Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет программной инженерии и компьютерной техники

Дисциплина: Базы данных

Отчёт по лабораторной работе № 4

Выполнил: Во Минь Тхиен Лонг

Нгуен Зюи Кхань

Студент группы: P33201

Преподаватель: Николаев В. В.

Санкт-Петербург

2021 г.

I. Текст задания

Для выполнения лабораторной работы №4 необходимо:

- Реализовать разработанную в рамках лабораторной работы №3 даталогическую модель в реляционной СУБД PostgreSQL.
- Заполнить созданные таблицы данными.
- Обеспечить целостность данных при помощи средств языка DDL.
- В рамках лабораторной работы должны быть разработаны скрипты для создания/удаления требуемых объектов базы данных, заполнения/удаления содержимого созданных таблиц.

Отчёт по лабораторной работе должен содержать:

- титульный лист;
- текст задания;
- описание предметной области;
- DDL-скрипты, часть DML-скриптов;
- выводы по работе.

II. РеализациядаталогическоймоделинаSQL

Типы перечислений

1. **Gender**

```
CREATE TYPE gender AS ENUM ('MALE', 'FEMALE',
'OTHER');
```

2. **Food_type**

```
CREATE TYPE food_type AS ENUM ('CARNIVORES',
'HERBIVORES', 'OMNIVORES');
```

3. **Age**

```
CREATE TYPE age AS ENUM ('CRETACEOUS', 'JURASSIC',
'TRIASSIC');
```

Стержневые сущности

1. Account

```
CREATE TABLE IF NOT EXISTS account (
    id SERIAL PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE CHECK
(LENGTH(username) >= 6 AND username ~*
'^(?![_.])(?!.*[_.]{2})[a-zA-Z0-9._]+(?<![_.])$'),
    password VARCHAR(50) NOT NULL CHECK
(LENGTH(password) >= 6),
    birthday DATE CHECK (birthday <= now()),
    email VARCHAR(50) UNIQUE CHECK (email ~* '^[A-
Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'),
    gender GENDER NOT NULL
);
```

```
 Column  |         Type          |                     Modifiers
----------+-----------------------+-----------------------------------------------------
 id       | integer               | not null default nextval('account_id_seq'::regclass)
 username | character varying(50) | not null
 password | character varying(50) | not null
 birthday | date                  |
 email    | character varying(50) |
 gender   | gender                | not null
Indexes:
    "account_pkey" PRIMARY KEY, btree (id)
    "account_email_key" UNIQUE CONSTRAINT, btree (email)
    "account_username_key" UNIQUE CONSTRAINT, btree (username)
Check constraints:
    "account_birthday_check" CHECK (birthday <= now())
    "account_email_check" CHECK (email::text ~* '^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+[.][A-Za-z]+$'::text)
    "account_password_check" CHECK (length(password::text) >= 6)
    "account_username_check" CHECK (length(username::text) >= 6 AND username::text ~* '^(?![_.])(?!.*[_.]{2})[a-zA-Z0-9._]+(?<![_.])$'::text)
Referenced by:
    TABLE "account_food" CONSTRAINT "account_food_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
    TABLE "account_material" CONSTRAINT "account_material_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
    TABLE "inventory" CONSTRAINT "inventory_acd_fkey" FOREIGN KEY (acd) REFERENCES account(id)
    TABLE "island" CONSTRAINT "island_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
```

2. Material

```
CREATE TABLE IF NOT EXISTS material (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE
);
```

```
 Column |         Type          |                     Modifiers
--------+-----------------------+-----------------------------------------------------
 id     | integer               | not null default nextval('material_id_seq'::regclass)
 name   | character varying(50) | not null
Indexes:
    "material_pkey" PRIMARY KEY, btree (id)
    "material_name_key" UNIQUE CONSTRAINT, btree (name)
Referenced by:
    TABLE "account_material" CONSTRAINT "account_material_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
    TABLE "build_requirement" CONSTRAINT "build_requirement_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
    TABLE "resource_material" CONSTRAINT "resource_material_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
    TABLE "upgrade_requirement" CONSTRAINT "upgrade_requirement_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
```

3. Island (1000)

```
CREATE TABLE IF NOT EXISTS island (
    id SERIAL PRIMARY KEY,
    account_id INTEGER REFERENCES account NOT NULL,
    type AGE DEFAULT 'JURASSIC',
    UNIQUE (account_id, type)
);
```

```
  Column   |  Type   |                    Modifiers
-----------+---------+-------------------------------------------------
 id        | integer | not null default nextval('island_id_seq'::regclass)
 account_id | integer | not null
 type      | age     | default 'JURASSIC'::age
Indexes:
    "island_pkey" PRIMARY KEY, btree (id)
    "island_account_id_type_key" UNIQUE CONSTRAINT, btree (account_id, type)
Foreign-key constraints:
    "island_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
Referenced by:
    TABLE "habitat" CONSTRAINT "habitat_island_id_fkey" FOREIGN KEY (island_id) REFERENCES island(id)
    TABLE "island_resource" CONSTRAINT "island_resource_island_id_fkey" FOREIGN KEY (island_id) REFERENCES island(id)
```

4. Environment

```
CREATE TABLE IF NOT EXISTS environment (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE
);
```

```
 Column |         Type          |                      Modifiers
--------+-----------------------+-----------------------------------------------------
 id     | integer               | not null default nextval('environment_id_seq'::regclass)
 name   | character varying(50) | not null
Indexes:
    "environment_pkey" PRIMARY KEY, btree (id)
    "environment_name_key" UNIQUE CONSTRAINT, btree (name)
Referenced by:
    TABLE "habitat" CONSTRAINT "habitat_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
    TABLE "item" CONSTRAINT "item_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
    TABLE "species_environment" CONSTRAINT "species_environment_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
```

5. Habitat_level

```
CREATE TABLE IF NOT EXISTS habitat_level (
    level SERIAL PRIMARY KEY,
    capacity INTEGER DEFAULT 5 CHECK (capacity > 0)
);
```

```
  Column  |  Type   |                          Modifiers
----------+---------+-------------------------------------------------------------
 level    | integer | not null default nextval('habitat_level_level_seq'::regclass)
 capacity | integer | default 5
Indexes:
    "habitat_level_pkey" PRIMARY KEY, btree (level)
Check constraints:
    "habitat_level_capacity_check" CHECK (capacity > 0)
Referenced by:
    TABLE "habitat" CONSTRAINT "habitat_level_fkey" FOREIGN KEY (level) REFERENCES habitat_level(level)
    TABLE "upgrade_requirement" CONSTRAINT "upgrade_requirement_level_fkey" FOREIGN KEY (level) REFERENCES habitat_level(level)
```

6. Habitat

```
CREATE TABLE IF NOT EXISTS habitat (
    id SERIAL PRIMARY KEY,
    level INTEGER REFERENCES habitat_level DEFAULT 1,
    island_id INTEGER REFERENCES island NOT NULL,
    environment_id INTEGER REFERENCES environment
NOT NULL,
    UNIQUE (island_id, environment_id)
);
```

```
    Column     |  Type   |                    Modifiers
---------------+---------+-------------------------------------------------
 id            | integer | not null default nextval('habitat_id_seq'::regclass)
 level         | integer | default 1
 island_id     | integer | not null
 environment_id | integer | not null
Indexes:
    "habitat_pkey" PRIMARY KEY, btree (id)
    "habitat_island_id_environment_id_key" UNIQUE CONSTRAINT, btree (island_id, environment_id)
Foreign-key constraints:
    "habitat_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
    "habitat_island_id_fkey" FOREIGN KEY (island_id) REFERENCES island(id)
    "habitat_level_fkey" FOREIGN KEY (level) REFERENCES habitat_level(level)
Referenced by:
    TABLE "dinosaur_habitat" CONSTRAINT "dinosaur_habitat_habitat_id_fkey" FOREIGN KEY (habitat_id) REFERENCES habitat(id)
    TABLE "dinosaur" CONSTRAINT "dinosaur_habitat_id_fkey" FOREIGN KEY (habitat_id) REFERENCES habitat(id)
```

7. Species

```
CREATE TABLE IF NOT EXISTS species (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE,
    earning INTEGER NOT NULL CHECK (earning > 0),
    max_energy INTEGER NOT NULL CHECK (max_energy > 0),
    price INTEGER NOT NULL DEFAULT 100 CHECK (price > 0),
    type FOOD_TYPE DEFAULT 'CARNIVORES',
    age AGE DEFAULT 'JURASSIC'
);
```

```
   Column   |         Type          |                     Modifiers
------------+-----------------------+---------------------------------------------------
 id         | integer               | not null default nextval('species_id_seq'::regclass)
 name       | character varying(50) | not null
 earning    | integer               | not null
 max_energy | integer               | not null
 price      | integer               | not null default 100
 type       | food_type             | default 'CARNIVORES'::food_type
 age        | age                   | default 'JURASSIC'::age
Indexes:
    "species_pkey" PRIMARY KEY, btree (id)
    "species_name_key" UNIQUE CONSTRAINT, btree (name)
Check constraints:
    "species_earning_check" CHECK (earning > 0)
    "species_max_energy_check" CHECK (max_energy > 0)
    "species_price_check" CHECK (price > 0)
Referenced by:
    TABLE "dinosaur" CONSTRAINT "dinosaur_species_id_fkey" FOREIGN KEY (species_id) REFERENCES species(id)
    TABLE "species_environment" CONSTRAINT "species_environment_species_id_fkey" FOREIGN KEY (species_id) REFERENCES species(id)
```

8. **Dinosaur (**trigger при добавлении начального **earning** и **max_energy** зависит от **species)**

```
CREATE TABLE IF NOT EXISTS dinosaur (
    id SERIAL PRIMARY KEY,
    energy INTEGER DEFAULT 0 CHECK (energy >= 0),
    gender GENDER NOT NULL,
    species_id INTEGER REFERENCES species NOT NULL,
    earning INTEGER CHECK (earning > 0),
    max_energy INTEGER CHECK (max_energy > 0)
);
```

```
   Column   |  Type   |                      Modifiers
------------+---------+-----------------------------------------------------
 id         | integer | not null default nextval('dinosaur_id_seq'::regclass)
 habitat_id | integer | not null
 energy     | integer | default 0
 gender     | gender  | not null
 species_id | integer | not null
 earning    | integer |
 max_energy | integer |
Indexes:
    "dinosaur_pkey" PRIMARY KEY, btree (id)
Check constraints:
    "dinosaur_earning_check" CHECK (earning > 0)
    "dinosaur_energy_check" CHECK (energy >= 0)
    "dinosaur_max_energy_check" CHECK (max_energy > 0)
Foreign-key constraints:
    "dinosaur_habitat_id_fkey" FOREIGN KEY (habitat_id) REFERENCES habitat(id)
    "dinosaur_species_id_fkey" FOREIGN KEY (species_id) REFERENCES species(id)
```

9. **Food**

```
CREATE TABLE IF NOT EXISTS food (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE,
    price INTEGER CHECK (price > 0),
    energy_provide INTEGER CHECK (energy_provide > 0),
    type FOOD_TYPE DEFAULT 'CARNIVORES'
);
```

```
     Column     |         Type          |                    Modifiers
----------------+-----------------------+-------------------------------------------------
 id             | integer               | not null default nextval('food_id_seq'::regclass)
 name           | character varying(50) | not null
 price          | integer               |
 energy_provide | integer               |
 type           | food_type             | default 'CARNIVORES'::food_type
Indexes:
    "food_pkey" PRIMARY KEY, btree (id)
    "food_name_key" UNIQUE CONSTRAINT, btree (name)
Check constraints:
    "food_energy_provide_check" CHECK (energy_provide > 0)
    "food_price_check" CHECK (price > 0)
Referenced by:
    TABLE "account_food" CONSTRAINT "account_food_food_id_fkey" FOREIGN KEY (food_id) REFERENCES food(id)
```

10. Resource

```
CREATE TABLE IF NOT EXISTS resource (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE,
    age AGE DEFAULT 'JURASSIC'
);
```

```
 Column |         Type          |                     Modifiers
--------+-----------------------+-----------------------------------------------------
 id     | integer               | not null default nextval('resource_id_seq'::regclass)
 name   | character varying(50) | not null
 age    | age                   | default 'JURASSIC'::age
Indexes:
    "resource_pkey" PRIMARY KEY, btree (id)
    "resource_name_key" UNIQUE CONSTRAINT, btree (name)
Referenced by:
    TABLE "island_resource" CONSTRAINT "island_resource_resource_id_fkey" FOREIGN KEY (resource_id) REFERENCES resource(id)
    TABLE "resource_material" CONSTRAINT "resource_material_resource_id_fkey" FOREIGN KEY (resource_id) REFERENCES resource(id)
```

11. Item

```
CREATE TABLE IF NOT EXISTS item (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE,
    environment_id INTEGER REFERENCES environment
);
```

```
     Column     |         Type          |                     Modifiers
----------------+-----------------------+-----------------------------------------------------
 id             | integer               | not null default nextval('item_id_seq'::regclass)
 name           | character varying(50) | not null
 environment_id | integer               |
Indexes:
    "item_pkey" PRIMARY KEY, btree (id)
    "item_name_key" UNIQUE CONSTRAINT, btree (name)
Foreign-key constraints:
    "item_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
Referenced by:
    TABLE "inventory" CONSTRAINT "inventory_item_id_fkey" FOREIGN KEY (item_id) REFERENCES item(id)
    TABLE "item_ability" CONSTRAINT "item_ability_item_id_fkey" FOREIGN KEY (item_id) REFERENCES item(id)
```

12. Building

```
CREATE TABLE IF NOT EXISTS building (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL UNIQUE,
    description VARCHAR(200) NOT NULL
);
```

```
   Column    |         Type          |                     Modifiers
-------------+-----------------------+-----------------------------------------------------
 id          | integer               | not null default nextval('building_id_seq'::regclass)
 name        | character varying(50) | not null
 description | character varying(200) | not null
Indexes:
    "building_pkey" PRIMARY KEY, btree (id)
    "building_name_key" UNIQUE CONSTRAINT, btree (name)
Referenced by:
    TABLE "build_requirement" CONSTRAINT "build_requirement_building_id_fkey" FOREIGN KEY (building_id) REFERENCES building(id)
```

13. **Ability**

```
CREATE TABLE IF NOT EXISTS ability (
    id SERIAL PRIMARY KEY,
    earning_increase INTEGER NOT NULL CHECK
(earning_increase > 0),
    max_energy_increase INTEGER NOT NULL CHECK
(max_energy_increase > 0)
);
```

```
      Column        |  Type   |                     Modifiers
--------------------+---------+---------------------------------------------------
 id                 | integer | not null default nextval('ability_id_seq'::regclass)
 earning_increase   | integer | not null
 max_energy_increase | integer | not null
Indexes:
    "ability_pkey" PRIMARY KEY, btree (id)
Check constraints:
    "ability_earning_increase_check" CHECK (earning_increase > 0)
    "ability_max_energy_increase_check" CHECK (max_energy_increase > 0)
Referenced by:
    TABLE "item_ability" CONSTRAINT "item_ability_ability_id_fkey" FOREIGN KEY (ability_id) REFERENCES ability(id)
```

Ассоциативные сущности

1. **Account_food (**Когда **account** добавляется в базу данных, trigger добавит **food** по умолчанию для этого пользователя**)**

```
CREATE TABLE IF NOT EXISTS account_food (
    account_id INTEGER REFERENCES account NOT NULL,
    food_id INTEGER REFERENCES food NOT NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (account_id, food_id)
);
```

```
   Column   |  Type   | Modifiers
------------+---------+-----------
 account_id | integer | not null
 food_id    | integer | not null
 amount     | integer | default 0
Indexes:
    "account_food_account_id_food_id_key" UNIQUE CONSTRAINT, btree (account_id, food_id)
Check constraints:
    "account_food_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "account_food_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
    "account_food_food_id_fkey" FOREIGN KEY (food_id) REFERENCES food(id)
```

2. **Account_material (**Когда **account** добавляется в базу данных, trigger добавит **material** по умолчанию для этого пользователя**)**

```
CREATE TABLE IF NOT EXISTS account_material (
    account_id INTEGER REFERENCES account NOT NULL,
    material_id INTEGER REFERENCES material NOT
NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (account_id, material_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 account_id  | integer | not null
 material_id | integer | not null
 amount      | integer | default 0
Indexes:
    "account_material_account_id_material_id_key" UNIQUE CONSTRAINT, btree (account_id, material_id)
Check constraints:
    "account_material_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "account_material_account_id_fkey" FOREIGN KEY (account_id) REFERENCES account(id)
    "account_material_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
```

3. **Island_resource (**trigger **resource** имеет тот же **age**, что и **island)**

```
CREATE TABLE IF NOT EXISTS island_resource (
    island_id INTEGER REFERENCES island NOT NULL,
    resource_id INTEGER REFERENCES resource NOT
NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (island_id, resource_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 island_id   | integer | not null
 resource_id | integer | not null
 amount      | integer | default 0
Indexes:
    "island_resource_island_id_resource_id_key" UNIQUE CONSTRAINT, btree (island_id, resource_id)
Check constraints:
    "island_resource_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "island_resource_island_id_fkey" FOREIGN KEY (island_id) REFERENCES island(id)
    "island_resource_resource_id_fkey" FOREIGN KEY (resource_id) REFERENCES resource(id)
```

4. **Dinosaur_habitat (**trigger **dinosaur** имеет тот же **environment**, что и **habitat**, и количество **dinosaur** не превышает вместимости **habitat)**

```
CREATE TABLE IF NOT EXISTS dinosaur_habitat (
    dinosaur_id INTEGER REFERENCES dinosaur NOT NULL UNIQUE,
    habitat_id INTEGER REFERENCES habitat NOT NULL,
    UNIQUE (dinosaur_id, habitat_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 dinosaur_id | integer | not null
 habitat_id  | integer | not null
Indexes:
    "dinosaur_habitat_dinosaur_id_habitat_id_key" UNIQUE CONSTRAINT, btree (dinosaur_id, habitat_id)
    "dinosaur_habitat_dinosaur_id_key" UNIQUE CONSTRAINT, btree (dinosaur_id)
Foreign-key constraints:
    "dinosaur_habitat_dinosaur_id_fkey" FOREIGN KEY (dinosaur_id) REFERENCES dinosaur(id)
    "dinosaur_habitat_habitat_id_fkey" FOREIGN KEY (habitat_id) REFERENCES habitat(id)
```

5. **Resource_material**

```
CREATE TABLE IF NOT EXISTS resource_material (
    resource_id INTEGER REFERENCES resource NOT NULL,
    material_id INTEGER REFERENCES material NOT NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (resource_id, material_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 resource_id | integer | not null
 material_id | integer | not null
 amount      | integer | default 0
Indexes:
    "resource_material_resource_id_material_id_key" UNIQUE CONSTRAINT, btree (resource_id, material_id)
Check constraints:
    "resource_material_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "resource_material_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
    "resource_material_resource_id_fkey" FOREIGN KEY (resource_id) REFERENCES resource(id)
```

6. **Build_requirement**

```
CREATE TABLE IF NOT EXISTS build_requirement (
    building_id INTEGER REFERENCES building NOT NULL,
    material_id INTEGER REFERENCES material NOT NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (building_id, material_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 building_id | integer | not null
 material_id | integer | not null
 amount      | integer | default 0
Indexes:
    "build_requirement_building_id_material_id_key" UNIQUE CONSTRAINT, btree (building_id, material_id)
Check constraints:
    "build_requirement_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "build_requirement_building_id_fkey" FOREIGN KEY (building_id) REFERENCES building(id)
    "build_requirement_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
```

7. **Upgrade_requirement**

```
CREATE TABLE IF NOT EXISTS upgrade_requirement (
    level INTEGER REFERENCES habitat_level NOT NULL,
    material_id INTEGER REFERENCES material NOT NULL,
    amount INTEGER DEFAULT 0 CHECK (amount > 0),
    UNIQUE (level, material_id)
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 level       | integer | not null
 material_id | integer | not null
 amount      | integer | default 0
Indexes:
    "upgrade_requirement_level_material_id_key" UNIQUE CONSTRAINT, btree (level, material_id)
Check constraints:
    "upgrade_requirement_amount_check" CHECK (amount > 0)
Foreign-key constraints:
    "upgrade_requirement_level_fkey" FOREIGN KEY (level) REFERENCES habitat_level(level)
    "upgrade_requirement_material_id_fkey" FOREIGN KEY (material_id) REFERENCES material(id)
```

8. Species_environment

```
CREATE TABLE IF NOT EXISTS species_environment (
    species_id INTEGER REFERENCES species NOT NULL,
    environment_id INTEGER REFERENCES environment
NOT NULL
);
```

```
    Column     |  Type   | Modifiers
---------------+---------+-----------
 species_id    | integer | not null
 environment_id | integer | not null
Foreign-key constraints:
    "species_environment_environment_id_fkey" FOREIGN KEY (environment_id) REFERENCES environment(id)
    "species_environment_species_id_fkey" FOREIGN KEY (species_id) REFERENCES species(id)
```

9. Item_ability

```
CREATE TABLE IF NOT EXISTS item_ability (
    item_id INTEGER REFERENCES item NOT NULL,
    ability_id INTEGER REFERENCES ability NOT NULL
);
```

```
   Column   |  Type   | Modifiers
------------+---------+-----------
 item_id    | integer | not null
 ability_id | integer | not null
Foreign-key constraints:
    "item_ability_ability_id_fkey" FOREIGN KEY (ability_id) REFERENCES ability(id)
    "item_ability_item_id_fkey" FOREIGN KEY (item_id) REFERENCES item(id)
```

10.Inventory

```
CREATE TABLE IF NOT EXISTS inventory (
    account_id INTEGER REFERENCES account NOT NULL,
    item_id INTEGER REFERENCES item NOT NULL,
    dinosaur_id INTEGER REFERENCES dinosaur
);
```

```
   Column    |  Type   | Modifiers
-------------+---------+-----------
 acd         | integer | not null
 item_id     | integer | not null
 dinosaur_id | integer |
Foreign-key constraints:
    "inventory_acd_fkey" FOREIGN KEY (acd) REFERENCES account(id)
    "inventory_dinosaur_id_fkey" FOREIGN KEY (dinosaur_id) REFERENCES dinosaur(id)
    "inventory_item_id_fkey" FOREIGN KEY (item_id) REFERENCES item(id)
```

Список сущностей

```
                List of relations
 Schema   |         Name          | Type  |  Owner
----------+-----------------------+-------+----------
 s273973  | ability               | table | s273973
 s273973  | account               | table | s273973
 s273973  | account_food          | table | s273973
 s273973  | account_material      | table | s273973
 s273973  | build_requirement     | table | s273973
 s273973  | building              | table | s273973
 s273973  | dinosaur              | table | s273973
 s273973  | dinosaur_habitat      | table | s273973
 s273973  | environment           | table | s273973
 s273973  | food                  | table | s273973
 s273973  | habitat               | table | s273973
 s273973  | habitat_level         | table | s273973
 s273973  | inventory             | table | s273973
 s273973  | island                | table | s273973
 s273973  | island_resource       | table | s273973
 s273973  | item                  | table | s273973
 s273973  | item_ability          | table | s273973
 s273973  | material              | table | s273973
 s273973  | resource              | table | s273973
 s273973  | resource_material     | table | s273973
 s273973  | species               | table | s273973
 s273973  | species_environment   | table | s273973
 s273973  | upgrade_requirement   | table | s273973
(23 rows)
```

III. Заполнение данными

1. **Account (500)**

```
INSERT INTO account (username, password, birthday,
email, gender) VALUES ('scotty_192', 'scotty194',
'1941-4-19', 'scotty@mail.ru', 'MALE');
INSERT INTO account (username, password, birthday,
email, gender) VALUES ('darren_24_3', 'D4rreN*~#',
'1982-3-24', 'darren@gmail.com', 'FEMALE');
INSERT INTO account (username, password, birthday,
email, gender) VALUES ('donald', '11231232312',
'1998-3-3', 'donald@mail.ru', 'MALE');
…
```

2. **Material**

```
INSERT INTO material (name) VALUES ('Gold');
INSERT INTO material (name) VALUES ('Sivel');
INSERT INTO material (name) VALUES ('Metal');
INSERT INTO material (name) VALUES ('Wood');
INSERT INTO material (name) VALUES ('Cement');
```

3. **Island (1000)**

```
INSERT INTO island (account_id) VALUES (4);
INSERT INTO island (account_id, type) VALUES (3,
'CRETACEOUS');
INSERT INTO island (account_id, type) VALUES (1,
'TRIASSIC');
…
```

4. **Environment**

```
INSERT INTO environment (name) VALUES ('aerial');
INSERT INTO environment (name) VALUES ('mountain');
INSERT INTO environment (name) VALUES ('plain');
INSERT INTO environment (name) VALUES ('forest');
INSERT INTO environment (name) VALUES ('valley');
INSERT INTO environment (name) VALUES ('desert');
INSERT INTO environment (name) VALUES ('swamp');
INSERT INTO environment (name) VALUES ('river');
INSERT INTO environment (name) VALUES ('lake');
INSERT INTO environment (name) VALUES ('sea');
```

5. **Habitat_level**

```
INSERT INTO habitat_level (level) VALUES (1);
INSERT INTO habitat_level (level, capacity) VALUES
(2, 7);
INSERT INTO habitat_level (level, capacity) VALUES
(3, 10);
INSERT INTO habitat_level (level, capacity) VALUES
(4, 15);
INSERT INTO habitat_level (level, capacity) VALUES
(5, 20);
INSERT INTO habitat_level (level, capacity) VALUES
(6, 30);
```

6. **Habitat (3000)**

```
INSERT INTO habitat (level, island_id,
environment_id) VALUES (1, 1, 3);
INSERT INTO habitat (level, island_id,
environment_id) VALUES (1, 2, 3);
INSERT INTO habitat (level, island_id,
environment_id) VALUES (1, 3, 3);
…
```

7. **Species (500)**

```
INSERT INTO species (name, earning, max_energy,
price) VALUES ('Allosaurus', 856, 1297, 489);
INSERT INTO species (name, earning, max_energy,
price, type) VALUES ('Apatosaurus', 873, 1555, 763,
'HERBIVORES');
INSERT INTO species (name, earning, max_energy,
price, type) VALUES ('Archaeopteryx', 661, 1532, 5,
'OMNIVORES');
…
```

8. **Dinosaur (6000)**

```
INSERT INTO dinosaur (habitat_id, energy, gender,
species_id) VALUES (41, 455, 'FEMALE', 1);
INSERT INTO dinosaur (habitat_id, energy, gender,
species_id) VALUES (1163, 714, 'MALE', 359);
INSERT INTO dinosaur (habitat_id, energy, gender,
species_id) VALUES (2954, 448, 'OTHER', 146);
…
```

9. Food

```
INSERT INTO food (name, price, energy_provide) VALUES
('pork', 143, 826);
INSERT INTO food (name, price, energy_provide, type)
VALUES ('grass', 109, 600, 'HERBIVORES');
INSERT INTO food (name, price, energy_provide, type)
VALUES ('sushi', 110, 854, 'OMNIVORES');
INSERT INTO food (name, price, energy_provide) VALUES
('chicken', 75, 463);
INSERT INTO food (name, price, energy_provide, type)
VALUES ('apple', 128, 781, 'HERBIVORES');
INSERT INTO food (name, price, energy_provide, type)
VALUES ('banana', 70, 797, 'HERBIVORES');
INSERT INTO food (name, price, energy_provide) VALUES
('beef', 113, 504);
```

10.Resource (50)

```
INSERT INTO resource (name) VALUES ('mine');
INSERT INTO resource (name) VALUES ('field');
INSERT INTO resource (name) VALUES ('mountain');
…
```

11. Item (100)

```
INSERT INTO item (name) VALUES ('sunglasses');
INSERT INTO item (name) VALUES ('swimsuit');
INSERT INTO item (name) VALUES ('ring');
…
```

12. Building

```
INSERT INTO building (name, description) VALUES
('Shop', 'Место для покупки еды, предметов и
динозавров');
INSERT INTO building (name, description) VALUES
('Mating room', 'Место для спаривания и размножения
динозавров');
INSERT INTO building (name, description) VALUES
('Admin House', 'Административное здание');
INSERT INTO building (name, description) VALUES
('Storehouse', 'Хранилище пользователя');
INSERT INTO building (name, description) VALUES
('Port', 'Гавань');
```

13. Ability (50)

```
INSERT INTO ability (earning_increase,
max_energy_increase) VALUES (17, 18);
INSERT INTO ability (earning_increase,
max_energy_increase) VALUES (10, 1);
…
```

Ассоциативные сущности

1. **Account_food (3500 = 500 * 7)**

```
INSERT INTO account_food (account_id, food_id,
amount) VALUES (1, 1, 41);
INSERT INTO account_food (account_id, food_id,
amount) VALUES (1, 2, 85);
…
```

2. **Account_material (2500 = 500*5)**

```
INSERT INTO account_material (account_id,
material_id, amount) VALUES (1, 1, 41);
INSERT INTO account_material (account_id,
material_id, amount) VALUES (1, 2, 21);
INSERT INTO account_material (account_id,
material_id, amount) VALUES (1, 3, 319);
…
```

3. **Island_resource (2525)**

```
INSERT INTO island_resource (island_id, resource_id,
amount) VALUES (1, 18, 5);
INSERT INTO island_resource (island_id, resource_id,
amount) VALUES (2, 20, 5);
INSERT INTO island_resource (island_id, resource_id,
amount) VALUES (2, 29, 4);
…
```

4. **Dinosaur_habitat (5000)**

```
INSERT INTO dinosaur_habitat (dinosaur_id,
habitat_id) VALUES (1, 42);
INSERT INTO dinosaur_habitat (dinosaur_id,
habitat_id) VALUES (2, 468);
INSERT INTO dinosaur_habitat (dinosaur_id,
habitat_id) VALUES (3, 335);
INSERT INTO dinosaur_habitat (dinosaur_id,
habitat_id) VALUES (4, 2501);
…
```

5. **Resource_material (60)**

```
INSERT INTO resource_material (resource_id,
material_id, amount) VALUES (42, 3, 35);
INSERT INTO resource_material (resource_id,
material_id, amount) VALUES (1, 5, 25);
INSERT INTO resource_material (resource_id,
material_id, amount) VALUES (29, 4, 63);
…
```

6. **Build_requirement (10)**

```
INSERT INTO build_requirement (building_id,
material_id, amount) VALUES (2, 3, 122);
INSERT INTO build_requirement (building_id,
material_id, amount) VALUES (1, 5, 119);
INSERT INTO build_requirement (building_id,
material_id, amount) VALUES (4, 4, 146);
…
```

7. **Upgrade_requirement (18)**

```
INSERT INTO upgrade_requirement (level, material_id,
amount) VALUES (1, 3, 122);
INSERT INTO upgrade_requirement (level, material_id,
amount) VALUES (1, 1, 130);
INSERT INTO upgrade_requirement (level, material_id,
amount) VALUES (1, 5, 115);
INSERT INTO upgrade_requirement (level, material_id,
amount) VALUES (2, 3, 72);
…
```

8. **Species_environment (700)**

```
INSERT INTO species_environment (species_id,
environment_id) VALUES (42, 8);
INSERT INTO species_environment (species_id,
environment_id) VALUES (335, 1);
INSERT INTO species_environment (species_id,
environment_id) VALUES (170, 2);
…
```

9. **Item_ability (200)**

```
INSERT INTO item_ability (item_id, ability_id) VALUES
(42, 18);
INSERT INTO item_ability (item_id, ability_id) VALUES
(35, 1);
INSERT INTO item_ability (item_id, ability_id) VALUES
(70, 25);
…
```

10. **Inventory (6000)**

```
INSERT INTO inventory (account_id, item_id,
dinosaur_id) VALUES (42, 68, 1);
INSERT INTO inventory (account_id, item_id,
dinosaur_id) VALUES (335, 1, 2);
INSERT INTO inventory (account_id, item_id,
dinosaur_id) VALUES (170, 25, 3);…
```

IV. Вывод

Мы научились вычленять данные из моего предметной области, составлять инфологическую и даталогическую таблицы.