Tentamen i DATABASER

Svar:

Obs! Lärare-version, med lösningar

DAG: lö, 23 okt 1999 **TID:** kl. 14.15 – 18.15 **SAL:** ML11 - ML18 (Maskinhuset)

Ansvarig:	Martin Weichert
Förfrågningar:	Martin Weichert, tel. 772 10 68
Resultat:	anslås den 11 nov 1999
Poängantal:	sammanlagt maximalt 60 poäng.
Betygsgränser:	CTH: 3:a 24 p., 4:a 36 p., 5:a 48 p. GU: Godkänd 28 p., Väl godkänd 48 p. 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) Till vad används *NULL*-värden?

Svar: När ett äkta värde: – inte finns (inte är tillämpbart), – är okänt, – är ännu inte bestämt.

– **1p.**

(b) Vilka är de fyra krav som ställs på ett databassystem med **transaktioner**? Till varje krav, ange en lämplig metod eller åtgärd som kan användas för att garantera detta krav.

Svar:

A. "Atomicity" - "Allt eller inget": Varje transaktion ska genomföras antingen helt eller inte alls. **Åtgärd:** En transaktion "rullas tillbaka" (görs ogjord, "ROLLBACK") om den blir avbruten och därför inte fullständigt genomförd.

C. "Consistency" - "Se upp för konsistensvillkoren": En transaktion som börjar i ett konsistent tillstånd måste leda till ett konsistent tillstånd. **Åtgärd:** Antingen: kravet ställs på användaren (t.ex. programmeraren av användarprogram), eller: Databasen utför konsistenskontroll (kontroll av alla angivna riktighetsvillkor) senast vid varje COMMIT; om transaktionen godkänns endast om alla villkor är uppfyllda och genomförs då, annars rullas den tillbaka.

I. "Isolation" - "Ingen Insyn": Inga (eventuellt inkonsistenta) mellantillstånd av en transaktion ska vara synliga till en annan. **Åtgärd:** t.ex. 2-fas-låsning.

D. "Durability" - "Det som är gjort är gjort" En transaktions ändringar som har avslutats och bekräftats med COMMIT, gäller. **Åtgärd:** Återställning (recovery) efter krasch, samt att alla ändringar verkligen skrivs till hårddisk (antingen i loggfil eller direkt i databasen) vid varje COMMIT (eller varje CHECKPOINT).

– $2 + 4 = 6$ **p.** ($\frac{1}{2}$ för varje krav; 1 för varje åtgärdat krav).

(c) Varför är relationsalgebra viktig för databaser?

Svar: Hela teorin om relationsdatabaser bygger på relationsalgebra. Uppdelning av relationer i delrelationer är en relationsalgebra-operation (projektion, π); förening av dessa till en större är en relationsalgebra-operation (samkörning, $\bowtie$); alla `select`-satser uttrycker relationsalgebra-operationer. Relationsalgebra är det matematiska fundament som relationsdatabaser bygger på.

– **2p.**

(d) Vad är och var används fragmentering? Vad är horisontell och vertikal fragmentering?

Svar: F. används vid distribuerade databaser. Tabellerna i en databas delas upp på olika maskiner/noder/servrar i ett nät.

Vid h.f. delas en tabell i mängder av rader (σ) och man återfår den med union ($\cup$).

Vid v.f. delas en tabell i mängder av kolumner (π) och återfås med samkörning ($\bowtie$). – 3p.

12 poäng.

Uppgift 2. Låt $r(A, B, C, D)$ och $s(A, C, E)$ vara tabeller med följande innehåll:

r:

A	B	C	D
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1
a_2	b_3	c_1	d_3

s:

A	C	E
a_1	c_2	e_1
a_1	c_1	e_2
a_2	c_3	e_3
a_1	c_2	e_4

Beräkna resultatet av följande relationsalgebra-uttryck!

(a) $\pi_A(\sigma_{C \neq c_1}(s))$

(b) $(\pi_A(s) \times \pi_C(s)) - \pi_{A,C}(r)$

(c) $r \bowtie s$ (naturlig samkörning)

Svar:

(a, b) $\pi_A(\sigma_{C \neq c_1}(s))$:

A
a_1
a_2

: **2p.**

$(\pi_A(s) \times \pi_C(s)) - \pi_{A,C}(r)$:

A	C
a_1	c_3
a_2	c_2
a_2	c_3

: **3p.**

(c) $r \bowtie s$:

A	B	C	D	E
a_1	b_1	c_1	d_2	e_2
a_1	b_2	c_2	d_1	e_1
a_1	b_2	c_2	d_1	e_4

: **3p.**

Kolumnerna kan även vara i annan ordning, t.ex. (A, C, B, D, E).

Poängavdrag ($-$ **1p.**) om det finns dubletter.

8 poäng.

Uppgift 3. Följande tabell har skapats i databasen:

```
create table ANSTÄLLDA (
  USERID   number(4),
  NAMN     varchar(15)     not null,
  CHEF     number(4)       default 1000,
  UPPGIFT  varchar(12),
  LÖN      number(5)       default 0,
  START    date,
  constraint VILLKOR1 primary key (USERID),
  constraint VILLKOR2 unique (NAMN),
  constraint VILLKOR3 foreign key (CHEF) references ANSTÄLLDA(USERID)
                      on delete cascade,
  constraint VILLKOR4 check (LÖN between 10000 and 25000)
);
```

Följande rader har redan matats in:

UserId	*Namn*	*Chef*	*Uppgift*	*Lön*	*Start*
1003	'Roberts'	1002	'biljettförs'	12000	1999-JUL-01
1004	'Ruskin'	1002	'biljettförs'	12000	1998-NOV-15
1002	'Price'	1000	'biljettchef'	15000	1996-APR-01
1000	'Powell'	NULL	'avd-chef'	20000	1994-JAN-01

Kontrollera de följande SQL-satserna väldigt noga. Vilka av dessa ändringsförsök kommer att bli accepterade, vilka inte? Ange hur ändringsförsöken berörs av de olika riktighetsvillkoren, så att de blir accepterade eller inte, och vad som kommer att ändras i tabellen. För dem som *inte* blir accepterade, ange *alla* riktighetsvillkor mot vilka de bryter!

(Attributlistan efter `insert` anger vilka attribut som matas in och i vilken följd. Attribut som saknas i listan får sitt förvalt värde.)

(a)
```
insert into ANSTÄLLDA (NAMN, USERID, CHEF, UPPGIFT, LÖN, START)
  values ('Raphael', 1005, 'Price', 'biljettförs', 12000, '1999-OCT-23');
```

Svar: fel: CHEF har fel typ – måste vara chefens USERID, ej chefens NAMN.

– 1p.

(b)
```
insert into ANSTÄLLDA (USERID, NAMN, CHEF, UPPGIFT, START)
  values (1006, 'Roberts', 1002, 'biljettförs', '1999-SEP-31');
```

Svar: Tre fel:

- namnet finns redan → bryter mot VILLKOR2
- ingen LÖN angiven → DEFAULT-värdet 0 används → bryter mot VILLKOR4
- felaktigt datum

– 3p.

(c)
```
insert into ANSTÄLLDA (USERID, NAMN, UPPGIFT, LÖN, START)
  values (1010, 'Ruskins', 'flygmek', 12000, '1999-OCT-23');
```

Svar: Raden accepteras. *NAMN* 'Ruskins' är skilt från 'Ruskin' → bryter EJ mot VILLKOR1. CHEF saknas → DEFAULT-värdet 1000 används → bryter EJ mot VILLKOR3, eftersom en referensrad med USERID = 1000 finns.

– 2p.

(d)
```
update ANSTÄLLDA set LÖN = 1.5 * LÖN where START < '1997-JAN-01';
```

Svar: Fel: Powells lön blir 30000 → bryter mot VILLKOR4.

– 1p.

(e)
```
delete from ANSTÄLLDA where UPPGIFT = 'avd-chef';
```

Svar: Hela tabellen töms! Avdelningschefen Powell tas bort, med honom försvinner (både Ruskins från (c) och) biljettchefen Price, och med Price även alla under biljettförsälare under honom.

– 1p.

8 poäng.

Uppgift 4. 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 med följande tabeller användas:

- *ledamot(PersonId, Namn, Född, LandId, PartiId, GruppId)*
  En lista över ledamöter, med *PersonId*, en unik identifierare för varje ledamot; *Namn*, ledamotens namn; *Född*, födelseåret; *LandId*, en två-bokstavs-kod för landet som ledamoten representerar; *PartiId*, en förkortning för partiet som ledamoten representerar; *GruppId*, en förkortning för den parlamentsgrupp som ledamoten tillhör. De passande attributen är samtidigt referenser till tabellerna *land*, *parti* och *grupp*.

- *land*:

LandId	*Namn*	*Invånare*
SE	Sverige	8 milj.
ES	Spanien	40 milj.
...	...	...

  En lista över alla 15 länder, med: *LandId*, en två-bokstavs-kod för landet; landets *Namn*; samt antalet *Invånare*. Inga två länder har förstås samma namn! Både *LandId* och *Namn* är nycklar till relationen.

- *grupp*:

GruppId	*Namn*
PPE	Europeiska folkpartiet (kristdemokrater)
PSE	Europeiska socialdemokratiska partiet
...	...

  En lista över alla 8 parlamentsgrupper.

- *parti*:

LandId	*PartiId*	*Namn*
SE	s	socialdemokraterna
SE	m	moderaterna
...	...	...

  En lista över alla nationella partier, med *LandId*, två-bokstavs-koden för landet (referens till *land*); *PartiId*, den förkortning som partiet vanligtvis använder; samt partiets fullständiga *Namn*. *PartiId* anses vara unikt inom varje land. Beakta dock att *PartiId* inte behöver vara unikt över ländernas gränser! Så finns t.ex. två vänsterpartier som båda heter "PDS" – ett i Tyskland och ett i Italien. *LandId* och *PartiId* tillsammans är nyckeln till relationen.

- *utskott*:

UtskottId	*Namn*
BUDG	Budget
ENVI	Miljö, folkhälsa, konsumentskydd
FEMM	Kvinnors rättigheter och jämställdhet
...	...

  En lista över alla 17 parlamentsutskott, med en identifierare *UtskottId*, som är unik för varje utskott och som alltid består av fyra bokstäver, och ett *Namn*, vilket också är unikt.

- *utskottMedlem(PersonId, UtskottId, Funktion)*
  Vem som är medlemmar i vilka utskott. *PersonId* och *UtskottId* är referenser till *ledamot* och *utskott*. *Funktion* kan vara 'ordförande', 'vice-ordf' eller tom.

- *förslag(FörslagNr, Rubrik, UtskottId, Föredragande, Datum, Text, ...)*
  En lista över förslag som behandlas i parlamentet.
  FörslagNr, en unik identifierare för varje förslag; *Rubrik*, vad det handlar om; *UtskottId*, en förkortning för utskottet som är ansvarig för förslaget (referens till *utskott*); *Föredragande*, vem som rapporterar förslaget från utskottet till plenum (referens till *ledamot*); *Datum*, när förslaget röstas; och *Text*, själva förslagstexten.

- *röstning(PersonId, FörslagNr, Röst)*
  Röstresultaten: anger vilken *Röst* ('ja', 'nej' eller 'nedlagd') ledamoten *PersonId* (referens till *ledamot*) gav angående *FörslagNr* (referens till *förslag*).

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

Först skall databasen skapas:

(a) Ange de *fullständiga* SQL-satserna som skapar tabellerna *land* och *ledamot* enligt beskrivningen ovan.

Svar:

```
create table LAND (
  LANDID   char(2),
  NAMN     varchar(30),
  INVÅNARE decimal(9),
  primary key (LANDID),
  unique (NAMN)
);
create table LEDAMOT (
  PERSONID decimal(3),
  NAMN     varchar(30),
  FÖDD     decimal(4),
  LANDID   char(2),
  PARTIID  varchar(8),
  GRUPPID  char(4),
  primary key (PERSONID),
  foreign key (LANDID) references LAND,
  foreign key (LANDID,PARTIID) references PARTI,
  foreign key (GRUPPID) references GRUPP );
```

– **4p.**

För resten av uppgiften antar vi att databasen är skapad och är fylld med data. Skriv SQL-satser för följande uppgifter:

(b) Gör en lista över Sveriges alla 22 ledamöter: För varje ledamot ska det anges namn, födelseår, samt namnen på det parti och den parlamentsgrupp som ledamoten tillhör. Utskriften ska vara sorterad från yngst till äldst.

Svar:

```
select L.NAMN, L.FÖDD, P.NAMN, G.NAMN
  from LEDAMOT L, PARTI P, GRUPP G
 where L.LANDID = P.LANDID
   and L.PARTIID = P.PARTIID
   and L.GRUPPID = G.GRUPPID
   and L.LANDID = 'SE'
 order by FÖDD desc;
```

Inga av villkoren får glömmas!
– 3p.

(c) Lista Sveriges alla ledamöter och hur de röstade angående lagstiftningresolutionen om *Främjande av förnybara energikällor* som har förslagnummer 'A4-0085/99'.

Svar:

```
select L.NAMN, R.RÖST
  from LEDAMOT L, RÖSTNING R
 where L.LANDID = 'SE'
   and L.PERSONID = R.PERSONID
   and R.FÖRSLAGNR = 'A4-0085/99';
```

Obs! Tabellen FÖRSLAG behövs inte. – 2p.

(d) Gör en lista över alla utskott, med antalet *svenska* medlemmar i varje utskott.

Svar:

```
select UTSKOTTID, count(*) as SVENSKAR
  from LEDAMOT L, UTSKOTTMEDLEM M
 where L.LANDID = 'SE'
   and L.PERSONID = M.PERSONID
 group by UTSKOTTID;
```

Bara de utskott som **har** svenska medlemmar behöver tas med. Vill man även ha utskott utan svenska medlemmar, så kan man lägga till:

```
...union
 select UTSKOTTID, 0
   from UTSKOTT
  where UTSKOTTID not in
    (select UTSKOTTID
       from LEDAMOT L, UTSKOTTMEDLEM M
      where L.LANDID = 'SE'
        and L.PERSONID = M.PERSONID);
```

– 2p.

(e) Lista det eller de utskott som har högsta antalet medlemmar.

Svar:

```
create view TMP as
select UTSKOTTID, count(*) as ANTAL
  from UTSKOTTMEDLEM
 group by UTSKOTTID;

select UTSKOTTID from TMP
 where ANTAL = ( select max(ANTAL) from TMP );
```

Obs! "`count(*)`" **måste** döpas om, t.ex. så som gjort här till `ANTAL`.

– 3p.

(f) Alla dessa informationer skall vara offentliga – ge därför rättigheten till alla att läsa (men inte ändra) i tabellerna.

Ge dessutom användaren *Admin* rättigheten att lägga in och ändra rader i tabellen *ledamot*.

Svar:

```
grant SELECT on LEDAMOT to PUBLIC;
grant SELECT on LAND to PUBLIC;
...
grant SELECT on RÖSTNING to PUBLIC;

grant INSERT,UPDATE on LEDAMOT to ADMIN;
```

– 2p.

Skapa gärna hjälpvyer för dina sökfrågor om du behöver.

16 poäng.

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

1. $ABC \to DE$
2. $E \to BC$

(a) Bestäm alla nycklar till relationen r.

Svar: $\{A, B, C\}$ och $\{A, E\}$. – **2p.**

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

Svar: Nej. Beroendet 2. $E \to BC$ bryter mot BCNF, eftersom determinanten E inte är nån (över)nyckel. – **1p.**

(c) Uppfyller r tredje normalformen? Motivera!

Svar: Ja. Beroendet 2. $E \to BC$ uppfyller 3NF, eftersom både B och C är nyckelattribut. (Beroendet 1. $ABC \to DE$ uppfyller 3NF eftersom det redan uppfyller det starkare BCNF.) – **1p.**

(d) Om vi delar upp relationen r för att uppfylla en bättre (d.v.s. strängare) normalform, vilka delrelationer får vi? Vad förlorar vi?

Svar: Vi delar upp med beroendet 2. $E \to BC$, och vi får relationer $ade(\underline{A, D, E})$ och $ebc(\underline{E}, B, C)$. Vi förlorar beroendet 1. $ABC \to DE$.
– **2p.**

6 poäng.

Uppgift 6. En relationsdatas brukar innehålla en *systemkatalog*, dvs. en beskrivning av hela databasens struktur, som även själv görs i form av tabeller. Följande informationer skall finnas med i systemkatalogen:

- I databasen finns två sorter tabeller (*TABLE*), nämligen bastabeller *(BASE_TABLE)* och vyer *(VIEW)*. Inga två tabeller i databasen kan ha samma namn, dvs. tabeller identifieras entydigt med sina namn.

  Till varje bastabell lagras vissa egenskaper, så som det aktuella antalet rader i den, antalet block den använder på hårddisk, och det senaste ändringsdatumet. (I verkligheten lagras många mer egenskaper än dessa!) Varje bastabell kan ha ett eller flera *index* (datatrukturer för snabbare åtkomst).

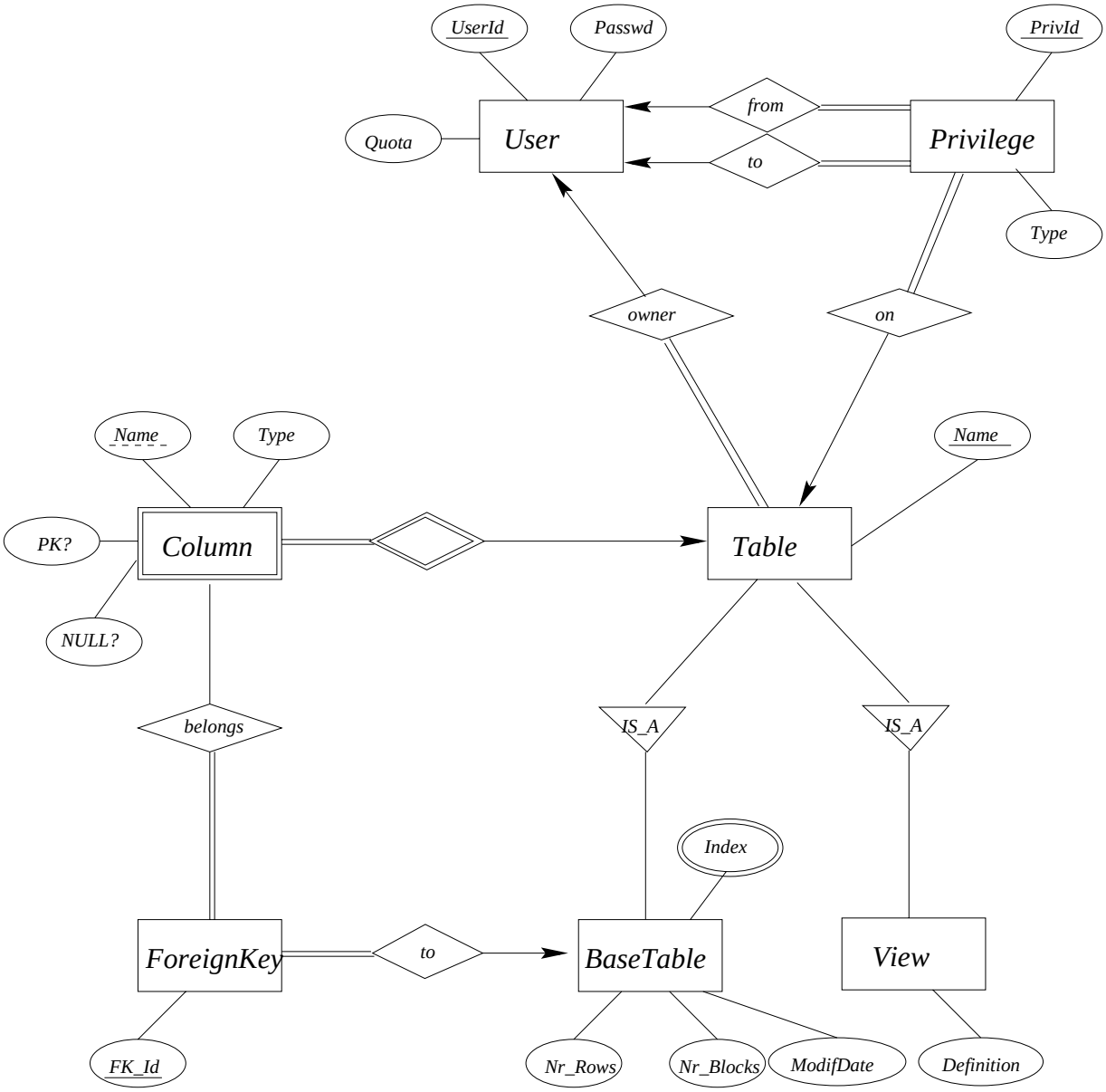
  Till varje vy lagras SELECT-satsen som den definierades med som en textsträng.
- Varje tabell har en mängd attribut (kolumner, *COLUMN*). Varje kolumn måste tillhöra en viss tabell. Två attribut i olika tabeller kan ha samma namn, däremot inte två attribut i samma tabell. Varje kolumn har en viss datatyp. För varje attribut lagras om det ingår i primärnyckeln eller ej, samt om det kan vara NULL eller ej.
- Det lagras också information om referenser (främmande nycklar) mellan tabellerna. Varje främmande nyckel (*FOREIGN_KEY*) pekar på *en* viss tabell (vilken måste vara en bastabell, ingen vy). I varje främmande nyckel ingår ett eller flera attribut; ett attribut kan ingå i noll, en, eller fler främmande nycklar.
- Varje tabell ägs av en databas-användare (*USER*). Varje användare har ett användarnamn (*userid*), ett lösenord och en *disk quota*.
- Användare kan dela ut rättigheter till andra användare. Varje rättighet (*PRIVILEGE*) ges av en viss användare och tas emot av en viss användare. Den gäller en viss tabell och har en viss typ.

Du kan införa egna nya nyckelattribut (löpande nummer eller dyl.) om det känns nödvändigt eller lämpligt. Om du gör antaganden, ange vilka.

Rita ett diagram enligt EER-modellen (*"Enhanced Entity-Relationship"*) som motsvarar ovanstående beskrivning.

Svar: Varje punkt i följande lista, om det är rätt använt, räknas som **1p.**

- entiteter (rektanglar)
- attribut (ovaler)
- samband (romber)
- unika attribut = nyckelattribut (attributnamn understruket)
- flervärde-attribut (dubbeloval)
- ett "måste"-samband (dubbelstreck)
- 1:N-samband (ett pilhuvud; måste vara åt rätt håll!)
- M:N-samband (inget pilhuvud)
- svag entitet (dubbel rektangel) med partiell nyckel (punkterad understruket) och identifierande samband (dubbelruta; måste vara 1:N) till identifierande ägare.
- arv ("IS_A").



10 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!