12.2.1

a) Finn alle Laptop-elementer med pris mindre enn 800.

```
let $products := doc("products.xml")
for $laptop in $products//Laptop[@price < "800"]
return $laptop
```

b) Finn alle Laptop-elementer med pris mindre enn 800, og produser sekvensen av disse elementene omgitt av en tag <CheapLaptops>.

```
let $laptopSeq := (
    let $products := doc("products.xml")
    for $laptop in $products//Laptop[@price < "800"]
    return $laptop
)
return <CheapLaptops>{$laptopSeq}</CheapLaptops>
```

} Fra forrige oppg.

12.2.1

c) Finn alle produsenter hvor alle laptops har en pris på max 1000.

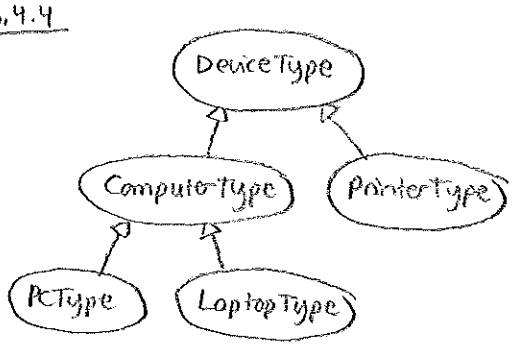
```
let $products := doc("products.xml")
for $maker in $products//Maker
where every $laptop in $maker/Laptop satisfies
        $laptop/@price <= "1000"
return $maker
```

d) Finn navn på alle produsenter som lager både PCer og laptops

```
let $products := doc("products.xml")
for $maker in $products//Maker
where count($maker/Laptop) > 0 and
      count($maker/PC) > 0
return $maker/@name
```

10.4.4

10.4.4 løst med SQL:1999




```
create type  ModelNameType as varchar(10);
create type  SpeedType as real;
create type  RamType as integer;
create type  HdType as integer;
create type  ScreenType as real;
create type  PriceType as real;
create type  ColorType as varchar(5);
create type  PrinterType as varchar(10);

create type  DeviceType as (
    model  ModelNameType,
    price  PriceType
)
ref is system generated;

create type ComputerType under DeviceType as (
    speed  SpeedType,
    ram    RamType,
    hd     HdType
);
create type  PCType under ComputerType as (
);

create type  LaptopType under ComputerType as (
    screen ScreenType
);

create type  PrinterType under DeviceType as (
    color  ColorType,
    type   PrinterType
);

create table Device of DeviceType (
    ref is deviceid system generated,
    primary key (model)
);
```

For at skal kunne tilordne objektidentifikator til relasjoner som er typet med DeviceType

subtype

Tabellen kan da ha objekter av alle subtyper av DeviceType, det er mer elegant enn den oppramsinge strukturen med tre tabeller (en for hver av pc, laptop, printer)

10.4.4 (forts.)

```
create type MakerNameType as varchar(20);
create
```

```
create type ProductType as (
    maker MakerNameType,
    model ref (DeviceType)
);
```

Trenger ikke lenger et attributt "type" som brukes til å avgjøre hvilken tabell opplysninger om produktet fins i, det kan vi gjøre ved å se på hvilken subtype objektene i tabellen Device har.

```
create table Product of ProductType (
    primary key (maker, model),
    model with options scope Device
);
```

← Restriktor til å referere DeviceType-objekter i Device-relasjonen

Burde kanskje valgt et annet navn for ikke å blande sammen med model i DeviceType...

10.5.2

10.5.2 løst med SQL:1999

a) Produsenter av PCer med hd > 80 GB:

```
select distinct p.maker
from Product p
where deref(p.model) is of (PCType) and
      p.model -> hd > 80;
```

} restriktor til objekter av rett subtype i Device-tabellen

peker til objekt av PCType

b) Produsenter av laserprintere:

```
select distinct p.maker
from Product p
where deref(p.model) is of (PrinterType) and
      p.model -> type = 'laser';
```

c) ?? var oppgavebeskrivelse .. Mones for hver produsent (manufacturer) den/de laptopene som har høyest ram? I såfall:

```
select q.maker, q.model->model
from (select p.maker, max(p.model -> ram) as maxram
      from Product p
      where deref(p.model) is of (LaptopType)
      group by p.maker) mp,
      Product q
where q.maker = mp.maker and
      q.model -> ram = mp.maxram;
```

10.4.3 Løst med PostgreSQL

10.4.3 PostgreSQL:

```
create type classtp as (
    class varchar(20),
    classtype char(2),
    country varchar(15),
    numGuns integer,
    bore integer,
    displacement integer
);
```

← lager som type for å se hvordan det funker. Får da ikke noen primary key (det brukes bare for tabeller). Burde kunnet skrive :: check (classtype = 'bb' or ...) her, men det protesterer systemet på. Så det kommer her, i stedet.

```
create table ship (
    name varchar(20) primary key,
    class classtp check (classtype = 'bb' or classtype = 'bc'),
    launched integer
);
```

← bruker altså en record som verdi her, så bryter INF

```
create table battle (
    name varchar(30) primary key,
    duration interval
);
```

← interval kan brukes for å angi fra-til

```
create table outcome (
    ship varchar(20) references ship(name),
    battle varchar(30) references battle(name),
    result varchar(7)
        check (result = 'sunk' or result = 'ok' or result = 'damaged'),
    primary key (ship, battle)
);
```

← antar at et skip kan være med i flere slag

Eks: ship

name	class	launched
Haruna	(Kongo, bc, Japan, 8, 14, 32000)	1915
Hiei	(Kongo, bc, Japan, 8, 14, 32000)	1914
Kongo	(Kongo, bc, Japan, 8, 15, 32000)	1913

feil – men modellen hindrer ikke dette (*)

(*) Ulempen med modellen er at det er ingen kontroll på at klassebetingelsen brukes "riktig": info/record med klasseopplysninger legges inn direkte under hvert skip, så vi får duplisering av data og fare for å skrive feil i noen av recordene (kan f.eks. ha info om class Bismarck hvor f.eks. 'numGuns' eller 'bore' på tvers av skipstuplet har forskjellige verdier selv om det gjelder samme klasse og altså burde hatt identiske verdier). På den annen side: Oppgavene under 10.5.3 blir enkle; man har direkte tilgang til klassedata for hvert skip uten joinoperasjoner.

10.4.4 løst med PostgreSQL

10.4.4 PostgreSQL: Har ikke reftype. Reformulert oppgave:

"Redesign produktdatabasen fra oppgave 2.4.1. Bruk typedeklarasjoner der dette passer. Utnytt muligheter for arv til å strukturere databasen bedre."

```
create table product (                    ← (lager automatisk en tilhørende
    maker varchar(20),                      type product)
    model integer,
    type varchar(10),
    primary key (maker, model)
);

create table PC (                         ← (lager automatisk typen PC)
    speed numeric,
    ram integer,
    hd integer,
    price integer,
    primary key (maker, model)            ← i PostgreSQL 8.4 arves ikke nøkler
) inherits (product);                     ← arver attributtene fra product

create table Laptop (
    speed numeric,
    ram integer,
    hd integer,
    screen numeric,
    price integer,
    primary key (maker, model)
) inherits (product);

create table Printer (
    color boolean,
    printertype varchar(10),              ← kan ikke gjenbruke 'type' som
    price integer,                          attributtnavn (product har et attributt type)
    primary key (maker, model)
) inherits (product);
```

NB Bedre forslag neste side!

10.4.4 (forts).

Bedre:

```
create table product (
    maker varchar(20),
    model integer,
    type varchar(10),          ← kan vurdere å droppe dette attributtet
    price integer,             ← alle har en pris, så ha den heller her
    primary key (maker, model)
);

create table computer (        ← felles for PC og Laptop
    speed numeric,
    ram integer,
    hd integer,
    primary key (maker, model)
) inherits (product);

create table PC (
    primary key (maker, model)
) inherits (computer);

create table Laptop (
    screen numeric,
    primary key (maker, model)
) inherits (computer);

create table Printer (
    color boolean,
    printertype varchar(10),
    primary key (maker, model)
) inherits (product);
```

← intet ekstra i forhold til computer (ingen nye attributter). Logisk likevel nyttigere/ryddigere å ha en egen tabell og ikke simpelthen legge alle PCer i computer-tabellen.

Antakelig skal ingen tupler legges inn i table product, og heller ikke i table computer; disse er å anse som 'abstrakte' i SQL:1999-terminologi, men jeg tror ikke PostgreSQL har noen mulighet for å definere "abstrakte tabeller".

10.5.2 PostgreSQL: (bruker 2. versjon av løsningsforslagene i 10.4.4)

a) Produsenter av PCer med harddisk over 80 Gb:

```
select distinct maker
from PC
where hd > 80;
```

b) Produsenter av laserskrivere:

```
select maker
from Printer
where printertype = 'laser';
```

c) For hver laptopmodell den laptopmodellen som har størst RAM fra samme forhandler:

```
select L.maker, L.model, K.model
from (select maker, max(ram) as maxram
      from Laptop
      group by maker) as M,
     Laptop L, Laptop K
where L.maker = M.maker and
      K.maker = M.maker and K.ram = M.maxram;
```

} de med mest RAM pr. forhandler

L er alle laptoper, hver matches med de (angitt av K) som har maks ram – men kan ikke bare bruke M fordi jeg ikke kan få ut modellen også under gruppering.

10.5.3 PostgreSQL:

10.5.3 løst med PostgreSQL

a) Slag hvor minst ett skip ble ødelagt:

```
select distinct battle
from outcome
where result = 'damaged';
```

b) Skip med minst 40000 tonn slagvolum (displacement):

```
select name
from ship
where (class).displacement >= 40 000;
```

c) Klasser med skip som ble lansert etter 1926:

```
select distinct (class).class
from ship
where launched > 1926;
```

d) Slag hvor minst ett engelsk skip ble ødelagt:

```
select distinct b.battle
from outcome b, ship s
where b.ship = s.ship and b.result = 'damaged' and
      (s.class).country = 'Gt. Britain';
```

2.5.1

a) PC med hastighet $< 3{,}00$ må være priset $\leq \$800$.

$$\sigma_{speed < 3.00 \text{ and } price > 800}(PC) = \emptyset$$

[Brytes av eksempeldataene, se modell 1001: speed = 2.66, pris $2114.]

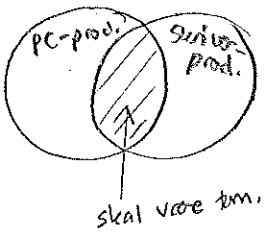
b) Laptop med skjerm mindre enn 15,4" må ha $\geq$ 120 Gb harddisk eller pris $< \$1000$.

Det betyr at mengden av laptops med skjerm $< 15{,}4"$, harddisk < 120 Gb og pris $\geq \$1000$, skal være tom:

$$\sigma_{screen < 15.4 \text{ and } hd < 120 \text{ and } price >= 1000}(Laptop) = \emptyset$$

[Brytes av eksempeldataene, se modell 2004: screen = 13,3, hd = 60, price = 1150]

c) Ingen PC-produsenter får lage skjermer.



Tmyk

$$\left(\sigma_{type='pc'}(Product)\right) \cap \left(\sigma_{type='printer'}(Product)\right) = \emptyset$$

Tmyk

[Brytes av eksempeldataene; produsent D lager begge typer.]

2.5.1

d) Hvis laptop med større memory enn en PC, skal prisen også være høyere.

Idé: Finn par av laptoper og PCer hvor laptopen har størst memory. Finn par hvor laptopen har høyest pris. Førstnevnte mengde skal være inneholdt i sistnevnte.

$L(m, s, r, h, sc, p) := Laptop;$ (renaming)

Answer(...) :=

$$\sigma_{r > ram}(L \times PC) \subseteq \sigma_{p > price}(L \times PC)$$

[Motseksempel i eksempeldataene:

Laptop 2002 har ram = 1024 og price = 949
PC 1002 har ram = 512 og price = 995]

2.5.1

e) En PC-produsent må også produsere en laptop med minst like stor speed.

laptop med større / enn PCen

Kan lage par av PCer og laptoper med hensyn på hastighet for samme produsent, projisér på PC, skal da ha noe som omfatter samtlige PCer.

Laptoper → $R(m, s) := \Pi_{maker, speed}(Product \bowtie Laptop)$

PCer → $S(maker, speed) := \Pi_{maker, speed}(Product \bowtie PC)$

PCer med en Laptop m/høyere speed fra samme produsent → $T(maker) := \Pi_{maker}(S \bowtie R)$

maker=m
and
speed<=s

Svar: $\Pi_{maker}(S) \subseteq T$

Alle PCer

Alle PCer hvor det er en Laptop med høyere hastighet

20.5.2

(i, j, M): node i sender M til node j, M ∈ {P, R, D, C, A}

Node 0 er koordinator, global transaksjon med subtransaksjoner på node 1 og 2.

a) Eksempel på sekvens av meldinger der node 1 vil cammite og node 2 aborterer:

(0, 1, P)
(0, 2, P)
(1, 0, R)
(2, 0, D)
(0, 1, A)
(0, 2, A)