CS-A1150 Tietokannat

Osa kalvoista perustuu Juha Puustjärven luentokalvoihin aikaisemmalta kurssikerralta sekä kurssin oppikirjaan

4.2.2019

Aloitusluento: tämän luennon jälkeen

- Tiedät, mitä sinun pitää tehdä kurssin suorittamiseksi.
- Tunnet kurssin järjestelyt.
- Tiedät, mitä tarkoittaa tietokanta ja tietokannan hallintajärjestelmä.
- Tiedät tietokannan hallintajärjestelmän osat ja niiden tehtävät.
- Osaat selittää, mitä tarkoittaa relaatiomalli, relaatio, attribuutti ja relaation avain.

CS-A1150 Tietokannat

- Kurssin tavoite: *Kurssin käytyäsi osaat suunnitella yksinkertaisia tietokantoja ja tehdä niihin kyselyitä.*
- Pääpaino relaatiotietokannoissa, mutta myös NoSQL-tietokantoja käsitellään lyhyesti.

Miksi tietokannat ovat tärkeitä?

- Tietokantoja on kaikkialla, esimerkiksi
  - väestörekisteri
  - terveydenhuollon rekisterit
  - Oodi
  - lento- ja junayhtiöiden lipunvarausjärjestelmät
  - nettikaupat
  - yritysten asiakasrekisterit ja laskutustiedot
  - pankkien järjestelmät
  - matkapuhelinoperaattoreiden tiedot asiakkaista ja heidän puhelintensa sijainneista

Käytännön asioita

- Luennot: Kerttu Pollari-Malmi
  - Periodeilla III ja V maanantaisin klo 12-14
  - Periodilla IV tiistaisin klo 12-14.
- Harjoitukset:
  - Joni Arponen, Henri Gröhn, Auli Mustonen, Venla Pesonen (harjoitukset) sekä Tatu Timonen (ratkaisujen arvostelut) ja Vesa Ala-Laurinaho (A+-järjestelmän ylläpito)
  - Kuusi eri ryhmää 2–4 viikon välein. Ke 16–18 ryhmä englanniksi.
  - Ryhmiin ei tarvitse ilmoittautua.
  - Harjoituksissa opiskelijat tekevät harjoitustehtäviä, mutta voivat kysyä neuvoa assarilta.
- Harjoitustyö (pakollinen)
  - Tehdään 2–3 hengen ryhmissä.
  - Palautus kahdessa vaiheessa: suunnitelma (UML-kaavio ym.) 3.4. mennessä, tietokannan toteutus SQL:llä 14.5. mennessä.
  - Tietokantaan ei tehdä käyttöliittymää.

Käytännön asioita, jatkuu

- Osasuoritukset:
  - Tentti (pakollinen)
  - Harjoitustyö (pakollinen): jos pistemäärä on vähintään 30/40, saa siitä yhden arvosanan korotuksen hyväksytyyn tenttisuoritukseen.
  - Harjoitustehtävät (vapaaehtoiset, 0–4 lisäpistettä hyväksyttyyn tenttisuoritukseen)
  - Kurssipalaute (vapaaehtoinen, 16 lisäpistettä harjoitustehtäviin, joiden maksimi ilman palautepisteitä on noin 160.)
- Oppikirja: Ullman, Widom: *A First Course in Database Systems*, 3. painos tai New International Edition. (Aalto-yliopiston kirjastossa on muutama kappale ja e-kirja, jota voi lukea korkeintaan 10 henkilöä samanaikaisesti.)
- Kurssin MyCourses-sivulla julkaistaan luentokalvot, joiden avulla on mahdollista jotenkin valmistautua tenttiin, mutta oppikirjan lukeminen on suositeltavaa.

Käytännön asioita, jatkuu

- Harjoitustehtävät ja niiden palautus:
  - Harjoitustehtäviä on 5 kierrosta ja lisätehtäviä.
  - Harjoitustehtäviä voi tehdä 1–3 hengen ryhmissä.
  - Harjoitustehtävät julkaistaan A+:ssa ja kurssin MyCourses-sivulla (alasivu Harjoitukset).
  - Osa tehtävistä arvostellaan automaattisesti A+:ssa, osa käsin käsin.
  - Kaikki ratkaisut palautetaan A+:aan.
  - Malliratkaisut julkaistaan MyCouresissa tehtäväkierroksen sulkeutumisen jälkeen.

Alustava luentoaikatalu

4.2.	Esittely, johdanto, relaatiomalli
11.2.	Relaatioalgebra
26.2.	SQL: perusteita
5.3.	UML-mallinnus
12.3	UML-kaavio relaatiokaavioksi; Funktionaaliset riippuvuudet
19.3.	Boyce-Codd-normaalimuoto; Moniarvoiset riippuvuudet
26.3.	SQL: alikyselyt, koosteoperaattorit
2.4.	SQL: taulujen määrittely, tietokannan tilan muuttaminen, eheysrajoitukset, näkymät
15.4.	Hakemistot ja niiden käyttö; transaktiot
29.4.	Laukaisimet, SQL-käskyjen liittäminen toisella kielellä kirjoitettuun ohjelmaan
6.5.	*Vierailuluento* Keijo Heljanko: NoSQL-tietokannat
13.5.	SQL-käskyjen liittäminen Python-kieliseen ohjelmaan
20.5.	Kertausta

Arvio kurssin työmäärän jakautumisesta

- Kurssin laajuus on 5 op, joka tarkoittaa noin 133 tuntia työtä. Tuntien on laskettu jakautuvan seuraavasti
  - luennot 13 x 2 h = 26 h (tai vastaava aika itseopiskeluun)
  - vapaaehtoisten harjoitusten tekeminen sekä harjoitustilaisuuksissa että niiden ulkopuolella ja malliratkaisuihin tutustuminen 5 x 6 h = 30 h
  - harjoitustyön tekeminen 45 h / henkilö
  - omatoiminen opiskelu (muu kuin edellä mainittu) 29 h
  - tenttiin osallistuminen 3 h

Viimevuotisesta palautteesta

- Harjoitustyötä pidettiin hyvin hyödyllisenä, samoin automaattisesti tarkastettavia SQL-tehtäviä.

  Tänä vuonna: Harjoitustyön aihe on vaihdettu, mutta harjoitustyö ja SQL-tehtävät hoidetaan viimevuotiseen tapaan.
- Automaattisesti tarkistettaviin relaatioalgebran lausekkeisiin toivottiin joitakin parannuksia.

  Tänä vuonna: Relaatioalgebran tehtävien käyttöliittymässä on pieniä parannuksia, esim. yhden rivin tekstikentän sijaan lausekkeita voi nyt kirjoittaa moniriviselle tekstialueelle ja käytössä on testitietokanta.

Viimevuotisesta palautteesta (jatkuu)

- Kritiikki: Kurssilla on liikaa teoriaa, olisi pitänyt opettaa vain SQL:ää.

  Vastaus: SQL-käskyjen kirjoittamista opetetaan myös ammatti- ja ammattikorkeakouluissa. Yliopistosta valmistuneen erottaa em. kouluista valmistuneista siitä, että hän tuntee myös taustalla olevan teorian ja pystyy sen vuoksi selvästi vaativampiin tehtäviin.

Viimevuotisesta palautteesta (jatkuu)

- Kritiikki: Kurssilla käytetty SQLite ei toteuta kaikki SQL-käskyjä.

  Vastaus: Mikään tarjolla oleva järjestelmä ei toteuta kaikki SQL-standardeissa määriteltyjä ominaisuuksia. Tunnetumpiin järjestelmiin (esim. Oracle, MySQL tai PostgreSQL) verrattuna SQLite on ylivoimainen siinä suhteessa, että jokainen voi helposti ottaa sen käyttöön ja ladata halutessaan omalle koneelleen ilman, että tarvitaan esimerkiksi ylläpidon apua tai ylläpidon luomia tunnuksia.

Viimevuotisesta palautteesta (jatkuu)

- Kritiikki: Harjoitustehtävistä sai liian vähän pisteitä niiden työmäärään verrattuna.
  Vastaus: Harjoitustehtäväpisteillä pystyy kuitenkin korottamaan tenttiarvosanaa kokonaisella numerolla. Harjoitustehtävästä saatavat bonuspisteet ovat aitoja bonuspisteitä eli tentistä voi saada viitosen ilman niitäkin. Jos harjoitustehtäväpisteiden merkitystä lisättäisiin nykyisestä, pitäisi muuta arvostelua muuttaa niin, että kurssista ei enää voisi saada parhaita arvosanoja ilman harjoitustehtäviä.
- Kritiikki: ryhmätyönä tehtävä harjoitustyö oli työläs.
  Vastaus: Ryhmätyölle on varattu kurssin mitoituksessa yli 1/3 kurssiin käytettävästä ajasta, yhteensä 45 tuntia opiskelijaa kohti.

Tietokanta ja tietokannan hallintajärjestelmä

- Tietokanta (database) on kokoelma jollain tavalla yhteen kuuluvaa tietoa.
- Tietokannan hallintajärjestelmä (database management system, dbms)
  - Mahdollistaa uusien tietokantojen määrittelyn tukemalla jotain tiedonmäärittelykieltä (data-definition language).
  - Mahdollistaa tietokantakyselyt ja -päivitykset tukemalla jotain kyselykieltä ja tiedonkäsittelykieltä (query language, data-manipulation language)
  - Varmistaa suurien tietomäärien (teratavuja tai enemmän) säilymisen pitkän ajanjakson niin, että samalla kuitenkin kyselyt ja päivitykset ovat mahdollisia.
  - Varmistaa tiedon säilymisen myös erilaisissa virhe- ja häiriötilanteissa.
  - Varmistaa sen, että samanaikaiset käyttäjät eivät "häiritse" toisiaan.

Miksi tiedostojärjestelmät eivät riitä?

- Tiedostoissa voidaan säilyttää dataa, mutta tiedostojärjestelmät eivät takaa sen säilymistä erilaisissa virhetilanteissa, jos dataa ei erikseen varmuuskopioida.
- Tiedostojärjestelmät eivät tue korkean tason kyselykieliä.
- Tiedostojärjestelmissä tiedon rakenteen määrittely rajoittuu tiedostorakenteeseen.
- Tiedostojärjestelmät eivät tue useamman käyttäjän samanaikaista käyttöä.

Tietokantojen historiasta

- Ensimmäiset kaupalliset tietokannan hallintajärjestelmät 1960-luvulla
  - Rakennettu tiedostojärjestelmien pohjalle.
  - Datamalli ja tiedon varastointi olivat tiivisti yhteydessä toisiinsa, suosituimmat puihin perustuva malli ja verkkorakenteisiin perustuva malli.
  - Esimerkkejä: pankkijärjestelmät, lentoyhtiöitten paikanvarausjärjestelmät, yhtiöitten kirjanpito-, varasto- ja henkilöstöjärjestelmät.
  - Kyselyt hankalia kirjoittaa ja usein hitaita suorittaa
- Relaatiotietokannat:
  - E.F. Codd: *A relational model for large shared data banks*, Communications of the ACM, 1970.
  - Datamalli ja tiedon varastointi erotettu toisistaan.
  - Korkean tason kyselykieli, SQL (esitelty 1976).
  - Edelleen hyvin tärkeä.

Nykyisiä suuntauksia

- Olio-ohjelmoinnin yhdistäminen relaatiotietokantoihin, ORM-työkalut. (Käyttäjä kirjoittaa olio-ohjelmaa, jonka käskyistä osan työkalu muuttaa tietokantakyselyiksi.)
- Pienemmät järjestelmät, jotka pyörivät PC:ssä
- Suuret järjestelmät, joissa relaatiotietokannan ja SQL:n ominaisuuksia on karsittu, jotta saadaan tehokas tietokanta määrättyyn tarkoitukseen.
- Keskusmuistitietokannat.
- Rakenteisten dokumenttien (XML, JSON) käsittely.
- Heterogeenisen tiedon kerääminen ja yhdistely monesta eri lähteestä, tietovarastot (data warehouses).

Tietokannan hallintajärjestelmän osat

- Tietokanta (data and metadata)
  - Sisältää varsinaisen tiedon (data) sekä kuvauksen tiedon rakenteesta (metadata).
- Muistinhallitsija (storage manager)
  - Huolehtii tiedonsiirrosta keskusmuistin ja esim. kovalevyllä olevan tietokannan välillä.
- Kyselynkäsittelijä (query processor)
  - Etsii mahdollisimman tehokkaan tavan kyselyiden ja päivitysten suorittamiseksi.
- Tapahtumankäsittelijä (transaction manager)
  - Kontrolloi samanaikaisten käyttäjien operaatioiden lomittumista (samanaikaisuuden hallinta, concurrency control) ja varmistaa päivitysten atomisuuden.

Muistinhallitsija

- Yleensä tehokkuussyistä tietokannan hallintajärjestelmät kontrolloivat itse ulkoisen muistin (esim. kovalevy) käyttöä sen sijaan, että ne käyttäisivät käyttöjärjestelmän tarjoamia palveluja.
- Muistinhallitsijan osat:
  - Tiedostonhallitsija (file manager)
    - Huolehtii tiedostojen sijoittelusta ulkoisessa muistissa sekä jaksojen (block) siirroista ulkoisen muistin ja keskusmuistin välillä.
  - Puskurinhallitsija (buffer manager)
    - Huolehtii keskusmuistin hallinnasta.
    - Tutkii, mille keskusmuistin sivulle tiedostonhallitsijalta saatu jakso sijoitetaan.
    - Pyytää tarvittaessa tiedostonhallitsijaa kirjoittamaan sivuja takaisin ulkoiseen muistiin.

Kyselynkäsittelijä

- Kääntää korkean tason kielellä (esim. SQL) esitetyn kyselyn suorituskaavioksi (query plan) ja suorittaa kyselyn.
- Usein kysely voidaan suorittaa monessa eri järjestetyksessä. Kyselynkäsittelijän tehtävänä on etsiä mahdollisimman tehokas suoritusjärjestys.
- Kyselynkäsittelijän tehtävänä myös optimoida yksittäisiä kyselyyn sisältyviä operaatioita esimerkiksi käyttämällä apuna hakemistoja (engl. index) silloin, kun se on mahdollista.

Tapahtumankäsittelijä

- Käyttäjä voi määritellä yhden tai useamman kyselyn tai päivityksen tapahtumaksi, transaktioksi (transaction).
- Tapahtumankäsittelijä huolehtii siitä, että
  - Transaktiolla on sama vaikutus tietokantaan riippumatta muista samanaikaisista tapahtumista. (Esim. varaukset menevät oikein, vaikka kaksi käyttäjää yrittäisi samaan aikaan varata paikkoja samalle lennolle.)
  - Kaikki transaktion päivitykset suoritetaan (esim. pankin tilisiirrossa sekä otto että pano eikä vain toista) tai niistä ei suoriteta mitään.
  - Tietokantaan tehdyt toimenpiteet eivät riko sille ennalta määrättyjä eheysehtoja (esim. otto ei vie tilin saldoa negatiiviseksi, jos sitä ei ole sallittu).
  - Transaktion päivitykset säilyvät tietokannassa (myös erilaisissa virhetilanteissa).
- Transaktioita käsitellään tarkemmin omalla luennollaan.

Relaatiomalli

- Käytössä suurimmassa osassa kaupallisista tietokannan hallintajärjestelmistä.
- Hyvin yksinkertainen
- Yksinkertainen korkean tason kyselykieli, joka kuitenkin tarjoaa hyvin monipuoliset mahdollisuudet.
- Mahdollista implementoida tehokkaasti.

Relaatiomalli, jatkoa

- Tietokanta koostuu kaksiulotteisista tauluista (table), joita kutsutaan *relaatioiksi* (relation).
- Jokaisella relaatiolla on joukko nimettyjä *attribuutteja* (attribute).
- Kullakin taulun rivillä eli monikolla (tuple) on arvot eri attribuuteille.
- Attribuuttien arvojen tulee olla atomisia (esim. yksittäinen lukuarvo tai merkkijono, ei esimerkiksi joukko tai monikko). Arvoilla on tyyppi.

Relation `Customers`

custNo	*name*	*born*	*bonus*	*address*	*email*
112233	Teemu Teekkari	1995	55	Servinkuja 3	tteekkari@gmail.com
554422	Riina Raksalainen	1993	43	Otaranta 8	riinar@yahoo.com
37856	Antti Virta	1970	12	Aaltokatu 4	antti@hotmail.com

Huomautuksia

- Relaatiot ovat monikoiden joukkoja, ei listoja. Tämä tarkoitttaa sitä, että monikoiden järjestyksellä relaation sisällä ei ole merkitystä ja että relaatio ei voi sisältää kahta täsmälleen samaa monikkoa (kaikkien attribuuttien arvot samat yhtä aikaa).
- Myös attribuutit voidaan esittää mielivaltaisessa järjestyksessä, kuitenkin niin, että säilyy tieto siitä, mikä arvo kuuluu millekin attribuutille.
- *Relaatiokaavio* (relation schema) määrittää, mitä attribuutteja relaatioon kuuluu ja mikä on attribuuttien tyyppi.
  - Esimerkiksi edellisen kalvon relaatiokaavio voidaan kirjoittaa joko `Customers(custNo, name, born, bonus, address, email)` tai tyyppien kanssa `Customers(custNo:string, name:string, born:integer, bonus:integer address:string, email:string)`
- *Tietokantakaavio* (database schema) sisältää kaikkien tietokantaan kuuluvien relaatioiden relaatiokaaviot.
- Relaation *instanssi* tarkoittaa relaation sisältämiä monikoita tietyllä hetkellä.

Erikoisarvo NULL

- Erikoisarvolla `NULL` voidaan kuvata sitä, että jonkin attribuutin arvoa ei jollain monikolla tiedetä tai ole määritelty.
- *Esimerkki:* lisätään edellä esitettyyn `Customers`-relaatioon uusi asiakas, jonka syntymävuosi ei ole lisääjän tiedossa:

Relation `Customers`

custNo	*name*	*born*	*bonus*	*address*	*email*
112233	Teemu Teekkari	1995	55	Servinkuja 3	tteekkari@gmail.com
554422	Riina Raksalainen	1993	43	Otaranta 8	riinar@yahoo.com
37856	Antti Virta	1970	12	Aaltokatu 4	antti@hotmail.com
41678	Anne Asiakas	NULL	0	Narutie 7	anne@iki.fi

NULL-arvojen vaikutuksia

- Halutaan hakea edellisen kalvon relaatiosta kaikki ne asiakkaat, jotka ovat syntyneet vuoden 1990 jälkeen eli joille ehto `born` $>$ 1990 on tosi.
- Anne Asiakas ei tule mukaan, koska hänelle ei ole määritelty syntymävuotta.
- Entä, jos ehto onkin `born` $>$ 1990 `OR` `born` $<=$ 1990?
- Vaikka ehdon voisi kuvitella kattavan kaikki asiakkaat, niin todellisuudessa Anne Asiakas ei tule hakutulokseen mukaan, koska sen `born`-attribuutilla on arvo `NULL`.

Avain

- Joukko relaation attribuutteja muodostaa relaation *avaimen*, jos millään kahdella relaation monikolla ei saa olla samoja arvoja kaikilla avaimen attribuuteilla.
- Relaatiossa `Customers` attribuutti `custNo` sopii relaation avaimeksi, koska jokaisella asiakkaalla on yksikäsitteinen asiakasnumero.
- Joissakin tapauksissa mikään attribuutti ei riitä yksistään relaation avaimeksi, vaan useampi attribuutti yhdessä muodostaa avaimen (esimerkki kohta).
- Avainattribuuttien arvojen pitää olla yksikäsitteisiä kaikilla mahdollisilla relaation instansseilla. Esimerkiksi attribuutti `born` ei ole relaation `Customers` avain, vaikka kalvon instanssissa jokaisella monikolla on eri arvo `born`-attribuutilla. On täysin sallittua lisätä uusi asiakas, jolla on sama syntymävuosi jonkun relaatiossa jo olevan asiakkaan kanssa.
- Relaatiokaaviossa avaimeen kuuluvat attribuutit merkitään usein alleviivaamalla, esimerkiksi `Customers(custNo, name, born, bonus, address, email)`

Toinen esimerkki relaatioista

- Oletetaan, että halutaan tallentaa tiedot yliopiston opiskelijoista, kursseista ja siitä, kuka opiskelija on suorittanut minkin kurssin.
- Kursseista tarvitaan vain perustiedot (ei esim. lukujärjestys- tai opettajatietoja).
- Oletetaan, että opiskelija voi suorittaa saman kurssin vain yhteen kertaan.
- Mitä relaatioita määrittelisit? Mitä avaimia näille relaatioille tarvitaan?

Opiskelijaesimerkki jatkuu

- Tarvitaan yksi relaatio opiskelijoiden ja toinen kurssien tietoja varten.
- Koska attribuuttien on oltava atomisia, ei tietoa suoritetuista kursseista voida tallentaa opiskelijan tietoja kuvaavaan monikkoon eikä kurssin suorittaneita opiskelijoita kurssin monikkoon.
- Tarvitaan siis kolmas relaatio, johon tallennetaan tieto siitä, kuka opiskelija on suorittanut minkin kurssin.
- Esimerkki järkevistä relaatiokaavioista:
  `Students(ID, name, program, year)`
  `Courses(code, name, credits)`
  `Grades(studentID, courseCode, date, grade)`
- Attribuutit `studentID` ja `courseCode` muodostavat yhdessä relaation `Grades` avaimen.