Lecture 24:
Query Execution

Monday, November 27, 2006

Outline

- Query optimization: algebraic laws 16.2

Example

Product(pname, maker), **Company**(cname, city)

```
Select Product.pname
From  Product, Company
Where Product.maker=Company.cname
   and  Company.city = "Seattle"
```

- How do we execute this query ?

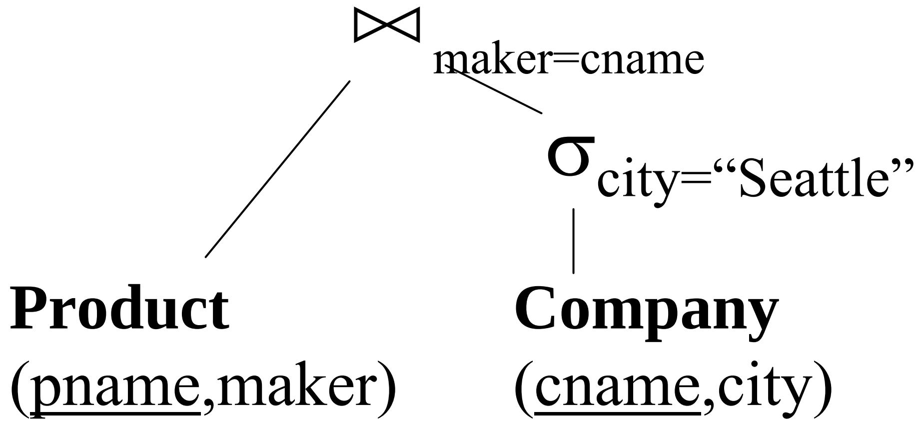
Example

Product(pname, maker), **Company**(cname, city)

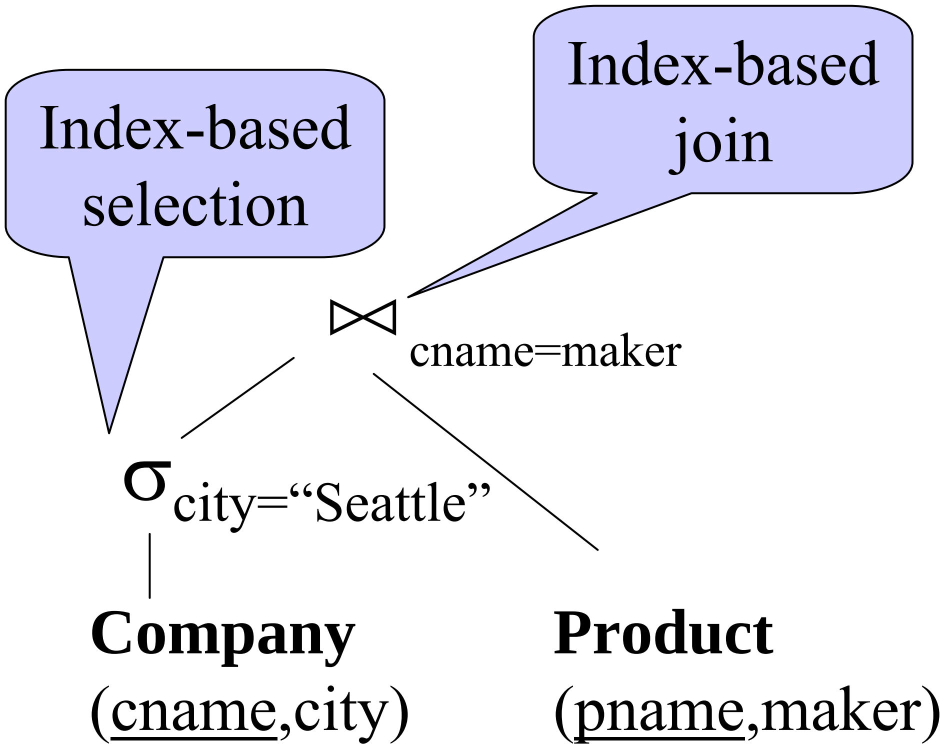
Assume:

Clustered index: **Product**.pname, **Company**.cname
Unclustered index: **Product**.maker, **Company**.city

Logical Plan:

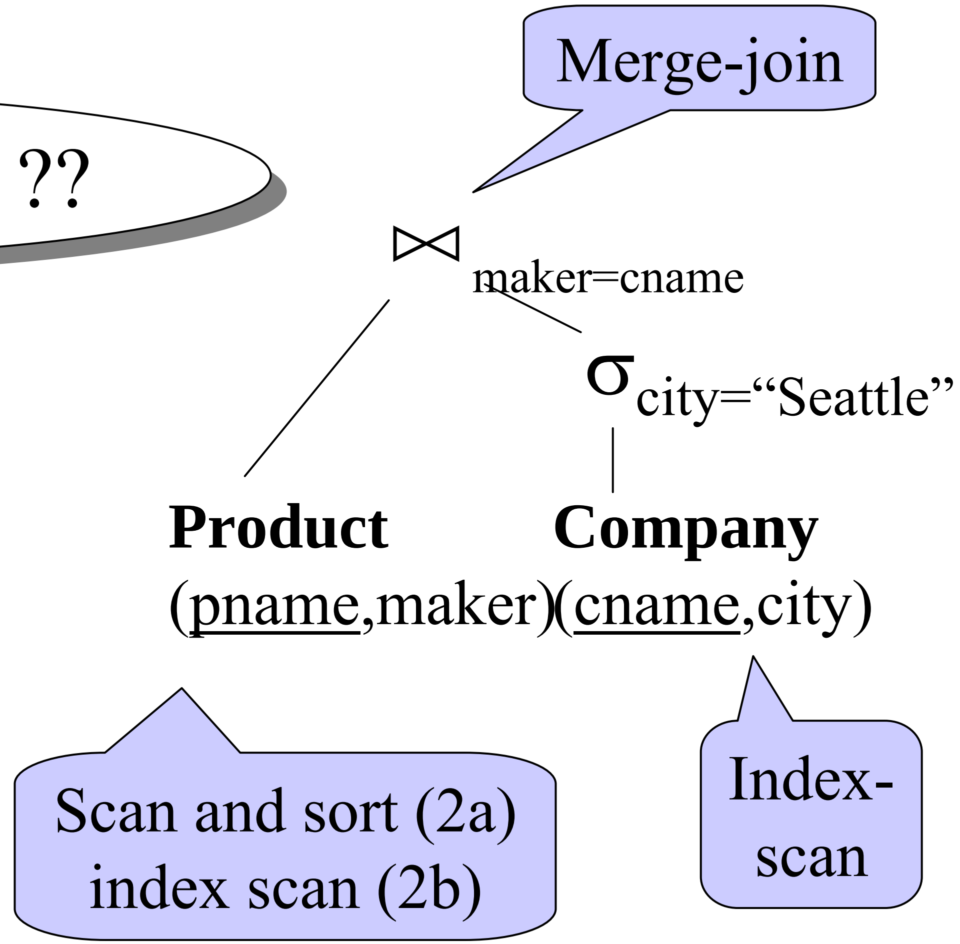


Physical plan 1:



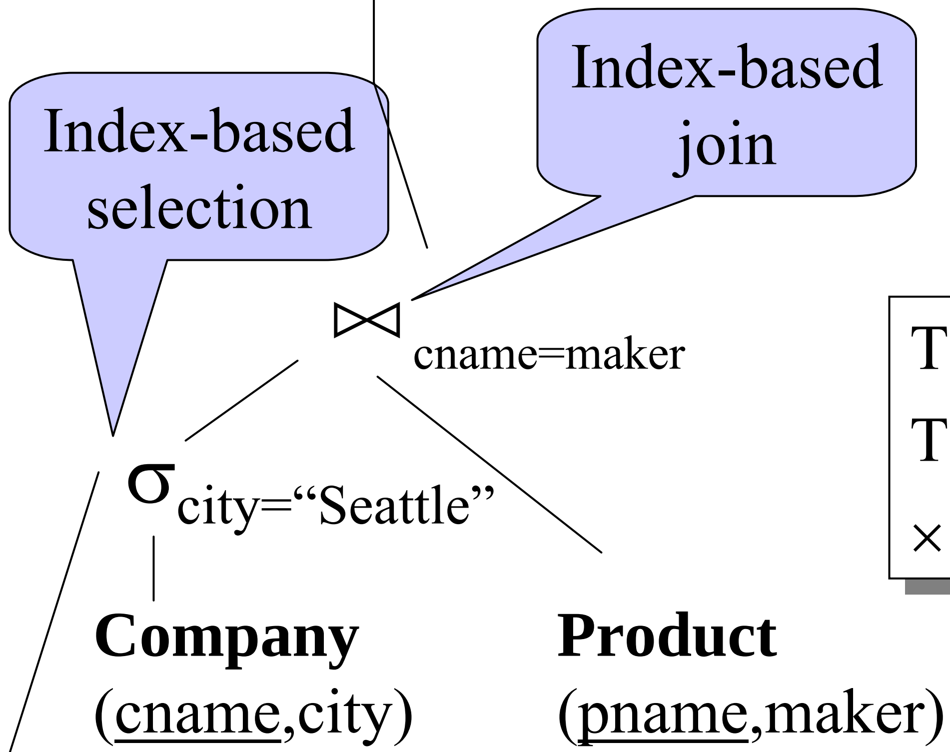
Physical plans 2a and 2b:

Which one is better ??



Physical plan 1:

$\times$ T(**Product**) / V(**Product**, maker)



Total cost:

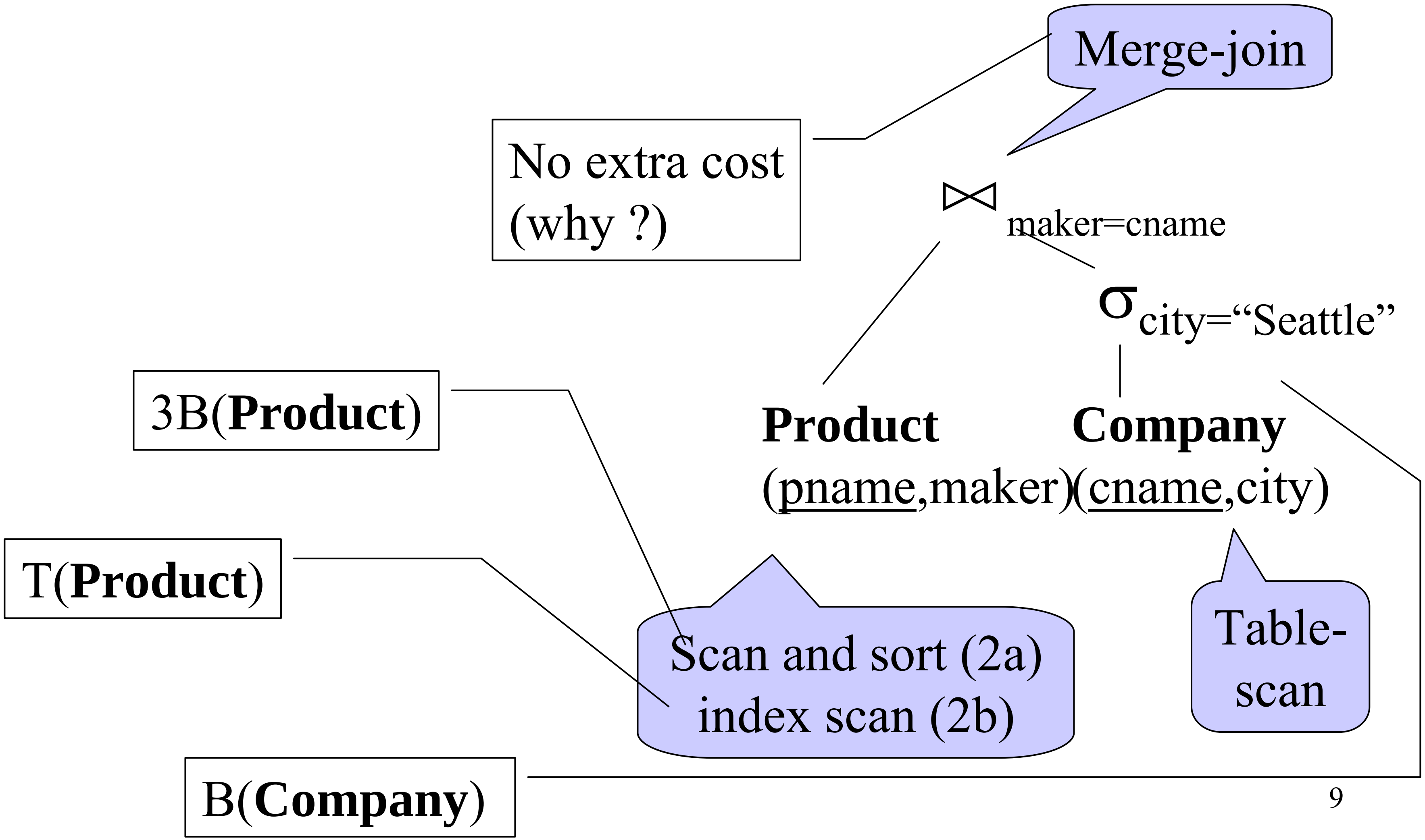
T(**Company**) / V(**Company**, city)

$\times$ T(**Product**) / V(**Product**, maker)

T(**Company**) / V(**Company**, city)

Total cost:
(2a): 3B(**Product**) + B(**Company**)
(2b): T(**Product**) + B(**Company**)

Physical plans 2a and 2b:



Plan 1: $T(\textbf{Company})/V(\textbf{Company},\text{city}) \times T(\textbf{Product})/V(\textbf{Product},\text{maker})$

Plan 2a: $B(\textbf{Company}) + 3B(\textbf{Product})$

Plan 2b: $B(\textbf{Company}) + T(\textbf{Product})$

Which one is better ??

It depends on the data !!

Example

T(**Company**) = 5,000 B(**Company**) = 500 M = 100
T(**Product**) = 100,000 B(**Product**) = 1,000

We may assume V(**Product**, maker) ≈ T(**Company**) (why ?)

- Case 1: V(**Company**, city) ≈ T(**Company**)

V(**Company**,city) = 2,000

- Case 2: V(**Company**, city) << T(**Company**)

V(**Company**,city) = 20

Which Plan is Best ?

Plan 1: T(**Company**)/V(**Company**,city) × T(**Product**)/V(**Product**,maker)
Plan 2a: B(**Company**) + 3B(**Product**)
Plan 2b: B(**Company**) + T(**Product**)

Case 1:

Case 2:

Lessons

- Need to consider several physical plan
  - even for one, simple logical plan
- No magic "best" plan: depends on the data
- In order to make the right choice
  - need to have ***statistics*** over the data
  - the B's, the T's, the V's

Query Optimzation

- Have a SQL query Q
- Create a plan P

HOW ??

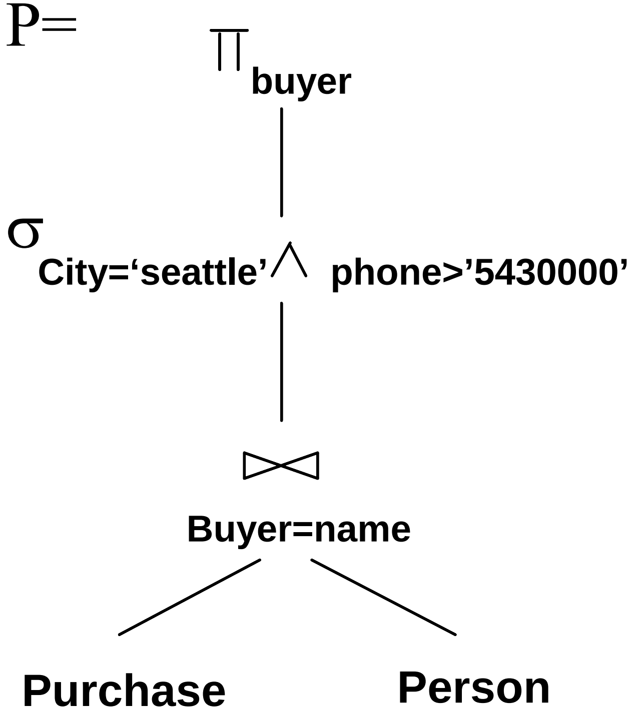
- Find equivalent plans P = P’ = P’’ = …
- Choose the “cheapest”.

Logical Query Plan

```
SELECT P.buyer
FROM Purchase P, Person Q
WHERE P.buyer=Q.name AND
      P.city='seattle' AND
      Q.phone > '5430000'
```

Purchasse(buyer, city)
Person(name, phone)

P=



In class:
find a "better" plan P'

Logical Query Plan

Q=

```
SELECT city, sum(quantity)
FROM sales
GROUP BY city
HAVING sum(quantity) < 100
```

P=

T2(city,p)

$\sigma_{p < 100}$

T1(city,p)

$\gamma_{city,\ sum(quantity) \rightarrow p}$

sales(product, city, quantity)

In class:
find a “better” plan P’

The three components of an optimizer

We need three things in an optimizer:

- Algebraic laws
- An optimization algorithm
- A cost estimator

Algebraic Laws

- Commutative and Associative Laws

  $R \cup S = S \cup R$, $R \cup (S \cup T) = (R \cup S) \cup T$

  $R \mid\times\mid S = S \mid\times\mid R$, $R \mid\times\mid (S \mid\times\mid T) = (R \mid\times\mid S) \mid\times\mid T$

  $R \mid\times\mid S = S \mid\times\mid R$, $R \mid\times\mid (S \mid\times\mid T) = (R \mid\times\mid S) \mid\times\mid T$

- Distributive Laws

  $R \mid\times\mid (S \cup T) = (R \mid\times\mid S) \cup (R \mid\times\mid T)$

Algebraic Laws

- Laws involving selection:

  $\sigma_{C \text{ AND } C'}(R) = \sigma_C(\sigma_{C'}(R)) = \sigma_C(R) \cap \sigma_{C'}(R)$

  $\sigma_{C \text{ OR } C'}(R) = \sigma_C(R) \cup \sigma_{C'}(R)$

  $\sigma_C (R \mid\times\mid S) = \sigma_C (R) \mid\times\mid S$

- When C involves only attributes of R

  $\sigma_C (R - S) = \sigma_C (R) - S$

  $\sigma_C (R \cup S) = \sigma_C (R) \cup \sigma_C (S)$

  $\sigma_C (R \mid\times\mid S) = \sigma_C (R) \mid\times\mid S$

Algebraic Laws

- Example: R(A, B, C, D), S(E, F, G)
  - $\sigma_{F=3}(R \mid\times\mid_{D=E} S) =$?
  - $\sigma_{A=5 \text{ AND } G=9}(R \mid\times\mid_{D=E} S) =$?

Algebraic Laws

- Laws involving projections

  $\Pi_M(R \mid\times\mid S) = \Pi_M(\Pi_P(R) \mid\times\mid \Pi_Q(S))$

  $\Pi_M(\Pi_N(R)) = \Pi_{M,N}(R)$

- Example R(A,B,C,D), S(E, F, G)

  $\Pi_{A,B,G}(R \mid\times\mid_{D=E} S) = \Pi_{?}(\Pi_?(R) \mid\times\mid_{D=E} \Pi_?(S))$

Algebraic Laws

- Laws involving grouping and aggregation:

  $\delta(\gamma_{A,\ agg(B)}(R)) = \gamma_{A,\ agg(B)}(R)$

  $\gamma_{A,\ agg(B)}(\delta(R)) = \gamma_{A,\ agg(B)}(R)$ if agg is "duplicate insensitive"

- Which of the following are "duplicate insensitive" ?
  sum, count, avg, min, max

$\gamma_{A,\ agg(D)}(R(A,B) \mid\times\mid_{B=C} S(C,D)) =$
$\quad \gamma_{A,\ agg(D)}(R(A,B) \mid\times\mid_{B=C} (\gamma_{C,\ agg(D)} S(C,D)))$

Optimizations Based on Semijoins

THIS IS ADVANCED STUFF; NOT ON THE FINAL

- $R \ltimes S = \Pi_{A1,\ldots,An} (R \bowtie S)$
- Where the schemas are:
  - Input: R(A1,…An), S(B1,…,Bm)
  - Output: T(A1,…,An)

Optimizations Based on Semijoins

Semijoins: a bit of theory (see [AHV])

- Given a query:

$$Q :- \Pi (\sigma (R_1 \ |x|\ R_2 \ |x|\ \ldots \ |x|\ R_n))$$

- A full reducer for Q is a program:

$$\begin{aligned} R_{i1} &:= R_{i1} \ltimes R_{j1} \\ R_{i2} &:= R_{i2} \ltimes R_{j2} \\ &\ldots\ldots \\ R_{ip} &:= R_{ip} \ltimes R_{jp} \end{aligned}$$

- Such that no dangling tuples remain in any relation

Optimizations Based on Semijoins

- Example:

  Q(A,E) :- R1(A,B) |x| R2(B,C) |x| R3(C,D,E)

- A full reducer is:

  R2(B,C) := R2(B,C) |x R1(A,B)

  R3(C,D,E) := R3(C,D,E) |x R2(B,C)

  R2(B,C) := R2(B,C) |x R3(C,D,E)

  R1(A,B) := R1(A,B) |x R2(B,C)

The new tables have only the tuples necessary to compute Q(E)

Optimizations Based on Semijoins

- Example:

$$\boxed{Q(E) :- R1(A,B) \;|x|\; R2(B,C) \;|x|\; R3(A,C, E)}$$

- Doesn’t have a full reducer (we can reduce forever)

Theorem a query has a full reducer iff it is “acyclic”

Optimizations Based on Semijoins

- Semijoins in [Chaudhuri'98]

```
CREATE VIEW DepAvgSal As (
        SELECT E.did, Avg(E.Sal) AS avgsal
        FROM Emp E
        GROUP BY E.did)

SELECT E.eid, E.sal
FROM Emp E, Dept D, DepAvgSal V
WHERE E.did = D.did AND E.did = V.did
        AND E.age < 30 AND D.budget > 100k
        AND E.sal > V.avgsal
```

Optimizations Based on Semijoins

- First idea:

```
CREATE VIEW LimitedAvgSal As (
        SELECT E.did, Avg(E.Sal) AS avgsal
        FROM Emp E, Dept D
        WHERE E.did = D.did AND D.buget > 100k
        GROUP BY E.did)

SELECT E.eid, E.sal
FROM Emp E, Dept D, LimitedAvgSal V
WHERE E.did = D.did AND E.did = V.did
        AND E.age < 30 AND D.budget > 100k
        AND E.sal > V.avgsal
```

Optimizations Based on Semijoins

- Better: full reducer

```
CREATE VIEW PartialResult AS
        (SELECT E.id, E.sal, E.did
        FROM Emp E, Dept D
        WHERE E.did=D.did AND E.age < 30
        AND D.budget > 100k)

CREATE VIEW Filter AS
        (SELECT DISTINCT P.did FROM PartialResult P)

CREATE VIEW LimitedAvgSal AS
        (SELECT E.did, Avg(E.Sal) AS avgsal
        FROM Emp E, Filter F
        WHERE E.did = F.did GROUP BY E.did)
```

Optimizations Based on Semijoins

```
SELECT P.eid, P.sal
FROM PartialResult P, LimitedDepAvgSal V
WHERE P.did = V.did AND P.sal > V.avgsal
```