

Trường Đại Học Bách Khoa Tp.HCM
Hệ Đào Tạo Từ Xa
Khoa Khoa Học và Kỹ Thuật Máy Tính



HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Chương 4. Các giải thuật xử lý và tối ưu hóa truy vấn

Bùi Hoài Thắng

Nội dung

- * Chuyển đổi các truy vấn SQL thành đại số quan hệ
- * Các giải thuật xếp thứ tự ngoại
- * Dùng các heuristic để tối ưu hóa truy vấn
- * Các giải thuật cho phép chọn và phép kết
- * Các giải thuật cho phép chiếu và các phép toán trên tập hợp
- * Hiện thực các tác vụ gộp và các phép kết ngoài
- * Kết hợp các phép toán bằng cách dùng kỹ thuật pipelining
- * Sử dụng các ước lượng chi phí để tối ưu hóa truy vấn
- * Các kỹ thuật tối ưu hóa truy vấn trong SQL-Server



BÀI 1 – TỔNG QUAN



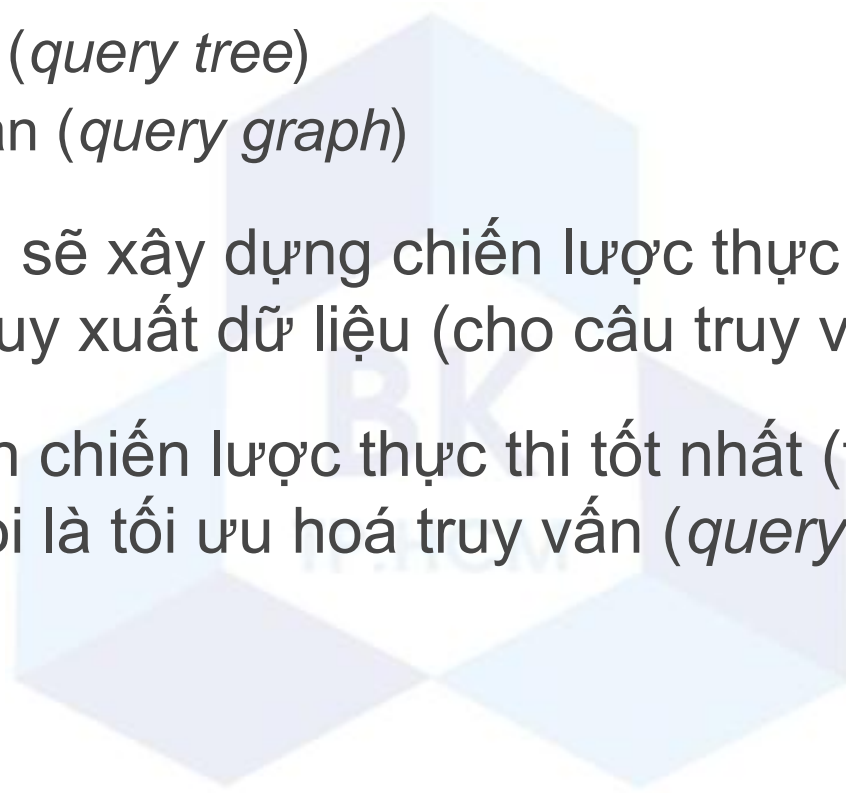
PHẦN 1. XỬ LÝ TRUY VẤN

Giới thiệu về xử lý truy vấn

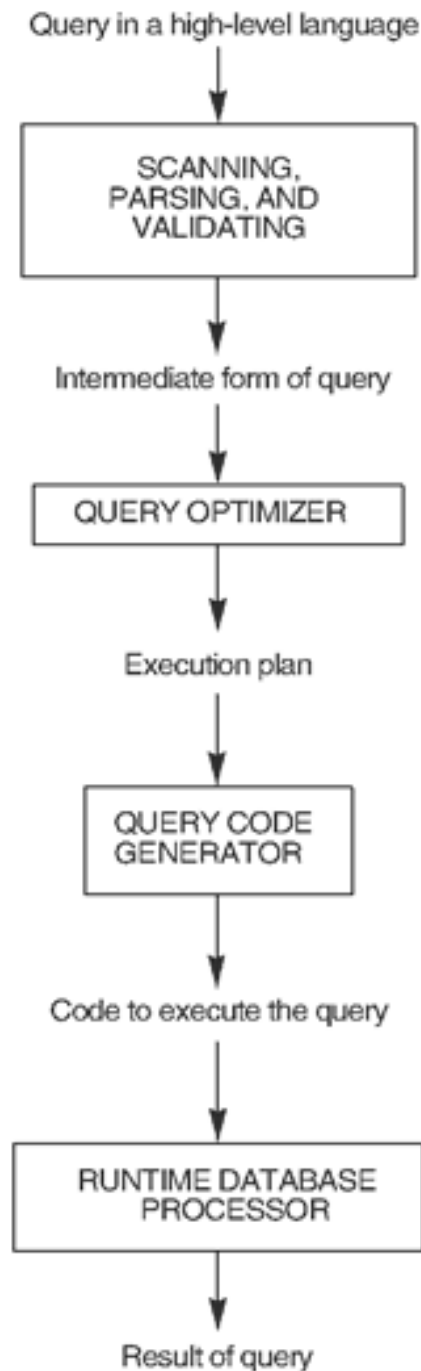
- * Câu truy vấn được biểu diễn ở ngôn ngữ truy vấn cấp cao như SQL (Structured Query Language)
 - Cần phải được quét, phân tích ngữ pháp và kiểm chứng
- * Quét (*scan*):
 - Nhận dạng các từ khoá (*keyword*), bảng, thuộc tính, ...
- * Phân tích ngữ pháp (*parse*):
 - Kiểm tra ngữ pháp của câu truy vấn
- * Kiểm chứng (*validate*):
 - Đảm bảo các bảng, thuộc tính, ... là đúng và có nghĩa trong CSDL cần truy vấn

Giới thiệu về xử lý truy vấn (tt.)

- * Hai cấu trúc thường được dùng để biểu diễn câu truy vấn sau bước phân tích:
 - Cây truy vấn (*query tree*)
 - Đồ thị truy vấn (*query graph*)
- * Sau đó DBMS sẽ xây dựng chiến lược thực thi (*execution strategy*) để truy xuất dữ liệu (cho câu truy vấn)
- * Quá trình chọn chiến lược thực thi tốt nhất (trong nhiều chiến lược) gọi là tối ưu hoá truy vấn (*query optimization*)



Các bước của quá trình xử lý câu truy vấn



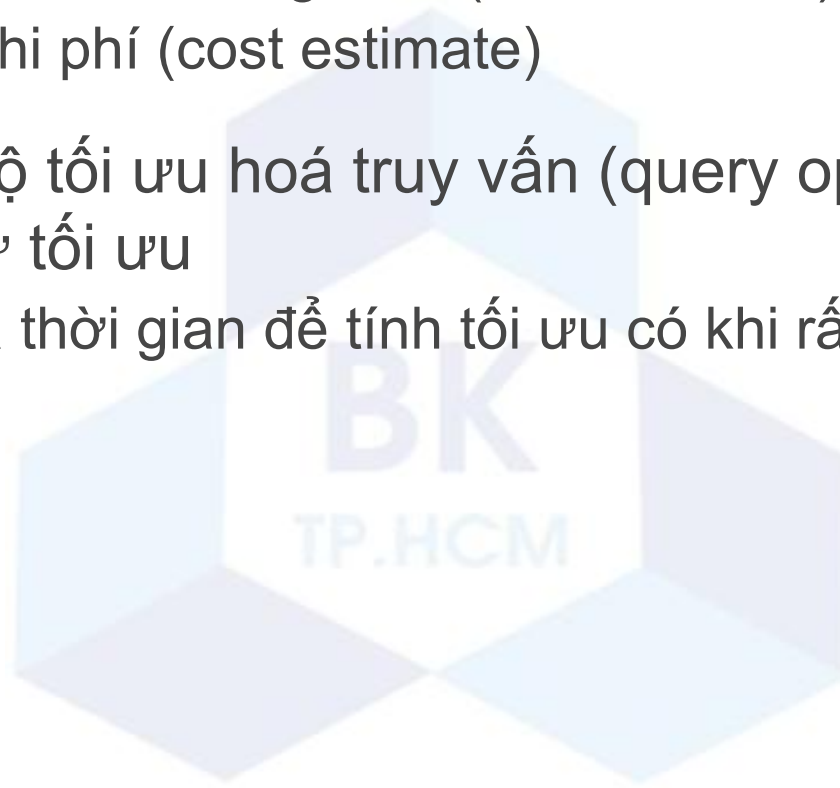
Code can be:

- Executed directly (interpreted mode)
- Stored and executed later whenever needed (compiled mode)

Hình 4.1

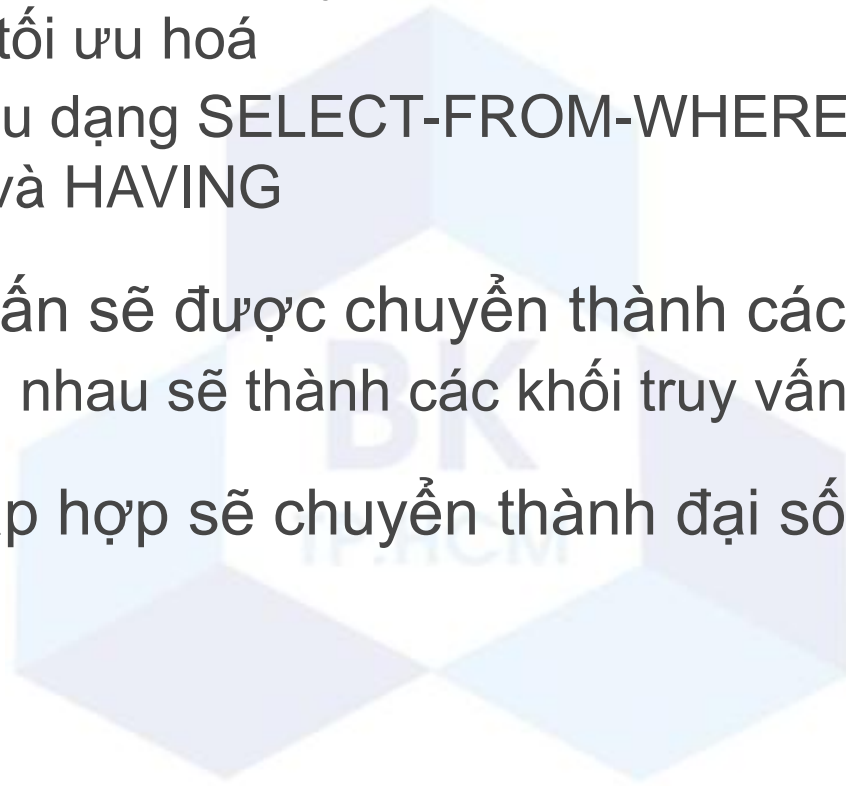
Ghi chú về tối ưu hoá truy vấn

- * Có 2 phương pháp chủ yếu về tối ưu:
 - Dùng các qui luật kinh nghiệm (heuristic rule)
 - Ước lượng chi phí (cost estimate)
- * Kết quả của bộ tối ưu hoá truy vấn (query optimizer) có thể không thực sự tối ưu
 - Do chi phí và thời gian để tính tối ưu có khi rất lớn

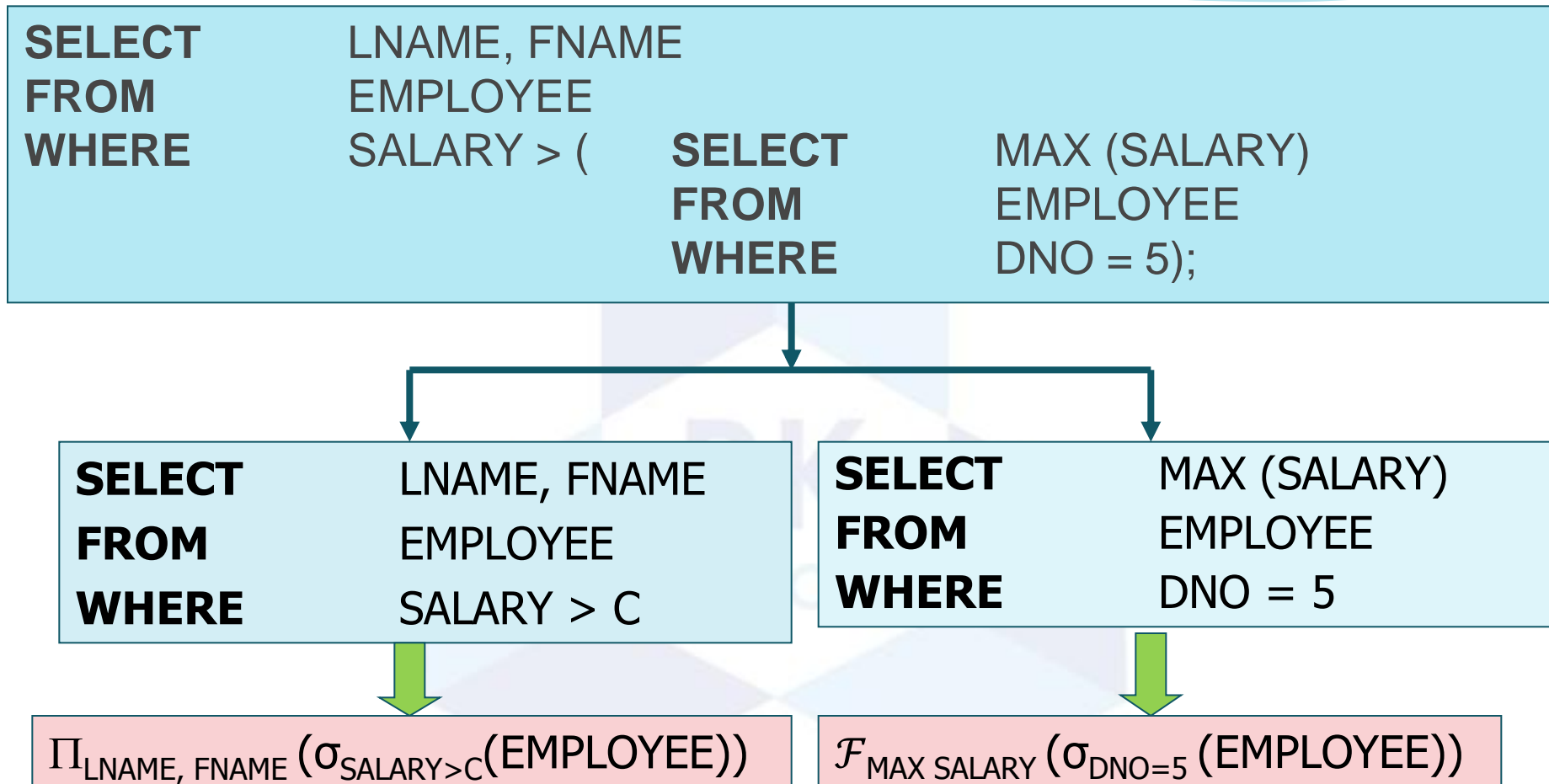


Chuyển từ SQL sang đại số quan hệ (Relational Algebra)

- * Khối truy vấn (query block):
 - Đơn vị cơ bản có thể chuyển thành các toán tử đại số (algebra operator) và tối ưu hoá
 - Chứa một câu dạng SELECT-FROM-WHERE và có thể có GROUP BY và HAVING
- * Một câu truy vấn sẽ được chuyển thành các khối truy vấn
 - Các câu lồng nhau sẽ thành các khối truy vấn riêng lẻ
- * Các toán tử tập hợp sẽ chuyển thành đại số mở rộng



Chuyển từ SQL sang đại số quan hệ (tt.)



Hình 4.2

Xếp thứ tự ngoại (External sort)

- * Xếp thứ tự là một trong các giải thuật quan trọng trong xử lý truy vấn:
 - ORDER BY
 - Dùng trong giải thuật sort-merge cho phép kết, UNION, ...
- * Xếp thứ tự ngoại (External sorting):
 - Dùng xếp thứ tự các tập tin lớn không chứa đủ trong bộ nhớ chính
- * Chiến lược Xếp-và-Trộn (Sort-Merge):
 - Ban đầu xếp thứ tự các phần nhỏ của tập tin (run)
 - Sau đó trộn chúng lại thành các run lớn hơn (có thứ tự)
 - Tiếp tục

Ví dụ sort-merge

1 block = 2 records

15	22	2	27	14	6	51	18	35	16	50	36	9	8	32	12	11	33	30	30	23	21	24	29
----	----	---	----	----	---	----	----	----	----	----	----	---	---	----	----	----	----	----	----	----	----	----	----

buffer = 3 blocks = 6 records



Giai đoạn sắp xếp:

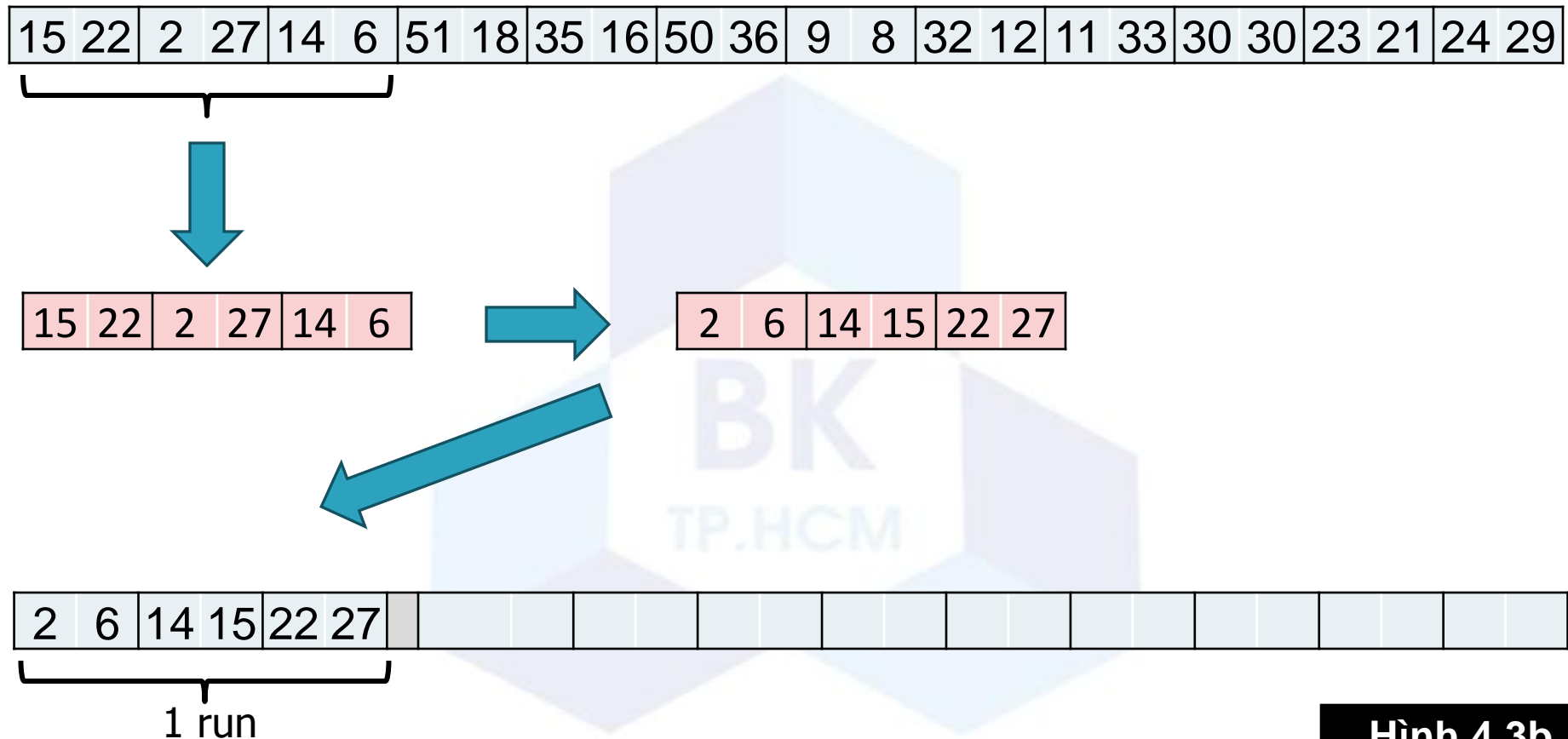
Đọc lần lượt 3 khối trên tập tin và xếp thứ tự

=> chép xuống thành các run có chiều dài là 3 blocks

Hình 4.3a

Ví dụ sort-merge (tt.)

Giai đoạn sắp xếp

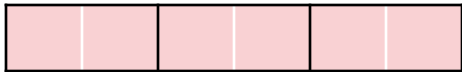


Hình 4.3b

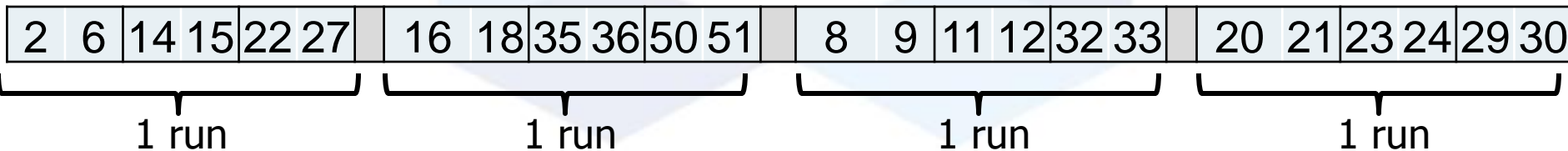
Ví dụ sort-merge (tt.)

Giai đoạn sắp xếp

15	22	2	27	14	6	51	18	35	16	50	36	9	8	32	12	11	33	20	30	23	21	24	29
----	----	---	----	----	---	----	----	----	----	----	----	---	---	----	----	----	----	----	----	----	----	----	----



Hình 4.3d



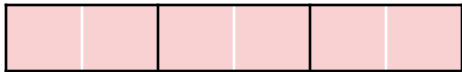
Ví dụ sort-merge (tt.)

2	6	14	15	22	27		16	18	35	36	50	51		8	9	11	12	32	33		20	21	23	24	29	30
---	---	----	----	----	----	--	----	----	----	----	----	----	--	---	---	----	----	----	----	--	----	----	----	----	----	----

Giai đoạn trộn:

Mỗi bước trộn:

- Đọc 1 block từ (k-1) run
- Trộn các block trên buffer vào block tạm
- Nếu block tạm đầy: ghi xuống file
- Nếu block của dữ liệu hết: đọc tiếp block kế của run

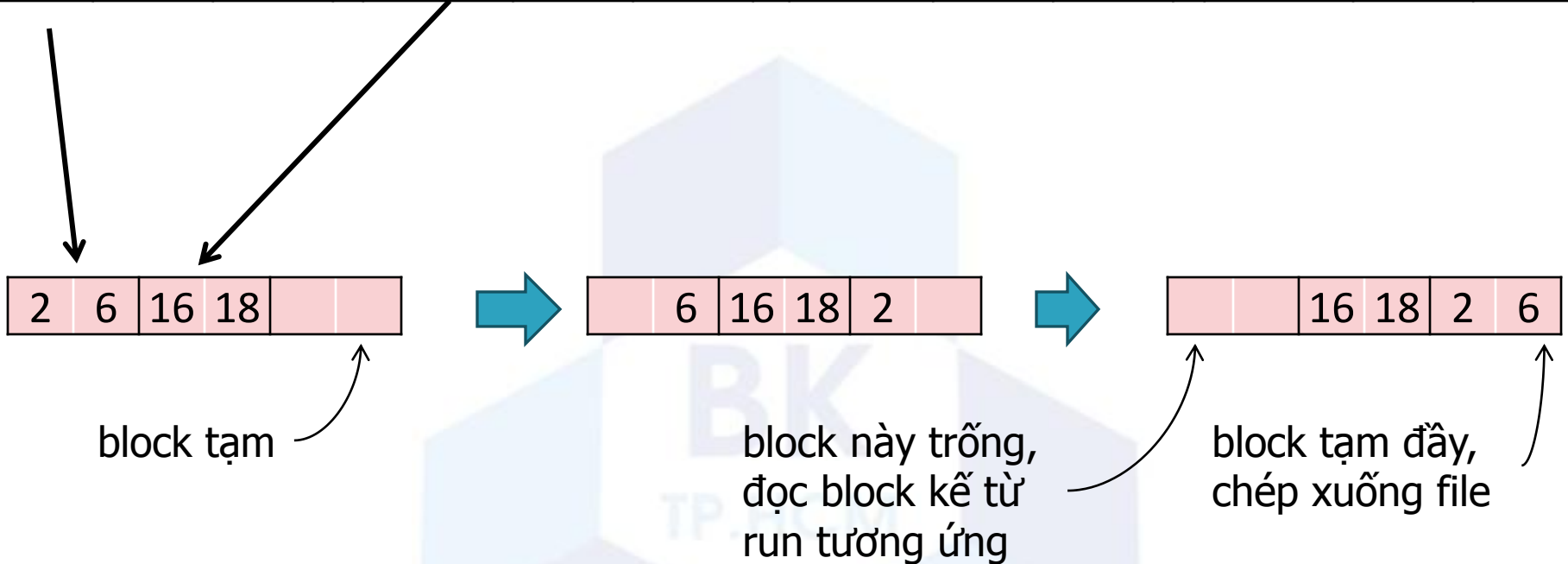


Hình 4.3e

Ví dụ sort-merge (tt.)

Giai đoạn trộn:

2	6	14	15	22	27		16	18	35	36	50	51		8	9	11	12	32	33		20	21	23	24	29	30
---	---	----	----	----	----	--	----	----	----	----	----	----	--	---	---	----	----	----	----	--	----	----	----	----	----	----

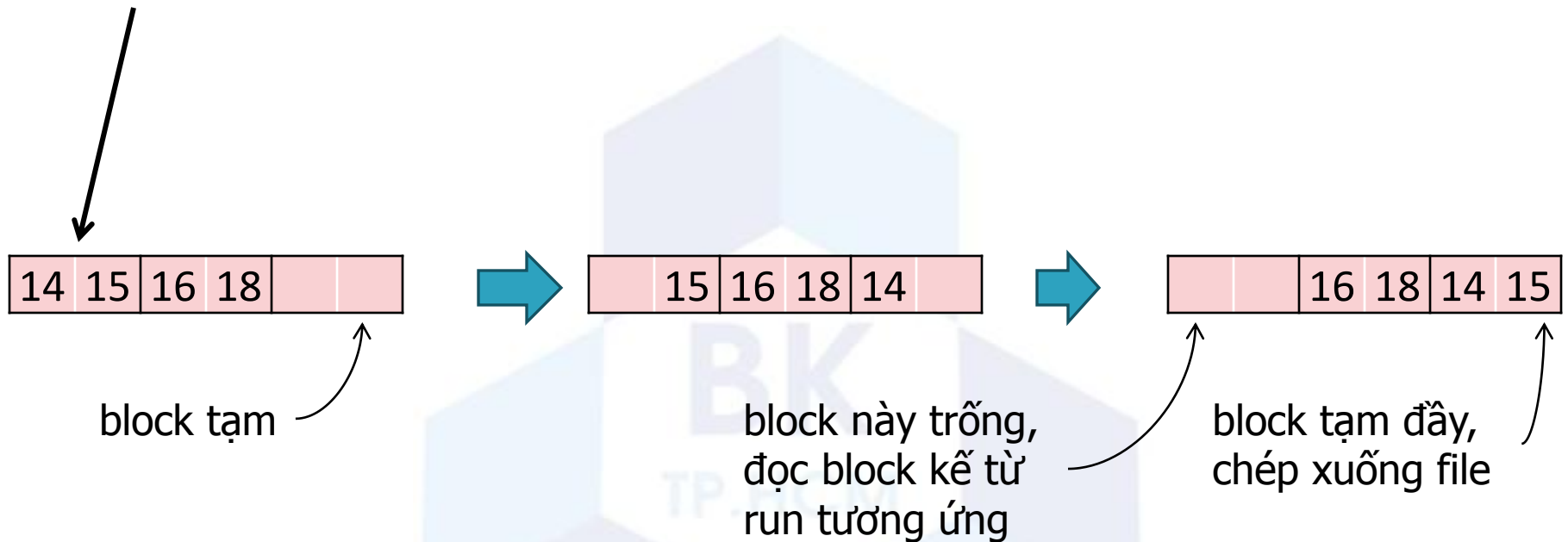


Hình 4.3f

Ví dụ sort-merge (tt.)

Giai đoạn trộn:

2	6	14	15	22	27		16	18	35	36	50	51		8	9	11	12	32	33		20	21	23	24	29	30
---	---	----	----	----	----	--	----	----	----	----	----	----	--	---	---	----	----	----	----	--	----	----	----	----	----	----

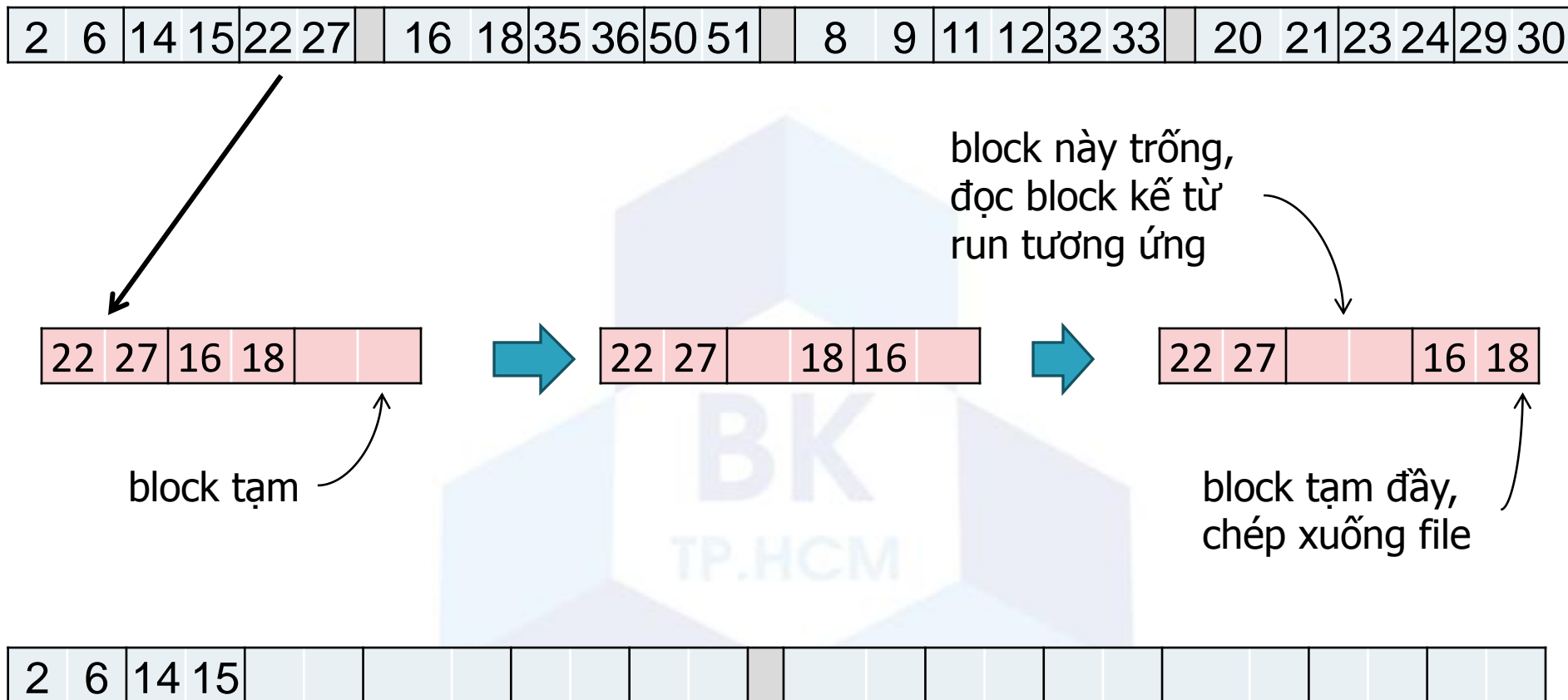


2	6																										
---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Hình 4.3g

Ví dụ sort-merge (tt.)

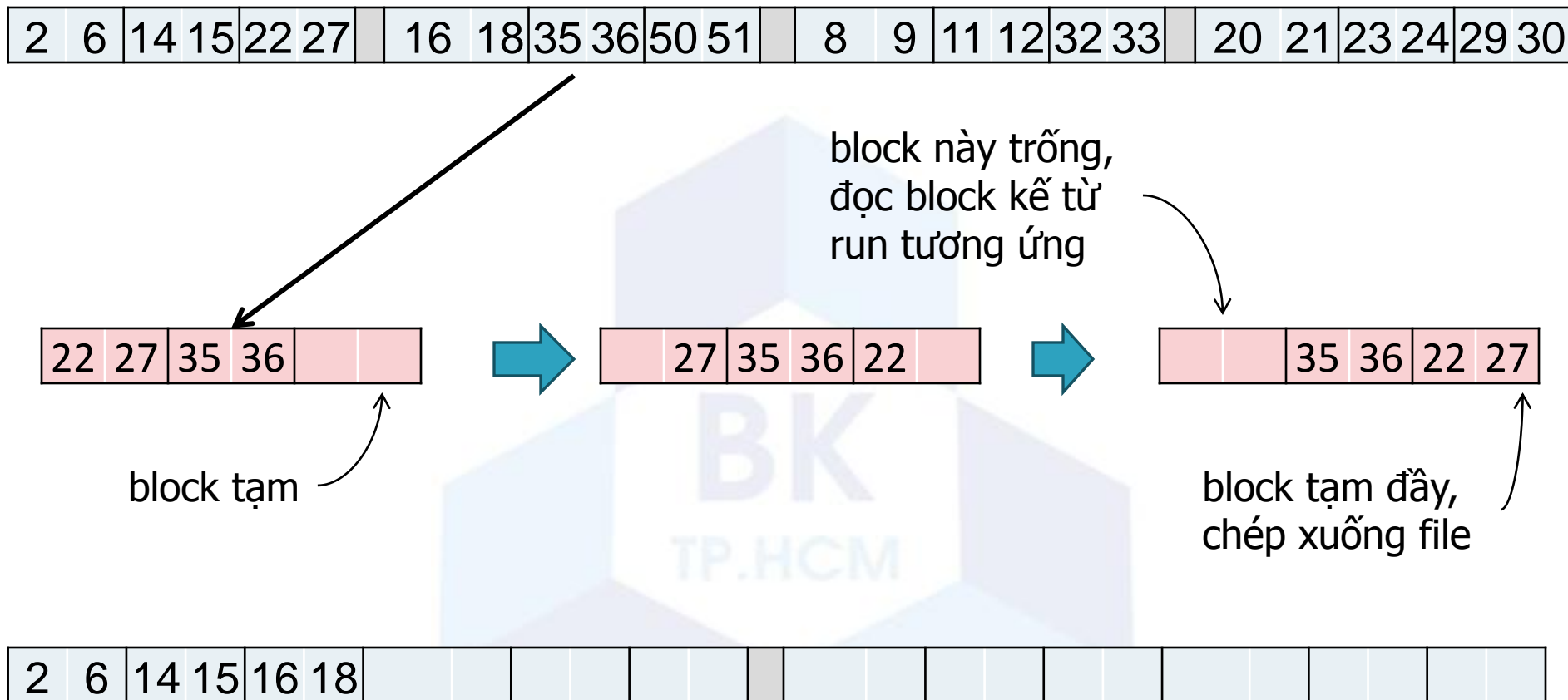
Giai đoạn trộn:



Hình 4.3h

Ví dụ sort-merge (tt.)

Giai đoạn trộn:



Hình 4.3i

Ví dụ sort-merge (tt.)

Giai đoạn trộn:

2	6	14	15	22	27		16	18	35	36	50	51		8	9	11	12	32	33		20	21	23	24	29	30
---	---	----	----	----	----	--	----	----	----	----	----	----	--	---	---	----	----	----	----	--	----	----	----	----	----	----

run này không còn block nào => loại run
này ra khỏi bước trộn này => chép các
block của run còn lại vào file

		35	36		
--	--	----	----	--	--

block tạm

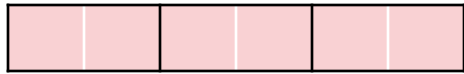
2	6	14	15	16	18	22	27																			
---	---	----	----	----	----	----	----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Hình 4.3j

Ví dụ sort-merge (tt.)

Giai đoạn trộn:

2	6	14	15	22	27		16	18	35	36	50	51		8	9	11	12	32	33		20	21	23	24	29	30
---	---	----	----	----	----	--	----	----	----	----	----	----	--	---	---	----	----	----	----	--	----	----	----	----	----	----



block tạm
1 run mới

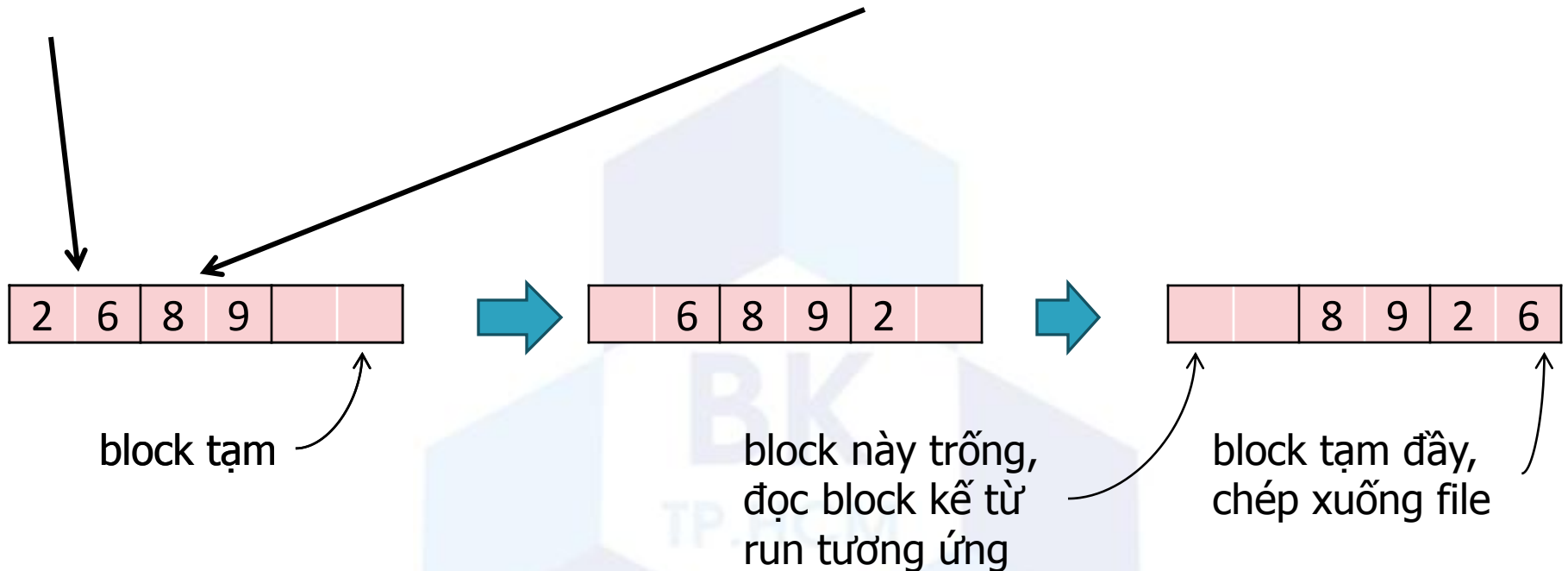
2	6	14	15	16	18	22	27	35	36	50	51															
---	---	----	----	----	----	----	----	----	----	----	----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Hình 4.3k

Ví dụ sort-merge (tt.)

Giai đoạn trộn: Lần trộn kế tiếp

2	6	14	15	16	18	22	27	35	36	50	51		8	9	11	12	20	21	23	24	29	30	32	33
---	---	----	----	----	----	----	----	----	----	----	----	--	---	---	----	----	----	----	----	----	----	----	----	----

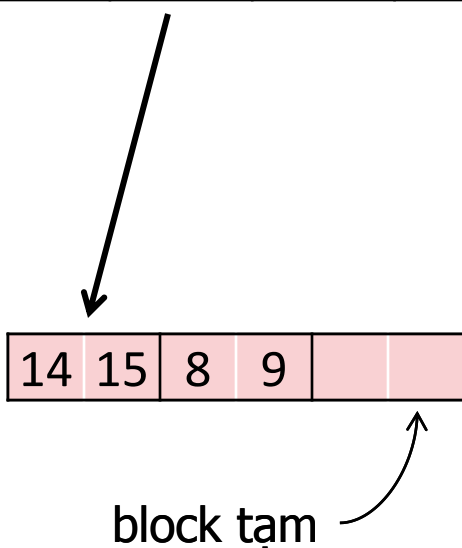


Hình 4.3I

Ví dụ sort-merge (tt.)

Giai đoạn trộn: Lần trộn kế tiếp

2	6	14	15	16	18	22	27	35	36	50	51		8	9	11	12	20	21	23	24	29	30	32	33
---	---	----	----	----	----	----	----	----	----	----	----	--	---	---	----	----	----	----	----	----	----	----	----	----



14	15	8	9		
----	----	---	---	--	--

block tạm

2	6																								
---	---	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Hình 4.3m

Ví dụ sort-merge (tt.)

Kết quả

2	6	8	9	11	12	14	15	16	18	20	21	22	23	24	27	29	30	32	33	35	36	50	51
---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Hình 4.3n

Giải thuật sort-merge

```
set  $i \leftarrow 1, j \leftarrow b$ ; /* size of the file in blocks */  
   $k \leftarrow n_B$ ; /* size of buffer in blocks */  
   $m \leftarrow \lceil (j/k) \rceil$ ;
```

```
{Sort phase}
```

```
while ( $i \leq m$ ) do
```

```
{
```

```
  read next  $k$  blocks of the file into the buffer or  
  if there are less than  $k$  blocks remaining,  
  then read in the remaining blocks;
```

```
  sort the records in the buffer and write as a temporary subfile;
```

```
   $i \leftarrow i+1$ ;
```

```
}
```

```
/*Merge phase: merge subfiles until only 1 remains */
set i  $\leftarrow$  1;
  p  $\leftarrow$   $\lceil \log_{k-1} m \rceil$ ; { p is the number of passes for the
                           merging phase }

  j  $\leftarrow$  m;
while (i  $\leq$  p) do {
  n  $\leftarrow$  1;
  q  $\leftarrow$   $\lceil (j/(k-1)) \rceil$ ;
  while ( n  $\leq$  q) do {
    read next k-1 subfiles or remaining subfiles
    (from previous pass) one block at a time
    merge and write as new subfile one block at a time;
    n  $\leftarrow$  n+1;
  }
  j  $\leftarrow$  q;
  i  $\leftarrow$  i+1;
}
```

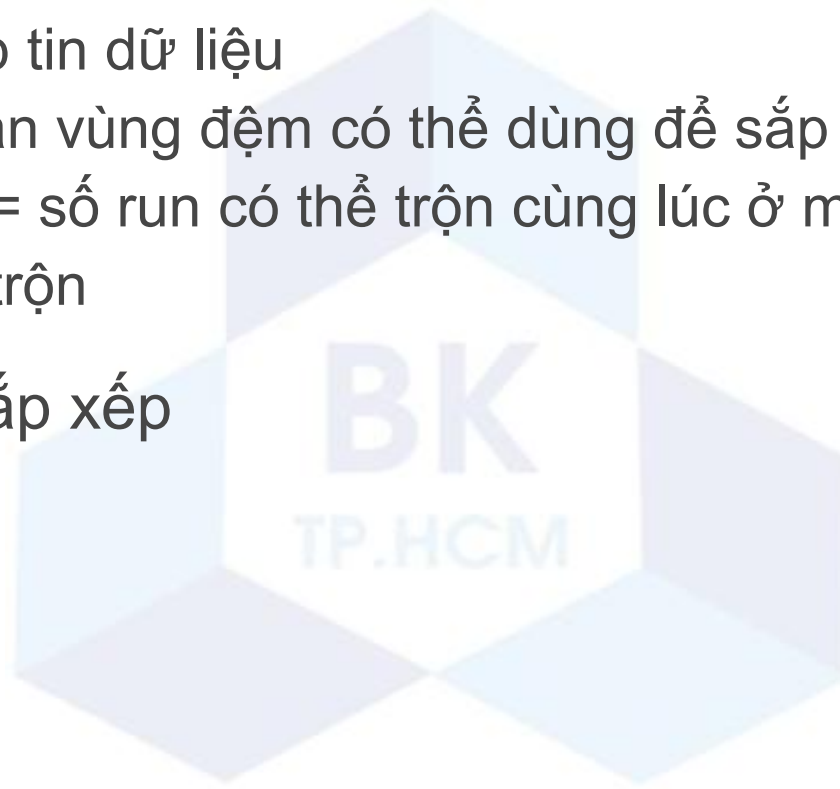
Phân tích giải thuật sort-merge

* Thông số:

- n_R : số run ban đầu
- b : số khối tập tin dữ liệu
- n_B : không gian vùng đệm có thể dùng để sắp xếp
- d_M : bậc trộn = số run có thể trộn cùng lúc ở mỗi bước trộn
- n_P : số bước trộn

* Ở giai đoạn sắp xếp

- $n_R = \lceil (b/n_B) \rceil$



Phân tích giải thuật sort-merge (tt.)

* Ở giai đoạn trộn

➤ $d_M = \min(n_B - 1, n_R)$

- Do cần 1 khối nhớ trống để làm vùng tạm trong quá trình sắp xếp trên bộ nhớ

➤ $n_P = \lceil (\log_{d_M}(n_R)) \rceil$

- Do sau mỗi bước trộn, chiều dài các run tăng lên
 - Ban đầu là n_R
 - Sau bước trộn 1 sẽ là $d_M * n_R$
 - Sau bước trộn 2 sẽ là $d_M * (\text{chiều dài ở bước 1}) = d_M^2 * n_R$

* Tổng cộng:

➤ $(2 * b) + (2 * (b * \lceil (\log_{d_M}(n_R)) \rceil))$

- $2*b$ cho lần sắp xếp đầu tiên: đọc và ghi b khối
- Các lần trộn, mỗi lần đọc và ghi b khối

Ví dụ phân tích sort-merge

- * Thông số:

- Không gian vùng đệm dùng sắp xếp $n_B = 5$ khối
- Số khối của tập tin $b = 1024$

- * Số run ban đầu:

- $n_R = \lceil (b/n_B) \rceil = 205$
- (Sau khi sắp xếp ở giai đoạn đầu, có 205 run có thứ tự, mỗi run có 5 khối, trừ run cuối có 4 khối)

- * Giai đoạn trộn:

- Bậc trộn $d_M = \min(n_B - 1, n_R) = 4$
- Số bước trộn $n_P = \lceil (\log_{d_M}(n_R)) \rceil = \lceil (\log_4(205)) \rceil = 4$
- (Sau bước 1 còn 52 run, sau bước 2 còn 13 run, sau bước 3 còn 4 run, sau bước 4 còn 1 run \Rightarrow tập tin có thứ tự)

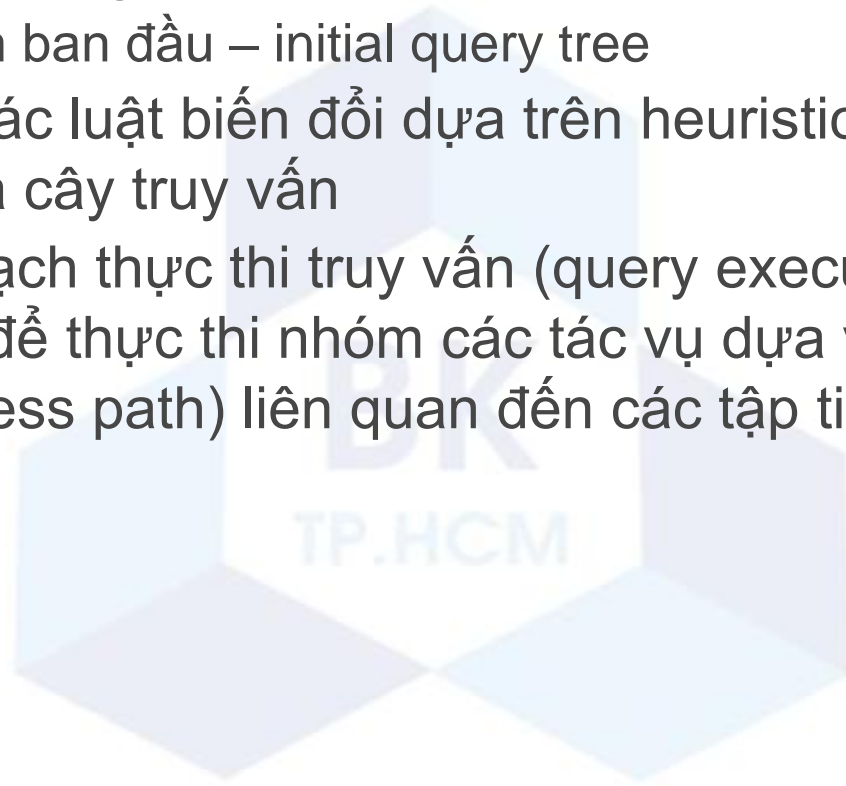


PHẦN 2. TỐI ƯU HOÁ DỰA TRÊN HEURISTIC

Dùng heuristic trong tối ưu hoá truy vấn

* Lộ trình:

- 1. Bộ phân tích ngữ pháp tạo ra một cấu trúc nội ban đầu:
 - cây truy vấn ban đầu – initial query tree
- 2. Áp dụng các luật biến đổi dựa trên heuristic (heuristics rules) để tối ưu hoá cây truy vấn
- 3. Một kế hoạch thực thi truy vấn (query execution plan) sẽ được tạo ra để thực thi nhóm các tác vụ dựa vào các đường đi truy đạt (access path) liên quan đến các tập tin trong câu truy vấn



Dùng heuristic trong tối ưu hoá truy vấn (tt.)

- * Cây truy vấn (Query tree):
 - Cấu trúc dữ liệu dạng cây biểu diễn biểu thức đại số quan hệ
 - Các tập tin dữ liệu dùng trong truy vấn là các nút lá
 - Các phép toán đại số quan hệ (relational algebra operation) là các nút nội (các nút khác nút lá)
- * Thi hành cây truy vấn:
 - Khi các nút con của một nút nội có dữ liệu (từ tập tin hoặc là kết quả của các tác vụ của nút đó), thực hiện phép toán đại số quan hệ và thay thế nút này bằng dữ liệu trung gian (mới được tạo ra)
- * Đồ thị truy vấn (Query graph):
 - Cấu trúc dữ liệu dạng đồ thị biểu diễn biểu thức đại số quan hệ
 - Không cần diễn đạt thứ tự thực hiện các phép toán
 - Chỉ có một đồ thị cho một câu truy vấn

Dùng heuristic trong tối ưu hoá truy vấn – Heuristic chính

- * Áp dụng trước các tác vụ thu gọn kích thước các kết quả trung gian
- * Ví dụ: Thực hiện tác vụ chọn (SELECT) và chiếu (PROJECT) trước khi thực hiện phép kết (JOIN)



Ví dụ cây và đồ thị truy vấn

- * Yêu cầu:

- Đối với từng dự án (project) thực hiện ở 'Stafford', truy xuất mã dự án (project number), mã của phòng quản lý dự án này (department number) và họ, địa chỉ và ngày sinh của người quản lý phòng (department manager's last name, address and birthdate).

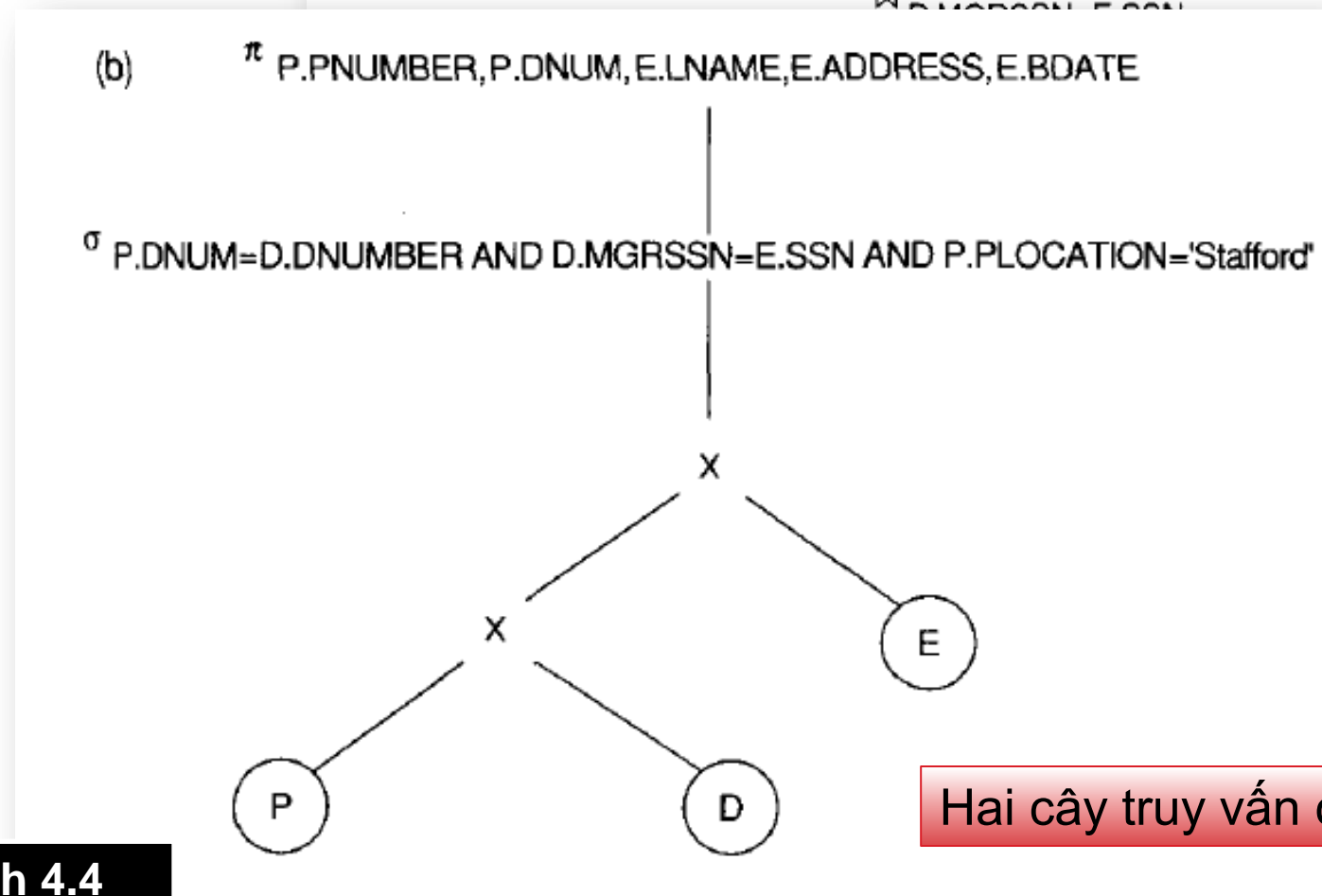
- * Đại số quan hệ:

π PNUMBER, DNUM, LNAME, ADDRESS, BDATE
((($\sigma_{PLOCATION='STAFFORD'}(PROJECT)$))
 \bowtie $\sigma_{DNUM=DNUMBER}(DEPARTMENT)$) \bowtie $\sigma_{MGRSSN=SSN}(EMPLOYEE)$

- * Câu truy vấn SQL:

Q2: SELECT P.NUMBER, P.DNUM, E.LNAME, E.ADDRESS, E.BDATE
FROM PROJECT **AS** P, DEPARTMENT **AS** D, EMPLOYEE **AS** E
WHERE P.DNUM=D.DNUMBER **AND** D.MGRSSN=E.SSN
AND P.PLOCATION='STAFFORD'

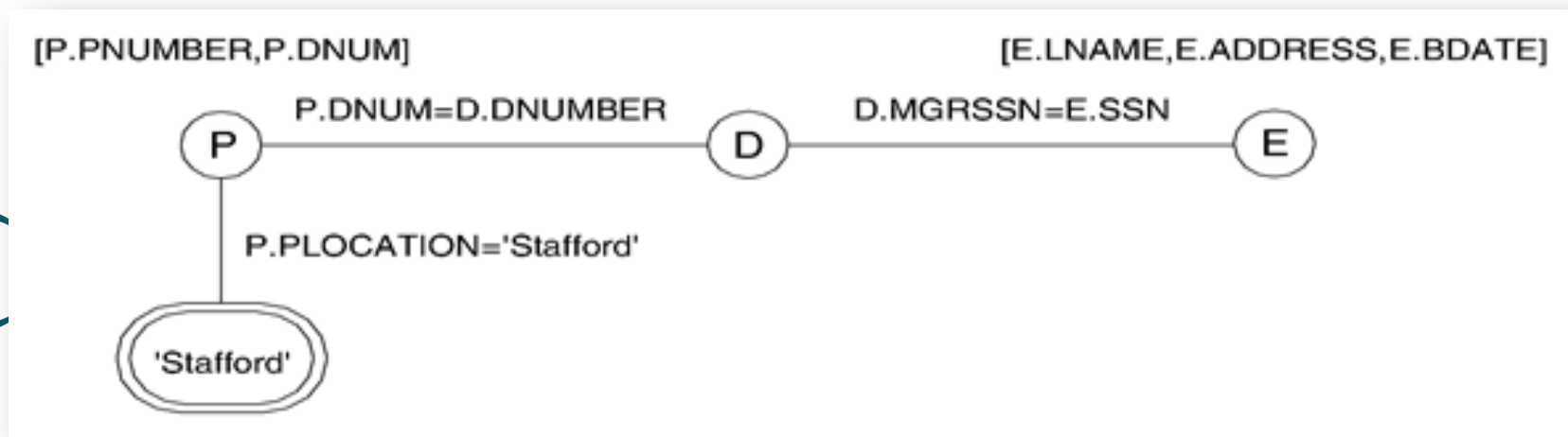
Ví dụ cây và đồ thị truy vấn (tt.)



Hai cây truy vấn cho Q2

Hình 4.4

Ví dụ cây và đồ thị truy vấn (tt.)



Đồ thị truy vấn cho Q2

Hình 4.5

Tối ưu hoá cây truy vấn dùng heuristic

* Vấn đề:

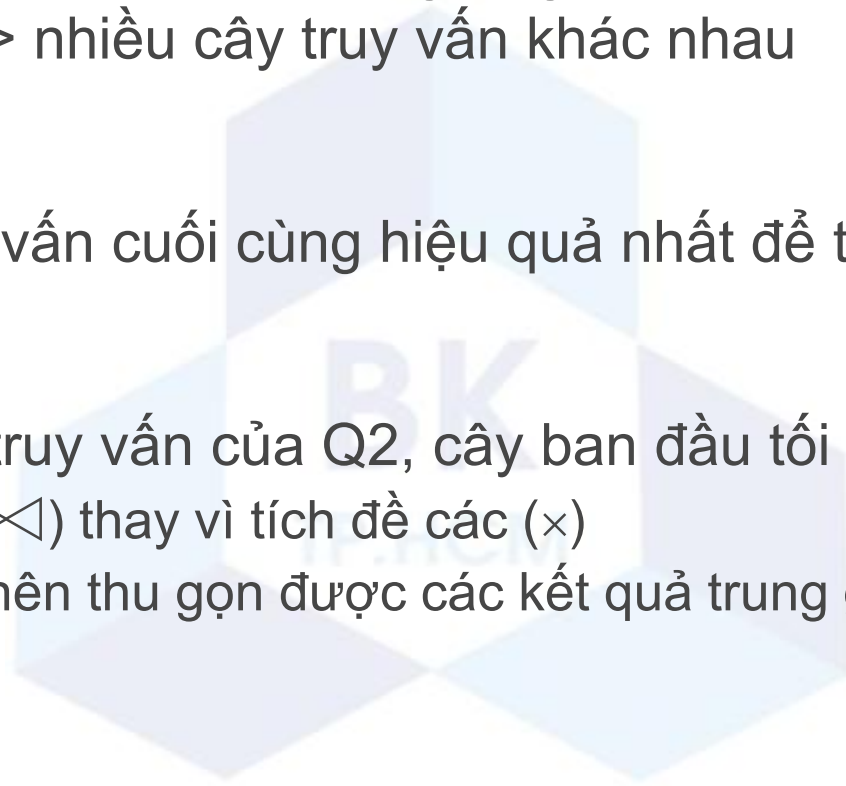
- Một câu truy vấn có thể tương ứng với nhiều biểu thức quan hệ khác nhau => nhiều cây truy vấn khác nhau

* Nhiệm vụ:

- Tìm cây truy vấn cuối cùng hiệu quả nhất để thi hành

* Ví dụ:

- Trong 2 cây truy vấn của Q2, cây ban đầu tối ưu hơn
 - Dùng kết (\bowtie) thay vì tích đề các (\times)
 - Chọn sớm nên thu gọn được các kết quả trung gian



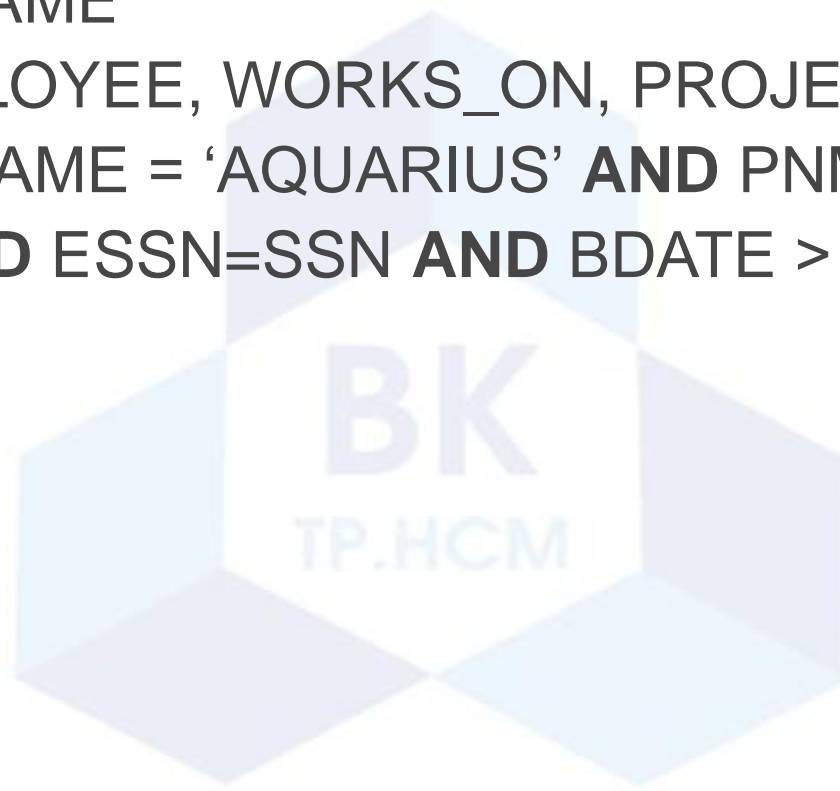
Ví dụ tối ưu hoá dùng heuristic

- * Tìm cây truy vấn tối ưu cho câu truy vấn sau:

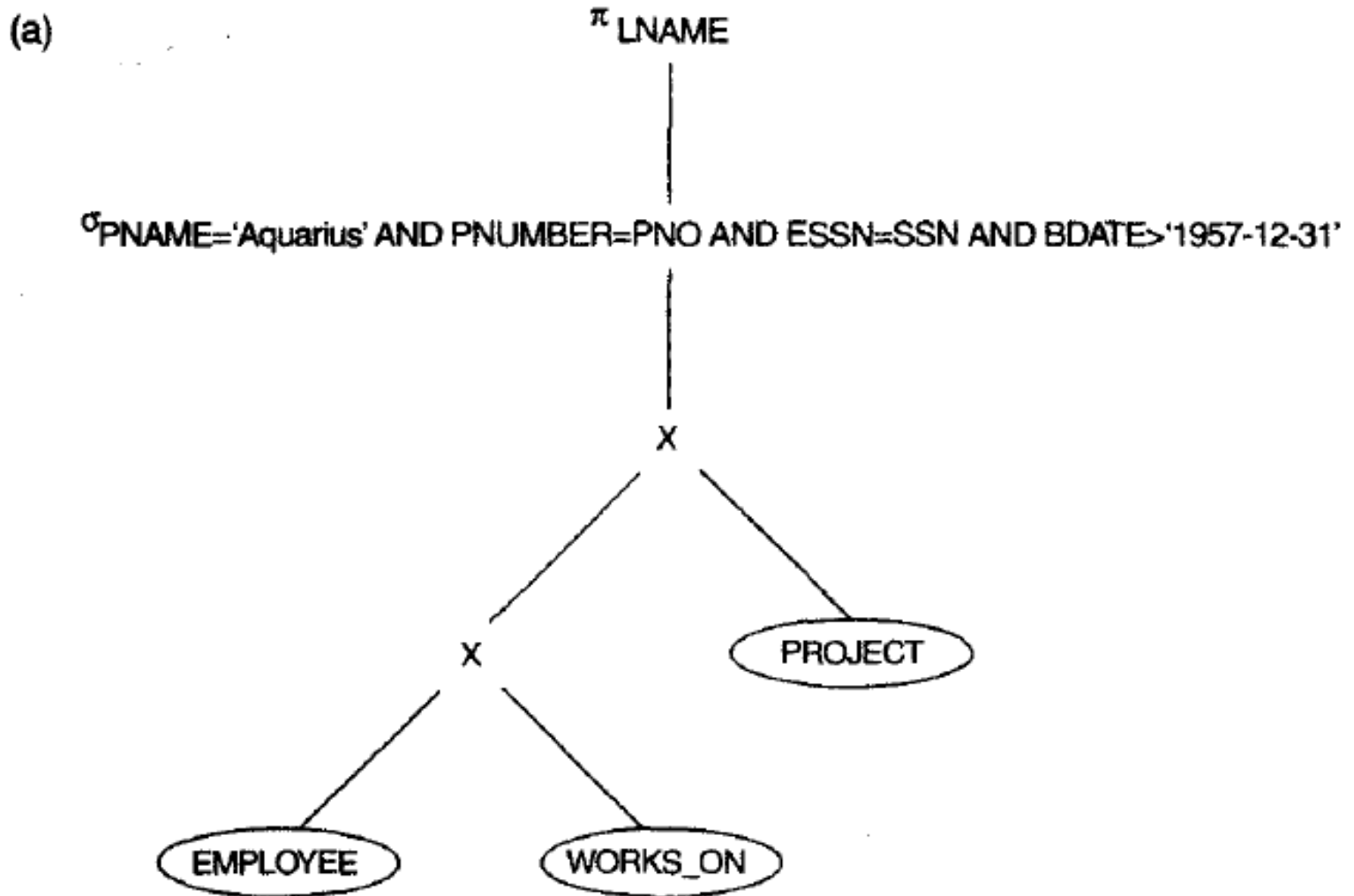
Q: SELECT LNAME

FROM EMPLOYEE, WORKS_ON, PROJECT

**WHERE PNAME = 'AQUARIUS' AND PNMUBER=PNO
AND ESSN=SSN AND BDATE > '1957-12-31';**



Ví dụ tối ưu hoá dùng heuristic (tt.)



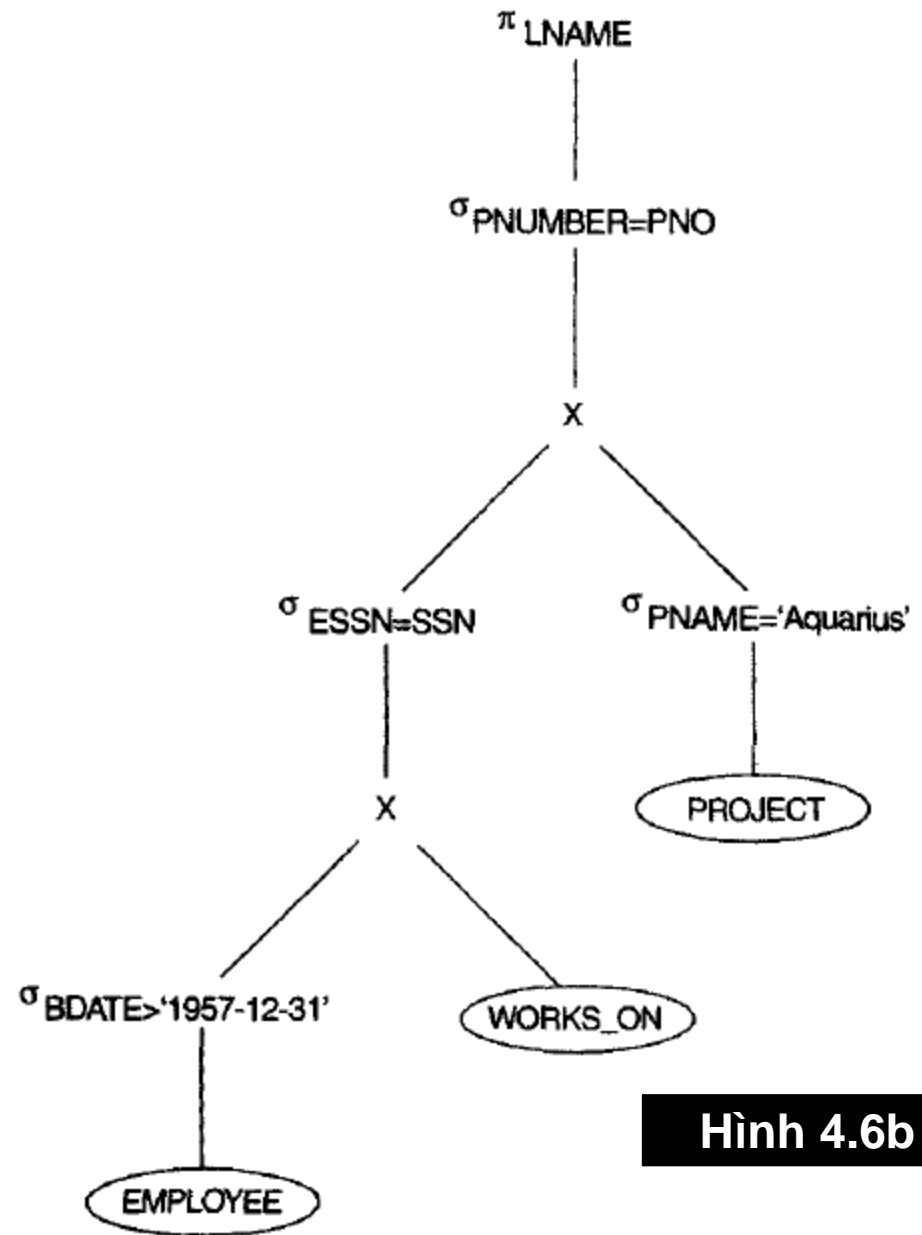
Hình 4.6a

Bước 1: tạo cây truy vấn ban đầu (Initial query tree)

Ví dụ tối ưu hoá dùng heuristic (tt.)

Bước 2: đưa các phép chọn xuống

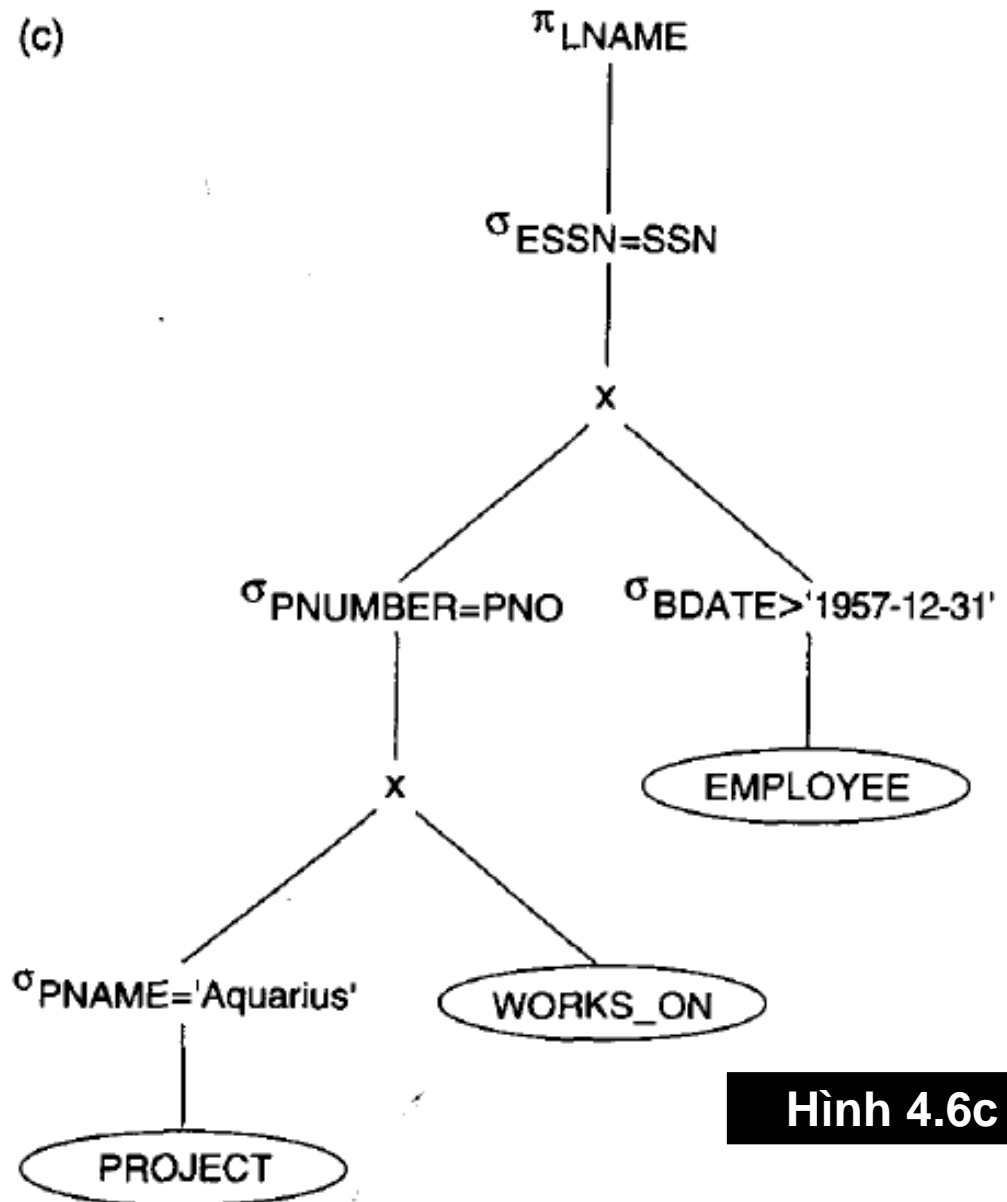
(b)



Hình 4.6b

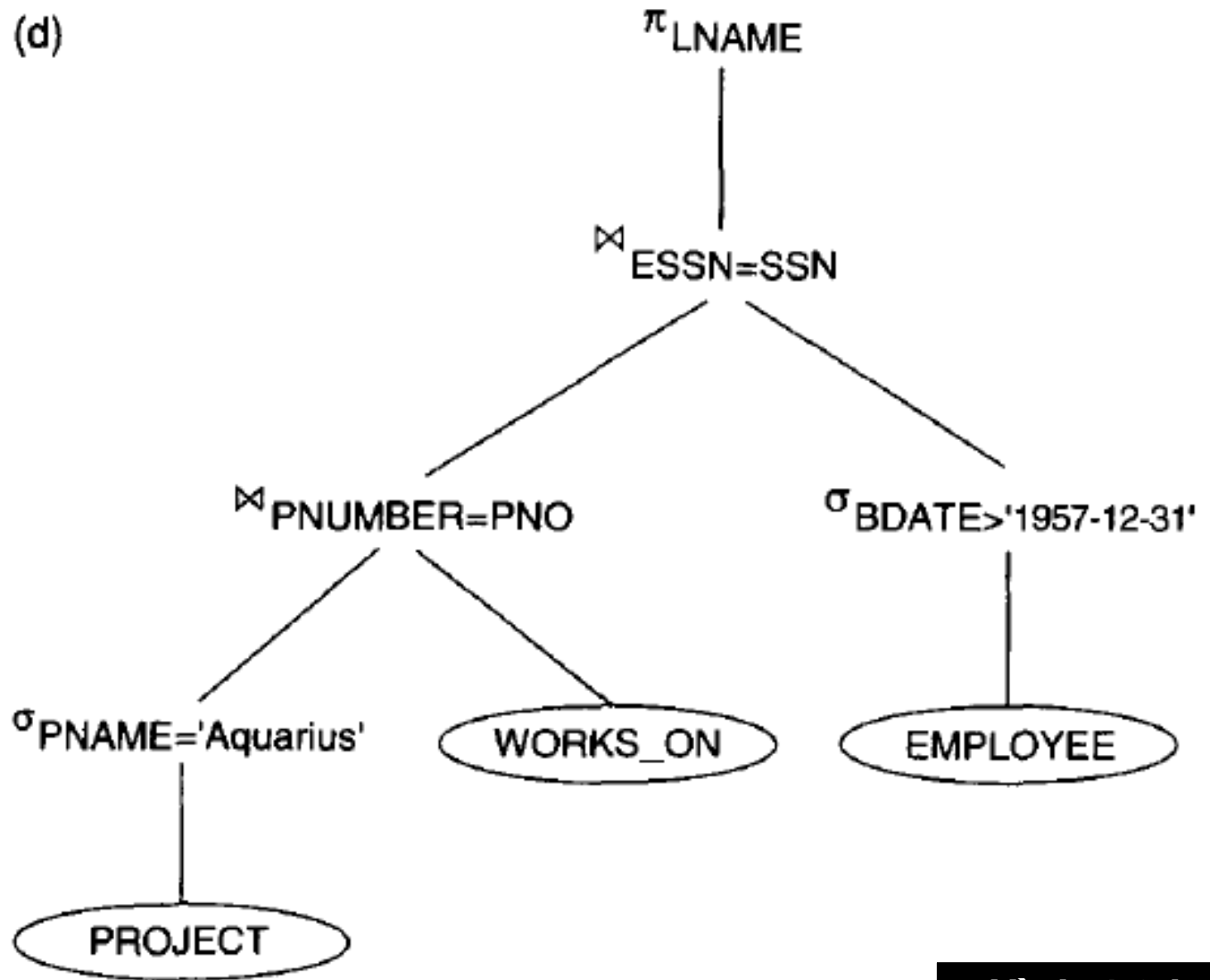
Ví dụ tối ưu hoá dùng heuristic (tt.)

Bước 3: thực hiện các phép chọn



Hình 4.6c

Ví dụ tối ưu hoá dùng heuristic (tt.)

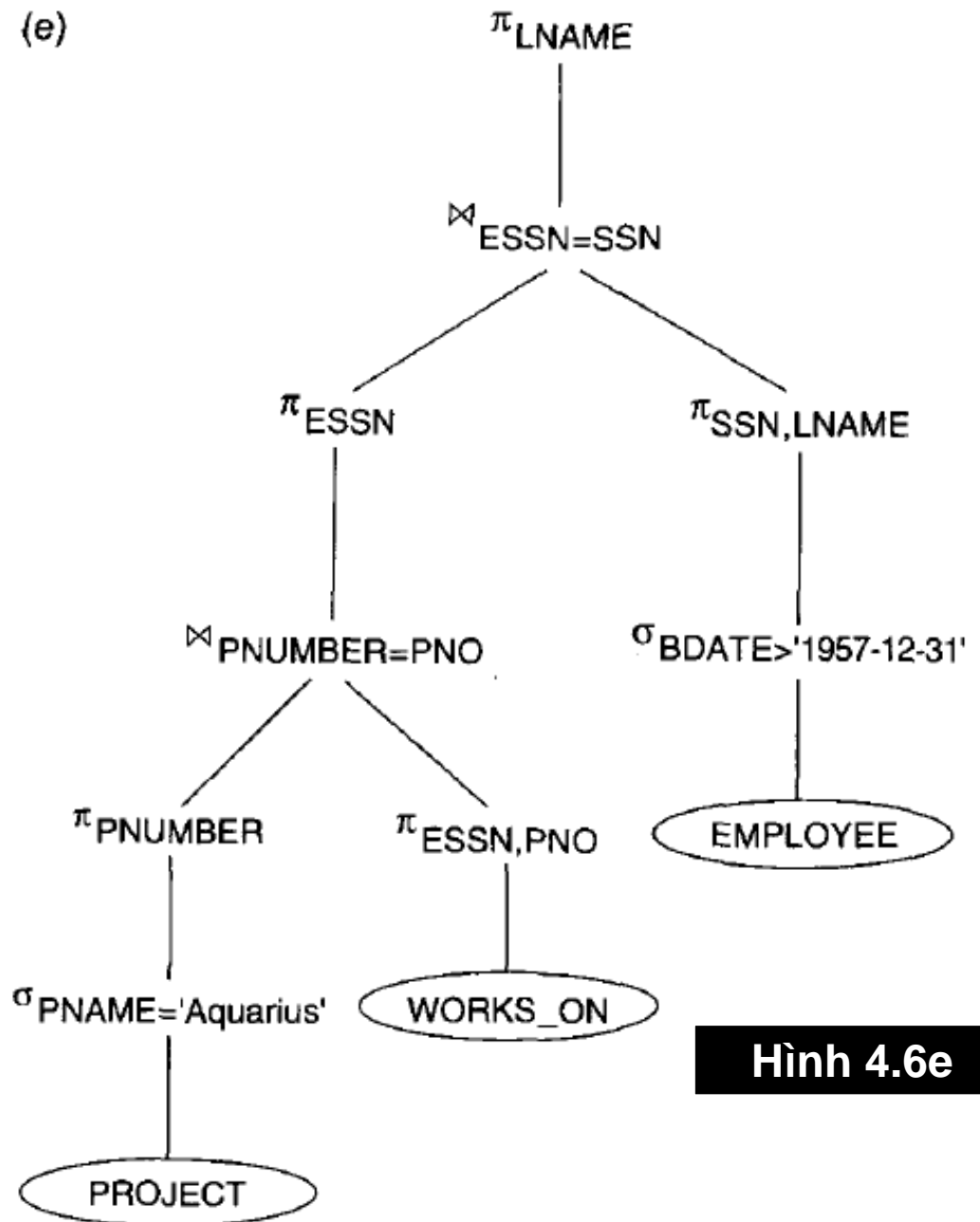


Hình 4.6d

Bước 4: thay thế phép tích đề các và phép chọn phù hợp thành phép kết

Ví dụ tối ưu hoá dùng heuristic (tt.)

Bước 5: chuyển các phép chiếu



Hình 4.6e

Các luật chuyển đổi cho các phép toán đại số quan hệ

- * 1. Tính dắt dây của phép chọn (Cascade of σ):

$$\sigma_{c1 \text{ AND } c2 \text{ AND } \dots \text{ AND } cn}(R) = \sigma_{c1}(\sigma_{c2}(\dots(\sigma_{cn}(R))\dots))$$

- * 2. Tính giao hoán của phép chọn (Commutativity of σ):

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

- * 3. Tính dắt dây của phép chiếu (Cascade of π):

$$\pi_{List1}(\pi_{List2}(\dots(\pi_{Listn}(R))\dots)) = \pi_{List1}(R)$$

Chú ý: $List1 \subseteq List2 \subseteq \dots \subseteq Listn$

- * 4. Tính giao hoán của phép chọn và phép chiếu:
 - Nếu phép chọn chỉ liên quan đến các thuộc tính chiếu

$$\pi_{A1, A2, \dots, An}(\sigma_c(R)) = \sigma_c(\pi_{A1, A2, \dots, An}(R))$$

Các luật chuyển đổi cho các phép toán đại số quan hệ (tt.)

- * 5. Tính giao hoán của phép kết (và tích đề các)

$$R \bowtie S = S \bowtie R; R \times S = S \times R$$

- * 6. Tính giao hoán của phép chọn và phép kết (hoặc tích đề các):

- Nếu tất cả các thuộc tính trong điều kiện chọn c chỉ liên quan đến các thuộc tính của một quan hệ trong phép kết, giả sử là R

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

- Nếu điều kiện chọn c có thể được tách ra thành $(c1 \text{ AND } c2)$, trong đó $c1$ chỉ liên quan đến R và $c2$ chỉ liên quan đến S

$$\sigma_c(R \bowtie S) = (\sigma_{c1}(R)) \bowtie (\sigma_{c2}(S))$$

Các luật chuyển đổi cho các phép toán đại số quan hệ (tt.)

- * 7. Giao hoán của phép chiếu và phép kết (hoặc tích đề các):
 - Giả sử rằng bộ thuộc tính chiếu $L = \{A_1, \dots, A_n, B_1, \dots, B_m\}$, trong đó A_i thuộc R và B_i thuộc S , và nếu điều kiện kết c chỉ liên quan đến các thuộc tính trong L

$$\pi_L(R \bowtie_c S) = (\pi_{A_1, \dots, A_n}(R)) \bowtie_c (\pi_{B_1, \dots, B_m}(S))$$

- Nếu trong c có một số thuộc tính không có trong L , giả sử là A_j, \dots, A_n thuộc R và B_k, \dots, B_m thuộc S , chúng có thể được thêm vào và loại bỏ ở bước cuối

$$\pi_L(R \bowtie_c S) = \pi_L((\pi_{A_1, \dots, A_n, A_j, \dots, A_n}(R)) \bowtie_c (\pi_{B_1, \dots, B_m, B_k, \dots, B_m}(S)))$$

Các luật chuyển đổi cho các phép toán đại số quan hệ (tt.)

- * 8. Tính giao hoán của một số phép toán tập hợp:
 - Phép hội (\cup) và phép giao (\cap) là giao hoán
 - Phép hiệu ($-$) không giao hoán
- * 9. Tính kết hợp của phép kết (\bowtie), tích đề các (\times), hội (\cup) và giao (\cap):
 - Giả sử θ là một trong các phép toán trên

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

- * 10. Tính giao hoán của phép chọn đối với phép toán tập hợp:
 - Giả sử θ là một trong các phép toán tập hợp

$$\sigma_c(R \theta S) = (\sigma_c(R)) \theta (\sigma_c(S))$$

Các luật chuyển đổi cho các phép toán đại số quan hệ (tt.)

- * 11. Tính giao hoán của phép chiếu và phép hội

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

- * 12. Chuyển một bộ liên tiếp phép chọn và tích đề các (σ , \times) thành phép kết \bowtie :

➤ Nếu điều kiện c của phép chọn theo sau một phép tích đề các tương đương một điều kiện kết

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

- * 13. Ngoài ra còn một số luật chuyển đổi khác

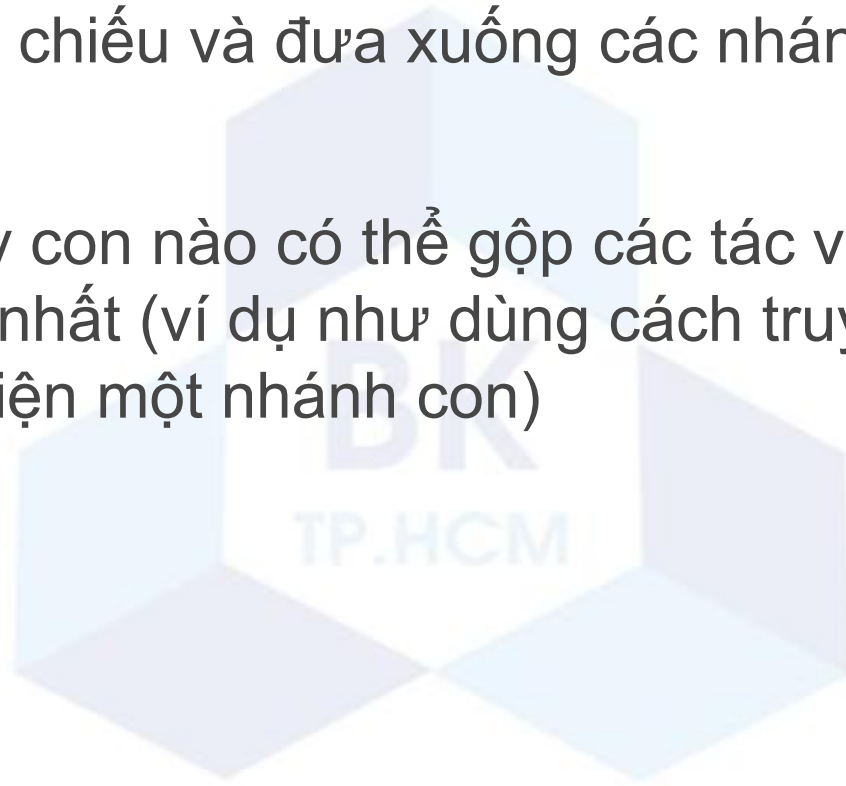
➤ Ví dụ: $\text{NOT } (c1 \text{ AND } c2) = (\text{NOT } c1) \text{ OR } (\text{NOT } c2)$

Giải thuật tối ưu hoá dùng heuristic trên

- * 1. Dùng luật 1 chuyển các điều kiện chọn giao thành các điều kiện chọn đơn dặt dây
- * 2. Dùng luật 2, 4, 6 và 10 về tính giao hoán của phép chọn với các phép toán khác để đưa các phép chọn xuống các nhánh con sâu đến mức có thể (chú ý các thuộc tính của các quan hệ trong cây truy vấn)
- * 3. Dùng luật 9 về tính hoán vị của các phép toán nhị nguyên (binary operation) để sắp xếp các nhánh lá của cây sao cho nút lá nào có các phép chọn có kết quả trung gian nhỏ nhất sẽ được thực thi trước trên cây (\Rightarrow đưa qua trái)
- * 4. Dùng luật 12 chuyển các tích đề các cùng các phép chọn phù hợp thành các phép kết

Giải thuật tối ưu hoá dùng heuristic trên (tt.)

- * 5. Dùng các luật 3, 4, 7 và 11 về tính dắt dây của phép chiếu và tính giao hoán của phép chiếu với các phép toán khác để chia nhỏ phép chiếu và đưa xuống các nhánh con càng sâu càng tốt
- * 6. Tìm các cây con nào có thể gộp các tác vụ lại trong một giải thuật duy nhất (ví dụ như dùng cách truyền theo đường ống để thực hiện một nhánh con)



Tóm tắt phương pháp tối ưu hoá dùng heuristic

- * 1. Heuristic chính là áp dụng trước các tác vụ thu gọn kích thước của kết quả trung gian
- * 2. Thi hành các tác vụ chọn sớm đến mức có thể để thu gọn số dòng dữ liệu và thi hành các tác vụ chiếu sớm đến mức có thể để thu gọn số thuộc tính của dữ liệu (trung gian)
 - Đẩy các tác vụ xuống càng sâu càng tốt
- * 3. Các tác vụ chọn và kết cho ra ít kết quả cần phải được thi hành trước các tác vụ tương tự
 - Thay đổi thứ tự thực thi các nút lá trong cây

Tóm tắt phương pháp tối ưu hoá dùng heuristic (tt.)

- * Kế hoạch thực thi truy vấn (query execution plan):
 - Một kế hoạch thực thi cho một câu truy vấn đại số quan hệ gồm một kết hợp của cây truy vấn đại số quan hệ và các thông tin về các đường đi truy đạt (đến các quan hệ tham gia vào truy vấn) và các phương pháp dùng để tính toán các tác vụ quan hệ trong cây
- * Sự định trị vật chất hoá (materialized evaluation):
 - Kết quả của một tác vụ được lưu thành các quan hệ tạm (temporary relation)
- * Dùng kỹ thuật truyền theo đường ống (Pipelining):
 - Đưa từng (dòng) kết quả của một tác vụ sang tác vụ kế tiếp thay vì đợi tác vụ trước hoàn thành

Tóm tắt bài 1

- * Giới thiệu chung
- * Chuyển đổi các truy vấn SQL thành đại số quan hệ
- * Các giải thuật xếp thứ tự ngoại
 - Sort-merge
- * Dùng các heuristic để tối ưu hóa truy vấn
 - Phương pháp
 - Các luật chuyển đổi các phép toán đại số quan hệ

