Spørsmålskompilering del 2

- Estimere størrelsen på mellomresultater
- Vurdere fysiske spørreplaner

Estimere resultatstørrelser

Estimere størrelser

- Ideelt sett ønsker vi regler som er
  - nøyaktige – en liten feil kan resultere i valg av en lite hensiktsmessig algoritme i den fysiske spørreplanen
  - enkle å beregne – minimal ekstrakostnad for å gjøre selve valget
  - logisk konsistente – ikke avhengig av konkret algoritme for operatoren
- Men ingen universell algoritme finnes for dette
- Heldigvis hjelper også omtrentlige estimater til å velge en god fysisk spørreplan

Notasjon

- B_R eller $B(R)$ angir antall blokker som er nødvendig for å lagre R tettpakket
- T_R eller $T(R)$ angir antall tupler i R
- $V_R(A)$ eller $V(R,A)$ angir antall ulike verdier for attributt A i R
  - Gjennomsnittlig antall tupler med like A-verdier er da $T_R/V_R(A)$
- S_R eller $S(R)$ er størrelsen av et tuppel i R i bytes
- $sizeof_R$ eller $sizeof(R)$ er størrelsen av R i bytes: $sizeof_R = T_R * S_R$

Størrelsen av en projeksjon

- Størrelsen av en projeksjon $\pi_L(R)$ kan beregnes nøyaktig:
  - ett resultattuppel for hvert argumenttuppel
    $T(\pi_{A, B, \ldots}(R)) = T_R$
  - endrer bare størrelsen av hvert tuppel:
    $sizeof(\pi_{A, B, \ldots}(R)) = T_R * (S_{R.A} + S_{R.B} + \ldots)$

Størrelsen av en projeksjon, eksempel

Relasjon R

A	B	C	D
1	cat	1999	a
2	cat	2002	b
3	dog	2002	c
4	cat	1998	a
5	dog	2000	c

A: 4 byte integer
B: 20 byte text string
C: 4 byte date (year)
D: 30 byte text string

$T_R = 5$
$S_R = 58$

$V_R(A) = 5$
$V_R(B) = 2$
$V_R(C) = 4$
$V_R(D) = 3$

$sizeof(\pi_{A, B, \ldots}(R)) = T_R * (S_A + S_B + \ldots)$

$sizeof_R = \ldots$

$sizeof(\pi_A(R)) = \ldots$

$sizeof(\pi_{A, B, C, D, (A+10)\rightarrow E}(R)) = \ldots$

Størrelsen av en seleksjon

- En seleksjon $\sigma_C(R)$ reduserer antall tupler, men størrelsen av hvert tuppel er uendret
  - sizeof($\sigma_C(R)$) = T($\sigma_C(R)$) * S_R
- Estimering av antall tupler avhenger av
  - seleksjonsbetingelsen C
  - fordeling av verdier for de aktuelle attributtene: vi antar en uniform fordeling hvor vi bruker $V_R(A)$ for å estimere antall tupler i resultatet

Størrelsen av en seleksjon (forts.)

For attributt A og konstant c:

- Likhet, $\sigma_{A = c}(R)$:
  Bruk selektivitetsfaktoren **$1/V_R(A)$**
  - $T(\sigma_{A=c}(R)) = T_R / V_R(A)$
- Intervall, $\sigma_{A < c}(R)$:
  Bruk selektivitetsfaktoren **1/3**
  - $T(\sigma_{A<c}(R)) = T_R / 3$
- Ulikhet, $\sigma_{A \neq c}(R)$:
  Bruk selektivitetsfaktoren **$1 - 1/V_R(A)$**
  - $T(\sigma_{A \neq c}(R)) = T_R * (1 - 1/V_R(A))$

Størrelsen av en seleksjon, eksempel

Relasjon R

A	B	C	D
1	cat	1999	a
2	cat	2002	b
3	dog	2002	c
4	cat	1998	a
5	dog	2000	c

A: 4 byte integer
B: 20 byte text string
C: 4 byte date (year)
D: 30 byte text string

$T_R = 5$ $V_R(A) = 5$
$S_R = 58$ $V_R(B) = 2$
$V_R(C) = 4$
$V_R(D) = 3$

$T(\sigma_{A=c}(R)) = T_R / V_R(A)$
$T(\sigma_{A=3}(R)) = \ldots$
$T(\sigma_{B = \text{'cat'}}(R)) = \ldots$

$T(\sigma_{A<c}(R)) = T_R / 3$
$T(\sigma_{A > 2}(R)) = \ldots$

$T(\sigma_{A \neq c}(R)) = T_R * (1 - 1/V_R(A))$
$T(\sigma_{B \neq \text{'cat'}}(R)) = \ldots$

Størrelsen av en seleksjon med AND eller NOT

- Seleksjon med flere betingelser med **AND**, $\sigma_{A \textbf{ AND } B \textbf{ AND} \ldots}(R)$:
  - estimer størrelsen ved hjelp av én selektivitetsfaktor for hver betingelse:
    - 1/3 for <, >, ≤, ≥
    - $1-1/V_R(A)$ for ≠ på attributt A
    - $1/V_R(A)$ for = på attributt A
  - $T(\sigma_{A \textbf{ AND } B \textbf{ AND} \ldots}(R)) = T_R * faktor_A * faktor_B * \ldots$
- Seleksjon med **NOT**, $\sigma_{\textbf{NOT } A}(R)$:
  Bruk selektivitetsfaktoren $\mathbf{1-T(\sigma_A(R))/\ T_R}$
  - $T(\sigma_{\textbf{NOT } A}(R)) = T_R - T(\sigma_A(R))$

Størrelsen av en seleksjon med AND eller NOT, eksempel

Relasjon R

A	B	C	D
1	cat	1999	a
2	cat	2002	b
3	dog	2002	c
4	cat	1998	a
5	dog	2000	c

A: 4 byte integer
B: 20 byte text string
C: 4 byte date (year)
D: 30 byte text string

$T_R = 5$ $V_R(A) = 5$
$S_R = 58$ $V_R(B) = 2$
$V_R(C) = 4$
$V_R(D) = 3$

$T(\sigma_{A \textbf{ AND } B \textbf{ AND} \ldots}(R)) = T_R * faktor_A * faktor_B * \ldots$

$T(\sigma_{C = 1999 \textbf{ AND } A < 4}(R)) = \ldots$

$T(\sigma_{\textbf{NOT } A}(R)) = T_R - T(\sigma_A(R))$

$T(\sigma_{NOT\ A = 3}(R)) = \ldots$

Størrelsen av en seleksjon med OR

- Seleksjon med flere betingelser med **OR**, $\sigma_{A \textbf{ OR } B \textbf{ OR} \ldots}(R)$
  - Alternativ 1:
    $T(\sigma_{A \textbf{ OR } B \textbf{ OR} \ldots}(R)) = T(\sigma_A(R)) + T(\sigma_B(R)) + \ldots$
  - Alternativ 2:
    $T(\sigma_{A \textbf{ OR } B \textbf{ OR} \ldots}(R)) = \min(T_R, (T(\sigma_A(R)) + T(\sigma_B(R)) + \ldots))$
  - Alternativ 3:
    - anta at m_1 tupler tilfredsstiller den første betingelsen, m_2 tilfredsstiller den andre betingelsen, ...
    - $1 - m_i / T_R$ er da andelen tupler som ikke tilfredsstiller den i-te betingelsen
    - $T(\sigma_{A \textbf{ OR } B \textbf{ OR} \ldots}(R)) = T_R * [1 - (1 - m_1/ T_R) * (1 - m_2/ T_R) * \ldots]$

Størrelsen av et produkt

Størrelsen av et kartesisk produkt $R \times S$ kan beregnes nøyaktig:

- Ett tuppel for hver mulige kombinasjon av tuplene i relasjonene R og S:
  $T(R \times S) = T_R * T_S$
- Størrelsen av hvert nye tuppel er summen av størrelsen til hvert av de opprinnelige tuplene:
  $S(R \times S) = S_R + S_S$
- sizeof$(R \times S) = T(R \times S) * S(R \times S) = T_R * T_S * (S_R + S_S)$

Størrelsen av naturlig join

Størrelsen av naturlig join $R(X,A) \bowtie S(A,Y)$ avhenger av hvordan verdiene til joinattributtet A fordeler seg mellom relasjonene R(X,A) og S(A,Y):

- Disjunkte sett med A-verdier – tomt resultat:
  $T(R \bowtie S) = 0$
- A er en fremmednøkkel fra R til S – hvert tuppel i R matcher ett tuppel i S:
  $T(R \bowtie S) = T_R$
- Nesten alle tuplene i R og S har samme A-verdi – kombiner alle tuplene i hver relasjon:
  $T(R \bowtie S) = T_R * T_S$

Størrelsen av naturlig join (forts.)

Antakelser:

- **Inkludering av verdimengde:**
  Hvis $V_R(A) \leq V_S(A)$, så anta at
  $$\delta(\pi_A(R)) \subseteq \delta(\pi_A(S))$$
  Dvs. hver A-verdi i R har en match i S
- **Bevaring av verdimengde:**
  Anta at verdimengden til ikke-join-attributter er den samme før og etter join, dvs.
  $$V(R \bowtie S, B) = V_R(B)$$
  der B er et attributt i R, men ikke i S

Størrelsen av naturlig join (forts.)

Antall tupler i $R(X, A) \bowtie S(A, Y)$ kan nå estimeres som følger:

- Hvis $V_R(A) \leq V_S(A)$, vil hvert tuppel i R matche anslagsvis $T_S / V_S(A)$ tupler i S:
  $T(R \bowtie S) = T_R * T_S / V_S(A)$
- Tilsvarende, hvis $V_S(A) \leq V_R(A)$:
  $T(R \bowtie S) = T_R * T_S / V_R(A)$
- Generelt:
  $T(R \bowtie S) = T_R * T_S / \max(V_R(A), V_S(A))$

Størrelsen av naturlig join (forts.)

Eksempel:

A(a, b)	B(b, c)	C(c, d)
$T_A = 10.000$	$T_B = 2.000$	$T_C = 5.000$
$V_A(a) = 5.000$	$V_B(b) = 100$	$V_C(c) = 100$
$V_A(b) = 1.000$	$V_B(c) = 1.000$	$V_C(d) = 100$

$T(A \bowtie B) = \ldots$

$T(A \bowtie B \bowtie C) = \ldots$

Størrelsen av naturlig join (forts.)

- Hvis det er *mer enn ett joinattributt,*
  $R(X, A_1, A_2, \ldots) \bowtie S(A_1, A_2, \ldots, Y)$, får vi:

$$T(R \bowtie S) = \frac{T_R * T_S}{\max(V_R(A_1), V_S(A_1)) * \max(V_R(A_2), V_S(A_2)) * \ldots}$$

  for hvert A_i -attributt som er felles for R og S

- For naturlig join mellom flere relasjoner $R_1 \bowtie R_2 \bowtie R_3 \bowtie \ldots \bowtie R_n$
  - start med maksimalt antall tupler $T(R_1) * T(R_2) * T(R_3) * \ldots * T(R_n)$
  - for hvert attributt A som forekommer i mer enn en relasjon, divider med alle unntatt den minste $V(R_i,A)$
- Størrelsen av equijoin beregnes som naturlig join
- Størrelsen av thetajoin $R \bowtie_c S$ beregnes ved å beregne størrelsen av $\sigma_c(R \times S)$

Størrelsen av en union

- Avhenger av om vi bruker **set**- eller **bag**-versjonen:
  - Bag:
    Resultatet er nøyaktig lik summen av tupler i argumentene:
    $T(R \cup_b S) = T_R + T_S$
  - Set:
    - Som for bag hvis relasjonene er disjunkte
    - Antall tupler i den største relasjonen hvis den minste er en delmengde av denne
    - Vanligvis et sted mellom disse, kan for eksempel bruke gjennomsnittet:
      $T(R \cup_s S) = T_R + T_S/2$
      hvor S er den minste relasjonen

Størrelsen av snitt og differanse

- Antall tupler i et snitt $R \cap S$ er
  - 0 hvis relasjonene er disjunkte
  - $\min(T_R, T_S)$ hvis den ene relasjonen er en delmengde av den andre
  - vanligvis et sted i mellom, kan for eksempel bruke gjennomsnittet: $\min(T_R, T_S) / 2$
- Antall tupler i en differanse $R - S$ er
  - T_R hvis relasjonene er disjunkte
  - $T_R - T_S$ hvis alle tuplene i S finnes i R
  - vanligvis et sted i mellom, kan for eksempel bruke gjennomsnittet: $T_R - T_S/2$

Størrelsen av en duplikateleminasjon

- Antall distinkte tupler som resultat av en duplikateliminasjon $\delta(R)$ er
  - 1 hvis alle tuplene er like
  - T_R hvis alle tuplene er forskjellige
- En tilnærming:
  - Gitt $V_R(A_i)$ for alle *n* attributter, vil maksimalt antall ulike tupler være $V_R(A_1) * V_R(A_2) * \ldots * V_R(A_n)$
  - La estimert antall tupler være det minste av dette tallet og $T_R/2$, dvs. $\min(V_R(A_1) * V_R(A_2) * \ldots * V_R(A_n), T_R/2)$
- Tilsvarende for gruppering
  (ser bare på grupperingsattributtene A_1, A_2, ... A_m)

Vurdere fysiske planer

Sammenligning av logiske spørreplaner

- Vi sammenligner ulike spørreplaner for en gitt spørring ved hjelp av størrelsen på temporære relasjoner
  - estimer resultatet av hver operator i spørreplanen
  - legg kostnadene inn i treet
  - kostnaden til planen er lik summen av alle kostnadene i treet, bortsett fra for:
    - roten – sluttresultatet
    - løvnodene – data lagret på disk

Sammenligning av logiske spørreplaner - eksempel

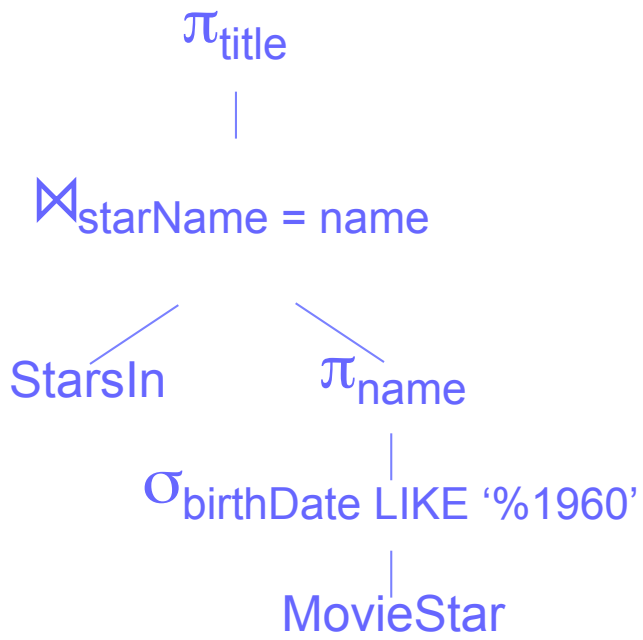
```
StarsIn(title, year, starName)
MovieStar(name, address, gender, birthDate)

SELECT title
FROM StarsIn
WHERE starName IN (
        SELECT name
        FROM MovieStar
        WHERE birthDate LIKE '%1960');
```

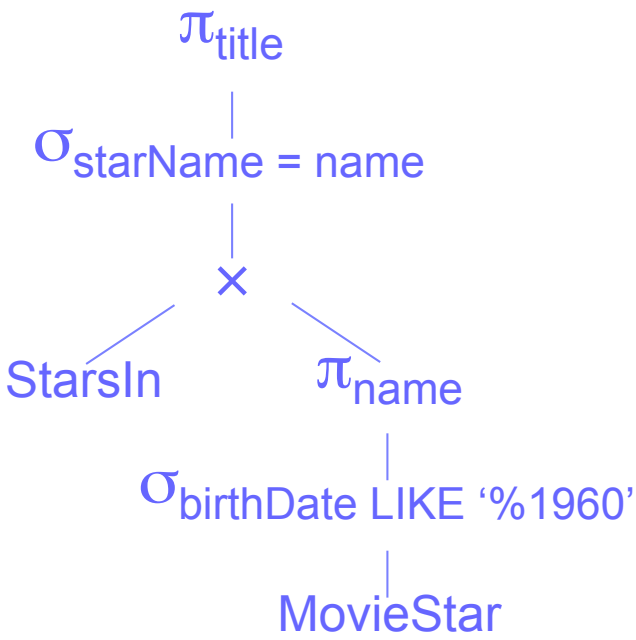
Statistikk:
T(StarsIn) = 10.000

V(StarsIn, starName) = 500
S(StarsIn) = 60

T(MovieStar) = 1.000

V(MovieStar, name) = 1.000
V(MovieStar, birthDate) = 50
S(MovieStar) = 100





Eksempel (forts.)

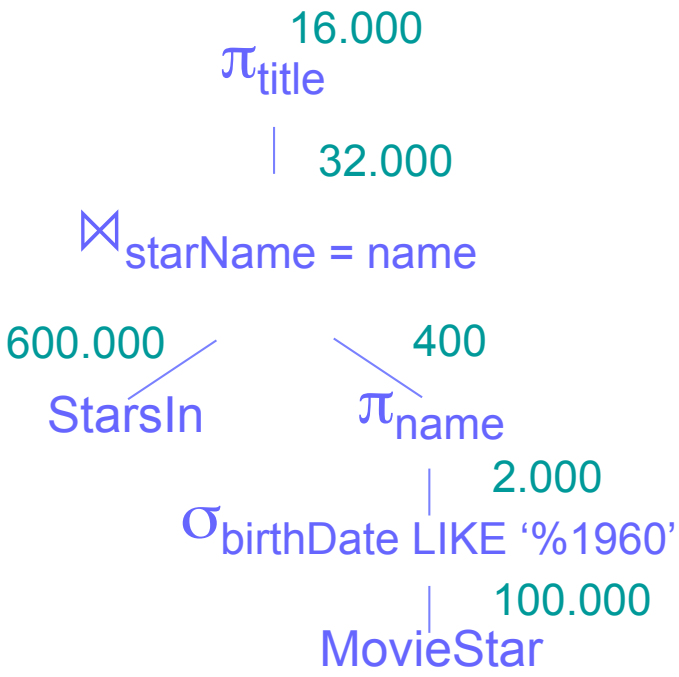
- $A_1 = \sigma_{\text{birthDate LIKE '\%1960'}}(MS)$:
  - $T(A_1) = T(\sigma(MS)) = T(MS) / V(MS, \text{birthDate}) = 1000 / 50 = 20$
  - $\text{sizeof}(A_1) = 20 * 100 = 2000$
- $A_2 = \pi_{\text{name}}(A_1)$:
  - $T(A_2) = T(\pi(A_1)) = T(A_1) = 20$
  - anta at name er 20 byte
  - $\text{sizeof}(A_2) = 20 * 20 = 400$
- $A_3 = SI \bowtie A_2$:
  - $T(A_3) = T(SI \bowtie A_2) =$
    $T(SI)*T(A_2) / \max[V(SI, \text{starName}), V(A_2, \text{name})] =$
    $10000 * 20 / \max(500, 20) = 400$
  - $S(A_2) = 20$
  - $\text{sizeof}(A_3) = 400 * (60 + 20) = 32000$
- $A_4 = \pi_{\text{title}}(A_3)$:
  - $T(A_4) = T(\pi(A_3)) = T(A_3) = 400$
  - anta at title er 40 byte
  - $\text{sizeof}(A_4) = 400 * 40 = 16000$

Statistikk:
T(SI) = 10.000

V(SI, starName) = 500
S(SI) = 60

T(MS) = 1.000

V(MS, name) = 1.000
V(MS, birthDate) = 50
S(MS) = 100



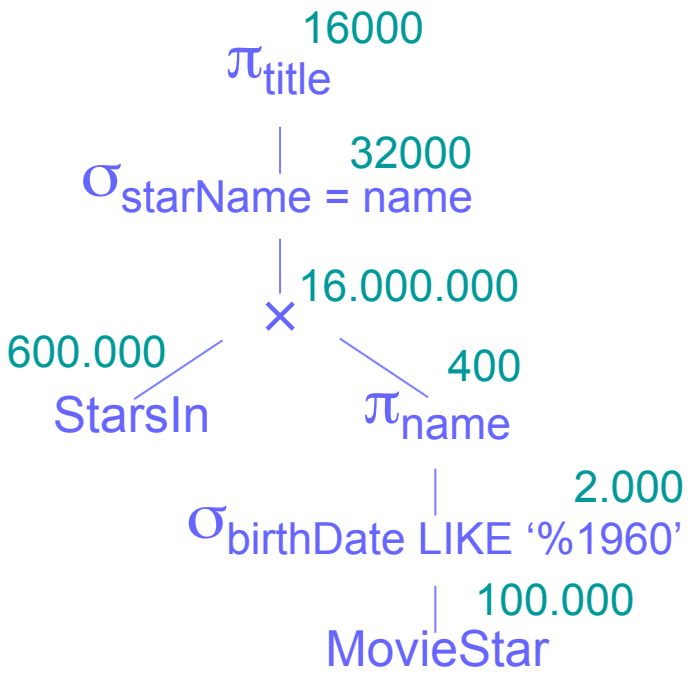
Eksempel (forts.)

Statistikk:
T(SI) = 10.000
V(SI, starName) = 500
S(SI) = 60

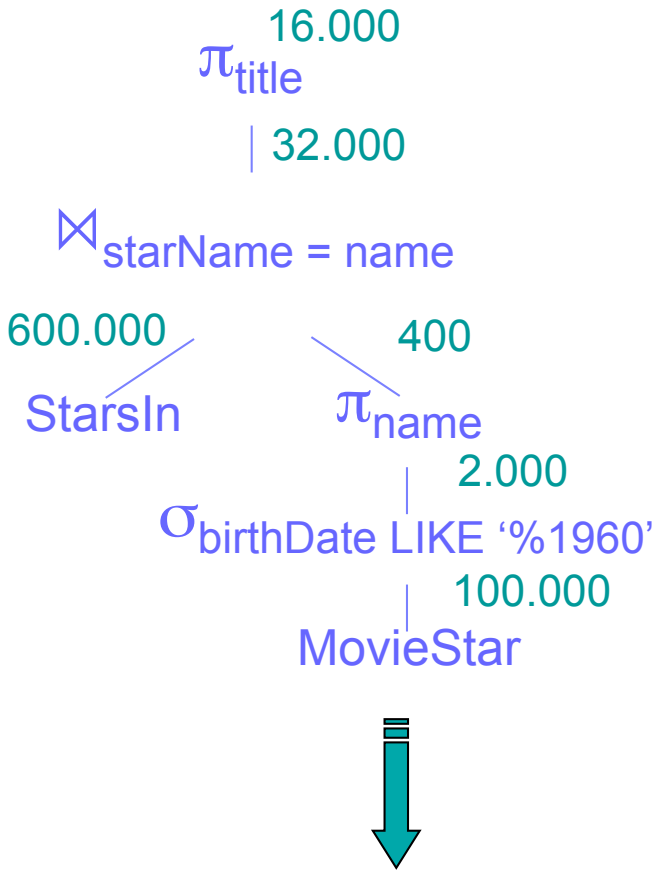
T(MS) = 1.000

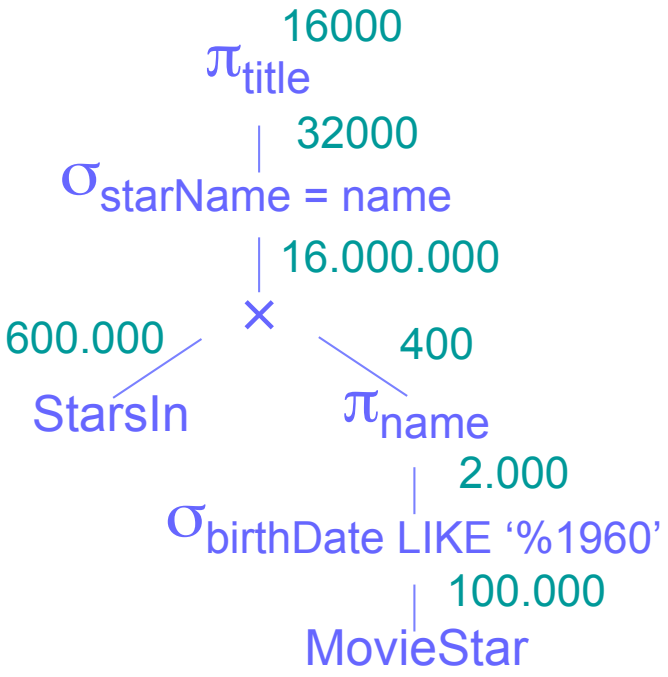
V(MS, name) = 1.000
V(MS, birthDate) = 50
S(MS) = 100

- $A_1 = \sigma_{\text{birthDate LIKE '\%1960'}}(MS)$ → som før: 2000, $T(\sigma(MS))=20$
- $A_2 = \pi_{\text{name}}(A_1)$ → som før: 400, $T(A_2) = T(A_1) = 20$
- $B_3 = SI \times A_2$:
  - $T(B_3) = T(SI \times A_2) =$
    $T(SI) * T(A_2) = 10000 * 20 = 200.000$
  - $S(A_2) = 20$
  - $\text{sizeof}(B_3) = 200.000 * (60 + 20) = 16.000.000$
- $B_4 = \sigma_{\text{starName = name}}(B_3)$:
  (Regelen for $T(\sigma_{A=B}(R))$ er ikke gjennomgått)
  - $T(B_4) = T(\sigma(B_3)) =$
    $T(B_3) / \max(V(B_3, \text{name}), V(SI, \text{starName}))$
    $= 200.000 / \max(20, 500) = 400$
  - $S(B_4) = S(SI) + S(B_3) = 60 + 20 = 80$
  - $\text{sizeof}(B_4) = 400 * 80 = 32000$
- $B_5 = \pi_{\text{title}}(B_4)$ → som A_4: 400 * 40 = 16000



Eksempel (forts.)





Totalt:
2000 + 400 + 32000 = 34400

Totalt:
2000 + 400 + 16000000 + 32000= 16034400

Valg av fysisk spørreplan

- Mulige alternativer for å velge “billigste” fysiske spørreplan:
  - exhaustive
  - heuristic
  - branch-and-bound
  - hill climbing
  - dynamic programming
  - Sellinger-style optimizations

Valg av fysisk spørreplan (forts.)

- Exhaustive:
  - se på alle mulige kombinasjoner av valg i planen
  - estimer kostnaden til hver plan
  - mange planer, kostbart/ikke gjennomførbart

- Heuristic (brute force):
  - velg en plan i henhold til heuristiske regler, dvs. basert på tidligere erfaringer som f.eks.:
    - bruk indeks på operasjoner som $\sigma_{A = 10}(R)$
    - bruk den minste relasjonen først ved join av mange relasjoner
    - hvis argumentene er sortert, bruk en sorteringsbasert operator
    - ...
  - rask, men bare basert på generelle regler

Valg av fysisk spørreplan (forts.)

- Branch-and-bound:
  - finn en plan ved hjelp av heuristiske regler
  - se på mindre deler av planen for å finne optimaliseringer

- Sellinger-style optimization:
  - for alle subuttrykk, ta vare på kostnaden og forventet type resultat
  - en operator kan ha høyere individuell kostnad, men hvis resultatet f.eks. er sortert, kan senere operatorer bruke dette
    - ingen effekt ved vurdering av størrelsen på mellomresultatet
    - ved vurdering av antall disk-IO kan første del av en sorteringsbasert operasjon spares

Valg av seleksjonsalgoritme

- Eksempel:
  - R(x,y,z), T_R = 5000, B_R = 200, $V_R(x)$ = 100, $V_R(y)$ = 500
  - indekser på alle attributtene, R er sortert på z, R ligger samlet
  - $\sigma_{x=1 \text{ AND } y=2 \text{ AND } z<5}(R)$
  - table-scan – les blokk for blokk:
    - kostnad: B_R = 200 disk-IO siden R ligger samlet
  - index-scan på x – finn tupler med x=1 ved bruk av indeksen, sjekk så y og z:
    - i verste fall ligger alle tuplene i ulike blokker
    - kostnad: $T_R / V_R(x)$ = 5000 / 100 = 50 disk-IO
  - index-scan på y – finn tupler med y=2 ved bruk av indeksen, sjekk så x og z:
    - i verste fall ligger alle tuplene i ulike blokker
    - kostnad: $T_R / V_R(y)$ = 5000 / 500 = 10 disk-IO
  - index-scan på z – finn tupler med z<5 ved bruk av indeksen, sjekk så x og y:
    - vi har estimert slike seleksjoner til 1/3 av tuplene, R er sortert på z og ligger samlet
    - kostnad: $B_R / 3$ = 67 disk-IO

Valg av joinalgoritme

- Hvis vi ikke vet hvor mye ressurser som er tilgjengelig:
  - velg en-pass og håp at det er tilstrekkelig med minneplass
  - velg sort join hvis...
    - ... begge argumentene allerede er sorterte
    - ... det joines tre eller flere relasjoner på samme attributt
  - velg index join hvis den ene relasjonen er liten og det finnes en passende indeks på den andre
  - velg ellers hash join siden det krever minst minne

Pipelining versus materialisering

Det siste viktige spørsmålet er hvordan overføre mellomresultater mellom operatorer:

- Pipelining:
  Send resultatet direkte videre til den nye operatoren, dvs. data forblir i minnet og operasjoner kan arbeide samtidig
  - kan være mer effektivt
  - krever mer minne – risikerer flere diskaksesser
- Materialisering:
  Lagre alle mellomresultater på disk inntil de trengs av en annen operator
  - mer disk-IO
  - kan tillate enklere algoritmer (f.eks. en-pass snarere enn to-pass) siden en operator kan bruke mer minne

Pipelining versus materialisering (forts.)

- *Unære operasjoner*, seleksjon og projeksjon, bør pipelines siden operasjonene utføres på ett tuppel om gangen.

- *Binære operasjoner* kan pipelines, men
  - antall buffere som trengs for beregningene varierer
  - resultatstørrelsene varierer
  - ⇨ valg mellom pipelining og materialisering avhenger av minneplassen