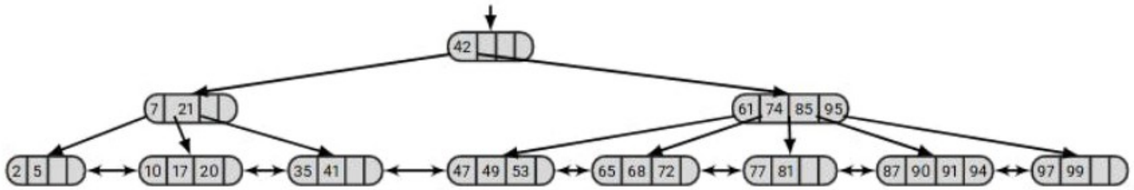
Gruppetime 13 - Indekser og spørreprosessering, repetisjon

Av Katrine <3

Hvordan leses det fra databasen?

- Databaser bruker datastrukturer for å søke mer effektivt
- Indeksstrukturer finner rader basert på verdien i en eller flere kolonner
- Databaseindekser lagres på disk (DDR eller SSD) og ikke i minne (RAM)
- Indeksstrukturer er optimert for å utføre så få diskoppslag som mulig
- De to indeks-hovedtypene:



B-tre-indekser

- Trestruktur, hvor node kan ha flere barn
- Nodene har samme størrelse som en disk-blokk
- (Disk-blokk er en sekvens med bits eller bytes som inneholder rader)
- Denne strukturen minimerer antall oppslag på disk
- Hver verdi i løvnodene har pekere til den tilhørende rad i den tilhørende tabellen
- Gir effektive oppslag på konkrete verdier, samt effektive intervall søk

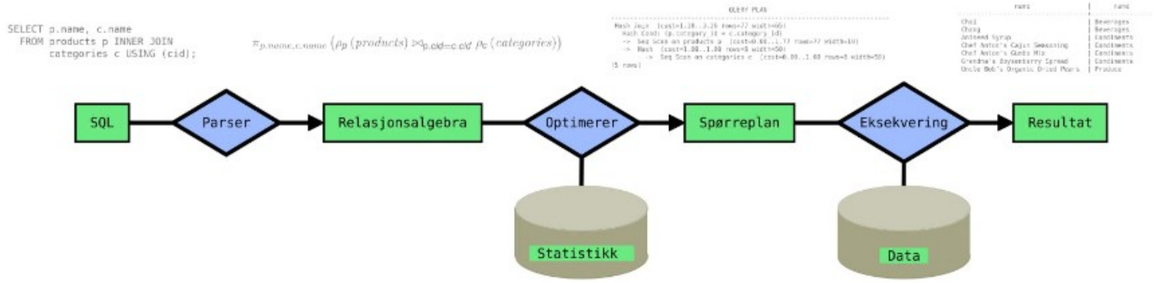
Hash-indekser

- Bruker en hash-funksjon for å oversette en verdi til en minneadresse
- (En hash-funksjon "mapper" data av tilfeldig størrelse til en verdi av en gitt størrelse)
- På minneadressen ligger en liste pekere til rader med denne verdien
- Mer effektive på oppslag på konkrete verdier
- Men kan ikke brukes for intervaller, siden det krever oppslag for hver mulig verdi i intervallet

```sql
DROP TABLE IF EXISTS t1;
DROP TABLE IF EXISTS t2;

CREATE TABLE t1(id int);
INSERT INTO t1
SELECT n*random() -- Genererer 10 mill. tilfeldige tall
FROM generate_series(1, 10000000) AS x(n);

CREATE TABLE t2 AS (SELECT * FROM t1); -- t2 inneholder samme data som t1

CREATE INDEX t1_ind ON t1(id); -- Lager index på t1

-- Følgende gjør at psql printer ut hvor lang tid spørringer tar
\timing on

SELECT * FROM t1 WHERE id = 100; --> Time:   0.792 ms
SELECT * FROM t2 WHERE id = 100; --> Time: 303.022 ms
```

Indekser med SQL

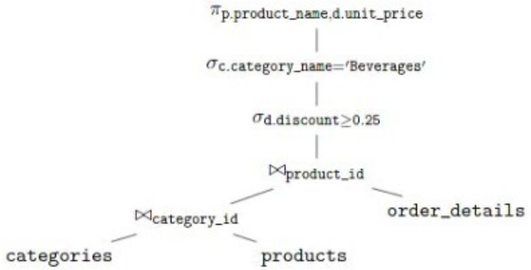
- CREATE INDEX <indeks_navn_her> ON <tabellen> (<kolonner>);
- Lages en passende indeks
- CREATE INDEX priser ON produkt(enhet_pris, enhet_navn);
- Flere kolonner gjør indeksen til en indeks over alle samtidig
- Databasen finnes selv ut når indeksen bør brukes



Spørreprosessering

- Spørreprosessering er alle stegene en spørring går gjennom til den blir evaluert
- Det første er parsing, som sjekker syntaksen

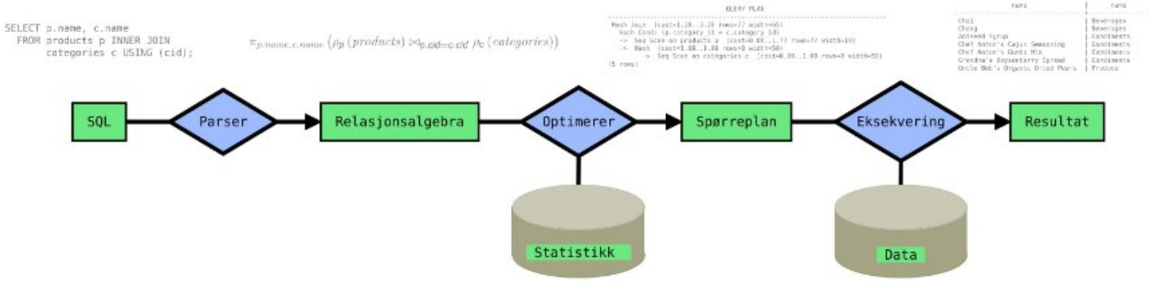
```sql
SELECT p.product_name, d.unit_price
FROM categories AS c JOIN
  products AS p USING (category_id) JOIN
  order_details AS d USING (product_id)
WHERE c.category_name = 'Beverages'
  AND d.discount >= 0.25;
```



$$\pi_{\text{p.product\_name,d.unit\_price}}\Big(\sigma_{\text{c.category\_name}='\text{Beverages}'}\Big(\sigma_{\text{d.discount}\geq 0.25}(\text{categories} \bowtie_{\text{category\_id}} \text{products} \bowtie_{\text{product\_id}} \text{order\_details})\Big)\Big)$$

Oversettelse til relasjonsalgebra

→ Deretter oversettes et til et spørringstre over relasjonsalgebraen

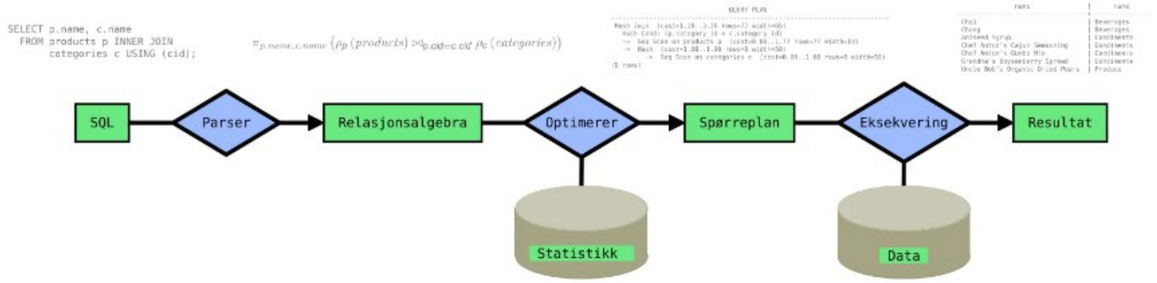


Optimering

- Grunnen til at du får samme resultat fra forskjellige spørringer
- Nå har vi spørringen på algebraisk form, som betyr at den kan manipuleres med algebra
- Dette brukes til å lage forskjellige spørringer med likt svar
- Som dere kanskje har lagt merke til, både med SQL og koding, så kan du få forskjellig kjøretid basert på kompleksitet
- Hver av dem tilordnes en verdi, og den med minst verdi er den som blir eksekvert
- Verdien gir ved bruk av statistikk over databasen, som antall rader og ulike verdier, skranker og indeksstruktur

```
psql=> EXPLAIN SELECT p.product_name, d.unit_price
FROM categories AS c JOIN
  products AS p USING (category_id) JOIN
  order_details AS d USING (product_id)
WHERE c.category_name = 'Beverages'
  AND d.discount >= 0.25;
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Hash Join  (cost=3.32..43.04 rows=20 width=21)
   Hash Cond: (d.product_id = p.product_id)
   ->  Seq Scan on order_details d  (cost=0.00..38.94 rows=154 width=6)
         Filter: (discount >= '0.25'::double precision)
   ->  Hash  (cost=3.20..3.20 rows=10 width=19)
         ->  Hash Join  (cost=1.11..3.20 rows=10 width=19)
               Hash Cond: (p.category_id = c.category_id)
               ->  Seq Scan on products p  (cost=0.00..1.77 rows=77 width=21)
               ->  Hash  (cost=1.10..1.10 rows=1 width=2)
                     ->  Seq Scan on categories c  (cost=0.00..1.10 rows=1 width=2)
                           Filter: ((category_name)::text = 'Beverages'::text)
(11 rows)
```

Spørreplan



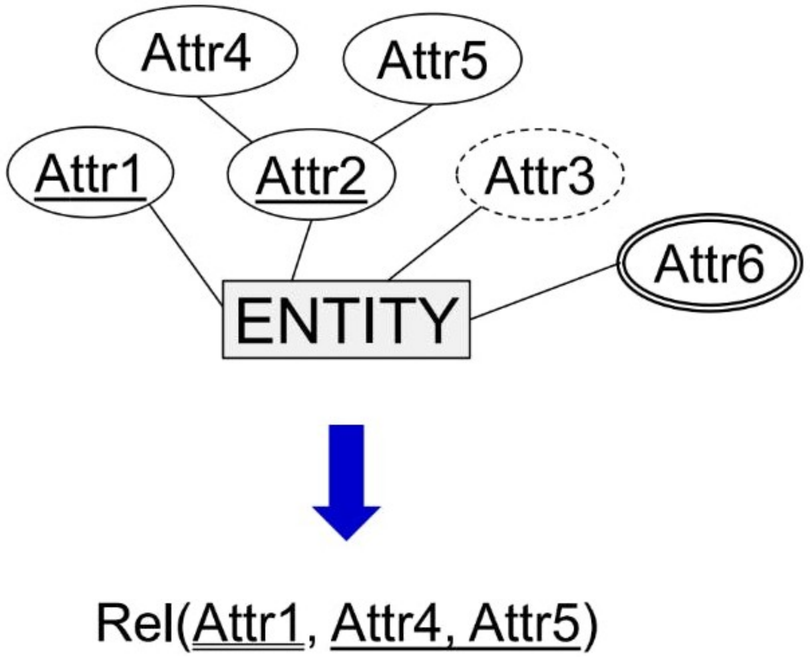
Evaluering

→ Og da har vi nådd eksekveringen, hvor spørringen blir evaluert over databasen

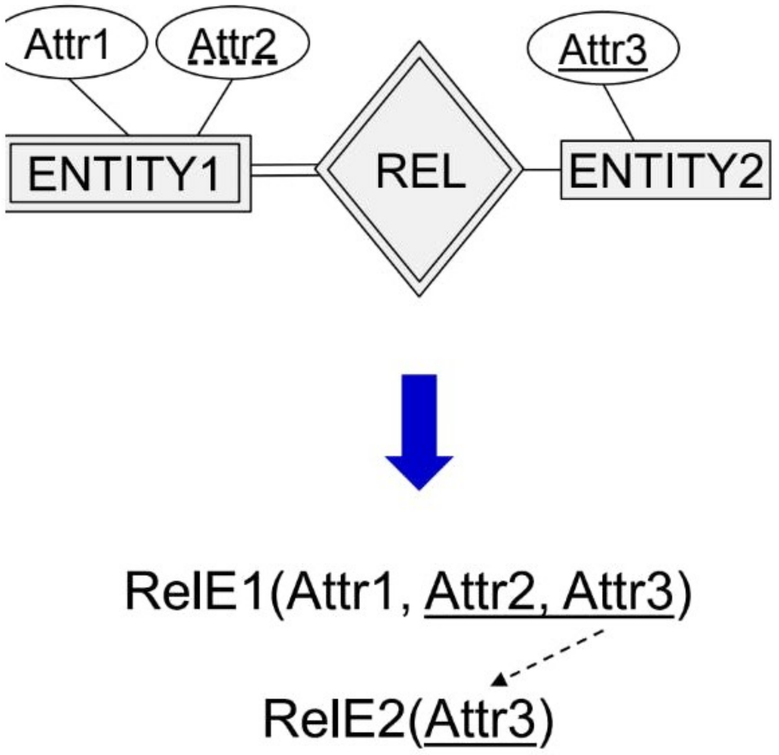
Repetisjon: ER til relasjonsmodell

7 steg for å være sikker på alt

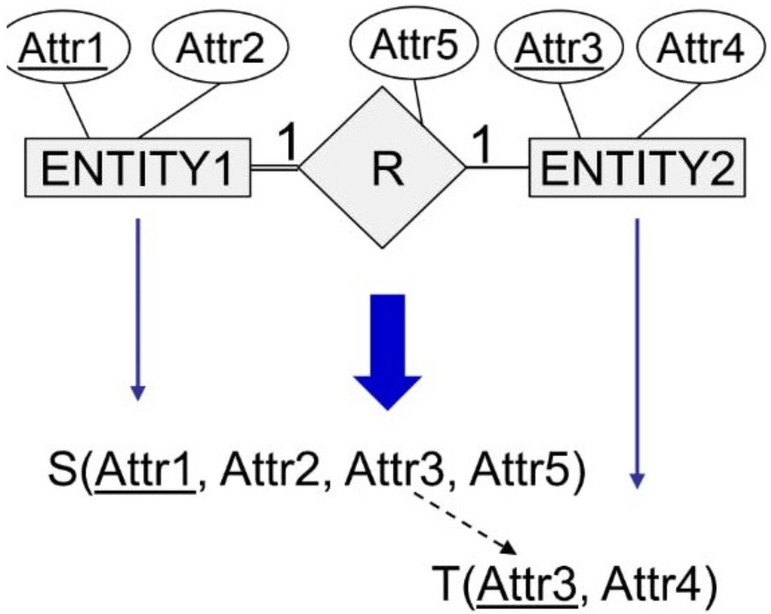
1. Kartlegg de vanlige entitetstypene
2. Kartlegg de svake entitetstypene
3. Kartlegg de binære 1-til-1 forholdene
4. Kartlegg de binære 1-til-N forholdene
5. Kartlegg de binære N-til-N forholdene
6. Kartlegg de attributtene med flere verdier
7. Kartlegg de ternære forholdene



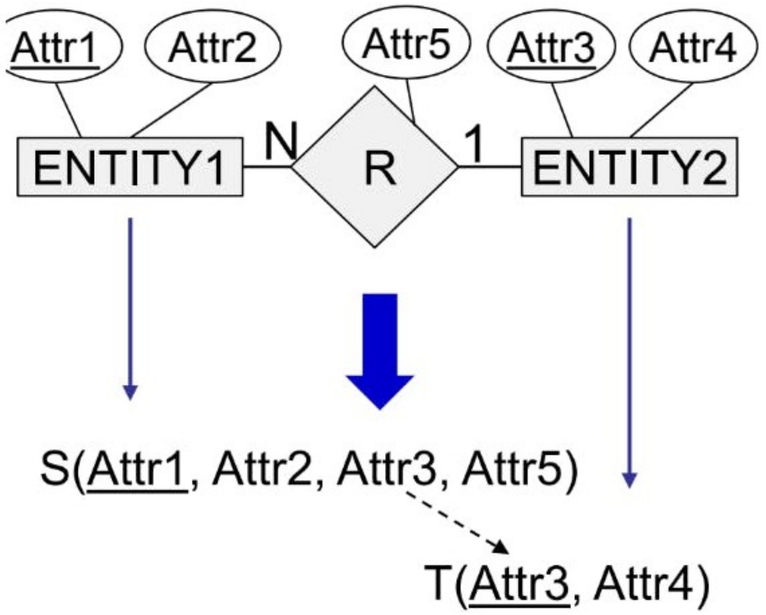
1. Kartlegg de vanlige entitetstypene



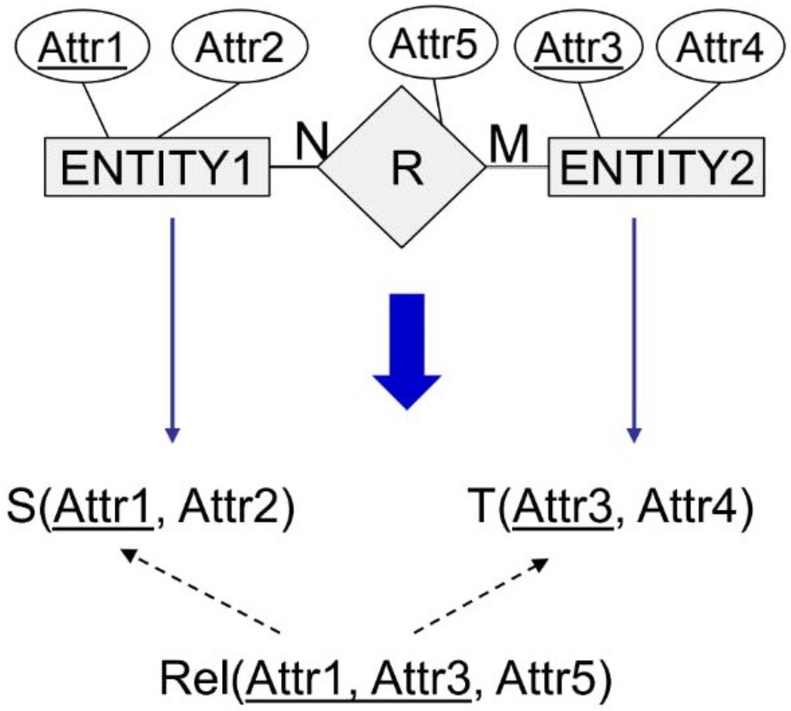
2. Kartlegg de svake entitetstypene



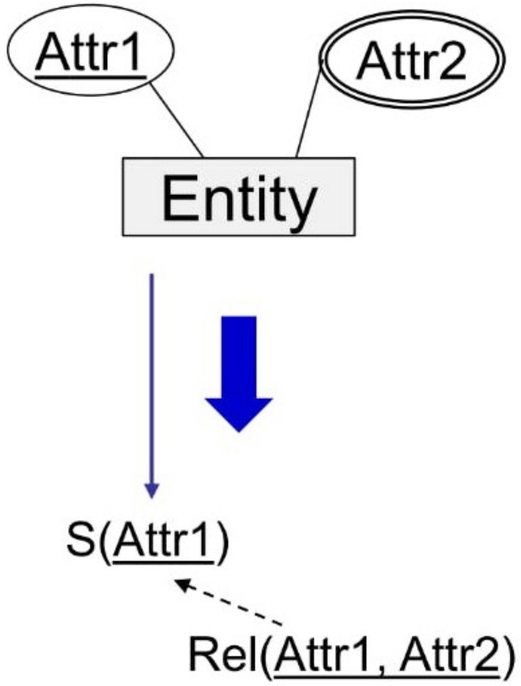
3. Kartlegg de binære 1-til-1 forholdene



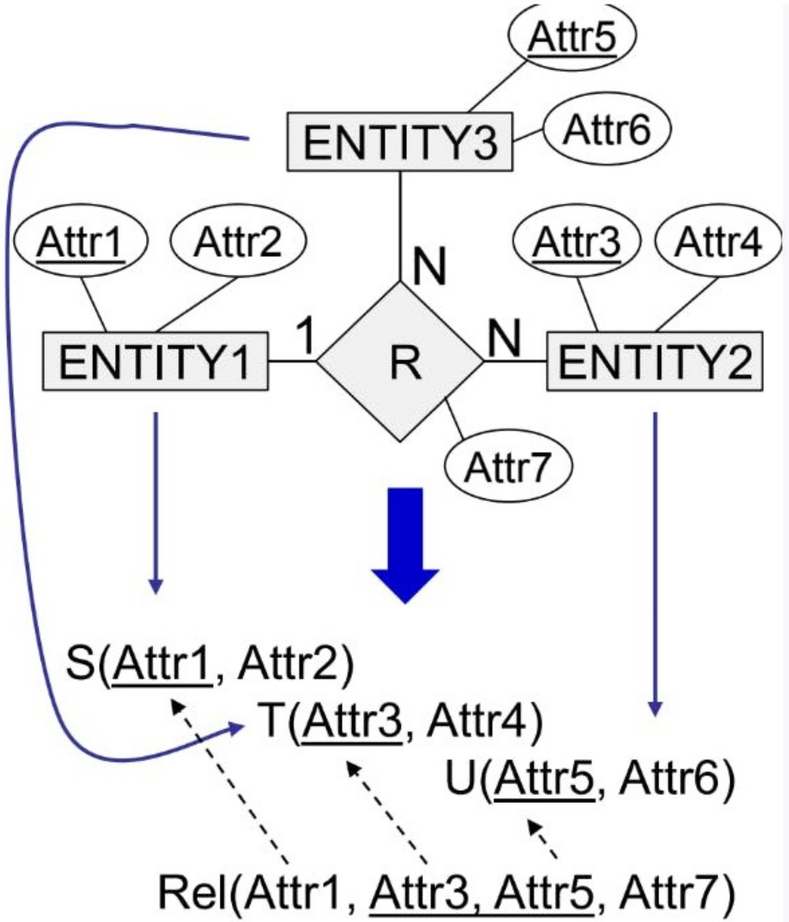
4. Kartlegg de binære 1-til-N forholdene



5. Kartlegg de binære N-til-N forholdene



6. Kartlegg de attributtene med flere verdier



7. Kartlegg de ternære forholdene

Repetisjon: Dekomponering

Algoritme:

Finn alle kandidatnøkler.
For hver tabell og hver FD $X \rightarrow A$:

1. Er X en supernøkkel?
   - Ja: BCNF sålangt, gå til neste FD
   - Nei: brudd på BCNF. Gå til 2.
2. Er A et nøkkelattributt?
   - Ja: 3NF sålangt, gå til neste FD
   - Nei: brudd på 3NF. Gå til 3.
3. Er X del av en kandidatnøkkel?
   - Nei: 2NF sålangt, gå til neste FD
   - Ja: brudd på 2NF og skjema er på 1NF, stopp.

Kandidatnøkler

- 1. Vi bruker FDene til å finne kandidatnøkler og normalform, så for dekomponering
- Kandidatnøkler: Finn ut hvilke attributter som bare er på høyre, venstre og på begge sider
- De på venstre vil alltid være en del av kandidatnøklene
- Om bare de på venstre ikke gir en tillukning som inneholder alle attributter, utvid med de fra begge sidene
- Om et attributt ikke er nok, utvid videre, så lenge det ikke er del av en annen kandidatnøkkel

Algoritme:

Finn alle kandidatnøkler.
For hver tabell og hver FD $X \rightarrow A$:

1. Er *X* en supernøkkel?
   - Ja: BCNF sålangt, gå til neste FD
   - Nei: brudd på BCNF. Gå til 2.
2. Er *A* et nøkkelattributt?
   - Ja: 3NF sålangt, gå til neste FD
   - Nei: brudd på 3NF. Gå til 3.
3. Er *X* del av en kandidatnøkkel?
   - Nei: 2NF sålangt, gå til neste FD
   - Ja: brudd på 2NF og skjema er på 1NF, stopp.

Normalformer

- 1. Inneholder X en av kandidatnøklene vi fant?
- {BCF} er en delmengde i {ABCF}. Fordi BCF er en minimal supernøkkel, så er ABCF en supernøkkel
- 2. Er A et attributt i en av kandidatnøklene vi fant?
- Gitt FDen BCF->DE, så tolken den som BCF->D og BCF->E. Er E et element i BCF, er D et element i BCF?
- 3. Er X en delmengde av en av kandidatnøklene vi fant?
- Som {CF} i {BCF}

Tapsfri dekomponering av $R(X)$ med FDer F:

1. For hver FD $Y \rightarrow A \in F$, hvis FDen er et brudd på BCNF:
   - 1.1 beregn Y^+,
   - 1.2 og dekomponer R til $S_1(Y^+)$ og $S_2(Y, X/Y^+)$.
2. Fortsett rekursivt (over S_1 og S_2) til ingen brudd på BCNF

Dekomponering

- Tapsfri dekomponering er når vi har med en felles nøkkel hver gang vi deler opp tabellen, slik at den kan slås sammen igjen
- 1. Finn hvilke FDer som bryter med BCNF på R
- 1.1 Start på en FD (gjerne en med flest attributter i Y), og ta tillukningen
- 1.2 Den første tabellen S1 består av tillukningen til Y
- 1.2 Den andre tabellen S2 skal bestå av 1# Y (felles nøkkel) #2 alle attributter som ikke er med i S1
- Fordi R har blitt delt opp, så vil ikke alle FDer, samt FDen vi dekomponerte på lenger være gjeldende
- 2. Finn FDene som fortsatt er gjeldende og ikke på BCNF, velg en å starte med, og repeter 1.1 og 1.2 til ingen FDer er gjeldende (på BCNF)
- Husk at dette er rekursjon, som betyr at R blir til S1 og S2, og at disse også skal vurderes og dekomponeres

Eksempel fra oblig 3

R(A, B, C, D, E, F, G)
CDE → B
AF → B
B → A
BCF → DE
D → G

Høyre:
G

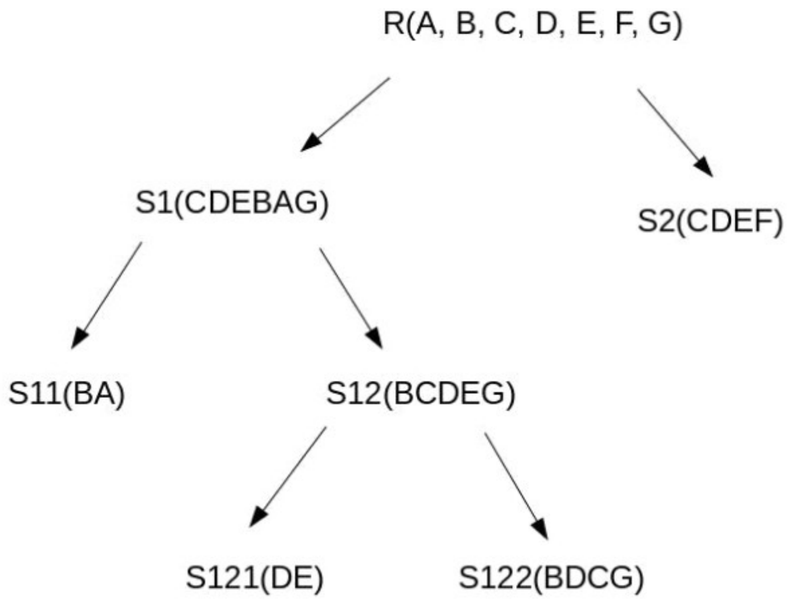
Venstre:
C, F

Begge:
A, B, D, E

CF+ = CF
CFA+ = CFABDEG
CFB+ = CFBADEG
CFD+ = CFDG
CFE+ = CFE
CFDE+ = CFDEBAG

KN = {ACF, BCF, DECF}

1. Er et element I KN et element I CDE?
2. Er B et attributt I et element I KN?
3. Er CDE en mengde I et element I KN?



FERDIG! DETTE VAR SISTE GRUPPETIME! TAKK FOR AT DERE KOM :D