Genomgång av tentamen

Databasteknik ITHS 4/3 2016

Tentans upplägg

Täcka in de 10 delar av kursen vi bedömt som viktigast

Säkerställa att det funnits såväl teoripass som övningspass kring varje del

Lätt kunna generera en lista på ingående delar som stämmer överens med tentan

10 frågor à 5 poäng ger statistik för skola, lärare och student över vilka moment som var svåra och behöver fördjupas

Vilka var de svåraste frågorna?

Fråga 9 - JDBC var den fråga som hade lägst genomsnittspoäng och även flest 0-poängssvar (den fråga som flest fått totalt 0 poäng på)

Samtidigt sa ett par på utvärderingen att JDBC var mest givande och användbart

Varför var det så lågt resultat på JDBC-frågorna?

Teori: De kräver en hel del förståelse för Java och systemutveckling

Vilka frågor var lättast?

Fråga 2 om SQLite hade högst genomsnittspoäng

Fråga 6 INSERT hade flest fullpoängssvar

Hur gick det på tentan?

8 VG

6 G

3 U

Max 50 poäng. Klassens genomsnittspoäng: 36.8 poäng (nära VG)

Vad tyckte ni? Var det för lätt eller för svårt? Var det relevant?

Fråga 1a Databas/Databashanterare

Beskriv skillnaden mellan en databas och en databashanterare där databashanterare används för att beskriva en kategori programvaror.

Databas: Data strukturerat i till exempel tabeller, kolumner och rader - sen samling relaterad data - en abstrakt benämning

Databashanterare (DBMS) - programvara som hanterar databaser och låter oss hämta, lagra och manipulera data i databaser

Fråga 1b Databas/Databashanterare

Beskriv hur begreppen databas, tabell, rad och kolumn förhåller sig till varandra.

Databas: En benämning på en eller flera tabeller med data

Tabell: En samling rader med relaterad data beskriven i kolumner

Kolumn: Beskrivning av en egenskap hos data i en rad/entitet. Kallas också attribut eller fält.

Rad: En entitet eller instans av data. Kallas också *record*.

En tabell av bilar kan beskriva attributen hos en bil (kolumnerna) och ha en mängd bilar beskrivna i rader där varje rad (helst!) motsvarar en unik bil i det att den har en unik kombination av värden på attributen (kolumnerna).

Fråga 1c - Databas/Databashanterare

Ge en motivering till att vi använder databaser och databashanterare

- Separera data från applikation
- Centralisering av datahantering/-lagring
- Databashanterare är bättre på att hantera data än vi

Fråga 1d - Databas/Databashanterare

Vilket språk har vi använt för att kommunicera med den databashanterare vi använt i kursen?

SQL - Structured Query Language

Fel svar:
Bash - är skalet/kommandotolken
SQLite - är databashanteraren
Java - är ett programmeringsspråk i största allmänhet
JDBC - är ett API för Java för databasåtkomst

Fråga 2 a - SQLite-databashanteraren

*Nämn två sätt att starta SQLite **interaktivt** och välja en existerande databas*

1. `$ sqlite3 “sökväg_till_databasfil”`
2. ```
   $ sqlite3
   sqlite> .open “sökväg_till_databasfil”
   ```
   ```

Fråga 2b - SQLite-databashanteraren

Förklara skillnaden på de instruktioner vi givit utifrån dessa två typer av instruktioner:

I Sådana som inleds med en punkt (och saknar semikolon)

II Sådana som avslutas med semikolon

I: SQLite-specifika kommandon för databashantering mm

II: SQL-satser (Create,Select,Insert,Delete,Update osv)

Fråga 2c - SQLite-databashanteraren

Hur kan man få SQLite att skriva resultatet till en fil i stället för standard out?

1. sqlite> .output “filnamn” -- alla efterföljande resultat skrivs till filen
2. sqlite> .once “filnamn” -- nästa resultat skrivs till filen (engångs)
3. Kör sqlite-kommandon från bash och omdirigera stdout till fil (fusk ;-))

Fråga 2d - SQLite-databashanteraren

Vad händer om man ger SQLite argumentet av en databas som inte existerar när man startar det interaktiva skalet?

Den skapar en ny fil för databasen om den behöver.

Fråga 2e - SQLite-databashanteraren

Förklara instruktionerna .tables och .schema

.tables - listar tabeller i vald/aktuell databas

.schema - listar tabelldefinitioner i vald/aktuell databas

Fråga 3a - SQL - SELECT mm

När vi hämtar data från en databas, vilken funktion har den så kallade WHERE-klausulen (konstruktionen som inleds med ordet WHERE)?

WHERE-klausulen inkluderar ett villkor som måste vara uppfyllt för att rader ska inkluderas i resultatet. Det specificerar egenskaper för de rader vi vill hämta.

Fråga 3b - SQL - SELECT mm

Vilket specialtecken använder vi för att symbolisera (“alla kolumner”)?

* (stjärna eller asterisk)

Fråga 3c - SQL - SELECT mm

Studera denna tabell (och tänk dig att det finns ett stort antal rader inlagda):

```
CREATE TABLE exam_results(student_name text, score integer);
```

Skriv en sats som listar student_name och score för alla studenter

```
SELECT * FROM exam_results;
```

Skriv en sats som listar student_name för alla studenter som har en score som överstiger 24 poäng

```
SELECT student_name FROM exam_results WHERE score > 24;
```

*Skriv en sats som listar (student_name och score) de 5 bästa studenterna (med avseende på score) med den bästa studenten först och så **fallande** poäng*
(Hint: ni måste både sortera och begränsa antalet till 5)

```
SELECT * FROM exam_results ORDER BY score DESC LIMIT 5;
```

Fråga 4a - UPDATE

Vad innebär det att utföra en update-sats som saknar en så kallad WHERE-klausul?

Det innebär att vi inte har något urval eller kriterium för vilka rader som ska uppdateras. Med andra ord så uppdateras samtliga rader i tabellen.

Fråga 4b - UPDATE

Utifrån tabellen i 3c och skriv en sats som ändrar en rad där student_name nu är ‘Plato’ så att raden ändrar namnet till ‘Pluto’ och score till 45. Du kan utgå från att det är unika namn på studenterna. Du behöver därför inte bry dig om vilken score ‘Plato’ hade innan ändringen. Tanken är att rätta ett fel där Pluto blivit felstavad till Plato och dessutom hade score blivit fel. Nu ska du rätta raden till student_name Pluto och score 45.

```
UPDATE exam_results SET student_name='Pluto', score=45

  WHERE student_name='Plato';
```

Fråga 4c - UPDATE

Om man är osäker på att man har rätt urval för en update som är tänkt att bara ändra en rad i databasen, vad kan man göra för att försäkra sig om att urvalet är rätt?

Man kan utföra en SELECT med exakt samma WHERE-klausul och säkerställa att den rad man var ute efter är den enda som listas.

Fråga 5a - DELETE (5p)

*Resonera kring varför man inte kan använda tecknet * före FROM i samband med en delete-sats (i databashanteraren SQLite är det inte tillåtet).*

En delete-sats syftar till att radera en eller flera rader i en tabell. Eftersom syftet är att radera hela rader så är det meningslöst att dessutom använda * som ju betyder “alla kolumner”. En hel rad innehåller ju alla kolumner.

Fråga 5b - DELETE (5p)

Vad är det för skillnad mellan att använda DROP TABLE och DELETE FROM (där delete-satsen saknar where-klausul)?

DROP TABLE: Radera en hel tabell och alla dess rader med data

DELETE FROM: Ta bort en eller flera rader med data från en tabell men behåll tabellen och ej berörda rader med data.

Fråga 5c - DELETE (5p)

Formulera en delete-sats som utifrån tabellen i 3c tar bort alla rader med studenter som har en score som understiger 20 poäng.

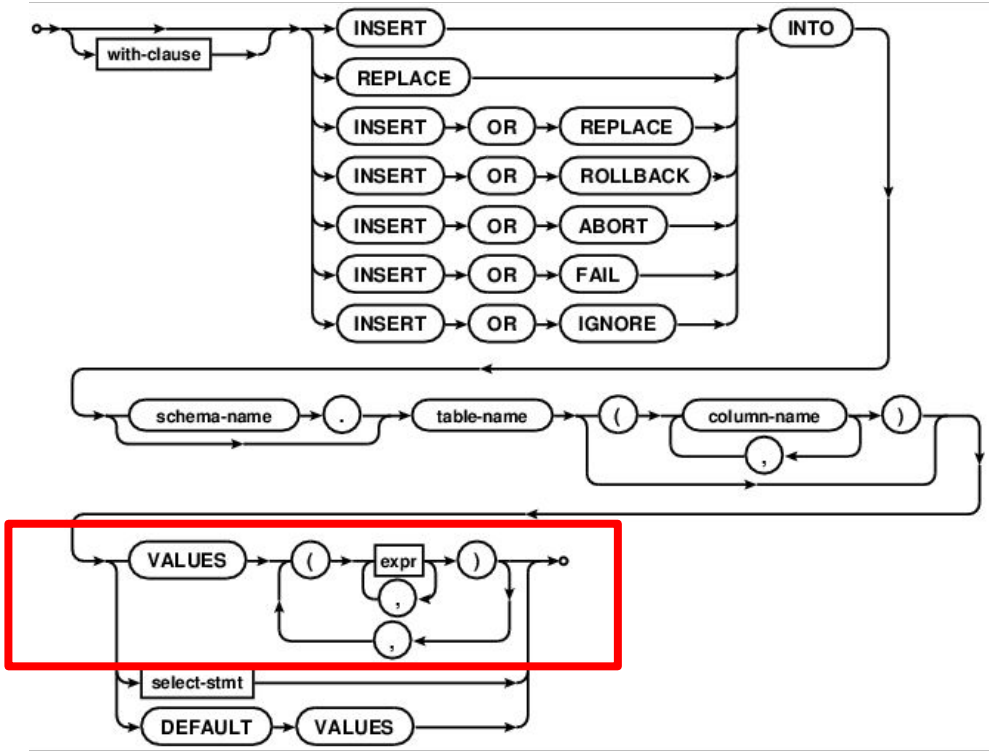
```
DELETE FROM exam_results WHERE score<20;
```

Fråga 6a - INSERT

Om vi fortsätter att betrakta tabellen från 3c, vad tror du följande sats resulterar i?

```
INSERT INTO exam_results
 VALUES ('Hewey', 29), ('Dewey', 47), ('Louie', 30);
```

Tre nya studenter med poäng sätts in i tabellen. Det går alltså att lista rader grupperat med parenteser och skiljt med kommatecken. Se schemat för INSERT.


with-clause
INSERT
REPLACE
INSERT OR REPLACE
INSERT OR ROLLBACK
INSERT OR ABORT
INSERT OR FAIL
INSERT OR IGNORE
INTO
schema-name
.
table-name
(
column-name
,
)
VALUES
(
expr
,
)
,
select-stmt
DEFAULT
VALUES

Fråga 6b - INSERT

Varför är det inte relevant (eller möjligt) att ha med en så kallad WHERE-klausul på toppnivå i en INSERT-sats?

INSERT sätter in nya rader i tabellen. WHERE används för att sätta kriterier på existerande data vid SELECT, UPDATE och DELETE.

Det är inte relevant att jämföra något när helt nya rader helt sonika ska sättas in i tabellen.

Fråga 6c - INSERT

Vi vill lägga in en student med namn ‘Gyro’ i tabellen från 3c. Score ska sättas till 50. Fortsätt följande påbörjade sats och se till att den avslutas korrekt:

```
INSERT INTO exam_results(score, student_name)
```

(Tänk på hur kolumn-listan relaterar till värde-listan)

```
... VALUES(50, 'Gyro');
```

Eftersom kolumnerna listas i ordningen score, sedan student_name måste values-klausulens värdelista ha samma ordning.

Fråga 7a - Normalisering mm

Nämn minst två skäl till att bryta upp en tabell i flera mindre tabeller (som kopplas samman med hjälp av någon kolumn som representerar samma sak i några av tabellerna).

Lagringseffektivitet, dataintegritet och skalbarhet.

Fråga 7b - Normalisering mm

Vi vill göra om tabellen i 3c så att det går att ha studenter med samma namn och i stället skilja på dem med hjälp av ett personnummer. Vi vill ha information om studenternas namn och personnummer i en tabell och en tabell med information om poäng på provet (score). Vi väljer följande design:

```
CREATE TABLE students(student_id integer primary key,
                      name text, id_number text);
CREATE TABLE "exam_results"(student_id integer, score integer);
```

*Skriv en sats som skriver ut name och score för varje student, sorterat på score (**högsta först och fallande**). Tänk på att kolumnen med namnet på studenten nu heter name.*

```
SELECT name,score FROM students NATURAL JOIN exam_results
 ORDER BY score DESC;
-- OR:
SELECT s.name, e.score FROM students s, exam_results e
 WHERE s.student_id=e.student_id ORDER BY score  DESC;
```

Fråga 8a - Constraints och modifiers

Tabellen students i uppgift 7b hade modifier primary key och typen integer för kolumnen student_id. Vilken constraint får man på köpet för kolumner som är primary key (åtminstone i de flesta databashanterare)?

UNIQUE constraint - en primärnyckel måste kräva unika värden

Fråga 8b - Constraints och modifiers

En primary key som är av typen integer (i SQLite) får en speciell egenskap vid INSERT som underlättar för oss när vi lägger in nya rader. Vilken?

Om vi sätter in rader utan att ange kolumnen som är INTEGER och PRIMARY KEY, så räknar SQLite ut ett nästa heltalsvärde åt oss och sätter det på tabellen. Det liknar konceptet “auto increment” som finns i många databashanterare.

Detta låter oss fokusera på de övriga kolumnerna och låta databashanteraren sköta värdet på sådana kolumner som typiskt är interna IDn.

Fråga 8c - Constraints och modifiers

Om vi vill förhindra att kolumnen id_number (som ska motsvara ett svenskt personnummer) får in två identiska personnummer i två rader, vilken constraint (begränsningsvillkor) kan vi använda för att forcera att bara unika värden tillåts för raderna i tabellen?

UNIQUE - detta är ett begränsningsvillkor för att kräva unika värden genom hela tabellen (inga två rader får ha samma värde på en sådan kolumn).

Fråga 8d - Constraints och modifiers

Om vi bara vill tillåta textsträngar för personnummer som har formen “NNNNNN-NNNN”, det vill säga sex stycken heltal, ett bindestreck och fyra heltal (t ex “200202-1111”), hur kan vi använda funktionerna CHECK och GLOB för att skapa ett constraint? Svara med ett helt CHECK-uttryck.

(Vi struntar här i att uttrycket kommer tillåta konstiga personnummer så som 009988-0000 - där månad och dag uppenbarligen är helt åt skogen liksom kontrollsiffran osv)

```
check (id_number glob '[0-9][0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')

-- OR

check ( glob ('[0-9][0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]', id_number) )
```

Fråga 9a - JDBC

Vad används metoden Class.forName(String) ofta till i ett jdbc-projekt?*

Till att ladda drivrutinens JDBC implementation (en klass som registrerar sig hos java.sql.DriverManager).

Class.forName(String) används för att få en referens till class-objektet för en namngiven klass. En sidoeffekt som man är ute efter här är att om klassen inte är laddad så läses den in i JVM innan referensen returneras.

Fråga 9b - JDBC

Vad returnerar metoden executeQuery från interface:et Statement?

Den returnerar en referens till ett ResultSet.

Signatur:

```
public abstract java.sql.ResultSet executeQuery(java.lang.String)
```

Fråga 9c - JDBC

Vad står förkortningen JDBC för?

JavaDataBaseConnectivity

Fråga 9d - JDBC

Beskriv i stora drag de steg som behövs för att köra en SQL SELECT via jdbc.
Utgå ifrån att SELECT-satsen är det första som sker i programmets main-metod. Tänk på vilka saker som måste ske och i vilken ordning. Vilka klasser behöver finnas/ha laddats? Vilka objekt behövs? Vilka metoder? Du behöver inte komma ihåg exakt vad metoderna heter men ge beskrivande namn.

* Drivrutinen måste ha laddats och därmed registrerat sig hos DriverManager (`Class.forName(...)`)
* Det måste finnas en Connection till databasen (`DriverManager.getConnection(...)`)
* Det måste finnas ett Statement som man fått från Connection (`con.createStatement()`)
* Nu kan man anropa `stm.executeQuery(...)` och få ett ResultSet tillbaka

Fråga 10a - SQL-injections

Beskriv vad “sql injection” är och hur det fungerar

SQL injection är när man utnyttjar en svaghet i en databaskopplad applikation genom att infoga (inject) SQL-syntax som en del av indata där applikationen (felaktigt) antar att det är ett konstant värde som infogas. När applikationen (felaktigt) bygger upp en SQL-sats så kommer det i stället för ett konstant värde i satsen finnas ny SQL som utför något som applikationen inte var tänkt att utföra. Ett exempel är en applikation som förväntar sig ett ID i heltalsform för att skapa ett WHERE-villkor på formen “id = N” (där N kommer från indata). Istället för endast ett heltal, skickar en angripare SQL-syntax, exempelvis om användaren skulle mata in 0 men i stället matar in:

```
0 OR 1=1
```

Tänkt resultat: `select name from table where id=0;` -- endast rad med id=0 returneras

Verkligt resultat: `select name from table where id=0 OR 1=1;` -- alla rader returneras

Fråga 10b - SQL-injections

Om vi har följande (mycket förenklade) metod:

```
public int updateScore(String id, int newScore){
  String sql="UPDATE exam_results SET score=" +
             newScore +
             " WHERE student_id=" + id;
  Statement stm=null;
  if(hasConnection()){ // assume this method exist!
    try{
      stm=con.createStatement();
      return stm.executeUpdate(sql);
    }catch(SQLException e){
      System.err.println("executeUpdate: "+e.getMessage());
    }finally{
      closeIt(stm); // assume this method exists and works!
    }
  }
  return -1;
}
```

Vad blir effekten av att metoden anropas med första argumentet som en String-referens som representerar texten:

"1 OR 1=1;--"

och andra argument som en int med värdet 0 (noll)?

Fråga 10b - SQL-injections

Relevanta avsnitt i metoden:

```
public int updateScore(String id, int newScore){
  String sql="UPDATE exam_results SET score=" +
             newScore +
             " WHERE student_id=" + id;
...
      stm=con.createStatement();
      return stm.executeUpdate(sql);
```

första argumente (id)t: `"1 OR 1=1;--"`
andra argument (newScore): en int med värdet 0

SQL-strängen blir:

```
UPDATE exam_results SET score=0 WHERE student_id=1 OR 1=1;--
```

Resultat: Alla rader ändras till att ha score = 0 och metoden returnerar antal rader som uppdaterats.

Fråga 10c - SQL-injections

Ett säkrare sätt att använda String-referenser som argument till en metod som använder referenserna som en del av en SQL-sats är att använda interface:et PreparedStatement. Hur fungerar PreparedStatement i stora drag?

Ett PreparedStatement låter oss använda parametrar i en SQL-sats. Man anger ? (frågetecken) på de platser som ska “parametriseras”. En fördel med PreparedStatements är att de skickas till databasen som kompilerar dem så att de kan återanvändas med nya värden för paramterarna utan att de behöver kompileras igen. En annan fördel är att parametrarna är säkrade mot SQL-injections genom att alla specialtecken i en parameter “escape:as” bort (ersätts med escape-sekvenser som gör att tecknen tolkas ordagrannt och inte som en del av språket SQL).

```
PreparedStatement updateScore  =
  con.prepareStatement("UPDATE exam_results SET score=? WHERE student_id=?");
updateScore.setInt(1, newScore);
updateScore.setInt(2, id);
updateScore.executeUpdate();
// Assumes both newScore and id are int-values!
```