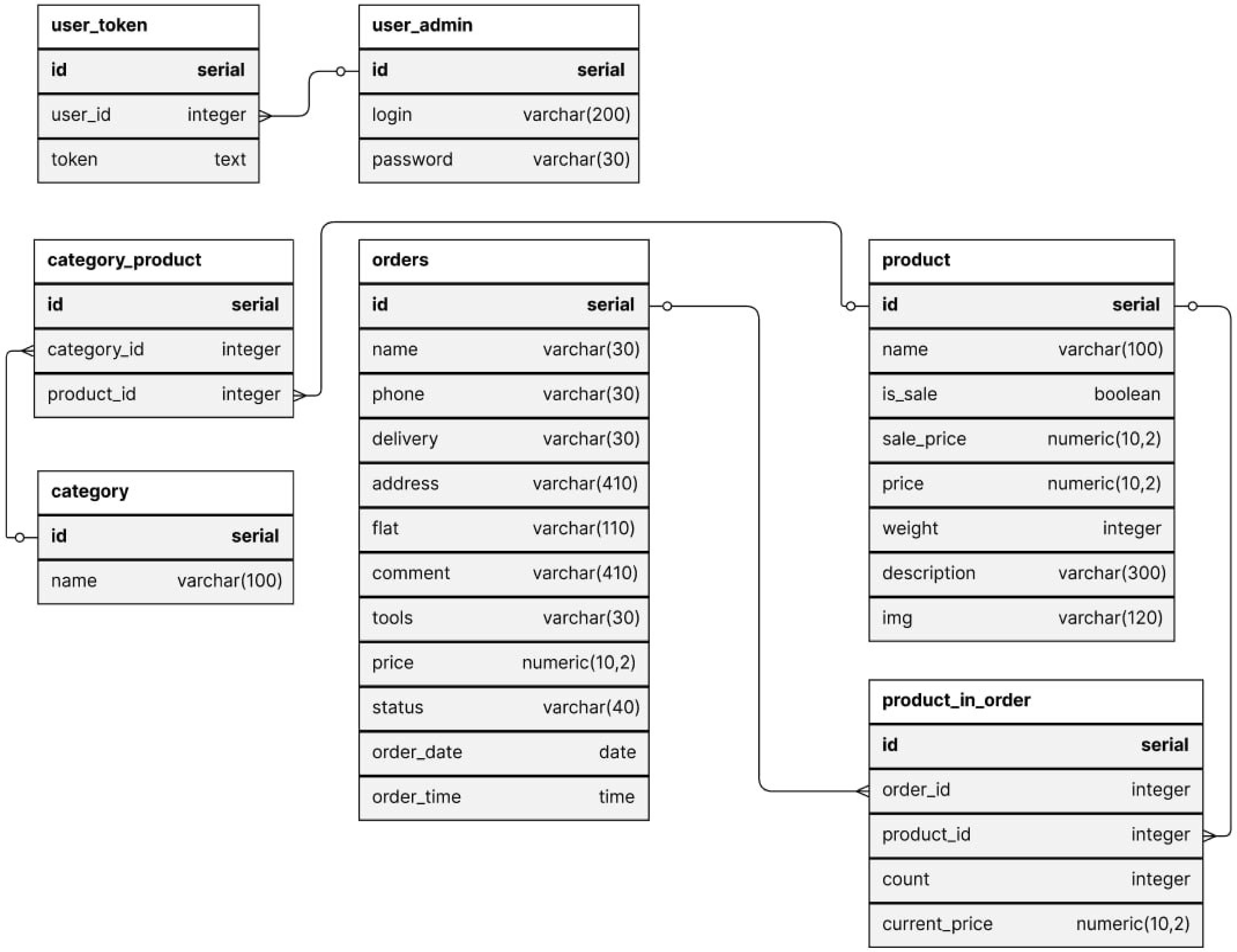

user_token	
id	**serial**
user_id	integer
token	text

user_admin	
id	**serial**
login	varchar(200)
password	varchar(30)

category_product	
id	**serial**
category_id	integer
product_id	integer

category	
id	**serial**
name	varchar(100)

orders	
id	**serial**
name	varchar(30)
phone	varchar(30)
delivery	varchar(30)
address	varchar(410)
flat	varchar(110)
comment	varchar(410)
tools	varchar(30)
price	numeric(10,2)
status	varchar(40)
order_date	date
order_time	time

product	
id	**serial**
name	varchar(100)
is_sale	boolean
sale_price	numeric(10,2)
price	numeric(10,2)
weight	integer
description	varchar(300)
img	varchar(120)

product_in_order	
id	**serial**
order_id	integer
product_id	integer
count	integer
current_price	numeric(10,2)

```
CREATE OR REPLACE FUNCTION save_product(
   name VARCHAR(100),
   is_sale BOOLEAN,
   sale_price NUMERIC(10, 2),
   price NUMERIC(10, 2),
   weight INTEGER,
   description VARCHAR(300),
   img_path VARCHAR(120),
   img_exp VARCHAR(15)
)
RETURNS INTEGER AS $$
DECLARE
   _id INTEGER;
BEGIN
   INSERT INTO product (
      name,
      is_sale,
      sale_price,
      price,
      weight,
      description,
      img
   )
   VALUES (
      name,
      is_sale,
      sale_price,
      price,
      weight,
      description,
      NULL
   )
   RETURNING id INTO _id;

   UPDATE product
   SET img = img_path || _id || '.' || img_exp
   WHERE id = _id;

   RETURN _id;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE VIEW category_with_all_products AS
WITH distinct_products AS (
   SELECT DISTINCT CP.category_id, CP.product_id
   FROM category_product CP
),
distinct_other_products AS (
   SELECT DISTINCT C.id AS category_id, P.id AS product_id
   FROM category C
   CROSS JOIN product P
   WHERE NOT EXISTS (
      SELECT 1
      FROM category_product CP2
      WHERE CP2.category_id = C.id
      AND CP2.product_id = P.id
   )
)
SELECT
   C.id AS id,
   C.name AS name,
   COALESCE(
      (SELECT JSON_AGG(
         JSON_BUILD_OBJECT(
            'id', P1.id,
                                          'name', P1.name,
                                          'price', P1.price,
                                          'isSale', P1.is_sale,
                                          'salePrice', P1.sale_price,
                                          'desc', P1.description
         ) ORDER BY P1.id DESC
      ) FROM (
         SELECT DISTINCT DP.product_id
         FROM distinct_products DP
         WHERE DP.category_id = C.id
      ) sub
      JOIN product P1 ON sub.product_id = P1.id),
      '[]'
   ) AS products,
   COALESCE(
      (SELECT JSON_AGG(
         JSON_BUILD_OBJECT(
            'id', P2.id,
                                          'name', P2.name,
                                          'price', P2.price,
                                          'isSale', P2.is_sale,
                                          'salePrice', P2.sale_price,
                                          'desc', P2.description
         ) ORDER BY P2.id DESC
      ) FROM (
         SELECT DISTINCT DOP.product_id
         FROM distinct_other_products DOP
         WHERE DOP.category_id = C.id
      ) sub
      JOIN product P2 ON sub.product_id = P2.id),
      '[]'
   ) AS other
FROM category C
GROUP BY C.id, C.name;
```

```
CREATE OR REPLACE VIEW category_with_included_products AS
WITH distinct_products AS (
   SELECT DISTINCT CP.category_id, CP.product_id
   FROM category_product CP
),
distinct_other_products AS (
   SELECT DISTINCT C.id AS category_id, P.id AS product_id
   FROM category C
   CROSS JOIN product P
   WHERE NOT EXISTS (
      SELECT 1
      FROM category_product CP2
      WHERE CP2.category_id = C.id
      AND CP2.product_id = P.id
   )
)
SELECT
   C.id AS id,
   C.name AS name,
   COALESCE(
      (SELECT JSON_AGG(
         JSON_BUILD_OBJECT(
            'id', P1.id,
                                          'name', P1.name,
                                          'price', P1.price,
                                          'isSale', P1.is_sale,
                                          'salePrice', P1.sale_price,
                                          'desc', P1.description,
            'weight', P1.weight,
            'image', P1.img
         ) ORDER BY P1.id DESC
      ) FROM (
         SELECT DISTINCT DP.product_id
         FROM distinct_products DP
         WHERE DP.category_id = C.id
      ) sub
      JOIN product P1 ON sub.product_id = P1.id),
      '[]'
   ) AS products
FROM category C
GROUP BY C.id, C.name;

CREATE OR REPLACE VIEW orders_with_cart AS
SELECT
   o.*,
   COALESCE(
      json_agg(
         json_build_object(
            'name', p.name,
            'img', p.img,
            'count', pio.count
         )
      ) FILTER (WHERE pio.id IS NOT NULL),
      '[]'::json
   ) AS cart
FROM orders o
LEFT JOIN product_in_order pio ON o.id = pio.order_id
LEFT JOIN product p ON pio.product_id = p.id
GROUP BY o.id;
```

К.Р. 21КИ06125 – 01 91 01

Изм	Лист	№ документа	Подпись	Дата
Разраб.		Валявкин М.А.		
Провер.		Глебов С.А.		
Т.контр				
Н.контр				
Утвердил		Глебов С.А.		

Структура базы данных

Литера	Масса	Масштаб
У		

Лист 1 Листов 1

КФ МГТУ им. Баумана
ИУК4-62Б

Подп. и дата

Инв. № дубль

Взам. инв №

Подп. и дата

Инв. № подл