# Omtentamen i DATABASER

**Svar:**

**Obs! Lärare-version, med lösningar**

**DAG:** lö, 20 nov 1999 **TID:** kl. 8.45 – 12.45 **SAL:** ML13 och ML14 (Maskinhuset)

| | |
|---|---|
| **Ansvarig:** | Martin Weichert |
| **Förfrågningar:** | Martin Weichert, tel. 772 10 68 |
| **Resultat:** | anslås den 9 dec 1999 |
| **Poängantal:** | sammanlagt maximalt 60 poäng. |
| **Betygsgränser:** | CTH: 3:a 24 p., 4:a 36 p., 5:a 48 p.<br>GU: Godkänd 28 p., Väl godkänd 48 p.<br>Doktorander: Godkänd 28 p. |
| **Hjälpmedel:** | utdraget ur *Oracle7 Server SQL Language* ("det gula pappret") resp. *Appendix B Syntax* ur referensmanualen för *Oracle SQL* samt *SQL*Plus Quick Reference*. |

**Observera:**

- Skriv tydligt och disponera pappret på ett lämpligt sätt.
- Börja varje uppgift på nytt blad. Skriv endast på en sida av pappret.
- Alla svar skall väl motiveras!
- Ange på tentan om du går på GU eller CTH. För CTH ange även vilken linje du går!

*Lycka till!*

**Uppgift 1.** (a) En transaktion i ett databassystem måste rullas tillbaka när den blir avbruten. Varför blir en transaktion avbruten? Nämn minst tre olika anledningar.

**Svar:**

- Användaren begär det (t.ex. skriver ABORT i sqlplus).
- Processen dör, eller hela datorn kraschar.
- Brott mot integritetsvillkor upptäcks.
- För att bryta upp deadlock.
- För att förhindra deadlock (vid vissa protokoll, t.ex. *wait-die*,...).
- Transaktionen läste *"dirty data"* skrivna av en annan transaktion som själv blev avbruten.

– 3p.

(b) För vad används en **loggfil**? Vilka informationer står i den?

**Svar:** Den används för återställning av ett konsistent tillstånd efter en systemkrasch (eller efter avbrutna transaktioner). Den innehåller följande operationer (alltid med namnet till den genomförande transaktionen): BEGIN TRANSACTION (inte nödvändigtvis), WRITE (alltid med det nya värdet; vid vissa protokoll även det gamla värdet), READ (vid vissa protokoll, för att avgöra om *cascading rollback* behövs), COMMIT, ABORT, CHECKPOINT (vid vissa protokoll). Absolut nödvändiga är WRITE och COMMIT. – 2p.

(c) Varför använder man **normalformer** vid databas-design? Nämn tre problem som kan uppstå i en relationsdatabas p.g.a. av att den inte är i normalform!

**Svar:** Normalformer hjälper att garantera vissa riktighetsvillkor, och dessutom att undvika redundans. I en onormaliserad databas, t.ex. *emp(ENo, EName, DNo, DName)* med $DNo \rightarrow DName$, kan följande problem uppstå:

- data som bryter mot funktionella beroenden ($DNo \rightarrow DName$),
- uppdateringsanomalier, t.ex. en ändring (t.ex. av *DName*) genomförs inte i alla motsvarande rader
- inläggningsanomalier, t.ex. vissa data (t.ex. en ny tuppel *(DNo, DName)*) kan inte matas in då de inte har ett värde för nyckelattributet
- borttagningsanomalier, t.ex. information om en viss *(DNo, DName)*) försvinner om sista *emp*-tuppel för dessa tas bort
- redundans: *DName* lagras (upprepas) onödigt många gånger
- NULL-värden

– **3p.**

8 poäng.

**Uppgift 2.** Givet är tabellerna $r(\underline{A}, B)$ och $s(\underline{B, C}, A)$ med följande innehåll:

$r$:

| $\underline{A}$ | $B$ |
|---|---|
| $a_1$ | $b_1$ |
| $a_2$ | $b_2$ |

$s$:

| $\underline{B}$ | $\underline{C}$ | $A$ |
|---|---|---|
| $b_2$ | $c_1$ | $a_2$ |
| $b_3$ | $c_2$ | $a_1$ |
| $b_2$ | $c_2$ | $a_2$ |

Beräkna resultatet på följande relationsalgebra-uttryck. I denna uppgift antar vi att det är NAMN som gäller när vi identifierar kolumnerna. **Obs!** Beakta att det alltid är **mängder** det handlar om!

(a) $\sigma_{C=c_2}(s)$

(b) $\pi_{AB}(s) - r$

(c) $r \times \pi_C(s)$

**Svar:** $\sigma_{C=c_2}(s)$

| $B$ | $C$ | $A$ |
|---|---|---|
| $b_3$ | $c_2$ | $a_1$ |
| $b_2$ | $c_2$ | $a_2$ |

$\pi_{AB}(s) - r$

| $A$ | $B$ |
|---|---|
| $a_1$ | $b_3$ |

$r \times \pi_C(s)$

| $A$ | $B$ | $C$ |
|---|---|---|
| $a_1$ | $b_1$ | $c_1$ |
| $a_1$ | $b_1$ | $c_2$ |
| $a_2$ | $b_2$ | $c_1$ |
| $a_2$ | $b_2$ | $c_2$ |

Poängavdrag om det finns dubletter.
– (a) **1p.** – (b) **2p.** – (c) **2p.**

5 poäng.

**Uppgift 3.** (a) Vi har en relation $r$ i vilken det funktionella beroendet $A, B \to C$ skall gälla (och $AB$ är **inte** en nyckel, d.v.s. det måste finnas minst ett attribut till). Av någon anledning har vi valt att **inte** dela upp relationen $r$, men vi vill ändå behålla en garanti att beroendet $A, B \to C$ uppfylls. För detta skall en sats

```
create assertion AB_ger_C check ( uttryck );
```

användas, där *uttryck* är ett villkor formulerat i SQL som ger värdet "sant" om relationen $r$ uppfyller beroendet $A, B \to C$ och "falskt" om så inte är fallet.

Ange detta *uttryck* i SQL.

**Svar:** Flera möjliga lösningar, t.ex.:

- *uttryck* = `not exists (select * from vy)`
  med en lämplig definition för *vy*, t.ex.:
- `not exists (select * from R group by A,B having count(distinct C) > 1 )`
- `not exists (select * from R X, R Y where X.A=Y.A, X.B=Y.B, X.C<>Y.C )`
- $|\pi_{ABC}(r)| = |\pi_{AB}(r)|$, d.v.s. *uttryck* =
  `(select count(distinct A,B,C) from R) = (select count(distinct A,B) from R)`
- $G_{\text{count(C)}}(r/(A,B)) = all\ 1$, d.v.s. *uttryck* =
  `1 = all (select count(distinct C) from R group by A,B )`

4 poäng.

(b) I den nyare SQL-92 kan man använda nyckelordet `join` i `select`-satser för samkörning av tabeller. Givet två tabeller $r(A, B)$ och $s(B, C)$ kan man i SQL-92 t.ex. skriva:

```
select *
  from R natural join S;
```

vilket motsvarar följande `select`-sats utan `join`, som ger samma resultat:

```
select A, R.B, C
  from R, S
 where R.B = S.B;
```

Med ordet `join` kan man även uttrycka operationen "outer join", t.ex.:

```
select *
  from R natural left outer join S;
```

Man kan uttrycka även denna operation i förra SQL, även om det blir något mera komplicerat. Skriv en motsvarande `select`-sats utan `join`!

**Svar:**

```
select A, R.B, C
  from R, S
 where R.B = S.B
 union
select A, B, NULL
  from R
 where B not in (select B from S);
```

4 poäng.

8 poäng

**Uppgift 4.** I en databas finns uppgifter om studenter som gör laborationen på en kurs. Databasen innehåller bl a följande tabeller:

- *student(Stud_Nr, Namn, Född, Grupp_Nr)*
  En lista över alla studenter, med ett nummer *Stud_Nr* som unikt identifierar varje student, studentens *Namn* och födelsedatum *Född*, samt *Grupp_Nr* av den labgrupp i vilken studenten deltar. *Grupp_Nr* är referens till *labgrupp*.
- *labgrupp(Grupp_Nr, Betyg)*
  Lista över alla labgrupper, med unikt *Grupp_Nr* och ett numeriskt värde *Betyg*, vilket är 0, 3, 4 eller 5.

Skriv SQL-satser för följande uppgifter:

(a) Skapa en vy *STUDENT_BETYG* över alla studenter med deras *Stud_Nr*, namn, labgruppnummer och betyg. (Du kan använda denna vy i de följande uppgifterna.)

**Svar:**

```
create view STUDENT_BETYG
 as
select STUD_NR,NAMN,STUDENT.GRUPP_NR,BETYG
  from STUDENT,LABGRUPP
 where STUDENT.GRUPP_NR = LABGRUPP.GRUPP_NR;
```

– **2p.**

(b) Lista innehållet av *STUDENT_BETYG* i gruppnummerordning, med namnen alfabetiskt sorterade i varje grupp.

**Svar:**

```
select *
  from STUDENT_BETYG
order by GRUPP_NR,NAMN;
```

– **2p.**

(c) Lista namn på alla studenter som har det högsta förekommande betyget.

**Svar:**

```
select NAMN
  from STUDENT_BETYG
 where BETYG = (select max(BETYG) from STUDENT_BETYG);
```

– **2p** för SELECT-sats med "BETYG=(nästlad SELECT-sats)" (rätt tolkat); men bara **1p** för SELECT-sats med "BETYG=5" (feltolkat),

(d) Ange genomsnittligt betyg över alla labgrupper.

**Svar:** `select AVG(BETYG) from LABGRUPP;` – **2p.**

Ingen "`group by`" och ingen vy behövs!

(e) Ange genomsnittligt betyg över alla studenter(!). (Observera att detta resultat kan skilja sej från det föregående eftersom det kan finnas olika många studenter i labgrupperna.)

**Svar:** `select AVG(BETYG) from STUDENT_BETYG;` – **2p.**

Ingen "`group by`" och ingen vy behövs!

(f) Skapa följande rättigheter:

- Den kursansvarige (med databas-användarnamn LÄRARE) ska ha rättigheten att lägga till och ta bort labgrupper. Den här rättigheten ska han/hon även kunna ge vidare till andra.
- Kursassistenten (med databas-användarnamn ASSISTENT) ska ha rättigheten att ändra labgruppernas betyg.
- Alla labgrupper (deras databas-användarnamn är lika med deras *GRUPP_NR*) ska ha rättigheten att se sina egna betyg.

**Svar:**

- `grant insert,delete on LABGRUPP to LÄRARE with grant option;`
- `grant update(BETYG) on LABGRUPP to ASSISTENT;`
- ```
  create view EGNA
   as
  select * from LABGRUPP
   where GRUPP_NR = USER;
  grant select on EGNA to public;
  ```

$2 + 1 + 2 =$ **5p.**

(g) Återta den rättighet som kursassistenten fick i uppgift (f).

**Svar:** `revoke update(BETYG) on LABGRUPP from ASSISTENT;`

eller: `revoke update on LABGRUPP from ASSISTENT;`

(Enligt *sqlplus* gäller egentligen bara den sista varianten: "`UPDATE may only be REVOKEd from the whole table, not by column`", men vi accepterar även den första varianten här.)

– **1p.**

16 poäng.
  ```

**Uppgift 5.** Givet relation $r(A, B, C, D, E)$ med funktionella beroenden

1. $AB \rightarrow C$
2. $EA \rightarrow B$
3. $C \rightarrow E$
4. $E \rightarrow DA$

(a) Bestäm alla nycklar till relationen $r$.

**Svar:** $\{A, B\}$, $\{C\}$ och $\{E\}$.

**3p.**

(b) Uppfyller $r$ Boyce-Codd-normalformen? Motivera!

**Svar:** Ja. Alla givna beroenden uppfyller BCNF, eftersom alla vänstersidor är övernycklar.

**2p.**

(c) Uppfyller $r$ tredje normalformen? Motivera!

**Svar:** Ja. BCNF medför 3NF.

**2p.**

(d) Vi vill skapa ovanstående tabell $r$ i SQL med:

```
create table R
( A char,
  B char,
  C char,
  D char,
  E char,
  ***
);
```

I stället för `***` ska det finnas rader som garanterar alla nyckelvillkor från uppgift (a). Hur ser dessa rader ut?

**Svar:**

```
primary key (A,B),
unique (C),
unique (E)
```

(eller också med $C$ eller $E$ som primärnyckel istället).

**3p.**

(e) När vi nu har lagt in dessa rader från uppgift (d), är det då möjligt att mata in data som bryter mot något av de funktionella beroendena ovan?

**Svar:** Nej (p.g.a. att vi har BCNF).

**1p.**

11 poäng.

**Uppgift 6.** Europaparlamentet har 626 ledamöter, som representerar 15 länder, kommer från ett hundratal nationella partier, fördelas i 8 parlamentsgrupper, och är verksamma i 17 olika parlamentsutskott.

För att hålla reda på parlamentets verksamhet skall en databas användas. Följande informationer ska finnas med:

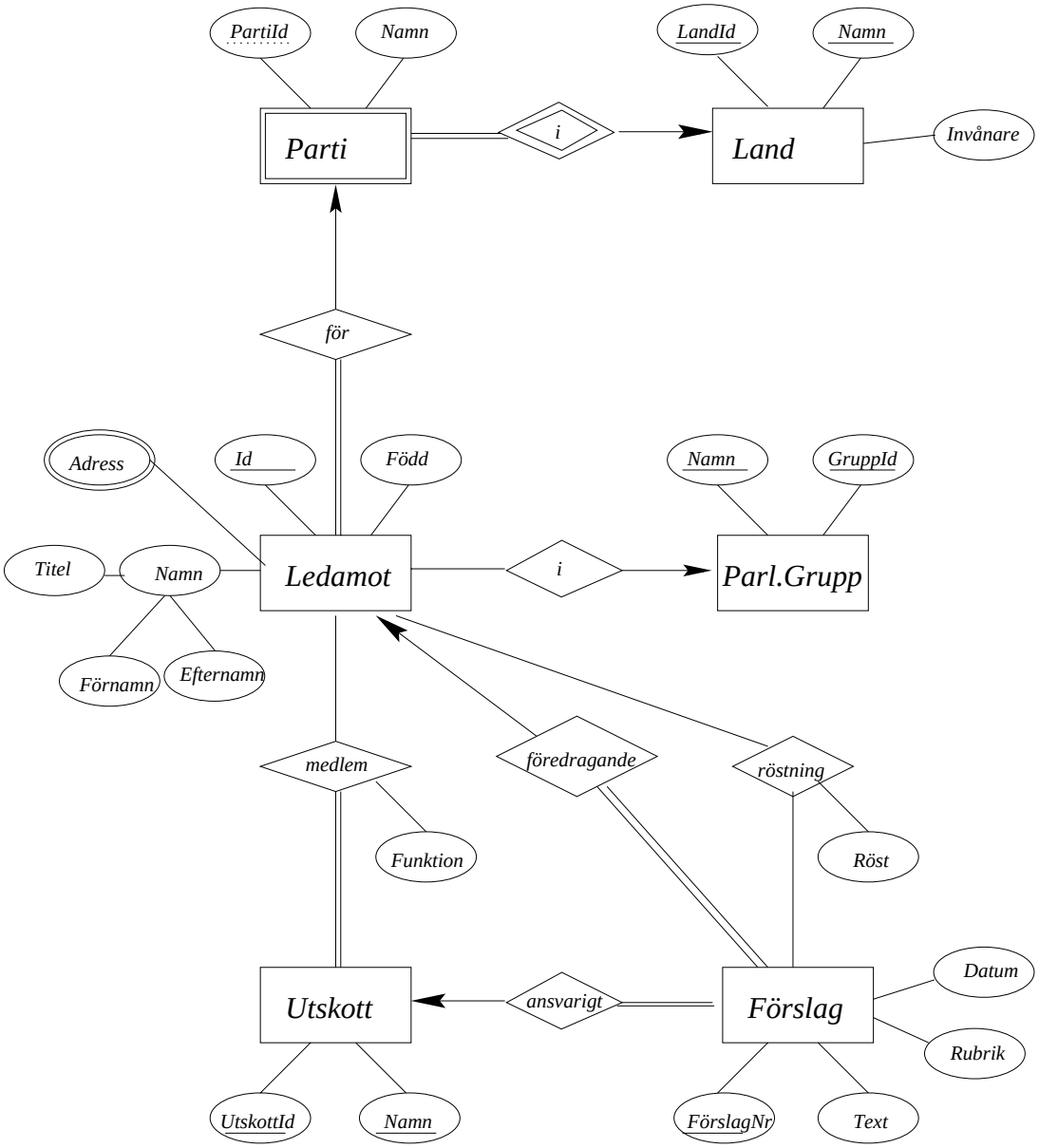
- Varje **ledamot** har ett namn, som består av titel (Mr., Mrs., Señor, Señora, ...), förenamn och efternamn; en unik person-identifierare; ett födelsedatum och en eller flera adresser. Varje ledamot väljs i **ett** land, tillhör **ett** nationellt parti och är medlem i högst **en** parlamentsgrupp.
- Varje **land** har en unik två-bokstavs-kod som identifierar det; har ett namn och har en storlek (antal invånare). Inga två länder har förstås samma namn!
- Ledamöterna delar in sig i 8 **parlamentsgrupper**. Varje grupp har ett unikt namn och en unik förkortning, t.ex. "Europeiska folkpartiet (Kristdemokrater)" har "PPE", "Europeiska socialdemokratiska partiet" har "PSE", osv.
- Varje ledamot tillhör ett nationellt parti. Varje parti tillhör ett visst land. Partiet har ett fullständigt namn, men också en förkortning som partiet vanligtvis använder. Denna förkortning anses vara unikt inom varje land. Beakta dock att förkortningen inte behöver vara unikt över ländernas gränser! Så finns t.ex. två vänsterpartier som båda heter "PDS" – ett i Tyskland och ett i Italien.
- Parlamentet har 17 olika **parlamentsutskott**. Varje utskott har en förkortning på fyra bokstäver som är unik för varje utskott, och har ett *Namn*, vilket också är unikt (t.ex. "BUDG" för "Budget", "ENVI" för "Miljö, folkhälsa, konsumentskydd", "FEMM" för "Kvinnors rättigheter och jämställdhet", osv. Varje utskott har minst en ledamot som medlem. En ledamot kan vara medlem i noll, ett eller flera utskott. En ledamot som är medlem i ett utskott kan ha en funktion i utskottet, t.ex. ordförande eller vice-ordförande.
- Parlamentet diskuterar och röstar om olika **förslag**. Varje förslag identifieras av ett unikt förslagsnummer och har en rubrik, som säger vad det handlar om. Varje förslag hör till ett visst utskott som är ansvarig för förslaget och har en föredragande, vilket är en ledamot som rapporterar förslaget från utskottet till plenum. Till förslaget hör ett datum när det röstas och även själva förslagstexten.

  Till varje förslag ska det lagras vem (dvs. ledamot) som röstade vad ('ja', 'nej' eller 'nedlagd') angående förslaget.

Rita ett ER-diagram som motsvarar ovanstående beskrivning.

*Informationer hämtade från Europaparlamentets webbsida: <http://www.europarl.eu.int/>*

**Svar:** ER-diagram:




12 poäng.

**Obs! – <u>Kursenkät</u> – Vad tycker du om kursen?**

På kursens webbsida finns en **kursenkät** att fylla i. Alla studenter bes att fylla i denna enkät för att vi ska kunna förbättra kursen i framtiden. Tack!