Harri Laine

Tietokantojen perusteet

Opetusmoniste D404

Helsingin yliopisto

Tietojenkäsittelytieteen laitos

11.1. 2000

1 Johdanto

Tietokanta (database) voidaan karkeasti määritellä jotakin käyttötarkoitusta varten laadituksi kokoelmaksi toisiinsa liittyviä säilytettäviä tietoja. Tämän määritelmän mukaan mitä tahansa säilytettävää tietokokoelmaa voidaan pitää tietokantana. Usein tietokantaan liitetään kuitenkin sen teknisiin ominaisuuksiin liittyviä lisävaateita, joiden perusteella tietokanta eroaa perinteisesti ohjelmointikielissä tarjolla olevasta tiedostosta (file). Seuraavassa käsitellään muutamia tällaisia ominaisuuksia.

1.1 Tiedon riippumattomuus niitä käsittelevistä ohjelmista

Samoja tietoja voidaan käyttää moneen tarkoitukseen. Kutakin tarkoitusta varten voisi olla perusteltua laatia eri ohjelma, esimerkiksi pankkiautomaatissa pankkitilejä käsitellään eri ohjelmalla kuin mitä pankkitoimihenkilöt käyttävät pankkipäätteillään. Voisi olla myös tarkoituksenmukaista käyttää eri ohjelmointikieliä näiden ohjelmien toteutukseen. Ohjelmointikielissä tiedostoa käyttävä ohjelma määrittelee tarvitsemansa tiedostot. Ohjelmointikielten näkemykset talletetusta tiedosta poikkeavat kuitenkin toisistaan, esimerkiksi C-kielen tiedostomäärittely on erilainen kuin Javan. Jotta useiden ohjelmointikielten käyttö saman aineiston käsittelyyn olisi mahdollista, pitäisi tiedot kyetä määrittelemään riippumattomasti ohjelmista ja ohjelmointikielistä. Tietokannoissa riippumattomuus saadaan aikaan siten, että tietokanta määritellään erityisen tietokantakaavion (database schema) avulla. Määrittely on ohjelmariippumaton (ei sisälly mihinkään sovellusohjelmista) ja joidenkin tietokantatyyppien kohdalla myös ohjelmointikieliriippumaton.

Riippumattomuutta tietojen ja ohjelmien välillä voidaan tarkastella myös toiselta kannalta: ohjelmien riippuvuutena niiden käsittelemistä tiedoista. Jos vaikkapa haluaisimme lisätä uuden tietoalkion kortistoon, meidän pitää tietenkin muuttaa niitä ohjelmia, jotka käsittelevät tuota uutta tietoalkiota. Mutta täytyykö meidän muuttaa ja kääntää uudelleen myös ne ohjelmat, jotka eivät tuota uutta alkiota käytä? Ohjelmien

tietoriippumattomuudella tarkoitetaan sitä, että ohjelmaa ei tarvitse muuttaa, ellei sen itse käyttämien tietojen rakenteissa tapahdu muutoksia.

1.2 Tietojen samanaikainen käyttö

Tekstinkäsittelyssä muokattava teksti ladataan kerralla apumuistista keskusmuistiin. Edistyksellisessä ympäristössä käyttöjärjestelmä huolehtii siitä, että kukaan ei pääse muokkaamaan tekstiä sillä aikaa kun se on jonkun toisen käyttäjän tai saman käyttäjän jossain toisessa ikkunassa muokattavana. Suurissa tietovarastoissa ei voi kuitenkaan harrastaa koko tietovaraston kerralla yhdelle käyttäjälle luovuttavaa käsittelyperiaatetta. Esimerkiksi pankin tilitietorekisteriin täytyy olla samanaikainen pääsy useammalla kuin yhdellä pankkitoimihenkilöllä. Käyttöjärjestelmien tarjoama tuki samanaikaiselle käytölle rajoittuu yleensä koko tiedoston varaamiseen yksinoikeutettuun käyttöön (lukitsemiseen). Tietokannat tarjoavat huomattavasti edistyneemmän samanaikaisuuden hallinnan, jossa pyritään yleensä tietojen mahdollisimman laajaan rinnakkaiseen käyttöön samalla kuitenkin minimoiden ne ongelmat, joita tämä tuo mukanaan.

1.3 Monipuoliset tiedonhakumahdollisuudet

Perinteisesti ohjelmointikielten tarjoamat mahdollisuudet tiedon osoittamiseen perustuvat siihen, että ohjelma käy peräkkäin läpi jotain tiedostoa tai hakee tietoa jostakin tietystä kohdasta tiedostoa. Tietokannat tarjoavat mahdollisuuden tiedon sisällön perusteella tapahtuvaan kohteiden valintaan ja käsittelyyn. Käsittelyyn voidaan esimerkiksi valita suoraan tietyn asiakkaan pankkitilit ilman, että ohjelmaan täytyy rakentaa silmukka, jossa nämä tilit käydään yksitellen poimimassa kaikkien pankin tilien joukosta.

1.4 Tietojen suojaus

Useimmat käyttöjärjestelmät suojaavat tietoja tiedosto- ja hakemistotasolla. Esimerkiksi UNIX-järjestelmässä tiedoston luku- ja kirjoitusoikeudet voidaan rajata vain käyttäjälle itselleen, tietylle ryhmälle tai sallia kaikille. Tällainen karkea suojaus ei aina riitä. Tietojen käyttöä voi olla tarpeen rajata paljon monipuolisemmin. Esimerkiksi opiskelija näkisi omat suoritustietonsa, mutta ei muiden opiskelijoiden suori-

tustietoja, johtaja näkisi omien alaistensa palkat, mutta ei esimiestensä palkkoja. Luentokurssien kohdalla jotkut kurssin liittyvät tiedot voisivat olla kaikkien luettavissa, osa rajautuisi vain opetus- ja hallintohenkilöstölle, osa vain kurssin vastuuhenkilölle. Tietokannat mahdollistavat tällaiset käyttörajoitukset ja vievät suojauksen huomattavasti käyttöjärjestelmiä tarkemmalle tasolle.

1.5 Automaattinen varmistus ja elpyminen häiriöistä

Varmistuskopion laatiminen tiedostosta aika ajoin takaa sen, ettei kovin suurta määrää tiedoston ylläpitoon käytettyä työtä mene hukkaan levyhäiriön sattuessa. Varmistustalletusten tekeminen esimerkiksi tekstiä muokattessa puolestaan takaa sen, että keskusmuistissa olevaa muokattua materiaalia ei katoa järjestelmän kaatumisen aiheuttavien häiriöiden yhteydessä. Yleensä varmistuskopion laatiminen estää tiedoston käytön kunnes kopio on valmis. Jos tiedostolla on useita käyttäjiä, voi olla vaikea löytää ajankohtaa varmistuskopion tekemiselle. Samoin käy, jos tiedosto on niin suuri, ettei sen käytössä löydy riittävän pitkää katkoaikaa kopion tekemiselle. Tietokantojen kohdalla järjestelmät pyrkivät takaaman sen, että käyttäjän suorittama onnistuneesti loppuunviety (vahvistettu / committed) operaatio jättää pysyvän jälkensä tietokantaan ilman, että käyttäjien täytyy kantaa tästä huolta.

1.6 Suuret ja kasvavat tietomäärät

Joissakin järjestelmissä käsiteltävät tietomäärät voivat olla hyvin suuria. Tällaisten käsittely edellyttää tehokkaita talletusrakenteita. Tietosisältö voi myös olla jatkuvasti kasvava tai muutenkin dynaamisesti muuttuva, jolloin talletusrakenteiden täytyy joustavasti mukautua muutoksiin. Talletusrakennetta pitää esimerkiksi pystyä virittämään paremman tehon aikaansaamiseksi eikä virittämisen pitäisi vaatia mitään muutoksia rakenteita käyttäviin ohjelmiin.

1.7 Mutkikkaat riippuvuudet tietojen välillä

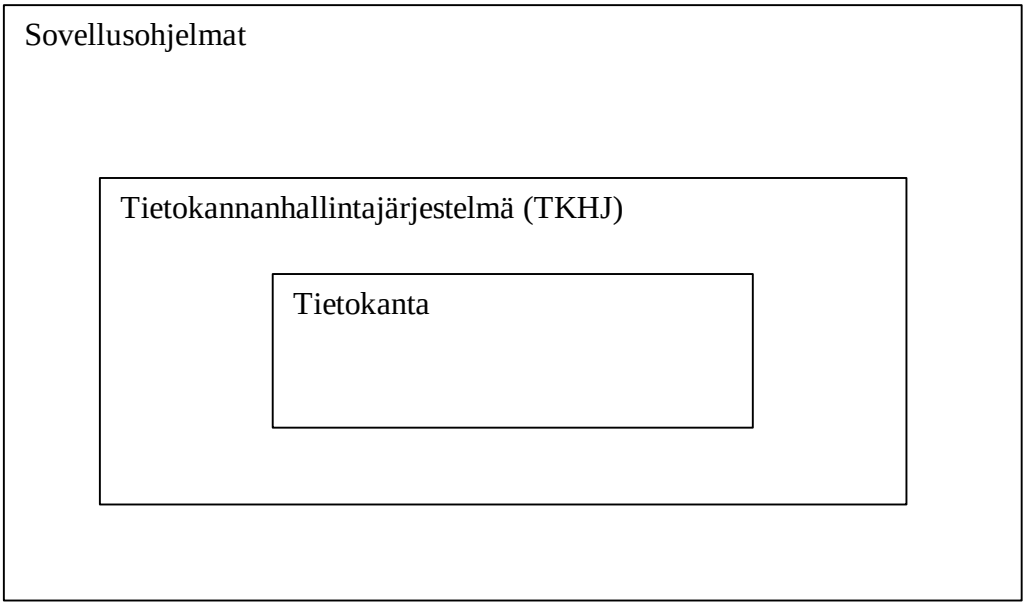
Tietojen välillä on usein erilaisia kytkentöjä ja riippuvuuksia. Nämä heijastavat niitä sääntöjä ja lainalaisuuksia, jotka ovat voimassa tietojen kuvaamalla kohdealueella. Tietojen oikeellisuuden kannalta on välttämätöntä, että näitä riippuvuuksia valvotaan. Riippuvuuksien valvontaa ei voi jättää yksittäisten tietoja käsittelevien ohjelmien vas-

tuulle, koska näitä tuottavat eri ohjelmoijat ja on varsin todennäköistä, että tarkistukset eivät toteudu samanlaisina jokaisessa ohjelmassa.

2 Tietokantajärjestelmän arkkitehtuuri

2.1 Tietokantajärjestelmä

Tietokantajärjestelmä (database system, DBS) muodostuu tietokannasta (database), tietokannanhallintajärjestelmästä (database management system, DBMS) ja tietokantaa käyttävistä sovellusohjelmista (application program) (kuva 2.1)



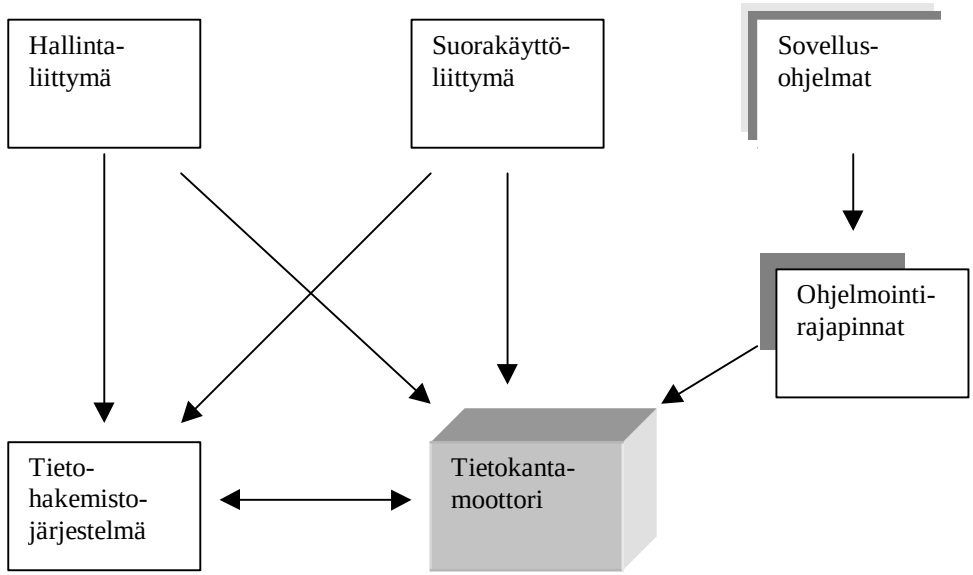
Kuva 2.1: Tietokantajärjestelmän kerrokset

Yllä esitetyssä karkeassa mallissa ei ole huomioitu tietoliikennettä ja sen hoitamiseen tarvittavaa tietoliikenneohjelmistoa. Sovellusohjelmat voivat kommunikoida tietokannanhallintajärjestelmän kanssa joko suoraan tai tietoliikenneohjelmiston välityksellä. Samoin tietokannanhallintajärjestelmä saattaa sisäisesti käyttää hyödykseen tietoliikenneohjelmiston palveluita.

2.2 Tietokannanhallintajärjestelmä

Tietokannanhallintajärjestelmä tarjoaa erilaisia palveluita tietokannan käsittelyyn (kuva 2.2):

- Suorakayttöliittymien (query interface) avulla käyttäjä voi tehdä tietokantaan kohdistuvia kyselyitä ja muokkausoperaatioita
- Hallintaliittymän (administration interface) avulla voidaan muokata tietokannan rakenteita, hallita käyttöoikeuksia ja säädellä tietokannan asetuksia.
- Ohjelmointirajapinnat (application programming interface, API) mahdollistavat tietokannan käytön sovellusohjelmien kautta



Kuva 2.2: Tietokannanhallintajärjestelmän pääosat

Tietokannanhallintajärjestelmät ovat rakenteiltaan erilaisia. Muutamia perusosasia järjestelmät kuitenkin sisältävät. Yllä mainittuja ulkoisia liittymiä varten tarvitaan käyttöliittymät ja kääntäjiä. Tietokantajärjestelmän keskeisenä komponettina voidaan pitää tietokantamoottoria (database manager, database engine). Sen vastuulla on varsinaisten tietokantaan kohdistuvien operaatioiden suorittaminen Toinen keskeinen osa on tietohakemistojärjestelmä (data dictionary system), jonka tehtävänä on hallita kaikkea tietokantaan liittyvää kuvaustietoa, ns. metatietoa. Tietohakemistojärjestelmä käyttää yleensä tietokantamoottoria ja tietokantaa itseään kuvaustiedon tallennukseen.

Tietokantamoottori on varsin monimutkainen ja laaja ohjelmisto. Toiminnallisuuden perusteella sen osia voisivat olla

- Pääsyn valvonta (authorization control), jonka tehtävänä on tarkastaa käyttäjien oikeudet heidän haluamiinsa operaatioihin.
- Kyselyn optimoija (query optimizer), jonka tehtävänä on laatia toteutussuunnitelma halutuille tietokantaoperaatiolle.
- Transaktion hallinta (transaction manager), jonka tehtävänä on valvoa samanaikaisia operaatioita sekä varata ja vapauttaa resursseja, huolehtia operaatioiden päättymisistä ja mahdollisista peruutuksista.
- Eheyden valvonta (integrity control), jonka tehtävänä on valvoa, etteivät tietokantaan kohdistuvat muutokset riko tietokannalle määriteltyjä oikeellisuussääntöjä.
- Suorittaja (command processor), joka ohjaa operaation suoritusta
- Puskurien hallinta (buffer management), jonka vastaa keskusmuistin ja apumuistin välisestä tiedonsiirrosta.
- Hakumenetelmät (access methods), joiden tehtävänä on suorittaa tiedonhaut toteutussuunnitelman mukaisesti

Tällä kurssilla keskitytään lähinnä tietokannanhallintajärjestelmän ulkoiseen käyttäytymiseen ja edellä mainittuihin ulkoisiin liittymiin. Tietokantamoottorin sisäinen toiminta jää myöhemmille kursseille.

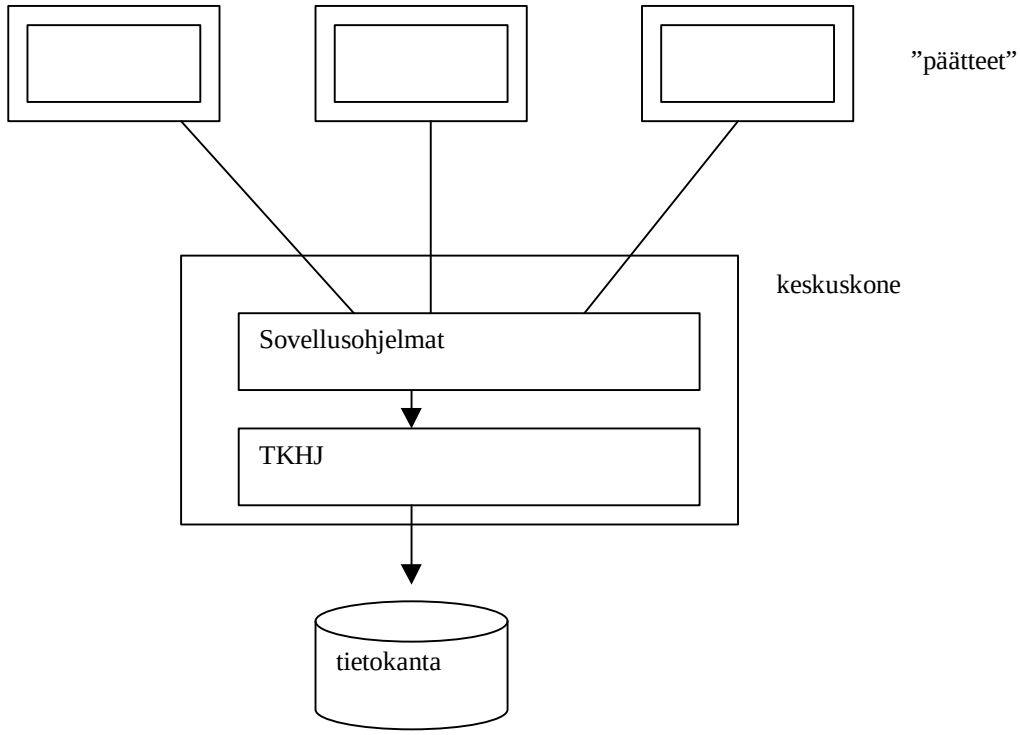
2.3 Tietokanta tietoverkossa

2.3.1 Henkilökohtainen tietokanta

Henkilökohtaiseen käyttöön tarkoitetussa järjestelmässä tietokanta, tietokannanhallintajärjestelmä, ja sovellusohjelmat voivat sijaita ja toimia samassa käyttäjän työasemassa.

2.3.2 Keskuskonemalli

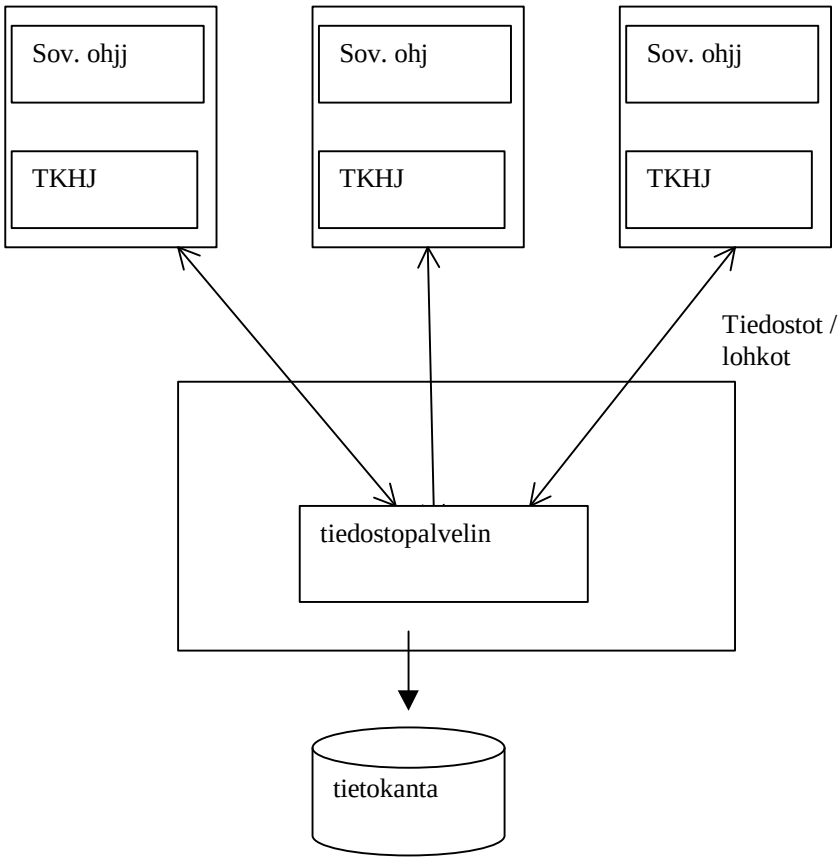
Vielä kymmenkunta vuotta sitten tietokantajärjestelmät olivat pääsääntöisesti keskuskonepohjaisia päätekäyttöisiä järjestelmiä. Tällaisessa järjestelmässä tietokanta sijaitsee keskuskoneen levyillä, tietokannanhallintajärjestelmä toimi kokonaisuudessaan samassa koneessa kuten myös kantaa käyttävät sovellusohjelmat (kuva 2.3). Tiedonvälitys tietokannanhallintajärjestelmän ja sovellusohjelmien välillä on ratkaisussa nopeaa, esimerkiksi yhteisiin muistialueisiin perustuvaa. Tällainen ratkaisu tietokantajärjestelmän perusarkkitehtuuriksi on edelleen mahdollinen. Tietokanta ja tietokannanhallintajärjestelmä sijaitsevat keskuskoneessa ja koneeseen liitetyt työasemat toimivat ’tyhminä’ päätteinä pelkästään näyttäen keskuskoneen tuottamia näyttöjä. Useat WWW-pohjaiset tietokantasovellukset toimivat nykyäänkin tämän mallin mukaisesti: tietokantapalvelimessa tuotetaan sivut, joita sitten näytetään WWW-selaimissa. Mallin etuna on se, että ohjelmat sijaitsevat yhdessä paikassa ja ovat siten helposti päivitettävissä. Jos käyttäjiä on runsaasti, asettaa tällainen malli suuret kapasiteettivaatimukset keskuskoneelle. Työasemilta sensijaan ei vaadita kovinkaan paljoa.



Kuva 2.3: Tietokanta keskuskoneessa

2.3.3 Tiedostopalvelin malli

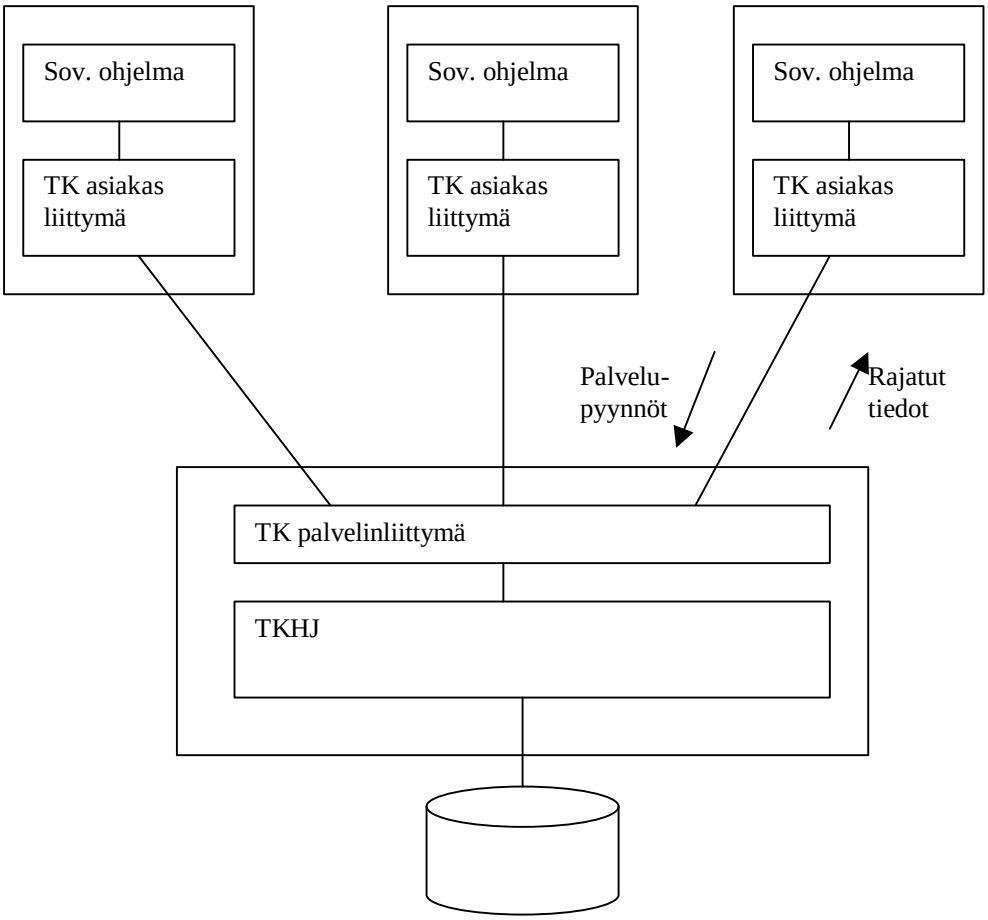
Nykyään työasemat on yleensä kytketty nopeaan lähiverkkoon. Verkossa on käytettävissä yksi tai useampia tiedostopalvelimia (file server), jotka tarjoavat talletuskapasitettia joko käyttäjän omien tai yhteiskäyttöisten tiedostojen säilytykseen. Tällaisessa ympäristössä tietokantajärjestelmä voidaan toteuttaa siten, että tietokanta sijaitsee tiedostopalvelimen levyillä ja tietokannanhallintajärjestelmä ja sovellusohjelmat toimivat jokaisessa kantaa käyttävässä työasemassa (kuva 2.4). Yleensä tällaisessa ratkaisussa tietokannanhallintajärjestelmänä käytetään halpoja työasemajärjestelmiä (esim. dBase, Access). Ratkaisu vähentää tiedostopalvelimen kuormaa koska sen tehtävänä on tarjota vain pääsy tiedostoihin. Käsittelyä varten tiedot on siirrettävä työasemiin. Niiltä vaaditaan puolestaan enemmän kapasiteettia. Ratkaisu kuormittaa myös tietoliikenneverkkoa, sillä siirrettävää tietoa ei voida rajata tiedon sisällön perusteella, koska kaikki valinnat tehdään työasemissa. Jokaisessa työasemassa tarvitaan kopio tietokannanhallintaohjelmistosta sekä sovellusohjelmista.



Kuva 2.4: Tiedostopalvelin malli

2.3.4 Asiakas-palvelin malli

Asiakas-palvelin mallissa (client- server) tietokannan käsittelyn työkuorma jaetaan tietokantapalvelimen (database server) ja työasemien välillä siten, että tietokanta ja tietokannanhallintajärjestelmä on sijoitettu tietokantapalvelimeen (kuva 2.5). Sovellusohjelmat toimivat työasemissa. Lisäksi sekä palvelimeen että työasemiin tarvitaan liittymäohjelmisto (esim. ODBC), joka mahdollistaa palvelupyyntöjen välityksen työasemista palvelimelle ja tulosten välityksen toiseen suuntaan. Halutessaan käsittelyyn joitain tietoja sovellusohjelma lähettää tietokantapalvelimelle kyseiset tiedot määrittelevän kyselyn ja saa vastauksena vain haluamansa tiedot, ei koko tiedostoa kuten tiedostopalvelinmallissa. Verkossa liikkuvan tiedon määrä vähenee. Sovellusohjelmat puolestaan vastaavat kokonaan käyttöliittymän toiminnasta. Tässä ratkaisussa palvelinkoneelta vaaditaan enemmän laskentakapasiteettia kuin tiedostopalvelinmallissa, mutta vähemmän kuin keskuskoneratkaisussa.



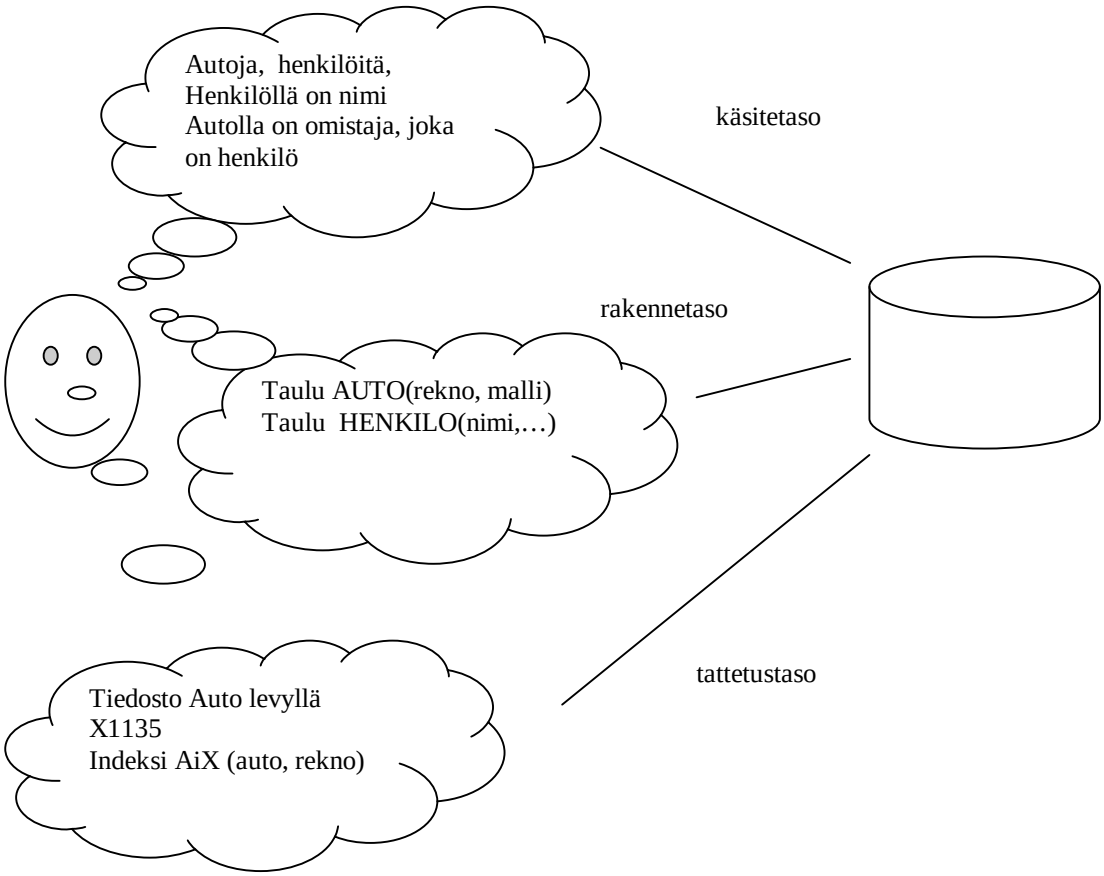
Kuva 2.5: Asiakas-palvelin malli

Asiakas-palvelin mallissa sovellukset voidaan toteuttaa myös siten, että sama asiakas käyttää useita tietokantapalvelimia. Nämä voivat näkyä asiakkaille erillisinä tietokantoina (monitietokanta, multi-database) tai yhtenä tietokantana (hajautettu tietokanta, distributed database). Tilanteissa, joissa halutaan tasata palvelinten kuormitusta voidaan kayttää ns. kolmitasoratkaisua (three-tier model), jossa asiakkaiden ja palvelinten väliin toteutetaan välikerros, joka toimii asiakkaana varsinaisille tietokantapalvelimille ja palvelimena suhteessa varsinaisiin asiakkaisiin.

3 Tietomallit

Tietokantaa ja sen sisältämiä tietoja voidaan tarkastella eri näkökulmista. Yleinen näkökulmajaottelu perustuu kolmeen abstraktiotasoon: käsitetasoon (conceptual level), rakennetasoon (logical level, structural level) ja talletustasoon (physical level, internal level) (kuva 3.1). Kuhunkin näkökulmaan liittyen on esitetty lukuisia tietomalleja (data model) eli käsitteistöjä, joihin pohjautuen tietokanta kyseisellä abstraktiotasolla kuvataan.

Käsitetason abstraktiossa tietokannasta pyritään määrittelemään sen sisältö: mitä tietoja tietokannassa on ja miten nämä liittyvät toisiinsa. Kuvauksen tietosisällön tulisi olla riippumaton kaikista tietokanta- ja ohjelmointitekniikoista. Tällainen kuvaus saadaan aikaan esimerkiksi laatimalla malli siitä kohdealueesta, mitä tietokannan tiedoilla on tarkoitus kuvata. Tunnetuimpia tietomalleja tietojen kuvaamiseen käsitetasolla ovat ER-malliperheen mallit (Entity-Relationship models, ER-models) [Chen76] ja nykyään oliomallit [BRJ98]. Tämän näkökulman mukaisia kuvauksia tarkastellaan kurssilla ‘Johdatus sovellussuunnitteluun’.



Kuva 3.1: Abstraktiotasot

Rakennetason abstraktiossa tietokantaa tarkastellaan lähinnä ohjelmoijan tai tietokannan suorakäyttäjän kannalta: millaisina rakenteina hän tietokannan hahmottaa. Tämän tason tietomalleista tunnetuin on relaatiomalli (the relational model of data), jonka mukaisia tietokantoja tällä kurssilla pääasiassa tarkastellaan. Jo enimmäkseen historiaan vaipuneita 70 ja 80 –lukujen tietokannanhallintajärjestelmissä käytettyjä tietomalleja ovat verkkomalli (network model) ja hierakkinen malli (hierarchical model).[EN99] Nykyisin rakennetason mallina on yleistynyt myös oliomalli.

Tallennustasolla kiinnostuksen kohteena on se, miten tietokanta on sijoitettu levylle, millaisia hakurakenteita käytetään, miten säädellään tietokantapuskureita, tilanvarauksia, jne. Kuvausvälineistö ja käsitteistö näiden esittämiseen on pitkälti järjestelmäkohtaista.

4 Relaatiotietokannat

Tunnetuin rakennetason tietomalli on relaatiomalli [Codd70]. Malliin pohjautuvat tietokannanhallintajärjestelmät ovat ylivoimaisesti laajimmin käytettyjä nykyisten tietokantojen toteutuksessa. Tunnetuimmat relaatiomalliin pohjautuvat tietokannanhallintajärjestelmät ovat IBM:n DB2, Oracle, Sybase, Informix ja Microsoftin SQL Server. Näiden lisäksi on markkinoilla lukuisia muita malliin pohjautuvia järjestelmiä.

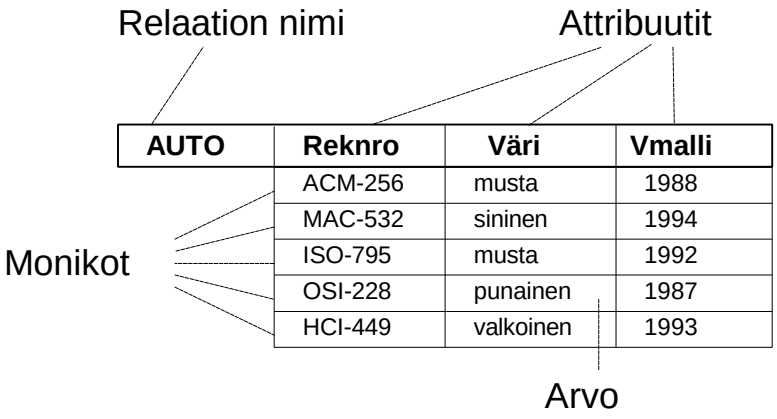
Relaatiomallin pohjana on joukko-oppi ja matematiikan relaatio-käsite. Näiden yksinkertaisten peruskäsitteiden pohjalta on määritelty operaatiot tietokannan käsittelyyn sekä kehitetty teorioita tietokannan suunnitteluun. Hyvä matemaattinen perusta on ollut yksi syy mallin saavuttamaan suosioon etenkin tutkijapiireissä. Yksinkertainen peruskäsitteistö puolestaan on helpottanut mallin ymmärtämistä ja lisännyt sen suosiota käyttäjien keskuudessa.

4.1 Relaatio

Relaatiomallissa tietokannan katsotaan koostuvan tietoalkiden muodostamista relaatioista (relation). Havainnollisesti relaatio voidaan esittää alla olevan kuvan (kuva 4.1) mukaisesti taulukkomuodossa tauluna (table)**.** Alkuperäisen määritelmä relaatiolle oli:

> *Olkoon D1, D2, ..., Dn arvojoukkoja (domain), joiden ei tarvitse olla erillisiä. Relaatio R on joukko monikkoja (tuple, n-tuple), joiden ensimmäinen arvo kuuluu joukkoon D1, 2. arvo joukkoon D2 jne. Relaatio on siis ristitulon* $D1 \times D2 \times ... \times Dn$ *osajoukko.*

Tässä määritelmässä arvojoukko on jokin atomisten arvojen joukko. Atomisella arvolla tarkoitetaan sellaista arvoa, mikä ei jakaudu enää pienempiin osiin.



Kuva 4.1: Relaation tauluesitys.

Joukkojen A={1,2,3) ja B={a,b} ristitulo (karteesinen tulo) A×B on kaikkien niiden parien (x,y) joukko, joissa ensimmäinen alkio kuuluu joukkoon A ja toinen joukkoon B eli joukko

{(1,a), (1,b), (2,a), (2,b), (3.a), (3,b)}.

Parien järjestyksellä joukossa ei ole merkitystä.

Kuvan 4.1 relaatiossa ensimmäiselle paikalle (sarakkeelle) on annettu nimi *Reknro*, toiselle *Väri* ja kolmannelle *Vmalli*. Paikan *Väri* arvojoukkoon kuuluvat ainakin arvot '*musta*', '*punainen*', '*sininen*' ja '*valkoinen*'. Arvojoukkoon voi kuulua muitakin värejä vaikka niitä kuvan relaatiossa ei esiinnykään. Itse asiassa arvojoukko pitäisi määritellä siten, että siihen kuuluvat kaikki mahdolliset autojen värit. Vastaavasti *Vmalli*:n arvojoukkoon kuuluisivat kaikki mahdolliset vuosiluvut, jotka tulevat kyseeseen auton vuosimallina ja *Reknro*:n arvojoukkoon vaikkapa kaikki mahdolliset suomalaiset rekisterinumerot. Merkitään paikkaan A liitettyä arvojoukkoa D(A):lla. Jos arvojoukot määriteltäisiin kuten yllä, voitaisiin ristitulon

$$D(\text{Reknro}) \times D(\text{Väri}) \times D(\text{Vmalli})$$

osajoukoilla esittää olemassaolevien rekisteröityjen suomalaisten autojen väri ja vuosimalli millä tahansa ajanhetkellä

Relaation paikoille voidaan antaa nimi, kuten yllä on tehty. Tällaista nimeä kutsutaan attribuutiksi (attribute). Tauluesityksessä luontevin nimitys attribuutille on sarakenimi (column name). Paikalle annetun nimen tulisi kuvata paikassa olevalla arvolla ilmaistavaa asiaa. Arvojoukko (domain) on tiettyyn attribuuttiin liittyvien mahdol-

listen arvojen joukko eli tietyssä sarakkeessa esiintyvien mahdollisten arvojen joukko. Myös arvojoukot voidaan tarvittaessa nimetä.

Relaation asteella (degree) ilmaistaan relaation attribuuttien (taulun sarakkeiden) lukumäärä. Unaarirelaatiossa sarakkeita olisi yksi, binäärirelaatiossa 2, kolmipaikkaisessa 3, jne. Relaation monikoiden (taulun rivien) määrää kutsutaan myös relaation kooksi (cardinality).

Relaatiokaavio (relation schema) määrittelee relaation rakenteen. Yksinkertaisimmillaan se voidaan esittää kaavion nimeen liitettynä attribuuttiluettelona , esimerkiksi

Auto(Reknro,Väri,Vmalli)

Attribuuttien järjestyksellä luettelossa ei ole mitään merkitystä.
Täydellisemmässä kaavioesityksessä mukaan otetaan myös arvojoukot, esimerkiksi

Auto(Reknro: Suomalaiset_rekisterinumerot,
Väri: Autovärit,
Vmalli: Vuosiluvut >1900)

Relaatio on relaatiokaavion ilmentymä (instance). Koska tietokannan sisältö tyypillisesti muuttuu ajan myötä, ovat saman relaatiokaavion eriaikaiset ilmentymät yleensä erilaisia.

Matemaattisessa joukossa tietty alkio esiintyy vain kerran. Koska relaatio on määritelmänsä mukaisesti matemaattinen joukko, seuraa edellisestä, että jokainen relaation monikko (taulun rivi) on erilainen. Matemaattisessa joukossa alkioiden järjestyksellä ei ole merkitystä., esimerkiksi {a,b,c,d} ≡ {b,d,a,c}. Täten myöskään relaation monikoiden järjestyksellä ei ole merkitystä.

4.1.1 Avain

Edellä todettiin, että kaikki relaation monikot ovat keskenään erilaisia. Monikot kyetään siis erottelemaan sisältönsä, niissä olevien arvojen, perusteella muista monikoista. Tällaiseen erottelun ei välttämättä tarvita edes kaikkia monikon arvoja vaan pelkästään joidenkin attribuuttien arvot. Kuvan 4.1 esimerkissä sekä attribuutin *Reknro* että attribuutin *Vmalli* arvot yksinään erottelevat (yksilöivät) taulun rivit, ts.

taulussa ei ole kahta riviä, jolla olisi sama arvo *Reknro* -sarakkeessa tai sama arvo *Vmalli* -sarakkeessa.

Relaation avain (key) on sellainen attribuutti tai niiden yhdistelmä, jolle pätee

- erilaisuus, eli missään relaatiokaavion ilmentymässä ei voi olla kahta tai useampaa riviä, joilla avainattribuuttien arvot ovat samat
- minimaalisuus, eli avain muodostuu mahdollisimman pienestä joukosta attribuutteja, ts. avainattribuuttien mikään osajoukko ei ole avain. Tämä tarkoittaa sitä, että avaimesta ei voi poistaa jotain attribuuttia ja jäljelle jääneiden attribuuttien arvot yhä erottelisivat relaation monikot.

Kuvan 4.1 esimerkissä attribuutti *Reknro* sopii avaimeksi. Sen määrittely avaimeksi tarkoittaa, että rekisterinumeroa (eli autoa) kohti taulussa on vain yksi rivi. *Vmalli*, vaikka se erotteleekin rivit kuvan esimerkissä, ei sensijaan tee sitä kaikissa relaatiokaavion ilmentymissä, koska saman vuosimallin autoja voi olla enemmän kuin yksi. *Vmalli* ei täten käy avaimeksi. Relaatiolla voi olla useita avainehdot täyttäviä attribuutteja tai niiden yhdistelmiä. Esimerkiksi yrityksen työntekijärekisterissä, *Työntekijä* relaatiossa, sekä attribuutti *Henkilötunnus* että *Työntekijänumero* voisivat täyttää avainehdot. Relaation pääavain (primary key) on avainehdot täyttävien ehdokkaiden joukosta valittu rivien osoittamiseen ja niihin viittaamiseen ensisijaisesti käytettävä ehdokas. *Työntekijä*-relaatiossa voitaisiin valita informaatiota sisältämätön *Työntekijänumero* pääavaimeksi yksilösuojan suhteen arveluttavan *Henkilötunnuksen* asemesta. Jatkossa termiä avain käytetään pääavaimen synonyyminä. Relaatiokaaviossa pääavain voidaan merkitä alleviivaamalla sen muodostavat sarakkeet, esimerkiksi

Auto(Reknro, Väri, Vmalli) .

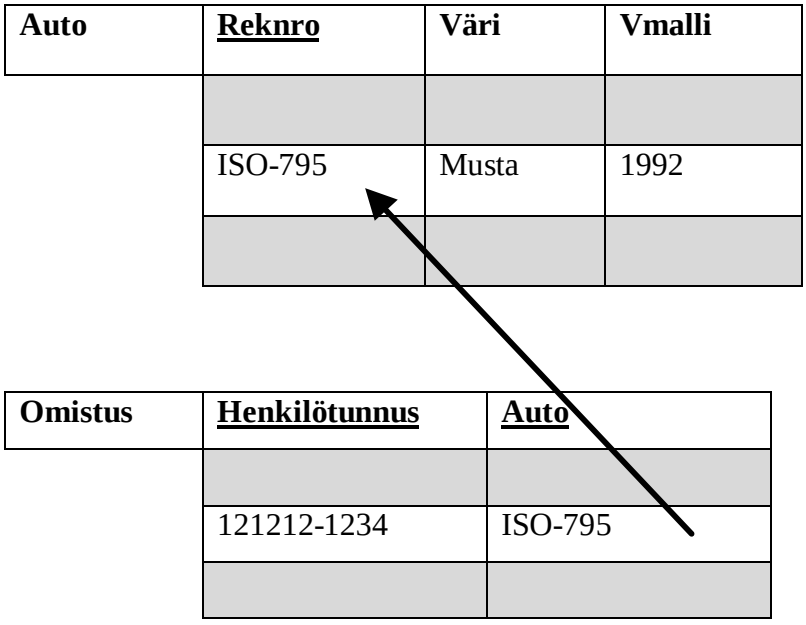
Aina ei ole mahdollista erotella rivejä pelkästään yhden attribuutin perusteella. Esimerkiksi relaation

PELIVARAUS (kenttänumero, alkuaika, kesto, nimi)

avaimeksi tarvitaan molemmat attribuutit *kenttänumero* ja *alkuaika*, koska kenttään voi kohdistua useita eriaikaisia varauksia (sama kenttänumero useilla riveillä) ja samaan aikaan voi olla varauksia eri kentille (sama alkuaika useilla riveillä).

4.1.2 Viiteavain

Tietokanta muodostuu yleensä useasta relaatiosta. Eri relaatiot eivät suinkaan ole toisistaan riippumattomia vaan relaatioiden tiedot kytkeytyvät usein yhteen. Kytkentä kahden monikon välille saadaan relaatiotietokannassa aikaan sisällyttämällä toisen monikon avainattribuuttien arvot toiseen monikkoon



Kuva 4.2: Yhteisellä arvolla yhteenkytketyt monikot.

Kuvassa 4.2 *Omistus*-relaation *Auto*-attribuutin arvona on relaation *Auto* avainattribuutin *Reknro* arvo. Attribuuttia tai attribuuttiyhdistelmää, jonka kaikki arvot ovat jonkin relaation avainattribuuttien arvoja kutsutaan viiteavaimiksi (foreign key). Attribuutti *Auto* relaatiossa *Omistus* on siis relaation *Auto* relaatioon *Omistus* kytkevä viiteavain. Relaatiomallissa viiteavain on ainoa tapa kytkeä relaatioita toisiinsa. Viiteavain voitaisiin perustaa mihin tahansa viitattavan relaation avaimeen, mutta

selkeintä on aina käyttää viittauksissa relaation pääavainta. Relaatiokaaviossa viiteavain voidaan esittää siten, että viiteavain-attribuutin perään sijoitetaan nuoli ja viitattavan relaation nimi. Esimerkiksi relaatiossa

Omistus(Henkilötunnus→Henkilö, Auto→Auto)

attribuutti *Henkilötunnus* on relaatioon *Henkilö* viittaava viiteavain ja attribuutti *Auto* on relaatioon *Auto* viittaava viiteavain. Jos viiteavain muodostuu useasta attribuutista nämä voidaan laittaa sulkeisiin ja ilmoittaa sulkeiden jälkeen viitattava relaatio [1], esimerkiksi

Osallistuu(Kuka→Opiskelija,
(Kurssikoodi, RyhmäNumero)→Harjoitustryhmä).

Tässä *Kurssikoodi* yhdessä *RyhmäNumeron* kanssa osoittaa harjoitusryhmän.

Edellä esitetyissä esimerkeissä viiteavaimet ovat sisältyneet viittaavan relaation pääavaimeen. Näin ei suinkaan tarvitse aina olla, esimerkiksi

Kurssi(Kurssikoodi, Luennoija →Opettaja).

Viiteavaimella voidaan kytkeä toisiinsa myös saman relaation eri monikoita, Esimerkiksi relaatiossa

Työntekijä(TyöntekijäNumero, …, Esimies→Työntekijä)

attribuutin *Esimies* arvona on kullakin rivillä jonkin toisen työntekijän työntekijänumero (kuva 4.3):

Työntekijä	TyöntekijäNumero	…	Esimies
	1010		
	1020		1010
	1030		1010

Kuva 4.3: Relaation sisäiset kytkennät

Viiteavaimen määrittelyyn liittyy vaatimus viite-eheydestä (referential integrity). Tällä tarkoitetaan sitä, että jokaisen viiteavain-attribuutin arvon on esiinnyttävä viitat-

[1] Tätä esitystekniikkaa voi käyttää vain jos viiteavaimen muodostavat attribuutit ovat relaatiokaaviossa peräkkäin, mikä on kaikin puolin suotavaa mutta ei toki välttämätöntä.

tavassa relaatiossa avainattribuutin arvona. Viiteavainsarakkeessa esiintyvien arvojen joukko on siis viitattavan taulun avainsarakkeessa esiintyvien arvojen joukon osajoukko. Joissakin tapauksissa sallitaan viiteavaimelle myös tyhjäarvo (null value) Tämä on erikoisarvo, joka tarkoittaa, että attribuutilla ei oikeasti ole arvoa (kuvan 4.3 ylin rivi). Jos viiteavaimella on tyhjäarvo ei monikkoa ole kytketty mihinkään toiseen monikkoon. Tietokannanhallintajärjestelmän edellytetään valvovan viite-eheyttä ja hylkäävän tai automaattisesti korjaavan tilanteet, joissa sitä yritetään rikkoa.

4.2 Relaatioalgebra

Relaatioalgebra muodostuu joukosta operaatioita, joilla relaatioista voidaan muodostaa uusia relaatioita. Alunperin operaatioita esiteltiin kahdeksan, myöhemmin niitä on esitelty lisää. Operaatiot voidaan kuitenkin ilmaista viiden perusoperaation valinnan, projektion, ristitulon, yhdisteen ja erotuksen avulla. Näistä yhdiste, erotus ja ristitulo ovat joukko-opin perusoperaatioita. Valinta ja projektio ovat relaatioalgebran omia lisäoperaatioita. Yleisesti käytettyjä johdettuja operaatioita ovat leikkaus ja liitos.

4.2.1 Perusoperaatiot

4.2.1.1 Yhdiste (union)

Yhdisteen avulla muodostetaan relaatio joka sisältää kummankin yhdistettävän relaation monikot:

R È **S** = **{ t** | **t** Î **R** Ú **t** Î **S }** missä R ja S ovat relaatioita ja t on monikko	*[eli niiden monikoiden t joukko, jotka kuuluvat relaatioon R tai relaatioon S]*

Mikäli sama monikko esiintyy kummassakin yhdistettävässä relaatiossa, se tulee mukaan tulosrelaatioon vain kertaalleen. Yhdiste edellyttää, että yhdistettävät relaatiot ovat samarakenteisia (union compatible) eli niillä on yhtä monta attribuuttia ja vastinattribuuteilla (samassa paikassa) on samat arvojoukot. Vastin-

attribuuttien ei tarvitse olla samannimisiä. Sovitaan, että tulosrelaation paikat nimetään yhdisteen ensimmäisen osapuolen mukaisesti.

Kuvassa 4.4 tarkastellaan esimerkkinä auton omistajia ja haltijoita:

Omistus	**Reknro**	**Henkilonumero**
	ISO-795	11111
	ACM-256	22222
	HCI-449	33333

Hallinta	**Reknro**	**Hnum**
	ACM-256	44444
	ACM-256	22222
	ACM-256	11111

Omistus È Hallinta	**Reknro**	**Henkilonumero**
	ISO-795	11111
	ACM-256	22222
	HCI-449	33333
	ACM-256	44444
	ACM-256	11111

Kuva 4.4: Yhdiste

4.2.1.2 Erotus (set difference)

Erotuksessa (kuva 4.5) tulosrelaatioon otetaan ne relaation monikot, jotka eivät sisälly erotettavaan relaatioon :

R - S = { t | t Î R Ù t Ï S }.	*t on R:n monikko ja t ei ole S:n monikko*

Myös erotuksessa relaatioiden pitää olla samarakenteisia.

Omistus - Hallinta	**Reknro**	**Henkilonumero**
	ISO-795	11111
	HCI-449	33333

Hallinta – Omistus	**Reknro**	**Hnum**
	ACM-256	44444
	ACM-256	11111

Kuva 4.5: Erotus

4.2.1.3 Ristitulo, karteesinen tulo (Cartesian product)

Ristitulossa R × S muodostetaan tulosrelaatioon monikoita, kokoamalla yhdeksi monikoksi arvot monikkopareista, joissa parin monikoista ensimmäinen kuuluu relaatioon R ja toinen relaatioon S. Yhdistetty monikko muodostetaan jokaisesta monikkoparista (kuva 4.6).

Omistus ´ Hallinta	**Omistus.Reknro**	**Henkilonumero**	**Hallinta.Reknro**	**Hnum**
	ISO-795	11111	ACM-256	44444
	ACM-256	22222	ACM-256	44444
	HCI-449	33333	ACM-256	44444
	ISO-795	11111	ACM-256	22222
	ACM-256	22222	ACM-256	22222
	HCI-449	33333	ACM-256	22222
	ISO-795	11111	ACM-256	11111
	ACM-256	22222	ACM-256	11111
	HCI-449	33333	ACM-256	11111

Kuva 4.6: Ristitulo

Kuten esimerkistä (kuva 4.6) huomaamme on ristitulon tuloksen koko suurempi kuin lähtörelaatioiden koko. Merkitään relaation R kokoa *koko(R)* ja astetta *aste(R).* Tällöin

$koko\ (R \times S) = koko(R) * koko(S)$ ja

$aste(R \times S) = aste(R) + aste(S)$

Ristitulossa tulosrelaatioon voisi tulla samannimisiä sarakkeita. Sovitaan, että mikäli tällaisia tulee, ne automaattisesti uudelleennimetään siten, että attribuutin nimen eteen liitetään pisteen seuraamana sen relaation nimi, josta attribuutti oli kotoisin eli relaation R attribuutti A olisi $R.A$ [2].

4.2.1.4 Projektio (projection)

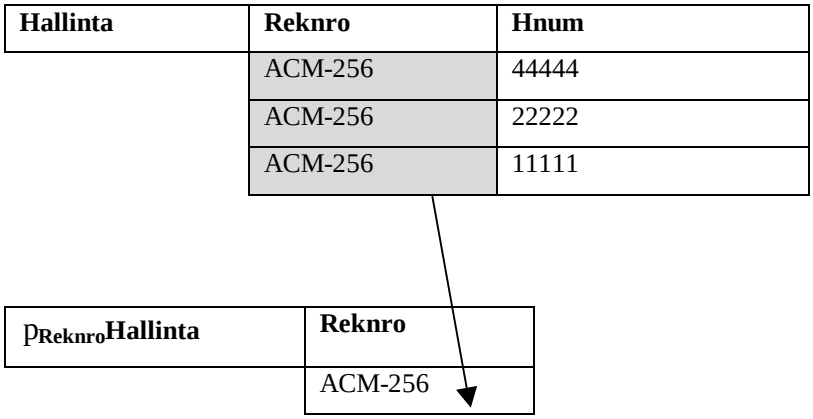
Edellä käsitellyt operaatiot olivat joukko-opin perusoperaatioita. Projektio sensijaan on relaatioalgebran oma operaatio. Se kohdistuu yhteen relaatioon. Operaation tuloksena saadaan uusi relaatio, jossa on ainoastaan valitut sarakkeet alkuperäisestä relaatiosta R. Operaatiosymbolina käytetään nykyään yleisesti merkkiä π (pikku pii).

$$\pi_{A_1,\ldots,A_n}(R) = \{ (a_1,\ldots,a_n) \mid x \in R,\ \forall i=1..n:\ a_i = x.A_i \},$$

missä $A_1,\ldots A_n$ ovat attribuutteja ja $x.A_i$ on attribuutin A_i arvo monikossa x.

Projektion tuloksen aste on siis pienempi tai yhtäsuuri kuin lähtörelaation aste. Tuloksen koko on myös pienempi tai yhtäsuuri kuin lähtörelaation koko. Se voi olla pienempi, sillä tilanteessa, jossa valittujen sarakkeiden arvot ovat samat useissa lähtörelaation monikoissa, arvoyhdistelmä tulee mukaan tulosrelaatioon vain kertaalleen. Tätä piirrettä kutsutaan toistuvien rivien eli duplikaattien (dublicate) karsimiseksi (kuva 4.7).

[2] Relaatioalgebrassa on myös uudelleennimentä operaatio, jota voitaisiin käyttä ratkomaan samannimisten sarakkeiden ongelmaa. Yksinkertaisuuden vuoksi tässä esityksessä käytetään kuitenkin sen asemasta nimeämissopimuksia.



Kuva 4.7: Projektio ja dublikaattien karsinta.

4.2.1.5 Valinta (selection)

Valintaoperaation kohteena on yksi relaatio. Valinnassa poimitaan tulosrelaatioon valintaehdon täyttävät lähtörelaation monikot. Operaatiosymbolina käytetään nykyisin yleisesti merkkiä σ (pikku sigma).

$\sigma_{ehto}(R)$ = { x | x$\in$R, ehto on voimassa, kun siinä esiintyvät attribuutit korvataan niiden arvoilla monikossa x}

Ehdossa vertailtavina voivat olla attribuutit ja vakiot. Vertailuoperaattoreina tulevat kyseeseen =, $\neq$, <, >, <= ja >=.

Auto	Reknro	Väri	Vmalli
	ACM-256	musta	1988
	MAC-532	sininen	1994
	ISO-795	musta	1992
	OSI-228	punainen	1987
	HCI-449	valkoinen	1993

$\sigma_{\text{Väri='musta'}}$**(Auto)**	Reknro	Väri	Vmalli
	ACM-256	musta	1988
	ISO-795	musta	1992

$\sigma_{\text{Vmalli<1990}}$**(Auto)**	Reknro	Väri	Vmalli
	ACM-256	musta	1988
	OSI-228	punainen	1987

$\sigma_{\text{Vmalli=1999}}$**(Auto)**	Reknro	Väri	Vmalli

Tässä tapauksessa tulos on siis tyhjä joukko.

Kuva 4.8: Valinta.

Valinta on määritelty yhteen vertailuehtoon perustuvana. Siinä voi kuitenkin käyttää myös ohjelmointikielissä käytettävien kaltaisia loogisia lausekkeita sillä nämä on muunnettavissa relaatioalgebran operaatioiksi. Esimerkiksi

$\sigma_{\text{ehto\_1 } \mathbf{or} \text{ ehto\_2}}(R) \equiv \sigma_{\text{ehto\_1}}(R) \cup \sigma_{\text{ehto\_2}}(R)$ ja

$\sigma_{\text{ehto\_1 } \mathbf{and} \text{ ehto\_2}}(R) \equiv \sigma_{\text{ehto\_1}}(R) \cap \sigma_{\text{ehto\_2}}(R)$

(huom. leikkausoperaatio esitellään myöhemmin johdettujen operaatioiden yhteydessä)

4.2.1.6 Sijoitus (assign)

Sijoitus on operaatio, jolla voidaan nimetä uudelleen kyselyn tulosrelaation sarakkeet. Se ei ole varsinainen relaatioalgebran operaatio. Sijoitus voidaan esittää muodossa

$R(A_1,\ldots,A_n)$ = kysely,

missä vasemmalla puolella annetaan tulosrelaation kaavio.

4.2.2 Johdetut operaatiot

Edellä on esitelty relaatioalgebran perusoperaatiot. Näiden ilmaisuvoima kyselyjä esitettäessä on sama kuin ensimmäisen kertaluokan predikaattilogiikan, johon pohjautuvia teoreettisia kyselykieliä, relaatiokalkyylejä (relational calculus) on myös esitetty. Kyselykieltä, jolla pystytään esittämään samat kyselyt kuin relaatioalgebran operaatioilla sanotaan relationaalisesti täydelliseksi (relationally complete). Käytännön kyselykielet ovat yleensä tätä ja ylikin. Kyselyn laatiminen relaatioalgebran perusoperaatioita käyttäen on aika hankalaa. Johdetut operaatiot helpottavat sitä, mutta eivät suinkaan tee relaatioalgebraa vielä käyttäjäystävälliseksi.

4.2.2.1 Leikkaus (intersection)

Leikkaus on joukko-opin operaatio, jolla saadaan tulokseksi kahden joukon yhteiset alkiot. Se voidaan esittää erotus-operaation avulla.

$$R \cap S = R - (R - S) - (S - R)$$

Esimerkki: Autot, joille on kirjattu sekä omistaja että haltija:

$\pi_{Reknro}(Omistus) \cap \pi_{Reknro}(Hallinta)$	**Reknro**
	ACM-256

Kuva 4.9: Leikkaus.

4.2.2.2 Liitos (join)

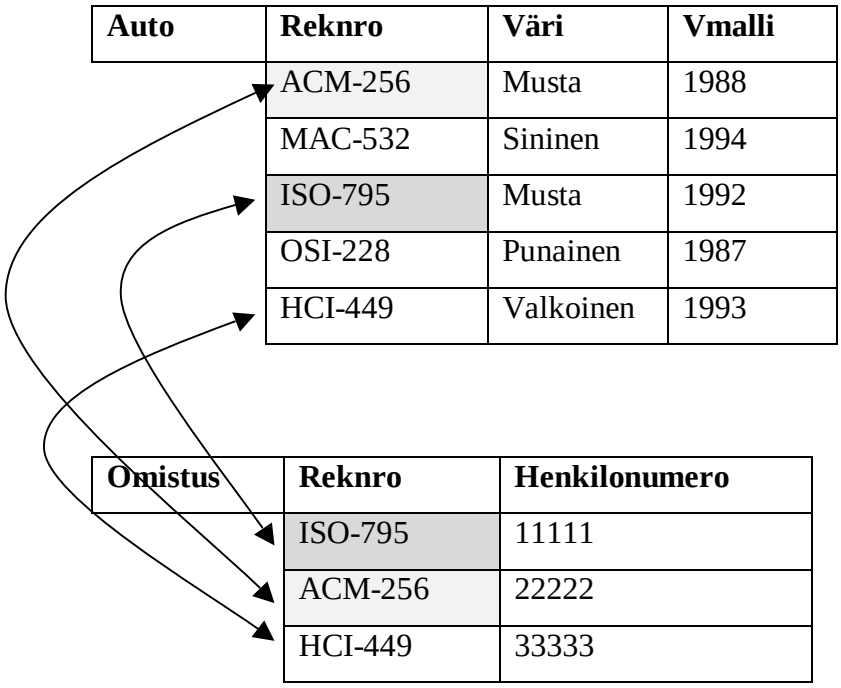
Liitosoperaatiot ovat johdettuja operaatioita joilla yhdistellään eri relaatioihin sisältyviä monikkoja jonkin ehdon avulla. Tyypillisesti relaatioiden välinen liitos perustuu siihen, että toisessa yhdisteltävässä monikossa olevan viiteavaimen täytyy olla sama kuin toisen monikon avain. Näin saadaan jokaiseen relaation monikkoon kytkettyä vuorollaan jokainen siihen toisessa relaatiossa viittaava monikko. Muunlaiset liitokset ovat käytännössä aika harvinaisia, mutta toki mahdollisia.

Yleinen liitos (join, theta join) on lyhennemerkintä ristitulon ja valinnan yhdistelmälle:

$$R \bowtie_{\text{liitosehto}} S \equiv \sigma_{\text{liitosehto}} (R \times S)$$

Tässä liitosehto on samanlainen kuin valinnassa käytettävä ehto, mutta ehto-lausekkeessa operandit ovat eri relaatioiden attribuutteja.

$$AO = \text{Auto} \bowtie_{\text{Auto.Reknro=Omistus.Reknro}} \text{Omistus}$$



Auto	Reknro	Väri	Vmalli
	ACM-256	Musta	1988
	MAC-532	Sininen	1994
	ISO-795	Musta	1992
	OSI-228	Punainen	1987
	HCI-449	Valkoinen	1993

Omistus	Reknro	Henkilonumero
	ISO-795	11111
	ACM-256	22222
	HCI-449	33333

Kuva 4.10: Liitoksessa yhdistettävät rivit.

$$AO = Auto \ \overset{\leftarrow}{\bowtie}_{Auto.Reknro=Omistus.Reknro} \ Omistus$$

Välitulos, johon on jätetty myös Auto-rivit, joille ei löydy ehdon täyttävää paria:

	Reknro	**Väri**	**Vmalli**	**Reknro**	**Henkilonumero**
	ACM-256	Musta	1988	ACM-256	22222
	MAC-532	Sininen	1994		
	ISO-795	Musta	1992	ISO-795	11111
	OSI-228	Punainen	1987		
	HCI-449	Valkoinen	1993	HCI-449	33333

Lopputulos:

AO	**Reknro**	**Väri**	**Vmalli**	**Reknro**	**Henkilonumero**
	ACM-256	Musta	1988	ACM-256	11111
	ISO-795	Musta	1992	ISO-795	22222
	HCI-449	Valkoinen	1993	HCI-449	33333

Kuva 4.11 Liitos viiteavaimeen perustuen.

Kuvien 4.10 ja 4.11 esimerkissä on liitetty rivit toisiinsa viiteavaimen ja viitattavan taulun avaimen yhtäsuuruusvertailuun perustuen. Kuhunkin autoon on siis liitetty tiedot sen omistajasta. Kaikille autoille ei kuitenkaan ole relaatiossa *Omistus* annettu omistajaa. Tällaisten autojen kohdalla ei ristitulosta löydy yhtään riviä, joka toteuttaisi liitosehdon (välitulos kuvassa 4.11). Täten näistä autoista ei tule mitään tietoa liitoksen tulosrelaatioon.

Liitoksen osapuolten ei välttämättä tarvitse olla eri relaatioita. Liitettäessä relaatio itseensä kytketään monikkoja toisiin saman relaation monikkoihin, esimerkiksi esimies alaisiinsa. Tällaisessa tilanteessa voidaan liitosehdossa käyttää osapuolista lisätarkenteita 1 ja 2, esim. *Auto.1.A* tarkoittaa ensimmäisen *Auto* osapuolen attribuuttia A. Tarkenteet voi myös jättää pois, jolloin ehto tulkitaan siten, että vasemmalla puolella oleva attribuutti liittyy ensimmäiseen osapuoleen ja oikealla oleva toiseen.

Liitoksessa voi käyttää mitä tahansa vertailuoperaattoria. Kuitenkin erisuuruus operaattorin suhteen on syytä pitää varansa; tuottaako se todella sen tuloksen jota halutaan. Esimerkiksi autoja, joilla ei ole omistajaa **ei saada selville** kyselyllä

$$\text{Omistajattomat} = \text{Auto} \overset{\text{Auto.Reknro} \neq \text{Omistus.Reknro}}{\rtimes} \text{Omistus},$$

vaan tämä kytkee jokaiseen autoon kaikkien muiden kuin auton itsensä omistustiedot eli tulos olisi

$$(\text{Auto} \times \text{Omistus}) - (\text{Auto} \overset{\text{Auto.Reknro=Omistus.Reknro}}{\rtimes} \text{Omistus})$$

eli vain kuvan 4.11 lopputulosrivit puuttuisivat ristitulosta. Vastauksen Omistajattomat autot saa sensijaan selville muodostamalla joukon komplementin 'Autot, joiden omistaja on tiedossa' ja erottamalla nämä kaikista autoista eli

$$\pi_{\text{Reknro}}(\text{Auto}) - \pi_{\text{Auto.Reknro}}(\text{Auto} \overset{\text{Auto.Reknro=Omistus.Reknro}}{\rtimes} \text{Omistus})$$

Viiteavaimen ja avaimen väliseen vertailuun perustuvan yhtäläisyysliitoksen (operaattorina =, equijoin) tulosrelaation koko on korkeintaan viittaavan relaation koko.

4.2.2.3 Luonnollinen liitos (natural join)

Luonnollisessa liitoksessa **R * S** ei tarvitse antaa liitosehtoa vaan tämä muodostetaan automaattisesti siten, että liitosehtona vaaditaan kaikkien vastinattribuuttien yhtäsuuruutta. Vastinattribuutilla tarkoitetaan tässä sellaista attribuuttia, joka esiintyy kummassakin relaatiossa. Edelleen, koska jokaisella vastinattribuutilla edellytetään olevan sama arvo kummassakin yhdistettävässä monikossa, attribuutti otetaan mukaan tulosrelaatioon vain kertaalleen.

Olkoot $A_1,\ldots,A_n$ R:n attribuutit, jotka eivät esiinny S:ssä ja $C_1,\ldots, C_m$ S:n attribuutit, jotka eivät esiinny R:ssä sekä $B_1,\ldots,B_k$ attribuutteja, jotka esiintyvät kummassakin relaatiokaaviossa. Tällöin

$$R * S \equiv \pi_{A1,\ldots,An,R.B1,\ldots,R.Bk,C1,\ldots,Cm} (R \overset{\text{R.B1=S.B1 and \ldots. and R.Bk=S.Bk}}{\rtimes} S)$$

Omistus * Hallinta =

$\pi_{\text{Omistus.Reknro,Henkilönumero,Hnum}}(\text{Omistus} \rtimes_{\text{Omistus.Reknro=Hallinta.Reknro}} \text{Hallinta})$

Omistus * Hallinta	**Reknro**	**Henkilonumero**	**Hnum**
	ACM-256	22222	44444
	ACM-256	22222	22222
	ACM-256	22222	11111

Kuva 4.12: Luonnollinen liitos.

Kuvan 4.12 esimerkissä liitos tehdään Reknro attribuuttien perusteella, koska attribuutti esiintyy kummassakin relaatiossa. Kummassakin esiintyy myös ’henkilönumero’, mutta erinimisenä, jolloin se ei valikoidu liitosattribuutiksi..

4.2.2.4 Ulkoliitos (outer join)

Ulkoliitos on yhdisteen ja liitoksen yhdistelmä, jolla saadaan mukaan tulos-relaation myös sellaiset lähtörelaation monikot, joille liitosehdon mukaisesti ei löydy paria toisesta lähtörelaatiosta (kuvan 4.11 välitulos).

$$R \rtimes_{\text{ehto}} S \equiv (R \rtimes_{\text{ehto}} S) \cup (R - \pi_{\text{att(R)}}(R \rtimes_{\text{ehto}} S)) \times \text{א},$$

missä att(R) tarkoittaa kaikkia R:n attribuutteja ja א on yksimonikkoinen relaatio, jonka kaavio on sama kuin relaatiolla S ja jonka jokainen arvo on tyhjäarvo. Kuvan 4.11 välitulos saavutetaan kyselyllä

$\text{Auto} \rtimes_{\text{Auto.Reknro=Omistus.Reknro}} \text{Omistus}$

Ulkoliitoksesta on myös olemassa molemminpuoleinen versio $R \rtimes\!\!\rtimes_{\text{ehto}} S$, jossa kummankin osapuolen parittomat täydennetään tyhjäarvoilla.

4.2.3 Esimerkkejä

Relatioalgebrassa kysely ilmaistaan muodostamalla relaatioalgebran operaatioista lauseke, joka tuottaa tuloksenaan halutun relaation. Sulkujen käytöllä voidaan ohjata laskentajärjestystä. Kyselyt kannattaa laatia muodostamalla väli- ja aputuloksia ja kokoamalla nämä lopulta yhteen. Seuraavissa esimerkeissä yritetään apu- ja välitulosten avulla kuvata päättelyä, joka johtaa lopputulokseen. Varsinainen tulos on kunkin esimerkin viimeisellä rivillä. Kyselyt voi esittää muullakin tavalla kuin tässä esitettävällä.

Käytettävät esimerkkirelaatiot ovat:

Auto(Reknro, Väri, Vmalli, Merkki)
Omistus(Reknro,Henkilönumero, Osoite, Nimi),
Katsastus(Reknro, Katsastusvuosi, KatsastusPvm, Hyväksyttiinkö),
Vika(Reknro, KatsastusPvm, Vika)

a) Vuosimallia 1996 olevat autot, joista on löytynyt vikoja vuoden 1999 katsastuksissa:

$\sigma_{Vmalli=1996}$ (Auto)

{vuosimallia 1996 oleva auto}

$\sigma_{Katsastusvuosi=1999}$ (Katsastus)

{vuoden 1999 katsastus}

Vika * $\sigma_{Katsastusvuosi=1999}$ (Katsastus)

{vuoden 1999 katsastuksissa löytyneet viat, liitos rekisterinumeron ja katsastuspäivän perusteella}

$\sigma_{Vmalli=1996}$ (Auto) * (Vika * $\sigma_{Katsastusvuosi=1999}$ (Katsastus))

{vuoden 1999 katsastuksessa vuosimallin 1996 autoista löytyneet viat, liitos rekisterinumeron perusteella}

$\pi_{\mathbf{Reknro}}$ **(** $\sigma_{\mathbf{Vmalli=1996}}$ **(Auto) * (Vika *** $\sigma_{\mathbf{Katsastusvuosi=1999}}$ **(Katsastus)))**

{lopputulokseksi rekisterinumero}

b) Omistajan nimi ja osoite vuosimallia 1995 tai sitä vanhemmista autoista, joiden vuoden 1999 katsastusta ei ole vielä tehty

$\sigma_{Katsastusvuosi=1999}$ (Katsastus)

{vuoden 1999 katsastukset}

$\sigma_{Vmalli <= 1995}$ (Auto)

{vuosimallia 1995 tai sitä vanhempi auto}

π_{Reknro} (Auto) – π_{Reknro} ($\sigma_{Katsastusvuosi=1999}$ (Katsastus))

{rekisterinumero autoista, joiden vuoden 1999 katsastusta ei ole tehty}

π_{Reknro} ($\sigma_{Vmalli<=1995}$ (Auto))– π_{Reknro} ($\sigma_{Katsastusvuosi=1999}$ (Katsastus))

{rekisterinumero vuosimallia 1995 olevista tai vanhemmista autoista, joiden vuoden 1999 katsastusta ei ole tehty}

Auto * Omistus

{Omistajatiedot liitetty autoon rekisterinumeron perusteella}

$\pi_{Nimi,Osoite}$ (Omistus * (π_{Reknro} ($\sigma_{Vmalli<=1995}$ (Auto)) –

π_{Reknro} ($\sigma_{Katsastusvuosi=1999}$ (Katsastus)))

{lopputulos}

c) Tiedot henkilöistä, jotka omistavat yhteisen auton (tiedot myös tästä) mutta asuvat eri osoitteissa.

Omistaja

$\bowtie_{Omistaja.1.Reknro=Omistaja.2.Reknro \text{ and } Omistaja.1.Henkilönrmero < Omistaja.2.Henkilönumero}$

Omistaja

{Saman auton omistavat henkilöparit. Tässä osapuoliin viitataan selvyyden vuoksi tarkentein 1 ja 2 [yleensä nuo jätetään pois]. Vertailuoperaattorilla ‘<’ saadaan aikaan se, ettei sama pari tule kahdesti eri järjestyksessä, erisuuruusvertailu johtaisi tähän}

p Auto. Reknro, Omistaja.1. Nimi, Omistaja.1.Osoite, Omistaja.2.Nimi, Omistaja.2.Osoite **(**

Auto * (Omistaja

⋈ Omistaja.1.Reknro=Omistaja.2.Reknro and

Omistaja.1.Henkilönrmero < Omistaja.2.Henkilönumerro

and Omistaja.1.Osoite ≠ Omistaja.2.Osoite

Omistaja**)**

)

{lopputulos}

d) Vuoden 1996 Opel Astroissa sekä Audi A4:ssa esiintyneet yhteiset viat vuoden 1999 katsastuksissa.

σ Vmalli=1996 and Merkki='Audi A4'(Auto)

{vuoden 1996 mallia oleva Audi A4}

π Vika (σ Vmalli=1996 and Merkki='Audi A4'(Auto) *

(Vika * σ Katsastusvuosi=1999 (Katsastus)))

{vuoden 1996 Audi A4 autojen viat vuoden 1999 katsastuksissa}

π Vika (σ Vmalli=1996 and Merkki='Audi A4'(Auto) *

(Vika * σ Katsastusvuosi=1999 (Katsastus))) ∩

π Vika (σ Vmalli=1996 and Merkki='Opel Astra'(Auto) *

(Vika * σ Katsastusvuosi=1999 (Katsastus)))

{lopputulos, leikkauksella saadaan yhteiset viat}

4.2.4 Relaatioalgebran käytännön merkityksestä

Relaatioalgebra muodostaa teoriaperustan relaatiotietokantojen käsittelylle. Sitä ei käytetä suoraan käytännön kyselykielenä, vaikka esimerkiksi SQL:stä löytyvät useimmat operaatiot miltei vastaavina. Relaatiotietokannanhallintajärjestelmien

toteutuksissa ytimenä ovat relaatioalgebran operaatioiden kaltaiset operaatiot. Näitä varten on toteutettu käsittelyalgoritmit. Kyselyn kääntäjä kääntää kyselyn näitä operaatioita käyttäväksi lausekkeeksi. Sen jälkeen kyselyn optimoija muokkaa lauseketta ja järjestää sen uudelleen tehokkaimman suoritustavan löytämiseksi. Operaatiot suoritetaan tämän suoritusjärjestyksen mukaisesti. Suorituksessa operaation lähtörelaatiot haetaan tarvittavassa laajuudessa apumuistista keskusmuistiin vertailuja varten ja tulosrelaatio kirjoitetaan takaisin apumuistiin, josta se taas luetaan seuraavan operaation lähtörelaationa. Joissakin tapauksissa tulosrelaation monikot voidaan toimittaa suoraan seuraavalle operaatiolle ilman että ne viedään välillä apumuistiin.

Keskusmuistiin ladattavien tietojen määrään voidaan vaikuttaa talletustason rakenteiden avulla, esimerkiksi indeksien avulla. Indeksejä käsitellään tarkemmin myöhemmin. Esimerkkinä niiden vaikutuksesta voidaan tarkastella edellä käytettyä Auto relaatiota. Jos relaatioon Auto olisi liitetty indeksi attribuuttiin Merkki perustuen ja valintaehdossa haettaisiin tietyn merkkisiä autoja, vähentäisi indeksin olemassaolo haettavien monikoiden määrän relaation kaikista monikoista vain kyseiseen automerkkiin liittyviin monikoihin.

Kyselyiden laatijoiden ja tietokannansuunittelijoiden on ainakin toistuvien kyselyiden osalta mietittävä myös kyselyn kustannuksia ja kyselyn suoritukseen kuvaa aikaa. Karkean arvion kyselyn kustannuksia voi tehdä muuntamalla kysely relaatioalgebran operaatioiksi ja arvioimalla näiden kustannukset vaikkapa käsiteltävien monikoiden lukumääränä.