Uppsala universitet
Institutionen för informationsteknologi
Kjell Orsborn

Tentamen 2003-08-22
DATABASTEKNIK - 1DL116

Datum ....Fredagen den 22 Augusti, 2003
Tid 8:00-13:00
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. Datamodeller - tre-schema-arkitekturen: 4p

   I databassammanhang är tre-schema-arkitekturen ett välkänt sätt att beskriva en databas i flera nivåer och datamodeller. Förklara innebörden av var och en av dessa tre nivåer samt förklara vilka fördelaktiga egenskaper man kan uppnå med denna indelning.

   Svar:

   The three-schema architecture introduces a multi-level architecture where each level represents one abstraction level (in 1978 the "standard" 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 "the real world": 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 DBMS's 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.

2. Fysisk databasdesign - indexering: 4p

   (a) förklara begreppen primärindex och sekundärindex deras uppbyggnad, användning och egenskaper.

   Svar:

   Primärindex består av en ordnad fil av dataposter med 2 fält. Första fältet är av samma typ som ordningsfältet (indexeringsfältet) för datafilen och det andra fältet är en pekare till ett datablock (blockpekare). Primärindex är ett glest index då det har en indexpost för varje block i data-filen. Primärindex kräver mycket mindre plats än motsvarande datafil.

   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.

Sekventiell skanning med hjälp av primärindex är effektivt men med sekundärindex är det dyrbart då varje post adressering kan betyda att ett nytt datablock måste hämtas från disk. Seckundärindex tar normalt mer plats än ett primärindex och längre söktid än ett primärindex. Förbättringen av söktiden för en godtycklig datapost är dock högre för ett sekundärindex än för ett primärindex om man jämför med att inte ha tillgång till något index.

(b) beräkna hur många diskaccesser som behöver genomföras för att hitta ett specifikt sökvärde i ett B^+-träd som omfattar 500000 sökvärden och nodstorleken i trädet är 100 (en nod anses fylla ett diskblock).
Svar:
4 disk accesses corresponding to an access of a node at each level of the tree where the maximum depth of the tree is calculated from $\lceil \log_{\lceil n/2 \rceil} N \rceil$, where N is the number of search values and n is the size of the tree nodes.

3. Transaktionshantering: 4p

(a) Vad är en seriell transaktionsplan och varför är en sådan plan betraktad som korrekt?
Svar:
En seriell transaktionsplan består av en exekveringsplan för en mängd transaktioner där för alla transaktioner skall gälla att alla deloperationer för en transaktionen skall exekveras efter varann i en följd utan att någon operation från en annan transaktion exekveras emellan. Det betyder att endast en transaktion är aktiv vid varje tillfälle.

(b) Vad är en serialiserbar (eng. serializable) transaktionsplan? Förklara gärna med exempel.
Svar:
En transaktionsplan som består av ett antal transaktioner är serialiserbar om den kan transformeras till en motsarande seriell transaktionsplan för samma uppsättning transaktioner. (Ref kap 19.5 i vita Elmasri/Navathe - 3:e ed)

4. Återhämtning (eng. recovery): 4p

Beskriv kortfattat proceduren för återhämtning enligt modellen omedelbar uppdatering (eng. immediate update) i en fleranvändarversion och där ”cascading rollback”kan krävas.

Svar:

Recovery according to the immediate update model:

1. Start from the last record in the log file and traverse backwards until a check point is reached. Create two lists: C transactions that have reached their commit points NC transactions that have not reached their commit points.

2. Create a list R with transactions that has read an item updated by a transaction in NC. (OBS that this step must be applied recursively).

3. Start from the last record in the log file and apply the UNDO procedure to all (Write,T,...) where $T \in NC \cup R$.

4. Start from the check point and REDO all transactions (Write,T,...) such that $(T \in C) \wedge (T \notin R)$.

5. Restart all failured transactions.

5. Säkerhet och integritet: 4p

   (a) Hur specificeras aktorisering (authorization) i moderna relationsdatabaser? (1p)

   Svar: By GRANT/REVOKE SQL commands that associates access rights with tables and views.

   (b) Varför är vyer användbara för aktorisering? (1p)

   Svar : Views make it possible to specify access rights covering subsets of tables or combinations of tables. E.g. dynamic access rights can be specifies where the accessible data depends on row values in tables.

   (c) När kan en användare transferera aktoriseringsrättigheter (authorization rights) till en annan användare? (1p)

   Svar: A user can transfer access rights if those where granted to him using the annotation WITH GRANT OPTION.

   (d) Vad är en accessmatris (access matrix) och hur används den för databassäkerhet? (1p)

   Svar: The access matrix is a 2D system object S x O -¿ R stating what access rights R a subject S has for objkect O. Subjects are e.g. users, accounts, roles. Objects are e.g. tables, views. Rights are e.g. READ, WRITE, UPDATE.

6. Objekt-orienterade databaser: 4p

   (a) Vilka är de viktigaste skillnaderna mellan ett objekt-orienterat (object-oriented) och en objektrelationellt (object-relational) databassystem? (3p)

   Svar: An object-oriented DBMS extends an existing programming with the capability to persistently save objects on disk. The object-oriented DBMS acts as an object store. An object-relational DBMS extends a relational DBMS with capabilities to model and query objects, rather than just 3rd normal form tables. Object-oriented DBMS have more limited query capabilities than object-relational DBMS and is less suitable for high transaction frequencies or many users. Object-oriented DBMSs are on the other hand very efficient if data is to be accessed with few concurrent users procedurally through a conventional programming language. Object-relational DBMS often contain hooks to extend the query optimizer.

(b) Vad är 'swizzling' och till vad används det? (1p)

Svar: Swizzling is a technique used by object-oriented DBMSs to access transparently persistent objects as if they were main-memory references.

7. Frågeoptimering: 4p

(a) Vad kallas de tre viktigaste join algoritmerna? (1p)

Svar: Nested-loop join, Sort-merge join, and Hash join.

(b) Beskriv hur de fungerar m.h.a. pseudokod. (3p)

Svar: 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'); // öppna resultatström R
  open(T,'i'); // öppna 1:a inströmmen
  open(U,'i'); // öppna 2:a inströmmen
  t = next(T); // läs 1:a raden till t
  u = next(U); // detsamma för u
  while (not(eof(T)) and not(eof(U))) // så länge båda inströmmarna
                                      // har mer data
  {
    if(t.k > u.k) u = next(U); // om nyckelfältet mindre i u så
                               // flytta fram U
    else if(t.k < u.k) t = next(T); // detsamma för T
    else emit(t + u, R); // skicka  t och u konkatenerade som
                         // nästa resultatrad
  }
```

```
  close(T);
  close(U);
  close(R); // stäng alla strömmar
}
```

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. Relationskalkyl: 4p

Vi har två relationer:

```
EMPLOYEE(SSN, NAME, SALARY, DNO, ADDRESS, PHONE)
DEPARTMENT(DNO, DNAME, MGR, SECR, LOCATION)
```

och ställer SQL frågan:

```
SELECT S.NAME FROM EMPLOYEE E, DEPARTMENT D, EMPLOYEE S
WHERE E.NAME = 'Kalle Karlsson' AND D.DNO = E.DNO AND
D.SECR = S.SSN
```

(a) Översätt frågan till 'tuple relational calculus'. (2p)
Svar:

```
{s.name | employee(s) and
              exist s, d
             (e.name = "Kalle Andersson" and
              d.dno = e.dno and
              d.secr = s.ssn)}
```

(b) Översätt frågan till 'domain relational calculus'. (2p)
Svar:

```
{name | exists ssn, salary, dno, address, phone, dname, mgr, location
            (employee(ssn, name, salary, dno, address, phone) and
             department(dno, dname, mgr, ssn, location)}
```

alt. if _ means 'don't care what value is':

```
    {name | exists ssn, dno
            (employee(ssn, name, _, dno, _, _) and
             department(dno, _, _, ssn, _)}
```

Lycka till!

/ Kjell och Tore