PA152: Efficient Use of DB

6. Query Processing

Vlastislav Dohnal

Query Processing

- Procedure:
  - Query
  - Checking syntax and semantics
    - Parse Tree
  - Logical query plan
    - Plan modifications
  - Physical query plan
  - Evaluation

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

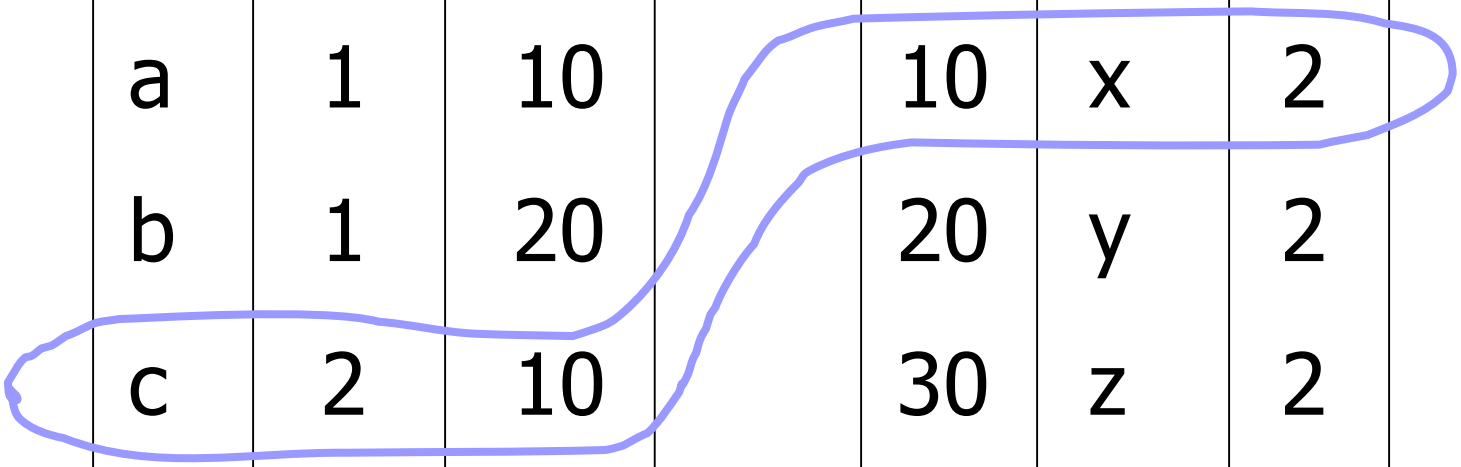
Example

R

A	B	C
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



Result:

B	D
2	x

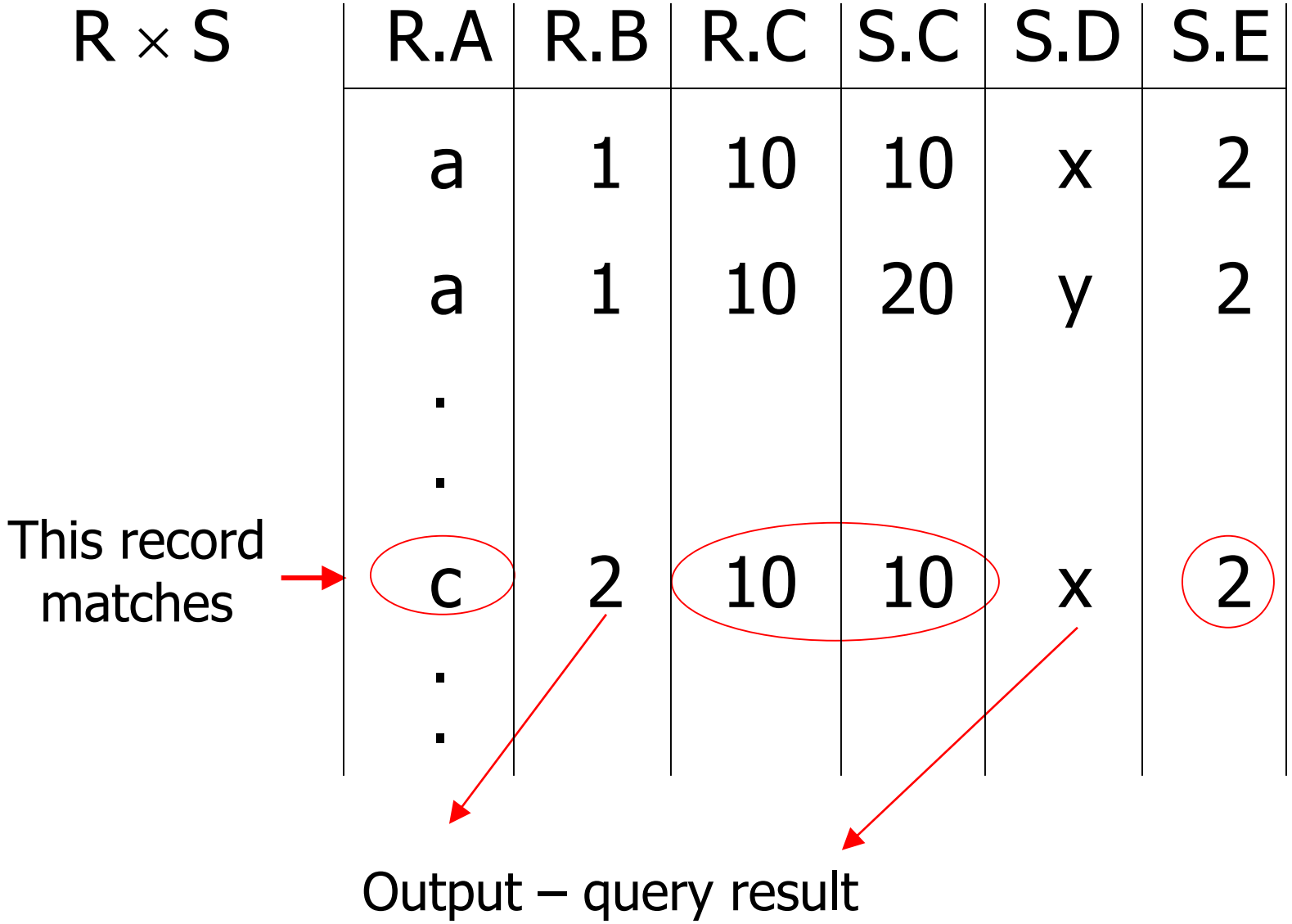
How to evaluate this query?

1. way

- Cartesian product
- Selecting records
- Projection

$R \times S$

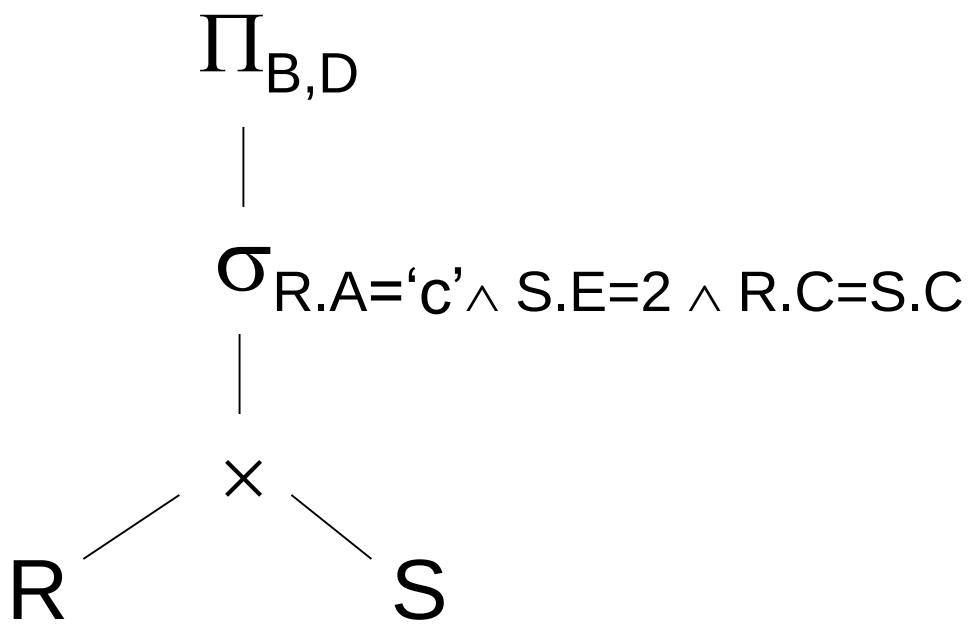
R.A	R.B	R.C	S.C	S.D	S.E
a	1	10	10	x	2
a	1	10	20	y	2
.					
.					
c	2	10	10	x	2
.					
.					



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

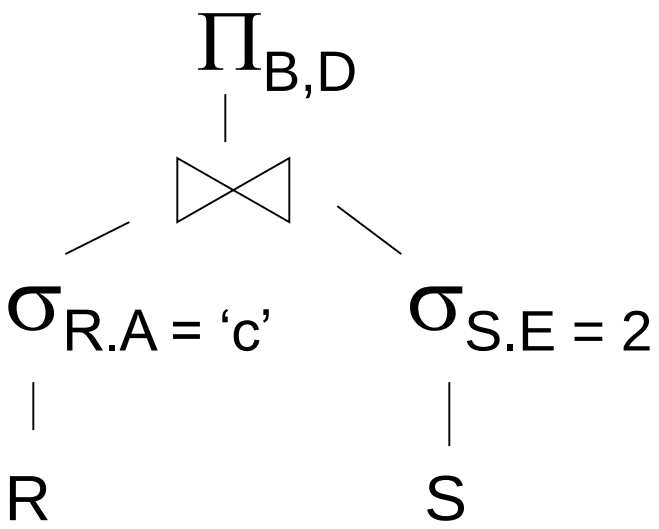
Describing Evaluation Plans

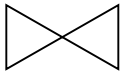
- Using relational algebra
  - $\Pi_{B,D} [\sigma_{R.A='c' \wedge S.E=2 \wedge R.C = S.C} (R \times S)]$
- Example of Plan 1:
  - Query plan



Describing Evaluation Plans

- Example of Plan 2:

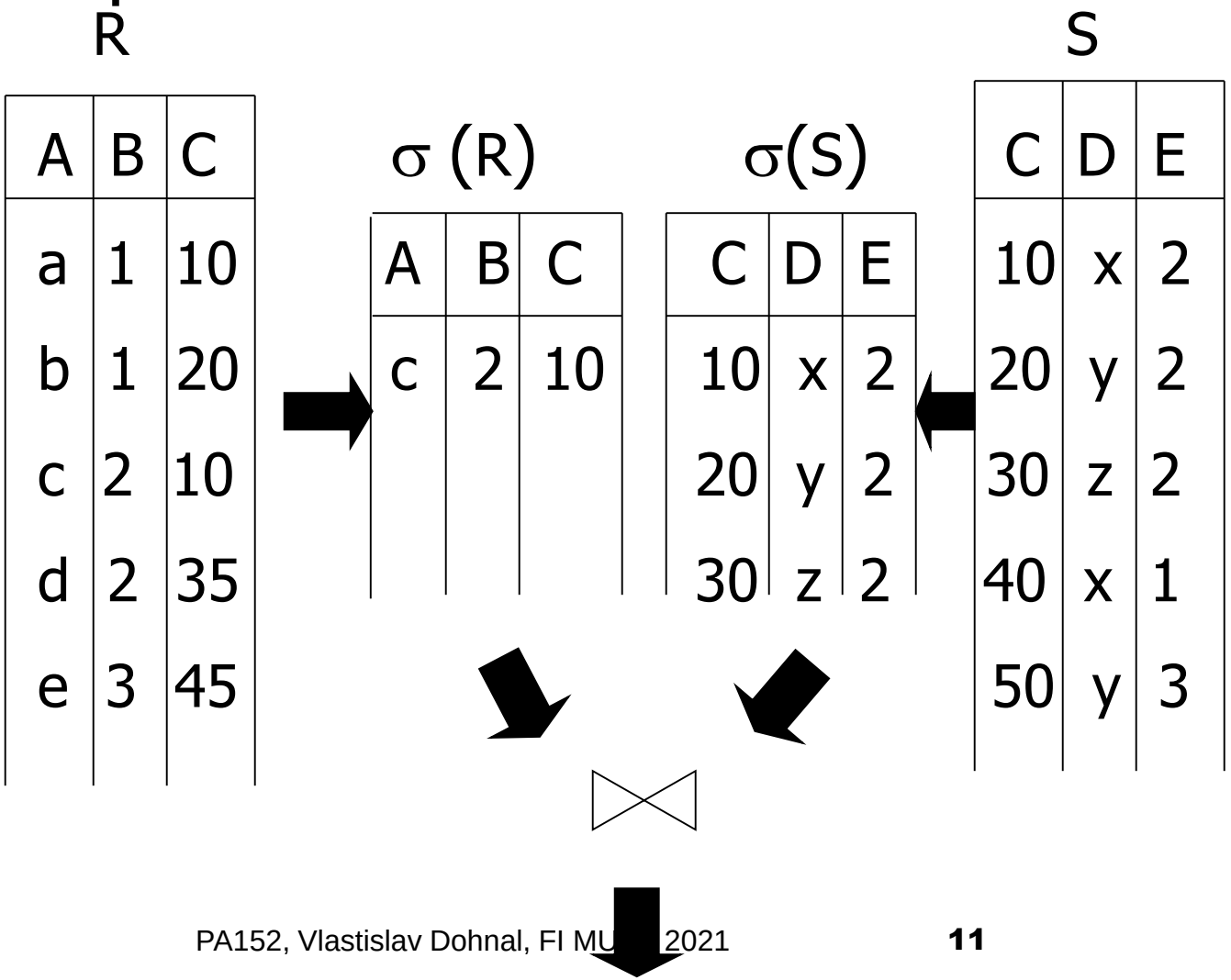




natural join

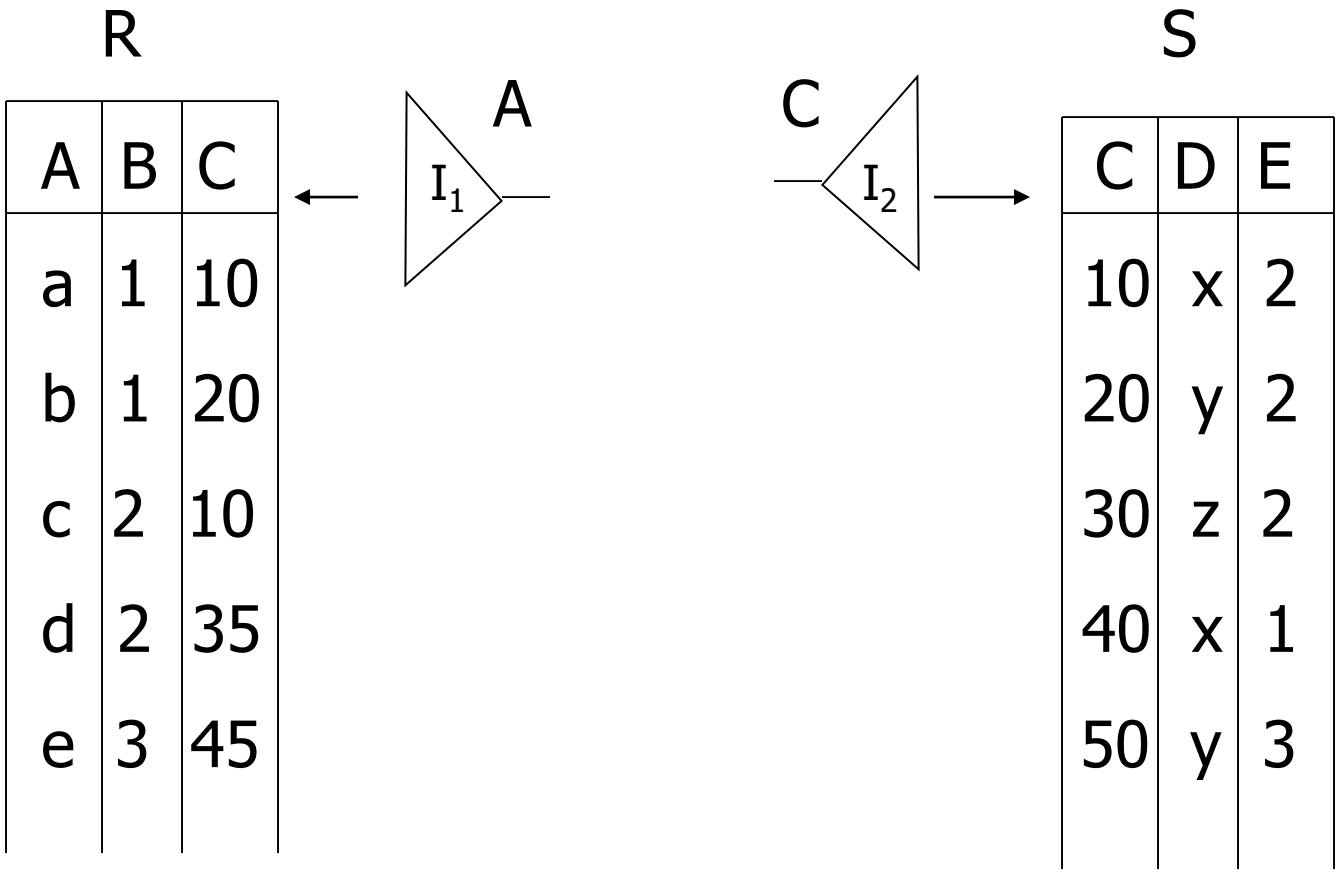
Physical Plan

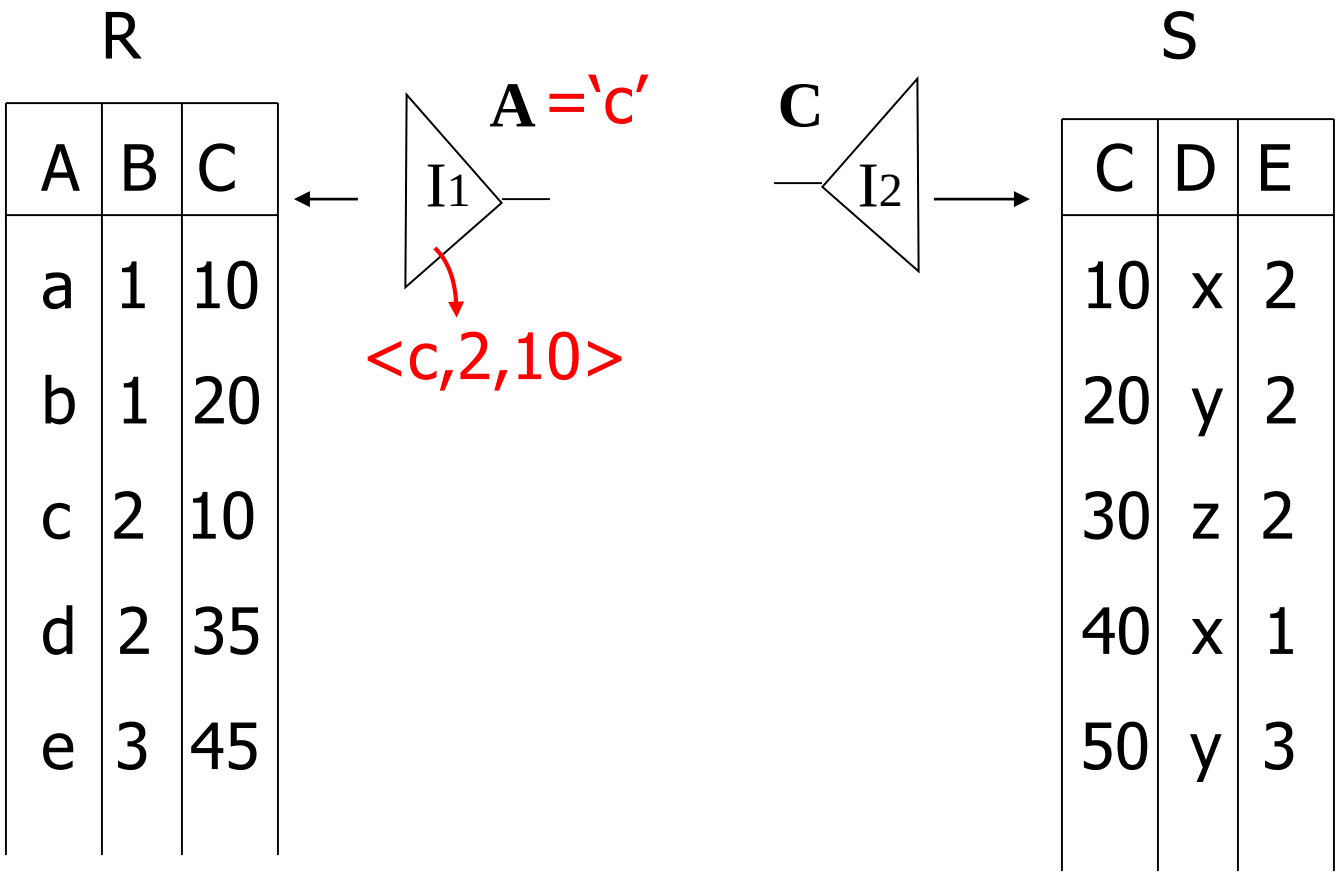
- Example of Plan 2:

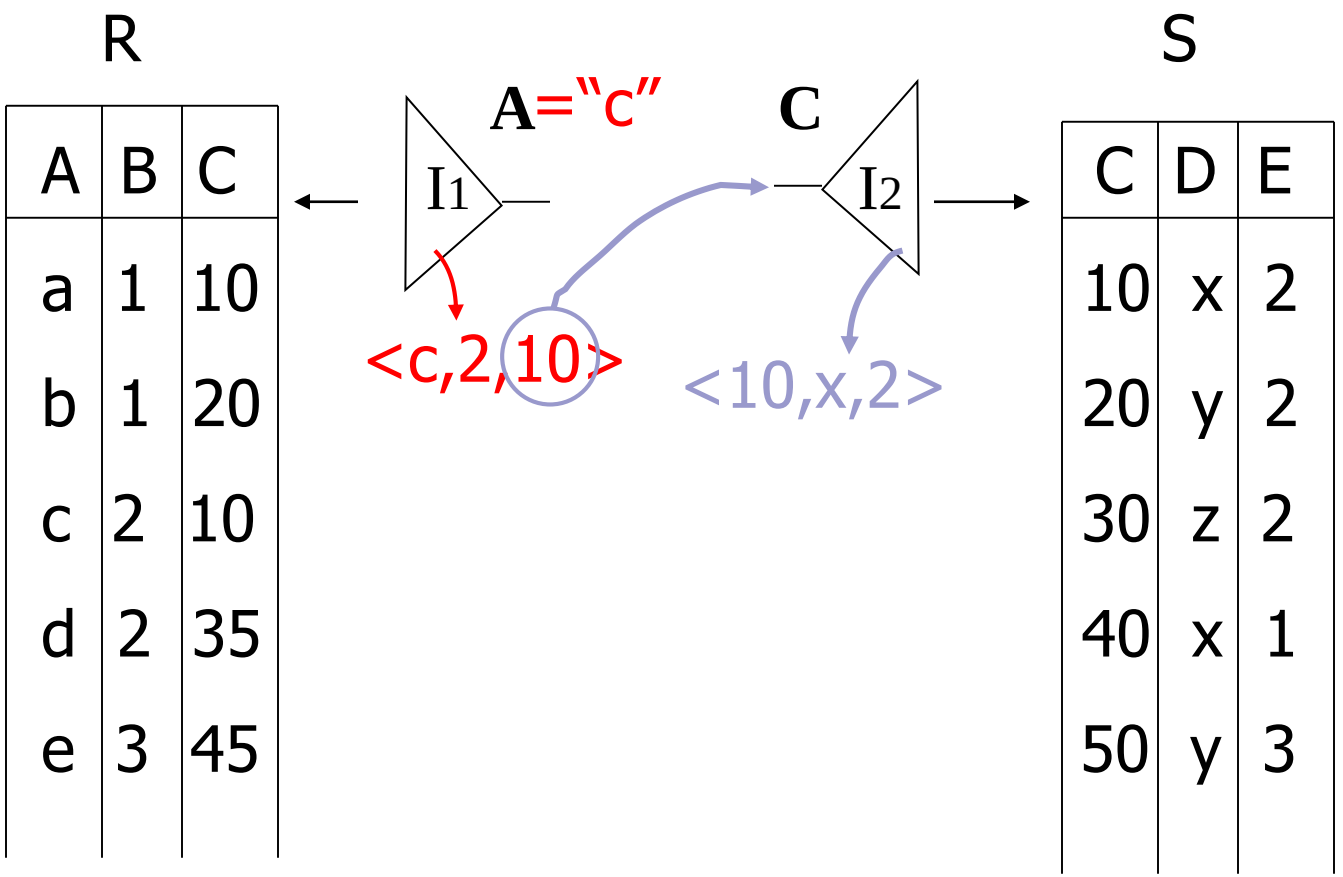


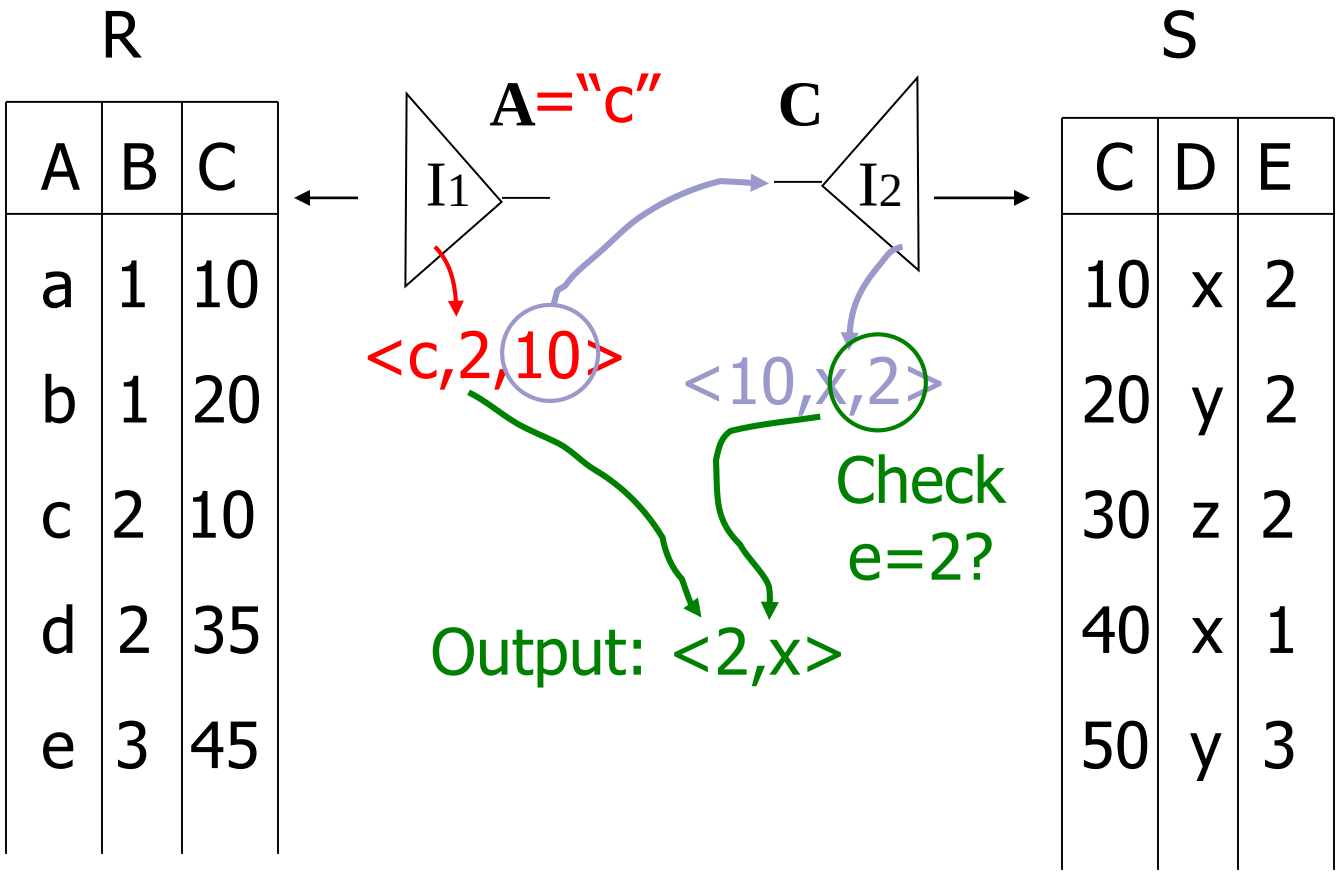
Describing Evaluation Plans

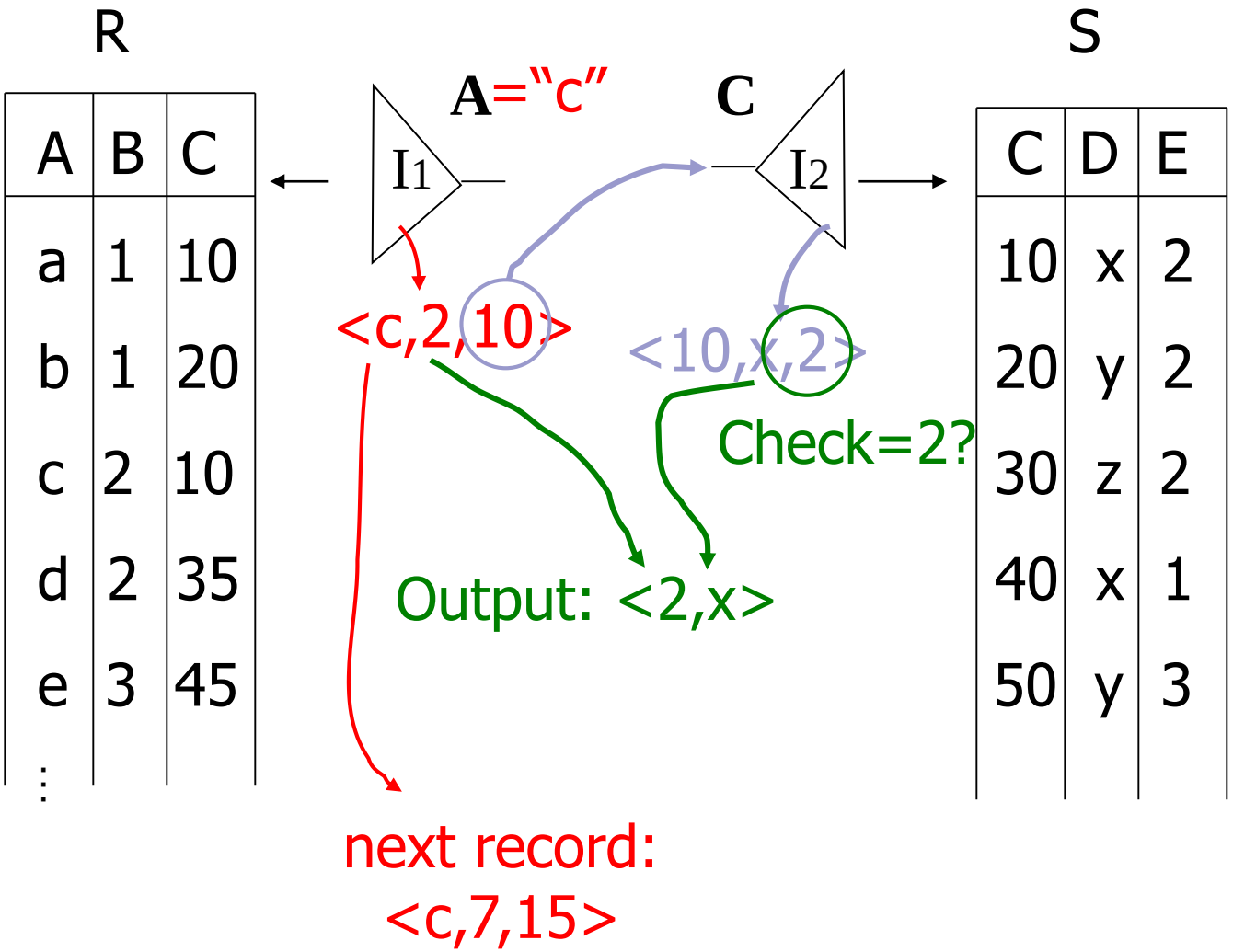
- Plan 3:
  - Assume an index on R.A and on S.C
  - Scan index R.A for looking up records of R satisfying R.A = “c”
    - For each record found, take value of R.C to scan index on S.C to get matching records of S
    - Filter out records of S, where S.E ≠ 2
  - Join the corresponding records of R and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


R
A
B
C
a 1 10
b 1 20
c 2 10
d 2 35
e 3 45
A
I_1
C
I_2
S
C
D
E
10 x 2
20 y 2
30 z 2
40 x 1
50 y 3

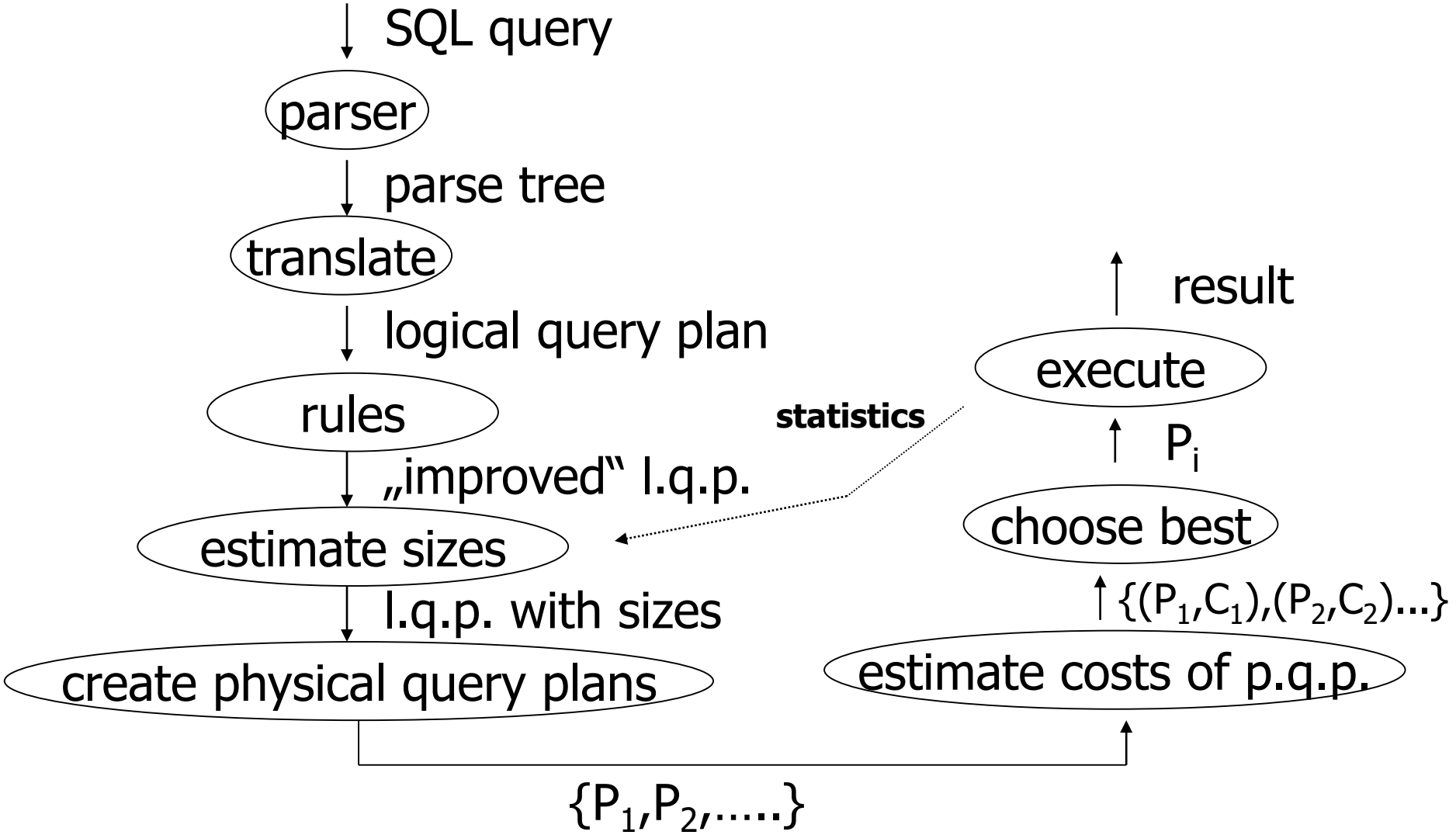

R
A
B
C
a
1
10
b
1
20
c
2
10
d
2
35
e
3
45
A ='c'
I1
<c,2,10>
C
I2
S
C
D
E
10
x
2
20
y
2
30
z
2
40
x
1
50
y
3


R
A
B
C
a
1
10
b
1
20
c
2
10
d
2
35
e
3
45
A="c"
I1
C
I2
<c,2,10>
<10,x,2>
S
C
D
E
10
x
2
20
y
2
30
z
2
40
x
1
50
y
3


R
A
B
C
a
1
10
b
1
20
c
2
10
d
2
35
e
3
45
A="c"
I1
C
I2
<c,2,10>
<10,x,2>
Check
e=2?
Output: <2,x>
S
C
D
E
10
x
2
20
y
2
30
z
2
40
x
1
50
y
3


R
A
B
C
a
1
10
b
1
20
c
2
10
d
2
35
e
3
45
A="c"
I1
C
I2
S
C
D
E
10
x
2
20
y
2
30
z
2
40
x
1
50
y
3
<c,2,10>
<10,x,2>
Check=2?
Output: <2,x>
next record:
<c,7,15>

Overview on Query Optimization

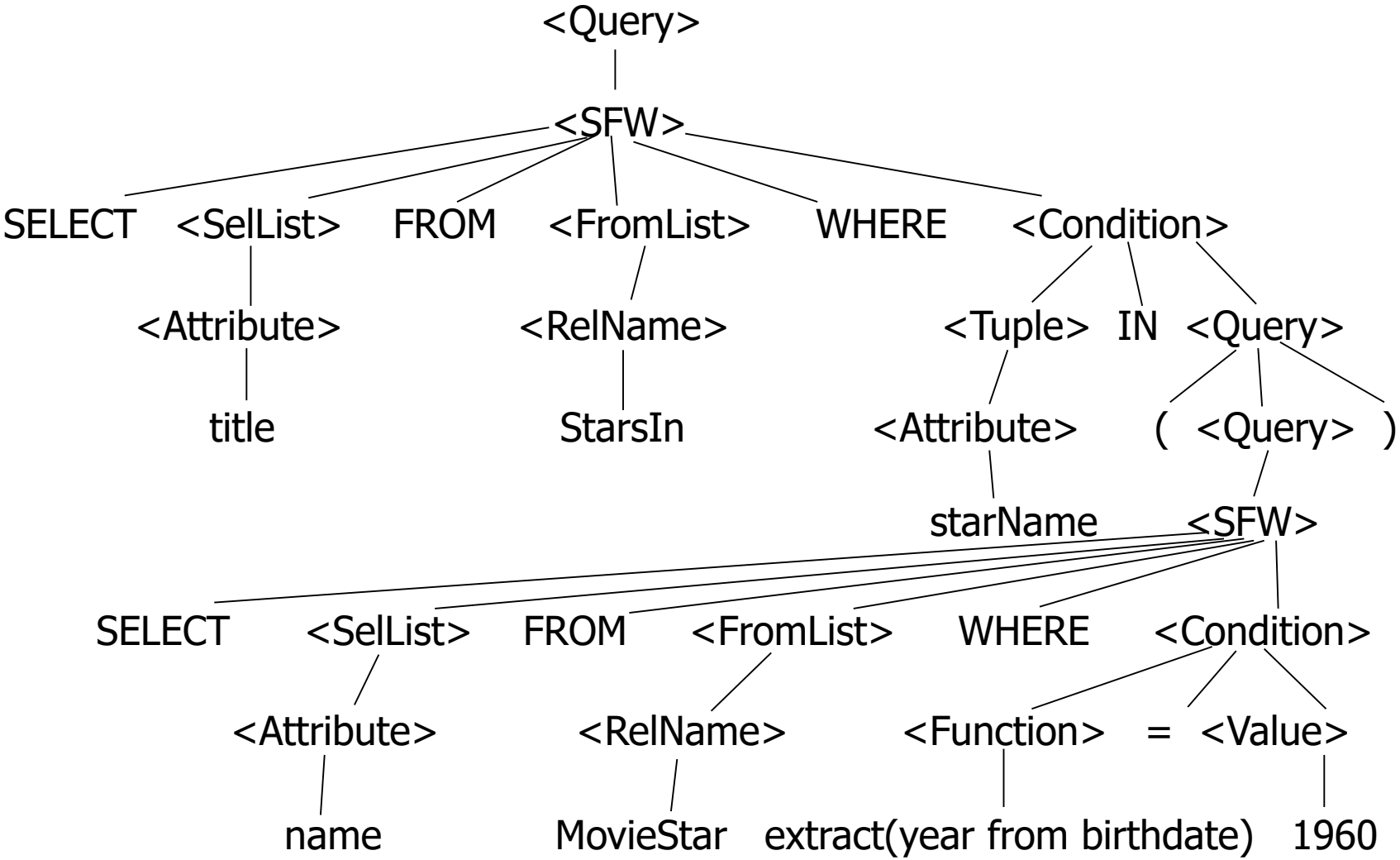


Example: SQL query

- Relations
  - StarsIn(title, year, starName)
  - MovieStar(name, birthdate)
- Query
  - Select movies with stars born in 1960:

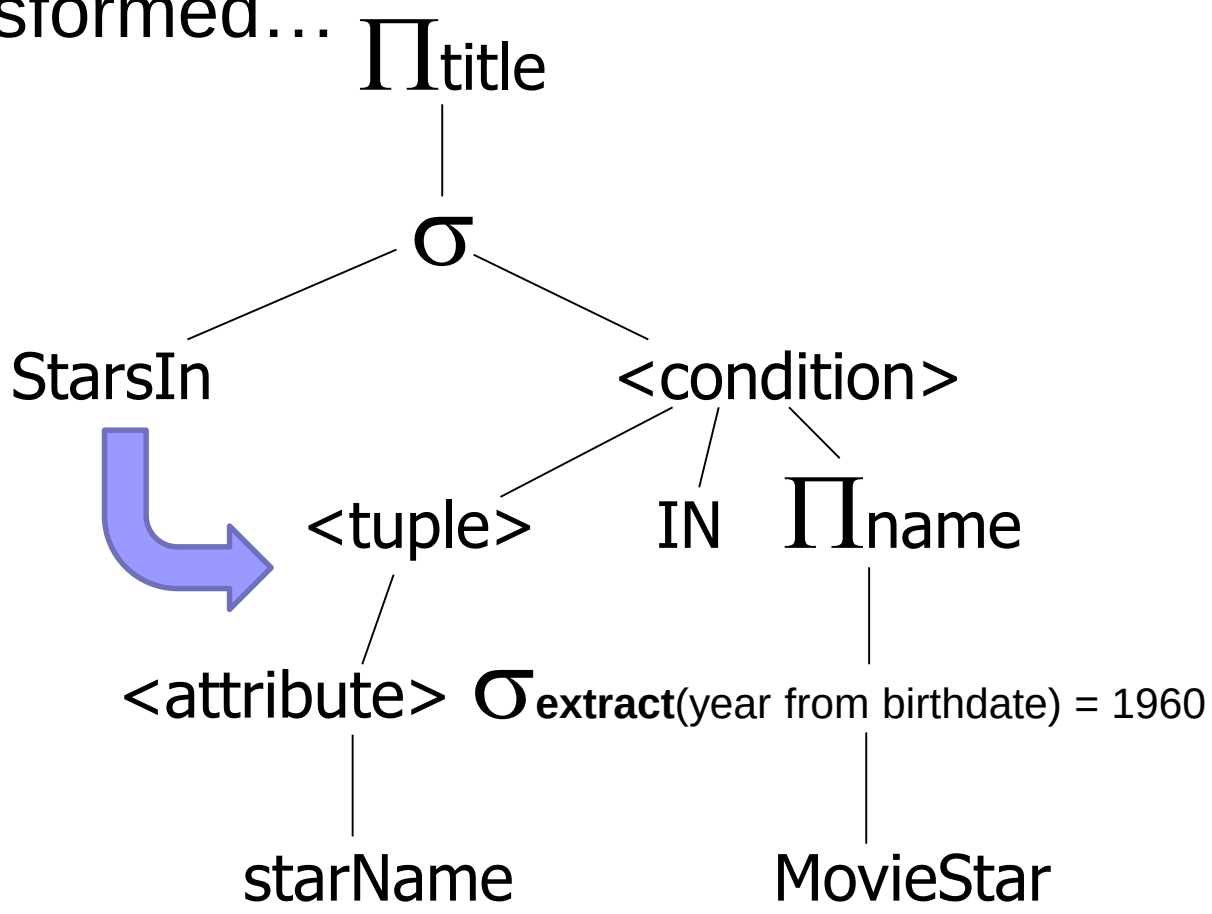
  - ```
    SELECT title
    FROM StarsIn
    WHERE starName IN (
                SELECT name
                FROM MovieStar
                WHERE extract(year from birthdate) = 1960
    );
    ```
    ```

Example: Parse Tree



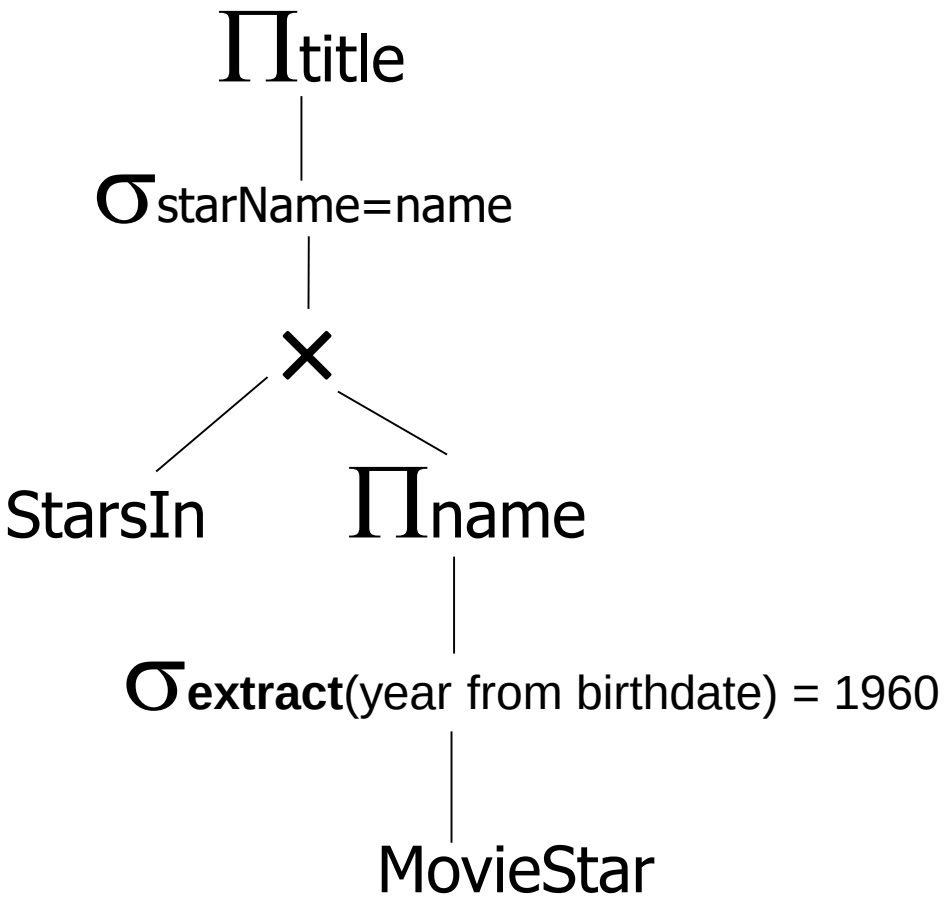
Example: Translation to Rel. Algebra

- Selection has two arguments
  - Must be transformed…
- IN operator
  - i.e., avoiding nested queries (sub-selects)



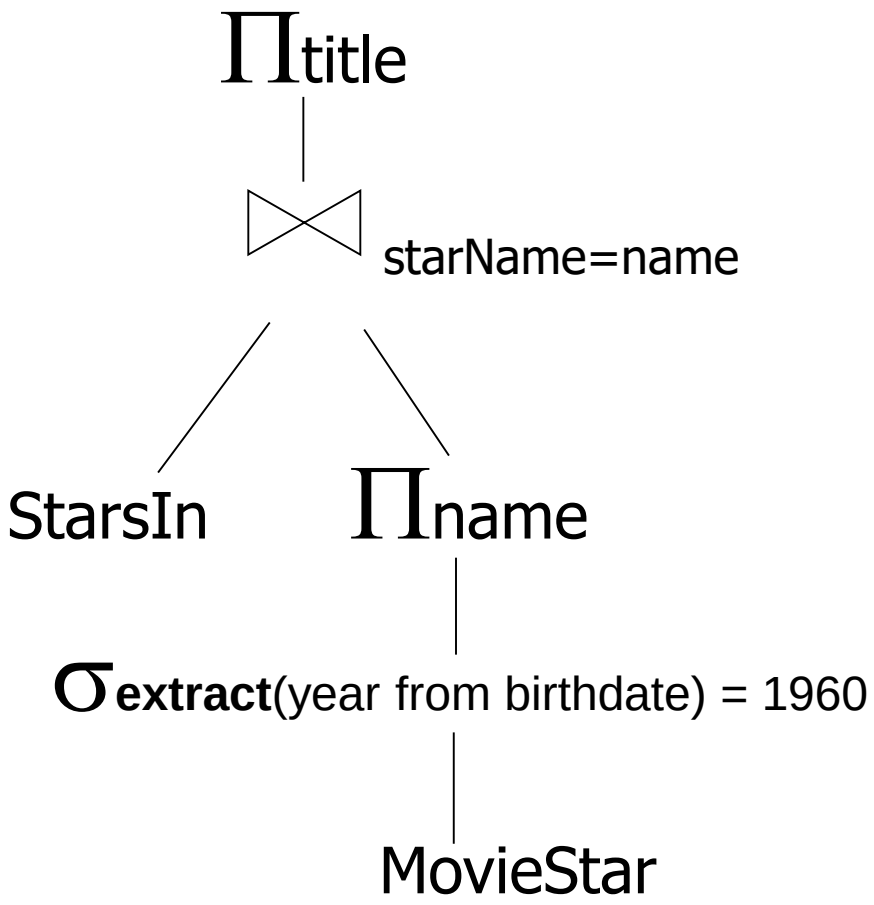
Example: Logical Query Plan

- IN operator replaced with product



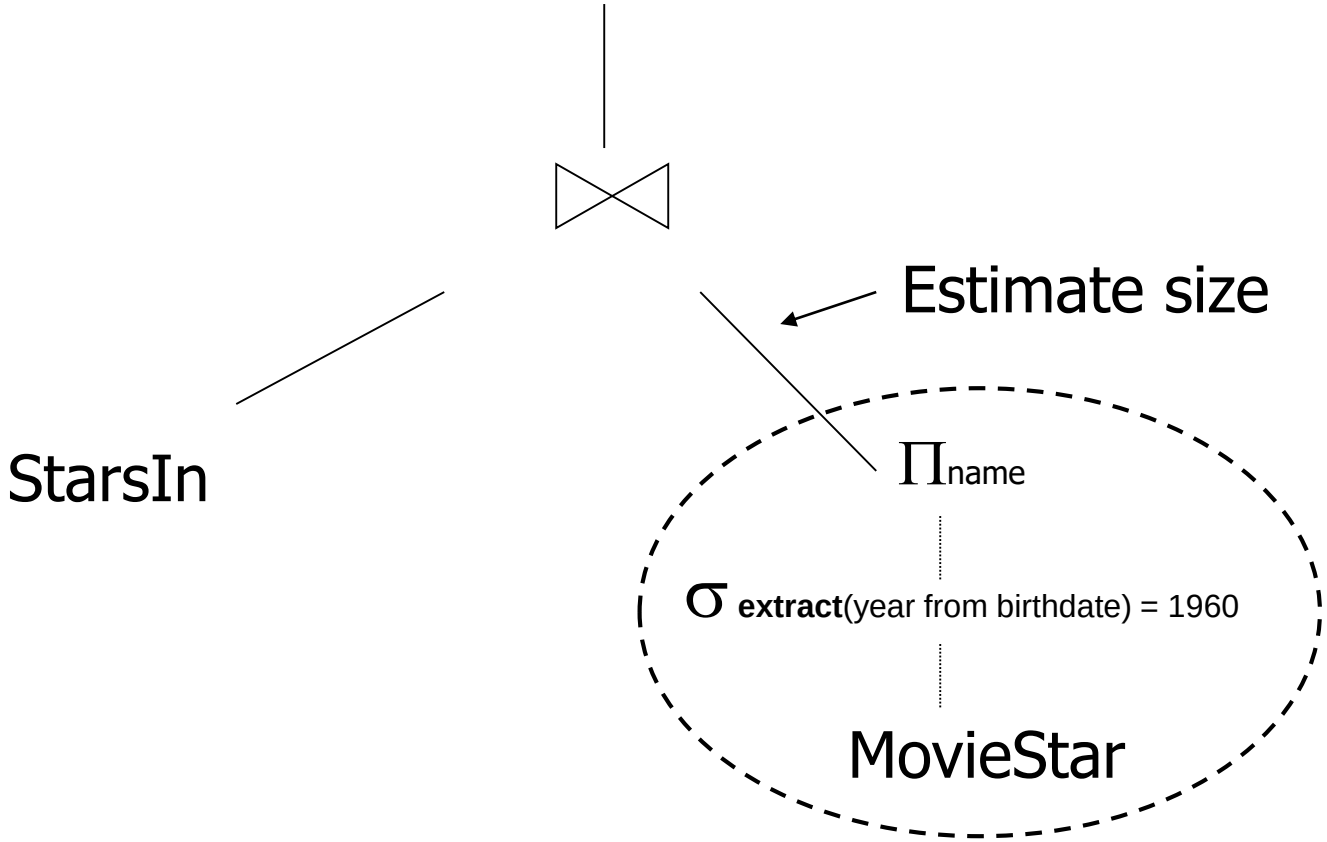
Example: Improving Logical Plan

- Substitution of product and selection
  - by join
- Next option
  - Push project to relation *StarsIn*?

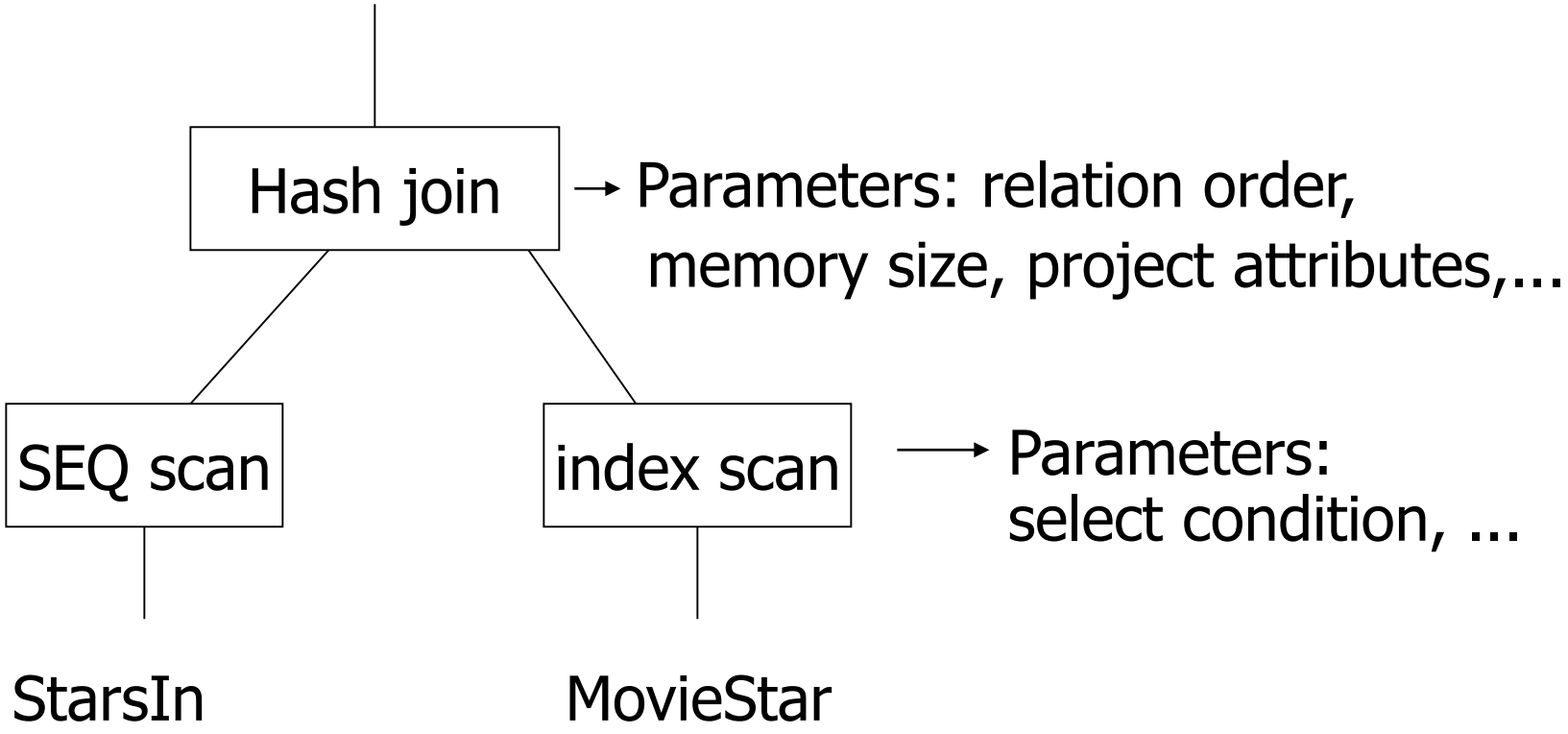


Example: Estimate Result Sizes

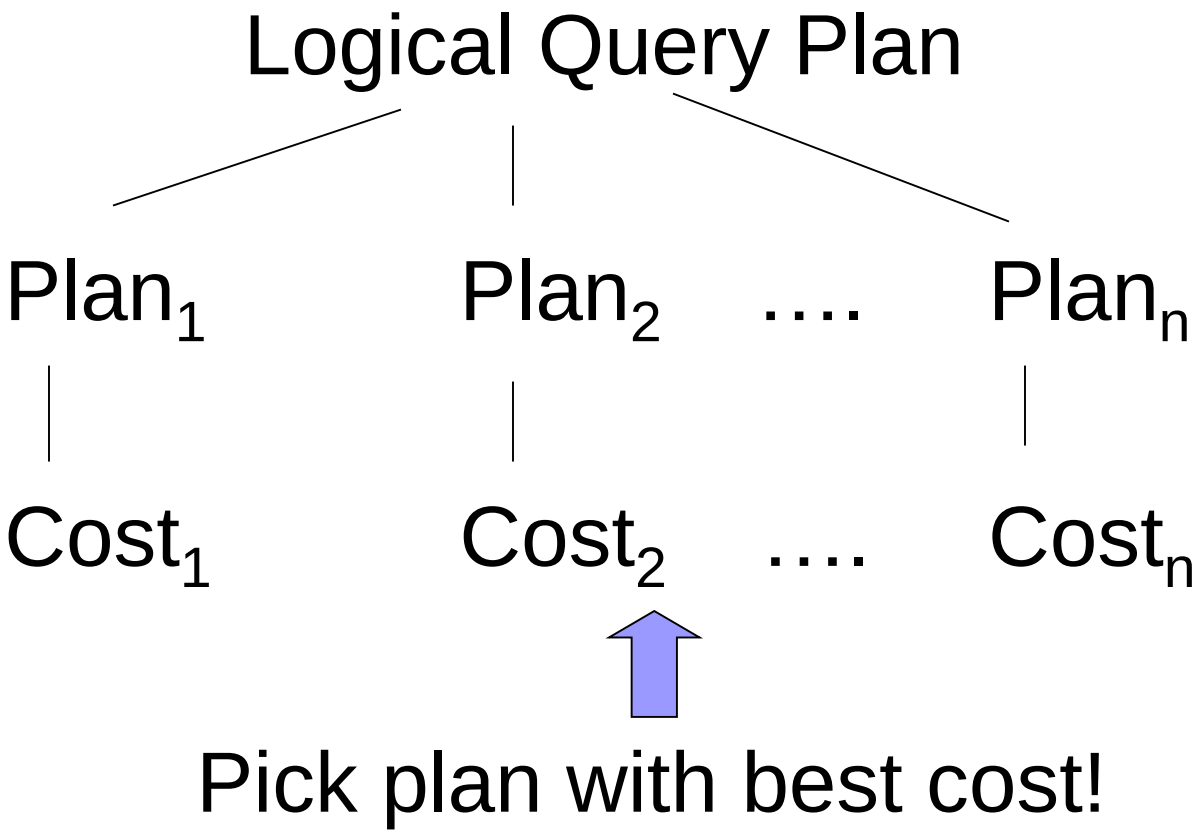
- Before generating physical plans
- Influence estimation of evaluation costs



Example: One Physical Plan



Example: Estimate Costs



Query Optimization

- Relational algebra level
- Detailed query plan level
  - Estimate costs
    - Without indexes
    - With indexes
  - Generate and compare plans

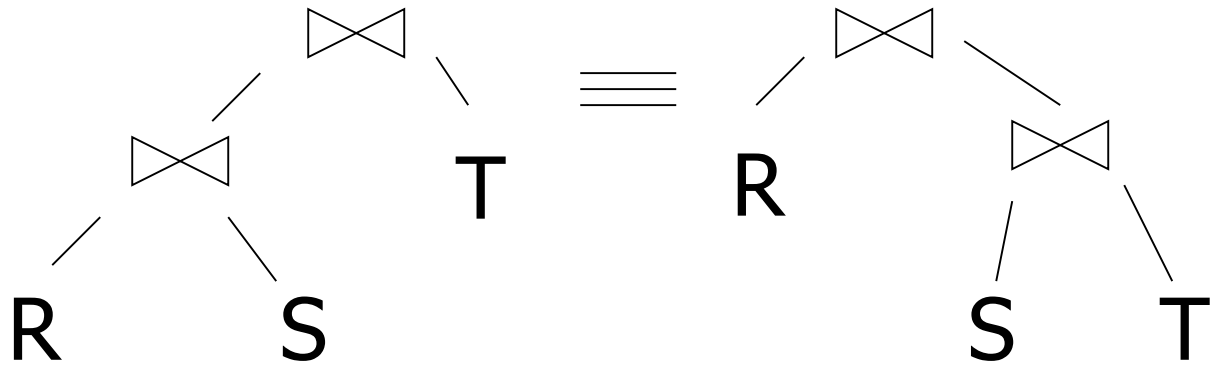
Relational Algebra Optimization

- Transformation rules
  - Must preserve equivalence
  - What are good transformations?

Transformation Rules

- 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
- Example
  - R = {a,a,b,b,b,c}
  - S = {b,b,c,c,d}
- $R \cap S = ?$
  - 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 SUM: Union

- Union two relations
  - $R \cup S$ SQL: UNION ALL
- Example
  - Representatives(id, year, party, …)
  - Senators(id, year, party, …)
  - $R = \pi_{year,party}(Senators)$ $S = \pi_{year,party}(Representatives)$

year	party
1997	ODS
2003	ČSSD
2007	SZ

year	party
1997	ODS
1998	KDU
1996	ČSSD

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)

    != (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 = 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)

$$\pi_x[\sigma_p(R)] = \pi_x(\sigma_p[\pi_{xz}(R)])$$

Transformation Rules

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

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

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

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

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

Transformation Rules

- Combining select and union

$$\sigma_p(R \cup_{sum} S) = \sigma_p(R) \cup_{sum} \sigma_p(S)$$

- 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* ?

Good Transformations

- Select early

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

- Project early

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

  - Example:
    - R(A,B,C,D,E,F,G,H,I,J) result={E}
    - Filter using P: (A=3) $\wedge$ (B="cat")

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

Good Transformations

$$\sigma_{p_1 \wedge p_2}(R) \rightarrow \sigma_{p_1}[\sigma_{p_2}(R)] \rightarrow \sigma_{p_2}[\sigma_{p_1}(R)]$$

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

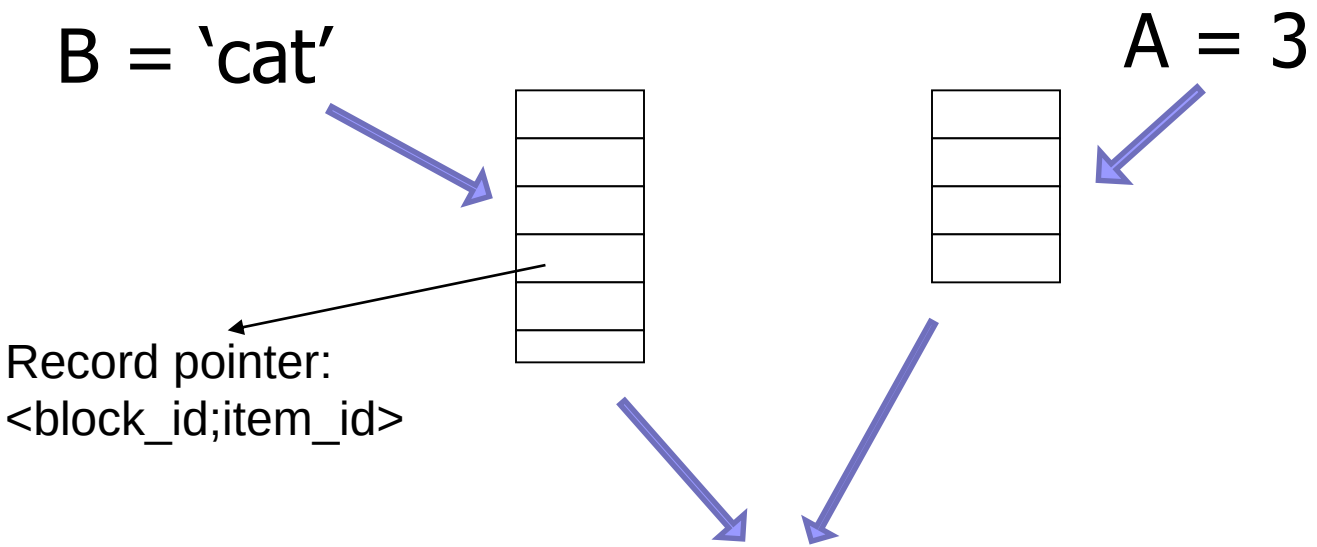
$$\sigma_{p_1 \vee p_2}(R) \rightarrow \sigma_{p_1}(R) \cup_{max} \sigma_{p_2}(R)$$

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

Good Transformations

- Assume indexes
  - On A and on B

$$\sigma_{(A=3) \wedge (B=\text{“cat”})}(R) = \sigma_{(A=3)}(R) \cap \sigma_{(B=\text{“cat”})}(R)$$



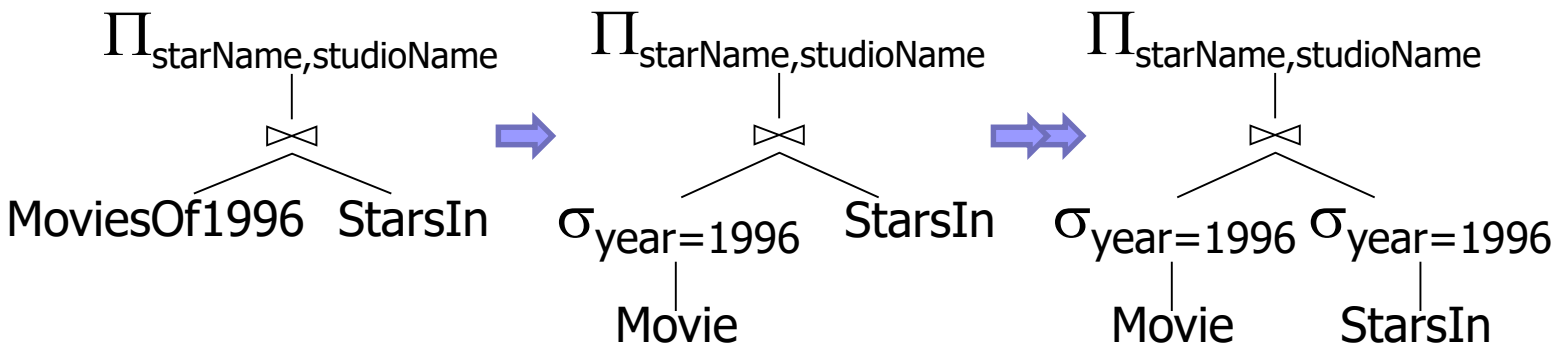
Intersect pointers to get pointers to matching tuples

Good Transformations

- Recommendations:
  - No transformation is always good.
  - Usually, good to
    - select early
    - project early
- Eliminate common sub-expressions
- Eliminate tuple duplicates

Good Transformations: Example

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



Query Evaluation: Overview

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

Estimating Cost of Query Plan

1. Estimate size of result
2. Estimate number of IOs

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

- Relation R
  - Attribute A – string, max. 20 B
    - S(R,A) = 3 ← average length
  - Attribute B – integer, 4 B
  - Attribute C – date, 8 B
  - Attribute D – string, 5 B
    - S(R,D) = 1
- Statistics
  - T(R) = 5 S(R) = 16
  - V(R,A) = 3 V(R,B) = 1
  - V(R,C) = 5 V(R,D) = 4

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

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 $W = \sigma_{Z=val}(R)$
  - $S(W) = S(R)$
  - $T(W) = ?$

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

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

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

V(R,A)=3
V(R,B)=1
V(R,C)=5
V(R,D)=4

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

Estimating Result Size

- Assumption of last estimate
  - Values are uniformly distributed over possible V(R,Z) values!
    - f(val) = 1 / V(R,Z)
    - $T(\sigma_{Z=val}(R)) = T(R) \cdot f(val)$
- Alternate assumption
  - Uniform distribution of values over domain with DOM(R,Z) values
    - # values in domain is denoted as DOM(R,Z)
    - f(val) = 1 / DOM(R,Z)

Estimating Result Size: Example

- Select $W = \sigma_{Z=val}(R)$
  - $T(W) = ?$
    - Podle 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 DOM(R,A)=10
V(R,B)=1 DOM(R,B)=10
V(R,C)=5 DOM(R,C)=10
V(R,D)=4 DOM(R,D)=10

Estimating Result Size

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

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

  - Alternate solution

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

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

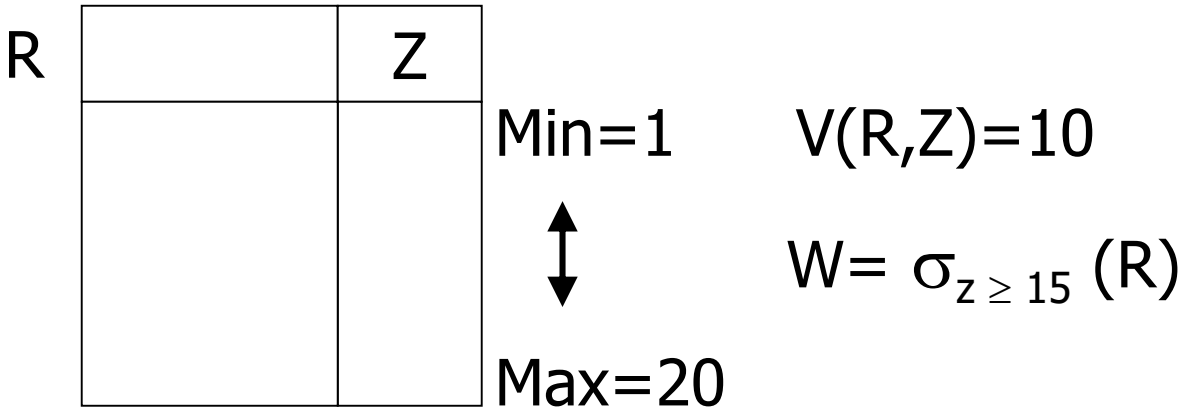
V(R,A)=3 DOM(R,A)=10
V(R,B)=1 DOM(R,B)=10
V(R,C)=5 DOM(R,C)=10
V(R,D)=4 DOM(R,D)=10

Estimating Result Size

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

Estimating Result Size

- Select – estimate values in range



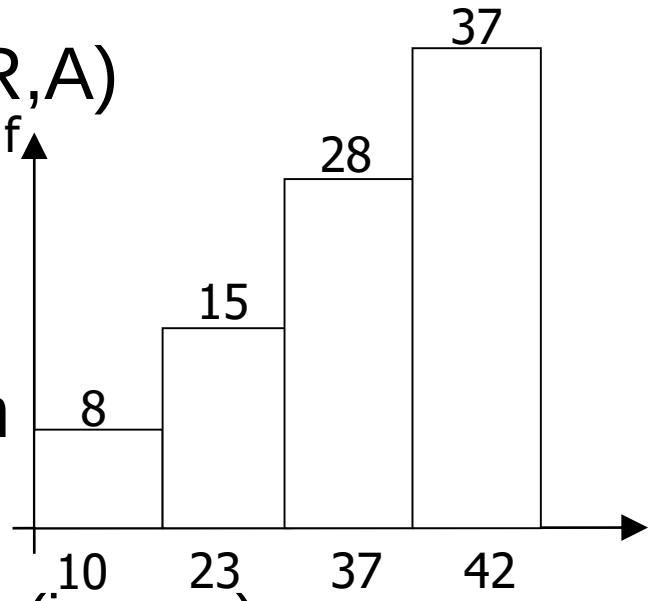
- Calculate fraction of unique values in range

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

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

Estimate # values: Histogram

- Histogram of attribute values
  - Replacing 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 $W = \sigma_{Z \neq val}(R)$
  - $T(W) = T(R) \cdot (1 - f(val)) = T(R) \cdot (1 - 1/V(R,Z))$

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

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

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

$R_1 \bowtie R_2$

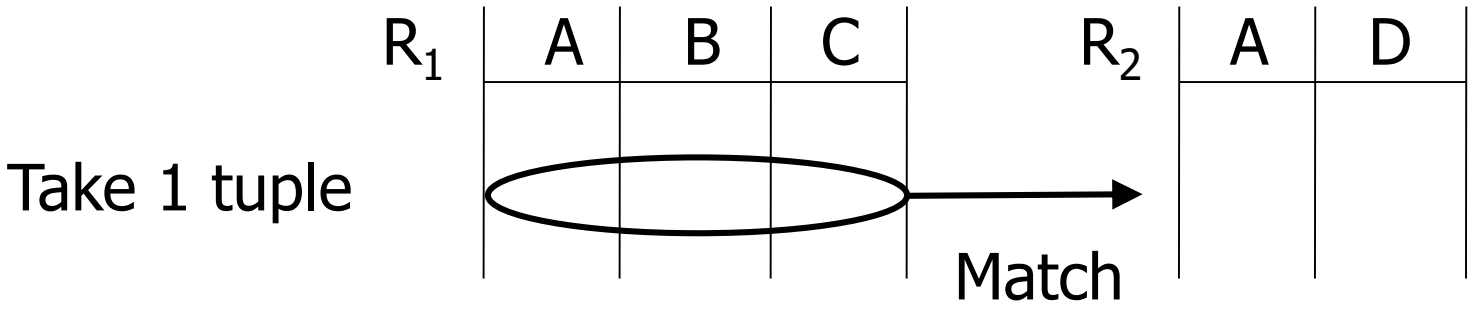
R_1	A	B	C

R_2	A	D

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

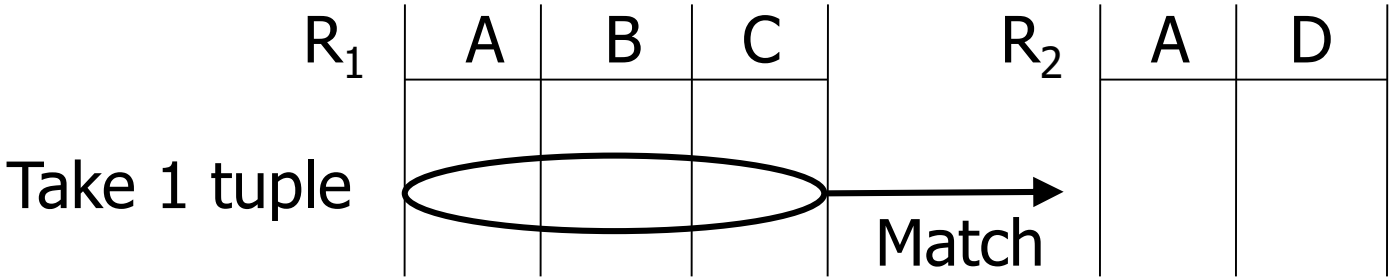
Estimating Result Size: Natural Join

- 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 $T(R_2)/DOM(R_2,A)$ recs.
- Result:

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

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 $n=T(R)$, $T(W) = n \cdot (1 - (1-m_1/n) \cdot (1-m_2/n))$
    - $m_1=T(R) / V(R,A)$ $\quad$ $m_2=T(R) / V(R,B)$

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

Estimating Result Size

- For complex expressions, intermediate results and their stats are needed
- Example
  - $W = [\underbrace{\sigma_{A=a}(R_1)}_{\text{denote as U}}] \bowtie R_2$
- $T(U) = T(R_1) / V(R_1,A)$ $\quad S(U) = S(R_1)$
  - Need V(U,*) to estimate costs of W!

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

Sth. in between

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
  - $V(U,*) = V(R,*)$, ie. $V(U,*) = T(U)$
- As a result, original V(R,*) can be used
  - $V(U,*) = \min \{ V(R,*), T(U) \}$

Estimate # values: Join

- $U = R_1(A,B) \bowtie R_2(A,C)$
- Estimates:
  - $V(U,A) = \min\{ V(R_1,A), V(R_2,A) \}$
  - $V(U,B) = V(R_1,B)$
    - ie. $\min \{ V(R_1,B), T(U) \}$
  - $V(U,C) = V(R_2,C)$

Estimate # values: Join

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

Estimate # values: Join

- Intermediate result
  - $U = R_1(A,B) \bowtie R_2(B,C)$
  - Estimates:
    - $T(U) = T(R_1) \cdot T(R_2) / \max\{ V(R_1,B), V(R_2,B) \} =$
      $= 1000 \cdot 2000 / 200 = 10\,000$
    - $V(U,A) = 50$
    - $V(U,B) = \min\{ V(R_1,B), V(R_2,B) \} = 100$
    - $V(U,C) = 300$

Estimate # values: Join

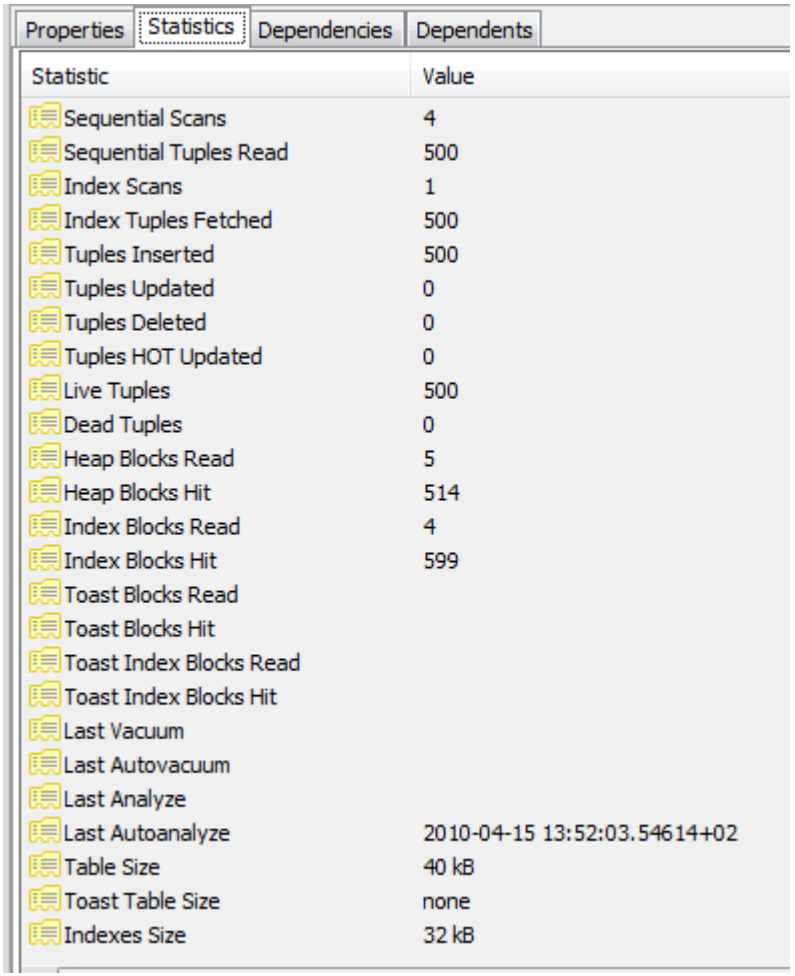
- Final result
  - $Z = U \bowtie R_3(C,D)$
    - U(A,B,C)
  - Estimates:
    - T(Z) = 10 000 · 3 000 / 300 = 100 000
    - V(Z,A) = 50
    - V(Z,B) = 100
    - V(Z,C) = 90
    - V(Z,D) = 500

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/13/view-pg-stats.html

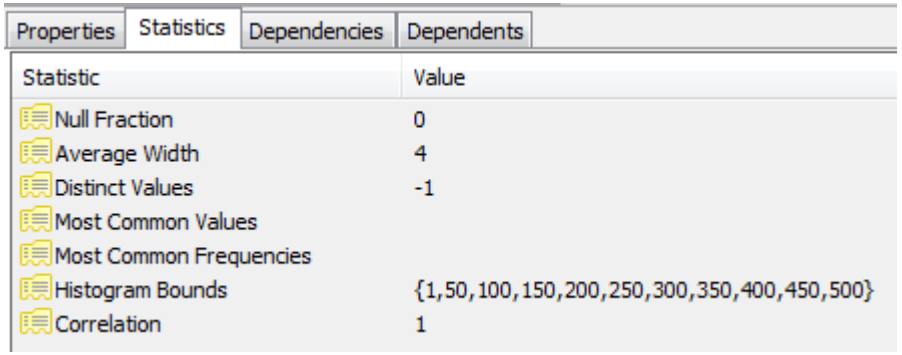
Example of Stats in PostgreSQL

- Relation hotel

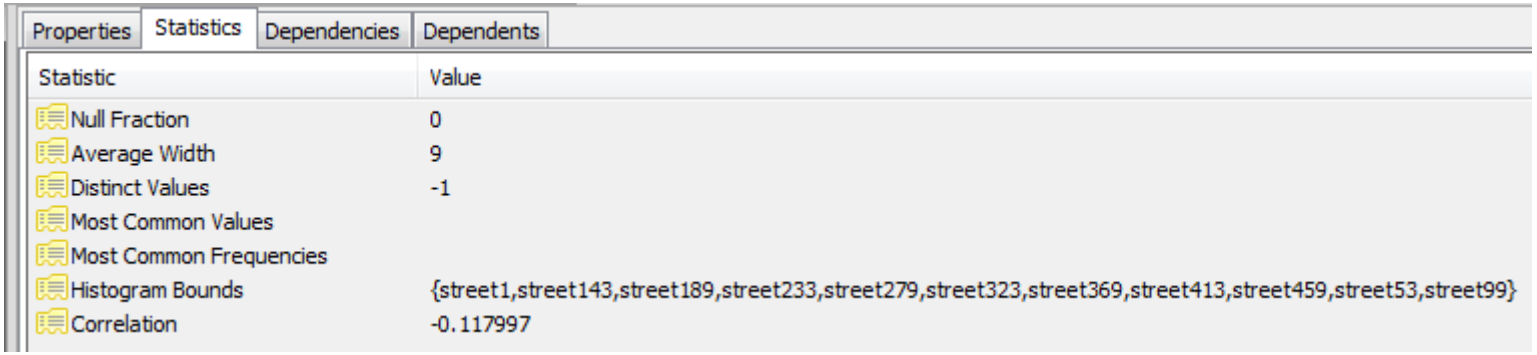


Example of Stats in PostgreSQL

- Attribute hotel.id

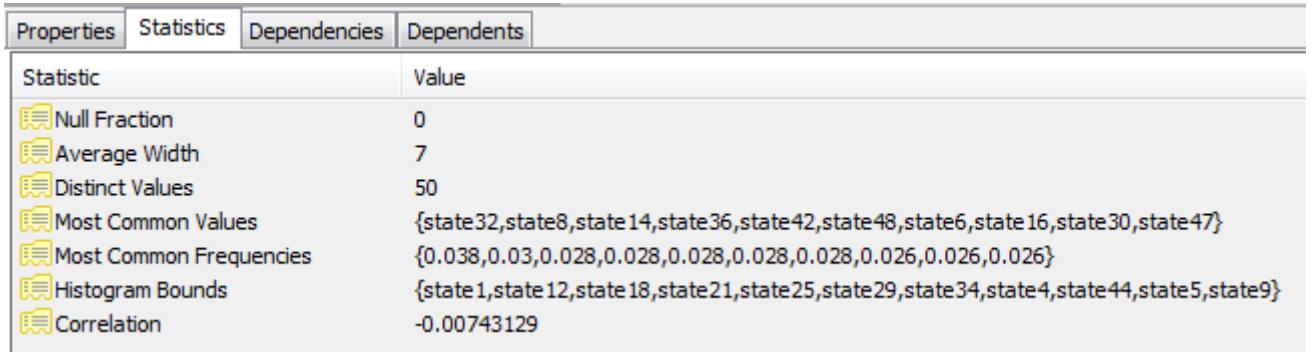


- Attribute hotel.name



Example of Stats in PostgreSQL

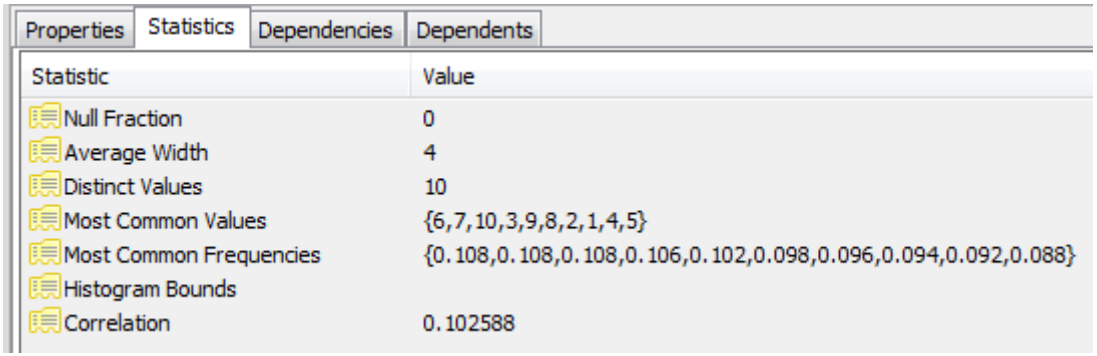
- Attribute hotel.state



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
    $\rightarrow$ 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
- $\rightarrow$ Do not update often

Statistics Update

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