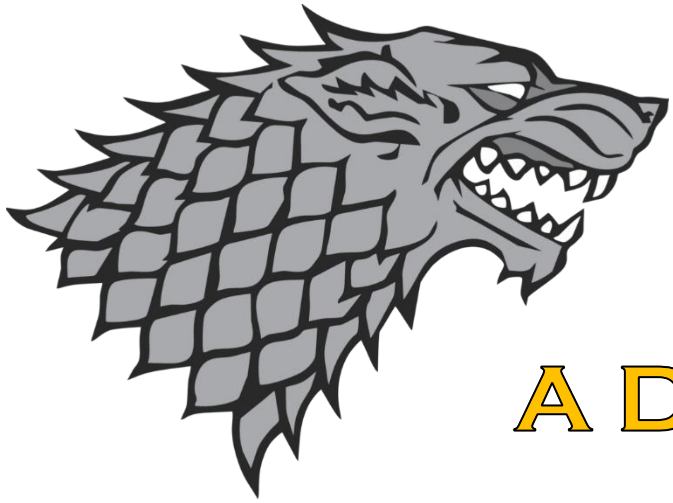


A Database of Ice and Fire

Designed by Liam Harwood

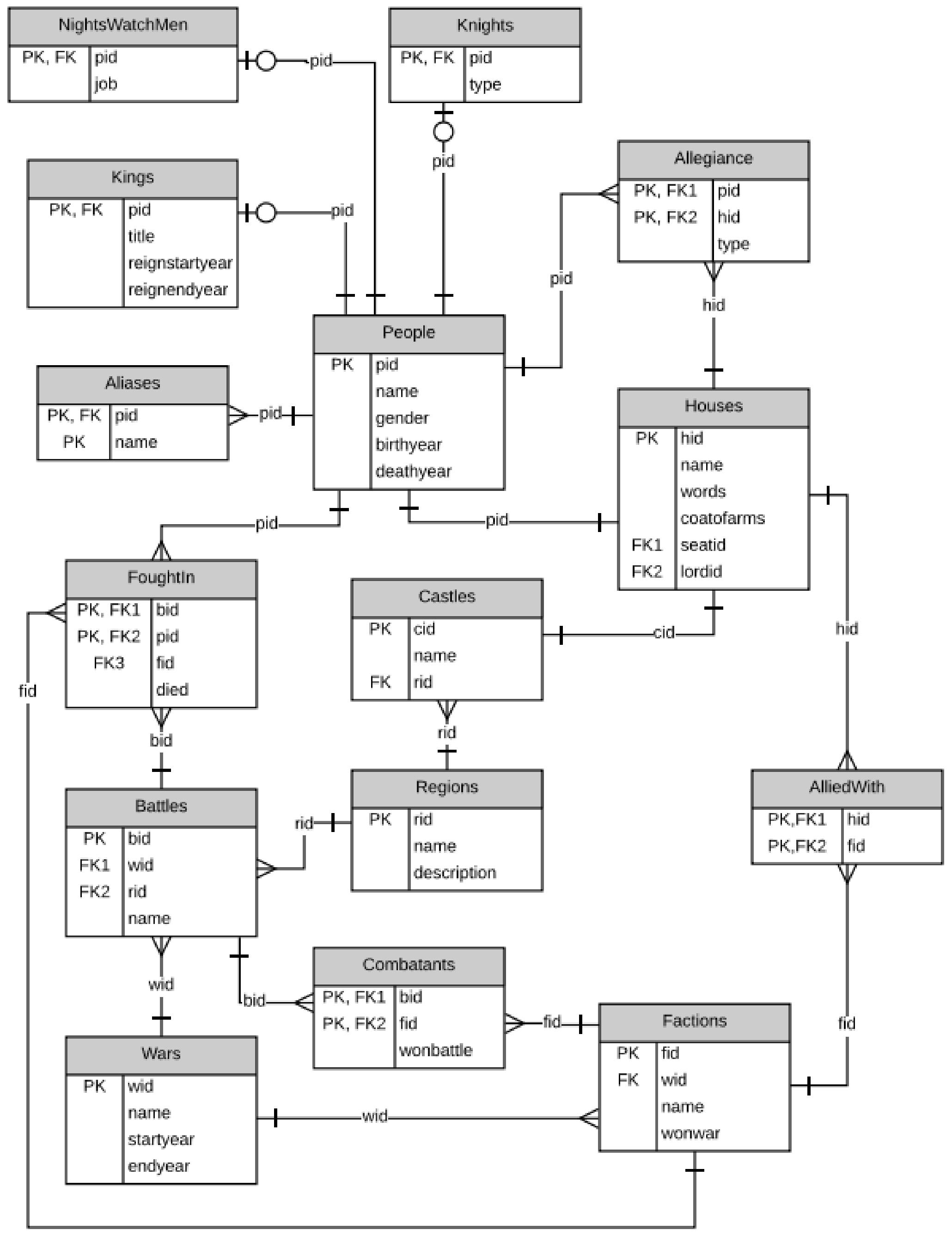
Table of Contents:

Executive Summary

This document outlines the design and implementation of a database which holds up-to-date historical data on the people, houses, and conflicts of Westeros. First, the ER diagram of the database will be presented, followed by a description of each table and its SQL create statements. Sample data for each table will also be shown. Next, the views, reports, stored procedures, and triggers implemented in this database will be discussed, along with sample output. Then, there will be a description of all roles and security privileges suggested for this database. Finally, notes on the implementation will be given, followed by known issues and ideas for future improvement.

This implementation is intended to be used by the maesters of the Citadel for the purpose of furthering the documentation and analysis of Westerosi history. This database will allow for easier keeping of records concerning the people, places, and events of Westeros. Maesters will be able to gain useful information from queries that will provide interesting and valuable historical data. The goal of this database implementation is to provide the Citadel with a functional, normalized database that will make the process of keeping track of historical events (past, present, and future) easier and more efficient.

ER Diagram


NightsWatchMen
PK, FK pid
job
Knights
PK, FK pid
type
Kings
PK, FK pid
title
reignstartyear
reignendyear
Allegiance
PK, FK1 pid
PK, FK2 hid
type
People
PK pid
name
gender
birthyear
deathyear
Aliases
PK, FK pid
PK name
Houses
PK hid
name
words
coatofarms
FK1 seatid
FK2 lordid
FoughtIn
PK, FK1 bid
PK, FK2 pid
FK3 fid
died
Castles
PK cid
name
FK rid
Regions
PK rid
name
description
AlliedWith
PK,FK1 hid
PK,FK2 fid
Battles
PK bid
FK1 wid
FK2 rid
name
Combatants
PK, FK1 bid
PK, FK2 fid
wonbattle
Factions
PK fid
FK wid
name
wonwar
Wars
PK wid
name
startyear
endyear
pid
hid
cid
rid
bid
wid
fid

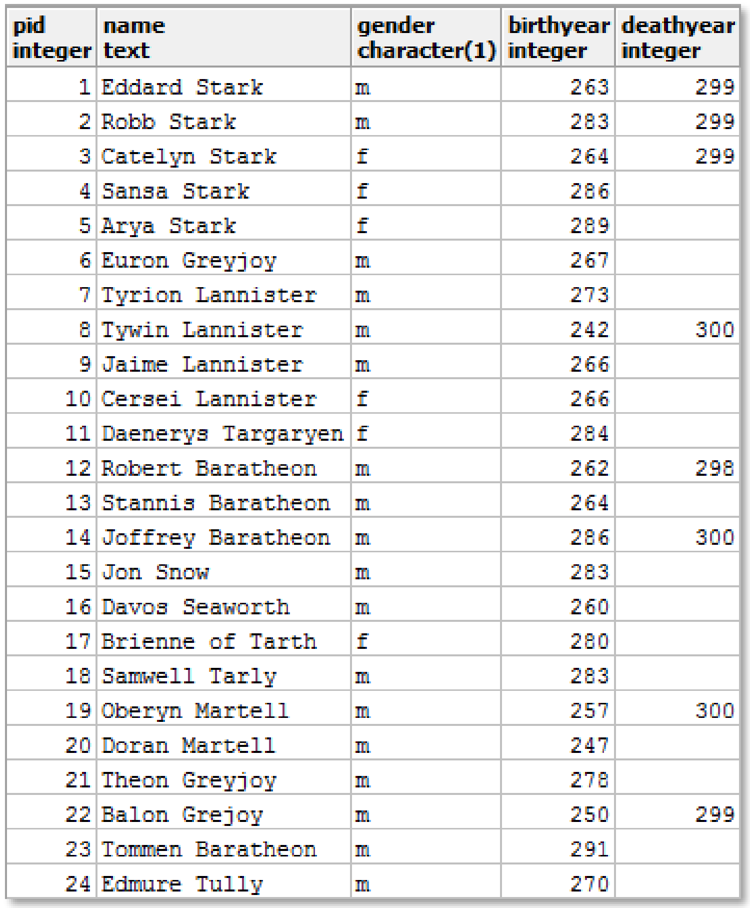
Tables

People: The people table contains all people and their basic attributes, which are shared with the subtypes Kings, Knights, and Night’s Watchmen.

```
create table People (
   pid       integer not null,
   name      text    not null,
   gender    char(1) not null check(gender='m' or gender='f'),
   birthyear integer not null,
   deathyear integer,
  primary key(pid)
);
```

Functional Dependencies:

pid → name, gender, birthyear, deathyear

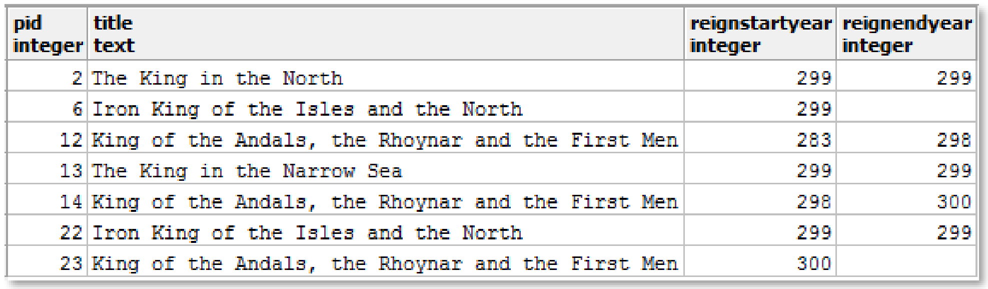


pid integer	name text	gender character(1)	birthyear integer	deathyear integer
1	Eddard Stark	m	263	299
2	Robb Stark	m	283	299
3	Catelyn Stark	f	264	299
4	Sansa Stark	f	286	
5	Arya Stark	f	289	
6	Euron Greyjoy	m	267	
7	Tyrion Lannister	m	273	
8	Tywin Lannister	m	242	300
9	Jaime Lannister	m	266	
10	Cersei Lannister	f	266	
11	Daenerys Targaryen	f	284	
12	Robert Baratheon	m	262	298
13	Stannis Baratheon	m	264	
14	Joffrey Baratheon	m	286	300
15	Jon Snow	m	283	
16	Davos Seaworth	m	260	
17	Brienne of Tarth	f	280	
18	Samwell Tarly	m	283	
19	Oberyn Martell	m	257	300
20	Doran Martell	m	247	
21	Theon Greyjoy	m	278	
22	Balon Grejoy	m	250	299
23	Tommen Baratheon	m	291	
24	Edmure Tully	m	270	

Sample Data 1: People

Kings: Subtype of People. Contains all people that are kings and basic information about their reign.

```
create table Kings (
   pid            integer not null references People(pid),
   title          text    not null,
   reignstartyear integer not null,
   reignendyear   integer,
  primary key(pid)
);
```



pid integer	title text	reignstartyear integer	reignendyear integer
2	The King in the North	299	299
6	Iron King of the Isles and the North	299	
12	King of the Andals, the Rhoynar and the First Men	283	298
13	The King in the Narrow Sea	299	299
14	King of the Andals, the Rhoynar and the First Men	298	300
22	Iron King of the Isles and the North	299	299
23	King of the Andals, the Rhoynar and the First Men	300	

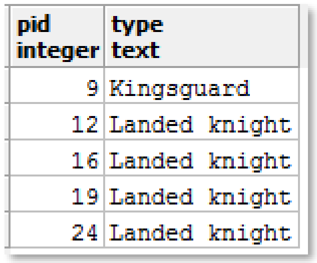
Sample Data 2: Kings

Functional Dependencies:

pid → title, reignstartyear, reignendyear

Knights: Subtype of People. Contains all people that are knights and the type of knight that each person is.

```
create table Knights (
   pid  integer not null references People(pid),
   type text,
  primary key(pid)
);
```



pid integer	type text
9	Kingsguard
12	Landed knight
16	Landed knight
19	Landed knight
24	Landed knight

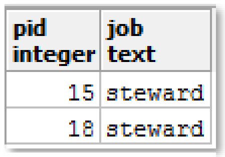
Sample Data 3: Knights

Functional Dependencies:

pid → type

Night's Watchmen: Subtype of People. Contains all people that are members of the Night's Watch as well as their job in the Watch.

```
create table NightsWatchMen (
   pid integer not null references People(pid),
   job text    check(job='ranger' or job='steward' or job='builder'),
  primary key(pid)
);
```



pid integer	job text
15	steward
18	steward

Sample Data 4: Night's Watchmen

Functional Dependencies:

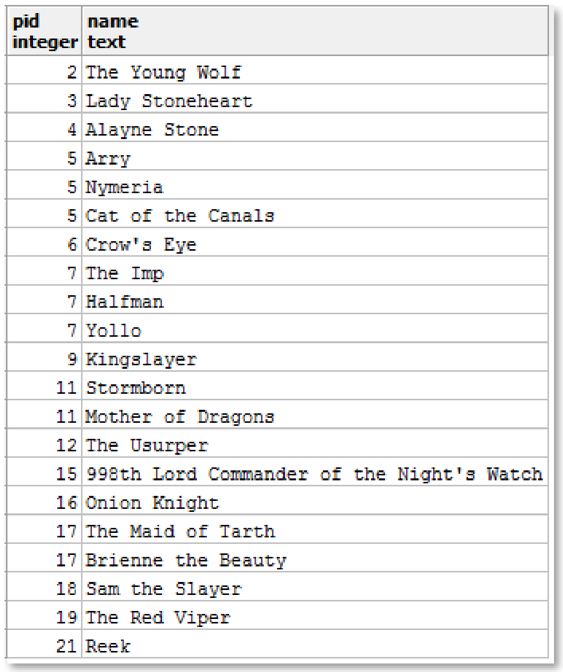
pid → job

Aliases: Contains any aliases, nicknames, or assumed identities that people have used.

```
create table Aliases (
   pid  integer not null references People(pid),
   name text    not null,
  primary key(pid, name)
);
```

Functional Dependencies:

(pid, name) →

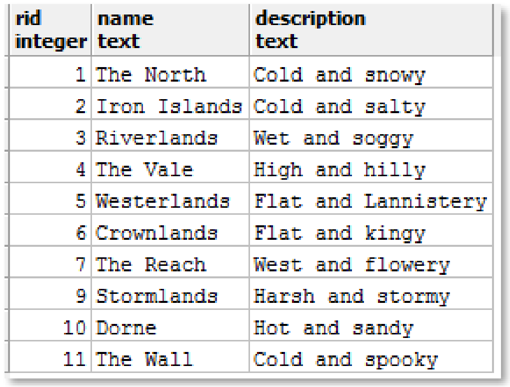


pid integer	name text
2	The Young Wolf
3	Lady Stoneheart
4	Alayne Stone
5	Arry
5	Nymeria
5	Cat of the Canals
6	Crow's Eye
7	The Imp
7	Halfman
7	Yollo
9	Kingslayer
11	Stormborn
11	Mother of Dragons
12	The Usurper
15	998th Lord Commander of the Night's Watch
16	Onion Knight
17	The Maid of Tarth
17	Brienne the Beauty
18	Sam the Slayer
19	The Red Viper
21	Reek

Sample Data 5: Aliases

Regions: Contains the different geographic regions of Westeros.

```
create table Regions (
   rid         integer not null,
   name        text    not null,
   description text    not null,
  primary key(rid)
);
```



rid integer	name text	description text
1	The North	Cold and snowy
2	Iron Islands	Cold and salty
3	Riverlands	Wet and soggy
4	The Vale	High and hilly
5	Westerlands	Flat and Lannistery
6	Crownlands	Flat and kingy
7	The Reach	West and flowery
9	Stormlands	Harsh and stormy
10	Dorne	Hot and sandy
11	The Wall	Cold and spooky

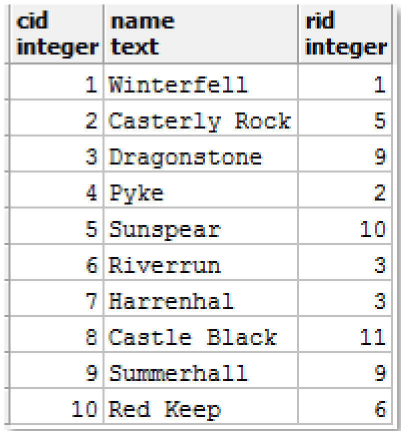
Sample Data 6: Regions

Functional Dependencies:

rid → name, description

Castles: Contains the different castles and strongholds as well as their locations.

```
create table Castles (
   cid  integer not null,
   name text    not null,
   rid  integer not null references Regions(rid),
  primary key(cid)
);
```



cid integer	name text	rid integer
1	Winterfell	1
2	Casterly Rock	5
3	Dragonstone	9
4	Pyke	2
5	Sunspear	10
6	Riverrun	3
7	Harrenhal	3
8	Castle Black	11
9	Summerhall	9
10	Red Keep	6

Sample Data 7: Castles

Functional Dependencies:

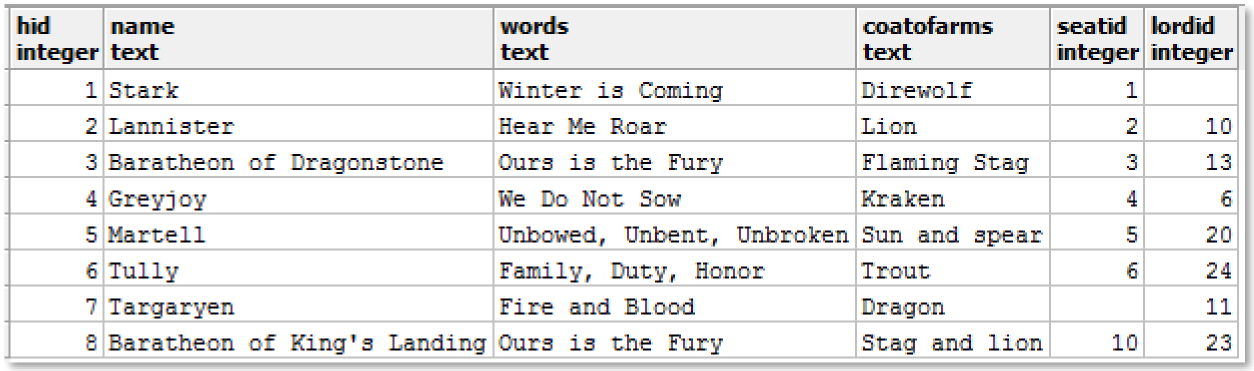
cid → name, rid

Houses: Contains the Great Houses (families) of Westeros as well as basic information about each house. (Note: “seatid” is a foreign key referencing the ID of the current castle where the house is based and “lordid” is a foreign key referencing the ID of the person who is currently the lord of the house.)

```
create table Houses (
   hid          integer not null,
   name         text    not null,
   words        text,
   coatofarms   text,
   seatid       integer references Castles(cid),
   lordid       integer references People(pid),
  primary key(hid)
);
```

Functional Dependencies:

Hid → name, words, coatofarms, seatid, lordid

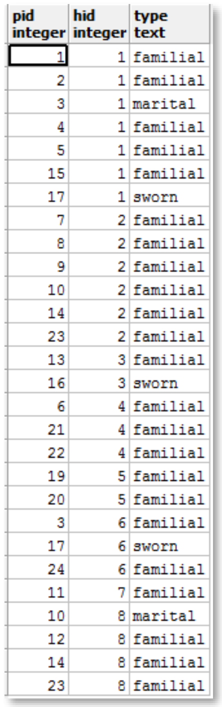


hid integer	name text	words text	coatofarms text	seatid integer	lordid integer
1	Stark	Winter is Coming	Direwolf	1	
2	Lannister	Hear Me Roar	Lion	2	10
3	Baratheon of Dragonstone	Ours is the Fury	Flaming Stag	3	13
4	Greyjoy	We Do Not Sow	Kraken	4	6
5	Martell	Unbowed, Unbent, Unbroken	Sun and spear	5	20
6	Tully	Family, Duty, Honor	Trout	6	24
7	Targaryen	Fire and Blood	Dragon		11
8	Baratheon of King's Landing	Ours is the Fury	Stag and lion	10	23

Sample Data 8: Houses

Allegiance: Contains the current allegiances of each person, i.e. the houses that they are loyal to, as well as the nature of this allegiance (familial, marital, or sworn vow).

```
create table Allegiance (
   pid  integer not null references People(pid),
   hid  integer not null references Houses(hid),
   type text    check(type='marital' or type='familial' or type='sworn'),
  primary key(pid, hid)
);
```



pid integer	hid integer	type text
1	1	familial
2	1	familial
3	1	marital
4	1	familial
5	1	familial
15	1	familial
17	1	sworn
7	2	familial
8	2	familial
9	2	familial
10	2	familial
14	2	familial
23	2	familial
13	3	familial
16	3	sworn
6	4	familial
21	4	familial
22	4	familial
19	5	familial
20	5	familial
3	6	familial
17	6	sworn
24	6	familial
11	7	familial
10	8	marital
12	8	familial
14	8	familial
23	8	familial

Sample Data 9: Allegiance

Functional Dependencies:

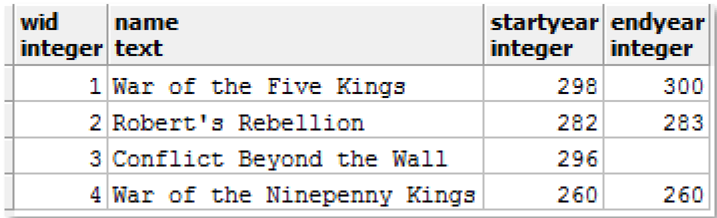
(pid, hid) → type

Wars: Contains different wars that have happened or are currently happening, as well as some basic information about each war.

```
create table Wars (
   wid        integer not null,
   name       text    not null,
   startyear  integer not null,
   endyear    integer,
  primary key(wid)
);
```

Functional Dependencies:

wid → name, startyear, endyear

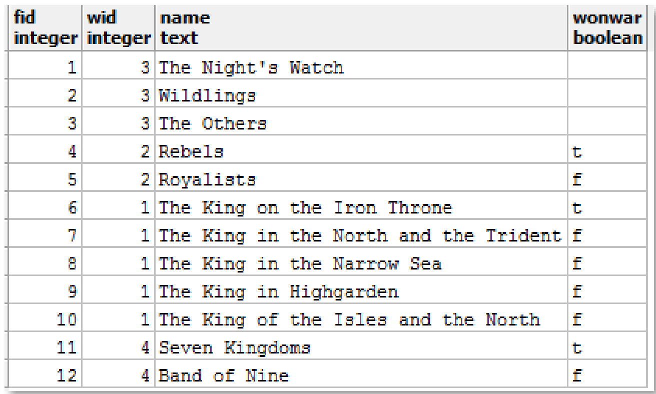


wid integer	name text	startyear integer	endyear integer
1	War of the Five Kings	298	300
2	Robert's Rebellion	282	283
3	Conflict Beyond the Wall	296	
4	War of the Ninepenny Kings	260	260

Sample Data 10: Wars

Factions: Contains the factions that were in conflict in each war as well as basic information about each faction. (Note: “wonwar” is true if the faction won the war, false if the faction lost the war, and null if the war is ongoing.)

```
create table Factions (
   fid          integer  not null,
   wid          integer  not null references Wars(wid),
   name         text     not null,
   wonwar       boolean,
  primary key(fid)
);
```



fid integer	wid integer	name text	wonwar boolean
1	3	The Night's Watch	
2	3	Wildlings	
3	3	The Others	
4	2	Rebels	t
5	2	Royalists	f
6	1	The King on the Iron Throne	t
7	1	The King in the North and the Trident	f
8	1	The King in the Narrow Sea	f
9	1	The King in Highgarden	f
10	1	The King of the Isles and the North	f
11	4	Seven Kingdoms	t
12	4	Band of Nine	f

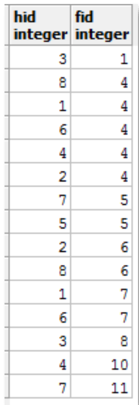
Sample Data 11: Factions

Functional Dependencies:

fid → wid, name, wonwar

AlliedWith: Contains the factions that each house was allied with in the different wars.

```
create table AlliedWith (
   hid integer not null references Houses(hid),
   fid integer not null references Factions(fid),
  primary key(hid, fid)
);
```



hid integer	fid integer
3	1
8	4
1	4
6	4
4	4
2	4
7	5
5	5
2	6
8	6
1	7
6	7
3	8
4	10
7	11

Sample Data 12: AlliedWith

Functional Dependencies:

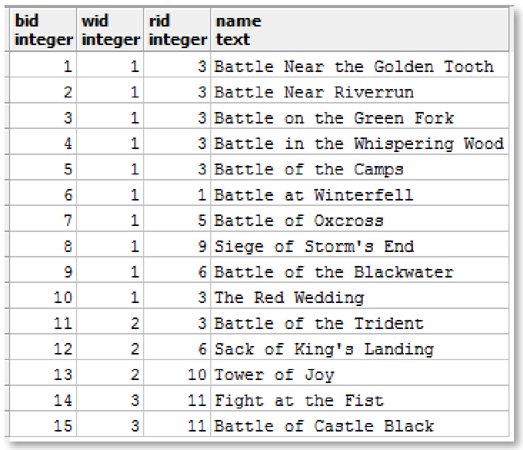
(hid, fid) →

Battles: Contains the battles that were fought during the different wars as well as basic information about each battle.

```
create table Battles (
   bid  integer not null,
   wid  integer not null references Wars(wid),
   rid  integer not null references Regions(rid),
   name text    not null,
  primary key(bid)
);
```

Functional Dependencies:

bid → wid, rid, name



bid integer	wid integer	rid integer	name text
1	1	3	Battle Near the Golden Tooth
2	1	3	Battle Near Riverrun
3	1	3	Battle on the Green Fork
4	1	3	Battle in the Whispering Wood
5	1	3	Battle of the Camps
6	1	1	Battle at Winterfell
7	1	5	Battle of Oxcross
8	1	9	Siege of Storm's End
9	1	6	Battle of the Blackwater
10	1	3	The Red Wedding
11	2	3	Battle of the Trident
12	2	6	Sack of King's Landing
13	2	10	Tower of Joy
14	3	11	Fight at the Fist
15	3	11	Battle of Castle Black

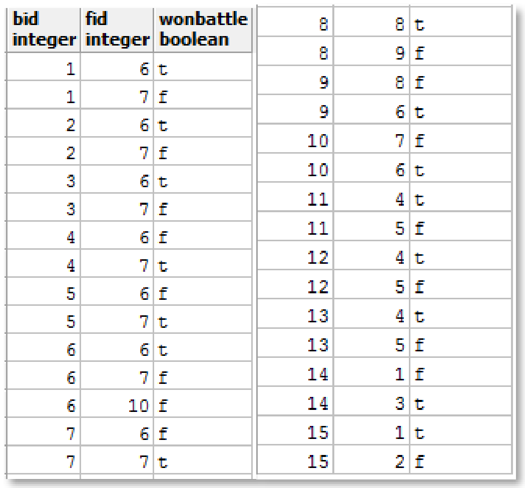
Sample Data 13: Battles

Combatants: Contains the factions that fought in each battle as well as whether or not they won the battle.

```
create table Combatants (
   bid       integer not null references Battles(bid),
   fid       integer not null references Factions(fid),
   wonbattle boolean,
  primary key(bid, fid)
);
```

Functional Dependencies:

(bid, fid) → wonbattle



bid integer	fid integer	wonbattle boolean
1	6	t
1	7	f
2	6	t
2	7	f
3	6	t
3	7	f
4	6	f
4	7	t
5	6	f
5	7	t
6	6	t
6	7	f
6	10	f
7	6	f
7	7	t
8	8	t
8	9	f
9	8	f
9	6	t
10	7	f
10	6	t
11	4	t
11	5	f
12	4	t
12	5	f
13	4	t
13	5	f
14	1	f
14	3	t
15	1	t
15	2	f

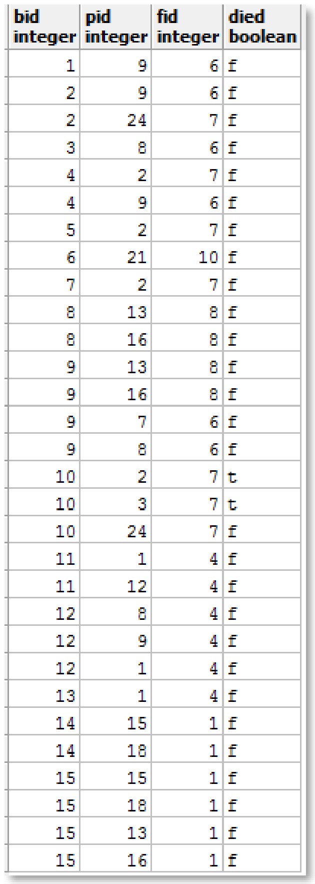
Sample Data 14: Combatants

FoughtIn: Contains the people that fought in each battle as well as the faction they fought for and whether or not they died at that battle.

```
create table FoughtIn (
   bid  integer not null references Battles(bid),
   pid  integer not null references People(pid),
   fid  integer not null references Factions(fid),
   died boolean not null,
  primary key(pid, bid)
);
```

Functional Dependencies:

(pid, bid) → fid, died



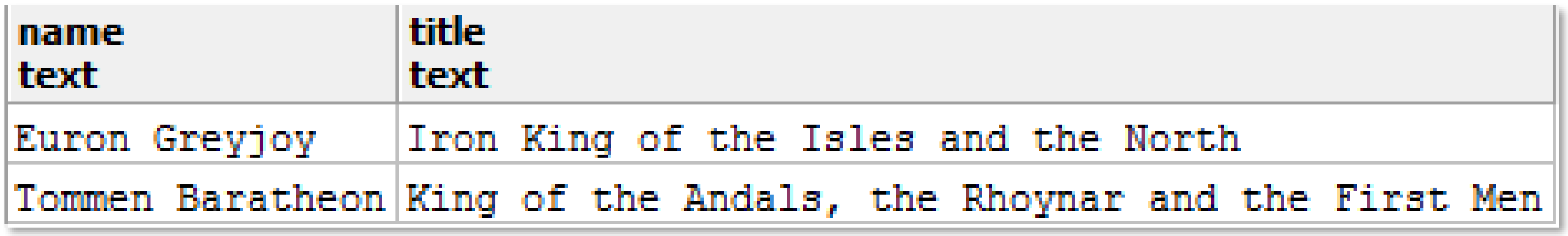
bid integer	pid integer	fid integer	died boolean
1	9	6	f
2	9	6	f
2	24	7	f
3	8	6	f
4	2	7	f
4	9	6	f
5	2	7	f
6	21	10	f
7	2	7	f
8	13	8	f
8	16	8	f
9	13	8	f
9	16	8	f
9	7	6	f
9	8	6	f
10	2	7	t
10	3	7	t
10	24	7	f
11	1	4	f
11	12	4	f
12	8	4	f
12	9	4	f
12	1	4	f
13	1	4	f
14	15	1	f
14	18	1	f
15	15	1	f
15	18	1	f
15	13	1	f
15	16	1	f

Sample Data 15: FoughtIn

Views

LivingKings: Lists the names and titles of all living, reigning kings

```
create view LivingKings as
  select name, title
  from people p inner join kings k on p.pid = k.pid
  where deathyear    is null
    and reignendyear is null
  order by name asc;
```

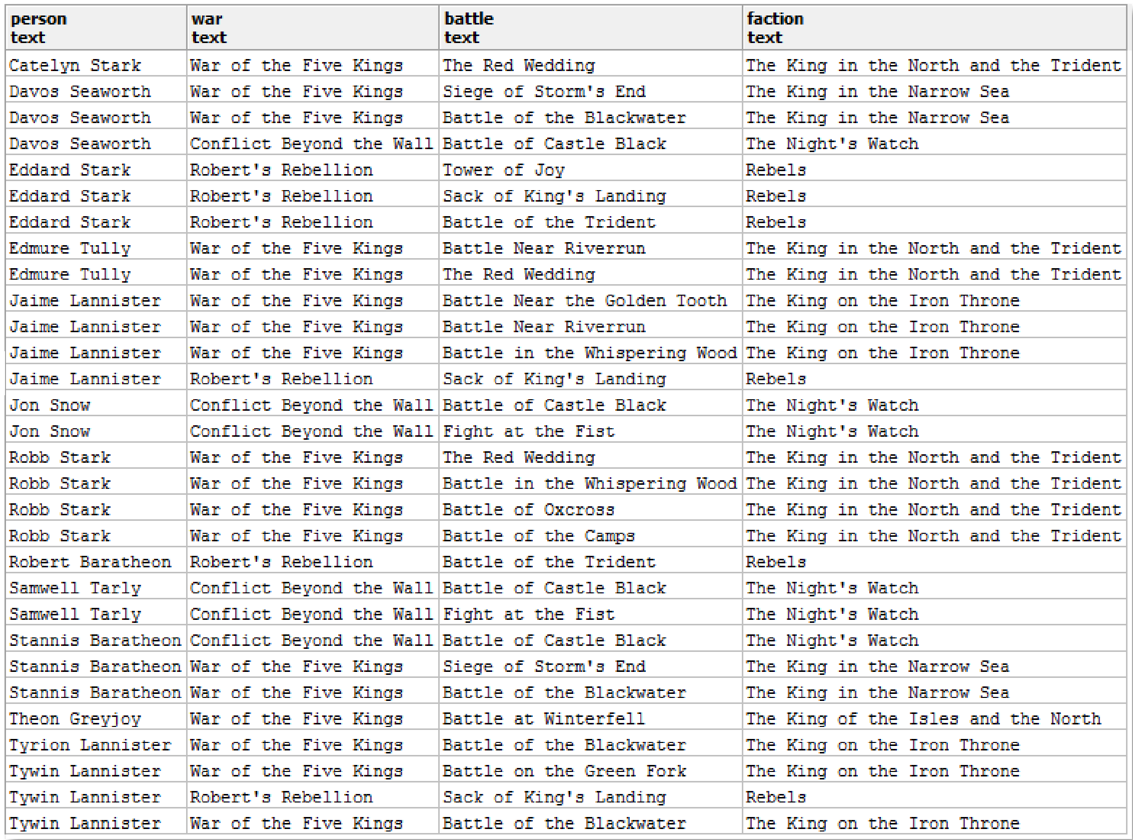


name text	title text
Euron Greyjoy	Iron King of the Isles and the North
Tommen Baratheon	King of the Andals, the Rhoynar and the First Men

Sample Output 1: LivingKings

BattleExperience: Lists the names of all people along with the wars they fought in, each battle they fought in for each war, and the side they were on for each battle

```
create view BattleExperience as
  select p.name as person, w.name as war, b.name as battle, fa.name as faction
  from   people p inner join foughtin fi on p.pid  = fi.pid
               inner join factions fa on fi.fid = fa.fid
                    inner join battles  b  on fi.bid = b.bid
                    inner join wars     w  on b.wid  = w.wid
  order by person asc;
```

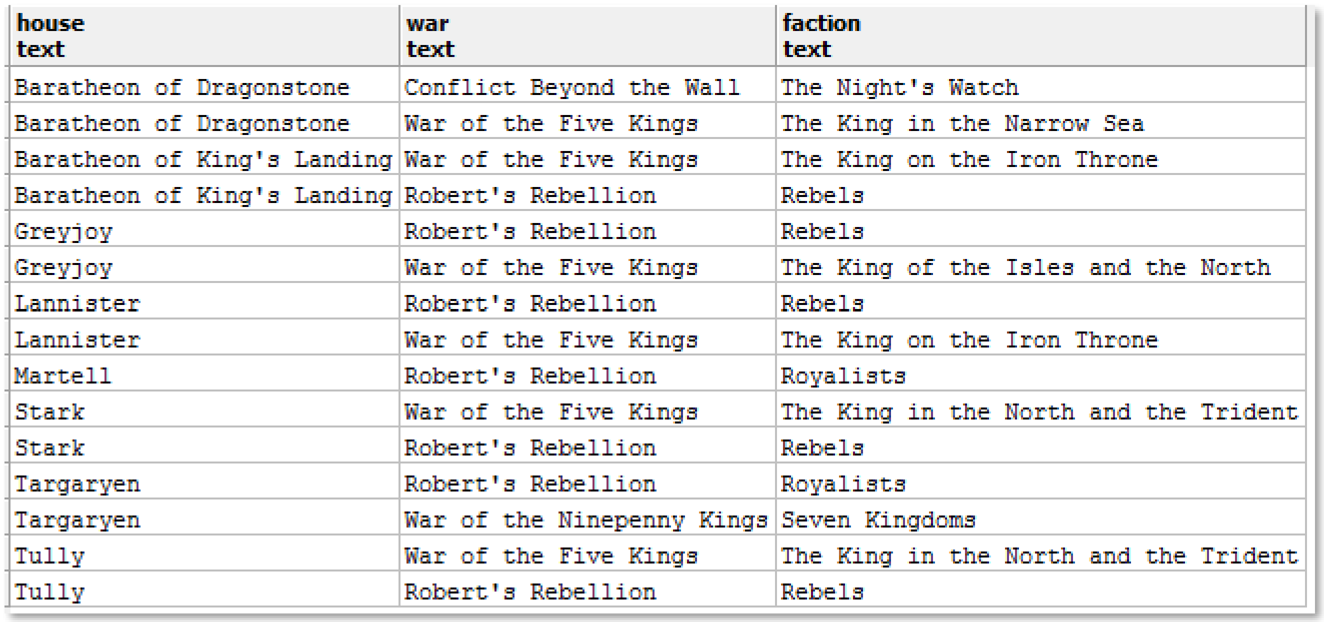


person	war	battle	faction
text	text	text	text
Catelyn Stark	War of the Five Kings	The Red Wedding	The King in the North and the Trident
Davos Seaworth	War of the Five Kings	Siege of Storm's End	The King in the Narrow Sea
Davos Seaworth	War of the Five Kings	Battle of the Blackwater	The King in the Narrow Sea
Davos Seaworth	Conflict Beyond the Wall	Battle of Castle Black	The Night's Watch
Eddard Stark	Robert's Rebellion	Tower of Joy	Rebels
Eddard Stark	Robert's Rebellion	Sack of King's Landing	Rebels
Eddard Stark	Robert's Rebellion	Battle of the Trident	Rebels
Edmure Tully	War of the Five Kings	Battle Near Riverrun	The King in the North and the Trident
Edmure Tully	War of the Five Kings	The Red Wedding	The King in the North and the Trident
Jaime Lannister	War of the Five Kings	Battle Near the Golden Tooth	The King on the Iron Throne
Jaime Lannister	War of the Five Kings	Battle Near Riverrun	The King on the Iron Throne
Jaime Lannister	War of the Five Kings	Battle in the Whispering Wood	The King on the Iron Throne
Jaime Lannister	Robert's Rebellion	Sack of King's Landing	Rebels
Jon Snow	Conflict Beyond the Wall	Battle of Castle Black	The Night's Watch
Jon Snow	Conflict Beyond the Wall	Fight at the Fist	The Night's Watch
Robb Stark	War of the Five Kings	The Red Wedding	The King in the North and the Trident
Robb Stark	War of the Five Kings	Battle in the Whispering Wood	The King in the North and the Trident
Robb Stark	War of the Five Kings	Battle of Oxcross	The King in the North and the Trident
Robb Stark	War of the Five Kings	Battle of the Camps	The King in the North and the Trident
Robert Baratheon	Robert's Rebellion	Battle of the Trident	Rebels
Samwell Tarly	Conflict Beyond the Wall	Battle of Castle Black	The Night's Watch
Samwell Tarly	Conflict Beyond the Wall	Fight at the Fist	The Night's Watch
Stannis Baratheon	Conflict Beyond the Wall	Battle of Castle Black	The Night's Watch
Stannis Baratheon	War of the Five Kings	Siege of Storm's End	The King in the Narrow Sea
Stannis Baratheon	War of the Five Kings	Battle of the Blackwater	The King in the Narrow Sea
Theon Greyjoy	War of the Five Kings	Battle at Winterfell	The King of the Isles and the North
Tyrion Lannister	War of the Five Kings	Battle of the Blackwater	The King on the Iron Throne
Tywin Lannister	War of the Five Kings	Battle on the Green Fork	The King on the Iron Throne
Tywin Lannister	Robert's Rebellion	Sack of King's Landing	Rebels
Tywin Lannister	War of the Five Kings	Battle of the Blackwater	The King on the Iron Throne

Sample Output 2: BattleExperience

HouseWarHistory: Lists the names of all houses along with the wars they were a part of and the faction they allied with for each war

```
create view HouseWarHistory as
  select h.name as house, w.name as war, f.name as faction
  from houses h inner join alliedwith a on h.hid = a.hid
                inner join factions   f on a.fid = f.fid
                inner join wars       w on f.wid = w.wid
  order by house asc;
```

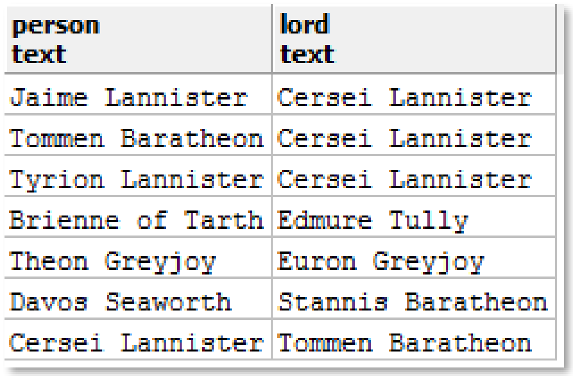


house text	war text	faction text
Baratheon of Dragonstone	Conflict Beyond the Wall	The Night's Watch
Baratheon of Dragonstone	War of the Five Kings	The King in the Narrow Sea
Baratheon of King's Landing	War of the Five Kings	The King on the Iron Throne
Baratheon of King's Landing	Robert's Rebellion	Rebels
Greyjoy	Robert's Rebellion	Rebels
Greyjoy	War of the Five Kings	The King of the Isles and the North
Lannister	Robert's Rebellion	Rebels
Lannister	War of the Five Kings	The King on the Iron Throne
Martell	Robert's Rebellion	Royalists
Stark	War of the Five Kings	The King in the North and the Trident
Stark	Robert's Rebellion	Rebels
Targaryen	Robert's Rebellion	Royalists
Targaryen	War of the Ninepenny Kings	Seven Kingdoms
Tully	War of the Five Kings	The King in the North and the Trident
Tully	Robert's Rebellion	Rebels

Sample Output 3: HouseWarHistory

Fealty: Lists each living person who swears fealty to a lord as well as that lord's name

```
create view Fealty as
  select p1.name as person, p2.name as lord
  from people p1 inner join allegiance a  on p1.pid   = a.pid
                 inner join houses     h  on a.hid    = h.hid
                 inner join people     p2 on h.lordid = p2.pid
  where p1.pid       != p2.pid
    and p1.deathyear is null
  order by lord asc;
```



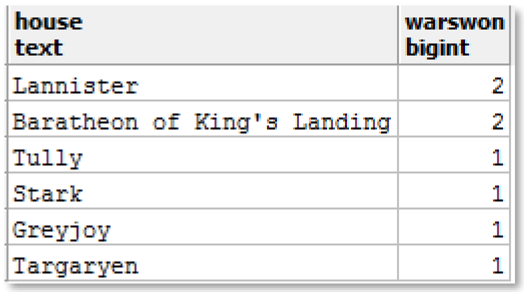
person text	lord text
Jaime Lannister	Cersei Lannister
Tommen Baratheon	Cersei Lannister
Tyrion Lannister	Cersei Lannister
Brienne of Tarth	Edmure Tully
Theon Greyjoy	Euron Greyjoy
Davos Seaworth	Stannis Baratheon
Cersei Lannister	Tommen Baratheon

Sample Output 4: Fealty

Reports and Interesting Queries

1. Query to return the number of wars won by each house

```
select h.name as house, count(h.hid) as warsWon
from houses h inner join alliedwith a on h.hid = a.hid
              inner join factions   f on a.fid = f.fid
where f.wonwar = true
group by h.name
order by warsWon desc;
```

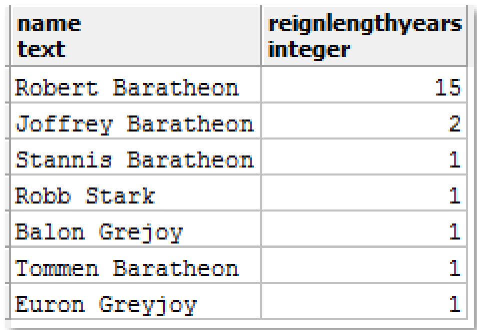


house text	warswon bigint
Lannister	2
Baratheon of King's Landing	2
Tully	1
Stark	1
Greyjoy	1
Targaryen	1

Sample Output 5

2. Query to return the length in years of each king’s reign

```
select p.name,
       case when coalesce((k.reignendyear - k.reignstartyear), (300 - reignstartyear)) = 0
then 1
            else coalesce((k.reignendyear - k.reignstartyear), (300 - reignstartyear))
       end as reignlengthyears
from people p inner join kings k on p.pid = k.pid
order by reignlengthyears desc;
```

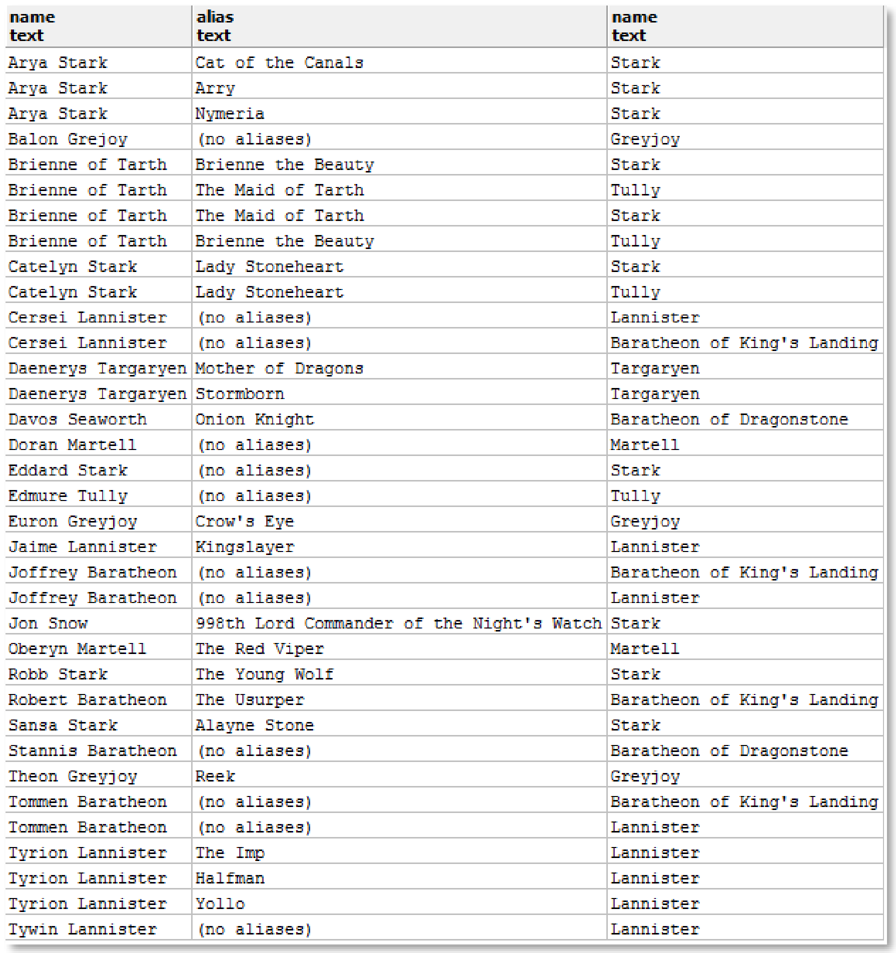


name text	reignlengthyears integer
Robert Baratheon	15
Joffrey Baratheon	2
Stannis Baratheon	1
Robb Stark	1
Balon Grejoy	1
Tommen Baratheon	1
Euron Greyjoy	1

Sample Output 6

3. Query to return the allegiance of each person as well as their assumed aliases

```
select p.name, coalesce(a1.name, '(no aliases)') as alias, h.name
from people p left outer join aliases    a1 on p.pid  = a1.pid
              inner join      allegiance a2 on p.pid  = a2.pid
              inner join      houses     h  on a2.hid = h.hid
order by p.name;
```



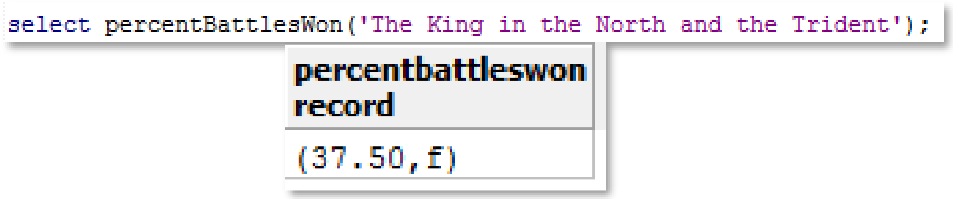
name	alias	name
text	text	text
Arya Stark	Cat of the Canals	Stark
Arya Stark	Arry	Stark
Arya Stark	Nymeria	Stark
Balon Grejoy	(no aliases)	Greyjoy
Brienne of Tarth	Brienne the Beauty	Stark
Brienne of Tarth	The Maid of Tarth	Tully
Brienne of Tarth	The Maid of Tarth	Stark
Brienne of Tarth	Brienne the Beauty	Tully
Catelyn Stark	Lady Stoneheart	Stark
Catelyn Stark	Lady Stoneheart	Tully
Cersei Lannister	(no aliases)	Lannister
Cersei Lannister	(no aliases)	Baratheon of King's Landing
Daenerys Targaryen	Mother of Dragons	Targaryen
Daenerys Targaryen	Stormborn	Targaryen
Davos Seaworth	Onion Knight	Baratheon of Dragonstone
Doran Martell	(no aliases)	Martell
Eddard Stark	(no aliases)	Stark
Edmure Tully	(no aliases)	Tully
Euron Greyjoy	Crow's Eye	Greyjoy
Jaime Lannister	Kingslayer	Lannister
Joffrey Baratheon	(no aliases)	Baratheon of King's Landing
Joffrey Baratheon	(no aliases)	Lannister
Jon Snow	998th Lord Commander of the Night's Watch	Stark
Oberyn Martell	The Red Viper	Martell
Robb Stark	The Young Wolf	Stark
Robert Baratheon	The Usurper	Baratheon of King's Landing
Sansa Stark	Alayne Stone	Stark
Stannis Baratheon	(no aliases)	Baratheon of Dragonstone
Theon Greyjoy	Reek	Greyjoy
Tommen Baratheon	(no aliases)	Baratheon of King's Landing
Tommen Baratheon	(no aliases)	Lannister
Tyrion Lannister	The Imp	Lannister
Tyrion Lannister	Halfman	Lannister
Tyrion Lannister	Yollo	Lannister
Tywin Lannister	(no aliases)	Lannister

Sample Output 7

Stored Procedures

percentBattlesWon: Takes a faction name as an argument and returns the percentage of battles that faction won and whether or not they won the war.

```
create or replace function percentBattlesWon(factionName text)
returns table(percent_won numeric, wonwar boolean) as $$
begin
  return query select trunc (
           (cast(
                 ( select count(c.fid) as battleswon
                   from combatants c inner join factions f on c.fid = f.fid
                                     inner join wars     w on w.wid = f.wid
                   where f.name       = factionName
                     and c.wonbattle = true
                 ) as decimal(5,2))
                     /
                     ( select count(c.fid) as battlesfought
                       from combatants c inner join factions f on c.fid = f.fid
                                         inner join wars     w on w.wid = f.wid
                       where f.name = factionName
                     ) * 100) , 2
              ) as percent_won, factions.wonwar
  from factions
  where name = factionName;
end; $$ language plpgsql;
```



```
select percentBattlesWon('The King in the North and the Trident');
percentbattleswon
record
(37.50,f)
```

Sample Output 8: percentBattlesWon

endWar: Returns a trigger that sets the endyear for a war to the current year if a faction is updated to have won the war and the war is not over (See Triggers, p.23)

```
create or replace function endWar() returns trigger as
$$
declare
-- Unfortunately, the Westerosi calendar is not supported by Postgres, necessitating the following
-- variable:
  currentYear integer := 300;
begin
  if new.wonwar = true
  and (select endyear
       from wars
       where wid = new.wid) is null
  then
      update Wars
      set endyear = currentYear
      where wid = new.wid;
  end if;
  return new;
end;
$$
language plpgsql;
```

endReign: Returns a trigger that sets the endreignyear of a king to their death year if they are updated to have died and their reign is not currently over (See Triggers, p.24)

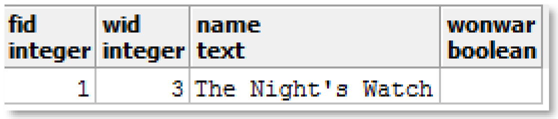
```
create or replace function endReign() returns trigger as
$$
begin
  if  new.deathyear is not null
  and (select reignendyear
       from kings
       where pid = new.pid) is null
  then
      update kings
      set reignendyear = new.deathyear
      where pid = new.pid;
  end if;
  return new;
end;
$$
language plpgsql;
```

Triggers

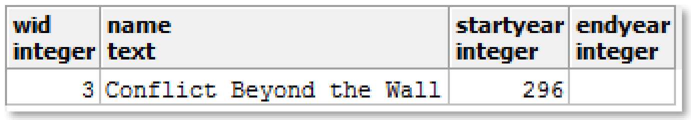
endWar: After updating the Factions table, executes the endWar() procedure (See Stored Procedures, p.21)

```
create trigger endWar

after update on Factions

for each row

execute procedure endWar();
```

Before Update

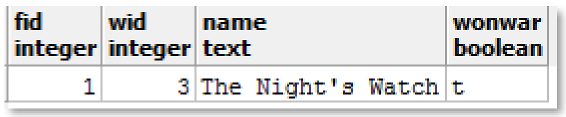


fid	wid	name	wonwar
integer	integer	text	boolean
1	3	The Night's Watch	

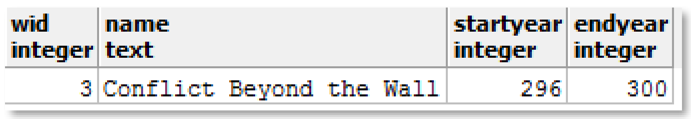


wid	name	startyear	endyear
integer	text	integer	integer
3	Conflict Beyond the Wall	296	

After Update



fid	wid	name	wonwar
integer	integer	text	boolean
1	3	The Night's Watch	t



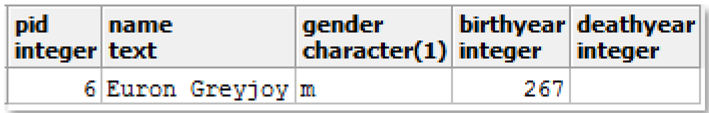
wid	name	startyear	endyear
integer	text	integer	integer
3	Conflict Beyond the Wall	296	300

Sample Output 9: endWar

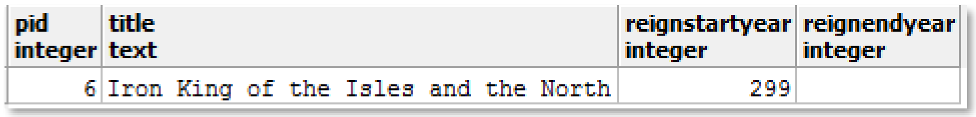
endReign: After updating the People table, executes the endReign() procedure (See Stored Procedures, p.22)

```
create trigger endReign

after update on People

for each row

execute procedure endReign();
```

Before Update

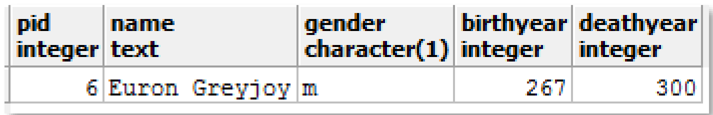


pid integer	name text	gender character(1)	birthyear integer	deathyear integer
6	Euron Greyjoy	m	267	

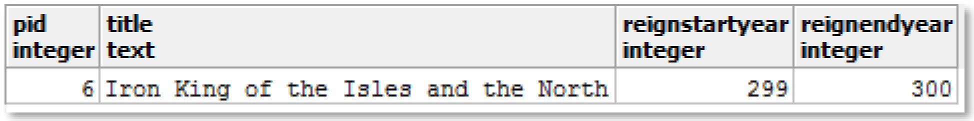


pid integer	title text	reignstartyear integer	reignendyear integer
6	Iron King of the Isles and the North	299	

After Update



pid integer	name text	gender character(1)	birthyear integer	deathyear integer
6	Euron Greyjoy	m	267	300



pid integer	title text	reignstartyear integer	reignendyear integer
6	Iron King of the Isles and the North	299	300

Sample Output 10: endReign

Security

Administrator Role: Represents the database administrator who has full access

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

Maester Role: Represents the maesters who work at the Citadel who need to access and change records for their studies and for archival purposes

```
create role maesters;
revoke all on all tables in schema public from maesters;
grant select on all tables in schema public to maesters;
grant insert on people, kings, nightswatchmen, knights,
                aliases, houses, allegiance, foughtin,
                battles, wars, combatants, factions,
                alliedwith
to maesters;
grant update on people, kings, nightswatchmen, knights,
                aliases, houses, allegiance, foughtin,
                battles, wars, combatants, factions,
                alliedwith
to maesters;
```

Visitor Role: Represents an outside visitor to the Citadel who needs to access data but is not allowed to change anything

```
create role visitors;
revoke all on all tables in schema public from visitors;
grant select on people, kings, nightswatchmen, knights,
                houses, castles, regions, foughtin,
                battles, wars, combatants, factions,
                alliedwith
to visitors;
```

Implementation Notes

- Postgres does not support the Westerosi calendar including years and dates. For this reason, years are stored as integers and the current year is manually entered when necessary.
- In all instances when definitive historical data is not yet available, the value is left as null. For instance, ongoing wars have a null endyear and living people have a null deathyear.
- All names of people, houses, wars, factions, battles, etc. do not have to be unique. For instance, a house could splinter into warring factions but maintain the same name in both separate houses. For this reason and similar others, names are not used as primary keys in any tables.
- For those not acquainted with Westerosi culture, a “maester” is a scholarly individual comparable to a teacher, doctor, or other learned person. They study at the Citadel and are often assigned to houses to serve as advisors and doctors.

Known Problems / Future Enhancements

- Due to the date problem described in Implementation Notes, the current year will have to be updated in the database each year.
- With the current implementation, it is difficult to tell exactly who is related to whom. One can see which people have familial allegiance to the same house, but even then it will not count people who have married into that house. Thus, a future version could improve on the current implementation by taking into account family trees and creating a hierarchy of sorts.
- There are many more tables that could be implemented in future versions of the database. For instance, there could be a “Groups” or “Organizations” table which has subtypes such as “Houses”, “Mercenaries”, or “Religions”. This would allow a greater level of detail showing different people’s allegiances to groups other than just Houses.