Uppsala universitet
Institutionen för informationsteknologi
Kjell Orsborn, Tore Risch

Tentamen 2004-12-20
DATABASTEKNIK - 1DL116, 1MB025

```
Datum .................................Måndagen den 20 December, 2004
Tid ..................................................14:00-19:00
Jourhavande lärare ...Kjell Orsborn, tel. 471 11 54 eller 070 425 06 91

Hjälpmedel ..............................................miniräknare
```

Anvisningar:

- Läs igenom hela skrivningen och notera eventuella oklarheter innan du börjar lösa uppgifterna. Förutom anvisningarna på skrivningsomslaget så gäller följande:
  - Skriv tydligt och klart. Lösningar som inte går att läsa kan naturligtvis inte ge några poäng och oklara formuleringar kan dessutom misstolkas.
  - Antaganden utöver de som står i uppgiften måste anges. Gjorda antaganden får förstås inte förändra den givna uppgiften.
  - Skriv endast på en sida av papperet och använd ett nytt papper för varje uppgift för att underlätta rättning och minska risken för missförstånd.
- För godkänt krävs det cirka 50% av maxpoäng.

1. **Database terminology:** 3pts

Explain the following concepts in a database context.

(a) database schema

Answer: Meta data, or the database schema, include data about data, i.e. a description of the database stored in the system catalog. Meta-data consist of information about structure of files, type and storage format of each data item, various constraints on the data and other types of information about data such as authorization privileges and access statistics. For the relational model this include descriptions of the relation names, attribute names, data types, primary keys, secondary keys, foreign keys, other constraints, views, storage structures and indexes, and security and authorization information.

(b) aggregation (in data modeling)

Answer: Aggregation is an abstraction concept to group entities into composite objects from their components. In three cases can aggregation be related to the EER model. The 1st case is an aggregation of attribute values of an object to form the whole object. The 2nd case is the representation of an aggregation relationship using an ordinary relationship. The 3rd case is not explicitly supported in EER but involve the possibility to combine related objects using a particular relationship instance into a higher-level aggregate object.

(c) secondary index

Answer: Sekundärindex är en ordnad fil av dataposter med 2 fält där första fältet är av samma typ som som indexeringsfältet, dvs vilket fält som helst i datafilen. Andra fältet är en blockpekare. Indexeringsfältet kan vara ett icke-nyckelfält eller ett sekundärnyckelfält och datafilen ej sorterad efter indexeringsfältet. Sekundärindex kan vara glesa eller täta. Index ger en avsevärd effetivisering vid sökning av dataposter. Vid updatering av datafilen måste också tillhörande index uppdateras vilket medför en viss ökad kostnad för dessa operationer.

2. **Data models:** 4pts

Explain the three-schema architecture and how it supports two forms of data independence.

Answer: The three-schema architecture introduces a multi-level architecture where each level represents one abstraction level (in 1978 the ÒstandardÓ architecture (ANSI/SPARC architecture) for databases was introduced). It consists of 3 levels where each level introduces one abstraction layer and has a schema that describes how representations should be mapped to the next lower abstraction level:

1) The internal level or internal schema - describes storage structures and access paths for the physical database. Abstraction level: files, index files etc. Is usually defined through the data definition language (DDL) of the DBMS.

2) Conceptual level or conceptual schema - an abstract description of the physical database. Constitute one, for all users, common basic model of the logical content of the database. This abstraction level corresponds to Òthe real worldÓ: object, characteristics, relationships between objects etc. The schema is created in the DDL according to a specific data model.

3) External level, external schemas, or views - a typical DB has several users with varying needs, demands, access privileges etc. External schemas describes different views of the conceptual database with respect to what different user groups would like to/are allowed to se. Some DBMSÕs have a specific language for view definitions (else the DDL is used).

Physical data independence: the possibility to change the internal schema without influencing the conceptual schema. E.g. the effects of a physical reorganization of the database, such as adding an access path, is eliminated.

Logical data independence: the possibility to change the conceptual schema without influencing the external schemas (views). E.g. add another field to a conceptual schema.

3. **SQL:** 3pts

Suppose that we in a database have two relations (tables) with the following schemas:

```
LINE(LID,P1ID,P2ID)
POINT(PID,X,Y)
```

(a) Formulate a relational algebra expression that retrieves all points in the first quadrant, i.e. both coordinates greater than zero. (1pt)

(b) Formulate an SQL query that retrieves the length of each line (hint: there is a SQL function `SQRT(`*value expression*`)` that calculates the square root) (2pts)

Answer:

$\pi_{<PID>}(\sigma_{X \geqslant 0.0 \texttt{ AND } Y \geqslant 0.0}(POINT))$

```
SELECT L.LID, SQRT(((P2.X - P1.X)*(P2.X - P1.X)) +
                   ((P2.Y - P1.Y)*(P2.Y - P1.Y))) AS LENGTH
FROM LINE L, POINT P1, POINT P2
WHERE P1.PID = L.P1ID AND
      P2.PID = L.P2ID;
```

4. **Normalization:** 2pts

(a) Explain functional dependency (1pt)

(b) Explain full functional dependency (1pt)

Answer:
Ett funktionellt beroende så att X bestämmer Y, X - Y, existerar då om för varje par av tupler $t_1, t_2 \in r(R)$ och för alla $r(R)$ följande gäller: om $t_1[X] = t_2[X]$ så gäller att $t_1[Y] = t_2[Y]$
Fullt funktionellt beroende anger att för ett funktionellt beroende gäller att det inte finns någon delmängd attribut $A \subseteq X$ så att $(X - \{A\})Y$.
Här gäller att R är ett relationsschema och $r(R)$ är en instans av schemat R med attributen $A_1, ..., A_n$ och $X, Y \subseteq \{A_1, ..., A_n\}$.
Alltså ett fullt funktionellt beroende är ett funktionellt beroende som inte innehåller något onödigt attribut i determinanten (vänsterledet i beroendet).

5. **Transactions management:** 4pts

Explain, preferably using a picture, (hint: state transition diagram), the different states that a transaction goes through from the point where it starts

until it is finished (the difference between a transaction carried out fully and if it is instead aborted should be included).

Answer: See for instance Elmasri/Navathe 4ed., Figure 17.4) En transaktion går in i ett aktivt tillstånd (eng. active state) direkt den startar där den kan utfärda läs- och skrivoperationer.

När transaktionen avslutas övergår den i ett partiellt överlämnat tillstånd (eng. partially comitted state). I detta tillstånd behöver man kontrollera att inget kan förhindra att transaktionen permanentas (vissa transaktionsprotokoll möjliggör detta genom att att förändringar av databasen registreras i loggen).

Om allt är OK så har transaktionen nått sin överlämningspunkt (eng. commit point) och kan övergå i ett överlämnat tillstånd (eng. comitted state).

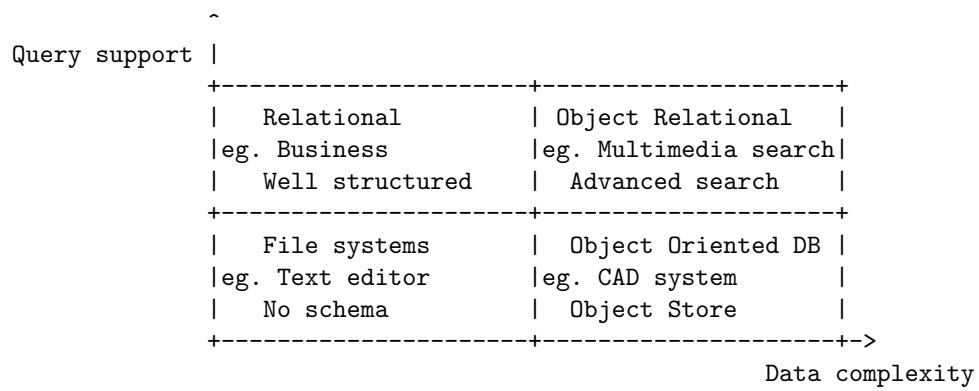
En transaktion kan också övergå i ett falerat tillstånd (eng. failed state) från sitt partiella överlämnade tillstånd eller från sitt aktiva tillstånd om någon kontroll gått fel eller om transaktionen avbrutits. En falerad transaktion kan behöva rullas tillbaka för att eleminera dess effekt.

överlämnade och falerade transaktioner lämnar slutligen systemet genom att övergå i ett avslutat tillstånd (eng. terminated state).

6. **Object-Oriented Databases:** 4pts

What are the important differences between File systems, Relational Databases, Object-Oriented Databases, and Object-Relational Databases and for what kind of applications are each kind suitable? Draw diagram.

Answer:



7. **Query Optimization:** 4pts

(a) What are the three most important join algorithms called? (1pt)

(b) Outline their functioning through pseudo code. (3pts)

Answer:

7a) Nested loop join, sort-merge join, hash join

7b)

```
Nested loop join:
{
   Stream T, U, R;
   Tuple t, u;

   open(R,'o');
   open(T,'i');
   while (not(eof(T)))
   {
     t = next(T);
     open(U,'i');
     while (not(eof(U))
     {
       u = next(U);
       if(t.K == u.K) emit(t + u, R);
     }
     close(U);
   }
   close(T);
   close(R);
}

Sort-merge join:
{
  Stream T, U, R;
  Tuple t, u;
  open(R,'o'); // open result stream R
  open(T,'i'); // open 1st input stream
  open(U,'i'); // open 2nd input stream
  t = next(T); // read 1st row to t
  u = next(U); // the same for u
  while (not(eof(T)) and not(eof(U))) // as long as both input
                                      // streams have more data
  {
    if(t.k > u.k) u = next(U); // if the key field less in u then
                               // move U forward
    else if(t.k < u.k) t = next(T); // the same for T
    else emit(t + u, R); // send t and u concatenated as
                         // next result row
  }
  close(T);
  close(U);
  close(R); // close all streams
}

Hash join:
{
   Stream T, U, R;
   Tuple t, u;
   HashTable h;

   open(R,'o');
   h = createHashTable();
   open(T, 'i');
   while (not(eof(T)))
   {
     t = next(T);
     putHash(h,t.k,t);
   }
```

```
    open(U, 'i');
    while (not(eof(U))
      {
        u = next(U);
        tm = getHash(ht, u.k);
        if(tm != NULL) emit(tm + u, R);
      }
    close(U);
    close(T);
    close(R);
}
```

8. **Active Databases:** 4pts

(a) What are the components of ECA in a relational database trigger? (3pts)

(b) Why is it often difficult to implement complex integrity constraints with triggers? (1pt)

Answer:

8a)

Event: An insert, update, or delete operation

Condition: A query over the database

Action: A set of database update operations.

8b) It is difficult to identify all possible event situations that might violate the constraint.

9. **Multi-Media Databases:** 4pts

(a) What is feature extraction and how is it used for querying multi-media databases? (2pts)

(b) What kind of database API primitives are needed for BLOBs? (2pts)

Answer:

9a) Feature extraction is the process of extracting features from multi-media objects before they are inserted into the database. A feature can be, e.g., color spectrum of an image, sharpness, lightness, etc. The features are stored in the database together with the multi-media object. When searching the database the features are extracted also from searched multi-media objects and used for finding similar multi-media objects stored in the database.

9b) Since a BLOB can be very large we cannot read it into the application's memory. Instead there must be a streamed interface with operations such as:

```
open(blob)
read(blob,pos)
write(blob,pos,len,section)
delete(blob,pos,len)
close(blob)
```

Good luck and a Merry, Merry Christmas!

/ Kjell och Tore