UNIVERSITETET I OSLO

Relasjonsdatabasedesign

Oppdateringsanomalier

Dekomponering

Normalformer

Hva kjennetegner god relasjonsdatabasedesign?

- Beslektet informasjon legges i samme skjema
  - Ubeslektet informasjon legges *ikke* i samme skjema
  - Brudd på dette vil ofte medføre dobbeltlagring av data og dermed forårsake ***oppdateringsanomalier***
- Så lite dobbeltlagring som mulig
  - Plassbehovet minimaliseres
  - Oppdatering forenkles
- Så få «glisne» relasjoner som mulig
  - Plassbehovet minimaliseres
  - Unngår problemer med håndtering av nil-verdier
- Korrekt totalinformasjon kan gjenskapes nøyaktig
  - Ingen falske data genereres

Eksempel: Grossistdatabase versjon 1

Produkt(Kode, Produktnavn, Produsent, #Enheter)

Bestilling(Kode, Kundenr, Navn, Adresse, #Bestilt)

Skranker:

1. Alle verdier i Kode i Produktskjemaet skal være unike
2. For hvert par av Kundenr, Kode i Bestillingskjemaet skal det være bare en mulig verdi av #Bestilt
3. Verdien i Kundenr bestemmer verdiene i Navn og Adresse
4. Kode i Bestillingskjemaet er fremmednøkkel til (refererer til) verdier i Kode i Produktskjemaet

Grossistdatabase versjon 1 med integritetsregler

Produkt(Kode, Produktnavn, Produsent, #Enheter)

Bestilling(Kode, Kundenr, Navn, Adresse, #Bestilt)

Integritetsregler:

1. Kode→Produktnavn Produsent #Enheter (i Produktskjemaet)
2. Kundenr Kode→#Bestilt (i Bestillingskjemaet)
3. Kundenr→Navn Adresse (i Bestillingskjemaet)
4. Kode i Bestilling-skjemaet er fremmednøkkel til Produkt

   Merk at {Kode} er en kandidatnøkkel i Produkt
   Den er den eneste slike i Produkt og blir derfor primærnøkkel

   Merk at 2 og 3 gir at {Kode, Kundenr} er supernøkkel i Bestilling
   Dette er dessuten en kandidatnøkkel, og den eneste slike i Bestilling, og den blir derfor primærnøkkel

Eksempelekstensjon Bestilling

Bestilling

Kode	Kundenr	Navn	Adresse	#Bestilt
1	1	A	a	3
2	1	A	a	8
1	2	B	b	2

Sekundær informasjon

«Navn» og «Adresse» er eksempler på ***sekundær informasjon***

Dette er nyttig informasjon om kundene, men den er unødvendig i en tabell over bestillingene

Oppdateringsanomalier

- **Innsettingsanomalier**
  - Opprettholde konsistente verdier
  - Håndtere sekundær informasjon
  - Håndtere nil i kandidat- og fremmednøkler
- **Slettingsanomalier**
  - Unngå tap av sekundær informasjon
- **Modifiseringsanomalier**
  - Opprettholde konsistente verdier
  - Oppdatere sekundær informasjon

Hvordan unngå oppdateringsanomalier

- Unngå/fjern oppdateringsanomalier og dobbeltlagring ved å splitte (dekomponere) relasjonene slik at dobbeltlagring blir borte!

Dekomposisjon av et relasjonsskjema

- Gitt et relasjonsskjema $R(A_1,A_2,\ldots,A_n)$.
  En ***dekomposisjon*** $D = \{R_1,R_2,\ldots,R_m\}$ av R er en samling med skjemaer $R_1,R_2,\ldots,R_m$ som er slik at følgende to krav holder:
  - alle attributtene i hver R_k er også attributt i R
  - samtlige attributter $A_1,A_2,\ldots,A_n$ kan gjenfinnes i minst ett av skjemaene $R_1,R_2,\ldots,R_m$

Eksempel: Grossistdatabase versjon 2

Produkt(Kode, Produktnavn, Produsent)
Kunde(Kundenr, Navn, Adresse)
Bestilling(Kode, Kundenr, #Bestilt)

Integritetsregler:

- Kode i Bestilling er fremmednøkkel til Produkt
- Kundenr i Bestilling er fremmednøkkel til Kunde

Eksempelekstensjoner Kunde, Bestilling

Ny modell

Kunde

Kundenr	Navn	Adresse
1	A	a
2	B	b

Bestilling

Kode	Kundenr	#Bestilt
1	1	3
2	1	8
1	2	2

Gammel modell

Bestilling

Kode	Kundenr	Navn	Adresse	#Bestilt
1	1	A	a	3
2	1	A	a	8
1	2	B	b	2

Naturlig join

- Vi ønsker å kunne rekonstruere den opprinnelige ekstensjonen
- ***Naturlig join*** er en operasjon der to relasjoner slås sammen til en ved at to og to tupler pares hvis og bare hvis de har like verdier i attributter med likt navn

Kunde⋈Bestilling

Kunde

Kundenr	Navn	Adresse
1	A	a
2	B	b

Bestilling

Kode	Kundenr	#Bestilt
1	1	3
2	1	8
1	2	2

Kunde ⋈ Bestilling

Kundenr	Navn	Adresse	Kode	#Bestilt
1	A	a	1	3
1	A	a	2	8
2	B	b	1	2

Eksempel: Grossistdatabase, versjon 3

Produkt(<u>Kode</u>, Produktnavn, Produsent)
Kunde(<u>Kundenr</u>, Navn, Adresse)
Koderegister(<u>Kode, Kundenr</u>)
Antall(<u>Kundenr, #Bestilt</u>)

Integritetsregler:

- Kode i Koderegister er fremmednøkkel til Produkt
- Kundenr i Koderegister er fremmednøkkel til Kunde
- Kundenr i Antall er fremmednøkkel til Kunde

Eksempelekstensjoner Kundereg, Bestilling

Koderegister

Kode	Kundenr
1	1
2	1
1	2

Antall

Kundenr	#Bestilt
1	3
1	8
2	2

Koderegister ⋈ Antall

Kode	Kundenr	#Bestilt
1	1	3
~~1~~	~~1~~	~~8~~
2	1	8
~~2~~	~~1~~	~~3~~
1	2	2

Naturlig join på de to tabellene gir flere tupler enn i den opprinnelige tabellen!
= **Falske tupler**

Retningslinjer for dekomposisjon av relasjoner

- **Motivasjon**: Fjerne oppdateringsanomalier
- **Teknikk**: Dekomponere problemrelasjoner
- **Betingelse**: Opprinnelig ekstensjon skal alltid kunne rekonstrueres nøyaktig ved naturlig join
- **Problemer**:
  - Hvordan vurdere objektivt om en samling relasjoner er god/dårlig?
  - Hvordan sikre at en dekomposisjon aldri gir falske tupler?

Normalformer

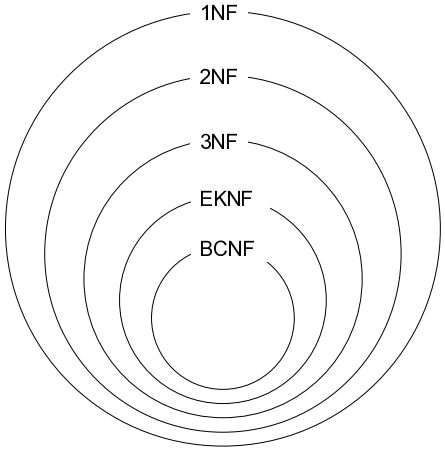
- Normalformer er et uttrykk for hvor godt vi har lykkes i en dekomposisjon
- Jo høyere normalform, jo færre oppdateringsanomalier
- Det fins algoritmer for å omforme fra lavere til høyere normalformer
- Det fins algoritmer for å sjekke om en dekomposisjon er ***tapsfri***, dvs. at naturlig join aldri vil gi falske tupler

Utgangspunkt for normalformene 1NF-BCNF

- Alle integritetsregler er i form av FDer

  (i tillegg til domeneskranker og fremmednøkler)

Normalformer, oversikt



Første normalform

- **Definisjon 1NF** (Codd 1972):
  - Alle domener består av atomære verdier
  - Funksjonsverdien til et tuppel for et gitt attributt skal være en slik atomær verdi (eller nil)
- Alle relasjoner er automatisk på 1NF

Andre normalform

- Repetisjon: En FD $X \rightarrow Y$ er ikketriviell hvis Y ikke er inneholdt i X, dvs. hvis $Y - X \neq \varnothing$
- **Definisjon 2NF** (Codd 1972):
  En relasjon R er på ***andre normalform*** hvis alle ikketrivielle FDer i R på formen $X \rightarrow A$, der X er en mengde attributter og A et attributt i R, tilfredsstiller minst ett av følgende:
  i. X er en supernøkkel i R
  ii. A er et nøkkelattributt i R
  iii. $X \not\subset K$ for noen kandidatnøkkel K i R

Egenskaper ved 2NF

- 2NF⊆1NF siden alle relasjoner er 1NF
- Når er en relasjon 1NF, men ikke 2NF?
  Svar: Når det fins en ikketriviell FD X→A og en kandidatnøkkel K hvor X⊂K, X ≠ K og A ikke er et nøkkelattributt
  Eks: R(A,B,C,D), F={BC→D, C→A}
  Dekomposisjon: R_1(A,C), R_2(B,C,D)
- 2NF er mest av historisk interesse;
  vi gjør sjelden feil som bryter 2NF

Tredje normalform

- **Definisjon 3NF** (Codd 1972):
  En relasjon R er på **tredje normalform** hvis alle ikketrivielle FDer i R på formen X→A tilfredsstiller minst ett av følgende:
  i. X er en supernøkkel i R
  ii. A er et nøkkelattributt i R

Egenskaper ved 3NF

- 3NF⊆2NF fordi kravene til 3NF er en skjerping av kravene til 2NF
- Når er en relasjon 2NF, men ikke 3NF?
  Svar: Når det fins en ikketriviell FD X→A hvor X ikke er en supernøkkel og A ikke er et nøkkel-attributt og X⊄K for noen kandidatnøkkel K
  Eks: R(A,B,C), F={A→BC, B→C}
  Dekomposisjon: R_1(A,B), R_2(B,C)
- Også 3NF er lett å oppnå

Elementære FDer og nøkler

- En FD X→A kalles *elementær* dersom
  - A er et atributt
  - X→A ikke er triviell
  - X er minimal
    (dvs. at hvis Y⊆X og Y→A, så er Y=X)
- En kandidatnøkkel K kalles *elementær* hvis det finnes en elementær FD K→B i R

Elementary Key Normal Form

- **EKNF** (Zaniolo 1982) er ikke pensum, men er tatt med for fullstendighets skyld:
  En relasjon R er på ***EKNF*** hvis alle ikketrivielle FDer i R på formen X→A tilfredsstiller minst ett av følgende:
  i. X er en supernøkkel i R
  ii. A er et attributt i en elementær kandidatnøkkel i R

Egenskaper ved EKNF

- EKNF⊆3NF fordi kravene til EKNF er en skjerping av kravene til 3NF
- Når er en relasjon 3NF, men ikke EKNF?
  Svar: Når det fins en ikketriviell FD X→A hvor X ikke er en supernøkkel og A er et nøkkel-attributt i en ikke-elementær kandidatnøkkel
  Eks: R(A,B,C), F={C→A,A→C}
  (Her kan A=epostadresse, B=emnekode og C=fødselsnummer på studenter)
  Mulig dekomposisjon: R_1(A,B), R_2(A,C)

Boyce-Codd Normalform

- **Definisjon BCNF** (Boyce & Codd 1974):
  En relasjon R er på ***Boyce-Codd normalform*** hvis alle ikketrivielle FDer i R på formen X→A tilfredsstiller følgende:
  i. X er en supernøkkel i R

Egenskaper ved BCNF

- BCNF⊆EKNF fordi kravene til BCNF er en skjerping av kravene til EKNF
- Når er en relasjon EKNF, men ikke BCNF?
  Svar: Når det fins en ikketriviell FD X→A hvor X ikke er en supernøkkel og A er et attributt i en elementær kandidatnøkkel
  Eks: R(A,B,C), F={AB→C, C→A}
  Dekomposisjon: R_1(B,C), R_2(A,C)
- Merk at FDen AB→C går på tvers av R_1 og R_2
  (Dvs. at vi må ta en join av R_1 og R_2 før vi kan teste om AB→C)

Oppsummering normalisering

- Brudd på normalformer gir opphav til dobbeltlagring
- Jo høyere normalform, desto mindre dobbeltlagring
- BCNF: Alle FDer er i form av kandidatnøkler.
  Dvs. ingen dobbeltlagring *innen skjemaet*.
  Men: Noen FDer kan gå på tvers av skjemaer
- EKNF (3NF): Ingen FDer går på tvers av skjemaer, men det kan være noen FDer innen skjemaer som gir opphav til dobbeltlagring
- Krav til dekomposisjon: Den må være ***tapsfri***, dvs. aldri kunne gi opphav til falske tupler
- Det er en fordel om dekomposisjonen også er ***FD-bevarende***
  (dvs. at ingen FDer går på tvers av skjemaer)

Hvorfor BCNF ikke alltid er gunstig

- Dersom vi har en dekomposisjon som ikke er FD-bevarende, må vi foreta en join mellom skjemaene i forbindelse med oppdateringer for å sjekke at integritetsreglene overholdes
- Vi har valget mellom
  Enten: Gå tilbake til 3NF (EKNF) og bryte BCNF, da får vi dobbeltlagring og må ta hensyn til dette under oppdatering
  Eller: Beholde BCNF og foreta join mellom relasjoner for å sjekke at FDer overholdes ved oppdateringer

Det fins algoritmer som…

- Tester om en dekomposisjon er tapsfri
- Tester om en dekomposisjon er FD-bevarende
- Dekomponerer skjemaer tapsfritt og FD-bevarende
  - Garanterer EKNF, men gir ikke alltid BCNF
- Dekomponerer skjemaer tapsfritt til BCNF
  - Er ikke alltid FD-bevarende