SQL

Tipična oblika SQL poizvedbe je v sledeči obliki. Oznaka A predstavlja atribute, r so relacije, P pa pogojni stavki (predikati)

select $A_1, A_2, ..., A_n$
from $r_1, r_2, ..., r_m$
where P

SELECT

Uporabimo ga kadar želimo dobiti izbrane atribute iz baze, kot rezultat poizvedbe. Rezultat poizvedbe so relacije.

S spodnjo poizvedbo želimo izbrati vse atribute `branch-name` iz tabele `loan`

select *branch-name*
from *loan*

Če bi namesto atributa `branch-name` uporabili `*` bi pomenilo, da želimo izbrati vse atribute iz tabele `loan`.

Če želimo odstraniti duplikate, bi uporabili sledečo poizvedbo:

select *branch-name*
from *loan*

V stavku select lahko nad atributi izvajajmo aritmetične izraze (seštevanje, odštevanje, množenje, deljenje)

select *loan-number, branch-name, amount* * 100
from *loan*

Z where stavkom postavljamo pogoje, kakšni morajo biti atributi, ki jih želimo izbrati. Primerjave se sestavljajo z logičnimi povezavami `and`, `or`, `not`

select *loan-number*
from *loan*
where *branch-name* = 'Perryridge' **and** *amount* > 1200

Še en primer stavka where:

select *loan-number*
from *loan*
where *amount* **between** 90000 **and** 100000

Stavek FROM predstavlja kartezični atribut različnih atributov. Z naslednjo poizvedbo želimo poiskati kartezični produkt tabel `borrower` in `loan`.

select *
from *borrower, loan*

```
select customer-name, borrower.loan-number, amount
from borrower, loan
where   borrower.loan-number = loan.loan-number  and
        branch-name = 'Perryridge'
```

Z stavkom `as` preimenujemo atribut v tabeli z našim imenom:

```
select customer-name, borrower.loan-number as loan-id, amount
from borrower, loan
where borrower.loan-number = loan.loan-number
```

Preimenovanje imen tabele:

```
select distinct T.branch-name
from branch as T, branch as S
where T.assets > S.assets and S.branch-city = 'Brooklyn'
```

Operacije nad nizi: `%` → poišči katerikoli podniz, `_` → ujemanje kateregakoli znaka, `||` → združevanje nizov

```
select customer-name
from customer
where customer-street like '%Main%'
```

`order by` razvrščanje rezultatov po atributih (ASC – naraščajoče, DESC -- padajoče)

```
select distinct customer-name
from   borrower, loan
where borrower loan-number - loan.loan-number and
        branch-name = 'Perryridge'
order by customer-name
```

Operacije nad množicami (unija, presek, negacija)

```
(select customer-name from depositor)
union
(select customer-name from borrower)

(select customer-name from depositor)
intersect
(select customer-name from borrower)

(select customer-name from depositor)
except
(select customer-name from borrower)
```

Agregacijske funkcije vključujejo stavke `avg`, `min`, `max`, `sum`, `count`

```
select avg (balance)
from account
where branch-name = 'Perryridge'
```

```
select count (*)
from customer
```

```
select count (distinct customer-name)
from depositor
```

Poišči število strank, po atributu `branch-name`. S stavkom `group by` naredimo kategorijo štetja v tem primeru:

```
select branch-name, count (distinct customer-name)
from depositor, account
where depositor.account-number = account.account-number
group by branch-name
```

```
select branch-name, avg (balance)
from account
group by branch-name
having avg (balance) > 1200
```

Gnezdenje poizvedb

Select-from-where.

Največkrat se uporabljajo, ko želimo pogledati vsebnost, primerjavo določene množice z drugo množico.

```
select distinct customer-name
from borrower
where customer-name in (select customer-name
                        from depositor)
```

```
select distinct customer-name
from borrower
where customer-name not in (select customer-name
                            from depositor)
```

Primerjava množic:

```
select distinct  T.branch-name
from branch as T, branch as S
where  T.assets > S.assets and
       S.branch-city = 'Brooklyn'
```

Enaka poizvedba kot zgornja je lahko zapisna tudi kot:

```
select branch-name
from branch
where assets > some
      (select assets
       from branch
       where branch-city = 'Brooklyn')
```

Spodnja poizvedba poišče vse račune po vseh `brancih`, ki so locirani v Brooklyinu.

```
select distinct S.customer-name
from depositor as S
where not exists (
    (select branch-name
    from branch
    where branch-city = 'Brooklyn')
    except
    (select R.branch-name
    from depositor as T, account as R
    where T.account-number = R.account-number and
          S.customer-name = T.customer-name))
```

Najdi vse stranki, ki imajo največ en račun v Perrydge branchu

```
select T.customer-name
from depositor as T
where unique (
    select R.customer-name
    from account, depositor as R
    where T.customer-name = R.customer-name and
          R.account-number = account.account-number and
          account.branch-name = 'Perryridge')
```

Poiščemo povprečno vrednost na računu, v tistih računih, ki imajo vsaj 1200 dolarjev povrečje

```
select branch-name, avg-balance
from (select branch-name, avg (balance)
      from account
      group by branch-name)
      as result (branch-name, avg-balance)
where avg-balance > 1200
```

DELETE → brisanje zapisov iz baze

Brisanje vseh zapisov iz baze, ki imajo branch-name

```
delete from account
where branch-name = 'Perryridge'

delete from account
where balance < (select avg (balance)
                 from account)
```

INSERT → vstavljanje zapisov v bazo

```
insert into account (branch-name, balance, account-number)
      values ('Perryridge', 1200, 'A-9732')
```

UPDATE → posodabljanje vrednosti v bazi

```
update account
set balance = balance * 1.06
where balance > 10000

update account
set balance = balance * 1.05
where balance ≤ 10000
```

DDL → data definition language

S tem jezikom ustvarimo tabele in njihove ralacije ter podamo omejitve posameznih atributov v podatkovni bazi.

```
create table branch
   (branch-name char(15),
   branch-city   char(30)
   assets        integer,
   primary key (branch-name),
   check (assets >= 0))
```

Primarni ključ tabele zagotavlja, da ključ ni NULL in se lahko tudi avtomatsko povečuje.

Spreminjanje tabel in ralacij:

alter table *r* **add** *A D*

alter table *r* **drop** *A*

Primer definicije podatkovne zbirke s tabelami, njihovimi relacijami in omejitvami:

```
create table customer
    (customer-name    char(20),
     customer-street  char(30),
     customer-city    char(30),
     primary key (customer-name))

create table branch
    (branch-name      char(15),
     branch-city      char(30),
     assets           integer,
     primary key (branch-name),
     check (assets >= 0))

create table account
    (account-number   char(10),
     branch-name      char(15),
     balance          integer,
     primary key (account-number),
     check (balance >= 0))

create table depositor
    (customer-name    char(20),
     account-number   char(10),
     primary key (customer-name, account-number))
```

To je bil pregled poizvedbenega jezika nad podatki v relacijskih podatkovnih bazah SQL. Za dodatne primere si lahko ogledate stran: https://www.w3schools.com/sql/