Teema 6. Päringud üldiselt, päringud SQL-keeles. SELECT lause.

- **Päringud üldiselt.Relatsioonialgebra.**

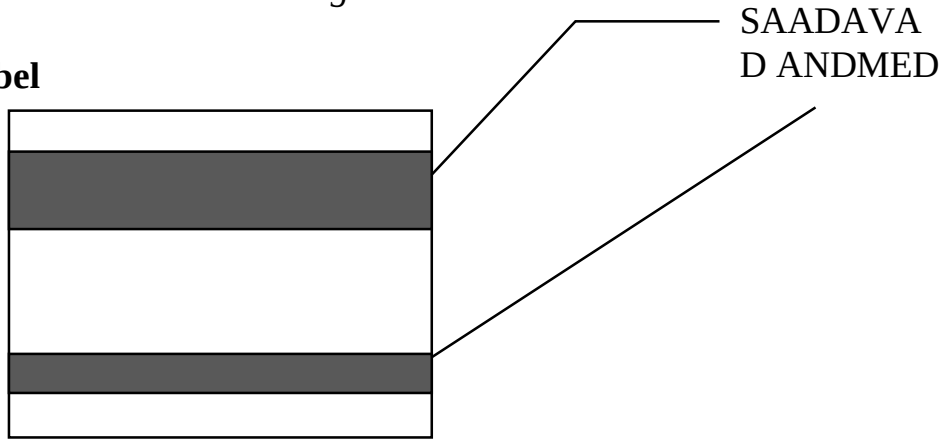
"Relatsiooniline mudel" andmebaaside jaoks. Relatsioonilist andmebaasi võib vaadelda kui hulka kahemõõtmelisi tabeleid. Põhimõisteid eelnevast - **tabelid, veerud, read, väljad (*tables,columns, rows, fields*).** Eelnevas jutus süsteemianalüüsist nimetasime samu asju järgmiselt - **infoobjekt, atribuut, kirje, väli.**

Relatsioonilisele andmebaasile rakendub matemaatika haru , mida nimetatakse *relatsioonialgebraks.* Relatsioonialgebra **objektideks** on andmebaasi **tabelid**, objektidega koos kasutatakse **relatsiooni operaatoreid.** Relatsioonioperaatorite rakendamisel tabelitele on võimalik saada andmebaasist andmeid (ja selle tulemusel luua uusi tabeleid).

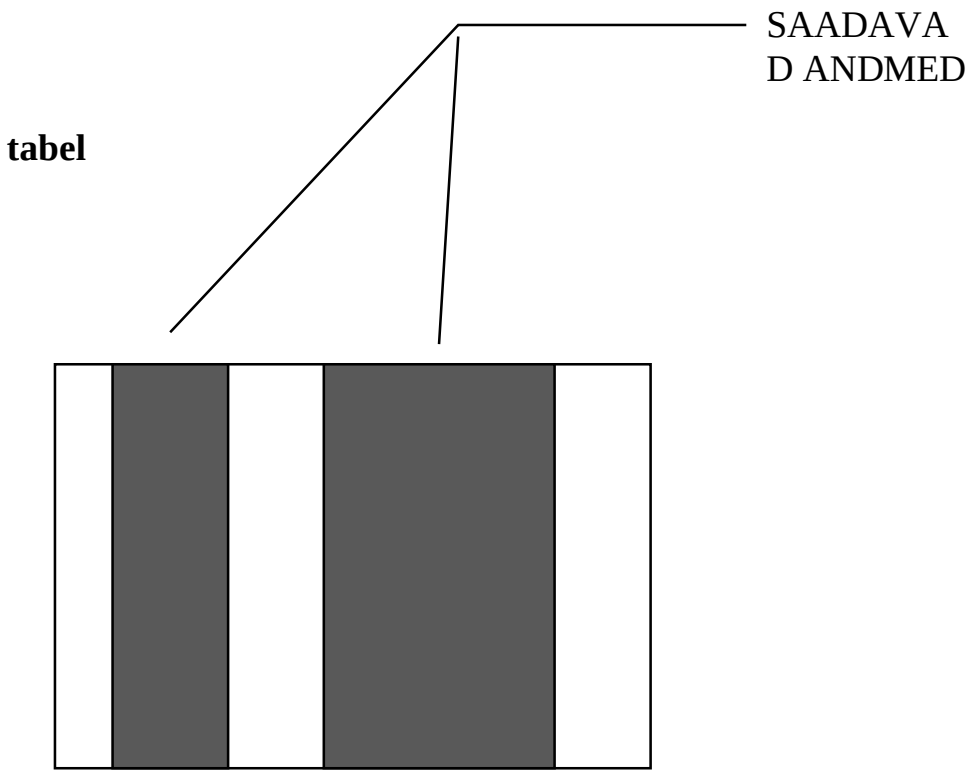
Põhimõtteliselt tähendab relatsioonialgebra mingite hulgatehete tegemist tabelite või päringute tulemustega.

Relatsioonioperaatorid:

PIIRANG (restriction) - on operatsioon, mis võtab andmeid tabelist.On võimalik küsida andmeid kogu tabelist (kõik read) või küsida ainult neid andmeid (ridu), mis vastavad mingile konkreetsele tingimusele/tingimustele. Piirangu tulemust nimetatakse teinekord *horisontaalseks alamhulgaks.*



PROJEKTSIOON (projection) - andmed teatud kindlatest tabeli veergudest (columns). Nimetatakse ka vertikaalseks alamhulgaks.



KORRUTIS (product) - kui saadud andmed on koond kahe andmehulga (andmetabeli) andmetest, mis on ühendatud.Kõik read esimesest andmehulgast on ühendatud kõigi ridadega teisest andmehulgast. Sageli võib korrutise tulemuseks olla väga suur andmehulk. Korrutisel on harva mingi mõte.

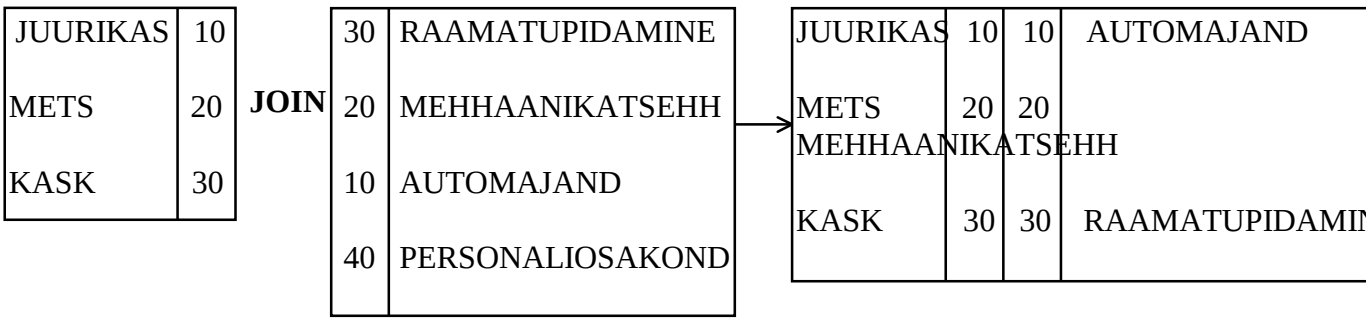
JUHAN JUURIKAS
MART METS
VIRVE KASK

AUTOJUHT
KEEVITAJA

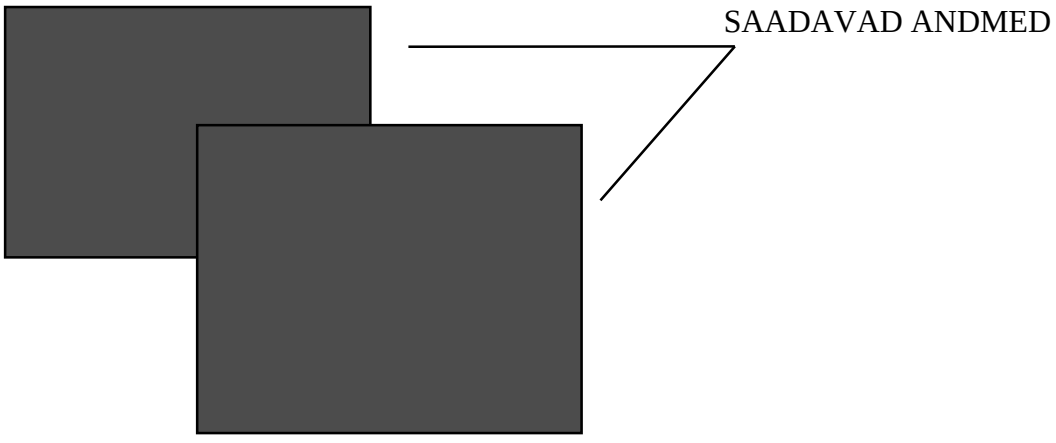
JUHAN JUURIKAS	AUTOJUHT
JUHAN JUURIKAS	KEEVITAJA
MART METS	AUTOJUHT
MART METS	KEEVITAJA
VIRVE KASK	AUTOJUHT
VIRVE KASK	KEEVITAJA

product ⟶

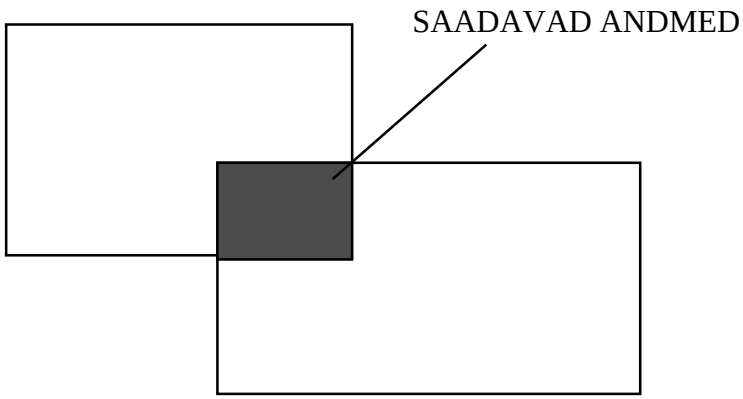
ÜHEND (join) - kui read (kirjed) kahest tabelist on ühendatud mingi kindla kriteeriumi järgi. (tuleta meelde relatsioone ja primaar- ning välisvõtmeid, *foregn key*)



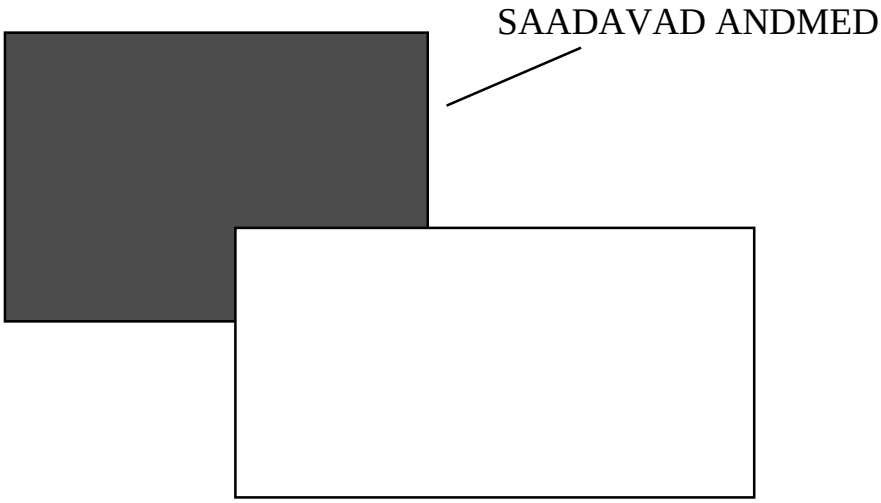
ÜHISOSA (union) - annab tulemuseks kõik read mõlemast tabelist.



LÕIGE (intersection) - annab tulemuseks read, mis on mõlemas tabelis samad.



VAHE (difference) - andmed, mis on ainult ühes tabelis kahest.



SQL-I süntaksis tähistab vahet MINUS. Näiteks: Oraaklis

```
SELECT part FROM orders_list1
MINUS
SELECT part FROM orders_list2;
```

- **Päringud. SELECT -lause**

Andmete valimiseks tabelitest ja vaadetest.

Süntaks: **SELECT [DISTINCT | ALL] {* | <val> [, <val> ...]}**
FROM <tableref> [, <tableref> ...]
[WHERE <search_condition>]
[GROUP BY col]
[HAVING <search_condition>]
[UNION <select_expr>]
[PLAN <plan_expr>]
[ORDER BY <order_list>]

- "SELECT" lause koosneb **SELECT** klauslist, mis loetleb veerud, millest andmeid valitakse, **FROM** klausel määrab tabelid ,kust andmed valitakse.
- SELECT lauses võib veel olla aritmeetilisi tehteid :
  SELECT kliendi_kood * 12 FROM klient ;
- veergudele võib anda teised nimed (aliases) mida näidatakse päringu väljundis.
  SELECT kliendi_kood kliendi_number FROM klient;
- lauses võib kasutada sidurdamist e. konkatenatsiooni - mitme välja kokkuliitmist päringus
  SELECT nimi || registr_kpv kliendi_registreerimine FROM klient;
- kui tabeli mingis väljas väärtus puudub, interpreteeritakse seda kui nullise väärtusega välja (*null value*).***Null value* ei ole sama mis null**. Kui numbrivälja väärtuseks on null, ei tähenda see, et väli on tühi.
- kui päringu tulemuseks satuvad identsed, duplikaatread, seda aga ei soovita, tuleb SELECT lausesse lisada DISTINCT klausel.

SELECT DISTINCT registr_kpv FROM klient;

- normaalselt on päringu tulemusel saadavad read suvalises järjekorras. Seda saab muuta, andes ette vastava sorteerimiseeskirja - millise veeru väärtuste järgi tuleb read sorteerida.

SELECT nimi, aadress FROM klient ORDER BY nimi;

Sorteeritekse - väiksemad numbrilised väärtused ettepoole
- varasemad kuupäevad ettepoole
- tekstiväljad tähestiku järjekorras

ORDER BY korral võib kasutada ka mitme veeru järgi korraga sorteerimist.
HAVING piirab nende ORDER BY järgi grupeeritud gruppide väljastamist, mis ei vasta päringutingimustega. Alumises näites küsitakse nende osakondade kassapidajate miinimum ja maksimumpalku, kus miinimumpalk jääb alla 1000 .

```
SELECT deptno, MIN(sal), MAX(sal)
    FROM emp
    WHERE job = 'CLERK'
    GROUP BY deptno
    HAVING MIN(sal) < 1000 ;
```

- WHERE klausel paneb peale **piirangu** (restriction) päringu tulemustele, selle klausliga määratakse, millistele tingimustele peavad päringu tulemusel saadavad read (rows) vastama. WHERE klausel selekteerib päringust välja mittesoovitavad andmed.

```
SELECT kliendi_kood, nimi FROM klient
    WHERE kliendi_kood < 3 ;
```

```
SELECT kliendi_kood, nimi FROM klient
    WHERE kliendi_kood < 3 ;
```

KLIENDI_KOOD	NIMI
1	Mati Jõgi
2	Andres Mets

```
SELECT kliendi_kood, nimi FROM klient
    WHERE nimi  =  “Mati Jõgi” ;
```

WHERE klausel nõuab kolme elementi : 1. veeru nimi
2. võrdlusoperaator
3. veeru nimi, konstant või väärtuste hulk

- WHERE klauslis in kasutatavad operaatorid BETWEEN … AND, IN(list), LIKE, IS NULL

```
SELECT kliendi_kood, nimi FROM klient
    WHERE kliendi_kood  BETWEEN 1 AND 3 ;
```

- WHERE klausli abil võib ühes päringus korraga kasutada mitut tingimust.

```
SELECT kliendi_kood, nimi, registr_kpv FROM klient
        WHERE registr_kpv < "12-MAY-1994"
        AND kliendi_kood  BETWEEN 1 AND 3 ;
```

- **Funktsioonid**

jagunevad:

- grupifunktsioonid
- ühe rea funktsioonid

Osa funktsioone annab väärtuse iga päringu (tabeli) rea kohta - i. k. *single row functions*
osa fuktsioone annab väärtuse ridade grupi kohta - i.k. *group functions*

Funktsioonidele võib argumentidena edasi anda väärtusi *f(x)*

grupifunktsioonid:

AVG(column) - keskmise veeru väärustest
SUM(column) - summa
MAX(column) - maksimaalne väärtus veerus
MIN(column) - minimaalne väärtus veerus
COUNT(column) - ridade arv veerus

Grupifunktsioonide kasutamine

SELECT SUM(kliendi_kood) FROM klient; - kogub andmeid tabeli kõikidest ridadest, kuid väljastab ainult ühe rea - kliendi koodide summa.

Ühe rea funktsioonid:

UPPER('ants')

Enamus andmebaasisüsteeme pakub lisaks ülaltoodud standardsetele funktsioonidele veel täiendavaid funktsioone. - n. trigonomeetrilised süsteemid, aja funktsioonid (SYSDATE).

- **Päringud rohkem kui ühest tabelist.**

Kui päringuga on tarvis küsida andmeid mitmest tabelist, kasutatakse **join** - meetodit (vt. eelmist punkti). Kirjed ühed tabelist on päringus ühendatud teise tabeli kirjetega nendes kirjetes asuvate ühiste väärtuste alusel.

On olemas kahte tüüpi **join** -tingimusi :

- **Equal-join :** kahest tabelist valitakse ainult need kirjed, mille vastavate veergude väärtused on võrdsed (võrduse operaator = on kasutusel.)

SELECT veerud
FROM tabelid
WHERE joini tingimus on …

SELECT nimi, osakonna_nimetus
FROM tootaja,osakond
WHERE tootaja.osakonna_nr = osakond.osak_nr;

Joini tingimuses on veergude nimetuses ära märgitud ka tabeli nimi (**tootaja.osakonna_nr**), see on vajalik vaid siis, kui veergude (atribuutide) nimed mõlemas tabelis on identsed.

- **Non-equal-join :** on juhul, kui seos kahe erinevas tabelis paiukneva kirje vahel on saavutatud teisiti, kui (=) operaatorit kasutades. Näiteks võib küside töötaja palgaastet vastavast tabelist teades töötaja palka. Palgaastmestiku tabelis on toodud vastava palgaastme jaoks miinimum- ja maksimumpalk. Päringus kasutatakse kirjete ühendamiseks BETWEEN operaatorit.

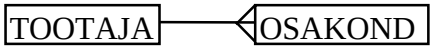
SELECT E.ENAME, E.SAL , S.GRADE
FROM EMP E, SALGRADE S
WHERE E.SAL **BETWEEN** S.LOSAL AND S.HISAL;

Siin on süntaksi lühendamiseks kasutatud nn. alias-nimesid - tabel EMP on ümber nimetatud E-ks ja SALGRADE S-ks. Selline ümbernimetamine kehtib ainult antud SQL-lause piirides.

Reegel tabelite kirjete ühendamiseks (join) päringus:
nn. joini tingimuste (WHERE-klauslite) minimaalne arv = tabelite arv - 1

- **Teisi joini meetodeid.**

  - **OUTER join :** Kui kirje mingis tabelis ei rahulda joini tingimust, jääb ta päringu tulemustest välja. Näiteks: kui küsida kõik töötajad ühest tabelist ja nendele vastavad osakonnad teisest tabelist, siis päringu tulemuses ei näe osakonda, millel pole ühtegi töötajat.




```
SELECT nimi, osakonna_nimetus
FROM tootaja, osakond
WHERE tootaja.osakonna_nr = osakond.osak_nr;
```

Puuduva osakonna andmeid on võimalik päringu tulemusena näha, kui **joini tingimusse lisada nn. *outer join* operaator - (+)**

```
SELECT nimi, osakonna_nimetus
 FROM tootaja, osakond
 WHERE tootaja.osakonna_nr (+) = osakond.osak_nr;
```

See operaator lisab ühe või rohkem NULL rida päringusse, millega ühendada teise tabeli neid kirjeid, millele vastet esimese tabelis ei leidu. Ülaltoodu päringu tulemus võiks OUTER JOIN-I korral olla näiteks

NIMI	OSAKONNA_NIMETUS
JUURIKAS	AUTOMAJAND
METS	MEHHAANIKATSEHH
KASK	RAAMATUPIDAMINE
-------	PERSONALIOSAKOND

- **Tabeli ühendamine iseendaga (joining table to itself):**

Tabelite alias-nimesid kasutades (vt. eelmisi näiteid) on võimalik ühendada päringus kirjeid samast tabelist nii, nagu oleks tegemist kahe erineva tabeliga.
Järgnev päring on näide sellest, kuidas teada saada kõigi nende töötajate nimesid, kes saavad palka rohkem, kui nende ülemused (managers). Sellist iseendaga ühendamist võib vaja minna siis, kui ühes päringus on vaja ühte ja sama tabelit pärida tegelikult kaks korda , nagu antud näites : on vaja küsida töötaja palk ja samal ajal on vaja teada ka tema ülemuse (manageri) palka samast tabelis.

```
SELECT E.NAME EMP_NAME,
       E.SAL EMP_SAL,
       M.ENAME MGR_NAME,
       M.SAL MGR_SAL
FROM EMP E, EMP M
WHERE E.MGR = M.EMPNO
AND E.SAL < M.SAL;
```

- **Relatsioonialgebra operaatorite kasutamine: UNION, INTERSECT, MINUS.** Aitavad kombineerida mitme SELECT- lause tulemusi üheks tulemuseks. Päring võib seetõttu koosneda mitmest SELECT lausest, mis on seostatud ülaltoodud operaatorite abil.

```
SELECT JOB
FROM EMP
WHERE DEPTNO  =  10
UNION ALL
SELECT JOB
FROM EMP
WHERE DEPTNO = 30;
```

```
SELECT JOB
FROM EMP
WHERE DEPTNO  =  10
INTERSECT
SELECT JOB
FROM EMP
WHERE DEPTNO = 30;
```

```
SELECT JOB
FROM EMP
WHERE DEPTNO  =  10
MINUS
SELECT JOB
FROM EMP
WHERE DEPTNO = 30;
```

Reeglit relatsioonioperaatorite kasutamiseks:

1. SELECT laused peavad andma tulemuseks võrdse arvu kirjeid.
2. kokkulangevad read peavad olema samade andmetüüpidega.
3. SELECT lauseid täidetakse algusest (esimesest lausest) lõpuni.
4. Duplikaatread eemaldatekse automaatselt tulemusest.

- **Alampäringud.(subqueries)**

Alampäring on päring , mis asub teise SQL-lause sees. Süntaks on sama, mis tavalisel päringul, va. puudub puudub ORDER BY klausel. Tuleb vahet teha, kas alampäringut täidetakse üks kord terve SQL -lause kohta või iga pealause kohta eraldi.
Näiteks: a) Leida Juhan Juurikaga samal kursusel õppivad tudengid:

```
SELECT pnimi, enimi FROM tudeng WHERE kursus =
        (SELECT kursus FROM tudeng
          WHERE pnimi = 'Juurikas' AND enimi =' Jaan');
```

Alampäring (leida Juhan Juurika kursus) täidetakse ainult üks kord.

b) Lisa nende tudengite nimed, kes on vanemal kursusel kui nende suund keskmiselt:

```
SELECT pnimi, enimi  FROM tudeng x  WHERE kursus >
        (SELECT AVG(kursus) FROM tudeng y
          WHERE x.suund = y.suund);
```

Alampäringut täidetakse korduvalt, kui alampäring viitab pealausest pärit olevale veerule.
Kui alampäring viitab pealausest pärit olevale veerule, on tegemist **korreleeruvate** päringutega (*correlated queries*). Need päringud on üksteisest sõltuvad päringud ja ei saa käivituda iseseisvalt). Igale välimise SELECT lause rea kohta käivitub eraldi alampäring.

Nimedest - kõigepealt otsitakse alampäringu sees, kui sealt ei leita, siis alles otsitakse pealausest.

Nested subqueries - kui üks päring paikneb teise päringu sees (vt. eelmine näide)

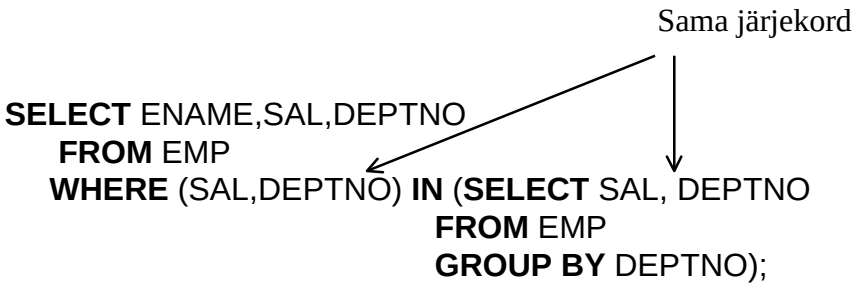
Alampäring võib vastuseks anda ühe rea (*single row subquery*) või mitu rida. Viimase näiteks:

```
SELECT ENAME, SAL, DEPTNO  FROM EMP
  WHERE SAL IN (SELECT MIN(SAL)
                FROM EMP
                GROUP BY DEPTNO);
```

Pööra tähelepanu sellele, et GROUP BY klausel tingib selle, et alampäring tagastab mitu rida (MIN() funktsiooni kasutatakse DEPTNO järgi grupeeritud gruppidele)

Reeglid alampäringutele:

*Alampäringus ei saa kasutada ORDER BY klauslit.
*Alampäringuid võib paigutada mitu tükki üksteise sisse (*nested subqueries*)
*Sisemine(alam)päring peab paiknema sulgudes WHERE klausli paremal pool.
*Kui sisemise päringu (inner query) valikulistis (selection list) on mitu veergu , siis peavad need veerud olema reastatud samas järjekorras kui välise päringu WHERE klauslis olevad veerud. Näiteks:



*ORDER BY klausel paikneb peamise SELECT-lause lõpus.
*Kõige enne täidetakse kõige sügavamal(most deeply nested) paiknevad alampäringud.

Alampäringud võivad:

*tagastada rohkem kui ühe kirje.
*tagastada rohkem kui ühe veeru.
*kasutada GROUP BY klauslit
*kasutada ühendit (join) kahest või rohkemast tabelist)
*alampäring võib kuuluda SELECT,UPDATE,DELETE,INSERT,CREATE TABLE lausete koosseisu.
*alampäring võib küsida andmeid teistest tabelitest kui väline (outer) päring.
*kasutada relatsioonialgebra operaatoreid (MINUS, INTERSECT jne.)

Miks kasutada **korreleeruvaid** päringuid:

- Korreleeruv päring on üks võimalus lugeda andmeid ühest tabelist ja samal ajal (samas päringus) võrrelda neid andmeid teiste andmetega, mille saamiseks on samuti vaja rakendada päringut. Alati, kui alampäring peab iga peapäringu rea kohta tagastama erinevaid väärtusi, tuleb kasutada peapäringuga korreleerivat alampäringut.

Päringutes on kasutatavad operaatorid EXISTS,SOME/ANY, ALL.

Päringud on kasutatavad ka andmete sisestamisel tabelitesse (INSERT -lauses)