# 5. Andmebaaside realiseerimise vahendid,erinevad andmebaasisüsteemid.Sissejuhatus SQL-keelde, põhimõisted.

* Andmebaasisüsteemid võib empiiriliselt jagada kaheks - **lokaalseteks** nn. desktop-andmebaasideks ja suurematеks, nn. **SQL-andmebaasideks**. Jaotada võib neid töödeldavate andmehulkade, andmebaasi projekteerimise(ülesehitamise) võimaluste hulga, andmetöötluse paralleelsuse (kas on andmete paralleeltöötlust või mitte), mitmeplatvormilisuse , andmete haldamise vahendite ja paljude muude parameetrite järgi.

Jaotamine iseenest ei ole väga tähtis, enamus andmebaasidega seotud põhiomadusi on kõikides süsteemides olemas, erinevused avalduvad vaid suuremate, keerukamate ja rohkemat jõudlust nõudvate süsteemide korral süsteemi realisatsioonietapil.

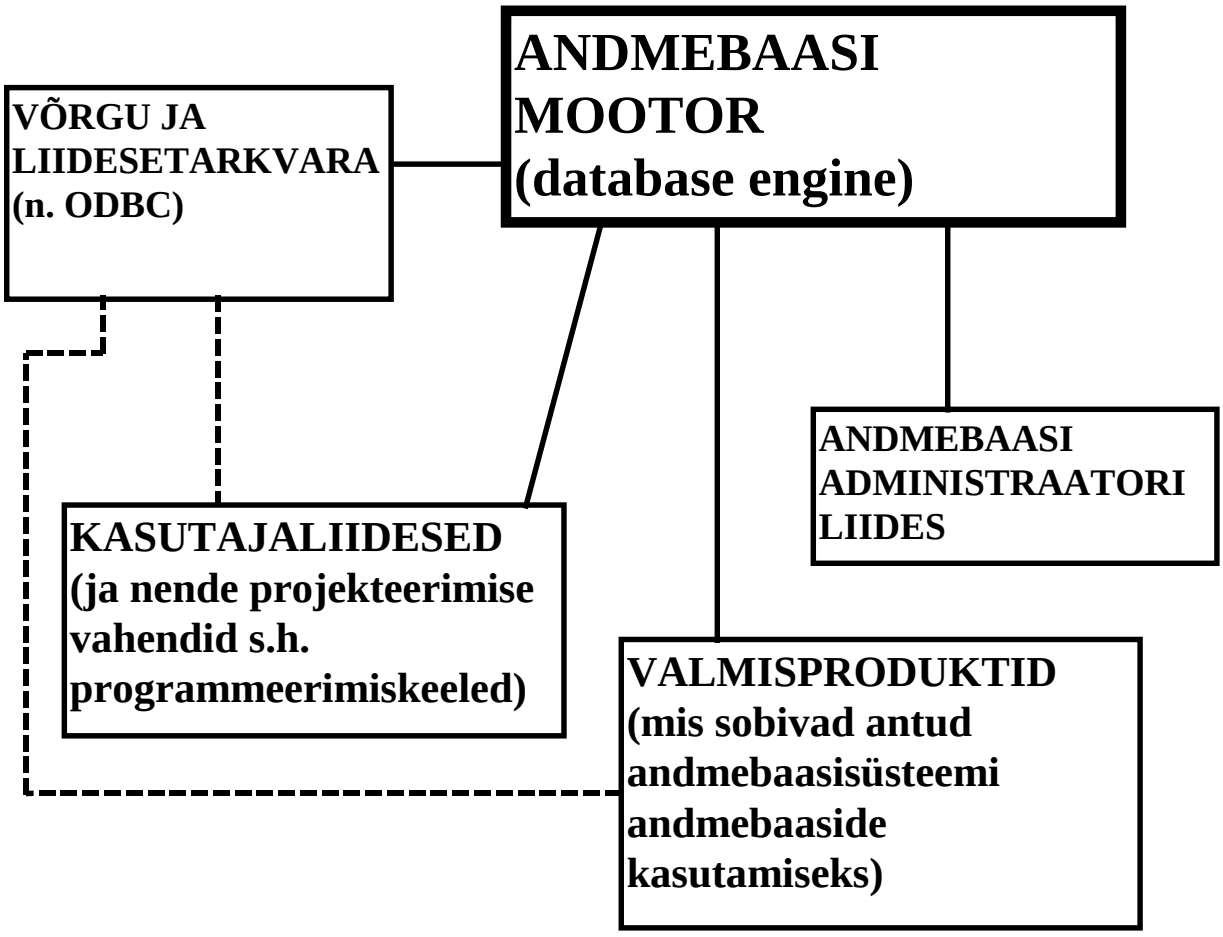
**Desktop (laua-arvuti) andmebaasisüsteemide eriomadused:**

- andmebaasisüsteemi on reegline integreeritud ka kasutajaliideste tegemise vahendid, aruannete genereerimise vahendid ja programmeerimiskeel.
- käsitletavad andmehulgad (mille puhul süsteemi toimimiskiirus on veel rahuldav) on suhteliselt väikesed.
- puudub või on nõrgalt organiseeritud andmete paralleeltöötlus.Sellega seoses töötavad sellised süsteemid võrguversioonidena (kui andmebaasil on läbi võrgu mitu kasutajat) suhteliselt aeglaselt ja ebakindlalt, eritikui on tegemist suuremate andmehulkadega. Sobivus hajussüsteemidesse jätab soovida.
- tavaliselt on sellised süsteemid ühe või kaheplatvormilised (st. andmebaasisüsteem

töötab kindlal operatsioonisüsteemil ja riistvaral - MS Windows, MacOS jne.)

SQL-andmebaaside eriomadused:

- andmebaasisüsteem on tavaliselt eraldiseisev toode, ilma kasutajaliideste ja muude lisavahenditeta. Ta kujutab endast tegelikult programmi, mis on võimeline looma andmebaasitabeleid ja muid andmebaasi objekte (indeksid, vaated, kasutajad jne.), andmeid salvestama ja vastama andmebaasi päringutele. Seda programmi nim. tavaliselt **andmebaasi mootoriks**(database engine). Selleks, et andmebaasisüsteemi kasutada , on vajalik mingi kasutajaliides, mille kaudu andmebaasisüsteemile "käske" jagada - andmebaasi administraatori kasutajaprogramm, samuti on vajalikud vahendid kasutajaliideste tegemiseks ning aruannete genereerimiseks andmebaasist, samuti side- ja võrgutarkvara andmebaasi ühendamiseks terviksüsteemi . Kõik need tarvara"tükid" tuleb hankida eraldi, vastavalt projekteeritava süsteemi iselooomule. Seetõttu on suurte infosüsteemide väljatöötamiseks vajalik terve tooterühm, millest ühe osa moodustab **andmebaasi mootor.**



- suuremad andmebaasisüsteemid töötavad tihti mitmetel platvormidel - kõige kõvem näide siin Oracle, mis töötab vähemalt paarikümnel eri platvormil (nii riist- kui tarkvara mõttes)
- arenenud andmete paralleeltöötlus, mitmekasutajareziim,töö võrkudes ja hajussüsteemides
- suur võimsus (suured töödeldavad andmehulgad)
- standardseks andmebaasidega töötamise vahendiks on SQL keel - **Standard Query Language**

- **STANDARD QUERY LANGUAGE- sissejuhatus**

**AJALUGU:**Relatsiooniline AB töötati välja E. F. Codd'i poolt 70-ndate alguses. SQL on pärit IBM-st, relats. AB prototüübist System R, 70-ndate keskel. Originaalne SQL keel (SEQUEL2) kirjeldati 1976 a. novembris IBM Journal of R&D. Esimene turule tulnud produkt oli 1979 Oracle Corp. poolt. Tänapäeval on kujunenud andmebaaside juurdepääsukeele standardiks.

**SQL - I üldiseloomustus:**

1. Mitteprotseduurne keel, sest
- Töödeldakse kirjete hulki, mitte üht kirjet korraga
- Juurdepääs (navigation) andmetele automaatne - ei nõua kasutajalt juurdepääsuviisi teadmist. SQL kasutab olemasolevaid indekseid automaatselt.

2. Kasutajate ring in väga lai:
- Süsteemi administraatorid
- Andmebaasi administraatorid
- Andmekaitse administraatorid
- Rakenduste programmeerijad
- Kõikvõimalikud lõppkasutajad

3. Ühtne keel
- Päringute esitamiseks
- Andmete lisamiseks, kustutamiseks, uuendamiseks
- Objektide (tabelid, indeksid...) loomiseks, eemaldamiseks, muutmiseks
- Juurdepääsuõiguste kontrollimiseks
- Andembaasi terviklikkuse tagamiseks.

4. Võimalik kasutada teiste protseduursete keelte sees (Embedded SQL), nagu Ada, C, COBOL, Fortran, Pascal, PL/1, Visual Basic, Java

**näide SQL-I kohta:** tabel KLIENT eelnevates näidetes toodud süsteemist (KLIENT - ARVE - KAUP)

| kliendi_kood | nimi | passi_number | aadress | registr_kpv |
|---|---|---|---|---|
| 1 | Mati Jõgi | K1012344 | Tallinn Sõpruse pst.34-45 | 11.11.1994 |
| 2 | Andres Mets | K1034564 | Paide Roheline 7 | 12.04.1995 |
| 3 | Tiiu Kask | N1904567 | Tartu Tähe 12 | 10.05.1993 |
| 4 | Anton Triik | G3455678 | Tartu Kase 12-44 | 10.03.1994 |

```
CREATE TABLE klient (kliendi_kood NUMERIC(10) NOT NULL PRIMARY
KEY,
nimi VARCHAR(100) NOT NULL,
passi_number VARCHAR(8),
aadress VARCHAR(100),
registr_kpv DATE NOT NULL);


INSERT INTO klient VALUES (1,'Mati Jõgi','K1012344',
'Tallinn Sõpruse pst.34-45','11.11.1994');

INSERT INTO klient VALUES (2,'Andres Mets','K1034564',
'Paide Roheline 7','12.04.1995');

INSERT INTO klient VALUES (3,'Tiiu Kask','N1904567',
'Tartu Tähe 12','10.05.1993');

INSERT INTO klient VALUES (4,'Anton Triik','G3455678',
'Tartu Kase 12-44','10.03.1994');
```

**SELECT nimi FROM klient;**

```
NIMI
==========================

Mati Jõgi
Andres Mets
Tiiu Kask
Anton Triik
```

**SELECT * FROM klient;**

```
KLIENDI_KOOD     NIMI          PASSI_NUMBER        AADRESS          REGISTR_KPV
==============   ======        ==============  ===============  ============

 1                Mati Jõgi     K1012344       Tallinn Sõpruse pst.34-45  11-NOV-1994
 2                Andres Mets   K1034564       Paide Roheline 7           12-APR-1995
 3                Tiiu Kask     N1904567       Tartu Tähe 12               10-MAY-1993
 4                Anton Triik   G3455678       Tartu Kase 12-44           10-MAR-1994
```

**GRANT ALL ON KLIENT TO PUBLIC;**
**GRANT ALL ON KLIENT TO “aleksander”;**

**nimed SQL - is:** 1-18 märki (Oracle - 30 märki)
Ei eristata suur- ja väiketähti
Nimi peab algama tähega
Võib sisaldada tähti ja numbreid (mõnikord ka _, $, #)
Ei tohi olla reserveeritud sõna, näit AND, CREATE, DECIMAL, TABLE jne.
1-18 märki (Oracle - 30 märki)
Ei eristata suur- ja väiketähti
Nimi peab algama tähega
Võib sisaldada tähti ja numbreid (mõnikord ka _, $, #)
Ei tohi olla reserveeritud sõna, näit AND, CREATE, DECIMAL, TABLE jne.

**KONSTANDID:** Tekst - 'Hello', 'John''s pen', '01.05.95', ''
Täisarv - 0, 123 (max. 15 numbrit, ainult pos.)

Arv - 25, +4.68, 0.5, 20e-02, -7 (Täpsusele max. 15 kohta, aste -130 ja 125 vahel)

**tehted, avaldistehted, funktsioonid:**
Aritmeetilised tehted:
unaarsed + - , * / + -

Tekstitehted:
|| - sidurdamine e. konkatenatsioon. 'Jaan '|| 'Kask'

Vôrdlustehted:
=, <>, <, >, >=, <=,
IN e. =ANY - Kontrollib sisalduvust loetelus
ANY e. SOME - vôrdleb loetelu elemendiga.
ALL - Vôrdleb kôigi loetelu elementidega
[NOT] BETWEEN x AND y - Vôrdleb vahemikku (servad k.a.)
EXISTS - Tôene, kui alampäring leidis vähemalt ühe rea
x [NOT] LIKE y - vôrdleb sarnasust. ... - suvaline märgijada, näit. pnimi LIKE '...mägi...'
ORACLE - % -suvaline märgijada, _ - üks märk
IS [NOT] NULL - kontrollib tühja väärtust. NB! 0 <> NULL

Loogilised tehted:
AND OR NOT

**keelekonvektsioonid:**
{|} - tuleb valida üks - { BEFORE | AFTER }
[] - vôib ära jätta, vôi valida loetelust ühe -
[DISTINCT | ALL]. Vaikimisi valik on allajoonitud.
... - Vôib korrata eelmist osa - [,expr]...
Keelekonventsioonid:
{|} - tuleb valida üks - { BEFORE | AFTER }
[] - vôib ära jätta, vôi valida loetelust ühe -
[DISTINCT | ALL]. Vaikimisi valik on allajoonitud.
... - Vôib korrata eelmist osa - [,expr]...

**ANDMETÜÜBID:** CHAR(n) e CHARACTER(n) (Oracle-CHAR)

Fikseeritud pikkusega tekst. Vaikimisi pikkus 1, max. n on 255. Kui andmed on lühemad, siis tühikud otsa, kui pikemad, siis lôgatakse tagant ära.

CHARACTER VARYING(n) e. CHAR VARYING(n) (Oracle-VARCHAR2(n) ja LONG Informix - VARCHAR(n))
Muutuva pikkusega tekst, n-max pikkus, ülempiir 2000 baiti (Oracle VARCHAR2), 32767 (Informix), 2GB (Oracle LONG)

INTEGER e INT, SMALLINT - Täisarv (Oracle-NUMBER(n))

DECIMAL(p,s) e DEC e NUMERIC - Fiks. komaga arvud (Oracle-NUMBER(p,s). p-numbrikohtade arv, s-skaala, e. kohtade arv komast paremal. Skaala vôib olla ka negat., siis on tegemist suurte arvudega. Vahemik -84...+127.

REAL e. SMALLFLOAT - reaalarv
FLOAT(b) e. DOUBLE PRECISION - topelttäpsusega reaalarv

DATE - kuupäev. Oracle-kuupäev koos kellaajaga, Informix-ainult kuupäev. Informixis selleks DATETIME x TO n, näit. DATETIME YEAR TO HOUR. Valikud on YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, FRACTION-tuhandik sekundit.

Peale selle on standardiväliselt:
Informix:
MONEY-raha, SERIAL-järjest.numereerimine, BYTE-binaarne, INTERVAL-ajavahemik
Oracle:
RAW ja LONG RAW-binaarne, ROWID-reanumber

- **SQL -keele lausendite tüübid:**

■ **päringud** (queries) andmete küsimiseks andmebaasist - “**SELECT”-**lause.
Päringulauses spetsifitseeritakse millistest tabelitest milliseid andmeid andmeid vajatakse (veerud, valiku kriteeriumid)

- **andmete defineerimise** laused (data definition statements) - defineerivad andmebaasistruktuuri, nendega luuakse kõik andmebaasi objektid - tabelid, vaated, indeksid, kasutajad, trigerid jne. "**CREATE" , "ALTER TABLE", "DROP"** laused.

- **andmete töötlemise** laused (data manipulation statements) - muudavad andmeid andmebaas - lisavad , muudavad, kustutavad andmeid. "**INSERT","UPDATE","DELETE"** laused.

- **Andmetele juurdepääsu kontrolli** laused (data-control statements) - kasutajatele privileegide ja andmete kasutamise õiguste jagamine, **"GRANT"** laused.

**SQL tegeleb puhtalt andmetega manipuleerimisel kõige kõrgemal loogilisel tasemel - nn. puhas andmetöötlus. SQL keeles on lausendid on võimalikult lihtsustatud, ainult andmetega tegelevad lausekonstruktsioonid.Need on loomuliku keele mõnevõrra organiseeritud ja reeglistatud lausendid.**

- **<u>Päringud. SELECT -lause</u>**

Andmete valimiseks tabelitest ja vaadetest.

Süntaks: **SELECT [DISTINCT | ALL] {* | <val> [, <val> ...]}**
**FROM <tableref> [, <tableref> ...]**
**[WHERE <search_condition>]**
**[GROUP BY col [COLLATE collation] [, col [COLLATE collation] ...]**
**[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.

- 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 ;**

- **<u>Funktsioonid</u>**

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. Alampäringud**.