لیست Entity ها و Weak Entity ها و strong Entity مربوط به آن ها

Entities:

Delivery (

Name,

Last Name,

Phone Number,

Delivery_id: pk

)

Sale Factor (

Date,

Factor_id : pk

)

Customers (

Phone number,

Lastname,

Firstname,

Age,

Personal ID: pk

)

Address (

Address_name,

Address,

LandLine

)

Buy Factor (

Date,

Buy_factor_id: pk

)

Stores (

Store_name,

Active,

StoreID:pk

)

Menu (

Current price,

Item_name: pk

)

Weak Entities:

Requested Items (

Request ID: pk,

Items_name: pk,

Number,

price

) Related Strong Entity: Menu

Bought Stuffs (

Stuff_name: pk,

boughtID: pk,

price

) Related Strong Entity: Price

لیست ویژگی های جدول ها

```
CREATE TABLE public.address (
    address_id integer NOT NULL,
    landline character varying(8) NOT NULL,
    address_name character varying(20) NOT NULL,
    address character varying(100) NOT NULL,
    personal_id character varying(20) NOT NULL
);

CREATE TABLE public.bought_stuffs (
    buy_factor_id integer NOT NULL,
    stuff_name character varying(20) NOT NULL,
    price bigint NOT NULL
);

CREATE TABLE public.buy_factor (
    buy_factor_id integer NOT NULL,
    date date NOT NULL,
    store_id integer
);

CREATE TABLE public.customers (
    personal_id character varying(20) NOT NULL,
    first_name character varying(20) NOT NULL,
    last_name character varying(20) NOT NULL,
    phone_number character varying(11) NOT NULL,
    age smallint NOT NULL
);
```

```
CREATE TABLE public.delivery (
    first_name character varying(20) NOT NULL,
    last_name character varying(20) NOT NULL,
    phone_number character varying(11) NOT NULL,
    delivery_id character varying(20) NOT NULL
);

CREATE TABLE public.menu (
    current_price bigint NOT NULL,
    item_name character varying(20) NOT NULL
);

CREATE TABLE public.requested_items (
    factor_id integer NOT NULL,
    item_name character varying(20) NOT NULL,
    number smallint NOT NULL,
    price bigint NOT NULL
);

CREATE TABLE public.sale_factor (
    factor_id integer NOT NULL,
    date date NOT NULL,
    personal_id character varying(20),
    address_id integer,
    delivery_id character varying(20)
);
```

```
CREATE TABLE public.stores (
    store_id integer NOT NULL,
    store_name character varying(20) NOT NULL,
    active boolean NOT NULL
);
```

لیست کلید های خارجی:

```
    ADD CONSTRAINT address_personal_id_fkey FOREIGN KEY (personal_id) REFERENCES
public.customers(personal_id) NOT VALID;


    ADD CONSTRAINT bought_stuffs_buy_factor_id_fkey FOREIGN KEY (buy_factor_id) REFERENCES
public.buy_factor(buy_factor_id);


    ADD CONSTRAINT buy_factor_store_id_fkey FOREIGN KEY (store_id) REFERENCES
public.stores(store_id) ON UPDATE SET NULL ON DELETE SET NULL NOT VALID;




    ADD CONSTRAINT requested_items_factor_id_fkey FOREIGN KEY (factor_id) REFERENCES
public.sale_factor(factor_id) NOT VALID;




    ADD CONSTRAINT sale_factor_address_id_fkey FOREIGN KEY (address_id) REFERENCES
public.address(address_id) NOT VALID;




    ADD CONSTRAINT sale_factor_delivery_id_fkey FOREIGN KEY (delivery_id) REFERENCES
public.delivery(delivery_id) ON UPDATE SET NULL ON DELETE SET NULL NOT VALID;
```

ADD CONSTRAINT sale_factor_personal_id_fkey FOREIGN KEY (personal_id) REFERENCES public.customers(personal_id) ON UPDATE SET NULL ON DELETE SET NULL NOT VALID;

لیست روابط

Delivers (

Deliver_id,

factor_id:pk

)

Requests(

Factor_id:pk,

Personal_id

)

Has_address(

Personal_id,

Address_id:pk

)

containItems (

factor_id,

item_name: pk,

Reques_id: pk,

)

Refers(

Request_id: pk,

Item_name:pk,

Item_name

)

Bought From (

Store_id,

Buy_factor_id:pk

)

Bought (

Buy_factor-id: pk,

Stuff_name,

boughtId

)

لیست کوری ها:

```
"INSERT INTO menu values (%s, %s)",
[current_price, item_name])
```

```
"insert into delivery values (%s, %s, %s, %s)",
[first_name, last_name, phone_number, delivery_id]
```

```
"insert into stores (store_name, active) values (%s, %s)",
[store_name, True]
```

```
"insert into sale_factor (date, personal_id, address_id, delivery_id) "
"values (%s, %s, %s, %s) RETURNING factor_id",
[datetime.date.today(), personal_id, address_id, delivery_id]
```

```
"select current_price from menu where item_name = %s", [item['item_name']]
```

```
"insert into requested_items values (%s, %s, %s, %s)",
[factor_id, item['item_name'], item['number'], price]
```

```
"insert into buy_factor (date, store_id)"
" values (%s, %s) RETURNING buy_factor_id",
[datetime.date.today(), store_id]
```

```
"insert into bought_stuffs values (%s, %s, %s)",
[buy_factor_id, item['item_name'], item['price']]
```

```
"update delivery set first_name = %s, last_name = %s, phone_number = %s "
"where delivery_id = %s", [first_name, last_name, phone_number, delivery_id]
```

```
"delete from delivery where delivery_id = %s", [delivery_id]
```

```
"update stores set store_name = %s, active = %s"
" where store_id = %s", [store_name, active, store_id]
```

```
"delete from stores where store_id = %s", [store_id]
```

```
"update menu set  current_price = %s where item_name = %s",
[price, item_name]
```

```
"delete from menu where item_name = %s", [item_name]
```

```
"select sum(price * number) from requested_items natural join sale_factor where date =
%s::date",
[date]
```

```
'select sum(price) from bought_stuffs natural join sale_factor where date = %s::date',
[date]
```

```
'select sum(price * number), factor_id from requested_items natural join sale_factor
where date = %s::date group by factor_id',
[date]
```

```
'select sum(price), buy_factor_id from bought_stuffs natural join sale_factor where
date = %s::date group by buy_factor_id',
[date]
```

```
"select store_id, store_name from stores where active = true"
```

```
"select * from  " + log_table,
```

```
"update customers "
"set first_name = %s, last_name = %s, phone_number = %s, age = %s "
"where personal_id = %s",
[first_name, last_name, phone_number, age, personal_id]
```

```
cursor.execute("INSERT INTO customers values (%s, %s, %s, %s, %s)",
               [personal_id, first_name, last_name, phone_number, age])
```

```
"INSERT INTO address (landline, address_name, address, personal_id) values (%s, %s,
%s, %s)",
[landline, address_name, address, personal_id])
```

```
"delete from customers where personal_id = %s", [personal_id]
```

```
"select * from sale_factor natural join requested_items where personal_id = %s order
by factor_id",
[personal_id]
```

```
"select sum(price) as t_price from sale_factor natural join requested_items where
personal_id = %s ",
[personal_id]
```

```
"WITH count_it as "
"(select item_name, count(*) as num "
"from sale_factor natural join requested_items where personal_id = %s"
" group by item_name) "
"select  num, item_name from count_it where num = (select max(num) from count_it)",
[personal_id]
```

```
'insert into restore values (%s, %s)', [table_name, query]
```

```
'select query from restore where table_name = %s', [table_name]
```

توضیحات پروژه:

برای ساخت این پروژه در قسمت backend از django استفاده شده است و در قسمت front end از vue استفاده شده

از postgres به عنوان DBMS استفاده کردیم

برای انجام log ها از trigger استفاده کردیم و برای هر جدول یک جدول لاگ درست کردیم و برای تمام trigger ها از یک function استفاده شده است به نام logger که در این تابع تمام کد های لازم برای لاگ کردن تمام جدول ها آمده است و سپس به تمام جدول ها یک trigger اضافه کردیم که تابع logger را اضافه میکنند

برای حفظ قیمت و نام غذا در فاکتور در هنگام تغییر قیمت و یا حذف آیتم از منو آمدیم فارن کی را در فرم یک trigger شبیه سازی کردیم و به به جدول requested_items به جای اضافه کردن فارن کی به منو تریگر نوشته شده را اضافه کردیم که چک میکند که آیتم در منو است یا نه که اگر نبود با استفاده از transaction عملیات انجام شده را rollback میکنیم در غیر این صورت ایتم را اضافه میکند و همچنین برای حفظ قیمت در برابر تغییر قیمت خریداری شده ی سفارش را در requested items نگه میداریم

برای بررسی کردن valid بودن شماره تلفن های اضافه شده از constraint به نام check استفاده کردیم و در آن از regex استفاده کردیم به عنوان مثال برای چک کردن شماره ی تلفن همراه از regex به شکل زیر استفاده کردیم

[0-9]{11}