The background of the image is a detailed illustration from the Fire Emblem Awakening game. It features a group of characters in dynamic poses. In the center, a man with long dark hair and a blue tunic (Marth) is shown in a powerful stance, holding a sword. To his right, a woman in a blue and red outfit (Lucina) is also prominent. Other characters include a man with a large axe (Meteo), a woman with long brown hair (Sully), and a man in a blue hooded cloak (Nephros). The art style is characteristic of the Fire Emblem series, with detailed armor and flowing hair. The title 'FIRE EMBLEM' is written in a large, stylized, black serif font, with 'Awakening' in a smaller, simpler font below it. The entire scene is set against a light gray background.

FIRE EMBLEM[™] Awakening

By Eric Schmidt

Table of Contents

Table of Contents	2
Executive Summary	3
Entity Relationship Diagram	4
Tables:	
Units Table.....	5
Weapons Table	7
Abilities Table	9
Items Table	10
CopyUnits Table	11
Support Table	12
Relationship Table	13
TeamMakeUp Table	14
Team Table	15



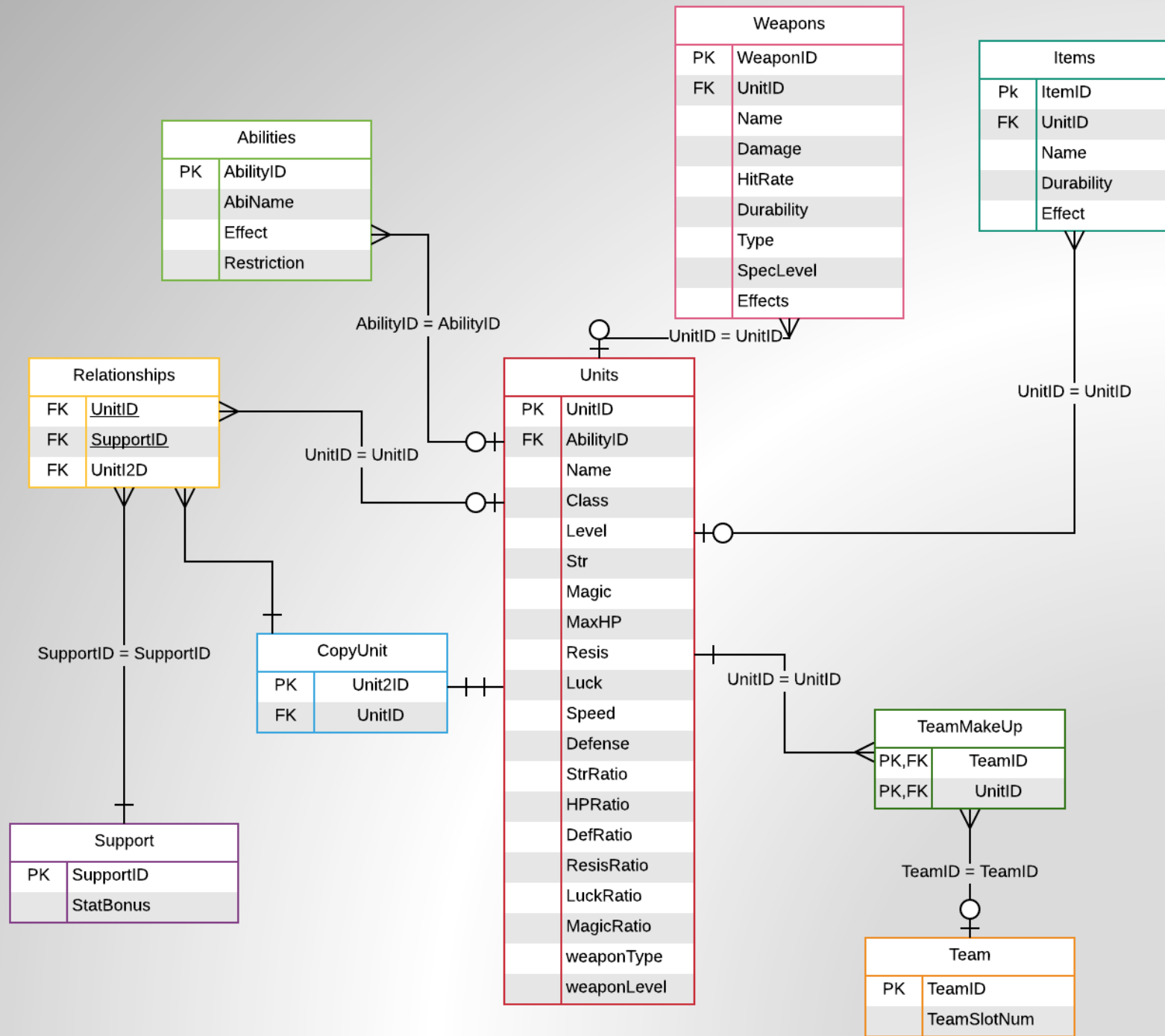
Executive Summary

Fire Emblem is a game where players are allowed to acquire units and use them to battle various strategy game modes. Each character can have weapons and items to aid them in these battles. Each character is also specialized in a type of combat (i.e. swords, bows etc.).

This database was created with the intent to allow users to easily access their units and view their available teams they have made. It also allows developers the ability to add more content if necessary over time.



Fire Emblem ER Diagram



CREATE TABLE units(

UnitID	int	not null,
name	text	not null,
class	text	not null,
weaponType	text	not null,
weaponLevel	text	not null,
level	int	not null,
str	int	not null,
magic	int	not null,
maxHP	int	not null,
resis	int	not null,
luck	int	not null,
speed	int	not null,
defense	int	not null,
strRatio	int	not null,
HPRatio	int	not null,
defRatio	int	not null,
resisRatio	int	not null,
speedRatio	int	not null,
luckRatio	int	not null,
magicRatio	int	not null,
AbilityID	int	references Ability(AbilityID),

check(weaponType = 'Sword' or weaponType = 'Axe' or weaponType = 'Tome' or
weaponType = 'Bow' or weaponType = 'Lance'),
primary key(UnitID)
);

Units Table

Functional Dependencies:
UnitID → name, class, weaponType, weaponLevel, level,
Str, magic, maxHP, resis, luck, speed, defense,
strRatio, HPRatio, defRatio, resisRatio, speedRatio, luckRatio,
magicRatio



<input type="checkbox"/>	unitid integer	name text	class text	weapontype text	weaponlevel text	level integer	str integer	magic integer	maxhp integer	resis integer	luck integer	speed integer	defense integer	strratio integer	hpratio integer	defratio integer	resisratio integer	speedratio integer	luckratio integer	magicratio integer
<input type="checkbox"/>	1	Ike	Hero	Sword	C	1	5	1	19	0	6	7	5	50	75	40	40	55	35	20
<input type="checkbox"/>	10	Hector	General	Axe	A	20	18	1	43	15	15	10	30	50	85	10	5	10	30	50
<input type="checkbox"/>	8	Julia	Priestess	Tome	B	15	3	26	37	21	8	17	6	10	90	10	14	30	30	100
<input type="checkbox"/>	23	Roy	Mercenary	Sword	C	11	11	1	27	0	6	12	8	80	75	40	20	65	30	10
<input type="checkbox"/>	33	Takumi	Sniper	Bow	B	7	12	0	24	6	11	10	9	50	60	45	25	55	50	0
<input type="checkbox"/>	61	Reinhardt	Mage Knight	Tome	A	20	13	20	48	17	18	14	12	45	75	60	80	55	70	90
<input type="checkbox"/>	54	Boyd	Fighter	Axe	D	2	7	0	30	0	4	6	5	60	75	25	25	45	35	5
<input type="checkbox"/>	102	Corrin	Princess	Sword	C	1	7	4	19	2	5	6	6	60	60	45	30	55	55	40
<input type="checkbox"/>	7	Alan	Dragon	Tome	A	20	22	9	51	17	12	20	33	90	100	50	30	45	25	40

-- No values can be null in this table --



Weapons Table

```
create table weapons(  
  WeaponID      int          not null,  
  UnitID        int          references units(UnitID),  
  Name          text         not null,  
  Damage        int          not null,  
  HitRate       int          not null,  
  Durability    int,         not null,  
  Type          text         not null,  
  SpecLevel     char,  
  Effects       text,  
  check(Name is NOT NULL),  
  primary key(WeaponID)  
);
```

Function Dependencies:

weaponID → Name, damage, HitRate, Durability, Type, SpecLevel, Effects



weaponid integer	unitid integer	name text	damage integer	hitrate integer	durability integer	type text	spelevel character (1)	effects text
1	54	Iron Axe	10	85	30	Axe	E	[null]
2	33	Fujin Yomi	12	90	[null]	Bow	[null]	Crit Rate + 10%
3	61	Thunder	14	75	40	Tome	D	Effective against Flying
6	102	Yato	12	70	[null]	Sword	[null]	[null]

Can be null do the fact that
some weapons are
unbreakable

Can be null because not all weapons have special effects



Abilities Table

```
create table abilities(  
  AbilityID      int           not null,  
  AbiName        text         not null,  
  effect         text         not null,  
  restriction     text,  
  primary key(AbilityID)  
);
```

Functional Dependencies:
AbilityID → AbiName, effect, restriction

abilityid integer	abiname text	effect text	restriction text
2	Vantage	50% chance to attack first always	Sword
5	Double-Take	Always Double Hit when initiating	Bow
4	Savage Blow	Deal 5 Damage to adjacent units after attacking	Tome




Items Table

```
create table items(  
  ItemID      int      not null,  
  UnitID      int      references units(UnitID),  
  name        text     not null,  
  durability  int,  
  effect      text,  
  primary key(ItemID)  
);
```

itemid integer	unitid integer	name text	durability integer	effect text
1	1	Elixir	3	Heal 10 HP
3	8	Master Seal	1	Evolve Unit
7	61	Boots	1	Increase Speed by 3
12	[null]	Goddess Icon	1	Increase Max HP by 5
15	[null]	Member Card	[null]	Gives Access to Secret Shops

Some items are unbreakable



Functional Dependencies:
ItemID → name, durability, effect



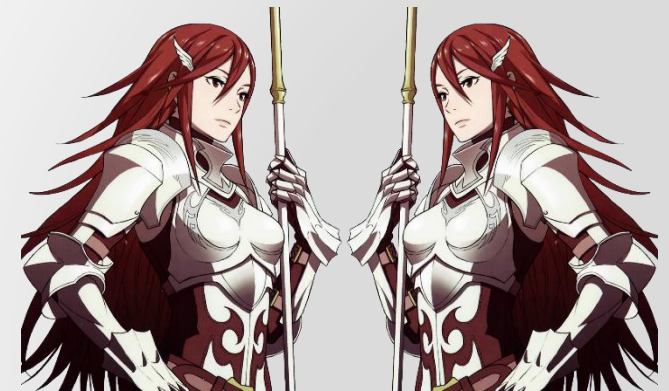
CopyUnit Table

Creates a copy ID of all units to allow relationships between two units

```
create table copyUnit(  
  UnitID      int      not null references units(UnitID),  
  Unit2ID     int      not null,  
  primary key(Unit2ID)  
);
```

Functional Dependencies:
Unit2ID → UnitID

unitid integer	unit2id integer
1	1
8	8
10	10
23	23
33	33
61	61
54	54
102	102
7	7



Support Table

```
create table support(  
  SupportID      int      not null,  
  StatBonus      int,  
  primary key(SupportID)  
);
```

-- Stat bonuses are treated as flat increases. Ex: 2 = +2 to all stats of a unit--

supportid integer	statbonus integer
1	2
2	4
3	1
4	5

Functional Dependencies:
SupportID → StatBonus



Relationships Table

```
create table relationships(  
  UnitID          int      not null references units(UnitID),  
  Unit2ID         int      not null references copyUnit(Unit2ID),  
  SupportID       int      not null references support(SupportID),  
  primary key(UnitID, SupportID)  
);
```

Functional Dependencies:
UnitID, SupportID → Unit2ID

unitid integer	unit2id integer	supportid integer
1	8	1
10	102	2
54	23	1
61	33	4



TeamMakeUp Table

```
create table teamMakeUp(  
  TeamID      int  
  UnitID      int  
  primary key(TeamID, UnitID)  
);
```

not null references teams(TeamID),
references units(UnitID),

teamid integer	unitid integer
1	1
1	102
1	8
2	1
2	61
2	54
3	33
3	54
3	8

Functional Dependencies:
None: Full Keyed Relation



Teams Table

```
create table teams(  
    TeamID      int      not null,  
    teamName    text      not null,  
    primary key(TeamID)  
);
```

Functional Dependencies:
TeamID → teamName

teamid integer	teamname text
1	OP
2	BackseatWarriors
3	OG Squad



useAbilities View

Displays All Units who can use the available Abilities:

CREATE VIEW useAbilities as

```
SELECT units.name, units.class, units.weaponType, abilities.effect, abilities.AbiName  
FROM units  
INNER JOIN abilities on units.weaponType = abilities.restriction;
```

name text	class text	weapontype text	effect text	abiname text
Corrin	Princess	Sword	50% chance to attack first always	Vantage
Roy	Mercenary	Sword	50% chance to attack first always	Vantage
Ike	Hero	Sword	50% chance to attack first always	Vantage
Takumi	Sniper	Bow	Always Double Hit when initiating	Double-Take
Alan	Dragon	Tome	Deal 5 Damage to adjacent units after attacking	Savage Blow
Reinhardt	Mage Knight	Tome	Deal 5 Damage to adjacent units after attacking	Savage Blow
Julia	Priestess	Tome	Deal 5 Damage to adjacent units after attacking	Savage Blow



CurrentTeam View

Displays the currently selected team:

CREATE VIEW currentTeam as

```
SELECT units.name,units.class,units.level, units.str, units.magic,  
       units.maxHP,units.resis,units.luck,units.speed,units.defense, teams.teamName  
FROM teamMakeup  
INNER JOIN units on units.UnitID = teamMakeUp.UnitID  
INNER JOIN teams on teamMakeUp.teamID = teams.teamID  
WHERE teams.teamID = 01;
```

name text	class text	level integer	str integer	magic integer	maxhp integer	resis integer	luck integer	speed integer	defense integer	teamname text
Ike	Hero	1	5	1	19	0	6	7	5	OP
Julia	Priestess	15	3	26	37	21	8	17	6	OP
Corrin	Princess	1	7	4	19	2	5	6	6	OP



Report – Equipped Weapons

-- show all units with their equipped or unequipped to check who still needs weapons--

```
SELECT units.name, units.class, weapons.name as Weapon_Name  
FROM Units  
FULL JOIN Weapons ON units.UnitID = weapons.UnitID  
ORDER BY units.name DESC;
```

name text	class text	weapon_name text
Takumi	Sniper	Fujin Yomi
Roy	Mercenary	[null]
Reinhardt	Mage Knight	Thunder
Julia	Priestess	[null]
Ike	Hero	[null]
Hector	General	[null]
Corrin	Princess	Yato
Boyd	Fighter	Iron Axe
Alan	Dragon	[null]

NULL values mean you still need to equip a weapon



Report - InRelationship

-- Show all units that are in a relationship --

```
SELECT Units.name  
FROM units  
INNER JOIN copyunit on units.UnitID = copyUnit.Unit2ID;
```

name text
Ike
Julia
Hector
Roy
Takumi
Reinhardt
Boyd
Corrin
Alan



Report - BestStats

-- Show Unit with best current Stats --

```
SELECT units.name, units.class, (units.str +  
                                units.magic +  
                                units.maxHP +  
                                units.luck +  
                                units.speed +  
                                units.defense +  
                                units.resis) as Total_Stat
```

```
FROM Units  
ORDER BY Total_Stat DESC  
LIMIT 1;
```

name text	class text	total_stat integer
Alan	Dragon	164

COOL!

Strong!

WOW!



Reports – Best Stat Ratio

-- Show Units with best Stat Ratios --

```
SELECT units.name,units.class, (units.strRatio + units.magicRatio +  
                                units.HPRatio +units.luckRatio +  
                                units.speedRatio + units.defRatio +  
                                units.resisRatio) as Total_Ratio
```

```
FROM Units
```

```
ORDER BY Total_Ratio DESC
```

```
LIMIT 1;
```

name text	class text	total_ratio integer
Reinhardt	Mage Knight	475

This is useful to find out who may be the most efficient to level



Report –Unit Stats at Max Level

-- Show possible stats for all units if not already max level of 20 --

Select units.name, units.str + ((20-units.level) * (units.strRatio) / 100) as Str,
units.magic + ((20-units.level) * (units.magicRatio) / 100) as Magic,
units.maxHP + ((20-units.level) * (units.HPRatio) / 100) as MaxHP,
units.luck + ((20-units.level) * (units.LuckRatio) / 100) as Luck,
units.speed + ((20-units.level) * (units.SpeedRatio) / 100) as Speed,
units.defense + ((20-units.level) * (units.defRatio) / 100) as Def,
units.resis + ((20-units.level) * (units.resisRatio) / 100) as Resis

FROM units;

name text	str integer	magic integer	maxhp integer	luck integer	speed integer	def integer	resis integer
Ike	14	4	33	12	17	12	7
Hector	18	1	43	15	10	30	15
Julia	3	31	41	9	18	6	21
Roy	18	1	33	8	17	11	1
Takumi	18	0	31	17	17	14	9
Reinhardt	13	20	48	18	14	12	17
Boyd	17	0	43	10	14	9	4
Corrin	18	11	30	15	16	14	7
Alan	22	9	51	12	20	33	17



Stored Procedure - ValidAbility

-- Ensures all abilities are inputted correctly --

```
CREATE OR REPLACE FUNCTION ValidAbility()  
RETURNS TRIGGER AS $$  
BEGIN  
    IF NEW.Restriction IS NULL THEN  
        RAISE EXCEPTION 'Restrictions must be entered. If no restriction, put none';  
    END IF;  
    IF NEW.Effects IS NULL THEN  
        RAISE EXCEPTION 'Effects must be entered. If no restrictions, put no effect';  
    END IF;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```



****Output Will be Shown in Trigger Slide****

Stored Procedure – TeamUnits

-- Creates a table that lists all unit's names in a table with their team number

```
CREATE OR REPLACE FUNCTION TeamUnits(IN TeamID INT)
    RETURNS TABLE (UnitName text, TeamNumber int) AS
    $$
```

```
BEGIN
```

```
RETURN QUERY SELECT DISTINCT units.Name, teams.teamID
    FROM TeamMakeUp
    INNER JOIN Units
    ON units.UnitID = teamMakeUp.UnitID
    INNER JOIN Teams
    ON teams.TeamID = teamMakeUp.TeamID
    WHERE teamMakeUp.teamID is NOT NULL;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

INPUT

```
select TeamUnits(1);
```

OUTPUT

teamunits record
(Corrin,1)
(Julia,1)
(Takumi,3)
(Ike,1)
(Reinhardt,2)
(Boyd,2)
(Julia,3)
(Ike,2)
(Boyd,3)



Trigger – ValidAbility

-- Creates a trigger whenever an ability is added to check and ensure all the ability's info is added --

```
CREATE TRIGGER ValidAbility  
BEFORE INSERT OR UPDATE ON Abilities  
FOR EACH ROW  
EXECUTE PROCEDURE ValidAbility();
```

Inputting New Abilities with NULL values

```
(09, 'Death Dance', NULL, NULL);|
```

Output

```
ERROR:  record "new" has no field "effects"  
CONTEXT:  SQL statement "SELECT NEW.Effects IS NULL"  
PL/pgSQL function validability() line 6 at IF  
***** Error *****
```



Security - Developers

-- creates a developer role to add and manipulate tables as they see fit --

```
CREATE ROLE Developers;
```

```
GRANT SELECT, UPDATE ON Units, Abilities, Items, Weapons, Support,  
CopyUnit
```

```
TO Developer;
```



Security - Players

-- creates a player role who only should have access to changing teams, units, items and weapons --

CREATE ROLE Player;

GRANT SELECT ON Teams, Units, Items, Weapons

TO Player;



Implementation Notes

- Copying information from .sql file to pgadmin has occasionally caused quotes to be entered incorrectly. This would cause all the text inputs to misfire and cause many error messages.
- This database only looks at a small sample set of units, weapons and items from the game, but is capable of adding additional units if desired.



Future Enhancements

- Allowing Units to have additional weapon specializations would allow for a more diverse and versatile team composition.
- Creating team captains as unique individuals set to only one team may make it easier to locate and personalize one team set. This was not possible from this small database because of the smaller amount of units available.



Known Problems

- Adding Units is a long and strenuous process so adding additional functionalities to separate the different portions of the unit's attributes may be useful.

