# lord of the rings trilogy database

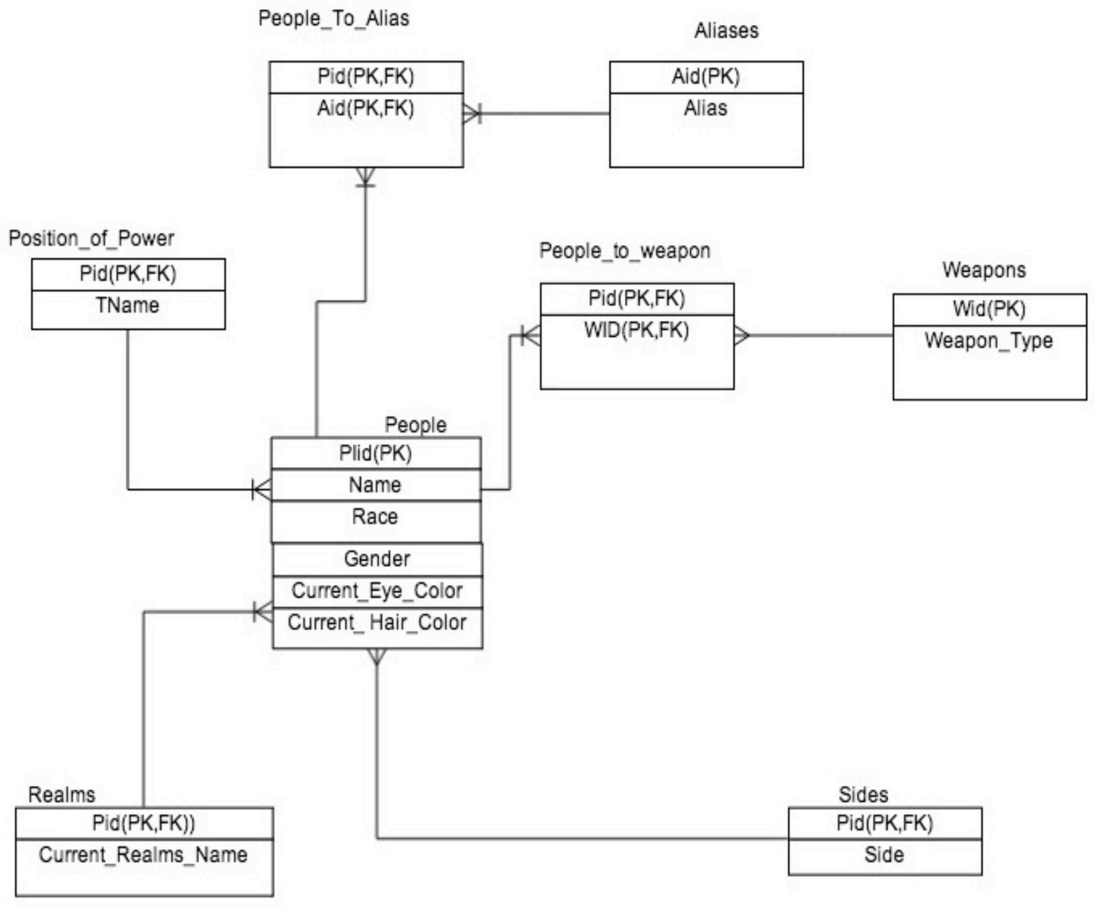
By:
Chris Maher

## table of contents

## executive summary

This database is designed to handle the characters and miscellaneous animals that populate Peter Jackson's interpretation of one of the greatest stories of all time, The Lord of the Rings. Written by J.R.R Tolkien, this series developed a world of its own with a classic battle of good versus evil. The database allows the storing of each the characters: Name, Race, Gender, Eye Color, and Hair Color. It also allows for the user to track each weapon that the character uses, Realm they are fighting for or represent, their many aliases used, and their current title. For example, it can be used to find out which characters belong to each realm, aliases used, and their current Title.

# create table statements



## create table statements

The section that follows is designed to show the Functional Dependencies of each of the tables in the database, along with the create statements for said tables, and some gorgeous snapshots of the tables.

### People Table

The People Table has 6 unique fields such as FName,  LName, Race, Gender, Eye_Color, Hair_Color. These are designed to keep track of the physical characteristic of each character.

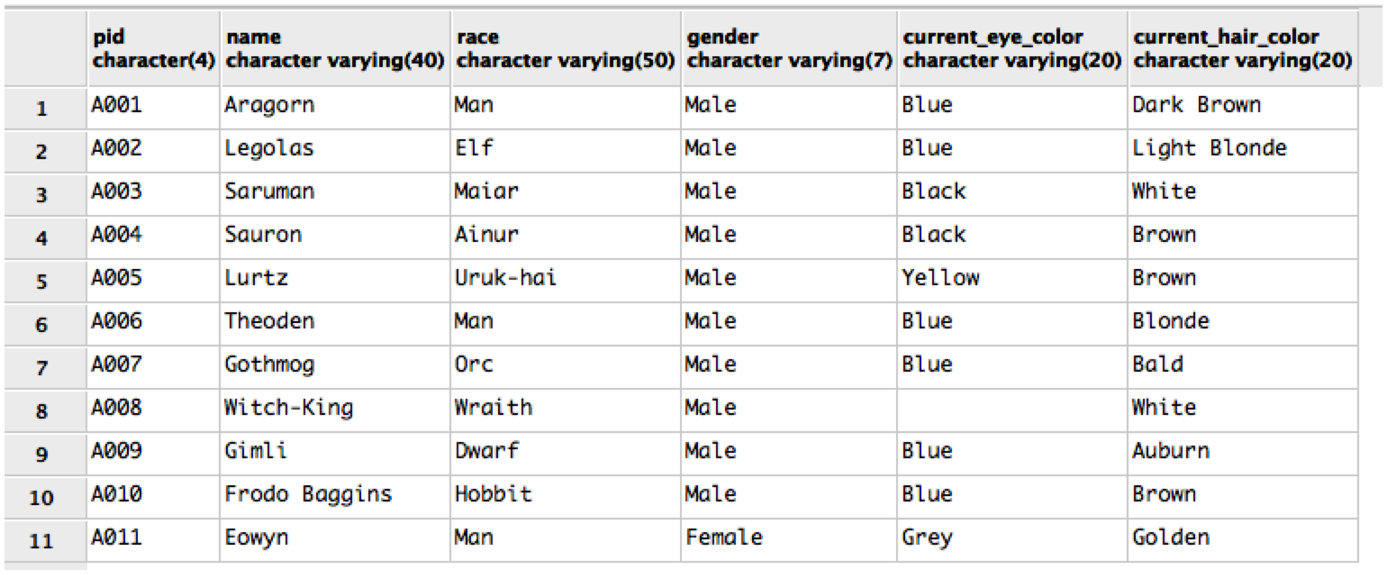
### SQL Create Statement

```
Drop table if exists People;
Create table People(
Pid                       char(4) not null,
Name                  varchar(40) not null,
Race                      varchar(50) not null,
Gender                varchar(7) not null,
Current_Eye_Color         varchar(20),
Current_Hair_Color        varchar(20) not null,
Primary Key (Pid)
);
```

### Functional Dependencies

**People**: Pid → Name, Race,Gender,Eye_Color, Hair_Color

## Sample Data



| | pid character(4) | name character varying(40) | race character varying(50) | gender character varying(7) | current_eye_color character varying(20) | current_hair_color character varying(20) |
|---|---|---|---|---|---|---|
| 1 | A001 | Aragorn | Man | Male | Blue | Dark Brown |
| 2 | A002 | Legolas | Elf | Male | Blue | Light Blonde |
| 3 | A003 | Saruman | Maiar | Male | Black | White |
| 4 | A004 | Sauron | Ainur | Male | Black | Brown |
| 5 | A005 | Lurtz | Uruk-hai | Male | Yellow | Brown |
| 6 | A006 | Theoden | Man | Male | Blue | Blonde |
| 7 | A007 | Gothmog | Orc | Male | Blue | Bald |
| 8 | A008 | Witch-King | Wraith | Male | | White |
| 9 | A009 | Gimli | Dwarf | Male | Blue | Auburn |
| 10 | A010 | Frodo Baggins | Hobbit | Male | Blue | Brown |
| 11 | A011 | Eowyn | Man | Female | Grey | Golden |

## People_To_Weapon Table

The People_To_Weapon Table is designed to remove the possibility of a many to many relation. The table has no unique fields.

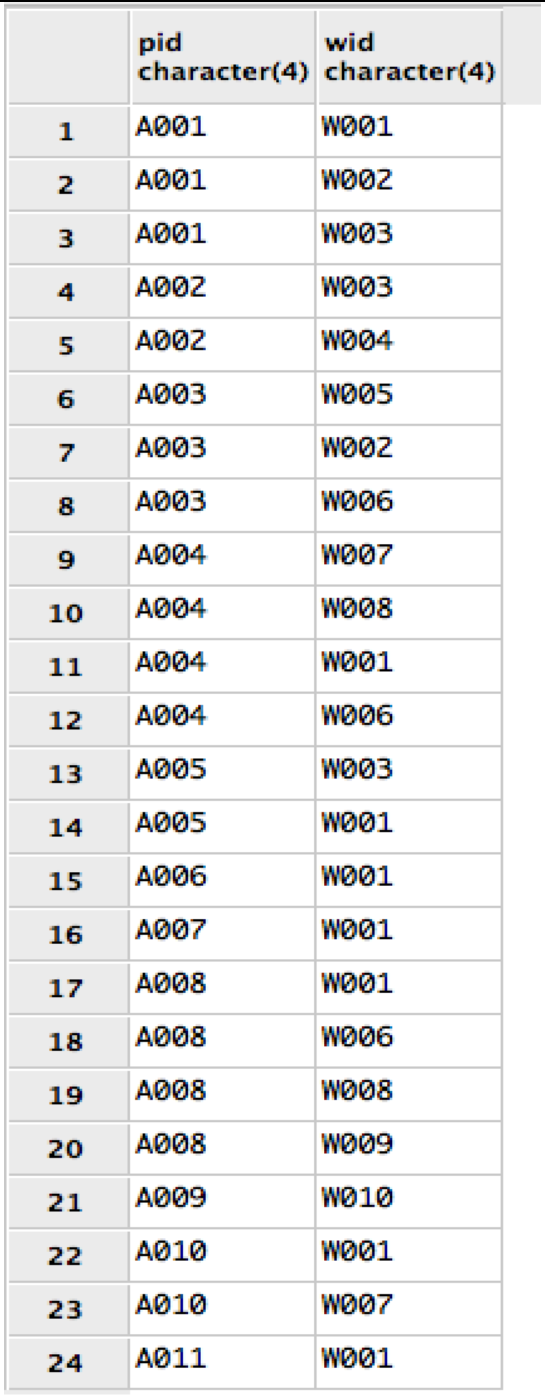
## SQL Create Statement

```
Drop table if exists People_To_Weapon;
 Create table People_To_Weapon(
 Pid    char(4) not null references People(Pid),
 Wid    char(4) not null references Weapons
(Wid),
Primary Key (Pid,Wid)
 );
```

## Functional Dependencies

**People_To_Weapons** : No Dependencies

## Sample Data



| | pid character(4) | wid character(4) |
|---|---|---|
| 1 | A001 | W001 |
| 2 | A001 | W002 |
| 3 | A001 | W003 |
| 4 | A002 | W003 |
| 5 | A002 | W004 |
| 6 | A003 | W005 |
| 7 | A003 | W002 |
| 8 | A003 | W006 |
| 9 | A004 | W007 |
| 10 | A004 | W008 |
| 11 | A004 | W001 |
| 12 | A004 | W006 |
| 13 | A005 | W003 |
| 14 | A005 | W001 |
| 15 | A006 | W001 |
| 16 | A007 | W001 |
| 17 | A008 | W001 |
| 18 | A008 | W006 |
| 19 | A008 | W008 |
| 20 | A008 | W009 |
| 21 | A009 | W010 |
| 22 | A010 | W001 |
| 23 | A010 | W007 |
| 24 | A011 | W001 |

## Weapons Table

The Weapons Table has 1 unique field  Weapon_Type. This field is designed to pair up a weapon type to each Wid. An example of a Weapon_Type is a bow.

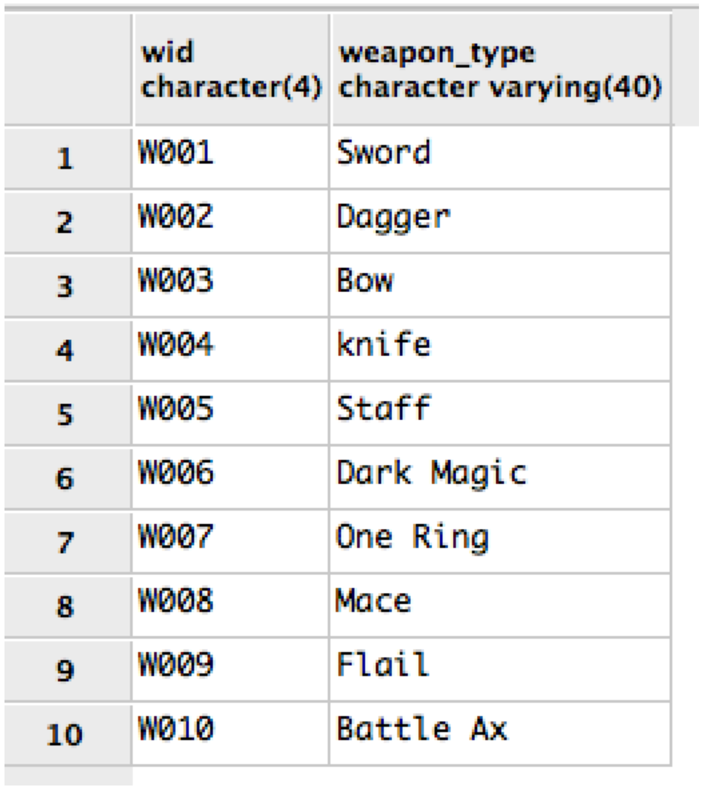
## SQL Create Statement

```
Drop table if exists Weapons;
Create table Weapons(
Wid                      char(4) not null,
Weapon_Type              varchar(40) not null,
Primary Key (Wid)
);
```

## Functional Dependencies

**People**: Wid → Weapon_Type

## Sample Data



| | wid character(4) | weapon_type character varying(40) |
|---|---|---|
| 1 | W001 | Sword |
| 2 | W002 | Dagger |
| 3 | W003 | Bow |
| 4 | W004 | knife |
| 5 | W005 | Staff |
| 6 | W006 | Dark Magic |
| 7 | W007 | One Ring |
| 8 | W008 | Mace |
| 9 | W009 | Flail |
| 10 | W010 | Battle Ax |

## Sides Table

The Sides Table has 1 unique field called side. These are designed to keep track of the characters side, good or evil.

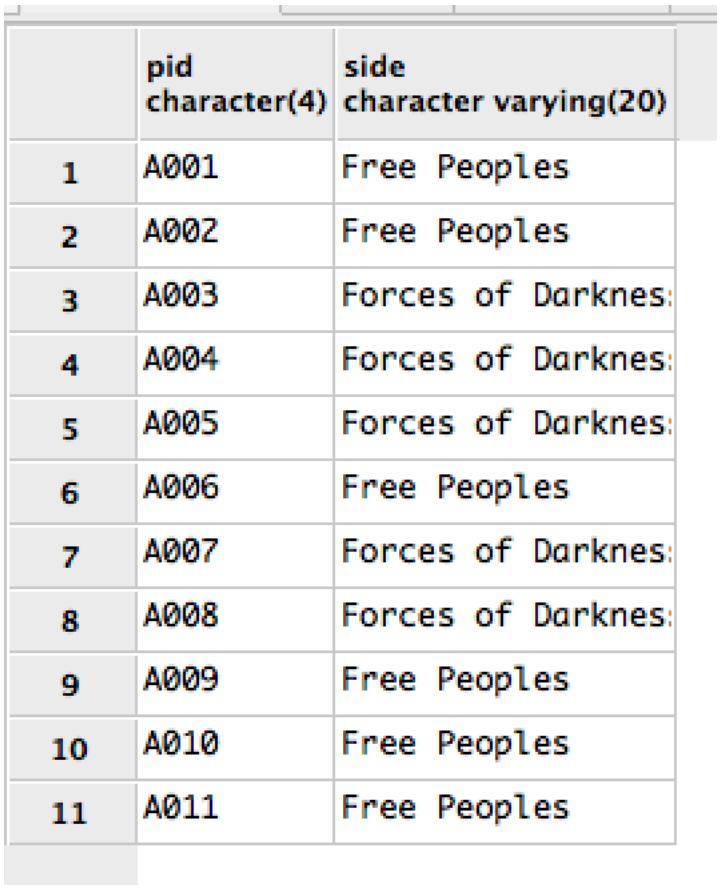
## SQL Create Statement

```
Drop table if exists Sides;
Create table Sides(
Pid                       char(4) not null,
Side                      varchar(20) not null,
Primary Key (Pid)
);
```

## Functional Dependencies

**Sides**: Pid → Side

## Sample Data



| | pid character(4) | side character varying(20) |
|---|---|---|
| 1 | A001 | Free Peoples |
| 2 | A002 | Free Peoples |
| 3 | A003 | Forces of Darknes: |
| 4 | A004 | Forces of Darknes: |
| 5 | A005 | Forces of Darknes: |
| 6 | A006 | Free Peoples |
| 7 | A007 | Forces of Darknes: |
| 8 | A008 | Forces of Darknes: |
| 9 | A009 | Free Peoples |
| 10 | A010 | Free Peoples |
| 11 | A011 | Free Peoples |

## Positon_of_Power Table

The Position_of_Power Table has 1 unique field called TName. TName is the Position that the character has earned him or herself. These are designed to keep track of who is in what position. A good example of this is how Aragorn is a King.

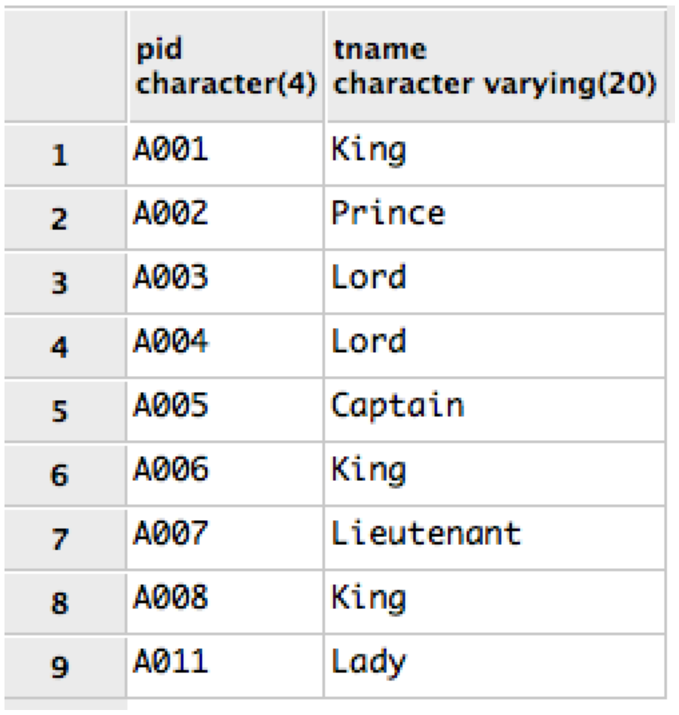
## SQL Create Statement

```
Drop table if exists Postion_of_Power;
Create table Position_of_Power(
Pid                 char(4) not null references
People(Pid),
TName           varchar(20) not null,
Primary Key (Pid)
);
```

## Functional Dependencies

**People**: Pid → TName

## Sample Data



| | pid character(4) | tname character varying(20) |
|---|---|---|
| 1 | A001 | King |
| 2 | A002 | Prince |
| 3 | A003 | Lord |
| 4 | A004 | Lord |
| 5 | A005 | Captain |
| 6 | A006 | King |
| 7 | A007 | Lieutenant |
| 8 | A008 | King |
| 9 | A011 | Lady |

## Realms Table

The Realm Table has 1 unique field Current_Realm_Name. This is designed to keep track of the Realm each character currently fights for or represents. For Example, Gondor is a Realm.

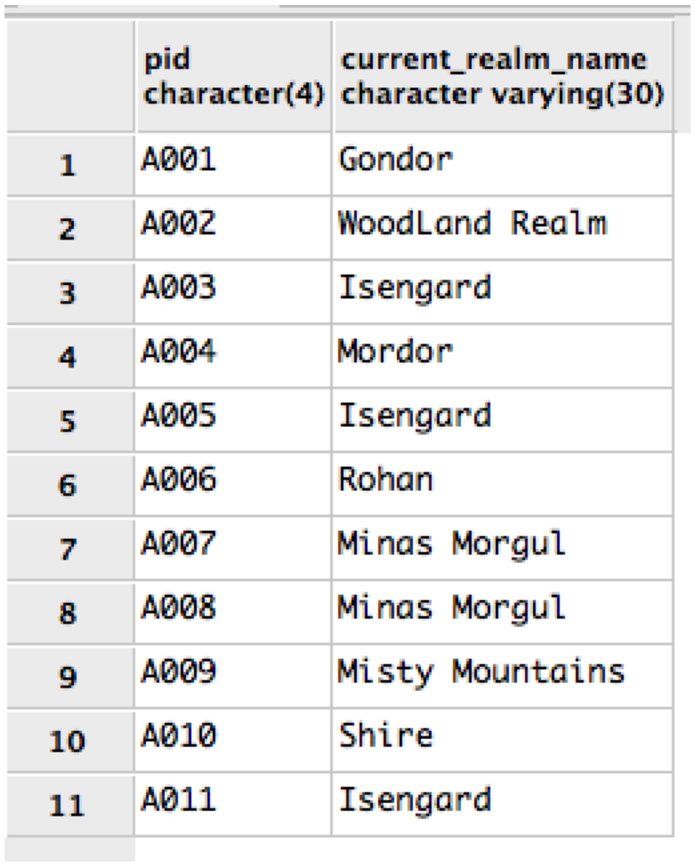
## SQL Create Statement

```
Drop table if exists Realms;
Create table Realms(
Rid                        char(4) not null,
Current_Realm_Name        varchar(40) not null,
Primary key(Pid)
);
```

## Functional Dependencies

**Realms**: Pid → Current_Realm_Name

## Sample Data



| | pid character(4) | current_realm_name character varying(30) |
|---|---|---|
| 1 | A001 | Gondor |
| 2 | A002 | WoodLand Realm |
| 3 | A003 | Isengard |
| 4 | A004 | Mordor |
| 5 | A005 | Isengard |
| 6 | A006 | Rohan |
| 7 | A007 | Minas Morgul |
| 8 | A008 | Minas Morgul |
| 9 | A009 | Misty Mountains |
| 10 | A010 | Shire |
| 11 | A011 | Isengard |

## People_To_Aliases Table

The People_To_Alias Table has no unique fields. Made only to prevent many to many relationship.

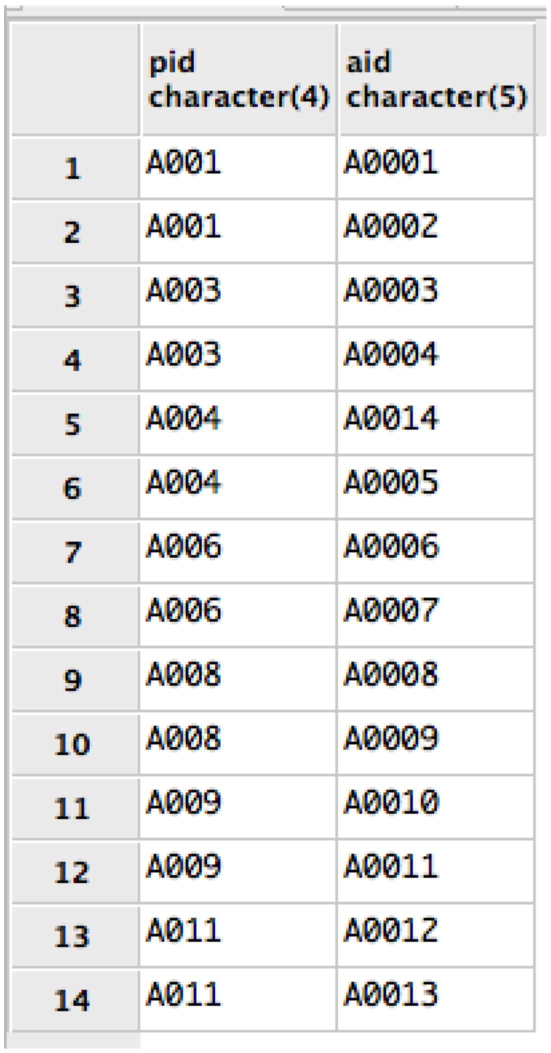
## SQL Create Statement

```
Drop table if exists People_To_Aliases;
Create table People_To_Aliases(
Pid             char(4) not null references
People(Pid),
Aid             char(5) not null references
Aliases(Aid),
Primary Key(Pid,Aid)
);
```

## Functional Dependencies

**People_To_Alias**: No Dependency

## Sample Data



| | pid character(4) | aid character(5) |
|---|---|---|
| 1 | A001 | A0001 |
| 2 | A001 | A0002 |
| 3 | A003 | A0003 |
| 4 | A003 | A0004 |
| 5 | A004 | A0014 |
| 6 | A004 | A0005 |
| 7 | A006 | A0006 |
| 8 | A006 | A0007 |
| 9 | A008 | A0008 |
| 10 | A008 | A0009 |
| 11 | A009 | A0010 |
| 12 | A009 | A0011 |
| 13 | A011 | A0012 |
| 14 | A011 | A0013 |

## Aliases Table

The Aliases Table has 1 unique field Alias. Each character can have multiple aliases.

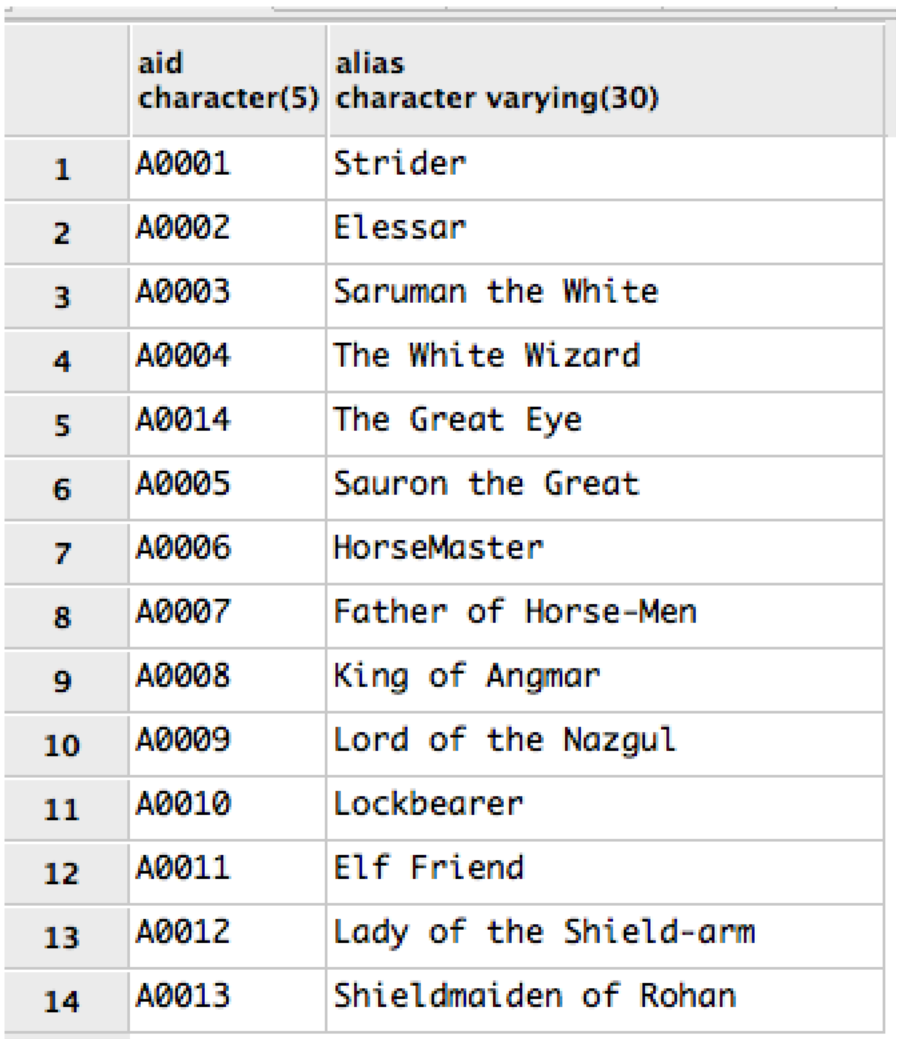
## SQL Create Statement

```
Drop table if exists Aliases;
Create table Aliases(
Aid                      char(5) not null,
Alias                varchar(40) not null,
Primary Key (Aid)
);
```

## Functional Dependencies

**People**: Aid → Alias

## Sample Data



| | aid character(5) | alias character varying(30) |
|---|---|---|
| 1 | A0001 | Strider |
| 2 | A0002 | Elessar |
| 3 | A0003 | Saruman the White |
| 4 | A0004 | The White Wizard |
| 5 | A0014 | The Great Eye |
| 6 | A0005 | Sauron the Great |
| 7 | A0006 | HorseMaster |
| 8 | A0007 | Father of Horse-Men |
| 9 | A0008 | King of Angmar |
| 10 | A0009 | Lord of the Nazgul |
| 11 | A0010 | Lockbearer |
| 12 | A0011 | Elf Friend |
| 13 | A0012 | Lady of the Shield-arm |
| 14 | A0013 | Shieldmaiden of Rohan |

## views

The following view is designed to show the selected characters current status and Position of Power in their individual realms along with the alias they may also go by.

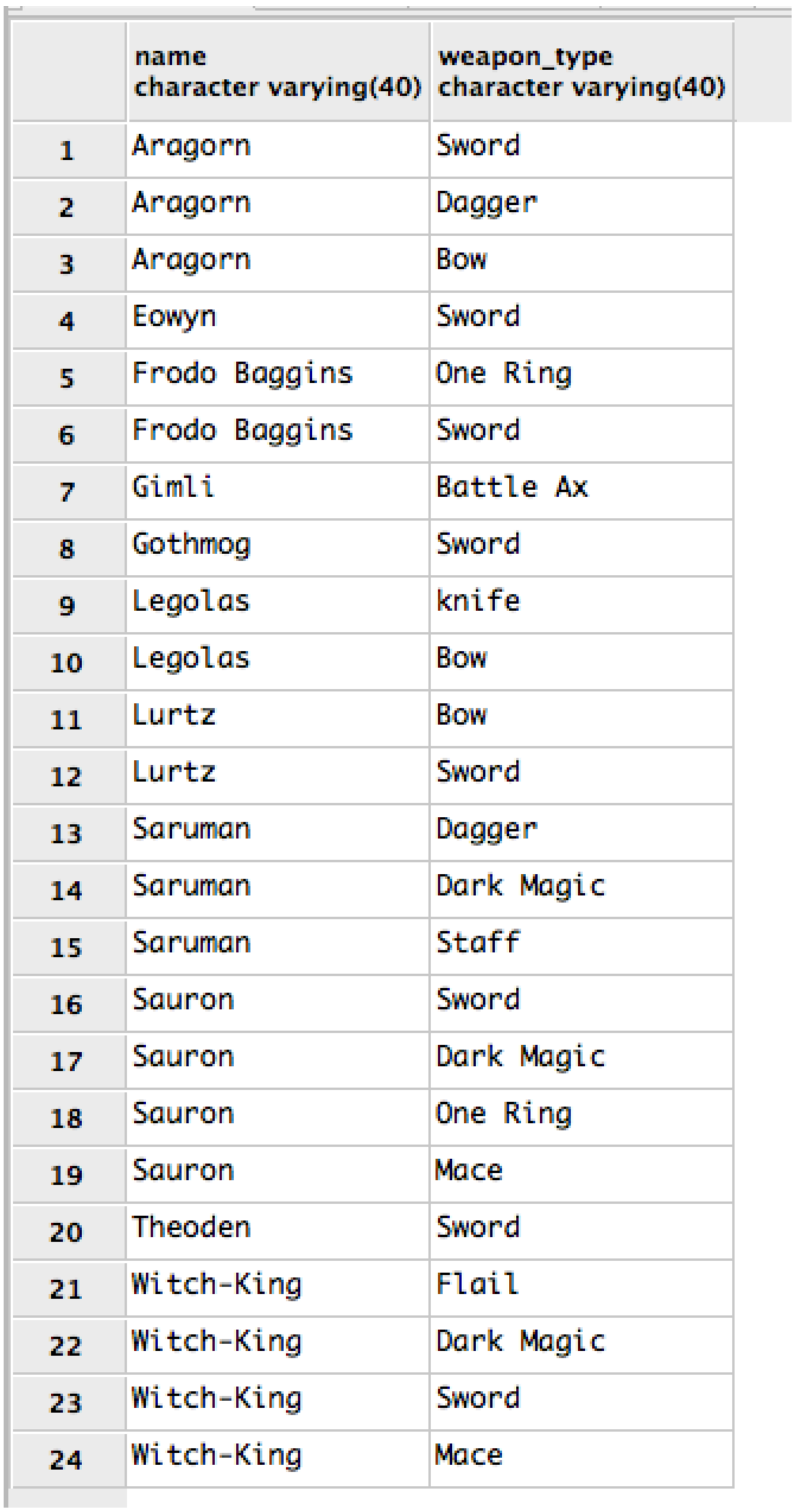
```
Create View Current_Character_Status as
Select
P.Name,A.Alias,PP.TName,R.Current_Realm_Name
From
People as P, Aliases as A, People_To_Aliases as
PA, Position_of_Power as PP, Realms as R
Where
 PP.Pid=P.Pid
And  P.Pid=PA.Pid
And PA.Aid=A.Aid
And P.Pid=R.Pid
```

## Reports and Queries

**These two reports show who wields which type of weapon and who is currently ruling the realm.**

### Query: Which weapon each individual character uses

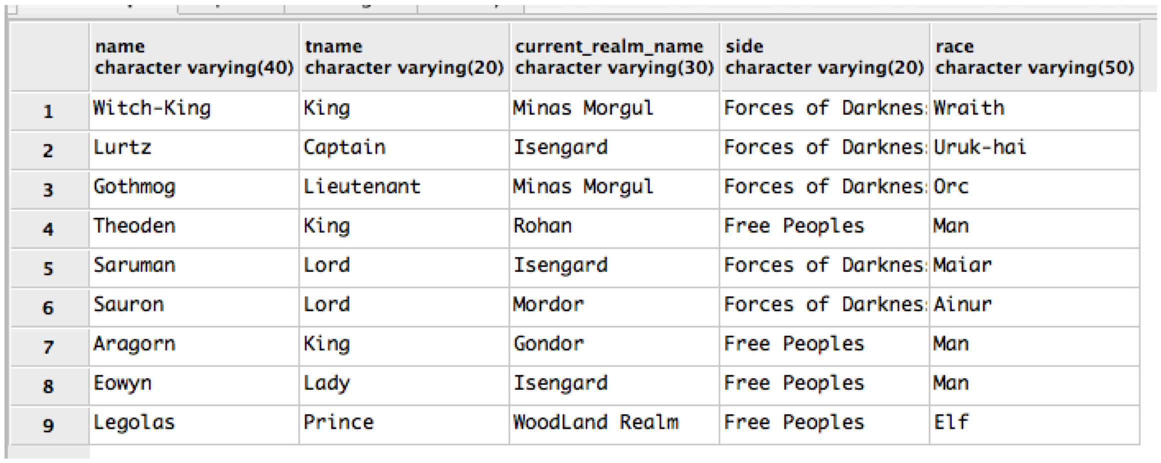
```
Select
 P.Name, W.Weapon_Type
From
People as P, People_to_Weapon as PW, Weapons as
W
Where
P.Pid=PW.Pid
and PW.Wid=W.Wid
order by P.Name ASC;
```



| | name<br>character varying(40) | weapon_type<br>character varying(40) |
|---|---|---|
| 1 | Aragorn | Sword |
| 2 | Aragorn | Dagger |
| 3 | Aragorn | Bow |
| 4 | Eowyn | Sword |
| 5 | Frodo Baggins | One Ring |
| 6 | Frodo Baggins | Sword |
| 7 | Gimli | Battle Ax |
| 8 | Gothmog | Sword |
| 9 | Legolas | knife |
| 10 | Legolas | Bow |
| 11 | Lurtz | Bow |
| 12 | Lurtz | Sword |
| 13 | Saruman | Dagger |
| 14 | Saruman | Dark Magic |
| 15 | Saruman | Staff |
| 16 | Sauron | Sword |
| 17 | Sauron | Dark Magic |
| 18 | Sauron | One Ring |
| 19 | Sauron | Mace |
| 20 | Theoden | Sword |
| 21 | Witch-King | Flail |
| 22 | Witch-King | Dark Magic |
| 23 | Witch-King | Sword |
| 24 | Witch-King | Mace |

## Query: Which Race rules which Realm along with who is currently ruling said realm.

```
Select
 distinct P.Name, PP.TName,
R.Current_Realm_Name,S.Side ,P.Race
from
People as P,
Position_of_Power as PP,
Realms as R,
Sides as S
where
PP.Pid=P.Pid
 and P.Pid=R.Pid
and P.Pid=S.Pid;
```



| | name character varying(40) | tname character varying(20) | current_realm_name character varying(30) | side character varying(20) | race character varying(50) |
|---|---|---|---|---|---|
| 1 | Witch-King | King | Minas Morgul | Forces of Darknes: | Wraith |
| 2 | Lurtz | Captain | Isengard | Forces of Darknes: | Uruk-hai |
| 3 | Gothmog | Lieutenant | Minas Morgul | Forces of Darknes: | Orc |
| 4 | Theoden | King | Rohan | Free Peoples | Man |
| 5 | Saruman | Lord | Isengard | Forces of Darknes: | Maiar |
| 6 | Sauron | Lord | Mordor | Forces of Darknes: | Ainur |
| 7 | Aragorn | King | Gondor | Free Peoples | Man |
| 8 | Eowyn | Lady | Isengard | Free Peoples | Man |
| 9 | Legolas | Prince | WoodLand Realm | Free Peoples | Elf |

## stored procedures

The following stored procedure is designed to find out which characters fight for which realm. It calls on both the Realms table and the People table.

```
CREATE FUNCTION
character_realm(Current_Realm_Name varchar(30))

Returns table (Name varchar(40),
Current_Realm_Name varchar(30)) as $$

Select
 P.Name, R.Current_Realm_Name
From People as P, Realms as R
Where P.Pid=R.Pid
$$ language ‘sql’;

select * from
character_realm(‘Current_Realm_Name’);
```

**Designed to prevent nulls by the possibility that a character isn’t even in the war. An example of this is the Tree beard at first.**

```
CREATE FUNCTION check_character_side() RETURNS
TRIGGER AS $$
```

```
Begin If (side=null )
Then
Update Sides set side= 'Neutral' where
Pid=null;
End if; End;
 $$language plpgsql
```

## triggers

The following trigger calls the previously created stored procedure check_character_side() to discover if there were any null values submitted for side. If there is in fact a null value, than it changes it to the specified value in the stored procedure

```
Create Trigger check_character_side Before
Insert on Sides
For EACH ROW EXECUTE PROCEDURE
check_character_side();
```

## SECURITY

Security is essentially the creation of different access levels on the database. The highest access level will be the Admin who is created to modify the database. This ranges from updating, inserting, viewing, and deleting. The other possible access level is simply User level. The User will only be able to view tables.

ADMIN CREATION:
```
Create Admin LOTRADMIN WITH  PASSWORD ‘alpaca’;

Grant select, insert, update, delete on People,
Realms, Sides, Position_of_Power, Weapons,
People_to_weapon, People_to_Aliases, Aliases TO
LOTRADMIN;
```

USER CREATION:
```
Create User LOTRUSER WITH PASSWORD ‘alpaca’;

Revoke all on People, Realms, Sides,
Position_of_Power, Weapons, People_to_weapon,
People_to_Aliases, Aliases FROM LOTRUSER;
```

## known problems, implementation and future enhancements

**Known Problems**: There can be the blatantly obvious problem of running out of unique characters for any single one of my ids. Another possible issue involves Gandalf, who rebirths himself into a white wizard changing everything about himself. Only possible solution is having Gandalf in the database twice as both the White and the Grey.

**Implementation**: Implementation involves copying and pasting the code I have in the document. If the copying and pasting process is a success, there should be no issue.

**Future Enhancements**: Besides from adding more data to the database, I can think of adding another two possible tables. Designed specifically for mounts. Another future Enhancement can be adding the specific names for each special weapon.