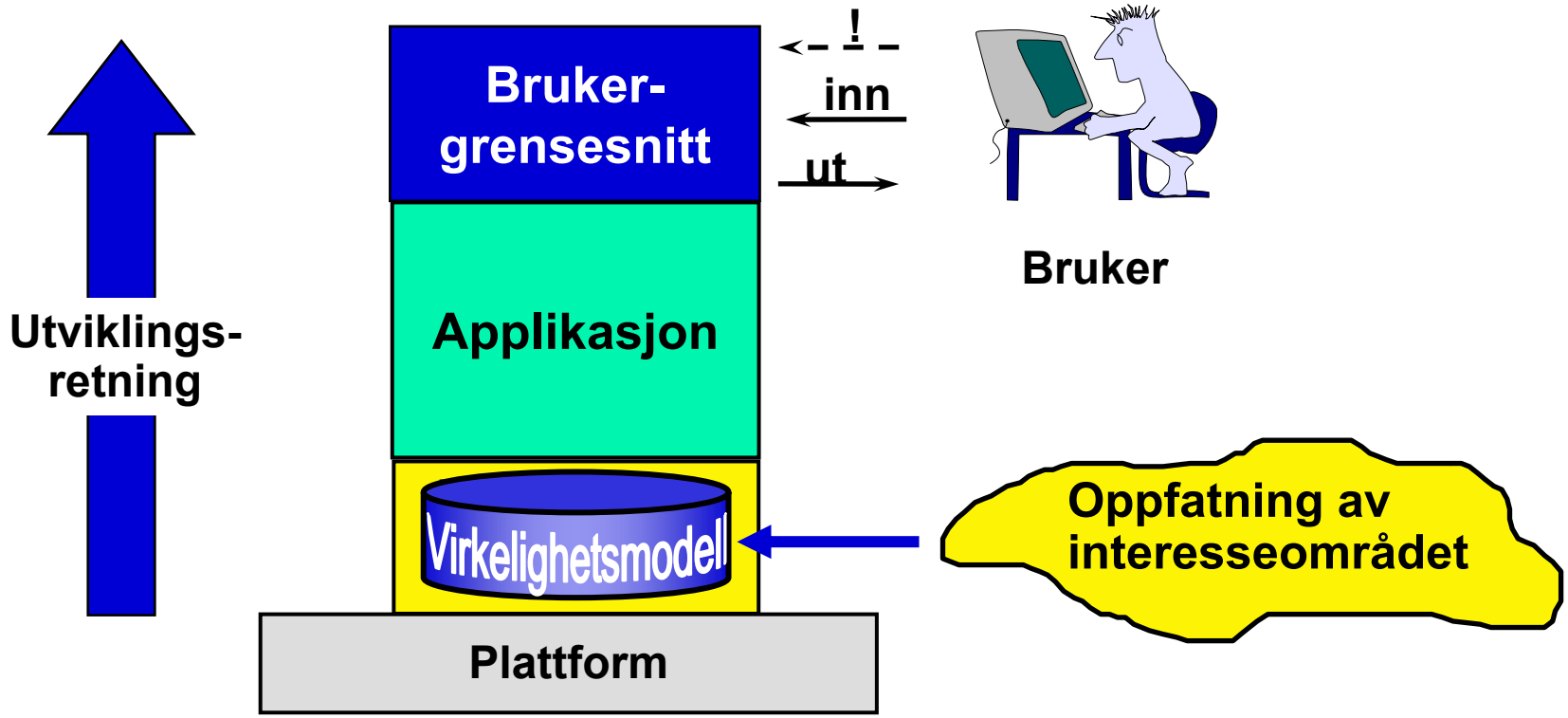
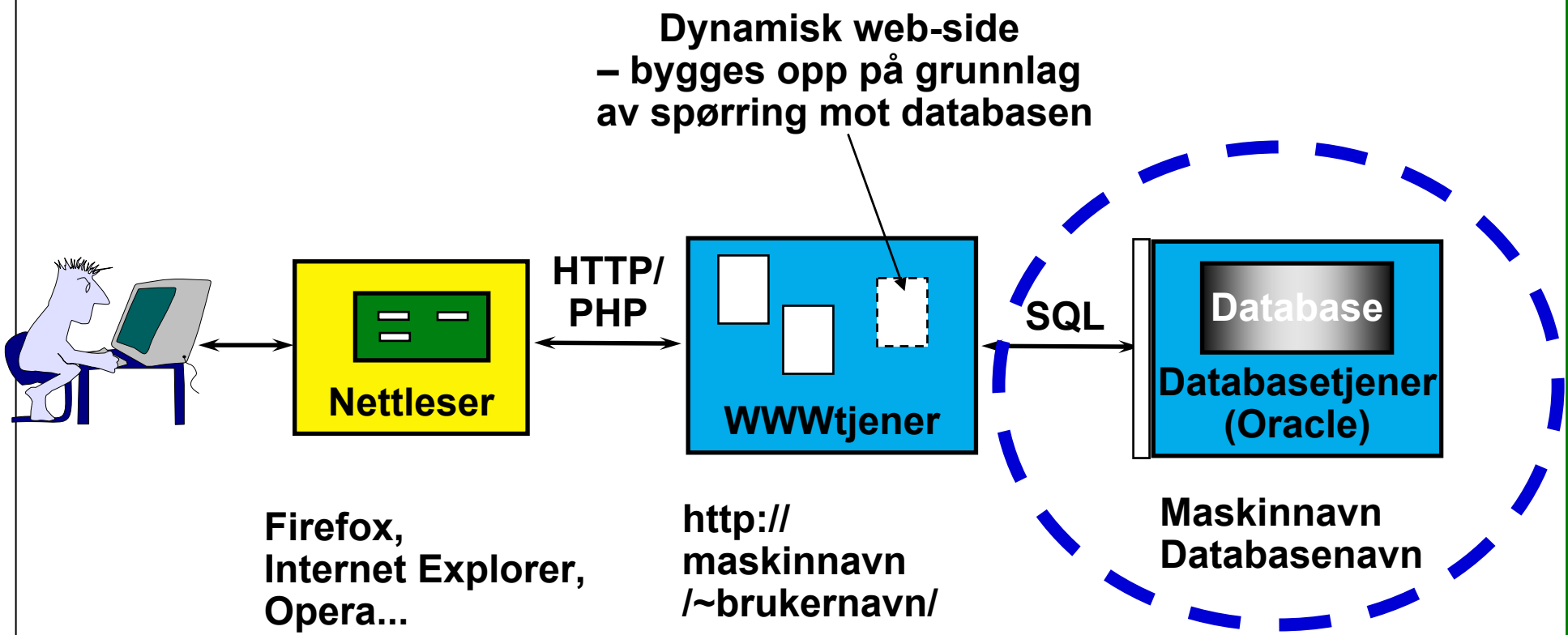
Utvikling fra kjernen og ut

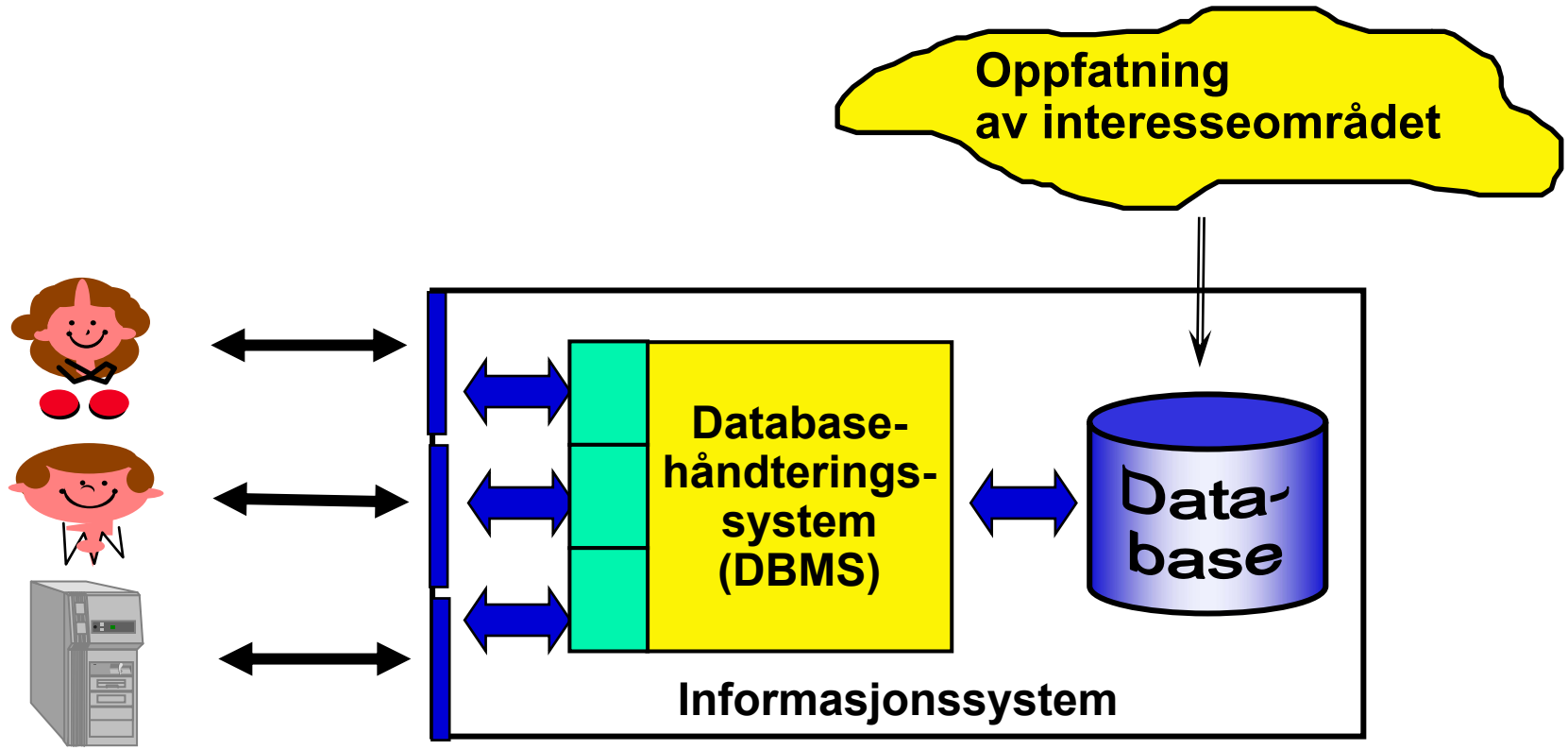


jfr. ”Systemutvikling – Fra kjernen og ut, fra skallet og inn” kapittel 4

PHP-arkitektur



Informasjonssystem bygd på et databasehåndteringssystem



”En database er en samling med data som er organisert for å tjene et bruksområde”

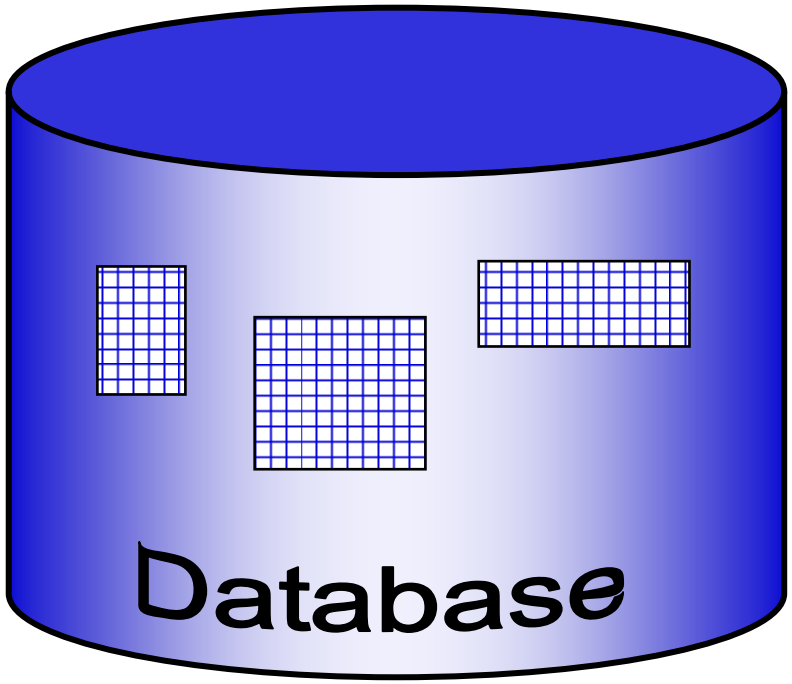
Hva gjør databasehåndteringssystemet?

- Tilbyr grensesnitt for brukere og programmerere
- Utfører (og optimaliserer) spørringer og oppdateringer
  - brukerdata
  - metadata (data om brukerdata)
- Håndhever skranker/integritetsregler (mer senere!)
- Håndterer flere brukere samtidig (gjelder ikke enbruker-DBMS)
- Gjennomfører oppdateringer som transaksjoner (mer senere!)
- Utøver tilgangskontroll
- Sikrer data

Databaseteknologier

- Hierarkiske databaser
- Nettverksdatabaser
- Tabelldatabaser/Relasjonsdatabaser
- Objektorienterte databaser

Relasjonsdatabaser



- I en relasjonsdatabase er data organisert i *tabeller*.
- Tabellene må oppfylle bestemte krav (mer senere).
- Flertallet av dagens informasjonssystemer bygd rundt databasehåndterings-systemer er basert på relasjonsdatabaser.

Relasjoner og relasjonsdatabaser

- **Relasjon**

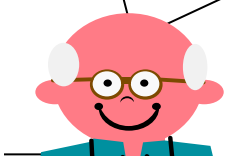
  Et matematisk begrep som kan tolkes som en tabell med verdier *der alle linjer i tabellen er forskjellige fra hverandre*

- **Relasjonsdatabase**

  En samling relasjoner

Relasjonsdatabaseteoriens fødsel:

E. F. Codd:
"A Relational Model for Large Shared Data Banks",
Communications of the ACM, Vol 13, Number 6 (June 1970)



Å regne med relasjoner

- Regning med tall (her: multiplikasjon): 3 * 2 → 6
- Regning med relasjoner (her: kartesisk produkt)

a1	a2
Per	17
Pål	19

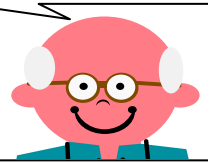
X

b1	b2	b3
Gro	18	X
Siv	19	Y
Åse	22	Z

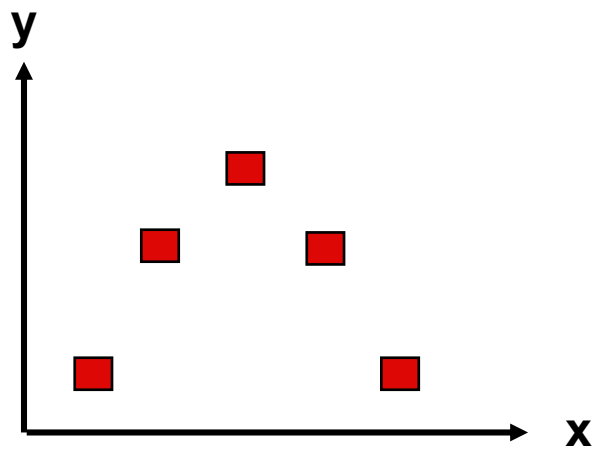
→

a1	a2	b1	b2	b3
Per	17	Gro	18	X
Pål	19	Gro	18	X
Per	17	Siv	19	Y
Pål	19	Siv	19	Y
Per	17	Åse	22	Z
Pål	19	Åse	22	Z

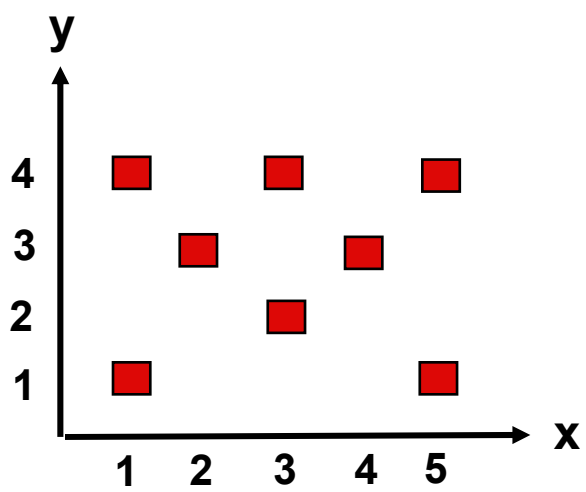
Operandene er hele relasjoner!



Hvorfor heter det “relasjon”?



- **Matematisk funksjon:**
  - **Til hver x hører bare én y**
    y = f(x)



x	y
1	1
1	4
2	3
3	2
3	4
4	3
5	1
5	4

- **Matematisk relasjon:**
  - **Til hver x kan høre flere y – og omvendt**
  - **Kan vises i en tabell med to kolonner – x og y**
  - **Kan generaliseres til n dimensjoner**

Figur 4-2. Relasjonsdatabaseterminologi

Verdiområde ("domain")

Relasjonsnavn

Attributt

Husholdningsavfall

Relasjonsskjema

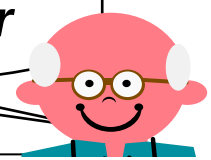
fylkenr	kommunenr	kommunenavn	avfallsmengde
01	0101	Halden	10228
01	0104	Moss	10423
01	0105	Sarpsborg	12600

Tupler/
Linjer/
Forekomster

Krav til relasjoner

- *Relasjonen* har et *entydig navn* innen databasen
- *Attributter* har et *entydig navn* innen relasjonen
- *Attributtenes rekkefølge* skal være uten betydning
- Verdiene er *atomiske*
- Alle verdier i et bestemt attributt er hentet fra *samme verdiområde*, eller er *NULL*
- Et verdiområde kan være endelig eller (teoretisk) uendelig
- *Tuplenes rekkefølge* skal være uten betydning
- I en relasjon kan det *ikke finnes to like tupler*

Mange av disse kravene er en konsekvens av at relasjonsdatabaseteorien definerer en relasjon som en matematisk mengde av likeartede tupler



Figur 4-4. Entydighetsskranke og primærnøkkel

Primærnøkkel

Entydighetsskranke

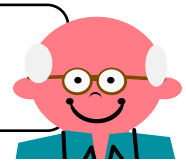
Alt. I

fylkenr	kommunenr	kommunenavn	avfalls-mengde	innbygger-tall
01	0101	Halden	10228	26417
01	0104	Moss	10423	25860
01	0105	Sarpsborg	12600	46692

Alt. II

fylkenr	kommunenr_ i_fylket	kommunenavn	avfalls-mengde	innbygger-tall
01	01	Halden	10228	26417
01	04	Moss	10423	25860
01	05	Sarpsborg	12600	46692

Entydighetsskranken håndheves av databasehåndteringssystemet!



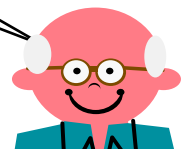
Fremmednøkkel

Et attributt – eller en attributtkombinasjon – som deler verdiområde med en primærnøkkel.

Fylke

fylkenr	fylkenavn
01	Østfold
02	Akershus
03	Oslo

Attributtene behøver ikke å ha samme navn - de må bare få sine verdier fra samme verdiområde



Husholdningsavfall

fylkenr	kommunenr	kommunenavn	avfallsmengde
01	0101	Halden	10228
01	0104	Moss	10423
02	0211	Vestby	11549

Referanseintegritet

Skranke **som uttrykker at verdiene i fremmednøkkelen også må finnes i primærnøkkelen.**
Håndheves av databasehåndteringssystemet.

Fylke

fylkenr	fylkenavn
01	Østfold
02	Akershus
03	Oslo

{subset}

Husholdningsavfall

fylkenr	kommunenr	kommunenavn	avfallsmengde
01	0101	Halden	10228
01	0104	Moss	10423
02	0211	Vestby	11549

Referanseintegriteten er her vist med en {subset}-skranke (delmengdeskranke)
Pass på at pilen peker riktig vei!

Skjema og forekomster (Intensjon og ekstensjon)

{subset}

Skjema

Husholdningsavfall

fylkenr	kommunenr	kommunenavn	avfallsmengde

Meta-data

Bruker-data

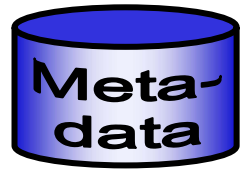
Databasens intensjon

Databasens ekstensjon

Linjer/ Forekomster

01	0101	Halden	10228
01	0104	Moss	10423
01	0105	Sarpsborg	12600

Også metadataene kan struktureres som tabeller



Tables

tablename	create_date
Fylke	20070127
Husholdningsavfall	20070127

{subset}

Attributes

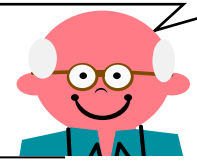
tablename	attributename	type	length
Husholdningsavfall	fylkenr	char	2
Husholdningsavfall	kommunenr	char	4
Husholdningsavfall	kommunenavn	char	25
Husholdningsavfall	avfallsmengde	int	

Formaliserte data

...er data som er under kontroll av et skjema

- **Fordeler med formaliserte data**
  - **Lettere bearbeiding**
  - **Mulighet for automatisering**
  - **Høyere datakvalitet**
  - **Styrende**
- **Ulemper med formaliserte data**
  - **Restriktivt**

"Riktig" skjema er viktig!

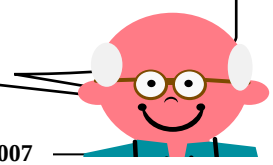


Tabeller kontra relasjoner

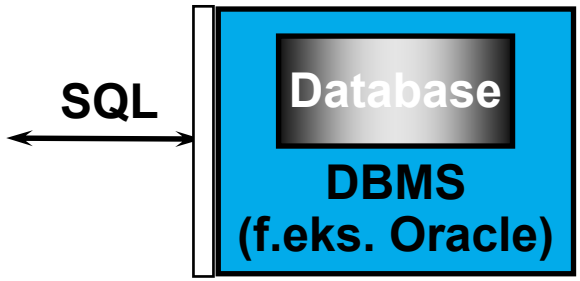
Teori er en ting, praksis er gjerne noe anna….:

- **I kommersielle databasehåndteringssystemer brukes tabeller – ikke relasjoner:**
  - **Like linjer er tillatt**
  - **Attributter er kvasi-atomiske**
- **Derfor snakker vi ofte om ”tabelldatabaser” istedenfor ”relasjonsdatabaser”.**
- **SQL er et programmeringsspråk for å håndtere *tabelldatabaser*.**

De to betydningsfulle avvikene er begrunnet i effektivitet og praktiske behov



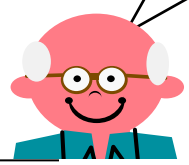
SQL - ”Structured Query Language”



ISO SQL-standardene

- **SQL-89**
  - **Originalstandarden**
  - **Addendum for Embedded SQL (1994)**
- **SQL-92**
  - **ISO/IEC 9075**
- **SQL:1999**
  - **ISO/IEC 9075:1999**

Det finnes i dag mer brukervennlige språk for å kommunisere med databaser, men SQL er standardisert....



SQLs tre kommandokategorier

- For forekomstmanipulering:
  DML – ”Data Manipulation Language” – Datamanipuleringsspråk
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- For skjemamanipulering:
  DDL – ”Data Definition Language” – Datadefinisjonsspråk
  - CREATE
  - ALTER
  - DROP
- For tilgangskontroll:
  DCL – ”Data Control Language” – Datakontrollspråk
  - GRANT
  - REVOKE

To eksempler på enkle SQL-spørringer

- Ta ut kolonner av en tabell

```
SELECT kommunenr, avfallsmengde
    FROM Husholdningsavfall;
```

Attributter vi ønsker å få ut

Tabell vi henter data fra

- Ta ut en hel tabell (lage en kopi)

```
SELECT *
    FROM Husholdningsavfall;
```

Alle attributtene

Legg merke til at resultatet av en spørring også er en tabell!

Filtrerende betingelser

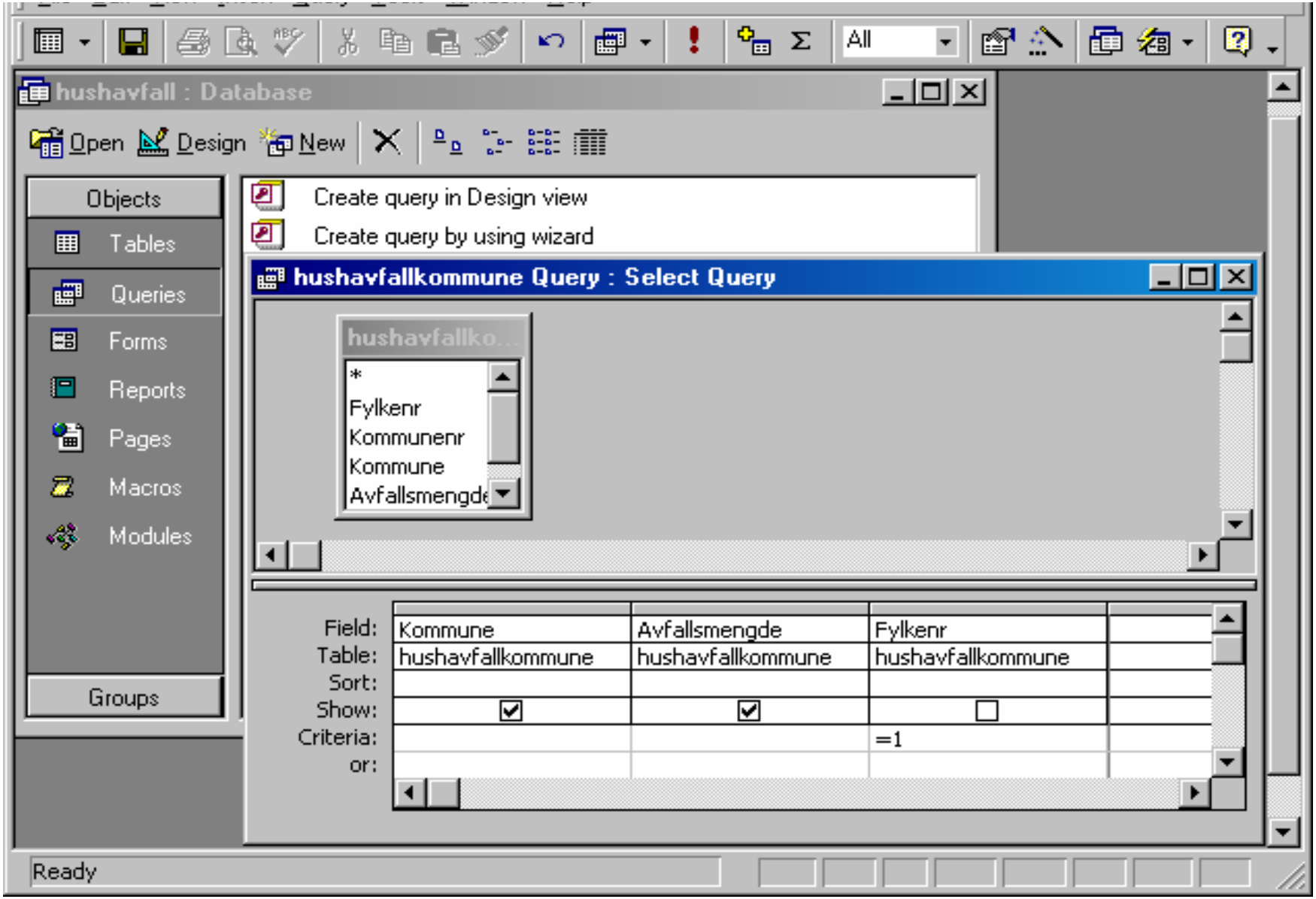
```
SELECT kommunenavn, avfallsmengde
    FROM Husholdningsavfall
    WHERE fylkenr = '01';
```

Attributter vi ønsker å få ut

Linjer vi ønsker å få ut

Bare linjer som oppfyller denne betingelsen vil komme med i resultatet

Figur 4-9 b. Eksempel på spørring i Query-by-example



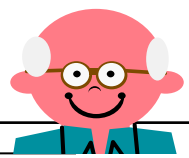
Spørringer mot flere tabeller (”join”)

```
SELECT Fylke.fylkenr,fylkenavn,kommunenr,
   kommunenavn,avfallsmengde
   FROM Husholdningsavfall, Fylke
   WHERE Husholdningsavfall.fylkenr=Fylke.fylkenr;
```

Legg merke til:

- *FROM-leddet inneholder flere tabeller*
- *WHERE-leddet inneholder join-betingelsen*
- *Det må være like mange join-betingelser som det er tabeller i FROM-leddet utover den første – hvis ikke får vi kartesiske produkter*
- *For å kunne referere til ulike attributter med samme navn kvalifiserer vi dem med tabellnavnet*

Resultat på neste lysark!



Eksempel på ”join”

Fylke

{subset}

fylkenr	fylkenavn
01	Østfold
02	Akershus
03	Oslo

SQL-kommandoen på forrige lysark

Husholdningsavfall

fylkenr	kommunenr	kommunenavn	avfallsmengde
01	0101	Halden	10228
01	0104	Moss	10423
02	0211	Vestby	11549

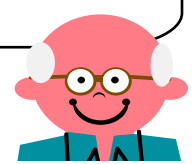
Resultat

fylkenr	fylkenavn	kommunenr	kommunenavn	avfallsmengde
01	Østfold	0101	Halden	10228
01	Østfold	0104	Moss	10423
02	Akershus	0211	Vestby	11549

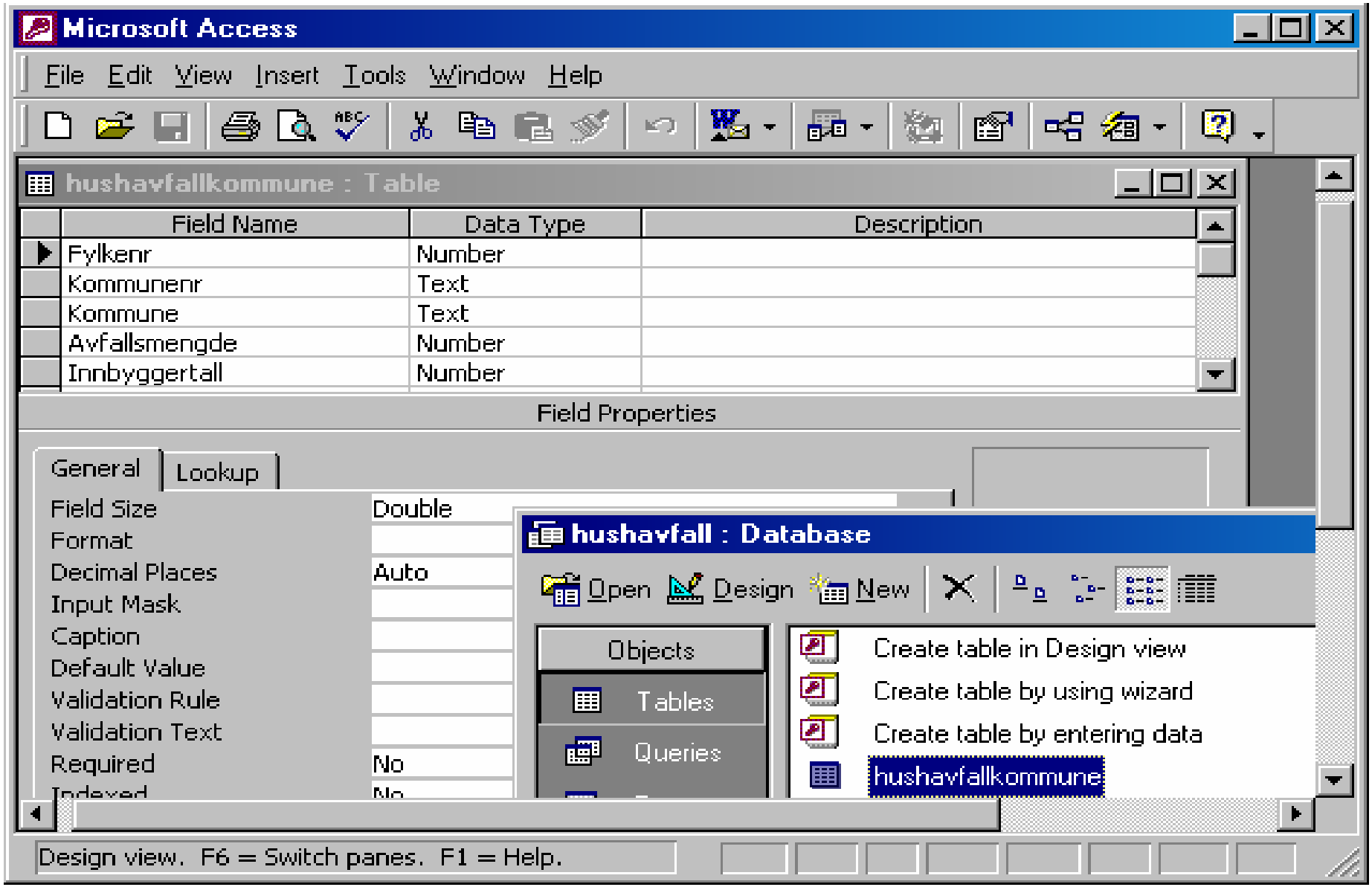
Oppretting av relasjonsdatabase med SQL

```
CREATE TABLE Husholdningsavfall (
Fylkenr              CHAR(2)              NOT NULL,
Kommunenr            CHAR(2)              NOT NULL,
Kommunenavn          VARCHAR(25)          NOT NULL,
Avfallsmengde INT                       ,
Innbyggertall INT                       NOT NULL,
CONSTRAINT EntydigKommunenr
     PRIMARY KEY (Kommunenr)
) ;


CREATE TABLE Fylke (… detaljene utelatt ) ;

ALTER TABLE Husholdningsavfall
ADD CONSTRAINT fylkenr_fk
 FOREIGN KEY(fylkenr) REFERENCES Fylke;
```

Rød tekst: Oppretting av skranker



Figur 4-8 b. Oppretting av relasjonsdatabase med grafisk dialog



Oppdatering – legge inn linjer

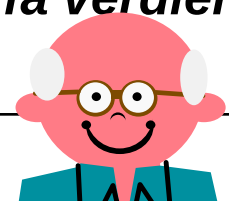
Nye linjer legges inn med **INSERT INTO:**

```
INSERT INTO tabellnavn VALUES(liste med verdier);
```

eller

```
INSERT INTO tabellnavn (liste med attributter) VALUES
   (liste med verdier);
```

I den første formen legges verdiene inn i tabellen fra venstre mot høyre (ikke å anbefale). Gjenværende attributter blir NULL.
I den andre formen velger vi hvilke attributter som skal få verdier, og i hvilken rekkefølge. Unevnte attributter blir NULL.



Legge inn linjer – eksempel

```
INSERT INTO Fylke VALUES ('13', NULL);
```

eller

```
INSERT INTO Fylke (fylkenr, fylkenavn)
  VALUES ('13', NULL);
```

eller

```
INSERT INTO Fylke (fylkenavn, fylkenr)
  VALUES (NULL, '13');
```

eller

```
INSERT INTO Fylke (fylkenr) VALUES ('13');
```

Oppdatering – fjerne linjer

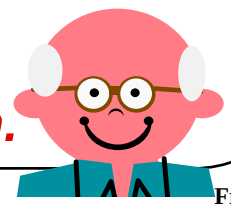
Eksisterende linjer fjernes med **DELETE FROM:**

```
DELETE FROM tabellnavn
WHERE betingelse for linjer som skal fjernes;
```

Eksempel:

```
DELETE FROM Fylke
WHERE fylkenr = '13';
```

SQL har ingen angrefunksjon!
Vær derfor sikker på at betingelsen er korrekt,
slik at det ikke fjernes linjer som ikke skulle vært fjernet.
En DELETE FROM uten WHERE fjerner alle linjer i tabellen.



Oppdatering – endre eksisterende linjer

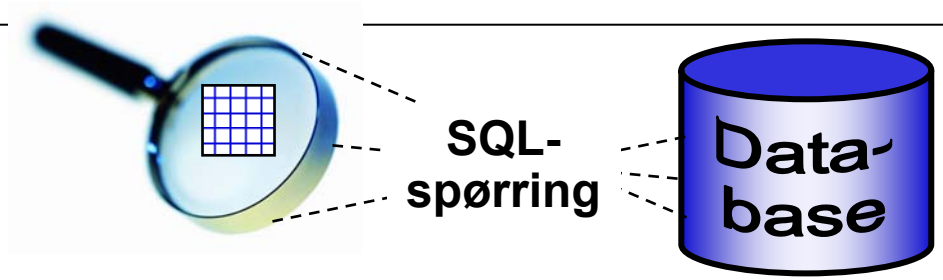
Eksisterende linjer endres med **UPDATE:**

```
UPDATE tabellnavn
   SET attributtnavn1 = Verdi eller uttrykk,
       attributtnavn2 = Verdi eller uttrykk,

           ...
   WHERE betingelse for linjer som skal endres;
```

Eksempel:

```
UPDATE Husholdningsavfall
   SET avfallPerInnbygger =
   avfallsmengde*1000/Innbyggertall
   WHERE innbyggertall > 0;
```

Virtuelle tabeller



- En virtuell tabell er et spørrings-resultat som kan brukes på lik linje med vanlige tabeller i etterfølgende SQL-kommandoer
- Forekomstene i virtuelle tabeller lagres ikke i databasen – endrer datagrunnlaget for spørringen seg, endres også den virtuelle tabellen
- Virtuelle tabeller brukes i hovedsak til:
  - ”Mellomregning” i kompliserte spørringer
  - Datagrunnlag for skjermbilder og applikasjonsprogrammer
  - Tilgangskontroll
- OBS:
  Begrensede muligheter for oppdatering av virtuelle tabeller

Bruk av virtuell tabell - eksempel

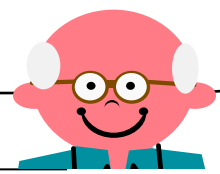
Vi oppretter en virtuell tabell:

```
CREATE VIEW AvfallAkershus AS
   SELECT Kommunenr,Kommunenavn,Avfallsmengde
   FROM Husholdningsavfall
   WHERE Fylkenr = '02' ;
```

Vi kan nå bruke den virtuelle tabellen:

```
SELECT * FROM AvfallAkershus ;
```

Mer SQL i gruppeundervisningen…
Se også "Systemutvikling
– fra kjernen og ut, fra skallet og inn"
Appendiks A



Applikasjonslag og brukergrensesnitt

Brukergrensesnitt

Applikasjon

OBS!

Virkelighetsmodell

Informasjonssystemets programmoduler

Ad-hoc spørring og oppdatering

(Query by Example/ SQL klientprogram)

Programmerbart SQL-grensesnitt (SQL API)

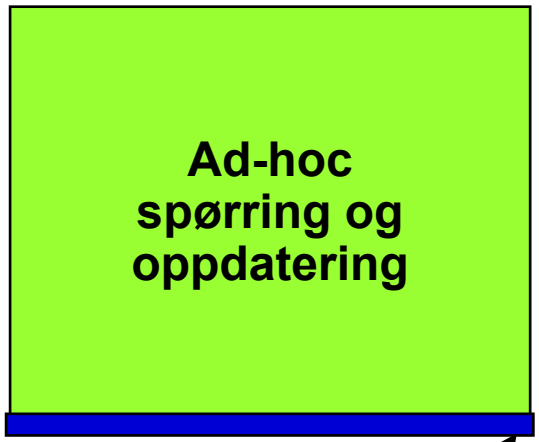
SQL

Databasehåndteringssystem

Operativsystem, datanettprogramvare

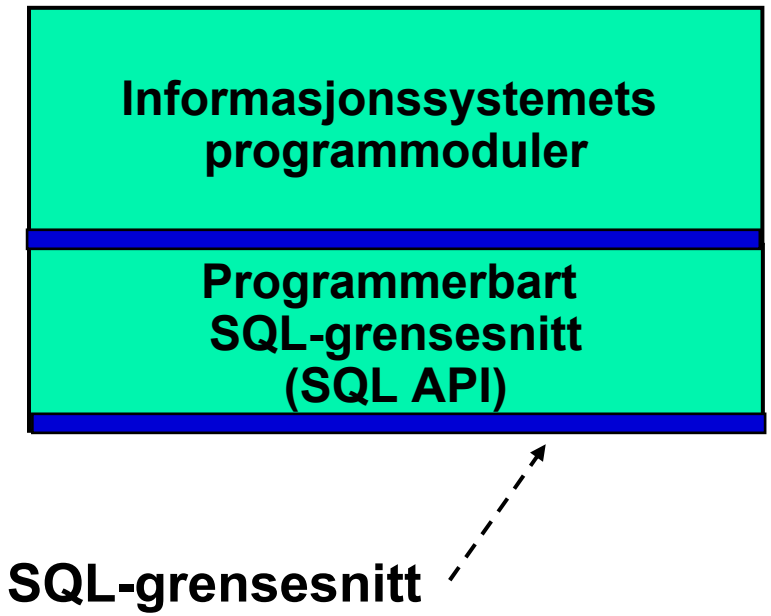
Maskinvare og datanett

Ad-hoc spørring og oppdatering



- Spørringer og oppdateringer som formuleres "på sparket"
- Formuleres enten i SQL. f.eks. i klientprogrammet SQL*Plus…
- …eller i et mer brukervennlig grensesnitt,
  f.eks. bygd på prinsippet "Query by Example"

Programmer med innebygde spørringer



- Kan genereres eller programmeres
- Forutsetter at programmeringsspråket har et programmerbart grensesnitt (API) for å kalle SQL-kommandoer
- Vær oppmerksom på "impedance-mismatch"-problemer

Database-transaksjoner

Transaksjon:

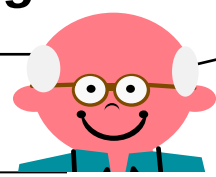
Et udelelig stykke arbeid mot databasen

Skal ideelt sett tilfredsstille "ACID"-kravene:

ACID

- **Atomiticy**
- **Consistency preservation (serializability)**
- **Durability**
- **Isolation**

Hva du gjør, gjør fullt og helt, og ikke stykkevis og delt!



Databaser og DBMS – en oppsummering

- ”En database er en samling med data som er organisert for å tjene et bruksområde”
  (definisjon fra Norsk Dataordbok)
- En database er selvbeskrivende:
  Den inneholder brukerdata og en beskrivelse av brukerdataene
- Metadata er brukerdatabeskrivelser og skranker
- En database håndteres av et databasehåndteringssystem
  (“Data Base Management System” - DBMS)
- Databasehåndteringssystemet letter tilgang, regulerer tilgang,
  gir samtidig tilgang for flere brukere, sikrer data
- Databasehåndteringssystemet sikrer at oppdateringer gjennomføres som transaksjoner
- Databasehåndteringssystemet optimaliserer spørringer
- Databasehåndteringssystemet håndhever skranker
- Moderne databasehåndteringssystemer kan håndtere
  både formaliserte og ikke-formaliserte data