

# Oppsummering

Thomas Lohne Aanes

Thomas Amble

14.11.04

# Kapittel 2: Data Modell

**Mål:** Data som skal brukes av applikasjonen blir spesifisert på en formell og likevel intuitiv måte.

**Resultat:** Vi får et konseptuelt skjema som på en lettfattelig måte viser tilgjengelig kunnskap om applikasjonsdataene.

## Essensielle ingredienser i ER-modellen:

**Entitet:** Kontainer av strukturert data.

**Attributt:** Beskriver entiteter. Uttrykker en egenskap til en entitet.

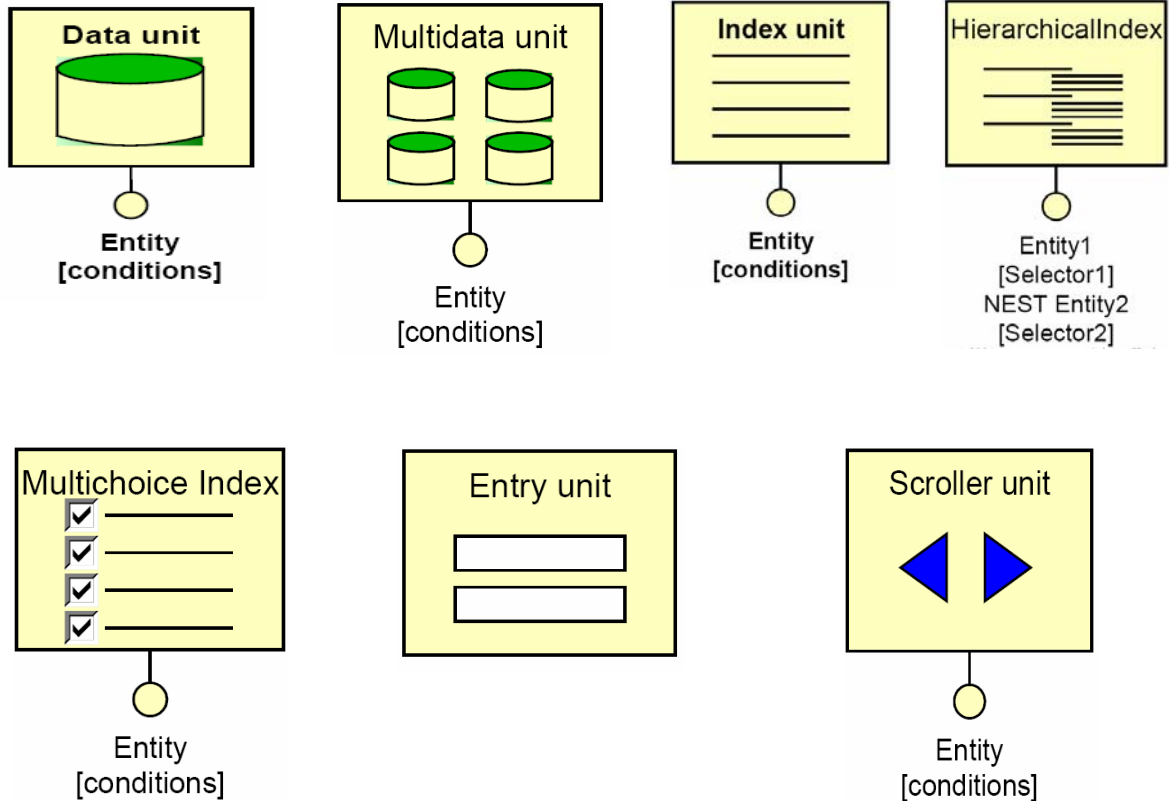
**IS-A Hierarki:** Uttrykker en utledning av et spesifikt konsept fra et mer generelt. (Arv i Objekt Programmering)

**Relasjon:** Forbindelse mellom entiteter. En beskrivelse av forholdet mellom to objekter.

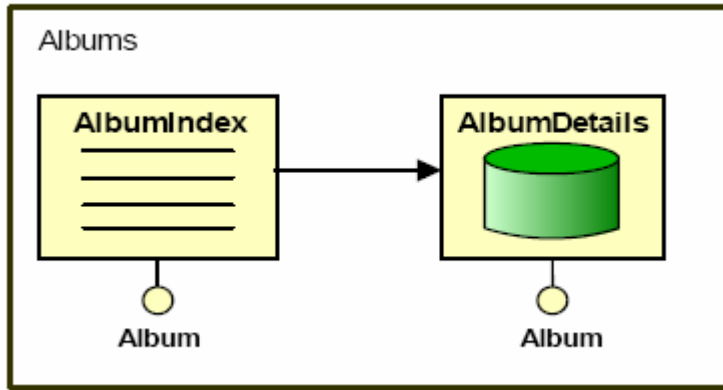
**Relasjonsrolle:** Beskriver forbindelsen mellom entitetene.

# Kapittel 3: Hypertext model

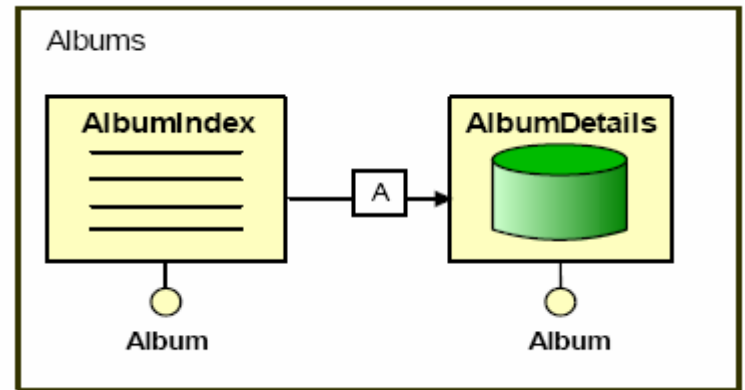
Enheter / Units: utgjør en side.



# Sider / Linker



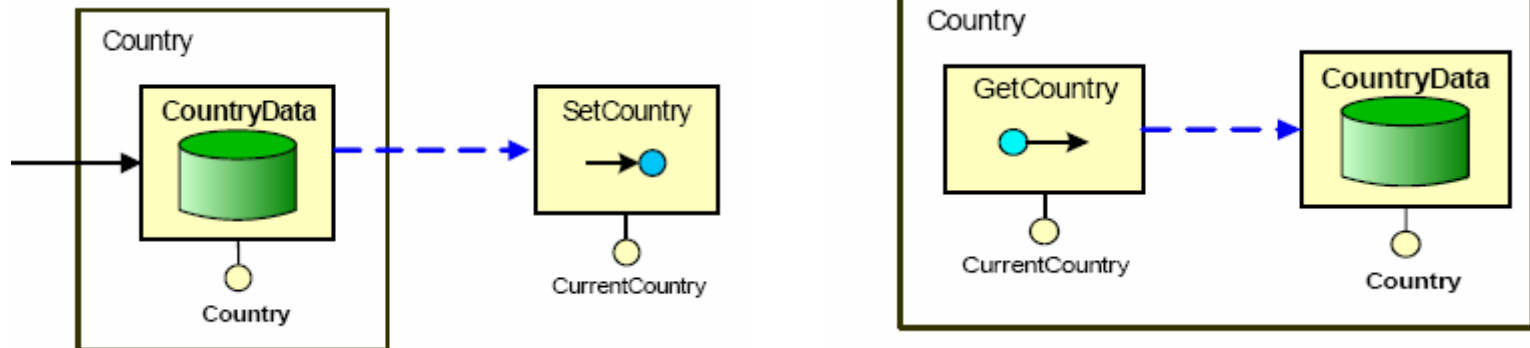
Params



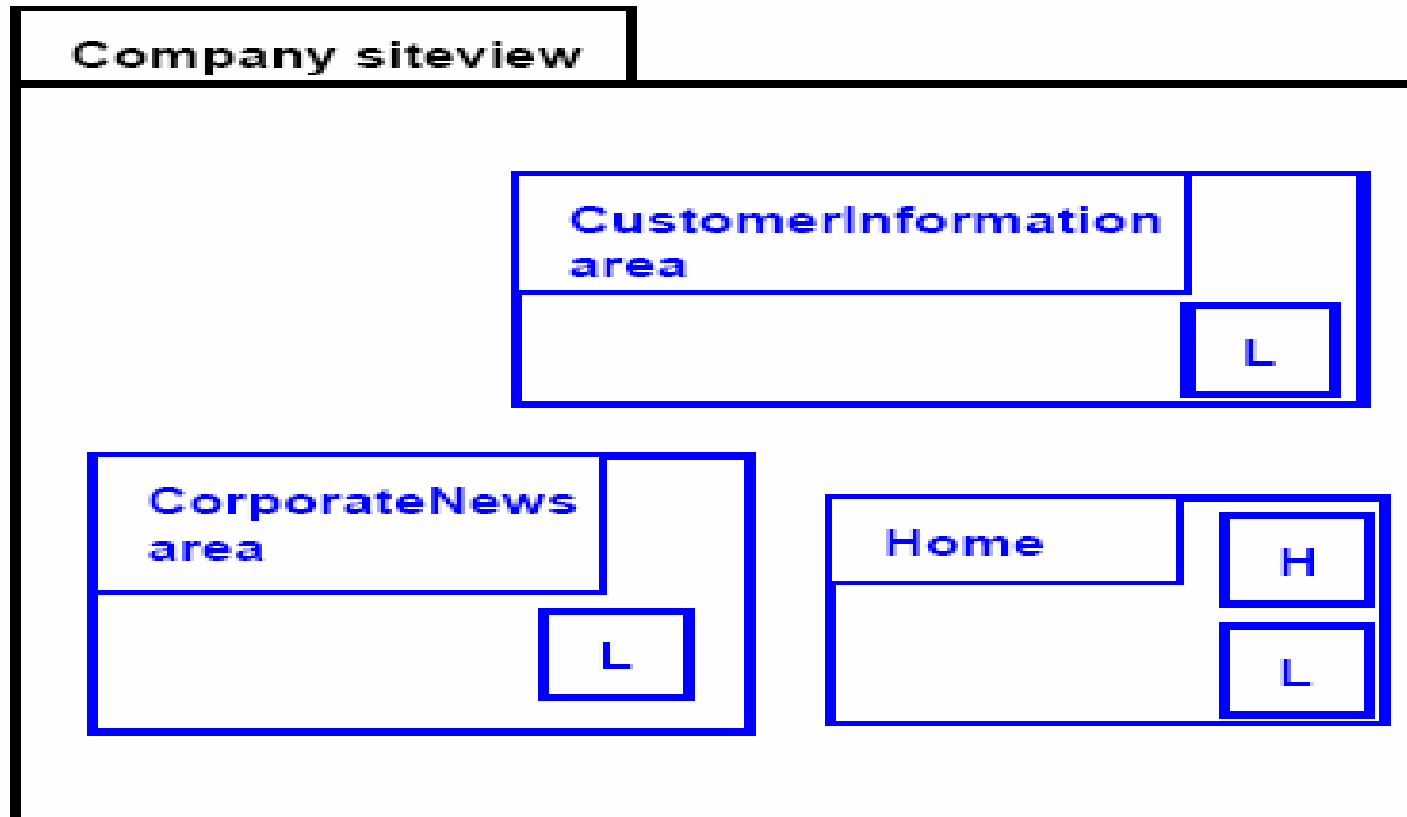
Params



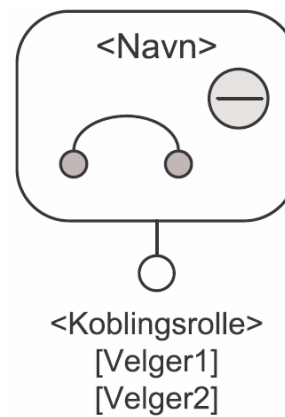
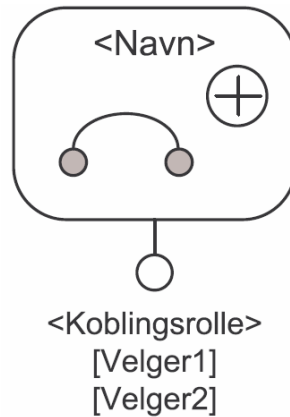
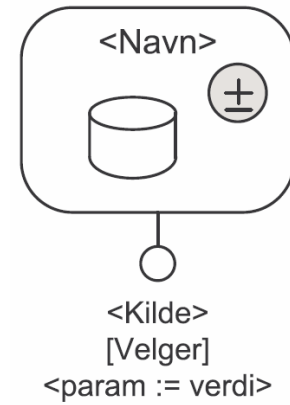
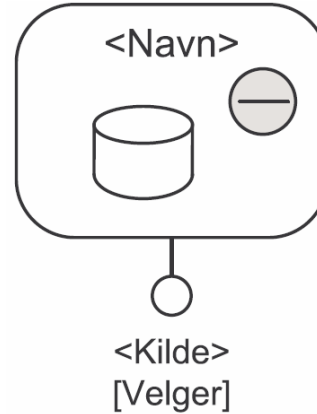
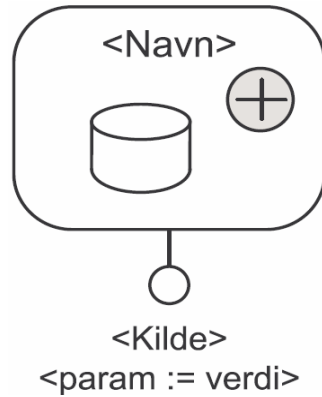
# Globale parameterere



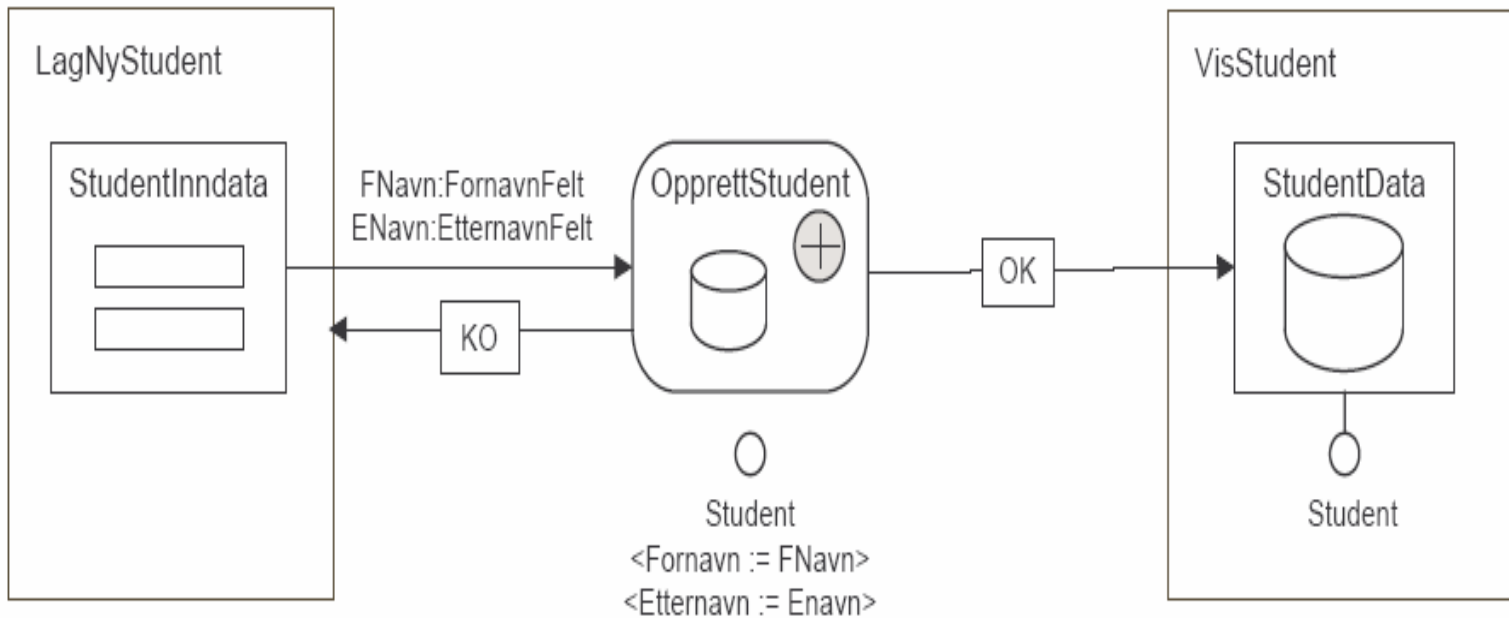
# Hypertekst Organisering



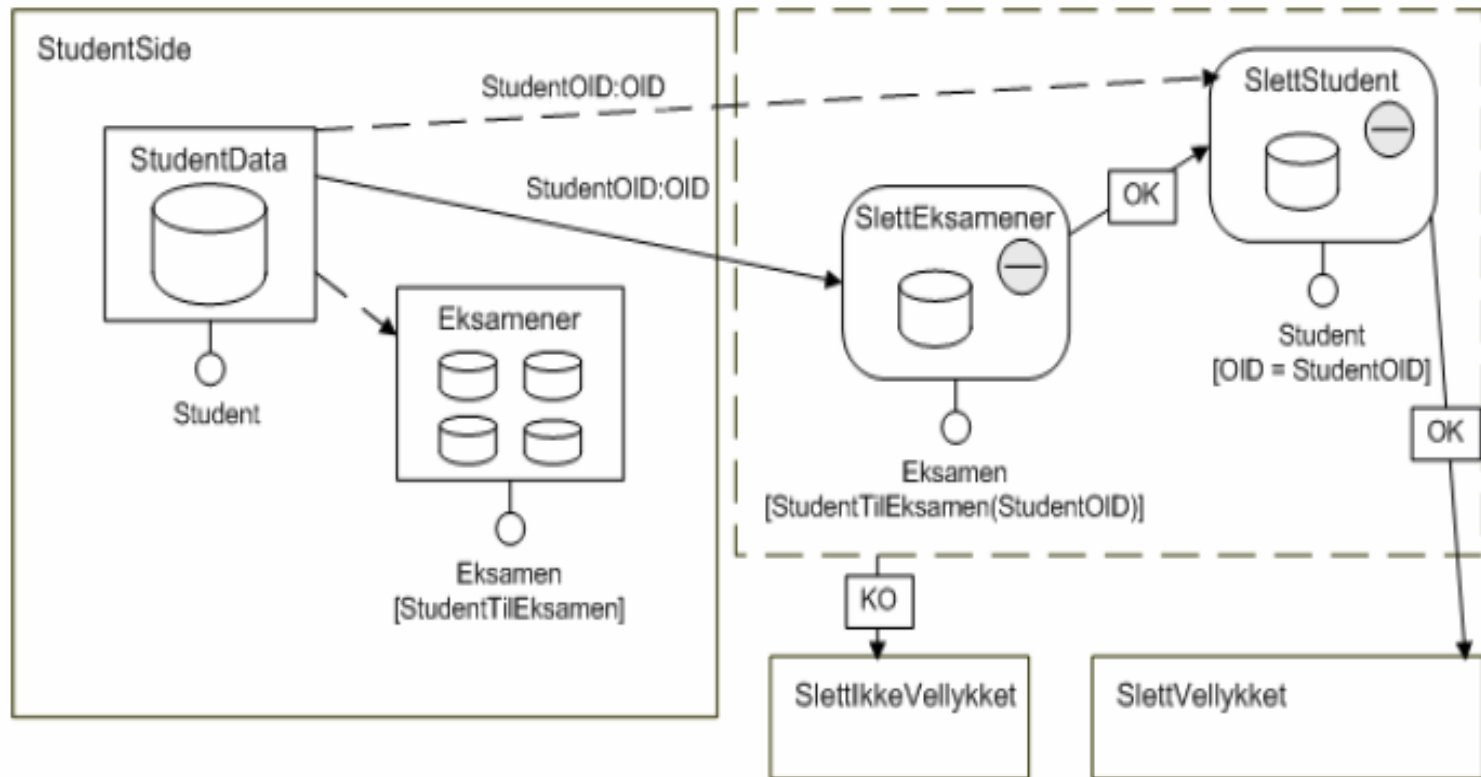
# Kapitel 4: Content Management model



# Opprette / slette / endre

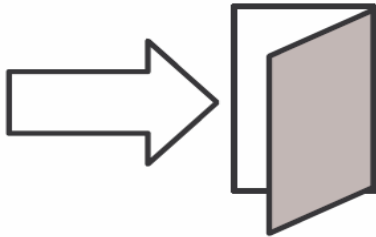


# Transaksjon

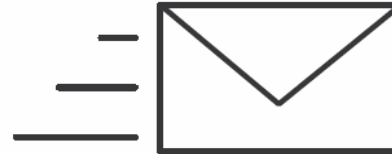


# Forhåndsdefinerte opprasjoner

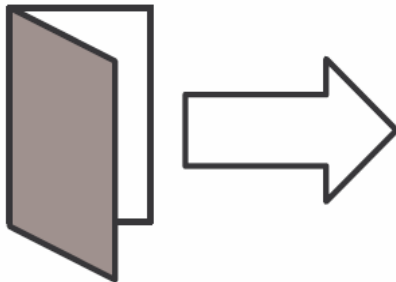
Innlogging



Send E-post



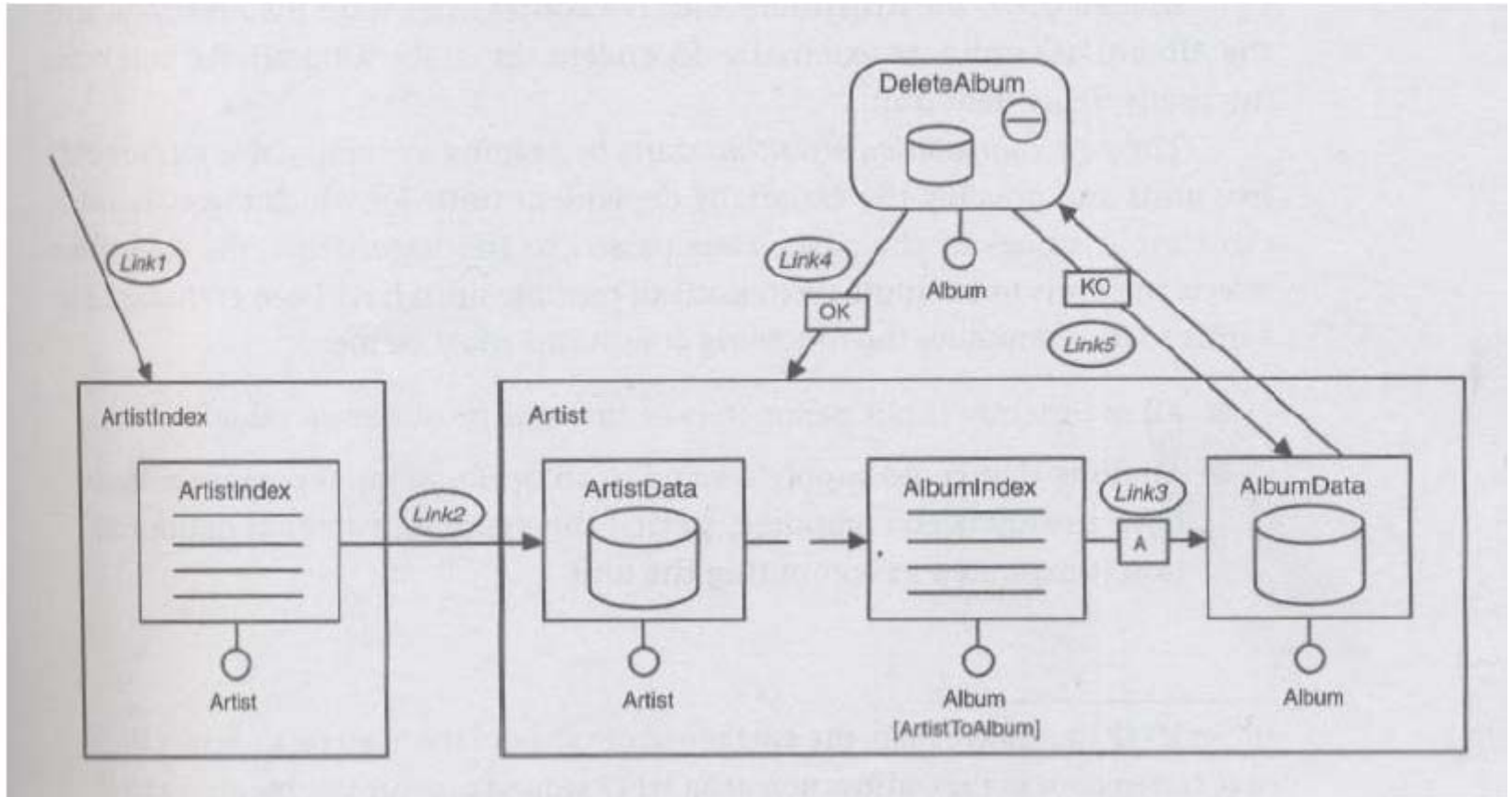
Utlogging



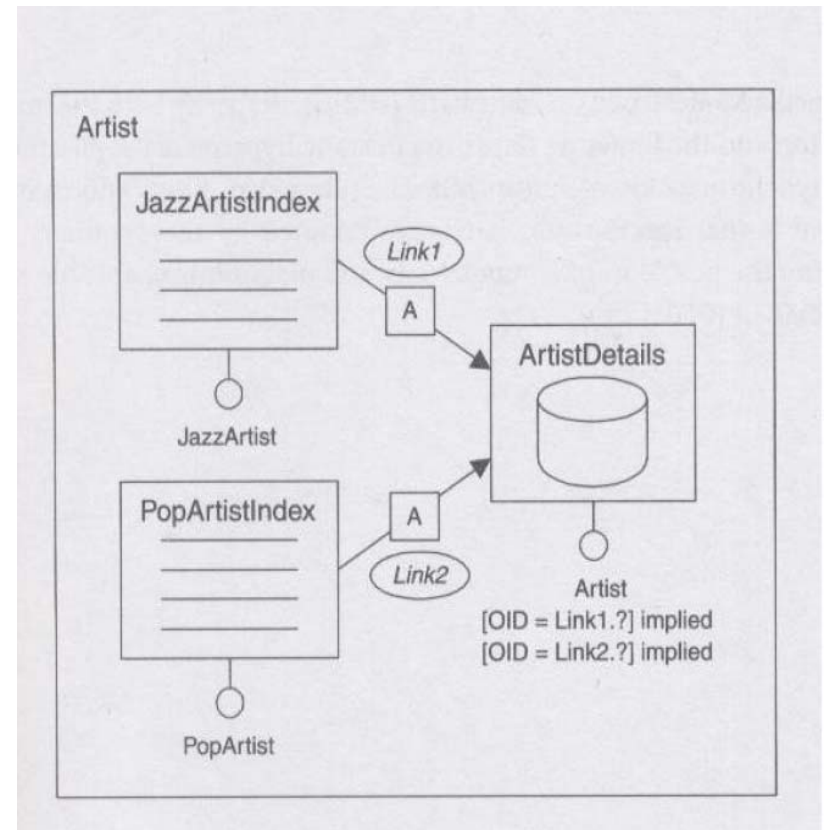
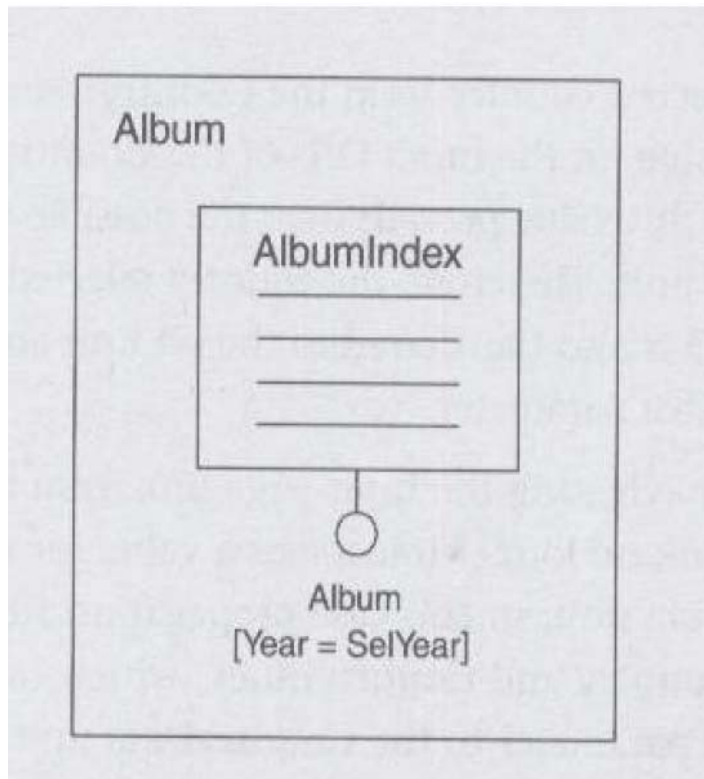
Navn



# Kap 5: Advanced Hypertext Model



# Ugyldige modeller



# Kapittel 6: Oversikt over utviklingsprosessen

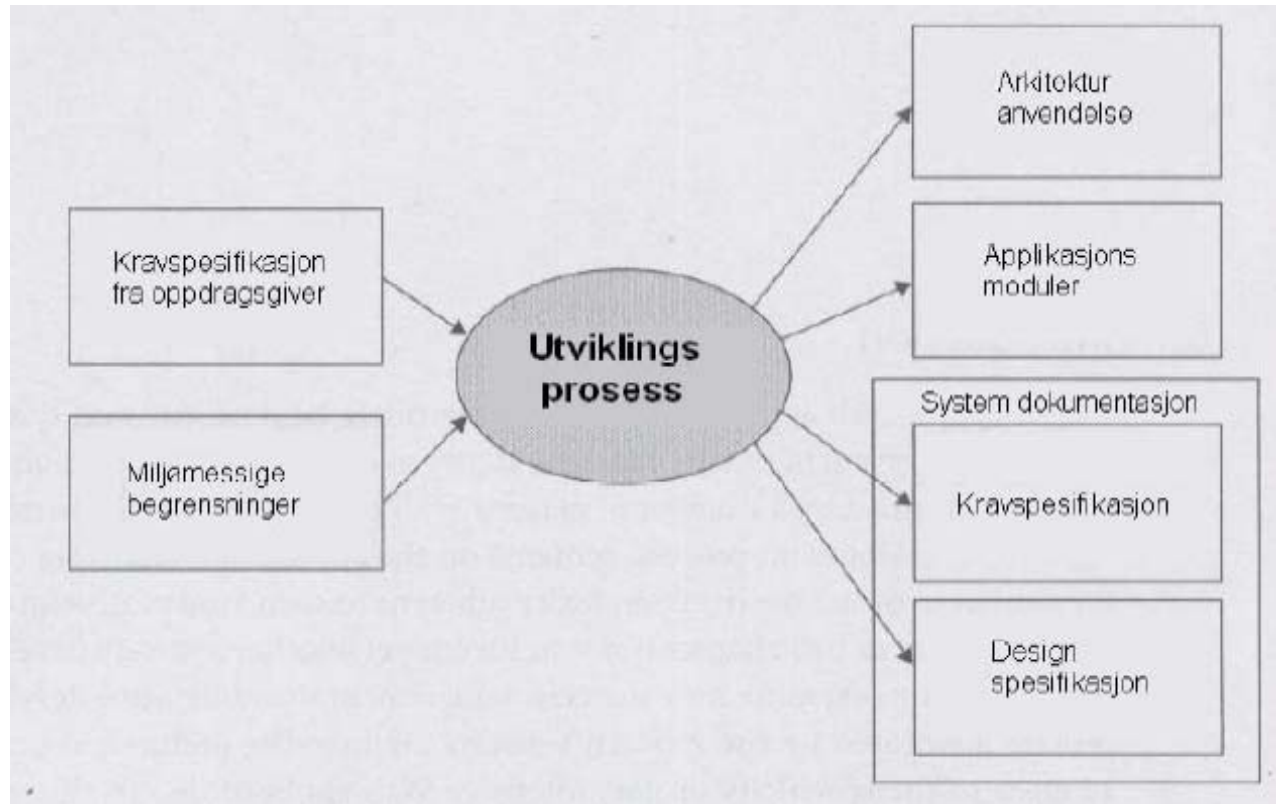
Å utvikle en web-applikasjon er en kompleks prosess. Utviklingen omfatter et bredt spekter av oppgaver, utført av mange personer samtidig.

## **Ulike roller i utviklingsprosessen:**

- Applikasjons analytiker
- Data arkitekt
- Applikasjons arkitekt
- Grafisk designer
- Utvikler/Administrator

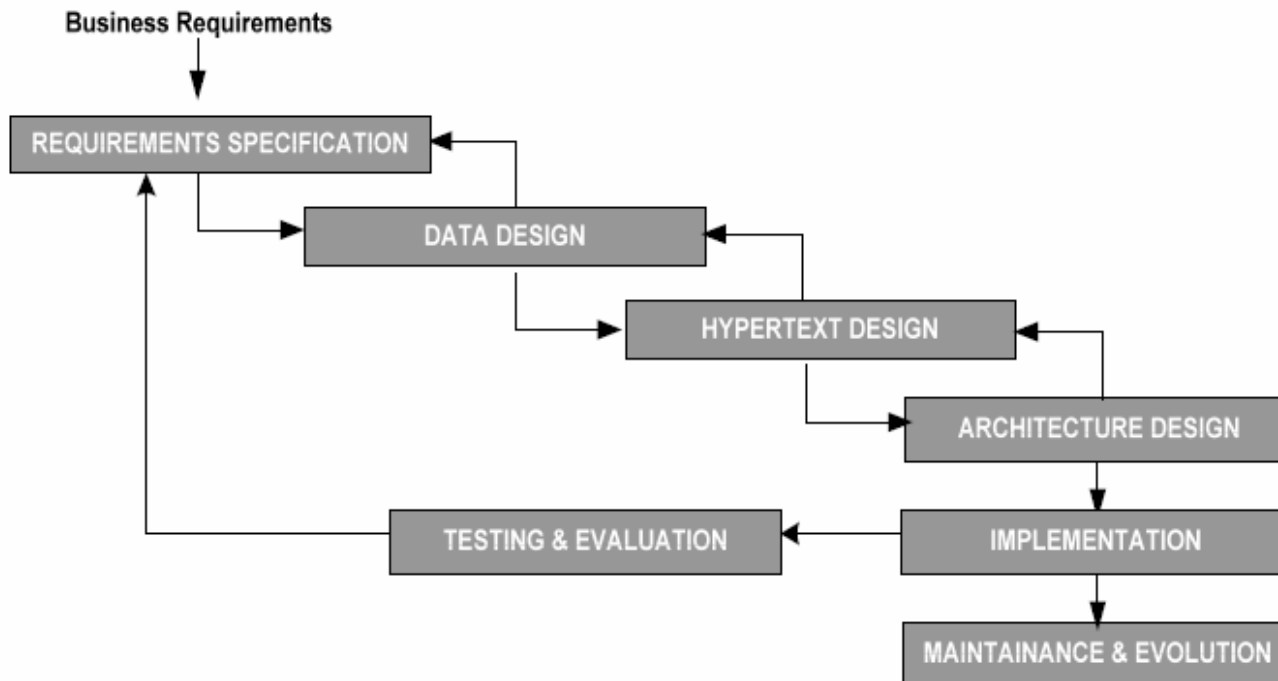
# INPUT

# OUTPUT



# Utviklingssyklus

- Utviklingsprosessen består av flere faser
- Fasene benyttes iterativt. De ulike oppgavene blir repetert og redefinert helt frem til endelig løsning.
- I hver iterasjon blir den foreløpige versjonen av systemet testet og evaluert, utvidet og endret.



# Kapittel 7: Kravspesifikasjoner

**Kravspesifikasjoner består av to underdeler:**

**Samling av krav:**

- Intervjue aktører
- Se på dokumentasjoner
- Hvem skal bruke applikasjon

**Analyse av krav:**

- Se gjennom og formalisere krav
- Lage spesifikasjoner som danner utgangspunkt for design av applikasjon

## **Sjekkliste for å identifisere krav:**

- Identifisere brukerne
- Funksjonelle krav
- Datakrav
- Personaliseringskrav
- Komponentspesifikke krav

# Kapittel 8: Data design

## **Lage ER modell av kravspesifikasjoner.**

### **2 scenario:**

- Ingen eksisterende database over hva applikasjonen skal inneholde. Datadesignet utvikles samtidig med innholdsdatabase.
- Det eksisterer innholdsdatabase over hva applikasjonen skal inneholde. Datadesignet må tilpasses helt/delvis.

## Tilpasning til eksisterende database

Tilpasningsprosessen må ikke forkludre datadesignet.

- Dårlig strukturert innhold.
- Godt hypertekst design reflekterer godt datadesign.
- Dårlig struktur i innholdsdatabasen gir dårlig hypertekstdesign.

# Kapittel 9: Hypertekstdesign

## **Grov design**

- lager et første utkast av hver «site view». Elementer fra dataskjemaet blir koplet til de områdene der de blir brukt.

## **Detaljert design**

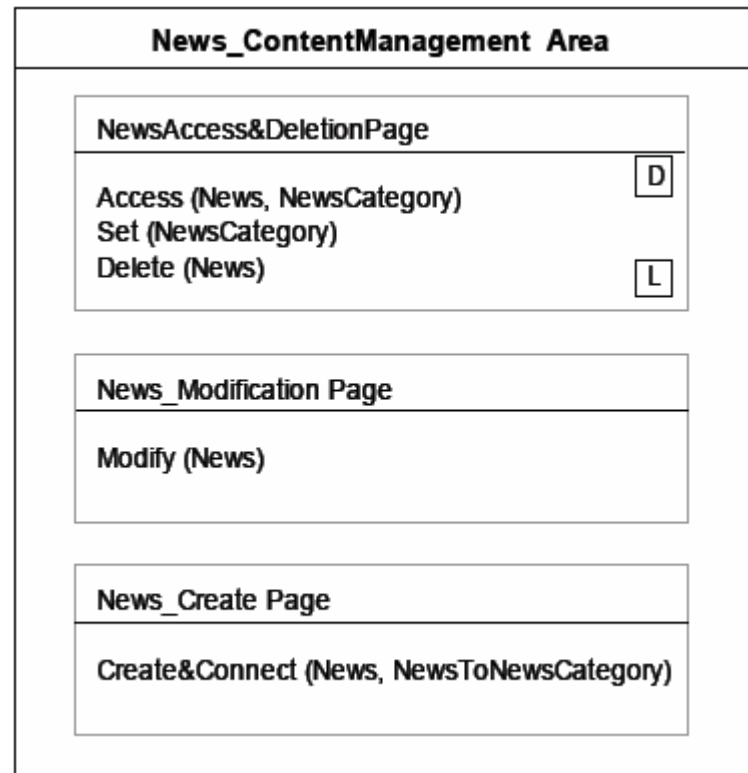
- raffinering av grov design der utkast blir gjort om til detaljerte WebML sider som støtter opp om de egenskapene som er beskrevet i den grove designen.

## Grov design

- Identifisering av område
- Spesifisering av områdesynlighet
- Spesifisering av innhold

## Detaljert design

- Identifisering av side
- Spesifisering av sidesynlighet
- Spesifisering av sideinnhold



# **Brukervennlig hypertekst**

Viktige punkter for brukervennligheten:

- Navigering
- Orientering
- Søk
- Konsistens

## Kappittel 10: Arkitektur design

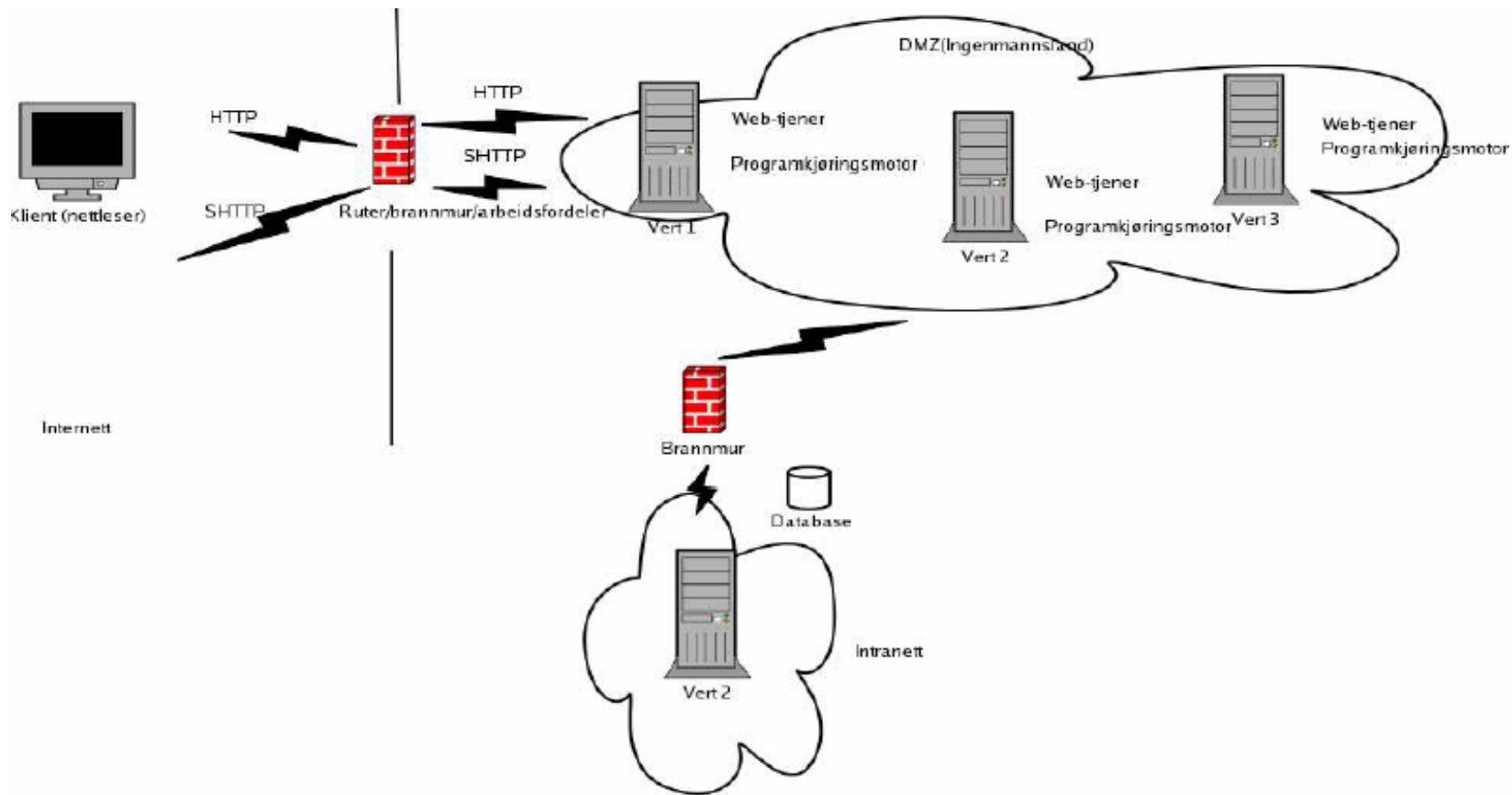
- Mål:
- Ytelse
- Skalerbarhet
- Tilgjengelighet
- Vedlikehold av tilstand for bruker
- Sikkerhet

# Begrensninger

- Kostnader
- Komplexitet
- Eksisterende systemer

# Installering av arkitektur

- Drifter selv, plassert hos seg selv
- Drifter selv, plassert hos andre
- Andre har og drifter systemet for dem



# Teknikker for testing og forbedre ytelsen

- Klarer arkitektur og ta unna trafikken?
  - Beregne forventet trafikk
  - Teste implementasjonen
- Finn resursene som evt. holder igjenn systemet(web-server, app-kode, db)
- Legg til flere resurser, forbedre koden, ta i bruk cacheing.

# Cache

Fordeler:

- redusere svartid på forespørsel
- reduserer bruk av resurser

Ulemper:

- Skal implementeres og vedlikeholdes
- Tar plass og mellomagre

Kan mellomagres:

- Beregnede sider, db spørringer

Cache på egne maskiner, ikke tilknyttet direkte til applikasjonen

# Kapittel 11: Designing Data-Intensive Web Applications

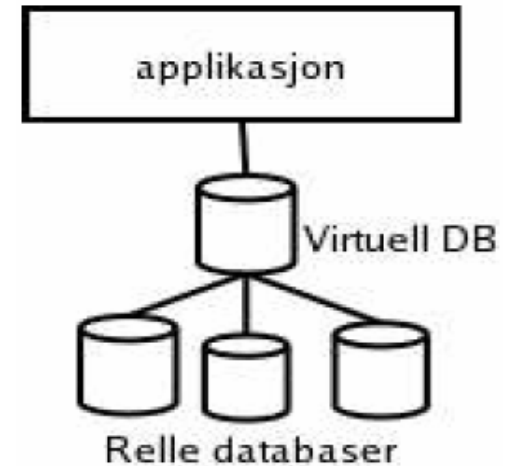
## **Dataimplementasjon**

- Avbilder data fra det konseptuelle skjemaet på konkrete datakilder.
- Muliggjør visning og håndtering av dataigjennom vevgrensesnitt.

# Databasetyper

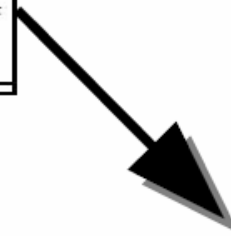
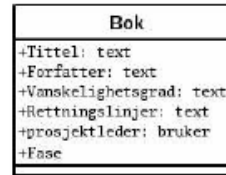
- Dedikert database.
- Replikert database.
- Online databaser
- Distribuerte databaser

Mange distinkte databaser, ofte på fysiske forskjellige steder, blir presentert av den distribuerte databasen som en enkelt database.



## Mapping (fra skjema til sql)

- Hver entitet blir eget skjema
- Hver attributt blir en kolonne



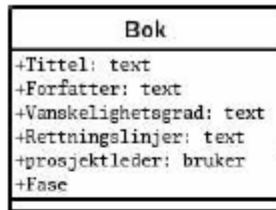
```
CREATE TABLE BOK {  
    oid SERIAL PRIMARY KEY,  
    forfatter TEXT,  
    vanskelighetsgrad TEXT,  
    retningslinjer TEXT,  
    prosjektleder INTEGER,  
    fase INTEGER  
};
```

# Avbildning av relasjoner

To generelle typer:

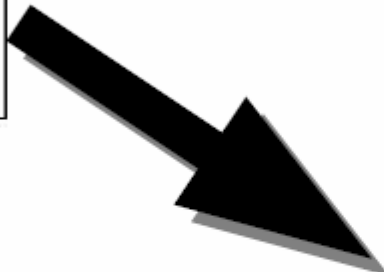
- mange-til-mange relasjoner.
- en-til-mange relasjoner.

Mange-til-en eksempel



1:1

0:N



```
CREATE TABLE BOK {  
  oid SERIAL PRIMARY KEY,  
  forfatter TEXT,  
  vanskelighetsgrad TEXT,  
  retningslinjer TEXT,  
  prosjektleder TEXT,  
  fase INTEGER,  
  FOREIGN KEY prosjektleder REFERENCES Bruker  
  ON DELETE CASCADE
```

```
};
```

## **Nyttige funksjoner:**

### **View:**

- Et view er en navngitt spørring som presenteres som en tabell for applikasjonen.
- Denne pseudotabellen kan ha felt som er utledet fra data i andre tabeller.

### **Indeksering:**

Indeksering øker søkehastigheten på de feltene som er indeksert.

## Kapittel 12: Hypertext Implementasjon

- modellen av web-applikasjonen blir gjort om til software komponenter
- Boken gir gode eksempler på hvordan dette IKKE skal gjøres

# Kapittel 13: Avansert Hypertext Implementasjon

- Hvordan skrive modell om til MVC
- Fordeler:
  - Lagdelt
  - fordele ansvar
  - penere kode
  - enklere å vedlikeholde
  - gjennbruk av kode
- EJB, dele logikk med ikke-web applikasjoner, bedre ytelse, bedre fordeling av last enn med servletcontainer
- css (utsende) xslt (layout)

