INF 329: Web-Teknologier

Dataimplementasjon

Fra Kapittel 11 i «Designing Data-Intensive Web Applications»

Presentasjonsdato: 17/10/2004

av:
Dag Viggo Lokøen (dagvl@ii.uib.no)
Kent Inge F. Simonsen (kentis@ii.uib.no)

Dataimplementasjon

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

Databasetyper

- Dedikert database.
  - Når innholdet ikke finnes før applikasjonen.
  - Konstruksjon av denne er en del av vevapplikasjonsutviklingen.
  - Data er eid av de som eier applikasjonen.
  - Applikasjonen har mulighet til å lese fra og skrive til databasen.
  - Vedlikehold på databasen må typisk gjøres av applikasjonsvedlikeholderene.

Databasetyper

- Replikert database.
  - Innhold finnes på en eller flere eksisterende kilder som blir kopiert til en database dedikert til vevapplikasjonen.
  - Vevapplikasjonen kan typisk ikke skrive til databasen.
  - Kan fort ha en del forsinkelser i databaselaget.

Databasetyper

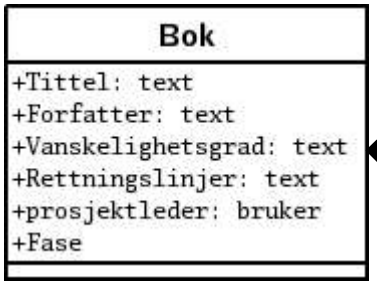
- Online databaser
  - Applikasjonen har direkte adgang til en eksisterende database.
  - All data finnes på i den eksisteredne databasen, (med unntak av noe caching og slik).
  - Kan by på ytelsesproblemer om mange bruker samme databasen.
  - Applikasjonen har både skrive og lesetilgang til databasen.

Databasetyper

- Disse og alle mulige kombinasjoner av disse finnes.
- Eksempel freedb-lignende databaser som er en onlinedatabase, tillater bare lesing og skriving av nye plater men ikke oppdatering av eksisterende.
- ER-diagram brukes uansett.
- For eksisterende databaser for å hjelpe utviklingen av siden.
- For nylagde (dedikert og replikert) for å lage databasen.

Mapping (fra skjema til sql)

- Hver entitet blir eget skjema.
  - Hver attributt blir en kolonne.
  - I tillegg er det ofte lurt med en egen nøkkelkolonne for primærnøkkelen.




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

Blob

- En type attributt som inneholder store binære objekter.
  - Kan lagres i som del av databasen.
    - Sentralisert styring på dataene.
  - Kan lagres i filsystemet utenfor databasen.
    - Kan være greiere å implementere.

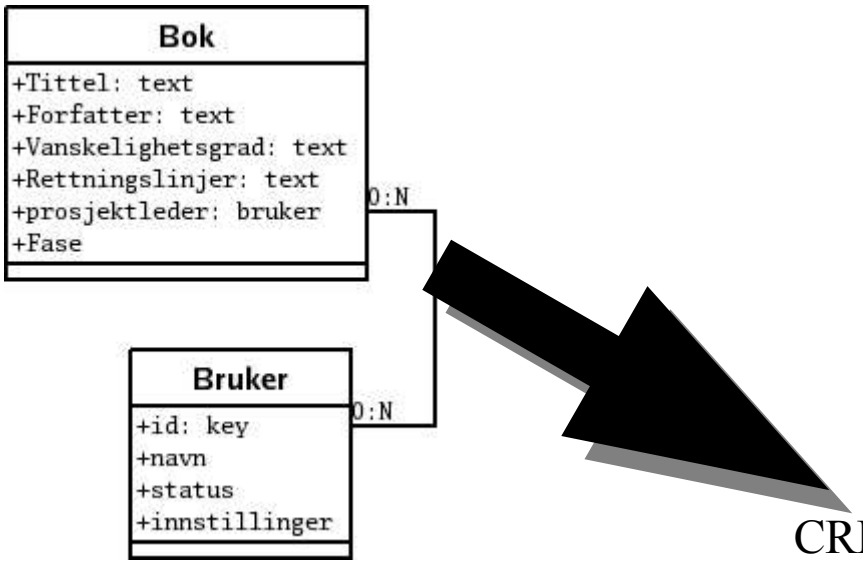
Blob

- Som del av databasen
  - I en dertil egnet kolonne i samme tabell som resten av entiteten.
  - I en dedikert tabell sammen med en nøkkel som kan refereres til fra tabellen der resten av entiteten er lagret.
- I filsystemet.
  - Explisit lenke i tabellen.
  - Implisitt lenke. Applikasjonen holder selv orden på hvilken data som hører til hvor.

Avbildning av relasjoner

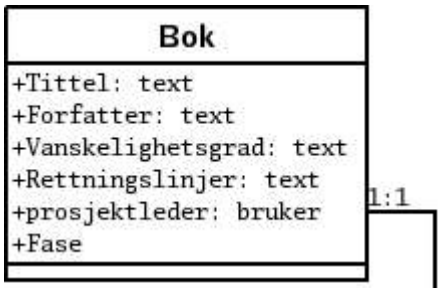
- To generelle typer
  - mange-til-mange relasjoner.
    - krever en koblingstabell.
  - en-til-mange relasjoner.
    - en-til-en relasjoner kan her ses på som et spesialtilfelle av en-til-mange relasjoner.
    - kan implementeres direkte ved krysstabellreferanser.

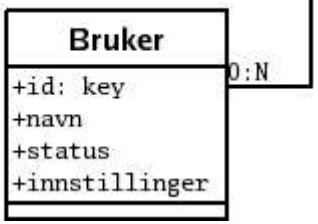
Mange-til-mange med koblingstabell

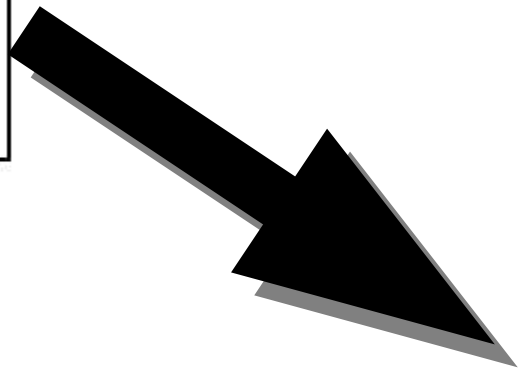



```
CREATE TABLE lederkobling {
        bok_oid INTEGER,
        bruker_oid int,

        PRIMARY KEY(bok_oid, bruker_oid),

        FOREIGN KEY bok_oid REFERENCES bok
        ON DELETE CASCADE,

        FOREIGN KEY bruker_oid REFERENCES bok
        ON DELETE CASCADE
};
```

Mange-til-en eksempel





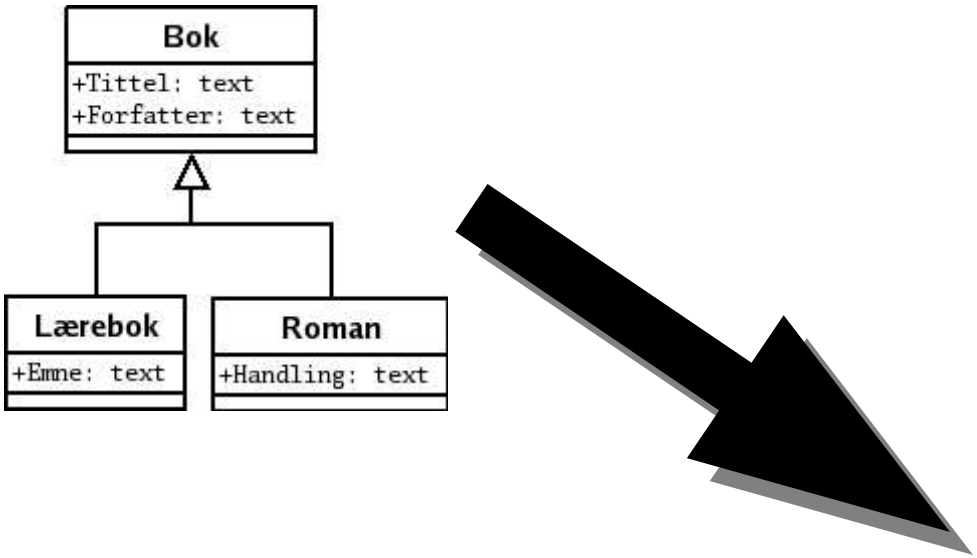


```
CREATE TABLE BOK {
        oid SERIAL PRIMARY KEY,
        forfatter TEXT,
        vanskelighetsgrad TEXT,
        retningslinjer TEXT,
        prosjektleder TEXT,
        fase INTEGER,
        FOREIGN KEY prosjektleder REFERENCES Bruker
        ON DELETE CASCADE
};
```

Avbildning av generaliseringshierakier

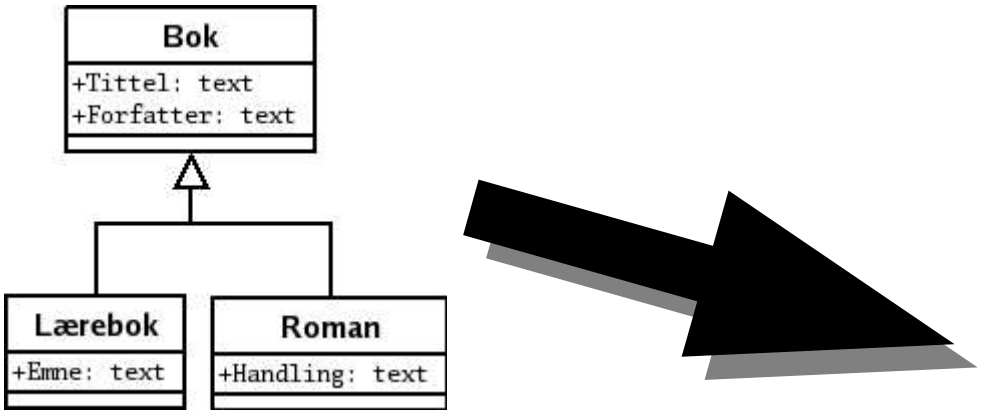
- Generaliseringshierarkier er støttet av SQL-99 og en rekke dbmser.
- Man kan også emulere slik støtte i rene relasjonelle dbmser.
  - Horisontalt ved å legge all data inn i en supertabell
  - Vertikalt ved å danne egne tabeller for hver underentitet.

Horisontal avbildning



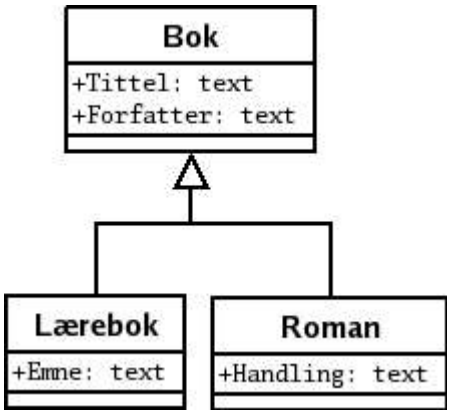

```
CREATE TABLE BOK{
        oid SERIAL PRIMARY KEY,
        tittel TEXT,
        forfatter TEXT,
        emne TEXT,
        handling TEXT
};
```

Vertikal avbildning




```
CREATE TABLE Bok{
        oid SERIAL PRIMARY KEY,
        tittel TEXT,
        forfatter TEXT,
};

CREATE TABLE Lærebok{
        oid SERIAL PRIMARY KEY,
        tittel TEXT,
        forfatter TEXT,
        emne TEXT NOT NULL
};

CREATE TABLE Roman{
        oid SERIAL PRIMARY KEY,
        tittel TEXT,
        forfatter TEXT,
        handling TEXT NOT NULL
};
```

Arv i SQL-99

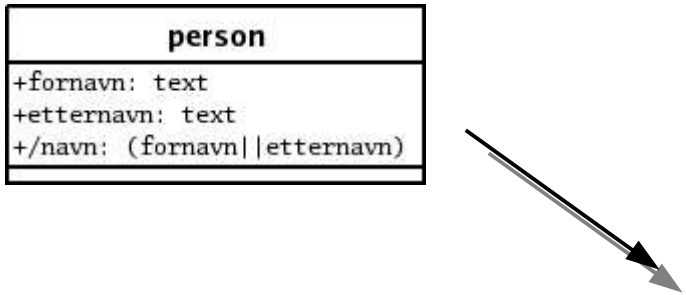



```
CREATE TABLE Bok{
        oid SERIAL PRIMARY KEY,
        tittel TEXT,
        forfatter TEXT,
};

CREATE TABLE Lærebok{
        emne TEXT NOT NULL
} INHERITS (Bok);

CREATE TABLE Roman{
        handling TEXT NOT NULL
} INHERITS (Bok);
```

Utledning av deriverte data vha views

- 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.

Et view




```
CREATE TABLE person{
        oid SERIAL PRIMARY KEY,
        fornavn TEXT,
        etternavn TEXT
};

CREATE VIEW fullenavn AS SELECT fornavn, etternavn,
fornavn||' '||etternavn AS navn FROM person;
```

Fysiske designtips

- Indeksering øker søkehastigheten på de feltenen som er indeksert.
  - Vil du søke på det? Indekser det!

Kapittel 11, INF329

Dataforvaltning og -arkitektur

Hovedarkitekturer

- Dedikert database
  - Standardskjemaet er databaseskjemaet
  - Ingen integrasjonsproblemer
- Replikert database
  - Standardskjemaet er databaseskjemaet
  - *Men* man må kopiere inn data fra eksterne kilder
- Onlinedatabase
  - Standardskjemaet er det *view* man har mot den eksterne databasen

Skjemaintegrasjon

- Standardskjemaet er det «beste» skjemaet for vevapplikasjonen
- I en idèell verden hadde skjemaet i allerede eksisterende databaser vært lik standardskjemaet.
- I virkeligheten så har man i datadesignet *abstrahert seg vekk fra den reelle organisasjonen av datakildene.*

Skjemaintegrasjon

- Man må derfor harmonisere forskjellene mellom standardskjemaet og de reelle datakildene
- Man kan begrense arbeidsmengden ved å ta disse hensynene allerede i datadesign-fasen, men forskjeller vil alltid oppstå pga:
  - Analysefeil
  - Endringer i dataskjema i eksterne kilder

Skjemaintegrasjon, metoder

- Endre standardskjemaet
  - Endrer dataskjemaet for å gjøre det konsistent med eksterne datakilder
  - Så «propagerer» man endringene til ER-skjemaet
- Endre skjemaet i de eksterne datakildene
  - Her endrer man skjemaet i de eksterne datakildene til å være konsistent med standardskjemaet
  - Dette kan være problematisk siden det ofte krever endringer i andre programmer også

Skjemaintegrasjon, metoder

- Lage en *view*
  - Man legger et presentasjonslag på toppen av de eksterne datakildene som omformer data til formatet påkrevd av standardskjemaet

Dataintegrasjon, problemer

- Fysisk format
  - XML vs PostgreSQL
- Syntaks
  - Eks: MM/DD/YYYY vs DD/MM/YYYY
- Semantikk
  - «Dato sendt» kan i to databaser bety enten dato sendt til kunde eller dato sendt fra leverandør

Replikerte databaser, arkitektur

- Med en replikert database så skiller man mellom den opprinnelige databasen og databasen til vevapplikasjonen.
- Så «pumper» man dataene fra den ene til den andre. Dette kan gjøres via enkle kopier av tabeller (hvis databaseskjemaet stemmer bra med standardskjemaet,) eller man kan gjøre transformasjoner på dataene hvis det er påkrevd pga. forskjeller i skjemaene.

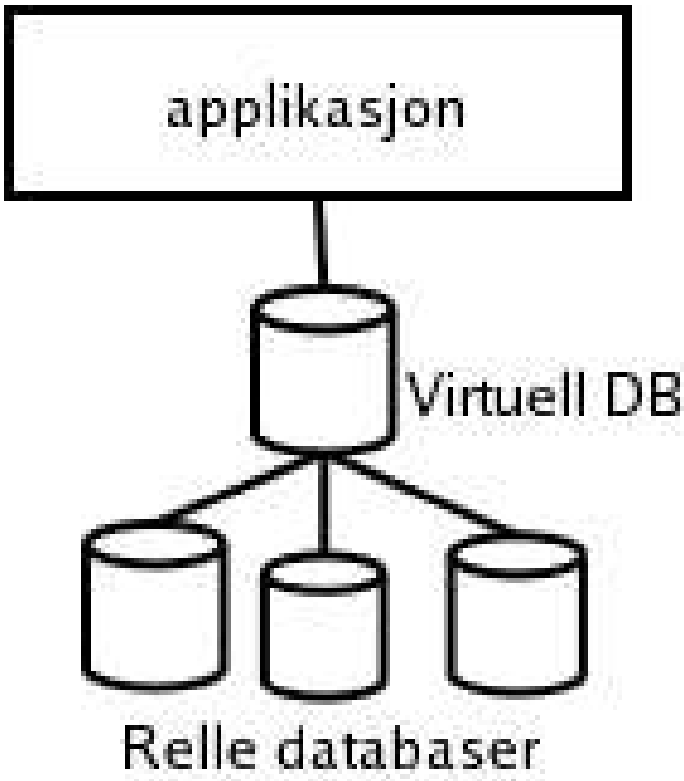
Onlinedatabaser, arkitektur

- Vevapplikasjonen er koblet direkte til bedriftens database, ingen replikering/datapumping.
- Krever ofte tilgang til distinkte databaser
- Beste arkitekturvalg er ofte «distribuerte databaser.»

Distribuerte databaser

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

- Dermed har man lokasjons- og oppdelingsuavhengighet for applikasjoner



Distribuerte databaser, tjenester

- En distribuert database tilbyr vanligvis følgende tjenester:
  - Distribuerte *views*
  - Distribuerte spørringer
  - Distribuerte transaksjoner
  - Distribuert administrering
  - God skalering av ytelse
- Ulemper ved distribuerte databaser:
  - Krever mye kunnskap fra teknisk personell.

Distinkte databaser, arkitektur

- Vevapplikasjonen må selv åpne koblinger mot alle de benyttede databaser.
- Fordelen med dette er at det er en veldig enkel arkitektur med lavere krav til databasene. Dermed blir det ofte mye billigere.
- Ulemper:
  - Lokasjonsuavhengighet er umulig
  - Avhengigheter mellom databaser må programmeres inn i applikasjonen for hånd

Distinkte databaser, arkitektur