Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление системного и прикладного программного обеспечения

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

курса «Информационные сисемы и базы данных»

по теме: «Построение инфологической и даталогической модели»

Вариант № 335047

Выполнил студент:

Тюрин Иван Николаевич

группа: P33102

Преподаватель:

Сагадак Алина Алексеевна

Санкт-Петербург, 2023 г.

Содержание

Лабораторная работа № 1
Построение инфологической и даталогической модели

1. Задание варианта № 335047

, , ,

Составить инфологическую и даталогическую модели по следующему тексту и реализовать базу данных PostgreSQL на helios.se.ifmo.ru.

Элли знала, что на заре истории нашей планеты в растительном мире царили такие же законы борьбы за выживание, как и в животном, а порой даже и более жестокие. Яд, который вырабатывали Serenna veriformans, был всего лишь незначительным примером многообразного арсенала "химического оружия"растений. Например, некоторые растения выделяют терпены /Органические соединения, находящиеся в смоле хвойных деревьев/, пропитывая почву ядом и подавляя тем самым рост других растений.

, , ,

2. Выполнение задания

Текст взят из рассказа про фантастический «Парк Юрского периода». Пришлось посмотреть окружающий текст, чтобы понять, что является предметной областью в моем случае. Среди важных сущностей я выделил следующие:

- Character – **стержневая** сущность, персоныжи книги, действующие лица. Например, Элли, «другие люди», профессор Дуглас.
- Location – **характеристическая** сущность, локация в которой что-то происходит. Например, бассейн, лес и пр.
- Biology Entity – **стержнева** сущность, сущность в иерархии живых существ, те кто представлены в Парке Юрского периода. Например, Selenna Veriformans.
- Biology Hierarchy – **характеристическая** сущность, иерархии, к которым принадлежат сущности представленные в книге.
- Live Epoch – **характеристическая** сущность, эпоха, в которой относятся сущности, из какой эпохи их возродили на острове.
- Protection Method – **ассоциативная** сущность, метод защиты биологических существ от других биологических существ, средства о которых идет речь в тексте.
- Character Join Biology Entity – **ассоциативная** сущность, объединяющая Character и Biology Entity

Так же определил связи между выделенными сущностями.

- Character знает одного или нескольких Biology Entity, находится в какой-то Location, имеет имя.
- Location задает название локации.
- Biology Entity имеет определенное место в иерархии живых существ, имеет один или несколько Protection Method от других существ, жило в какой-то определенной Live Epoch и располагается в данных момент в Location.
- Biology Hierarchy определяет название иерархии и какие-то нестрого структурированные атрибуты этой иерархии.
- Live Epoch определяет название эпохи и ее временной интервал с помощью нестрого структурированных атрибутов.

- Protection Method определяет связь между двумя живыми существами и свойства этой связи.

Сущности были предсавлены на концептуальной (инфологической) и даталогической модели в виде ER-диаграммы (см. 1.1).

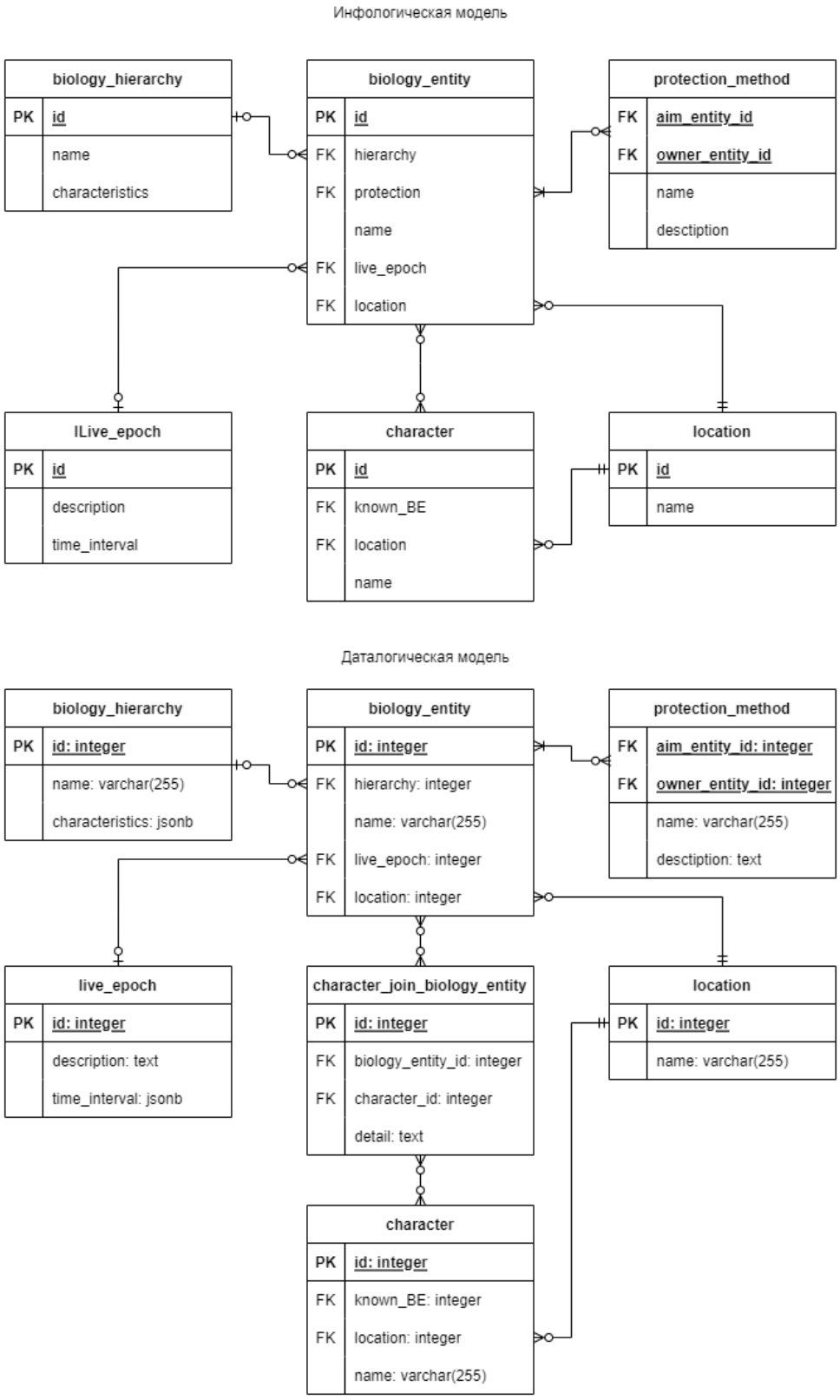

Рис. 1.1: ER-диаграмма инфологической и даталогическая модели

SQL скрипты для создания нужных таблиц представлены в листингах 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7.

```sql
CREATE TABLE IF NOT EXISTS "character"
(
    "id" integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    "name" character varying(255) NOT NULL,
    "location" integer,
    CONSTRAINT "location_fk"
    FOREIGN KEY ("location")
        REFERENCES "location" ("id")
        ON DELETE RESTRICT
);

ALTER TABLE IF EXISTS "character"
    OWNER TO s335047;
```

Листинг 1.1: DDL Character

```sql
CREATE TABLE IF NOT EXISTS "biology_entity"
(
    "id" integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    "hierarchy" integer,
    "name" character varying(255) NOT NULL,
    "live_epoch" integer,
    "location" integer NOT NULL,
    CONSTRAINT "biology_hierarchy_fk"
    FOREIGN KEY ("hierarchy")
        REFERENCES "biology_hierarchy" ("id")
        ON DELETE NO ACTION,
    CONSTRAINT "live_epoch_fk"
    FOREIGN KEY ("live_epoch")
        REFERENCES "live_epoch" ("id")
        ON DELETE NO ACTION,
    CONSTRAINT "location_fk"
    FOREIGN KEY ("location")
        REFERENCES "location" ("id")
        ON DELETE RESTRICT
);

ALTER TABLE IF EXISTS "biology_entity"
    OWNER TO s335047;
```

Листинг 1.2: DDL Biology Entity

Написанные скрипты выполнялись с помощью `psql` в базе данных `studs`. Для этого нужно было пробросить порт с локальной машины на сервер и запустив утилиту, выполнить команду `\i` с указанием скрипта, который должен быть выполнет.

Дальнейшее заполнение таблиц данными выполнялось с помощью графического приложения pgAdmin и с использованеим SQL/DML скриптов (см. 1.8).

```sql
CREATE TABLE If NOT EXISTS "biology_hierarchy"
(
    "id" integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    "name" character varying(255) NOT NUll,
    "characteristics" jsonb
);

ALTER TABLE IF EXISTS "biology_hierarchy"
    OWNER TO s335047;
```

Листинг 1.3: DDL Biology Hierarchy

```sql
CREATE TABLE IF NOT EXISTS "live_epoch"
(
    "id" integer CONSTRAINT "live_epoch_pk" PRIMARY KEY GENERATED ALWAYS
    AS IDENTITY,
    "description" text NOT NULL,
    "time_interval" jsonb
);

ALTER TABLE IF EXISTS "live_epoch"
    OWNER TO s335047;
```

Листинг 1.4: DDL Live Epoch

```sql
CREATE TABLE IF NOT EXISTS "location"
(
    "id" integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    "name" character varying(255) NOT NULL
);

ALTER TABLE IF EXISTS "location"
    OWNER TO s335047;
```

Листинг 1.5: DDL Location

```sql
CREATE TABLE IF NOT EXISTS "protection_method"
(
    "aim_entity_id" integer NOT NULL,
    "owner_entity_id" integer NOT NULL,
    "name" character varying(255) NOT NULL,
    "description" text NOT NULL,
    CONSTRAINT "aim_entity_fk"
    FOREIGN KEY ("aim_entity_id")
        REFERENCES "biology_entity" ("id")
        ON DELETE CASCADE,
    CONSTRAINT "owner_entity_fk"
    FOREIGN KEY ("owner_entity_id")
        REFERENCES "biology_entity" ("id")
        ON DELETE CASCADE,
    CONSTRAINT "protection_method_pk"
    PRIMARY KEY ("aim_entity_id", "owner_entity_id")
);

ALTER TABLE IF EXISTS "protection_method"
    OWNER TO s335047;
```

Листинг 1.6: DDL Protection Method

```sql
CREATE TABLE IF NOT EXISTS "character_join_biology_entity"
(
    "id" integer PRIMARY KEY GENERATED ALWAYS AS IDENTITY,
    "character_id" integer NOT NULL,
    "biology_entity_id" integer NOT NULL,
    "detail" text,
    CONSTRAINT "character_id_fk" FOREIGN KEY ("character_id")
        REFERENCES "character" ("id")
        ON DELETE CASCADE,
    CONSTRAINT "biology_entity_id_fk" FOREIGN KEY ("biology_entity_id")
        REFERENCES "biology_entity" ("id")
        ON DELETE CASCADE
);

ALTER TABLE IF EXISTS "character_join_biology_entity"
    OWNER TO s335047;
```

Листинг 1.7: DDL Character Join Biology Entity

```sql
INSERT INTO TABLE "character" (name, location)
SELECT 'Элли', loc.id
FROM (SELECT FIRST(id) FROM "location" WHERE name LIKE 'Бассейн') loc;

INSERT INTO TABLE "character" (name, location)
SELECT 'Дуглас', loc.id
FROM (SELECT FIRST(id) FROM "location" WHERE name LIKE 'В пределах планеты
    Земля') loc;
```

Листинг 1.8: DML Add Character

```
INSERT INTO TABLE "biology_entites" ("hierarchy", "name", "live_epoch", "
    location")
SELECT h."id", 'Папоротник', NULL, loc."id"
FROM (SELECT FIRST(id) FROM "biology_hierarchy" WHERE "name" LIKE '
    Папоротник') h,
      (SELECT FIRST(id) FROM "locatin" WHERE "name" LIKE 'Бассейн') loc;

INSERT INTO TABLE "biology_entites" ("hierarchy", "name", "live_epoch", "
    location")
SELECT h."id", 'Ели Дугласа', NULL, loc."id"
FROM (SELECT FIRST(id) FROM "biology_hierarchy" WHERE "name" LIKE 'Ели
    Дугласа') h,
      (SELECT FIRST(id) FROM "locatin" WHERE "name" LIKE 'Лес') loc;
```

Листинг 1.9: DML Add Biology Entity

```
INSERT INTO "biology_hierarchy"
VALUES (DEFAULT, 'Папоротники', 'царство{"": растения"", род"": папаротники""}');

INSERT INTO "biology_hierarchy"
VALUES (DEFAULT, 'Хищники', '{ класс"": хищники"" }');
```

Листинг 1.10: DML Add Biology Hierarchy

```
INSERT INTO TABLE "live_epoches" ( "description", "characteristics")
VALUES ( 'очень давно', 'прям{"оченьдавно__": рил""}');

INSERT INTO TABLE "live_epoches" ( "description", "characteristics")
VALUES ( 'kek', '{"epoch": "lol"}');

INSERT INTO TABLE "live_epoches" ( "description", "characteristics")
VALUES ( 'Настоящее время', '{"epoch": "now"}');

INSERT INTO TABLE "live_epoches" ( "description", "characteristics")
VALUES ( 'Not stated', '{"epoch": null}');
```

Листинг 1.11: DML Add Live Epoch

```
INSERT INTO TABLE "location" ("name")
VALUES ("Бассейн");

INSERT INTO TABLE "location" ("name")
VALUES ("Лес");

INSERT INTO TABLE "location" ("name")
VALUES ("В пределах планеты Земля");
```

Листинг 1.12: DML Add Location

```
INSERT INTO TABLE "protection_method" ( aim_entity_id, owner_entity_id,
    name, description)
SELECT aim.id, owner.id, 'феромоны', 'отпугивают насекомых, сообщаяют другим
    деревьям об опасности'
FROM (SELECT id FROM "biology_entity" WHERE name LIKE 'Ели Дугласа') aim,
    (SELECT id from "biology_entity" WHERE name LIKE 'Насекомые') owner;
```

Листинг 1.13: DML Add Protection Method

```
INSERT INTO "character_join_biology_entity" (character_id,
    biology_entity_id)
SELECT ch.id, be.id
FROM (SELECT id FROM "character" WHERE name LIKE 'Элли') ch,
    (SELECT id FROM "biology_entity" WHERE name LIKE 'Папоротник') be
```

Листинг 1.14: DML Add Character Join Biology Entity

3. Вывод

В резульате выполнения работы изучил правилами составления инфологических и даталогичкских моделей, написания SQL скриптов для БД PostgreSQL; поработал с утилитой `psql` и приложением pgAdmin.

Литература

[1] Ссылка на личный репозиторий GitHub: https://github.com/e1turin/itmo-comp-math/tree/dev-lab-1/lab-1