Høgskolen i Østfold

EKSAMEN

Emnekode: ITF30307	Emne: Databaseadministrasjon og databasesystemer
Dato: 04.12.13	Eksamenstid: 09.00 - 12.00.
Hjelpemidler: ingen	Faglærer: Edgar Bostrøm / Per O. Bisseberg
Oppgavesettet består av 2 sider. Vedlegget består av 1 side. På mange av oppgavene kan det lønne seg å svare punktvis. I noen tilfeller holder det med en setning eller tre, i andre tilfeller bør det gjøres en beskrivelse/kommentar/drøfting på hvert av disse punktene. Tidsangivelsen pr. oppgave gir indikasjon på hvor mye man bør svare. Hver deloppgave teller likt.	
Sensurdato: 8. januar 2014 Karakterene er tilgjengelige for studenter på studentweb senest dagen etter oppgitt sensurfrist.	

Oppgave 1. Tid: 60 minutter.

a) Hvilke oppgaver har en databaseadministrator? Trekk gjerne inn stoff fra gjesteforeleser (utover det å drikke mye kaffe☺).
Kort: Sammenlign rollen til en databaseadministrator og en dataadministrator.

b) Hva er replikering, og hvilke fordeler og ulemper gir replikering i forhold til ikke-replikerte systemer?

c) Hva menes med recovery av en database? Boka (og delvis også forelesningsnotatene) beskriver ulike teknikker for recovery – forklar!

Oppgave 2. Tid: 60 minutter.

For delspørsmål a) og b) tar vi utgangspunkt i en forenklet ordre-ordrelinje-vare-struktur.

ORDRE
Ordrenr Ordredato Kundenr

VARE
Varenr Varenavn

ORDRELINJE
Ordrenr Varenr Antall

a) Lag utsagn i relasjonsalgebra for:
- Kundenr for den/de som har kjøpt 100 stk. "Grønne spikermatter" i en og samme ordre. Bruk natural join, og slik at du kobler sammen alle tabellene først («innerst»).
- Samme som over, men vi ønsker i stedet å **ha med alt i relasjonen Ordre**, og vi ønsker at det skal gjøres **effektivt**. Tips: det kan også være lurt å bruke noe annet enn natural join (helt eller delvis) for å gjøre utsagnet enklere.

b) Lag utsagn i relasjonsalgebra for:
   - Varenr og varenavn på varer som ikke er solgt i det hele tatt (dvs. at de ikke finnes på noen ordrelinje).
   - Kundenr for de som har kjøpt alle varene som finnes i vare-relasjonen.
   - **Kort:** Læreboka og andre kilder definerer opp en grupperingsoperator. Gi et eksempel på et relasjonsalgebrauttrykk som bruker denne.

c) Begrepene fra relasjonsalgebra kan brukes for å forklare hva som gjøres når man overfører/transformerer data fra et vanlig OLTP-system til et datavarehus. Forklar!

Tips for del a) og b): Se syntaks i vedlegget.

Oppgave 3. Tid: 60 minutter.

a) Forklar og sammenlign virkemåtene til triggere, lagrede prosedyrer og lagrede funksjoner.

b) Utred fordeler og ulemper ved plassering av programlogikk på databasenivå. Hvilke alternativer har vi til plassering av programlogikk?

c) Forelesningene tok opp ulike databasemodeller/former (bl.a. hierarkisk/nettverks-, relasjonsdatabaser og OODB) og satte disse i sammenheng. Forklar, gjerne ved bruk av en tabell. Hvordan passer XML som databasemodell inn i dette mønsteret?

NB: Dropper kriteriet på en join når dette er opplagt (f.eks. PK/FK og korresponderende navn)
NB: Like god løsning om utsagnet er splittet i deler, R1:= …., R2:= …..

a) Lag utsagn i relasjonsalgebra for:

- Kundenr for den/de som har kjøpt 100 stk. "Grønne spikermatter" i en og samme ordre. Bruk natural join, og slik at du kobler sammen alle tabellene først («innerst»).

  $\pi_{\text{kundenr}}(\sigma_{\text{antall = 100}}(\sigma_{\text{varenavn = «Grønne spikermatter»}}(\textbf{Ordre} \triangleright\triangleleft \textbf{Ordrelinje} \triangleright\triangleleft \textbf{Vare})))$

- Samme som over, men vi ønsker i stedet å **ha med alt i relasjonen Ordre**, og vi ønsker at det skal gjøres **effektivt**. Tips: det kan også være lurt å bruke noe annet enn natural join (helt eller delvis) for å gjøre utsagnet enklere.

  $\textbf{Ordre} \triangleright \sigma_{\text{antall = 100}}(\textbf{Ordrelinje}) \triangleright \sigma_{\text{varenavn = «Grønne spikermatter»}}(\textbf{Vare})$. ***Også greit u/ semijoin på høyre del.***

b) Lag utsagn i relasjonsalgebra for:

- Varenr og varenavn på varer som ikke er solgt i det hele tatt (dvs. at de ikke finnes på noen ordrelinje).

  $\textbf{Vare} \triangleright (\pi_{\text{varenr}}(\textbf{Vare}) \setminus \pi_{\text{varenr}}(\textbf{Ordrelinje}))$

  evt

  $\pi_{\text{varenr,varenavn}}(\textbf{Vare} \triangleright\triangleleft (\pi_{\text{varenr}}(\textbf{Vare}) \setminus \pi_{\text{varenr}}(\textbf{Ordrelinje})))$

  evt.

  $\textbf{Vare} \setminus \pi_{\text{varenr, varenavn}}(\sigma_{\text{ordre.varenr is null}}(\text{Vare} \bowtie \text{Ordre}))$. Kunne projisert bort litt på vare og ordre innerst. Enklere, men mindre effektivt: $\textbf{Vare} \setminus \pi_{\text{varenr, varenavn}}(\textbf{Vare} \bowtie \textbf{Ordre})$.

- Kundenr for de som har kjøpt alle varene som finnes i vare-relasjonen.

  Normalt ville vi tenkt noe med Ordrelinje / Vare for å få de ordrelinjene som innehold alle varer. Siden vi skal ha med kunder må vi først joine Ordre og Ordrelinje for å få fram kundene.

  $\pi_{\text{kundenr, varenr}}(\textbf{Ordre} \triangleright\triangleleft \textbf{Ordrelinje}) / \pi_{\text{varenr}}(\textbf{Vare})$

- **Kort:** Læreboka og andre kilder definerer opp en grupperingsoperator. Gi et eksempel på et relasjonsalgebrauttrykk som bruker denne.

  Å lage denne ut fra lærebokas syntaks var en del av oblig 2. Annen syntaks, f.eks. fra nettsteder er ok. Det følgende er direkte kopi fra løsningsforslag, men på en oppgave de nok neppe tenkte at de fikk.

  $\rho_{R}(\textbf{prosjektnr,prosjektlinjer,prosjekttimer})\ {}_{\text{prosjektnr}}\Im_{\text{COUNT prosjektnr, SUM timetall}}(\textbf{PrAns})$

c) Begrepene fra relasjonsalgebra kan brukes for å forklare hva som gjøres når man overfører/transformerer data fra et vanlig OLTP-system til et datavarehus. Forklar!

De mest typiske transformasjonene som gjøres fra et OLTP til OLAP, er

- ***å overføre enkelte rader, dvs. en restriksjon, σ***
- ***å overføre enkelte kolonner, men ikke alle, dvs. en projektsjon π***
- ***å koble sammen tabeller, f.eks. i forbindelse med laging av stjerneskjemaer, vi bruker $\triangleright\triangleleft$ eller andre join-operasjoner***
- ***En union av relasjoner, f.eks. dersom man vil lage et DW bl.a. fra to firmaer som har slått seg sammen, men har samme struktur, $\cup$.***
- ***Tilsvarende også med snitt, $\cap$, og mengdedifferanse,$\setminus$, f.eks. ta med alle kunder som ikke har kjøpt noe de siste 2 årene.***
- ***Også andre operatorer som er gjennomgått kan være aktuelle.***
- ***å summere eller gjøre annen gruppering av data, $\Im$***

En typisk transformasjon kan dermed skrives som et relasjonsalgebrautsagn med en eller flere av disse operatorene. Siden det er prosydurelt, kan man i motsetning til SQL beskrive rekkefølgen som et mer komplisert utsagn skal utføres på. Det kan være en fordel, andre ganger er optimizeren bedre.
En kortere, uten opplisting av alle disse operasjonene er også greit, men bør ha med konkret sammenheng mellom operatorer, eller i alle fall begrep, og transformasjonsprosesssen.

Til del 3c:
Dette var en del av forelesningsnotatene som ble gitt. Forelesningen tok opp 4 typer data, og satte disse i sammenheng med ulike databaseformer. Data og metadata finnes i alle. Som vi ser, i siste figur, utspenner de to siste alle kombinasjoner av Data som instrukser (i.e. programmer) og pekere til data. Dette kan fungere som en god kategorisering.
NB! Jeg har lyst en gang til å skrive en artikkel om dette, så ikke bruk dette videre i en slik sammenheng. Å bruke det i en forelesning er selvsagt greit.

~~DB-modell~~ → ↓ Type data	Hierarkisk/ nettverk	Relasjons	Relasjons- m/ triggere/SP	OO/OR
Data	✓	✓	✓	✓
Data som beskriver data	✓	✓	✓	✓
Data som instruksjoner			✓	✓
Pekere til data	✓[1]			✓

Figur 1 Karakterisering av ulike databaseformer.

Noen poenger: Man kan karakterisere relasjonsdatabaser i forhold til andre databasemodeller bl.a. ved å si at "there are no pointers" ([Date, 2004 s. 63]). Dermed er den eneste muligheten at sammenhengen mellom data i flere tabeller[2] også må være via vanlige data (dvs. fremmednøkler), ikke pekere. Figuren viser også at "rene" relasjonsdatabaser pr. definisjon er den enklest mulige databaseformen (ingen pekere, ingen eksekverbare strukturer). Dette har vist seg å ha en rekke fordeler. Sammen med enkelhet i metadata (databaseskjemaet) og i spørrespråket, samt et sunt teorigrunnlag, er dette antagelig hovedgrunnen til at rene relasjonsdatabaser fremdeles er dominerende.

Når både metadata og spørrespråket blir mer komplisert i et OO/OR-system, har det sammenheng med ønsket om å kunne mappe en komplisert virkelighet på en bedre måte, ikke å gjøre det mer innviklet i og for seg. At OO/OR også har data som er instruksjoner, er ikke noe annet enn at man kan legge funksjoner/metoder/operasjoner inn som en del av skjema. Relasjonsdatabasesystemer med mulighet for triggere og lagrede funksjoner/prosedyrer (SP-er) vil tilsvarende ha data som er instruksjoner som en integrert del av sin databasemodell.

Som nevnt inneholder alle databaseformene data og metadata. En interessant observasjon i denne sammenhengen er: figuren over viser at **alle kombinasjoner** av de to gjenværende (data som instruksjoner og pekere til data) er utnyttet på ulik måte i databasemodeller. Dermed vil disse **databasemodellene spenne ut rommet over muligheter som finnes** med de 4 kategorier av data som vi beskriver i artikkelen.

~~DB-modell~~ → ↓ Type data	Hierarkisk/ nettverk	Relasjons	Relasjons- m/ triggere/SP	OO/OR
Data som instruksjoner			✓	✓
Pekere til data	✓			✓

I forhold til XML vil det være naturlig med en diskusjon om at det strengt tatt er hierarkisk, dokumentbasert, at metadata ikke behøver å være så strukturert, taggingstenkning, DTD/Skjema, XPointer m.m.

[1] Hierarkiske databaser trenger i praksis pekere for å kunne fungere.
[2] evt. internt i samme tabell, såkalte egenrelasjoner/rekursive relasjoner..

Relasjonsalgebra - vanlige operasjoner.

Mengdeoperasjoner:	*Notasjon, variant 1*	*Notasjon, variant 2*
Union	$R \cup S$	R union S
Snitt	$R \cap S$	R intersect S
Mengdedifferanse	R - S R \ S	R difference S R minus S
Mengdeprodukt, kartesisk produkt ("alle mot alle")	$R \times S$	R product S R times S
Spesielt for relasjoner:		
Horisontalt utvalg	$\sigma_{<beting.>}(R)$	R where <bet.> R where <bet.>
Vertikalt utvalg	$\pi_{<feltliste>}(R)$	R[<feltliste>]
Mengdedivisjon. (Gitt R[c,d] og S[d]. c er med i mengden R dividert med S hvis c i R forekommer sammen med alle d-er som finnes i S.)	$R \div S$ R / S	R divideby S
Spesialiteter av produkt:		
θ-join (produkt med en eller annen betingelse på kompatible attributter, f.eks. >, <, og kombinasjoner	$R \bowtie_{<bet.>} S$	R join$_{<betingelse>}$ S (R join S) where <bet.>
Equi-join (θ-operasjonen er =)	som over	som over
Natural join (Equi-join hvor felles attributt kommer bare en gang) ** den mest vanlige jointypen **	som over	som over
Varianter for produkt:		
Outer join, normalt venstre. (alle i R, samt alle fra S som oppfyller koblingsbetingelsen)	$R \supset\!\bowtie S$	R left join$_{<bet.>}$ S
Full join (alle i R, alle i S, samt alle som oppfyller koblingsbet.)	$R \supset\!\subset S$	R full join$_{<bet.>}$ S
Semijoin (de i R som tilfredsstiller R join$_{<betingelse>}$ S)	$R \triangleright_{<bet.>} S$	R semijoin$_{<bet.>}$ S

Legg merke til at operasjonene her er på mengder, slik at evt. duplikater tas bort – tilsvarende select distinct i SQL.

Dersom betingelsen er på entydige primær/fremmednøkkelkombinasjoner, droppes ofte <bet>.