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

Bài 2 – Các giải thuật hiện thực các phép toán đại số quan hệ

Bùi Hoài Thắng

PHẦN 3. CÁC GIẢI THUẬT CHO PHÉP CHỌN

Một số phép chọn (SELECT) cơ bản

* CSDL dùng minh hoạ cho phần này:

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATIONS

DNUMBER	DLOCATION

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM

WORKS_ON

ESSN	PNO	HOURS

Hình 4.7

Một số phép chọn cơ bản (tt.)

Ta xem xét các ví dụ lựa chọn sau

* (OP1): $\sigma_{SSN='123456789'}(EMPLOYEE)$
* (OP2): $\sigma_{DNUMBER>5}(DEPARTMENT)$
* (OP3): $\sigma_{DNO=5}(EMPLOYEE)$
* (OP4): $\sigma_{DNO=5\ AND\ SALARY>30000\ AND\ SEX=F}(EMPLOYEE)$
* (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}(WORKS\_ON)$

Phương pháp cho các điều kiện chọn đơn giản

* S1. Phép tìm tuần tự (linear search) (vét cạn - brute force):
  - Truy xuất lần lượt tất cả các bản ghi trong tập tin,
  - Kiểm tra các thuộc tính của bản ghi có thoả mãn điều kiện chọn (selection condition) hay không
  - Ví dụ:
    - (OP1): $\sigma_{SSN='123456789'}(EMPLOYEE)$
    - (OP2): $\sigma_{DNUMBER>5}(DEPARTMENT)$
    - (OP3): $\sigma_{DNO=5}(EMPLOYEE)$
    - (OP4): $\sigma_{DNO=5\ AND\ SALARY>30000\ AND\ SEX=F}(EMPLOYEE)$
    - (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}(WORKS\_ON)$

Phương pháp cho các điều kiện chọn đơn giản (tt.)

* S2. Phép tìm nhị phân (binary search):
  - Khi điều kiện chọn là một phép so sánh bằng (equality comparison) trên thuộc tính khoá của tập tin có thứ tự (theo thuộc tính khoá này)
  - Hiệu quả hơn phép tìm tuần tự
  - Ví dụ:
    - (OP1): $\sigma_{SSN='123456789'}(EMPLOYEE)$
* S3. Sử dụng chỉ mục sơ cấp hoặc phép băm:
  - Khi điều kiện chọn là một phép so sánh bằng trên thuộc tính khoá có chỉ mục sơ cấp hoặc tập tin băm trên khoá này
  - Ví dụ:
    - (OP1): $\sigma_{SSN='123456789'}(EMPLOYEE)$

Phương pháp cho các điều kiện chọn đơn giản (tt.)

* S4. Dùng chỉ mục sơ cấp để truy xuất nhiều bản ghi:
  - Khi điều kiện chọn là các phép so sánh >, ≥ , <, hoặc ≤ trên mục tin chỉ mục sơ cấp
  - Dùng chỉ mục tìm bản ghi đầu tiên thoả điều kiện =
  - Truy xuất tất cả các bản ghi về cuối (phép > hoặc ≥) hoặc đầu (phép < hoặc ≤) tập tin (có thứ tự)
  - Loại bản ghi đầu tiên nếu phép so sánh không có điều kiện =
  - Ví dụ:
    - (OP2): $\sigma_{DNUMBER>5}(DEPARTMENT)$

Phương pháp cho các điều kiện chọn đơn giản (tt.)

* S5. Dùng chỉ mục cụm để truy xuất nhiều bản ghi:
  - Khi điều kiện chọn là phép so sánh bằng trên mục tin chỉ mục cụm không (là) khoá
  - Ví dụ:
    - (OP3): $\sigma_{DNO=5}(EMPLOYEE)$

Phương pháp cho các điều kiện chọn đơn giản (tt.)

* S6. Dùng chỉ mục thứ cấp (cây B+):
  - Dùng cây B+ cho điều kiện chọn của phép so sánh bằng sẽ truy xuất một bản ghi (nếu mục tin chỉ mục là duy nhất) hoặc nhiều bản ghi (nếu mục tin chỉ mục không duy nhất)
  - Có thể dùng cây B+ để truy xuất nhiều bản ghi cho các điều kiện >,>=, <, or <=
    - Mức lá là danh sách liên kết nên có thể truy xuất như trên tập tin có thứ tự
  - Các ví dụ:
    - (OP1): $\sigma_{SSN='123456789'}(EMPLOYEE)$
    - (OP2): $\sigma_{DNUMBER>5}(DEPARTMENT)$
    - (OP3): $\sigma_{DNO=5}(EMPLOYEE)$

Phương pháp cho các điều kiện chọn phức tạp

* S7. Phép chọn giao (Conjunctive selection):
  - Nếu một điều kiện chọn trong biểu thức điều kiện chọn giao có thể dùng được một trong các phương pháp từ S2 đến S6, dùng phương pháp đó để truy xuất các bản ghi
  - Kiểm tra nếu bản ghi thoả các điều kiện còn lại
  - Ví dụ:
    - (OP4): $\sigma_{DNO=5\ AND\ SALARY>30000\ AND\ SEX=F}(EMPLOYEE)$
    - (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}(WORKS\_ON)$
* S8. Dùng chỉ mục tổ hợp (composite index):
  - Khi điều kiện chọn là phép so sánh bằng và phù hợp với một chỉ mục tổ hợp (hoặc phép băm) đã có, dùng chỉ mục tổ hợp
  - Ví dụ:
    - (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}(WORKS\_ON)$

Phương pháp cho các điều kiện chọn phức tạp (tt.)

* S9. Giao các con trỏ bản ghi cho điều kiện chọn giao:
  - Nếu một số thuộc tính trong điều kiện chọn (phép so sánh bằng) có chỉ mục thứ cấp tồn tại và các chỉ mục thứ cấp dùng con trỏ bản ghi (chứ không phải con trỏ khối) thì dùng các chỉ mục để tìm tập các con trỏ bản ghi thoả từng điều kiện thành phần
  - Giao các tập con trỏ bản ghi để tìm tập con trỏ bản ghi thoả tất cả các điều kiện thành phần
  - Truy xuất các bản ghi dùng tập con trỏ bản ghi
  - Kiểm tra các bản ghi này có thoả các điều kiện thành phần còn lại không
  - Ví dụ:
    - (OP4): $\sigma_{DNO=5\ AND\ SALARY>30000\ AND\ SEX=F}(EMPLOYEE)$
    - (OP5): $\sigma_{ESSN=123456789\ AND\ PNO=10}(WORKS\_ON)$

Phương pháp cho các điều kiện chọn phức tạp (tt.)

* Điều kiện chọn hội (disjunctive selection conditions):
  - Giống phép toán hội (UNION)

PHẦN 4. CÁC GIẢI THUẬT CHO PHÉP KẾT, CHIẾU, TẬP HỢP

Các phép kết (JOIN) cơ bản

* Tập trung chủ yếu vào phép kết bằng (equi-join) hoặc phép kết tự nhiên (natural join)
* Có 2 dạng kết:
  - Kết 2 chiều (two–way join): kết 2 tập tin
    - Ví dụ: $R \bowtie_{A=B} S$
  - Kết đa chiều (multi-way join): kết nhiều hơn một tập tin
    - Ví dụ: $R \bowtie_{A=B} S \bowtie_{C=D} T$
* Ví dụ:
  - (OP6): $EMPLOYEE \bowtie_{DNO=DNUMBER} DEPARTMENT$
  - (OP7): $DEPARTMENT \bowtie_{MGRSSN=SSN} EMPLOYEE$

Các giải thuật cho phép kết (tt.)

* $R \bowtie_{A=B} S$
* J1. Kết lặp-lồng (Nested-loop join):
  - Ứng với mỗi bản ghi t trên R truy xuất tất cả bản ghi s trên S và kiểm tra nếu hai bản ghi thoả điều kiện kết (join condition) t[A] = s[B].

```
for each block b_R in R
    for each block b_S in S
        for each t in b_R
            for each s in b_S
                if t[A] = s[B] then
                    output <t,s>
```

Các giải thuật cho phép kết (tt.)

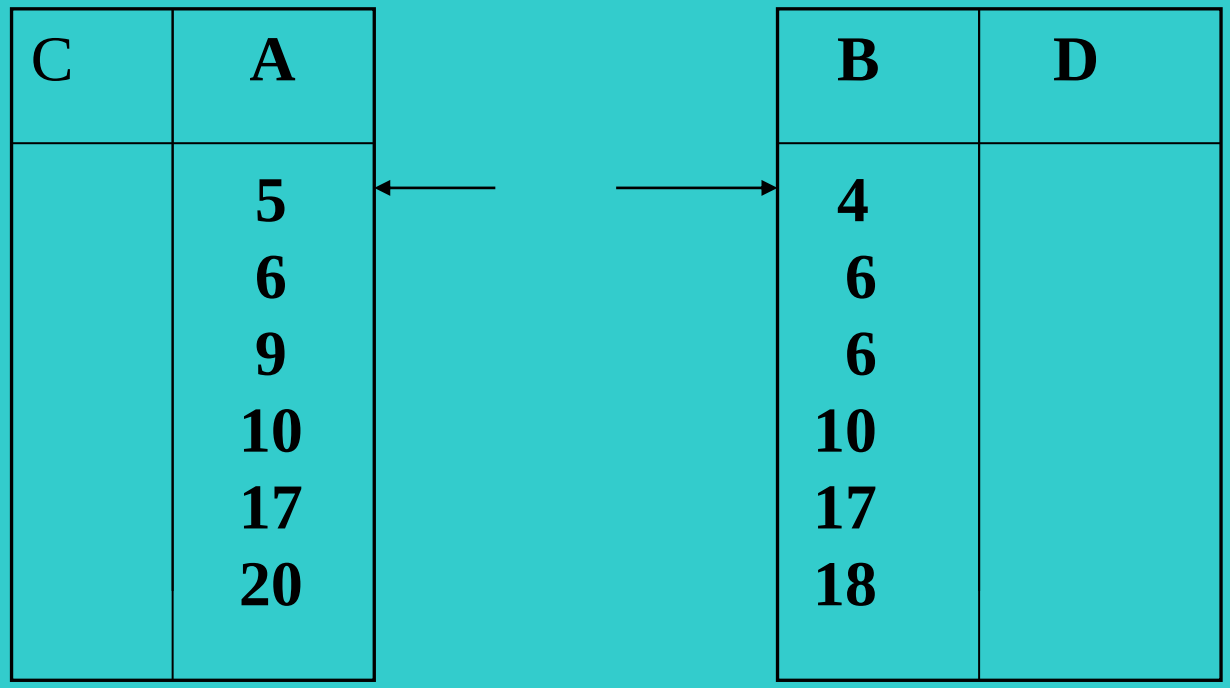
* $R \bowtie_{A=B} S$
* J2. Kết lặp-đơn (Single-loop join)
  - Nếu một trong hai thuộc tính kết có chỉ mục (hoặc tập tin băm trên thuộc tính đó), giả sử B trên S, ứng với mỗi bản ghi t trên R, truy xuất các bản ghi tương ứng s trên S so cho s[B] = t[A]

```
for each block b_R in R
  for each t in b_R
     find s in S such that index entry s[B] = t[A]
     if found then
        output <t,s>
```

Các giải thuật cho phép kết (tt.)

* J3. Kết xếp-và-trộn (Sort-merge join):
  - Khi các bản ghi trên R và S đã được sắp thứ tự theo các thuộc tính kết (A và B tương ứng)
  - Quét R và S đồng thời để truy xuất các bản ghi tương ứng kết được với nhau

Các giải thuật cho phép kết (tt.)



Sort-merge join

Hình 4.8

Các giải thuật cho phép kết (tt.) (*)

```
sort the tuples in R on attribute A;  /* assume R has n tuples  */
sort the tuples in S on attribute B;  /* assume S has m tuples  */
set  i ← 1, j ← 1;
while (i ≤ n) and (j ≤ m)
do { if R(i)[A] > S[j][B] then set  j ← j + 1
     elseif R(i)[A] < S[j][B] then set  i ← i + 1
     else { /* output a matched tuple  */
            output the combined tupe <R(i), S(j)> to T;
            /* output other tuples that match R(i), if any */
            set  j1 ← j + 1 ;
            while ( j1 ≤ m) and (R(i)[A] = S[j1][B])
            do { output the combined tuple <R(i), S[j1]> to T;
                 set  j1 ← j1 + 1 }
            /* output other tuples that match S(j), if any */
            set  i1 ← i+1
            while ( i1 ≤ n) and (R(i1)[A] = S[j][B])
            do { output the combined tuple <R(i1), S[j]> to T;
                 set  i1 ← i1 + 1 }
            set i ← i1, j ← j1;
          }
   }
```

(a) Sort-merge join $T \leftarrow R \bowtie_{A=B} S$

Các giải thuật cho phép kết (tt.)

* J4. Kết băm (Hash-join) (*):
  - Các bản ghi của R và S sẽ được chứa vào cùng một tập tin băm và dùng cùng một hàm băm trên thuộc tính A và B tương ứng.
  - Bước 1: băm tất cả các bản ghi của tập tin ít bản ghi hơn, giả sử R, vào các thùng trong tập tin băm
  - Bước 2: băm tất cả các bản ghi của S vào các thùng tương ứng và kết bản ghi từ S với các bản ghi tương ứng của R trong cùng thùng

Các giải thuật cho phép kết (tt.)

* Nhận xét:
  - Các tập tin phân dữ liệu thành từng khối
  - Các giải thuật kết phải làm việc trên khối để kết

Giải thuật cho phép chiếu (PROJECTION)

* $\Pi_{<\text{attribute list}>}(R)$
* Phép chiếu sẽ loại bỏ các hàng trùng nhau
* Giải thuật:
  - Tạo các bản ghi chỉ lấy các thuộc tính được chiếu từ R
  - Nếu bộ thuộc tính không chứa khoá chính
    - Sắp xếp
    - Duyệt và lấy các bản ghi không trùng nhau
    - Hoặc dùng bảng băm để lấy các giá trị không trùng nhau

Giải thuật cho phép chiếu (tt.) (*)

```
(b) create a tuple t[<attribute list>] in T' for each tuple t in R;
    /* T' contains the projection result before duplicate
elimination */
    if <attribute list> includes a key of R
    then T ← T'
    else  { sort the tuples in T';
            set i ← 1, j ← 2;
            while i ≤ n
             do { output the tuple T'[i] to T;
                  while T'[i] = T[j] and j≤ n do j ← j+1;
                  set i ← j, j ← j+1;
                }
          }
/* T' contains the projection result after duplicate
elimination */
```

$T \leftarrow \prod_{<attribute\ list>}(R)$

Giải thuật cho các phép toán tập hợp (SET) (*)

* Các phép toán tập hợp:
  - hội – UNION
  - giao – INTERSECTION
  - hiệu - SET DIFFERENCE
  - tích đề các - CARTESIAN PRODUCT
* Ngoại trừ trong phép tích đề các, trong các phép toán tập hợp còn lại bộ thuộc tính của hai tập tin dữ liệu phải tương đương nhau (cả về thứ tự xuất hiện)

  R(A1, A2, …, An)

  UNION/ INTERSECT/ SET DIFFERENCE

  S(A1, A2, …, An)

Giải thuật cho các phép toán tập hợp (tt.)

* Tích đề các (CARTESIAN PRODUCT) của R và S ($R \times S$)
  - Chứa tất cả các bộ dữ liệu gồm thuộc tính từ cả R và S và là sự kết hợp của các bản ghi của R và S
  - Nếu R có n bản ghi và j thuộc tính và S có m bản ghi và k thuộc tính, thì kết quả có n*m bản ghi và j+k thuộc tính
  - Tác vụ này rất tốn kém và cần phải tránh sử dụng tối đa

Giải thuật cho các phép toán tập hợp (tt.) (*)

* Hội – (UNION) $R \cup S$
  - 1. Sắp xếp R và S dùng cùng một thứ tự thuộc tính
  - 2. Quét và trộn (scan and merge) cả R và S đồng thời. Nếu có các bản ghi trùng nhau thì chỉ giữ lại một bản ghi trong kết quả
* Giao (INTERSECTION) $R \cap S$
  - 1. Sắp xếp R và S dùng cùng một thứ tự thuộc tính
  - 2. Quét và trộn R và S đồng thời, giữ lại những bản ghi nào xuất hiện trong cả R và S
* Hiệu (SET DIFFERENCE) R-S
  - 1. Sắp xếp R và S dùng cùng một thứ tự thuộc tính
  - 2. Quét và trộn R và S đồng thời, giữ lại những bản ghi nào xuất hiện trong R và không xuất hiện trong S

Giải thuật cho các phép toán tập hợp (tt.) (*)

```
    sort the tuples in R and S using the same unique
sort attributes;
    set i ← 1, j ← 1;
    while (i ≤ n) and (j ≤ m) do
    {
        if R(i) > S(j)
        then
        {   output S(j) to T;
            set j ← j+1
        }
        elseif R(i) < S(j)
        then
        {    output R(i) to T;
             set i ← i+1
        }
        else set j← j+1  /* R(i) = S(j), so we skip one of
the duplicate tuples  */
    }
   if (i ≤ n) then add tuples from R(i) to R(n) to T;
   if (j ≤ m) then add tuples from S(i) to S(m) to T;
```

Union: $T \leftarrow R \cup S$

Giải thuật cho các phép toán tập hợp (tt.) (*)

```
    sort the tuples in R and S using the same unique
sort attributes;
    set i ← 1, j ← 1;
    while (i ≤ n) and (j ≤ m) do
    {
        if R(i) > S(j)
        then
           set j ← j+1
        elseif R(i) < S(j)
        then
           set i ← i+1
        else
        {  output R(i) to T;  /* R(i) = S(j), so we skip one
of the duplicate tuples  */
            set  i ← i+1, j← j+1
        }
    }
```

Intersection $T \leftarrow R \cap S$

Giải thuật cho các phép toán tập hợp (tt.) (*)

```
    sort the tuples in R and S using the same unique
sort attributes;
    set i ← 1, j ← 1;
    while (i ≤ n) and (j ≤ m) do
    {
        if R(i) > S(j)
        then
            set j ← j+1
        elseif R(i) < S(j)
        then
       {  output R(i) to T;  /* R(i) has no matching S(j),
so output R(i)  */
            set i ← i+1
        }
        else
            set  i ← i+1, j← j+1
     }
    if (i ≤ n) then add tuples R(i) to R(n) to T;
```

Difference T ← R – S

Ví dụ về phép hiệu (*)

NV

S#	SName
s1	NV 1
s2	NV 2
s3	NV 3
s4	NV 4
s5	NV 5
s6	NV 6
s7	NV 7

NV1

S#	SName
s2	NV 2
s5	NV 5
s6	NV 6

NV – NV1

S#	SName
s1	NV 1
s3	NV 3
s4	NV 4
s7	NV 7

Hình 4.9

Giải thuật cho các phép toán gộp (Aggregation)

- Các toán tử gộp (Aggregate operators) thường dùng : MIN, MAX, SUM, COUNT và AVG
- Có thể dùng các quét trên tập tin dữ liệu (Table Scan) hoặc chỉ mục (index) để thực hiện các tác vụ này
- Ví dụ:
  - SELECT MAX(SALARY) FROM EMPLOYEE;
  - Nếu tập tin dữ liệu được tạo chỉ mục (tăng dần) trên thuộc tính SALARY -> phần tử lớn nhất chính là phần tử cuối cùng trên chỉ mục
    - Nếu là cây B, dùng các con trỏ phải cùng (right-most pointer) trên các nút của cây

Giải thuật cho các phép toán gộp (tt.)

* SUM, COUNT và AVG
  - Nếu có chỉ mục dày (dense index) (mỗi bản ghi có một mục chỉ mục): tính trên mỗi mục chỉ mục
  - Nếu có chỉ mục thưa (non-dense index): số bản ghi thực sự cho mỗi mục chỉ mục cần phải được tính toán
* Có mệnh đề GROUP BY:
  - Các toán tử cần phải được tính riêng cho từng nhóm
  - 1. Sắp xếp hoặc băm (hashing) trên các thuộc tính nhóm để phân hoạch các bản ghi thành từng nhóm
  - 2. Tính toán trên từng nhóm
* Nếu có chỉ mục cụm (clustering index) trên thuộc tính nhóm?
  - …

Giải thuật cho các phép kết ngoài (Outer Joins)

* Các phép kết ngoài:
  - Kết (ngoài) trái - LEFT OUTER JOIN
  - Kết (ngoài) phải - RIGHT OUTER JOIN
  - Kết ngoài đầy đủ - FULL OUTER JOIN.
    - Tương đương với phép hội của phép kết trái và phép kết phải

* Ví dụ:

```
SELECT  DISTINCT FNAME, DNAME
FROM  EMPLOYEE  LEFT OUTER JOIN  DEPARTMENT
        ON DNO = DNUMBER
```

Giải thuật cho các phép kết ngoài (tt.)

* Cách hiệu chỉnh các giải thuật kết cho phép kết ngoài
  - Kết ngoài trái (left outer join): dùng tập tin phía trái làm quan hệ ngoài (outer relation) trong vòng lặp ngoài (outer loop) và kết với từng bản ghi của tập tin phía phải. Nếu không tìm thấy bản ghi của tập tin phải nào thoả thì nối thêm các giá trị null để tạo thành bản ghi kết quả
  - Kết ngoài phải (right outer join) và kết ngoài đầy đủ (full outer join) cũng tương tự

Giải thuật cho các phép kết ngoài (tt.)

* Cách dùng nhiều phép toán:
  - 1. {Kết EMPLOYEE và DEPARTMENT}

    $TEMP1 \leftarrow \pi_{FNAME,DNAME}(EMPLOYEE \bowtie_{DNO=DNUMBER} DEPARTMENT)$
  - 2. {Tìm các bản ghi trong EMPLOYEE không kết được}

    $TEMP2 \leftarrow \pi_{FNAME}(EMPLOYEE) - \pi_{FNAME}(Temp1)$
  - 3. {Thêm vào mỗi dòng trong TEMP2 mục tin DNAME là NULL}

    $TEMP2 \leftarrow TEMP2 \times$ 'null'
  - 4. {Hội hai kết quả tạm để có kết quả LEFT OUTER JOIN}

    $RESULT \leftarrow TEMP1 \cup TEMP2$

Chú ý: cách này chưa phải là trường hợp tổng quát. Ví dụ: khi bảng EMPLOYEE có chứa mục tin DNO không có trong DEPARTMENT

Ví dụ phép kết ngoài

EMPLOYEE

SSN	FNAME	DNO
s1	NV 1	p1
s2	NV 2	p2
s3	NV 3	p3
s4	NV 4	p3
s5	NV 5	
s6	NV 6	

DEPARTMENT

DNUMBER	DNAME
p1	Phòng 1
p2	Phòng 2
p3	Phòng 3
p4	Phòng 4

$TEMP1 \leftarrow \pi_{FNAME,\ DNAME}(EMPLOYEE \bowtie_{DNO=DNUMBER} DEPARTMENT)$

FNAME	DNAME
NV 1	Phòng 1
NV 2	Phòng 2
NV 3	Phòng 3
NV 4	Phòng 3

$TEMP2 \leftarrow (\pi_{FNAME}(EMPLOYEE) - \pi_{FNAME}(TEMP1)) \times$ 'null'

FNAME	'null'
NV 5	(null)
NV 6	(null)

$RESULT \leftarrow TEMP1 \cup TEMP2$

FNAME	DNAME
NV 1	Phòng 1
NV 2	Phòng 2
NV 3	Phòng 3
NV 4	Phòng 3
NV 5	(null)
NV 6	(null)

Hình 4.10

Kỹ thuật truyền theo đường ống (Pipelining)

* Vấn đề:
  - Một câu truy vấn là một chuỗi các tác vụ
  - Mỗi tác vụ tạo ra một kết quả tạm (trung gian)
  - Tạo và lưu kết quả tạm trên đĩa tốn thời gian và chi phí
* Một giải pháp:
  - Tránh tạo ra các dữ liệu tạm đến mức có thể
  - Dùng kỹ thuật truyền theo đường ống qua các tác vụ kế tiếp nhau mà không cần đợi tác vụ trước hoàn thành mới thực hiện tác vụ sau

Kỹ thuật truyền theo đường ống (tt.)

* Ví dụ:
  - Cần thực hiện một phép kết, 2 phép chọn trên 2 tập tin nguồn, và cuối cùng là một phép chiếu trên kết quả của phép kết.
* Thực hiện:
  - Khi thực hiện kết, ứng với từng lần thực hiện:
    - đồng thời thực hiện phép chọn trên từng bản ghi của 2 tập tin nguồn
    - nếu kết được, chiếu trên bản ghi kết quả tạm này
    - Xuất kết quả cuối cùng
  - Như vậy, thay vì tạo ra 4 tập tin trung gian, chỉ có một tập tin kết quả
* Cần phải sửa mã trong các giải thuật
* Còn gọi là *xử lý dựa trên dòng* (stream-based processing)

Tóm tắt bài 2

* 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