CS-A1150 Tietokannat

20.5.2019

Kertausluento

- Tällä luennolla kerrataan lyhyesti tenttivaatimuksissa esitettyjä asioita ja samalla tarkastellaan sitä, mitä niistä voidaan kysyä tentissä.
- Huomaa kuitenkin, että kaikki kurssialueen asiat asiat kuuluvat tenttivaatimuksiin, jos ei ole erikseen mainittu, että ne eivät kuulu. Kysymyksiä voi olla myös erilaisia kuin mitä tässä on esitetty esimerkkeinä.
- Vierailuluennon (NoSQL-tietokannat) asiat eivät kuulu tenttivaatimuksiin.
- Tämän luennon kalvot eivät sovellu niillä mainittujen asioiden itseopiskeluun, koska asiat on esitetty niillä liian lyhyesti. Asiat on selitetty tarkemmin vastaavien luentojen kalvoissa.
- Voit luennon aikana lähettää kysymyksiä ja kommentteja myös sivulla `http://presemo.aalto.fi/tietokannat2019`

Tärkeä käytännön asia

- Ensimmäiseen tenttiin ei tarvitse ilmoittautua erikseen, jos on ilmoittautunut kurssille. Rästitentteihin pitää ilmoittautua viimeistään viikko ennen tenttiä WebOodissa.
- Toukokuun tentin jälkeen on vielä kolme rästitenttiä, joissa keväällä 2019 tehty ja hyväksytty harjoitustyö sekä harjoitustehtävistä saatavat lisäpisteet ovat voimassa. Todennäköiset päivät ovat 5.9.2019, 25.10.2019 ja 21.2.2020. (**Tarkista päivät, kun tiedot on julkaistu WebOodissa, sillä näihin voi vielä tulla muutoksia.**)

Mitä tämän kurssin jälkeen?

- CS-C3170 Web Software Development
  - Ei varsinaisesti tietokantakurssi, mutta kurssilla tehdään harjoitustyönä web-ohjelma, joka käyttää tietokantaa. Harjoitustyössä käytetään Django-kehitysympäristöä.
- CS-E4640 Big Data Platforms
  - Kurssi sisältää lisätietoa NoSQL-tietokannoista
- Mahdollisesti muitakin kursseja, kun alan professoritilanne CS-laitoksella vakiintuu.

Esimerkkitietokanta

- Tämän luennon esimerkit käsittelevät aikaisempien luentojen esimerkkitietokantaa, joka koostuu seuraavista relaatioista

  `Customers(custNo, name, born, bonus, address, email)`
  `Products(number, prodName, description, price, manufID)`
  `Manufacturers(ID, manufName, phone)`
  `Orders(orderNo, deliver, status, custNo)`
  `BelongsTo(orderNo, productNo, count)`

Esimerkkitenttikysymyksiä tietokannoista ja tietokannan hallintajärjestelmästä yleisesti

- Mitä etuja tietokannan hallintajärjestelmä tarjoaa siihen verrattuna, että dataa säilytettäisiin tavallisissa tekstitiedostoissa.
- Tietokannan hallintajärjestelmän osat ja niiden tehtävät.

UML-mallinnus

- UML-kaavio on graafinen tapa tietokannan mallintamiseen.
- Keskeisiä käsitteitä: luokka, assosiaatio, assiosiaatioluokka, aliluokka, kompositio ja aggregaatio.
- Huomaa erot silloin, kun UML-kaaviolla mallinnetaan tietokantaa vs. UML-kaaviolla mallinnetaan olio-ohjelmaa.

Mahdollisia tenttikysymyksiä UML-kaavioista

- Laadi UML-kaavio annettua kuvausta vastaavaan tilanteeseen.
- Muunna laatimasi tai tehtävässä annettu UML-kaavio relaatiokaavioiksi.
- Vastaa annetusta UML-kaaviosta esitettyihin kysymyksiin. (Esim. onko tämän kaavion perusteella mahdollista, että ...?)

Tyypillisiä virheitä UML-kaavioita koskevissa vastauksissa

- UML-kaaviossa luokalla on ylimääräisiä attribuutteja, esimerkiksi assosiaation kautta tulevia tietoja (toisin kuin olio-ohjelmia mallinnettaessa, assosiaation kautta tulevia tietoja ei merkitä luokan omiksi attribuuteiksi).
- Assosiaatio on useamman kuin kahden luokan välinen (UML-kaavioissa assosiaatio on aina kahden luokan välinen).
- Avaimia ei ole merkitty tai ne on merkitty väärin.
- Ei ole merkitty oikein, että avaimeen tarvitaan myös toisen luokan avainattribuutti.
- Assosiaatioluokalla on avainattribuutteja (tällöin siitä pitäisi tehdä tavallinen luokka).
- Assosiaation valitsevuutta ei ole merkitty (tällöin se on tasan yksi) tai se on merkitty väärin.
- Jotain asiaa on kuvattu attribuuttina, vaikka sen arvo on joukko tai lista.

Tyypillisiä virheitä UML-kaavioita koskevissa vastauksissa, jatkuu

- Kun luokkia ja assosiaatioita on muutettu relaatioiksi, niin attribuutteja on joko liikaa tai liian vähän (osa avainattribuuteista puuttuu relaatiosta tai sitten assosiaatiosta tehdyssä relaatiossa on mukana ylimääräisiä attribuutteja).
- Osaa avaimeen kuuluvista attribuuteista ei ole alleviivattu tai on alleviivattu liikaa attribuutteja.
- Monesta moneen -assosiaatiosta ei ole tehty omaa relaatiota, kun UML-kaavio on muunnettu relaatioiksi.

Avaimella on väliä!

- Mitä eroa on relaatioilla

  `Products(number, prodName, description, price, manufID)`

  ja

  `Products(number, prodName, description, price, manufID)`

Avaimella on väliä!

- Mitä eroa on relaatioilla

  `Products(`<u>`number`</u>`, prodName, description, price, manufID)`

  ja

  `Products(`<u>`number`</u>`, prodName, description, price,` <u>`manufID`</u>`)`
- Jälkimmäisessä sama tuote (tuotenumero) voi esiintyä relaatiossa `Products` monta kertaa, jos sillä on joka monikossa eri valmistaja.

Funktionaaliset riippuvuudet ja tietokannan normalisointi

- *Ongelma:* Mitä relaatioita tietokantaan pitäisi määritellä ja mitä attribuutteja näillä pitäisi olla?
- Samat tiedot voidaan esittää useilla eri tietokantakaavioilla. Jotkin niistä ovat parempia kuin toiset.
- Keskeisiä huonojen relaatiokaavioiden aiheuttamia ongelmia:
  - Tiedon toisteisuus (redundancy)
  - Päivitysanomaliat (update anomalies)
  - Poistoanomaliat (deletion anomalies)
- Huonot relaatiokaaviot voidaan muuttaa parempaan muotoon normalisoimalla. Normalisoinnissa tarvitaan tietoa relaatioiden attribuuttien funktionaalisista riippuvuuksista.

Boyce-Codd-normaalimuoto

- Jos relaatio on Boyce-Codd-normaalimuodossa (Boyce-Codd normal form, BCNF), siinä ei ole lainkaan funktionaalisista riippuvuuksista johtuvaa toisteista tietoa.
- Relaatio R on BCNF:ssä (Boyce-Codd-normaalimuodossa) jos ja vain jos kaikille R:ssä voimassa oleville epätriviaaleille funktionaalisille riippuvuuksille $A_1\ A_2\ \ldots A_n\ \rightarrow B_1\ B_2\ \ldots\ B_m$ pätee se, että $\{A_1, A_2, \ldots, A_n\}$ on relaation R yliavain.
- Esimerkki: relaatio

  `Products1(number, prodName, description, price, manufID, manufName, phone)`

  ei ole BCNF:ssä, koska relaatiolle pätee riippuvuus

  $$\texttt{manufID} \rightarrow \texttt{manufName phone}$$

  mutta riippuvuuden vasen puoli ei ole relaation yliavain, vaan relaation avain on $\{\texttt{number}\}$

Relaation osittaminen Boyce-Codd-normaalimuotoon

- Laske annettujen epätriviaalien riippuvuuksien vasempien puolten sulkeumat.
- Jos yksikin sulkeumista ei sisällä relaation kaikkia attribuutteja (eli vasen puoli ei ole relaation yliavain), relaatio ei ole BCNF:ssä, ja se pitää osittaa.
- Valitse **yksi** niistä riippuvuuksista, jossa vasen puoli ei ole relaation yliavain ja ota tämän riippuvuuden vasemman puolen sulkeuma.
- Jaa ositettava relaatio kahdeksi uudeksi relaatioksi:
  - Ensimmäiseen tulee em. sulkeuma.
  - Toiseen tulee edellä valitun riippuvuuden vasen puoli ja lisäksi ne ositettavan relaation attribuutit, jotka eivät kuulu vasemman puolen sulkeumaan.
- Tutki, mitkä riippuvuudet ovat voimassa uusissa relaatioissa.
- Laske uusien relaatioiden riippuvuuksien vasempien puolien sulkeumat. Jos jokin sulkeuma ei sisällä uuden relaation kaikkia attribuutteja, uusi relaatio ei ole BCNF:ssä ja se pitää osittaa samalla tavalla.

Mahdollisia tenttikysymyksiä funktionaalisesta riippuvuudesta ja BCNF:stä

- Tämä on erittäin tärkeä asia, ja on hyvin todennäköistä, että asiasta kysytään jokaisessa tentissä.
- Todennäköisin tehtävätyyppi: on annettu relaatio ja siinä esiintyvät funktionaaliset riippuvuudet. Kysytään, onko relaatio BCNF:ssä (perustelut) ja pyydetään osittamaan se tarvittaessa BCNF:ään.
- Myös moniarvoiset riippuvuudet ovat mahdollisia tenttitehtävien aiheita. Tästä aihepiiristä mahdollinen tehtävä on sellainen, jossa vaaditaan moniarvoisen riippuvuuden käsitteen ymmärtämistä. Tehtävissä ei kysytä mitään 4. normaalimuodosta.

BCNF-tehtävä toukokuun 2018 tentistä

- Tarkastellaan relaatiota $R(A, B, C, D, E)$, jossa on voimassa riippuvuudet $A \rightarrow B$, $C \rightarrow B$ ja $C\ D \rightarrow E$.
  - Perustele, miksi relaatio ei ole BCNF:ssä.
  - Osita relaatio BCNF:ään käyttämällä kurssilla (ja oppikirjassa) esitettyä algoritmia. Perustele lyhyesti jokainen muodostamasi uusi relaatio. Jatka osittamista niin pitkälle, että jäljellä on vain BCNF:ssä olevia relaatioita. Perustele, miksi lopulliset relaatiosi ovat BCNF:ssä.
  - Ratkaisu esitetään luennolla.

Tyypillisiä virheitä tenttivastauksissa BCNF-tehtävissä

- Kun on kysytty, onko relaatio BCNF:ssä, ei ole laskettu funktionaalisten riippuvuuksien vasempien puolien sulkeumia, vaan on annettu jokin erikoinen perustelu, joka on joko selvästi väärä tai sitten liian ylimalkainen.
- Relaatiota ei ole ositettu annetulla algoritmilla, vaan jollain itse keksityllä tavalla (esimerkiksi yhteen uuteen relaatioon on aina otettu kaikki yhdessä riippuvuudessa esiintyvät attribuutit), joka voi johtaa informaation katoamiseen
- On annettu aivan oikeat uudet relaatiot, mutta ei ole kunnolla perusteltu sitä, miten niihin on päädytty tai miksi uudet relaatiot ovat BCNF:ssä. Katso MyCoursesin Luennot-sivulta esimerkki vastauksesta, jossa on kaikki vaaditut asiat, mutta ei mitään turhaa.
- Relaatio on ositettu aivan oikein yhden kerran, mutta ei ole huomattu sitä, että toinen tai molemmat ositetuista relaatioista ei vielä ole BCNF:ssä ja ositusta pitäisi jatkaa.

Mahdollisia tenttikysymyksiä relaatioalgebrasta

- Pyydetään esittämään kysely relaatioalgebran lausekkeilla.
- On annettu relaation instanssi ja relaatioalgebran lauseke. Kysytään, mikä on lausekkeen tulos.
- Pyydetään selvittämään jotain yleisempää relaatioalgebran operaatioista, esim. miten jonkin operaation voi korvata toisilla operaatioilla tai miten monta monikkoa voi esiintyä lausekkeen tulosrelaation instanssissa, jos monikoiden määrät lähtörelaatioiden instansseissa tunnetaan.

Tyypillisiä virheitä relaatioalgebran tenttivastauksissa

- Lausekkeissa on käytetty luonnollista liitosta, vaikka attribuutit, joiden yhtäsuuruuden mukaan liitos tehdään, ovat erinimisiä.
- Samaan ehtoon on yritetty sisällyttää liikaa. Esimerkiksi: haettava niiden asiakkaiden asiakasnumerot, joilla on tilaus, jonka tila on 'in post', mutta ei yhtään tilausta, jonka tila on 'delivered'.
  - Lauseke

    $$\pi_{custNo}(\sigma_{status=\texttt{'in post' AND } status\neq\texttt{'delivered'}}(\texttt{Orders}))$$

    ei tuota haluttua vastausta, koska se tarkastelee vain yhden tilauksen tietoja kerrallaan.
  - Ehto ei estä sitä, että asiakkaalla olisi toinen tilaus, jonka status on 'delivered'.

SQL: yksinkertaiset kyselyt ja liitokset

- Esimerkki 1

```
SELECT prodName, description
FROM Products
WHERE price > 100.0;
```

- Esimerkki 2:

```
SELECT Customers.custNo, name
FROM Customers, Orders
WHERE Customers.custNo = Orders.custNo AND
          status = 'returned';
```

SQL: Alikyselyt

- SQL-kyselyn sisään voi kirjoittaa WHERE- tai FROM-osan sisään toisen kyselyn. Tällaista kyselyä sanotaan *alikyselyksi* (subquery).
- Esimerkki relaation tuottavista alikyselystä:

```
SELECT DISTINCT orderNo
FROM BelongsTo
WHERE productNo IN
    (SELECT number
     FROM Products
     WHERE description = 'camera'
    );
```

SQL: koosteoperaattorit

- SQL tarjoaa koosteoperaattorit SUM, AVG, MIN, MAX ja COUNT, joiden avulla voidaan laskea tilastoja jonkin relaation jonkin attribuuttien arvoista.
- Koosteoperaattoreita käytettäessä monikoita voidaan ryhmitellä GROUP BY-operaattorin avulla.
- Ryhmille voidaan antaa myös jokin koosteoperaattoria käyttävä ehto HAVING-osan avulla.
- Esimerkki:

```
SELECT manufID, manufName, AVG(price)
FROM Products, Manufacturers
WHERE id = manufID
GROUP BY manufID
HAVING MAX(price) > 100;
```

SQL: mahdollisia tenttikysymyksiä

- Tyypillisessä tenttikysymyksessä pyydetään kirjoittamaan jokin SQL-kysely.
- Kyselyissä voi tarvita mitä tahansa niistä ominaisuuksista, joita kurssin luentokalvoissa ja harjoitustehtävissä on käytetty.
- Myös taulujen luomiseen, monikoiden lisäämiseen ja tietojen päivittämiseen tarvittavia käskyjä voi tarvita, vaikka niitä ei olekaan tällä kertausluennolla esitelty.
- Eheysehtojen kirjoittamisesta tarvitsee osata niin paljon kuin luentokalvoilla ja harjoitustehtävissä on esitetty.
- Tentissä ei pyydetä kirjoittamaan laukaisimia, mutta on tunnettava laukaisimien toimintaperiaate ja ymmärrettävä, mikä ero on FOR EACH ROW-tyyppisillä ja FOR EACH STATEMENT-tyyppisillä laukaisimilla.

SQL: mahdollisia tenttikysymyksiä (jatkuu)

- Myös näkymien määrittely ja niiden käyttö on osattava siinä laajudessa kuin mitä luentomateriaalissa on määritelty.
- SQL-käskyjen liittämistä muulla kielillä kirjoitettuun ohjelmaan (joko kirjastofunktioita tai sulautettua SQL:ää käyttämällä) ei kysytä tentissä.

Tyypillisiä virheitä SQL-tenttitehtävissä

- Liitosehdot tai osa niistä puuttuu.
- Alikyselyn edessä käytetty IN-operaattorin sijasta yhtäsuuruusmerkkiä, vaikka alikyselyn tuloksena on relaatio eikä yksittäinen arvo.
- Ei ole ymmärretty, että WHERE-osassa oleva ehto käsittelee aina yhtä FROM-osassa olevien relaatioiden karteesisen tulon monikkoa kerrallaan. On kirjoitettu ehtoja, joiden pitäisi tutkia samalla kerralla useita monikoita.
- Kyselyssä pitäisi laskea koostefunktioiden arvoja tietyille ryhmille (esim. kunkin valmistajan tuotteiden hintojen keskiarvo), mutta GROUP BY-osa puuttuu kyselystä.
- Koostefunktiota koskeva ehto on kirjoitettu WHERE-osaan, vaikka se pitäisi olla HAVING-osassa.
- On käytetty sellaisia operaattoreita, joita kurssilla ei ole opetettu, mutta ei ole ymmärretty, mitä ne tarkoittavat. Esimerkiksi LEFT JOIN tarkoittaa samaa kuin LEFT OUTER JOIN. Monessa vastauksessa tätä on kuitenkin käytetty silloinkin, kun tarkoituksena on tehdä tavallinen liitos eikä ulkoliitos.

Hakemistot, tenttitehtävätyyppejä

- Tentissä voi olla tehtävä, jossa joutuu arvioimaan, mitä hakemistoja kuvatulle tietokannalle kannattaa luoda.
- Tentissä voidaan myös pyytää selostamaan yleisemmin, millaisia asioita pitää ottaa huomioon, kun päätetään tietokantaan tulevista hakemistoista.
- B-puusta ei tentissä tarvitse osata muuta kuin pystyä arvioimaan se, miten hakemiston käyttö vaikuttaa levyhakujen määrään kyselyissä ja päivityksissä.

Transaktiot (tapahtumat)

- Ohjelmoija voi suojata tietokannan monilta erilaisilta häiriöiltä, esimerkiksi sähkökatkoilta, laiterikoilta ja muiden samanaikaisten käyttäjien aiheuttamilta häiriöiltä määrittelemällä transaktioita, jotka voivat koostua useista tietokantaoperaatioista.
- Kun ohjelmoija on määritellyt joidenkin operaatioiden muodostavan transaktion, niin tietokannan hallintajärjestelmä pitää huolen siitä, että tietokanta toteuttaa seuraavalla kalvolla mainitut transaktioilta vaaditut ominaisuudet.

Transaktioilta vaadittavat ominaisuudet

- Atomicity (atomisuus): kaikki transaktion sisältämät käskyt suoritetaan tai mitään niistä ei suoriteta.
- Consistency (eheys): jos tietokannassa määritellyt eheysehdot ovat voimassa ennen transaktion suoritusta, niin ne ovat voimassa myös sen jälkeen.
- Isolation (serializability, sarjallistuvuus): transaktio suoritetaan niin kuin muita transaktioita ei suoritettaisi samaan aikaan. Toisin sanoen: jos useita transaktioita suoritetaan samanaikaisesti, niin lopputulos on sama kuin jos samat transaktiot olisi suoritettu jossain järjestyksessä peräkkäin yksi kerrallaan.
- Durability (pysyvyys): Jos transaktiot on suoritettu onnistuneesti loppuun (on suoritettu transaktion commit-operaatio), niin sen vaikutukset eivät katoa tietokannasta.
- Näitä ominaisuuksia kutsutaan usein *ACID-ominaisuuksiksi*.

Transaktioihin liittyviä tenttikysymyksiä

- Transaktiot on tärkeä aihe tentissä.
- Tentissä voidaan suoraan kysyä transaktioilta vaadittavia ominaisuuksia ja pyytää selittämään niitä tai osaa niistä tarkemmin.
- Tentissä voidaan myös kysyä, mitä muita eristyvyystasoja SQL:ssä on mahdollista määritellä transaktioille sarjallistuvuuden sijasta.