NTNU
Institutt for datateknikk
og informasjonsvitenskap

Eksamensoppgave i

TDT4145 – Datamodellering og databasesystemer - kontinuasjonseksamen

Tirsdag 10. september 2010, kl. 09:00 - 13:00

Oppgaven er utarbeidet av faglærer Kjell Bratbergsengen.
Kontaktperson under eksamen er Kjell Bratbergsengen (mobiltelefon 906 17 185)

Språkform: Bokmål
Tillatte hjelpemidler: D
Ingen trykte eller håndskrevne hjelpemidler tillatt.
Bestemt, enkel kalkulator tillatt.

Sensurfrist: Tirsdag tirsdag 31. august 2010.

Les oppgaveteksten nøye. Finn ut hva det spørres etter i hver oppgave.
Dersom du mener at opplysninger mangler i en oppgaveformulering gjør kort rede for de antagelser og forutsetninger som du finner det nødvendig å gjøre.

Oppgave 1 (30 %)

a) Vi skal lage en database for registrering av plantefunn. Databasen skal brukes til å overvåke utbredelsen av planter over tid. Noen planter er utrydningstruet (rødliste), mens andre igjen både er på fremmarsj og uønsket som for eksempel ”Tromsøpalmen” (svarteliste). Databasen skal være åpen og interaktiv, og skal selv inneholde dokumentasjon for utviklingen av sitt innhold.

Databasen skal inneholde en oversikt over plantene som er gruppert i et hierarkisk system: en art tilhører en slekt som tilhører en familie som tilhører en orden som tilhører en klasse. Hierarkiet går høyere, men vi stopper der. På hvert nivå er det et norsk og et latinsk navn. Til hver art kan det være en beskrivende tekst, ett eller flere bilder eller tegninger, og en kode for giftighet og status i Norge (rødliste, svarteliste).

Viktige bidragsytere til innholdet i databasen er amatører som observerer planter (observatører). Når en plante blir observert kan en være usikker på artsbestemmelsen. En observasjon skal registreres med følgende data: observatør, dato, sannsynlig artsnavn (observatørens antakelse), funnsted - enten gps-koordinater eller en kartreferanse, og en tekstlig beskrivelse. En observasjon kan også

dokumenteres av fotografier. Et funn kan kommenteres av registrerte brukere av databasen. Kommentarene skal registreres sammen med dato og den som kommenterer. En kommentar kan også være relatert til en eller flere andre kommentarer. Observasjoner bearbeides av fagpersoner som er tilknyttet databasen. En fagperson kan vurdere artsbestemmelsen til å være sikker, sannsynlig, eller usikker. Identiteten til den som vurderer og dato for vurderingen skal registreres. Personer er registrert med personnummer, navn, adresse og status (fagperson, observatør, gjest).

Tegn EER-diagram for en database som reflekterer opplysningene vi har gitt over. Redegjør for eventuelle antakelser som du gjør.

b) Omform EER-diagrammet til en relasjonsdatabase. Vis hvilke felter som er nøkler og fremmednøkler.

Oppgave 2 (25 %)

Ansatte i en bedrift er registrert i en database med følgende innhold:

PERSON (PNr, EtterNavn, ForNavn, *AdresseId*)
ADRESSE(AdresseId, Gate_Vei, HusNr, *PostNr*)
POSTSTED(PostNr, StedsNavn)
AVDELING(AvdId, AvdNavn)
ANSATT(*PNr, AvdId*, Stilling, Lønn)

a) Lag en view eller virtuell tabell som inneholder persondata (personnummer, for- og etternavn) og fullstendig adresse (navn på gate eller vei, husnummer, postnummer og poststed).

b) Er denne virtuelle tabellen oppdaterbar? Begrunn svaret ditt.

c) Skriv ut en rapport over antall personer som bor i samme gate eller vei.

d) Vi finner ut at databasen ikke dekker behovene for å lagre historiske data om en persons tilsettinger og lønn. For å kunne ta vare på endringene som har skjedd, endrer vi tabellen ANSATT til ANSATTH. Tabellen inneholder nå følgende data:

ANSATTH (*PNr, AvdId, FraDato*, TilDato, Stilling, Lønn)

For et ansettelsesforhold som fortsatt løper har vi den konvensjon at TilDato har verdien ’2099-12-31’

Skriv ut en rapport over ansettelseshistorikken til alle personer i Innkjøpsavdelingen. Rapporten skal være sortert på person og ansettelses- eller endringsdato, de siste endringene kommer først. For hver linje skal vi ha med personens nummer og navn, stilling, lønn og hvilken periode dette gjelder for (fra – og tildato).

e) For at vi skal kunne kjøre gamle programmer som før, skal du lage en virtuell tabell (view) som maskerer den endringen vi har gjort i databasen, altså overgangen fra ANSATT til ANSATTH.

Oppgave 3 (25 %)

A	B	C
A	b	100
D	b	100
E	a	200
B	c	400
E	b	100
D	a	200
C	b	100
E	c	400

a) Hvilke funksjonelle avhengigheter kan eksistere for tabellen over, ut fra de data som akkurat nå står i tabellen?

b) Hva er forskjellen på de funksjonelle avhengigheter vi analyserer oss fram til, og de vi kan finne ved å observere data i en tabell på et gitt tidspunkt?

c) Hvordan er definisjonen på BCNF (Boyce Codd Normal Form)?

d) For den universelle relasjon R(abcdef) har vi følgende funksjonelle avhengigheter:
e→ab, f→a, d→b, cb→f.
Finn nøkkelen(e) til R.

e) Dekomponer R til tabeller på BCNF.

Oppgave 4 (20 %)

a) Hva menes med at en utførelsessekvens er henholdsvis:
1) Ikke gjenopprettbar (not recoverable),
2) gjenopprettbar (evt. med galopperende abort (cascading abort)),
3) gjenopprettbar, ikke galopperende abort og
4) strikt?

b) For at en transaksjon skal kunne utføres korrekt må den både være gjenopprettbar og serialiserbar. Vi har følgende historie:

H: r1(x), r2(y), w2(y), r3(z), r1(y), a2, r3(x), r1(z), w1(z), w3(x) …

Ved … er alle gjenværende transaksjoner klare til å komitte. Kan de det?

c) Strikt tofaselåsing sikrer både gjenopprettbarhet og serialiserbarhet.
Beskriv strikt tofaselåsing.

d) Hvordan ville utførelsessekvensen blitt for historien H i b) hvis databasen brukte strikt tofaselåsing?

kjb/21.7.2010/

TDT4145 – Datamodellering og databasesystemer
Forslag til løsning

Tirsdag 10. september 2010, kl. 09:00 - 13:00

Oppgave 1 (30 %)

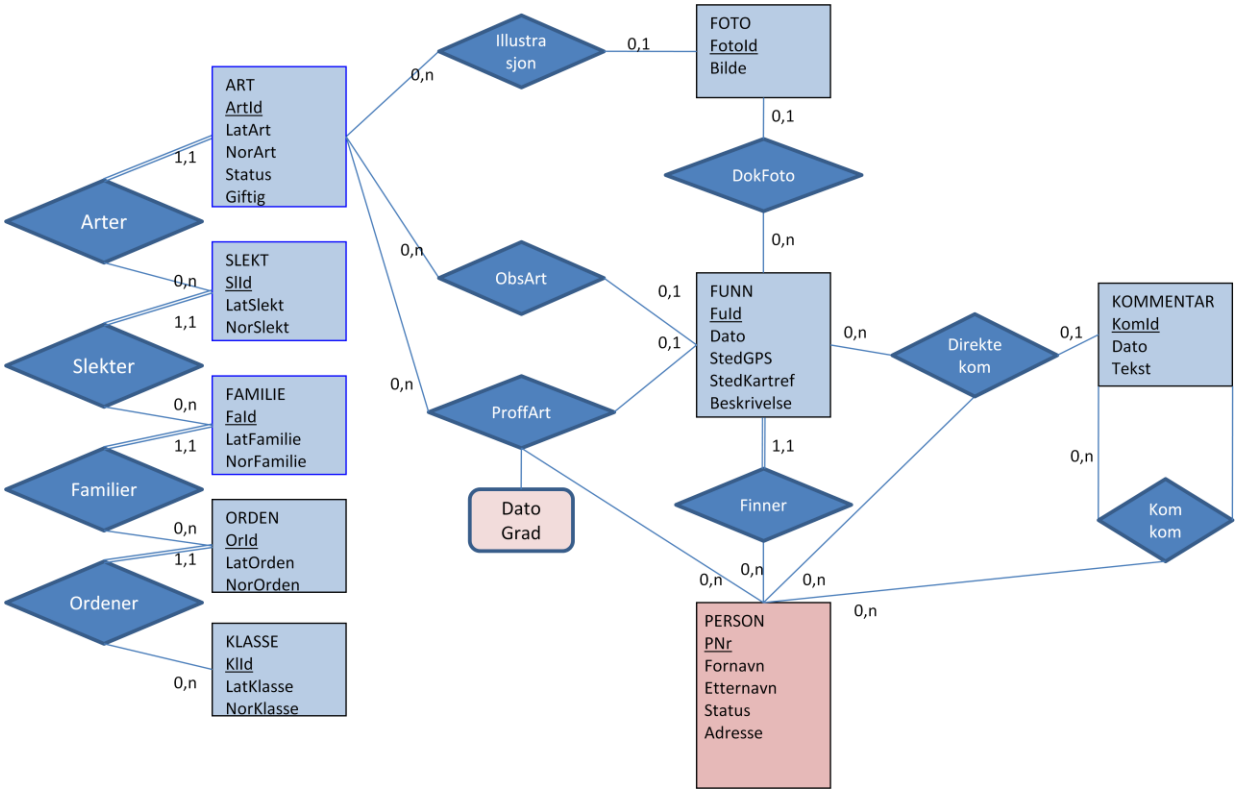
a) Vi skal lage en database for registrering av plantefunn. Databasen skal brukes til å overvåke utbredelsen av planter over tid. Noen planter er utrydningstruet (rødliste), mens andre igjen kanskje både er på fremmarsj og uønsket som for eksempel "Tromsøpalmen" (svarteliste). Databasen skal være åpen og interaktiv, og skal selv inneholde dokumentasjon for utviklingen av sitt innhold.

Databasen skal inneholde en oversikt over plantene som er gruppert i et hierarkisk system: en art tilhører en slekt som tilhører en familie som tilhører en orden som tilhører en klasse. Hierarkiet går høyere, men vi stopper der. På hvert nivå er det et norsk og et latinsk navn. Til hver art kan det være en beskrivende tekst, ett eller flere bilder eller tegninger, og en kode for giftighet og status i Norge (rødliste, svarteliste).

Viktige bidragsytere til innholdet i databasen er amatører som observerer planter. Når en plante blir observert kan en være usikker på artsbestemmelsen. En observasjon skal registreres med følgende data: observatør, dato, sannsynlig artsnavn (observatørens antakelse), funnsted - enten gps-koordinater eller en kartreferanse, og en tekstlig beskrivelse. En observasjon kan også dokumenteres av fotografier. Et funn kan kommenteres av registrerte brukere av databasen. En kommentar er en tekst på inntil 400 tegn. Kommentarene skal registreres sammen med dato og den som kommenterer. En kommentar kan også være relatert til en eller flere andre kommentarer. Observasjoner bearbeides av fagpersoner som er tilknyttet databasen. En fagperson kan vurdere artsbestemmelsen til å være sikker, sannsynlig, eller usikker. Identiteten og til den som vurderer og dato for vurderingen skal registreres. Personer er registrert med personnummer, navn, adresse og status (fagperson, amatør, gjest).

Løsning

Kommentarer:
Entitetene er gitt løpenummer som identifikator. Adresse er registrert "in line" fordi adressene ikke er spesielt viktige som egne objekter i denne databasen.

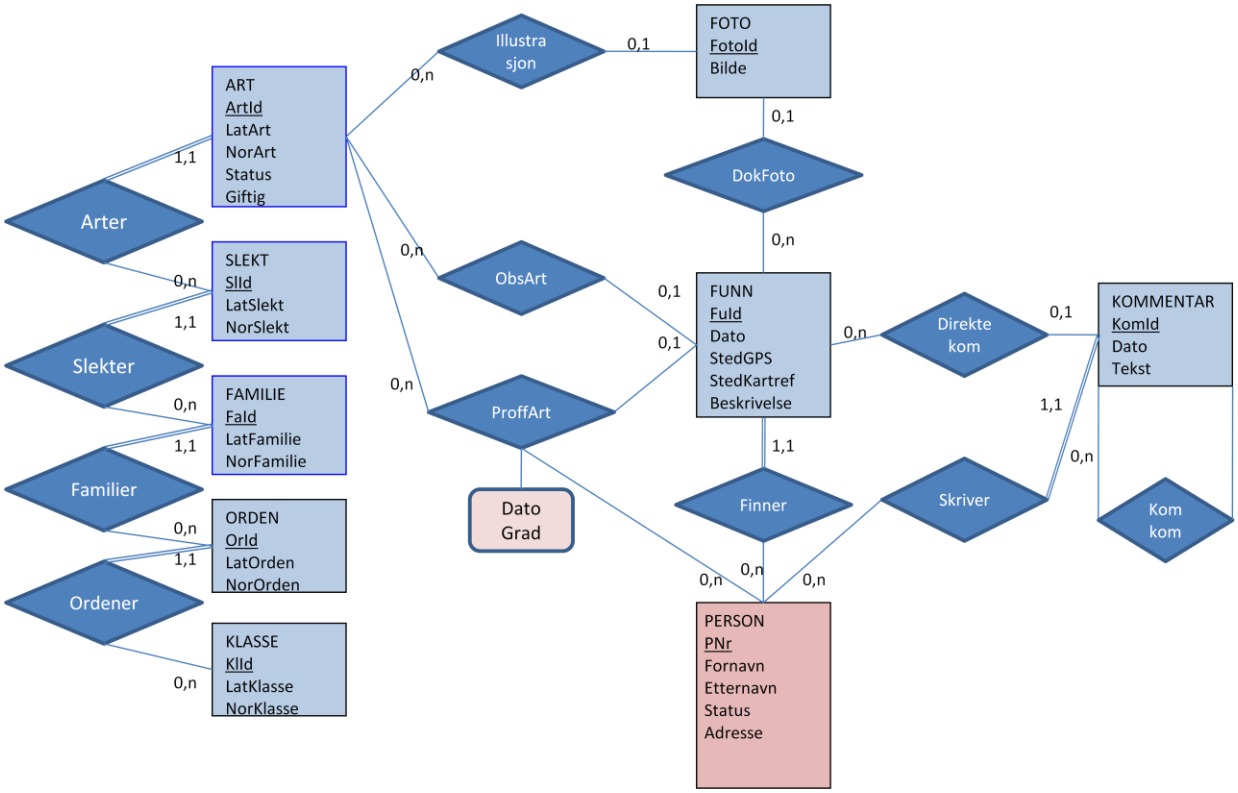


b) Omform EER-diagrammet til en relasjonsdatabase. Vis hvilke felter som er nøkler og fremmednøkler.

KLASSE (KlId, LatKlasse, NorKlasse)
ORDEN (OrId, LatOrden, NorOrden, *KlId*)
FAMILIE (FaId, LatFamilie, NorFamilie, *OrId*)
SLEKT (SlId, LatSlekt, NorSlekt, *FaId*)
ART (ArtId, LatArt, NorArt, *SlId*)
FOTO (FotoId, Bilde, *Illustrasjon_ArtId*, *DokFoto_FuId*)
FUNN (FuId, Dato, StedGPS, StedKartref, Beskrivelse, *ObsArt_ArtId*, *ProffArt_ArtId*, ProfArt_Dato, ProfArt_Grad, *ProffArt_PNr*, *Finner_PNr*)
PERSON (PNr, Fornavn, Etternavn, Status, Adresse)
KOMMENTAR (KomId, Dato, Tekst, *Skriver_PNr*, *DirekteKom_FuId*)
KomKom (*KomId*, *KomId_Om*)

Alle relasjoner unntatt én er en-til-mangerelasjoner som ikke trenger egne tabeller. Den profesjonelle artsbestemmelen tas inn i FUNN-tabellen. Dato for bestemmelsen og graden av sikkerhet tas inn som data, fagpersonen som gjør bestemmelsen tas inn med fremmednøkkel til person. En kommentar kan gjelde mange andre kommentarer og en kommentar kan bli kommentert av flere, dermed er det har en mange-til-mangerelasjon som må ha en egen tabell.

DirekteKom og KomKom er begge triple relasjoner i ER-diagrammet. Hvem som skriver kommentaren kunne også vært en binær relasjon mellom kommentar og person. Tabellene er utformet etter en slik variant. Dermed slipper en å gjenta personnøkkelen i tabellen som viser kommentarer om kommentarer. Diagrammer nedenfor viser dette:



Oppgave 2 (25 %)

Ansatte i en bedrift er registrert i en database med følgende innhold:

PERSON (PNr, EtterNavn, ForNavn, *AdresseId*)
ADRESSE(AdresseId, Gate_Vei, HusNr, *PostNr*)
POSTSTED(PostNr, StedsNavn)
AVDELING(AvdId, AvdNavn)
ANSATT(*PNr, AvdId*, Stilling, Lønn)

a) Lag en view eller virtuell tabell som inneholder persondata (personnummer, for- og etternavn) og fullstendig adresse (navn på gate eller vei, husnummer, postnummer og poststed).

Svar:

```
CREATE VIEW PersonData AS
SELECT PNr, EtterNavn, ForNavn, AdresseId, Gate_Vei, HusNr, PostNr, StedsNavn
FROM PERSON NATURAL JOIN ADRESSE NATURAL JOIN POSTSTED;
```

eller:

```
CREATE VIEW PersonData AS
SELECT PNr, EtterNavn, ForNavn, AdresseId, Gate_Vei, HusNr, PostNr, StedsNavn
FROM PERSON p, ADRESSE a, POSTSTED s
WHERE p.AdresseId=a.AdresseId AND a.PostNr=s.PostNr;
```

b) Er denne virtuelle tabellen oppdaterbar? Begrunn svaret ditt.

Svar:

Den virtuelle tabellen er ikke oppdaterbar. Det er ikke et entydig forhold mellom virtuell attributt og opprinnelse.

c) Skriv ut en rapport over antall personer som bor i samme gate eller vei.

Svar:

```
SELECT Gate_Vei, COUNT(*) AS Antall_i_Gata
FROM PersonData
GROUP BY Gate_Vei;
```

En kan her peke på at det i en database kan finnes mange forskjellige gater og veier med same navn. Vi forutsetter at det ikke finnes flere gater eller veier med samme navn under *samme* postnummer. Tar vi hensyn til denne komplikasjonen kan vi løse oppgaven ved følgende SQL-setning:

```
SELECT Gate_Vei, Antall_i_Gata
FROM
  SELECT PostNr, Gate_Vei, COUNT(*) AS Antall_i_Gata
  FROM PersonData
  GROUP BY  PostNr, Gate_Vei;
```

Denne løsningen er ikke krevet til eksamen.

Start oppgave d)

Vi finner ut at databasen ikke dekker behovene for å lagre historiske data om en persons tilsettinger og lønn. For å kunne ta vare på endringene som har skjedd opp gjennom tiden endrer vi tabellen ANSATT til ANSATTH. Tabellen inneholder nå følgende data:

ANSATTH (*PNr, AvdId, FraDato*, TilDato, Stilling, Lønn)

For et ansettelsesforhold som fortsatt løper har vi den konvensjon at TilDato har verdien ’2099-12-31’

d) Skriv ut en rapport over ansettelseshistorikken til alle personer i Innkjøpsavdelingen. Rapporten skal være sortert på person og ansettelses- eller endringsdato, de siste endringene kommer først. For hver linje skal vi ha personens nummer og navn, stilling, lønn og hvilken periode dette gjelder for (fra – og tildato).

Svar:

Legg merke til at nøkkelen i ANSATTH er utvidet med DatoFra. Det er nødvendig fordi samme person vil kunne ha forskjellige stillinger og fremfor alt forskjellig lønn i forskjellige perioder ved samme avdeling. For å samle opplysningene om en person må vi sortere på personnummer og for å få siste periode først må vi også sortere på FraDato i synkende orden.

```
SELECT PNr, ForNavn; EtterNavn, Stilling, FraDato, TilDato
FROM PersonData p, AVDELING a, ANSATTH h
WHERE a.AvdNavn LIKE ’Innkjøpsavdeling*’
AND a.AvdId = h.AvdId
AND h.PNr = p.PNr
ORDER BY PNr, FraDato DESC;
```

e) For at vi skal kunne kjøre gamle programmer som før, skal du lage en virtuell tabell (view) som maskerer den endringen vi har gjort i databasen, altså overgangen fra ANSATT til ANSATTH.

Svar:

Vi velger ut de attributter som er i den gamle tabellen ANSATT, og for å få den løpende ansettelsen så bruker vi konvensjonen som er oppgitt i oppgaveteksten: ”For et ansettelsesforhold som fortsatt løper har vi den konvensjon at TilDato har verdien ’2099-12-31’”. Det var et hovedpoeng med deloppgaven.

CREATE VIEW ANSATT AS
SELECT PNr, AvdId, Stilling, Lønn
FROM ANSATTH h
WHERE TilDato = DATO’2099-12-31’;

Denne tabellen er forøvrig heller ikke oppdaterbar da den ikke inneholder alle attributter som inngår i nøkkelen til basetabellen - den tabellen de er basert på. Ikke spurt om dette til eksamen. Det betyr i praksis at gamle programmer som oppdaterer ANSATT må skrives om!

Oppgave 3 (25%)

A	B	C
A	b	100
D	b	100
E	a	200
B	c	400
E	b	100
D	a	200
C	b	100
E	c	400

a) Hvilke funksjonelle avhengigheter kan eksistere for tabellen over, ut fra de data som akkurat nå står i tabellen?

Svar:
De funksjonelle avhengighetene vi finner ved å se på data på et bestemt tidspunkt i en tabell er flere enn, eller like mange som - de som finnes i virkeligheten. Dess færre data det er i tabellen – dess flere FA (falske) kan vi finne. En systematisk gjennomgang er krevende – særlig for tabeller med mange kolonner, vi må se på alle kombinasjoner av mulige nøkler med 1 attributt, 2 attributter, 3 attributter osv.

Systematisk gjennomgang:

A $\nrightarrow$ B, vi finner både Db og Da. (A $\nrightarrow$ B leses slik: A bestemmer ikke B.)

A $\nrightarrow$ C, vi finner D100 og D200

B $\rightarrow$ C, ingen “motstridende” verdipar.

B $\nrightarrow$ A, bA, bD

C $\nrightarrow$ A, 100D, 100E

C →B,

AB → C, kan også avledes av Armstrongs aksiomer
AC → B, samme

BC ↛ A

Sammentrukket:
B → C
C →B
AB → C
AC → B

I de FA-er som er gulet ut er venstresiden en determinant, dvs. bestemmer andre attributter. Avhengigheter som er merket med blått følger av andre FA-er med en mindre nøkkel. (FA – funksjonell avhengighet)

b) Hva er forskjellen på de funksjonelle avhengigheter vi analyserer oss fram til og de vi kan finne ved å observere data i en tabell på et gitt tidspunkt?

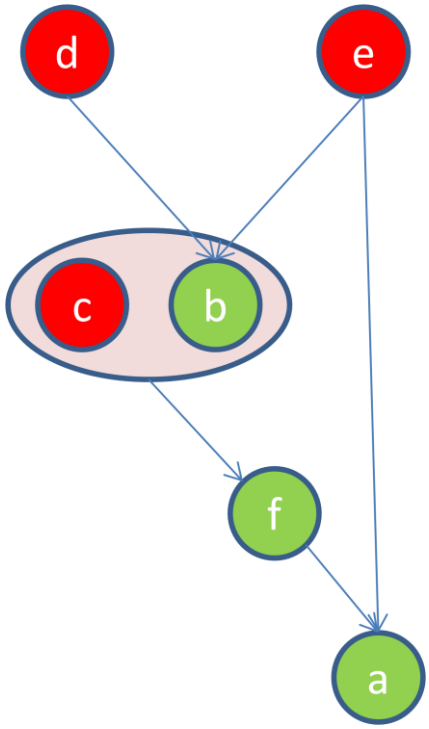
Svar:
De FA vi finner fram til under systemanalysen er intensjoner, de skal gjelde for hele systemets levetid. De vi finner ved å se på data i en tabell på et bestemt tidspunkt er tilfeldig. De må tilfredsstille de FA vi finner i analysen, men tilsynelatende kan det også være mange andre FA, men disse er altså ikke sanne. Feilen blir større dess færre data vi analyserer.

c) Hvordan er definisjonen på BCNF (Boyce Codd Normal Form)?

Svar:
Alle determinanter er nøkler.

d) For den universelle relasjon R(abcdef) har vi følgende funksjonelle avhengigheter:
e→ab, f→a, d→b, cb→f.
Finn nøkkelen(e) til R.

Svar:



Nøkkel er cde. Ingen alternativer.

e) Dekomponer R til tabeller på BCNF.

Svar:

For hver (tapsfri) oppdeling må vi passe på at felles attributter i de to nye relasjoner til sammen inneholder alle attributter og at fellesattributtene (attributter som finnes i begge tabeller) er nøkkel i en av dem. Fortsetter oppdelingen til *alle determinanter er nøkler* - BCNF

$R1 \cap R2 = R1 \mid R2$

R(abcdef)
/ \
R1(bcdef) R2(fa) felles f, f→a

/ \
R11(bcde) R12(bcf) felles bc, bc→f

/ \
R111(cde) R112(db) felles d, d→b, nøkkel i R112

S = R2(fa), R12(bcf) , R112(db), R111(cde)

Det kan være mange korrekte løsninger på denne oppgavetypen. Det avhenger av hvilke attributter vi velger å ta ut i første splitting. En alternativ løsning er:

R(abcdef)
/ \
R1(bcdef) R2(ea) felles e, e→a

/ \
R11(bcde) R12(bcf) felles bc, bc→f

/ \
R111(cde) R112(db) felles d, d→b, nøkkel i R112

S = R2(ea), R12(bcf) , R112(db), R111(cde)

Her ble løsningene veldig like. Derimot hvis vi starter med uttak av eab blir resultatet temmelig ulikt.

R(abcdef)
/ \
R1(cdef) R2(eab) felles e, e→ab

/ \
R11(cde) R12(cdf) felles cd, cd→f, har benyttet en transitiv avhengighet ved at d bestemmer b

S = R2(eab), R12(cdf) , R11(cde)

Denne oppdelingsmetoden bevarer ikke funksjonelle avhengigheter.

Oppgave 4 (20 %)

a) Hva menes med at en utførelsessekvens er henholdsvis:
1) Ikke gjenopprettbar (not recoverable),
2) gjenopprettbar (evt. med galopperende abort (cascading abort)),
3) gjenopprettbar, ikke galopperende abort og
4) strikt?

Svar: Se læreboka.

b) For at en transaksjon skal kunne utføres korrekt må den både være gjenopprettbar og serialiserbar. Vi har følgende historie:

H: r1(x), r2(y), w2(y), r3(z), r1(y), a2, r3(x), r1(z), w1(z), w3(x) …

Ved … er alle gjenværende transaksjoner klare til å komitte. Kan de det?

Svar:

T1	T2	T3
R(X)		
	R(Y)	
	W(Y)	
		R(Z)

R(Y)		
	abort	
		R(X)
R(Z)		
W(Z)		
abort		W(X)
		commit

Svaret er både ja og nei. T1 må abortere fordi den har gjort en ”dirty read”. Når skriver av Y som er T2 aborterer må også T1 abortere, vi har et eksempel på galopperende abort. Tabellen viser konflikterende operasjoner i samme farge. Etter at både T1 og T2 har abortert er det bare en transaksjon igjen. T3 har bare lest komitterte verdier og kan gjøre confirm. Vi får altså ingen serialiseringskonflikt fordi det bare er en transaksjon som har kjørt. De andre er annullert.

c) Strikt tofaselåsing sikrer både gjenopprettbarhet og serialiserbarhet.
Beskriv strikt tofaselåsing.

Svar: I fase 1 kan en bare sette låser, dvs. sette leselås, sette skrivelås eller konvertere leselås til skrivelås hvis ressursen ikke holdes av flere leselåser enn egen transaksjons leselås. I fase 2 kan en bare frigi ressurser. Strikt betyr at alle låser oppheves samtidig i det øyeblikk transaksjonen komitter. Alternativet til strikt er at låsene avgis gradvis i fase to – altså ikke strikt. Tofaselåsing fører til venting på ressurs som er låst, og en kan få flere transaksjoner som venter på hverandre i sirkel og en har vranglås.

d) Hvordan ville utførelsessekvensen blitt for historien H i b) hvis databasen brukte strikt tofaselåsing?

Svar:

T1	T2	T3
R(X)		
	R(Y)	
	W(Y)	
		R(Z)
R(Y)		
venter	abort	
R(y)		
		R(X)
R(Z)		
W(Z)		
venter		W(X)
		venter
VRANGLÅS		

Når det oppstår en konflikt vil transaksjonen som ber om ressurs vente. Enten til det oppstår vranglås eller til at ressursen blir frigitt. Ved vranglås må en av de involverte transaksjoner ofres for å komme videre - systemstyrt abort. Når T1 ber om Y, må den vente. Ved T2s abort frigis Y og første transaksjon i kø som er T1 kan nå få den.
Når T1 skal skrive Z må låsen konverteres fra en delt lås til en eksklusiv lås. Dermed må T1 vente igjen til T3 frigir Z. For å fullføre må T3 skrive X som holdes av T1 og T3 i delt leselås. T1 venter allerede på T3, og vi har en vranglås.

H: r1(x), r2(y), w2(y), r3(z), r1(y), a2, r3(x), r1(z), w1(z), w3(x) …

Modifisert H blir da:

H': r1(x), r2(y), w2(y), r3(z), a2, r1(y), r3(x), r1(z), vranglås

Tofaselåsing kan medføre vranglås. Denne må løses opp ved at en av de involverte transaksjonene blir abortert av databasesystemet. Aborten for T2 er en abort som kommer fra selve transaksjonen, den oppgir seg selv av en eller annen grunn.

kjb/24.07.2010/