



PA152: Efficient Use of DB

3. Query Processing

Vlastislav Dohnal

Query Processing

■ Procedure:

- Query in SQL
- Checking syntax and semantics
 - Parse Tree – in relational algebra
- Logical query plan
 - Plan modifications, estimate relation sizes
- Physical query plan
 - Concrete algorithms for individual operations
 - Estimate costs
- Evaluation
 - Choose the “best” physical query plan

Example

■ Relation

- R(A,B,C)

- S(C,D,E)

■ Query

- select B,D

- from R,S

- where R.C=S.C and R.A='c' and S.E=2

Example

R	A	B	C
	a	1	10
	b	1	20
	c	2	10
	d	2	35
	e	3	45

S	C	D	E
	10	x	2
	20	y	2
	30	z	2
	40	x	1
	50	y	3

select B,D from R,S where R.C=S.C and R.A='c' and S.E=2

Example

R	A	B	C	S	C	D	E
a	1	10	10	10	x	2	
b	1	20	20	20	y	2	
c	2	10	30	30	z	2	
d	2	35	40	40	x	1	
e	3	45	50	50	y	3	

Result:

B	D
2	x

How to evaluate this query?

1. way

- Cartesian product
- Selecting records
- Projection

$R \times S$

R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2
a	1	10	20	y	2
.					
.					
c	2	10	10	x	2
.					
.					

$R \times S$	R.A	R.B	R.C	S.C	S.D	S.E
	a	1	10	10	x	2
	a	1	10	20	y	2
	.					
	.					
This record matches →	c	2	10	10	x	2
	.					
	.					

Output – query result

select B,D from R,S where R.C=S.C and R.A='c' and S.E=2

Describing Evaluation Plans

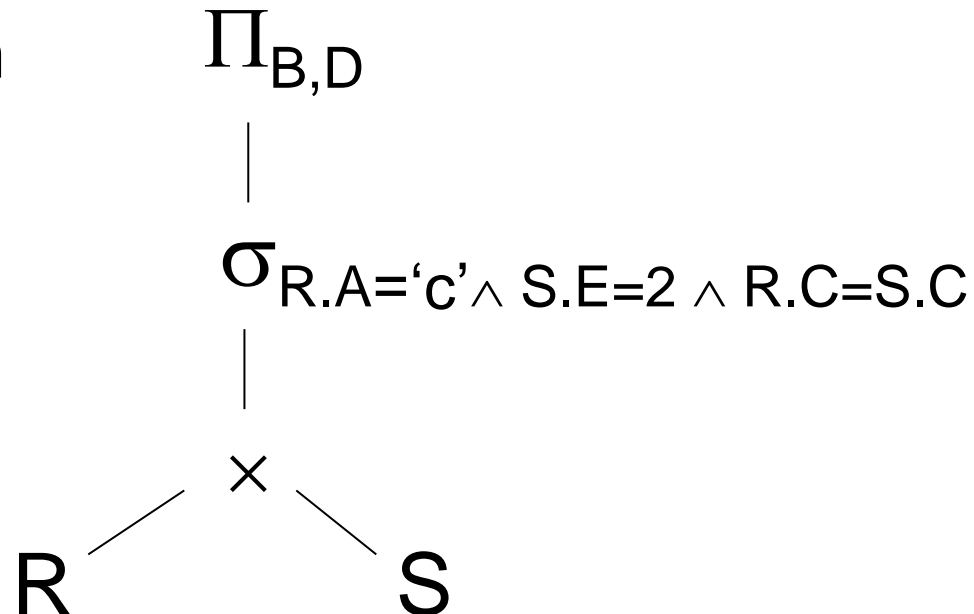
■ Using relational algebra

□ select B,D from R,S where R.C=S.C and R.A='c' and S.E=2

□ $\Pi_{B,D} [\sigma_{R.A='c' \wedge S.E=2 \wedge R.C=S.C} (R \times S)]$

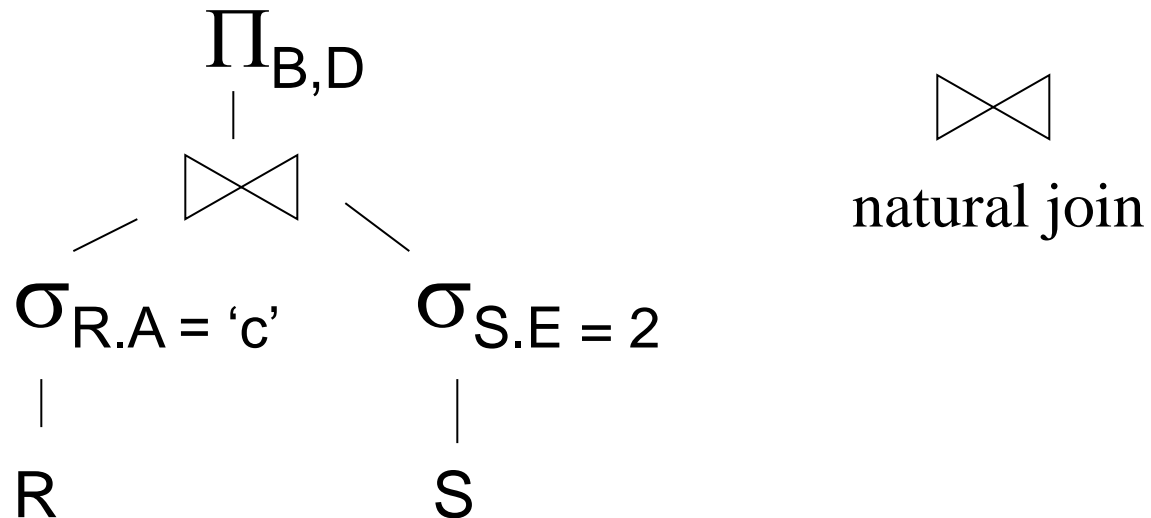
■ Example of Plan 1:

□ Query plan



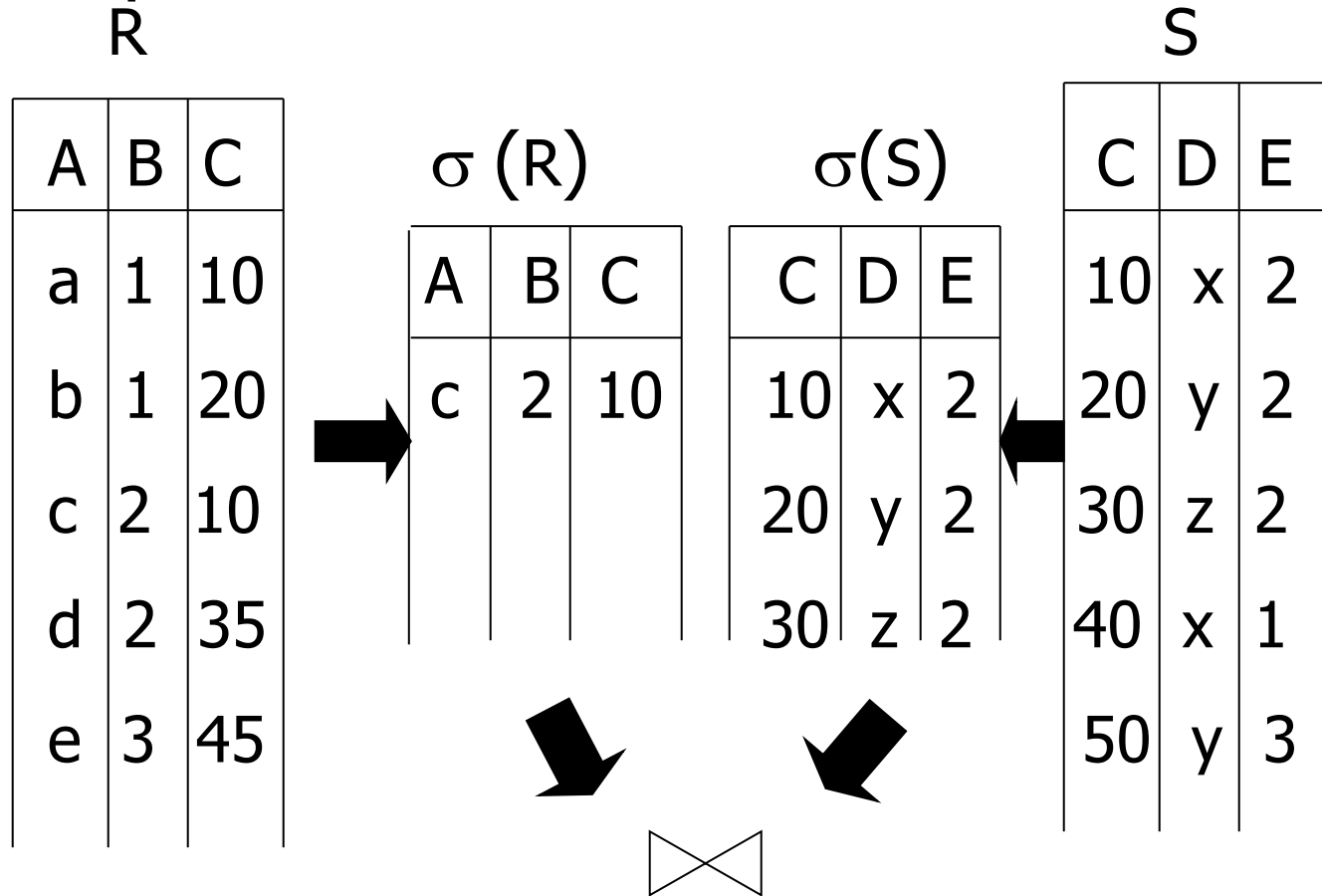
Describing Evaluation Plans

- Example of Plan 2:



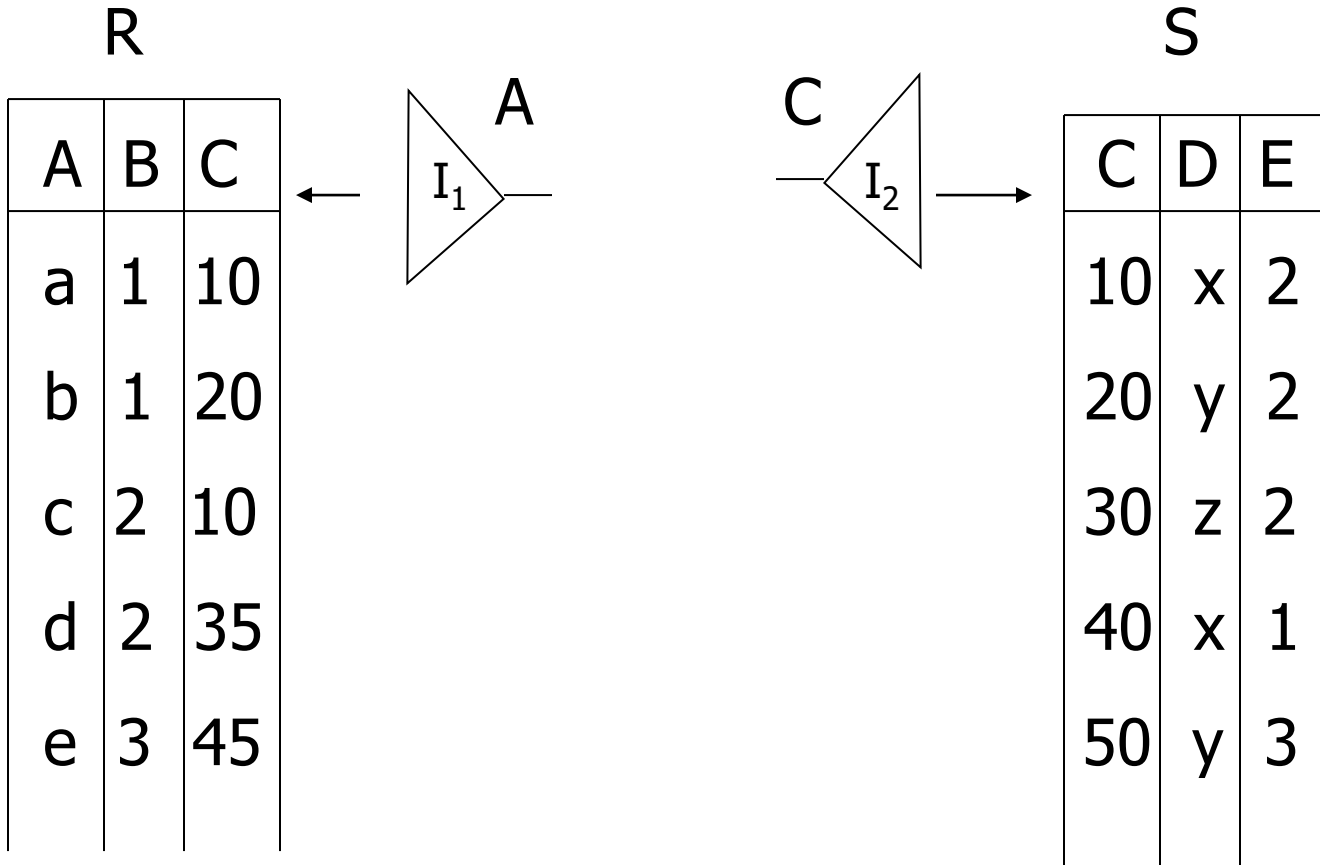
Describing Evaluation Plans

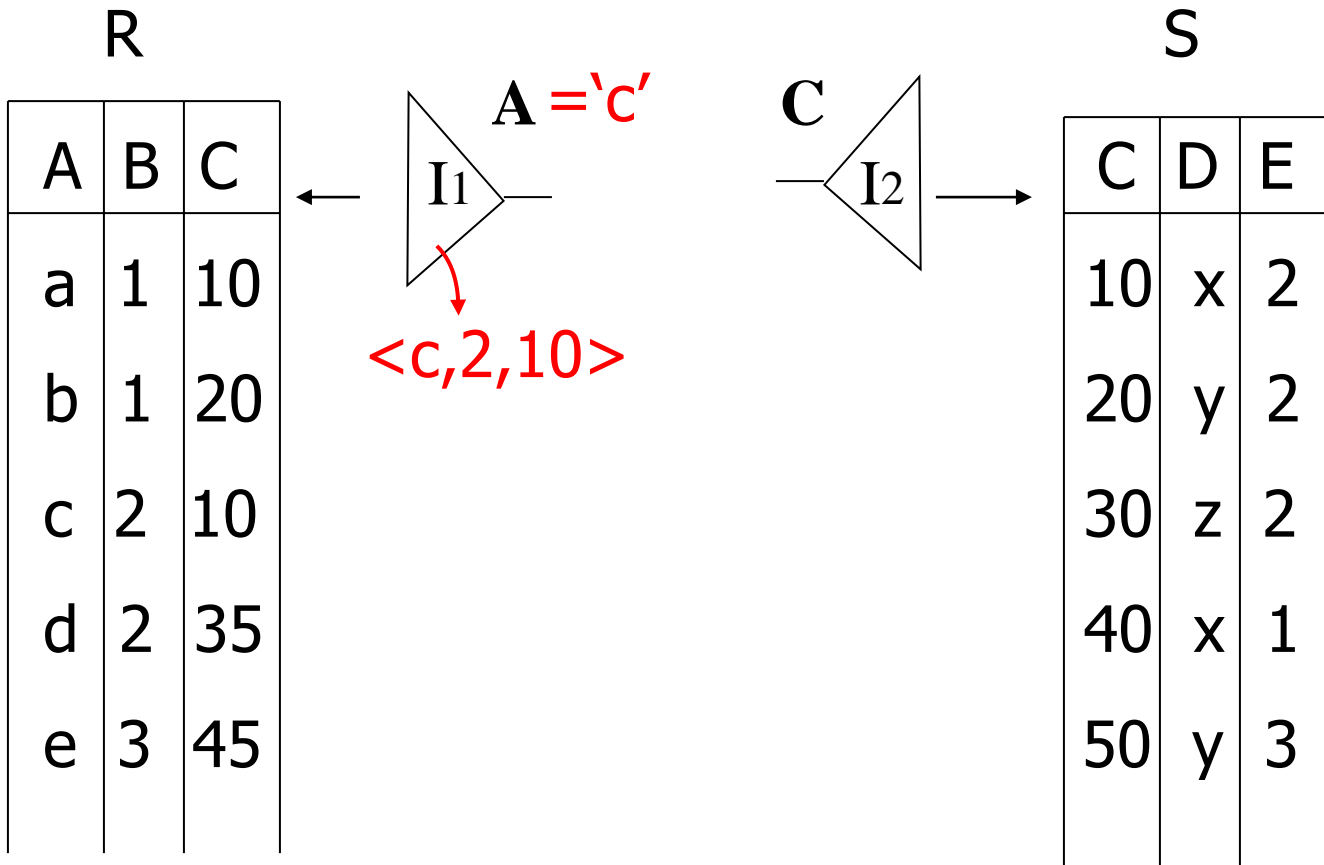
- Example of Plan 2:

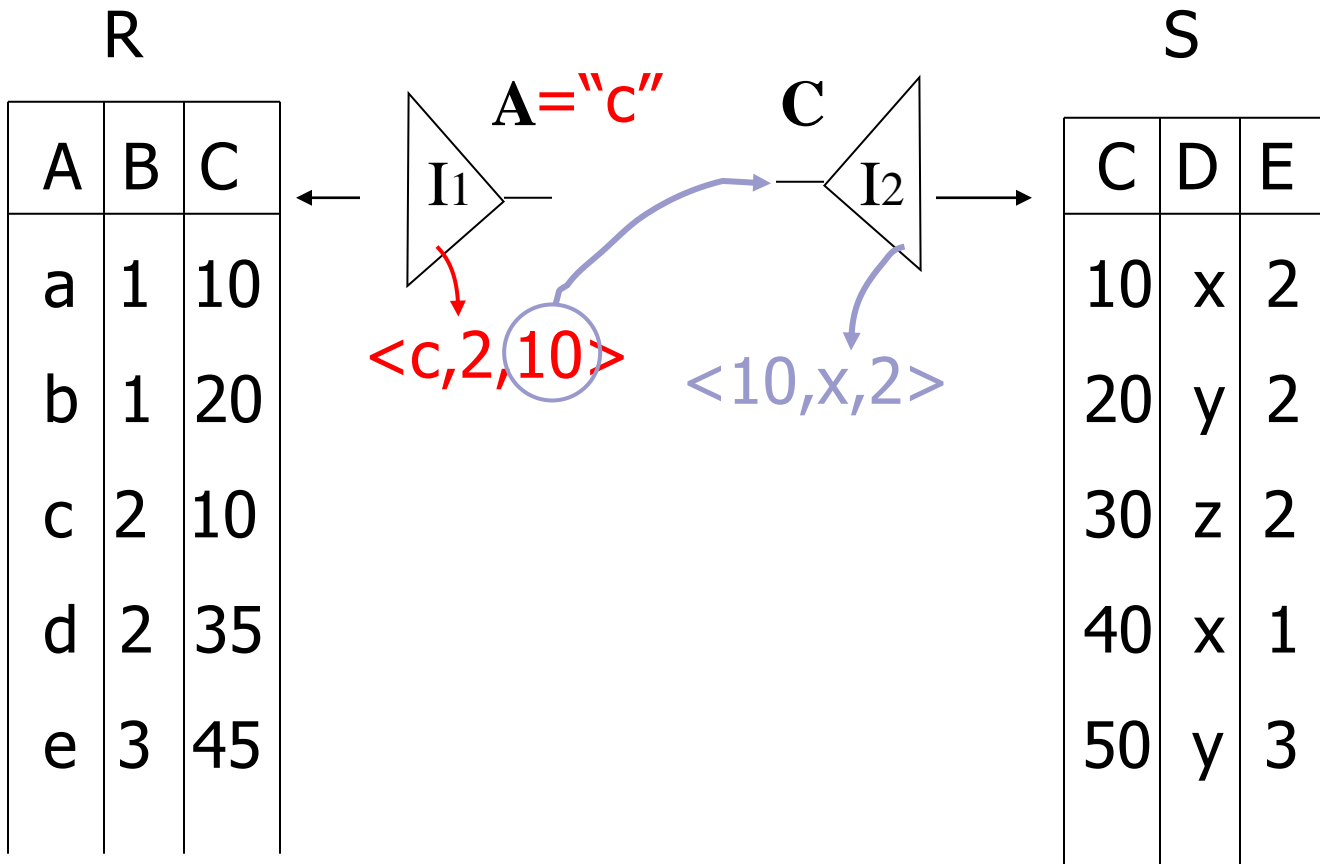


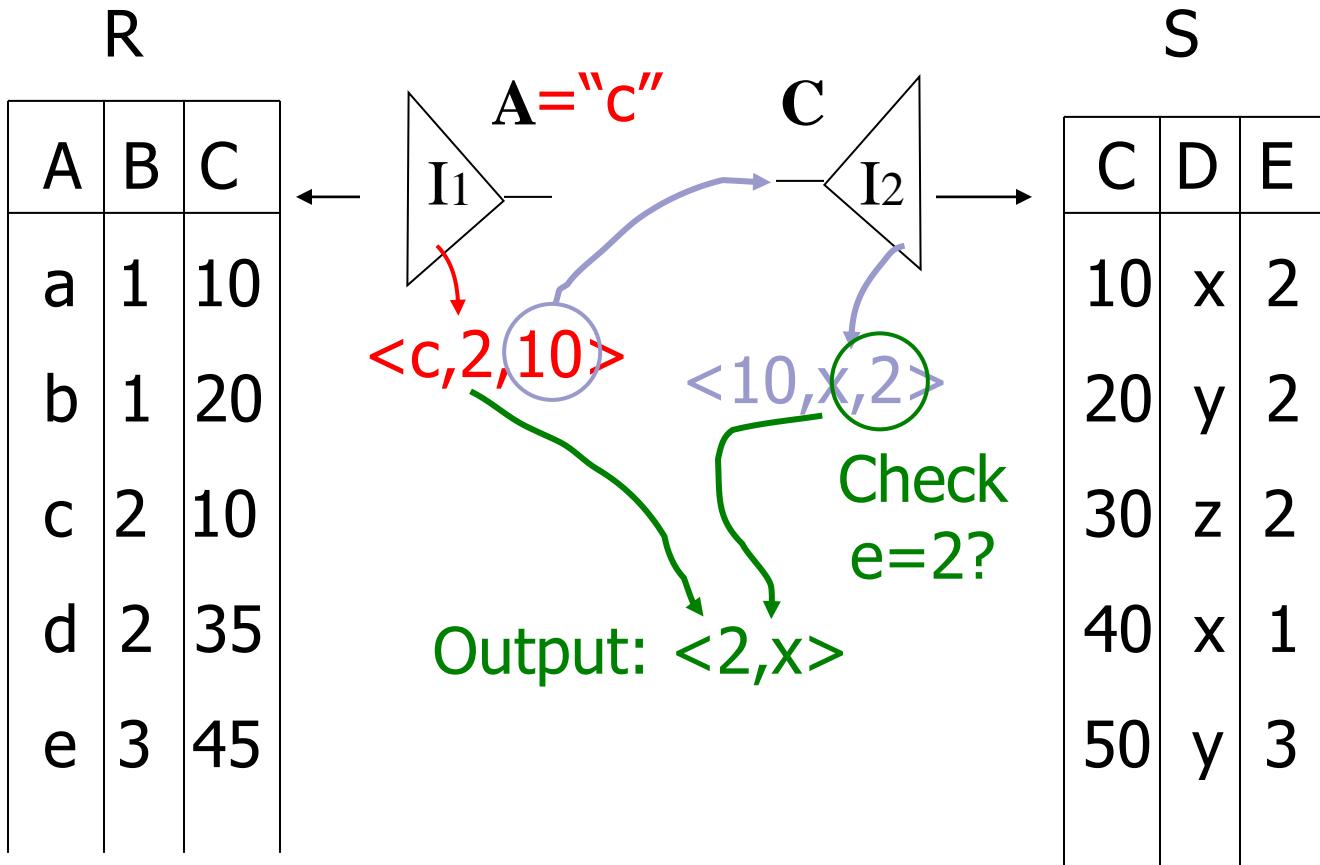
Describing Evaluation Plans

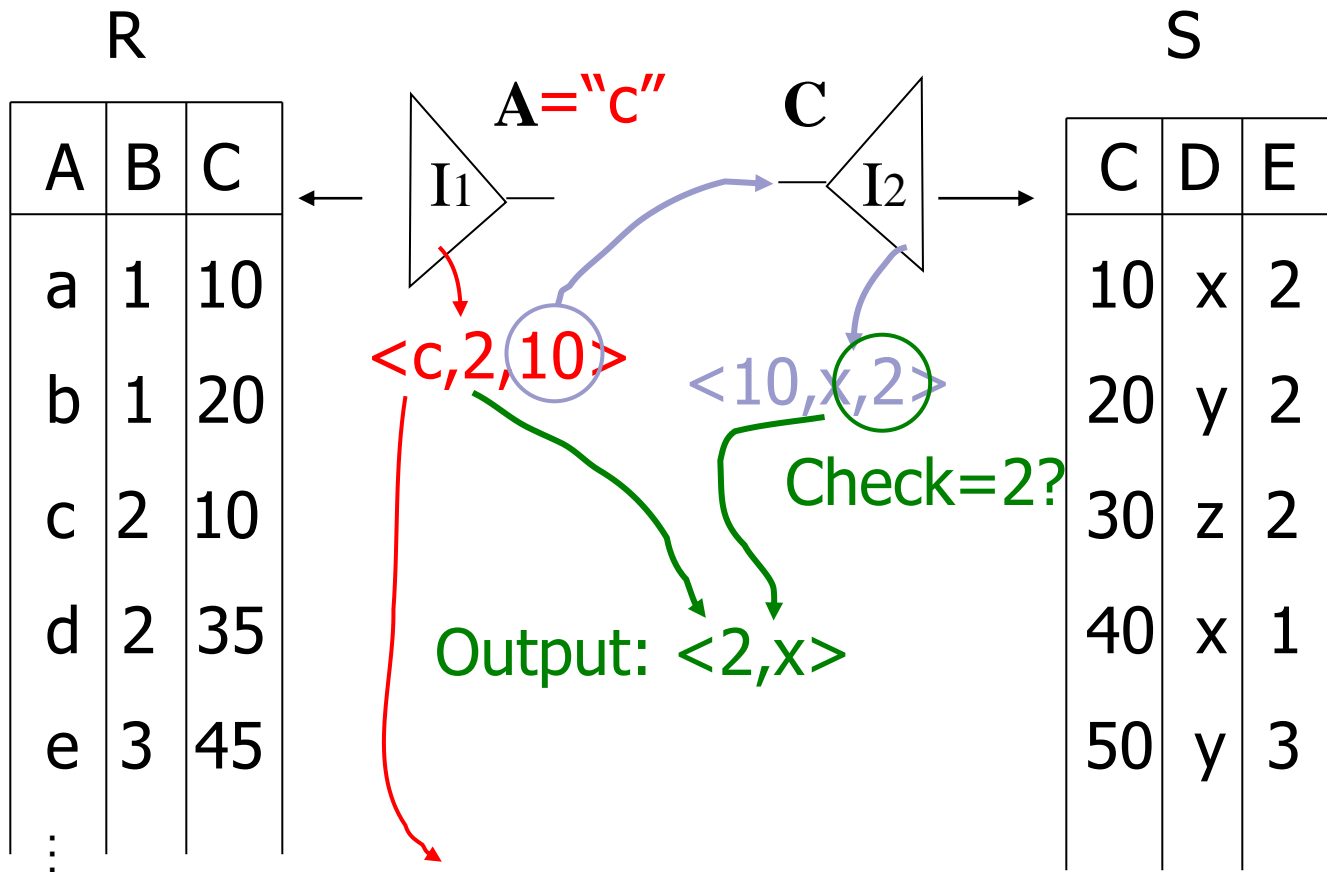
- Plan 3: (physical plan)
 - Assume an index on R.A and on S.C
 - Scan index R.A for looking up records of R satisfying $R.A = "c"$
 - For each record found, take value of R.C to scan index on S.C to get matching records of S
 - Filter out records of S, where $S.E \neq 2$
 - Join the corresponding records of R and S
 - Project on B,D



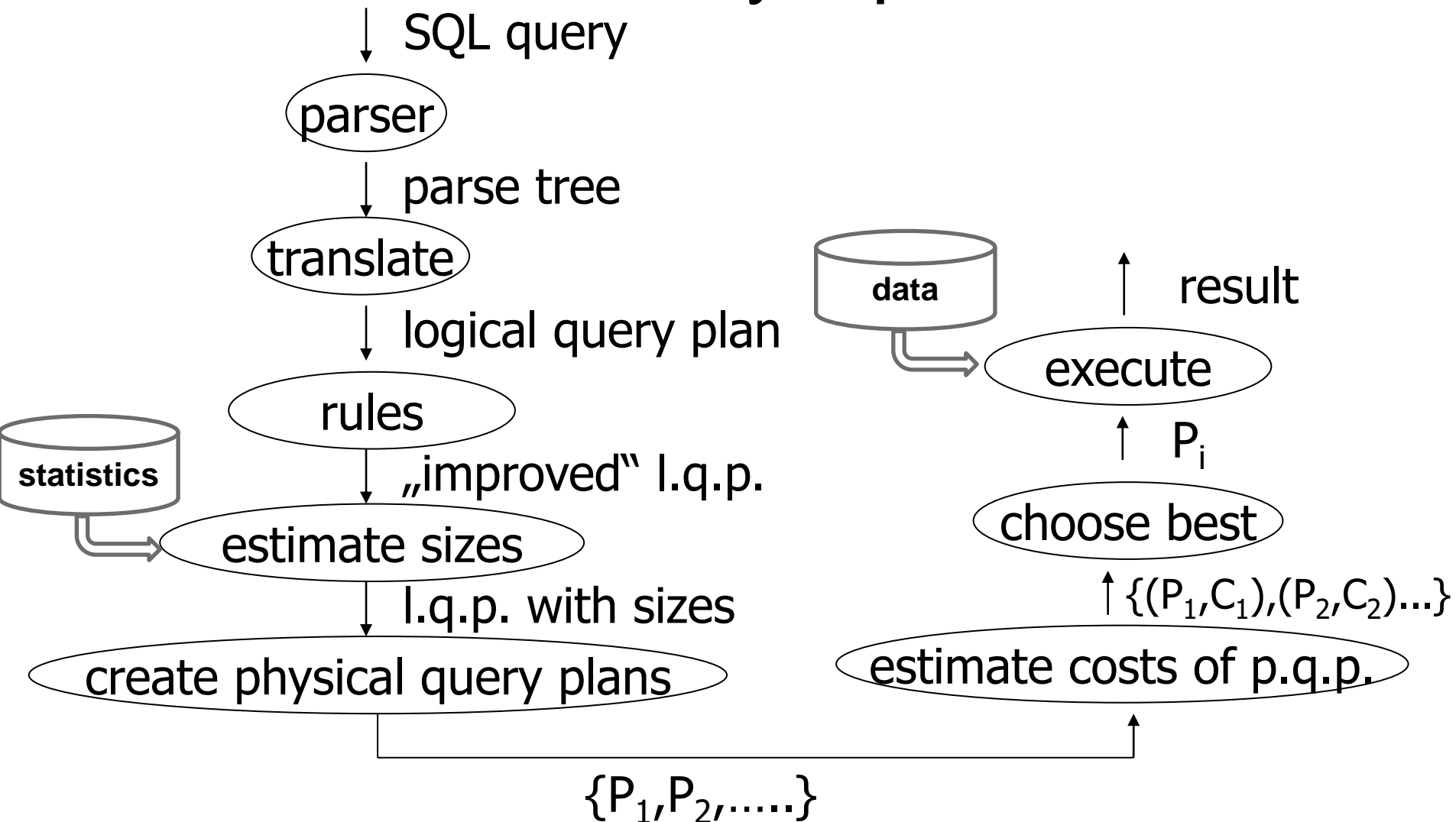








Overview on Query Optimization



Example: SQL query

■ Relations

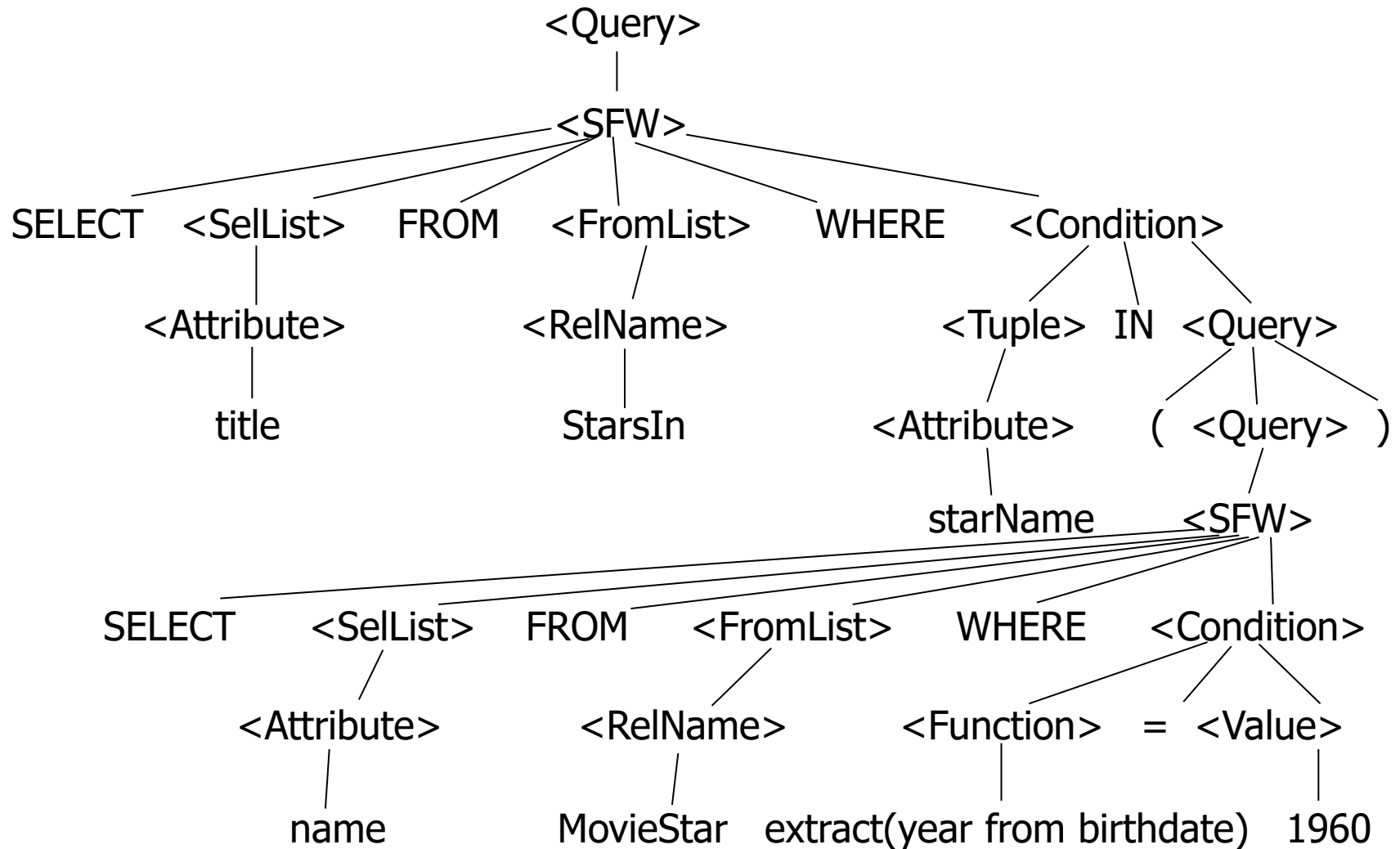
- StarsIn(title, year, starName)
- MovieStar(name, birthdate)

■ Query

- Select movies with stars born in 1960:

```
□ SELECT title
  FROM StarsIn
 WHERE starName IN (
     SELECT name
   FROM MovieStar
  WHERE extract(year from birthdate) = 1960
 );
```

Example: Parse Tree



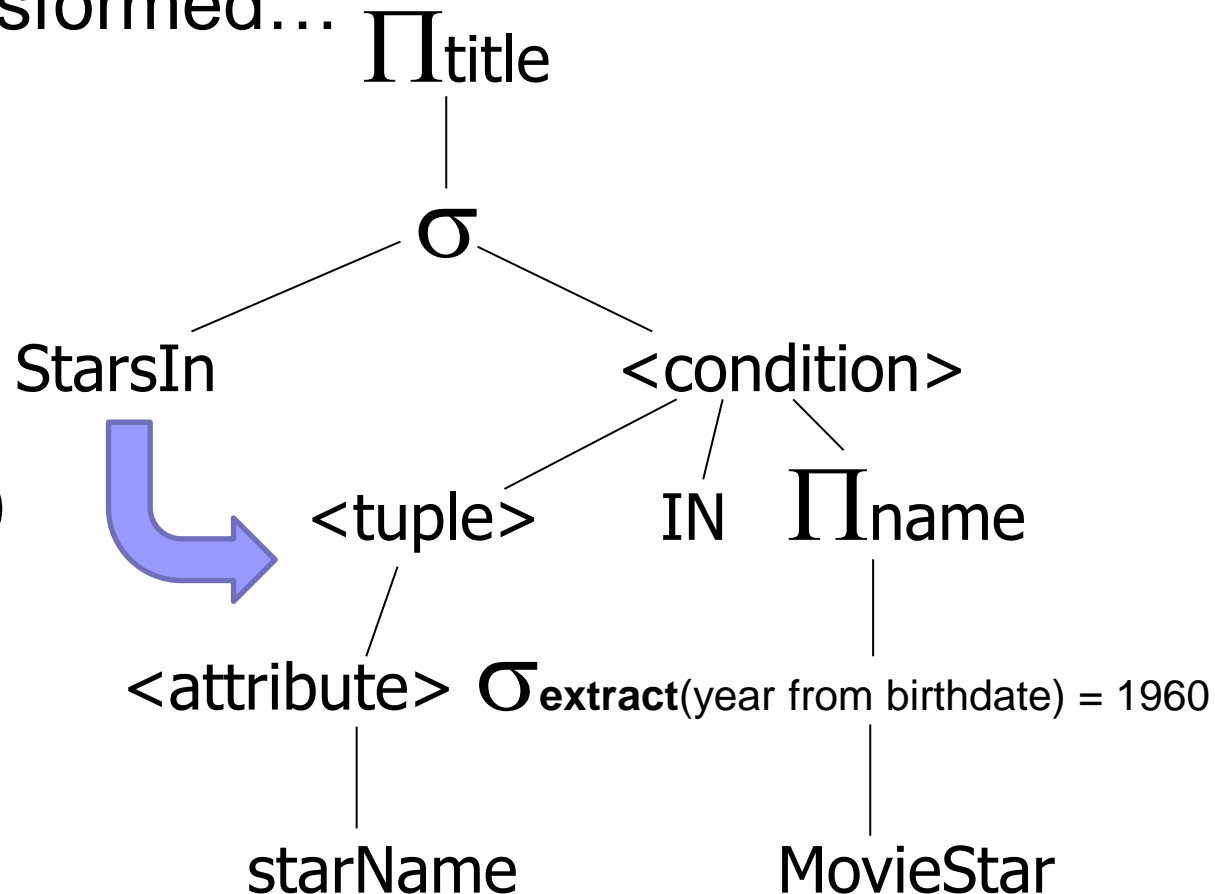
Example: Parse Tree in Rel. Algebra

- Selection has two arguments

 - Must be transformed...

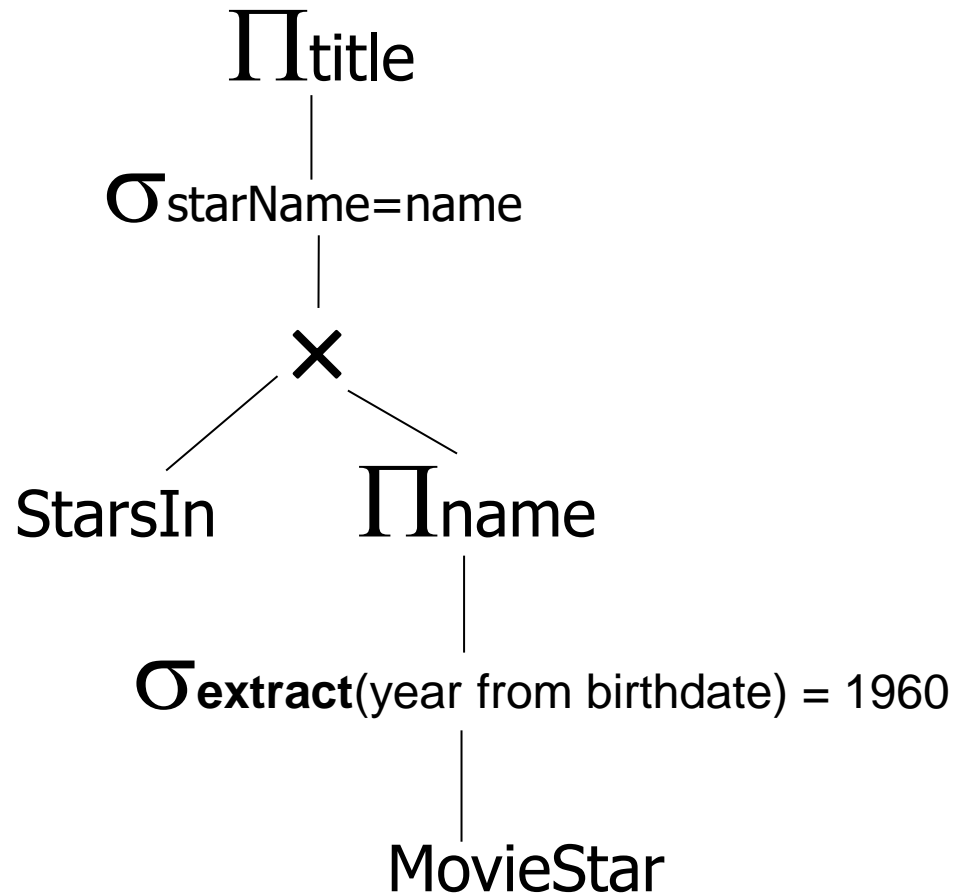
- IN operator

 - i.e., avoiding nested queries (sub-selects)



Example: Logical Query Plan

- IN operator replaced with product



Example: Improving Logical Plan

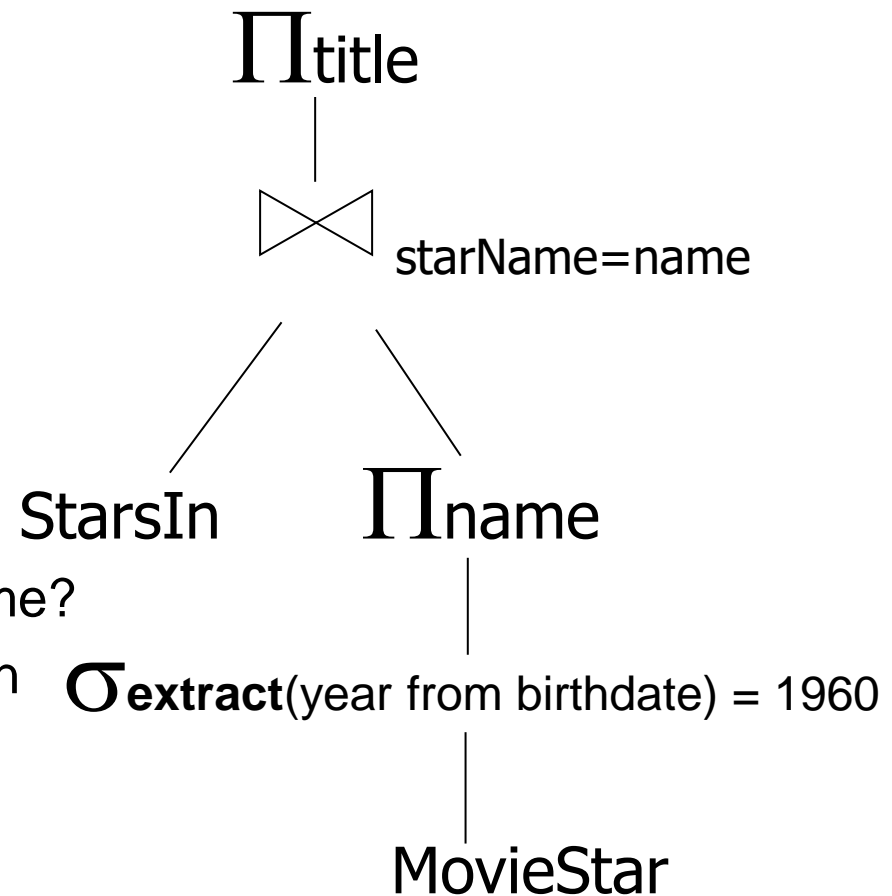
- Substitution of cross product and selection
 - by join

- Next option

- Push project to relation *StarsIn*?

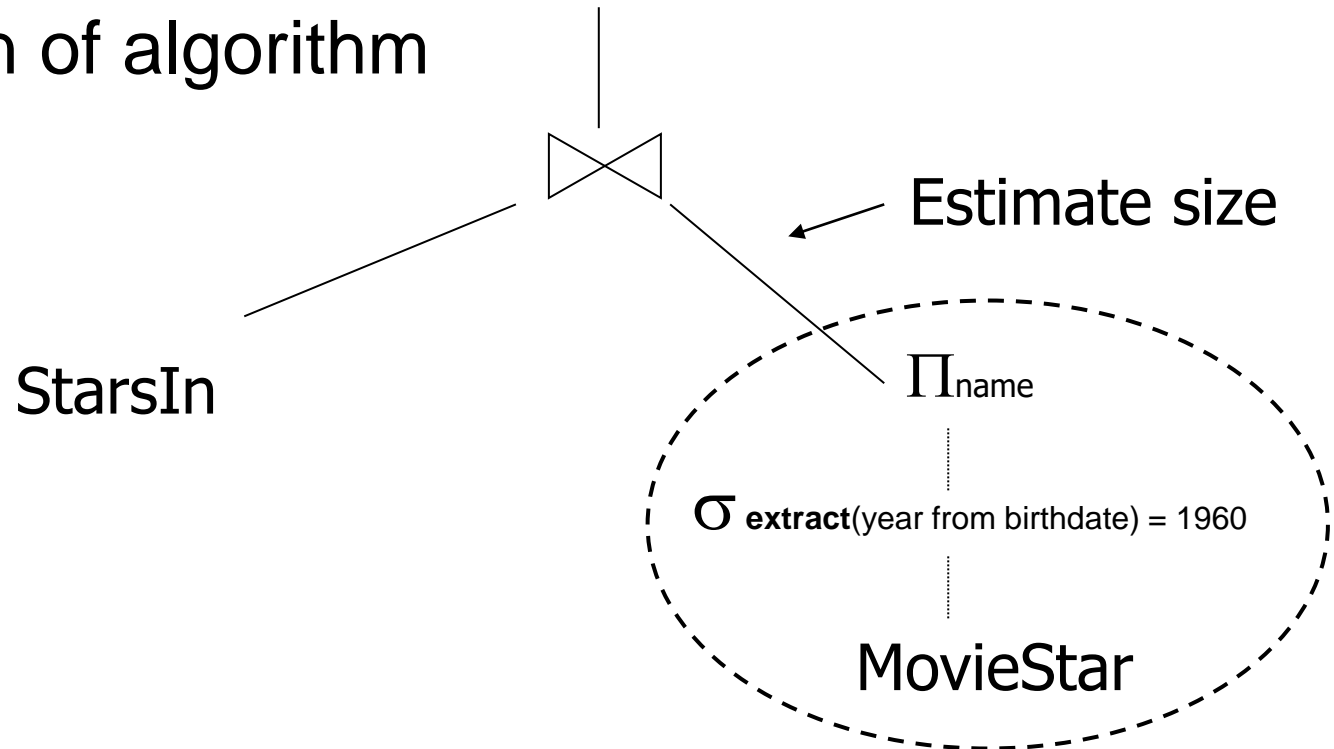
- Depends on:

- index on *StarsIn.starName*?
- many attributes in *StarsIn*

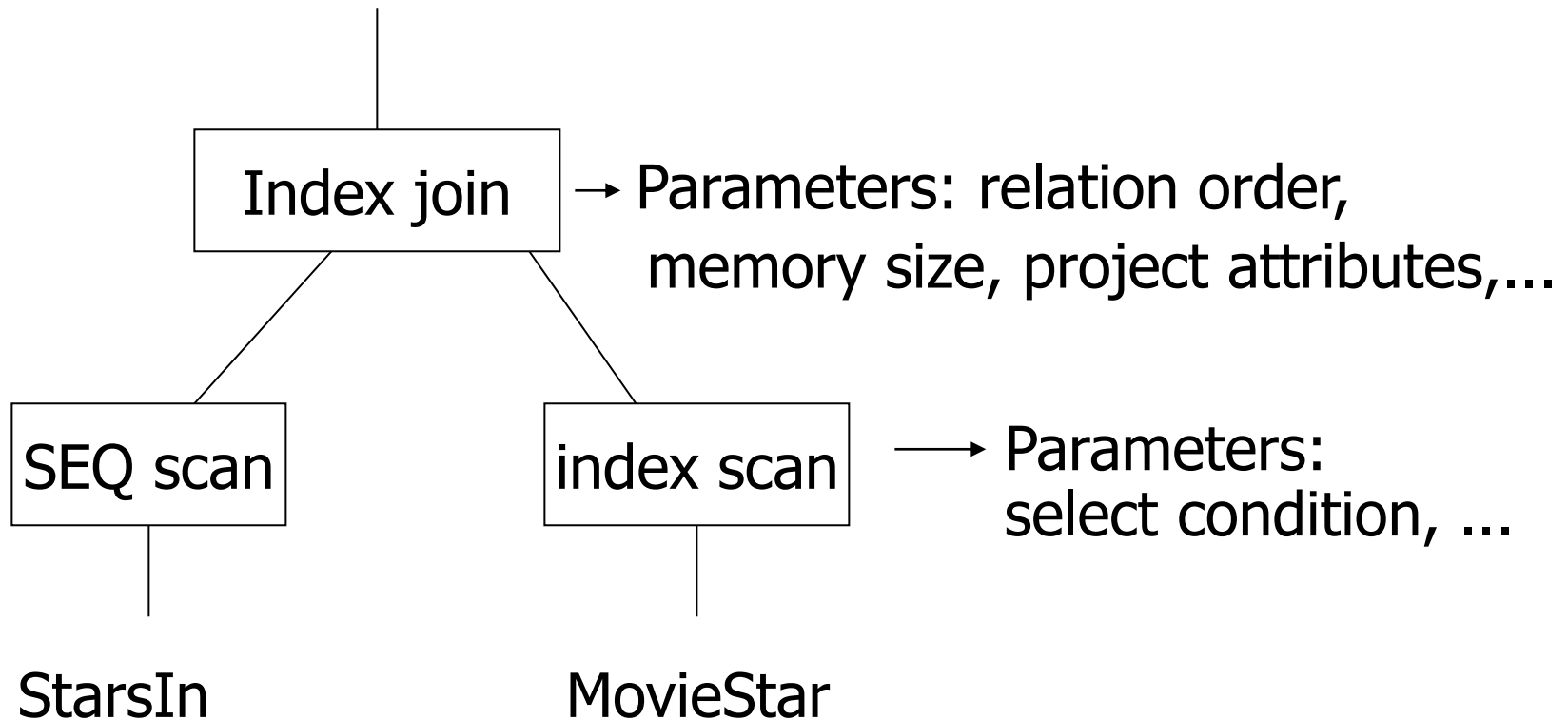


Example: Estimate Result Sizes

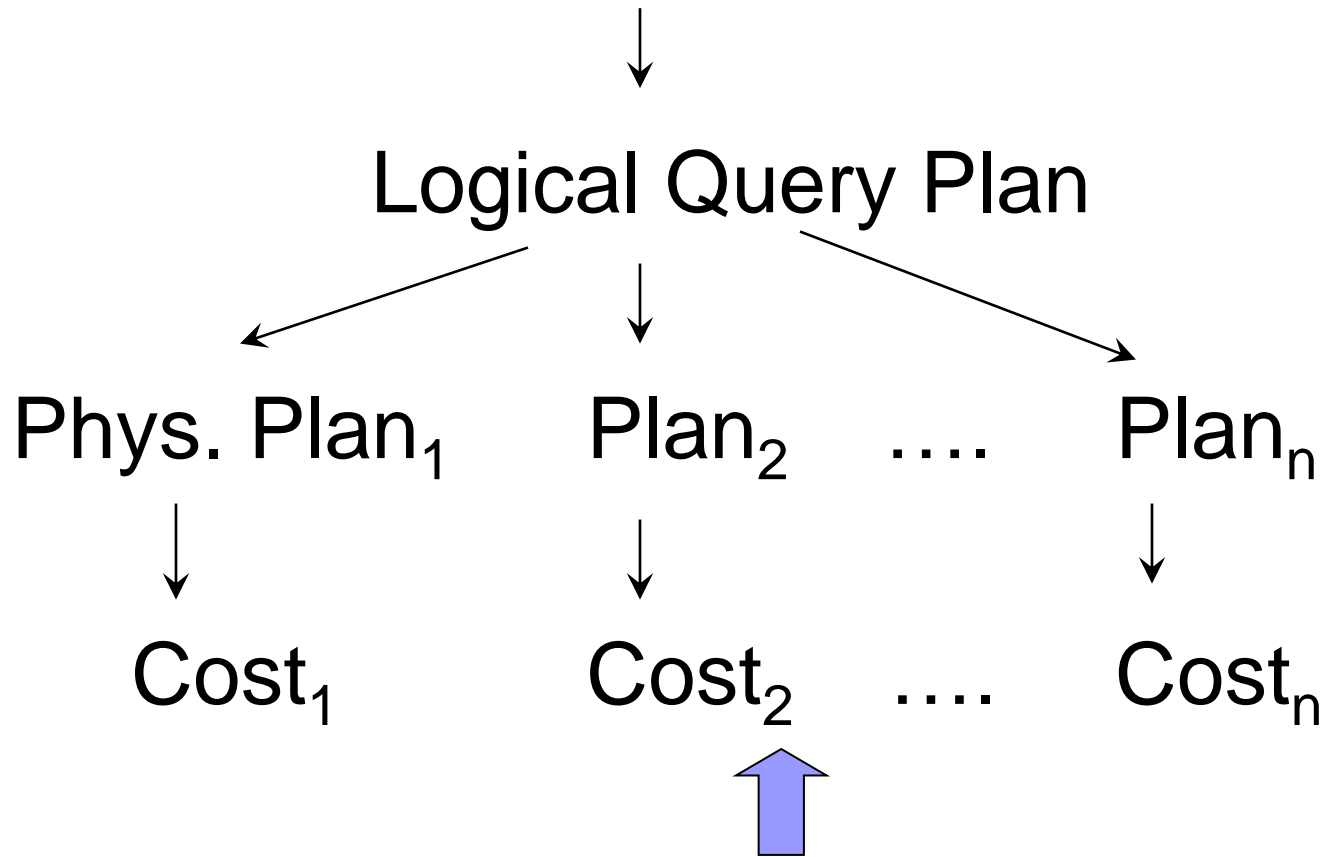
- Before generating physical plans
- Influences
 - estimation of evaluation costs
 - selection of algorithm



Example: Possible Physical Plan



Example: Estimate Costs



Pick the plan with the best cost!

Query Optimization

- Relational algebra level
- Detailed query plan level
 - Estimate costs
 - Without indexes
 - With indexes
 - Consider size of memory buffers
 - Generate and compare plans

Optimization in Relational Algebra

- Transformation rules

- Must preserve equivalence

- What are good transformations?

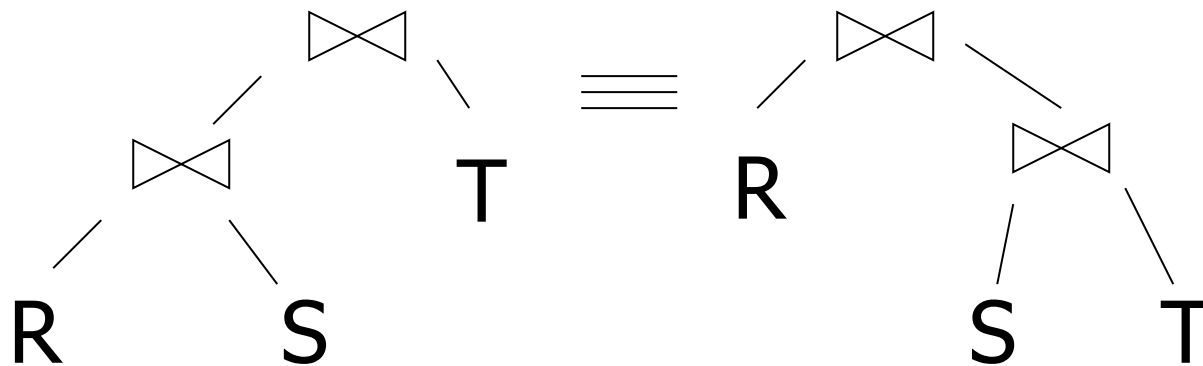
Transformation Rules in R.A.

- Natural join

- Relation order is not important since all attributes are preserved

- Example: $R \bowtie S = S \bowtie R$

$$(R \bowtie S) \bowtie T = R \bowtie (S \bowtie T)$$



Transformation Rules

- Same for cartesian product and union

$$R \times S = S \times R$$

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

$$R \cup S = S \cup R$$

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

Transformation Rules

■ Selects

$$\sigma_{p1 \wedge p2}(R) = \sigma_{p1} [\sigma_{p2}(R)]$$

$$\sigma_{p1 \wedge p2}(R) = [\sigma_{p1}(R)] \cap [\sigma_{p2}(R)]$$

$$\sigma_{p1 \vee p2}(R) = [\sigma_{p1}(R)] \cup [\sigma_{p2}(R)]$$

Question of Tuple Duplicates

■ Sets vs. bags?

- Relations are bags (in reality)

■ Example

- $R = \{a, a, b, b, b, c\}$

- $S = \{b, b, c, c, d\}$

R and S can be observed as output of some operations, e.g., $\Pi_{B,D} [\sigma_{A='c'}(tbl1)]$

■ $R \cap S = ?$

- MIN: $R \cap S = \{b, b, c\}$

SQL: INTERSECT ALL

■ $R \cup S = ?$

- SUM: $R \cup S = \{a, a, b, b, b, b, b, c, c, c, d\}$ SQL: UNION ALL

- MAX: $R \cup S = \{a, a, b, b, b, c, c, d\}$

Option MAX: Select Decomposition

- Select decomposition:

$$\sigma_{p_1 \vee p_2}(R) = \sigma_{p_1}(R) \cup \sigma_{p_2}(R)$$

- Example: $R = \{a, a, b, b, b, c\}$

- a, b satisfy p_1 ; b, c satisfy p_2

$$\sigma_{p_1 \vee p_2}(R) = \{a, a, b, b, b, c\}$$

$$\sigma_{p_1}(R) = \{a, a, b, b, b\}$$

$$\sigma_{p_2}(R) = \{b, b, b, c\}$$

$$\sigma_{p_1}(R) \cup_{\max} \sigma_{p_2}(R) = \{a, a, b, b, b, c\}$$

Choice of Correct Option

- Pragmatic solution for \cup
 - Use “SUM” for bag union
 - “MAX” for dividing disjunctive predicates (\vee)
- Some rules cannot be applied to bags!
 - Associativity of except: $R - (S - T)$
 - Distributivity: $R \cap (S \cup T)$
 $\neq (R \cap S) \cup (R \cap T)$

Transformation Rules

■ Notation:

- X = set of attributes
- Y = set of attributes
- $XY = X \cup Y$

■ Project

$$\pi_{xy}(R) = \pi_x[\pi_y(R)]$$

Transformation Rules

- Combining select and natural join

- Let

p = expr. containing only attrs. of R

q = expr. containing only attrs. of S

m = expr. containing attrs. of both R, S

$$\sigma_p (R \bowtie S) = [\sigma_p (R)] \bowtie S$$

$$\sigma_q (R \bowtie S) = R \bowtie [\sigma_q (S)]$$

Transformation Rules

- Combining select and natural join
 - Other rules can be derived

$$\sigma_{p \wedge q} (R \bowtie S) = [\sigma_p (R)] \bowtie [\sigma_q (S)]$$

$$\sigma_{p \wedge q \wedge m} (R \bowtie S) = \sigma_m \left[(\sigma_p (R)) \bowtie (\sigma_q (S)) \right]$$

$$\sigma_{p \vee q} (R \bowtie S) =$$

$$\left[(\sigma_p (R)) \bowtie S \right] \cup_{\max} \left[R \bowtie (\sigma_q (S)) \right]$$

Transformation Rules

- Combining select and natural join
 - Example of rule derivation

$$\sigma_{p \wedge q} (R \bowtie S) =$$

$$\sigma_p [\sigma_q (R \bowtie S)] =$$

$$\sigma_p [R \bowtie \sigma_q (S)] =$$

$$[\sigma_p (R)] \bowtie [\sigma_q (S)]$$

Transformation Rules

- Combining select and natural join

- Example of rule derivation

- Let

- $m = \text{expr. containing only attrs. common in R and S, but does not compare them}$

$$\sigma_m (R \bowtie S) = [\sigma_m (R)] \bowtie [\sigma_m (S)]$$

Transformation Rules

- Combining project and select

- Let

x = attribute subset of R

z = attributes referenced in expr. P
(subset of R)

$$\pi_x[\sigma_P(R)] = \pi_x \left(\sigma_P \left[\overset{\pi_{xz}}{\cancel{\pi_x}}(R) \right] \right)$$

Transformation Rules

- Combining project and natural join
- Let
 - x = attribute subset of R
 - y = attribute subset S
 - z = attributes common in R and S

$$\pi_{xy} (R \bowtie S) =$$

$$\pi_{xy} \left(\left[\pi_{xz} (R) \right] \bowtie \left[\pi_{yz} (S) \right] \right)$$

Transformation Rules

- Combining previous with select

$$\pi_{xy} (\sigma_p (R \bowtie S)) =$$

$$\pi_{xy} (\sigma_p [\pi_{xz'} (R) \bowtie \pi_{yz} (S)])$$

$$z' = z \cup \{\text{attributes referenced in } P\}$$

Transformation Rules

- Combining project, select and Cartesian product

$$\pi_{xy} (\sigma_p (R \times S)) =$$

$$\pi_{xy} (\sigma_p [\pi_{x'} (R) \times \pi_y (S)])$$

$$x' = x \cup \{ \text{attributes referenced in P} \}$$

Transformation Rules

- Combining select and union

$$\sigma_P(R \cup_{\text{sum}} S) = \sigma_P(R) \cup_{\text{sum}} \sigma_P(S)$$

- Combining select and except

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

- Select can also be applied to S

- May be convenient for shrinking relation before doing except

- Are there some limits on P in both above?

Good Transformations

- Select early

$$\sigma_p (R \bowtie S) \rightarrow [\sigma_p (R)] \bowtie S$$

- Project early

$$\pi_x [\sigma_p (R)] \rightarrow \pi_x (\sigma_p [\pi_{x'} (R)])$$

□ Example:

$$x' = x \cup \{ \text{attributes referenced in } P \}$$

- $R(A,B,C,D,E,F,G,H,I,J)$ result={E}

- Filter using $P: (A=3) \wedge (B=\text{"cat"})$

$$\pi_E (\sigma_p (R)) \quad \text{vs.} \quad \pi_E (\sigma_p (\pi_{ABE}(R)))$$

Good Transformations

$$\begin{aligned}\sigma_{p_1 \wedge p_2} (R) &\rightarrow \sigma_{p_1} [\sigma_{p_2} (R)] \rightarrow \sigma_{p_2} [\sigma_{p_1} (R)] \\ &\rightarrow [\sigma_{p_1} (R)] \cap [\sigma_{p_2} (R)]\end{aligned}$$

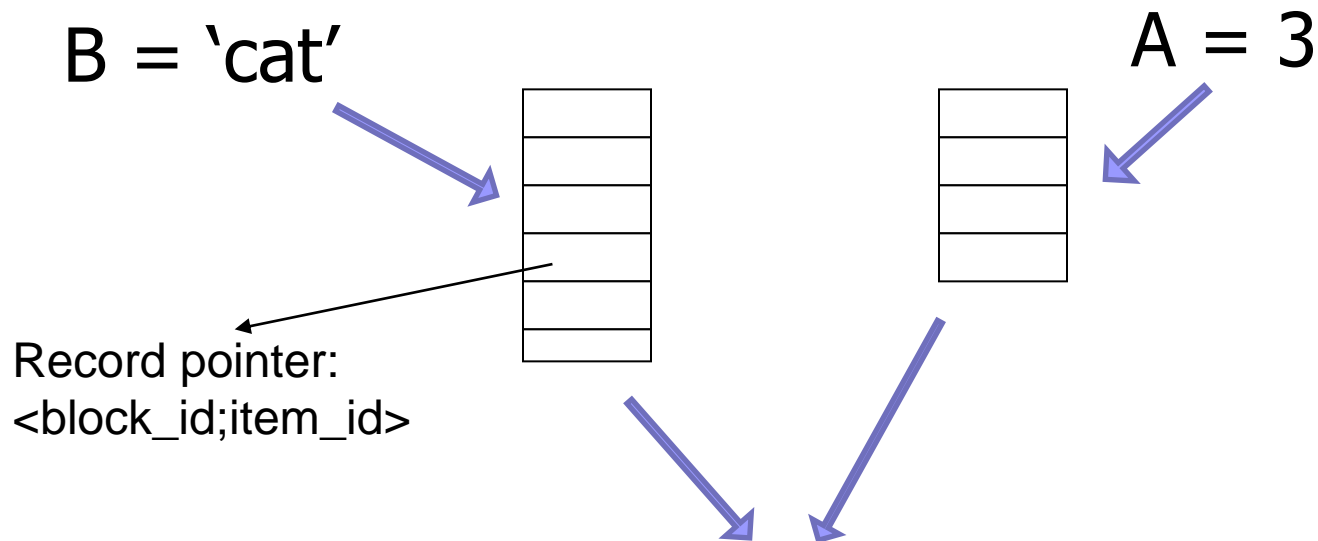
$$\sigma_{p_1 \vee p_2} (R) \rightarrow \sigma_{p_1}(R) \cup_{\max} \sigma_{p_2}(R)$$

$$R \bowtie S \rightarrow S \bowtie R$$

Good Transformations

- Assume indexes
 - On A and on B

$$\begin{aligned}\sigma_{(A=3) \wedge (B=\text{"cat"})}(R) \\ = \sigma_{(A=3)}(R) \cap \sigma_{(B=\text{"cat"})}(R)\end{aligned}$$



Intersect pointers to get
pointers to matching tuples

Good Transformations

■ Previous case in SQL and PostgreSQL

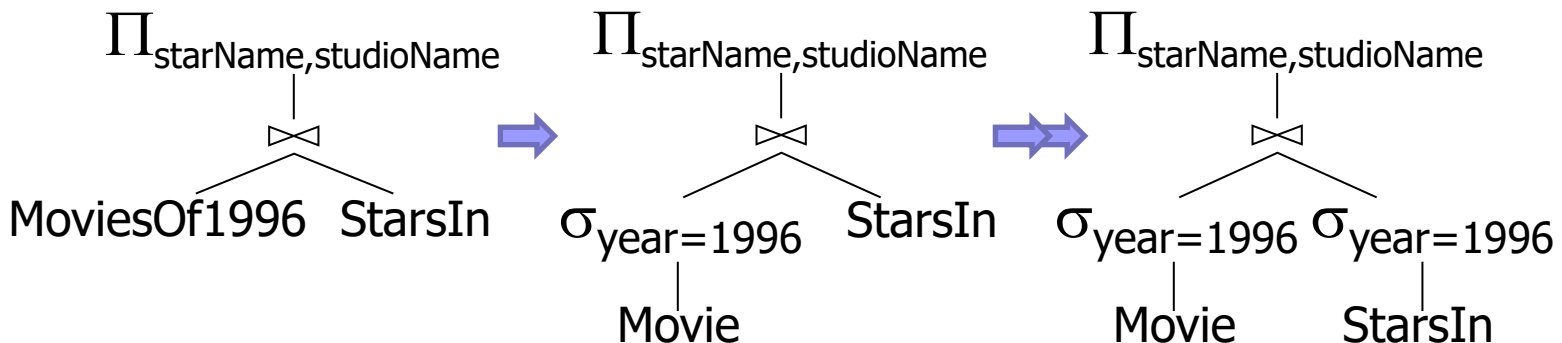
```
SELECT * FROM customer
WHERE username < 'user100' AND customerid < 1000;
```

```
Bitmap Heap Scan on customers (cost=25.76..61.62 rows=10 width=13)
  Recheck Cond: ((username)::text < 'user100'::text)
  -> BitmapAnd (cost=25.76..25.76 rows=10 width=0)
    -> Bitmap Index Scan on ix_cust_username (cost=0.00..5.75 rows=200)
        Index Cond: ((username)::text < 'user100'::text)
    -> Bitmap Index Scan on customers_pkey (cost=0.00..19.75 rows=1000)
        Index Cond: (customerid < 1000)
```


Good Transformations

- Push select to relations → apparently OK
 - But: Firstly, move them as far as possible; next push them back
- Example:
 - Relations:
 - $StarsIn(\underline{title}, \underline{year}, \underline{starName})$
 - $Movie(\underline{title}, \underline{year}, \underline{studioName})$
 - View: create view *MoviesOf1996* as
select * from *Movie* where *year* = 1996;
- Query: select *starName*, *studioName*
from *MoviesOf1996* natural join *StarsIn*;

prerequisite to be able to do it!!!



Good Transformations: Summary

■ Recommendations:

- No transformation is always good.
- Usually, good to
 - select early
 - project early

■ Eliminate common sub-expressions

■ Eliminate tuple duplicates

Query Processing: Overview

- Relational algebra level
 - Transformation rules
 - Apply good rules
- Detail (physical) query plan level
 - Estimate costs
 - Generate and compare plans

Estimating Cost of Query Plan

- Why? To “grade” a physical plan.
 - Choose proper algorithms to save time and RAM during evaluation.
 - Many rows may need more RAM buffers.
 - Multiple scans over data to save RAM.
- Why saving RAM?
 - More queries to be processed at parallel.
 - Individual operations of one query may need to keep all data in RAM buffers.
 - May lead to storing it to a disk.

Estimating Cost of Query Plan

- The costs are in an arbitrary unit.
 - not milliseconds or any other time unit.
- Process:
 1. Estimate size of result
 - Contributes to CPU costs
 2. Estimate number of I/O operations
 - Contributes to “slowness” of the query
- Done for each operation
 - in the logical query plan
- Based on statistics (metadata about tables)

Estimating Result Size

- Keep statistics for relation R
 - $T(R)$ – # tuples in R
 - $S(R)$ – # of bytes in each R tuple
 - $S(R,A)$ – length in bytes of values of attribute A
 - $B(R)$ – # of blocks to hold all R tuples
 - $V(R, A)$ – # distinct values in R for attribute A
- Good estimates need
 - Statistics up to date!

Statistics Example

Mind encoding – utf8 takes more bytes per char.

■ Relation R

- Attribute A – string, max. 20
 - $S(R,A) = 3$ ← average length
- Attribute B – integer, 4 B
- Attribute C – date, 8 B
- Attribute D – string, max. 5
 - $S(R,D) = 1$

A	B	C	D
cat	1	10.2.98	a
cat	1	20.3.98	b
dog	1	30.4.98	a
dog	1	14.6.98	c
bat	1	15.6.98	d

■ Statistics

- $T(R) = 5$
- $V(R,A) = 3$
- $V(R,C) = 5$
- $S(R) = 16$
- $V(R,B) = 1$
- $V(R,D) = 4$

Estimating Result Size

- Cartesian product $W = R_1 \times R_2$
 - $T(W) = T(R_1) \cdot T(R_2)$
 - $S(W) = S(R_1) + S(R_2)$

Estimating Result Size

- Select $W = \sigma_{Z=val}(R)$

- $S(W) = S(R)$

- $T(W) = ?$

- $W = \sigma_{A='cat'}(R)$

$$T(W) = \frac{T(R)}{V(R,A)} = 5/3$$

- $W_2 = \sigma_{B=2}(R)$

$$T(W_2) = ?$$

A	B	C	D
cat	1	10.2.98	a
cat	1	20.3.98	b
dog	1	30.4.98	a
dog	1	14.6.98	c
bat	1	15.6.98	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

Estimating Result Size

■ Assumption of last estimate

□ Values are uniformly distributed over possible $V(R,Z)$ values!

■ $f(\text{val}) = 1 / V(R,Z)$

■ $T(\sigma_{Z=\text{val}}(R)) = T(R) \cdot f(\text{val})$

■ Alternative

□ Uniform distribution of values over domain with $\text{DOM}(R,Z)$ values

■ # values in domain is denoted as $\text{DOM}(R,Z)$

■ $f(\text{val}) = 1 / \text{DOM}(R,Z)$

Estimating Result Size

- Select $W = \sigma_{Z=val}(R)$

- $T(W) = ?$

- using $DOM(R,*)$

- Formula derivation

- $W = \sigma_{C=val}(R)$

- $T(W) = f(val) \cdot T(R)$
 $= 1/10 * 5 = 0,5$

- $W = \sigma_{B=val}(R)$

- $T(W) = (1/10)*5$

- $W = \sigma_{A=val}(R)$

- $T(W) = 0,5$

A	B	C	D
cat	1	10.2.98	a
cat	1	20.3.98	b
dog	1	30.4.98	a
dog	1	14.6.98	c
bat	1	15.6.98	d

$V(R,A)=3$

$V(R,B)=1$

$V(R,C)=5$

$V(R,D)=4$

$DOM(R,A)=10$

$DOM(R,B)=10$

$DOM(R,C)=10$

$DOM(R,D)=10$

Estimating Result Size

- Select $W = \sigma_{Z=val}(R)$

- Original solution

$$T(W) = \frac{T(R)}{V(R,Z)}$$

- Alternative

$$T(W) = \frac{T(R)}{DOM(R,Z)}$$

- **Mostly used:**

- histogram over the most frequent ones
- plus, uniform distribution of remaining ones.

A	B	C	D
cat	1	10.2.98	a
cat	1	20.3.98	b
dog	1	30.4.98	a
dog	1	14.6.98	c
bat	1	15.6.98	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

$$DOM(R,A)=10$$

$$DOM(R,B)=10$$

$$DOM(R,C)=10$$

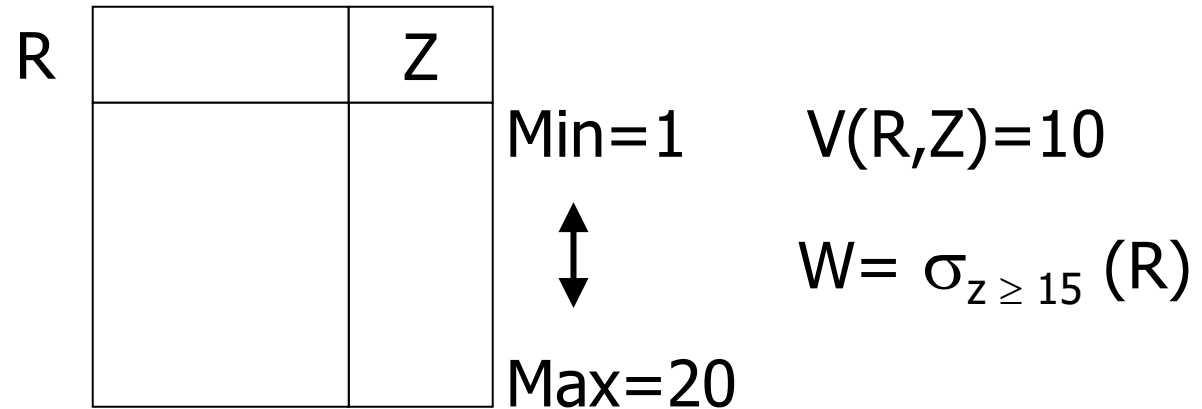
$$DOM(R,D)=10$$

Estimating Result Size

- Select $W = \sigma_{Z \geq \text{val}}(R)$
 - Solution 1
 - $T(W) = T(R) / 2$
 - Solution 2
 - $T(W) = T(R) / 3$
 - **Solution 3**
 - Estimate values in range – using histogram.

Estimating Result Size

- Select – estimate values in range



- Calculate fraction of unique values in range

$$f = \frac{20-15+1}{20-1+1} = \frac{6}{20}$$

$$\square T(W) = f \cdot T(R)$$

Estimate # values: Histogram

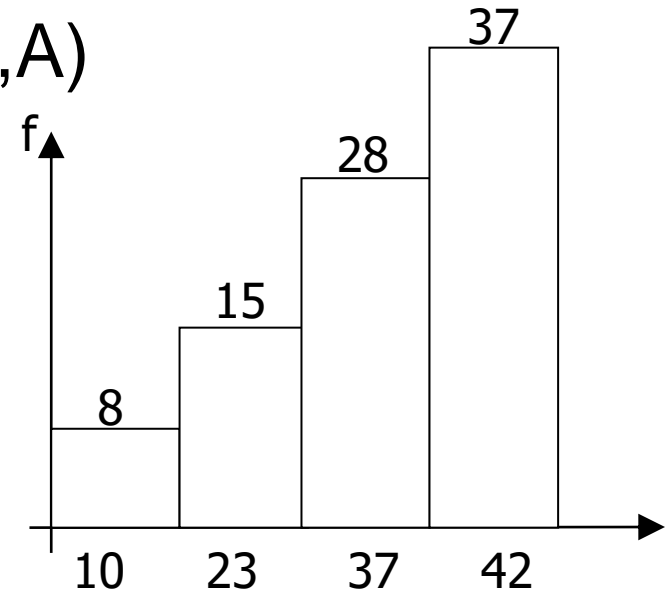
■ Histogram of attribute values

- Replaces $V(R,A)$ and $DOM(R,A)$
- More precise estimates

■ Distinct values

- Few → abs. number per each
- Many → quantization

- Ranges (intervals) of equal size (in recs)
- Percentiles
- Most frequent ones only
 - others all together (i.e., uniformly distributed)



Estimating Result Size

- Select $W = \sigma_{z \neq \text{val}}(R)$

- $T(W) = T(R) \cdot (1 - f(\text{val})) = T(R) \cdot (1 - 1/V(R,Z))$
 $= T(R) - \frac{T(R)}{V(R,Z)}$

- Typical solution

- $T(W) = T(R)$

- Mind special cases

- e.g., $V(R,Z) = 1$ and the value is “val”.

Estimating Result Size

- Natural join $W = R_1 \bowtie R_2$

- Notation

- X – attributes of R_1
 - Y – attributes of R_2

- Case 1

- $X \cap Y = \emptyset$

- Same as $R_1 \times R_2$

- Case 2

- $X \cap Y = Z$

- Follows...

Estimating Result Size: Natural Join

$R_1 \bowtie R_2$

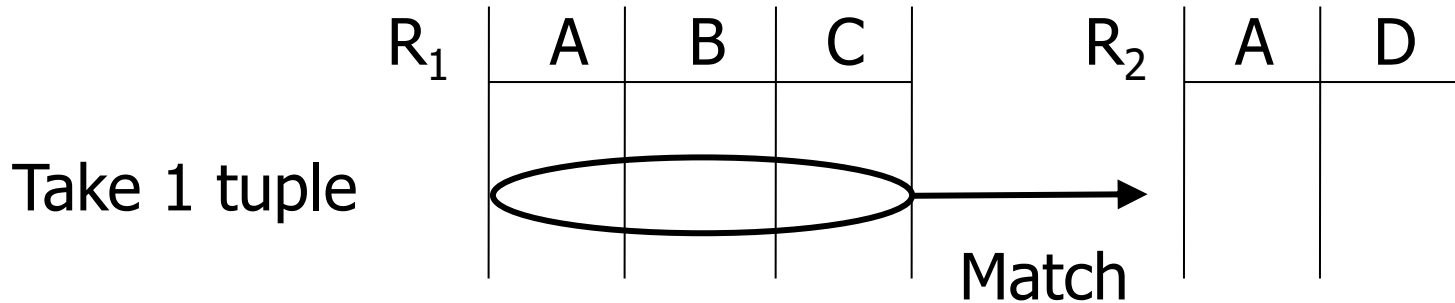
R_1	A	B	C

R_2	A	D

- Assumption $Z=\{A\}$ and also:
 - $V(R_1, A) \leq V(R_2, A)$
→ every A value in R_1 is in R_2
 - $V(R_1, A) \geq V(R_2, A)$
→ every A value in R_2 is in R_1

Estimating Result Size: Natural Join

- $V(R_1, A) \leq V(R_2, A)$



- One record is matched with $T(R_2) / V(R_2, A)$ records

- Assumption of uniform distribution

- Result:
$$T(W) = T(R_1) \cdot \frac{T(R_2)}{V(R_2, A)}$$

Estimating Result Size: Natural Join

- Overview of both variants

- $V(R_1, A) \leq V(R_2, A)$

$$T(W) = T(R_1) \cdot \frac{T(R_2)}{V(R_2, A)}$$

- $V(R_2, A) \leq V(R_1, A)$

$$T(W) = T(R_2) \cdot \frac{T(R_1)}{V(R_1, A)}$$

- Difference is in denominator

Estimating Result Size: Natural Join

- In general

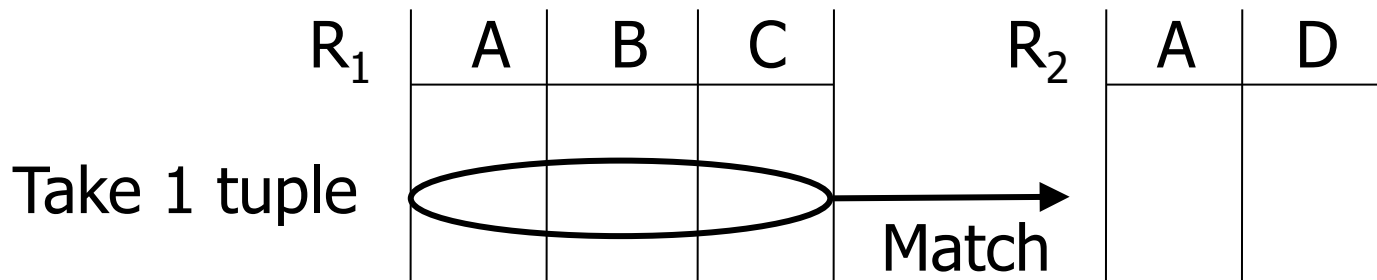
- $W = R_1 \bowtie R_2$

$$T(W) = \frac{T(R_1) \cdot T(R_2)}{\max \{ V(R_1, A), V(R_2, A) \}}$$

Estimating Result Size: Natural Join

- Alternate solution

- Values uniformly distributed over domain



- One rec. matches with $T(R_2)/\text{DOM}(R_2, A)$ recs.

- Result:

$$T(W) = \frac{T(R_1) \cdot T(R_2)}{\text{DOM}(R_2, A)} = \frac{T(R_1) \cdot T(R_2)}{\text{DOM}(R_1, A)}$$

assuming to be same

Estimating Result Size: Natural Join

- $W = R_1 \bowtie R_2$
 - $R_1(X), R_2(Y), X \cap Y = Z$
- Size of record
 - $S(W) = S(R_1) + S(R_2) - S(R_1, Z)$
 - Valid for all cases
- Number of tuples if Z contains more attrs.?
 - Assume they are independent

$$T(W) = \frac{T(R_1) \cdot T(R_2)}{\max\{V(R_1, A_1), V(R_2, A_1)\} \cdot \max\{V(R_1, A_2), V(R_2, A_2)\}}$$

Estimating Size: Project, Select

- Project $W = \Pi_{AB}(R)$

- $T(W) = T(R)$

- $S(W) = S(R, AB)$

- Select $W = \sigma_{A=a \vee B=b}(R)$

- $S(W) = S(R)$

- Let $n = T(R)$, $T(W) = n \cdot (1 - (1 - m_1/n) \cdot (1 - m_2/n))$

- $m_1 = T(R) / V(R, A)$

- $m_2 = T(R) / V(R, B)$

Estimating Size: Set Operations

■ Union, intersect, except

□ $\cup, \cap, -$

- $T(W)$ – choose average size

□ E.g.,

- $T(R \cup S) = T(R) + T(S)$... if \cup means UNION ALL

- $T(R \cup S) = [\max\{T(R), T(S)\}, T(R) + T(S)]$

□ So, use: $T(R \cup S) = \text{avg}\{ \max\{T(R), T(S)\}, T(R) + T(S) \}$

- If *set union* is evaluated

- $T(R - S) = T(R) - 1/2T(S)$

- $T(R \cap S) = \text{avg}\{ 0, \min\{T(R), T(S)\} \}$

■ DISTINCT

□ All attributes

- $\min\{ 1/2T(R), (V(R,A)*V(R,B)*...) \}$

Estimating Result Size

- For complex expressions, intermediate results and their stats are needed

- Example

$$\square W = [\underbrace{\sigma_{A=a}(R_1)}] \bowtie R_2$$

denote as U

- $T(U) = T(R_1) / V(R_1, A)$ $S(U) = S(R_1)$

- Need $V(U, *)$ to estimate costs of W!

- at least for the join attributes

Estimate Number of Values

- To estimate $V(U, *)$
 - $U = \sigma_{A=a}(R_1)$
 - Assume $R_1(A, B, C, D)$

Estimate # values: Example

- Relation R_1

- $U = \sigma_{A=a}(R_1)$

A	B	C	D
cat	1	10.2.98	a
cat	1	20.3.98	b
dog	1	30.4.98	a
dog	1	14.6.98	c
bat	1	15.6.98	d

$$V(R,A)=3$$

$$V(R,B)=1$$

$$V(R,C)=5$$

$$V(R,D)=4$$

- Estimates

- $V(U,A) = 1$

- $V(U,B) = 1$

- $V(U,C) = 1 \dots (T(R_1) / V(R_1,A))$

- $V(U,D) = 1 \dots (T(R_1) / V(R_1,A))$

sth. in
between

Not easy, if there is no
dependence btw. A and
B/C/D.

→ functional
dependencies

Estimate # values: Reality

■ Common solution

- $U = \sigma_{A=a}(R_1)$

- $V(U,A) = 1$

- $V(U,K) = T(U)$

- Primary key K of R_1 is an exception

- $V(U,*) = V(R,*)$ or $V(U,*) = T(U)$

■ As a result, original $V(R,*)$ can be used

- $V(U,*) = \min \{ V(R,*), T(U) \}$

Estimate # values

■ Join

$$\square U = R_1(A,B) \bowtie R_2(A,C)$$

■ Estimates:

$$\square V(U,A) = \min\{ V(R_1,A), V(R_2,A) \}$$

$$\square V(U,B) = \min\{ V(R_1,B), T(U) \}$$

$$\square V(U,C) = \min\{ V(R_2,C), T(U) \}$$

Estimate # values

■ Example

□ $Z = R_1(A,B) \bowtie R_2(B,C) \bowtie R_3(C,D)$

- $T(R_1) = 1000$ $V(R_1,A)=50$ $V(R_1,B)=100$
- $T(R_2) = 2000$ $V(R_2,B)=200$ $V(R_2,C)=300$
- $T(R_3) = 3000$ $V(R_3,C)=90$ $V(R_3,D)=500$

□ Intermediate result

- $U = R_1(A,B) \bowtie R_2(B,C)$
- $T(U) = T(R_1) \cdot T(R_2) / \max\{ V(R_1,B), V(R_2,B) \} = 10k$
- $V(U,A) = 50$
- $V(U,B) = \min\{ V(R_1,B), V(R_2,B) \} = 100$
- $V(U,C) = 300$

Estimate # values

■ Final result

□ $Z = U \bowtie R_3(C,D)$

■ $U(A,B,C)$

□ Estimates:

■ $T(Z) = 10\,000 \cdot 3\,000 / 300 = 100\,000$

■ $V(Z,A) = 50$

■ $V(Z,B) = 100$

■ $V(Z,C) = 90$

■ $V(Z,D) = 500$

Example of Stats in PostgreSQL

- Connect to student instance of PostgreSQL
 - See the first lecture for instructions
- Check the schema *xdohnal*
 - Relations: *predmet*, *skupina*, *hotel*
 - Observe both the relation and attribute statistics
 - Explanation of individual items in doc
 - <https://www.postgresql.org/docs/current/view-pg-stats.html>








Example of Stats in PostgreSQL

■ Relation hotel

Statistic	Value
Sequential Scans	4
Sequential Tuples Read	500
Index Scans	1
Index Tuples Fetched	500
Tuples Inserted	500
Tuples Updated	0
Tuples Deleted	0
Tuples HOT Updated	0
Live Tuples	500
Dead Tuples	0
Heap Blocks Read	5
Heap Blocks Hit	514
Index Blocks Read	4
Index Blocks Hit	599
Toast Blocks Read	
Toast Blocks Hit	
Toast Index Blocks Read	
Toast Index Blocks Hit	
Last Vacuum	
Last Autovacuum	
Last Analyze	
Last Autoanalyze	2010-04-15 13:52:03.54614+02
Table Size	40 kB
Toast Table Size	none
Indexes Size	32 kB

Example of Stats in PostgreSQL

■ Attribute hotel.id








Properties	Statistics	Dependencies	Dependents
Statistic		Value	
 Null Fraction		0	
 Average Width		4	
 Distinct Values		-1	
 Most Common Values			
 Most Common Frequencies			
 Histogram Bounds		{1,50,100,150,200,250,300,350,400,450,500}	
 Correlation		1	

■ Attribute hotel.name








Properties	Statistics	Dependencies	Dependents
Statistic		Value	
 Null Fraction		0	
 Average Width		9	
 Distinct Values		-1	
 Most Common Values			
 Most Common Frequencies			
 Histogram Bounds		{street1,street143,street189,street233,street279,street323,street369,street413,street459,street53,street99}	
 Correlation		-0.117997	

Example of Stats in PostgreSQL

■ Attribute hotel.state

Properties	Statistics	Dependencies	Dependents
Statistic		Value	
 Null Fraction		0	
 Average Width		7	
 Distinct Values		50	
 Most Common Values		{state32,state8,state14,state36,state42,state48,state6,state16,state30,state47}	
 Most Common Frequencies		{0.038,0.03,0.028,0.028,0.028,0.028,0.028,0.026,0.026,0.026}	
 Histogram Bounds		{state1,state12,state18,state21,state25,state29,state34,state4,state44,state5,state9}	
 Correlation		-0.00743129	

■ Attribute hotel.distance_to_center

Properties	Statistics	Dependencies	Dependents
Statistic		Value	
 Null Fraction		0	
 Average Width		4	
 Distinct Values		10	
 Most Common Values		{6,7,10,3,9,8,2,1,4,5}	
 Most Common Frequencies		{0.108,0.108,0.108,0.106,0.102,0.098,0.096,0.094,0.092,0.088}	
 Histogram Bounds			
 Correlation		0.102588	

Summary

- Estimating size of results is an “art”
- Do not forget:
 - Statistics must be kept up to date for precise estimates
→ necessity to maintain them during table updates
 - What are the costs to update stats?

Statistics Update

- Stats do not change rapidly
 - in short time
- Inaccurate stats may also help
- Instant stats update
 - Can become a bottleneck
 - Stats are used very often
- → Do not update often

Statistics Update

- Run periodically
 - After some time elapses
 - After some number of updates are made
- Slow for $V(R,A)$
 - Especially if histograms are kept
 - → Use a data sample
 - If almost all are distinct → $V(R,A) \approx T(R)$
 - If not many are distinct → we likely saw most of them

Estimating Costs of Query Plan: Outline

- Estimating the size of results
 - Already done
- Estimating # of IOs
 - Next lecture...

- Generate and compare plans

Takeaways

- Query plan
 - What it is
 - How to transform it and why
- Term “costs”
- Types of statistics maintained by DBMS
- How to estimate
 - result size of an operation
 - and the other statistics