Tampereen yliopisto

Olli Nurmi

CAD-TIETOKANNAT

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Toukokuu 2023

TIIVISTELMÄ

Olli Nurmi: CAD-tietokannat
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2023

Tämä pro gradu -tutkielma on kirjallisuuskatsaus CAD-tietokannoista. CAD (Computer-Aided Design) tarkoittaa tietokoneavusteista suunnittelua. CAD-ohjelmilla mallinnettavat objektit ovat tyypillisesti monimutkaisia rakenteeltaan. Niissä on toisiinsa yhteydessä olevia entiteettejä, ja niiden rakenne on usein hierarkkista. Objektit voivat olla koottuja objekteja, jotka sisältävät toisia objekteja. Tämä aikaansaa tietokannalle erityisiä vaatimuksia.

CAD-tietokannan tulee pystyä käsittelemään suuria datamääriä, ja datan monimutkaiset keskinäiset suhteet ovat esitettävä riittävästi. Tietokannan täytyy pystyä käsittelemään datan dynaamista luonnetta. Tämä johtuu siitä, että käyttäjä voi suunnitteluprosessin edetessä määritellä uusia dataluokkia ja muuttaa niitä. Tietokannan tulee myös muun muassa olla riittävän joustava, tukea eri versioita ja huolehtia datan eheydestä. Rajoitusten käsittelykin on yleensä poikkeuksellista CAD-tietokannoissa, johtuen muun muassa poikkeuksellisen pitkistä tapahtumasarjoista.

Yleisimmät CAD-järjestelmissä käytettävät tietokantamallit ovat hierarkkinen, verkkomallin, relaatio-, oliomallin ja olio-relaatiomallin tietokanta. Näissä kaikissa malleissa on hyvät ja huonot puolensa. Hierarkkista ja verkkomallin tietokantoja käytetään edelleen, vaikka ne ovat vanhoja. Relaatiotietokantaa pidetään hierarkkista ja verkkomallin tietokantoja parempana muun muassa sen loogisuuden ja yksinkertaisuuden takia. Oliomallin tietokantaa pidetään monissa lähteissä parhaana CAD-järjestelmiin. Se ei kuitenkaan ole syrjäyttänyt muita tietokantamalleja.

Avainsanat: CAD, tietokanta, tietokantamalli

1 Johdanto

CAD eli tietokoneavusteinen suunnittelu tarkoittaa tietokonejärjestelmän käyttöä auttamaan suunnitelman luontia, muuttamista, analysointia tai optimointia [Sarcar *et al.* 2008, s. 3]. CAD-järjestelmän osakomponentit kommunikoivat yhteisen tietokannan kautta [Neumann 2005]. Tietokanta on kokoelma dataa eli se on varastoalue, johon voidaan varastoida dataa, jota voidaan prosessoida [Naik 2013, 1]. Tietokannan hallintajärjestelmä (DBMS) hallitsee kaikkea dataa järjestelmässä. DBMS tarjoaa tietomallin ja kielen sen käsittelemiseksi. [Neumann 2005]

CAD-datalla on ainutlaatuisia ominaisuuksia, jotka vaikeuttavat niiden hallintaa. CAD-objektilla on yleensä monimutkainen rakenne, joka sisältää suuria datamääriä. Lisäksi suunnitellussa objektissa on yleensä paljon tallennettavaa dataa, jota voidaan päivittää myöhemmin. CAD-objekti voi myös koostua alemman tason komponenteista. Kun alemman tason komponenttia muutetaan, sen sisältäneen ylemmän tason komponentin tulee joko muuttua automaattisesti tai muuttua mitättömäksi. Kaikki nämä datan eri osa-alueet on tallennettava, ja niitä on käsiteltävä oikein integroidussa ympäristössä, johon eri CAD-apuohjelmat voivat päästä käsiksi, jotta näiden objektien säilytys-, ylläpito- ja käyttökustannukset olisivat minimaaliset. [Liu 2003] Tämä aiheuttaa erityishaasteita CAD-järjestelmän tietokannalle.

CAD-entiteetit kuvaavat ensisijaisesti geometrisia elementtejä, kuten viivoja, ympyröitä jne. Monimutkaisemmat CAD-objektit ovat graafisia objekteja, kuten kokoonpanoja, jotka on luotu ryhmittelemällä yhteen joukko geometrisia elementtejä. [Liu 2003]

Tässä tutkielmassa on tutkittu CAD-järjestelmissä käytettäviä tietokantoja. Minkälaisia erityispiirteitä ja vaatimuksia niillä on, ja mitä tietokantamalleja niissä käytetään. Luvussa 2 on käsitelty CAD:ia yleisesti. Siinä käsitellään muun muassa CAD-objektia ja sen mallintamista. Luvussa 3 on käsitelty tietokantoja. Siinä kerrotaan tietokannoista yleisellä tasolla ja siitä, mitä vaatimuksia CAD tietokannoille on asetettu sekä, mitä erityispiirteitä CAD-tietokannoilla on. Luvussa 4 on käsitelty tietokantamalleja, joita CAD-järjestelmissä käytetään. Näitä ovat hierarkkinen, verkkomallin, relaatiomallin, oliomallin, olio-relaatiomallin, XML-, spatiaalinen ja deduktiivinen tietokanta. Luvussa 5 on esitetty esimerkkejä eri tietokantamalleista CAD-esimerkeillä. Luvussa 6 on pohdittu eri tietokantamallien suosiota CAD-järjestelmissä.

2 Tietokoneavusteinen suunnittelu (CAD)

CAD määritellään tietokonejärjestelmän käyttöä auttamaan suunnitelman luontia, muuttamista, analysointia tai optimointia. Tietokonejärjestelmä koostuu tietokoneesta, yhdestä tai useammasta näytöstä, muista laitteista ja ohjelmasta. Modernit CAD-järjestelmät perustuvat interaktiiviseen tietokonegrafiikkaan. CAD:in käytölle on neljä hyvää syytä [Sarcar *et al.* 2008, s. 3-4]:

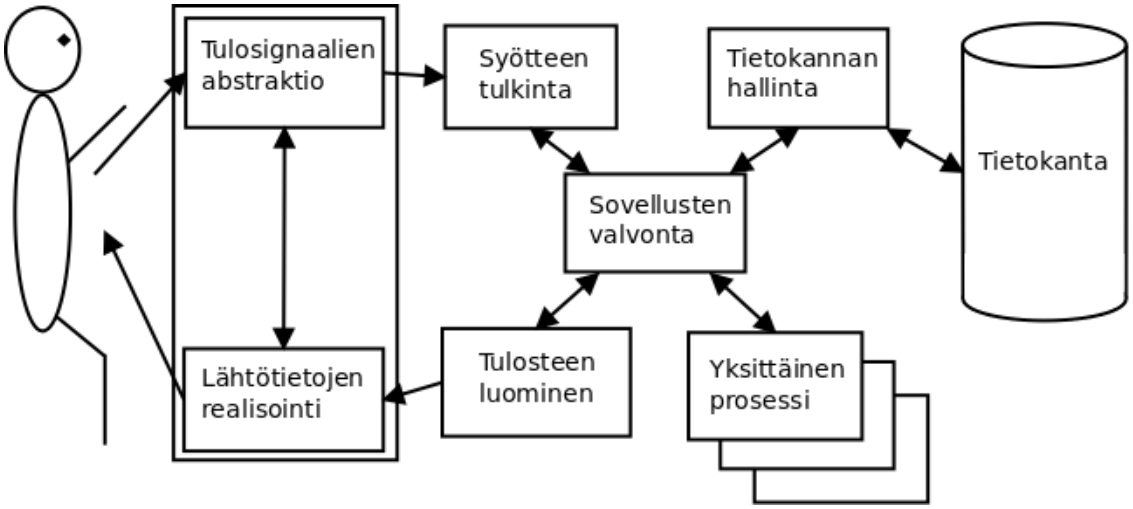
- Lisää suunnittelijan tuottavuutta.
- Parantaa suunnitelman laatua.
- Parantaa kommunikaatiota dokumenttien avulla.
- Luo tietokannan valmistusta varten.

CAD-ohjelmistoja on käytetty 1950-luvulta lähtien [Bi ja Wang 2020, 1.2]. 1980-luvun alussa CAD-ohjelmistot vähensivät teknisten piirrosten tekijöiden tarvetta teollisuudessa. 1990-luvun puolivälissä CAD-ohjelmistoista tuli riittävän edullisia, jonka ansiosta insinöörit pystyivät itse tekemään niillä teknisiä piirroksia, ja myös jossain määrin analyyttistä työtä [Chang 2014, 3.1].

Nykyisin CAD-ohjelmistoilla voidaan tehdä suoraan ns. solidi malli, josta saadaan riittävästi dataa esimerkiksi materiaali, prosessit, dimensiot ja toleranssit. Solidista mallista on selitetty tarkemmin luvussa 2.2.3. Tekniset piirrokset eivät ole enää välttämättömiä. Nämä solidit mallit tukevat mm. tuotteen virtuaalista valmistusta, prototyyppien tekemistä, valmistusprosessin suunnittelua, hinta-arviota ja tarjoavat tietovaraston arkistointia varten. Solidien mallintaminen onkin oleellisin asia CAD:ssa [Chang 2014, 3.1] Kuvassa 1 on esitetty CAD-järjestelmän pääorganisaatio.

CAD-ohjelmistot toimivat kolmen ristiriitaisen vaatimuksen alaisina:

- Suuret monimutkaiset ongelmat vaativat turvallista ja organisoitua tiedonkäsittelyä.
- Prosessori intensiiviset algoritmit ovat riippuvaisia nopeasta tiedonsaannista.
- Muisti rajoittaa ohjelman käsittelemän ongelman kokoa. [Soukup 1991]



Kuva 1. CAD-järjestelmän pääorganisaatio [Yagiu 2012, s.2].

2.1 CAD-objekti

Suunnittelu- ja suunnittelutietokantojen mallintamat objektit ovat yleensä osien kokoonpanoja, jotka koostuvat itse pienemmistä osista. Objektien kokoonpanoja kutsutaan kootuksi objektiksi (composite object). Objekteja, jotka eivät ole kokoonpanoja, kutsutaan atomiobjekteiksi (atomic object). Kootuilla objekteilla on hierarkkinen rakenne. Tämän hierarkian lehtisolmut ovat atomiobjekteja ja haarautumasolmut ovat koottuja objekteja. [Ketabchi ja Berzins 1987]

Pohjimmiltaan CAD-objekteilla on monimutkainen rakenne, ja ne voivat lisäksi sisältää suuren määrän komponentteja. Toiminnallisesti komponentit voidaan määrittää itsenäisesti tai kiinteästi sidottuna kohteeseen. CAD-objekteissa on myös ominaisuuksia, joita käyttäjä voi muokata. CAD-objekti määrittelee seuraavat asiat: sen rakenteeseen liittyvät ominaisuudet, rajoitukset, jotka määrittelevät sen ominaisuuksien välisiä suhteita ja toiminnalliset linkit, jotka määrittävät laskentasäännöt objektin johtamiseksi muista. [Rieu ja Nguyen 1986]

2.2 CAD-objektin mallintaminen

2.2.1 Rautalankamalli

Rautalankamalli on yksinkertaisin mallinnusmetodi. Siinä esitetään objektin särmät [Bi ja Wang 2020, 2.4.1]. Särmät voivat olla esimerkiksi janoja tai käyriä. Rautalankamallin etu on siinä, että sen mallintaminen on yksinkertaista. Myös mallintamiseen vaadittavan ohjelman toteuttaminen on suhteellisen helppoa. [Chang 2014, 3.2.1]

Rautalankamallin heikkous on siinä, että sillä ei voi määritellä kiinteää objektia yksikäsitteisesti. Tästä seuraa esimerkiksi se, että sillä ei voi laskea kappaleen massaa eikä sillä voi yleensä analysoida objektin rakenteellisia ominaisuuksia. Sillä ei myöskään pysty esittämään kaarevia pintoja tietyissä tilanteissa. [Chang 2014, 3.2.1] Mallinnetun

datan valmistelu, tuonti ja käsittely ovat aikaa vieviä ja alttiita virheille rautalankamallissa. [Bi ja Wang 2020, 2.4.1]

2.2.2 Pintamalli

Pintamalli esittää osan pinnan muotoja. Se on tavallaan rautalankamallin päälle vedetty kuori. Pintamallissa on tietoa osan pinnoista ja särmistä. Jotta pintamallilla voisi esittää solidin osan, sen täytyisi olla täysin ilmatiivis ja suljettu. Ilmatiivistä ja suljettua mallia kutsutaan rajapintamalliksi (Boundary representation, B-Rep) (Boundary Surface Modelling [Bi ja Wang 2020, 2.4.3]). [Chang 2014, 3.2.2] Ou-Yangin ja Liun [1999] mukaan rajapintamalli on yleisin CAD-tietokannoissa käytetty esitystapa.

Pintamallia käytetään erityisesti vapaamuotoisten osien mallinnuksessa, ja se on hyvä, kun halutaan visualisoida monimutkaisia pintoja. Sitä käytetään paljon myös FE-analyyseissä (finite element analysis). Pintamalli toimiikin yleensä hyvin solideja osia mallinnettaessa. [Chang 2014, 3.2.2] Se ei kuitenkaan sisällä tietoa osan tilavuudesta tai massasta [Bi ja Wang 2020 2.4.2, 2.4.3].

2.2.3 Solidi malli

Solidi malli sisältää tiedot kappaleen särmistä pinnoista ja sisäosasta. Se on lopullinen tapa esittää yleisiä objekteja. Solidi malli antaa täyden tiedon fyysisestä kappaleesta. Solidille mallille voidaan laskea esimerkiksi massa, jota tarvitaan liikesimulaatioissa. Se tukee myös törmäystarkasteluita, joita tarvitaan kokoonpanossa ja kinemaattisissa analyyseissa. [Chang 2014, 3.2.3]

Solideja malleja voidaan mallintaa esimerkiksi pintaoperaatioilla, ulokeoperaatioilla piirrepohjaisella mallinnuksella, Boolen operaatioilla ja parametrisella mallinnuksella. Pintaoperaatiossa osan pintaa voidaan esimerkiksi leikata. Ulokeoperaatioissa piirretään ensiksi kaksiulotteinen piirros (sketch). Tämän jälkeen muodostetaan piirroksesta kolmiulotteinen uloke. Uloke voidaan tehdä esimerkiksi suoraan (extruding) tai kiertämällä pyörähdysakselin ympäri. Piirrepohjainen mallinnus tukee pääasiassa valmistusta. Tällä tarkoitetaan esimerkiksi reikien mallinnusta, joita valmistetaan poraamalla. Boolen operaatioilla yhdistetään kaksi kappaletta, josta saadaan esimerkiksi niiden leikkaus, unioni ja erotus. Parametrisella mallinnuksella muutetaan osan geometrisia muotoja. [Chang 2014, 3.2.3]

2.2.4 Abstraktiomuodot mallintamisessa

Mekaanisen CAD-datan mallintamiseen on olemassa kaksi erittäin hyödyllistä abstraktia muotoa: yhdistäminen (aggregation) ja yleistäminen. Yhdistämisessä käsitellään toisiinsa

liittyviä objekteja korkeamman tason objektina. Esimerkiksi mekaanisen osan pintamallissa särmä voidaan nähdä olevan yhdistelmä kahdesta solmusta ja kaaresta. Yhdistämismuoto on hyödyllinen kun mallinnetaan osa-komponentti-hierarkioita. [Spooner 1991]

Yleistämisessä saman luokan samankaltaiset oliot voidaan nähdä yleisesti korkeamman tason oliotyypin tapauksina. Esimerkiksi solidi malli voidaan nähdä CSG-puun (Constructive Solid Geometry) ja rajapintamallin yleistämisenä [Requicha ja Voelcker 1983]. Yleistämismuoto on hyödyllistä, kun mallinnetaan suunnitteluvaihtoehtoja. [Spooner 1991]

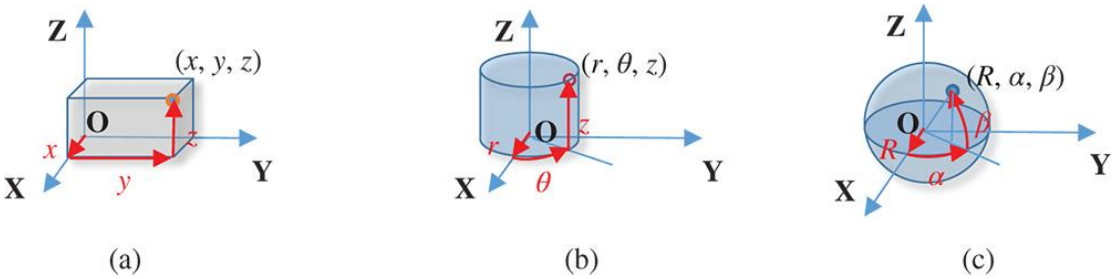
Yhdistämistä ja yleistämistä voidaan käyttää abstraktiohierarkioiden tekemiseen [Smith ja Smith 1977]. Nämä hierarkiat voidaan yhdistää hierarkioiden hierarkiaan siten, että jokainen objekti osallistuu samanaikaisesti sekä yhdistämis- että yleistysabstraktioihin. Siten esimerkiksi objekti, joka edustaa osaa yleistyshierarkiassa, voi määritellä osan peruskomponentit yhdistämishierarkiaksi. Tämän koontihierarkian voivat sitten periä kaikki yleistyshierarkian osan erikoisalat, ja sitä voidaan edelleen jalostaa kuvaamaan osan kunkin vaihtoehdon yksityiskohdat. Tämä tarjoaa tehokkaan työkalun tietojen mallintamiseen CAD-ympäristössä. [Spooner 1991]

2.3 Geometriset elementit

Osan, tuotteen tai järjestelmän suunnittelu alkaa yleensä geometrisella mallinnuksella. Geometrisen mallinnusta käytetään luomaan virtuaalinen geometrinen esitys todellisesta tai kuvitteellisesta kohteesta, joka sisältää tietoa kohteen muodosta, mitoista ja materiaaleista. [Bi ja Wang 2020, 2.1]

2.3.1 Koordinaatistot

Kolme yleisintä koordinaatistoa kolmiulotteisessa avaruudessa ovat karteesinen koordinaatisto, sylinterikoordinaatisto ja pallokoordinaatisto. Karteesisessa koordinaatistossa on kolme lineaarista akselia, jotka ovat toisiinsa kohtisuorassa. Sylinterikoordinaatistossa on yksi kiertoakseli ja kaksi lineaarista akselia: säde ja korkeus. Pallokoordinaatistossa on yksi translaatioakseli ja kaksi kiertoakselia. Pisteen sijainti pallokoordinaatistossa määritellään siis kolmen muuttujan avulla: etäisyys origosta eli säde, atsimuuttikulma ja napakulma. Koordinaatistot on esitetty kuvassa 2. [Bi ja Wang 2020, 2.2.1]



Kuva 2. a) karteesinen koordinaatisto b) sylinterikoordinaatisto c) pallokoordinaatisto [Bi ja Wang 2020, 2.2.1].

2.3.2 Kaavat karteesisessa koordinaatistossa

Jana kolmiulotteisessa avaruudessa määritellään kahdella pisteellä (x_1, y_1, z_1) ja (x_2, y_2, z_2), jotka on tallennettu tietokantaan. Seuraavilla kaavoilla voidaan laskea kaikki janan pisteet, kun y käy arvot y_1 :stä y_2 :teen:

$$x = \frac{(x_2 - x_1)(y - y_1)}{y_2 - y_1} + x_1$$

$$z = \frac{(z_2 - z_1)(y - y_1)}{y_2 - y_1} + z_1.$$

Ympyrän, jonka keskipiste on (x_0, y_0), matemaattinen esitys implisiittisessä muodossa on seuraava:

$$(x - x_0)^2 + (y - y_0)^2 = R^2.$$

Tämä voidaan johtaa muotoon:

$$x = \sqrt{(R^2 - (y - y_0)^2)} + x_0.$$

Kolmiulotteiset käyrät, joita ei voida analyyttisesti määritellä, voidaan määritellä interpoloimalla paloittain. Tarvitaan vain riittävästi kontrollipisteitä. Yksi esimerkki interpoloinnista on Lagrangen kaava:

$$P(t) = \sum_{1=0}^{n-1} L_i(t) \cdot P_i,$$

jossa

$$L_i(t) = \frac{\prod(t - t_i)}{\prod(t_j - t_i)}.$$

Kaavassa P_i on valitut kaksi muuttuja (x_i, y_i tai z_i) kontrollipisteestä ja t on jäljelle jäänyt muuttuja, joka käy kyseisen muuttujan arvot läpi. Myös vapaamuotoinen pinta, jota ei voida määritellä analyyttisesti, voidaan määritellä paloittain interpoloimalla. Pinnassa tarvitaan kaksi riippumatonta muuttujaa.

Pallo voidaan tallentaa tietokantaan, kun tiedetään sen keskipiste (x_0, y_0, z_0) ja säde R. Pallon matemaattinen esitys implisiittisessä muodossa on seuraava:

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2.$$

Tämä voidaan johtaa muotoon:

$$x = \sqrt{(R^2 - (y - y_0)^2 - (z - z_0)^2)} + x_0.$$

[Bi ja Wang 2020, 2.3.2]

3 Tietokanta

Tietokanta on kokoelma dataa. Eli se on varastoalue, johon voidaan varastoida dataa, jota voidaan prosessoida. Tietokannan pitäisi sisältää täsmällistä, johdonmukaista ja oleellista dataa, joka voidaan jakaa eri ohjelmille. Datojen välille voidaan määritellä suhteita, jos ne liittyvät toisiinsa. Tietokannassa olevaa dataa voidaan hakea, päivittää tai poistaa. [Naik 2013, 1]

Tietokantaa tarvitaan tehokkaaseen ja helppoon datatallenteiden varastointiin, hakuun, päivittämiseen ja poistamiseen. Sitä tarvitaan myös monimutkaisen ja suurivolyymisen datan varastointiin sekä monidimensioisen datan hallintaan. Tietokannasta on myös hyötyä, kun halutaan jakaa dataa eri sovelluksille. [Naik 2013, 1]

Data on raakaa faktaa, joka voi olla esimerkiksi numeroita, nimiä, päivämääriä. Aiheeseen liittyvä data (related data) tarkoittaa dataa, joka kuuluu samaan entiteettiin. Kun prosessoidaan aiheeseen liittyvää dataa, saadaan informaatiota. Dataa voidaan kuvata tallentamalla datan tyyppi, koko, rajoitteet jne. Käyttämällä tätä informaatiota tietokannan hallintajärjestelmä luo tietohakemiston, joka sisältää dataa datasta eli metadataa. [Naik 2013, 1.2] Tietokannan hallintajärjestelmästä on kerrottu tarkemmin luvussa 3.1. Attribuutti on informaation aliryhmä entiteetissä. Entiteetti on olio, joka esittää dataryhmää tai datakategoriaa. [Stephens ja Plew 2001]

3.1 Tietokannan hallinta

Tietokannan datan hallintaprosessia kutsutaan tietokannan hallinnaksi. Tietokannan hallintaa varten tarvitaan tietokannan hallintajärjestelmä (DBMS), joka on kokoelma sovellusohjelmia, joilla voidaan hallita tietokantaobjekteja, kuten tauluja, käyttäjiä, funktioita jne. Tietokannan hallintaan sisältyy seuraavat asiat: uusien datatiedostojen luominen, datatiedostojen rakenteiden päivittäminen, datatiedostojen poistaminen, suhteiden asettaminen datatiedostojen välille, data-arvojen lisääminen, data-arvojen poistaminen, data-arvojen päivittäminen, tietohakemiston ylläpitäminen ja tietokantaobjektien (kuten näkymien, funktioiden, triggereiden jne.) luominen, päivittäminen ja poistaminen. [Naik 2013, 1.4-1.5]

DBMS:llä huolehditaan tietoturvasta, samanaikaisen käytön synkronisoinnista ja kaatumisilta suojautumiselta sekä niistä palautumiselta. Se mahdollistaa myös informaation esittämisen eri näkymistä. DBMS tukee kahta eri tyyppistä käyttöliittymää: Ohjelmointikäyttöliittymää ja interaktiivista käyttöliittymää. Ohjelmointikäyttöliittymässä voidaan kirjoittaa proseduraalisella kielellä kuten Pascal. Se tukee kannanmäärittelykieltä (data definition language eli DDL) ja kannankäsittelykieltä (data manipulation language eli DML). [Neumann 2005]

Jos DBMS:llä on määrä hallita suunnittelutietoja tehokkaasti ja tarjota integroitu ympäristö suunnittelulle, sillä on oltava jonkin verran tietoa hallinnoimiensa tietojen käyttötarkoituksesta. Toisin sanoen DBMS:llä on oltava jonkin verran tietoa sen tallentamien tietojen semantiikasta. [Spooner *et al.* 1986]

Maier [1986] mukaan suurin osa CAD-järjestelmistä käyttää niiden omaa tiedonhallintaa käyttöjärjestelmän tiedostojärjestelmän päällä. Myös Anumban [1996] mukaan tuon ajan tallennusmekanismit olivat riittämättömiä CAD-datalle, ja CAD DBMS:lle on laadittu vaatimuksia useiden toivottujen ominaisuuksien korostamiseksi. Tietokannan hallintajärjestelmän huolellinen suunnittelu on välttämätöntä CAD-järjestelmän tehokkuuden kannalta [Anumba 1996].

3.2 Tapahtumasarjat ja datan johdonmukaisuus

Tietokannan voidaan ajatella olevan kokoelma entiteettejä. Entiteetit liittyvät yleensä toisiinsa ja täyttävät joitain johdonmukaisuusrajoituksia. Jos kaikki rajoitukset täyttyvät, tietokanta on johdonmukaisessa tilassa. Jos tietokannalle tehdään jotain toimintoja (esimerkiksi lukeminen, kirjoittaminen), tietokanta saattaa ajautua epäjohdonmukaiseen tilaan. Tapahtumasarja (tietokantatapahtuma, transaction) on joukko toimia, joka muuttaa tietokannan johdonmukaisesta tilasta uuteen johdonmukaiseen tilaan. [Neumann 2005] Kun käyttäjä lukee/kirjoittaa/poistaa dataa tietokannasta tai tietokantaan kyseessä on tapahtumasarja [Naik 2013, 8.2]. Tietokannan hallintajärjestelmä on vastuussa johdonmukaisuuden ylläpidosta. Käyttäjän täytyy vain aina ilmaista tapahtumasarjan alku ja loppu. Johdonmukaisuus täytyy säilyttää esimerkiksi seuraavissa tilanteissa [Neumann 2005]:

- Jos useita tapahtumasarjoja suoritetaan samaan aikaan.
- Jos järjestelmä kaatuu.
- Jos tapahtumasarja keskeytetään.

Tapahtumasarjojen samanaikaisesta suorittamisesta voi seurata päivityksen menetys, virheellinen luku tai luku, jota ei voida toistaa. Päivityksen menetys voi tapahtua tilanteessa, jossa tapahtumasarja T1 päivittää tallenteen, jota tapahtumasarja T2 on aiemmin päivittänyt. Jos nyt T2 peruutetaan myös T1 peruuntuu. Väärä luku voi tapahtua tilanteessa, jossa T1 päivittää tallennetta, jota T2 lukee. Jos nyt T1 peruutetaan, T2 on lukenut tallenteen, jota ei ole koskaan ollut olemassa. Luku, jota ei voida toistaa, voi tapahtua tilanteessa, jossa T1 lukee tallenteen, jota T2 tämän jälkeen muuttaa. Jos nyt T1 lukee uudelleen tallenteen, näkee se eri arvon. [Neumann 2005]

Tapahtumasarjan tulos on sitoutunut (committed), kun se luopuu oikeudesta peruuttaa kirjoitusta. Tapahtumasarjalla saavutetaan johdonmukainen tila seuraavissa tilanteissa: Jos tapahtumasarja ei ylikirjoita muiden tapahtumasarjojen sitoutumatonta dataa. Jos tapahtumasarja ei sido yhtäkään kirjoitusta ennen kuin se on suorittanut kaikki kirjoituksensa. Jos tapahtumasarja ei lue muiden tapahtumasarjojen sitoutumatonta dataa. Tai jos

muut tapahtumasarjat eivät muuta dataa, jota tapahtumasarja lukee, ennen kuin se on valmis. [Neumann 2005]

Johdonmukaisuus voidaan pakottaa käyttämällä tiettyjä lukkoprotokolia. Lukot voivat olla jaettuja tai eksklusiivisia. Tapahtumasarja noudattaa johdonmukaista lukitusprotokollaa jos seuraavat ehdot täyttyvät [Neumann 2005]:

- Tapahtumasarja asettaa eksklusiivisen lukon kaikkeen dataan, jota se tekee epäjohdonmukaiseksi.
- Tapahtumasarja asettaa jaetun lukon kaikkeen dataan, jota se lukee.
- Kaikki lukot säilytetään tapahtumasarjan loppuun asti.

Lukittavien osien valinta on kompromissi samanaikaisuuden ja yleisrasitteen välillä. Lukittavia osia voivat olla esimerkiksi segmentit, relaatiot, monikot tai kentät. Eli samanaikaisuus kasvaa, kun hyvin lukkiutuva osa valitaan. Monimutkaisissa tapahtumasarjoissa, joissa täytyy päästä suureen määrään tallenteita, hyvin lukkiutuvan osan valinta olisi kallis. Tällaisen tapahtumasarjan täytyisi asettaa ja nollata todella monta lukkoa. Tästä syystä samassa systeemissä olisi hyvä olla lukittavia yksiköitä eri rakeisuudella. [Neumann 2005]

Kuvitellaan esimerkki tietokannasta, joka on hierarkkiselta rakenteeltaan seuraava:

Tietokanta
|
Segmentit
|
Relaatiot
|
Monikot

Jokainen solmu voidaan lukita. Jos lukitaan jokin tietty solmu, kaikki sen jälkeläissolmut täytyy myös lukita. Solmun A jälkeläissolmujen lukitsemisessa on tärkeää estää solmun A edeltäjiä lukitsemasta solmuja, koska ne voisivat lukita solmut yhteensopimattomaan muotoon. Tarkoitustilaa käytetään ilmaisemaan, että lukitseminen tehdään matalammalla tasolla. Oikea toimintatapa lukita solmun A jälkeläiset on lukita ensin kaikki sen edeltäjät tarkoitustilaan, ja tämän jälkeen lukita solmu A. Tämä pätee sekä eksklusiiviseen että jaettuun lukitsemiseen. [Neumann 2005]

Järjestelmän täytyy säilyttää datan johdonmukaisuus ei-toivottujen tapahtumien varalta. Ei toivottuja tapahtumia ovat esimerkiksi järjestelmän kaatumiset tai lukkiutumiset. Järjestelmä suojelee datan johdonmukaisuutta toteuttamalla alkeistransaktioita (atomic transactions). Alkeistransaktiosta tulee pysyvä, kun se on suorittanut kaikki päivityksensä. Jos alkeistransaktio abortoidaan tai jos järjestelmään tulee toimintahäiriö, sen päivitykset eivät näy muille tapahtumasarjoille. [Neumann 2005]

Muisti on lukittu kiinteänpituisiin jako- ja siirtoyksiköihin primääri- ja sekundaarimuistin välille. Primäärimuisti on epävakaa, koska sen sisältö menetetään epäonnistumisissa. Sivun siirrosta voi seurata jokin seuraavista tapahtumista: onnistuminen (sivu saa uuden arvon), osittainen epäonnistuminen (sivu on sekaisin) tai täydellinen epäonnistuminen (sivu säilyy muuttumattomana). Palautumisen hallinnan täytyy havaita nämä virheet ja korjata ne. Osittainen virhe voidaan korjata toistamalla kirjoitukset ja lukea jälkeenpäin. Sivun vanha kopio hylätään vasta, kun uusi kopio on kirjoitettu onnistuneesti. [Neumann 2005]

3.3 Vaatimukset CAD-tietokannalle

Oleellisin luonnehdinta CAD-tietokannalle on, että se on informoiva malli todellisen maailman osasta, joka sisältää entiteettejä, joita ihmiset luovat, aistiva, manipuloivat ja ajattelevat. Tämä perustavanlaatuinen, yksinkertaiselta vaikuttava ja lähes itsestään selvä fakta saa aikaan monia ainutlaatuisia vaatimuksia CAD:lle ja usein myös CAM:lle (computer aided manufacturing eli tietokoneavusteinen valmistus). [Yagiu 2012] Sarcarin ja muiden [2008] mukaan tärkein ominaisuus CAD-ohjelmalle on sen täysin kolmiulotteinen, keskitetty, assosiatiivinen ja integroitu tietokanta.

Yagiu [2012, s. 1] määrittelee seuraavia haasteita CAD:lle:

- Käyttäjäystävällisen käyttöliittymän määrittely.
- Tasapainon jako kuormituksen sekä toiminnallisuuden välillä paikallisen grafiikan ja keskusprosessorin välillä.
- Rakenteiden graafisten entiteettien ja niiden manipuloinnin erittely kuormituksen jaon mukaisesti.
- Rakenne ja hallinta tietokannalle, joka kuvaa suunniteltavia objekteja johdonmukaisesti ja avoimen laajennettavissa olevalla tavalla ja täten antaa pääsyn eri käyttäjille eri tarkoituksiin.
- Kehittää sopiva matemaattinen esitys spatiaalisille objekteille ja tehokkaat algoritmit niiden geometrian laskemiseen.
- Suunnitella valvontamekanismi, joka valvoo eri moduuleita sekä ohjelmia ja kontrolloi informaatio virtaa niiden välillä.

Yagiu [2012, s. 21] määrittelee kolme lähestymistapaa CAD-tietokantojen vaatimuksille. Ensimmäisenä on induktiivinen lähestymistapa. Siinä tutkitaan mahdollisimman laajasti yksittäisiä toteutuksia tietokannan hallintaohjelmistoista. Näistä kerätään ominaisuuksia ja funktioita, jotka ovat sopivia CAD:iin. Nämä otetaan talteen ja ilmaistaan ne abstraktimmin. Toinen lähestymistapa on deduktiivinen, jossa vaatimukset tietokannalle johtaa abstraktinpiin ja kattaviin käsityksiin suunniteltavista objekteista ja prosesseista. Kolmatta lähestymistapaa kutsutaan sieppaavaksi (abductive). Siinä tutkitaan vaatimuksia

valitussa konkreettisessa esimerkissä, ja tarkennetaan ja ekstrapoloidaan ne tähän tilanteeseen.

Katragaddan [1999] mukaan CAD-tietokannoille on kolme vaatimusta: Datan integraatio, varasto monimutkaisille rakenteille ja redundanssin minimointi. Datan integraatiolla tarkoitetaan sitä, että tarvitaan integroitu tietokanta. Eri CAD-ohjelmistoilla saattaa olla erilainen näkemys dataan. Kaikki eri näkökulmat voitaisiin varastoida eri tietokantoihin. Tämä johtaisi kuitenkin päivitysten myötä datan epäjohdonmukaisuuteen. Integroidussa tietokannassa suunniteltavan objektin eri näkökulmat täytyy varastoida ja ne täytyy olla helposti saatavilla.

Monimutkaisten rakenteiden varastolla tarkoitetaan sitä, että CAD-malleissa on monimutkaisia rakenteita, joissa on toisiinsa yhteydessä olevia entiteettejä. Tämä on tärkeä osa CAD:in informaatiota. Redundanssin minimoinnilla tarkoitetaan sitä, että CAD-tietokanta täytyy suunnitella niin, että datan toistuvuutta on mahdollisimman vähän. CAD-datassa on yleensä paljon toistuvaa dataa. [Katragadda 1999]

Liu [2003] mainitsee samat vaatimukset kuin Katragadda [1999], mutta näiden lisäksi vaatimuksena on myös tehokas pääsy dataan. Tämä tarkoittaa, että sekä tiedonhaun että päivitysten tulee olla nopeaa ja helppoa. CAD-tietokannasta haetaan usein informaatiota, ja dataan tulee päästä helposti ja nopeasti. Tätä voidaan helpottaa tehokkailla säilytysmekanismeilla.

Anumba [1996] määrittelee seuraavia vaatimuksia CAD DBMS:lle:

- Tuki dynaamiselle mallin määrittelylle, joka heijastaa suunnittelun kehittyvää luonnetta, jota ei voida määrittää etukäteen. Suunnittelijan tulee pystyä määrittelemään uusia dataluokkia, jalostamaan niitä tai määrittämään ne uudelleen ja käyttämään tallennettua dataa koko suunnitteluprosessin ajan.
- CAD-mallin määritelmään sisältyviä laajempia tietotyyppejä tulisi tukea. Tähän liittyy läheisesti kyky käsitellä riittävästi sekä graafista että ei-graafista suunnitteluinformaatiota.
- CAD-tietokannan tulee pystyä käsittelemään suuria määriä CAD:iin liittyvää dataa. Tämän datan monimutkaiset keskinäiset suhteet on myös esitettävä riittävästi; tämä varmistaa tehokkaan tiedonhaun ja rajoittaa vastausaikoja.
- Datan eheyden ylläpito on äärimmäisen tärkeää. CAD-data muuttuu koko ajan, joten vaaditaan mekanismi tietojen eheyden ja johdonmukaisuuden ylläpitämiseksi.
- Versionhallinnan on mahdollistettava vaihtoehtoisten suunnittelusuunnitelmien kehittäminen, jotka voidaan arvioida globaalisti ennen tietokannan pysyvää päivittämistä
- CAD-tietokannassa tarvitaan mahdollisuus pitkien tapahtumien käsittelyyn. Insinööri työskentelee yleensä pitkiä aikoja datajoukon parissa interaktiivisessa tilassa; päivitystoimissa on otettava tämä huomioon.

- Suunnitteluprojektissa on yleensä mukana useita osapuolia, ja CAD-tietokannan on tarjottava samanaikainen pääsy. Tämä usean käyttäjän pääsy CAD-tietokantaan taas aiheuttaa jonkinlaisen suojaus- tai kulunvalvontatoiminnon tarpeen tietojen vioittumisen tai häiriöiden estämiseksi.
- Suunnitteluprosessin alustava ja iteratiivinen luonne vaatii suurta joustavuutta, jota CAD-tietokannan tulisi tarjota. Suunnittelijaa ei saa rajoittaa yritys ja erehdys -syklin läpikäymisessä.
- Teknisessä suunnittelussa on joskus tarpeen nähdä dataa useista eri näkökulmista tai eri tasoilla. Näin ollen CAD-tietokannan pitäisi pystyä tukemaan objektien moniesitystä/moninäkymää ja yksityiskohtatason säätöä. Tarkasteltava data pysyy pohjimmiltaan samana, mutta järjestelmä pystyy välittämään suunnittelutiedon eri näkökohtia loppukäyttäjälle/suunnittelijalle hänen näkökulmastaan riippuen.
- CAD-tietokannan pitäisi tyydyttää tarve saada informaatiota, joka on johdettava tallennetusta datasta. Tämä edellyttää kykyä upottaa semanttista informaatiota tai tiedon kapselointia tietokantaan. Tietoon perustuvan lähestymistavan odotetaan helpottavan informaation päättelyä ja päätöksentekoa.
- CAD DBMS tulisi olla helppokäyttöinen ilman ennakkotietoa tietokannan arkkitehtuurista ja tarjota helpon pääsyn asiaankuuluvaan CAD-dataan. [Anumba 1996]

3.4 CAD-tietokantojen erityispiirteet

Suurin osa CAD-järjestelmistä käyttää niiden omaa tiedonhallintaa käyttöjärjestelmän tiedostojärjestelmän päällä. Ne järjestelmät, jotka on rakennettu relaatiotietokannan päälle, käyttävät niitä lähinnä vain valintoihin ja joskus projektioihin. Liitokset sen sijaan tehdään yleensä suunnittelutyökalun yksityisessä ohjelmanmuistissa. CAD-järjestelmät käyttävät tietokannan hallintajärjestelmää indeksipakettina tukemaan assosiatiivista pääsyä, silloin kun ne ylipäänsä niitä käytävät. Kaupallisia tietokantajärjestelmiä ei pidetä riittävän nopeina tukemaan simulaattoreita tai interaktiivisia suunnittelutyökaluja. [Maier 1986]

CAD DBMS vaatii samanlaisia laitteistoja kuin perinteinen DBMS, mutta sen on oltava paljon joustavampi selviytyäkseen CAD-tietojen erityispiirteistä [Anumba 1996]. CAD-tietokannat eroavat perinteisistä tietokannoista erityisesti siinä, että perinteisissä tietokannoissa entiteetit ja niiden väliset suhteet tiedetään etukäteen. Suunnittelutietokantojen yhteydessä tämä oletus on kohtuuton. Suunnitteluprosessin edetessä lisätään entiteettejä (eli objekteja) ja suhteita eri entiteettien välillä. Suunnittelutietokantojen tulee siis pystyä käsittelemään tiedon dynaamista luonnetta ja keskinäisiä suhteita. [Du ja Ghanta 1987]. Yksittäisiä CAD-sovelluksia varten on kehitetty lukemattomia ad hoc -tietokantoja. [Yagiu 1991]

CAD-järjestelmät tarjoavat myös mahdollisuuden kehittää tuotteen valmistukseen tarvittavan tietokannan. Valmistustietokanta (manufacturing database) on integroitu CAD/CAM-tietokanta. Se sisältää kaiken datan, jota tuotteesta on luotu suunnitteluprosessin aikana. [Sarcar *et al.* 2008, s. 18]

CAD-suuntautuneiden tietokantajärjestelmien pääominaisuus on Rieun ja Nguyen [1986] mukaan integroida seuraavat asiat: monimutkaisten rakenteiden mallintaminen, spesifit tekniset työkalut, johdonmukaisuuden kontrollointi- ja täytäntöönpanosäännöt sekä semanttiset suhteet objektien välillä.

CAD-sovellukset käsittelevät laajempaa valikoimaa tietotyyppejä, mukaan lukien graafista, tekstiä, menettelyä koskevia ja muita tietoja. Eri CAD-tietoelementtien väliset suhteet ovat paljon monimutkaisempia kuin suhteellisen yksinkertaiset liiketietoelementtien väliset suhteet. CAD-tietoelementtien välillä on monia syklisiä, rekursiivisia ja muita objektikohtaisia suhdetyyppejä, ja ne on esitettävä sopivasti. [Anumba 1996]

Datan luonne voi vaihdella tiettyjä graafisia objekteja edustavasta tietorakenteesta eri osien kustannuksia kuvaaviin taulukoihin. Entiteettien väliset suhteet ovat yleensä monesta moneen. Yksinkertainen esimerkki on rajaviiva, johon kuuluu useita pintoja. Myös järjestys on tärkeää. Esimerkiksi tilanteessa, jossa kaarre koostuu useista käyräsegmenteistä. Segmenttien järjestyksellä on merkitystä. [Neumann 2005]

Rakenteet ovat usein hierarkkisia useissa tasoissa. Esimerkiksi virtapiiri, joka koostuu osapiireistä, jotka taas koostuvat osapiireistä jne. CAD-ohjelman täytyy pystyä esittämään osapiirit vaihtelevalla monimutkaisuudella eri hierarkian tasoilla. [Neumann 2005]

Datan johdonmukaisuus on myös oleellinen asia CAD-tietokannoissa. Niissä voi esiintyä toissijaista dataa. Jos ensisijainen data muuttuu, täytyy toissijainen data mitätöidä. Päivityksiä ei pitäisi suoraan kohdistaa toissijaiseen dataan. [Neumann 2005] Toissijainen data on datalähde, joka ei toimita dataa suoraan sen kerääjälle [Gesang 2020].

Käyttäjä esittää yleensä paljon kyselyitä tietokannalle, mutta jokainen yksittäinen kysely esitetään suhteellisen harvoin. Ad hoc -kyselyitä täytyy tukea ohjelmointikäyttöliittymässä. Datan ulkoiselle mallille täytyy olla selkeä konsepti, koska suunnitteluprosessissa on yleensä mukana useiden eri alojen asiantuntijoita, ja he voivat tulkita samaa dataa eri tavalla. [Neumann 2005]

Suunnitteluprosessi on interaktiivinen proseduuri. Usein löydetään uusia riippuvuuksia entiteettien ja attribuuttien välille, jonka seurauksena käsitteellistä tietomallia täytyy muuttaa. [Neumann 2005]

CAD-ohjelmassa käyttäjä käsittelee objekteja, kuten "siipi", "transistori" jne. ja tietokone käsittelee bittijonoja. Abstraktit objektit on kartoitettava tietokoneen sisäiseen esitykseen. Abstraktien objektien kartoittamiseen on olemassa kaksi tasoa: 1. Käyttäjä on kartoitettu informaation keskeiseen käsitemalliin. 2. käsitemalli on kartoitettu johonkin fyysiseen esitykseen. Kahta eri tasoa käytetään, koska eri käyttäjillä on eri havainnot da-

tasta. Kaikille käyttäjille ei anneta pääsyä kaikkeen arkaluonteiseen informaatioon. Suunnitteluprosessin aikana voidaan luoda uusia dataobjekteja, joiden ei pitäisi vaikuttaa ohjelmiin, joille nämä uudet kohteet eivät ole merkityksellisiä. [Neumann 2005]

CAD-sovellusten tietorakenteet ovat paljon monimutkaisempia kuin perinteiset sovellukset. Manipuloidut objektit määritellään usein muiden rakennettujen objektien yhdistämisen perusteella, ja objektisuhteita on monenlaisia. [Santana *et al.* 1989] Lisäksi objektien välillä on keskinäisiä riippuvuuksia, jotka voidaan esittää oikein puurakenteissa vain asettamalla rakenteeseen lisärajoituksia tai toistamalla yhtä tai useampaa ominaisuutta ja/tai attribuuttia [Iafrate *et al.* 1988].

Suunnittelu on tiimityötä. Jokainen tiimin jäsen voi aloittaa tapahtuman, joka sisältää suuren määrän dataa ja kestää pitkään. Suunnittelusovelluksissa tapahtuvia tapahtumia ei näin ollen voida pitää palautuksen yksikkönä, koska huomattava määrä työtä voidaan perua vian sattuessa. [Ketabchi ja Berzins 1987]

Ketabchin ja Berzinsin [1987] mukaan suunnittelutapahtumiin liittyvän datamäärän ja näiden tapahtumien keston vuoksi tavanomaisia lukitus- ja aikaleimatekniikoita ei voida käyttää samanaikaisuuden hallintaan.

Perinteisissä DBMS:ssä objektin käyttöikä alkaa, kun se lisätään tietokantaan, ja päättyy, kun se poistetaan tietokannasta. Sen sijaan suunnittelusovellukset käsittelevät myös kohteita, jotka elävät kauemmin kuin tietokannan pysyvät objektit. Nämä objektit edustavat yleisiä primitiivisiä komponentteja, jotka ovat käytettävissä useammassa kuin yhdessä suunnittelutietokannassa. [Ketabchi ja Berzins 1987]

Suunnittelusovellusten tiedon jakamisessa on erityispiirteitä. Yleisiä primitiivisiä komponentteja kuvaavat tiedot jaetaan useiden suunnitteluprojektien tietokantoihin. Suunnitteluprojektin tiedot jaetaan suunnittelutiimin jäsenten kesken. Yhden suunnittelijan käsittelemiä tietoja ei voi käyttää toinen suunnittelija. [Ketabchi ja Berzins 1987]

Alan laajuisten standardien puute pakottaa insinööritoimiston, joka tekee sopimuksen tietystä asiakkaasta, omaksumaan kyseisen asiakkaan käytännöt piirustuksia tai muita asiakirjoja laatiessaan. Jokaisella yrityksellä on oma menettelytapansa esimerkiksi otsikkolohkojen (piirustukseen sisältyvän laitteen yhteenvetotiedot), muistiinpanojen, nimikkeiden ja putkinumeroiden muotoiluun. Tämä edellyttää erityistä monipuolisuutta CAD-järjestelmältä ja sen tietokannalta. Esimerkiksi pumpun otsikkolohko voi koostua laitenumerosta, laitteen nimestä, virtausnopeudesta, paine-erosta imu- ja poistovirtauksen välillä sekä nesteen tiheydestä. Jotkut yritykset haluavat kaiken tämän tiedon piirustukseen, toiset haluavat vain laitenumeron ja sen palvelun. Jos virtausnopeus otetaan mukaan, on olemassa useita mahdollisia yksiköitä, joissa se voidaan näyttää, jotka voivat olla erilaisia kuin yksiköt, joissa tiedot on tallennettu tietokantaan. [Buchmann ja Perez de Celis 1985]

3.4.1 Alitietokannat ja muut osat

Vuorovaikutteisten suunnittelutapahtumien pituus CAD-järjestelmissä on tärkeä syy alitietokantojen generoinnissa. Tämän lisäksi ei pidä unohtaa datan käyttötapoja, sillä useiden sovellusten on päästävä samaan dataan. Mikään yksittäinen tietokanta ei voi tehokkaasti tukea tarvittavia useita pääsypolkuja. Oliomallin järjestelmässä fyysinen klusterointi on mahdollista vain yhden kriteerin mukaan ja kaikki muu klusterointi on tehtävä indeksointimekanismien kautta. Poimimalla osia globaalista tietokannasta ja jäsentämällä ne haluttujen pääsypolkujen mukaan, voidaan vasteaikaa parantaa huomattavasti. [Buchmann ja Perez de Celis 1985] Oliomallin tietokannasta on kerrottu tarkemmin luvussa 4.4.

Yksi syy alitietokantojen luomiseen liittyy datan laatuun. Projektin laajuinen tietokanta on monotonisesti kasvava kokoelma hyväksyttyä dataa. Teknisesti hyväksytyllä datalla tarkoitetaan dataa, joka on käynyt läpi usean eri tason varmentamisen ja valtuutuksen, yleensä suunnittelijan, ryhmänjohtajan, projektipäällikön ja asiakkaan edustajan toimesta. Hyväksyttyyn dataan tehdyt muutokset on kirjattava versioina ja muutoksen on hyväksyttävä sama hyväksyntähierarkia. Koska malli hyppää harvoin edestakaisin versioiden välillä, on hyväksyttävää, että vanhat versiot puretaan, jotta tietokanta ei sotkeudu. Data, joka ei ole saavuttanut täydellistä hyväksyntää, on merkittävä sellaiseksi, jotta johdetut mallit katsottaisiin myös alustaviksi ja edellyttävät lopullista hyväksyntää sille datalle, josta ne on johdettu. Vaikka koko tietokannan päivittämiselle ei olisikaan datan laaturajoituksia, skeeman pelkkä koko ja datan määrä suuressa projektissa tekevät mahdottomaksi sallia suoria päivityksiä projektin laajuiseen tietokantaan muutoin kuin lopullisten suunnitelmien integroimisen. [Buchmann ja Perez de Celis 1985]

Buchmannin ja Perez de Celisin [1985] mukaan alitietokannat voidaan jakaa kolmeen eri tyyppiin: paikallinen tietokanta, työaluetietokanta ja testialue (scratchspace). Niitä kaikkia voidaan kuitenkin kutsua työalueeksi (workspace).

Paikalliset tietokannat ovat otteita koko hankkeen laajuisesta hyväksytystä tietokannasta ja niiden sisällön määrittelevät sovellukset, joille paikallinen tietokanta on tarkoitettu. Poimittavat objektit määräytyvät sovelluksen tyypin ja sovelluksen esiintymän mukaan. Paikallinen tietokanta rakennetaan fyysisesti uudelleen tietohakemistossa olevien käyttötietojen perusteella tietyn sovelluksen suorituskyvyn optimoimiseksi. [Buchmann ja Perez de Celis 1985]

Paikallinen tietokanta on vakaa kopio, josta voidaan luoda muita kopioita yksinkertaisella tiedostojen kopiointipyynnöllä. Tämä on paljon nopeampaa kuin uuden tiedoston poiminta suuresta projektin laajuisesta tietokannasta. Toiseksi, jos suunnittelija päättää, että hänen viimeisen tunnin aikana tekemänsä suunnittelu on epäonnistunut, hän voi aloittaa alusta vakaasta paikallisesta tietokannasta eikä tarvitse kumota jo tehtyjä muutoksia. On myös tärkeää pitää mielessä tietojen laadun ja varmennusprosessin eri tasot. Usein on helpointa todentaa suunnittelu suorittamalla uudelleen yhdistetty loki (consolidated log) suunnittelijan tekemistä suorituksista. Yhdistetty loki on loki, joka on pelkistetty vain

niihin päivityksiin, joilla on pysyvä vaikutus. Jos ohjaaja havaitsee virheitä tai jos hänellä on parempi idea, hän voi keskeyttää joitain osia tarkastettavasta suunnittelusta. Jos tämä tehtäisiin koko projektin tietokannassa, se voisi aiheuttaa vakavia häiriöitä. [Buchmann ja Perez de Celis 1985]

Työaluetietokannat (workarea database) ovat paikallisen tietokannan kopioita, joissa suunnittelija suorittaa tarvittavat suunnittelutoimenpiteet. On mahdollista, että kaksi tai kolme suunnittelijaa työskentelee rinnakkain ja he saattavat haluta jokaiselle oman työalueensa. Työalueista on tehtävä yhteensopivia, ennen kuin ne integroidaan projektin laajuiseen tietokantaan. [Buchmann ja Perez de Celis 1985]

Testialue (scratchspace) on alue, jossa suunnittelija voi testata villiä ideaansa. Koska nämä ideat voivat riippua aiemmin suoritetuista suunnittelutoimista, jotka on jo tehty työalueella, hän saattaa haluta luoda yhden tai useamman "mitä jos" -kopion testatakseen erilaisia suunnitteluvaihtoehtoja. Vaikka työalueiden on oltava keskenään yhteensopivia, testialueet yleensä hylätään ja korkeintaan yksi valitaan oikeaksi ratkaisuksi. [Buchmann ja Perez de Celis 1985]

Tietohakemistolla voi olla kaksoistoiminto CAD-järjestelmissä: CAD-tietokantojen suunnittelun aikana ja järjestelmän käytön aikana. Tietokannan suunnittelun tukidatan tulee olla helposti tietokannan suunnittelijan saatavilla tietokannan suunnittelutyökalun kautta. Kuitenkin, kun tietokannan suunnittelu on valmis, tämä data voidaan ladata ja arkistoida. [Buchmann ja Perez de Celis 1985]

Käytön aikana tietohakemiston tulee sisältää tiedot datan määrittelystä, esimerkiksi yksiköistä, joihin data on tallennettu, ja datan käyttötiedot tukemaan klusterointialgoritmeja, jotka määrittävät, kuinka data tulee klusteroida paikallis- ja työaluetietokantoihin. Tietohakemiston tulee myös sisältää datan postitusluettelot, joita käytetään tukemaan päivityksen etenemismekanismia ja osoittavat, onko käyttäjän valtuutettava päivitys vai riittääkö pelkkä ilmoitus. Tietohakemisto sisältää myös datan käyttötiedot siitä, kuka on tarkastanut minkäkin objektin, jotta käyttäjälle voidaan ilmoittaa muutoksista, jotka koskevat häntä. [Buchmann ja Perez de Celis 1985]

Kaavaintietokanta (template database) sisältää tietohakemiston tavoin metadataa. Näiden ero on siinä, että kaavaintietokannassa mallitiedot ovat projektikohtaisia, ja ne on räätälöitävä jokaiselle suunnitteluyrityksen asiakkaalle. Tietohakemistossa taas metadata on melko vakaata ja kuvastaa suunnitteluyrityksen menettelytapoja. [Buchmann ja Perez de Celis 1985]

Luettelo (catalogue) ja tukitietokanta sisältävät standardoituja teknisiä komponentteja. Ne ovat sekä apu suunnittelijalle, jotta hän voi valita niistä kelvollisia komponentteja, että rajoitusten lähde validointimekanismille. [Buchmann ja Perez de Celis 1985]

Rajoituspohjat ovat rajoitusten ja näiden rajoitusten poikkeuksien arkistot. Rajoitukset tallennetaan merkkijonoina, joissa on rajoitteen kuvaus ja erityisesti projektikohtaisten rajoitteiden osalta viitearvot voidaan ottaa voimassa olevista tukitietokannoista. [Buchmann ja Perez de Celis 1985]

3.4.2 Rajoitusten erityispiirteet

Perinteisten rajoitusten, kuten viittaus- ja toimialuerajoitusten, lisäksi suunnittelusovellusten on täytettävä toinen ryhmä rajoituksia, joita kutsutaan suunnittelusäännöiksi. Suunnittelusääntöjen tarkistaminen vaatii yleensä huomattavan määrän dataa ja laskentaa. Toisin kuin perinteiset rajoitukset, joita DBMS voi valvoa automaattisesti, suunnittelusäännöt tarkistetaan yleensä suunnittelutyökaluilla, eikä niitä voida noudattaa automaattisesti. [Ketabchi ja Berzins 1987]

Johdonmukaisuusrajoituksia pidetään joskus keinona validoida tietokannan tiedot tai ainakin toimenpiteiden spesifikaatioina, jotka estävät tietokantaa joutumasta epäjohdonmukaiseen tilaan. Itse asiassa rajoitukset ovat olennainen osa tietokannassa esitettyjen objektien määritelmää. Rakenteelliset kuvaukset ja johdonmukaisuusrajoitukset täydentävät toisiaan. [Lacroix ja Pirotte 1981]

Teoreettiselta kannalta katsottuna on sama asetetaanko tietorakennemääritellyt osat rakennekuvauksiin vai johdonmukaisuusrajoituksiin. Siksi useat erilaiset tietorakenteen määrittelystrategiat ovat periaatteessa samanarvoisia. Ne eroavat toisistaan vain siinä, kuinka suuri osa sovelluksen semantiikasta on rakenteellisissa säännöissä ja kuinka paljon ilmaistaan rajoituksilla, jotka rajoittavat rakenteellisia sääntöjä. Tällainen vapaus ei välttämättä ole hyvä asia määrittely- ja suunnitteluprosessin kokonaistehokkuuden kannalta. [Lacroix ja Pirotte 1981]

Strategialla, jossa rakenteelliset säännöt valitaan niin, että ne sopivat mahdollisimman tiukasti todellisiin esineisiin, on useita etuja:

- Monissa toteutuksissa ja ainakin intuitiivisesti rakenteelliset säännöt määrittelevät mallit tietojen järjestämiselle, kun taas rajoitukset sanovat, että jotkin datayhdistelmät näiden mallien sisällä eivät ole sallittuja. Rakenteelliset säännöt muodostavat siten datan kuvauksen selkärangan. Mitä korkeammalla tasolla tällainen datakuvaus on, sitä yksinkertaisempaa on dokumentaatio, joka yhdistää datakuvauksen sovellusuniversumiin.
- Yrittämällä laittaa rakenteellisiin kuvauksiin mahdollisimman paljon semantiikkaa, eri suunnittelijat pääsevät todennäköisemmin yksimielisyyteen käsiteskeeman "standardista".
- Strategioiden ongelmat, jotka rakentavat löysät rakenteelliset kuvaukset, tulevat hyvin vakaviksi, kun rajoituksia ei kuvata sellaisenaan. Tällaisia käytäntöjä kohdataan ympäristöissä, joissa ajattelutyökalut ovat tietueiden ja osoittimien tasolla. Usein johdonmukaisuusrajoitteet hautautuvat monimutkaisiin tulkintasääntöihin, jotka dokumentoivat tietorakenteita silloin, kun dokumentaatiota on saatavilla. Huonossa tapauksessa dokumentaatio on vain sovellusasiantuntijoiden päässä. Käytännössä rajoitukset lakkaavat olemasta pakollisia, niistä tulee vain hyvän

käytännön sääntöjä, joita on usein erittäin vaikea valvoa tehokkaasti. [Lacroix ja Pirotte 1981]

CAD-sovelluksissa ei ole vain tärkeää pystyä käsittelemään rajoituksia tehokkaasti, vaan on myös tarpeen käsitellä poikkeuksia. Poikkeuksia pidetään tässä yhteydessä rajoitusten lieventämisenä. Rajoitusten ja poikkeuksien käsittelyyn on sama syy kuin attribuuttien periytymiseen semanttisessa tietomallissa: molempien on perittävä osaobjektit. Tämä on välttämätöntä, koska rajoitukset määritetään usein korkeilla abstraktiotasoilla. [Buchmann *et al.* 1986]

Voimakkaalla tyypityksellä, josta sekä perinteiset tietokantalähestymistavat että ohjelmointikielet ovat riippuvaisia rajoitusten täytäntöönpanosta, on kaksi perusseurausta: Rajoitukset on tarkistettava aina, eikä niitä voi poistaa väliaikaisesti käytöstä, ja tietokannan suunnittelijan (tai ohjelmoijan) täytyy ennakoida kaikkia mahdollisia rajoituksia määritelmän uudelleenkääntämisen rangaistuksen vuoksi. [Buchmann *et al.* 1986]

CAD-tietokanta voi sisältää jopa yli 10 000 attribuuttia. Tästä syystä on mahdotonta määrittää tietokannan suunnitteluvaiheessa kaikkia rajoituksia, jopa niitä, jotka voidaan tyypillisesti määrittää luokkatasolla. Lisäksi on olemassa useita rajoituksia, jotka on määriteltävä tietylle projektille millä tahansa abstraktiotasolla. Nämä erityiset rajoitukset riippuvat useista tekijöistä, kuten asiakasyrityksen suunnittelukäytännöistä, maantieteellisen sijainnin asettamista rajoituksista, materiaalien ja komponenttien saatavuudesta ulkomailla, paikallisista suunnittelusäännöistä jne. Monet näistä rajoituksista neuvotellaan asiakkaan kanssa samalla, kun suunnittelu on jo alkanut ajan säästämiseksi. [Buchmann *et al.* 1986]

CAD-järjestelmät on kehitetty geneerisinä työkaluina, ja niitä voidaan räätälöidä vain tietyn yrityksen tai käyttäjäryhmän tarpeisiin "oppimisjakson" kautta, jonka aikana rajoituksia lisätään ja säädetään niin, että järjestelmä voi toimia yrityksen odotusten mukaisesti. Suurin ongelma tässä lähestymistavassa on se, että avaimet käteen -periaatteella CAD-järjestelmän ostavilla suunnitteluyrityksillä on harvoin henkilökuntaa tekemään muutoksia skeematasolla ja suorittamaan laajoja uudelleenkäännöksiä ja ohjelmistotestejä taustalla olevasta DBMS:stä. Siksi tällä periaatteella toimivan CAD-järjestelmän perustana olevan DBMS:n tavoitteena tulisi olla rajoitusten dynaamisen määrittelyn mekanismi, joka edellyttää käyttäjältä vain asiaankuuluvaa insinööriosaamista, eikä vaadi laajaa tietokantajärjestelmän tuntemusta ja uuden kielen oppimista rajoitusten määrittelemiseen. [Buchmann *et al.* 1986]

Toinen tyypillinen CAD-tietokantojen ominaisuus, jolla on vaikutusta rajoitusten tarkistustoimintoon, on se että tapahtumat ovat poikkeuksellisen pitkiä. Tämä voi johtaa seuraaviin tilanteisiin:

- Tapahtuma koostuu usein suuresta määrästä ydintoimintoja ja suunnittelija saattaa haluta tarkistaa työnsä johdonmukaisuuden muissakin kohdissa kuin tapahtuman lopussa.

- Tietokanta on juuri täytetty suunnittelusovelluksissa ja yhdelle attribuutille määritetty rajoitus ei ehkä ole vielä pakotettavissa, koska yhtä tai useampaa rajoitteeseen vaikuttavaa attribuuttia ei ole määritetty.
- Joidenkin rajoitusten tarkistusten monimutkaisuus ja suunnitteluprosessin vuorovaikutteisuus tekevät usein mahdottomaksi arvioida rajoituksia lennossa, koska tämä voi olla liian aikaa vievää ja johtaa suunnittelijan ajatteluprosessin keskeytykseen.
- Suunnitteluvaiheesta riippuen rajoitusten rikkominen voi vaatia erilaisia toimia. Tapahtuman peruuttaminen rajoitusten rikkomisen vuoksi, voi aiheuttaa useiden tuntien tai jopa päivien työmäärän menettämisen. Siksi suunnittelijalla tulisi olla mahdollisuus määritellä suoritettavat toimet.
- Monet suunnittelutietokannan päivitykset, erityisesti työaluetietokannassa, ovat luonteeltaan vain alustavia ja niitä muutetaan tai mitätöidään myöhemmin. Näiden alustavien päivitysten rajoitusten tarkistaminen on ajan haaskausta. [Buchmann *et al.* 1986]

Rajoitusten tarkistus on aikaa vievä tehtävä ja monet rajoitukset voivat muuttua vääriksi tai määrittämättömiksi, ellei suunnittelu ole edennyt tiettyyn valmiusasteeseen. Tästä syystä on toivottavaa, että käytössä on globaali kytkin, jonka avulla käyttäjä voi työskennellä ilman rajoitusten tarkistamista työaluetietokannassa, kunnes hän haluaa varmistaa johdonmukaisuuden. Rajoituksen tarkistuskytkin tulee määrittää istunnon alussa. Jos rajoitusten tarkistus on valittu jokaiselle päivitykselle, käyttäjällä tulee olla oikeus määrittää, haluaako hän tehdä takautuvan rajoitteen tarkistuksen ennen kuin hän jatkaa vai haluaako hän tarkistaa vain nykyisestä tilasta eteenpäin. Jokaisessa työaluetietokannassa on tilakenttä, joka on asetettu arvoon "epäluotettava", elleivät kaikki rajoitukset ole tosi joko alkuperäisessä muodossaan, tai koska poikkeus on määritetty. Jos päivitys suoritetaan ilman rajoitusten tarkistusta, tilaksi asetetaan "epäluotettava", kunnes yleinen rajoitusten tarkistus suoritetaan. [Buchmann *et al.* 1986]

4 Tietokantamallit CAD:ssa

Tässä luvussa käsitellään tietomalleja ja tietokantamalleja. Tietomalli kuvaa dataa ja sen määritelmää. Tietomalli on konsepteihin perustuva logiikka, kun taas sen toteutus on DBMS. [Naik 2013, 2.2] Tietokantamalli taas tarkoittaa tietokannan kuvausta ja strukturointiformalismia [Hammer ja McLeod 1981]. Tietokantamalleja on paljon erilaisia.

Tietomallin spesifisyys koskee esimerkiksi seuraavia asioita [Lacroix ja Pirotte 1981]:

- Tietorakenteiden määrittämiseen käytettävissä olevien tärkeimpien ja ilmeisimpien konstruktioiden luonne.
- Pääasiallisten datan strukturointirakenteiden teho tai rikkaus, eli tarkkuus, jolla sovellusuniversumin yksityiskohtainen rakenne voidaan mallintaa.
- Luonnollisuus tai riittävyys, jolla strukturointirakenteet mahdollistavat erilaisten esineiden ja tilanteiden mallintamisen sovellusalueen asiantuntijoiden käsityksen mukaan.

Oletetaan, että tietokanta kuvastaa oikein tavoiteltua minimaailman (mini word) semantiikkaa. Tietomallien jäykän kehyksen vuoksi minimaailman ja sen tietokantaesityksen välille jää edelleen semanttinen aukko. Toisin sanoen on yleensä mahdotonta esittää kaikkea kiinnostavaa semantiikkaa tietokannassa. Tietokantaa käyttävien sovellusohjelmien on kaapattava "jäännös" ja/tai se on osa käyttäjän itsensä tekemää tulkintaa tietokantakyselyjen tuloksesta. Tietokantajärjestelmien perimmäisenä tavoitteena on kuitenkin tarjota käsitteitä, jotka mahdollistavat semanttisen aukon pitämisen mahdollisimman pienenä ja täten mahdollistaa tärkeimpien semantiikkojen esittämisen tietokannassa itsessään. [Dittrich 1986]

Kalay [1985] määrittelee viisi kriteeriä tietokantamallille CAD-järjestelmissä: löytyvyys (completeness), kompaktius (conciseness), yksinkertaisuus, joustavuus ja laajennettavuus.

Löytyvyys on tärkein kriteeri. Sillä tarkoitetaan sitä, että mallin täytyy pystyä esittämään ja mahdollistaa pääsy kaikkeen objektiin liittyvään informaatioon. Jos malli rajoittaa esitettävän informaation typpiä, se ei ole yleiskäyttöinen. [Kalay 1985]

Kompaktius on tärkeä ominaisuus informaation tallentamisessa ja kommunikoinnissa datalinkkien välillä. Yksinkertaisuus on tärkeä mallin yleiskäyttöisyyden kannalta. Useilla käyttäjillä voi olla tarve päästä dataan käsiksi, eikä kaikilta voida olettaa olevan ymmärrystä monimutkaisista, toisiinsa liittyvistä tietorakenteista kuten niistä, joita käytetään data-päivitys-malleissa. Jos data esitetään loogisesti yksinkertaisessa muodossa edistää se sen hyväksyntää ja käyttöä. [Kalay 1985]

Joustavuus vastaa dataa käyttävien eri ohjelmien erilaisiin tarpeisiin. Se mahdollistaa kyselyn ja datan talteenoton kyseisen ohjelman kannalta sopivimmassa muodossa mahdollisesti uudelleenjärjestelemällä data. Laajennettavuus on tärkeä kriteeri, kun ohjelmavalikoiman, jossa mallia voidaan käyttää, oletetaan olevan laaja. Laajennettavuus mahdollistaa uusien data-alkioiden ja attribuuttien lisäämisen olemassa oleviin alkioihin mitätöimättä edellisiä. [Kalay 1985]

Du ja Ghanta [1987] määrittelevät seuraavia vaatimuksia hyvälle CAD-tietokantamallille. Ensimmäinen vaatimus: Tietokannan täytyy olla dynaaminen. Ohjelmat vaihtelevat merkittävästi rakenteeltaan ja sisällöltään, joten mallin on tuettava kehittyvää suunnittelu dataa.

Toinen vaatimus on objektin integroitu käsittely. Riippumatta siitä, onko objekti primitiivinen vai ei, ja riippumatta siitä, onko objekti eksplisiittisesti tallennettu vai ei, kaikkia objekteja on käsiteltävä yhtenäisellä tavalla.

Kolmas vaatimus: Rajoituksen määrittely ja käsittely. Mallin tulisi mahdollistaa rajoitusten helppo määrittely. Automaattista rajoitusten leviämistä tulisi tukea, aina kun mahdollista. Jos leviäminen ei jostain syystä ole mahdollista, järjestelmän tulee raportoida käyttäjälle rajoituksista, joita suunnittelijan ehdottama muutos suunnitteluun aikoo rikkoa. Suunnittelijan on lisättävä ja poistettava rajoituksia joustavasti järjestelmän objektien välillä suunnittelun edetessä. Itse asiassa rajoitukset muodostavat hierarkian. Joitakin rajoituksia on käsiteltävä useammin kuin toisia. Niiden laajuuden perusteella joitakin on noudatettava jokaisen pienenkin tapahtuman jälkeen. Muita saatetaan väliaikaisesti rikkoa. Rajoituksilla voi kuitenkin olla erilaiset soveltamisalasäännöt, jotka niiden tulisi täyttää. Joissakin tapauksissa suunnittelija saattaa tahallaan haluta olla täyttämättä joitain rajoituksia väliaikaisesti, koska suunnittelun valmistuttua nämä rikotut rajoitukset täyttyvät. Tällaisessa tapauksessa ei ole mitään syytä, miksi järjestelmän pitäisi vaatia joitakin pikakorjauksia täyttääkseen rajoitukset väliaikaisesti keskeneräiseen suunnitteluun.

Neljäs vaatimus on tuki erityyppisille datoille. Koska suunnitteluohjelmissa ei useinkaan ole mahdollista saada tietoja vain yhdessä muodossa joustavuutta menettämättä, mallin on sallittava tiedot eri muodoissa, kuten grafiikkadata, toiminnallinen data ja looginen data.

Viides vaatimus on nimien ja määritelmien uudelleenkäytettävyys. Mallin ei pitäisi rajoittaa käyttäjää nimeämään objektejaan siten, että ne ovat koko järjestelmässä yksilöllisiä. Eri tarkoituksiin käytettävien objektien tai määritelmien nimiä tulisi voida käyttää uudelleen kunhan ne ovat ainutlaatuisia kontekstissaan tai näkymässään.

Kuudes vaatimus on objektien jaettavuus. Tehokkuussyistä (kuten tallennusvaatimusten vähentäminen tai vaadittujen tietorakenteiden koon pienentäminen) virhekorjattuja suunnitteluosia, joista tulee osa julkista kirjastoa, on käsiteltävä primitiivisinä objekteina ellei halua selvittää yksityiskohtia kyseisestä objektista. Samoin on parasta olla kutsumatta ja luomatta eksplisiittistä kopiota osaobjektista, ellei ainakin yhtä sen arvoista ole muutettu.

Seitsemäs vaatimus on tuki objektin eri versioille. Kahdeksas vaatimus on operaatioiden laajennettavuus. Suunnittelijan on parannettava tai mahdollisesti määritellä uudelleen joitain mallin oletusarvoisesti tarjoamia toimintoja. Mallissa on oltava runsas joukko perustoimintoja sekä mahdollisuus järjestelmän ominaisuuksien parantamiseksi, koska on erittäin vaikeaa tarjota kaikkia haluttuja ominaisuuksia ja toimintoja, joita kaikki järjestelmän käyttäjät tarvitsevat.

Yhdeksäs vaatimus on tallennusrakenteiden laajennettavuus. Tallennusrakennetta, joka olisi paras mahdollinen kaikille sovelluksille, ei ole olemassa. Tästä syystä järjestelmän on sallittava datan hallinta käyttäjän toimesta sen sijaan, että se pakottaisi hänet mukautumaan tarjottuihin rakenteisiin.

Kymmenes vaatimus on tehtävätapahtuman hallinta. Eri suunnitelmien erilaiset tapahtumat on koordinoitava, jotta koko järjestelmän laajuinen toiminta jatkuisi sujuvasti. Perinteisissä tietokannoissa tapahtumalla on kaikki tai ei mitään -tulkinta. Tämä ominaisuus ei sovellu joihinkin suunnittelutapahtumiin, jotka vaativat jopa päiviä. Tapahtumahallinnan tulee siis käsitellä sekä tapahtumia, jotka ovat jossain määrin perinteisten tietokantojen tapahtumia muistuttavia, että niitä, jotka menevät päällekkäin ja kattavat pidemmän ajanjakson, jolloin lukitseminen ei ole toivottavaa.

Yhdestoista vaatimus on samanaikaisuuden kontrollointi. Projektin koon kasvaessa joudutaan jossain vaiheessa siirtymään hajautettuun ympäristöön. Mallin on kyettävä käsittelemään eri suunnittelijoiden eri tapahtumien rinnakkaisoperaatioita ilman umpikujaa ja muita vastaavia ei-toivottuja ominaisuuksia, samalla kun säilytetään maksimaalinen rinnakkaisuus.

Kahdestoista vaatimus on käyttäjien välisen viestinnän integrointi tietokantaan: Suunnitteluprosessin aikana esiintyy usein epäjohdonmukaisuuksia, jotka johtuvat ihmisten välisen viestinnän epävirallisesta roolista. Tietokantamallin täytyy minimoida suunnittelun läpimenoaika ja vähentää kalliita virheitä, jotka livahtavat dokumentoimattomaan käyttäjäviestintään. Näin ollen mallin on käsiteltävä myös esimerkiksi asiaankuuluvat sähköpostit osana suunnittelutietoja.

Kolmastoista vaatimus on näkymien hallinta. Eri suunnittelijat huolehtivat järjestelmän eri osista. Näin ollen heillä on erilaisia näkemyksiä koko järjestelmästä. Epämuodollisesti näkymää voidaan pitää porttina tai ikkunana tietokannan osaan. Mallin tulee olla riittävän joustava mahdollistaakseen näkymän määrittelyn ja liikkumisen dynaamisesti sen sijaan, että tietokannan hallitsija (master) pakotettaisiin määrittelemään kaikki mahdolliset näkymät ennen suunnitteluprosessin alkamista.

Neljästoista vaatimus on palautuminen. Viat ovat väistämättömiä käytännön järjestelmissä. Mallissa tulee olla ominaisuuksia, kuten varmuuskopiointiproseduurit, palautuslohkojärjestelmät (recovery block schemes) ja vaiheittaiset vedokset, jotta minimoidaan mahdolliset virheistä johtuvat vauriot, toipumisprosessin aikana. Toisin sanoen malli ei saa olettaa laitteiston olevan ihanteellinen, joka toimii jatkuvasti.

Viidestoista vaatimus on suojaus ja turvallisuus. Suojauksen ja turvallisuuden välillä on pieni ero siinä, että toinen käsittelee tahattomia tekoja ja toinen tahallisia toimia. Ne molemmat ovat erittäin tärkeitä. Esimerkiksi suunnittelija ei ehkä halua muiden suunnittelijoiden tietävän hänen keskeneräisen objektinsa yksityiskohtia, tai suunnittelija voi nimenomaisesti kieltää pääsyn muilta suunnittelijoilta, joiden hän epäilee olevan haitallisia.

Nämä vaatimukset voidaan nähdä Dun ja Ghantan [1987] mukaan hyvinä tavoitteina tietokantamallille. Parhaassa tapauksessa malli täyttää kaikki vaatimukset. Seuraavissa kohdissa esitellään tietokantamallit, joita CAD-järjestelmissä käytetään.

4.1 Hierarkkinen tietokanta

Hierarkkinen tietomalli oli ensimmäinen tietomalli. Se kehitettiin 1960-luvulla. Hierarkkisessa tietomallissa entiteetit ja niiden väliset suhteet hallitaan puurakenteen avulla (juurellinen puu). Puurakenteessa vanhemmalla voi olla useita lapsia, mutta lapsella voi olla vain yksi vanhempi. Jos siis jollain entiteetillä on useita vanhempia, täytyy luoda useita erillisiä vanhempi solmuja, joiden lapseksi linkitetään kyseinen entiteetti. Yhtä entiteetin esiintymää kutsutaan segmentiksi. Juurena oleva segmentti on juurisegmentti. Muita segmenttejä kutsutaan riippuviksi segmenteiksi. [Naik 2013, 2.2]

Hierarkkisen tietokannan etuna on Naikin [2013, 2.2] mukaan se, että yhdestä moneen suhteet voidaan esittää tehokkaasti. Stephensin ja Plewin [2001, 2] mukaan etuna on datan hakemisen nopeus ja datan eheyden helpompi hallinta.

Huonoja puolia hierarkkisessa tietokannassa on Naikin [2013, 2.2] mukaan se, että monesta moneen suhteita ei voida esittää, koska lapsella voi olla vain yksi vanhempi. Jos segmentti poistetaan kaikki sen alle jäävät segmentit poistuvat. Myös segmentin päivittäminen on vaikeaa, koska niiden etsiminen on hankalaa. [Naik 2013, 2.2] Huonoja puolia Stephensin ja Plewin [2001, 2] mukaan on, että käyttäjän täytyy tuntea tietokannan rakenne hyvin. Tietokantaan myös tallennetaan tarpeetonta dataa [Stephens ja Plew 2001, 2] [Anumba 1996].

4.2 Verkkomallin tietokanta

Verkkomallin tietomallissa (Network datamodel) käytetään linkkejä tallenteiden (record) välissä. Jos tallenteet liittyvät toisiinsa monesta moneen suhteella, ne ovat yhdistetty toisiinsa liitintallenteella, jota kutsutaan joukoksi (set). [Naik 2013, 2.3]

Verkkomallin tietokannan etuna on se, että monesta moneen suhteet voidaan esittää helposti [Naik 2013, 2.3]. Dataan on helppo päästä käsiksi ja kaikkeen dataan päästään käsiksi mistä tahansa tallenteesta. Myös monimutkaisten kyselyiden luominen on helpompaa. [Stephens ja Plew 2001, 2]

Huonoja puolia ovat verkkomallin tietokannan monimutkaisuus [Naik 2013, 2.3] [Anumba 1996]. Tietokannan rakennetta on vaikea muokata [Stephens ja Plew 2001, 2]. Jos rakennetta kuitenkin muokataan, sillä on vaikutusta ohjelmiin, jotka tietokantaa käyttävät [Stephens ja Plew 2001, 2]. Huono puoli on myös se, että käyttäjän täytyy tuntea tietokannan rakenne [Stephens ja Plew 2001, 2].

Loogiset suhteet eivät salli datan strukturointia luonnollisella tavalla verkko- ja hierarkkisessa mallissa. Käyttäjän on ajateltava tietokoneisiin liittyviä käsitteitä, kuten fyysisiä pääsypolkuja ja loogisia osoittimia. [Santana *et al.* 1989]

4.3 Relaatiotietokanta

Relaatiotietokannassa data on tallennettu relaatioihin, joita kutsutaan tauluiksi. Relaatiot liittyvät toisiinsa suhteilla. Relaatio on yhdistelmä toisiinsa liittyviä kokonaisuuksien esiintymiä. Relaatio esittää oikean maailman entiteettejä tai entiteettien välistä suhdetta. Entiteettien väliset suhteet käsitellään pää- ja viiteavaimilla. Relaatio voidaan myös määritellä kaksiulotteisena rakenteena, joka koostuu monikoista ja attribuuteista. Entiteetti voidaan määritellä, niin että se on kokoelma toisiinsa liittyviä kokonaisuuksien esiintymiä. [Naik 2013, 2.4, 3]

Kokonaisuuksien esiintymät esitetään monikkona (tuple) relaatiossa. Jokainen rivi esittää yhtä monikkoa. Mahtavuus (cardinality) tarkoittaa monikkojen eli rivien määrää relaatiossa. Attribuutit ovat objektin ominaisuuksia. Jokaisessa kokonaisuuden esiintymässä on attribuutteja. Aste (degree) tarkoittaa attribuuttien määrää relaatiossa. Määrittelyjoukko (domain) tarkoittaa esimääriteltyä joukkoa, johon attribuutin arvo kuuluu. [Naik 2013, 2.4, 3] Uusia attribuutteja kuten myös uusia relaatioita voidaan lisätä ilman, että se vaikuttaa edellisiin [Kalay 1985].

Relaatiomallissa rivien järjestyksellä ei ole väliä. Jokaisen rivin täytyy olla uniikki. Tämä saavutetaan yleensä määrittelemällä uniikki tunnus jokaiselle monikolle. Sarakkeiden järjestyksellä on merkitystä, koska se vastaa relaation määrittelyjoukkojen järjestystä. Sarakkeet ilmaistaan nimeämällä ne kyseisen määrittelyjoukon nimellä. [Naik 2013, 2.4, 3]

Uniikkia tunnusta, jolla tunnistetaan rivi, kutsutaan avaimeksi. Avain käsittää yhden tai useamman attribuutin. Näitä attribuutteja kutsutaan avainattribuuteiksi. Avain voi esimerkiksi olla pääavain (primary key) tai viiteavain (foreign key). Pääavaimella tunnistetaan jokainen rivi. Pääavaimia voi olla relaatiossa vain yksi, mutta se voi sisältää useita attribuutteja. Nämä attribuutit eivät saa sisältää null-arvoa. Viiteavain on avain, joka vastaa toisen taulun, pääavainta. Sen täytyy siis olla samaa määrittelyjoukkoa, mutta attribuutin nimi saa olla eri. [Naik 2013, 2.4, 3]

Relaatiomalli esitettiin ensimmäisen kerran vuonna 1970 [Codd 1970]. Relaatiomallin yleistyessä se ratkaisi hierarkkiseen ja verkkomallin tietokantoihin liittyviä ongelmia.

Kun datan tyyppiä, ominaisuuksia tai rakennetta muutettiin, ohjelman piti muuttua. [Naik 2013, 2.4, 3]

Relaatiomallin tärkein panos on ollut osoittaa, että loogisten ja fyysisten tietorakenteiden kuvaus on selvästi erotettava toisistaan. Sen jälkeen syntyi tärkeitä käsitteitä, kuten tietojen riippumattomuus ja monitasoinen DBMS-arkkitehtuuri. Tuolloin ajateltiin, että relaatiomalliin voitaisiin suunnitella yksinkertaisia ja korkean tason tiedonkäsittelykieliä. [Lacroix ja Pirotte 1981]

Malli on jonkin verran rikkaampi kuin hierarkkinen ja verkkomalli siinä mielessä, että tietorakenteiden määrittelyyn voidaan tuoda yksityiskohtaisempia tietoja. Hajottamisen ja normalisoinnin teoria tarjoaa työkaluja tiedon hienorakenteen analysointiin. [Lacroix ja Pirotte 1981]

Lacroixin ja Pirotten [1981] mukaan relaatiomalli on edistänyt huomattavaa edistystä tietokannan suunnittelun ja manipuloinnin metodologiassa. Siitä huolimatta sen kyky mallintaa monimutkaisia rakenteita kohtuullisen luotettavalla tavalla oli tuohon aikaan suhteellisen rajallinen.

Relaatiomalli on loogisesti yksinkertainen. Sen varastoima data voidaan järjestellä uudelleen muihin formaatteihin, joten se on myös joustava. [Kalay 1985] Relaatiomallilla on paljon muitakin etuja: Dataan pääsee käsiksi nopeasti. Käyttäjän ei tarvitse tietää, miten data on tallennettu, koska se on esitetty loogisesti. Monimutkaisten kyselyiden luonti on helppoa. Data on yleensä tarkempaa. Datan eheyttäminen on helppoa. Myös sovellusohjelmia on helppo luoda ja muokata. [Stephens ja Plew 2001, 2]

Kalayn [1985] mukaan relaatiomalli täyttää parhaiten luvussa 4 luetellut kriteerit tietokantamallille CAD-järjestelmissä: löytyvyys, kompaktius, yksinkertaisuus, joustavuus ja laajennettavuus.

Huonoja puolia relaatiomallissa on se, että eri tauluja pitää liittää datan hakemiseksi. Käyttäjän täytyy myös tietää taulujen väliset suhteet. Ongelmaksi nähdään myös se, että käyttäjän täytyy osata SQL kieltä. [Stephens ja Plew 2001, 2] Anumban [1996] mukaan relaatiomallilla on rajalliset mahdollisuudet datan abstraktiointiin, ja sen luontainen redundanssi tekee normalisointitoimista tarpeellista. Relaatiotietokantajärjestelmiä on myös moitittu niiden rajallisista kyvyistä datan abstrahointiin; uskotaan, että abstraktien tietotyyppien tuki voi parantaa niiden käyttöä CAD-tietojen tallentamiseen ja hallintaan.

Stonebrakerin ja muiden [1983] mukaan relaatiotietokanta ei tuohon aikaan sopinut erityisen hyvin CAD-sovelluksille. Siinä oli seuraavia ongelmia:

- Tuki uusille datatyypeille kuten monikulmioille ja suorakulmioille.
- Tuki tehokkaalle spatiaaliselle haulle.
- Tuki monimutkaisille eheysrajoituksille.
- Tuki suunnitteluhierarkioille ja usealle esityksille.

Lacroixin ja Pirotten [1981] mukaan relaatiomalli edustaa merkittävää edistystä suhteessa hierarkkisiin ja verkkomalleihin. Se ei kuitenkaan salli monimutkaisten objektien useiden näkökohtien suoraa ja tarkkaa esitystä monille CAD-sovelluksille.

Relaatiomalli soveltuu hyvin kuvaamaan dataa "litteällä" rakenteella, joka on tyypillistä monille liiketoiminta- ja hallintosovelluksille, mutta se ei salli luonnostaan sisäkkäisen rakenteen omaavien objektien luonnollista esittämistä. Myöskään alityyppejä, tyyppien yhdisteitä tai muunnelmatietueita ei voida esittää suoraan. Relaatiomallissa on myös hankala tarkastella samaa sovellusaluetta useilla abstraktiotasoilla. [Lacroix ja Pirotte 1981]

Relaatiomalli ei pysty erottamaan objektin ominaisuuksia ja suhteita, koska molemmat mallinnetaan käyttämällä samaa taulurakennetta. [Santana *et al.* 1989] Relaatiomalli levittää dataa CAD-objektista useissa normalisoiduissa suhteissa ja vaatii erittäin kalliin liitosoperaation käyttöä tietojen yhdistämiseen [Liu 2003].

CAD-työkalut käyttävät ohjelmointikielien tallennusrakenteita saadakseen tarvittavaa nopeutta yksittäisten arvojen hakemiseen ja varastointiin [Maier 1986]. Maierin [1986] mukaan nämä operaatiot olivat alun perin liian hitaita relaatiotietokantaan neljästä syystä. Ensimmäinen syy oli, että jokainen haku ja tallennus synnyttää proseduurikutsun sovellusohjelmasta tietokantaan. Tietojenkäsittelytapahtuma, joka pääsee käsiksi kaikkiin monikkoihin relaatiossa, on mitätön lisäkustannus, mutta yksittäiseen monikkoon käsiksi pääseminen on merkittävä taakka. Proseduurikutsu ei voi kilpailla yksinkertaisen siirtymäosoitteen kanssa ohjelmamuistin kenttään käsiksi pääsemisessä.

Toinen syy oli se, että relaatiotietokannassa tarvitaan ainakin yksi osoitteen käännös, jotta päästään avaimen arvosta monikon sijaintiin. Ohjelman muistissa tallenteet voivat osoittaa muihin tallenteisiin suoraan.

Kolmas syy oli se, että monimutkaisten suunnittelurakenteiden normalisointi ja muu koodaus tasoittaa entiteetin ja osakomponentin välisen epäselvyyden tasoa entisestään. Kyselyprosessorin (query processor) kutsuminen vain muutaman monikon liitoksen optimoimiseksi on ylivoimaista.

Neljäntenä syynä olivat lukitseminen ja kirjautuminen. Nämä aiheuttavat paljon yleisrasitetta tapahtumin, jotka päivittävät yksittäisiä lukuja. Kumpaakaan ei ole validoitu optimaaliseksi lähestymistavaksi ympäristössä, jossa on pitkiä tapahtumia ja tietokenttiä, jotka voivat muuttua monta kertaa ennen sitoutumista. [Maier 1986]

Korkean tason datakieli, kuten SQL, asettaa vaatimuksen löytää tehokkain pääsypolku DBMS:ään. Relaatiotietokannassa kyselyiden evaluointi sisältää tyypillisesti valinnan (select), projektion (projection) ja liitoksen (join). Sillä on seuraava kaava: Käytä valintaa relaatioon A, josta saadaan A', käytä valintaa relaatioon B, josta saadaan B'. Liitä A' ja B' muodostamaan relaatio C ja projektoi sarakkeet C:stä. Kyselyissä nopeimman polun määrittäminen riippuu parametrien määrästä. Systeemin täytyy pitää tilastoa, arvioidakseen kuinka monta monikkoa haetaan. Esimerkiksi seuraavat parametrit huomioi-

daan kysymysten optimoinnissa: Relaation mahtavuus, monikkojen keskimääräinen lukumäärä sivulla, sivujen kokonaismäärä segmentissä (joka sisältää relaation), liitosfiltterin tehokkuus, keskimääräinen (avain, monikkoavain) parien määrä indeksin sivulla. [Neumann 2005]

db-engines.com sivuston mukaan relaatiotietokanta on selvästi suosituin tietokantamalli tällä hetkellä. Suosituimpia tietokannan hallintajärjestelmiä, jotka käyttävät relaatiomallia, ovat sivuston mukaan Oracle, MySQL, Microsoft SQL Server ja PostgreSQL.

4.4 Oliomallin tietokanta

Oliomallin tietokannassa entiteettityyppi on esitetty luokkana. Luokka sisältää dataa ja metodeja. Data on olion attribuutteja. Attribuuttina voi olla toinen luokka. Attribuutteja ja metodeja ei tarvitse määritellä samassa luokassa. Jokainen tallenne on olio. [Naik 2013, 2.5] Olio määritellään seuraavasti: Oliolla on olemassaolo riippumatta sen arvosta. Kahdella eri oliolla voi olla sama arvo, jolloin ne ovat yhtä suuria. Kaksi eri oliota voivat olla samoja, jolloin ne ovat identtisiä. [Atkinson *et al.* 1990] Oliomallin tietokanta tukee kaikkia tyypillisiä olio-ohjelmoinnin metodeja kuten luokan perintä, metodien ylikirjoitus, polymorfismi jne. [Naik 2013, 2.5]. Oliomalli mahdollistaa mielivaltaisen monimutkaisten olioiden määrittelyn mielivaltaiseen syvyyteen sisäkkäisiksi olioiksi [Kim *et al.* 1989]. Selkeä määritelmä oliomallin tietokannalle annettiin ensimmäisen kerran vuonna 1990, mutta sitä oli käytetty jo sitä ennen [Atkinson *et al.* 1990].

Ketabchi ja Berzins [1987] määrittelevät oliomallin tietokannan niin, että se koostuu malleista (template) ja esiintymistä. Mallit ovat entiteettityyppien kuvauksia. Ilmentymät ovat entiteettiesiintymien kuvauksia. Jokaisella mallilla on yksilöllinen käyttäjän määrittämä tunniste. Jokaisella esiintymällä on pysyvä ja yksilöllinen järjestelmän määrittämä tunniste. Malli on sen esiintymien luokitus. Sekä mallit että esiintymät ovat olioita. Oliot ovat käteviä koontitietoja, jotka kuvaavat todellisen maailman esineitä. Toisin kuin muiden tietomallien entiteetit, oliomallissa oliot ovat sekä aktiivisen että passiivisen tiedon kasautumia. Olion metodit tarjoavat sovelluskohtaisen käyttöliittymän tietokantaan. [Ketabchi ja Berzins 1987]

Dittrichin [1986] mukaan oliomalli mahdollistaa yhden minimaailman kokonaisuuden (sen monimutkaisuudesta ja rakenteesta riippumatta) edustamisen täsmälleen yhdellä tietokannan oliolla. Koska entiteetit voivat koostua aliyksiköistä, jotka ovat itsenäisiä entiteettejä, oliopohjaisen tietomallin on sallittava myös rekursiivisesti muodostetut oliot.

Oliomallin tietokanta sopii hyvin monimutkaisiin datatyyppeihin [Atkinson *et al.* 1990]. Santanan ja muiden [1989] mukaan se parantaa käsitteellisiä mallinnustekniikoita käyttämällä aikaisemmista mallisukupolvista hankittua kokemusta ja tietoa. Chen ja Leen [2018] mukaan oliomalli sopii CAD:iin. Sen etuna on, että oikean maailman asia ja käsitteellinen malli ovat hyvin samankaltaisia. Siinä voidaan käyttää olio-ohjelmointikieliä,

joita muutenkin käytetään ohjelmissa. Ei siis tarvita mitään erillisiä rajapintoja. [Cattell *et al.* 2000] Ennen oliomallia sen aikaiset tietomallit olivat liian alhaisella käsitteellisellä tasolla teknisille tietokannoille [Eastman 1981]. Spoonerin ja muiden [1986] mukaan sen aikaisilla tietomalleilla ja DBMS:llä ei ollut riittävää joustavuutta ja suorituskykyä CAD-järjestelmille.

Oliomallin tietokannalla voidaan mallintaa käyttäytymistä. Käyttäytymisen mallintamisella tarkoitetaan, että tietokannassa on suoritusmalli, ja operaatiot voidaan liittää olioluokkiin. [Maier 1986]

Samalla olio-ohjelmointikielellä voidaan luoda tietokanta, määritellä oliot, kirjoittaa ohjelma ja kääntää ohjelma. Etuna on myös se, että iso osa ohjelman prosesseista on automatisoitu. Oliomalli on myös yhteensopivampi olio-ohjelmointityökalujen kanssa, ja olioita on teoriassa helpompi hallita. [Stephens ja Plew 2001, 2]

Tsichritzisin ja Nierstraszin [1989] mukaan oliomallin tietokanta sopii dataintensiivisiin sovelluksiin, joissa käytetään suurta määrää rakenteeltaan samankaltaisia olioita. Eli silloin kun olioita on selvästi enemmän kuin niiden luokkia.

Huonoja puolia oliomallin tietokannassa on se, että sitä on vaikea ymmärtää ja käyttää verrattuna esim. relaatiotietokantaan. Siinä ei ole ad hoc -kysely mahdollisuutta. Jos päivitetään tai poistetaan olio, sen periytyvä olio ei automaattisesti poistu tai päivity. [Naik 2013, 2.5] Huono puoli on myös Stephensin ja Plewin [2001, 2] mukaan se, että käyttäjän täytyy osata olio-ohjelmointi.

Oliomallit voidaan luokitella kolmeen luokkaan. Rakenteellisessa oliolähestymistavassa (structural object-orientation) voidaan esittää ja manipuloida korkearakenteisia entiteettejä. Käyttäytymisen oliolähestymistavassa (behavioral object-orientation) voidaan lisätä käyttäjän määrittelemiä tyyppi sidonnaisia operaatioita data entiteetteihin. Täydessä oliolähestymistavassa (full object-orientation) on yhdistetty edellä mainitut lähestymistavat. [Dittrich 1988]

Oliomallissa abstraktit datatyypit mahdollistavat monimutkaisten tietorakenteiden määrittelyn, ja samaan aikaan tarjoavat semanttista informaatiota datasta operaatioilla, jotka on määritelty tyypeille. Ohjelmat voidaan formalisoida operaatioiksi korkean tason abstrakteille data tyypeille, mikä mahdollistaa datan semanttisen lisähallinnan. Olio-ohjelmointi parantaa datan semanttista sisältöä abstraktien datatyyppien välille, määrittelemällä suhteita ominaisuuksien ja operaatioiden periytymisellä. Nämä dataorganisointi menetelmät vaikuttavat sopivilta CAD:iin, koska perusperiaate niissä on luoda ja manipuloida olioita. Samaa asiaa tehdään suunnitteluprosessissa. Tämän lisäksi luvussa 2.2.4 käsitellyt yleistyshierarkiat ovat hyvin samanlaisia kuin oliokeskeisissä ympäristöissä löydetyt olioluokkahierarkiat. [Spooner 1991]

Oliomalli tarjoaa joukon yhtenäisiä toimintoja, jotka tukevat kokoonpanojen jalostusta, vaihtoehtoja ja versioita. Perustavoitteena on tarjota mallinnusmahdollisuudet,

jotka mahdollistavat suunnittelusovelluksissa syntyvien tilanteiden mallintamisen tarkasti ja ilman liiallista tietokantasuunnittelutyötä, sekä tarjota operaatioita suunnittelutietojen käsittelyyn helposti ja tehokkaasti. [Ketabchi ja Berzins 1987]

Monet oliotietokannat tukevat skeeman evoluution hallintaa, joka on erityisen hyödyllinen suunnitteluympäristöissä. Oliotietokannat tukevat yleensä myös versionhallintaa vertailua ja optimointia varten, jota epäperinteiset sovellukset vaativat. Useimmat kaupalliset oliotietokannat ovat passiivisia; ne vain käsittelevät tietoja vastaten käyttäjien tai sovellusohjelmien ulkoisiin pyyntöihin. Tästä syystä nykyinen suuntaus on sisällyttää aktiiviset ominaisuudet oliotietokantoihin. Monet tutkijat pyrkivät laajentamaan perinteisiä passiivisia tietokantoja ominaisuuksilla, jotka tukevat automaattista reagointia tiettyihin olosuhteisiin. Tietokantajärjestelmän aktivoinnin tulos ei ainoastaan helpota aikarajoitteisten sovellusten toteuttamista, vaan myös laajentaa tietokantaa helposti parannetuilla tietojen mallinnusmahdollisuuksilla. [Dong ja Goh 1998]

Maierin [1986] mukaan oliomalliin perustuva tietokannan hallintajärjestelmä on nopeampi relaatiotietomalliin perustuvaan järjestelmään verrattuna seuraavista syistä: Suoritusmallilla yksi sovellusohjelmasta lähetetty viesti voi tehdä useita pääsyjä tietokantaan yhden proseduurikutsun kustannuksella. Suunnitteluoperaatiot eivät varsinaisesti ole yksittäisiä noutoja ja tallennuksia, vaan niihin liittyy tyypillisesti polun seuraaminen toiseen kokonaisuuteen tai uuden objektin täyttäminen luokassa.

Toinen syy on, että oliot voivat viitata alikomponentteihin niiden identiteetin perusteella, eivät niiden tilan perusteella. Globaalit oliotunnisteet voidaan siirtää paikallisiin muistiosoitteisiin, kun olio on päämuistissa. Myös osia globaalista tunnisteista voidaan tallentaa välimuistiin paikallista osoitekartoitusta varten, jolloin päästään lähelle aikaa, joka ohjelmointikielen tietuerakenteissa on.

Kolmas syy on, että monimutkaiset suunnitteluentiteetit voidaan esittää suoremmin, vähemmällä koodauksella. Neljäs syy on se, että pitkät suunnittelutapahtumat edellyttävät, että tietokannan käyttäjät ovat tietoisia toisistaan tai että tietokanta tukee useita versioita objektista. Nämä ehdot yhdessä objektin identiteetin kanssa saavat palautumisen näyttämään lupaavalta optimistisen samanaikaisuudenhallinnan (OCC) osalta. Jos jokainen sovellus saa "henkilökohtaisen" kopion tietokannasta, ei tarvitse kirjata muutoksia tai lukita kohteita ennen käyttöä. [Maier 1986]

Dittrichin [1986] mukaan on myös olemassa kaksi suuntaa, joissa käytetään oliosuuntauksen käsitettä:

- Olio-toteutus: (tietokanta)järjestelmä ohjelmistona rakennetaan abstraktien tietotyyppi-instanssien joukoksi. Eli käytetään tietynlaista modularisointia. Jopa ei-olio-tietokantajärjestelmissä voi olla olio-toteutus.
- Olio-käyttöliittymä/ohjelmointirajapinta: tietokantajärjestelmän käyttöliittymä esitetään käyttäjä- tai sovellusohjelmoijalle olio-ohjelmointiparadigman innoittamana. Vaikka tämä sopii hyvin oliopohjaiseen tietokantajärjestelmään, se voidaan tarjota myös minkä tahansa muun tietokantajärjestelmän päälle.

Datan hakemiseen on myös kehitetty kieli OQL (object query language), joka perustuu SQL:ää. OQL:ssä on joitain lisäominaisuuksia, joita ei ole SQL:ssä, jotka mahdollistavat datan tallentamisen ja dataan pääsemisen olioina. [Cluet 1998] [Stephens ja Plew 2001, 2] Myös ER-mallista (entity-relationship-malli) on olemassa laajennettuja muunnelmia, jotka sisältävät edistyneitä mallinnuskonsepteja oliomallista [Urban *et al*. 2000]. Suosituimpia tietokannan hallintajärjestelmiä, jotka käyttävät oliomallia, ovat db-engines.com sivuston mukaan InterSystems Caché, InterSystems IRIS, Db4o, Actian NoSQL Database ja ObjectStore.

4.5 Olio-relaatiomallin tietokanta

Olio-relaatiomallin tietokanta on yhdistelmä oliomallin tietokantaa ja relaatiotietokantaa [Naik 2013, 2.6]. Tarkoitus on esittää oliomallin ja relaatiomallin parhaat puolet [Stephens ja Plew 2001, 2]. Olio-relaatiomallin pitäisi toteuttaa SQL monimutkaisille datatyypeille [Naik 2013, 2.6]. Monimutkaisia datatyyppejä voi olla esimerkiksi monikot ja joukot [Katragadda 1999]. Oliorelaatiotietokannan hallintajärjestelmät (ORDBMS) on yleensä tarkoitettu tarjoamaan parempaa tukea olio-ohjelmointikielelle [Soutou 2001].

Olio-relaatioteknologia laajentaa perinteisen relaatiotaulukon käsitteen kohdetaulukoiden käsitteellä, jossa oliotaulukon monikot luovat oliotunnisteita, kuten oliotietokantajärjestelmien luokkien esiintymissä. Taulukoiden väliset suhteet muodostetaan sitten käyttämällä olioviittauksia. Polkulausekkeita voidaan myös sisällyttää SQL-kyselyihin tiiviimmän, oliosuuntautuneen lähestymistavan saamiseksi kyselylausekkeisiin. Lisäksi oliorelaatiomallit sisältävät käsitteen käyttäjän määrittämät tyypit tai abstraktit tietotyypit, sen lisäksi, että ne tarjoavat sisäänrakennetun tuen suurille oliolle ja aggregaattityypeille, kuten joukoille, taulukoille ja sisäkkäisille taulukoille. Tämän seurauksena oliotaulukon tai perinteisen relaatiotaulukon attribuutit eivät enää rajoitu atomiarvoihin. Näiden ominaisuuksien olemassaolo vaatii uutta näkemystä siihen, kuinka objektirelaatiokonsepteja voidaan käyttää tehokkaasti käsitteellisten tietokantasuunnitelmien toteutuksessa. [Urban *et al*. 2000]

Olio-relaatiomallin tietokannan etuna on, että käyttäjä voi itse määritellä datatyyppejä oliomallin tietokannan tapaan, mutta tietoja voidaan silti lukea SQL-kielellä. Huono puoli on, että käyttäjän määrittelemät datatyypit ja operaatiot pitäisi säilyttää tietohakemistossa, mikä vaatii sen uudelleensuunnittelua. Käyttäjän määrittelemien datatyyppien indeksointi voi myös aiheuttaa ongelmia. Myös muistin rakenteesta tulee monimutkainen. [Naik 2013, 2.6]

Stephensin ja Plewin [2001, 2] mukaan etuna olio-relaatiomallissa on se, että siinä relaatiotietokannalla on enemmän kolmiulotteisuutta. Huonona puolena käyttäjän täytyy ymmärtää sekä relaatiomalli että oliomalli [Stephens ja Plew 2001, 2].

Urbanin ja muiden [2000] mukaan Oracle 8:n (Oracle on yksi DBMS, joka tukee olio-relaatiomallia docs.oracle.com sivuston mukaan.) käyttö osoittaa, että olio-relaatiojärjestelmät tarjoavat käyttökelpoisen tietokantatekniikan suunnittelusovelluksiin. Kieli, kuten EXPRESS, on helpompi yhdistää oliorelaatiokäsitteisiin perinteisten relaatiotaulukoiden sijaan. Lisäksi oliorelaatioteknologialla toteutetut suunnittelusovellukset eivät vain hyödy oliopohjaisten ominaisuuksien saatavuudesta, vaan myös hyötyvät relaatiotekniikan perinteisestä kyselykielen eduista. Oliorelaatiojärjestelmän, ensisijainen etu oliotietokannan käyttöön verrattuna löytyy SQL:n käytöstä.

Yun ja muiden [2008] mukaan olio-relaatiomalli mahdollistaa monimutkaisen datan mallintamisen ja mukautuu erilaisiin tietolähdemuotoihin ja –standardeihin. Hybriditeknologiana se on perinyt kypsän relaatioperustansa vankat tapahtumanhallintaominaisuudet ja suorituskyvyn hallintaominaisuudet sekä oliosuuntautuneen manipuloinnin joustavuuden. Toisin sanoen se on relaatiotietokantajärjestelmä, jonka avulla käyttäjät voivat määritellä omat funktionsa ja tietotyyppinsä; attribuutteja voidaan yhdistää uuden tietotyypin luomiseksi, esimerkiksi "tekijän nimi, tekijän liitto ja vuosi". Tietokantasuunnittelijat voivat työskennellä tuttujen taulukkorakenteiden ja DDL:n kanssa samalla kun ne omaksuvat uusia olionhallintamahdollisuuksia.

Monimutkaisen datamallinnuksen näkökulmasta katsottuna olio-relaatiomallilla on joitakin ominaisuuksia, joita relaatiotietomalli ei pysty esittämään ja toteuttamaan ytimekkäästi ja tehokkaasti: oliotyypit (käyttäjän määrittämiä tietotyyppejä, joiden avulla tietokantajärjestelmän kehittäjät voivat mallintaa monimutkaisia reaalimaailman kokonaisuuksia), tyypin perintä ja tyypin evoluutio. Tietotyyppejä voidaan helposti luoda, muokata tai kehittää attribuutissa, metodeissa, tyyppimääritelmässä tai metatietotasossa muutosten tallentamiseksi automaattisesti. [Yu *et al.* 2008]

4.6 XML-tietokanta

XML (Extensible Markup Language) tietokannoissa tiedostot on tallennettu XML-formaatissa [Chen ja Lee 2018]. XML kuvaa XML-dokumenteiksi kutsuttujen tieto-objektien luokan ja kuvaa osittain niitä käsittelevien tietokoneohjelmien käyttäytymistä [Bray *et al.* 1998].

XML-dokumentit koostuvat tallennusyksiköistä, joita kutsutaan elementeiksi ja jotka sisältävät joko jäsennettyä tai jäsentämätöntä dataa. Jäsennetty data koostuu merkeistä, joista osa muodostaa merkkidataa ja osa merkintöjä. Merkintä koodaa asiakirjan tallennusasettelun ja loogisen rakenteen kuvauksen. XML tarjoaa mekanismin, jolla asetetaan rajoituksia tallennusasettelulle ja loogiselle rakenteelle. [Bray *et al.* 1998]

XML-dokumentin tyypin määrittely sisältää tai viittaa merkintämäärittelyihin, jotka tarjoavat kieliopin tietylle asiakirjaluokalle. Tämä kielioppi tunnetaan DTD:nä (Document Type Definition). [Bray *et al.* 1998] Yksinkertaisesti sanottuna DTD on tapa, jolla saman toimialan ihmiset voivat sopia viestinsä merkityksestä. [Rezayat 2000] XML

Schema on XML-pohjainen kieli, joka laajentaa DTD:n kykyä lisäämällä nimiavaruuden ja tietotyyppimääritelmiä sekä määrittää monimutkaisempia tietomalleja [Wang 2011].

Wangin [2011] mukaan XML:n käyttö voi yksinkertaistaa tietokannan kehittäjän työtä ja tehostaa tietojen käyttöä ja toimintaa. XML ja sen käsittely ovat myös riittävän joustavia, jotta tehtäviä voi suorittaa eri tavoin. XML:n vahvuus on kuitenkin myös sen heikkous: XML:n muoto ja käsittely ovat erittäin joustavia ja laajennettavissa. Siksi melko usein on olemassa lukemattomia tapoja, joilla XML-tekniikoita voi käyttää saman tavoitteen saavuttamiseksi.

XML-semantiikan (eli DTD:n) määrittäminen toimialalle, jolla on valtavat kokoelmat erittäin säänneltyjä asiakirjoja, on monimutkaista. Lisäksi suurin osa vanhasta datasta ja informaatiosta on lukittu patentoituihin muotoihin. Kaikki tämä tarkoittaa, että jokaisen toimialan on tehtävä alkuinvestointeja omien räätälöityjen DTD-tiedostojensa luomiseen ja että siihen liittyvää XML:ää tulee käyttää rajapintana patentoitujen sovellusten ja dokumenttijärjestelmien välillä. [Rezayat 2000]

Rezayat [2000] on esittänyt KBPD:n (Knowledge-Based Product Development) ja DTD:n, jota kutsutaan CADML:ksi (CAD Markup Language). Niitä käytetään CAD-datan vaihtoon XML:ää hyödyntäen. Shchekin ja Tribushinin [2020] mukaan XML on suositeltu tekniikka tiedonsiirtoon CAD-sovellusten välillä.

4.7 Spatiaalinen tietokanta

Spatiaaliset tietokannat eroavat tavanomaisista tietokannoista kahdella oleellisella tavalla. Ensimmäinen on vaatimus varastoida monimutkaista dataa, kuten pisteitä, janoja ja monikulmioita. Toinen on vaatimus sisältää tarvittavia toimintoja prosessoida monimutkaisia datatyyppejä spatiaalisilla operaatioilla, jotka ovat selvästi monimutkaisempia kuin tavanomaisten tietokantojen operaatiot. [Yeung ja Hall 2007]

Spatiaalista dataa kutsutaan toisinaan myös geospatiaaliseksi dataksi. Spatiaalista dataa voidaan esittää muokata ja analysoida spatiaalisen attribuutin avulla, joka ilmaisee sijainnin. Spatiaalinen attribuutti esitetään yleensä koordinaattiparien muodossa, jotka mahdollistavat spatiaalisen ominaisuuden sijainnin ja muodon mittaamisen sekä graafisen esittämisen. [Yeung ja Hall 2007]

Spatiaalinen data on tallennettu kahteen perusmuotoon, joita kutsutaan vektoriksi ja rasteriksi. Vektorimuodossa oleva spatiaalinen data on maantieteellinen objekti, joka on tunnistettavissa oleva erillinen ilmiö tai piirre esitettynä pisteenä, janana tai monikulmiona. Rasterimuodossa oleva data on yleensä neliöruudukko solun muodossa tai pikselinä, jotka sijaitsevat samankokoisten pikselien ruudukossa. [Yeung ja Hall 2007]

Günther ja Buchmann [1990] näkivät mekaanisen CAD:in potentiaalisena sovellusalueena spatiaaliselle tietokannalle. Mekaaninen CAD on sovellus, jolla on kenties kaikkein vaativimmat geometriset vaatimukset.

4.8 Deduktiivinen tietokanta

Deduktiiviset tietokantajärjestelmät ovat tietokannan hallintajärjestelmiä, joiden kyselykieli ja yleensä tallennusrakenne on suunniteltu loogisen datamallin ympärille. Koska relaatioita pidetään luonnollisesti loogisen predikaatin arvona ja relaatiokielet, kuten SQL, ovat syntaktisia loogisia ilmaisumuotoja, on helppo nähdä deduktiiviset tietokantajärjestelmät relaatiojärjestelmien kehittyneinä muotoina. [Ramakrishnan ja Ullman 1995]

Deduktiiviset järjestelmät ovat deklaratiivisia, toisin sanoen antavat käyttäjälle mahdollisuuden kysyä tai päivittää sanomalla, mitä hän haluaa, sen sijaan, että toiminto suoritetaan. Deduktiivisen tietokantatekniikan ja sen synnyttämän deklaratiivisuuden nähdään tunkeutuvan tietokantajärjestelmien muihin haaroihin, erityisesti oliomaailmaan, jossa on yhä tärkeämpää liittää olio- ja loogisia paradigmoja ns. DOOD (Deklaratiivinen ja oliotietokanta) järjestelmät. Deduktiiviset tietokantajärjestelmät sopivat parhaiten sovelluksiin, joissa on käytettävä suuria määriä tietoa ja tuettava monimutkaisia kyselyitä [Ramakrishnan ja Ullman 1995]

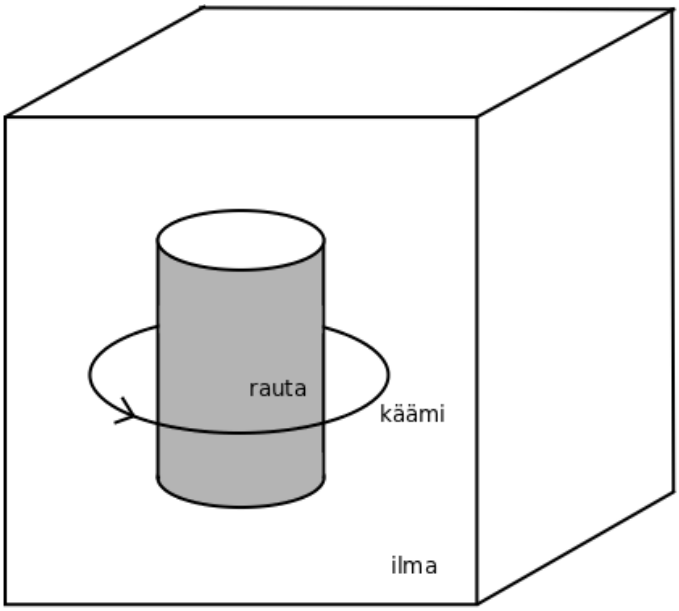
Liun [2003] tekemän analyysin perusteella pysyvät deduktiiviset olio-relaatio- tai oliopohjaiset tietokantajärjestelmät ovat ihanteellisia tukemaan CAD:ia, koska ne tarjoavat suoraa tukea suurten tietomäärien tehokkaalle tallentamiselle ja tehokkaalle pääsylle monimutkaisiin rakenteisiin levymuistissa sekä suorittavat myös päätelmät ja laskelmat.

Liun [2003] mukaan yksi jäljellä oleva kysymys on, voidaanko CAD-järjestelmää rakentaa käyttämällä ei-deduktiivista olio-relaatiotietokantajärjestelmää. Se on todennäköisesti mahdollista, koska olio-relaatiotietokantajärjestelmät tarjoavat hyvän tavan tallentaa ja käyttää CAD-dataa. Kaikki deduktiivisten olio-relaatiotietokantojen päättely- ja laskentamekanismit on kuitenkin toteutettava olio-relaatiotietokantajärjestelmän päälle. Tämän seurauksena CAD-järjestelmä olisi suuri ja monimutkainen, mikä ei ole toivottavaa.

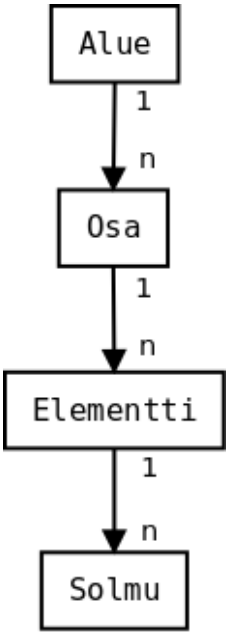
5 Esimerkit tietokantamalleista

5.1 Hierarkkinen tietokanta

Lähteessä Santana *et al.* [1989] on esitetty yksinkertainen esimerkki FE-tehtävästä: Sylinterimäiselle rautakappaleelle on määritelty tietty tilavuusluku. Lohko samoin kuin ympärillä oleva ilma on verkotettu ja hajotettu kiinteiksi elementeiksi. Tilanne on esitetty kuvassa 3. FE-tehtävän esitys hierarkkisessa tietokannassa kuvassa 4.



Kuva 3. FE-tehtävä [Santana *et al.* 1989].

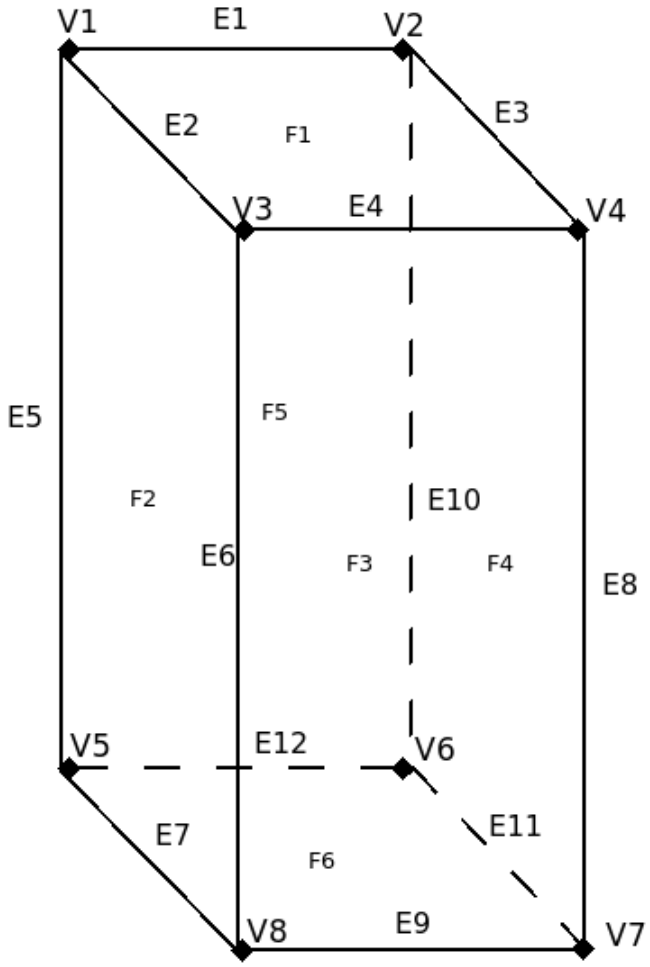


Kuva 4. Esimerkki hierarkkisesta tietokannasta [Santana *et al.* 1989].

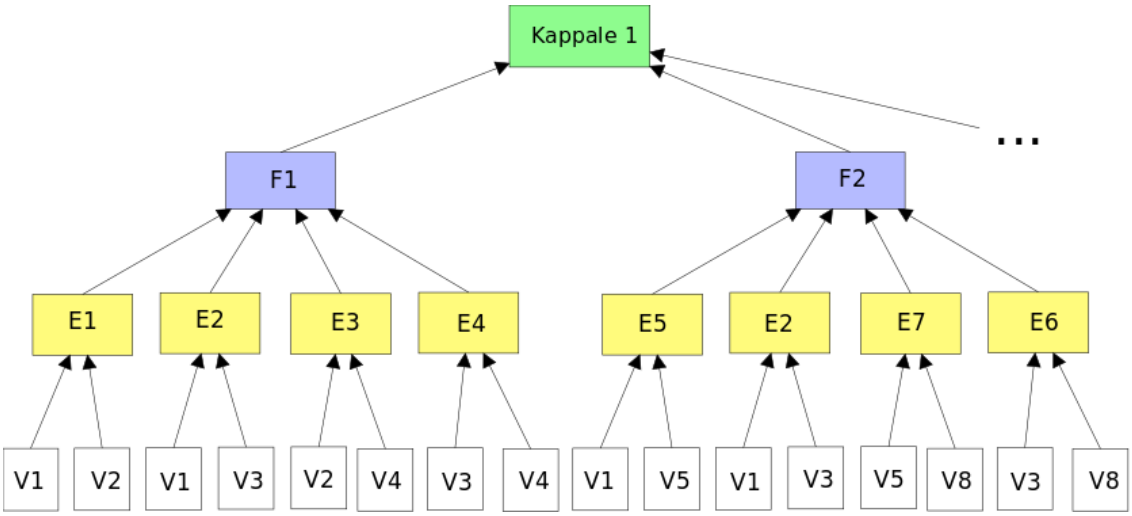
Kuvan 4 esimerkissä alueessa voi olla useita osia, mutta osa kuuluu vain yhteen alueeseen. Jos siis sama osa kuuluu useaan alueeseen, joudutaan se monistamaan, jolloin tietokantaan tulee redundanssia. Vastaavasti osaan voi kuulua useita elementtejä, mutta elementti kuuluu vain yhteen alueeseen. Samoin elementtiin voi kuulua useita solmuja, mutta solmu voi kuulua vain yhteen elementtiin.

Kuvassa 5 on esimerkkikappale, jossa pisteet on merkitty V-kirjaimella, särmät on merkitty E-kirjaimella ja pinnat on merkitty F-kirjaimella. Karteesisen koordinaatiston origo on pisteessä V5. X-akseli kasvaa pisteen V6 suuntaan, Y-akseli kasvaa pisteen V1 suuntaan ja Z-akseli kasvaa pisteen V8 suuntaan.

Lähteessä Bi ja Wang [2020, 2.3.1] on hierarkkinen tietokanta esitetty kuvan 6 tapaan. Esimerkkiobjektina on kuvan 5 kappale.



Kuva 5. Esimerkkikappale.

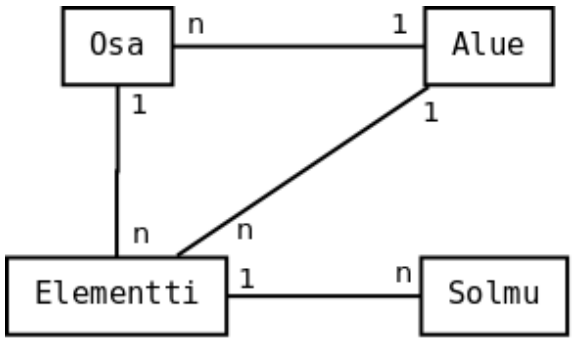


Kuva 6. Esimerkki hierarkkisesta tietokannasta Bi ja Wang [2020, 2.3.1] mukaan. Kaavioon ei ole piirretty näkyviin kaikkea dataa.

Kuvassa 6 kappaleen 1 lapsientiteettejä ovat pinnat. Pintojen lapsientiteettejä ovat särmät. Särmien lapsientiteettejä ovat pisteet. Särmiä ja pisteitä on jouduttu monistamaan, joten tietokantaan syntyy redundanssia.

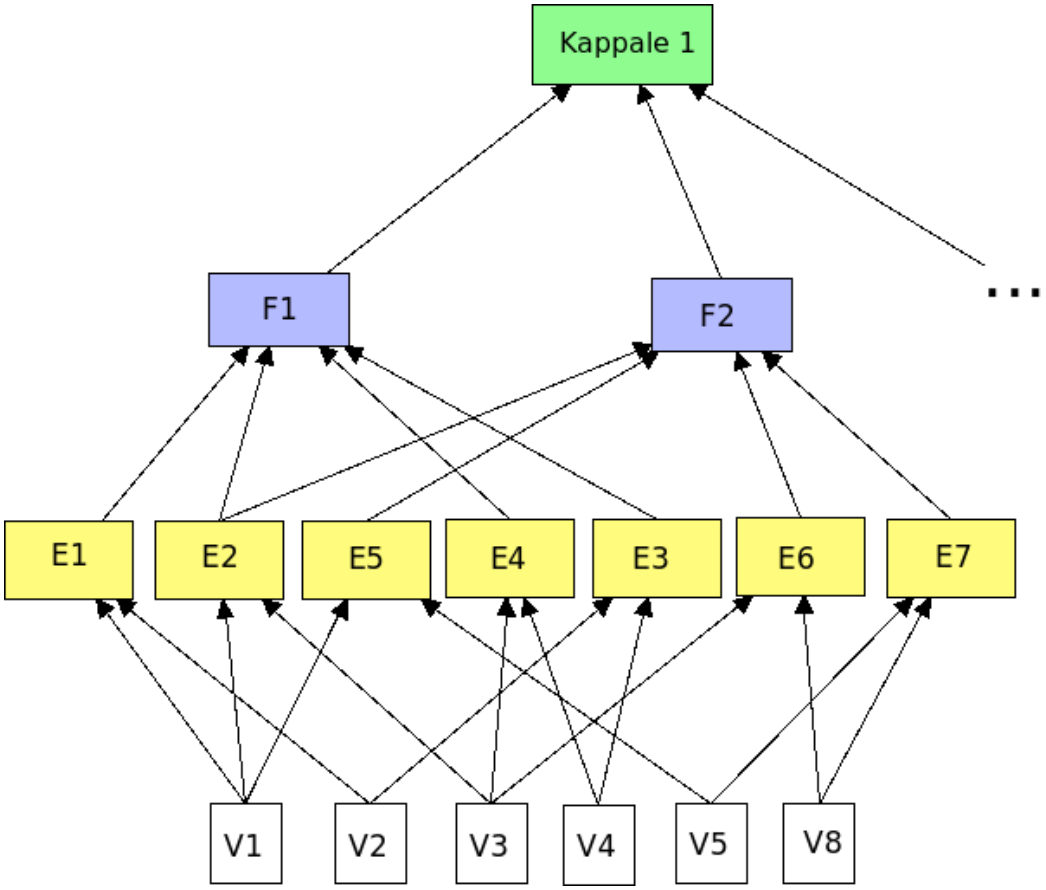
5.2 Verkkomallin tietokanta

Lähteessä Santana *et al.* [1989] esitetään kuvan 3 FE-tehtävä verkkomallin tietokannassa kuvassa 7. Alueesta on linkitys sekä osaan että elementtiin. Se eroaa siis tältä osin hierarkkisesta tietokannasta.



Kuva 7. Esimerkki verkkomallin tietokannasta [Santana *et al.* 1989].

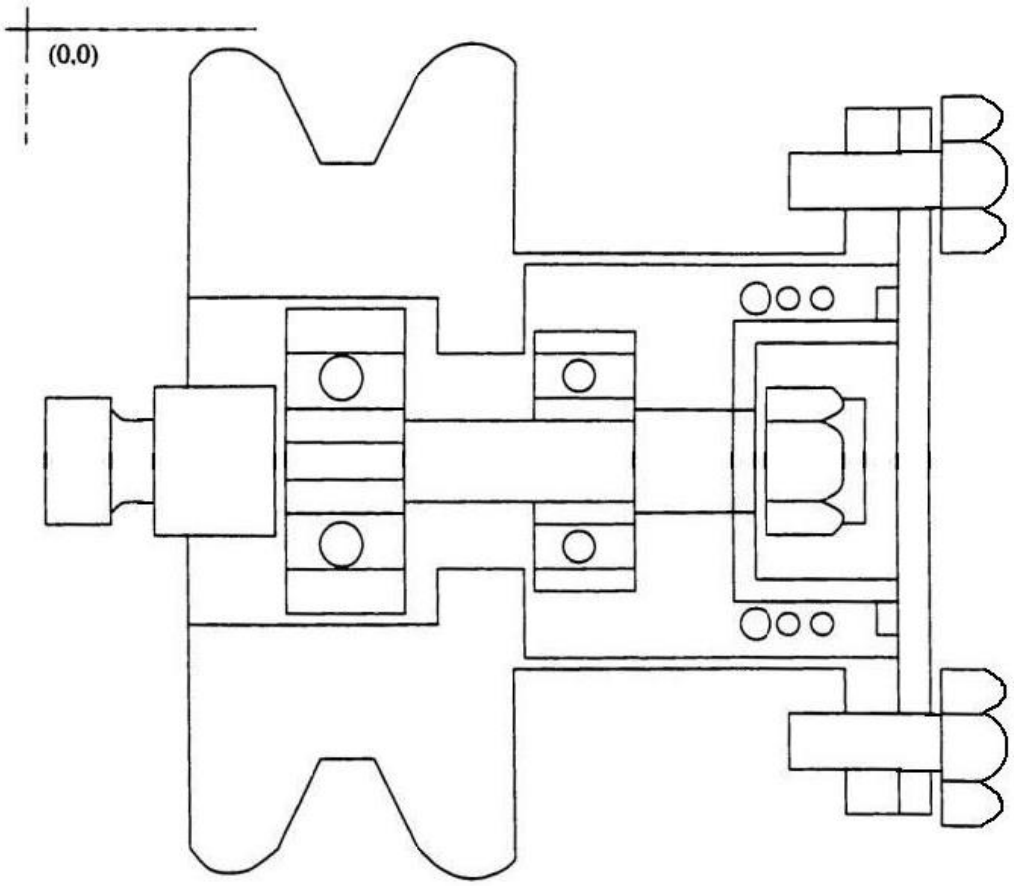
Lähteessä Bi ja Wang [2020, 2.3.1] on verkkomallin tietokanta esitetty kuvan 8 tapaan. Esimerkkiobjektina on kuvan 5 kappale.



Kuva 8. Esimerkki verkkomallin tietokannasta Bi ja Wang [2020, 2.3.1] mukaan. Kaavioon ei ole piirretty näkyviin kaikkea dataa.

Kuvasta 8 nähdään, että verkkomallin tietokantaan ei tarvitse monistaa särmiä eikä pisteitä, niin kuin hierarkkisessa tietokannassa. Samasta pisteestä voidaan tehdä linkitys useampaan särmään. Samoin samasta särmästä voidaan tehdä linkitys useaan pintaan.

5.3 Relaatiotietokanta



Kuva 9. Tuulettimen hihnapyörä [Katragadda 1999]. Alkuperäistä kuvaa on muokattu.

Yksi tapa varastoida kuvan 9 hihnapyörän data relaatiotietokantaan, on käyttää erillisiä relaatioita jokaiselle entiteetille. Taulukoissa 1-6 on esimerkki tästä. Taulukossa 1 on janat, taulukossa 2 on suorakaiteet, taulukossa 3 on ellipsit, taulukossa 4 on murtoviivat, taulukossa 5 on kaaret ja taulukossa 6 on lohkot. [Katragadda 1999]

Jokaisella relaatiolla on attribuutti id, joka tarkoittaa lohkon id-numeroa, johon kyseinen entiteetti kuuluu. Muut attribuutit ovat entiteettien koordinaatteja. Lohko-relaatiossa on attribuuttina, jokaisen lohkon id-numero, alilohkojen id-numero ja alilohkon koordinaatit. [Katragadda 1999]

Tässä esimerkissä relaatiotietokannan heikkous on siinä, että data on tallennettu useisiin eri relaatioihin. Päästäksemme käsiksi kyseisen kokoonpanon kaikkeen dataan, täytyy käyttää useita liitos-operaatioita, joka on kallis operaatio. Tietokannan pitäisi myös tarjota keinoja tunnistamaan kaikki alilohkot kyseisestä kokoonpanosta. Tämä tarkoittaa transitiivista sulkemista lohko-relaatiosta. Relaatiotietokannassa tämä on vaikeaa. Ongel-

mia aiheuttaa myös se, että lohkolla on ainoastaan lähteet alilohkoihin ja niiden sijainteihin. Alilohkojen data täytyy laskea suhteessa niiden peruspisteisiin. Tämä täytyy tehdä tietokannan ulkopuolella. [Katragadda 1999]

Janat				
id	**x1**	**y1**	**x2**	**y2**
1	4	5	4	46
1	3	5	27	5
1	2	47	27	47
1	4	18	25	18
1	4	39	25	39
2	6	21	47	21
2	6	40	48	40
3	4	17	38	17

Taulukko 1. Janat-relaatio.

Suorakaiteet				
id	**x1**	**y1**	**x2**	**y2**
2	7	8	49	50
3	4	6	40	40
4	3	17	39	35
5	83	92	129	116
5	128	88	202	118
5	201	85	235	116
5	42	78	82	128
5	12	88	31	123

Taulukko 2. Suorakaiteet-relaatio.

Ellipsit				
id	**x1**	**y1**	**x2**	**y2**
2	18	21	38	38
3	13	16	29	29
4	245	61	256	69
6	261	62	273	68
6	275	63	281	68
6	246	158	257	168
6	260	158	271	165

Taulukko 3. Ellipsit-relaatio.

Murtoviivat			
id	**viiva nro.**	**x**	**y**
1	1	27	10
1	1	36	10
1	1	35	43
1	1	24	43
6	2	292	75
6	2	292	59
6	2	282	59
6	2	282	69

Taulukko 4. Murtoviivat-relaatio.

Kaaret			
id	**kaari nro.**	**x**	**y**
1	1	24	5
1	1	28	9
1	1	28	15
1	1	24	19
1	2	22	17
1	2	27	25
1	2	26	33
1	2	25	37

Taulukko 5. Kaaret-relaatio.

Lohkot			
id	**alilohko**	**x**	**y**
5	1	235	71
5	3	160	47
5	3	159	113
5	2	79	40
5	2	77	110
6	5	13	16
6	1	248	87
6	3	173	63

Taulukko 6. Lohkot-relaatio.

Lähteessä Stonebraker *et al.* [1983] on esitetty relaatiomalli muuten samalla tavalla kuin lähteessä Katragadda [1999], mutta siinä on lisäksi kerros-attribuutit janaa, suorakaidetta ja murtoviivaa vastaavissa relaatioissa.

Lähteessä Kalay [1985] on esitetty relaatiotietokanta taulukoissa 7-10 esitetyllä tavalla. Taulukossa 7 on pisteet, taulukossa 8 on kehät, taulukossa 9 on pinnat ja taulukossa 10 on särmät. Esimerkkiobjektina on kuvan 5 kappale.

Piste			
pTunnus	**x**	**y**	**z**
1	0	2	0
2	1	2	0
3	0	2	1
4	1	2	1
5	0	0	0
6	1	0	0
7	1	0	1
8	0	0	1

Taulukko 7. Piste-relaatio.

Kehä		
kTunnus	**seuraava**	**pinta**
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6

Taulukko 8. Kehä-relaatio.

Pinta	
pTunnus	**kehä**
1	1
2	2
3	3
4	4
5	5
6	6

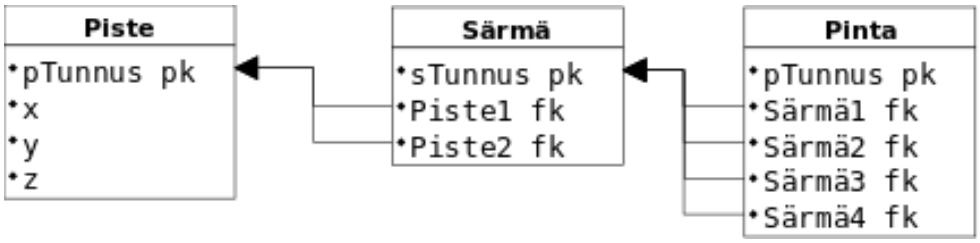
Taulukko 9. Pinta-relaatio.

Särmä								
sTunnus	**kehä1**	**sEdell1**	**sSeur1**	**solmu1**	**kehä2**	**sEdell2**	**sSeur2**	**solmu2**
1	1	2	3	1	5	10	5	2
2	1	4	1	1	2	5	6	3
3	1	1	4	2	4	8	10	4
4	1	3	2	3	3	6	8	4
5	5	1	12	1	2	7	2	5

Taulukko 10. Särmä-relaation viisi ylintä riviä.

Kalay [1985] esityksessä on siis neljä taulua. Objekti muodostetaan niin, että ensin etsitään kehä tai kehät, jotka rajaavat pintaa n kehä-taulusta. Tämän jälkeen etsitään särmät-taulusta kaikki särmät, joista kehä tai kehät muodostuvat, ja muodostetaan näistä särmistä uusi taulu. Seuraavaksi järjestetään särmät niin, että siinä järjestyksessä kehän voi kulkea läpi. Lopuksi etsitään Piste-taulusta särmien päätepisteitä vastaavien pisteiden koordinaatit. [Kalay 1985]

Lähteessä Bi ja Wang [2020, 2.3.1] on relaatiotietokanta niin, että pisteille särmille ja pinnoille on omat taulunsa. Särmä- ja pinta-taulu esitetty taulukoissa 11-12. Piste-taulu on sama kuin taulukossa 7. Esimerkkiobjektina on kuvan 5 kappale. Kuvassa 10 on relaatiotietokannan kaavio Bin ja Wangin [2020, 2.3.1] mukaan. Tässä esimerkissä on se etu Kalayn [1985] esimerkkiin verrattuna, että tauluja on yksi vähemmän.



Kuva 10. Kuvan 5 kappaleen relaatiotietokannan kaavio Bi ja Wang [2020, 2.3.1] mukaan. pk-lyhenne tarkoittaa pääavainta ja fk-lyhenne viiteavainta.

Särmä		
sTunnus	**piste1**	**piste2**
1	1	2
2	1	3
3	2	4
4	3	4
5	1	5
6	3	8
7	5	8
8	4	7
9	7	8
10	2	6
11	6	7
12	6	5

Taulukko 11. Särmä-relaatio.

Pinta				
pTunnus	**särmä1**	**särmä2**	**särmä3**	**särmä4**
1	1	3	4	2
2	2	6	7	5
3	4	8	9	6
4	3	10	11	8
5	1	10	12	5
6	12	7	9	11

Taulukko 12. Pinta-relaatio.

Lähteessä Santana *et al.* [1989] esitetään kuvan 3 FE-tehtävä relaatiomallin tietokannassa taulukoissa 13-14. Tässä esimerkissä on omat taulunsa alueille, sijainnille, elementeille, fysikaalisille ominaisuuksille ja lineaarisille materiaaleille.

Alue			**Sijainti**			**Elementit**	
tilavuus	alue		elementti	tilavuus		elementin nro	solmun nro
v1	alue1		e1	v1		e1	1
v2	alue2		e2	v1		e1	3
v3	alue3		e5	v2		e1	23
…	…		e6	v1		e2	3

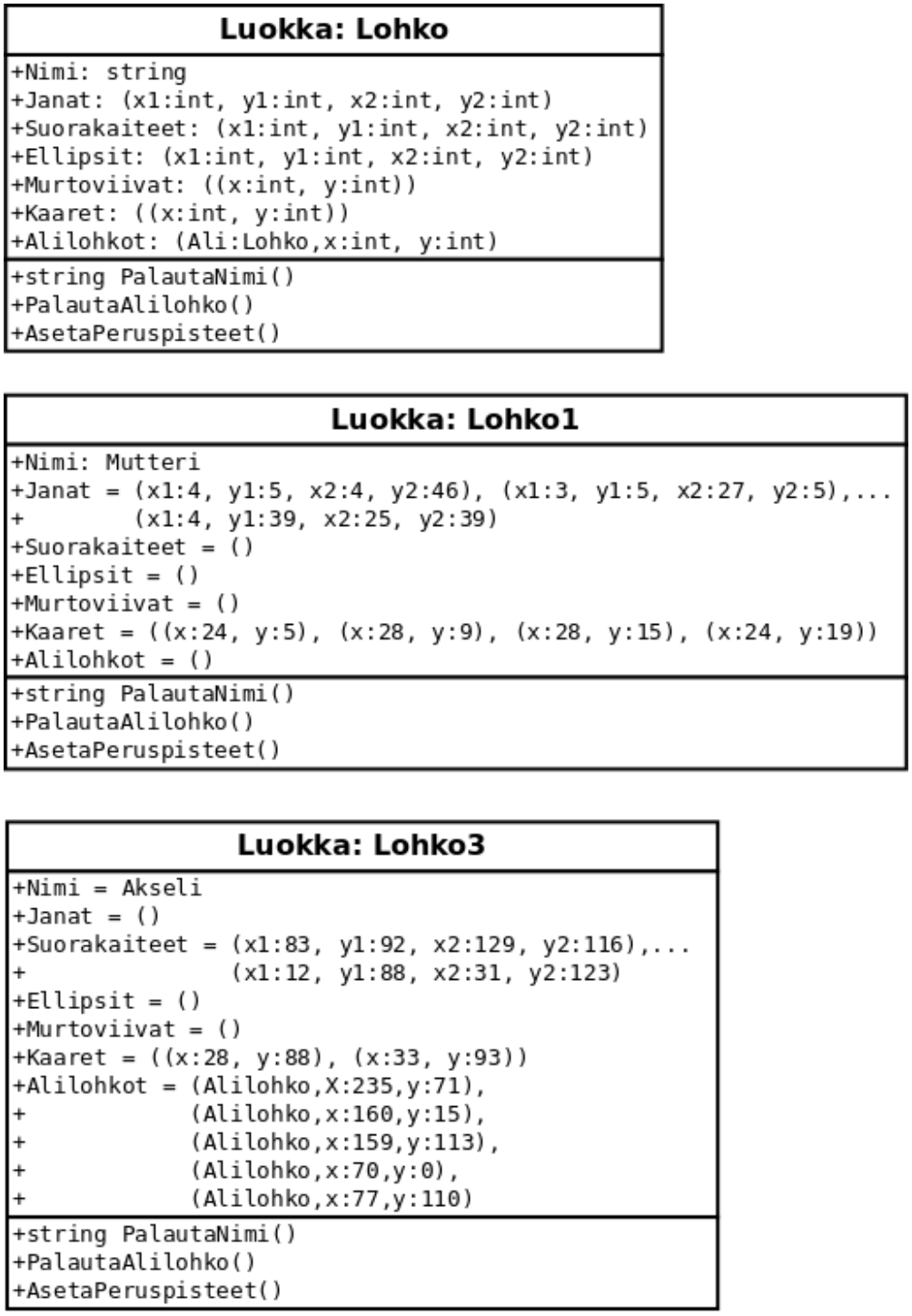
Taulukko 13. Alue-, sijainti- ja elementit-relaatiot Santana *et al.* [1989].

Fysikaaliset ominaisuudet			Lineaarinen materiaali				
alue	ominaisuudet		nimi	σ	μ	isotrooppisuus	J
alue1	ilma		ilma	0	1	…	…
alue2	rauta		rauta	…	…	…	…
…	…		…	…	…	…	…

Taulukko 14. Fysikaaliset ominaisuudet- ja lineaarinen materiaali-relaatiot [Santana *et al.* 1989].

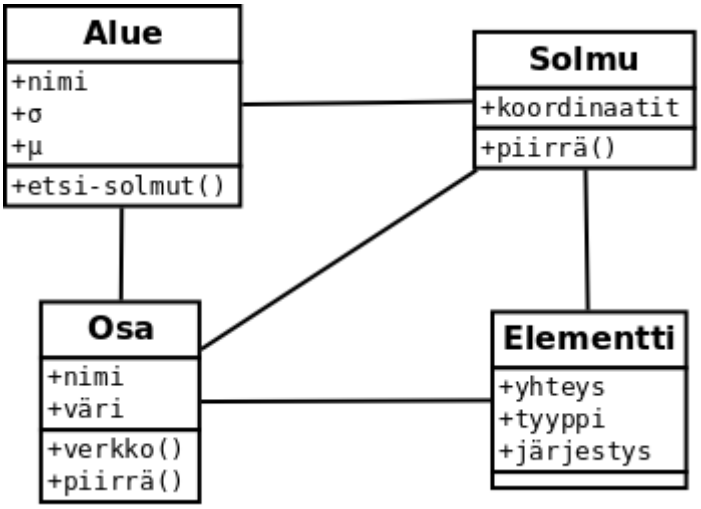
5.4 Oliomallin tietokanta

Katragadda [1999] esittää oliomallin tietokannan niin, että käytetään luokkaa jokaiselle osalle. Oliomallin tietokanta mahdollistaa käyttäjää määrittelemään metodit, joilla voidaan operoida olioita. Käyttäjä voi määritellä metodit, jotka löytävät lohkon kaikki alilohkot. Myös laskeminen voidaan tehdä alilohkojen peruspisteistä eli lohkon kaikki data voidaan laskea näillä metodeilla. Katragaddan [1999] esimerkki UML-kaaviona on esitetty kuvassa 11. Tässä esimerkissä janat, suorakaiteet ja ellipsit on ilmaistu kahdella koordinaattipisteellä. x- ja y-koordinaatit ovat kokonaislukuja (Kyseessä on kaksiulotteinen esimerkki). Murtoviivat ja kaaret on ilmaistu n-määrällä koordinaattipisteitä. Metodeja ovat PalautaNimi(), PalautaAlilohko() ja AsetaPeruspisteet().



Kuva 11. UML-kaavio oliomallin tietokannasta [Katragadda 1999].

Lähteessä Santana *et al.* [1989] esitetään kuvan 3 FE-tehtävä oliomallin tietokannassa kuvan 12 tapaan. Luokat sisältävät attribuutteja ja metodeja. Esimerkiksi alueella on attribuutteina nimi, jännitys (σ) ja kitkakerroin (μ). Alueella on metodi etsi-solmut().



Kuva 12. Esimerkki oliomallin tietokannasta UML-kaaviona [Santana *et al.* 1989].

Oliomalli mahdollistaa Santanan ja muiden [1989] mukaan korkeamman abstraktiotason, mutta näiden abstraktiomekanismien toteuttamiseen liittyvät yleiskustannukset eivät ehkä ole hyväksyttäviä elementtisovelluksissa.

5.5 Olio-relaatiomallin tietokanta

Katragaddan [1999] esimerkki olio-relaatiomallille on esitetty taulukkomuodossa taulukossa 15. Olio-relaatiomalli mahdollistaa kaiken datan tallentamisen yhteen relaatioon. Tämä on toteutettavissa käyttämällä luettelo- (set) ja monikko-rakentajia. Jana-attribuutti koostuu luettelosta monikkoja. Jokainen monikko kuvaa yhtä janaa. Monikossa on attribuutit alku ja loppu, jotka ovat solmuja (vertex). Solmulla taas on yksi koordinaattipiste (X, Y). Suorakaide ja ellipsi attribuutit koostuvat samankaltaisesta rakenteesta, jossa kaksi pistettä on diagonaalisesti vastakkain. Murtoviiva koostuu joukosta solmuja. Alilohko koostuu alilohkon numeroista ja sen koordinaateista. Koska olio-relaatiomallin tietokannassa kaikki data on tallennettu yhteen relaatioon, dataan pääsy ja datan päivittäminen on helpompaa kuin relaatiotietokannassa. [Katragadda 1999]

id	Nimi	Janat				Suorakaiteet				Kaaret		Alilohkot		
		Alku		Loppu		Alku		Loppu						
		solmu		solmu		solmu		solmu		solmu			x	y
		x	y	x	y	x	y	x	y	x	y			
1	Mutteri	4	5	4	46					24	5			
		3	5	27	5					28	9			
		2	47	27	5					28	15			
		4	18	25	18					24	19			
		4	39	25	39					22	17			
2	Laakeri	6	21	47	21	7	8	49	50					
		6	40	48	40									
5	Akseli					83	92	129	116	28	88	1	235	71
						128	88	202	118	33	93	3	160	15
						201	85	239	116	36	93	3	159	113
						42	78	82	128	40	93	2	70	0
						12	88	31	123	29	120	2	77	110
										34	117			
										37	118			
										40	120			

Taulukko 15. Oliorelaatiomalli havainnollistettu taulukoksi [Katragadda 1999].

5.6 XML-tietokanta

Esimerkki XML-tiedostosta:

```
<?XML version=“1.0”?>
<!DOCTYPE product SYSTEM “automotive.dtd”>

<!ELEMENT AUTOMOBILE (KDC+, KPC+) >
<!ELEMENT KDC (#PCDATA) *>
<!ELEMENT KPC (#PCDATA) *>
<!ATTLIST AUTOMOBILE
   make  CDATA    #REQUIRED
   model CDATA    #REQUIRED
   vin   CDATA    #REQUIRED>
```

```
<PRODUCT>
   <AUTOMOBILE make=“M” model=“R” vin=“4ab”>
      <KDC part_num=“x” rev=“4”>pinnan_viimeistely</KDC>
      <KPC part_num=“y” mach=“mil”>Toleranssi</KPC>
   </AUTOMOBILE>
</PRODUCT>
```

Ylläolevassa koodissa tuotteen tyyppi on *AUTOMOBILE.* AUTOMOBILE-elementin pakollisia attribuutteja ovat valmistus (make), malli (model) ja ajoneuvon valmistenumero (vin). Valmistus (make) on *M*, malli (model) on R ja ajoneuvon valmistenumero (vin) on *4ab.* AUTOMOBILE-elementti sisältää vähintään yhden KDC-elementin ja vähintään yhden KPC-elementin. KDC tarkoittaa tärkeimpiä suunnittelu ominaisuuksia (Key Design Characteristics) ja KPC tärkeimpiä prosessin ominaisuuksia (Key Process Characteristics). KDC on esimerkissä pinnan_viimeistely ja KPC on Toleranssi. Attribuutit ovat KDC:ssä osan numero (part_num) ja kierros (rev). KPC:ssä attribuutteja ovat osan numero ja kone (mach). Osan numero on KDC:ssä *x* ja KPC:ssä *y*. KDC:ssä kierros on *4* ja KPC:ssä kone on *mil.* Attribuutit eivät ole pakollisia KPC:ssä eikä KDC:ssä. PRODUCT-elementtiä ei ole määritelty lähteessä. [Rezayat 2000]

Metadata XML Schemalla CAD/CAE -datan integraatioon Yi ja Hua [2011] mukaan:

```
<?xml version} ="1.0"?}
<xs: schema
xmlns:xs= ".http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"version="1.0">
<xs: element name="XML-schema-instance">
<xs: element name="shared parameter" type=
"parameter.class">
<xs: complxType name="parameter.class">
<xs: sequent>
<xs: element name="name" type= "xs: string">
<xs: element name="description" type= "xs: string">
<xs: element name="value" type= "xs: float">
<xs: sequence>
<xs: attributeGroup ref="id.att">
</xs: complexType>
</xs: element>
…
</xs: schema>
```

Tässä esimerkissä elementin nimenä on *shared parameter*, joka on tyypiltään *parameter.class.* Parametre.class-tyyppi pitää sisällään elementit *name*, *description* ja *value*. Attribuuttien Name ja description tyyppejä ovat string eli merkkijono. Attribuutin Value tyyppi on float eli liukuluku. Vastaava CAD:n ja CAE:n välillä vaihdettu XML-dokumentti on seuraava:

```
<XML-schema-instance
xmlns:xsi1/4 ".http://www.w3.org/2001/XMLSchema-
instance"
<XML-schema-instnce
xsi:noNamespaceSchemaLcation="schema_list.xsd">
   <parameter>
      <name>'DH1'</name>
      <description>'kansilaippalevyn paksuus
      </description>\
      <value>26<value>
   </parameter>
…
</XML-schema-instance>
```

Nimi (name) on *DH1*, selitys (description) on *kansilaippalevyn paksuus* ja arvo (value) on *26*. [Yi ja Hua 2011]

6 Pohdintaa

Tutkielmassa käytiin läpi CAD:ia sekä tietokantoja yleisellä tasolla. Tämän jälkeen tarkasteltiin erityispiirteitä ja vaatimuksia, joita CAD-tietokannoilla on, ja mitä tietokantamalleja niissä käytetään. Tutkielmassa on myös esitetty erilaisia tapoja, miten CAD-dataa voidaan tallentaa eri tietomalleja hyödyntäen.

Tietomallien suosiosta paljastui seuraavia asioita. Relaatiomallista oli eniten mainintoja lähteissä, jotka on julkaistu ennen oliomallia. Tämä viittaa siihen, että sitä pidettiin parhaana tuohon aikaan. Hierarkkisessa ja verkkomallin tietomallissa on puutteita verrattuna relaatiomalliin. Oliomallin tietokantaa pidetään useimmissa lähteissä (esimerkiksi Spooner [1991]) parhaana. Vanhemmista tietomalleista eli relaatiomallista, hierarkkisesta ja verkkomallista on mainintoja myös uudemmissa lähteissä esimerkiksi Sarcar et al. [2008] ja Bi ja Wang [2020]. Ne ovat siis säilyttäneet asemansa, vaikka niissä onkin puutteita oliomalliin verrattuna. Esimerkiksi www.cati.com -sivuston mukaan Solidworks PDM käyttää relaatiomallia.

XML-tietokannasta ei luonnollisestikaan ole 80-luvun ja 90-luvun alun lähteissä mainintoja, koska XML esiteltiin vasta vuonna 1998 [Bray *et al.* 1998]. Siitä kuitenkin on mainintoja monissa 2000-luvulla julkaistuissa tutkimuksissa, joissa vaihdetaan CAD-dataa ohjelmien välillä.

Spatiaalisesta tietokannasta on mainintoja rakennusten ja ympäristön suunnitteluun tarkoitettujen ohjelmien yhteydessä (esimerkiksi [Hasan et al. 2016]). Muista tietokantamalleista kuten graafitietokannoista ei löytynyt juurikaan mainintoja. Tämä on mielenkiintoista, koska graafitietokanta on samankaltainen kuin verkkomallin tietokanta, mutta siinä on lisäominaisuuksia [Angles ja Gutierrez 2008].

7 Yhteenveto

CAD-ohjelmilla mallinnettavat objektit voidaan mallintaa rautalankamallina, pintamallina tai solidina mallina. Nykyisin solidien mallintaminen on oleellisin asia CAD:ssa. CAD-objektit ovat tyypillisesti monimutkaisia rakenteeltaan. Niissä on toisiinsa yhteydessä olevia entiteettejä. Rakenne on myös usein hierarkkista. Objektit voivat olla koottuja objekteja, jotka sisältävät toisia objekteja. Tämä aikaansaa tietokannalle erityisiä vaatimuksia.

CAD-tietokanta on informoiva malli todellisen maailman osasta, joka sisältää entiteettejä, joita ihmiset luovat, aistiva, manipuloivat ja ajattelevat. Tietokannan tulee pystyä käsittelemään suuria datamääriä, ja datan monimutkaiset keskinäiset suhteet on esitettävä riittävästi. Tietokannan täytyy pystyä käsittelemään datan dynaamista luonnetta. Tämä johtuu siitä, että käyttäjä voi suunnitteluprosessin edetessä määritellä uusia dataluokkia ja muuttaa niitä. Tietokannan tulee myös muun muassa olla riittävän joustava, tukea eri versioita ja huolehtia datan eheydestä. Dataan pääsyn täytyy myös olla riittävän tehokasta. Rajoitusten käsittelykin on yleensä poikkeuksellista CAD-tietokannoissa. Kaikkia rajoituksia ei voida noudattaa automaattisesti.

CAD-järjestelmissä käytetään useita eri tietokantamalleja. Yleisimmät tietokantamallit ovat hierarkkinen, verkkomallin, relaatio-, oliomallin ja olio-relaatiomallin tietokanta.

Hierarkkisessa tietokannassa entiteettien väliset suhteet on esitetty juurellisena puuna, jossa vanhemmalla voi olla useita lapsia, mutta lapsella voi olla vain yksi vanhempi. Hierarkkista tietokantaa käytetään edelleen, vaikka siinä on ongelmansa. Erityisesti ongelmaksi muodostuu datan redundanssi.

Verkkomallin tietokannassa tallenteiden väliset suhteet hoidetaan linkeillä. Sitä on pidetty parempana kuin hierarkkista tietokantaa, koska monesta moneen suhteet voidaan esittää helpommin. Verkkomallin tietokantaa pidetään kuitenkin monimutkaisena.

Relaatiotietokantaa pidettiin parhaana tietokantamallina CAD-järjestelmiin ennen oliomallin keksimistä. Relaatiotietokannassa on monia etuja hierarkkiseen ja verkkomallin tietokantaan verrattuna. Sitä pidetään loogisena ja yksinkertaisena. On olemassa erilaisia tapoja siitä, miten CAD-data voidaan tallentaa relaatiotietokantaan.

Oliomallin tietokannassa entiteettityyppi on esitetty luokkana. Luokka sisältää dataa (eli attribuutteja) ja metodeja. Attribuuttina voi olla toinen luokka. Jokainen tallenne on olio. Oliomallin tietokanta pidetään parhaana CAD-järjestelmiin. Sen etuna on, että oikean maailman asia ja käsitteellinen malli ovat hyvin samankaltaisia. Siinä voidaan käyttää olio-ohjelmointikieliä, joita muutenkin käytetään ohjelmissa.

Olio-relaatiomallissa on pyritty yhdistämään relaatiomallin ja oliomallin parhaat puolet. Oliorelaatiojärjestelmän, ensisijainen etu oliomallin tietokantaan verrattuna on mahdollisuus käyttää SQL:ää.

Myös muita tietomalleja käytetään CAD-järjestelmissä. XML-tekniikkaa käytetään erityisesti datan siirtämisessä CAD-ohjelmien välillä. Spatiaalista tietokantaa käytetään rakennusten ja ympäristön suunnitteluun tarkoitetuissa CAD-ohjelmissa.

8 Viiteluettelo

Angles, Renzo and Claudio Gutierrez. 2008. Survey of graph database models. *ACM Computing Surveys (CSUR)* 40,1, 1-39.

Anumba, C. J. 1996 Data structures and DBMS for computer-aided design systems. *Advances in Engineering Software* 25,2-3, 123-129.

Atkinson, Malcolm, David DeWitt, David Maier, François Bancilhon, Klaus Dittrich and Stanley Zdonik. 1990. The object-oriented database system manifesto. In: *Deductive and Object-Oriented Databases*, Elsevier B.V., 223–240.

Bi, Zhuming and Xiaoqin Wang. 2020. *Computer Aided Design and Manufacturing. New York, N.Y*: The American Society of Mechanical Engineers.

Bray, Tim, Jean Paoli and C. M. Sperberg-McQueen. 1998. Extensible markup language (XML) 1.0.

Buchmann, Alejandro P. Rolando S. Carrera and M. A. Vazquez-Galindo. 1986. A generalized constraint and exception handler for an object-oriented CAD-DBMS. In: *Proceedings on the 1986 International Workshop on Object-Oriented Database systems,* 38-49.

Buchmann, Alejandro P. and Concepcion Perez de Celis. 1985. An architecture and data model for CAD databases. In: *Proceedings of VLDB 85, Stockholm,* 105-114.

Cattell, Roderic Geoffrey Galton, Douglas K. Barry, Mark Berler, Jeff Eastman, David Jordan, Craig Russell, Olaf Schadow, Torsten Stanienda and Fernando Velez. 2000*. The Object Data Standard: ODMG 3.0.* Morgan Kaufmann.

Chang, Kuang-Hua. 2014. *Product Design Modeling Using CAD/CAE: The Computer Aided Engineering Design Series*. San Diego: Elsevier Science & Technology.

Chen, Jeang-Kuo and Wei-Zhe Lee. 2018. A study of NoSQL database for enterprises. In: *2018 International Symposium on Computer, Consumer and Control (IS3C). IEEE,* 436-440.

Cluet, Sophie. 1998. Designing OQL: Allowing objects to be queried. *Information Systems* 23,5, 279-305.

Codd, Edgar F. 1970. A relational model of data for large shared data banks. *Communications of the ACM* 13(6) 377-387.

Dittrich, Klaus. R. 1988. Advances in object-oriented database systems In: *Proceedings/2nd International Workshop on Object-Oriented Database Systems, Bad Münster am Stein-Ebernburg, DEU,* September 1988. Berlin: Heidelberg.

Dittrich, Klaus. R. 1986. Object-oriented database systems the notions and the issues. In: *Proceedings on the 1986 International Workshop on Object-Oriented database systems,* 2-4.

Dong, Yue and Angela Goh. 1998. An intelligent database for engineering applications. *Artificial Intelligence in Engineering* 12,1-2, 1-14.

Du, H. C. and Ghanta, S. 1987. A framework for efficient IC/VLSI CAD databases, In: *1987 IEEE Third International Conference on Data Engineering*. IEEE, 619–625.

Eastman, Charles M. 1981. Database facilities for engineering design. *Proceedings of the IEEE 69,10,* 1249-1263.

Gesang, Nandhi Roro and Supriyono Supriyono. 2020. Effect Of Motivation, Wages, Labor And Social Security Personnel On The Performance Of Hygiene District Department Of Environmental Nganjuk. *REVITALISASI: Jurnal Ilmu Manajemen* 9.1, 119-126.

Günther, Oliver and A. Buchmann. 1990. Research issues in spatial databases. *ACM Sigmod Record* 19,4, 61-68.

Hammer, Michael and Dennis McLeod. 1981 Database description with SDM: A semantic database model. *ACM Transactions on Database Systems (TODS)* 6,3, 351-386.

Hasan, Ahmad, Ashraf Qadir, Ian Nordeng and Jeremiah Neubert. 2016. Construction inspection through spatial database. *ArXiv Preprint ArXiv:1611.03566*.

Iafrate, J. Ph, O. Santana and J. L. Coulomb. 1988 DB-LIB: a Tool Box of Data Description and Management. *IEEE Transactions on Magnetics* 24,1, 342-345.

Kalay, Yehuda. 1985. A database management approach to CAD/CAM systems integration. In: *22nd ACM/IEEE Design Automation Conference*, *IEEE Press*, 111–16.

Katragadda Shilpesh. 1999. *DrawCAD: Using Deductive Object-relational Databases In CAD.* Regina

Ketabchi, M. A. and V. Berzins. 1987. Modeling and managing CAD databases. *Computer,* 20,02, 93-102.

Kim, Won, Elisa Bertino and Jorge F. Garza. 1989. Composite objects revisited. *ACM SIGMOD Record 18,2,* 337-347.

Lacroix, Michel and Alain Pirotte. 1981. Data structures for CAD object description. In: *18th Design Automation Conference. IEEE,* 653-659.

Liu, Mengchi. 2003. DrawCAD: using deductive object-relational databases in CAD. *Software: Practice and Experience* 33,2, 143-172.

Maier, David. 1986. Why object-oriented databases can succeed where others have failed. In: *Proceedings on the 1986 International Workshop on Object-Oriented Database Systems,* 227.

Naik, Shefali. 2013. *Concepts of Database Management System.* 1st edition. Pearson.

Neumann, Thomas. 2005 CAD data base requirements and architectures. In: *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*: CREST Advanced Course Darmstadt, 8.–19. September 1980, 262-292.

Ou-Yang, C. and P. Y. Liu. 1999. Applying the topological relationships of form features to retrieve part files from a CAD system. *IIE transactions* 31,4, 323-337.

Ramakrishnan, Raghu and Jeffrey D. Ullman. 1995. A survey of deductive database systems. *The Journal of Logic Programming* 23,2, 125-149.

Requicha, A. A. G. and H. Voelcker. B. 1983. Solid modeling: current status and research directions. *IEEE Computer Graphics and Applications* 3,7, 25–37.

Rezayat, Mohsen. 2000. Knowledge-based product development using XML and KCs. *Computer-Aided Design* 32,5-6, 299-309.

Rieu, Dominique and Gia Toan Nguyen. 1986. Semantics of CAD objects for generalized databases. In: *23rd ACM/IEEE Design Automation Conference.* IEEE, 34-40.

Santana, O., B.T. Chia, J.L. Coulomb and J.Ph. Iafrate. 1989. Data bases for CAD applications. *IEEE Transactions on Magnetics* 25,4, 2956-2958.

Sarcar, M. M. M., K. Mallikarjuna Rao and K. Lalit Narayan. 2008. *Computer Aided Design and Manufacturing.* PHI Learning Pvt. Ltd.

Shchekin, A. V. and I. N. Tribushinin. 2020. XML-based network integration of information in CAD systems. *Russian Engineering Research* 40, 1073-1077.

Smith, John Miles and Diane C.P. Smith. 1977. Database abstractions: Aggregation and generalization. *ACM Transactions on Database Systems (TODS)* 2,2, 105-133.

Soukup, Jiri. 1991. Organized C: A unified method of handling data in CAD algorithms and databases. In: *Proceedings of the 27th ACM/IEEE Design Automation Conference,* 425-430.

Soutou, Christian. 2001. Modeling relationships in object-relational databases. *Data & Knowledge Engineering* 36,1, 79-107.

Spooner, David L. 1991. Towards an object-oriented data model for a mechanical CAD database system. In: *On Object-Oriented Database Systems*, Berlin, Heidelberg, 189–205.

Spooner, David L., Michael A. Milicia and Donald B. Faatz. 1986. Modeling mechanical CAD data with data abstraction and object-oriented techniques. In: 1986 *IEEE Second International Conference on Data Engineering. IEEE,* 416-424.

Stephens, Ryan and Ronald Plew. 2001. *Database Design.* Sams Publishing.

Stonebraker, Michael, Brad Rubenstein, Antonin Guttman. 1983. Application of abstract data types and abstract indices to CAD data bases. In: *Pcoc. Annual Meeting Database Week,* 107-113.

Tsichritzis, D. C. and Oscar Nierstrasz. 1989. Directions in object-oriented research. In: *Object-Oriented Concepts, Databases, and Applications,* 523–536.

Urban, Susan D., Michael Tjahjadi and Jami J. Shah. 2000. A case study in mapping conceptual designs to object-relational schemas. *Concurrency: Practice and Experience* 12,9, 863-907.

Wang, J. 2011. *Oracle Database 11g Building Oracle XML DB Applications.* 1st Edition. New York: McGraw-Hill.

Yagiu, Takaaki. 2012 *Modeling Design Objects and Processes.* Springer Science & Business Media.

Yagiu, Takaaki. 1991. Criticism of past and current data models. In: *Modeling Design Objects and Processes.* Springer Science & Business Media, 47-93.

Yeung, Albert KW and G. Brent Hall. 2007. *Spatial database systems: Design, implementation and project management.* Vol. 87. Springer Science & Business Media.

Yi, Zhang and Li Hua. 2011. A method of CAD/CAE data integration based on XML. In: *2011 6th International Conference on Computer Science & Education (ICCSE). IEEE,* 1311-1314.

Yu, Hairong, Davis, M., Wilson, C. S. and Cole, F. T. 2008. Object-relational data modelling for informetric databases. *Journal of Informetrics* 2,3, 240-251.