SISÄLLYS

LIITTEET

SQL-Trainer tietokannan luonti-SQL

1 Johdanto (KS)

Tähän loppudokumenttiin on yhdistetty toteutus-, testaus- ja ylläpitodokumentit. Dokumentissa esitellään sellaiset järjestelmän osat, jotka muuttuivat suunnitelluista toteutusvaiheessa ja testauksen aikana. Lisäksi on esitelty järjestelmän osat, joita ei mahdollisesti osattu suunnitteluvaiheessa kuvata riittävästi, tai joiden tarpeellisuutta ei osattu ottaa huomioon suunnittelun yhteydessä.

Järjestelmälle suoritettu testaus on käsitelty kyseisiä toimintoja käsittelevien lukujen yhteydessä, kuten mahdolliset järjestelmän parannusehdotukset. Oma kappaleensa käsittelee järjestelmän ylläpitoa ja asennusta.

1.1 Dokumentin kirjoittajat

Tässä dokumentissa esiintyneet kirjoittajat on mainittu kappaleiden otsakkeissa nimikirjaimin. Lyhenteet on esitetty taulukossa 1.

KS	Kimmo Sinkko
JN	Jaakko Nurro
ST	Santtu Toivakka
GE	George El-Khouri

Taulukko 1. Nimikirjaimet

2 SQL-laajennukset (ST,KS)

2.1 Laajennukset

Harjoitusohjelman SQL -puolta laajennettiin sisältämään uusia tarkistuksia kyselyn ja vastauksen oikeellisuudesta. Kyselyssä olevista tauluista tarkistetaan sisältävätkö ne jotain järjestelmän tauluja, jos näin on niin ohjelmisto palauttaa virheilmoituksen, joka näyttää Oraclelta tulleelta ilmoitukselta siitä, että taulua ei löydy. Lisäksi tarkistetaan, että kysely alkaa sanalla ”select”. Tämä varmistaa, että ohjelmalla ei voi tehdä kuin kyselyjä kantaan.

Vastauksen oikeellisuuden tarkistamiseen luotiin kaksi uutta metodia, joilla tarkistetaan onko taulussa kaikki tarpeelliset taulut ja että mitään tarpeettomiksi määriteltyjä tauluja ei ole mukana. Kumpikin tarkistuksista toimii siten, että verrataan opiskelijan kyselyssä antamien taulujen nimiä niihin mitä kantaan on tallennettu tehtäviä laadittaessa.

2.2 Luokkien kuvaukset

Luokka:

```
public class OAnswerInfo
```

Luokka SQL-tehtävien suorittamiseen, siitä tietojen hakemiseen sekä käyttöliittymän luomiseen.

Metodit:

```
private boolean neededTabs(boolean finnish, StringBuffer problems,
OTaskInfo tsk, Vector tnames)
```

Tarkistaa ovatko kaikki tarpeelliset taulut kyselyssä mukana vertaamalla vectoria, joka pitää sisällään kyselyssä esiintyneiden taulujen nimet ja merkkijonoa jossa on tieto tauluista jotka vaaditaan että tehtävä olisi oikein.

```
private boolean needlessTabs(boolean finnish, StringBuffer problems,
OTaskInfo tsk, Vector tnames)
```

Tarkistaa onko mukana tarpeettomiksi määriteltyjä tauluja vertaamalla vectoria joka pitää sisällään kyselyssä esiintyneiden taulujen nimet ja merkkijonoa jossa on tieto tauluista joita kyselyssä ei saa olla.

Luokka:

```
public class TabnameChecker
```

Luokkaa käytetään tarkistamaan yksittäisten taulunimien tai kokonaisen SQL –lauseen sisältämien taulunimien sallittavuutta.

Konstruktori:

```
public TabnameChecker(Connection conn)
```

Ilmentymää luodessa luetaan muistiin sellaiset taulunimet USER_TABLES ja USER_SYSNONYMS –järjestelmätauluista, jotka eivät ole SQL-Trainer järjestelmän systeemitauluja. Yksittäiset tarkistukset tämän jälkeen tapahtuvat muistissa olevasta rakenteesta, eivätkä siten rasita TKHJ:tä turhilla kyselyillä.

Metodit:

```
public boolean checkTabname(String checkName)
```

Palauttaa true, jos argumenttina annettu taulunimi on sallittujen taulunimien joukossa. Muuten palautetaan false.

```
public boolean checkSql(String Sql)
```

Palauttaa true, jos argumenttina annetun SQL –lauseen sisältämät taulunimet ovat kaikki sallittujen taulunimien joukossa. Muuten palautetaan false. Metodi käyttää apumetodinaan getTableNames:ia, joka poimii taulunimet SQL –lauseesta.

`public void showAll()`
Tulostaa stderr:iin kaikki muistissa olevat sallitut taulunimet.

`public static Vector getTableNames (String query)`

Palauttaa vektorin, jossa on argumenttina saadun SQL –lauseen sisältämät taulunimet.

2.3 SQL:n testaus.

SQL:n toimivuutta testattiin antamalla erilaisia toimivia ja toimimattomia syötteitä (=kyselyitä). Testauksen alla olivat uudet ominaisuudet, eli kyselystä laittomien taulujen etsintä ja vastauksesta puuttuvien sekä ylimääräisten taulujen etsiminen.

Seuraavia kyselyitä testattiin:

```
select * from pstask
select * from teacher
select * from eitälläistäole
select * from student.
```

Tehtävänannossa oli määrätty, että vastauksena on kyselyn ”select * from student” tulos ja tarpeettomaksi tauluksi oli määritelty taulu teacher.

Kyselyistä ensimmäinen epäonnistui sen vuoksi, että kyselystä löytyi järjestelmätaulun nimi. Järjestelmä tuotti odotetunlaisen virheilmoituksen. Seuraava kysely onnistui ja vastaus tulostettiin käyttäjälle kuten oli tarkoituskin, virheellinen vastaus tulkittiin oikein johtuvaksi sekä puutteellisesta taulun nimestä student, että tarpeettomasta taulun nimestä teacher johtuvaksi. Kolmas kysely tuotti asiaan kuuluvan virheilmoituksen siitä, että kyseistä taulua ei ole olemassa. Viimeinen näistä kyselyistä tuotti oikean vastauksen.

Edellisen kaltaisia kyselyitä testattiin useilla erilaisilla taulun nimillä ja kaikki tuottavat saman kaltaisia vastauksia, joten niitä ei listata tässä.

3 SQL –ylläpito-operaatiot (ST)

3.1 Perusteet

Lähtökohtana oli laajentaa tunnettujen SQL –operaatioiden valikoimaa INSERT, UPDATE ja DELETE käskyillä. Perusvaatimuksena oli se, että opiskelijan suorittaman tietoa muuttavat operaatiot eivät saa muuttaa perustehtävätaulujen sisältöä.

Ylläpito-operaatiot toteutettiin niin, että ne uudelleen kohdennetaan suoritettaviksi tauluun tmp_(alkuperäisen taulun nimi), joka on muuten saman muotoinen alkuperäisen taulun kanssa, paitsi että siihen on lisätty kenttä SID pitämään yllä tietoa minkä opiskelijan luomasta rivistä on kysymys.

Luokkarakenne tehtiin vastamaan Sqltr:n olemassa olevaa luokkarakennetta, ja käyttämään hyväksi valmiita luokkia aina kun se vaan oli mahdollista.

3.2 Vaatimukset ja rajoitteet

Laajennettu sqltr hyväksyy seuraavan tyyppiset ylläpito-operaatiot suoritettaviksi:

```
Insert into taulu values [(…)]
Insert into taulu select[ …]
Delete from taulu
Delete from taulu where […]
Update taulu set […]
Update taulu set [… where …]
```

Muu kuin hakasulkeissa oleva on pakollista, jotta järjestelmä tulkitsee operaation suorituskelpoiseksi, jos syntaksi on oikein niin operaatio yritetään suorittaa, ja sen jälkeisistä mahdollisista virheilmoituksista vastaa TKHJ.

Koska tämä laajennus tehtiin kokonaan omaksi osiokseen ei sillä pystytä tekemään muita kuin ylläpitoharjoituksia, eli select kyselyitä ei voida suorittaa ylläpitopalikalla.

Harjoitusvälineen toimiminen vaatii myös sen että oikean muotoiset tmp_ -alkuiset taulut luodaan kantaan samalla kun sinne luodaan muutkin harjoitussetin taulut.

Koska taulujen muutokset ovat voimassa vain sen hetken kun tehtävää tehdään, eivätkä muutokset näy todellisessa harjoituskannassa, niin ylläpitotehtävät eivät saa riippua toisistaan. Toisin sanoen tehtävät eivät voi olla muotoa ”ota tehtävässä n saatu tulos ja…” eikä tehtävää ei voida jatkaa toisessa tehtävässä kuten esimerkiksi ”jatka tehtävää n siten että…”

3.3 Luokkien kuvaukset

Uusista luokista on kuvaus kaikista metodeista, muokatuista vain niistä joihin on koskettu.

Luokka:

public class OUpdateAnswerInfo (muokattu OAnswerInfosta)

Luokka ylläpitotehtävien vastausten tallentamiseen, tietojen hakemiseen sekä käyttöliittymän luomiseen.

Konstruktorit:

public OUpdateAnswerInfo(String lang, String id, String sname, String tid, int trcnt, Connection conn, String query, String ccred, int errorCode, String emessage) throws DatabaseException

Tällä konstruktorilla talletetaan opiskelijan antama vastaus kantaan. Parametreina annetaan kieli, opiskelijan tunnus, opiskelijan nimi, tehtävän tunnus, yrityksen numero, yhteys, opiskelijan tekemä kysely, saako tehtävästä pisteitä, virhekoodi, virheteksti.

public OUpdateAnswerInfo(String lang, String id,String sname, String tid, int tries, String ccred, boolean aForThis, String forTid, Connection conn) throws DatabaseException

Tämä kontsruktori hakee edellisen vastauksen tiedot kannasta. Parametreina kieli, opiskelijan tunnus, opiskelijan nimi, tehtävän numero, yrityskerta, saako tehtävästä pisteitä, ?, ?, connection.

Metodit:

`public void buildAnswerForm(PrintWriter out)`

Metodi tulostaa käyttöliittymästä vastauksen antamiseen liittyvät osat, eli nappulat ja tekstikentän. Parametrina annetaan Printwriter johon tulostetaan.

`public void print(PrintWriter out)`

Metodi tulostaa käyttöliittymän kutsumalla buildAnswerFormia ja lisäksi tulostamalla itse edellisen vastauksen tiedot. Parametrina annetaan Printwriter
johon tulostetaan.

Luokka:

`public class Update (uusi)`

Luokka jonka avulla voidaan suorittaa ylläpito-operaatioita siten että ne eivät kohdistu alkuperäiseen tauluun vaan tauluun joka on saman muotoinen, mutta sisältää yhden uuden attribuutin, joka pitää sisällään tiedon rivit luoneen opiskelijan tunnuksesta.

Konstruktori:

`public Update(boolean fi, Connection conn, String qry, String sid)`

Luo uuden ilmentymän, jolle annetaan parametreinä kieli, Connection, kysely ja opiskelijan tunnus.

Metodit:

`public boolean copy() throws DatabaseException`
Kopioi tiedot harjoitustaulusta taulu tmp_taulu nimiseen tauluun lisäten mukaan opiskelijan tunnuksen. Kyselyn ollessa väärän muotoinen palautetaan false. Jos kopiointi epäonnistuu virheen takia niin koitetaan tyhjentää tmp_ -taulun sisältö sinne mahdollisesti jääneistä riveistä kutsumalla delete()-metodia.

`public void execute() throws DatabaseException`

Suorittaa muokatun ylläpito-operaation tmp_ -tauluun.

`public void delete() throws DatabaseException`

Poistaa tmp_ -taulusta johon tiedot kopioitiin kaikki rivit, jotka kuuluvat tälle opiskelijalle.

`public String getResultsetQuery()`

Palauttaa merkkijonona kyselyn, jonka suorittamisella saadaan tutkittua mitä tmp_ -taulussa on ylläpito-operaation jälkeen.

`public int getModifiedRows()`

Palauttaa ylläpito-operaatiossa muuttuneiden rivien lukumäärän.

`private String getQuery()throws DatabaseException`

Palauttaa muokatun ylläpito-operaation. Operaatiota muokataan siten että sinne laitetaan oikeaan kohtaan opiskelijan tunnus.

`private boolean validateUpdateStatement()`

Tarkistaa onko kysely järjestelmän kannalta hyväksyttävä ylläpito-operaatio, ja palauttaa true jos on.

`private String getTableName()`

Hakee kyselystä sen taulun nimen johon ylläpito-operaatio kohdistuu. Koska ylläpito-operaatiot kohdistuvat vain yhteen tauluun, niin haetaan ainoastaan ensimmäinen taulun nimi.

Luokka:

`public class OUpdateTaskPage extends HttpDBServletOra (muokattu OTask-Page)`

Hakee tiedot tehtävästä ja luo käyttöliittymän, silloin kun tehtävä ollaan valittu tehtävien valinta sivulta ja käyttöliittymä luodaan ensimmäisen kerran

Metodit:

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
throws ServletException,IOException
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException,IOException
```

Luokka:

```
public class OUpdateAnswerForm extends HttpDBServletOra (muokattu
OAnswerFormista)
```

Luokka käsittelee ylläpito-operaatio tehtävien vastaukset, tulostaa taulun operaation jälkeen ja testaa vastauksen oikeellisuuden.

Metodit:

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
        throws ServletException,IOException
```

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
throws ServletException,IOException
```

```
private boolean rowCount(boolean finnish, StringBuffer problems, OTas-
kInfo tsk, int rcount)
```

```
private boolean neededTabs(boolean finnish, StringBuffer problems,
OTaskInfo tsk, Vector tnames)
```

Tarkistaa onko kaikki tarpeelliset taulut kyselyssä mukana vertaamalla Vectoria joka pitää sisällään kyselyssä esiintyneiden taulujen nimet ja merkkijonoa jossa on tieto tauluista jotka vaaditaan että tehtävä olisi oikein.

```
private boolean needlessTabs(boolean finnish, StringBuffer problems,
OTaskInfo tsk, Vector tnames)
```

Tarkistaa onko mukana tarpeettomiksi määriteltyjä tauluja vertaamalla Vectoria joka pitää sisällään kyselyssä esiintyneiden taulujen nimet ja merkkijonoa jossa on tieto tauluista joita kyselyssä ei saa olla.

3.4 Ylläpito-operaatioiden testaus.

Ylläpito-operaatioiden suorittamista oikein testattiin seuraavilla tauluilla:

STHARKKA

ID (int)	NAME (varchar2)	TEXT(varchar2)
1	ST	Hipsu
2	ST	Nipsu
3	MT	Kipsu
4	MT	Vipsu

STHARKKA2

ID (int)	NAME (varchar2)	TEXT(varchar2)
1	ST	Hipsu
5	ST	Nipsu
6	MT	Kipsu
7	MT	Vipsu

TMPSTHARKKA

ID (int)	NAME (varchar2)	TEXT(varchar2)	**SID**

Tauluista kaksi ensimmäistä on harjoittelutauluja ja kolmas on väliaikaistaulu johon tehtävät suoritetaan.

Seuraavia ylläpitotehtäviä testattiin:

Delete from stharkka	Ok
Delete from stharkka where id>2	Ok
Delete from stharkka where id in (select id from stharkka2)	Ok
Delete from tmp_stharkka	Taulua tai näkymää ei ole.
Delete from jokutaulu	Taulua tai näkymää ei ole.
Delete * from stharkka	Syntax error (virheellinen ylläpitolause)
Insert into stharkka values (1,'st','st')	Eheysvirhe (yritettiin laittaa samaa pääavainta)
Insert into stharkka values (10,'','')	Ok
Insert into stharkka select * from stharkka2	Eheysvirhe (yritettiin laittaa samaa pääavainta)
Insert into stharkka select * from stharkka2 where id>5	Ok
Insert into stharkka (values (10,'',''))	Syntax error (virheellinen ylläpitolause)
Update stharkka set name='kissakala'	Ok
Update stharkka set name='kissakala' where id in (select id from stharkka2)	Ok

Samankaltaisia kyselyitä testattiin myös todellisessa harjoitus kannassa, operaatiot näyttivät tuottavan oikeanlaisen tuloksen.

4 Relaatioalgebran käyttöliittymä (ST,KS)

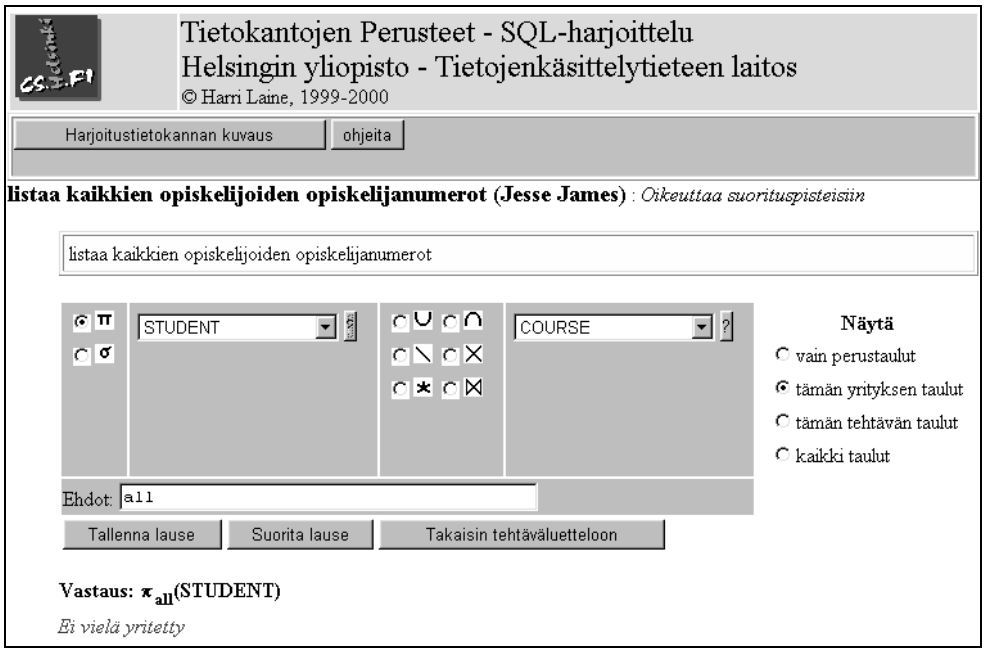
4.1 Yleistä

Tässä kappaleessa selitetään relaatioalgebran harjoitteluun liittyvä uusi käyttöliittymä, luokat, niiden sisältö ja mahdollisesti olemassa muutokset olemassa oleviin luokkiin.

Relaatioalgebran harjoittelu poikkeaa Sql:n vastaavasta siinä, että relaatioalgebra lauseketta joudutaan antamaan osissa. Servlet -puolella osat kootaan yhdeksi lausekkeeksi järkevästi joka kerralla, kun käyttäjä lisää uuden osan lausekkeeseen. Kun käyttäjä on tyytyväinen vastaukseensa ja painaa valmis -painiketta, lauseke on pystyttävä suorittamaan, jotta vastauksen oikeellisuutta voidaan tarkistaa.

Relaatioalgebran vastauksen oikeellisuuden tarkistaminen on samanlaista kuin SQL -kyselyn tarkistaminen, mutta lauseke on muutettava SQL -kyselyksi ennen tätä. Tästä johtuen on rakennettu AlgerbraParser -luokka hoitamaan muunnosta lausekkeesta kyselyyn.

Relaatioalgebran harjoittelun sivu eroaa SQL -harjoittelun sivusta siinä että kysely annetaan tekstikentän sijasta kuvassa 1. näkyvällä käyttöliittymällä.



Kuva 1. Relaatioalgebran käyttöliittymä

Tehtävänanto annetaan suomeksi tai englanniksi samaan tapaan kuin SQL -tehtävissäkin, lisäksi samaan tapaan sivu sisältään painikkeet, joilla pääsee opasteisiin käsiksi.

Käyttö tapahtuu askeleittain siten, että opiskelija valitsee haluamansa operaatioon sekä tarvittavan taulun tai taulut näytöltä vapaassa järjestyksessä. Valintojen jälkeen ”Tallenna lause” –painiketta painamalla kyseinen operaatio talletetaan kantaan niin että se saadaan näkymään taululistassa. Näytä –valinnasta opiskelija voi valita mitä tauluja (ja lauseita) hän haluaa taululistassa näkyvän

Harjoittelu etenee alkeisoperaatioita yhdistelemällä askel kerrallaan. Kun opiskelija katsoo saavuttaneensa tehtävänannossa pyydetyn tuloksen, hän lähettää vastauksen tarkastettavaksi näytöllä olevasta ”Suorita lause” -nappulasta.

Tauluvalinnan oikealla puolella olevasta ”?” –nappulasta voidaan tarkastella kyseisestä valintalistasta valitun taulun tai lausekkeen sisältöä. Sisältö avataan omaan ikkunaansa.

4.2 Toiminta

Suurin osa käyttöliittymän toiminnoista on tehty javascriptillä ja html:n lomakkeella. Sivun toiminnallinen osa on laitettu lomakkeeseen joka pitää sisällään useita piilotettuja (hidden) kenttiä, joissa on tieto opiskelijan tunnuksesta, tehtävän numerosta jne. Käyttäjälle näkyviä toiminnallisia kohteita ovat operaation valinta -nappulat, taululistat, ehdot -kenttä, taululistojen näkyvyyttä säätelevät painikkeet ja nappulat ”tallenna..” ja ”suorita…”. Kaikki edellä mainutut painikkeet kutsuvat erilaisia javascriptin funktioita, niin että tilanteesta riippuen joko päivitetään vastaus -kenttää tai sitten lähetetään lomake servletin käsiteltäväksi.

Operaatioiden valinta on tehty ’radio’-painikkeina, näitten painikkeiden avulla opiskelija voi valita minkä operaation hän suorittaa.

Ehdot-kenttään tulee laittaa operaatioon tarvittava lisätieto. Projektiota tehtäessä listataan halutut taulun sarakkeet pilkulla erotettuna.

Valintaa tai liitosta tehtäessä annetaan valintakriteeri loogisena lausekkeena, jossa saa esiintyä attribuuttien nimiä, sql:n arvojoukkojen arvoja, vertailuoperaattorit =, <, >, <=, >= ja <> sql:n syntaksin mukaisesti, sekä loogisia operaattoreita not, and ja or.

Opiskelija voi tallentaa minkä tahansa haluamansa välituloksen Tallenna-painikkeella, niin että se on valittavissa taululistasta. Näytön oikeassa reunassa olevat valintanapit on tarkoitettu suodattamaan listoja. Niillä voi valita näkyykö listoissa vain harjoitusjärjestelmän perustaulut, vain tämän yrityksen välitaulut, vain tämän tehtävän välitaulut vai kaikki käyttäjän kaikille tehtäville tallentamat välitaulut. Välitulokset esitetään taululistassa käyttäen, niin että relaatioalgebran operaatioita kuvataan kirjaimilla johtuen html:n rajoituksista, muuten relaatioalgebraa kuvataan mahdollisuuksien mukaan sen näköisen kun se pitääkin esittää.

Lomakkeelta välitetään seuraavat parametrit servletille ORAAnswerForm:

- operation
- kertoo minkä submitin käyttäjä on antanut
- htanswer
- vastaus html-muodossa
- answer
- vastaus algebraparserin hyväksymässä muodossa
- lang
- kieli
- sid
- opiskelijan tunnus
- tid
- tehtävän numero
- cred
- saako tehtävästä pisteitä
- trynum
- yrityksen numero
- sname
- käyttäjän nimi
- tables
- mitkä taulut näytetään

Loput parametreistä ovat vain käyttöliittymän javascript -osuuden käyttöön ja niitä ei hyödynnetä mitenkään muualla.

4.3 Javascript funktiot

`function opSelect(i)`

Päivittää vastauskentät htAnswer ja answer, sekä tulostaa vastauksen käyttäjälle näkyviin. Tätä kutsutaan kun käyttäjä vaihtaa operaation toiseksi.

`function removeTags(var s)`

Funktio poistaa merkkijonosta html -tagit ja korvaa ne tilanteesta riippuen suluilla tai tyhjällä. Funktiota käytetään htanswerin muuntamiseksi algebraparserin ymmärtämään muotoon

`function updateAnswer()`

Ehdot -kenttä ja taululistasta uuden valinta kutsuvat tätä funktiota, joka selvittää ensin mikä operaatio on valittuna ja sen jälkeen kutsuu opSelectiä.

`function saveStm()`

Tallenna -nappulan kutsuma funktio, joka vaihtaa operation parametrin arvoksi ’save’ ja lähettää lomakkeen eteenpäin.

`function tableContents(i)`

Luo uuden ikkunan ja kutsuu ShowTable-servlettiä, niin että käyttäjä näkee sen hetkisen taululistan valinnan sisällön, jos se vain on suoritettava lause.

`function backToList()`

Tehtävälistaan -nappulan kutsuma funktio, joka vaihtaa operation parametrin arvoksi ’back’ ja lähettää lomakkeen eteenpäin.

`function executeStm()`

Suorita -nappulan kutsuma funktio, joka vaihtaa operation parametrin arvoksi ’execute’ ja lähettää lomakkeen eteenpäin.

```
function updateTableLists()
```

Näytä -valinnan kutsuma funktio, joka vaihtaa operation parametrin arvoksi ’update’ ja lähettää lomakkeen eteenpäin.

4.4 Luokkien kuvaukset

`AlgebraParser`

Tämä luokka sisältää tarvittavat metodit SQL-kyselyn tuottamiseksi Relaatioalgebra-lausekkeesta. Luokan metodit eivät käytä ulkoisia tietovarastoja, parseNaturalJoin -metodia lukuunottamatta, jolloin luokka käyttää tietokantaa hyväksi luonnollisen liitoksen saman nimisten attribuuttien selvittämisessä.

Tässä luokassa Relaatioalgebra operaatioiden kuvaamiseksi käytetään seuraavassa taulukossa esitetyt merkintätavat. Nämä merkintätavat eivät käyttäjälle asti näy.

Relaatioalgebra operaatio	Tässä luokassa käytettävä merkintä tapa
Projektio	p(sarakkeet pilkulla erotettuna)(lauseke/taulu) ALL tarkoittaa kaikki attribuutit.
Valinta	s(ehdot loogisessa muodossa)(lauseke/taulu)
Ristitulo	(Taulu/lauseke) x (taulu/lauseke)
Liitos	(Taulu/lauseke) l(ehdot loogisessa muodossa) (taulu/lauseke)
Luonnollinen liitos	(Taulu/lauseke) * (taulu/lauseke)
Erotus	(Taulu/lauseke) / (taulu/lauseke)
Leikkaus	(Taulu/lauseke) I (taulu/lauseke)
Yhdiste	(Taulu/lauseke) U (taulu/lauseke)

Metodit:

`parse`

Koko muuntamisprosessin aloituskohta. Metodi on rekursiivinen, ja se käyttää luokan kaksi muuta metodia hyväkseen. Metodia **`split`** tämä metodi käyttää erottamaan kaksi saman tason lauseketta toisistaan, ja metodia **`parseNaturalJoin`** generoimaan luonnollisesta liitoksesta tavallista liitosta ehtoineen.

Parametrit : Relaatioalgebran lauseke
Palauttaa : Muunnettu relaatioalgebran lauseke SQL -kyselynä

`split`

Erottaa relaatioalgebralausekkeesta, jos sen rakenne on sellainen, että se on kaksi saman tasoista lauseketta yhdistettynä jollakin operaatiolla, kaksi saman tasoista Relaatioalgebra lauseketta toisistaan. Kahden saman tasoisen lausekkeen välillä on yksi seuraavista operaatioista: yhdiste, leikkaus, ristitulo, liitos, luonnollinen liitos ja erotus.

Parametrit : Relaatioalgebra-lauseke
Palauttaa : Merkkijono taulukko, jossa on joko kolme- tai yksi-paikka. Taulukko on kolmepaikkainen, jos parametrina annettu lauseke sisälsi kaksi saman tasoista lauseketta. Tällöin ensimmäisessä taulukon paikassa on ensimmäinen osa lausekkeesta, toisessa on lausekkeiden välinen operaatio ja kolmannessa on toinen lauseke. Jos taulukossa on vain yksi paikka, parametrina annettu lauseke ei sisältänyt kaksi saman tasoista lauseketta.

`getAnswerAttributes`

Tämä metodi suorittaa parametrikseen saamansa SQL-kyselyn, ja palauttaa kyseisen kyselyn tuottaman taulun kaikkien attribuuttien nimet Vectoriin sijoitettuna.

Parametrit: SQL-kysely
Palauttaa: Vector, joka sisältää parametrina annetun kyselyn vastauksen kaikkien attribuuttien nimet.

parseNaturalJoin

Tämä metodi tuottaa luonnollisesta liitoksesta vastaava ristituloon, valintaan ja projektioon perustuva SQL -kysely. Metodi käyttää hyväkseen pars -metodia, tuottaakseen luonnollisen liitoksen molemmista osapuolista SQL -kyselyitä. Tämän jälkeen metodi suorittaa molemmat kyselyt tietokannassa. Tällöin tietokannasta tulevista vastauksista saadaan molempien kyselyiden attribuuttien nimet, joista valitaan sellaiset, jotka sisältyvät molempiin tauluihin. Näiden ja alkuperäisten lausekkeiden avulla rakennetaan vastaava SQL -kyselyä.

Parametrit : Relaatioalgebralauseke, joka koostuu kahdesta saman tasoisesta lausekkeesta joita yhdistää luonnollisen liitoksen operaatio.
Palauttaa : parametrina annetun relaatioalgebralauseketta vastaava SQL -kysely.

getRistiTulo

Tämä metodi tuottaa ristitulosta vastaavan SQL -kyselyn. Tämä metodi käyttää edellä mainittua getAnswerAttributes -metodia, saadakseen listaa kaikista attribuuteista, jotka muodostavat jokaisen ristituloon osallistuneen kyselyn vastaus. Tämän jälkeen käydään molemmat listat läpi, ja kun saadaan sellaisen attribuutin nimi, joka esiintyy molempien kyselyjen vastauksissa, lisätään ensin oikealla puolella olevan kyselyn attribuutin alkuun sana ”second.”. Jos sellainenkin attribuutin nimi esiintyy, lisätään tämän attribuutin loppuun ”_” niin kauan kunnes se on erilainen kuin kaikki muut.

Parametrit: Ristitulon molemmat osapuolet (kyselyt).
Palauttaa : Ristitulosta sellainen SQL -lauseke, joka voidaan suorittaa tietokannassa.

Luokka:

public class ORelAnswerInfo (muokattu OAnswerInfosta)

Luokka relaatioalgebran tehtävien vastausten tallentamiseen, vastauksen tietojen hakemiseen sekä käyttöliittymän luomiseen.

Konstruktorit:

```
public ORelAnswerInfo(String lang, String id, String sname, String
tid,
            int trcnt, Connection conn,
            String query, String ccred, int errorCode, String emessa-
ge)
            throws DatabaseException
```

Tämä tallettaa vastauksen kantaan.

```
public ORelAnswerInfo(String lang, String id,String sname, String tid,
                  int tries, String ccred,
                   boolean aForThis,  String forTid, Connection conn)
      throws DatabaseException
```

Tämä hakee edellisen vastauksen tiedot kannasta.

Metodit:

```
public void buildAnswerForm(PrintWriter out)
```

Metodi tulostaa käyttöliittymän lomakkeen. Parametrina annetaan Printwriter johon tulostetaan.

```
public void print(PrintWriter out)
```

Metodi tulostaa käyttöliittymän osan jossa kerrotaan edellisen vastauksen tuloksesta, lisäksi tässä kutsutaan buoldAnswerForm metodia. Parametrina annetaan Printwriter johon tulostetaan.

```
public void setTableList(Connection conn, String tabs) throws Data-
baseException
```

Tällä haetaan kannasta tieto mitkä kaikki taulut tulostetaan käyttöliittymän taululistaan.

Luokka:

```
public class RelTaskPageMenu (muokattuu TaskPageMenusta)
```

Tulostaa käyttöliittymään painikkeet, joista saadaan esiin opasteita.

Konstruktori:

```
public RelTaskPageMenu(String lang)
```

Metodit:

```
public void print(PrintWriter out)
```

Luokka:

```
public class ORelTaskPage extends HttpDBServletOra (muokattu OTaskPa-
gesta)
```

Hakee tiedot tehtävästä ja luo käyttöliittymän, silloin kun tehtävä ollaan valittu tehtävien valinta sivulta ja käyttöliittymä luodaan ensimmäisen kerran.

Metodit:

```
public void doPost (HttpServletRequest req, HttpServletResponse res)
        throws ServletException,IOException

public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException,IOException  {
```

Luokka:

```
public class ShowTable extends harkka.HttpDBServletOra
```

Tämä oma servlet –luokkansa on tehty ?-painikkeen toteuttamista varten relaatioalgebran käyttöliittymässä. Tarkoituksena on näyttää käyttäjän valitseman relaation sisältö omassa popup –selainikkunassa, jos relaatio ei sisällä turvatarkistuksen kannalta kiellettyjä tauluja.

Metodit:

public void doPost (HttpServletRequest req, HttpServletResponse res)
throws ServletException,IOException

Servletin toteutus on doPost luokassa. Req –ilmentymästä etsitään kenttää alStatement, joka on järjestelmän sisäisessä muodossa esitetty relaatioalgebran lauseke. Jos kenttää ei löydy, annetaan virheilmoitus.

Jos kenttä löytyy, käytetään AlgebraPareser –luokkaa kääntämään lause SQL –muotoiseksi. Tämän jälkeen käytetään TabnameChecker- luokkaa tarkistamaan taulunimet SQL –lauseesta. Jos taulunimet ovat sallittuja, tulostetaan HTML –muodossa argumenttina annetun relaation sisältö.

```
public void doGet (HttpServletRequest req, HttpServletResponse res)
        throws ServletException,IOException
```

Ainoastaan käyttää doPost –metodia.

```
private void makeHeaderRow(ResultSet rsh,PrintWriter out)
```

Luo HTML tulosteen, jossa avataan taulukko ja tulostetaan sen ylimmälle riville servletin saaman relaation sisältämien kenttien nimet. Tähän tarvitaan ResultSet -olio, jonka metadatasta kenttien nimet noudetaan.

4.6 Relaatioalgebran käyttöliittymän testaus

Relaatioalgebrassa käytetty käyttöliittymä testattiin siten että lomakkeen kohde vaihdettiin osoittamaan tyhjää ja lähetysmuodoksi vaihdettiin 'get'. Näillä muutoksilla oli mahdollista varmistaa selaimen osoitekentässä näkyvä parametri rivi on oikeanlainen. Tämä tosin tehtiin vasta sitten kun käyttöliittymän javascript -funktiot oli todettu ainakin suurimmaksi osaksi toimiviksi halutulla tavalla.

Tärkeimpänä kohteena testissä oli se että tuottaako käyttöliittymä oikean laista tulostetta htanswer ja answer parametreihin. Jotta testit olisivat mielekkäitä, niin ainoaan käyttäjän vapaasti muokattavaan kenttää jätettiin oletusarvo all, joka vastaa lauseessa sitä että ehdoiksi annettaisiin kaikki taulun sarakkeet. Tämä testi siis suoritettiin kaikille kahdeksalle operaatiolle käyttäen erilaisia taulun nimiä ja todettiin että parametrin välityksen pitäisi näiltä osin olla kunnossa.

Käyttäjän antamaa ehdot -kentän arvoa ei missään vaiheessa tarkisteta, joten sen kautta on mahdollista syöttää virheellistä tietoa järjestelmää, tosin virheellinen tieto aiheuttaa sql-virheen kun parsittu relaatioalgebranlause yritetään suorittaa sql:nä. Tämän vuoksi ei ehdot kentässä tässä testeissä käytetty muita arvoja.

Käyttöliittymän ja sen alla olevan osan välillä suoritettiin testejä suorittamalla muutamia annettuja tehtäviä. Tällaisia olivat esimerkiksi: "valitse taulun teacher kaikki rivit" tai "valitse kaikki 'Eskola' sukunimen omaavat taulusta student". Nämä lauseet suoritettiin onnistuneesti parserilla.

Lisäksi testattiin muuta parametrin välitystä tallentamalla erilaisia lauseita 'tallenna' -painikkeella ja katsomalla mitä tauluja taululistoissa näkyy koittamalla 'näytä' -valinnassa erilaisia arvoja.

Samalla edellä mainituilla testeillä tuli myös testattua luokkien ORAAnswerForm ja ORelAnswerInfo toimintaa.

4.5 AlgebraParserista toteuttamatta jääneet osat

Ajan puutteen, ja toteutuksen monimutkaisuuden vuoksi relaatioalgebran parserista ei voitu tehdä täydellistä eli sellaista, joka osaisi kääntää kaikki relaatioalgebran lausekkeet SQL:ksi.

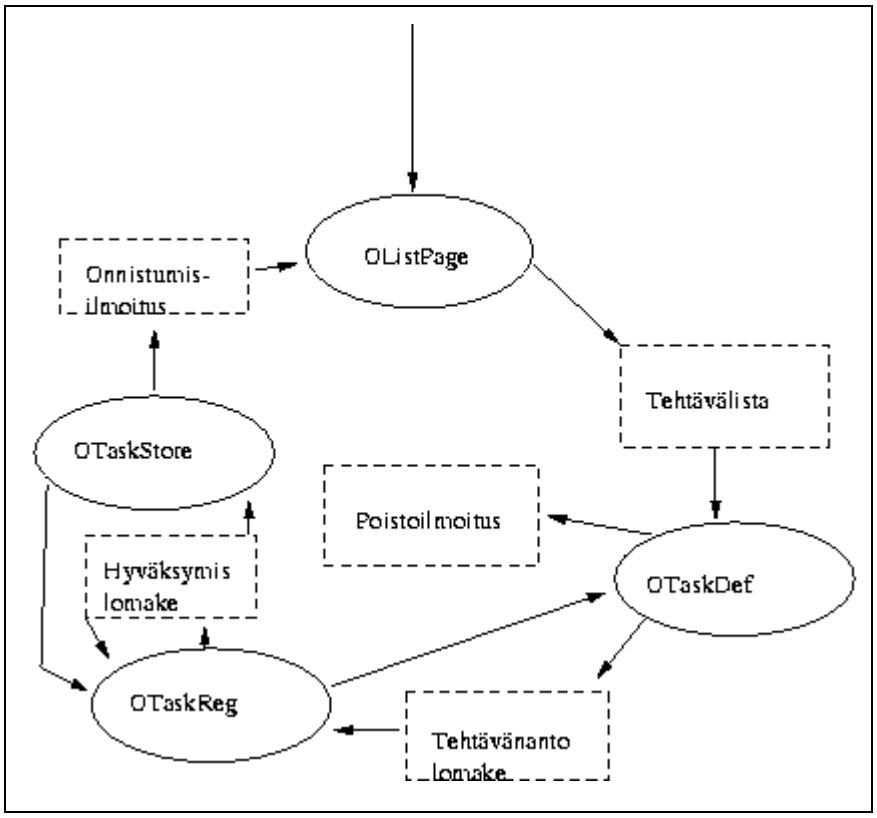
Parserissa käännetään kaikki mahdolliset relaatioalgebralausekkeet vastaaviksi SQL -kyselyiksi. Tarkistuksia, ja uudelleen nimeämisiä on toteutettu parserin toimintoina. Ainoa relaatioalgebran osa, johon ei uudelleen nimeämisiä, eikä tarkistuksia ole toteutettu on liitosoperaatio (tavallinen liitos).

Liitosoperaation kääntäminen SQL -lausekkeeksi käy parserilta hyvin, mutta ongelmia esiintyy parserin tuottaman SQL -lausekkeen suorittamisessa, kun molemmat liitettävät sisältävät samannimisiä attribuutteja. Niiden uudelleennimeäminen ei ole vaikeata (vrt. ylempänä), mutta jos nämä attribuutit esiintyvät myös liitoksen ehto -osassa niin toteuttaminen monimutkaistuu.

Ajan puutteen vuoksi tätä ei pystytä toteuttamaan, mutta samanlaisen operaation saa aikaan ottamalla ensin tauluista tai lausekkeista projektion, jossa uudelleen nimetään tarvittavat attribuutit ja vasta tämän jälkeen liitoksen.

5 Tehtävänantojärjestelmä (JN)

Järjestelmä on tarkoitettu hieman helpottamaan SQL-Trainerin tehtävätietokannan ylläpitoa webipohjaisin lomakkein. Tähän tarkoitukseen järjestelmä tarjoaa mahdollisuuden lisätä, muokata ja poistaa kannan tehtävänantoja. Järjestelmä koostuu kolmesta perusnäytöstä.



Kuva 2. Tehtävänannon ivurakenne

- Tehtävälistasivu, jolta voi valita tehtävän, jota haluaa käsitellä tai jonka haluaa poistaa. Mukana on myös kokonaan uuden tehtävänannon antamisvaihtoehto.

- Tehtävänantolomake, jolla käyttäjä antaa tehtävänannon tiedot, tai muokkaa niitä. Lisäksi tällä lomakkeella katsomisen yhteydessä tehtävä voidaan poistaa kannasta.

- Tehtävänannon hyväksymislomake, jonka avulla käyttäjä voi hyväksyä aiemmin antamansa tehtävänannon tallennettavaksi kantaan. Tällä lomakkeella voi myös muuttaa tehtävänantoa.

Tämän lisäksi on erilaisia viestinäyttöjä, jotka antavat palautetta käyttäjälle.

5.1 Järjestelmän toiminnot

Järjestelmällä pystyy siis lisäämään tehtävänantoja kantaan, muokkaamaan kannassa jo olevia tehtävänantoja, ja poistamaan tehtävänantoja kannasta.

5.1.1 Tehtävän antaminen

Tehtävän antaminen onnistuu siten, että käyttäjä painaa tehtävälistasivulta Lisää uusi -painiketta. Tästä painikkeesta saa järjestelmältä lomakkeen, johon voi syöttää tehtävänannon tiedot. Tehtävänannon tiedoista välttämättömiä ovat tehtävänumero ja tehtävän suomenkielinen otsikko. Täyttämättä näitä ei kantaan saa mitään.

Relaatioalgebratehtävän laatiminen onnistuu samalla tavalla kuin SQL-kyselytehtävänkin laatiminen. Käyttäjän täytyy vain muistaa merkitä tehtävä relaatioalgebratehtäväksi kohdasta 'Tehtävän tyyppi'.

Tehtävänantolomakkeen kentät kuvauksineen:

Tehtävänumero: tähän tulee syöttää tehtävänumero. Järjestelmä sallii käyttäjän itse valita tehtävien numeroinnin aika vapaasti, joten käyttäjä joutuu itse pitämään huolta, ettei vahingossakaan yritä syöttää tässä jo kannasta löytyvää tehtävänumeroa. Tehtävänumero 0 (nolla) on kuitenkin tässä järjestelmässä erityisasemassa, joten sitä ei suositella käytettäväksi, jos käyttäjä haluaa pystyä vielä muokkaamaankin tehtävää, jolle tämän numeron antaisi.

Nimi suomeksi: tähän tulee antaa tehtävännimi suomeksi, sillä järjestelmä tarjoaa uudestaan lomaketta täytettäväksi, jos tästä puuttuu tieto.

Nimi englanniksi: tähän kenttään voi antaa ylläannetun tehtävän otsikon englanninkielisen version.

Teksti suomeksi: tähän kenttään voi antaa tehtävänannon tarkemman kuvauksen.

Teksti englanniksi: tähän kenttään voi antaa tehtävänannon tarkemman kuvauksen englanniksi.

Mallivastaus: tähän tulee antaa tehtävän mallivastaus, josta lasketaan tehtävän tarkistustiedot, ja jota käytetään muutenkin tarvittaessa esimerkkivastauksena. Tehtävistä ei tule mielekkäitä, jos tämän jättää täyttämättä. Lisäksi järjestelmä varoittaa aiheesta.

Järjestetty: tähän kenttään tulee E, jos vastauksen järjestyksellä ei ole väliä, T jos sillä on. Kentän arvoa ei kuitenkaan erityisemmin tarkisteta.

Samat: tähän kenttään voi laittaa listan vastauksen sarakepareista, jotka ovat ekvivalentteja. Lista tulee antaa muodossa 'sarake1=sarake2,sarake3=sarake4', kuitenkin ilman lainausmerkkejä. Ylimääräiset merkit tässä kohdassa saattavat johtaa siihen, että järjestelmä ei hyväksy opiskelijoiden vastauksia tehtävään.

Aihepiiri: valikko, josta voi valita tehtävän aihepiirin. Tämän kentän laiminlyöminen saattaa johtaa siihen, että tehtävä ei näy opiskelijoille. Tämän takia aihepiiri pitää valita.

Tehtävän tyyppi: valintapainikkeet, joista tulee valita onko tehtävä SQL-kyselytehtävä, SQL-ylläpitotehtävä vai relaatioalgebratehtävä. Tehtävän tyyppiin on otettava kantaa. Järjestelmä käsittelee
tehtävätyypit eri lailla, joten väärä valinta tässä vaiheessa voi johtaa ennalta odottamattomiin tuloksiin.

Järjestysperuste: jos tehtävän vastauksen järjestyksellä on väliä, tähän käyttäjä voi kopioida esimerkkikyselyn order by -ehdon, tai jonkin muun vihjeen, jonka haluaa opiskelijalle antaa.

Turhat taulut: tähän voi käyttäjä listata tehtävän ratkaisun kannalta tarpeettomia tauluja muodossa 'A,B,C', ilman lainausmerkkejä. Ylimääräiset merkit (jopa välilyönnit) tässä kohdassa saattavat johtaa siihen, että järjestelmä ei hyväksy opiskelijoiden vastauksia tehtävään.

Lomakkeen täyttämisen jälkeen käyttäjä pääsee eteenpäin painamalla jatka-painiketta. Lomakkeen saa halutessaan aloittaa uudelta pohjalta tyhjennettyä kokonaan. Uudelle sivulle tulostuu esimerkkikyselyn vastaustaulu ja tehtävänantolomake uudestaan mahdollisia korjauksia varten. Vastaustaulun sarakkeista voi käyttäjä valita valintanapeilla sen, josta lasketaan tehtävän tarkistussumma. Tämän valitsematta jättäminen johtaa siihen, että tehtävää ei voi ratkaista, joten

valitsemista suositellaan. Jos suorituksessa ilmeni ongelmia, käyttäjä saa niistä virheilmoituksen. Virheiden ilmetessä kannattaa korjata lomakkeen tiedot, ja suorittaa esimerkkikysely uudestaan lomakkeen alla olevasta painikkeesta. Käyttäjän kannattaa varmistaa, että on valinnut sarakkeen, josta on tarkoitus laskea tarkistussumma. Virheellisiä tehtävänantoja ei kannata yrittää saattaa kantaan. Se voi onnistua, mutta siitä ei ole mitään hyötyä.

Kun käyttäjä on antanut tehtävänannon tiedot mielestään oikein, hän voi painaa Talleta tehtävä -painiketta tallentaakseen tehtävän. Jos tehtävän tallennus onnistui, aiheesta tulee ilmoitussivu, jolta pääsee takaisin tehtävälistaan. Virheistä tulee virheilmoitus tai järjestelmä palaa takaisin hyväksymislomakkeeseen. Tallennuksessa tapahtuvista virheistä joutuu palaamaan selaimen back -painikkeella takaisin.

5.1.2 Tehtävänannon muokkaaminen

Käyttäjä voi muokata valmiita tehtävänantoja valitsemalla tehtävän vastaavasta painikkeesta tehtävälistasta. Poistamis-painiketta on syytä varoa, sillä siitä ei erikseen varoiteta, eikä poistamistoimintoa voi perua.

Muokkaaminen tehdään samanlaisella lomakkeella kuin tehtävänannon
antaminenkin (ks. yltä), tosin lomakkeessa on jo valmiiksi tehtävästä kantaan
aiemmin tallennetut tiedot. Tehtävänumeron muokkaaminen ei onnistu enää sen antamisen jälkeen. Muokkaamisenkin yhteydessä poistamista on syytä varoa. Hyväksyminenkin onnistuu samalla tavalla kuin kokonaan uutta tehtävää laadittaessa, mutta käyttäjän on syytä pitää huolta siitä, että käyttäjä myös valitsee sarakkeen, josta tarkistussumma lasketaan. Järjestelmä saattaa unohtaa sen.

5.1.3 Tehtävän poistaminen

Tehtävän poistaminen onnistuu helposti. Käyttäjä voi halutessaan poistaa tehtävän poistamispainikkeella suoraan tehtävälistassa painamalla, tai sitten hän voi päättää poistamisesta tutustuttuaan tehtävään ensin painamalla muokkaamislomakkeen tehtävänpoistopainiketta. Poistamisessa suositellaan harkintaa, sillä poistamisesta tulee ainoastaan ilmoitus onnistumisesta, eikä sitä varmisteta tai voi perua.

5.1.4 Virhe- ja ongelmatilanteet

Käytössä saattaa esiintyä ongelmatilanteita. Tunnetut tilanteet käsitellään tässä.

Virhe:

"Esimerkkikyselyn suorittaminen ei onnistunut."

Tämä voi johtua monista syistä. Ilmoituksen yhteydessä tulevasta tietokantajärjestelmän virheilmoituksesta pystyy päättelemään vian. Yleensä virhe johtuu kielioppivirheellisestä esimerkkikyselystä.

"Muutettavaa tehtävä ei löydy."

Tehtävä ei joko ole kannassa, tai yhteydessä on tullut ongelmia.

"Suoritus ei onnistunut."

Tehtävän tallettamisessa tai päivittämisessä kantaan tapahtui virhe. Lisätietoja saa tietokantajärjestelmän virheilmoituksesta ja tulostuvasta taulunpäivityslauseesta.

"Tarkistussumman laskemisessa tuli virhe."

Tarkistussumman laskeminen ei onnistunut. Kannattaa yrittää uudestaan hyväksyttää tehtävänantoa tai suorittaa esimerkkikysely uudestaan. Paluu takaisin hyväksymislomakkeelle back - painikkeella.

"Tehtävälistan teko ei onnistunut."

Tarkempi syy on ongelmat kantayhteydessä tai kannan sisällössä. Virheilmoituksen yhteydessä tulevasta järjestelmän virheilmoituksesta voi päätellä enemmän.

"Tehtävän tyyppi ja esimerkin toimintaperiaate eivät täsmää."

"Ylläpitotehtävä alkaa selectillä."

Tehtävä on merkitty ylläpitotehtäväksi (lomakkeen kohdassa Tehtävän tyyppi), mutta se on kyselytehtävä. Tämän korjauksena tulee muuttaa joko esimerkkikyselyä tai tehtävän tyyppiä.

"Tietokantavirhe."

Tehtävän hakeminen tai muokkaaminen epäonnistui.

"Tietokantavirhe poistettaessa tehtävänantoa."

Taustalla ovat ongelmat kantayhteydessä tai kannan sisällössä. Virheilmoituksen yhteydessä tulevasta järjestelmän virheilmoituksesta voi päätellä enemmän. Tehtävä voi poistua tai olla poistumatta kannasta tämän virheen yhteydessä.

"Tietokantayhteydessä ongelmia"

Tämä tarkoittaa myös sitä, että ongelma johtuu järjestelmästä, ei käyttäjästä. Ilmoitus on varoituksenomainen.

"Tulostaulun tulostaminen ei onnistunut."

Tämäkin voi johtua monista syistä. Tietokannan virheilmoitus auttaa ymmärtämään missä virhe on.

"Väliaikaistauluja poistettaessa tuli virhe."

Tämä on lähinnä varoitus mahdollisesta ongelmasta. Käyttäjällä ei ole juurikaan mitään osaa virheeseen

"Yhteyttä ei saatu"

Tehtävää ei tietokantayhteyden puutteessa talletettu kantaan.

Ongelma:

Hyväksymislomakkeelta ei pääse eteenpäin, vaan saa sen eteensä uudestaan ja uudestaan

Tehtävästä on kerrottava tehtävänumero ja suomenkielinen otsikko. Järjestelmä ei anna tästä virheilmoitusta (ks. parannusehdotukset), joten tämä voi hämmentää käyttäjää.

Poistettaessa tehtävää kannasta poistuu väärä tehtävä

Tämä johtuu siitä, että käyttäjä on muuttanut tehtävänumeroa muokatessaan tehtävää, ja sitten painanut poistamispainiketta. Ikävän yhteensattuman takia kannasta poistetaan tehtävä, jolla on sama tehtävännumero kuin jota harkittiin poistettavalle tehtävälle.

'Rivejä tuloksessa, kun esimerkkikyselystä on poistettu distinct -määreet' on nolla, vaikka ei mitenkään pitäisi

Ylläpitotehtävistä ei lasketa noita rivejä, joten niiksi tulee nolla. Kyseessä on järjestelmän virhe, jota ei ole ehditty korjata. Jos tulos tulee jossain muussa tilanteessa, kannattaa korjata esimerkkikyselyä.

Tehtävää ei voi ratkaista, vaan SQL-Trainer valittaa hämmentävistä asioista, jotka ovat ihan oikein tehtävässä

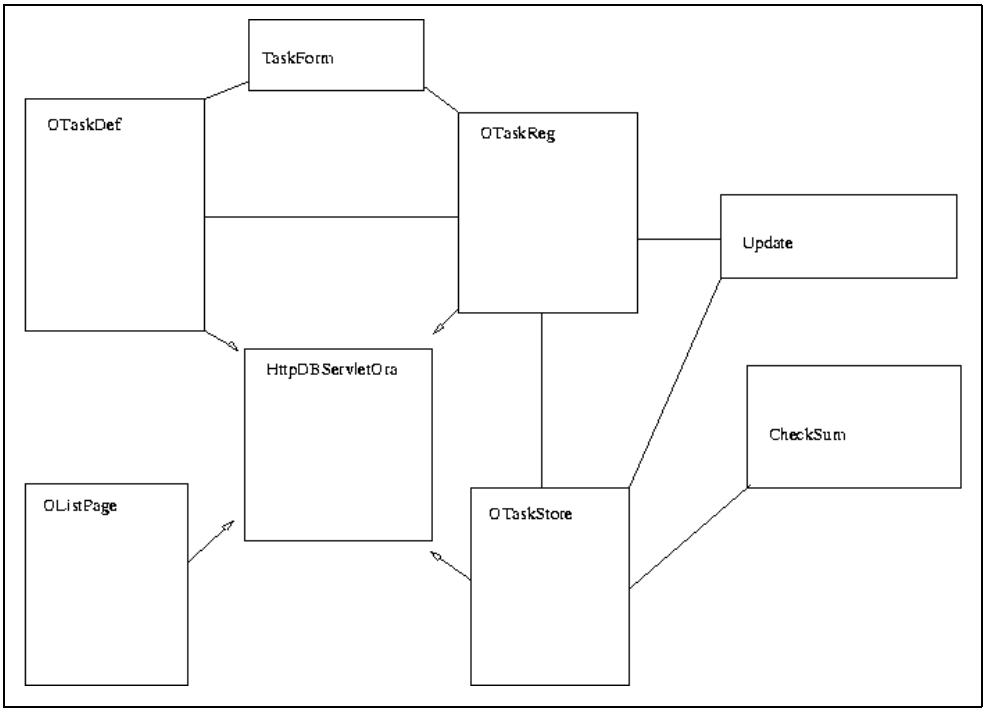
Syy voi olla puutteellinen tehtävänanto. Kannattaa ottaa tehtävänanto muokattavaksi, ja katsoa että kaikki tiedot ovat pilkulleen kuten pitääkin. Erityisesti tarkistussumman laskemisesta kannattaa olla tarkka, ja muistaa valita tarkistussarake aina ennen kuin hyväksyy tehtävän kantaan.

Yrittäessä muokata tehtävää järjestelmä antaakin tyhjän tehtävänantolomakkeen

Voi johtua siitä, että tehtävänumeroksi on laitettu nolla. Numero näkyy tehtävälistassa, ja sen voi muuttaa joko poistamalla tehtävän ja tekemällä uuden, toimivan, tai sitten suoraan kannasta säätämällä.

5.2 Luokkien kuvaukset

Tehtävänantojärjestelmä koostuu neljästä servletistä, OListPage, OTaskDef, OTaskReg ja OTaskStore, jotka mahdollistavat taulun pstask tehtävien ylläpitämisen. Servletit käyttävät apuluokkia, joista TaskForm vastaa tehtävänannon syöttämistä varten tarkoitettua lomaketta, ja kuvataan tässä servlettien yhteydessä. Tässä esitetyille servleteille ja luokalle TaskForm ei juurikaan ole muuta käyttöä kuin tämän järjestelmän osasina. Servletit tukevat sekä post- että get-operaatioita, eli korvaavat yliluokastaan HttpDBServletOra metodit doGet ja doPost. Toteutus on kaikissa neljässä servletissä kummallekin operaatiolle sama. Lomakkeiden kenttien nimet pyrkivät vastaamaan miltei täysin taulun pstask vastaavien attribuuttien nimiä.



Kuva 3. Tehtävänannon luokkarakenne

Luokka:

`OListPage`

Luokka tuottaa avaussivun, jolta voidaan valita tehtävä, ja mitä sille tehdään. Sivulta voi myös lähteä tekemään uutta tehtävää.

Metodit:

doGet ja doPost

Metodit ovat ainoat tässä luokassa toteutetut metodit. Ne hakevat kannasta tehtäväpiirien otsikot, numerot ja kelpuutusajat sekä tehtävien suomenkieliset nimet ja tunnukset ja tulostavat kursorisilmukassa näistä sivun, jolla tehtävien nimet ovat tehtävännumerojärjestyksessä, yläotsikkoinaan tehtäväpiirin otsikot. Metodit tulostavat jokaista taulun pstask tehtävää tehtävää kohden oman html-lomakkeen, jolla on painikkeet tehtävän muokkaamista tai poistamista varten. Lomakkeiden tiedot lähetetään servletille OTaskDef.

Jos tehtävälistaa varten tehtävä kysely ei jostain syystä onnistu, vaan tuottaa poikkeuksen SQLException, metodit tulostavat virheilmoituksen aiheesta ja siivoavat jäljet. Virhe tapahtuu yleensä ottaen vain jos kantayhteydessä on vikaa tai kannassa ei ole tarvittavia tietoja.

Luokka:

OTaskDef

Luokka käsittelee tehtävien käsittelyyn tulevat pyynnöt, kuten muokkaamis- tai luomislomakkeen tulostamisen tai tehtävän poiston kannasta.

Metodi:

delTask (int tid, PrintWriter out)

Metodi suorittaa tehtävän poiston kannasta. Se saa parametrikseen tehtävän tunnusluvun (muuttuja tid) ja tulosteita varten viitteen PrintWriter -olioon (muuttuja out). Metodi pyytää yliluokalta HttpDBServlerOra perityltä ConnectionBroker -oliolta yhteyden tietokantaan, ja poistaa tämän yhteyden kautta taulusta pstask rivin, jonka task_id -arvo vastaa metodin parametrin tid arvoa. Tämän jälkeen metodi tulostaa tiedon tapahtuman onnistumisesta. Jos poistoa suoritettaessa tapahtuu jokin virhe, järjestelmä tulostaa virheilmoituksen.

doGet ja doPost

Metodit hoitavat tehtävien käsittelypyynnöt. Ne määrittävät HttpServletRequest -olion parametrien perusteella, pitääkö tehtävä poistaa, onko kyseessä uusi tehtävä, vai muokataanko jo olemassa olevaa tehtävää. Järjestelmä tunnistaa historiallisista syistä uuden tehtävän siitä, että sen task_id on arvoltaan 0, minkä takia jos järjestelmään syöttää tehtävän tunnuksella 0, tätä tehtävää ei pysty muokkaamaan.

Jos kyseessä on jo olemassa oleva tehtävä, sen tiedot haetaan kyselyllä, muussa tapauksessa tehtävän oletusarvoiksi tulee tyhjää. Tehtävän tiedoista rakennetaan TaskForm -luokan olio, joka tulostetaan osaksi lomaketta, jolla tehtävänanto alustavasti annetaan servletille OtaskReg joko hyväksymistä tai tehtävän poistoa varten.

Luokka:

`OTaskReg`

Luokka tulostaa lomakkeen, jolla tehtävän voi lopullisesti hyväksyä kantaan tallennettavaksi. Tätä varten servlet suorittaa esimerkkikyselyn ja laskee pstaskiin tulevat arvot, joita käyttäjän ei tarvitse syöttää. Lisäksi servlet pyytää servlettiä OTaskDef poistamaan tehtävän kannasta, jos siltä on tätä pyydetty.

Metodit:

`String nonDistinctify(String query)`

Metodi on yksityinen apumetodi, jota doPost -metodi käyttää saadakseen esimerkkikyselystä version, josta on poistettu distinct -määreet. Algoritmi on yksinkertaisuudessaan seuraava: metodin parametrinaan saama kysely annetaan StringTokenizer -oliolle, jonka avulla se kopioidaan pala palalta StringBuffer -olioon. Ainut pala, jota ei kopioida, on avainsanaa select seuraava distinct -määre. Metodi saa palautusarvokseen viitteen String-olioon, joka saadaan käytetyn StringBuffer -olion toString-metodilla.

`String extractTables(String query)`

Metodi on yksityinen apumetodi, jota doPost -metodi käyttää saadakseen esimerkkikyselystä merkkijonon, joka kertoo kyselyssä käytetyt taulut. Merkkijono koostetaan käyttämällä Tab-

leNameParser-luokan staattista metodia getTableNames ja katenoimalla tuloksena saadun Vector -olion sisältämät String -oliot yhteen, pilkulla erotettuina. Metodi palauttaa viitteen String -olioon, joka listasta syntyy.

`doPost ja doGet`

Metodi käsittelee lomakkeen, jolla annetaan tehtävänannon tiedot. Jos lähetyspyynnön viesti on "Poista tehtävä", pyyntö ohjataan uudelleen OTaskDef -servletin delTask -metodille. Muussa tapauksessa tulostetaan lomake, jolla tehtävänannon voi hyväksyä tai suorittaa uudestaan.

Lomaketta varten metodi laskee muut pstaskiin tarvittavat arvot kuin tarkistussummasarake ja itse tarkistussumma. Ylläpitotehtävät käsitellään eri tavalla kuin kyselytehtävät, eli niissä käytetään Update -luokkaa, jotta saataisiin tulostaulu, josta laskea arvot. Kyselytehtävän esimerkkikysely suoritetaan kahdesti, ensin ilman distinct -määreitä, jolloin tuloksesta lasketaan nonunique -kentän arvo. Kyselyn suorittamisessa mahdollisesti aiheutuvat poikkeukset käsitellään tulostamalla virheilmoitus sivulle. Jos tässä vaiheessa tehtävänumero tai nimi on jätetty tyhjäksi, servlet kutsuu itseään uudestaan.

Lomakkeen alkuun tulostetaan kursorisilmukalla taulukko, jossa on esimerkkikyselyn vastaus (tai ylläpito-operaation tapauksessa taulu muutosten jälkeen). Sarakkeiden otsikkojen kohdalle tulostetaan valintanapit, joista voi valita, mistä sarakkeesta tarkistussumma lasketaan. Silmukassa lasketaan myös vaadittujen rivien ja sarakkeiden määrä, sekä muodostetaan merkkijono, joka sisältää vaadittujen sarakkeiden listan pilkulla eroteltuna.

Laskettu data tulostetaan sekä käyttäjän näkyville, että piilokentäksi lomakkeeseen. Lisäksi lomakkeelle tulostetaan piilokentäksi nimeltä old tieto siitä ollaanko muokkaamassa jo olemassa olevaa tehtävää vai luomassa uutta. Tätä tietoa tarvitaan jatkossa. Arvo 'ohyes' tarkoittaa, että käsiteltävästä tehtävästä on jo kannassa versio.

Lopuksi metodi luo tehtävän tiedoista TaskForm -olion, joka tulostetaan lomakkeelle ennen lähetyspainikkeita. Lähetyspainikkeita tulostetaan kaksi, toinen hyväksymistä, toinen uudelleensuorittamista varten.

Luokka:

`OTaskStore`

Luokka päivittää kantaan tauluun pstask lomakkeelta saamansa tehtävän tiedot, tai sitten lisää tauluun kokonaan uuden tehtävänannon. Lisäksi se voi suorittaa esimerkkikyselyn uudestaan ohjaamalla sen takaisin servletille OTaskReg.

Metodit:

`int calcCheckVal (ResultSet rs, String cn)`

Metodi on yksityinen apuväline tarkistussumman laskemiseen. Se käyttää Checksum -oliota tarkistussumman laskemiseen parametrina saamastaan ResultSet -oliosta, käyttäen tarkistussa-rakkeena saraketta, jonka nimi vastaa String -oliona saatua sarakkeennimeä. Metodi palauttaa kokonaisluvun, joka on parametrina saadun ResultSetin tarkistussumma laskettuna sarakkeesta, jonka nimi annettiin parametrina.

`doPost ja doGet`

Jos lähetyspyyntö on "Suorita esimerkkikysely uudestaan", metodi ohjaa pyynnön takaisin servletin OTaskReg käsiteltäväksi. Muuten metodi muuttaa merkkijonoina saamansa lukutiedon kokonaisluvuiksi ja laskee calcCheckVal -metodin avulla esimerkkikyselyn tarkistussumman parametrina saadusta sarakkeesta. Jos kyseessä on ylläpitotehtävä, tarkistussumma lasketaan Update -olion tuottamasta tulostaulusta (joka on ylläpito-operaation käsittelemä taulu operaation jälkeen). Mahdollisista poikkeuksista tulostetaan virheilmoitus.

Tarkistussumman laskemisen jälkeen tehtävä joko lisätään uutena tauluun pstask, tai sitten tau-lun tehtävänantoa muokataan, jos tehtävä ei ole uusi. Lopuksi tulostetaan tieto päivityksen on-nistumisesta, ja nappi, jolla pääsee takaisin OListPagen tuottamaan tehtävälistaan.

Luokka:

`TaskForm`

Luokka vastaa tehtävänantolomaketta. Sitä käyttävät apunaan servletit OTaskDef ja OTaskReg.

Konstruktorit:

Luokalla on kaksi konstruktoria. Parametriton konstruktori asettaa tehtävälomakkeen 'nollatilaan'. Parametrillinen konstruktori vaatii parametrikseen sarjan arvoja, joilla täyttää lomakkeen kentät.

Metodit:

`print(Printwriter out, Connection con)`

Metodi tulostaa tehtävänantolomakkeen olion kenttien arvojen mukaisesti parametrikseen saamalla PrintWriter -oliolla. Mitään muuta erikoista metodissa ei olekaan, paitsi että aihepiirit tulostetaan select -listana lomakkeelle kursorisilmukassa, vastausjoukosta kyselyyn, jolla haetaan aihepiirinumerot ja aihepiirin otsikot taulusta pstopic.

Lomakkeella käytetään tehtävän tyypistä koodikirjaimia S (SQL-kysely), U (SQL-ylläpitotehtävä) ja R (relaatioalgebrakysely).

5.3 Tehtävienantotyökalun testaus

Testausmenetelmänä on käytetty 'valistunutta' toimintotestausta, eli toiminnoista on testattu vastaavatko ne määrittelyjään, kuitenkin pohjaten tietoon järjestelmän toteutuksesta, ja sitä kautta kenties löytyvistä ongelmakohdista. Tehtävänantojen antamista on täten siis testattu täyttämällä lomakkeita ja katsomalla Oraclen tietokannasta suoraan kyselyillä, tulevatko lomakkeille annetut tiedot kantaan oikein.

Tässä yhteydessä varoitetaan, että tehtävänantotyökalun määrittely on kehittynyt sekä määrittely- että suunnitteluvaiheessa esitetyistä.

Testattuja toimintoja ovat siis tehtävänannon antaminen, muokkaaminen ja kannasta poistaminen. Testausta ei tässä dokumentointiteknisistä syistä kuvata taulukkona.

5.3.1 Tehtävän antaminen

Oikeellinen tehtävänanto:

```
Tehtävänumero:            10000
Nimi suomeksi:            Testi1
Nimi engl.:               Test1
Teksti suomeksi:          Testi 1: listaa opettajat.
Teksti engl. :            Test2: list the teachers.
Mallivastaus(SQL-kysely): select distinct * from teacher
Järjestetty:              E
Samat (A=B,C=D):          (tyhjä kenttä)
Aihepiiri: SQL:n          perusteet
Tehtävän tyyppi:          SQL-kyselytehtävä
Järjestysperuste:         (tyhjä kenttä)
Turhat taulut:            (tyhjä kenttä)
```

Testin tarkoituksena oli selvittää, onnistuuko oikeanmuotoista tehtävänantoa tallettaa kantaan. Esimerkkikysely tarkistettiin suorittamalla sama kysely itse oraclen kannassa. Tarkistussumma laskettiin kentästä tfname. Tehtävä tallentui kantaan odotetusti ja oikein.

Puutteellisin tiedoin täytettyä lomaketta testattiin jättämällä yksitellen edellä mainitusta lomakkeesta kenttiä tyhjiksi. Vastaavasti kenttiin arvojen laittamista testattiin lisäämällä yllä tyhjiin kenttiin arvot. Muuten lomake täytettiin kuten yllä. Järjestelmän odotettiin korvaavan tyhjät kentät tyhjillä merkkijonoilla tai tehtävänumerolla 0. Jos tauluun yritetään laittaa tehtävää, jolla ei ole tehtävänumeroa tai suomenkielestä otsikkoa, järjestelmä palaa odotetusti takaisin lomakkeelle, tallentamatta mitään kantaan. Muut kohdat voi jättää täyttämättä, jolloin tehtävän tarkis-

tuksessa ei ole käytettävissä kaikkia tietoja, ja esim. tarkistussarakkeen tapauksessa tehtävää ei voi järjestelmän mukaan ratkaista.

On huomattava, että tehtävänumero nolla (0) on alkuperäisessä järjestelmässä uuden tehtävän merkki, joten jos sen antaa, tehtävää ei voi editoida.

5.3.2 Tehtävän muokkaaminen

Tehtävän muokkaamista kokeiltiin tekemällä muutoksia edellä mainittuun testitehtävään. Koska pohjajärjestelmä on miltei täysin samanlainen, testattiin pitkälti samoja asioita ja samalla strategialla kuin tehtävän luomista testatessa. Tarkoitus oli katsoa, että tehtävänannon muokkaaminen nimenomaan muuttaa tehtävänantoa, eikä riko, poista tai kopioi tehtävää.

Jos tehtävän numeroa yritti muuttaa, muutosta kantaan ei tässä tapauksessa tehty, kuten oli tarkoituskin. Ilmoitus 'päivitys suoritettu' saattaa olla täten hieman hämmentävä.

5.3.3 Tehtävän poistaminen kannasta

Tarkoituksena oli tutkia, toimiiko tehtävän poistaminen kannasta. Testaus suoritettiin käyttämällä poista -nappia aiemmissa testeissä käytettyjen roskatehtävänantojen poistoon. Tehtävänannon poistuminen tarkistettiin katsomalla kannasta suoraan kyselyllä. Poistaminen toimi, eikä siitä jää rippeitä kantaan.

Jos tehtävännumeron paikalle laittaa vääriä lukuja ennen poistamista, järjestelmä yrittää poistaa tehtävännumeroa vastaavaa tehtävänantoa kannasta. Tämän kanssa kannattaa olla varovainen, sillä järjestelmä ei kysy mitään poistosta, ja jos painaa poista -nappia muutettuaan tehtävän numeroa, saattaa tulla poistaneeksi jonkin toisen tehtävän, jota ei ollut katsomassa.

5.3.4 Aihepiirin lisääminen

Järjestelmällä ei voi lisätä kantaan aihepiirejä, mutta jos kannan taulua pstopic muutetaan, tämän pitäisi näkyä järjestelmässäkin. Tätä on testattu lisäämällä kantaan aihepiirien tiedot, ja katsomalla tehtävänantolomakkeen aihepiirilistasta, että kaikki aihepiirit esiintyvät siellä. Testauksessa havaittiin, että järjestelmä hakee aihepiirien tiedot kannasta oikein.

5.4 Tehtävänantotyökalun parannusehdotukset

Koko järjestelmä tulisi ohjelmoida uusiksi olioperiaatteiden mukaisesti. Nyt on pyritty rakentamaan jo olemassa olleen järjestelmän päälle, jolloin oliorakenteen muodostaminen olisi ollut hankalaa. Oliorakenne olisi kuitenkin modulaarinen, helpommin paranneltava ja ylläpidettävä.

Muuten parannusehdotukset liittyvät pitkälti käytettävyyteen. Nyt järjestelmä vaatii käyttäjältä hieman liikaa kurinalaisuutta, muuten tehtävänannot jäävät puutteellisiksi.

Tehtävänannon poistamisesta pitäisi ehkä kysyä 'Oletko varma?' -tyyppinen kysymys, tai sitten se pitäisi voida perua. Edellinen on tällä lomakepohjaisella järjestelmällä helpommin toteutettavissa.

Tehtävänannossa on kenttiä, jotka ovat opiskelijan vastauksen tarkistamiselle välttämättömiä, kuten ok_checkable ja ok_checkvalue. Tehtävänantolomakkeessa voi kuitenkin tässä versiossa jättää näihin tyhjät arvot aivan liian helposti. Tästä tulisi varoittaa käyttäjää.

Käsin annettavat listat, kuten 'Samat' -kenttä ja turhien taulujen lista, on annettava varsin tarkalleen oikeassa muodossa, sillä opiskelijan vastauksen tarkistustyökalut eivät ymmärrä edes ylimääräisiä välilyöntejä. Tämä tulisi korjata, tai ainakin varoittaa käyttäjää, jos hän on tekemässä jotain väärin.

On esitetty toive, että kun muokataan vanhaa tehtävää, saataisiin suoraan OtaskReg -servletin tuottama muoto lomakkeesta tauluun talletettujen tietojen perusteella.

Ylläpitotehtäville ei lasketa rivien määrää ylläpito-operaation osakyselystä, josta on jäänyt pois distinct -määre. Tämä on monimutkainen ja kenties Tietokantojen perusteet -kurssin tehtävien puitteissa hieman tarpeetonkin korjata.

Joistain virhetilanteista on palattava selaimen back -painikkeella. Normaalissa käytössä näihin ei juurikaan törmää, virheellisen datan testauksessa tällaisia käyttötapauksia tuli ilmi.

6 Järjestelmän ylläpito ja asentaminen (GE)

Luokka HttpDBServletOra ja sen metodi printRelHeader(PrintWriter out) sisältävät merkkijonomuuttujat pathToServlets, pathToPictures ja pathPicsRemove, jotka pitää päivittää kun järjestelmä asennetaan uuteen ympäristöön. Kaksi ensimmäistä sisältävät polkuasetukset servleteille ja kuville. Kolmanteen tulee sama polku kuin kuville, mutta kaikki / -merkit pitää korvata merkkijonolla \\/ (keno, keno, kautta). pathToServlets muutujan voi jättää tyhjäksi, jos kaikki servletit ovat samassa paikassa.

Ooldinfo, OstatusInfo, OupdateAnswerInfo, RelTaskPageMenu ja TaskPageMenu sisältävät kovakoodattuna servlettikutsuja. Jos kaikki servletit ovat samassa paikassa näihin ei tarvitse koskea.

6.1 Asennusohje

Tässä kappaleessa käsitellään järjestelmän asentamista. Tämä ohje kattaa vain Apachen jserv-servletajurin, eikä ota kantaa muihin servletajureihin.

Järjestelmän asentamiseksi tarvitaan servletajuri, JDBC -ajuri (tietokannan käsittelyä varten), DBConnectionBroaker ja paketti sqltr.tar. Koska järjestelmä on toteutettu java -servletteinä, asennuksessa myös tarvitaan JDK:n (java development kit) vähintään 1.2.2-versiota.

Servletajurin asentamisen jälkeen määritellään servletalue esim. sqltr (tätä nimeä käytetään myöhemmin viittaamaan kyseiseen servletalueeseen). Sqltr -servletalueen asetuksiin määritellään hakemisto, josta kyseisen alueen tiedostot (luokat ja pakkaukset) löytyvät. Tiedostoon kirjoitetaan esimerkiksi: repositries=/home/local/sqltr/servlet/repositry/. Tämä kertoo servletajurille, että kyseisen servletalueen tiedostot löytyvät hakemistosta */home/local/sqltr/servlet/repositry/*. Jos näitä hakemistoja on enemmän kuin yksi, ne annetaan listassa pilkulla eroteltuina.

HttpDBServletOra -luokkassa alustetaan servletin muissa luokissa käytettävät muuttujat. Servletajurille pitää kertoa, että kyseistä luokkaa täytyy käsitellä, kun servletlueeseen viitataan en-

simmäisen kerran. Seuraava rivi ilmoittaa servletajurille kyseisestä ominaisuudesta: servlets.startup=harkka.HttpDBServletOra;, ja se kirjoitetaan servletialueen asetuksiin .

Lopuksi puretaan pakkaus sqltr.tar sqltr-servletalueen servleteille tarkoitettuun hakemistoon ja muokataan tiedostoa dbconfig vastaamaan haluttuja tietokanta-asetuksia. Lisäksi luodaan jserv-servletajurin käynnistysskripti, jolla ajuri käynnistetään. Käynnistysskriptissä määritellään tarvittavien pakkausten ja luokkien sijainti (classpath).

Liitteenä on esimerkit yllämainituista tiedostoista.

6.1.1 Servlet -alueen asetukset

```
##############################################################################
#
#                                          Servlet  Zone  Configuration  File
#
##############################################################################
#

################################     W     A     R     N     I     N     G
################################
#
# Unlike normal Java properties, JServ configurations have some important
# extentions:
#
#    1) commas are used as token separators
#    2) multiple definitions of the same key are concatenated in a
#       comma-separated list.
#
##############################################################################
#

# List of Repositories
########################

# The list of servlet repositories controlled by this servlet zone
# Syntax: repositories=[repository],[repository]...
# Default: NONE
repositories=/home/local/sqltr/servlet/repository/

# Classloader parameters
##########################

# Enable servlet class autoreloading.
# Syntax: autoreload.classes=[true,false] (boolean)
# Default: true
autoreload.classes=true

# Enable servlet resourced autoreloading (properties and other loaded re-
sources)
# Syntax: autoreload.file=[true,false] (boolean)
# Default: true
autoreload.file=true

# Set the number of millisecond to wait before giving up on initializing a
servlet.
# (a timeout of zero means no timeout)
# Syntax: init.timeout=(long)>0
# Default: 10000 (10 secs)
init.timeout=10000
```

```
# Set the number of millisecond to wait before giving up on destroying a
servlet.
# (a timeout of zero means no timeout)
# Syntax: destroy.timeout=(long)>0
# Default: 10000 (10 secs)
destroy.timeout=10000

# Set whether or not to use cookies to maintain session state.
# If false, then response.encodeUrl() will always be the method
# to maintain session state. If true, then the servlet engine will
# attempt to set a cookie when request.getSession(true) is called.
# Syntax: session.useCookies=[true,false] (boolean)
# Default: true
session.useCookies=true

# Set the number of millisecond to wait before invalidating an unused session.
# Syntax: session.timeout=(long)>0
# Default: 1800000 (30 mins)
session.timeout=1800000

# Set how frequently (milliseconds) to check for timed-out sessions.
# Syntax: session.checkFrequency=(long)>0
# Default: 30000 (30 secs)
session.checkFrequency=30000

# SingleThreadModel Servlets parameters
########################################

# Set the initial capacity of the STM servlets pool.
# Syntax: singleThreadModelServlet.initialCapacity=(int)>1
# Default: 5
singleThreadModelServlet.initialCapacity=5

# Set the number of servlet instances should be added to the pool if found
empty.
# Syntax: singleThreadModelServlet.incrementCapacity=(int)>1
# Default: 5
singleThreadModelServlet.incrementCapacity=5

# Set the maximum capacity of the STM pool
# Syntax: singleThreadModelServlet.maximumCapacity=(int)>1
# Default: 10
singleThreadModelServlet.maximumCapacity=10

################## S E R V L E T       P A R A M E T E R S
######################

################################## N O T E
####################################
# When "classname" is specified, it means a Java dot-formatter full class name
# without the ".class". For example, a class with source file named
# "Dummy.java" with a package name "org.fool" is defined as "org.fool.Dummy".
#
# Since each servlet may have lots of private initialization data, Apache
JServ
# allows you to store those servlet initArgs in a separate file. To do this,
# simply do not set any initArgs in this file: Apache JServ will then look for
# a file named "[servlet classname].initargs" in the same directory of that
# class. Note that this may work with even class archives.
#############################################################################
#

# Startup Servlets
####################

# Comma or space delimited list of servlets to launch on startup.
# This can either be a class name or alias.
# Syntax: servlets.startup=[classname or alias],[classname or alias],...
# Default: NONE
# servlets.startup=hello,snoop,org.fool.Dummy
servlets.startup=harkka.HttpDBServletOra;

# Servlet Aliases
```

```
##################

# This defines aliases from which servlets can be invoked.
# Each alias give a new instance of the servlet. This means that if a servlet
# is invoked both by class name and by alias name, it will result in _TWO_
# instances of the servlet being created.
# Syntax: servlet.[alias].code=[classname] (String)
# Default: NONE
# servlet.snoop.code=SnoopServlet
# servlet.hello.code=org.fool.Dummy

# Global Init Parameters
#########################

# Parameters passed here are given to each of servlets. You should put
# configuration information that is common to all servlets.
#
# The value of the property is a comma delimited list of "name=value" pairs
# that are accessible to the servlet via the method getInitParameter()
# in ServletConfig.
# Syntax: servlets.default.initArgs=[name]=[value],[name]=[value],...
# Default: NONE
# servlets.default.initArgs=common.to.everybody=Hi everybody!

# Servlet Init Parameters
##########################

# These properties define init parameters for each servlet that is invoked
# by its classname.
# Syntax: servlet.[classname].initArgs=[name]=[value],[name]=[value],...
# Default: NONE
# servlet.org.fool.Dummy.initArgs=message=I'm a dummy servlet

# Aliased Servlet Init Parameters
##################################

# These properties define init parameters for each servlet that is invoked
# by its alias.
# Syntax: servlet.[alias].initArgs=[name]=[value],[name]=[value],...
# Default: NONE
# servlet.snoop.initArgs=message=I'm a snoop servlet
# servlet.hello.initArgs=message=I say hello world to everyone
```

6.1.2 Startjserv-skripti

```
#!/bin/sh
# Launch jserv in manual mode.

SERVLET_DIR=/home/local/sqltr/servlet # servletalueen juurihakemisto

jsdk=/home/www/java/jsdk.jar #javn tarvittavat pakkaukset
jserv=/home/www/java/ApacheJServ.jar # Apachen omat pakkaukset
oracle=/home/local/sqltr/servlet/oracle/ # jdbc-ajurit
props="$SERVLET_DIR/jserv.properties" # servletajurin asetukset
log="$SERVLET_DIR/jserv.log"  # jdbc:n käyttämä loki
classes="$CLASSPATH:$jsdk:$jserv:$oracle"
java -classpath "$classes" org.apache.jserv.JServ "$props" &> "$log"
```

6.1.3 Dbconfig-tiedosto

```
dbDriver      oracle.jdbc.driver.OracleDriver
dbServer      jdbc:oracle:thin:@kontti.helsinki.fi:1522:tkta
dbLogin       harkka
dbPassword    info
minConns      1
maxConns      10
logFileString //home/local/sqltr/servlet/log
```

```
maxConnTime   .1
```

LIITE 1. SQL-Trainer tietokannan luonti-SQL

```
/*==============================================================*/
/* Database name:  ORACLE Version 8                             */
/* DBMS name:      ORACLE Version 8                             */
/* Created on:     27.7.2000 12:30:58                           */
/* Edited by:      Kimmo Sinkko 28.8. 16:04:00                  */
/*==============================================================*/


drop table psstate cascade constraints;

drop table psreltemp cascade constraints;

drop table psterm cascade constraints;

drop table psanswer cascade constraints;

drop table pstask cascade constraints;

drop table pstopic cascade constraints;

drop table pscourse cascade constraints;

drop table psstudent cascade constraints;


/*==============================================================*/
/* Table : psstudent                                            */
/*==============================================================*/

create table psstudent (
   sid                  VARCHAR(10)                    not null,
   lname                VARCHAR(48)                    not null,
   fname                VARCHAR(48),
   last_visit           DATE                           not null,
   HETU                 VARCHAR(11),
   SS                   VARCHAR(12),
   stnumber             VARCHAR(20),
   constraint PK_PSSTUDENT primary key (sid)
);


/*==============================================================*/
/* Table : pscourse                                             */
/*==============================================================*/

create table pscourse (
   year                 INTEGER                        not null,
   term                 CHAR                           not null,
   startdate            DATE,
   enddate              DATE,
   constraint PK_PSCOURSE primary key (year, term)
);


/*==============================================================*/
/* Table : pstopic                                              */
/*==============================================================*/

create table pstopic (
   tno                  INTEGER                        not null,
   ttitle_fi            VARCHAR(60),
   ttitle_en            VARCHAR(60),
   tasks                INTEGER                        not null,
   credits_upto         DATE                           not null,
   constraint PK_PSTOPIC primary key (tno)
);


/*==============================================================*/
```

```
/* Table : pstask                                                */
/*==============================================================*/

create table pstask (
   task_id              INTEGER                         not null,
   title_fi             VARCHAR(60)                     not null,
   title_en             VARCHAR(60),
   task_text_fi         VARCHAR(500),
   task_text_en         VARCHAR(500),
   ok_rows              INTEGER,
   nonunique            INTEGER,
   ok_cols              INTEGER,
   ok_checkable         VARCHAR(32),
   ok_checkvalue        INTEGER,
   ordered              CHAR,
   neededcols           VARCHAR(120),
   eqpairs              VARCHAR(120),
   tno                  INTEGER,
   included             VARCHAR(120),
   col_max              INTEGER,
   needed_tabs          VARCHAR(120),
   needless_tabs        VARCHAR(120),
   task_type            CHAR,
   task_answer          VARCHAR(500),
   constraint PK_PSTASK primary key (task_id),
   constraint FK_PSTASK_REF_44_PSTOPIC foreign key (tno)
         references pstopic (tno)
/*         on delete restrict*/
);


/*==============================================================*/
/* Table : psanswer                                             */
/*==============================================================*/

create table psanswer (
   sid                  VARCHAR(10)                     not null,
   task_id              INTEGER                         not null,
   trycnt               INTEGER                         not null,
   answ                 VARCHAR(1000),
   trydate              DATE,
   response             INTEGER,
   extra                VARCHAR(500),
   constraint PK_PSANSWER primary key (sid, task_id, trycnt),
   constraint FK_PSANSWER_REF_55_PSTASK foreign key (task_id)
         references pstask (task_id)
/*         on delete restrict*/,
   constraint FK_PSANSWER_REF_58_PSSTUDEN foreign key (sid)
         references psstudent (sid)
/*         on delete restrict*/
)
storage
(
initial 5M
next 3M
pctincrease 5
);


/*==============================================================*/
/* Table : psterm                                               */
/*==============================================================*/

create table psterm (
   year                 INTEGER                         not null,
   term                 CHAR                            not null,
   tno                  INTEGER                         not null,
   constraint PK_PSTERM primary key (year, term, tno),
   constraint FK_PSTERM_REF_71_PSCOURSE foreign key (year, term)
         references pscourse (year, term)
/*         on delete restrict*/,
   constraint FK_PSTERM_REF_76_PSTOPIC foreign key (tno)
         references pstopic (tno)
```

```
/*         on delete restrict*/
);


/*==============================================================*/
/* Table : psreltemp                                            */
/*==============================================================*/

create table psreltemp (
   sid                  VARCHAR(10)                     not null,
   task_id              INTEGER                                 not null,
   trycount                  integer                            not null,
   relcount                  integer                            not null,
   reldate              DATE,
   sqlhtmrel            VARCHAR(1000),
   sqlrel               VARCHAR(1000),
   constraint PK_PSRELTEMP primary key (sid, task_id, trycount,relcount),
   constraint FK_PSRELTEM_REFERENCE_PSSTUDEN foreign key (sid)
         references psstudent (sid)
/*         on delete cascade*/,
   constraint FK_PSRELTEM_REFERENCE_PSTASK foreign key (task_id)
         references pstask (task_id)
/*         on delete cascade*/
);


/*==============================================================*/
/* Table : psstate                                              */
/*==============================================================*/

create table psstate (
   sid                  VARCHAR(10)                     not null,
   task_id              INTEGER                                  not null,
   lasttry              INTEGER,
   state                INTEGER,
   constraint PK_PSSTATE primary key (sid, task_id),
   constraint FK_PSSTATE_REF_61_PSSTUDEN foreign key (sid)
         references psstudent (sid)
/*         on delete restrict*/,
   constraint FK_PSSTATE_REF_64_PSTASK foreign key (task_id)
         references pstask (task_id)
/*         on delete restrict*/
);
```