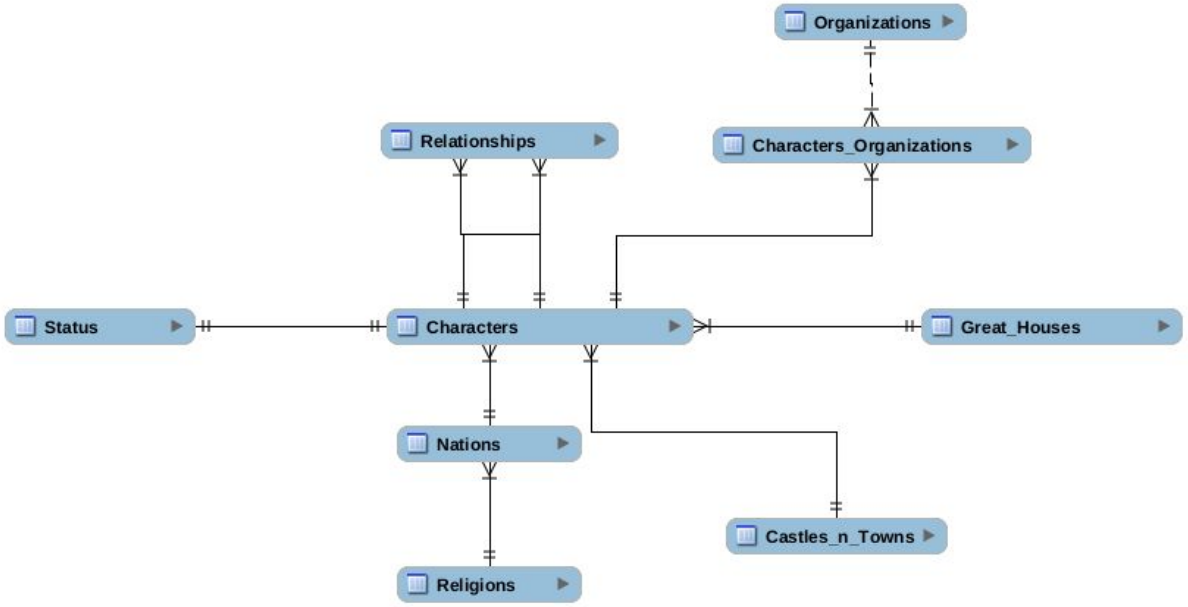
База данных “Игра престолов”

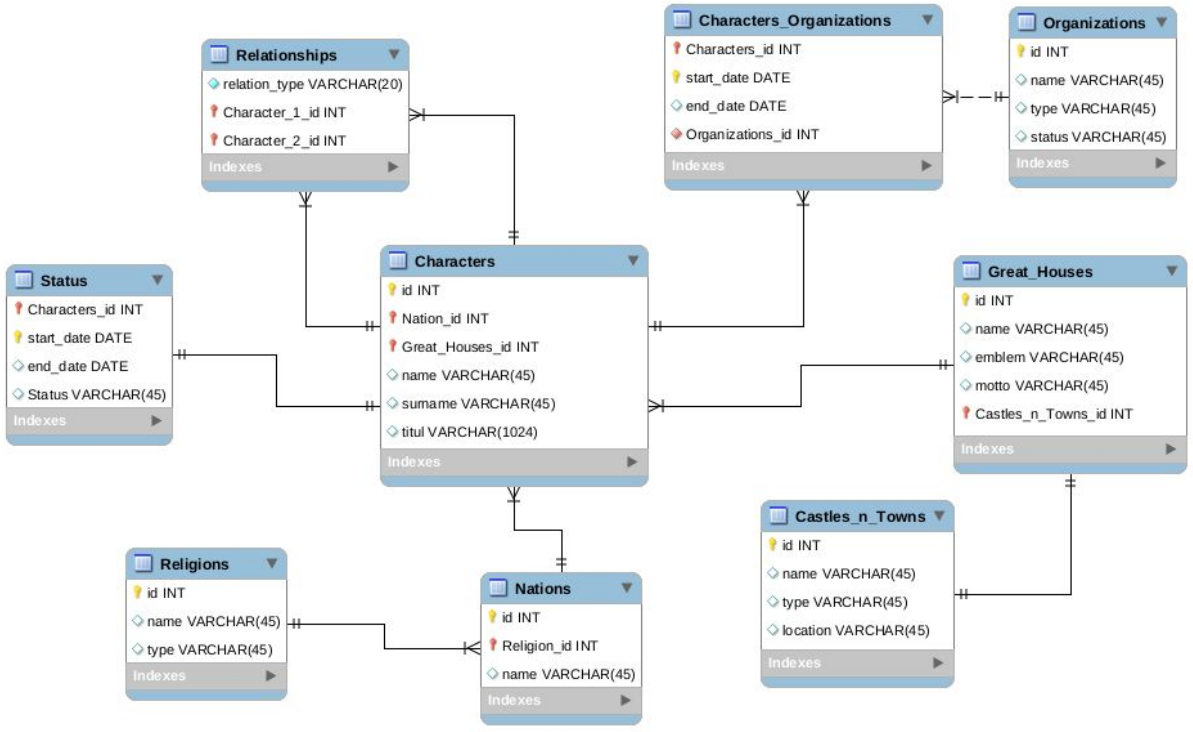
Кухтенков Алексей Б05-811

Весна 2019 г.

1. Концептуальная модель:



2. Логическая модель:



3. Создание физической модели:

- **Схема `GoT`:**

```
DROP SCHEMA IF EXISTS GoT CASCADE;
CREATE SCHEMA GoT;
```

- **Таблица `Religions`, в которой содержатся уникальный номер, название и тип религий:**

```
DROP TABLE IF EXISTS GoT.Religions CASCADE ;

CREATE TABLE GoT.Religions (
     id   SERIAL PRIMARY KEY,
     name VARCHAR(45) NOT NULL,
     type VARCHAR(45)
);
```

- **Таблица `Nations`, в которой содержатся уникальный номер, уникальный номер религии и название наций:**

```
DROP TABLE IF EXISTS GoT.Nations CASCADE ;

CREATE TABLE GoT.Nations (
     id        SERIAL PRIMARY KEY,
     Religion_id INT,
     name      VARCHAR(45)         NOT NULL,
     FOREIGN KEY (Religion_id) REFERENCES
GoT.Religions(id)
);
```

- **Таблица `Castles_n_Towns`, в которой содержатся уникальный номер, название, тип и местоположение городов и замков:**

```
DROP TABLE IF EXISTS GoT.Castles_n_Towns CASCADE ;

CREATE TABLE GoT.Castles_n_Towns (
     id        SERIAL    NOT NULL PRIMARY KEY,
     name      VARCHAR(45) NOT NULL,
     type      VARCHAR(45),
     location VARCHAR(45)
```

```
);
```

- **Таблица `Great_Houses`, в которой содержатся уникальный номер, уникальный номер родового поместья, название, эмблема и девиз великих домов:**

```
DROP TABLE IF EXISTS GoT.Great_Houses CASCADE ;

CREATE TABLE GoT.Great_Houses (
     id   SERIAL    NOT NULL PRIMARY KEY,
     castles_n_towns_id INT REFERENCES
GoT.Castles_n_Towns(id),
     name   VARCHAR(45) NOT NULL,
     emblem VARCHAR(45),
     motto  VARCHAR(100)
);
```

- **Таблица `Organizations`, в которой содержатся уникальный номер, название, тип и статус организаций:**

```
DROP TABLE IF EXISTS GoT.Organizations CASCADE ;

CREATE TABLE GoT.Organizations (
  id SERIAL NOT NULL PRIMARY KEY ,
  name VARCHAR(45) NOT NULL,
  type VARCHAR(45),
  status VARCHAR(45)
);
```

- **Таблица `Characters`, в которой содержатся уникальный номер, уникальный номер нации, уникальный номер великого дома, имя, фамилия и титул персонажа:**

```
DROP TABLE IF EXISTS GoT.Characters CASCADE ;

CREATE TABLE IF NOT EXISTS GoT.Characters (
  id SERIAL NOT NULL PRIMARY KEY,
  nation_id INT REFERENCES GoT.Nations(id),
  great_houses_id INT REFERENCES GoT.Great_Houses(id),
  name VARCHAR(45) NOT NULL,
  surname VARCHAR(45),
  titul VARCHAR(1024)
);
```

- **Таблица `Status`, в которой содержатся уникальный номер персонажа, даты начала и конца актуальности данной записи и статус персонажа:**

```
DROP TABLE IF EXISTS GoT.Status CASCADE ;

CREATE TABLE GoT.Status (
  Characters_id INT REFERENCES GoT.Characters(id),
  start_date DATE NOT NULL,
  end_date DATE,
  status VARCHAR(45),
  PRIMARY KEY (Characters_id, start_date)
);
```

- **Таблица `Relationships`, в которой содержатся тип отношения, уникальные номера персонажей 1 и 2:**

```
DROP TABLE IF EXISTS GoT.Relationships CASCADE ;

CREATE TABLE IF NOT EXISTS GoT.Relationships (
  relation_type VARCHAR(20),
  Character_1_id INT REFERENCES GoT.Characters(id),
  Character_2_id INT REFERENCES GoT.Characters(id),
  PRIMARY KEY (Character_1_id, Character_2_id)
);
```

- **Таблица `Characters_Organizations` - таблица связи между `Characters и Organizations(спасибо, кэп)`:**

```
DROP TABLE IF EXISTS GoT.Characters_Organizations;

CREATE TABLE GoT.Characters_Organizations (
     Characters_id INT REFERENCES GoT.Characters(id),
     start_date DATE,
     end_date DATE,
     organizations_id INT REFERENCES
GoT.Organizations(id),
     PRIMARY KEY (Characters_id, start_date)
);
```

4. INSERTS:

- **Religions**

```
INSERT INTO GoT.Religions VALUES (DEFAULT, 'Faith in seven',
'Monotheism');
INSERT INTO GoT.Religions VALUES (DEFAULT, 'God drowned',
'Monolatria');
INSERT INTO GoT.Religions VALUES (DEFAULT, 'Many faces',
'Monotheism');
INSERT INTO GoT.Religions VALUES (DEFAULT, 'Old gods',
'Polytheism');
INSERT INTO GoT.Religions VALUES (DEFAULT, 'Great stallion',
'Henotheism');
INSERT INTO GoT.Religions VALUES (DEFAULT, 'Valyrian',
'Monotheism');
```

- **Nations**

```
INSERT INTO GoT.Nations VALUES (DEFAULT, 5, 'Dothraki');
INSERT INTO GoT.Nations VALUES (DEFAULT, 2, 'Ironborn');
INSERT INTO GoT.Nations VALUES (DEFAULT, 4, 'Northerners');
INSERT INTO GoT.Nations VALUES (DEFAULT, 1, 'Dornishmen');
INSERT INTO GoT.Nations VALUES (DEFAULT, 1, 'Andals');
INSERT INTO GoT.Nations VALUES (DEFAULT, 6, 'Valyrians');
INSERT INTO GoT.Nations VALUES (DEFAULT, NULL, 'White Walkers');
INSERT INTO GoT.Nations VALUES (DEFAULT, 4, 'Swamp inhabitants');
```

- **Castles_n_Towns**

```
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'King_s Landing',
'Town', 'Royal lands');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Storm_s End',
'Castle', 'Stormlands');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Dragonstone',
'Castle', 'Dragonstone island');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Highgarden',
'Castle', 'Open space');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Winterfell',
'Castle', 'North');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Casterly Rock',
'Castle', 'Westerlands');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Blackhaven',
'Castle', 'Stormlands');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Riverrun',
'Castle', 'Riverlands');
INSERT INTO GoT.Castles_n_Towns VALUES(DEFAULT, 'Pyke', 'Castle',
'Iron islands');
```

- **Great_Houses**

```
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 5, 'Stark',
'Direwolf', 'Winter is coming');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 6, 'Lannister', 'Gold
lion', 'Lannisters always pay their debts');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 2, 'Baratheon', 'Black
deer', 'Us - rage');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 4, 'Tyrell', 'Gold
rose', 'Growing up - stronger');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 3, 'Targaryen',
'Three-headed red dragon', 'Flame and Blood');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 7, 'Dondarrion',
'Split purple lightning', NULL);
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 8, 'Tully', 'Jumping
silver trout', 'Family, duty, honor');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, NULL, 'Torn', 'Mace on
the scarlet field', NULL);
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, 9, 'Greyjoy', 'Gold
kraken', 'We do not sow');
INSERT INTO GoT.Great_Houses VALUES (DEFAULT, NULL, 'Tollet', 'Gray
and black wedges', 'At the darkest hour');
```

- **Organizations**

```
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Faceless Men',
'Religious mercenary killers', 'Active');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Unsullied',
'Mercenaries', 'Active');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Golden Company',
'Mercenaries', 'Active');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Kingsguard',
'Guardians', 'Active');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Night_s Watch',
'Military', 'Active');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Sparrows',
'Religious', 'Destroyed');
INSERT INTO GoT.Organizations VALUES (DEFAULT, 'Brotherhood Without
Banners', 'Outcasts', 'Active');
```

- **Перед вставкой данных в `Characters` создадим триггеры с соответствующими функциями:**

```
CREATE OR REPLACE FUNCTION characters_in_row() RETURNS TRIGGER AS $$
     BEGIN
     INSERT INTO GoT.Status VALUES (NEW.id, '2011-04-17',
'9999-12-31', 'Alive');
     RETURN NEW;
     END;
     $$ LANGUAGE plpgsql;

CREATE TRIGGER characters_in
     AFTER INSERT ON GoT.Characters
     FOR EACH ROW
     EXECUTE PROCEDURE characters_in_row();
```

Данный триггер автоматически добавляет статус `Alive` после добавления персонажа.

```
CREATE OR REPLACE FUNCTION characters_upd_row() RETURNS TRIGGER AS
$$
     BEGIN
     UPDATE GoT.Status SET Characters_id = new.id WHERE
Characters_id = old.id;
     RETURN NEW;
     END;
     $$ LANGUAGE plpgsql;

CREATE TRIGGER characters_upd
     AFTER UPDATE ON GoT.Characters
     FOR EACH ROW
     EXECUTE PROCEDURE characters_upd_row();
```

Данный триггер автоматически обновляет `id` персонажа в `Status` после изменения информации персонажа.

```
CREATE OR REPLACE FUNCTION characters_del_row() RETURNS TRIGGER AS
$$
     BEGIN
     DELETE FROM GoT.Status
     WHERE Characters_id = OLD.id;
     IF NOT FOUND THEN RETURN NULL; END IF;
     RETURN OLD;
     END;
     $$ LANGUAGE plpgsql;

CREATE TRIGGER characters_del
     BEFORE DELETE ON GoT.Characters
     FOR EACH ROW
     EXECUTE PROCEDURE characters_del_row();
```

Данный триггер автоматически удаляет персонажа в `Status` до удаления персонажа.

- **А также функция для упрощения умерщвления персонажей:**

```
CREATE OR REPLACE FUNCTION characters_death(d date, id int) RETURNS
BOOLEAN
     AS $$
     BEGIN
     IF id NOT IN (SELECT Characters_id FROM GoT.Status) THEN RETURN
FALSE; END IF;
     UPDATE GoT.Status SET end_date = d WHERE characters_id = id;
     INSERT INTO GoT.Status VALUES (id, d, '9999-12-31', 'Dead');
     RETURN TRUE;
     END;
$$ LANGUAGE plpgsql;
```

- **Characters…:**

```
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Jon', 'Snow',
'King of the North');
UPDATE GoT.Status SET start_date = '2016-04-24' WHERE characters_id
IN (SELECT id FROM GoT.Characters WHERE name = 'Jon');
INSERT INTO GoT.Status VALUES (1, '2011-04-17', '2015-06-14',
'Alive');
INSERT INTO GoT.Status VALUES (1, '2015-06-14', '2016-04-24',
'Dead');
INSERT INTO GoT.Characters VALUES (DEFAULT, 6, 5, 'Daenerys',
'Targaryen', 'Burenated from the house of Targaryen, from the blood
of ancient Valyria, the first of this name, the Queen of the Andals,
the Roynars and the First Men, the mistress of the seven kingdoms,
and the keeper of the regions, the burning mistress, the ruler of
Myerin, Khalisi of the Great Grass Sea, the Destroyer of the Bondage
and the Mother of Dragons');
INSERT INTO GoT.Characters VALUES (DEFAULT, 7, NULL, 'Night',
'King', 'BUUUUUUUU');
UPDATE GoT.Status SET start_date = '2014-04-13', end_date =
'2019-04-28', status = 'Alive?' WHERE characters_id IN (SELECT id
FROM GoT.Characters WHERE name = 'Night');
INSERT INTO GoT.Status VALUES (3, '2019-04-28', '9999-12-31',
'Destroyed');
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Arya', 'Stark',
'Princess');
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 2, 'Cersei',
'Lannister', 'Queen of the Andals and First People');
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 2, 'Joffrey',
'Baratheon', 'King of the Andals and First People');
SELECT characters_death('2014-04-13', 6) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Eddard', 'Stark',
'Lord Winterfell');
SELECT characters_death('2011-06-12', 7) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 2, 'Jaime',
'Lannister', 'Lord Commander of the Royal Guard');
INSERT INTO GoT.Characters VALUES (DEFAULT, 6, 5, 'Rhaegar',
'Targaryen', 'The Last Dragon');
UPDATE GoT.Status SET status = 'Dead' WHERE characters_id IN (SELECT
id FROM GoT.Characters WHERE name = 'Rhaegar');
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 6, 'Beric',
'Dondarrion', 'Lord of the Blackhaven');
```

```
UPDATE GoT.Status SET status = 'Dead', start_date = '2019-04-28'
WHERE characters_id IN (SELECT id FROM GoT.Characters WHERE name =
'Beric');
INSERT INTO GoT.Status VALUES (10, '2011-04-17', '2011-05-15',
'Alive');
INSERT INTO GoT.Status VALUES (10, '2011-05-15', '2011-05-22',
'Dead');
INSERT INTO GoT.Status VALUES (10, '2011-05-22', '2013-04-28',
'Alive');
INSERT INTO GoT.Status VALUES (10, '2013-04-28', '2013-05-05',
'Dead');
INSERT INTO GoT.Status VALUES (10, '2013-05-05', '2016-04-24',
'Alive');
INSERT INTO GoT.Characters VALUES (DEFAULT, NULL, NULL, 'Grey',
'Worm', 'Commander unsullied');
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Sansa', 'Stark',
'Lady Winterfell');
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Rickon', 'Stark',
'Prince');
SELECT characters_death('2016-06-19', 13) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Brandon',
'Stark', 'Three-eyed raven');
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 7, 'Catelyn',
'Stark', 'Prince');
SELECT characters_death('2013-06-02', 15) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, 1, 'Robb', 'Stark',
'Young wolf');
SELECT characters_death('2013-06-02', 16) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 8, 'Alliser', 'Torn',
'First scout');
SELECT characters_death('2016-05-05', 17) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 8, NULL, 'Bowen',
'Marsh', 'First steward');
SELECT characters_death('2016-05-05', 18) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 3, NULL, 'Olly', NULL,
'Steward');
SELECT characters_death('2016-05-05', 19) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, NULL, 'Othell',
'Yarwyck', 'First builder');
SELECT characters_death('2016-05-05', 20) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, NULL, 'Varys', NULL,
'Master of whispering');
INSERT INTO GoT.Characters VALUES (DEFAULT, 2, 9, 'Euron',
'Greyjoy', 'King of the Iron Isles');
```

```
INSERT INTO GoT.Characters VALUES (DEFAULT, NULL, NULL, 'Shaggydog',
NULL, 'Good boy');
SELECT characters_death('2016-05-05', 23) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 10, 'Eddison',
'Tollett', 'Lord Commander Night Watch');
SELECT characters_death('2019-04-28', 24) AS operation_result;
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 2, 'Tywin',
'Lannister', 'Guardian of the West');
SELECT characters_death('2014-06-15', 25);
INSERT INTO GoT.Characters VALUES (DEFAULT, 5, 3, 'Robert',
'Baratheon', 'The King of the Andals and the First Men');
SELECT characters_death('2014-06-15', 26);

-----------------------SPOILERS-----------------------------

SELECT characters_death('2019-05-13', 5) AS operation_result;
SELECT characters_death('2019-05-13', 8) AS operation_result;
SELECT characters_death('2019-05-13', 21) AS operation_result;
SELECT characters_death('2019-05-13', 22) AS operation_result;
```

- **Relationships:**

```
INSERT INTO GoT.Relationships VALUES ('Father_child', 9, 1);
INSERT INTO GoT.Relationships VALUES ('Father_child', 8, 6);
INSERT INTO GoT.Relationships VALUES ('Mother_child', 5, 6);
INSERT INTO GoT.Relationships VALUES ('Brother_Sister', 8, 5);
INSERT INTO GoT.Relationships VALUES ('Father_child', 7, 4);
INSERT INTO GoT.Relationships VALUES ('Father_child', 7, 12);
INSERT INTO GoT.Relationships VALUES ('Father_child', 7, 13);
INSERT INTO GoT.Relationships VALUES ('Father_child', 7, 14);
INSERT INTO GoT.Relationships VALUES ('Father_child', 7, 16);
INSERT INTO GoT.Relationships VALUES ('Brothers_Sisters', 14, 16);
INSERT INTO GoT.Relationships VALUES ('Husband_wife', 7, 15);
INSERT INTO GoT.Relationships VALUES ('Husband_wife', 26, 5);
```

- **Characters_Organizations:**

```
INSERT INTO GoT.Characters_Organizations VALUES (1, '2011-04-24',
'2015-06-14', 5);
INSERT INTO GoT.Characters_Organizations VALUES (4, '2015-04-12',
'2016-06-26', 1);
INSERT INTO GoT.Characters_Organizations VALUES (10, '2011-04-17',
'2019-04-28', 7);
INSERT INTO GoT.Characters_Organizations VALUES (11, '2014-05-19',
'9999-12-31', 2);
INSERT INTO GoT.Characters_Organizations VALUES (17, '2011-04-17',
'9999-12-31', 5);
INSERT INTO GoT.Characters_Organizations VALUES (18, '2011-04-17',
'9999-12-31', 5);
INSERT INTO GoT.Characters_Organizations VALUES (19, '2014-04-27',
'9999-12-31', 5);
INSERT INTO GoT.Characters_Organizations VALUES (20, '2011-04-17',
'9999-12-31', 5);
INSERT INTO GoT.Characters_Organizations VALUES (24, '2011-04-17',
'9999-12-31', 5);
```

5. SELECTS, VIEWS, FUNCTIONS, SELECT:

```
SET search_path = GoT, public;
```

- **Ищем всех детей Эддарда Старка:**

```
SELECT name, surname
FROM Characters c  INNER JOIN Relationships r
ON c.id = r.Character_2_id
WHERE r.relation_type = 'Father_child' AND
  r.Character_1_id in (SELECT id FROM Characters WHERE
name = 'Eddard' AND surname = 'Stark');
```

- **Ищем всех персонажей, которые воскресали за историю сериала:**

```
SELECT name, surname
FROM Characters c INNER JOIN Status s
ON c.id = s.Characters_id
GROUP BY id
HAVING count(status) > 2;
```

- **Ищем всех предателей Джона Сноу из ночного дозора:**

```
SELECT name, surname, titul
     FROM Characters c INNER JOIN Characters_Organizations
co
     ON c.id = co.Characters_id
     WHERE organizations_id IN (SELECT id FROM
Organizations WHERE name = 'Night_s Watch' AND NOW()
BETWEEN start_date AND end_date)
INTERSECT
SELECT name, surname, titul
     FROM Characters c INNER JOIN  Status s
     ON c.id = s.Characters_id
     WHERE start_date = '2016-05-05' AND status = 'Dead';
```

- **Ищем великие дома, которые соединены брачным союзом:**

```
SELECT h1.name, h2.name
FROM Great_Houses h1 JOIN Great_Houses h2 ON h1.id <>
h2.id
WHERE h1.id IN
     (SELECT great_houses_id FROM
     (SELECT c1.great_houses_id FROM Characters c1
          INNER JOIN Relationships r ON Character_1_id =
c1.id
          INNER JOIN characters c2 ON r.Character_2_id =
c2.id
          WHERE relation_type = 'Husband_wife') AS Q1)
     AND
     h2.id IN
     (SELECT great_houses_id FROM
     (SELECT c2.great_houses_id FROM Relationships r
          INNER JOIN characters c2 ON r.Character_2_id =
c2.id
          WHERE relation_type = 'Husband_wife' AND
Character_1_id = (SELECT id FROM Characters WHERE
great_houses_id = h1.id AND id IN (SELECT Character_1_id
FROM Relationships WHERE r.Character_2_id = Character_2_id
AND r.relation_type = 'Husband_wife'))) AS Q2);
```

- **Спойлерное представление:**

```
CREATE OR REPLACE VIEW spoilers (Name, Surname) AS
     SELECT name, surname
     FROM Characters c INNER JOIN Status s
     ON c.id = s.Characters_id
     WHERE start_date = '2019-05-13';
```

- **Представление всех персонажей с их жизненными статусами:**

```
CREATE VIEW Characters_Status (Character, Status) AS
     SELECT c.name, c.surname, s.status
     FROM GoT.Characters c INNER JOIN GoT.Status s ON c.id
= s.characters_id
     WHERE now() BETWEEN s.start_date AND s.end_date;
```

- **Представление с живыми персонажами дома Старков и Ланнистеров:**

```
CREATE VIEW Alive_Starks (Name_s, Surname_s) AS
     SELECT name, surname
     FROM Characters c INNER JOIN Status s
     ON c.id = s.Characters_id
     WHERE great_houses_id IN (SELECT id FROM Great_Houses
WHERE name = 'Stark') AND
          NOW() BETWEEN start_date AND end_date AND
          status = 'Alive';

CREATE VIEW Alive_Lannisters (Name_l, Surname_l) AS
     SELECT name, surname
     FROM Characters c INNER JOIN Status s
     ON c.id = s.Characters_id
     WHERE great_houses_id IN (SELECT id FROM Great_Houses
WHERE name = 'Lannister') AND
          NOW() BETWEEN start_date AND end_date AND
          status = 'Alive';
```

- **Функция, считающая количество персонажей, принадлежащих великому дому, подаваемому аргументом:**

```
CREATE OR REPLACE FUNCTION Count_Great_Housers(Great_House
varchar(45)) RETURNS BIGINT AS
     $$
     SELECT count(name) FROM Characters
     WHERE great_houses_id IN
          (SELECT id FROM Great_Houses
          WHERE name = Great_House);
     $$ LANGUAGE SQL;
```

- **Затерявшийся `select`, который выводит процентное соотношение живых персонажей дома Старков и Ланнистеров:**

```
SELECT CAST(cnt1 AS FLOAT) / cast(Count_Great_Housers('Lannister') AS
FLOAT) * 100 AS Lannister_percent_alive,
       CAST(cnt2 AS FLOAT) / cast(Count_Great_Housers('Stark') AS FLOAT) *
100 AS Stark_percent_alive
 FROM (SELECT count(Name_l) AS cnt1
       FROM Alive_Lannisters) as Q1,
      (SELECT count(Name_l) AS cnt2
       FROM Alive_Starks) as Q2;
```

6. ROLES:

- **`VIP` роль, которая может делать все что угодно с нашими табличками, включая разрешение доступа к ним:**

```
CREATE ROLE vip WITH LOGIN PASSWORD '1337';
GRANT ALL PRIVILEGES ON TABLE GoT.Characters TO vip WITH
GRANT OPTION;
```

- **Так себе роль, которая не может ничего, кроме обзора таблички статуса персонажей:**

```
CREATE ROLE Nina_skazala WITH LOGIN PASSWORD 'blblblbl';
GRANT SELECT ON TABLE GoT.Status TO Nina_skazala;
```