Uppsala universitet
Institutionen för informationsteknologi
Kjell Orsborn, Tore Risch

Tentamen 2005-07-29
DATABASDESIGN FÖR INGENJÖRER - 1DL124

```
Datum .....................................Fredagen den 29 Juli, 2005
Tid .....................................................13:00-18:00
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. **Databasterminologi:** 4 p

Förklara följande databasbegrepp:

(a) meta data

Svar: 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) "dödlig låsning"(eng. deadlock)

Svar: Dödlig låsning (eng. dead lock) är en situation som kan uppstå när alla transaktioner i en mängd av två eller flera transaktioner väntar på att få accessrättinghet till någon dataartikel som är låst av någon annan av transaktionerna.

(c) fullt funktionellt beroende

Svar:

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).

(d) sekundärindex

Svar: 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. **Datamodeller:** 4 p

Förklara begreppen primärnyckel (eng. primary key) för relationsdatamodellen och objektidentifierare (eng. object identifier) i en objektdatamodell samt jämför deras viktigaste egenskaper.

Svar: En primärnyckel är en minimal supernyckel, utvald bland kandidatnycklarna att utgöra nyckel för en relation. En minimal supernyckel består av en minimal delmängd av relationens attribut som unikt identifierar alla tupler i relationen.

En objektidentifierare är en unik och ofta logisk och systemgenererad identifierare som används för att unikt identifiera objekt under hela dess existens, samt för att hantera referenser mellan objekt.

De huvudsakliga skillnaderna mellan en P.N. och en OID är att en P.N. unikt identifierar tupler (rader) i en tabell medan en OID unikt identifierar objekt i hela databasen. Vidare så är en OID alltid systemgenererad, ändrar aldrig värde

och används internt för att referera till objekt. En P.N. utgörs normalt av en eller flera attribut (kolumner) från en relation, de väljs av databaskonstruktören och kan följaktligen ändras eller korrigeras.

3. **Konceptuell modellering:** 4 p

Med utökad entitets-relations-modellering (eng. enhanced entity-relationship) kan man vid specifikationen av en specialisering eller generalisering (eng. specialization/generalization) definiera olika bivillkor (eng. constraints). Förklara i detta sammanhang kortfattat följande begrepp:

(a) disjointness constraint

(b) completeness constraint

Svar:

(a) disjointness constraint innebär att subklasserna till en superklass kan specificeras till att vara *disjunkta*, dvs att en instans kan endast vara medlem av en av dessa subklasser, eller *överlappande* (eng. overlapping), dvs disjunkhetsvillkoret krävs så att instanser tillåts vara medlemmar till fler än en av subklasserna.

(b) completeness constraint specificerar klassificeringen till att vara *total* eller *partiell*. Om klassificeringen är total så måste instanserna till superklasserna vara medlem i någon subklass och för en partiell klassificering behöver ej instanserna till superklassen vara medlem av någon subklass.

Notera att villkoren a), b) är ortogonala, dvs de är oberoende av varann så att alla kombinationer av dessa kan existera.

4. **Transaktionshantering:** 4 p

Beskriv de egenskaper som man ofta vill att transaktioner skall uppfylla i databassammanhang (ledning: ACID).

Svar:

To preserve the integrity of data, the DBMS must ensure ACID properties:

- Atomicity (atomic or indivisible): a logic processing unit (all operations of the transaction) is carried out in its whole or not at all.
- Consistency (preservation): a correct execution of a transaction in isolation should preserve the consistency of the database (from one consistent state to another).
- Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of of other concurrently executing transactions. The updates of a transaction shall be isolated from other transactions until after the commit point.
- Durability (or permanency): If a transaction completes successfully, the changes it has made to the database must persist and should not be lost in a later system failure.

5. **Dataintegritet:** 4 p

(a) Förklara inom relationsdatamodellen begreppet referensintegritet (referential integrity). (2p)

Svar: Referensintegritet kräver att om en tupel i en relation refererar till en annan relation så måste den referera till en existerande tupel.

(b) Hur kan man testa att referensintegritet upprätthålls för en databasuppdatering med UPDATE? (2p)

Svar: There are two cases:

Consider relationship set R between entity sets E_1 and E_2. The relational schema for R includes the primary keys K_1 of E_1 and K_2 of E_2. Then K_1 and K_2 form foreign keys on the relational schemas for E_1 and E_2 respectively.

Case 1: If a tuple t_2 is updated in relation r_2 and the update modifiesvalues for the foreign key α, then a test similar to the insert case is made. Let $t_2^{'}$ denote the new value of tuple t_2. The system must ensure that: $t_2^{'}[\alpha] \in \Pi_{K_1}(r_1)$

Case 2: If a tuple t_1 is updated in r_1, and the update modifies values for the primary key (K_1), then a test similar to the delete case is made. The system must compute $\sigma_{\alpha=t_1[K_1]}(r_2)$ using the old value of t_1 (the value before the update is applied). If this set is not empty, the update may be rejected as an error, or the update may be cascaded to the tuples in the set (cascading update), or the tuples in the set may be deleted (cascading delete).

6. **Objektdatabaser:** 4 p

(a) Beskriv vilka tre former av utbyggbarhet som objekt-relationella databashanterare kan tillhandahålla för att stödja avancerade tillämpningar. (3p)

(b) Vilka av ovanstående utvidgningsmekanismer saknas eller är svaga i enkla objekt-orienterade databashanterare (dvs. i s.k. "object stores")? (1p)

6a. Användardefinerade datatyper och operationer

Användardefinerade datarepresentationer och index

Användardefinerad frågeoptimering (säsom kostnadsmodeller och optimeringsalgoritmer)

6b. Det saknas stöd för användardefinerad frågeoptimering

Det saknas stöd för användardefinerade index

7. **Frågeoptimering:** 4 p

(a) Vad kallas de tre viktigaste join algoritmerna och hur mycket minne behöver de när de körs? (3p)

Svar:

Sort-merge join: I princip behövs endast minne för att hantera två poster samtidigt för jämförelse.

Nested-loop join: I princip behövs endast minne för att hantera två poster samtidigt för jämförelse.

I praktiken så läses flera poster in samtidigt från båda tabellerna i varsin buffert och dessutom finns en resultatbuffert. Alltså behövs minnesutrymme för tre minnesbuffertar.

Hash join: Här behövs minnesutrymme för en hashtabell över ena tabellen samt för att hantera en post för den andra tabellen. Även här hanteras i praktiken buffertar med flera poster samtidigt för data- och resultatsströmmar.

(b) Varför är det viktigt att använda 'prepare' i JDBC/ODBC? (1p)
Svar: Prepare förkompilerar en parametriserad fråga så att den vid senare (upprepad) exekvering ej behöver kompileras vilket eleminerar onödig frågeoptimering.

8. **Datalager:** 4 p

Ett konsult behöver analysera sin verksamhet och tänker därför utnyttja datalagerteknik. Man vill analysera uppdragens inkomster per kund och typ av uppdrag (t.ex. seminarium, design, implementering, testning).

(a) Hur ser datakuben ut som sammanfattar ovanstående? Ge exempel. (1p)
Svar:

```
Uppdrag \ Kund | abb | sca | s:a
---------------------------------
seminarium     | 100 |  90  | 190
design         |  75 |  80  | 155
implementering | 125 | 120  | 245
testning       | 50  |  65  | 115
---------------------------------
s:a            | 350 | 355  | 705
```

(b) Designa ett stjärnschema för att lagra datakuben i en relationsdatabas. Ge exempel på tabellinnehåll. (2p)
Svar:

```
Kund(kundid, namn, + andra egenskaper)
Uppdrag(uppdragid, namn, + andra egenskaper)
Kunsultinkomster(kundid, uppdragid, inkomst)
```

(c) Hur uttrycker man en fråga i SQL m.h.a. cube-operatorn för att konstruera datakuben från relationsdatabasen? Hur ser resultattabellen ut? (1p)
Svar:

```
select kundid, k.namn as knamn, uppdragid, u.namn as unamn,
       sum(inkomst)
from Kund k, Uppdrag u, Konsultinkomster ki
where k.kundid = ki.kundid and
      u.uppdragid = ki. uppdragid
group by  kundid, k.namn, uppdragid, u.namn
with cube;
```

resultattabell:

```
kundid | knamn | uppdragid | unamn          | inkomst
-------------------------------------------------------
k1     | abb   | u1        | seminarium     | 100
k1     | abb   | u2        | design         | 75
k1     | abb   | u3        | implementering | 125
k1     | abb   | u4        | testning       | 50
k2     | sca   | u1        | seminarium     | 90
k2     | sca   | u2        | design         | 80
k2     | sca   | u3        | implementering | 120
k2     | sca   | u4        | testning       | 65
null   | null  | u1        | seminarium     | 190
null   | null  | u2        | design         | 155
null   | null  | u3        | implementering | 245
```

```
null   | null  | u4        | testning       | 115
k1     | abb   | null      | null           | 350
k2     | sca   | null      | null           | 355
null   | null  | null      | null           | 705
-------------------------------------------------------
```

Lycka till och ha en solig sommar!

/ Kjell och Tore