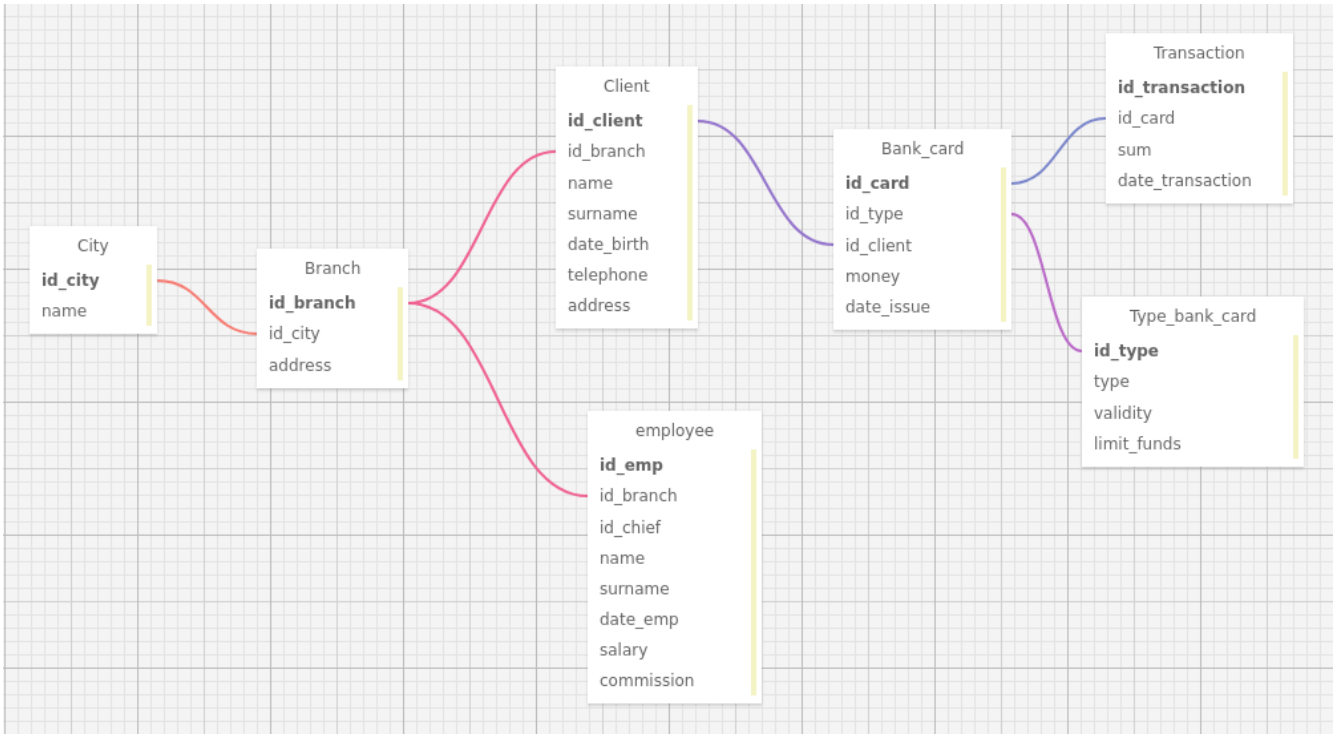
Crearea bazei de date in SGBD Oracle.

Subiectul:
«Banca comerciala»

1. Crearea schemei bazei de date:



Schema bazaei de date "Banca comerciala"

2. Operatorii Oracle/PLSQL pentru crearea tabelului:

```
1  create table City
2  (
3      id_city number(2) not null,
4      name varchar2(15) not null,
5      constraint pk_city primary key (id_city)
6  );
```

Results | Explain | Describe | Saved SQL | History

```
Table created.

0.04 seconds
```

exemplu crearei tabelului "City" (Orasul)

```
create table City
(
    id_city number(2) not null,
    name varchar2(15) not null,
    constraint pk_city primary key (id_city)
);
```

`create table` — operatorul pentru crearea tabelului.

`number(2)` – tip de date numerice, care stocheaza un numar din doua cifre.

`varchar(15)` – tip de date simbolice cu lungime variabile, care in cazul meu nu poate depasi 15 simboli

`constraint (primary key)` - interzice mai multe campuri să aibă aceeași valoare în aceeași coloană sau combinație de coloane și interzice valorile să fie nule.

`primary key` - este un singur câmp sau o combinație de câmpuri care definește unicitatea unei înregistrări. Câmpurile care fac parte din cheia primară nu pot conține NULL. Un tabel poate avea o singură cheie primară.

```
create table Branch
(
    id_branch number(2) not null,
    id_city number(2) not null,
    address varchar2(20) not null,
    constraint pk_branch primary key (id_branch),
    constraint fk_city foreign key (id_city)
        references City(id_city)
);
```

`foreign key` - înseamnă că valorile dintr-un tabel trebuie să apară și în celălalt tabel. Tabelul de referință se numește tabel părinte, iar tabelul cu „foreign key” se numește tabel copil. Cheia străină din tabelul copil se referă de obicei la cheia primară din tabelul părinte.

```
-- selectarea tutulor datelor din tabelul Client
select *
from Client;

-- selectarea datelor unde atributul nume se începe cu litera ‘N’ din tabelul Client
select *
from Client
where name like 'N%'

-- selectarea prenumelui, numelui din tabelul Client, selectarea tipului de card bancar și
banilor clientilor, care au pe card mai mult de 10 mii
-- conexiunea tabelelor Client și Bank_card este făcută de operatorul join, care sunt unite
prin id clientului.
select
    client.name,
    client.surname,
    card.id_type type,
    card.money
from client
    join Bank_card card
    on card.id_client = client.id_client
where card.money > 10000;

-- selectarea prenumelui și numelui clientului banii caraia pe card depășesc 10 mii, făcută
prin conditia unde id clientului este legal ca id clientului în tabelul Bank_card, unde banii
pe card depășesc summa data.

select
    client.name,
    client.surname
from Client
where id_client in(
    select id_client
    from Bank_card
    where money > 10000
);
```

```sql
-- Selectarea lucratorilor în ordinea de la cel mai mare salariu la cel mai mic.
-- se utilizeaza operatorul order by cu optiunea desc.
select *
from Employee
order by salary desc;

-- selectarea adresei filialilor bancii, și  numărul lucratroilor în fie care filial.
-- se utilizeaza group by pentru crearea grupelor unde addressele sunt identice și functia
count pentru numerarea lucratorilor.
select
    br.address branch,
    count(id_emp) employees
from Branch br
    join Employee emp
    on emp.id_branch = br.id_branch
group by br.address;

-- selectam prenumele, numele și anii fiecarui client, pentru asta extragem din data curenta
anul și scădem din el anul extras din data nasterei clientului. Utilizam functia extract
pentru extragerea anului, functia sysdate pentru a primi data curenta.
select
    name,
    surname,
    extract(year from sysdate) - extract(year from date_birth) years
from client


-- selectam prenumele, numele, și suma pe care a cheltuit client în tot timpul utilizarii
cardului bancar. Utilizam join de doua ori pentru a crea legătura dintre tabelele client,
Bank_card și transaction, grupam prenumele și numele utilizatorului pentru a exclude repetarea
acestor date și pentru a rezuma banii cheltuiți fiecarului client.
select
    cl.name,
    cl.surname,
    sum(sum)
from client cl
    join Bank_card card
    on card.id_client = cl.id_client
    join transaction tr
    on tr.id_card = card.id_card
group by cl.name, cl.surname

-- selectam prenume, nume și banii cheltuiți fiecarui client într-un câmp. Utilizam functia
upper pentru a converti toate caracterele în majuscule. Functia concat penru a uni datele
într-un sir. Functia rpad pentru introduce sirul de dreapta la dimensiunea specificata cu
caractere propuse. Functia length pentru a primi marimia sirului.
select
    upper(
        concat(
            rpad( cl.name || ' ' || cl.surname || ' ',  60 - length(cl.name || cl.surname),
':'), ' ' || sum(sum)
        )
    )
from client cl
    join Bank_card card
    on card.id_client = cl.id_client
    join transaction tr
    on tr.id_card = card.id_card
group by cl.name, cl.surname
```

```
-- Crearea tabelului „Client”
create table Client
(
    id_client number(3) not null,
    id_branch number(2) not null,
    name varchar2(15) not null,
    surname varchar2(15) not null,
    date_birth date not null,                   -- date - tipul de date.
    telephone varchar2(12) not null,
    address varchar2(30),
    constraint pk_client primary key (id_client),
    constraint fk_branch foreign key (id_branch)
        references Branch(id_branch)
);
-- cerearea tabelului „Employee”
create table Employee
(
    id_emp number(3) not null,
    id_branch number(2) not null,
    id_chief number(3),
    name varchar2(15) not null,
    surname varchar2(15) not null,
    date_emp date not null,
    salary number(5) not null,
    commision number(4,2),
    constraint pk_emp primary key (id_emp),
    constraint fk_branch_emp foreign key (id_branch)
        references Branch(id_branch)
);
-- crearea tabelului „Type_bank_card”
create table Type_bank_card
(
    id_type number(2) not null,
    type varchar2(15) not null,
    validity number(1) not null,
    limit_funds number(6),
    constraint pk_type primary key (id_type)
);
-- crearea tabelului „Bank_card”
create table Bank_card
(
    id_card number(4) not null,
    id_type number(2) not null,
    id_client number(3) not null,
    money number(9,2) not null,
    date_issue date not null,
    constraint pk_card primary key (id_card),
    constraint fk_type foreign key (id_type)
        references Type_bank_card(id_type),
    constraint fk_client foreign key (id_client)
        references Client(id_client)
);
-- crearea tabelului „Transactie”
create table Transaction
(
    id_transaction number(5) not null,
    id_card number(4) not null,
    sum number(7,2) not null,
    date_transaction date not null,
    constraint pk_transaction primary key (id_transaction),
    constraint fk_card foreign key (id_card)
```

```
        references Bank_card(id_card)
);
-- Exemple de introducerea datelor în tabele.
insert into City
values (1, 'Chisinau');

insert into Branch
values (1, 1, 'sos.Hancesti 55/4');

insert into Employee
values (1, 1, 5, 'Igor', 'Fresca', to_date('01.12.2004', 'dd.mm.yyyy'), 5400, 2.5);

insert into Client
values (1, 1, 'Nicita', 'Soldanescu', to_date('05.11.1977', 'dd.mm.yyyy'), '+37368643227',
'str.Mircea Batrin 16/1');

insert into Type_bank_card
values (1, 'bisnes', 6, 990000);

insert into Bank_card
values (1, 2, 1, 567.15, to_date('01.02.2019', 'dd.mm.yyyy'));

insert into Transaction
values (1, 1, 230.00, to_date('09.03.2019', 'dd.mm.yyyy'));
```