



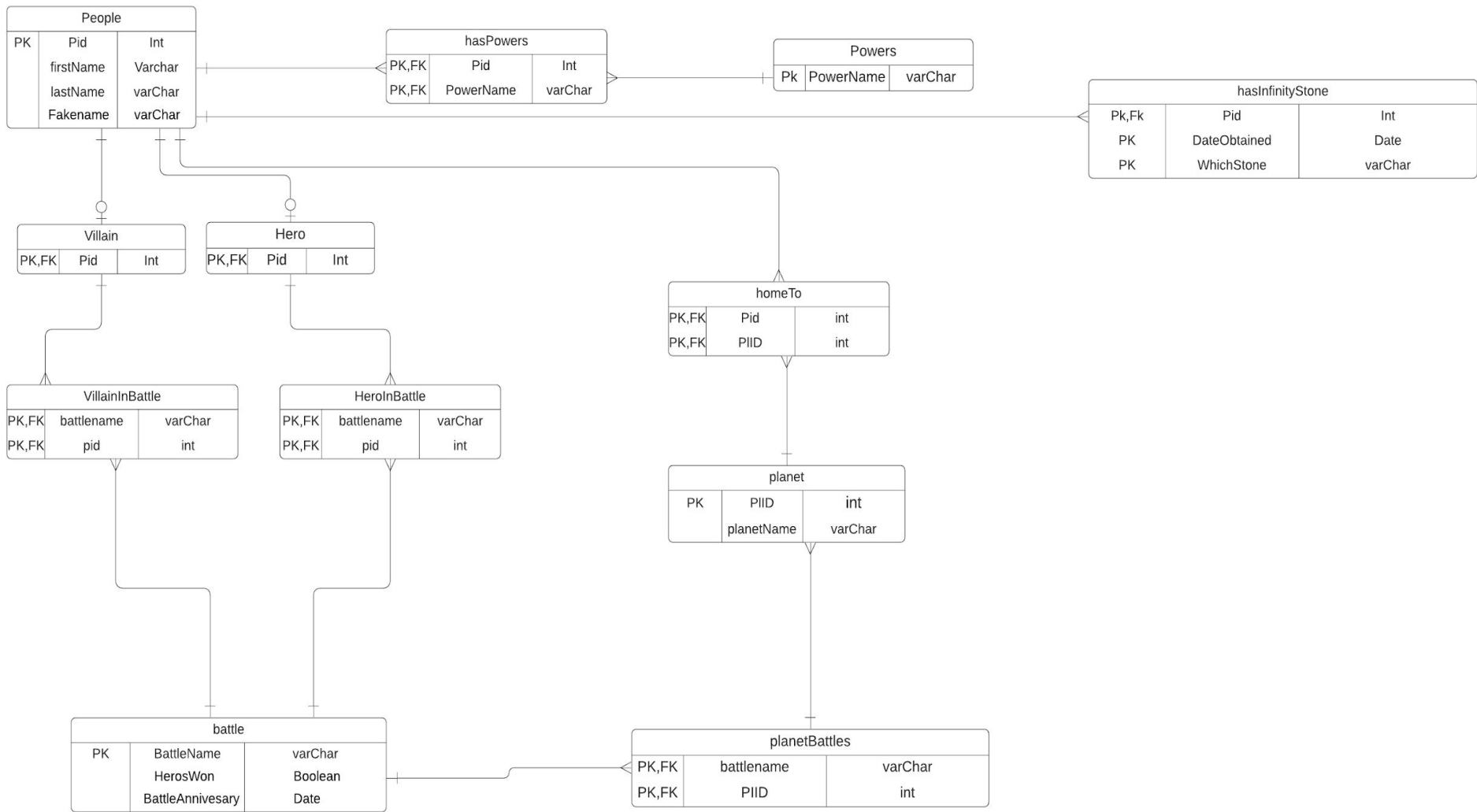
Design Project Marvel Edition

Executive Summary



Captain American hired me to create a database for the marvel universe. After the death of Tony Stark, he didn't believe in the avengers anymore so he wanted to create an elite team that could defeat any enemy. To do this, we have to create the best relational database with the people having the most wins in battle and the most powers.

The tables needed to complete this database are people, villians, battles, and more. We will need reports, views, locks and more for faster access into the database and tables. The entity-relationship diagram will lay out the basis for everything. There we can decide what goes together, what datatypes need to be used for every column and more.



People Table

```
create table people (  
  pid Serial primary key,  
  firstname varchar(50),  
  lastname varchar(50),  
  fakeName varChar(50)  
);
```

	pid [PK] integer	firstname character varying (50)	lastname character varying (50)	fakename character varying (50)
51	53	Proxima	Midnight	None
52	54	Ebony	Maw	None
53	55	Adrian	Toomes	Vulture
54	56	Alexander	Pierce	None
55	57	Ultron	None	None
56	58	Proxima	Midnight	None
57	59	Walter	Lawson	Captain Marvel
58	60	Alan	Labouseur	Professor

This table is used to display all of the people in the Marvel universe. The table will be subdivided into heros and villains to further precise the data.

Functional Dependencies: pid -> firstname, lastname, fakeName

Tables: hero and villain

The people's table is subdivided into the villains and heroes table. The pid column is a foreign key to the people's pid column.

No functional dependencies. They are just foreign keys to the people's table.

	pid [PK] integer
8	11
9	12
10	13
11	16
12	17

	pid integer
1	9
2	31
3	8
4	53
5	15
6	14
7	21

	pid [PK] integer	firstname character varying (50)	lastname character varying (50)	fakename character varying (50)
8	10	Tchalla	Udaku	Black Panther
9	11	Natasha	Romanoff	Black Widow
10	12	Walter	Lawson	Captain Marvel
11	13	Ancient One	None	None
12	14	Thanos	None	None
13	15	Dormammu	None	None
14	16	Wanda	Maximoff	Scarlet Witch
15	17	Clint	Barton	Hawkeye
16	18	Janet	Dyne	Wasp

People's Table

Battle Table

```
create table battle(  
    battleName varchar(50) default 'Unknown',  
    outcome text not null,  
    battleAniversary varchar default 'Unknown'  
);
```

	<div><div>battleName</div><div>character varying (50)</div></div>	<div><div>outcome</div><div>text</div></div>	<div><div>battleanniversary</div><div>character varying</div></div>
1	Battle of Earth	Tony Stark Saves the day in a 1 in 13 million chance	October 2023
2	Thor and Loki vs Hela	Hela wins	Unknown
3	Battle of New York	The avengers defeated	May 4th, 2012
4	Civil war	Tony Stark and Steve Rogers Settled conflict	May 6th, 2016

The battle table is used to display the battles, what the outcome of the battle was and the battle anniversary. This will make it easier for time travel since we know the dates.

Function dependencies:

battleName → outcome, battleanniversary

VillainInbattle Table

This table is used because we can't have a many to many relationship between tables. There are many villains in battles. Battles have many villains. We need this to keep our data consistent. This is called an associate table. This will connect villains with battles.

Data Output	Explain	Messages	Notifications
 pid [PK] integer	 battlename [PK] character varying (50) 		
1	8	Battle of New York	
2	14	Battle of Earth	
3	21	Battle of Earth	

```
create table VillianInBattle (  
    pid int references villan(pid),  
    battleName varchar(50) references battle(battleName),  
    primary key(pid,battleName)  
);
```

There are no functional dependencies. There is only a multi-valued key.

HeroInBattle Table

This table has the same purpose as the VillainInBattle. It will connect the battles table and the hero's table.


	pid [PK] integer	battleName [PK] character varying (50)
1	3	Battle of New York
2	1	Battle of New York
3	6	Battle of New York
4	61	Battle of New York
5	61	Civil war

There are no functional dependencies. There is only a multi-valued key.

```
create table HeroInBattle (  
    pid int references hero(pid),  
    battleName varchar(50) references battle(battleName),  
    primary key(pid,battleName)  
);
```


Powers Table



	powername [PK] character varying (50) 
1	superHumanStrength
2	speed
3	durability
4	fly
5	time manipulation
6	fast healing


This table is used to display all of the powers. I thought powers was superior enough to have its own table because there are so many powers in the marvel universe. I used the the power name as the primary key. Not all people have powers such as Nick Fury.

```
create table powers(  
    powerName varchar(50) not null,  
    primary key(powerName)  
);
```

There are no functional dependencies

hasPowers Table

The hasPowers table is an associate table. This table connects people to Powers. People can have multiple powers and powers belong to multiple people.




	powername [PK] character varying (50)	pid [PK] integer
5	superHumanStrength	14
6	superHumanStrength	10
7	speed	1
8	speed	12
9	speed	42
10	speed	61
11	fly	4
12	fly	1

```
create table hasPowers(  
  powerName varchar(50) not null references powers(powerName),  
  pid int not null references People(pid),  
  primary key(powerName,pid)  
);
```

There are no functional dependencies

Planets Table




	plid [PK] integer		planetname character varying (50)	
1		2	Maveth	
2		1	Asgard	
3		3	Xandar	
4		4	Earth	
5		5	Titan	

This table consists of all of the planets in the Marvel universe. These planets are where a lot of people and aliens are born. The planetname column depends on the plid.

```
create table planet(  
    plid int not null primary key,  
    planetName varchar(50) not null  
);
```

plid → planetName

HomeTo Table



	pid [PK] integer		plid [PK] integer	
1		9		5
2		1		1
3		61		4
4		5		4
5		3		4
6		8		1

This table is an associate table called HomeTo Table. This table is used to correlate the planet table and the people table together to find out someone's birth place.

```
create table homeTo(  
  pid int not null references People(pid),  
  PlID int not null references planet(PlID),  
  primary key(pid,PlID)  
);
```

This is a multi-valued key. No functional dependencies.

planetBattles Table




	battleName [PK] character varying (50)	plid [PK] integer
1	Battle of Earth	4
2	Battle of New York	4
3	Thor and Loki vs Hela	1

This table is used to correlate the battles to which planets they happened on. I did this because planets deserve their own table as there is over 20 planets visited in the marvel universe.

There are no functional dependencies.

```
create table planetBattles(  
    BattleName varchar(50) not null references battle(BattleName),  
    PlID int not null references planet(PlID),  
    primary key(PlID, BattleName)  
);
```

HasInfinityStones Table



```
create table hasInfinityStone(  
  pid int not null references people(pid),  
  dateObtained date not null,  
  whichStone Text not null,  
  primary key(pid,dateObtained,whichStone)  
);
```

	pid [PK] integer		dateobtained [PK] date		whichstone [PK] text
1	1		2001-01-02		SoulStone
2	6		2001-04-23		Power
3	12		2019-06-20		Power

The infinity stones are a huge part of the marvel universe. We need to know where they are at all times. They were sent back to their original timeline and we have no idea where they could be now.

There are no functional dependencies.
This is a multi-valued key. I did a multi-valued key because the same people can have multiple infinity stones on the same date. For example, Thanos.

Views

```
create view getAvengersInNewYork as
select firstname,lastname
from People
where pid in (select pid
              from hero
              where pid in( select pid
                           from HeroInBattle
                           where battleName = 'Battle of New York')));

select * from getAvengersInNewYork
```

	firstname character varying (50)	lastname character varying (50)
1	Tony	Stark
2	Bruce	Banner
3	Thor	Odinson
4	Steve	Rogers

This View gets all of the Avengers that fought in the battle of New York.

The view uses a subquery to get the first and last name from the people table and correlates the pid to the HeroInBattle table where the battle was New York.

Views

The getMVP view function will return the most valuable person. This is the person who has been in the most battles where the hero's have won. This person will get a reward handed to him by no other than Captain America.

```
create or replace view getMVP as
select pe.firstname,pe.lastname,count(b.outcome)
from people pe inner join hero h on h.pid = pe.pid
inner join HeroInBattle hi on hi.pid = h.pid
inner join battle b on b.battlename = hi.battlename
where b.outcome = true
group by firstname,lastname
order by count desc
limit 1;
```

```
select * from getMVP
```

firstname character varying (50)	lastname character varying (50)	count bigint
Steve	Rogers	2

Views

```
create view battlePlanets as
```

```
select pe.firstname, pe.lastname, pl.planetName, count(pb.battleName)
from people pe full outer join HomeTo H on pe.pid = h.pid
inner join planet pl on pl.PlID = h.PlID
inner join planetBattles pb on pb.PlID = h.PlID
group by planetName, pe.firstname, pe.lastname
order by count desc
limit 1;
```

	firstname character varying (50)	lastname character varying (50)	planetname character varying (50)	count bigint
1	Tony	Stark	Earth	2

This view is called battlePlanets. This will return the People with the most times a planet has been in a battle. There may be a correlation with the battles and the people living their.

Views



```
create or replace view PeopleWithoutPowers
as
select *
from people
where pid not in (select pid
                  from hasPowers);

select * from PeopleWithPowers
```

This is a great view to have because there is many people without powers. One of the people being Nick Fury and he is one of the most important hero's in the marvel universe

24	25	Nick	Fury	None
----	----	------	------	------

Stored Procedures

```
CREATE OR REPLACE FUNCTION getPeople(varChar,varchar,refcursor) RETURNS refcursor AS
$$
  declare
    PowerID varChar = $1;
    nameID varChar = $2;
    resultset refcursor = $3;
  begin
    open resultset for
      select firstname, lastname
      from people
      where pid in (select pid
                    from hasPowers
                    where PowerName in(
                                select PowerName
                                from Powers
                                where PowerName = PowerID  ))
      and
      pid in (select pid
              from hero
              where pid in(select pid
                            from HeroInBattle
                            where battleName = nameID
                            ));
    return resultset;
  end;
$$
language plpgsql;
```

```
SELECT getPeople('superHumanStrength','Battle of New York','results');
FETCH ALL FROM results;
```

	<div>firstname</div> <div>character varying (50)</div>	<div>lastname</div> <div>character varying (50)</div>
1	Steve	Rogers
2	Thor	Odinson
3	Tony	Stark

Trigger

```
CREATE OR REPLACE FUNCTION deleteEverything() RETURNS trigger AS
$$
begin
    delete from HeroInBattle
    where pid = old.pid;

    delete from hero
    where pid = old.pid;

    delete from VillainInBattle
    where pid = old.pid;

    delete from villain
    where pid = old.pid;

    delete from hasPowers
    where pid = old.pid;


    return old;
end;
$$
language plpgsql;
```

```
create trigger OneDelete
before Delete on People
for each row
execute procedure deleteEverything();
```

```
delete from people where pid = 61;
```

This trigger function deletes everything from last in first out order. The foreign key constraint prevents us from deleting so I made a trigger to delete everything in order. I like the idea of this trigger because it saves so much time for programmers.

Reports



I thought this was interesting because there are many heros who are also villains in the marvel universe. I thought it would be useful to know who's trustworthy or not for Captain America's Team.

```
create view bothVillainAndHero as
select distinct pe.firstname, pe.lastname
from people pe inner join villain v on v.pid = pe.pid
inner join hero h on h.pid = pe.pid
```

```
select *
from bothVillainAndHero
```

firstname character varying (50)	lastname character varying (50)
Loki	Odinson
Nebula	None
Bucky	Barnes
Gunter	Otto

Security - Grant and Revoke for Users and Groups

```
create role Avengers;  
grant all on all tables in schema public to Avengers;  
  
create role Villains;  
Revoke all on all tables in schema public from Villains;  
grant select on all tables in schema public to Villains;  
  
create role Trainees;  
Revoke all on all tables in schema public from Trainees;  
grant select,delete,update on all tables in schema public to Trainees;
```

For security purposes, Villains won't be allowed to do any actions but, select. Trainees are people who Captain America is training. They get to learn the basics of SQL to take over the business. All of the avengers get all the permissions.

Known Problems



One known problem is the 'battleanniversary' column in the battle table. This a problem because I made it into a varchar data type. This means someone could put an incorrect date in without even knowing. The reason I did this was because not all dates of the battles can be found so I wanted a default value. Dates can't have default values.

Another known problem during the marvel universe is the idea of time travel. I did not think of that implementation before I did the project. That could mess up the whole database. Someone could time travel back in time and steal an infinity stone (like the movies) and that would mess up the dates of tables. That would change the past and then change the future. It is hard to account for time travel.

Future Enhancements



Some future enhancements would be to add time travel table. In Avengers End game, the avengers time traveled and went back into the past to get the infinity stones. It would be a hard implementation because everything could change. There would need to be an extra table for every table.

Another enhancement could be to add a statistics table. For example, We could put how many kills people have, how many times they've gotten down, how many times they've won a battle. This would make this better for Captain America since he wants only the best team.

A final enhancement is more triggers. Triggers are fun to have and make programmers jobs easier. I believe triggers are the best thing in SQL. Putting an update trigger would make things easier for future assessments. Instead of deleting everything and putting it back manually, we could use a trigger.

Implementation notes



During implementation, I looked up a bunch about the Marvel Universe. I looked up peoples powers, who they fought for(villain or hero) , planet names and battle names. I had a lot of fun learning about all of this during the project. Now that I know the marvel universe is more than just the movies, it makes me look forward to them more. I changed a few things during implementation such as changing many data types. At first, I decided to use a varchar data type for the outcome of battles. I didn't think this really fit the assignment Captain America assigned me. He said he wanted statistics so that's when I changed the data type to a boolean. I thought this fit the situation better. I could now do a count on the table and see how many battles people won and what planet they were on.