# **ANDMEBAAS**

***DEF.Andmebaas on struktureeritud teabe või andmete järjestatud kogum, mis tavaliselt salvestatakse elektrooniliselt arvutisüsteemi. Andmebaasi haldab andmebaasihaldussüsteem***.

Kaasaegsete andmebaaside levinumate tüüpide andmed salvestatakse tavaliselt ridade ja veergude kujul, mis moodustavad tabeli. Neid andmeid saab hõlpsasti hallata, muuta, ajakohastada, jälgida ja korrastada. Andmebaase on mitut tüüpi, millel on oma eelised ja puudused ning mida kasutatakse teatud ülesannete jaoks.

## Relatsiooni andmebaas (Relational database)

See on tüüp, **milles teavet salvestatakse tabelite kujul**, mis koosnevad veergudest ja ridadest. Iga sellise andmebaasi tabelis sisalduv rida on kordumatu identifikaatoriga kirje, mida nimetatakse võtmeks. Tabeli veergudel on andmeatribuudid ja iga kirje sisaldab tavaliselt iga atribuudi väärtust, mis muudab andmeelementide vaheliste seoste loomise lihtsaks. Relatsiooniandmebaasi on mugav kasutada, kui on vaja andmeelemente omavahel siduda ning neid turvaliselt ja usaldusväärselt hallata, kuna relatsioonimudelis eeldab see loogilist andmestruktuuri: tabeleid, vaateid ja indekseid.[2] Relatsiooni andmebaasid skaleeruvad vertikaalselt. See tähendab, et andmemahu suurenemisega suurendavad need arvutusprotsessorite võimsust ja serveri koormust

**Loogiline struktuur erineb kirje füüsilisest struktuurist selle poolest, et võimaldab töötada füüsilise salvestussüsteemiga ja seega mitte muuta tabelites olevaid andmeid**. Füüsilised ja loogilised struktuurid erinevad toimingute poolest. Tõeväärtuse toimingud võimaldavad rakendustel määratleda nõuded nõutavale sisule. Füüsilised toimingud määravad, kuidas andmetele juurde pääsetakse ja ülesandeid täidetakse. Selleks, et relatsiooni andmebaasid oleksid täpsemad ja ligipääsetavamad, on vaja rakendada normaliseerimis meetodeid. Relatsioonimudel säilitab kõige tõhusamalt andmete terviklikkuse rakenduste ja andmebaasi eksemplaride vahel. Relatsiooniandmebaasid sobivad suurepäraselt andmete terviklikkuse säilitamiseks mitmes andmebaasi eksemplaris korraga.

**Neli olulist omadust määratlevad relatsiooniandmebaasi tehingud: aatomilisus, järjepidevus, isoleeritus ja vastupidavus – mida tavaliselt nimetatakse ACID-ks.**

* **Aatomilisus määratleb kõik elemendid, mis moodustavad täieliku andmebaasi tehingu.**
* **Järjepidevus määratleb reeglid andmepunktide õiges olekus hoidmiseks pärast tehingut.**
* **Isolatsioon hoiab segaduse vältimiseks tehingu mõju teistele nähtamatuks kuni selle sooritamiseni.**
* **Vastupidavus tagab, et andmete muudatused muutuvad pärast tehingu sooritamist püsivaks.**

Relatsiooniandmebaasid luuakse andmebaasihaldussüsteemis tuginedes SQLe, mis on kõige levinum standardiseeritud keel, mida kasutatakse andmebaasidele juurdepääsuks. SQL põhineb relatsioonialgebra algoritmidel ja selgel matemaatilisel struktuuril. ANSI SQL standardit toetavad kõik populaarsed relatsiooniandmebaasi mootorid. Mõned tuumad sisaldavad ka ANSI SQL-standardi laiendusi, mis toetavad nende tuumade spetsiifilisi funktsioone. SQL-i kasutatakse andmeridade lisamiseks, värskendamiseks ja kustutamiseks, tehingute töötlemise ja analüütiliste rakenduste andmekogumite toomiseks ning andmebaasi kõigi aspektide haldamiseks Microsoft Access on ka relatsiooniline andmebaasihaldussüsteem loodud Microsoft Corporationi, millel on lai valik funktsioone, sealhulgas lingitud päringud, lingid välistele tabelitele ja andmebaasidele. Tänu sisseehitatud VBA keelele saate kirjutada rakendusi, mis töötavad andmebaasidega Accessis.[16]

Relatsioonilised andmebaasid on nõutavad paljudes organisatsioonides, kus neid kasutatakse raamatupidamiseks, laos olevate kaupade arvu jälgimiseks, tarnijate, partnerite ja klientide personali andmete haldamiseks. Samuti on nad leidnud oma rakenduse Interneti-tehnoloogiates registreeritud kasutajakontode salvestamise korraldamiseks, sõnumitekstide salvestamiseks Interneti erinevates piirkondades, näiteks foorumites, suhtlusvõrgustikes ja uudistevoogudes.[28]



Joonis 1.1 Relatsiooniandmebaas

Plussid:

* Relatsioonilise andmebaasi mudel ei nõua keerulisi päringuid, kuna sellel pole päringute töötlemist ega struktureerimist, nii et andmete käsitlemiseks piisab lihtsatest SQL-päringutest.
* Kasutajad saavad hõlpsasti juurde pääseda oma nõutavale teabele või seda hankida mõne sekundi jooksul, ilma andmebaasi keerukusest järele andmata. SQL kasutatakse keerukate päringute täitmiseks.
* Relatsiooniandmebaasid on rangelt määratletud ja hästi organiseeritud, nii et andmed ei dubleerita. Relatsiooniandmebaasid on nende struktuuri tõttu täpsed, ilma andmete dubleerimiseta.
* Relatsiooniandmebaasi haldussüsteeme kasutatakse laialdaselt ka andmete terviklikkuse tagamiseks, kuna need tagavad järjepidevuse kõigis tabelites. Andmete terviklikkus tagab sellised funktsioonid nagu täpsus ja kasutuslihtsus.
* Andmed on turvalised, kuna relatsiooniandmebaasi haldussüsteem võimaldab andmetele otse juurdepääsu ainult volitatud kasutajatel. Ükski volitamata kasutaja ei pääse teabele juurde.
* Andmebaasi juurde pääseb mitu kasutajat, et saada teavet korraga ja isegi siis, kui andmeid värskendatakse.
* Andmebaasi normaliseerimine tagab ka selle, et relatsiooniandmebaasi struktuur pole erinevus ja seda saab täpselt manipuleerida.

Miinused:

* Relatsiooniandmebaasi haldamine muutub aja jooksul andmete suurenemise tõttu keeruliseks.
* Relatsiooniandmebaas koosneb ridadest ja veergudest, mis nõuab palju füüsilist mälu, kuna iga sooritatav toiming sõltub eraldi salvestusruumist. Füüsilise mälu nõuded võivad suureneda koos andmete arvu suurenemisega.
* Kui kasutada relatsiooniandmebaasi mitmes serveris, muutub selle struktuur ja muutub raskesti käsitletavaks, eriti kui andmete hulk on suur. Tänu sellele ei ole andmed erinevatel füüsilistel salvestusserveritel skaleeritavad. Lõppkokkuvõttes mõjutab see selle jõudlust, st andmete ja laadimisaja puudumine jne. Kuna andmebaas muutub suuremaks või jaotub suurema hulga serveritega, on sellel negatiivsed tagajärjed, nagu latentsus- ja kättesaadavusprobleemid, mis mõjutavad üldist jõudlust.
* Relatsiooniandmebaasid suudavad salvestada andmeid ainult tabeli kujul, mis muudab keerukate objektidevaheliste suhete esitamise keeruliseks. See on probleem, sest paljud rakendused nõuavad rohkem kui ühte tabelit, et salvestada kõik vajalikud andmed, mida nende rakendusloogika nõuab. [11]

### **1.1.1. Seosed tabelite vahel**

Seosed kahe või enama tabeli vahel luuakse primaar- ja välisvõtmete abil. Primaarvõti on väli, mida kasutatakse iga tabeli kirje unikaalseks tuvastamiseks. Välisvõti on ühe tabeli väli või mitu välja, mis viitavad teise tabeli primaarvõtmele. Tabelite vahel on kolme tüüpi seoseid: üks-ühele, üks-mitmele, mitu-mitmele.

Üks-ühele tähendab, et ühe tabeli rea saab siduda teise tabeli ühe reaga. Näitena saab luua tabeli "tudengid", kus on õpilase täisnimi ja sünniaeg ning teine tabel "Ülikoolidandmed" sisaldab tema maatrikskoodi ja ülikooli kirja.



Joonis 1.2 Seos üks-ühele näiteks

Sellist ühendust peetakse üks-ühele, kuna igal üliõpilasel on oma unikaalne maatrikskood ja e-post, samuti kuuluvad iga ülikooli andmed ainult ühele üliõpilasele.

Üks-mitmele tähendab, et ühe tabeli rea saab siduda teise tabeli paljude ridadega. Näitena võib tuua tabeli "Näitlejad", kus asuvad näitleja isikuandmed ja "Rollid", kus on tegelase nimi ja millises filmis see tegelane oli.



Joonis 1.3 Seos üks-mitmele

Seda suhet peetakse üks-mitmele, kuna näitleja võib erinevates filmides mängida rohkem kui üht tegelast, kuid kogu filmi jooksul saab tegelast mängida ainult üks inimene.

Mitu-mitmele tähendab et ühe tabeli rida võib olla seotud paljude teise tabeli ridadega ja vastupidi. Sellist ühendust aga otse luua ei saa. Seda tehakse läbi kolmanda tabeli. Mitu-mitmele suhte näiteks on tabelid "Õpilased" ja "Kooliained" vastavalt sellele, mitu õpilast saab ühes tunnis olla ja üks õpilane käib mitmes tunnis. [17]



Joonis 1.4 Seos mitu-mitmele

### **1.1.2. Normaliseerimine**

***Normaliseerimine on andmebaasi teisendustehnika, mis minimeerib andmete liiasust ja kõrvaldab liiasuse soovimatud tagajärjed, nagu erinevad kõrvalekalded, jõudluse halvenemine, ning muudab andmehalduse paindlikuks ja mugavamaks.*** Üleliigsuse kõrvaldamisel peate jagama suure tabeli väiksemateks ja ühendama need suhete abil. Normaliseerimata kujul võivad tabeli veerud ja väärtused salvestada teavet kahe või enama olemi kohta. Normaliseerimisel on 9 vormi, mis sisaldavad oma reegleid ja kriteeriume. Viib tabeli järgmisele vormile ja järgides reegleid, on võimalik vältida erinevaid anomaaliaid ja viia tabel miinimumi liiasuseni. [18][19]

Normaliseerimata vorm (UNF): Sellel kujul on tabelis palju kõrvalekaldeid ja dubleerivaid väärtusi.[29]

Tabel 1.1 Näiteks normaliseerimata vorm

|  |  |
| --- | --- |
| Inimine | E-mail |
| Kentaro Miura | Isiklik post berserk1989@gmail.com; Töö post Miura.Kentaro@gmail.com |
| Edward Elric | Ed.ferrum@gmail.com |
| Edward Elric | Ed.ferrum@gmail.com |
| Thorkell Long | IamBig@gmail.com; Thorkell.Long@gmail.com |

Esimene normaalvorm (1NF): Sellel kujul ei tohi tabelis olla topeltridu, igal lahtril peab olema mitteliitväärtus ja iga kirje peab olema kordumatu.[30]

Tabel 1.2 Näiteks esimene normaalvorm

|  |  |  |
| --- | --- | --- |
| Inimene | E-mail | Tüpp |
| Kentaro Miura | berserk1989@gmail.com | Isiklik post |
| Kentaro Miura | Miura.Kentaro@gmail.com | Töö post |
| Edward Elric | Ed.ferrum@gmail.com |  |
| Thorkell Long | IamBig@gmail.com |  |
| Thorkell Long | Thorkell.Long@gmail.com |  |

Teine normaalvorm (2NF): Sellel kujul peab tabel olema esimesel kujul ja sellel peab olema üks veerg primaarvõtmega.[31]

Tabel 1.3 Näiteks teine normaalvorm

|  |  |  |  |
| --- | --- | --- | --- |
| Number | Inimene | Eriala | Kirjeldus |
| 1 | Kentaro Miura | Kunstnik | Joonistab Jaapani koomikseid |
| 2 | Edward Elric | Keemik | Uurib aine koostist ja selle omadusi |
| 3 | Thorkell Long | Ehitaja | Ehitab hooneid |

Kolmas normaalvorm (3NF): Sellel kujul peab tabel olema teisel kujul ja sellel ei tohi olla transitiivset sõltuvust, see tähendab, et võtmeta veerud ei tohi sõltuda muude mittevõtmeteta veergude väärtustest.[32]

Tabeli 1.3 tõlkimiseks kolmandale vormile tuleb see jagada kaheks tabelisse, kuna elukutse kirjeldus ei ole seotud inimese endaga.

Tabel 1.4 Tabel inimeste kolmandal normaalvormil

|  |  |  |
| --- | --- | --- |
| Number | Inimene | Eriala |
| 1 | Kentaro Miura | 1 |
| 2 | Edward Elric | 2 |
| 3 | Thorkell Long | 3 |

Tabel 1.5 Tabel erialade kolmandal normaalvormil

|  |  |  |
| --- | --- | --- |
| Number eriala | Eriala | Kirjeldus |
| 1 | Kunstnik | Joonistab Jaapani koomikseid |
| 2 | Keemik | Uurib aine koostist ja selle omadusi |
| 3 | Ehitaja | Ehitab hooneid |

Boyce-Coddi normaalvorm (BCNF): Sellel kujul peab tabel olema kolmandal kujul ja liitvõtme võtmeatribuudid ei tohi sõltuda võtmeatribuutidest.[33]

Tabel 1.6 Tabel õpilased

|  |  |
| --- | --- |
| Number õpilane | Õpilane täisnimi |
| 1 | Kentaro Miura |
| 2 | Edward Elric |
| 3 | Thorkell Long |

Tabel 1.7 Tabel ainete

|  |  |  |
| --- | --- | --- |
| Number aine | Aine | Õpetaja |
| 1 | Matemaatika | E.Vasiljeva |
| 2 | Keemia | A.Kuld |
| 3 | Kehalise kasvatus | K.Miyamoto |

Tabel 1.8 Suhtlemine õpilaste ja õppeainete vahel

|  |  |
| --- | --- |
| Number õpilane | Number aine |
| 1 | 1 |
| 1 | 3 |
| 2 | 1 |
| 2 | 2 |
| 3 | 3 |

Neljas normaalvorm (4NF): Sellel kujul peab tabel olema kolmandal kujul (kui primaarvõti on lihtne) või Boyce-Coddi kujul ning sellel tabelis ei pea olema mitmeväärtuslikud sõltuvused.[34]

Viies normaalvorm (5NF): Sellel kujul peab tabel olema neljandal kujul ja selles tabelis määras 12iga keerulise ja ebatavalise ühenduse sõltuvus selle tabeli potentsiaalse võtmega.[35]

Domeenivõtme normaalvorm (DKNF): Sellel kujul peab tabel olema viiendal kujul ja et iga tabelile kehtestatud piirang on loogiline tagajärg teatud väärtuse atribuudile kehtestatud piirangutele , mis on kehtestatud sellele atribuudile, mis näivad olevat kandidaatvõti.[36]

Kuues normaalvorm (6NF): Sellisel kujul olevat tabelit ei saa ilma kadudeta edasi lagundada.[37]

## 1.2. NoSQL andmebaas

NoSQL-andmebaase nimetatakse ka mitterelatsioonilisteks andmebaasideks, mis salvestavad andmeid muus vormingus kui relatsioonitabelid, kuid NoSQL-andmebaasid suudavad käsitleda tohutul hulgal kiiresti muutuvaid struktureerimata andmeid. NoSQL-i andmebaasidest saab aga päringuid teha idiomaatiliste keelte API-de, deklaratiivsete struktureeritud päringukeelte ja näidispäringukeelte abil. NoSQL-i andmebaasi peamine eelis on skaleeritavus ja paindlikud skeemid. Kui relatsiooni andmebaasid skaleeruvad vertikaalselt, siis NoSQL-i andmebaasid horisontaalselt. See tähendab, et andmete suurenedes lisatakse täiendavaid andmebaase või jaotatakse andmebaas väiksematele sõlmedele.

***Relatsioonilise andmebaasi haldussüsteem on aluseks SQL-keelele, mis võimaldab kasutajatel kõrgelt struktureeritud tabelites andmetele juurde pääseda ja neid töödelda. See on andmebaasisüsteemide, nagu MS SQL Server, IBM DB2, Oracle ja MySQL, aluseks olev mudel. Kuid NoSQL-i andmebaaside puhul võib andmetele juurdepääsu süntaks erinevates andmebaasides olla erinev.***

Erinevalt relatsioonilistest andmebaasihaldussüsteemidest saavad NoSQL-i andmebaasid andmeid salvestada ilma eelnevalt skeemi määratlemata, võimaldades teil kiiresti navigeerida ja itereerida, määratledes andmemudeli töö käigus. See lähenemisviis võib olla kasulik konkreetsete ärinõuete nt graafikud, veerud, dokumendid täitmiseks või võtmeväärtuste hoidlana kasutamiseks.

NoSQL-i andmebaase on sõltuvalt nende andmemudelist erinevat tüüpi. Peamised tüübid on dokument, võtmeväärtus, kolonn-perekond ja graafik.



Joonis 1.5 Erinevus relatsioonilise ja NoSQL andmebaasi vahel



Joonis 1.6 Tüübid NoSQL andmebaas

Plussid:

* ***NoSQL-i andmebaasid on väga paindlikud, kuna need võivad salvestada ja kombineerida mis tahes tüüpi andmeid, nii struktureeritud kui ka struktureerimata, erinevalt relatsioonilistest andmebaasidest, mis suudavad andmeid salvestada ainult struktureeritud viisil.***
* NoSQL-i andmebaasid võimaldavad teil skeemi dünaamiliselt värskendada, et see muutuvate nõuetega muutuks, tagades samal ajal, et see ei põhjusta teie rakendusele katkestusi ega seisakuid.
* Selle asemel, et suurendada serverite lisamisega, saavad NoSQL-i andmebaasid standardse riistvaraga horisontaalselt skaleerida. Scale-out võimaldab NoSQL-i andmebaasidel saada suuremaks ja võimsamaks, nii et need on eelistatud valik andmekogumite kasvatamiseks.
* NoSQL-i andmebaasid kopeerivad andmeid automaatselt mitme serveri, andmekeskuse või pilveressursside vahel. See omakorda vähendab kasutajate latentsust olenemata nende asukohast. See funktsioon aitab vähendada ka andmebaasihalduse koormust, mis vabastab aega ja võimaldab keskenduda muudele prioriteetidele.
* NoSQL-i andmebaasid on mõeldud hajutatud andmesalvedele, mis nõuavad suurt salvestusruumi. Just see teeb NoSQL-ist ideaalse valiku suurandmete ja reaalajas veebirakenduste jaoks.[8]

Miinused:

* Puudub standard, mis määratleks NoSQL-i andmebaaside reeglid ja rollid. NoSQL-i andmebaaside kujundus- ja päringukeeled on erinevate NoSQL-toodete puhul väga erinevad – palju laiemalt kui traditsiooniliste SQL-andmebaaside puhul.
* Varukoopiad on NoSQL-i andmebaaside puuduseks. Kuigi mõned NoSQL-i andmebaasid, nagu MongoDB, pakuvad varundamiseks mõningaid tööriistu, pole need tööriistad piisavalt küpsed, et tagada andmete täielik varunduslahendus.
* NoSQL seab esikohale skaleeritavuse ja jõudluse, kuid kui rääkida andmete järjepidevusest, siis NoSQL ei võta palju arvesse, nii et see muudab selle relatsiooniandmebaasiga võrreldes vähe ebaturvaliseks, nt NoSQL-i andmebaasides, kui sisestate sama andmekomplekti uuesti, see võtab seda ilma vigadeta, samas kui relatsiooniandmebaasid tagavad, et andmebaasidesse ei satuks dubleerivaid ridu. [12]

### **1.2.1. Tüüpid NoSQL andmebaasi**

Graafiku andmebaas (Graph database) on tüüpi andmebaas, mis salvestab andmeid sõlmedevaheliste linkide abil. Sõlmed on graafiku andmepunktid, mis on andmetega ühendatud servade kaudu ja serva kasutatakse andmete vaheliste suhete salvestamiseks.[2] Serval on alati algussõlm, lõppsõlm, tüüp ja suund. Servad saavad kirjeldada vanemate ja laste suhteid, toiminguid, omandiõigust jne.[38]

Need andmebaasid on väga paindlikud, mistõttu on andmete linkimine lihtne, kuid neil on halb skaleeritavus.[13] Graafikandmebaasid sobivad sotsiaalvõrgustike loomiseks ja saavad hakkama ka finantstehingutega.[3][38]

Näiteks: Infinite Graph, TITAN, Sparksee, GraphBase, ArangoDB, Neo4J.[14]



Joonis 1.7 Graafikuandmebaasi näiteks

Dokumenti andmebaas (Document database) on andmetüüp, mis salvestab andmed JSON- või XML-dokumentides. Dokumente hoitakse hoidlates ja need ei vaja skeeme. Kui dokumente lisatakse hoidlatesse, sisestatakse need andmebaasi indeksitesse, mis võimaldab kiiret otsingut ka keerulise salvestusstruktuuriga. Kuna andmebaas ei vaja skeemi, võib iga üksik dokument koosneda suvalisest arvust unikaalsetest väljadest, mis võib põhjustaa skaleerimise probleeme. Dokumendiandmebaasid näitavad end hästi teabe salvestamisel kataloogi, näiteks loovad elektroonikaseadmete katalooge veebipoes. [3][4][10]

Näiteks: CouchDB, Couchbase, MongoDB, ArangoDB, Elastic. [14]



Joonis 1.8 Dokumenti andmebaasi näiteks

Võtmeväärtuse andmebaas (Key-value database) on tüüpi andmebaas, milles igale väärtusele on määratud oma kordumatu võti. Selline andmebaas ei pea looma skeeme ja väärtuste vahelisi seoseid. Sellistel andmebaasidel on väga hea skaleeritavus, mis võimaldab töötada suurte andmemahtudega. Sellised andmebaasid pole aga mõeldud kiireteks otsinguteks, kuna see nõuab kogu kollektsiooni skannimist või üksikute indeksiväärtuste loomist. Selliseid andmebaase kasutatakse objektide vahemäludena.[3][15]

Näiteks: ArangoDB, Redis, DynamoDB, Couchbase, Aerospike. [14]



Joonis 1.9 Võtmeväärtuse andmebassi näiteks

Veerupere või laia veeruga andmebaas (Column family database) on tüüp, kus andmed salvestatakse veergudesse, mis seejärel kombineeritakse alamrühmadesse. Erinevalt relatsiooniandmebaasidest, kus tabelid on skeemide sees lingitud, saab laiades veergudes salvestada iga veeru eraldi või kui on sarnased seotud veerud, siis kombineeritakse need veeruperekondadeks, misjärel salvestatakse need üksteisest eraldi.

Veeruperekond on jagatud kahte tüüpi. See on standardne veerg, milles on võtmeveerg ja väärtustega veerud (sarnaselt tabeliga). Ja superveerud, kus igal superveerul on nimi ja väärtus, mis kaardistab superveeru mitme erineva veeruga. Sellistel andmebaasidel on hea skaleeritavus, kuid need ei ole võrgutehingute töötlemisel kuigi tõhusad.[5] Neid kasutatakse konkreetse ülesande jaoks, näiteks mitme miljoni artikli pealkirjade loetlemiseks.[13]

Näiteks: Apache Flink, Hypertable, Cassandra, MonetDB. [14]



Joonis 1.10 Kolonn-perekonna andmebaasi näiteks

## 1.3. Teised andmebasside haldussüsteemid

Residentide andmebaas (mälusisene andmebaas (in-memory database) või põhimälu andmebaas (main memory database)) on andmebaasihaldussüsteem, mis salvestab arvuti andmed arvuti põhimällu, mitte kettale, et tagada kiirem reageerimisaeg. Sellised andmebaasid on kiiremad kui kettale mälu salvestavad andmebaasid, kuna juurdepääs kettale on palju aeglasem ja sisemised optimeerimisalgoritmid on lihtsamad ja täidavad vähem protsessori käske.

Andmed laaditakse alla tihendatud ja mitterelatsioonilises vormingus ning need on vormingus, mida on lihtne otse kasutada ilma tihendamise või krüptimise takistusteta. See võimaldab otsest navigeerimist registrist reale või veergu ja on kirjutuskaitstud süsteem.

Kuigi elanike andmebaas suudab kiiresti teavet kuvada, on sellel märkimisväärne puudus. Krahhi või ühenduse katkemise korral lähevad andmed kaotsi. Andmete kadumise vältimiseks peate tegema andmetest hetktõmmiseid, kirjutama tehingud kõvakettale, installima NVRAM ja kasutama välkmälu, kuid sellel on välkmälu kustutamise ja ülekirjutamise arv piiratud. mälu. Residentide andmebaas on leidnud tee näiteks pankades enne laenu väljastamist klienti analüüsima, kindlustusseltsides riske arvutama või interaktiivseid online mänge, sest võimaldab reaalajas andmeid analüüsida ja aruandeid koostada. [20][21][22]

Näiteks: Redis, Apache Ignite, Tarantool. [14]

Mitme mudeliga andmebaas (Multi-model database) on haldussüsteem, mis ühendab mitut tüüpi andmebaase ühe taustaprogrammiga ning salvestab, pärib ja indekseerib erinevate mudelite andmeid. Mitme mudeliga andmebaasid pakuvad mitmekeelse püsivuse modelleerimise eeliseid, ilma et peaks leidma võimalusi erinevate mudelite kombineerimiseks. Paindlik lähenemine võimaldab salvestada andmeid erineval viisil.

Mitme mudeliga andmebaasid võivad teisendada andmeid ühest vormingust teise, mis annab täiendavat paindlikkust ja hõlbustab konkreetsete projektinõuete täitmist.[23]

Näiteks: ArangoDB, OrientDB, FoundationDB. [14]

Objektorienteeritud andmebaas (Object Oriented Database) on andmebaasihaldussüsteem, kus andmed on esitatud objektidena, mida kasutatakse objektorienteeritud programmeerimiskeeltes. Igal objektil on oma omadused ja meetodid ning seda hallatakse koos objektorienteeritud programmeerimiskeelega, et hõlbustada objektorienteeritud andmete salvestamist ja otsimist. Objektid moodustuvad lihtsatest objektidest tänu konstruktoritele. Lihtsamad objektid on: numbrid, sümbolid, suvalise pikkusega märgistringid, tõeväärtuslikud muutujad jne. Komplekssed objektikonstruktorid on: korteežid, hulgad, multihulgad, loendid ja massiivid. Komplektid esindavad reaalse maailma objektide kogusid loomulikul viisil. Kordad pakuvad viisi olemite atribuutide esitamiseks. Loendid ja massiivid esindavad reaalse maailma objektide elamiskõlblikkust ning esindavad ka laia valikut matemaatilisi objekte.

Objektorienteeritud andmebaasid toetavad kapseldamist, mis peidab andmeid, ja programmeerimiskoode andmetega manipuleerimiseks. Objekt on jagatud liidese- ja teostusosadeks. Liidese osa on objektil lubatud toimingute komplekti spetsifikatsioon ja see objekti osa on nähtav teiste objektide meetoditele. Rakendusosa koosneb andmetest, mis kirjeldavad objekti olekut, ja protseduuridest, mis teostavad objektil toiminguid. Samuti on olemas pärimise tugi, mis võimaldab luua uusi klasse kasutades teiste klasside andmeid ja meetodeid.

Üks peamisi eeliseid relatsioonilise andmebaasihaldussüsteemi ees on see, et objektorienteeritud andmebaasid suudavad luua kiireid seoseid keeruliste andmetega ja keeruliste seostega, kuna andmebaasi struktuur on programmeerimisobjektidele väga lähedane. Kuid kuna igal objektorienteeritud programmeerimiskeelel on oma päringu süntaks ja puuduvad üldtunnustatud standardid, peate erinevalt SQL-i kasutavast relatsioonilise andmebaasi haldussüsteemist kohanema iga keelega.[25][26][27]

Näiteks: db4o, ZODB, ObjectDB. [14]

# **KASUTATUD KIRJANDUSE LOETELU**

1. <https://www.oracle.com/cis/database/what-is-database/#WhatIsDBMS> (veebiartikkel)
2.
3. <https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases> (veebiartikkel)
4.
5. <https://www.oracle.com/cis/database/what-is-a-relational-database/> (veebiartikkel)
6. <https://www.oracle.com/cis/database/nosql/what-is-nosql/> (veebiartikkel)
7. <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html> (veebiartikkel)
8. <https://en.wikipedia.org/wiki/Document-oriented_database> (veebiartikkel)
9. <https://databasetown.com/relational-database-benefits-and-limitations/> (veebiartikkel)
10. <https://technologypoint.in/advantages-and-disadvantages-of-nosql-databases/> (veebiartikkel)
11.
12. <https://hostingdata.co.uk/nosql-database/> (veebiartikkel)
13. <https://www.freecodecamp.org/news/the-pros-and-cons-of-different-data-formats-key-values-vs-tuples-f526ad3fa964/> (veebiartikkel)
14.
15. <https://www.youtube.com/watch?v=pYRPYfqWiUo&t=223s> (video)
16.
17. <https://www.guru99.com/database-normalization.html> (veebiartikkel)
18. <https://www.heavy.ai/technical-glossary/in-memory-database> (veebiartikkel)
19.
20. <https://en.wikipedia.org/wiki/In-memory_database> (veebiartikkel)
21. <https://phoenixnap.com/kb/multi-model-database> (veebiartikkel)
22. <https://euroteq.eurotech-universities.eu/> (veebiartikkel)
23. [https://elearning.sumdu.edu.ua/free\_content/lectured:eaeb836f0e4d45b9a509d6c3f79a9ecf6ab05395/20141129091116/49214/index.html](https://elearning.sumdu.edu.ua/free_content/lectured%3Aeaeb836f0e4d45b9a509d6c3f79a9ecf6ab05395/20141129091116/49214/index.html) (veebilehekülg loengu)
24. <https://www.mongodb.com/databases/what-is-an-object-oriented-database> (veebiartikkel)
25.
26. <http://mech.math.msu.su/~shvetz/54/inf/databases/chApplications.xhtml> (veebillehekülg)
27. <https://www.pragimtech.com/blog/contribute/article_images/2220211210231003/what-is-a-relational-database.jpg> (pild)
28. <https://media.geeksforgeeks.org/wp-content/uploads/20220405112418/NoSQLDatabases.jpg> (pild)
29. [https://miro.medium.com/max/960/1\*A-ggJLL9Q\_w1zt87RV8uSg.jpeg](https://miro.medium.com/max/960/1%2AA-ggJLL9Q_w1zt87RV8uSg.jpeg) (pild)
30. [https://miro.medium.com/max/768/1\*Jhy4UUszi7neyQFhfT6Eow.png](https://miro.medium.com/max/768/1%2AJhy4UUszi7neyQFhfT6Eow.png) (pild)
31. <https://media.geeksforgeeks.org/wp-content/uploads/20200522103130/Document3.png> (pild)
32. <https://www.influxdata.com/wp-content/uploads/Key-Value-Database.jpg> (pild)