

INF1300

Introduksjon til databaser

Dagens tema:

- **Oppdateringsanomalier**
- **Normalformer**

Hva kjennetegner god relasjonsdatabasedesign?

- Relasjonene samler beslektet informasjon
- Så lite dobbeltlagring som mulig
- Så få "glisne" relasjoner som mulig
- Korrekt totalinformasjon kan gjenskapes nøyaktig ved join

Eksempel:

Grossistdatabase versjon I (GDB1)

Produkt(Kode, Produktnavn, Produsent, #Enheter)

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

Integritetsregler i tillegg til primærnøklerne:

- A. Verdiene i Navn og Adresse er funksjonelt avhengige av verdien i Kundenr
- B. Kode i Bestilling er fremmednøkkel til Kode i Produkt

➤ Relasjonene samler beslektet informasjon:

- Tekstlig nærhet gjenspeiler logisk nærhet (Med tekstlig nærhet menes her samlokalisering i en relasjon)
- Brudd på dette prinsippet har en tendens til å påtvinge duplisering av data og dermed forårsake oppdateringsanomalier

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

Eksempelinstans 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

➤ Så lite dobbeltlagring som mulig:

- Plassbehovet minimaliseres
- Oppdatering forenkles

➤ Så få "glisne" relasjoner som mulig:

- Plassbehovet minimaliseres
- Unngår problemer med hvordan join på nil-verdier skal håndteres
- Unngår problemer med hvordan aggregeringsfunksjoner skal håndtere nil-verdier

Hvordan unngå dobbeltlagring

- Splitt (dekomponer) relasjonene slik at dobbeltlagring blir borte!
 - Instanser for de dekomponerte relasjonene fremkommer fra opprinnelig instans ved projeksjon

Eksempel:

Grossistdatabase versjon 2 (GDB2)

Produkt(Kode, Produktnavn, Produsent, #Enheter)

Kunde(Kundenr, Navn, Adresse)

Ordre(Kode, Kundenr, #Bestilt)

Integritetsregler i tillegg til primærnøkklene:

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

Eksempelinstanser

Kunde, Ordre

Ny
modell

Kunde

Kundenr	Navn	Adresse
1	A	a
2	B	b

Ordre

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

Krav til dekomposisjoner

- Vi ønsker å kunne rekonstruere den opprinnelige instansen
- Dekomposisjon av relasjoner må derfor gjøres på en måte som sikrer at vi alltid kan gjenskape den opprinnelige instansen ved **naturlig join**

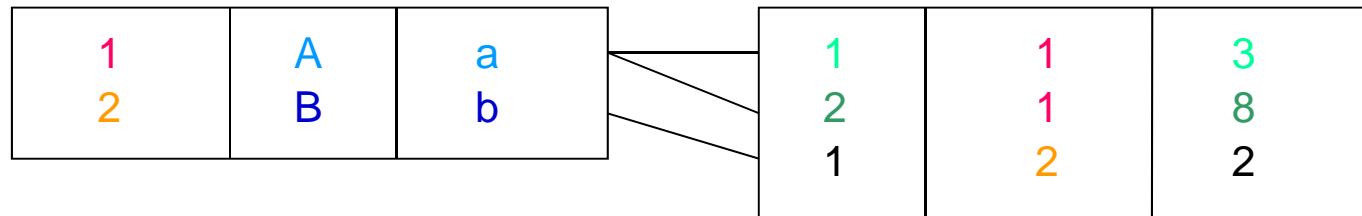
Kunde ⋈ Ordre

Kunde

Kundenr	Navn	Adresse
1	A	a
2	B	b

Ordre

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



Kunde ⋈ Ordre = Bestilling

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

1	A	a	1	3
1	A	a	2	8
2	B	b	1	2

Eksempel:

Grossistdatabase, versjon 3 (GDB3)

Produkt(Kode, Produktnavn, Produsent, #Enheter)

Kunde(Kundenr, Navn, Adresse)

Koderegister(Kode, Kundenr)

Antall(Kundenr, #Bestilt)

Integritetsregler:

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

Eksempelinstanser

Koderegister, Antall

Koderegister

Kode	Kundenr
1	1
2	1
1	2

Antall

Kundenr	#Bestilt
1	3
1	8
2	2

1	1	⊗	1	3
2	1		1	8
1	2		2	2

Koderegister ⊗ Antall ≠ Ordre!

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

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**

➤ Korrekt totalinformasjon kan gjenskapes nøyaktig ved join:

- Ingen falske tupler genereres

Normalformer

- **Problem:** Hvordan vurdere objektivt om en samling relasjoner er god/dårlig?
- **Normalformer** er et uttrykk for hvor godt vi har lykkes i en dekomposisjon
- At et skjema er på en normalform sikrer at visse typer dobbeltlagring ikke forekommer
- Jo høyere normalform, jo mindre dobbeltlagring

Litt definisjoner (gamle og nye)

I de videre definisjonene tenker vi oss gitt en relasjon $R(A_1, A_2, \dots, A_n)$ **med tilhørende integritetsregler.**

- La $X \subseteq R$ bety at $X \subseteq \{A_1, A_2, \dots, A_n\}$.
- Hvis t er et tuppel i en instans av R , betegner $t[X]$ verdiene i t 's X -attributter.

Integritetsregler

- Integritetsregler begrenser mengden av lovlige instanser for et databaseskjema.
 - **Primærnøkler** uttrykker én type integritetsregler.
 - Primærnøkler er spesialtilfeller av **kandidatnøkler**.
 - Kandidatnøkler er spesialtilfeller av **funksjonelle avhengigheter**.
 - (I tillegg fins andre typer integritetsregler.)

Nøkler

- X er en **supernøkkel** i R : $X \subseteq R$, og ingen instans av R får inneholde to forskjellige tupler t_1 og t_2 hvor $t_1[X] = t_2[X]$.
- X er en **kandidatnøkkel** i R : X er en supernøkkel i R , og for alle $A \subset X$ er A *ikke* en supernøkkel i R .
(Dvs. X er en minimal supernøkkel.)
- X er en **primærnøkkel** i R : X er en spesielt utpekt kandidatnøkkel i R .

Nøkkelattributt

- Et **nøkkelattributt** er et attributt som er med i en kandidatnøkkel.
- Et **ikke-nøkkelattributt** er et attributt som ikke er med i noen kandidatnøkkel.

Funksjonell avhengighet (repetisjon)

- Gitt en relasjon R og skranker for R , og gitt $X, Y \subseteq R$.
 - Y er **funksjonelt avhengig av** X hvis vi for enhver lovlig instans av R har at hvis instansen inneholder to tupler t_1 og t_2 hvor $t_1[X] = t_2[X]$, så må $t_1[Y] = t_2[Y]$.
 - I så fall skriver vi $X \rightarrow Y$.
- Ofte snakker vi for korthets skyld om «FDen $X \rightarrow Y$ » (der **FD** står for Functional Dependency)
- Vi sier at « Y følger av X », eller at « X bestemmer Y »
- Integritetsregelen $X \rightarrow A_1 A_2 \dots A_k$ kan alternativt representeres ved k FDer $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ (hvor høyresidene består av bare ett attributt).

Funksjonell avhengighet og kandidatnøkler

- Merk at hvis X er en supernøkkel, så holder $X \rightarrow Y$ for enhver Y .
 - Så hvis X er en kandidatnøkkel, holder $X \rightarrow Y$ for enhver Y .
- Omvendt: Hvis $X \rightarrow Y$ for enhver Y , så er X en supernøkkel.
- Spesielt betyr dette at hvis vi angir at $R(A_1, A_2, \dots, A_n)$ har en kandidatnøkkel X , betyr det at R har FDen $X \rightarrow A_1 A_2 \dots A_n$.
 - Hvis $R(A_1, A_2, \dots, A_n)$ har en primærnøkkel X , betyr det at R har FDen $X \rightarrow A_1 A_2 \dots A_n$.

GDB1 med integritetsregler

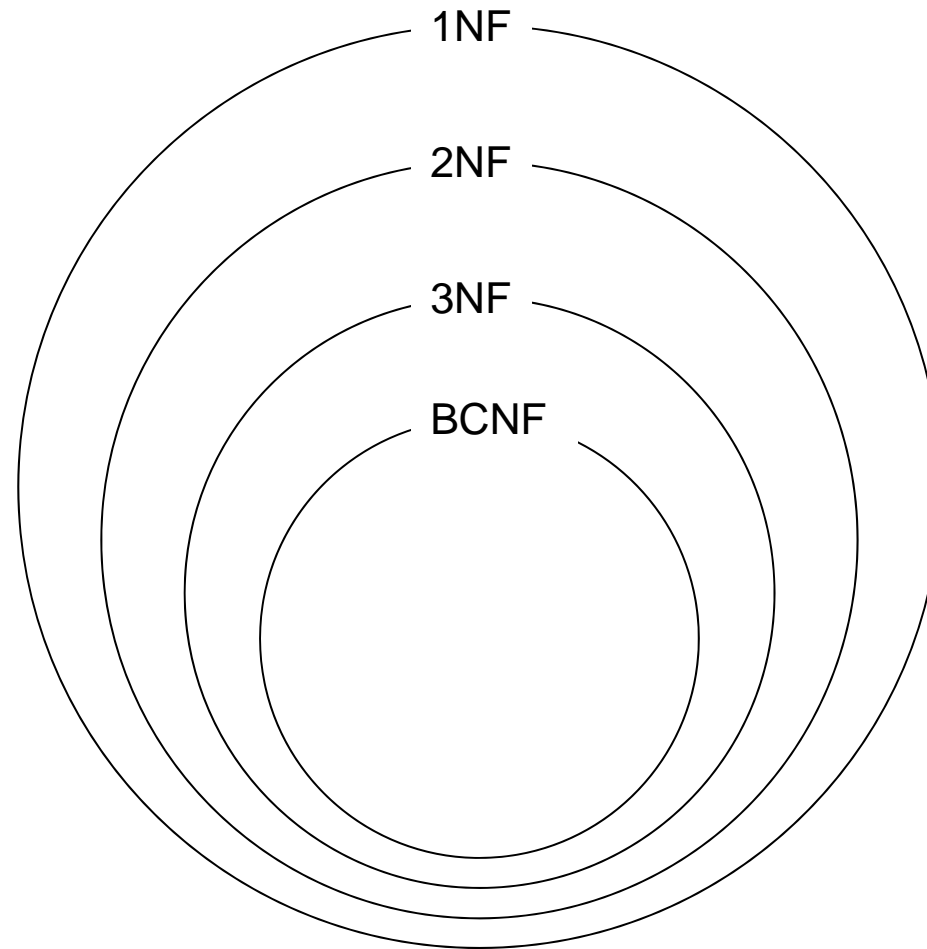
Produkt(Kode, Produktnavn, Produsent, #Enheter)

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

Integritetsregler:

1. **Kode** → **Produktnavn, Produsent, #Enheter** (i Produkt)
fordi Kode er primærnøkkel i Produkt
2. **Kode, Kundenr** → **Navn, Adresse, #Bestilt** (i Bestilling)
fordi (Kode, Kundenr) er primærnøkkel i Bestilling
3. **Kundenr** → **Navn, Adresse** (i Bestilling)
fordi verdiene i Navn og Adresse er funksjonelt avhengige av verdien i Kundenr (integritetsregel A på lysark 3)
4. Kode i Bestilling er fremmednøkkel til Produkt
(integritetsregel B på lysark 3)

Normalformer, oversikt



Utgangspunkt for normalformene

1NF-BCNF

Alle integritetsregler er i form av FDer

(i tillegg til domeneskranker og fremmednøkler)

Første normalform

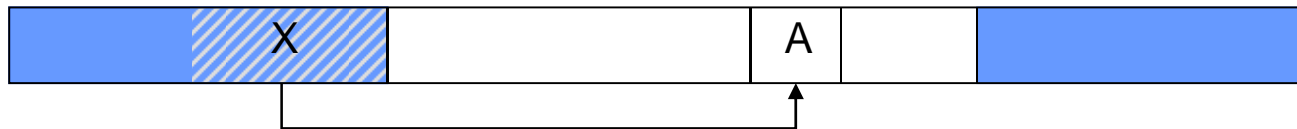
- En relasjon er 1NF hvis det bare er tillatt med atomære verdier i attributtene

Andre normalform

- En relasjon R er 2NF hvis enhver ikke-triviell FD $X \rightarrow A$ (hvor $X \subseteq R$ og $A \in R$) tilfredsstiller minst ett av følgende tre krav:
 - X er en supernøkkel
 - A er et nøkkelattributt
 - $X \not\subseteq W$ for noen kandidatnøkkel W i R
- R bryter 2NF hvis det fins en ikke-triviell FD $X \rightarrow A$ hvor A er et ikke-nøkkelattributt og det fins en kandidatnøkkel W slik at $X \subset W$ (ekte delmengde).

Brudd på andre normalform

FD $X \rightarrow A$ hvor X utgjør noen av, men ikke alle, attributtene i en av kandidatnøkklene og A er et ikke-nøkkelattributt.



X er skravert.

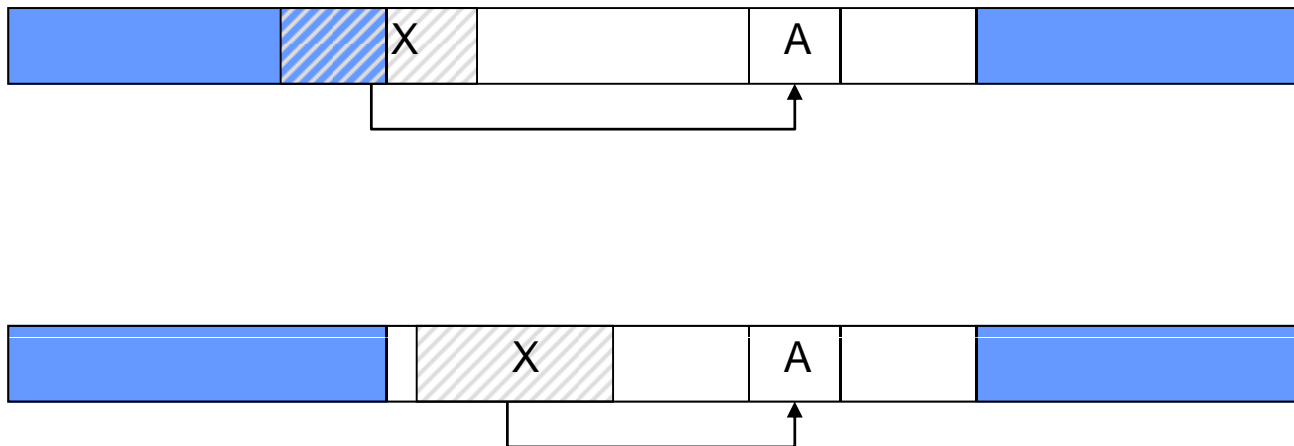
Kandidatnøkler er markert med lyseblått.

Tredje normalform

- En relasjon R er 3NF hvis enhver ikke-triviell FD $X \rightarrow A$ tilfredsstiller minst ett av følgende to krav:
 - X er en supernøkkel
 - A er et nøkkelattributt
- R bryter 3NF hvis det fins en ikke-triviell FD $X \rightarrow A$ hvor A er et ikke-nøkkelattributt og X ikke er en supernøkkel.

Brudd på tredje, men ikke andre, normalform

FD $X \rightarrow A$ hvor (noen av) attributtene i X er ikke-nøkkelattributter, eventuelle nøkkelattributter i X ikke omfatter noen fullstendig kandidatnøkkel, og A er et ikke-nøkkelattributt.

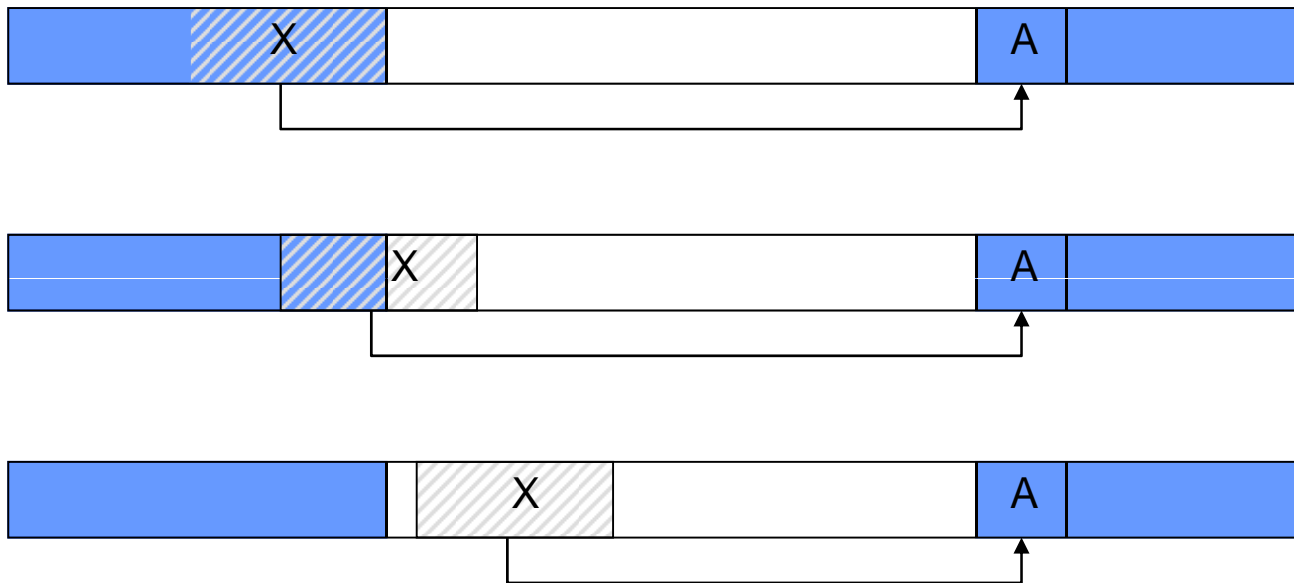


Boyce-Codd normalform

- En relasjon R er BCNF hvis enhver ikke-triviell FD $X \rightarrow A$ tilfredsstiller følgende krav:
 - X er en supernøkkel
- R bryter BCNF hvis det fins en ikke-triviell FD $X \rightarrow A$ hvor X ikke er en supernøkkel.

Brudd på Boyce-Codd, men ikke tredje, normalform

FD $X \rightarrow A$ hvor A er et nøkkelattributt og X ikke inneholder en kandidatnøkkel.



Brudd på normalformer i GDB1

Produkt(Kode, Produktnavn, Produsent, #Enheter)

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

FDer:

1. **Kode** → **Produktnavn, Produsent, #Enheter** (i Produkt)
fordi Kode er primærnøkkel
2. **Kode, Kundenr** → **Navn, Adresse, #Bestilt** (i Bestilling)
fordi (Kode, Kundenr) er primærnøkkel
3. **Kundenr** → **Navn, Adresse** (i Bestilling)
fordi verdiene i Navn og Adresse er funksjonelt avhengige av verdien i Kundenr (integritetsregel A på lysark 3)
Denne bryter 2NF (se lysark 29), så Bestilling er på 1NF, men ikke 2NF.

Normalisering

- **Normalisering** går ut på å dekomponere relasjoner med lav normalform til relasjoner med høyere normalform.
- **Regel:** Gitt en relasjon $R(XYZ)$ med en FD $X \rightarrow Y$. Hvis R dekomponeres til $S_1(XY)$, $S_2(XZ)$, vil vi *aldri* kunne få falske tupler ved naturlig join av S_1 og S_2 .
 - Hvis vi dekomponerer $R(XYZ)$ til $S_1(XY)$, $S_2(XZ)$ og det *ikke* er slik at $X \rightarrow Y$ holder, vil vi generelt få falske tupler ved naturlig join av S_1 og S_2 .

Dekomponering av GDB1

- Bruk regelen til å dekomponere GDB1 (lysark 3):

Relasjon: Bestilling(Kode, Kundenr, Navn, Adresse, #Bestilt)
FD: Kundenr → Navn, Adresse

X = {Kundenr}
Y = {Navn, Adresse}
Z = {Kode, #Bestilt}

Da blir resultatet GDB2 (lysark 10) :

Kunde(Kundenr, Navn, Adresse)
Ordre(Kode, Kundenr, #Bestilt)

- I GDB2 er det ingen brudd på normalformene; alle relasjonene er på BCNF.

Dekomponering av GDB2

- I dekomponeringen av GDB2 til GDB3 (lysark 14), blir ikke regelen fulgt. Da får vi en database som gir falske tupler, så GDB3 har ikke samme egenskaper som GDB2 og representerer derfor en annen modell av virksomhetsområdet enn den vi ønsket oss.

Normalisering – begrensninger

- Det lar seg alltid gjøre å normalisere til 3NF uten at noen funksjonelle avhengigheter går på tvers av relasjonene
- Det lar seg ikke alltid gjøre å normalisere til BCNF uten at noen funksjonelle avhengigheter går på tvers av relasjonene