

Национальный исследовательский Университет ИТМО  
Мегафакультет компьютерных технологий и управления  
Факультет программной инженерии и компьютерной техники

# Информационные системы и базы данных

Курсовой проект

**Работу**

**выполнили:**

Н. В. Кулаков,

Н. К. Нестеров

Группа: Р33312

**Преподаватель:**

Д. М. Шешуков

Санкт-Петербург  
2022

# Содержание

<b>1. Этап 1</b>	<b>3</b>
1.1. Описание предметной области . . . . .	3
1.2. Описание бизнес процессов . . . . .	3
1.3. Список сущностей и их классификации . . . . .	3
<b>2. Этап 2</b>	<b>4</b>
2.1. Инфологическая модель . . . . .	4
2.2. Даталогическая модель . . . . .	5
<b>3. Этап 3</b>	<b>5</b>
3.1. Запросы . . . . .	5
3.1.1. Для администраторов системы . . . . .	5
3.1.2. Для поставщика . . . . .	5
3.1.3. Для владельца кафе . . . . .	5
3.1.4. Для пользователя . . . . .	6
3.2. Создание объектов . . . . .	6
3.2.1. Таблицы . . . . .	6
3.2.2. Функции . . . . .	8
3.2.3. Процедуры . . . . .	8
3.3. Удаление объектов . . . . .	9
3.3.1. Таблицы . . . . .	9
3.3.2. Процедуры . . . . .	9
3.3.3. Функции . . . . .	10
3.4. Начальные значения . . . . .	10

# 1. Этап 1

## 1.1. Описание предметной области

В мире существует множество кофеен, которые объединяются в целые сети. Владелец таких заведений нужно иметь возможность мониторить их состояние и управлять ими, а покупателям (которые знают SQL, конечно же) всегда удобнее, когда у кофейни доступно актуальное меню в виде БД!

Без такой системы владельцы тратят миллионы на товары, которые в конце концов будут выброшены, так как успели просрочиться. А покупателям приходится неделями стоять в очереди за любимым блюдом или напитком, в ожидании того, что кофейня закупит нужные ингредиенты.

## 1.2. Описание бизнес процессов

Владелец может для каждой конкретной кофейни записывать доступное количество ингредиентов, изменять его при использовании ингредиентов, смотреть цены на разные товары у поставщиков, пополнять запасы.

Владелец может создавать меню с различными блюдами, привязанными к конкретному заведению и менять их в зависимости от необходимости. Эти меню будут видны и покупателям.

Продавцы продуктов могут выставлять свои товары, которые будут доступны для покупки в любом заведении, подключенном к этой системе.

## 1.3. Список сущностей и их классификации

### Стержневые сущности:

- Поставщик товаров (название, описание)
- Кафе (адрес, комментарий)
- Меню (название, комментарий)
- Рецепт (название, описание, информация о питательных свойствах)
- Ингредиент (название, комментарий, единицы измерения)

### Ассоциативные сущности:

- Цена ингредиента (количество, цена за единицу) - Поставщик М-М Ингредиент
- Ингредиент рецепта (количество, обязательность) - Рецепт М-М Ингредиент
- Запас продукта (количество, срок годности, дата покупки) - Кафе М-М Ингредиент
- Меню кафе - Кафе М-М Меню
- Пункт меню (цена) - Меню М-М Рецепт

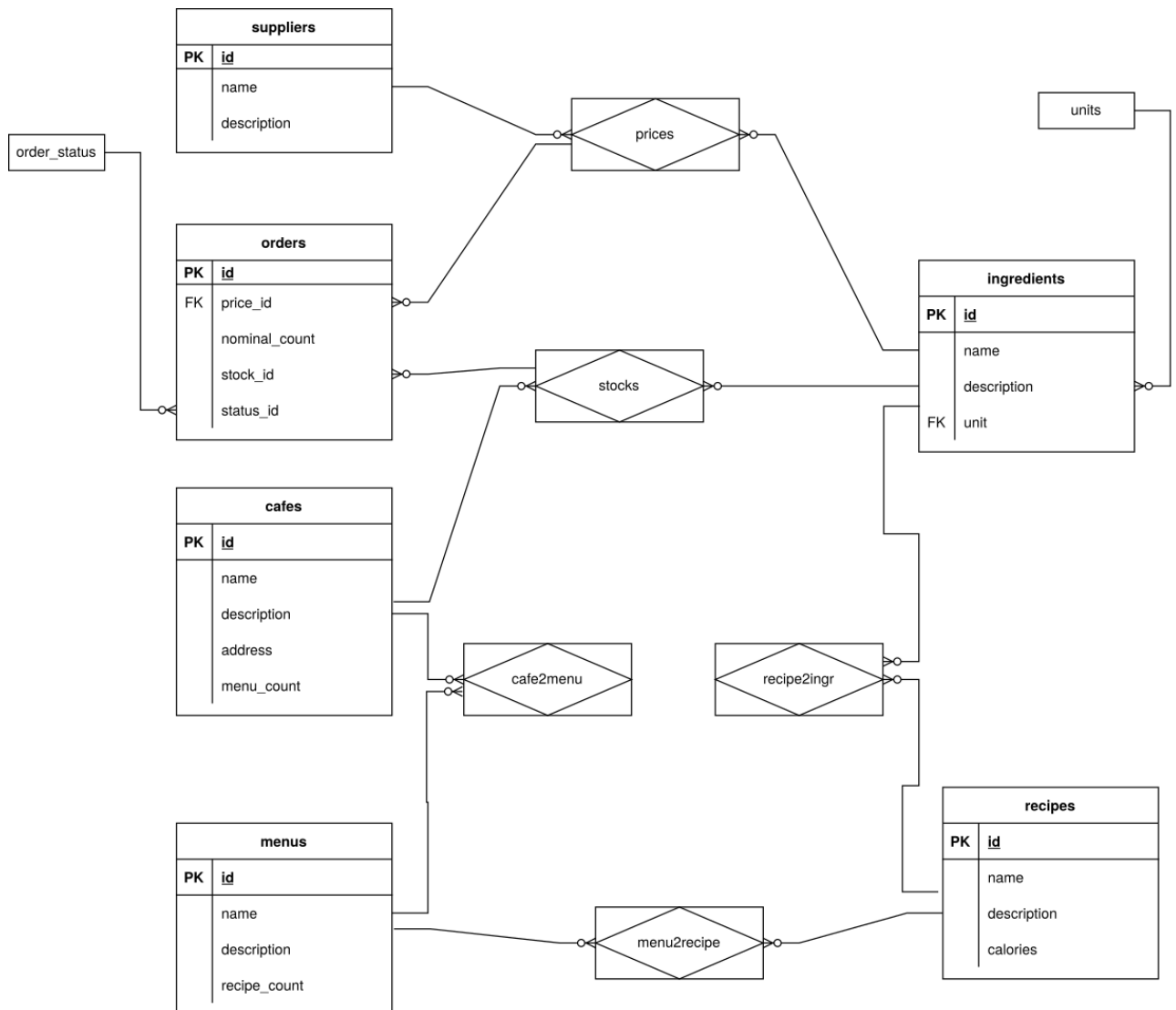
### Характеристические сущности:

- Единицы измерения (название) - к ингредиентам

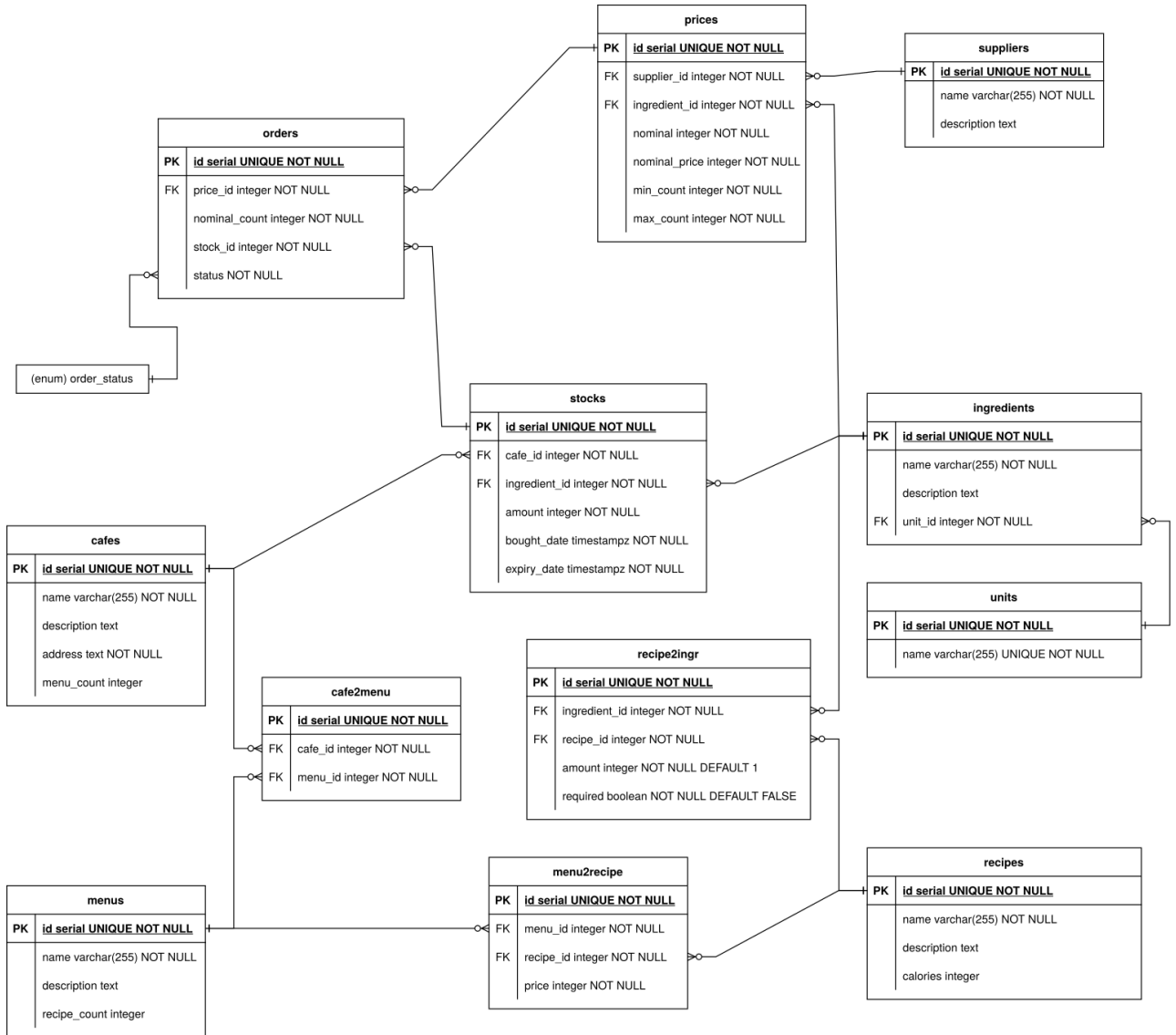
Характеристическая сущность (характеристика) — связь вида "многие-к-одной" или "одна-к-одной" между двумя сущностями (частный случай ассоциации). Цель характеристики - описание или уточнение некоторой другой сущности.

## 2. Этап 2

### 2.1. Инфологическая модель



## 2.2. Даталогическая модель



### 3. Этап 3

### 3.1. Запросы

### 3.1.1. Для администраторов системы

- Редактирование доступных единиц измерения
- Редактирование доступных ингредиентов

### 3.1.2. Для поставщика

- Создание и удаление аккаунта поставщика
- Размещение цен на товары

### 3.1.3. Для владельца кафе

- Добавление, изменение, удаление кафе
- Добавление, удаление, изменение меню
- Добавление, удаление меню в кафе

- Добавление, изменение, удаление рецептов
- Добавление, удаление рецептов в меню
- Поиск товаров у поставщиков
- Покупка товара у поставщиков
- Получение информации о купленных товарах, т.е. кассового чека, и обновление склада при изменении статуса чека на получено
- Редактирование и удаление записей об инвенторе

#### 3.1.4. Для пользователя

- Просмотр меню кафе
- Просмотр рецептов блюд

### 3.2. Создание объектов

Идет в правильном порядке запуска скриптов.

#### 3.2.1. Таблицы

```
-- Characteristic
-- NOTE: mb replace with enum
create table units(
  id serial primary key,
  name varchar(255) not null unique
);

create type order_status as enum();

-- Core
create table suppliers(
  id serial primary key,
  name varchar(255) not null,
  description text
);

create table cafes(
  id serial primary key,
  name varchar(255) not null,
  description text,
  address text not null,
  menu_count integer not null
);

create table menus(
  id serial primary key,
  name varchar(255) not null,
  description text,
  recipe_count integer not null
);

create table recipes(
  id serial primary key,
  name varchar(255) not null,
  description text,
  calories integer
  check(calories >= 0)
);
```

```

create table ingredients(
  id serial primary key,
  name varchar(255) not null,
  description text,
  unit_id integer not null references units
    on update cascade on delete cascade
);

-- Associative
create table prices(
  id serial primary key,
  supplier_id integer not null references suppliers
    on update cascade on delete cascade,
  ingredient_id integer not null references ingredients
    on update cascade on delete cascade,
  nominal integer not null
    check(nominal > 0),
  nominal_price integer not null
    check(nominal_price >= 0),
  min_count integer not null
    check(min_count > 0),
  max_count integer,
  constraint price_nominal_count check(min_count <= max_count or max_count is null)
);

create table orders(
  id serial primary key,
  price_id integer not null,
  nominal_count integer not null,
  stock_id integer not null,
  status order_status not null
);

create table recipe2ingr(
  id serial primary key,
  recipe_id integer not null references recipes
    on update cascade on delete cascade,
  ingredient_id integer not null references ingredients
    on update cascade on delete cascade,
  amount integer default 1 not null
    check(amount >= 0),
  required boolean default false not null
);

create table stocks(
  id serial primary key,
  cafe_id integer not null references cafes
    on update cascade on delete cascade,
  ingredient_id integer not null references ingredients
    on update cascade on delete cascade,
  amount integer not null
    check(amount >= 0),
  bought_date timestamp with time zone not null,
  expiry_date timestamp with time zone not null,
  constraint stock_bought_expiry_date check(bought_date < expiry_date)
);

create table cafe2menu(
  id serial primary key,

```

```

    cafe_id integer not null references cafes,
    -- on update cascade on delete cascade,
    menu_id integer not null references menus
    -- on update cascade on delete cascade
);

create table menu2recipe(
    id serial primary key,
    menu_id integer not null references menus
        on update cascade on delete cascade,
    recipe_id integer not null references recipes
        on update cascade on delete cascade,
    price integer not null
        check(price >= 0)
);

```

### 3.2.2. Функции

### 3.2.3. Процедуры

```

-- NOTE: if cafe_menu relations contain menu_id that are not present in table menu
create or replace function tf_cafe_menu_count()
returns trigger
language plpgsql
as $$
begin
    select count(id) into NEW.menu_count from cafe2menu where cafe2menu.menu_id = NEW.id;
    return NEW;
end
$$;

create or replace trigger tr_cafe_menu_count
before insert on cafes
for each row execute procedure tf_cafe_menu_count();

create or replace function tf_cafe_update_count()
returns trigger
language plpgsql
as $$
begin
    if (TG_OP = 'INSERT') then
        update cafe set menu_count = menu_count + 1 where id = NEW.cafe_id;
    elseif (TG_OP = 'DELETE') then
        update cafe set menu_count = menu_count - 1 where id = OLD.cafe_id;
    return OLD;
    end if;
    return NEW;
end
$$;

create or replace trigger tr_cafe_update_count
before insert or delete on cafe2menu
for each row execute procedure tf_cafe_update_count();

-- NOTE: if menu_item relations contain item_id that are not present in table menu
create or replace function tf_menu_recipe_count()
returns trigger
language plpgsql

```



```

as $$
begin
    select count(id) into NEW.recipe_count from menu2recipe where menu2recipe.menu_id = NEW
        .id;
    return NEW;
end
$$;

create or replace trigger tr_menu_recipe_count
before insert on menus
for each row execute procedure tf_menu_recipe_count();

create or replace function tf_menu_update_count()
returns trigger
language plpgsql
as $$
begin
    if (TG_OP = 'INSERT') then
        update menu set recipe_count = recipe_count + 1 where id = NEW.menu_id;
    elseif (TG_OP = 'DELETE') then
        update menu set recipe_count = recipe_count - 1 where id = OLD.menu_id;
        return OLD;
    end if;
    return NEW;
end
$$;

create or replace trigger tr_menu_update_count
before insert or delete on menu2recipe
for each row execute procedure tf_menu_update_count();

```

### 3.3. Удаление объектов

#### 3.3.1. Таблицы

```

drop table if exists
units,
suppliers,
cafes,
menus,
recipes,
ingredients,
prices,
orders,
recipe2ingr,
stocks,
cafe2menu,
menu2recipe
cascade;

drop type order_status;

```

#### 3.3.2. Процедуры

```

-- Trigger functions
drop function if exists
tf_cafe_menu_count,
tf_cafe_update_count,
tf_menu_recipe_count,
tf_menu_update_count

```

```
cascade;
```

### 3.3.3. Функции

## 3.4. Начальные значения

```
alter type order_status add value 'paid';
alter type order_status add value 'received';

insert into units(name) values('milliliter');
insert into units(name) values('unit'); -- колво- штук

insert into units(name) values('gram');

-- ounce
insert into units(name) values('ounce');

create or replace function ounce_to_gram(cnt double precision)
returns double precision
as $$
begin
    return cnt * 28.35;
end
$$
language plpgsql;

create or replace function gram_to_ounce(cnt double precision)
returns double precision
as $$
begin
    return cnt / 28.35;
end
$$
language plpgsql;

-- pounds
insert into units(name) values('pounds');

create or replace function pound_to_ounce(cnt double precision)
returns double precision
as $$
begin
    return cnt * 16;
end
$$
language plpgsql;

create or replace function ounce_to_pound(cnt double precision)
returns double precision
as $$
begin
    return cnt / 16;
end
$$
language plpgsql;

drop function if exists
    ounce_to_gram,
    gram_to_ounce,
    pound_to_ounce,
```

```
ounce_to_pound  
cascade;
```