Nguyen Tat Thanh Institute of International Education (NIIE)

DATABASE MANAGEMENT SYSTEMS

(Creadits 3)

MSc. Luong Tran Ngoc Khiet

May - 2021

Course content

Chap 1. Overview

Chap 2. Data storage management

Chap 3. Programming with Cursors

Chap 4. Query optimization

Chap 5. Continuous transaction processing

Chap 4.

Query optimization

MSc. Luong Tran Ngoc Khiet

NTT Institute of International Education (NIIE)

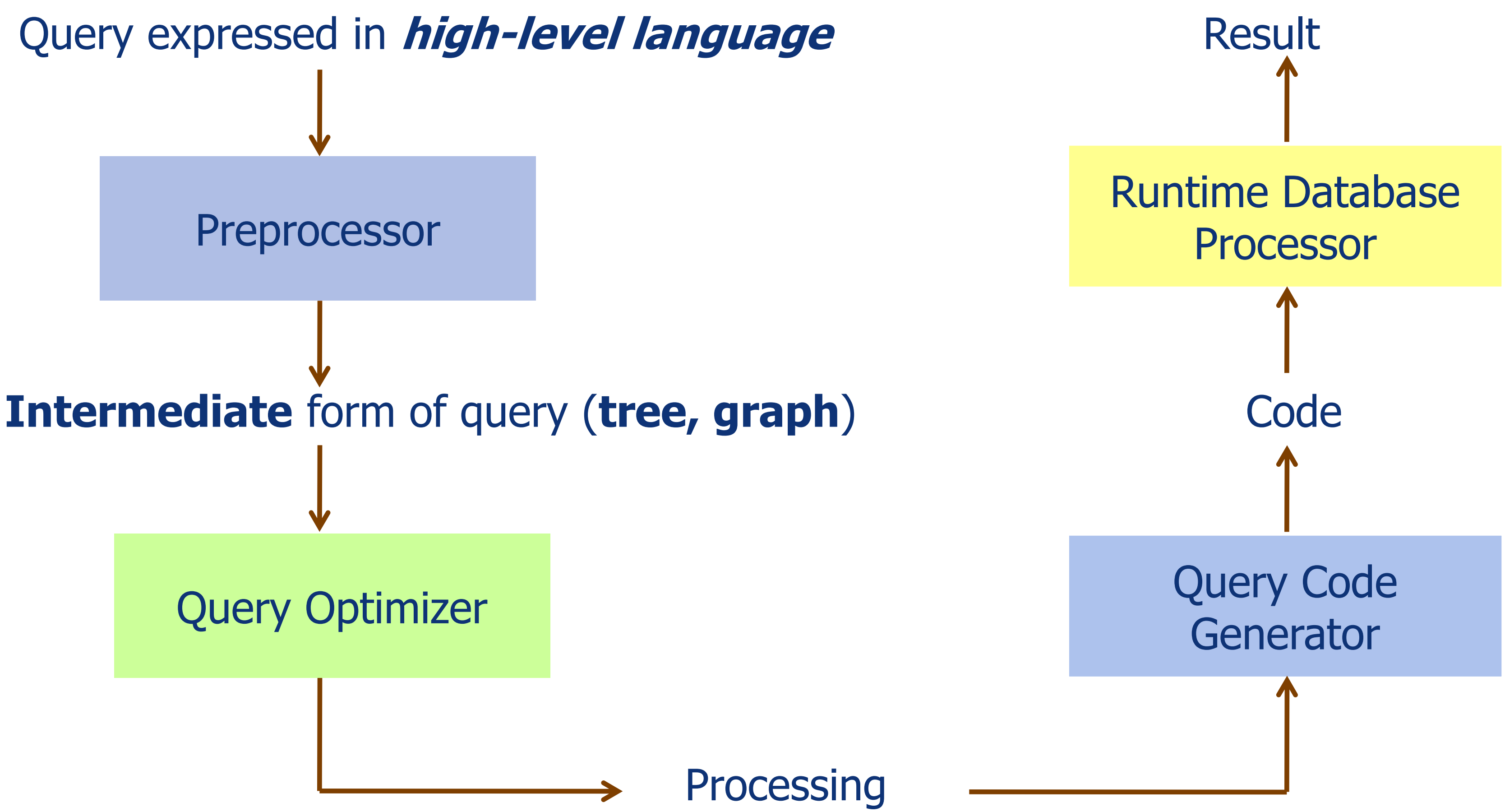
Target - Content

Understand the process of executing queries

Build queries effectively

1. Query execution process
2. Preprocess the query
3. Transform the query
4. Optimize the query

1. Query execution process



1. Query execution process

Preprocessor

Scanning: Identify keywords, attribute names, relationship names, etc.

Parsing: Check language syntax, Parse Tree representation

Validating: Check semantics: relationships, attributes, data types

1. Query execution process

Query Optimizer

Choose the appropriate execution strategy for query processing

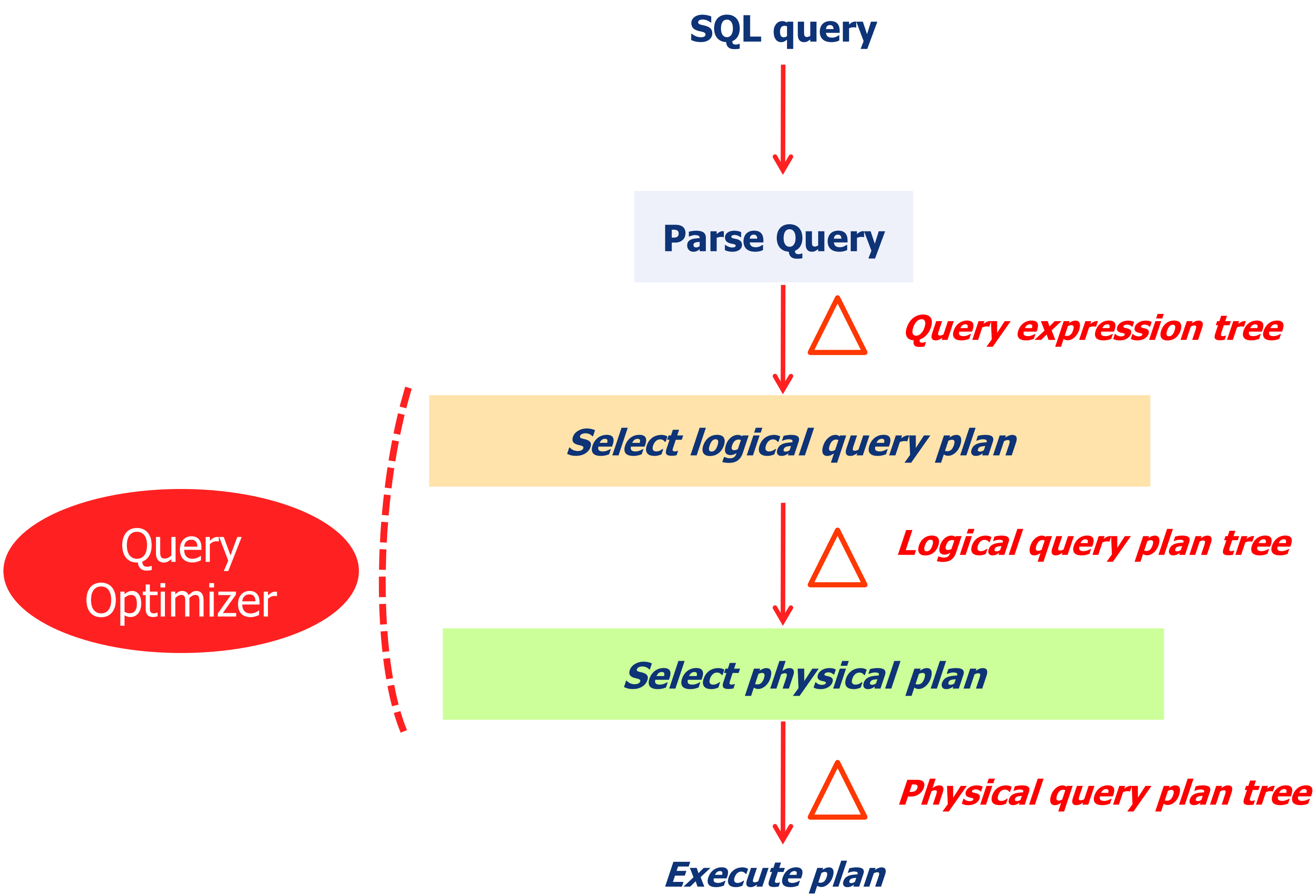
Query Code Generator

Generate code to execute the selected plan

Runtime Database Processor

Compile the query code to return the query results

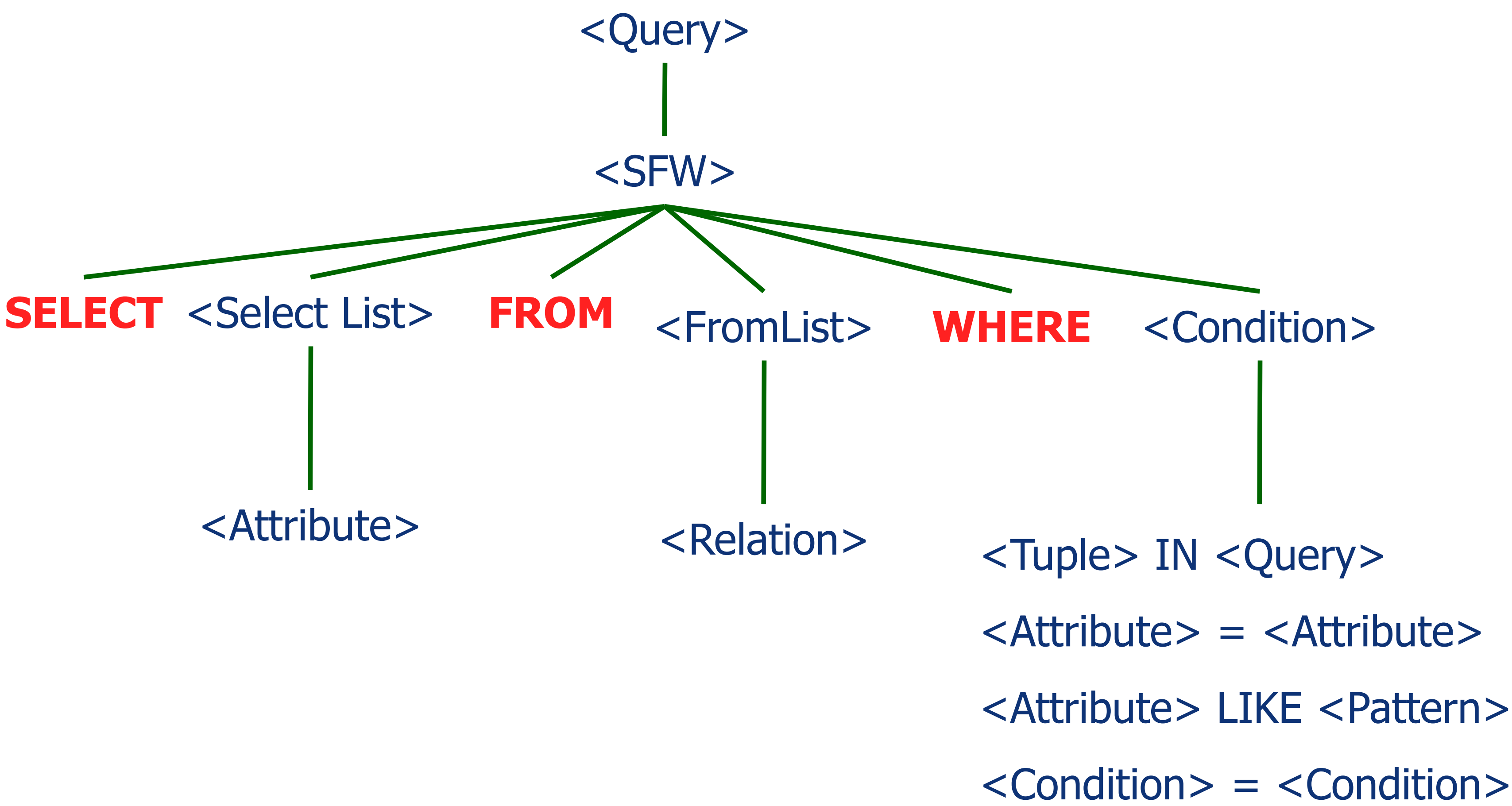
1. Query execution process



Content

1. Query execution process
2. **Preprocess the query**
3. Transform the query
4. Optimize the query

2. Preprocess the query



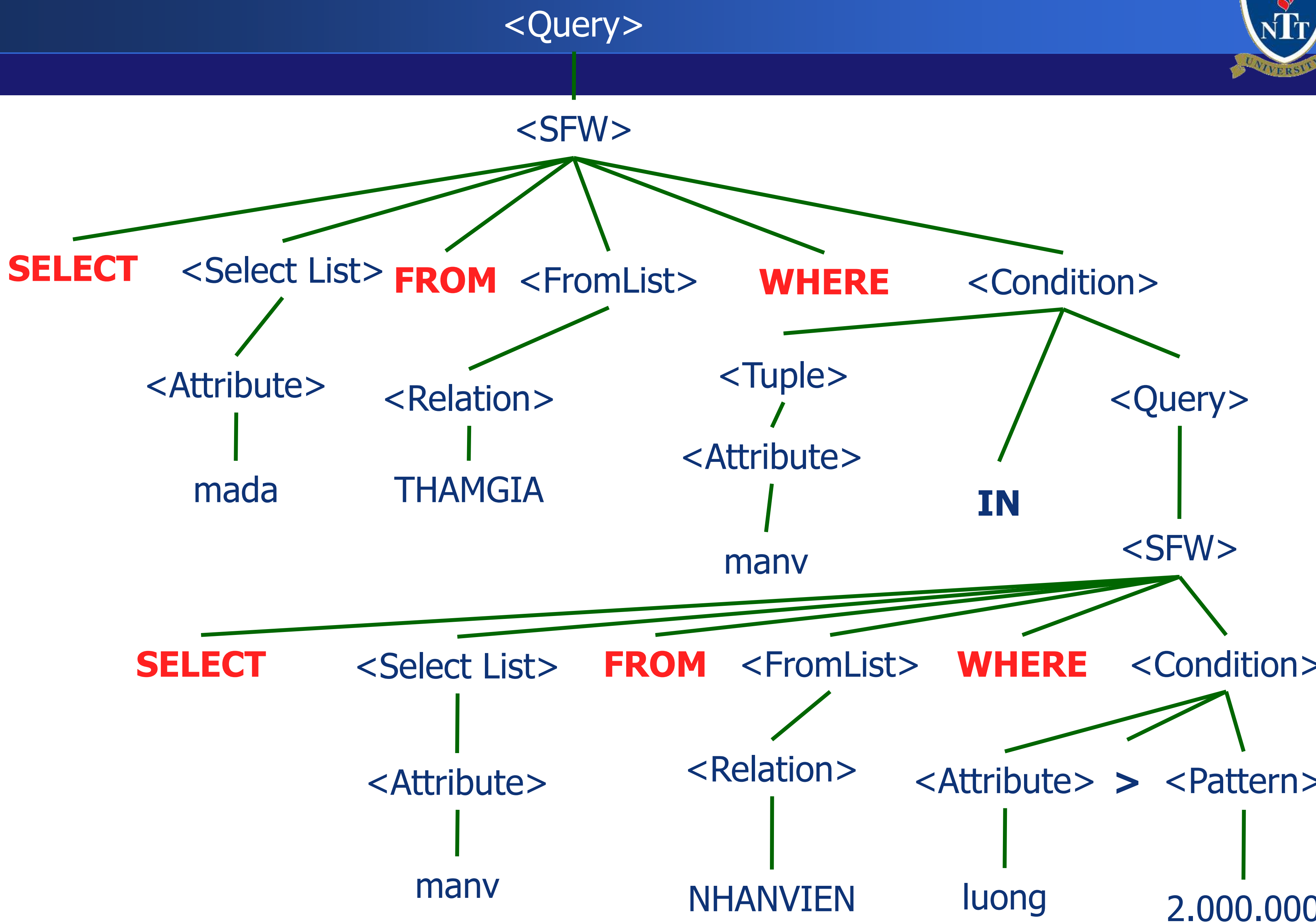
2. Preprocess the query

NHANVIEN (manv, tennv, ngaysinh, phai, luong)
THAMGIA (mada, manv, ngaybd, ngaykt)
================================

List project codes for which employees participating with salary >2,000,000

SELECT mada FROM THAMGIA
WHERE manv IN (
SELECT manv FROM NHANVIEN
WHERE luong >'2.000.000')

Query expression tree


<Query>
<SFW>
SELECT
<Select List>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<Relation>
<Tuple>
<Query>
mada
THAMGIA
<Attribute>
IN
manv
<SFW>
SELECT
<Select List>
FROM
<FromList>
WHERE
<Condition>
<Attribute>
<Relation>
<Attribute>
>
<Pattern>
manv
NHANVIEN
luong
2.000.000

Content

1. Query execution process
2. Preprocess the query
3. **Transform the query**

   3.1 Convert from SQL to Relational Algebra

   3.2 Equivalent transformation rules in Relational Algebra
4. Optimize the query

3.1 Convert from SQL to Relational Algebra

- Query block: SELECT-FROM-WHERE-GROUP BY-HAVING unit query block used to switch to Relational Algebra

- Nested query: split into command blocks into unit query blocks (query blocks)

3.1 Convert from SQL to Relational Algebra

```
SELECT  honv, tennv
FROM    NHANVIEN
WHERE   luong> (SELECT   MAX (luong)
                FROM     NHANVIEN
                WHERE    maphong = 5);
```

SELECT	honv, tennv
FROM	NHANVIEN
WHERE	luong > C

$\pi_{honv,\ tennv}(\sigma_{luong>C}(NHANVIEN))$

SELECT	MAX (luong)
FROM	NHANVIEN
WHERE	maphong = 5

$\mathcal{F}_{MAX\ luong}(\sigma_{maphong=5}(NHANVIEN))$

Content

1. Query execution process
2. Preprocess the query
3. Transform the query

   3.1 Convert from SQL to Relational Algebra

   3.2 Equivalent transformation rules in Relational Algebra

4. Optimize the query

3.2 Equivalent transformation rules – R1

R1: Handling AND operators in conditions

$$\sigma_{c1 AND c2 \ldots AND cn}(R) \equiv \sigma_{c1}(\sigma_{c2}(\ldots\sigma_{cn}(R))\ldots)$$

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

$\sigma_{\text{maphong = 'KT' AND phai = 'NAM'}}$ **(NHANVIEN)**

$\equiv$

$\sigma_{\text{maphong = 'KT'}}(\sigma_{\text{phai = 'NAM'}}$ **(NHANVIEN))**

3.2 Equivalent transformation rules – R2

R2: Change the order of selections

$$\sigma_{c1}(\sigma_{c2}(R)) \equiv \sigma_{c2}(\sigma_{c1}(R))$$

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

$$\sigma_{\text{maphong = 'KT'}}(\sigma_{\text{phai = 'NAM'}}(\text{NHANVIEN}))$$

$$\equiv$$

$$\sigma_{\text{phai = 'NAM'}}(\sigma_{\text{maphong = 'KT'}}(\text{NHANVIEN}))$$

R3: Handling projections

$$\pi_{<DS1>}\left(\pi_{<DS2>}\left(\ldots\pi_{<DSn>}(R)\right)\ldots\right) \equiv \pi_{<DS1>}(R)$$

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

$\pi_{\text{manv, honv, tennv}}(\pi_{\text{manv, honv, tennv, ngaysinh}}(\text{NHANVIEN}))$

$\equiv$

$\pi_{\text{manv, honv, tennv}}(\text{NHANVIEN})$

3.2 Equivalent transformation rules – R4

R4: Change the order of selections and projections

$$\pi_{A1,A2,\ldots,An}\left(\sigma_c\left(R\right)\right) \equiv \sigma_c\left(\pi_{A1,A2,\ldots,An}\left(R\right)\right)$$

Nếu như c $\subset$ [A1…An]

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

$\pi_{\text{manv, honv, tennv, phai}}(\sigma_{\text{phai = 'NAM'}}(\text{NHANVIEN}))$

$\equiv$

$\sigma_{\text{phai= 'NAM'}}(\pi_{\text{manv, honv, tennv, phai}}(\text{NHANVIEN}))$

3.2 Equivalent transformation rules – R5

R5: Commutative properties of natural join and Cartesian join

$$\left(R \bowtie_C S\right) = (S \bowtie_C R) \qquad \left(R \times S\right) = (S \times R)$$

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

PHONGBAN (<u>maphong</u>, tenphong, maql)

(NHANVIEN $\bowtie_{\text{NV.maphong= PB.maphong}}$ PHONGBAN)

≡

(PHONGBAN $\bowtie_{\text{NV.maphong= PB.maphong}}$ NHANVIEN)

3.2 Equivalent transformation rules – R6a

R6a: Change the order between selection and union

$$\sigma_c(R \bowtie S) \equiv (\sigma_c(R)) \bowtie S$$

Nếu như c $\subset$ R (hay c $\subset$ S)

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

PHONGBAN (<u>maphong</u>, tenphong, maql)

$$\sigma_{phai = \text{'NAM'}}(\text{NHANVIEN} \bowtie \text{PHONGBAN})$$

$$\equiv$$

$$(\sigma_{phai = \text{'NAM'}}(\text{NHANVIEN})) \bowtie \text{PHONGBAN}$$

3.2 Equivalent transformation rules – R6b

R6b: Distribution between selection and combination

$$\sigma_c(R \bowtie S) \equiv (\sigma_{c1}(R)) \bowtie (\sigma_{c_2}(S))$$

Nếu c = c1 and c2, (c1 $\in$ R và c2 $\in$ S)

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

PHONGBAN (<u>maphong</u>, tenphong, maql)

$\sigma_{\text{phai= 'NAM' AND tenphong= 'Kế toán'}}$(NHANVIEN $\bowtie$ PHONGBAN)

$\equiv$

($\sigma_{\text{phai = 'NAM'}}$ (NHANVIEN)) $\bowtie$ ($\sigma_{\text{tenphong= 'Kế toán'}}$ (PHONGBAN))

3.2 Equivalent transformation rules – R7a

R7a: Distribution between projection and conjunction

$$\prod_L (R \bowtie_C S) \equiv (\prod_{A_1, A_2, A_3, \ldots A_N}(R)) \bowtie_C (\prod_{B_1, B_2, B_3, \ldots B_M}(S))$$

$L = \{A_1,\ldots,A_N,B_1,\ldots,B_M\}$; $R(A_1,\ldots,A_N)$; $S(B_1,\ldots,B_M)$ Với $c \subset L$

NHANVIEN (<u>manv</u>, honv, tennv, ngaysinh, phai, luong, maphong)

PHONGBAN (<u>maphong</u>, tenphong, maql)

$$\pi_{manv,tennv,maphong,tenphong}(NHANVIEN \bowtie_{NV.maphong=PB.maphong} PHONGBAN)$$

$$\equiv$$

$$(\pi_{manv,\ honv,\ maphong}(NHANVIEN)) \bowtie_{NV.maphong=PB.maphong} (\pi_{tenphong,\ maphong}(PHONGBAN))$$

3.2 Equivalent transformation rules – R7b

R7b: Distribution between projection and conjunction

$$\prod_L (R \bowtie_C S) \equiv (\prod_{A_1, A_2, A_3, \ldots A_N, A_{N+1} A_{N+2} \ldots A_{N+K}} (R)) \bowtie_C (\prod_{B_1, B_2, B_3, \ldots B_M B_{M+1} B_{M+2} \ldots B_{M+P}} (S))$$

Với c $\not\subset$ L, R($A_1,\ldots,A_N, A_{N+1},\ldots A_{N+K}$) S($B_1,\ldots,B_M, B_{M+1},\ldots,B_{M+P}$)

NHANVIEN (manv, honv, tennv, ngaysinh, phai, luong, maphong)

PHONGBAN (maphong, tenphong, maql)

$$\pi_{\text{manv, tennv, tenphong}} (\text{NHANVIEN} \bowtie_{\text{NV.maphong=PB.maphong}} \text{PHONGBAN})$$

$$\equiv$$

$$(\pi_{\text{manv, tennv, maphong}}(\text{NHANVIEN})) \bowtie_{\text{NV.maphong=PB.maphong}} (\pi_{\text{tenphong,maphong}}(\text{PHONGBAN}))$$

3.2 Equivalent transformation rules – R8

R8: Commutation of union and intersection

$$R \cup S \equiv S \cup R$$

$$R \cap S \equiv S \cap R$$

3.2 Equivalent transformation rules – R9

R9: Combination of nature join, Cartesian join, union and intersection

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

Which θ ***is one of the operations*** $\bowtie$, X, $\cap$, $\cup$

R10: Distribution of selection over operations

$$\sigma_c(\mathrm{R}\,\theta\,\mathrm{S}) = (\sigma_{\mathrm{c}}(\mathrm{R}))\,\theta\,(\sigma_{\mathrm{c}}(\mathrm{S}))$$

Which θ ***is one of the operations*** $\cap$, $\cup$, $-$

3.2 Equivalent transformation rules – R11

R11: Distribution of projection with respect to mathematical operations

Which θ ***is one of the operations*** $\cap, \cup, -$

$$\prod_L (\mathrm{R}\ \theta\ \mathrm{S}) = (\prod_{\mathrm{L}} (\mathrm{R}))\ \theta\ (\prod_{\mathrm{L}} (\mathrm{S}))$$

R12: Convert math operations (σ, ×) become a join

$$\sigma_c(R \times S) = R \bowtie_c S$$

De Morgan:

c ≡ NOT (c1 AND c2) ≡ NOT (c1) OR NOT (c2)

c ≡ NOT (c1 OR c2) ≡ NOT (c1) AND NOT (c2)

Content

1. Query execution process
2. Preprocess the query
3. Transform the query
4. **Optimize the query**
   4.1 Heuristic algorithm
   4.2 Cost estimation

4.1 Heuristic algorithm

1. Apply **R1**, splits multiple selections into a sequence of selections.
2. Apply **R2, R4, R6 and R10**, to push the selection as deep as possible
3. Apply **R9** to reorganize the syntax tree so that the selection is made most profitable (least select) → heuristic
4. Combine the Cartesian join and the appropriate projections that follow
5. Apply **R3, R4, R7 and R11** to push the projection down as deep as possible (new projection may arise)
6. Focus on selections
7. Apply **R3** to remove useless projections

4.1 Heuristic algorithm

List full name NHANVIEN born after 1960 and working on project 'ABC'

SQL

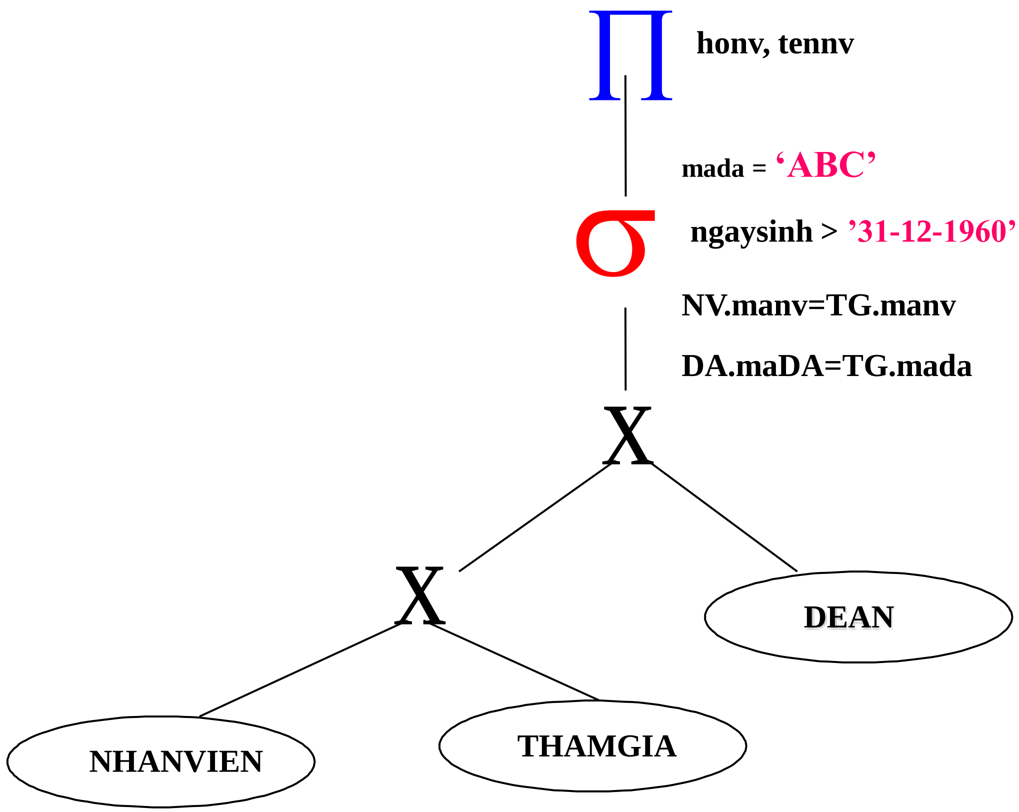
```
SELECT honv, tennv
FROM NHANVIEN NV, DEAN DA, THAMGIA TG
WHERE  mada='ABC' AND NV.manv=TG.manv AND
DA.mada=TG.mada AND ngaysinh> '31-12-1960'
```

Relational algebra

$$\Pi_{honv,\ tennv}(\sigma_{mada\ =\ 'ABC'\ \wedge\ ngaysinh\ >\ '31-12-1960'\ \wedge\ NV.manv=TG.manv\ \wedge\ DA.mada=TG.mada}(NHANVIEN \times DEAN \times THAMGIA))$$

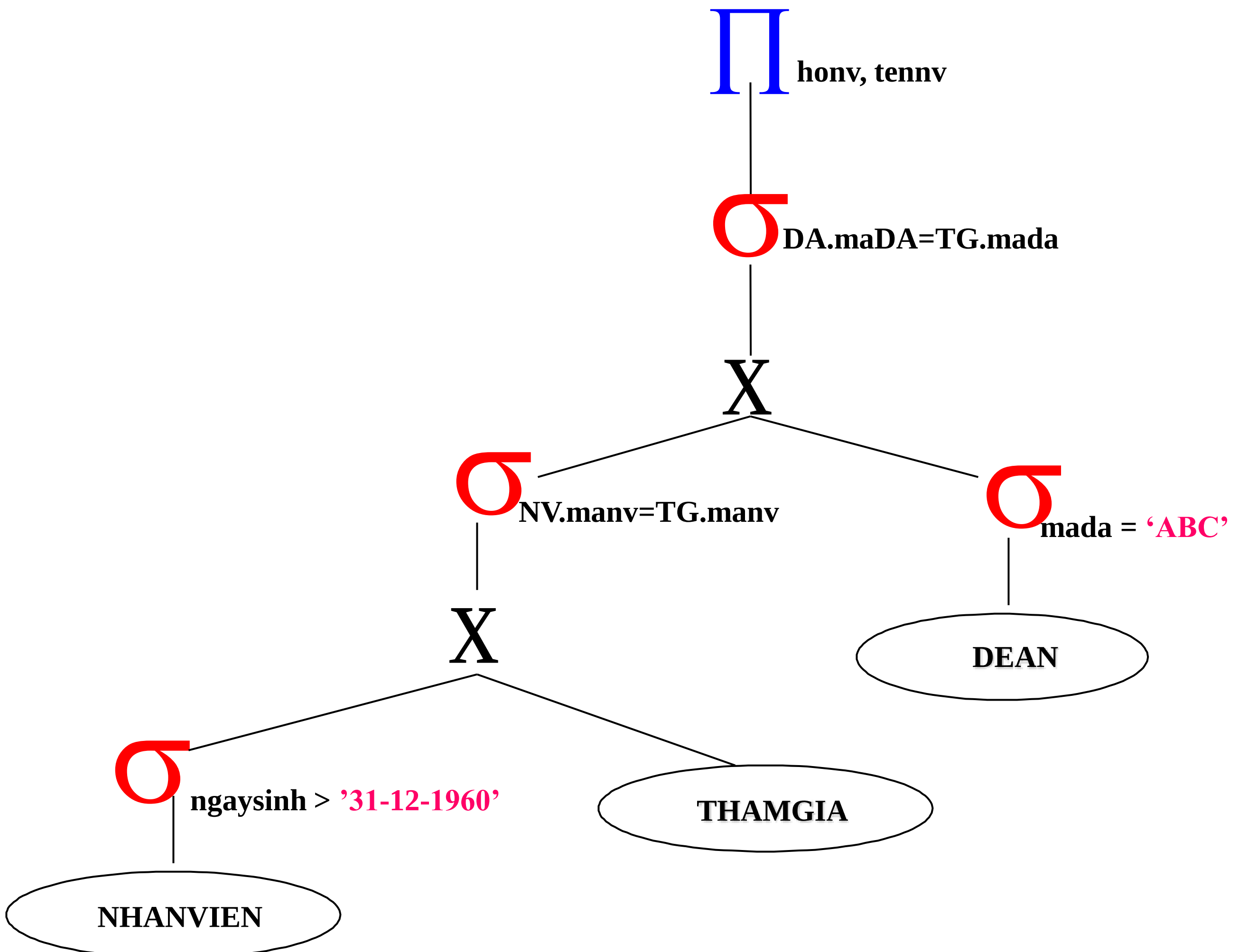
4.1 Heuristic algorithm

$\Pi_{\text{honv, tennv}}(\sigma_{\text{mada = 'ABC'} \wedge \text{ngaysinh > '31-12-1960'} \wedge \text{NV.manv=TG.manv} \wedge \text{DA.mada=TG.mada}}$ (NHANVIEN x DEAN x THAMGIA))



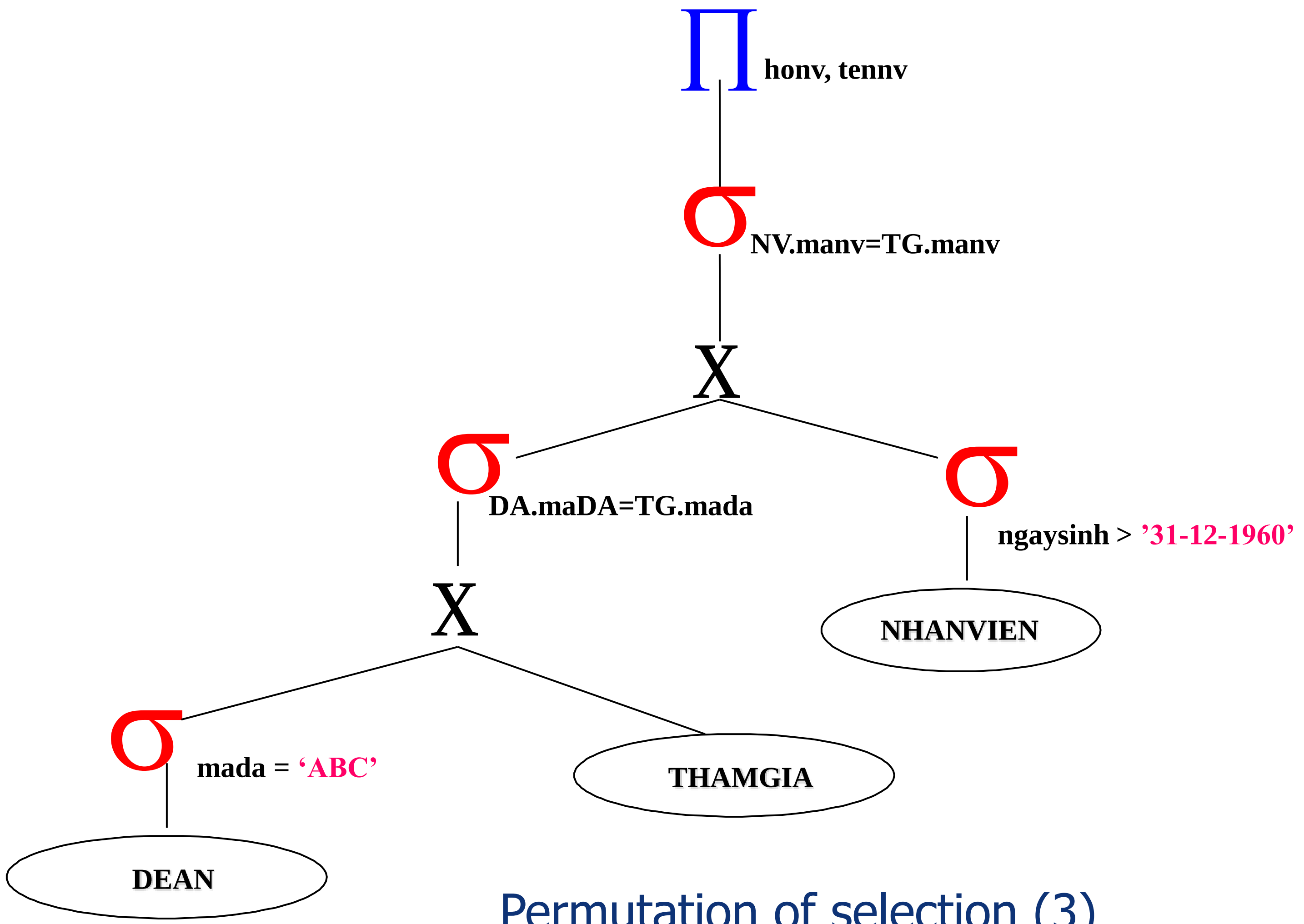
Tree representing query expressions (1)

4.1 Heuristic algorithm



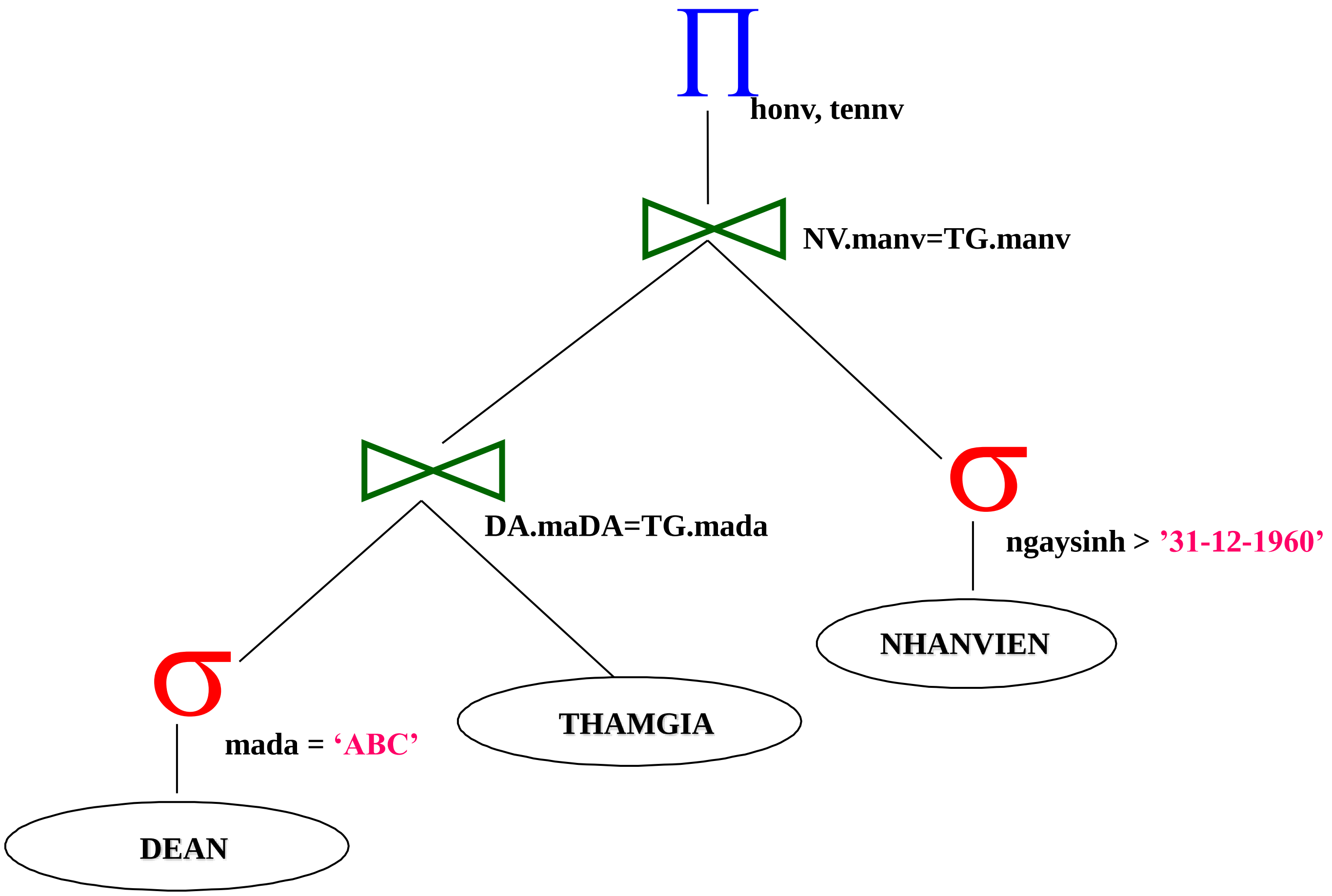
Move the selection down to the branches (2)

4.1 Heuristic algorithm



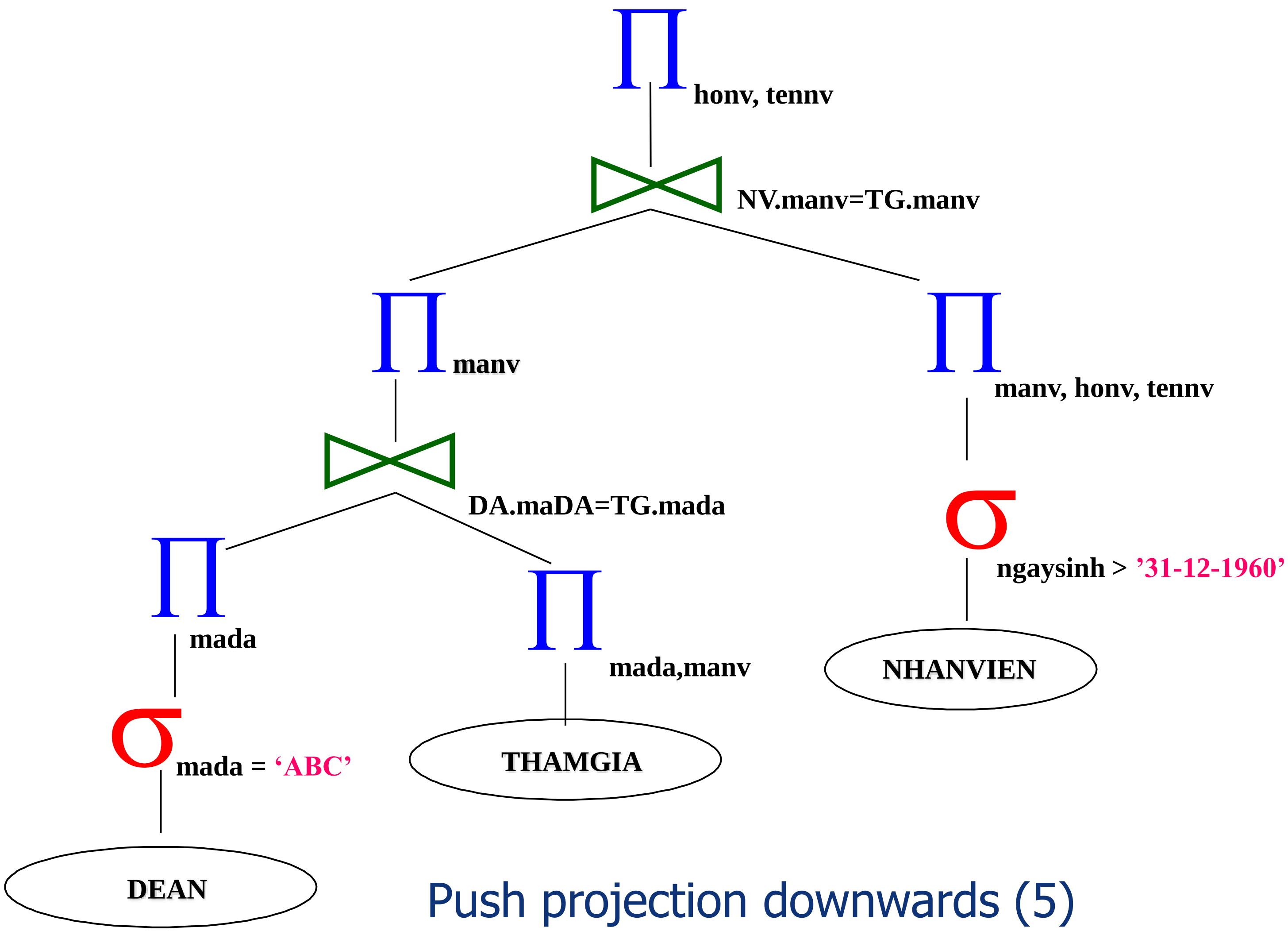
Permutation of selection (3)

4.1 Heuristic algorithm



Replace Cartesian products and selections with conjunctions (4)

4.1 Heuristic algorithm



Push projection downwards (5)

Relational algebra

$$\Pi_{honv,tennv}((\pi_{mada}(\sigma_{mada = 'ABC'}(DEAN))) \underset{DA.mada=TG.mada}{\bowtie} (\pi_{mada,\ manv}(THAMGIA)) \underset{NV.manv=TG.manv}{\bowtie} (\pi_{manv,honv,tennv}(\sigma_{ngaysinh > '31-12-1960'}(NHANVIEN))))$$

SQL

```
SELECT honv, tennv
FROM
        (SELECT mada FROM DEAN
        WHERE mada = 'ABC') AS DA INNER JOIN
        (SELECT mada, manv FROM THAMGIA) AS TG
        ON  DA.mada=TG.mada INNER JOIN
        (SELECT manv, honv, tennv FROM NHANVIEN WHERE ngaysinh> '31-12-1960'  ) NV
        ON  NV.manv=TG.manv
```

Given the following database schema:

•**SVIEN (<u>MASV</u>, TENSV, NAM, MAKHOA)**

•**HPHAN (<u>MAHP</u>, MAMH, HOCKY, NAM, GV)**

•**KQUA (<u>MASV, MAHP</u>, DIEM)**

Use heuristic algorithms to optimize query execution

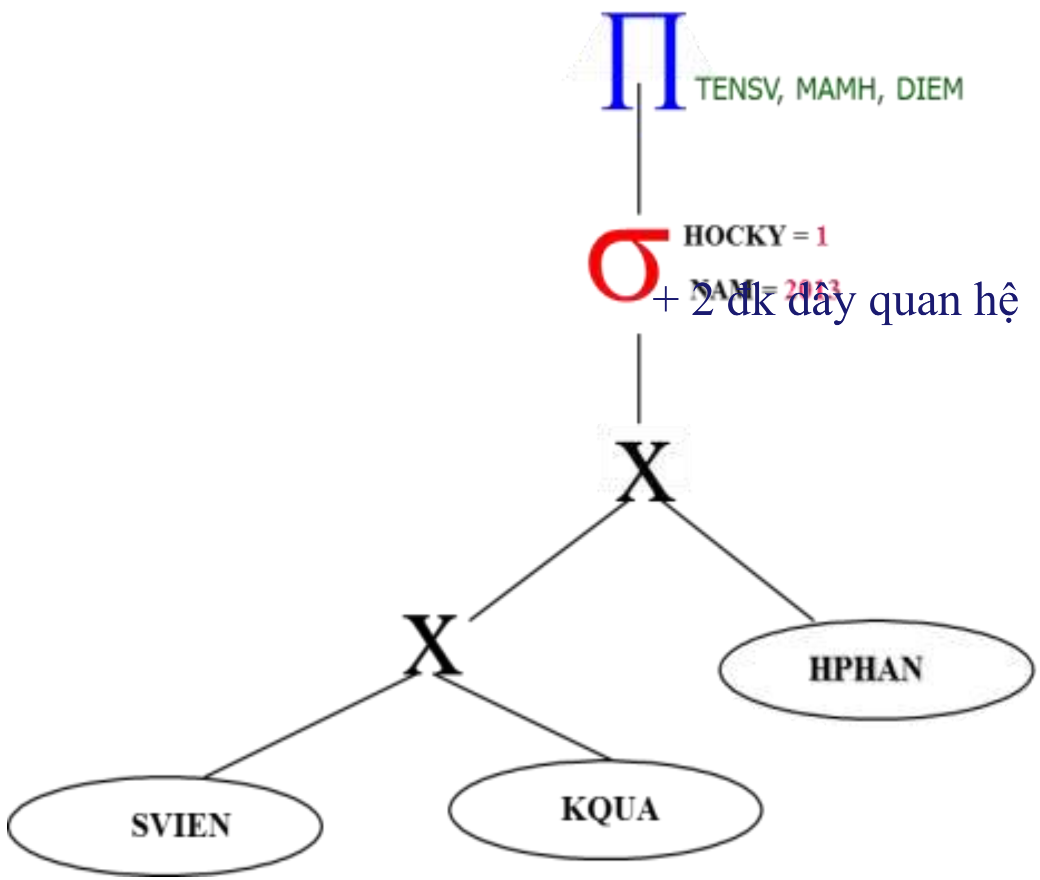
«List student names, course codes, and student scores in semester 1, school year 2013»

$\Pi_{\text{TENSV, MAMH, DIEM}}($

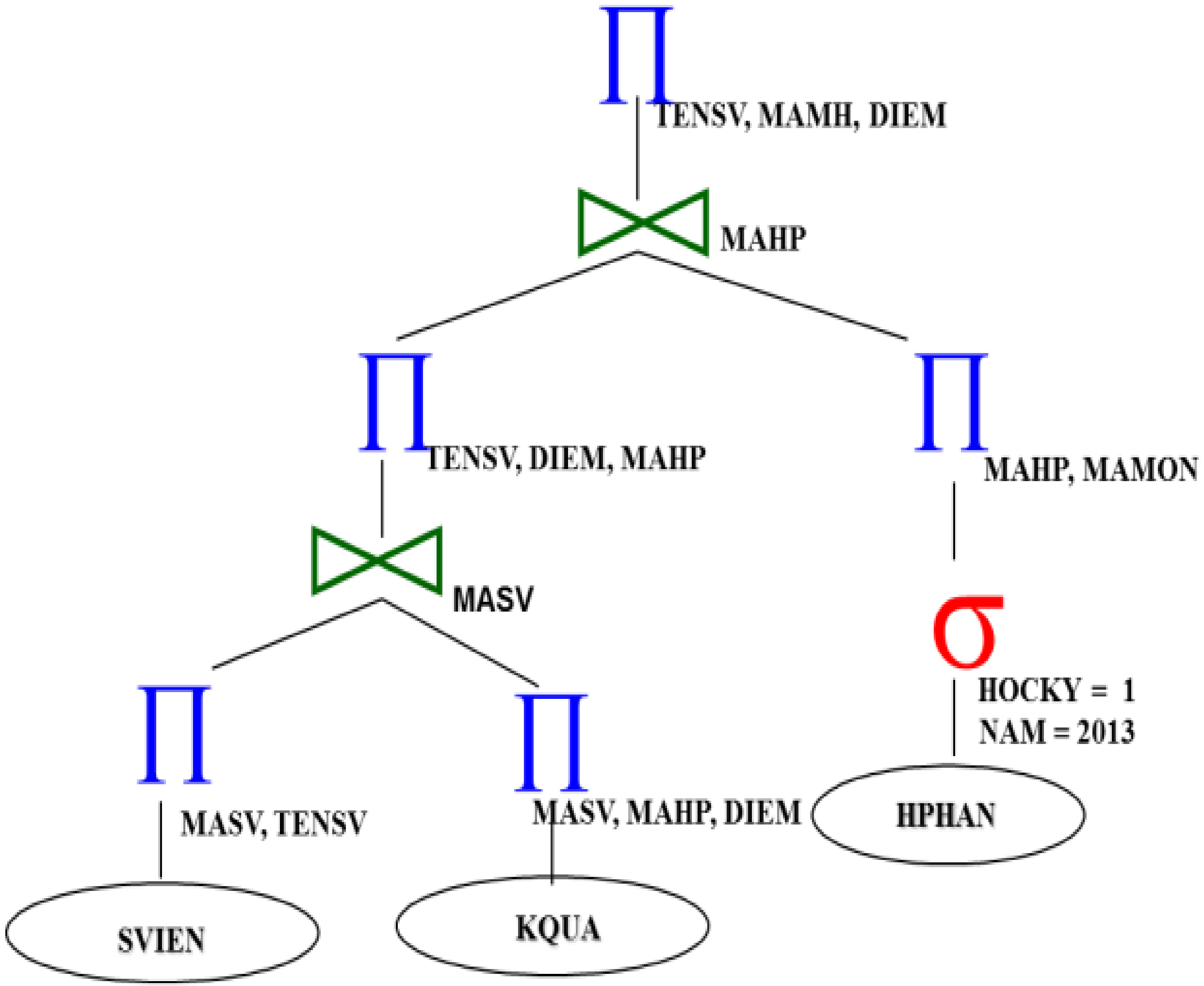
$\sigma_{\text{HOCKY = 1} \wedge \text{NAM=2013} \wedge \text{relationship wire conditions}}$

$(\text{SVIEN x KQUA x HPHAN}))$

$$\Pi_{TENSV,\ MAMH,\ DIEM}(\sigma_{HOCKY = 1\ \wedge\ NAM=2013}(SVIEN \times KQUA \times HPHAN))$$

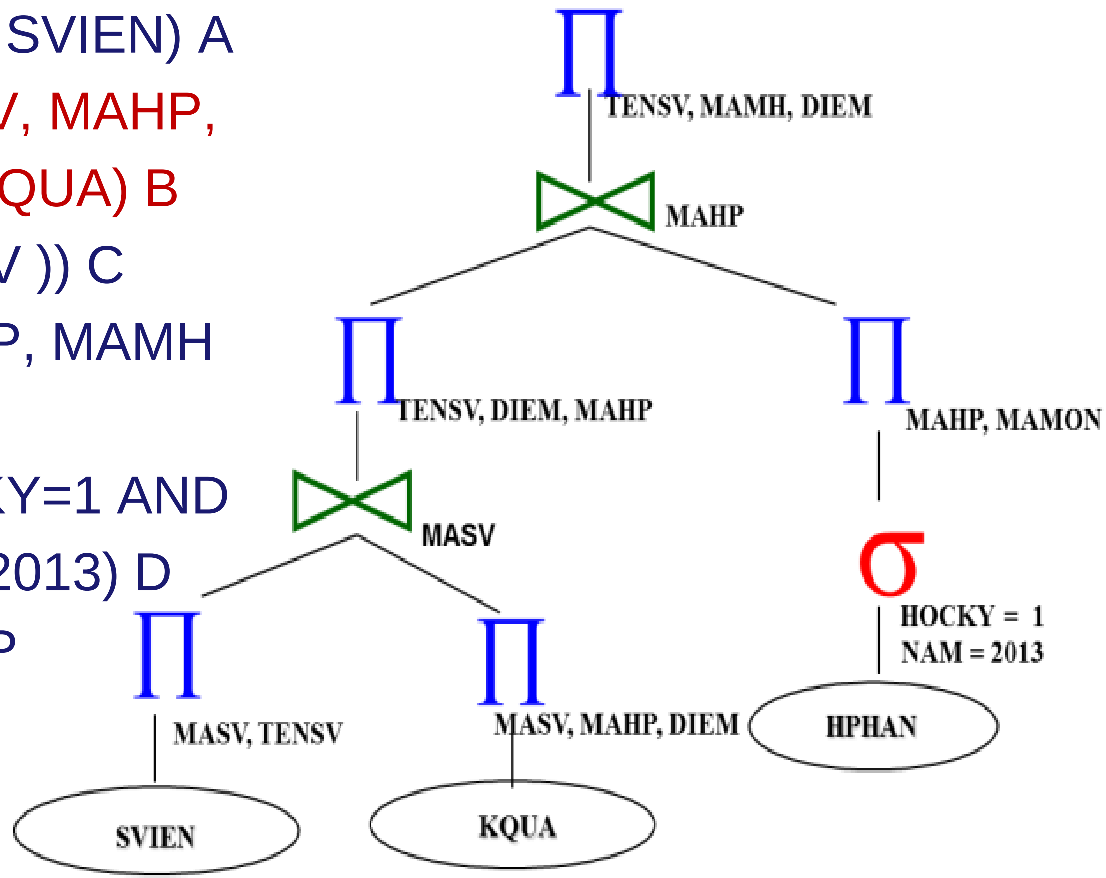


Tree representing query expressions (1)



TENSV, MAMH, DIEM
MAHP
TENSV, DIEM, MAHP
MAHP, MAMON
MASV
HOCKY = 1
NAM = 2013
MASV, TENSV
MASV, MAHP, DIEM
HPHAN
SVIEN
KQUA

```
SELECT A. TENSV, B. DIEM, C. MAMH
FROM  ( SELECT  TENSV,DIEM,MAHP
            FROM   (SELECT MASV,
                     TENSV FROM SVIEN) A
             JOIN (SELECT MASV, MAHP,
                     DIEM FROM KQUA) B
             ON A.MASV=B.MASV )) C
            JOIN  (SELECT MAHP, MAMH
                     FROM HPHAN
                     WHERE HOCKY=1 AND
                                NAM=2013) D
            ON C.MAHP=D.MAHP
```



Given the following database schema:

CAUTHU(<u>**MACT**</u>, TENCT, *MACLB*, NGAYSINH)

CLB(<u>**MACLB**</u>, TENCLB, SOLUONGCT)

Write the query 'List the player ID (MACT), player name (TENCT), date of birth (NGAYSINH) belonging to the club with the number of players (SOLUONGCT) less than 12 players'.

Optimize the execution of the above query using the heuristic algorithm.

Given the following database schema:

CAUTHU(**MACT**, TENCT, *MACLB*, NGAYSINH)

CLB(**MACLB**, TENCLB, SOLUONGCT)

Write the query 'List the player ID (MACT), player name (TENCT), date of birth (NGAYSINH) belonging to the club with the number of players (SOLUONGCT) less than 12 players'.

```
SELECT A.MACT, A.TENCT, A.NGAYSINH
FROM (SELECT MACT,TENCT, NGAYSINH, MACLB
      FROM CAUTHU) A
     JOIN
     (SELECT MACLB,SOLUONGCT
      FROM CLB WHERE SOLUONGCT<12) B
     ON A.MACLB=B.MACLB
```

Content

1. Query execution process
2. Preprocess the query
3. Transform the query
4. Optimize the query
   4.1 Heuristic algorithm
   4.2 Cost estimation
      4.2.1 Cost function for Select
      4.2.2 Cost function for Join

4.2 Cost estimation

❖ Compare costs between query execution options: choose the one with the lowest cost

- Secondary storage costs
- Storage costs
- Calculation costs
- Cost of memory usage
- Communication costs

4.2 Cost estimation

- ***File size parameters***

NHANVIEN

manv	tenv	phai	hsl
NV01	An	Nam	1.5
NV02	Bình	Nam	1.5
NV03	Dung	Nữ	3
NV04	Duyên	Nữ	2.5

manv: char (20)

tennv: nvarchar (50)

phai: nvarchar (10)

hsl (hệ số lương): double

- Number of records in the table (tuples): **T(R)**
- Size of 1 piece of records: **S(R)**
- Total number of blocks containing all tuples: **b**
- Number of records in 1 block: **bfr**
- Number of different values of attribute A (size of value domain): **V(R,A)**

Sample

R

A	B	C	D
x	1	10	a
x	1	20	b
y	1	30	a
y	1	40	c
z	1	50	d

A: character 20 bytes
B: integer 4 bytes
C: date 8 bytes
D: character 68 bytes

1 block = 1024 bytes
(block header: 24 bytes)

T(R) = 5
S(R) = 100*
B(R) = 1

V(R, A) = 3
V(R, C) = 5

V(R, B) = 1
V(R, D) = 4

Example exercises:

For relations R(a,b,c)

With:	a,b integer	4 Bytes
	c string	100 Bytes
Header of each set is		12 Bytes
1 Block		1024 Bytes
Block Header		24 Bytes

Record number of the table T(R)= 10 000

Calculate number of records in 1 block?

Number of Blocks needed to store 10,000 records?

Calculate the minimum file size that can contain the above number of records?

Example exercises:

For relations R(a,b,c) with:

a,b integer 4 Bytes; c string 100 Bytes

Header of each set is 12 Bytes

1 Block 1024 Bytes

Block Header 24 Bytes

Calculate number of records in 1 block?

bfr= 1000/120 ≈ 8 records

Number of Blocks needed to store 10,000 records

B(R)= 10 000/8= 1250 (blocks)

What is the minimum file size that can hold the above number of records? (1250*1024) Bytes

Example exercises:

For relations R(a,b,c)

With:	a,b integer	4 Bytes
	c string	100 Bytes
Header of each set is		12 Bytes
1 Block		1024 Bytes
Block Header		24 Bytes

Record number of the table T(R)= 10 000

If S= $\prod_{a+b,c}(R)$ Calculate B(R) (Hint: 1 Tuple 116 Bytes)

If U= $\prod_{a,b}(R)$ Calculate B(R)

4.2.1 Cost function for Select

[Ullman + 2001]

❖ Estimate W= $\sigma_{A=x}$ (R) *(for condition equal =)*

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

4.2.1 Cost function for Select

[Ullman + 2001]

❖ Estimate W= $\sigma_{A>x}$ (R) *(for condition >, >=, <, <=)*

Cách 1

$$T(W) = \frac{T(R)}{2}$$

Cách 2

$$T(W) = \frac{T(R)}{3}$$

4.2.1 Cost function for Select

❖For example:

R (A, B, C), cost calculation S= $\sigma_{A=10 \wedge B<20}$ (R)

With T(R)=10.000; V(R,A) = 50

We have:

$$T(W) = \frac{T(R)}{V(R,A)} = \frac{10000}{50 \text{ x } 3}$$

4.2.1 Cost function for Select

[Elmasri+2003]

- The function calculates the cost for Select according to the search method P_i: **S_i**
- Block access cost calculated by function S_i: **C_{Si}**

4.2.1 Cost function for Select

❖S1. Linear search

- Browse each record, and check whether the attribute value of that record meets the selection condition (not necessarily the = condition) or not.
- Complexity: O(n)

4.2.1 Cost function for Select

❖S1. Linear search

- For non-key attributes

$$C_{S1a} = b$$

- For condition =, key attribute

$$C_{S1b} = (b/2)$$

  - especially, if no scraps are found $C_{S1b} = b$

4.2.1 Cost function for Select

❖ S2. Binary Search

- If the selection condition (=) on the attribute is ordered, binary search is more efficient than linear search.
- Complexity : $O(\log_2 n)$

4.2.1 Cost function for Select

❖S2. Binary Search

$$C_{S2} = \log_2 b + [(s/bfr)] - 1$$

s: number of records that satisfy condition = on attribute A_k

- Especially for = conditions on key (or UNIQUE) attributes

$$C_{S2} = \log_2 b$$

4.2.1 Cost function for Select

❖ **Example: Given a relational schema**

Nhanvien (manv, honv, tennv, ngaysinh, gioitinh, luong, maphong)

Phongban (maphong, tenphong, ngaythanhlap, maql)

Question: Calculate the cost for the following query

Query: $\sigma_{maphong>5 \wedge manv='NV05'}$ (Nhanvien)

With r_{NV} = 10.000 records, b_{NV}=2000 blocks

4.2.1 Cost function for Select

❖Query: $\sigma_{maphong>5 \wedge manv='NV05'}(Nhanvien)$

- With conditional ***maphong>5***

  C_{S1a}= b=2000

- With conditional ***manv='NV05'***

  C_{S1a}= b/2=1000

  $C_{S2} = \log_2 b = \log_2 2000 = \log_2 2.10^3 = 1 + 3\log_2 10 \approx 11$

→ *So select the condition manv='NV05' to execute first*

4.2.2 Cost function for Join

[Ullman,+ 2001]

R1 (A, B, C); R2 (A, D) $W = R1 \bowtie R2$

Case 1: V(R1,A) ≤ V(R2,A)

$$T(W) = T(R1) \text{x} \frac{T(R2)}{V(R2, A)}$$

Case 2: V(R2,A) ≤ V(R1,A)

$$T(W) = T(R2) \text{x} \frac{T(R1)}{V(R1, A)}$$

4.2.2 Cost function for Join

[Ullman + 2001]

❖ Overview:

$$T(W) = \frac{T(R1) \text{x}\, T(R2)}{\max\,(V(R1,A), V(R2,A))}$$

❖ The number of attribute values that do not participate in the union

V (W, A) = min {V (R1, A), V(R2, A)}

V (W, B) = V (R1, B)

V (W, C) = V (R1, C)

V (W, D) = V (R2, D)

4.2.2 Cost function for Join

[Elmasri+2003]

- The function calculates the cost for Join according to the search method P_i : $\mathbf{J_i}$
- Block access cost calculated by function J_i : $\mathbf{C_{ji}}$

Note: the cost function is only based on the number of blocks transferred from memory to disk (not mentioning calculation time, storage costs and other factors)

4.2.2 Cost function for Join

- Selectivity of combination (js)

  $js = |(R \bowtie_C S)| / |R \times S| = |(R \bowtie_C S)| / (|R| * |S|)$

  $0 <= js <= 1$

- The size of the result set after performing the union $|(R \bowtie_C S)| = js * |R| * |S|$

- R.A=S.B
  - If A is the key of R then $|(R \bowtie_C S)| <= |S|, js <= 1/|R|$
  - If B is the key of S then $|(R \bowtie_C S)| <= |R|, js <= 1/|S|$

4.2.2 Cost function for Join

❖J1. Join combination

– Suppose R is the outer loop

$$C_{J1} = b_R + (b_R * b_S) + ((js * |R| * |S|)/bfr_{RS})$$

4.2.2 Cost function for Join

❖ Example: Calculate the cost for the following conclusion

Query: NHANVIEN $\bowtie_{NV.maphong=PB.maphong}$ PHONGBAN

With: $r_{Nhanvien}=10000$, $r_{Phongban}=125$, $b_{Nhanvien}=2000$, $b_{Phongban}=13$, $brf_{NV\_PB}=4$

4.2.2 Cost function for Join

- $js = 1/|PHONGBAN| = 1/125$

- Using J1 with NHANVIEN is the outer loop

$$C_{J1} = b_{NV} + (b_{NV} * b_{PB}) + ((js * r_{NV} * r_{PB}) / brf_{NV\_PB})$$

$$= 2000 + (2000*13) + ((1/125 * 10000 * 125)/4) = 30500$$

- Using J1 with PHONGBAN is the outer loop

$$C_{J1} = b_{PB} + (b_{PB} * b_{NV}) + ((js * r_{NV} * r_{PB}) / brf_{NV\_PB})$$

$$= 13 + (13*2000) + ((1/125 * 10000 * 125)/4) = 28513$$

So use PHONGBAN as the outer loop

Discussion