Московский авиационный институт

(национальный исследовательский университет) «МАИ»

Институт №3 — «Системы управления, информатика и электроэнергетика»

Кафедра 316 — «Системное моделирование и автоматизированное проектирование»

Финальная работа №1

«Проектирование базы данных»

Выполнил:

студент группы 3О-218М-19

Пономарев Роман

Принял:

Моргунов Евгений Павлович

Москва 2020

Описание предметной области

В качестве предметной области используется вариант фитнес-клуба. Клуб реализует продажу абонементов клиентам, продажу товаров и услуг. В клубе проводятся разные виды тренировок, предоставляются различные виды услуг. Тренировки проводятся по расписанию, под руководством определенного тренера. Абонементы рассчитаны на определенное количество посещений, которые необходимо учитывать. Абонементы продаются клиентам, минимальной информацией о которых, клубу также необходимо обладать. Необходимо разработать базу данных для такого фитнес-клуба, так как это позволит сотрудникам оптимальным способом хранить и систематизировать информацию, а также легко и удобно извлекать нужную информацию.

Требования к базе данных

Требования к данным:

Есть различные виды тренировок, услуг и товаров. Расписание содержит информацию сотруднике, проводящем тренировку, виде тренировке, виде услуги, номер записи. Продажа товара должна обладать информацией о количестве проданного товара, номере продажи, и клиенте, которому был продан товар. В клубе содержится информация о фамилии и имени клиентов. Для продажи абонемента нужно указывать количество оставшихся посещений, информацию о клиенте, которому продан абонемент. В каждом абонементе хранится информация о виде тренировки, цене, количестве посещений, видах услуг. Также в базе данных содержится информация о сотрудниках клуба: фамилия, имя, оклад.

Требования к транзакциям:

- Различные выборки услуг, видов тренировок, видов абонементов
- Обновление/удаление сведений о клиентах клуба
- Ввод сведений о новых абонементах, услугах, тренировках клуба

Концептуальная модель

Концептуальная модель содержит описание понятий и объектов предметной области.

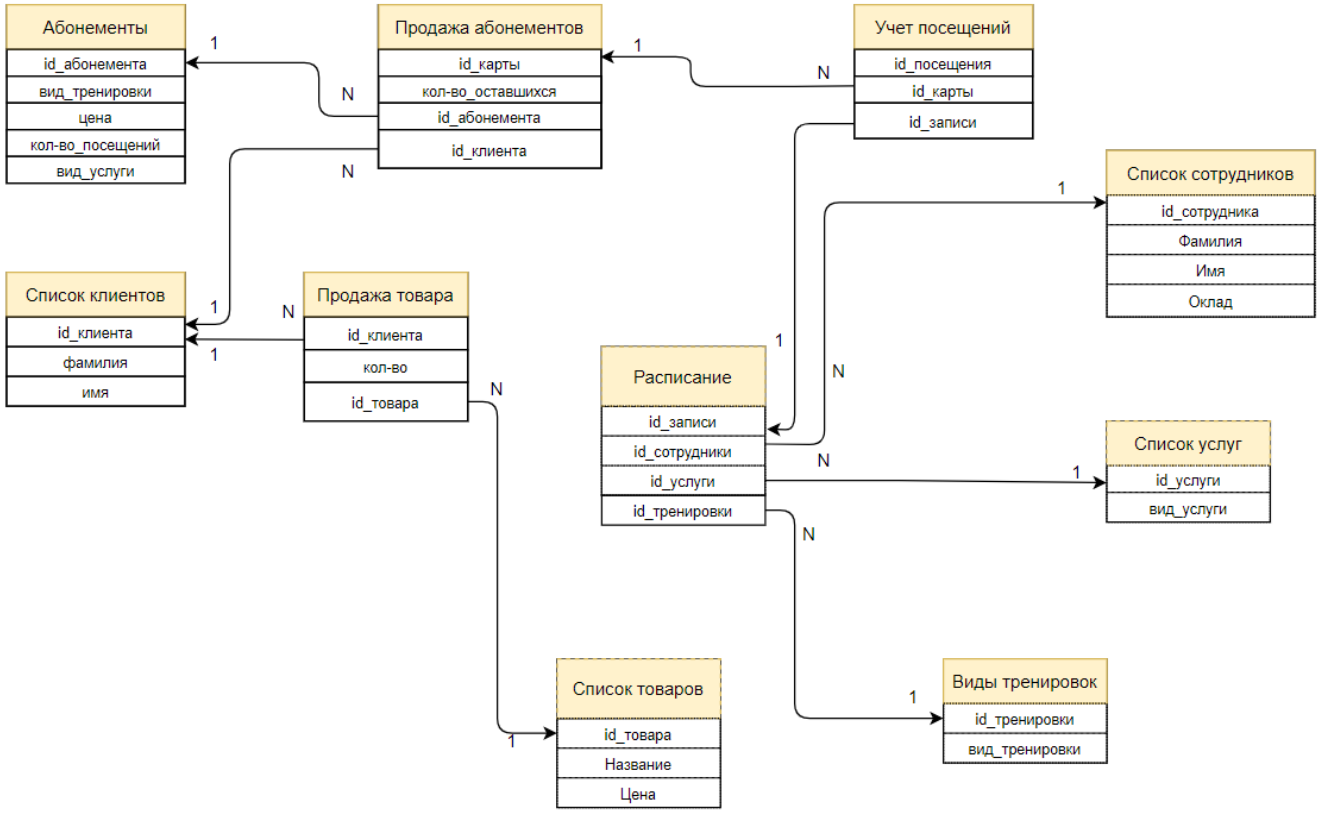

Рисунок 1 – Концептуальная схема

Логическая модель

Схема БД состоит из 10 таблиц:

1. tickets(ticket_id, training_type, price, visits_quantity, service_type)
2. client_list(client_id, surname, name)
3. goods_sale(client_id, counts, sale_id, good_id)
4. tickets_sales(card_id, remaining_number, ticket_id, client_id)
5. goods_list(good_id, name, price)
6. schedule(record_id, employee_id, service_id, training_id)
7. training_types(training_id, training_type)
8. services_list(service_id, service_type)
9. employees_list(employee_id, surname, name, salary)
10. visit_tracking(visit_id, card_id, record_id)

Таблица tickets представляет абонементы: номер абонемента, вид спорта,

Таблица tickets представляет абонементы: номер абонемента(уникальный), вид спорта, цену, количество посещений, вид услуги.

Таблица client_list представляет список клиентов: номер клиента(уникальный), фамилия, имя.

Таблица goods_sale содержит информацию о покупках каждого клиента: количество товара, номер товара, номер покупки(уникальный), номер товара.

Таблица goods_list содержит названия всех продаваемых в клубе товаров.

Таблица tickets_sales содержит информацию о проданных абонементах: номер абонемента, id клиента, которому продан абонемент, номер карты(уникальный), позволяющий составлять расписание, количество оставшихся занятий.

Таблица visits_tracking содержит информацию посещений конкретной карты: id посещений(уникальный), номер записи.

Таблица schedule объединяет информацию о каждой тренировке: номер записи(уникальный), номер работника, номер услуги, номер вида спорта.

Таблица employees_list содержит информацию о каждом тренере клуба: номер работника(уникальный), фамилия, имя, зарплата.

Таблица services_list содержит информацию о видах услуг клуба: номер услуги(уникальный) и вид услуги.

Таблица training_types содержит информацию о видах спорта в клубе: номер вида спорта(уникальный) и вид спорта.

Логическая модель описывает базу данных более подробно. Указываются поля таблиц, а также ключи. На ней изображены нормальные формы таблиц базы данных. Всего нормальных форм 7:

Первая нормальная форма: отношение, в котором на пересечении каждой строки и каждого столбца содержится одно и только одно значение.

Вторая нормальная форма: отношение, которое находится в 1 НФ, и каждый атрибут которого, не входящий в состав первичного ключа, характеризуется полной функциональной зависимостью от этого первичного ключа.

Третья нормальная форма: отношение, которое находится в первой и во второй нормальных формах и не имеет атрибутов, не входящих в первичный ключ, которые находились бы в транзитивной функциональной зависимости от этого первичного ключа.

Нормальная форма Бойса-Кодда: отношение должно находиться в 3 НФ, и каждый его детерминант является потенциальным ключом.

Все таблицы базы данных находятся в 3НФ, потому что все они имеют первичные ключи и в них отсутствует транзитивная функциональная зависимость от первичного ключа.

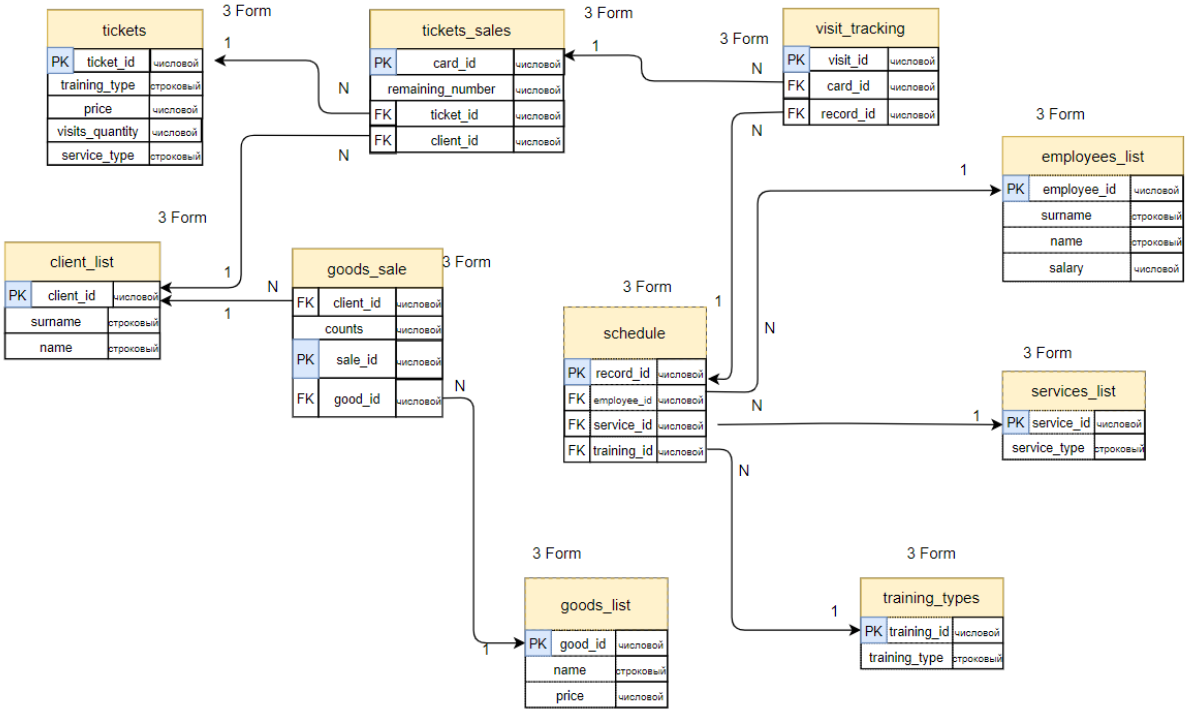

Рисунок 2 - Логическая схема

Физическая модель

Физическая модель отображает полностью все связи таблиц, названия таблиц, названия полей, типы данных, ключи и прочие атрибуты так, как они должны быть указаны при реализации базы данных.

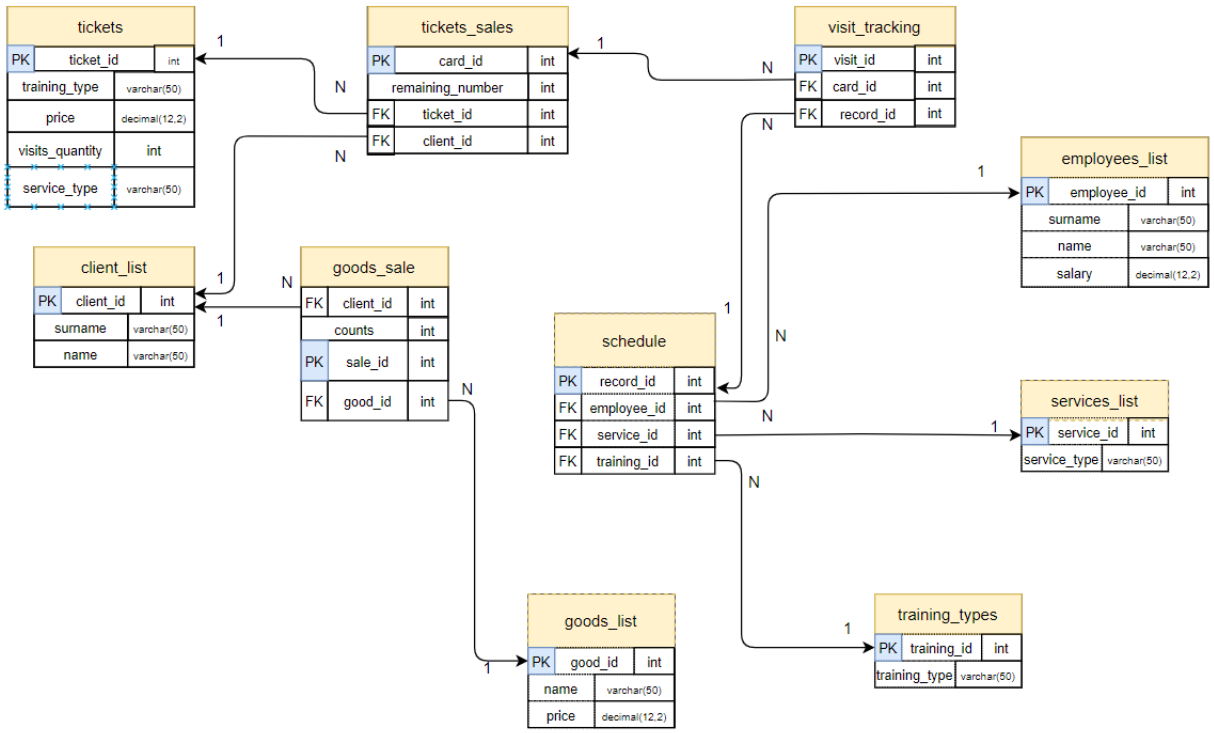

Рисунок 3 - Физическая схема

Запросы

1) Демонстрация всех функций в БД;

\x;

\df;

```
fitness_club1=# \x
Расширенный вывод включён.
fitness_club1=# \df
Список функций
-[ RECORD 1 ]----------+----------------
Схема                  | public1
Имя                    | count_good
Тип данных результата  | bigint
Типы данных аргументов | good_type text
Тип                    | функ.
-[ RECORD 2 ]----------+----------------
Схема                  | public1
Имя                    | ndfl
Тип данных результата  | trigger
Типы данных аргументов |
Тип                    | функ.
-[ RECORD 3 ]----------+----------------
Схема                  | public1
Имя                    | nds
Тип данных результата  | trigger
Типы данных аргументов |
Тип                    | функ.
-[ RECORD 4 ]----------+----------------
Схема                  | public1
Имя                    | sport_coach
Тип данных результата  | text
Типы данных аргументов | sport_type text
Тип                    | функ.
```

Рисунок 4 - функции в БД

2) Демонстрация работы функций;

SELECT count_good('protein');

SELECT sport_coach('MMA');

```
fitness_club1=# SELECT count_good('protein');
-[ RECORD 1 ]-
count_good | 2

fitness_club1=# SELECT sport_coach('MMA');
-[ RECORD 1 ]----------
sport_coach | Kovalenko
```

Рисунок 5 - Результат работы функций

3) Демонстрация работы триггера;

INSERT INTO employees_list VALUES (6, 'Kosenko', 'Ksenia', 90000);

SELECT * FROM employees_list;

```
fitness_club1=# INSERT INTO employees_list values(6, 'Kosenko', 'Ksenia', 90000);
INSERT 0 1
fitness_club1=# SELECT * FROM employees_list;
-[ RECORD 1 ]-----------
employee_id | 1
surname     | Smirnov
name        | Sergey
salary      | 70000.00
-[ RECORD 2 ]-----------
employee_id | 2
surname     | Kovalenko
name        | Kirill
salary      | 90000.00
-[ RECORD 3 ]-----------
employee_id | 3
surname     | Cherchesov
name        | Stanislav
salary      | 40000.00
-[ RECORD 4 ]-----------
employee_id | 4
surname     | Ignatenko
name        | Igor
salary      | 70000.00
-[ RECORD 5 ]-----------
employee_id | 5
surname     | Kozlov
name        | Anton
salary      | 35000.00
-[ RECORD 6 ]-----------
employee_id | 6
surname     | Kosenko
name        | Ksenia
salary      | 101700.00
```

Рисунок 6 - Результат работы триггера1

INSERT INTO goods_list VALUES (4, 'chocolate', 120);

SELECT * FROM goods_list;

```
fitness_club1=# INSERT INTO goods_list values(4, 'chocolate', 120);
INSERT 0 1
fitness_club1=# SELECT * FROM goods_list;
-[ RECORD 1 ]------
good_id | 3
name    | barbell
price   | 7000.00
-[ RECORD 2 ]------
good_id | 2
name    | protein
price   | 2000.00
-[ RECORD 3 ]------
good_id | 1
name    | towel
price   | 500.00
-[ RECORD 4 ]------
good_id | 4
name    | chocolate
price   | 144.00
```

Рисунок 7 - Результат работы триггера2

4) Вывести фамилии тренеров, которые проводят занятия по виду спорта MMA:

```
SELECT DISTINCT coaches.surname as coach FROM employees_list as coaches
        JOIN schedule ON coaches.employee_id = schedule.employee_id
        WHERE schedule.training_id = (SELECT DISTINCT types.training_id FROM training_types as types
        JOIN schedule ON schedule.training_id = types.training_id
        WHERE types.training_type = 'MMA');

--------   coach   |
-------------------
-------- Kovalenko
--------(1 строка)
```

Рисунок 8 – Подзапрос

5) Вывести клиентов и продукты, которые они купили:

```
fitness_club1=# WITH goods AS
(
SELECT sales.client_id, list.name FROM goods_sale as sales
JOIN goods_list as list ON sales.good_id = list.good_id
)
SELECT client_list.surname as surname, client_list.name, goods.name as product FROM client_list
JOIN goods ON client_list.client_id = goods.client_id;
 surname |  name  | product
---------+--------+---------
 Ivanov  | Petr   | protein
 Ivanov  | Petr   | towel
 Sidorov | Pavel  | barbell
 Ivanov  | Victor | towel
 Musaev  | Semen  | barbell
(5 строк)
```

Рисунок 9 - CTE

6) Демонстрация работы оконной функции.

Вывести отсортированный список клиентов относительно цен их абонементов, указать вид спорта и цену абонемента:

```
fitness_club1=# WITH ticket as
(
SELECT ticket_id, training_type, price,
row_number() OVER (ORDER BY sum(price) DESC) as rating
FROM tickets
GROUP BY ticket_id ,training_type, price)
SELECT clients.surname as surname, clients.name as name, ticket.training_type, ticket.price, ticket.rating FROM
client_list as clients JOIN tickets_sales as sales ON clients.client_id = sales.client_id
JOIN ticket ON sales.ticket_id = ticket.ticket_id
fitness_club1-# ORDER BY rating;
 surname |  name  | training_type |  price   | rating
---------+--------+---------------+----------+--------
 Ivanov  | Victor | MMA           | 30000.00 |      1
 Ivanov  | Petr   | MMA           | 15000.00 |      2
 Beglov  | Ivan   | swimming      | 15000.00 |      3
 Petrov  | Ivan   | Boxing        | 12000.00 |      4
 Sidorov | Pavel  | Fitness       | 10000.00 |      5
 Musaev  | Semen  | Fitness       |  9000.00 |      6
(6 строк)
```

Рисунок 10 - Оконная функция

Защита работы

Запрос, который вычисляет сумму денег, которую принесли в клуб клиенты, тренирующиеся у каждого из тренеров, т.е. сколько денег заработал для клуба каждый тренер, и проранжировать тренеров по этому показателю.

```
fitness_club1=# WITH coaches as (SELECT employees_list.surname as coach, tickets_sales.card_id as card_id FROM employees_list
INNER JOIN schedule on employees_list.employee_id = schedule.employee_id
INNER JOIN visit_tracking ON schedule.record_id = visit_tracking.record_id
INNER JOIN tickets_sales ON visit_tracking.card_id = tickets_sales.card_id)
, clients as (SELECT client_list.surname as client_name, price, tickets_sales.card_id as card_id FROM client_list
INNER JOIN tickets_sales on client_list.client_id = tickets_sales.client_id
INNER JOIN tickets ON tickets_sales.ticket_id = tickets.ticket_id)
SELECT SUM(clients.price) as money, coaches.coach FROM clients
INNER JOIN coaches ON coaches.card_id = clients.card_id
GROUP BY coaches.coach
ORDER BY money DESC;
  money   |   coach
----------+------------
 30000.00 | Kozlov
 25000.00 | Ignatenko
 24000.00 | Kovalenko
  9000.00 | Cherchesov
(4 строки)
```

Вывод

В ходе выполнения практической работы были закреплены на практике механизмы работы СУБД PostgreSQL. Были спроектированы физическая, логическая и концептуальная модели, определяющие сущность базы данных в выбранной предметной области. По моделям была реализована база данных, в которую были занесены данные, соответствующие выбранной тематике. Были добавлены вспомогательные функции. Также были добавлены триггеры, обеспечивающие автоматические изменения некоторых полей в определенных таблицах БД.

Приложение А

Листинг запросов

```
-- Для установки функций в БД
-- psql -d fitness_club1 -f club_pg_script.sql -U postgres
SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

DROP TABLE IF EXISTS tickets CASCADE;
DROP TABLE IF EXISTS client_list CASCADE;
DROP TABLE IF EXISTS goods_sale CASCADE;
DROP TABLE IF EXISTS tickets_sales CASCADE;
DROP SCHEMA IF EXISTS goods_list CASCADE;
DROP SCHEMA IF EXISTS schedule CASCADE;
DROP SCHEMA IF EXISTS visit_tracking CASCADE;
DROP SCHEMA IF EXISTS training_types CASCADE;
DROP SCHEMA IF EXISTS services_list CASCADE;
DROP SCHEMA IF EXISTS employees_list CASCADE;
DROP DATABASE IF EXISTS  fitness_club1;

CREATE DATABASE fitness_club1;

\connect fitness_club1;
```

```
SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;
SET row_security = off;

--
-- Name: public; Type: SCHEMA; Schema: -; Owner: -
--
CREATE SCHEMA public1;



--
-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: -
--

COMMENT ON SCHEMA public IS 'standard public schema';

SET search_path = public1;

--
-- Name: now(); Type: FUNCTION; Schema: public; Owner: -
--

CREATE OR REPLACE FUNCTION now() RETURNS timestamp with time zone
    LANGUAGE sql IMMUTABLE COST 0.00999999978
    AS $$SELECT '2016-10-13 17:00:00'::TIMESTAMP AT TIME ZONE 'Europe/Moscow';$$;
```

```
--
-- Name: FUNCTION now(); Type: COMMENT; Schema: public; Owner: -
--

COMMENT ON FUNCTION now() IS 'Момент времени, относительно которого сформированы данные';

SET default_tablespace = '';

SET default_with_oids = false;

--Create Tables:


CREATE TABLE client_list (
	client_id int NOT NULL,
	surname varchar(50) NOT NULL,
	name varchar(50) NOT NULL);
CREATE TABLE

CREATE TABLE tickets (
	ticket_id int NOT NULL,
	training_type varchar(50) NOT NULL,
	price decimal(12,2) NOT NULL,
	visits_quantity int NOT NULL,
	service_type varchar(50) NOT NULL);
CREATE TABLE

CREATE TABLE goods_sale (
```

```
        client_id int NOT NULL,
        counts int NOT NULL,
        sale_id int NOT NULL,
        good_id int NOT NULL);
CREATE TABLE


CREATE TABLE tickets_sales (
        card_id int NOT NULL,
        remaining_number int NOT NULL,
        ticket_id int NOT NULL,
        client_id int NOT NULL);
CREATE TABLE


CREATE TABLE goods_list (
        good_id int NOT NULL,
        name varchar(50) NOT NULL,
        price decimal(12,2) NOT NULL);
CREATE TABLE


CREATE TABLE schedule (
        record_id int NOT NULL,
        employee_id int NOT NULL,
        service_id int NOT NULL,
        training_id int NOT NULL);
CREATE TABLE
```

```
CREATE TABLE visit_tracking (
        visit_id int NOT NULL,
        card_id int NOT NULL,
        record_id int NOT NULL);
CREATE TABLE
                        ^
CREATE TABLE training_types (
        training_id int NOT NULL,
        training_type varchar(50) NOT NULL
);
CREATE TABLE


CREATE TABLE services_list (
        service_id int NOT NULL,
        service_type varchar(50) NOT NULL
);
CREATE TABLE

CREATE TABLE employees_list (
        employee_id int NOT NULL,
        surname varchar(50) NOT NULL,
        name varchar(50) NOT NULL,
        salary decimal(12,2) NOT NULL);
CREATE TABLE

--Adding Constraints

--Primary Keys:
```

```
ALTER TABLE tickets ADD
	CONSTRAINT PK_tickets PRIMARY KEY
	(
	ticket_id
	)
	;

ALTER TABLE client_list ADD
	CONSTRAINT PK_client_list PRIMARY KEY
	(
	client_id
	);

ALTER TABLE tickets_sales ADD
	CONSTRAINT PK_tickets_sales PRIMARY KEY
	(
	card_id
	);

ALTER TABLE goods_sale ADD
	CONSTRAINT PK_goods_sale PRIMARY KEY
	(
	sale_id
	);

ALTER TABLE goods_list ADD
	CONSTRAINT PK_goods_list PRIMARY KEY
	(
	good_id
	);
```

```
ALTER TABLE schedule ADD
	CONSTRAINT PK_schedule PRIMARY KEY
	(
	record_id
	);

ALTER TABLE visit_tracking ADD
	CONSTRAINT PK_visit_tracking PRIMARY KEY
	(
	visit_id
	);

ALTER TABLE training_types ADD
	CONSTRAINT PK_training_types PRIMARY KEY
	(
	training_id
	);

ALTER TABLE employees_list ADD
	CONSTRAINT PK_employees_list PRIMARY KEY
	(
	employee_id
	);

ALTER TABLE services_list ADD
	CONSTRAINT PK_services_list PRIMARY KEY
	(
	service_id
	);
```

```
--FOREIGN KEYS:

ALTER TABLE goods_sale ADD
	CONSTRAINT FK_goods_sale__client_list FOREIGN KEY
	(
	client_id
	) REFERENCES client_list (
	client_id
	)
	ON DELETE CASCADE ON UPDATE CASCADE
	;

ALTER TABLE goods_sale ADD
	CONSTRAINT FK_goods_sale__goods_list FOREIGN KEY
	(
	good_id
	) REFERENCES goods_list (
	good_id
	)
	ON DELETE CASCADE ON UPDATE CASCADE
	;

ALTER TABLE tickets_sales ADD
	CONSTRAINT FK_tickets_sales__tickets FOREIGN KEY
	(
	ticket_id
	) REFERENCES tickets (
	ticket_id
	)
```

```
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE tickets_sales ADD
    CONSTRAINT FK_tickets_sales__client_list FOREIGN KEY
    (
    client_id
    ) REFERENCES client_list (
    client_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE schedule ADD
    CONSTRAINT FK_schedule__employees_list FOREIGN KEY
    (
    employee_id
    ) REFERENCES employees_list (
    employee_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE schedule ADD
    CONSTRAINT FK_schedule__services_list FOREIGN KEY
    (
    service_id
    ) REFERENCES services_list (
    service_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE schedule ADD
```

```
    CONSTRAINT FK_schedule__training_types FOREIGN KEY
    (
    training_id
    ) REFERENCES training_types (
    training_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE visit_tracking ADD
    CONSTRAINT FK_visit_tracking__tickets_sales FOREIGN KEY
    (
    card_id
    ) REFERENCES tickets_sales (
    card_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE visit_tracking ADD
    CONSTRAINT FK_visit_tracking__schedule FOREIGN KEY
    (
    record_id
    ) REFERENCES schedule (
    record_id
    )
    ON DELETE CASCADE ON UPDATE CASCADE;


---INSERT VALUES INTO TABLES:----------------------------------------------------
-------------
```

---------tickets----------

---SELECT * FROM tickets;

--- training_type | price | visits_quantity | service_type | ticket_id

--------------+-------+----------------+-------------+-----------

---(0 строк)

insert into tickets values('MMA', 15000, 20, 'group_training', 1990);

insert into tickets values('Boxing', 12000, 20, 'group_training', 1895);

insert into tickets values('Fitness', 9000, 15, 'gym_training', 1896);

insert into tickets values('Fitness', 10000, 17, 'gym_training', 1996);

insert into tickets values('swimming', 15000, 17, 'swimming_pool', 1936);

insert into tickets values('MMA', 30000, 20, 'individual_training', 1946);

-------SELECT * FROM tickets;

----training_type | price | visits_quantity | service_type | ticket_id

--------------+----------+----------------+--------------------+-----------

---MMA | 15000.00 | 20 | group_training | 1990

---Boxing | 12000.00 | 20 | group_training | 1895

---Fitness | 9000.00 | 15 | gym_training | 1896

---Fitness | 10000.00 | 17 | gym_training | 1996

---swimming | 15000.00 | 17 | swimming_pool | 1936

---MMA | 30000.00 | 20 | individual_training | 1946

----(6 строк)

---------client_list----------

-------SELECT * FROM client_list;

-------client_id | surname | name

-----------+---------+------

------(0 строк)

^

```
insert into client_list values(135, 'Ivanov', 'Petr');
insert into client_list values(178, 'Petrov', 'Ivan');
insert into client_list values(58, 'Musaev', 'Semen');
insert into client_list values(98, 'Sidorov', 'Pavel');
insert into client_list values(149, 'Beglov', 'Ivan');
insert into client_list values(243, 'Ivanov', 'Victor');
```

------- SELECT * FROM client_list;

--------client_id | surname | name

```
-----------+---------+--------
-------       135 | Ivanov  | Petr
-------       178 | Petrov  | Ivan
-------        58 | Musaev  | Semen
-------        98 | Sidorov | Pavel
-------       149 | Beglov  | Ivan
-------       243 | Ivanov  | Victor
```

------- (6 строк)

----------tickets_sales----------

```
insert into tickets_sales values(34, 7, 1990, 135);
insert into tickets_sales values(56, 15, 1895, 178);
insert into tickets_sales values(15, 15, 1896, 58);
insert into tickets_sales values(98, 13, 1996, 98);
insert into tickets_sales values(41, 17, 1936, 149);
insert into tickets_sales values(66, 19, 1946, 243);
```

----------SELECT * FROM tickets_sales;

----------card_id | remaining_number | ticket_id | client_id

------------------+-----------------+-----------+-----------

---------- 34 | 7 | 1990 | 135

---------- 56 | 15 | 1895 | 178

---------- 15 | 15 | 1896 | 58

---------- 98 | 13 | 1996 | 98

---------- 41 | 17 | 1936 | 149

---------- 66 | 19 | 1946 | 243

----------(6 строк)

------------------goods_list----------

insert into goods_list values(3, 'barbell', 7000);

insert into goods_list values(2, 'protein', 2000);

insert into goods_list values(1, 'towel', 500);

-------SELECT * FROM goods_list;

------- good_id | name | price

---------------+---------+---------

------- 3 | barbell | 7000.00

------- 2 | protein | 2000.00

------- 1 | towel | 500.00

-------(3 строки)

------------------goods_sale----------

insert into goods_sale values(135, 2, 1, 2);

insert into goods_sale values(135, 1, 2, 1);

insert into goods_sale values(98, 1, 3, 3);

insert into goods_sale values(243, 3, 4, 1);

insert into goods_sale values(58, 2, 5, 3);

-------SELECT * FROM goods_sale;

------- client_id | counts | sale_id | good_id

-----------------+--------+---------+---------

------- 135 | 2 | 1 | 2

------- 135 | 1 | 2 | 1

------- 98 | 1 | 3 | 3

------- 243 | 3 | 4 | 1

------- 58 | 2 | 5 | 3

-------(5 строк)

----------training_types----------

insert into training_types values(1, 'MMA');

insert into training_types values(2, 'Boxing');

insert into training_types values(3, 'Fitness');

insert into training_types values(4, 'swimming');

--------SELECT * FROM training_types;

-------- training_id | training_type

--------------------+--------------

-------- 1 | MMA

-------- 2 | Boxing

-------- 3 | Fitness

-------- 4 | swimming

--------(4 строки)

----------services_list----------

```
insert into services_list values(1, 'group_training');
insert into services_list values(2, 'gym_training');
insert into services_list values(3, 'swimming_pool');
insert into services_list values(4, 'individual_training');
----------SELECT * FROM services_list;
---------- service_id |    service_type
-----------------------+--------------------
----------          1 | group_training
----------          2 | gym_training
----------          3 | swimming_pool
----------          4 | individual_training
----------(4 строки)


------------------employees_list---------------

insert into employees_list values(1, 'Smirnov', 'Sergey', 70000);
insert into employees_list values(2, 'Kovalenko', 'Kirill', 90000);
insert into employees_list values(3, 'Cherchesov', 'Stanislav', 40000);
insert into employees_list values(4, 'Ignatenko', 'Igor', 70000);
insert into employees_list values(5, 'Kozlov', 'Anton', 35000);
---------SELECT * FROM employees_list;
--------- employee_id |  surname   |   name    |  salary
---------------------+------------+-----------+----------
---------           1 | Smirnov    | Sergey    | 70000.00
---------           2 | Kovalenko  | Kirill    | 90000.00
---------           3 | Cherchesov | Stanislav | 40000.00
---------           4 | Ignatenko  | Igor      | 70000.00
---------           5 | Kozlov     | Anton     | 35000.00
```

---------(5 строк)

----------schedule----------

insert into schedule values(134, 2, 1, 1);
insert into schedule values(201, 3, 2, 3);
insert into schedule values(156, 1, 1, 2);
insert into schedule values(172, 4, 1, 2);
insert into schedule values(57, 5, 3, 4);
insert into schedule values(68, 2, 4, 1);

---------SELECT * FROM schedule;
--------- record_id | employee_id | service_id | training_id
-------------------+-------------+------------+-------------
--------- 134 | 2 | 1 | 1
--------- 201 | 3 | 2 | 3
--------- 156 | 1 | 1 | 2
--------- 57 | 5 | 3 | 4
--------- 172 | 4 | 1 | 2
--------- 68 | 2 | 4 | 1
---------(6 строк)

----------visit_tracking----------

insert into visit_tracking values(4, 34, 134);
insert into visit_tracking values(2, 15, 134);

```
insert into visit_tracking values(7, 98, 172);
insert into visit_tracking values(15, 41, 172);
insert into visit_tracking values(1, 66, 57);
insert into visit_tracking values(76, 15, 201);

--------SELECT * FROM visit_tracking;
-------- visit_id | card_id | record_id
------------------+---------+-----------
--------        4 |      34 |       134
--------        2 |      15 |       134
--------        7 |      98 |       172
--------       15 |      41 |       172
--------        1 |      66 |        57
--------       76 |      15 |       201
--------(6 строк)


-----------functions-------------------

---------Function that returns quantity of goods sold----------
CREATE OR REPLACE FUNCTION count_good( good_type text)
 RETURNS bigint AS $$
 SELECT SUM(counts) FROM goods_sale as g_sale
 JOIN goods_list as g_list ON g_sale.good_id = g_list.good_id
 WHERE g_list.name = good_type;
$$ LANGUAGE sql;

------SELECT count_good('protein');
------ count_good
------------------
```

```
------          2
------(1 строка)


--------Function returns coach's surname that training 'sport_type'
CREATE OR REPLACE FUNCTION sport_coach(sport_type text)
RETURNS text AS $$
SELECT DISTINCT coaches.surname as coach FROM employees_list as coaches
JOIN schedule ON coaches.employee_id = schedule.employee_id
WHERE schedule.training_id = (SELECT DISTINCT types.training_id FROM
training_types as types
JOIN schedule ON schedule.training_id = types.training_id
WHERE types.training_type = sport_type);
$$ LANGUAGE sql;

-----SELECT sport_coach('Boxing');
----- sport_coach
-----------------
----- Ignatenko
-----(1 строка)




----------------triggers-----------------------------------------------------
CREATE OR REPLACE FUNCTION ndfl()
RETURNS trigger AS
$$
BEGIN
 NEW.salary = NEW.salary*1.13;
```

```
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;

DROP TRIGGER IF EXISTS ndfl ON employees_list;

CREATE TRIGGER ndfl BEFORE INSERT OR UPDATE ON employees_list
 FOR EACH ROW EXECUTE PROCEDURE ndfl();



CREATE OR REPLACE FUNCTION nds()
RETURNS trigger AS
$$
BEGIN
 NEW.price = NEW.price * 1.2;
 RETURN NEW;
END;
$$ LANGUAGE plpgsql;



CREATE TRIGGER nds BEFORE INSERT OR UPDATE ON goods_list
 FOR EACH ROW EXECUTE PROCEDURE nds();
```