist Characters

The Antagonist Characters Table is a subtype of Characters and contains all the characters that the antagonists, along with the organization the character is affiliated.

```
CREATE TABLE AntagonistCharacters (
   antagonistID       int not null,
   characterID          int,
   organizationID int,
 primary key(antagonistID),
 foreign key(characterID) references Characters(characterID),
 foreign key(organizationID) references
      Organizations(organizationID)
);
```

Functional Dependencies:
antagonistID → characterID, organizationID



	antagonistid [PK] integer	characterid integer	organizationid integer
1	1	16	5
2	2	17	5
3	3	18	6
4	4	19	6
5	5	20	6
6	6	21	6
7	7	22	8
8	8	23	9
9	9	24	9
10	10	25	9
11	11	26	10
12	12	27	10

Tables

Backstory Characters

The Backstory Characters Table is a subtype of Characters and contains all the characters that provide the backstory to the main storyline, along with the organization the character is affiliated.

```
CREATE TABLE AntagonistCharacters (
   antagonistID       int not null,
   characterID          int,
   organizationID int,
 primary key(antagonistID),
 foreign key(characterID) references
      Characters(characterID),
 foreign key(organizationID) references
      Organizations(organizationID)
);
```

Functional Dependencies:
backstoryID → characterID, organizationID

	backstoryid [PK] integer	characterid integer	organizationid integer
1	1	28	5
2	2	29	4
3	3	30	6
4	4	31	8

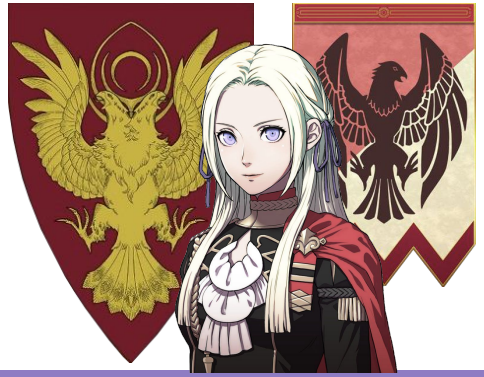
Views

Organization Place

View 1: Returns the organizationID, organizationName, and placeName of all the organizations

```
CREATE OR REPLACE VIEW OrganizationPlace as
select o.organizationID, o.organizationName, p.placeName
from Organizations o full outer join Places p on
      o.placeID = p.placeID;

select *
from CharacterOrganizationClass
order by characterID ASC;
```



	organizationid integer	organizationname text	placename text
1	1	Black Eagles	Adrestian Empire
2	2	Blue Lions	Holy Kingdom of Faerghus
3	3	Golden Deer	Leicester Alliance
4	4	Holy Kingdom of Faergh…	Holy Kingdom of Faerghus
5	5	Adrestian Empire	Adrestian Empire
6	6	Leicester Alliance	Leicester Alliance
7	7	Garreg Mach Monastery	Garreg Mach Monastery
8	8	Church of Seiros	Garreg Mach Monastery
9	9	Those Who Slither in th…	Agartha
10	10	Children of the Goddess	Nabatea

Views

Characters Organization Place

View 2: Returns the characterID, characterName, organizationName, and className of all the characters in the game

```
CREATE OR REPLACE VIEW CharacterOrganizationClass as
select ch.characterID, ch.characterName,
      o.OrganizationName, c.className
from Characters ch full outer join Organizations o on
            ch.organizationID = o.organizationID
      full outer join Classes c on
            ch.classID = c.classID;

select *
from CharacterOrganizationClass
order by characterID ASC;
```

	characterid integer	charactername text	organizationname text	classname text
1	1	Byleth	Church of Seiros	Swordmaster
2	2	Edalgard	Black Eagles	Fortress Knight
3	3	Hubert	Black Eagles	Noble
4	4	Ferdinand	Black Eagles	Noble
5	5	Dimitri	Blue Lions	Paladin
6	6	Felix	Blue Lions	Noble
7	7	Mercedes	Blue Lions	Commoner
8	8	Claude	Golden Deer	Wyvern Rider
9	9	Lysithea	Golden Deer	Mage
10	10	Hilda	Golden Deer	Warrior
11	11	Seteh	Church of Seiros	Wyvern Rider
12	12	Flayn	Church of Seiros	Priest

Reports and Interesting Queries

Characters Older than 1000 Years Old

Query 1: Get the characterID, characterName, gender, race, preTimeskipAge, and PostTimeskipAge of all the characters over the age of 1000

```
select characterID, characterName, gender, race, preTimeskipAge, postTimeskipAge
from Characters
where preTimeskipAge >= 1000 and postTimeskipAge >= 1000;
```

	characterid [PK] integer	charactername text	gender genderidentity	race raceidentity	pretimeskipage integer	posttimeskipage integer
1	11	Seteh	Male	Earth Dragon	1000	1000
2	12	Flayn	Female	Light Dragon	1000	1000
3	22	Rhea	Female	Sky Dragon	1222	1222
4	26	Indech	Male	Water Dragon	1000	1000
5	27	Macuil	Male	Wind Dragon	1000	1000
6	31	Sothis	Female	Goddess	1250	1250

Reports and Interesting Queries

Characters in the Black Eagle House

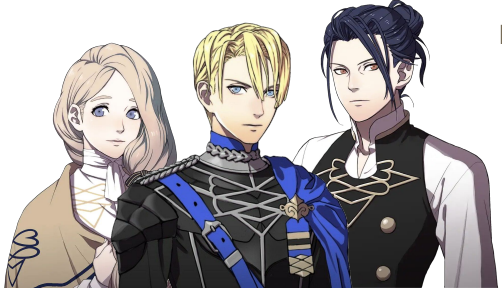
Query 2: Get the characterID, characterName, organizationName, className, gender, race, preTimeskipAge, and PostTimeskipAge of all the characters over in the Black Eagles House

```
select ch.characterID, ch.characterName, o.organizationName, c.className, ch.gender, ch.race,
ch.preTimeskipAge, ch.postTimeskipAge
from Characters ch left join Organizations o on (ch.organizationID = o.organizationID)
                           left join Classes c on (ch.classId = c.classID)
where ch.organizationID = '001';
```

	characterid integer	charactername text	organizationname text	classname text	gender genderidentity	race raceidentity	pretimeskipage integer	posttimeskipage integer
1	2	Edalgard	Black Eagles	Fortress Knight	Female	Human	18	24
2	3	Hubert	Black Eagles	Noble	Male	Human	21	26
3	4	Ferdinand	Black Eagles	Noble	Male	Human	18	24

Reports and Interesting Queries

Characters in the Blue Lion House



Query 3: Get the characterID, characterName, organizationName, className, gender, race, preTimeskipAge, and PostTimeskipAge of all the characters over in the Blue Lions House

```
select ch.characterID, ch.characterName, o.organizationName, c.className, ch.gender, ch.race,
ch.preTimeskipAge, ch.postTimeskipAge
from Characters ch left join Organizations o on (ch.organizationID = o.organizationID)
                           left join Classes c on (ch.classId = c.classID)
where ch.organizationID = '002';
```

	characterid integer	charactername text	organizationname text	classname text	gender genderidentity	race raceidentity	pretimeskipage integer	posttimeskipage integer
1	5	Dimitri	Blue Lions	Paladin	Male	Human	18	24
2	6	Felix	Blue Lions	Noble	Male	Human	18	23
3	7	Mercedes	Blue Lions	Commoner	Female	Human	23	29

Reports and Interesting Queries

Characters in the Golden Deer House

Query 4: Get the characterID, characterName, organizationName, className, gender, race, preTimeskipAge, and PostTimeskipAge of all the characters over in the Golden Deer House

```
select ch.characterID, ch.characterName, o.organizationName, c.className, ch.gender, ch.race,
ch.preTimeskipAge, ch.postTimeskipAge
from Characters ch left join Organizations o on (ch.organizationID = o.organizationID)
                           left join Classes c on (ch.classId = c.classID)
where ch.organizationID = '003';
```

	characterid integer	charactername text	organizationname text	classname text	gender genderidentity	race raceidentity	pretimeskipage integer	posttimeskipage integer
1	8	Claude	Golden Deer	Wyvern Rider	Male	Human	18	24
2	9	Lysithea	Golden Deer	Mage	Female	Human	16	21
3	10	Hilda	Golden Deer	Warrior	Female	Human	19	24

Stored Procedures

Locate Character

Stored Procedure 1: Create a function that allows the player to input the name of the character that they are looking for find the organization they are affiliated with

```
create or replace function locateCharacter(text)
returns table(name text, organization text) as $$
declare
  findCharacter text := $1;
begin
  return query
  select ch.characterName, o.organizationName
  from Characters ch inner join Organizations o on (ch.organizationID = o.organizationID)
  where ch.characterName = findCharacter;
end;
$$ language plpgsql;

select locateCharacter('Claude');
select locateCharacter('Dimitri');
select locateCharacter('Edalgard');
select locateCharacter('Alan');
```

	locatecharacter record
1	(Claude,"Golden Deer")

	locatecharacter record
1	(Dimitri,"Blue Lions")

	locatecharacter record
1	(Edalgard,"Black Eagles")

	locatecharacter record
1	(Alan,"Holy Kingdom of Faerghus")

Stored Procedures

Classes Available Based on Level

Stored Procedure 2: Create a function that lets the player view all of the different classes the player can choose based off of the level the class is available at

```
create or replace function allAvailableClasses(int)
returns table(level int, class text, levelType levelOfClasses, test needed) as $$
declare
  availableLevel int := $1;
begin
  return query
  select c.levelAvailable, c.className, c.classLevel, c.certificationExamNeeded
  from Classes c
  where c.levelAvailable = availableLevel
  group by c.levelAvailable, c.className, c.classLevel,
      c.certificationExamNeeded
  order by c.levelAvailable;
end;
$$ language plpgsql
```

	allavailableclasses record
1	(10,Lord,Intermediate,Yes)
2	(10,Mage,Intermediate,Yes)
3	(10,Priest,Intermediate,Yes)

	allavailableclasses record
1	(20,Assassin,Advanced,Yes)
2	(20,"Fortress Knight",Advanced,Yes)
3	(20,Paladin,Advanced,Yes)
4	(20,Swordmaster,Advanced,Yes)
5	(20,Trickster,Special,Yes)
6	(20,Valkyrie,Special,Yes)
7	(20,"War Monk",Special,Yes)
8	(20,Warrior,Advanced,Yes)
9	(20,"Wyvern Rider",Advanced,Yes)

	allavailableclasses record
1	(30,"Dark Knight",Master,Yes)
2	(30,Gremory,Master,Yes)
3	(30,"Holy Knight",Master,Yes)
4	(30,"Wyvern Lord",Master,Yes)

	allavailableclasses record
1	(0,Archbishop,Non-Playable,No)
2	(0,Commoner,Unique,No)
3	(0,Dancer,Unique,No)
4	(0,Emperor,None,No)
5	(0,Goddess,None,No)
6	(0,"Lord of the Desert",Monster,...
7	(0,"Lord of the Lake",Monster,No)
8	(0,Noble,Unique,No)
9	(0,"Prime Minister",None,No)

Triggers

Trigger 1: Create a trigger that checks if the new class is downloadable content. If it is, delete the class from database.

```
create or replace function downloadableClass()
returns trigger as $$
begin
  if (new.classLevel = 'DLC') then
  delete from Classes where classLevel = new.classLevel;
  end if;

  return NEW;
  end;
  $$ language plpgsql

create trigger downloadableClass
after insert on Classes
for each row
execute procedure downloadableClass();

INSERT INTO Classes
VALUES (030, 'Death Knight', 'DLC', '0', 'No')

select *
from Classes;
```

	classid [PK] integer	classname text	classlevel levelofclasses	levelavailable integer	certificationexamneeded needed
19	19	Valkyrie	Special	20	Yes
20	20	Dark Knight	Master	30	Yes
21	21	Holy Knight	Master	30	Yes
22	22	Wyvern Lord	Master	30	Yes
23	23	Gremory	Master	30	Yes
24	24	Archbishop	Non-Playable	0	No
25	25	Lord of the Lake	Monster	0	No
26	26	Lord of the De…	Monster	0	No
27	27	Emperor	None	0	No
28	28	Prime Minister	None	0	No
29	29	Goddess	None	0	No

Security

Roles

```
create role admin;
grant all on all tables in schema public to admin;

create role authorityFigures;
revoke all on all tables in schema public from authorityFigures;
grant select on all tables in schema public to authorityFigures;
grant insert on Characters, PlayableCharacters, NonplayableCharacters, AntagonistCharacters,
BackstoryCharacters
to authorityFigures;
grant update on Characters, PlayableCharacters, NonplayableCharacters,AntagonistCharacters,
BackstoryCharacters
to authorityFigures;

create role players;
revoke all on all tables in schema public from players;
grant select on Characters, PlayableCharacters, NonplayableCharacters,
  AntagonistCharacters, BackstoryCharacters
to players;
```

Implementation Notes

- Postgres does not support the Fóldan calendar and because of this, the months will have to be replaced with the Gregorian calendar. The calendar starts the year off in April and will end in March. The following are the months according to the Fóldan calendar and its equivalent is in the Gregorian calendar: Great Tree Moon (April), Hartstrings Moon (May), Garland Moon (June), Blue Sea Moon (July), Verdant Rain Moon (August), Horsebow Moon (September), Wyvern Moon (October), Red Wolf Moon (November), Ethereal Moon (December), Guardian Moon (January), Pegasus Moon (February), Lone Moon (March).
- The age of several characters is unknown, therefore I did my best to estimate the character's age based on their class and the organization that they may be affiliated with, as well as the art of the said character.
- Several characters have been alive way before the main storyline that it is hard to estimate when exactly they were born before the "Imperial Year" era, which is why their age does not change even after the timeskip.
- For those who are not familiar with Fire Emblem: Three Houses, the player has to take a Certification Exam to advance to the next class or reclass.

Known Problems/Future Enhancements

- Due to the date issue at hand, the dates will have to be roughly translated to the Gregorian calendar as best as possible.
- Due to the specific age of certain characters, in a future version, some ages can play shown with some indication that it is a rough estimate or that the character may be immortal
- More information about characters can be added to a future version, such as, any aliases they may have, what type of relationship they have with another character, as well as when the character first appears.