BEGIN;

CREATE TABLE company

(

company\_id integer PRIMARY KEY,

company\_name varchar(50) NOT NULL,

company\_email varchar(30) NOT NULL UNIQUE,

company\_address text NOT NULL,

company\_phone numeric(11) NOT NULL UNIQUE,

company\_city varchar(80) NOT NULL

);

CREATE TABLE IF NOT EXISTS emp\_pos

(

pos\_id integer NOT NULL PRIMARY KEY,

pos\_name varchar(50) NOT NULL

);

CREATE TABLE IF NOT EXISTS product

(

product\_id serial NOT NULL PRIMARY KEY,

product\_name varchar(40) NOT NULL,

product\_desc varchar(500) NOT NULL ,

product\_price money NOT NULL

);

CREATE TABLE priority\_task

(

priority integer PRIMARY KEY,

priority\_name varchar(30) NOT NULL

);

CREATE TABLE IF NOT EXISTS client

(

client\_id integer PRIMARY KEY,

client\_name varchar(50) NOT NULL,

client\_phone numeric(11) NOT NULL UNIQUE,

client\_email varchar(30) NOT NULL UNIQUE,

client\_postal\_address text NOT NULL,

client\_city varchar(80) NOT NULL,

company\_id integer NOT NULL REFERENCES company(company\_id)

);

CREATE TABLE IF NOT EXISTS contract

(

contract\_id serial PRIMARY KEY,

contract\_description text,

start\_date date NOT NULL,

finish\_date date NOT NULL,

total\_price money NOT NULL,

del\_address varchar(100) NOT NULL,

client\_id integer NOT NULL REFERENCES client(client\_id),

product\_id serial NOT NULL REFERENCES product(product\_id)

);

CREATE TABLE IF NOT EXISTS staff

(

staff\_id serial PRIMARY KEY,

staff\_name varchar(80) NOT NULL,

staff\_email varchar(40) NOT NULL UNIQUE,

staff\_phone numeric(11) NOT NULL UNIQUE,

staff\_login varchar(30) NOT NULL UNIQUE,

staff\_password text NOT NULL,

position\_id integer NOT NULL REFERENCES emp\_pos(pos\_id)

);

CREATE TABLE IF NOT EXISTS task

(

task\_id serial PRIMARY KEY,

start\_date date,

finish\_date date,

task\_description text NOT NULL,

task\_status BOOLEAN NOT NULL DEFAULT FALSE,

contract\_id integer REFERENCES contract(contract\_id),

author\_id integer REFERENCES staff(staff\_id),

executor\_id integer REFERENCES staff(staff\_id),

client\_id integer NOT NULL REFERENCES client(client\_id),

priority integer NOT NULL REFERENCES priority\_task(priority)

);

INSERT INTO company (company\_id, company\_name, company\_email, company\_address,

company\_phone, company\_city)

VALUES

(13,'MusicSoul','musicsoul@gmail.com','ул. Мечты, д.1', 89990070813, 'Москва');

INSERT INTO client (client\_id, client\_name, client\_phone, client\_email,

client\_postal\_address, client\_city, company\_id)

VALUES

(1,'Чон Юнги',89535448909,'jeonsuga@mail.ru','ул. Милых котят, д.1','Дэгу',13),

(2, 'Соен',89167244090,'g\_idlequeen@mail.ru','ул. Звёзд, д.98', 'Сеул',13),

(3, 'Оливер Сайкс',89216213248,'daddyissues@outlook.com','ул. Змеиначинаютпеть, д.20','Сидней',13);

INSERT INTO emp\_pos (pos\_id, pos\_name)

VALUES

(1, 'master'),

(2, 'worker');

INSERT INTO product (product\_name, product\_desc, product\_price)

VALUES

('microphone','the best one', 899900),

('micsher','almost the best one', 163820),

('acoustic system','slaaaay', 738738);

INSERT INTO priority\_task (priority, priority\_name)

VALUES

(1, 'низкий'),

(2, 'средний'),

(3, 'высокий');

INSERT INTO contract (contract\_description, start\_date, finish\_date, total\_price, del\_address, client\_id, product\_id)

VALUES

('Подарок на свадьбу', '2022-12-16', '2023-12-16', 163820, 'Дэгу, ул. Милых котят, д.1', 1, 2),

('Подарок на новоселье', '2022-08-31', '2023-08-31', 899900, 'Сеул, ул. Звёзд, д.98', 2, 1),

('Подарок на развод', '2021-10-25', '2022-4-25', 738738, 'Лондон, ул. Змеиначинаютпеть, д.20',3, 3);

INSERT INTO staff (staff\_name, staff\_email, staff\_phone, staff\_login, staff\_password, position\_id)

VALUES

('Дмитрий', 'pu$$yboy@mail.ru', 88005553535, worker\_1', MD5('airplanept2') , 2),

('Станислав', 'stasinator3000@gmail.com', 80030003000, 'master\_1', MD5('durakonline3000') , 1),

('Джони', 'sexofnct@gmail.com', 81271234523, 'master\_2', MD5('wayvmaster') , 1),

('Олег', 'oleglsp@mail.ru', 89210130414, 'worker\_2', MD5('monetka') , 2),

('Мирон', 'versusking@gmail.com', 89356742567, 'worker\_3', MD5('bezumiye') , 2);

INSERT INTO task (start\_date, finish\_date, task\_description, task\_status, contract\_id, author\_id, executor\_id, client\_id, priority)

VALUES

('2022-12-10', '2022-12-20', 'уточнить детали сделки с клиентом', FALSE, 1, 2, 4, 1, 2),

('2021-11-20', '2021-12-10', 'связаться с логистическим узлом', TRUE, 3, 3, 5, 3, 1),

('2022-10-15', '2023-01-18', 'сборка оборудования', FALSE, 2, 2, 1, 2, 3);

DROP ROLE IF EXISTS worker;

DROP ROLE IF EXISTS master;

CREATE ROLE worker;

CREATE ROLE master;

ALTER TABLE task ENABLE ROW LEVEL SECURITY;

GRANT SELECT ON ALL TABLES IN SCHEMA public

TO worker, master;

GRANT CONNECT ON DATABASE musicsoul\_db TO worker, master;

GRANT INSERT ON client, contract, company, task

TO master;

GRANT DELETE ON task

TO master, worker;

GRANT UPDATE (start\_date, finish\_date, task\_description, task\_status, contract\_id, author\_id, executor\_id, client\_id, priority) ON task

TO master, worker;

GRANT UPDATE (task\_status, finish\_date, priority)

ON task

TO master, worker;

GRANT UPDATE (executor\_id)

ON task

TO master;

CREATE POLICY insert\_task ON task

FOR INSERT

TO master

WITH CHECK (true);

CREATE POLICY update\_by\_author ON task AS PERMISSIVE

FOR UPDATE

TO master, worker

USING (true)

WITH CHECK (((( SELECT staff.staff\_login AS login FROM staff

WHERE (task.author\_id = staff.staff\_id)))::text = CURRENT\_USER));

CREATE POLICY update\_by\_executor ON task AS PERMISSIVE

FOR UPDATE

TO master, worker

USING (true)

WITH CHECK (((( SELECT staff.staff\_login AS login FROM staff

WHERE (task.executor\_id = staff.staff\_id)))::text = CURRENT\_USER));

CREATE POLICY see\_own\_tasks ON task

FOR SELECT

TO worker, master

USING (

(

SELECT staff\_login FROM staff

WHERE (staff\_id = author\_id)

) = CURRENT\_USER

OR

(

SELECT staff\_login FROM staff

WHERE (staff\_id = executor\_id)

) = CURRENT\_USER

);

CREATE PROCEDURE new\_task(

IN fin\_date date,

IN task\_desc text,

IN status boolean,

IN cont\_id integer,

IN author integer,

IN executor integer,

IN client integer,

IN priority\_num integer)

LANGUAGE 'sql'

AS $BODY$

INSERT INTO task(finish\_date, task\_description, task\_status,

contract\_id, author\_id, executor\_id, client\_id, priority)

VALUES (fin\_date, task\_desc, status,

cont\_id, author, executor, client, priority\_num);

$BODY$;

ALTER PROCEDURE public.new\_task(date, text, boolean, integer, integer, integer, integer, integer)

OWNER TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_task(date, text, boolean, integer, integer, integer, integer, integer) TO master;

GRANT EXECUTE ON PROCEDURE public.new\_task(date, text, boolean, integer, integer, integer, integer, integer) TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_task(date, text, boolean, integer, integer, integer, integer, integer) TO PUBLIC;

-- функция автоматической генерации даты создания при внесении ее в таблицу

CREATE FUNCTION gen\_start\_date() RETURNS TRIGGER

AS $$

BEGIN

UPDATE task SET start\_date = CURRENT\_DATE WHERE start\_date IS NULL;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_start\_date

AFTER INSERT ON task

FOR EACH ROW

EXECUTE FUNCTION gen\_start\_date();

GRANT EXECUTE ON FUNCTION gen\_start\_date() TO master;

DROP PROCEDURE create\_user(varchar, varchar, numeric, varchar, bytea, integer);

CREATE EXTENSION IF NOT EXISTS pgcrypto;

CREATE PROCEDURE create\_user(staff\_name varchar(80), staff\_email varchar(40),

staff\_phone numeric(11), staff\_login varchar(30), staff\_password text,

position\_id integer) AS $$

DECLARE exec\_role TEXT;

BEGIN

IF (SELECT COUNT(\*) FROM pg\_roles WHERE rolname = staff\_login) THEN

RAISE EXCEPTION 'Такой пользователь уже есть.';

ELSE

INSERT INTO staff(staff\_name,staff\_email,staff\_phone,

staff\_login, staff\_password, position\_id)

VALUES(staff\_name, staff\_email, staff\_phone, staff\_login,

MD5(staff\_password), position\_id);

EXECUTE format('CREATE ROLE %I WITH LOGIN PASSWORD %L', staff\_login, staff\_password);

exec\_role := (SELECT pos\_name FROM emp\_pos WHERE

pos\_id = position\_id);

EXECUTE FORMAT('GRANT %I TO %I', exec\_role, staff\_login);

--EXECUTE FORMAT('SET ROLE %I', employee\_log);

EXECUTE FORMAT('GRANT CONNECT ON DATABASE musicsoul\_db TO %I', staff\_login);

END IF;

END;

$$ LANGUAGE plpgsql;

CALL create\_user('Irina', 'algorithms@mail.ru', 89345672347, 'paralelky', 'idiotblytb', 1);

-- функция автоматич. генерации даты завершения задания

CREATE FUNCTION gen\_finish\_date() RETURNS TRIGGER

AS $$

BEGIN

UPDATE task SET finish\_date = CURRENT\_DATE WHERE (finish\_date >= CURRENT\_DATE) AND (task\_status = TRUE);

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger\_finish\_date

AFTER UPDATE OF task\_status ON task

FOR EACH ROW

EXECUTE FUNCTION gen\_finish\_date();

GRANT EXECUTE ON FUNCTION gen\_finish\_date() TO master,worker;

CREATE OR REPLACE FUNCTION delete\_task\_12() RETURNS trigger AS $$

BEGIN

DELETE FROM task

WHERE (task\_status = true) AND AGE(CURRENT\_DATE, finish\_date) > INTERVAL '12 months';

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER delete\_task\_12\_trigger

BEFORE INSERT OR UPDATE ON task

FOR EACH STATEMENT

EXECUTE FUNCTION delete\_task\_12();

GRANT EXECUTE ON FUNCTION delete\_task\_12 TO master,worker;

-- поиск клиентов по их атрибутам

CREATE INDEX client\_name\_ind ON client(client\_name);

CREATE INDEX client\_city\_ind ON client(client\_city);

CREATE INDEX client\_email\_ind ON client(client\_email);

--поиск сотрудника по логину

CREATE INDEX staff\_login ON staff(staff\_login);

-- поиск по автору/исполнителю

CREATE INDEX executor\_task\_ind ON task(executor\_id);

CREATE INDEX author\_task\_ind ON task(author\_id);

-- Функция выгрузки данных в .csv формат

CREATE OR REPLACE PROCEDURE export\_data\_csv(staff\_num smallint,

start\_task date, end\_task date, path\_file text)

AS $$

DECLARE

save\_values text; -- команда сохранения подсчитанных данных

all\_tasks int; -- все задания сотрудника in interval

completed\_tasks int; -- задания, выполненные в срок

proccess\_task int; -- незавершенные и просроченные задания

BEGIN

all\_tasks := (SELECT COUNT(\*) FROM task

WHERE (executor\_id = staff\_id)

AND (start\_date >= start\_task)

AND (finish\_date <= end\_task));

completed\_tasks := (SELECT COUNT(\*) FROM task

WHERE (executor\_id = staff\_num)

AND (task\_status = true)

AND (start\_date >= start\_task)

AND (finish\_date <= end\_task));

proccess\_task := (SELECT COUNT(\*) FROM task

WHERE (executor\_id = staff\_num)

AND (task\_status = false)

AND (start\_date >= start\_task)

AND (finish\_date <= end\_task));

save\_values := format('COPY (SELECT %L AS all\_tasks,

%L AS completed\_tasks,

%L AS proccess\_task)

TO %L CSV HEADER;',

all\_tasks, completed\_tasks,

proccess\_task, path\_file

);

EXECUTE save\_values;

END $$

LANGUAGE plpgsql;

GRANT EXECUTE ON PROCEDURE export\_data\_csv TO master;

CREATE OR REPLACE PROCEDURE export\_report\_json(directory text) AS $$

BEGIN

EXECUTE format('COPY (SELECT json\_agg(row\_to\_json(ts))

FROM (SELECT \* FROM task) AS ts) TO %L', directory);

END;

$$ LANGUAGE plpgsql;

GRANT EXECUTE ON PROCEDURE export\_report\_json TO master;

CREATE PROCEDURE dismissal(id\_worker integer, id\_manager integer)

LANGUAGE plpgsql

AS $$

BEGIN

UPDATE task SET executor\_id = id\_manager WHERE ((executor\_id = id\_worker) AND (task\_status = FALSE));

IF ((SELECT position\_id FROM staff WHERE staff\_id = id\_worker) = 2) THEN

DELETE FROM staff WHERE staff\_id = id\_worker;

COMMIT;

ELSE

ROLLBACK;

END IF;

END;

$$;

DROP PROCEDURE dismissal(integer, integer);

GRANT EXECUTE ON PROCEDURE public.dismissal(integer, integer) TO master;

GRANT EXECUTE ON PROCEDURE public.dismissal(integer, integer) TO postgres;

CALL dismissal(10,3)

ALTER TABLE client ALTER COLUMN client\_phone TYPE character varying(15);

ALTER TABLE client ALTER COLUMN client\_phone TYPE integer USING (client\_phone::integer);

CREATE OR REPLACE PROCEDURE public.new\_client(

IN client\_id integer,

IN client\_name character varying(50),

IN client\_phone character varying(15),

IN client\_email character varying(30),

IN client\_postal\_address text,

IN client\_city character varying(30),

IN company\_id integer)

LANGUAGE 'sql'

AS $BODY$

INSERT INTO client(client\_id,client\_name,client\_phone,client\_email,client\_postal\_address,client\_city,company\_id)

VALUES (client\_id,client\_name,client\_phone,client\_email,client\_postal\_address,client\_city,company\_id);

$BODY$;

ALTER PROCEDURE public.new\_client(integer, character varying(50), character varying(15), character varying(30), text, character varying(30), integer)

OWNER TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_client(integer, character varying(50), character varying(15), character varying(30), text, character varying(30), integer) TO master;

GRANT EXECUTE ON PROCEDURE public.new\_client(integer, character varying(50), character varying(15), character varying(30), text, character varying(30), integer) TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_client(integer, character varying(50), character varying(15), character varying(30), text, character varying(30), integer) TO PUBLIC;

CREATE OR REPLACE PROCEDURE public.new\_contract(

IN contract\_id integer,

IN contract\_description text,

IN start\_date date,

IN finish\_date date,

IN total\_price money,

IN del\_address character varying(100),

IN client\_id integer,

IN product\_id integer)

LANGUAGE 'sql'

AS $BODY$

INSERT INTO contract(contract\_id, contract\_description, start\_date, finish\_date, total\_price, del\_address, client\_id, product\_id)

VALUES (contract\_id, contract\_description, start\_date, finish\_date, total\_price, del\_address, client\_id, product\_id);

$BODY$;

ALTER PROCEDURE public.new\_contract(integer, text, date, date, money, character varying(100), integer, integer)

OWNER TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_contract(integer, text, date, date, money, character varying(100), integer, integer) TO master;

GRANT EXECUTE ON PROCEDURE public.new\_contract(integer, text, date, date, money, character varying(100), integer, integer) TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_contract(integer, text, date, date, money, character varying(100), integer, integer) TO PUBLIC;

CREATE OR REPLACE PROCEDURE public.new\_product(

IN product\_id integer,

IN product\_name character varying(40),

IN product\_desc character varying(500),

IN product\_price money)

LANGUAGE 'sql'

AS $BODY$

INSERT INTO product(product\_id, product\_name, product\_desc, product\_price)

VALUES (product\_id, product\_name, product\_desc, product\_price);

$BODY$;

ALTER PROCEDURE public.new\_product(integer, character varying(40), character varying(500), money)

OWNER TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_product(integer, character varying(40), character varying(500), money) TO master;

GRANT EXECUTE ON PROCEDURE public.new\_product(integer, character varying(40), character varying(500), money) TO worker;

GRANT EXECUTE ON PROCEDURE public.new\_product(integer, character varying(40), character varying(500), money) TO postgres;

GRANT EXECUTE ON PROCEDURE public.new\_product(integer, character varying(40), character varying(500), money) TO PUBLIC;

CREATE VIEW not\_done\_task AS

SELECT ta.task\_id,

ta.start\_date,

ta.finish\_date,

ta.task\_description,

ta.executor\_id,

st.staff\_name,

st.staff\_email

FROM task ta, staff st

WHERE ((ta.executor\_id = st.staff\_id) AND (ta.task\_status = FALSE));

END;