"Those who use databases and eat sausages would probably be more comfortable if they didn't watch either being made." --Al Foster

Mõisted: andmebaas (AB, ingl.k. DB), andmebaasi juhtimissüsteem (ABJS, ingl.k. DBMS), abstraktsioonitasemed: kasutajavaade, kontseptuaalne vaade, füüsiline andmebaas; andmebaasi kirjelduskeel, andmebaasi manipuleerimiskeel, andmebaasi administraator (ABA, ingl.k. DBA), andmebaasisüsteemide klassifikatsioon: struktuursete andmetega – täisteksti andmebaasid; hajusad andmebaasid; andmemudeliga määratud andmebaasid (relatsiooniline, objekt-orienteeritud jne.); eriotstarbelised andmebaasid. Andmekeskne maailmapilt.

1. Sissejuhatus

1.1 Miks on meil vaja andmebaase?

Me elame infosüsteemide ajastul. Me kasutame igapäevaselt kümneid infosüsteeme: üliõpilasena registreerime kursustele ÕIS-s, ostame bussipileteid T-pileti kaudu, tellime muusikat ja raamatuid Amazon'ist, maksame makse e-maksuameti kaudu, astume ülikooli SAIS'i abil, õppejõudude teadustöö kohta saame ülevaate ETIS'e kaudu jne. Kõik mainitud süsteemid on veebipõhiste kasutajaliidestega infosüsteemid, mille kesksek osaks on andmebaas. Selle kursuse eesmärgina te omandate oskuse andmebaaside loomiseks, et saaks nende põhjal infosüsteeme ehitada. See ei ole lihtsalt omandatav oskus, sest andmebaasi loomisest on kasu ainult siis, kui ta on hästi loodud – temal põhinev infosüsteem vastab tellijate vajadustele, on mugav (loe: intuitiivne) kasutada, teda on mugav ja lihtne täiendada jne.

Tosinkond aastat tagasi, kui IT-hariduse õppekavasid ülikoolides hakati arutama IT firmade esindajatega, esines pragune Majandus- ja kommunikatsiooniministeeriumi riigi infosüsteemide osakonna juhataja Margus Püüa ettekandega, mis väga kujukalt näitab milliste raskuste ees andmebaaside loojad seisavad. Mõned viis mõtet tema tollasest ettekandest.

1. See, mida organisatsioon vajab, see mida ta tellib IT firmalt/osakonnalt ja see mida ta saab – need on kolm täiesti erinevat asja.
2. Eduka IT lahenduse juurutamiseks peab: probleem olema õige, lahendus olema õige, plaan viima lahenduseni, teostus ja juhtimine vastama plaanile.
3. Tehnoloogiast saab olulist kasu siis ja ainult siis, kui tehnoloogia likvideerib mõne takistuse organisatsioonis.
4. Kui likvideerida takistused, aga mitte muuta reegleid, siis on tulemus õudne.
5. 4 sammu eduks:
    1) Mis on IT peamine võime? (arvutmiskiirus, suurte andmehulkade haldamine)
    2) Millise takistuse see võime suudab likvideerida meie organisatsioonis?
    3) Millised reeglid ja töökorraldus aitasid meid enne selle takistusega hakkama saada?
    4) Millised on uued reeglid ja töökorraldus.

Teie tulevane tarkus peaks tagama selle edu! Need mõttekillud näitavad, et andmebaaside loojad peavad väga hästi aru saama organisatsiooni probleemidest, milleinfosüsteemi jaoks nad andmebaasi loovad.

NB! Mitte unustada: kõik algab probleemist! Näide: TÜ õppetulemuste arvestus. Oli aeg, kui tollane TÜ Matemaatikateaduskond võttis igal aastal vastu alla 50 tudengi. Kõik arvestus- ja eksamiprotokollid olid paberil, paralleelselt oli igal tudengil nn. matrikliraamat, kuhu kõik tulemused ka sisse kanti, s.t. oli topelt raamatupidamine. Kui tudengid 5 aasta pärast lõpetama hakkasid, oli vaja kontrollida, kas nad on

õppekava täitnud ja diplomite juurde koostada hinnetelehed. Selleks oli dekanaadis tööl inimene, kes juba pool aastat enne lõpetamist diplomite juurde kuuluvaid hinnetelehti koostas ja paberprotokollide ja matrikliraamatute põhjal kontrollis. Täna teeb kogu seda tööd ÕIS. Ühe lõpetaja dokumentide (diplom ja hinneteleht) väljatrükkimiseks ÕIS teenusena kulub ca 3 minutit.

1.2 Põhiterminid

Alustuseks tuleb vahet teha kahel mõistel: **tarkvara**, mille abil andmebaase luuakse ja andmetega täidetud **andmebaas ise**. Võrdle: tekstiredaktor ja selle abil loodud tekstidokumendid. Nii, nagu tekstidokumendi loomist ei ole vaja alustada tekstiredaktori programmeerimisest, nii pole ka andmebaasi loomiseks vaja hakata ise andmebaasi ehitamise tarkvara kirjutama, sest selline tarkvara, paljudes erinevates variantides, on juba olemas. Seda tarkvara, mille abil andmebaase luuakse ja loodud andmebaase kasutatakse nimetatakse **andmebaaside juhtimissüsteem** e. ABJS (ingl.k. *database management systems e. DBMS*). Sellise tarkvara arendamisega tegelevad tavaliselt suured firmad ja läbi aegade on loodud väga palju erinevaid andmebaaside juhtimissüsteeme. Populaarsemad tänapäeval kasutatavad süsteemid on näiteks: Oracle, MySQL, Access, SyBase jne.

Andmetega täidetud andmebaasid on tavaliselt asutuse mingi infosüsteemi keskne osa. Andmebaase on vaja siis, kui meil on palju andmeid ja nendega peavad töötama paljud erinevate vajadustega inimesed. Näiteks vaatame TÜ ÕIS'i, kus erinevate teaduskondade üliõpilased näevad vaid neile ette nähtud andmeid (õppekavad, õppeained, oma hinded jne.), õppejõud näevad lisaks ka veel kõiki oma ainele registreerunud üliõpilasi , ja ka dekanaadil ja õppeosakonnal on andmebaasi andmetele oma vaade.

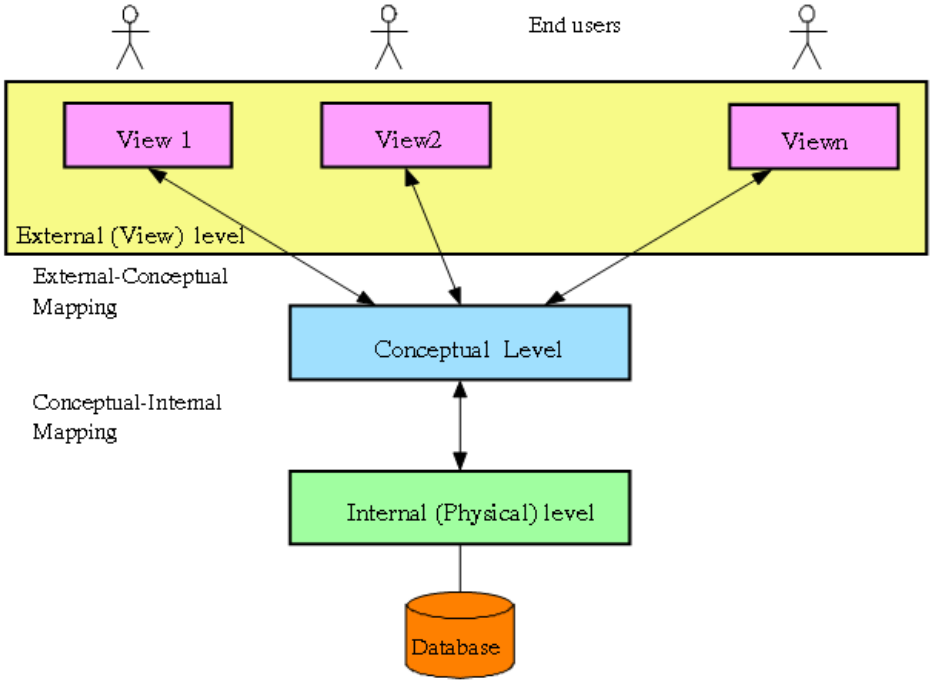
Et mitmeid kasutajagruppe teenindada ilma andmeid dubleerimata, koostatakse andmebaasis olevate andmete kohta selles andmebaasis hoitavate andmete üldine (e. kontseptuaalne) kirjeldus. Selle loomisel jälgitakse reaalsuses tekkivate andmete struktuuri, mitte erinevatele kasutajatele vajalikku struktuuri. Erinevate kasutajate infovajaduse järgi aga määratakse, milliseid andmeid üldse on vaja andmebaasis hoida. Igale kasutajale luuakse üldise kirjelduse põhjal neile vajalike andmete kirjeldus e. kasutajavaade. Oluline on, et kõik kasutajale vajaminevad andmed baasis olemas ja kättesaadavad oleks. Kui näiteks üliõpilaste TÜ Teadusliku raamatukogu lugejapileti numbreid andmebaasis ei ole, raamatukogu aga fikseerib külastuse pileti numbriga, ei saa ka raamatukogu juhtkonnale näidata teaduskondade kaupa üliõpilaste raamatukogu külastamise sagedusgraafikuid.

Kuna andmebaasid peavad võimaldama andmete efektiivset kättesaamist, siis on oluline see, kuidas andmed arvuti välismälus on organiseeritud. Ka selle kohta on andmebaasis vastav kirjeldus olemas, mida nimetatakse **andmebaasi füüsilise taseme kirjelduseks**.

2. Andmebaasi abstraktsioonitasemed.

Seega vaadatakse andmebaase tavaliselt kolmelt tasemelt: füüsiline, kontseptuaalne ja kasutajatase. Iga keerulist süsteemi aitab lihtsamalt käsitleda see, kui vaadata süsteemi mitmetasemelisena, igal tasemel omad eesmärgid ja ülesanded. Sellise nn. 3-tasemelise andmebaasi struktuuri pakkus juba 1975.a. välja Ameerika Standardite organisatsiooni üks komitee ANSI-SPARC (*American National Standards Institute, Standards Planning And Requirements Committee).* Vt. joonis 1.

Kontseptuaalne tase (*Conceptual level*) annab loodavast andmebaasist tervikliku pildi, ilma tehniliste üksikasjadeta.
Kasutaja tase (*view, end user or external level*) loob andmetest igale kasutajate grupile just neile vajaliku vaate, varjates mittevajalikud andmed ja esitades vajalikud andmed vajalikus vormis.
Füüsiline (või sisemine) **tase** tegeleb andmete füüsilise paiknemisega andmekandjatel, kirjeldab loodud juurdepääsuteid jms.



Joonis 1. Traditsiooniline 3-tasemeline andmebaasi arhitektuur (nimet. ka ANSI-SPARC arhitektuur)
http://cnx.org/content/m28150/latest/graphics6.png

Kui tavaliselt on andmed seotud konkreetse programmiga, mis andmeid töötleb, siis andmebaasides elavad andmed oma elu ja rakendusprogrammid suhtlevad andmetega **ainult läbi ABJS'i**.

Andmebaasidega tegelemisel on suureks abiks hea abstraktse mõtlemise oskus. Kuigi andmebaasides hoitakse reaalseid andmeid (näit. ÕIS andmebaasis muude asjade seas reaalsete tudengite isikuandmeid ja nende tudengite poolt sooritatud eksamite/arvestuste tulemusi), siis andmebaaside loomisel tegeldakse põhiliselt nende andmete kirjeldustega e. andmebaasi skeemiga. Skeem kirjeldab, mille kohta andmeid kogutakse (tudengid, ained, õppetulemused, õppekavad jne.) ja milliseid tunnuseid iga objekti kohta säilitatakse.

Andmebaasidega tegelemisel eristatakse kahte keelt: andmebaasi kirjelduskeelt (*Data Description Language e. DDL)* ja andmemanipuleerimiskeelt (*Data Manipulation Language e. DML).* DDL määrab, millised tunnuste tüübid (täisarv, sümboljada, kuupäev jne.) on olemas andmetunnuste kirjeldamiseks ja kuidas saame kirjeldada erinevate objektide vahelisi seoseid (näit. kuidas siduda tudengit tema poolt sooritatud eksamite/arvestustega). DML määrab, kuidas saame andmeid andmebaasis uuendada ja

milliste kriteeriumite alusel saame andmeid andmebaasist otsida. Reaalsetes süsteemides on nende kahe keele vahendid tihti integreeritud üheks andmebaasi keeleks (Näit. Structured Query Language e. SQL.

Kokkuvõtteks põhiterminite osas:

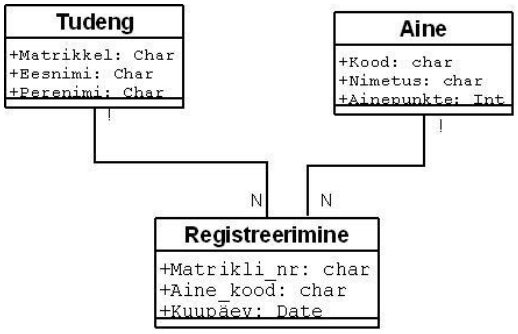
Andmebaas, AB (ingl.k. *Data Base - DB*) - omavahel seotud andmete kogu, mida hoitakse alaliselt arvutis.
NB! See pole definitsioon, vaid selgitus. !
Mõelge, millised andmed vastavad sellele selgitusele, mis aga ei ole andmebaas.
Täpsema selgituse saate esimese loengu jooksul.

Andmebaasi juhtimissüsteem ABJS (ingl.k. *DataBase Management System - DBMS*) - tarkvara, mis võimaldab andmebaasi luua ja käsitseda. Tihti kasutatakse lihtsamat terminit „andmebaasisüsteem".

3. Lihtne näide

Vaatame ÕIS'i ühe hästi lihtsa osa näidet. Meil on vaja säilitada andmeid üliõpilaste, õppeainete ja ainetele registreerimiste kohta. Üliõpilastest olgu meil vaja ees- ja perekonnanime ning matriklinumbrit; aine kohta tema koodi, nimetust ja ainepunkte; registreerumise kohta ainekoodi, matriklinumbrit ja ainele registreerumise kuupäeva.

Sellise lihtsustatud süsteemi kontseptuaalne mudel võiks välja näha selline (vt. joonis 2):



Joonis 2. Tudengite ainetele registreerimise valdkonna kontseptuaalse mudeli näide.

Kontseptuaalse mudeli kastides olevaid asju nimetatakse olemi tüübiks. Igale olemitüübile vastab andmebaasis palju olemi eksemplare (olemi tüüp – Tudeng, olemi eksemplarid on konkreetsed tudengi kohta käivad andmed). Olemitüüpide vahelised seosed on näidatu skeemil joontena – iga seose juures näidata seose tüüp. Ülaloleva Tudeng-Registreerimise seose tüüp näitab, et iga tudeng võib teha Palju registreerimisi, iga registreerimine kuulub aga alati vaid ühele tudengile.

Sellele skeemile vastavad andmed (olemi eksemplarid) võime näiteks säilitada tabelitena . Tabelid on omavahel seotud tunnuste väärtuste kaudu – näit. matrikli number esineb nii tabelis tudeng kui ka tabelis registreerimine . Kahe tabeli: Tudengid ja Registreerimine abil saame luua uue tabeli, kus on näiteks Ulvi Usina registreeringud - see oleks siis Ulvi Usina vaade antud andmebaasile.

Andmebaasi tabelid võivad olla väga mahukad. Näiteks Tartu Ülikoolis on ca 18 tuhat üliõpilast, ka erinevaid aineid on tuhandeid – registreerimiste arv on üsna suur.

Tabel: **Tudeng**

Matrikkel	Eesnimi	Perenimi
A034	Toomas	Tubli
B187	Ulvi	Usin
A729	Liisu	Laisk

Tabel: **Aine**

Kood	Nimetus	Punkte
MTAT.03.264	Andmebaasid	6
MTAT.03.100	Programmeerimine	6

Matrikli_nr	Aine_kood	Kuupäev
A034	MTAT.03.100	02.09.2012
B187	MTAT.03.105	05.09.2012
B187	MTAT.03.100	06.09.2012
A729	MTAT.03.100	07.09.2012

Tabel: **Registreerimine**

Joonis 3. Tudengite ainetele registreerimise andmebaasi tabelite näited.

Joonised 2 ja 3 illustreerivad andmebaasi kontseptuaalse mudeli ja andmebaasi enda sisu vahelist erinevust.

4. Andmebaaside terminoloogia

Siit edasi hakkame kasutama spetsialistidele omast terminoloogiat. Andmebaaside kontseptuaalsete mudelitega tegelemiseks on meil hädasti vaja tunda termineid: olem, olemi tüüp, võti ja seosed.

a) **Olem.** Olem (objekt) - on reaalselt eksisteeriva ja identifitseeritava asja või nähtuse esitus meie loodavas mudelis. Näiteks: on olemiks konkreetne tudeng ja konkreetne aine ÕIS ülesandes; ühe kliendi tellimus väikefirma haldamise ülesandes; mingi alustatav projekt IT-firma juhtimise ülesandes jne. Millised olemid meid huvitavad, see sõltub konkreetse ülesande (näit. õpihaldusülesande) eesmärkidest.

b) **Olemi tüüp.** Andmebaase on vaja seal, kus on palju andmeid. Andmed tekivad reaalses maailmas, andmebaasiga loome reaalsuses toimivale süsteemile infomudeli. Tüüpiliselt jagunevad reaalelulised objektid, millede kohta me andmebaasis tahame andmeid hoida, sarnaste objektide klassideks. Näiteks ÕIS hoiab andmeid ülikoolis õppivate tudengite kohta. ÕIS seisukohast on kõik tudengid teatud mõttes sarnased – nende kohta hoitakse samu andmeid: Eesnimi, Perekonnanimi, matrikli number jne. Teeme vahet olemi eksemplaride (üksik tudeng) ja olemite hulka kajastava olemitüübi vahel. Andmebaasi kirjelduses kirjeldatakse olemitüüpe. Olemi eksemplari esindavad andmebaasis selle olemieksemplari tunnuste väärtused. Konkreetset tudengit esindab andmebaasis tema ees- ja perekonnanimi ning matrikli number. Tunnuseid peab olema vähemalt niipalju, et nende väärtuste järgi identifitseerida reaalset olemi eksemplari. Näiteks juba matriklinumbrist piisab, et eristada tudengit ühe ülikooli piires. Kui tahame andmebaasis käsitleda näiteks kalakaupluses müüdavaid kalu, siis pole konkreetset kala esindavaid tunnuseid niisama lihtne leida. Kuidas eristada kaht letil olevat sama kaaluga latikat? Paneme lõpuse külge kollase plastnumbri, nagu nüüd lehmadel on? Võib-olla polegi kalade

müümise juures iga kala olemina arvele võtta vaja – piisab tootest " latikas", tunnusteks näiteks kilo hind ja toote identifikaator, mille saame hulgimüügifirmalt.

c) **Võti.** Neid tunnuseid, mille väärtused identifitseerivad olemi eksemplari üheselt, nimetatakse olemitüübi võtmetunnusteks. Võtmetunnuste komplekte võib olla palju. Tudeng on näiteks identifitseeritav väga mitme tunnuse põhjal: isikukood, passinumbri või lugejapilet ja ka näiteks kolmik: eesnimi, perenimi, sünnikuupäev peaks identifitseerimiseks sobima. Üks potentsiaalsetest võtmetest kuulutatakse selle olemitüübi andmebaasi võtmeks. Teised võtmed märgitakse tavaliselt ära kui unikaalsed, mittekorduva väärtusega tunnused – siis saab andmebaasisüsteem kontrollida lisatava eksemplari tunnuse unikaalsust. Näiteks, kuigi tudengi võtmeks ÕIS's on matriklinumber, ei luba süsteem lisada uut tudengit, kelle isikukood kattub juba olemasoleva, erineva tudengiga. Sellised kitsendused aitavad andmete sisestamisel vigu avastada.

d) **Seosed.** Olemitüüpide vahel on seosed. Näiteks: seos tudengi ja ainele registreerimise vahel. Seoseid klassifitseeritakse selle järgi, mitu ühe olemitüübi eksemplari saavad olla seotud teise olemitüübi eksemplaridega. Näiteks üks tudeng saab registreeruda mitmele ainele, aga iga registreerumine on seotud kindlasti parajasti ühe tudengiga. Seega on olemitüübi Tudeng ja olemitüübi Registreerumine vahel 1:n ehk üks mitmele seos. Kui vaatame näiteks bakalaureusetööde juhendamist, siis ühte tudengit võib meil juhendada ka kaks (või isegi enam) õppejõudu. Juhendamise seos tudengi olemitüübi ja õppejõu olemitüübi vahel mitu-mitmele e. n:m seos. Ühte tudengit saab juhendada mitu õppejõudu, aga ühel õppejõul võib olla mitu juhendatavat tudengit.

   Esineb ka olukordi, kus seos on 1:1, näiteks laevad ja kaptenid on seotud 1:1 seosega: igal laeval on parajasti 1 kapten ja igal kaptenil saab antud ajahetkel olla vaid üks laev. Klassikaline abielu on 1:1 seos olemitüübi mees ja olemitüübi naine vahel. Mida kajastaks siis analoogne 1:n ja n:m seos?

5. Andmebaaside konstrueerimine (ing.k. *Database Design*)

Muutuse, mis sisaldab ka andmebaasi loomist, tõukejõuks on probleem mingis organisatsioonis. See probleem tuleb kirjeldada ja sõnastada visioon, kuidas IT abil see probleem lahendada. Kui probleem ja visioon on kirjas, kutsutakse tüüpiliselt appi mingi IT-firma (suuremates organisatsioonides ka oma IT-osakond) ja asutakse visioonis kirjeldatud nägemust looma. Sageli on vaja selle käigus luua kõigile tulevast süsteemi kasutama hakkavatele gruppidele ühine antud probleemi lahendada aitav kontseptuaalne mudel ja selle põhjal luua andmebaas. Selle andmebaasi loomine algab nõuete analüüsist. Andmebaasi loomine ise on keeruline protsess ja parim viis selle protsessi juhtimiseks on – jagada ta mitmeks erinevaks etapiks ning vaadata neid etappe eraldi.

a) Nõuete analüüs

Tarkvaratehnoloogia õpetab meid, et ülesandele vastava tarkvara loomine algab probleemi sõnastusest e. üldisest ülesande püstitusest ja seejärel nõuete analüüsist, mille jooksul koos tellijatega pannakse kirja nii funktsionaalsed (mida tarkvara tegema peab) kui ka mittefunktsionaalsed (üldised) nõuded. Näiteks: ÕIS peab igal süsteemi sisseloginud tudengil ennast valitud ainele registreerida lubama (funktsionaalne nõue). Kuna TÜ teised infosüsteemid kasutavad

Oracle andmebaasi juhtimissüsteemi, siis peab ka ÕIS kasutama sama ABJsüsteemi (mittefunktsionaalne nõue).

b) Andmebaasi skeem

Lähtudes nõuetest, koostatakse andmebaasi kontseptuaalne e. loogiline skeem. Selleks on olemas spetsiaalsed graafilised vahendid. Seejärel koostatakse andmebaasi füüsiline mudel – millisel kujul andmed andmebaasis füüsiliselt on, milliste tunnuste järgi peavad nad kiiresti leitavad olema jne.

c) Vaated

Erinevatele kasutajagruppidele luuakse neile vajalikud vaated andmebaasi osadele. Näiteks saab õppejõud vaadata oma ainele registreerunud tudengite nimekirju.

d) Andmehõive

Selles etapis lahendatakse probleemid: kuidas toimub andmete alglaadimine andmebaasi (teisest baasist, algsisestamisega vms.) ja kuidas tagatakse igal hetkel andmebaasi andmete vastavus tegelikkusele. Nõuetest peab olema näha, kui palju aega andmete uuendamiseks võib kulutada.

e) Rakenduste loomine

Kasutajad ei suhtle andmebaasi süsteemil andmebaasiga, vaid neile luuakse spetsiaalsed rakendusprogrammid. Nende abil saavad kasutajad spetsiaalse, tihti veebipõhise, kasutaja liidese. Põhilisteks vahenditeks selliste liideste loomisel on SQL andmebaasiga suhtlemiseks, graafilise kasutajaliidese loomise vahendid ja programmeerimiskeeled.

f) Andmebaasi administreerimine

Tihti on andmed andmebaasis palju kallimad kui riistvara, millel andmebaas jookseb, kallimad isegi kui hoone, milles see riistvara asub. Vastavalt tuleb neisse ka suhtuda ja neid hooldada. Inimest (või inimeste rühma), kes konkreetset andmebaasi haldab, nimetatakse andmebaasi administraatoriks. See on amet, mis nõuab head andmebaaside alast haridust, pikaajalist kogemust ja on maailmas üsna hästitasustatav ametikoht.

Seda sammude jada nimetatakse ka andmebaasi elutsükliks. Elutsükli erinevate sammudega tegelevad suurfirmades erinevad spetsialistid: analüütikud, süsteemiarhitektid, rakendusprogrammeerijad, andmebaasi administraatorid jne.

6. Andmebaasi juhtimissüsteemi omadused ja klassifikatsioon

Andmebaasi juhtimissüsteem (ABJS) peab tagama:

- Andmete kirjeldamise vahendid, et andmebaasi kirjeldust luua ja vajadusel seda muuta.
- Andmete lisamise, eemaldamise ja parandamise vahendid, et andmete seisu baasis hoida vastavuses reaalses elus toimuvaga.
- Andmete kiire kättesaamise vahendid, et mistahes kriteeriumite järgi efektiivselt vastata päringutele.

Selleks kõigeks on andmebaasi juhtimissüsteemide jaoks olemas vastavad andmebaasi keeled: aAndmete kirjelduskeel (ingl.k. *data description languga* e. DDL ja andmetega manipuleerimise keel (ingl.k. data

manipulation language e. DML). Paljudes süsteemides ühendatakse need keeled üheks keeleks. Tuntuim selline ühendatud keel on **Structured Query Language e. SQL.**

Lisaks kõigele sellele peab ABJS tagama

- **Andmete säilimise** (ka siis, kui näiteks elekter mõneks tunniks ära läheb, mäluseade tõrgub, mõni programm vea tõttu hunnikusse jookseb jne).
- **Mitme kasutaja samaaegse töö juhtimine**. Pisinäide: Vanaisa kannab teile, kui vastsele tudengile Internetipangas üle 50€, samal ajal olete teie poes ja tasute pangakaardiga arvet. Oletame, et sündmuse juhtuvad selles järjekorras: Vanaisa pangaülekanne küsib pangast teie arve seisu kirje, siis küsib kaupluse rakendusprogramm teie arve seisu, siis kirjutab vanaisa ülekande programm 50€ võrra suurendatud seisu tagasi, seejärel kirjutab kauplus saadud vanast seisust jäägi tagasi. Tulemus – vanaisa ülekannet nagu polekski olnud!
- **Andmekaitse** – näit. teisi inimesi ei saa ainetele registreerida, teiste üliõpilaste andmeid ei saa teie muuta, ei endale ega kellelegi teisele hindeid ise kirjutada, isegi mitte teiste inimeste andmeid vaadata jne.
- Kui andmebaas paikneb laiali mitmes kohas, ei pea enamus kasutajaid sellest aru saama. Kas te teate, mitmes arvutis on laiali TÜ ÕIS andmed? Seega katab andmebaasi juhtimissüsteem kasutajate eest vajadusel ära **andmebaasi hajususe**.
- Kui andmebaas võtab enda peale infovahetuse kasutaja ja andmebaasis paiknevate andmete vahel, peab ta tegelema ka **mälu jagamise probleemidega** – andmebaasiga võivad korraga tegeleda sajad kasutajad.
- Kuna ainult ABJS teab, kuidas andmed välismälus paiknevad, siis peab tema ka tegelema **päringute optimeerimisega**.

Kokkuvõttes: andmebaasi juhtimissüsteem on keeruline süsteem, mis koosneb väga paljudest osadest.

Andmebaasi juhtimissüsteeme klassifitseeritakse väga erinevate kriteeriumite alusel.

- **Andmebaasis hoitava info järgi** – bibliograafilised andmebaasid, täisteksti andmebaasid, piltide/fotode andmebaasid, personaliandmete andmebaasid jne.
- **Andmemudelite j**ärgi s.t. nende vahendite järgi, mille abil andmebaase kirjeldatakse: hierarhilised andmebaasid, relatsioonilised andmebaasid, objekt-orienteeritud andmebaasid, jne. Neist levinuim on relatsiooniline andmebaas.
- Mittehajusad e. **tsentraalsed andmebaasid** ja **hajusandmebaasid,** sõltuvalt sellest, kas andmed paiknevad kõik ühel serveril (tsentraliseeritud andmebaasid) või hajusalt paiknevatel mitmetel serveritel (hajusandmebaasid).

Harjutus 2. Saate koos naabriga lehe, kus on kümmekond õpikutest/veebist korjatud andmebaasi definitsiooni. Neile tuginedes koostage uus definitsioon, mis sisaldab teie arvates olulisimaid asju.

7. Miks andmebaasid tekkisid?

Juba 1960-ndatel aastatel hoiatasid USA teadlased, et kui majandus sama tempoga edasi areneb, siis peavad paarikümne aasta pärast kõik USA töövõimelised inimesed raamatupidajateks hakkama, et seda tohutut tootmise ja teenuste osutamisega seotud dokumentide laviini käsitseda. Õnneks jõudis arvutite areng niikaugele, et isegi väikesed perefirmad saavad oma tegevuseks vastavaid andmebaasidele ehitatud majandamise tarkvara kasutada. Hoo andmebaaside loomiseks andis mäluseadmete areng – tekkisid otsepöördusega välismäluseadmed, kus vajaliku informatsiooni leidmine ei sõltunud enam lineaarselt andmete paiknemisest magnetlindil. Lisaks sellele hakkas ühe asutuse arvuti abil hallatavate ülesannete hulk kasvama ja nendel ülesannetel olid osaliselt kattuvad andmed. Näiteks TÜ-s tekkisid eraldi personaliarvestamise süsteem, õppetöö haldamise süsteem, raamatukogu laenutuste süsteem ja need kõik vajavad nii üliõpilaste kui õppejõudude andmeid. On mõttetu igas süsteemis eraldi tagada andmete vastavust reaalsele elule kui igal aastal lahkuvad ülikoolist tuhanded tudengid ja igal sülgi võetakse vastu uued tuhanded. Andmebaasid pidid looma olukorra, kus mitu rakendust said kasutada ühiseid andmeid ja seetõttu oli vaja koos andmetega salvestada ka andmete kirjeldused. Programmid tuli seejärel koostada nii, et need oskaksid neid kirjeldusi kasutada. Andmed kaotasid sõltuvuse programmidest ja hakkasid oma elu elama. Seda uut situatsiooni võrreldi sellega kuidas Nicolaus Copernicus XVI sajandil muutis meie arusaama päikesesüsteemi struktuurist geotsentrilisest - heliotsentriliseks. Andmebaaside tekkega muutus arvuti- (ja programmide-) keskne maailmapilt andmete keskseks. Ja selle paradigmaatilise muutuse keskmes oligi uut tüüpi tarkvara: andmebaasi juhtimissüsteem.

8. Miks on vaja tunda andmebaaside teooriat?

Vt. üht andmebaaside teooria ja praktikaga seotud veebisaiti: *Database Debunkings* URL=< http://www.dbdebunk.com/p/about_4.html > Põhjuste hulgas, miks vastav sait üldse loodi, on C.J.Date'i, paljude andmebaasi- õpikute autori, järgmine arutlus.

" Current DBMS deficiencies are, it seems to me, directly due to the widespread lack of understanding (not least on the part of vendors), of fundamental database principles. Certainly it is undeniable that they flout those principles in numerous ways. And the practical consequences are all too obvious: First, users must understand where the deficiencies lie; second, they have to understand just why they ***are*** *deficiencies; third, they have to understand how to work around them; and fourth, they have to devote time and effort in persuading the vendors to remedy them.*

The trouble is, of course, users too tend to be unaware of those same fundamental principles and, hence, find themselves unable to carry out their side of the "contract" (a "contract" that should not have been allowed, or agreed to in the first place, of course). It's a vicious cycle. What is more, this sad state of affairs is not likely to change given the apparent lack of interest on the part of the trade press--itself ignorant of these same principles--in trying to improve matters. "

Kokkuvõttes: Selles kursuses omandatud teadmised ja oskused on olulised mitte ainult andembaasi administraatorile, vaid ka analüütikutele, kõigile programmeerijatele (sest enamus rakendusi kasutab

andmebaaside andmeid), testijatele ja peaksid loomulikult kuuluma ka IT-lahendusi tellivate organisatsioonide esindajate haridusse.

Kirjandus

H.Darwen, An Introduction to Relational Database Theory (saabtasuta alla laadida 231lk. .pdf faili) http://bookboon.com/uk/student/it/an-introduction-to-relational-database-theory