Sample exam questions

Database exam TIG058

Distribution of topics covered

1. Grundläggande om Databaser och Databashanterare (5p)
2. SQLite-databashanteraren (5p)
3. SQL - SELECT, ORDER BY, WHERE, LIMIT (5p)
4. SQL - UPDATE (5p)
5. SQL - DELETE (5p)
6. SQL - INSERT (5p)
7. SQL - Normalisering, kopplingar mellan tabeller (JOIN) (5p)
8. SQL - Constraints och modifiers (5p)
9. JDBC basics (5p)
10. JDBC other (5p)

Databaser och Databashanterare

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

Databaser och Databashanterare

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

En databas är en strukturerad samling data där data om en sak beskrivs i en tabell (det kan finnas många tabeller).

En tabell beskriver en sak och består av en definition (ett schema) som beskriver vilka beståndsdelar och dessas typ en sak består av, samt data i rader med data för respektive kolumn.

En kolumn har ett namn och en typ för en beståndsdel av den sak som beskrivs i tabellen.

SQLite-databashanteraren

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

- Sådana som inleds med en punkt (och saknar semikolon)
- Sådana som avslutas med semikolon

SQLite-databashanteraren

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

- Sådana som inleds med en punkt (och saknar semikolon)
  - Detta är instruktioner specifika för SQLite, t ex .schema .tables .headers on
- Sådana som avslutas med semikolon
  - Detta är satser i språket SQL, ett standardiserat sätt att
    - Skapa tabeller
    - Hämta data från tabeller
    - Modifiera, radera eller sätta in data i tabeller

SQL - SELECT, ORDER BY, WHERE, LIMIT

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
- Skriv en sats som listar student_name för alla studenter som har en score som är minst 25 poäng
- 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)

SQL - SELECT, ORDER BY, WHERE, LIMIT

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;

-- Eller

SELECT student_name, score FROM exam_results;
```

SQL - SELECT, ORDER BY, WHERE, LIMIT

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 för alla studenter som har en score som är minst 25 poäng

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

-- eller

SELECT student_name FROM exam_results WHERE score >= 25;
```

SQL - SELECT, ORDER BY, WHERE, LIMIT

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) 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;
```

SQL - UPDATE

Utgå från tabellen exam_results i förra frågan och skriv en sats som ändrar en rad där student_name nu är ‘Plato’ så att raden ändrar namnet till ‘Pluto’ och score till 40. 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 40.

SQL - UPDATE

Utgå från tabellen exam_results i förra frågan och skriv en sats som ändrar en rad där student_name nu är ‘Plato’ så att raden ändrar namnet till ‘Pluto’ och score till 40. 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 40.

```
UPDATE exam_results SET student_name='Pluto', score=40
       WHERE student_name='Plato';
```

SQL - DELETE

Resonera kring varför man inte kan använda tecknet * före FROM i samband med en delete-sats (i databashanteraren SQLite är det inte tillåtet). T ex så är detta fel syntax:

```
DELETE * FROM students;
```

SQL - DELETE

Resonera kring varför man inte kan använda tecknet * före FROM i samband med en delete-sats (i databashanteraren SQLite är det inte tillåtet). T ex så är detta fel syntax:

```
DELETE * FROM students;
```

Hint: Tänk på vad DELETE får för effekt - *VAD* är det DELETE tar bort?

SQL - DELETE

Resonera kring varför man inte kan använda tecknet * före FROM i samband med en delete-sats (i databashanteraren SQLite är det inte tillåtet). T ex så är detta fel syntax:

```
DELETE * FROM students;
```

Hint: Tänk på vad DELETE får för effekt - *VAD* är det DELETE tar bort?

DELETE opererar på *hela rader* så det är oklart vad * skulle betyda.

* i en SELECT betyder *alla kolumner* men det är underförstått om man tar bort en hel rad!

SQL - INSERT

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

SQL - INSERT

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

Hint: Tänk efter vad INSERT har för effekt på en tabell

SQL - INSERT

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

Hint: Tänk efter vad INSERT har för effekt på en tabell

Effekten av INSERT är att nya rader med nya data (som kanske inte finns i tabellen) sätts in i tabellen.

Vad skulle då WHERE betyda? WHERE är ett urvalskriterium som opererar på *befintliga rader* t ex i SELECT, UPDATE och DELETE.

SQL - Normalisering, kopplingar mellan tabeller

Vi vill göra om tabellen `exam_results` 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) i en annan tabell, `exam_results`. 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.

SQL - Normalisering, kopplingar mellan tabeller

```
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; -- eller
SELECT name, score FROM students JOIN exam_results  ON
  students.student_id = exam_results.student_id ORDER BY score DESC; -- eller
SELECT s.name, er.score FROM students s, exam_results er
  WHERE s.student_id = er.student_id ORDER BY score DESC;
```

SQL - 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). Kolumnen heter id_number.

SQL - 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). Kolumnen heter id_number.

```
--(glob som infix operator):
CHECK(id_number GLOB '[0-9][0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]')
--(glob som funktion):
CHECK(GLOB ('[0-9][0-9][0-9][0-9][0-9][0-9]-[0-9][0-9][0-9][0-9]',
id_number))
```

JDBC Basics

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.

JDBC Basics

Beskriv i stora drag de steg som behövs för att köra en SQL SELECT via jdbc.

- Hämta och lagra en `Connection` från `DriverManager`
- Skapa ett `Statement` från `Connection`-objektet
- Be `Statement` utföra en query (ge SQL som en `String` som argument till metoden `executeQuery(sql)` - spara resultatet i en referens till ett `ResultSet`
- Om du vill hämta rader och kolumner, loopa igenom ditt `ResultSet` med `while (rs.next())` och hämta ut data från kolumnerna i varje varv med metoden `getString(columnName) t ex:`

```
while (rs.next()) {
  System.out.println(rs.getString("name"));
}
```

JDBC Other

Använd en PreparedStatement för att köra SQL-frågan SELECT * from students WHERE id = (där programmet fyller i värdet på id!).

Ni har en Java-int som heter id som har korrekt värde. Ni ska använda denna int för att sätta värdet på det som kommer efter likhetstecknet i SQL-saten ovan.

```
Connection con = DriverManager.getConnection("jdbc:sqlite:exam.db");
PreparedStatement ps =
  con.prepareStatement("SELECT * FROM students WHERE student_id=?");
int id = 3;
ps.setInt(1, id);
ResultSet rs = ps.executeQuery();
rs.next();
System.out.println(rs.getString("name")); // for instance!
```