



НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ
Кафедра програмного забезпечення та комп'ютерних систем

Лабораторна робота №3

з дисципліни
«Бази даних і засоби управління»

Виконав: студент III курсу
ФПМ групи КП-83
Палій Дмитро Володимирович
Перевірів:
Радченко К.О.

Київ – 2020

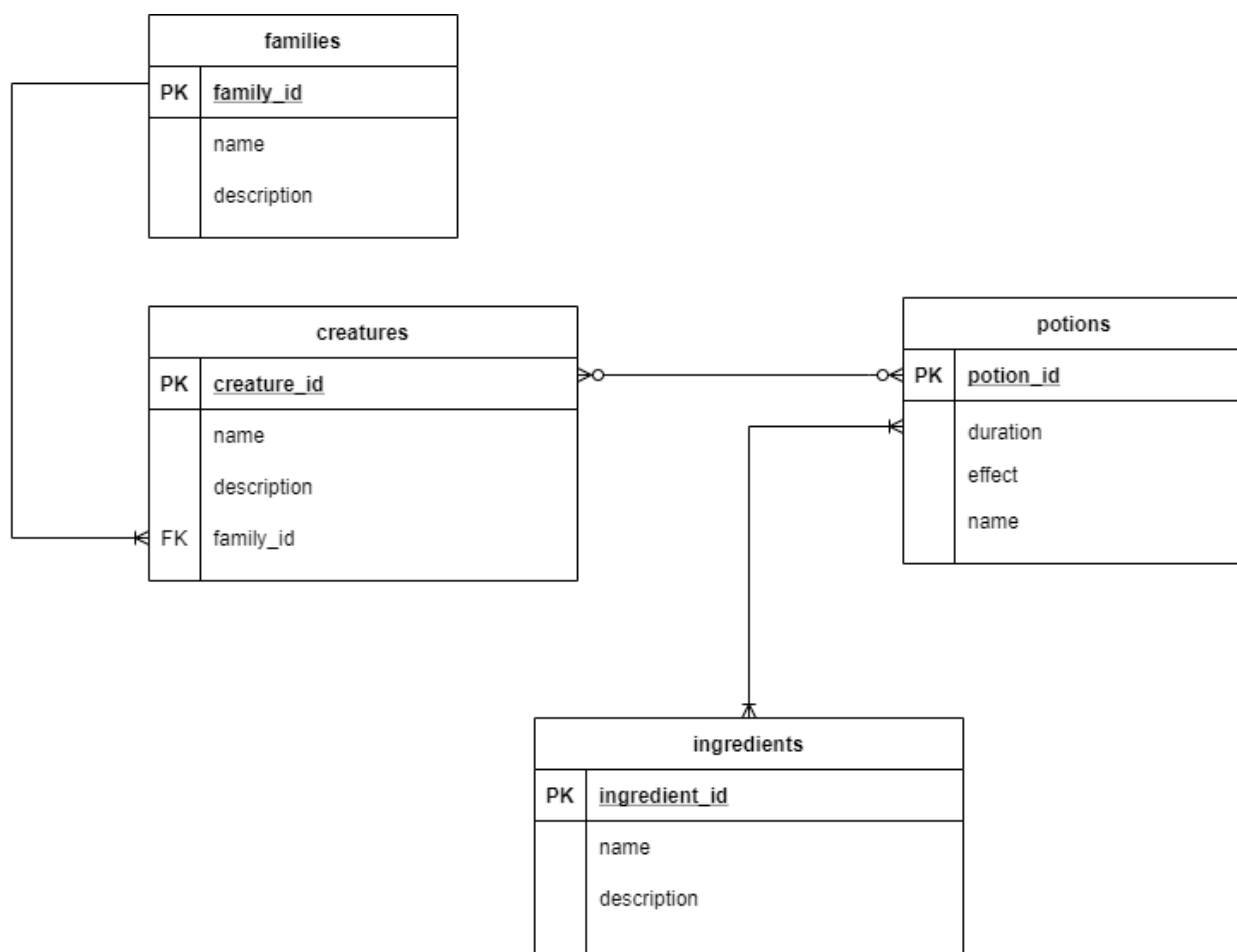
Завдання роботи

Варіант 16

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи No2 у вигляд об’єктно-реляційної проєкції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL (за варіантом – GIN, Hash).
3. Розробити тригер бази даних PostgreSQL PostgreSQL (за варіантом – after delete, insert).

Завдання 1

Структура БД:



Класи ORM:

```
class PotionIngredient(Base):
    __tablename__ = 'potion_ingredient'

    id = Column(Integer, primary_key=True)
```

```

    potion_id = Column(Integer, ForeignKey('potions.id'))
    ingredient_id = Column(Integer, ForeignKey('ingredients.id'))

    def __init__(self, potion_id=None, ingredient_id=None):
        self.potion_id = potion_id
        self.ingredient_id = ingredient_id

class PotionAgainstCreature(Base):
    __tablename__ = 'potion_against_creature'

    id = Column(Integer, primary_key=True)
    potion_id = Column(Integer, ForeignKey('potions.id'))
    creature_id = Column(Integer, ForeignKey('creatures.id'))

    def __init__(self, potion_id=None, creature_id=None):
        self.potion_id = potion_id
        self.creature_id = creature_id

class Creature(Base):
    __tablename__ = 'creatures'

    id = Column(Integer, primary_key=True)
    family_id = Column(Integer, ForeignKey('families.id'))
    name = Column(Text)
    description = Column(Text)

    potions_against_creatures = relationship(PotionAgainstCreature)

    def __init__(self, family_id=None, name=None, description=None):
        self.family_id = family_id
        self.name = name
        self.description = description

class Family(Base):
    __tablename__ = 'families'

    id = Column(Integer, primary_key=True)
    name = Column(Text)
    description = Column(Text)

    creatures = relationship('Creature')

    def __init__(self, name=None, description=None):
        self.name = name
        self.description = description

class Ingredient(Base):
    __tablename__ = 'ingredients'

    id = Column(Integer, primary_key=True)
    name = Column(Text)
    in_inventory = Column(Integer)

    potions_ingredients = relationship(PotionIngredient)

    def __init__(self, name=None, description=None, in_inventory=None):
        self.name = name
        self.description = description
        self.in_inventory = in_inventory

```

```

class Potion(Base):
    __tablename__ = 'potions'

    id = Column(Integer, primary_key=True)
    name = Column(Text)
    effect = Column(Text)
    duration = Column(Time)

    potions_ingredients = relationship(PotionIngredient)
    potions_against_creatures = relationship(PotionAgainstCreature)

    def __init__(self, name=None, duration=None, effect=None):
        self.name = name
        self.duration = duration
        self.effect = effect

```

Затити у вигляді ORM:

```

def pairs_from_str(self, string):
    lines = string.split(',')
    pairs = {}

    for line in lines:
        key, value = line.split('=')
        pairs[key.strip()] = value.strip()
    return pairs

def filter_by_pairs(self, query, pairs, cls):
    for key, value in pairs.items():
        field = getattr(cls, key)
        query = query.filter(field == value)
    return query

def get(self, tname, condition):
    object_class = TABLES[tname]

    query = session.query(object_class)

    if len(condition) > 0:
        pairs = self.pairs_from_str(condition)
        query = self.filter_by_pairs(query, pairs, object_class)

    return query.all()

def insert(self, tname, columns, values):
    columns = [c.strip() for c in columns.split(',')]
    values = [v.strip() for v in values.split(',')]

    pairs = dict(zip(columns, values))
    object_class = TABLES[tname]
    obj = object_class(**pairs)

    session.add(obj)

def commit(self):
    session.commit()

def delete(self, tname, condition):
    pairs = self.pairs_from_str(condition)
    object_class = TABLES[tname]

```

```

objects = session.query(object_class)
objects = self.filter_by_pairs(objects, pairs, object_class)

objects.delete()

def update(self, tname, condition, statement):
    pairs = self.pairs_from_str(condition)
    new_values = self.pairs_from_str(statement)
    object_class = TABLES[tname]

    objects = session.query(object_class)
    objects = self.filter_by_pairs(objects, pairs, object_class)

    for obj in objects:
        for field_name, value in new_values.items():
            setattr(obj, field_name, value)

```

Завдання 2

GIN індекс:

```

ALTER TABLE potions
ADD COLUMN ts_vector tsvector;

UPDATE potions
SET ts_vector = to_tsvector(effect)
WHERE true;

CREATE INDEX gin_index
ON potions
USING gin(ts_vector)

```

Пошук по частковому співпадінню

- *без індексів:*

```

EXPLAIN ANALYSE SELECT *
FROM potions
WHERE effect LIKE 'ab%'

```

	QUERY PLAN
	text
1	Seq Scan on potions (cost=0.00..318.19 rows=1 width=83) (actual time=0.119..2.699 rows=31 loops=1)
2	Filter: (effect ~~ 'ab%':text)
3	Rows Removed by Filter: 9984
4	Planning Time: 0.483 ms
5	Execution Time: 2.719 ms

- з індексами:

```
EXPLAIN ANALYZE
SELECT *
FROM potions
WHERE ts_vector @@ to_tsquery('ab:*');
```


	QUERY PLAN
	text
1	Bitmap Heap Scan on potions (cost=164.25..168.52 rows=1 width=83) (actual time=1.577..1.635 rows=38 loops=1)
2	Recheck Cond: (ts_vector @@ to_tsquery('ab:*::text'))
3	Heap Blocks: exact=36
4	-> Bitmap Index Scan on gin_index (cost=0.00..164.25 rows=1 width=0) (actual time=1.561..1.562 rows=38 loops=1)
5	Index Cond: (ts_vector @@ to_tsquery('ab:*::text'))
6	Planning Time: 0.519 ms
7	Execution Time: 1.731 ms

Повнотекстовий пошук з індексами швидший, бо кількість проіндексованих лексем менша за кількість записів у таблиці, а пошук запису в таблиці після знаходження лексеми у дереві лексем майже не займає часу.

Hash індекс:

```
CREATE INDEX ON potions USING hash(name);
```

```
EXPLAIN ANALYZE SELECT *
FROM potions
```

QUERY PLAN	
text	
Seq Scan on potions (cost=0.00..33.14 rows=1014 width=67) (actual time=0.077..0.251 rows=1014 loops=1)	
Planning Time: 0.098 ms	
Execution Time: 0.288 ms	

```
EXPLAIN ANALYZE SELECT *
FROM potions
WHERE name = 'Cat';
```

QUERY PLAN

text



Index Scan using potions_name_idx on potions (cost=0.00..8.02 rows=1 width=67) (actual time=0.090..0.091 rows=1 loops=1)

Index Cond: (name = 'Cat'::text)

Planning Time: 0.256 ms

Execution Time: 0.189 ms

Пошук за хеш індексом є швидшим, адже дозволяє шукати перебором не усіх значень таблиці, а лише тих, у які хеш-функція могла віднести записи з даним ключем.

Завдання 3

AFTER INSERT

Після створення нового зілля, активується тригер, що додає до назви зілля префікс 'Weak', якщо тривалість його дії менша за 12 годин, або префікс 'Strong' в іншому випадку

```
CREATE OR REPLACE FUNCTION afterInsertPotionFunc()  
RETURNS trigger AS  
  
$$  
  
BEGIN  
    IF NEW.duration < '12:00:00'::time  
    THEN  
        UPDATE potions SET name= 'Weak ' || NEW.name  
WHERE id= NEW.id;  
    ELSE  
        UPDATE potions SET name= 'Strong ' || NEW.name  
WHERE id= NEW.id;  
    END IF;  
    return NEW;  
END;  
  
$$  
  
LANGUAGE 'plpgsql';  
  
CREATE TRIGGER potionsInsertTrigger  
    AFTER INSERT  
    ON "potions"  
    FOR EACH ROW  
    EXECUTE PROCEDURE afterInsertPotionFunc();
```

До виконання команди, що активує тригер:






id [PK] integer	duration time without time zone	effect text	name text
1	08:00:00	Renders i...	Golden Oriole
2	00:05:00	Grants si...	Cat
9	04:50:11	fff2b66ab...	c662aefe0a
10	14:29:22	49ad6cea...	13fafa283d
11	20:38:58	be32318...	b381122c27
12	11:33:25	1e79e5ba...	81257ede68
13	12:03:45	6954f688...	86a3a4f332
14	23:00:30	f0270f55...	93b67e5b37
15	08:19:42	3837f6af...	b4b14c43e3
16	03:02:33	f3636b03...	23cb2a7e27
17	04:51:46	cc439b7...	07309a37a8
18	08:44:48	756513d...	85b434aeb2
19	02:17:32	60e15f9f...	f4726c9db5
20	10:10:32	8c6dff8b...	2900c1d474

Після виконання команди, що активує тригер:

```
INSERT INTO potions(name,effect,duration)
VALUES('cool potion','cool effect','10:00:00')
```


1	101020	10:00:00	cool effect	Weak cool potion
2	1	08:00:00	Renders immune to p...	Golden Oriole
3	2	00:05:00	Grants sight in total d...	Cat
4	9	04:50:11	fff2b66ab811b1a	c662aefe0a
5	10	14:29:22	49ad6ceae83d3cb	13fafa283d
6	11	20:38:58	be32318b256d8ba	b381122c27
7	12	11:33:25	1e79e5bab9b4c27	81257ede68
8	13	12:03:45	6954f688cf2bb03	86a3a4f332
9	14	23:00:30	f0270f554f11b46	93b67e5b37
10	15	08:19:42	3837f6af8cff7e7	b4b14c43e3
11	16	03:02:33	f3636b038bcb9ee	23cb2a7e27
12	17	04:51:46	cc439b79cb21cc6	07309a37a8
13	18	08:44:48	756513d666dd3d5	85b434aeb2
14	19	02:17:32	60e15f9fe4d22f8	f4726c9db5
15	20	10:10:32	8c6dff8b40e10fa	2900c1d474

```
INSERT INTO potions(name,effect,duration)
VALUES('greater potion','greater effect','20:00:00')
```

	 id [PK] integer 	duration time without time zone 	effect text 	name text 
1	111021	20:00:00	greater ef...	Strong greater potion
2	101427	18:34:59	7f485e42...	Strong 2bbee78e0e
3	101428	17:32:11	938ae93...	Strong 8406aa4b1e
4	101429	05:24:05	b7d38e7...	Weak cd3a40321c
5	101430	16:54:19	c247d30...	Strong 84dd20b4ee
6	101431	22:56:11	762b57e...	Strong 7f46e21ded

AFTER DELETE

Після видалення сімейства створінь, активується тригер, який видаляє всіх створінь, що належали до даного сімейства з таблиці ‘creatures’ та створює аналогічні записи в таблиці ‘extinct_creatures’.

```
CREATE OR REPLACE FUNCTION afterDeleteFamilyFunc()
RETURNS trigger AS $$

DECLARE
    creatures_ cursor is select * from creatures where
    family_id IS NULL;
```

```

BEGIN
FOR creature IN creatures_ LOOP
    DELETE FROM creatures
    WHERE id = creature.id;
    INSERT INTO extinct_creatures (name,
description)
        VALUES (creature.name, creature.description);
    end loop;
    return OLD;
END
$$
LANGUAGE 'plpgsql';

CREATE TRIGGER afterDeleteFamily
AFTER DELETE
ON "families"
FOR EACH ROW
EXECUTE PROCEDURE afterDeleteFamilyFunc();

```

```


DELETE FROM families
WHERE id=1

```

‘creatures’ до видалення:

id [PK] integer	family_id integer	name text	description text
4	1	Arches...	Archespores loo...
5	1	Berserk...	Berserkers are t...
6	1	Werew...	Werewolves are ...
1	[null]	Aracha...	Described as an...
2	[null]	Arachn...	Arachnomorphs...
3	[null]	Sandcr...	Sandcrabs have...
7	[null]	Basilisk	Basilisk is a dra...

‘extinct_creatures’ після видалення:

id [PK] integer 	name text 	description text 
1	Aracha...	Described as an...
2	Arachn...	Arachnomorphs...
3	Sandcr...	Sandcrabs have...
4	Basilisk	Basilisk is a dra...
5	Arches...	Archespores loo...
6	Berserk...	Berserkers are t...
7	Werew...	Werewolves are ...