Teema 6järg. Self-joini'id,alampäringud.

* self-joins - on nagu päringud mitmest tabelist, kusjuures tabelite ühendamiseks kasutatakse join-i tingimust , ainult et kahe erineva tabeli asemel on tegemist ühe tabeliga - tabel on ühendatud iseendaga (päringu tulemuses on teatud read tabelist ühendatud selle sama tabeli teatud ridadega).Virtuaalsete tabelitega on self-join-i puhul tegemist - tegelikult on üks tabel.

Self-joini on mõtet kasutada siis, kui ühest tabelist oleks vajaliku info saamiseks vaja küsida mitu korda. Näiteks: järgnevast tabelist olek vaja välja võtta kõik need, kelle palk on suurem kui Juhan Juurikal. Selleks oleks esiteks vaja kätte saada Juurika palk ja sellest palgast lähtuvalt kätte saada kõik need, kelle palk on suurem.

palgatabel

Töötaja	Palk
Juhan Juurikas	3000
Mart Mets	4000
Joonas Jõgi	2000
Andres Aas	3500
.....	

Üheks võimaluseks tulemuse saamise jaoks on kasutada alapäringut, mille tulemust kasutatakse põhipäringu tegemiseks: küsida Juurika palk tabelist ja saadud tulemust kasutada põhipäringu WHERE-klauslis

SELECT töötaja FROM palgatabel
WHERE palk > (SELECT palk FROM töötaja WHERE töötaja = "Juhan Juurikas");

Teiseks võimaluseks on kasutada self-join'i, nüüd on tegemist otsekui kahe erineva tabeliga, millest esimest saame kasutada Juurika palga leidmiseks ja teist Juurikast kõrgemapalgaliste töötajate leidmiseke, seda kõike ühe SELECT lause sees, kõik vajalikud tingimused saab nüüd esitada selle SELECT lause WHERE- klauslis.

Self-join'i puhul kasutatakse virtuaalsete tabelite eristamiseks nn. alias-nimesid.

SELECT A.töötaja
FROM palgatabel A, palgatabel B
WHERE A.palk < B.palk
AND B.töötaja = "Juhan Juurikas";

Päringu tulemus võetakse nüüd kahest virtuaalsest tabelist, mis tegelikult on ühe tabeli nö peegeldused. Kuid: need virtuaalsed tabelid on erinevad, seda määravad

WHERE-klauslis olevad tingimused selle virtuaaltabeli kohta. Nii on tabelis B (töötaja) ainult üks kirje (Juhan Juurikas). Seetõttu piisab teiseks tingimuseks "A.palk < B.palk" mis võrdlebki töötajate palku Juhan juurika palgaga.

* Alampäringud - nested queries

Mitu nö üksteise sisse pandud päringut - iga järgmine (välimine) päring kasutab alampäringust saadud andmeid WHERE klauslis.

SELECT töötaja FROM palgatabel
WHERE palk > (SELECT palk FROM töötaja WHERE töötaja = "Juhan Juurikas");
.

Veel näiteid - võetakse andmeid sellest tabelist ainult siis, kui tabelis on olemas mingile kriteeriumile vastav kirje

SELECT töötaja FROM palgatabel
WHERE EXISTS (SELECT töötaja FROM töötaja WHERE töötaja = "Juhan Juurikas" OR töötaja = "Mart mets");

valitakse töötajate nimekiri tabelist vaid siis, kui töötajate nimekirjas on Juhan Juurikas.

Alternatiivne variant sellest päringust:

SELECT töötaja FROM palgatabel
WHERE töötaja IN ("Juhan Juurikas", "Mart Mets");

Variant päringust, mis toob töötajate tabelist välja ainult need töötajad, kes on samaaegselt ka eesrindlaste tabelis.

SELECT töötaja FROM palgatabel
WHERE töötaja IN (SELECT eesrindlane FROM eesrindlased);

* rohkemat kui ühte alapäringut sisaldavad päringud - compound nested queries.

Päringud võivad sisaldada rohkem kui ühte alampäringut. Mitu alampäringut võivad paikneda samal tasemel ja olla ühendatud võtmesõnaga **AND**. Järgnevas näites valitakse töötaja tabelist ainult need töötajad, kelle palk ületab Juhan Juurika palka ja kes samal ajal on ka tööeesrindlaste tabelis.

SELECT töötaja FROM palgatabel
WHERE (palk > (SELECT palk FROM töötaja WHERE töötaja = "Juhan Juurikas")) AND (töötaja IN (SELECT eesrindlane FROM eesrindlased));

Alampäringud võivad olla ka teiste alapäringute sees (nende WHERE klauslites) st. siis mitte enam samal tasemel.

```
SELECT klient FROM klient
WHERE klient IN
   (SELECT klient FROM klient
     WHERE ostud > 100
       AND teenindus_piirkond IN
            (SELECT teenidus_piirkond  FROM teenindus_piirkond
                WHERE linn_maakod = "Tallinn")
```

klient teeninduspiirkond

Klient	ostud	teenindus_piirkond
Juhan	345	1
Mart	200	2

päringu abil k
piirkond asu

Teenindus_piirkond	linn_maakond
1	"Tallinn"

Sellest näiteks peaks olama ka näha andmebaasi normaliseerimise ja tabeliteks lahutamise varjuküljed - mida enam "laiali" paiknevad andmed tabelites , seda keerukamateks ja aeglasemateks lähevad päringud, seda tihedamini vajadus alapäringite järele.

* korreleeruvad alapäringud
üldiselt käivitub alampäring üks kord pärigu jooksul ja igale peapäringu reale vastab sama alapäringu tulemus.

Korreleeruva alapäringu korral nõuab alapäring infot peapäringu ridade kohta ja käivitub iga kord uuesti iga peapäringu rea korral. Alapäring on seotud korrelleruv peapäringuga ja ei ole temast sõltumatu.

Alampäring on korreleeriv peapäringuga siis, kui tema WHERE-klauslis on viide peapäringule.

```
SELECT töötaja FROM töötaja A
WHERE A.töötaja = (SELECT töötaja FROM eesrindlased
WHERE eesrindlane = A.töötaja);
```

Selles päringus viitab alampäringu WHERE klausel peapäringu reale (A.töötaja) - st. alampäringu tegemisel tuleb alati võrrelda peapäringu vastava reaga, seega peab alapäring käivituma just nii palju kordi , kui on peapäringu ridasid, iga kord uuesti.

Selleks , et alampäringust saaks viidata peapäringule, peab peapäringus kasutama alias nimesid tabelitele (antud juhul **töötaja - A**).

Pärigu täitmine käib siin järgnevalt - iga töötaja korral käivitub alampäring, mis pärib eesrindlase tabelist eesrindlase (nime või koodi) , mis on võrdne töötajaga töötaja tabelis (võrdlus **eesrindlane = A.töötaja**). Seega uurib alampäring, kas on antud töötajale vastavat eesrindlast eesrindlaste tabelis (kas töötaja on eesridlane). Kui on , siis osutub õigeks võrdus

A.töötaja = (SELECT töötaja FROM eesrindlased
WHERE eesrindlane = A.töötaja);
ja antud töötaja läheb päringu tulemusse. Siis võetakse ette järgmine töötaja, käivitatakse alampäring uuesti jne.