UNIVERSITETET
I OSLO

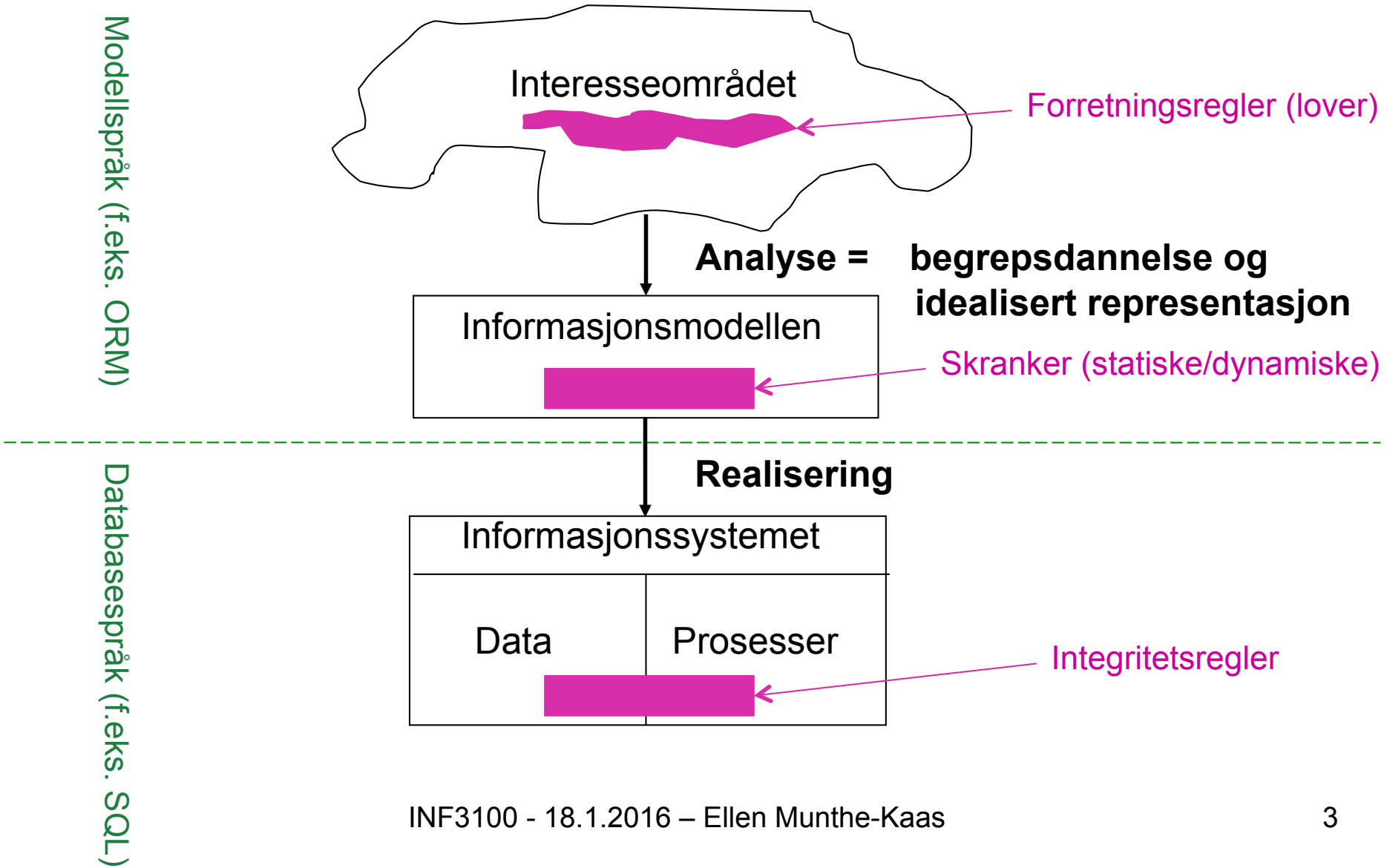
INF3100
Databasesystemer

Dagens tema:

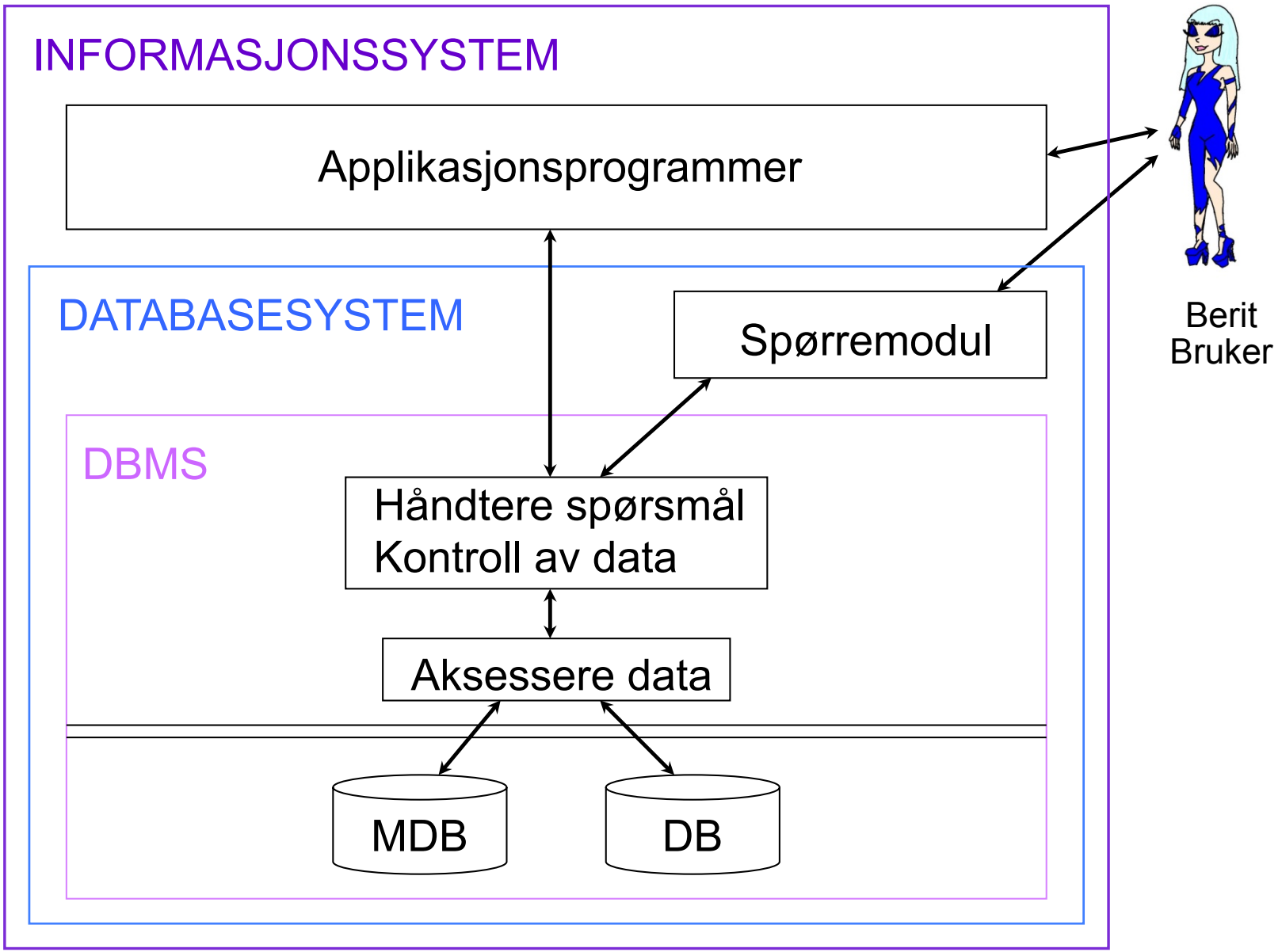
- Databaser og informasjonssystemer; datamodeller, databasemodeller og informasjonsmodeller
- 100%-prinsippet
- Litt databasehistorie
- 3-skjemaarkitekturen
- Databasesystemers oppbygning

Databaser og informasjonssystemer; datamodeller, databasemodeller og informasjonsmodeller

Informasjonssystemer

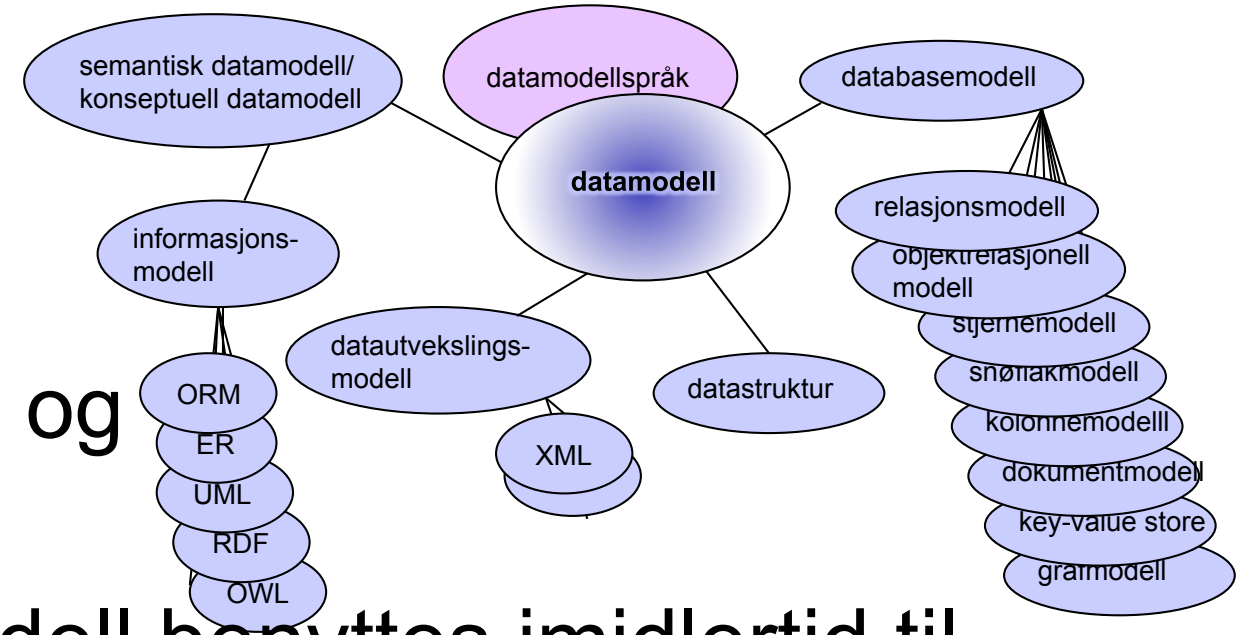


Informasjonssystemer vs. databaser



Hva er en datamodell?

- **Datamodell**: Beskriver hvordan data representeres og aksesseres



- Ordet datamodell benyttes imidlertid til forskjellige formål i alle ledd av informasjonsdesignprosessen og på forskjellige nivåer innen hvert ledd

Informasjonsmodell

- **Semantisk datamodell / konseptuell datamodell / informasjonsmodell**: En fullstendig beskrivelse av interesseområdet
  - En informasjonsmodell beskrives i et modellspråk
  - Eksempler på modellspråk: ORM, OWL, ER, UML klassediagrammer

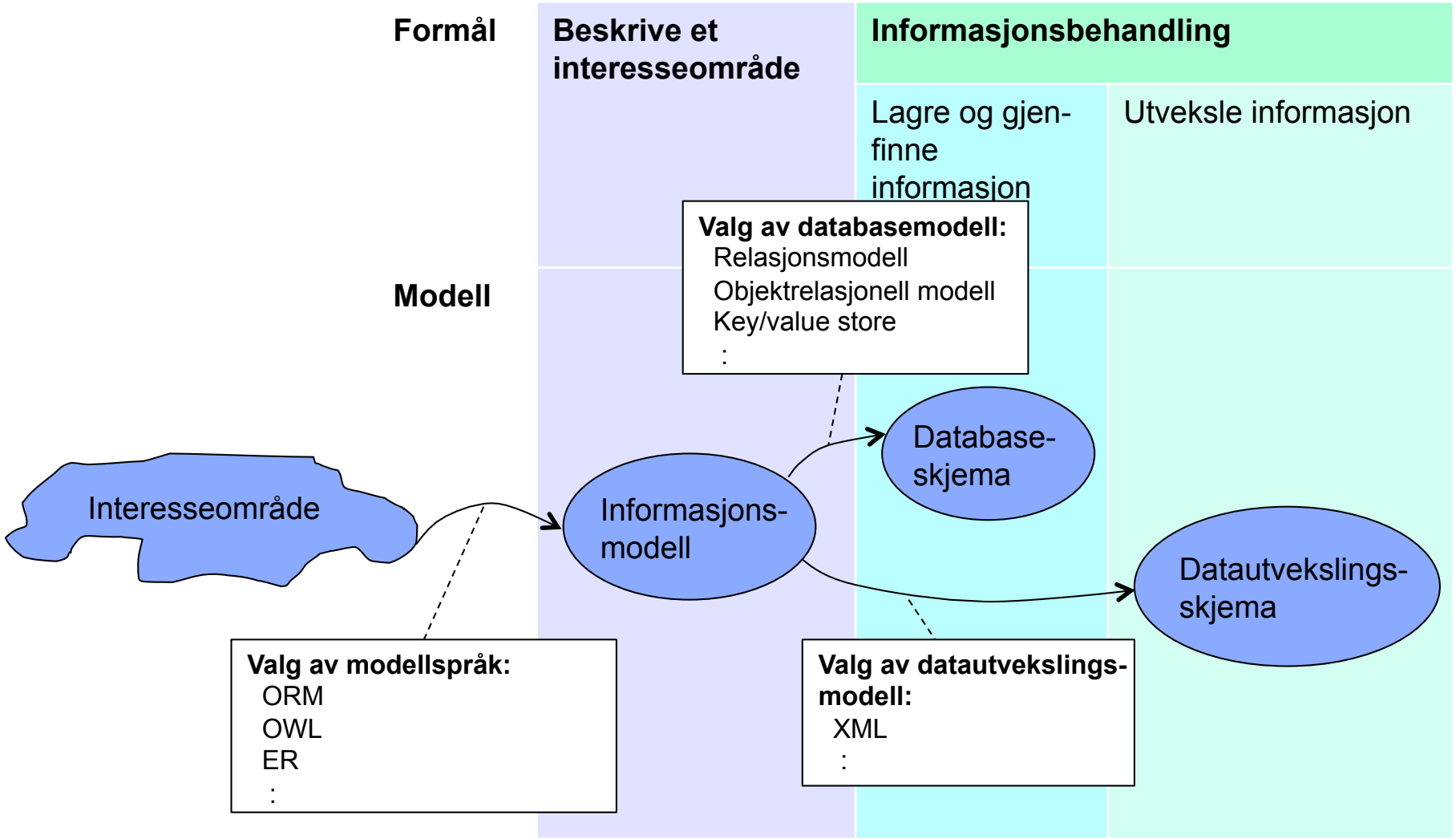
Databasemodell

- **Databasemodell**: Beskriver prinsippene for hvordan en database struktureres og brukes
  - Eksempler på databasemodeller: Relasjonsmodell, objektdatabasemodell, objektrelasjonell modell, key-value store, grafmodell
  - Som språk for relasjonsmodellen og den objektrelasjonelle modellen brukes vanligvis forskjellige versjoner av SQL
- **Databaseskjema** / **begrepsmessig skjema**: Beskriver en konkret database
  - Instans av en databasemodell
  - Reflekterer en informasjonsmodell

Datautvekslingsmodell

- **Datautvekslingsmodell**: Beskriver prinsippene for hvordan data skal struktureres ved utveksling av data mellom forskjellige systemer
  - Eksempel på datautvekslingsmodell (og modellspråk for datautveksling):
    eXtensible Markup Language (XML)
- **Datautvekslingsskjema**: Beskriver dataformat for et konkret datautvekslingsformål
  - Instans av en datautvekslingsmodell

Sammenheng mellom modeller



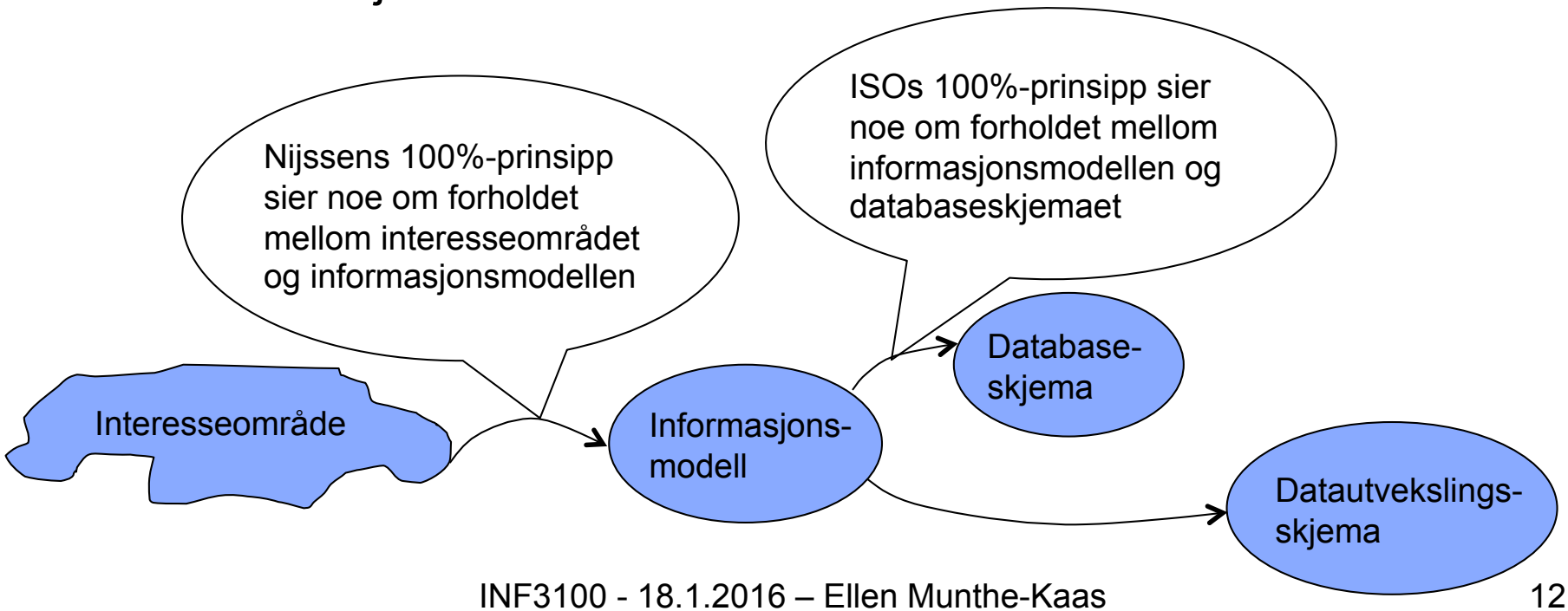
100%-prinsippet

Informasjonsmodeller og 100%-prinsippet

- En fullstendig beskrivelse av interesseområdet kalles en **informasjonsmodell**
- *100%-prinsippet* sier at det er mulig å lage en (endelig) informasjonsmodell på norsk eller et annet naturlig språk
- Hvorvidt dette er riktig, er et filosofisk spørsmål

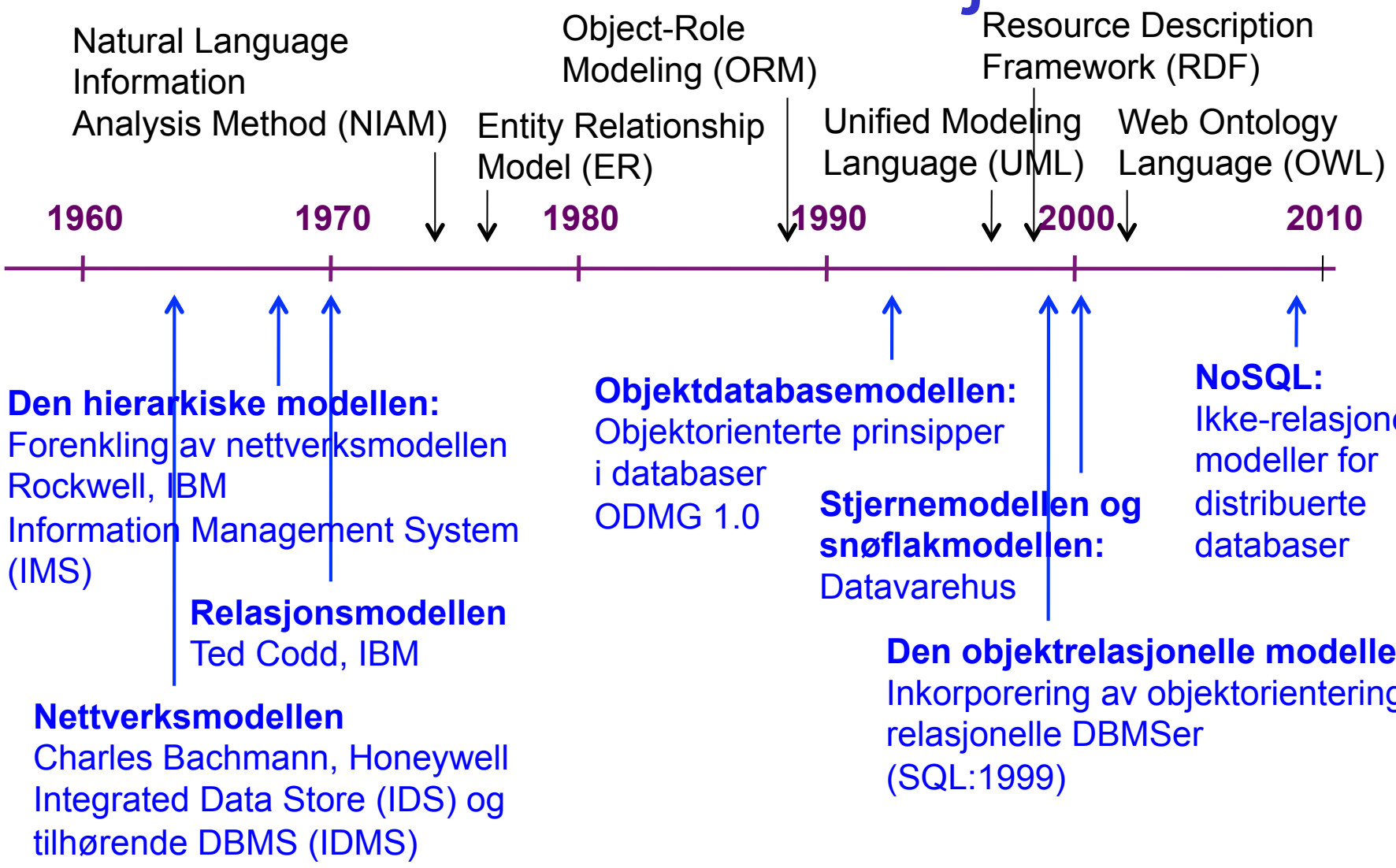
100%-prinsippet i litteraturen

- Merk at det i litteraturen finnes *to forskjellige «100%-prinsipper»:*
  - **Nijssens 100%-prinsipp** (forrige lysark): Det fins et språk som kan beskrive alt vi har behov for å beskrive
  - **ISOs 100%-prinsipp**: «Alt står i informasjonsmodellen» – databaseskjemaet får ikke inneholde noe mer enn det informasjonsmodellen omtaler



Litt databasehistorie

Historisk tidslinje



Databasemodeller

- **Databasemodell**
  Mengde av begreper for å beskrive strukturen til data lagret i en database
- Mulige databasemodeller:
  - Nettverksmodell
  - Hierarkisk modell
  - Relasjonsmodell
  - Objektdatabasemodell
  - Objektrelasjonell modell
  - Stjernemodell, snøflakmodell (for datavarehus)
  - Under NoSQL-paraplyen (teknologier for distribuerte databaser): Kolonnemodell, dokumentmodell, key-value store, grafmodell

Begynnelsen

- Bachman & Williams:
  A General Purpose Programming System for Random Access Memories (1964) beskrev IDS (Integrated Data Store)
- IDS var det første kommersielle DBMS (utviklet av General Electric, Bachman var prosjektleder)
- Dette var første gang to programmer kunne ha aksess til de samme dataene samtidig (kvasiparallell aksess)
- IDS ble i 1966 solgt og fra da markedsført som IDMS (Integrated Data Management System)

Maskinvare i 1964

- En stor maskin (1 mill. 1964-$) hadde
  - 512 Kbyte RAM
  - 50 Mbyte disk
  - Annet I/O-utstyr (magnet- og papirbånd, hullkortleser og linjeskriver)
- En slik maskin var vannkjølt, krevde stor plass (2–300 m^2), brukte mye strøm (ca 50 kW), og en stab på 10-12 personer for å holde den i gang

Nettverksdatabaser

- IDMS var en *nettverksdatabase*; skjemaet bestod av to typer datastrukturer:
  - Posttyper
  - Sett-typer (1:n-relasjon mellom to posttyper kalt henholdsvis eier- og medlemstype)
- Hver enkelt post kunne delta i vilkårlig mange sett, som eier i noen og medlem i andre, men bare en gang i hver sett-type
- Topologisk er et nettverksdatabaseskjema en rettet graf med posttypene som noder
- IDMS var designet for bruk fra et programmeringsspråk (vertsspråk)

Hierarkiske databaser

- Et hierarkisk databaseskjema har to datastrukturer:
  - Posttyper
  - 1:n relasjoner, kalt foreldre–barn-relasjoner, mellom to posttyper
- Foreldre-barn-relasjonene danner hierarkier (trær)
- I realiteten finnes bare ett kommersielt hierarkisk DBMS, IMS, som ble utviklet av IBM og Rockwell International (North American Aviation) og lansert av IBM i 1968
- 95% av *Fortune-1000*-firmaene bruker fortsatt IMS

Relasjonsdatabaser

- I 1970 presenterte E.F.Codd sin relasjonsmodell
- Dette var en teoretisk beskrivelse av en ny type databaser kalt relasjonsdatabaser
- Relasjonsdatabaser er enkle å beskrive og bruke, men vanskelige å lage DBMS for
- Først i 1977 klarte Oracle å lage et DBMS som fortjener betegnelsen relasjonell

Relasjonsmodellen

- Relasjonsdatabaser har én type datastruktur - relasjoner (tabeller), som tilsvarer posttyper i hierarkiske databaser
- Kolonnene har navn og kalles attributter
- Linjene er navnløse og kalles tupler (forekomster)
- Tuplenes attributtverdier er atomiske
- Alle logiske sammenhenger mellom relasjoner er basert på verdilikhet (fremmednøkler, join)
- SQL er ISO-standard for definisjon og bruk av relasjonsdatabaser

Egenskaper relasjoner

- Hver av verdiene i et tuppel er hentet fra et domene eller er nil
  - Et attributt kan ha verdien nil bl.a. hvis det ikke er lagt inn noen verdi ennå, eller det ikke gir noen mening å ha en verdi for attributtet, eller hvis verdien er ukjent.
  - I SQL brukes **null** for å betegne nil-verdier
- Et domene kan være endelig eller uendelig
- To attributter i en relasjon kan ha samme domene, men ikke samme navn
- Tuplenes rekkefølge i en instans er vilkårlig
- Verdienes rekkefølge i et tuppel er vilkårlig
  - Kan alltid bruke attributtnavn til å identifisere verdiene i et tuppel entydig
- I en instans kan det ikke finnes to like tupler
  - Men: de fleste (alle?) **implementasjoner**/DBMSer tillater like tupler

Definisjon av nøkler

Gitt en relasjon $R(A_1,A_2,\ldots,A_n)$ **med tilhørende integritetsregler**

La X være en delmengde av {A1,A2,…,An}.
Hvis t er et tuppel i en instans av R, betegner t[X] verdiene i t's X-attributter.

- **Supernøkkel**: En delmengde X av $\{A_1,A_2,\ldots,A_n\}$ som er slik at hvis t og u er to tupler hvor t≠u, så er t[X] ≠u[X].
  Dvs. t og u skal alltid ha forskjellig verdi i minst ett av attributtene i X
- **Kandidatnøkkel**: En minimal supernøkkel.
  Dvs: Fjerning av et hvilket som helst attributt fører til at de gjenværende attributtene ikke lenger utgjør en supernøkkel
- **Primærnøkkel**: En utvalgt blant kandidatnøklene.
  Alle relasjoner skal ha nøyaktig én primærnøkkel
- **Nøkkelattributt** (*prime attribute*): Attributt som er med i (minst) en kandidatnøkkel.

Supernøkler benyttes til å uttrykke integritetsregler

Påkrevde integritetsregler i relasjonsdatabaser

- **Entitetsintegritet**:
  Alle relasjoner skal ha en og bare en primærnøkkel
  Ingen av attributtene i primærnøkkelen får være nil
- **Referanseintegritet**:
  Hvis fremmednøkkelen ikke er nil, så skal det finnes et tuppel i den refererte relasjonen hvor primærnøkkelen har samme verdi som fremmednøkkelen (dvs. at det refererte tuppelet skal eksistere)
- **Domeneintegritet**:
  Alle verdier skal være atomære og hentet fra vedkommende attributts domene
  (Dessuten kan nil være tillatt «verdi» for noen attributter)
- I tillegg kan databasen ha andre integritetsregler, for eksempel kandidatnøkler som ikke er primærnøkler

Objektdatabaser

- Dataelementene er objekter
- Objektene kan stå i relasjon (relationship) til hverandre; slike relasjoner er alltid toveis
- Gjeldende standard er ODMG 3.0 (2000), men ingen har til nå laget en full implementasjon av denne
- Implementasjonsmessig er objektdatabaser en generalisering av nettverksdatabaser, mens det tilhørende spørrespråket OQL er en beregningskomplett SQL-etterligning (OQL returnerer et objekt; SQL returnerer en relasjon)

Objektrelasjonelle databaser

- Motivasjon: Utvide relasjonsmodellen med objekt-orienterte ideer
- Relasjonen beholdes som fundamental abstraksjon
  - bakoverkompatibilitet med eksisterende relasjonelle databasesystemer
- Tupler spiller rollen som objekter

NoSQL

- Motivasjon: Datamodeller for distribuerte databaser
- NoSQL er et paraplybegrep som omfatter en rekke ulike typer DBMSer
- Datamodellene under NoSQL avviker fra relasjons-modellen
  - Kolonneorientert modell
  - Dokumentorientert modell
  - Key-value store
  - Grafmodell

2-skjemaarkitektur

- IDS var en 2-skjemaarkitekturdatabase
  - Ett *skjema* ga en fullstendig beskrivelse av databasen
  - Ett *subskjema* tilpasset applikasjonen fungerte som applikasjonens vindu mot databasen
- Hver database hadde bare ett skjema, men kunne ha vilkårlig mange subskjemaer

Datauavhengighet

- 2-skjemaarkitekturen tilbød *datauavhengighet*:
  - Skjemaet definerte både struktur og lagringsformat for dataene
  - Subskjemaet definerte navn og format på de dataene applikasjonen brukte
  - Det ble dermed mulig å forandre lagrings-formatet uten å forandre programmene

3-skjemaarkitektur

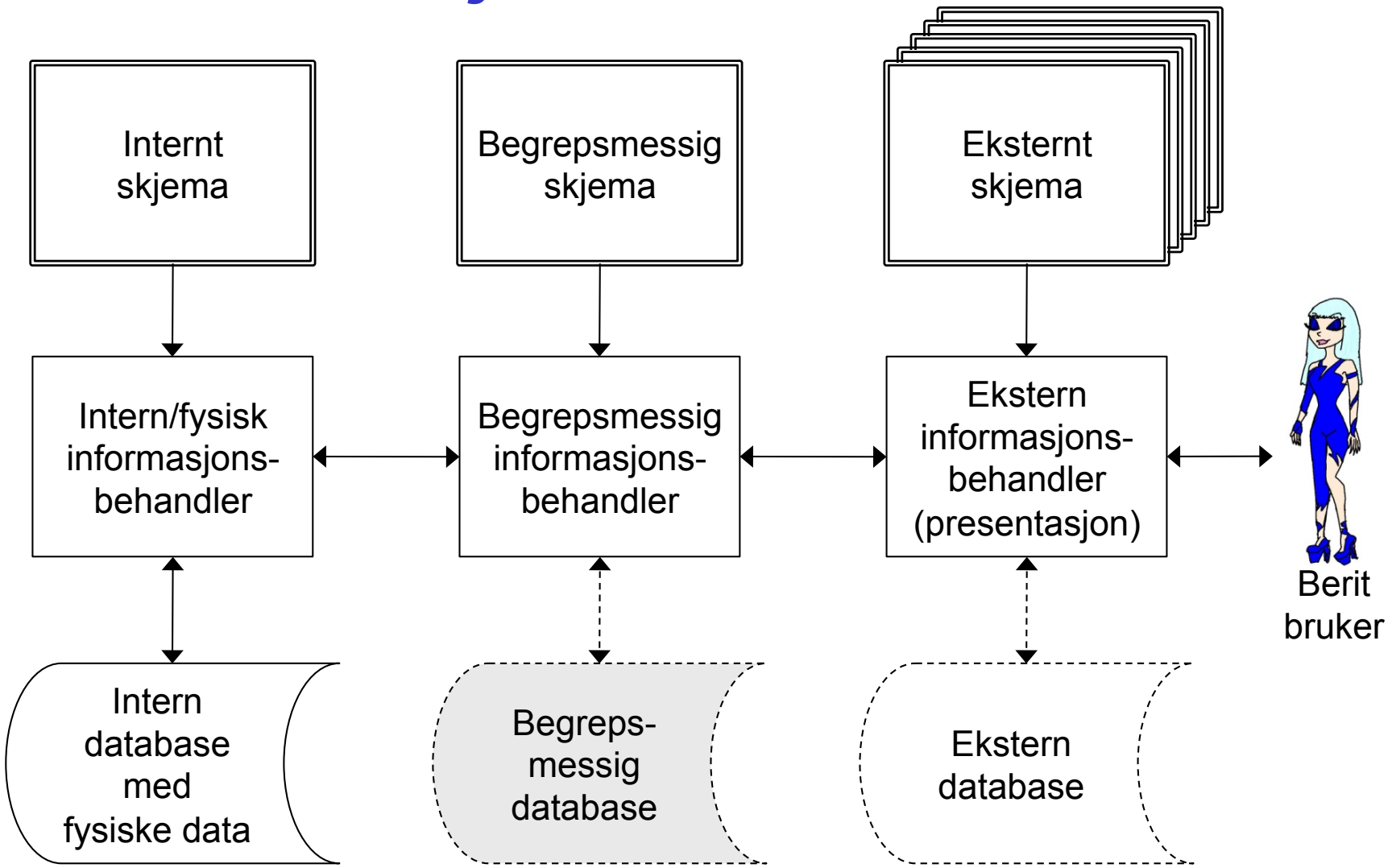
- I 1971 foreslo CODASYL DBTG (COmmittee on DAta SYstems Languages, Data Base Task Group) å splitte det sentrale skjema i to:
  - Et logisk (begrepsmessig) skjema
  - Et fysisk (internt) skjema
- ANSI/SPARC (ANSI Standards Planning And Requirement Committee) foreslo dette som standard i 1978
- Det begrepsmessige skjema ble ISO-standard i 1982 (sub-skjemaene ble her kalt eksterne skjemaer)
- Vi fikk dermed to former for datauavhengighet:
  - Fysisk mellom internt og begrepsmessig skjema
  - Logisk mellom begrepsmessig og eksternt skjema

3-skjemaarkitekturen

3-skjemaarkitekturen for databaser

- **Presentasjonslaget**
  beskrives med eksterne skjemaer («**views**») – hvordan informasjon skal presenteres for ulike brukere
- **Det logiske laget**
  beskrives i det begrepsmessige skjemaet – hva som kan lagres, og hva som er lovlige forandringer
- **Det fysiske laget**
  beskrives i det interne skjemaet – hvordan informasjon lagres, forandres og behandles

3-skjemaarkitektur



Fysisk og logisk datauavhengighet

- **Fysisk datauavhengighet**
  betyr at vi kan forandre det interne skjemaet så lenge det ikke strider mot det begrepsmessige skjemaet.
  Vi kan f.eks. forandre lagringsformatet av tall fra binært til BCD (Binary Coded Decimal) eller tekst uten å forandre det begrepsmessige skjemaet.
- **Logisk datauavhengighet**
  betyr at vi kan beholde eksterne skjemaer selv om vi forandrer det begrepsmessige skjemaet. Dermed slipper vi å forandre gamle programmer

3-lagsarkitektur i web-applikasjoner

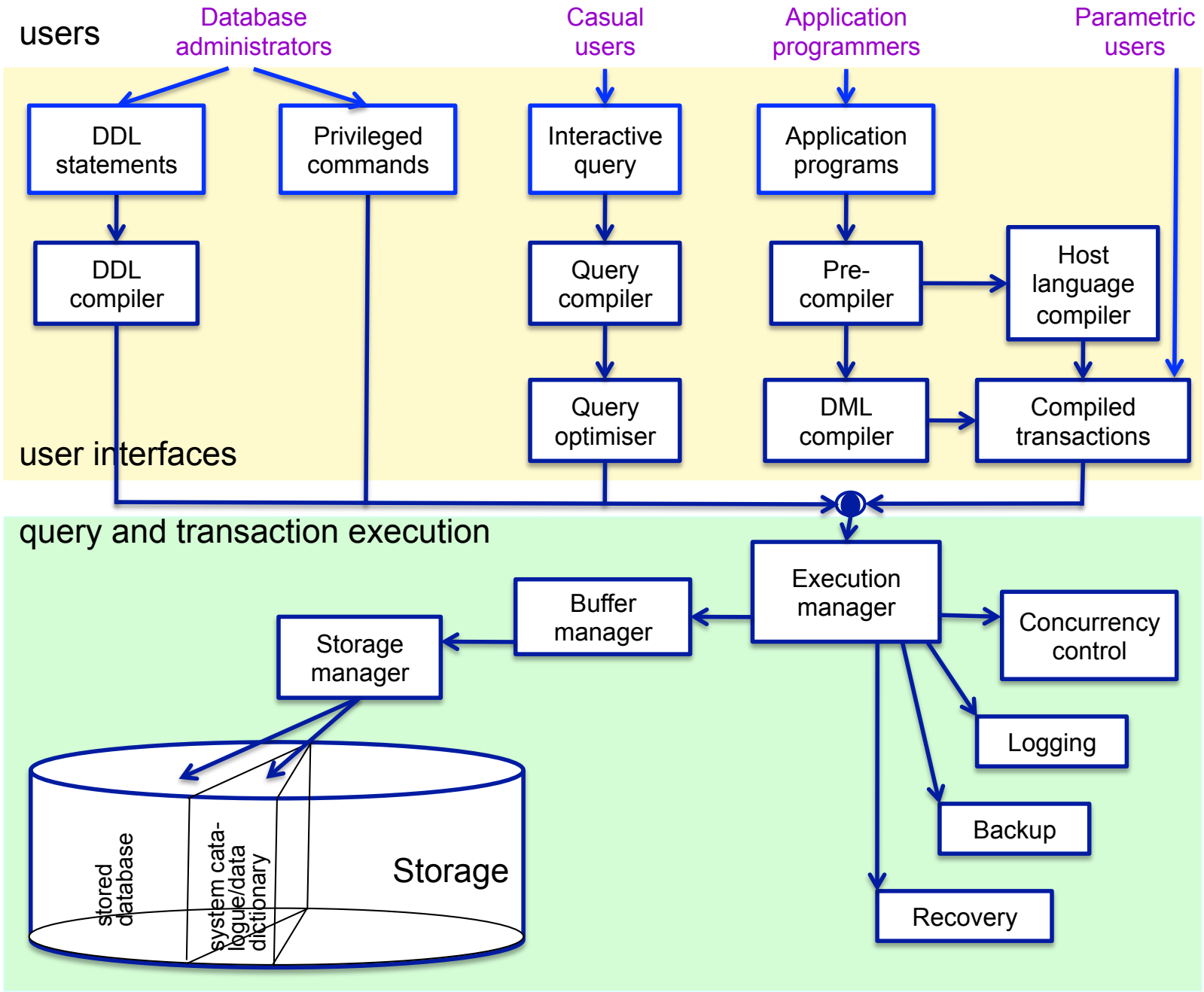
- Benytter ideene fra 3-skjemaarkitekturen i design av distribuerte systemer
  - Presentasjonslag: Webserver
  - Forretningslogikk: Applikasjonsserver
  - Datalag: Databaser, legacysystemer, …

Databasesystemers oppbygning

DBMS – Database Management System

- Spesialisert SW
- Karakteristika:
  - Persistens
  - Transaksjonshåndtering
    - **A** tomicity
    - **C** onsistency
    - **I** solation
    - **D** urability
  - Programmeringsgrensesnitt (API)

Typiske komponenter i et DBMS



Komponentene i et DBMS

- Øverste halvdel (gul): Brukergrensesnitt og tilhørende komponenter
- Nederste halvdel (grønn): De delene av DBMSet som står for datalagring og transaksjonsprosessering
- Brukere:
  - Databaseadministratorer (DBAer): Innehar spesielle privilegier
    - Opprettelse og endring av databaseskjemaer
    - Tuning av databasen
  - Tilfeldige brukere (casual users): Spørsmål (queries) til databasen
  - Applikasjonsprogrammerere: Programmering i et vertsspråk
  - Applikasjonsbrukere (parametric users): Spørsmål og innlegging av data via forhåndsdefinerte transaksjoner

Databaseadministratorer

- Skjemakommandoer (DDL statements)
  - DDL compiler:
    - Prosesserer skjemadefinisjoner beskrevet i et Data Definition Language (DDL)
    - Lagrer definisjonene i systemkatalogen
- Andre administrative kommandoer (privileged commands)

Tilfeldige brukere

- Brukergrensesnitt som kan ta imot spørringer (interactive query)
  - Query compiler: Parserer og sjekker spørringene, kompilerer dem til et internformat
  - Query optimiser: Omformer spørringene for å oppnå optimale eksekveringer, velger ut hvilke algoritmer som skal benyttes under eksekveringene. Resultatet er queryplaner som sendes til execution manager for utførelse

Applikasjonsprogrammerere. Applikasjonsbrukere

- Applikasjonsprogrammerere: Brukergrensesnitt som kan ta imot applikasjonsprogrammer skrevet i et vanlig programmeringsspråk (vertsspråk)
- Precompiler: Skiller ut databasespesifikke instruksjoner i applikasjonsprogrammet
  - Databaseinstruksjonene sendes til DML-kompilatoren (DML – Data Management Language) som lager objektkode for disse
  - Resten av applikasjonsprogrammet sendes til en kompilator for vertsspråket (host language compiler)
  - De to kompilerte delene lenkes til slutt sammen til eksekverbar kode – hermetiserte transaksjoner (canned transactions)
- Applikasjonsbrukere kan instansiere hermetiserte transaksjoner for å få utført faktiske transaksjoner mot databasen

Spørsmåls- og transaksjonsprosessering

- Execution manager mottar
  - administrative kommandoer
  - eksekverbare queryplaner
  - transaksjoner
- Transaksjonsprosessering involverer subkomponentene concurrency control, logging, backup og recovery
- Storage manager kontrollerer plassering av data på disk og flytting av data mellom disk og primærminnet. Buffer manager administrerer buffere i primærminnet

Ekstramateriale

Interesseområdet (UoD = Universe of Discourse)

- Lovene som styrer virkeligheten, kaller vi **forretningsregler**
- Forretningsregler og naturlover har mange likhetstrekk
- Vi ser effekten av dem, men de kan være vanskelige å finne

Informasjonsmodeller og skranker

- En fullstendig beskrivelse av interesseområdet kalles en **informasjonsmodell**
- Beskrivelsen av forretningsreglene kalles **skranker**
- *Statiske skranker* beskriver begrensninger på mulige tilstander i interesseområdet
- *Dynamiske skranker* beskriver begrensninger på mulige forandringer i interesseområdet

Det begrepsmessige skjema Integritetsregler

- Informasjonsmodellen brukt som regelverk (**preskripsjon**) for hvordan informasjonssystemet skal oppføre seg, kalles **det begrepsmessige skjema**
- Det begrepsmessige skjema uttrykkes i et språk som passer for den databaseteknologien vi skal bruke, f.eks. SQL (Structured Query Language) for relasjonsdatabaser
- I det begrepsmessige skjemaet kaller vi skrankene for **integritetsregler**
- Integritetsreglene bestemmer hva som er lovlig å *lagre* i informasjonssystemet (lovlige tilstander) og hva som er lovlige *forandringer* (lovlige transisjoner)

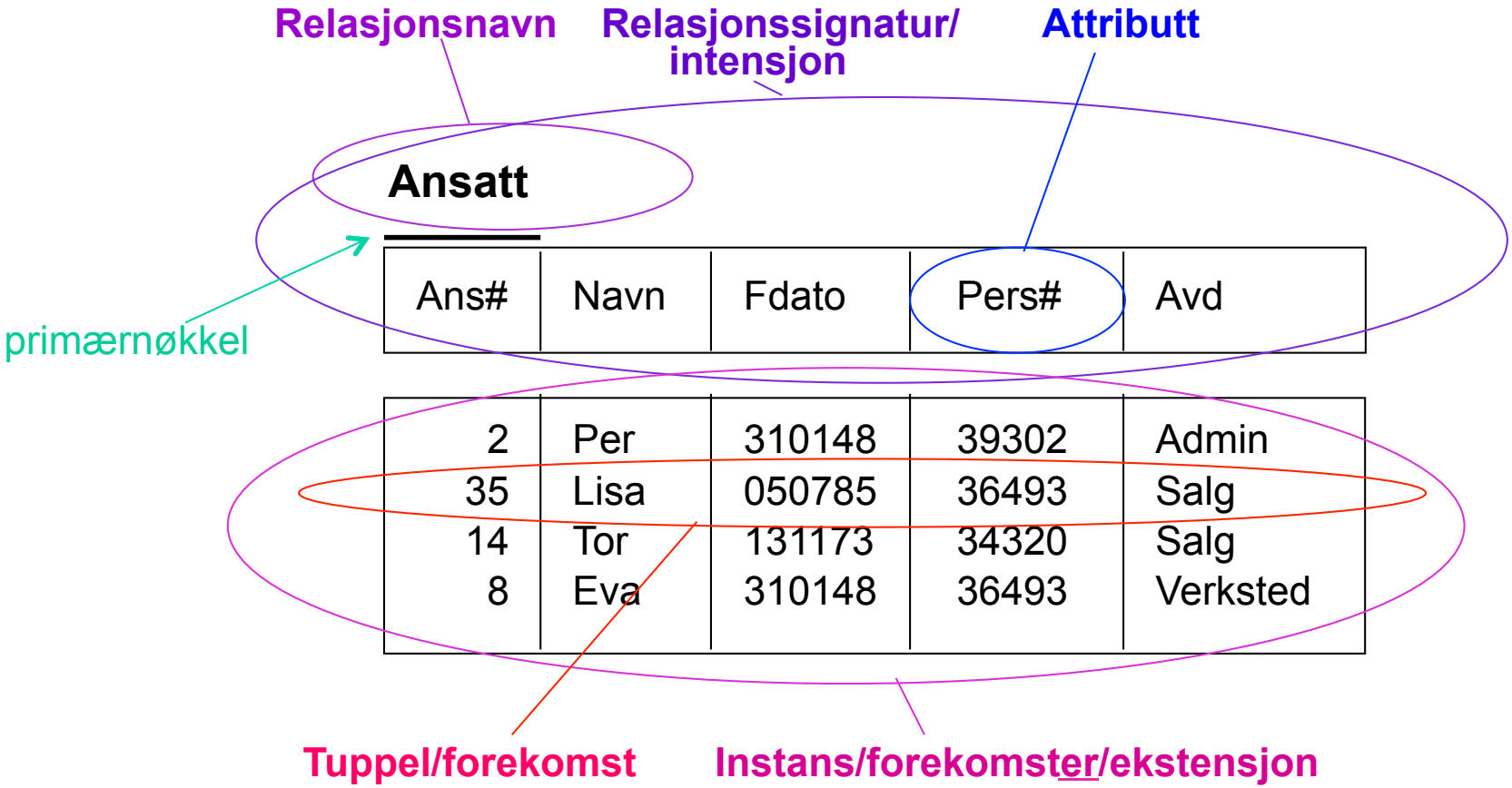
Relasjoner og relasjonsdatabaser

Ansatt

Ans#	Navn	Fdato	Pers#	Avd
2	Per	310148	39302	Admin
35	Lisa	050785	36493	Salg
14	Tor	131173	34320	Salg
8	Eva	310148	36493	Verksted

- **Relasjon**: Et matematisk begrep som kan tolkes som en tabell med verdier. Presist: En mengde av **tupler**
- **Relasjonsdatabase**: En samling relasjoner

Relasjoner - terminologi



Formelle definisjoner

- **Domene**: En mengde *atomære* verdier
- **Attributt**: Et navn på en rolle spilt av et domene («*kolonnenavn*»)
- **Relasjonssignatur** $R(A_1, A_2, \ldots, A_n)$: En navngitt mengde attributter $R = \{A_1, A_2, \ldots, A_n\}$ der R er **relasjonsnavnet**
  n kalles relasjonens *grad* eller *aritet*
- **Instans** av en relasjonssignatur $R(A_1, A_2, \ldots, A_n)$:
  En mengde $\{t_1, t_2, \ldots, t_m\}$ der hver t_k er et n-tuppel av verdier fra domenene til $A_1, A_2, \ldots, A_n$ (noen verdier kan være **nil**/**null**)
- **Relasjon**: Relasjonssignatur + tilhørende instans
  Relasjonssignaturen kalles relasjonens **intensjon**
  Instansen kalles relasjonens **ekstensjon**
- Vi slurver og sier «**relasjon**» selv om vi strengt tatt mener **relasjonssignaturen**

Relasjonsdatabaser - definisjoner

- **Databaseskjema** (omtales ofte som bare «**skjema**»):
  Samling av relasjonssignaturer + integritetsregler
- **Databaseinstans**:
  Samling av relasjonsinstanser
- **Database** = databaseskjema + databaseinstans

Fremmednøkler

Prosjekt

Pnavn	Deltaker
Budsjett2010	2
Budsjett2010	14
Budsjett2010	3
Kampanje0903	14
Kampanje0903	17

- Vi vil at Deltaker skal referere til forekomster av Ans# i **Ansatt**-tabellen

Fremmednøkler

Fremmednøkkel:
Ett eller flere attributter som sammen peker ut/refererer primærnøkkelen i en annen relasjon

Ansatt

Ans#	Navn	Fdato	Pers#	Avd
2	Per	310148	39302	Admin
35	Lisa	050785	36493	Salg
14	Tor	131173	34320	Salg
8	Eva	310148	36493	Verksted
3	Ida	190267	39421	Verksted
17	Odd	010482	34323	Salg

Prosjekt

Pnavn	Deltaker
Budsjett2010	2
Budsjett2010	14
Budsjett2010	3
Kampanje0903	14
Kampanje0903	17

Fremmednøkler

- Fremmednøkkelen må ha samme antall attributter som primærnøkkelen i den relasjonen den peker ut, og attributtene må ha parvis samme domener
  - Noen databasesystemer tillater fremmednøkler også til andre kandidatnøkler enn primærnøkkelen
- Korresponderende attributter behøver ikke å ha samme navn
- Det er lov å ha fremmednøkler til «seg selv»
- Fremmednøkler benyttes til å uttrykke integritetsregler