Query Processing and Optimization

Dr. L.M. Jenila Livingston

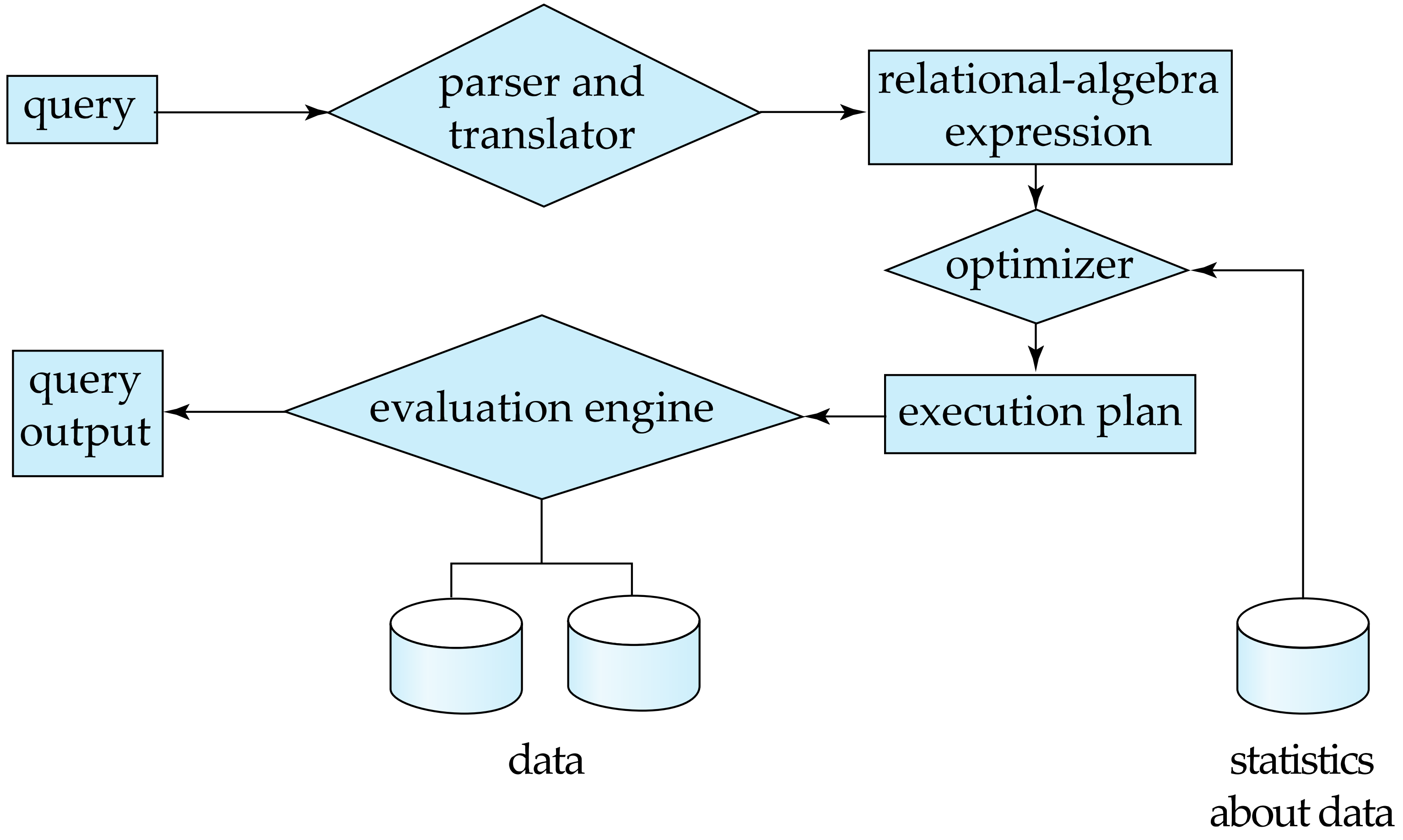
Introduction to Query Processing

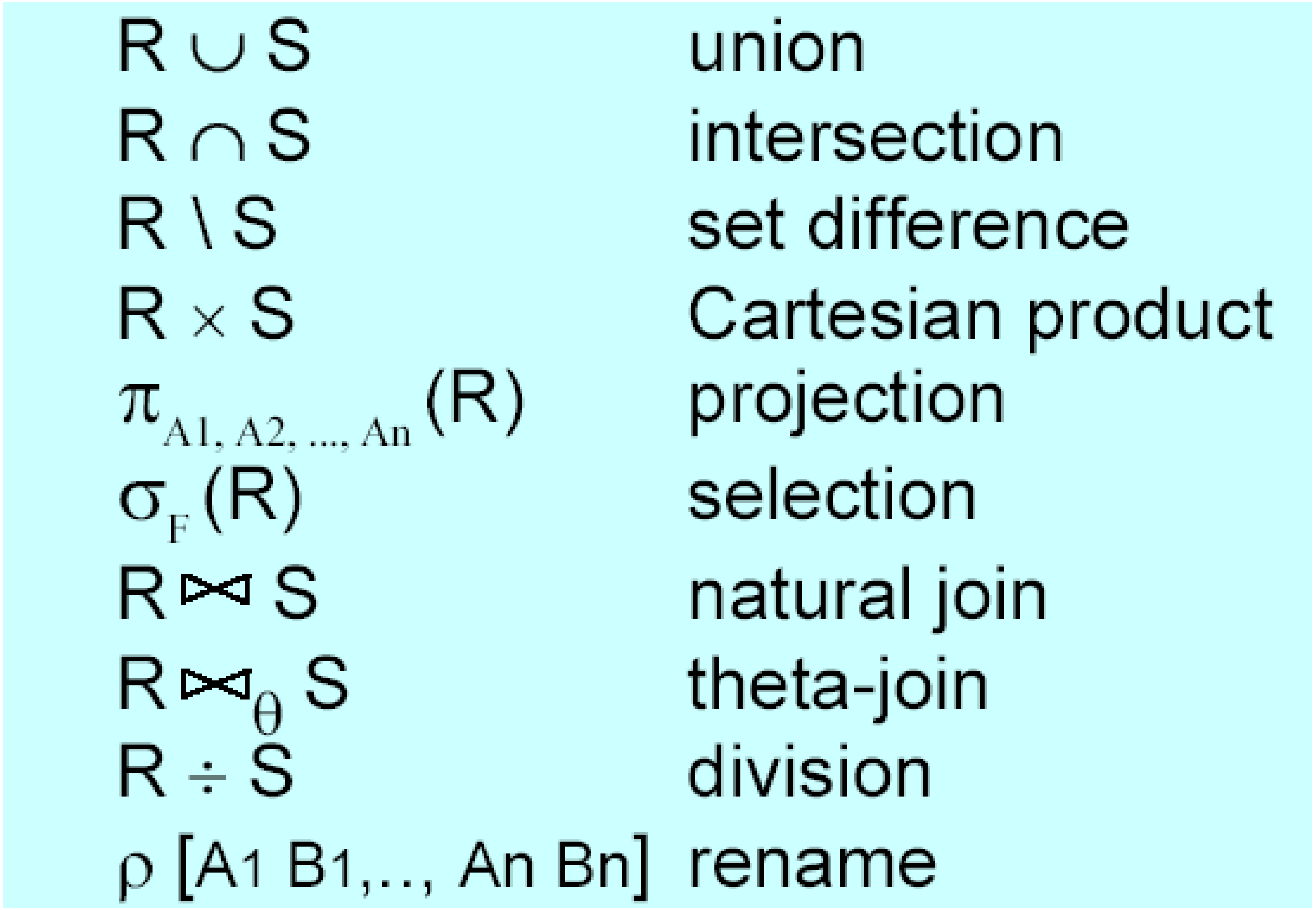
- **Query optimization**: the process of choosing a suitable execution strategy for processing a query.

- Internal representation of a query
  - **Query Tree**

Basic Steps in Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation





$R \cup S$	union
$R \cap S$	intersection
$R \setminus S$	set difference
$R \times S$	Cartesian product
$\pi_{A1, A2, \ldots, An}(R)$	projection
$\sigma_F(R)$	selection
$R \bowtie S$	natural join
$R \bowtie_\theta S$	theta-join
$R \div S$	division
$\rho\,[A_1\,B_1, \ldots, A_n\,B_n]$	rename

https://www.coursera.org/lecture/data-manipulation/relational-algebra-operators-theta-join-ppEF5

Translating SQL Queries into Relational Algebra

- **Query block:** the basic unit that can be translated into the algebraic operators and optimized.
- A query block contains a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clause if these are part of the block.
- **Nested queries** within a query are identified as separate query blocks.
- Aggregate operators in SQL must be included in the extended algebra.

Translating SQL Queries into Relational Algebra (2)

```
SELECT      LNAME, FNAME
FROM        EMPLOYEE
WHERE       SALARY > (     SELECT      MAX (SALARY)
                           FROM        EMPLOYEE
                           WHERE       DNO = 5);
```

```
SELECT      LNAME, FNAME
FROM        EMPLOYEE
WHERE       SALARY > C
```

```
SELECT      MAX (SALARY)
FROM        EMPLOYEE
WHERE       DNO = 5
```

$\pi_{\text{LNAME, FNAME}} (\sigma_{\text{SALARY>C}}(\text{EMPLOYEE}))$

$\pi g_{\text{MAX(SALARY)}} (\sigma_{\text{DNO=5}} (\text{EMPLOYEE}))$

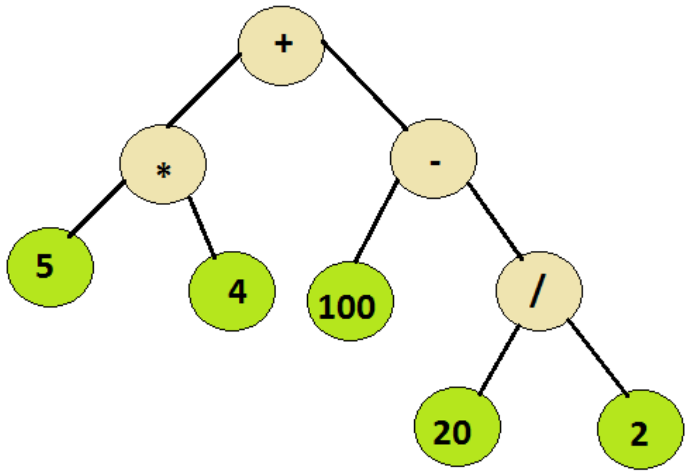
Using Heuristics in Query Optimization

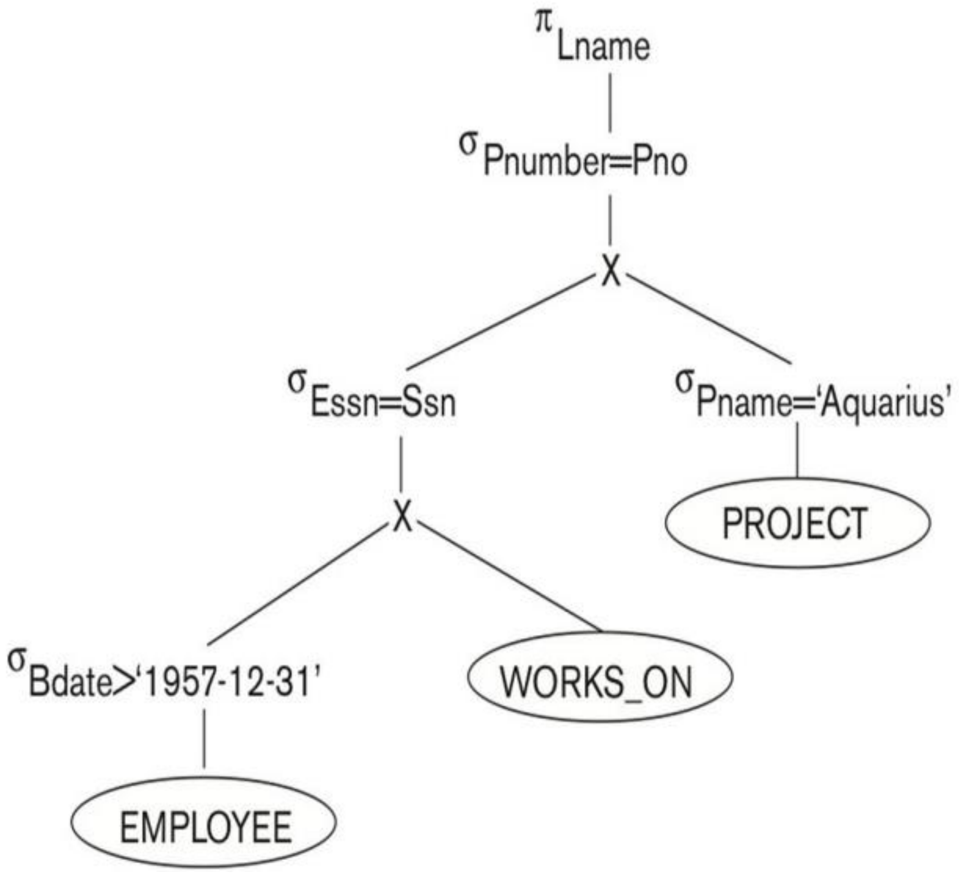
- ***query tree*** is used to represent a ***relational algebra*** or extended relational algebra expression, whereas a ***query graph*** is used to represent a *relational calculus expression*.

Using Heuristics in Query Optimization

- **Query tree**: a tree data structure that corresponds to a relational **algebra expression**. It represents the input relations of the query as ***leaf nodes*** of the tree, and represents the relational algebra operations as ***internal nodes***.
- An execution of the query tree consists of executing an internal node operation whenever its operands are available and then replacing that internal node by the relation that results from executing the operation.

Expression Tree Vs Query Tree



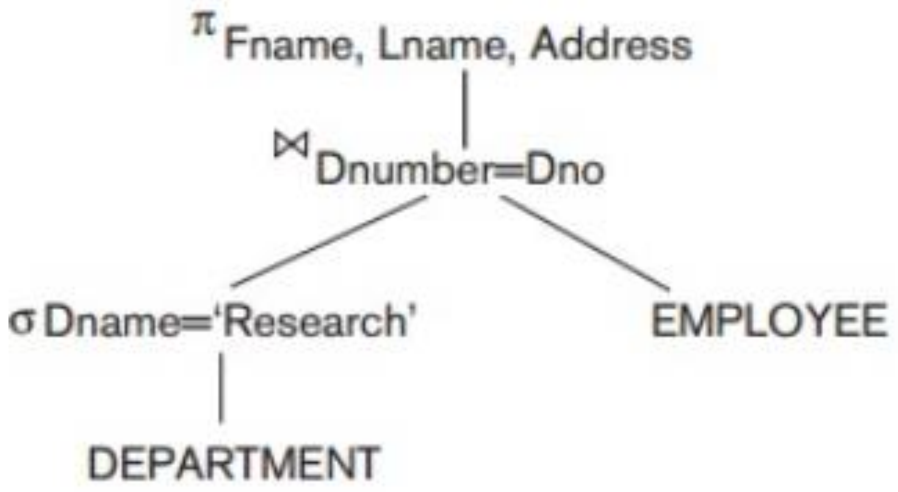


Query Tree

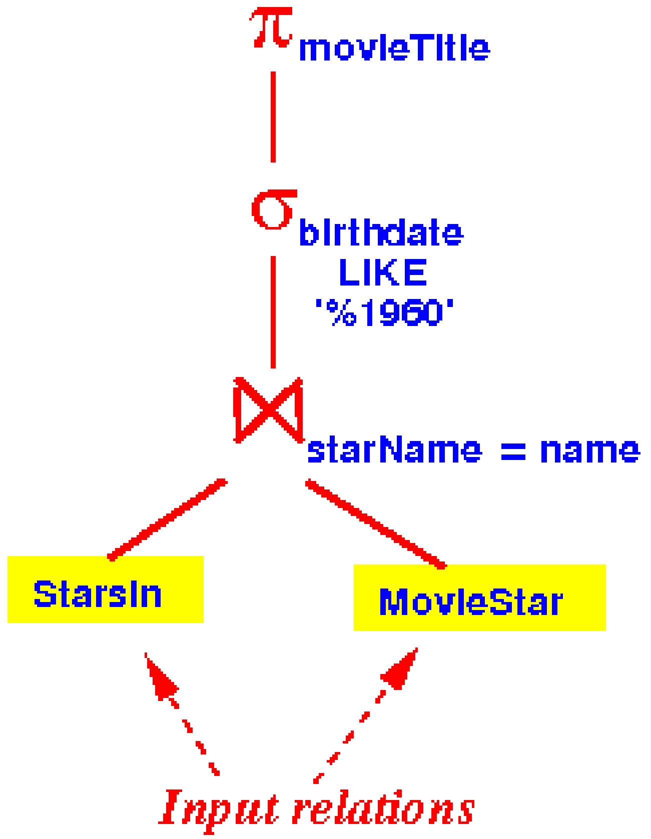
Relational algebra

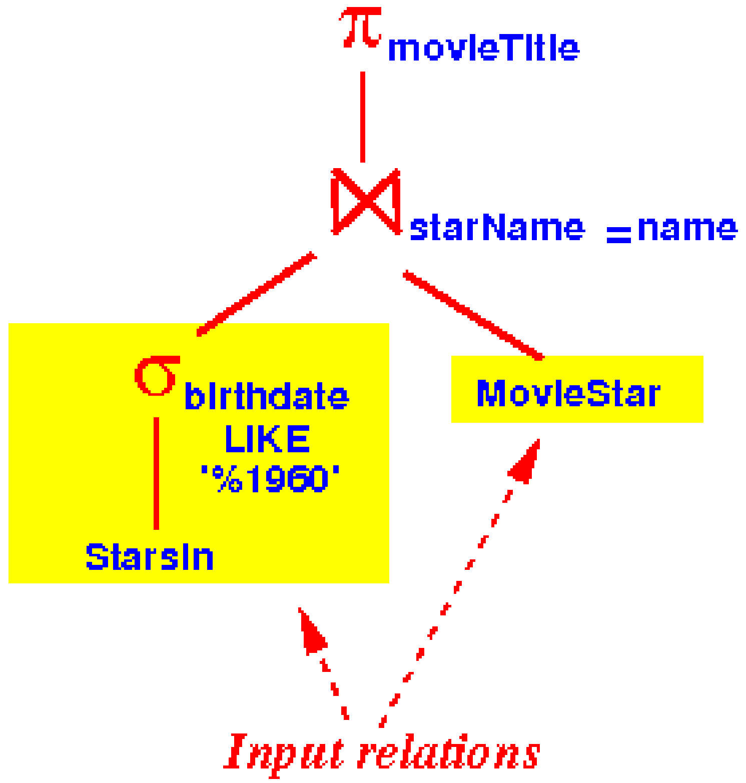
$$\pi_{\text{Fname, Lname, Address}}(\sigma_{\text{Dname='Research'}}(\text{DEPARTMENT}) \bowtie_{\text{Dnumber=Dno}} \text{EMPLOYEE})$$

Corresponding Query Tree



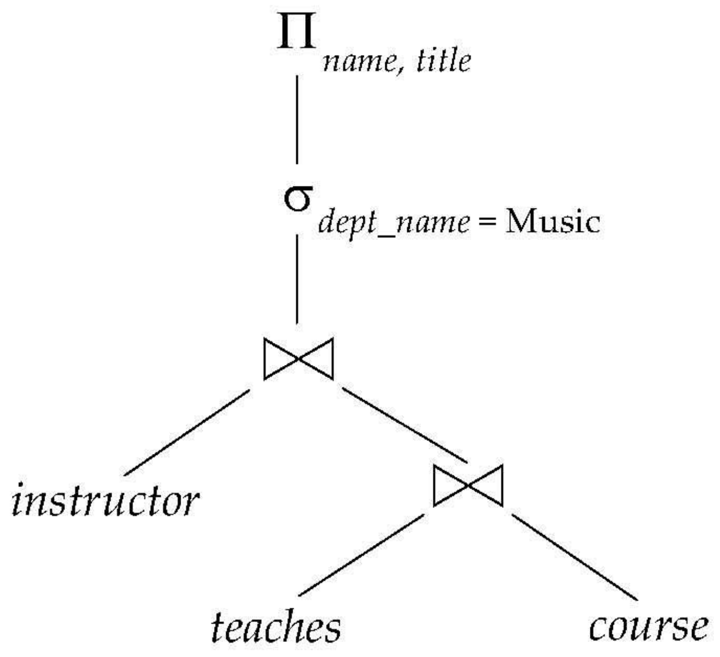
Query Tree

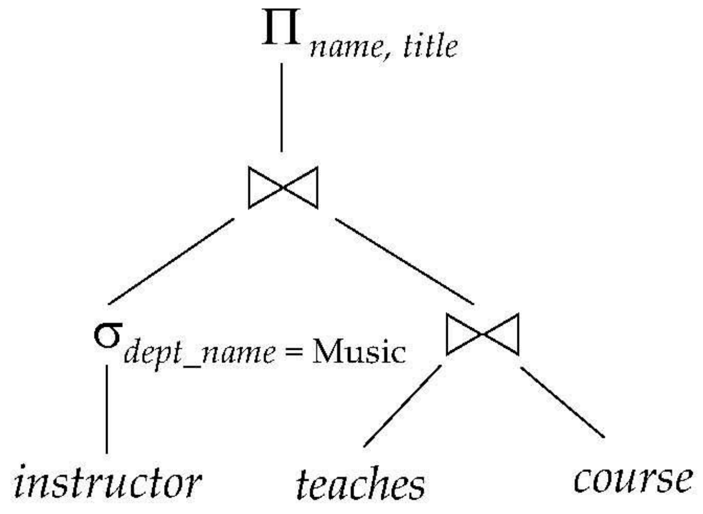

π movieTitle
σ birthdate LIKE '%1960'
⋈ starName = name
StarsIn
MovieStar
Input relations


π movieTitle
⋈ starName = name
σ birthdate LIKE '%1960'
StarsIn
MovieStar
Input relations

Query Tree

- Alternative ways of evaluating a given query
  - Equivalent expressions
  - Different algorithms for each operation





Using Heuristics in Query Optimization

General Transformation Rules for Relational Algebra Operations:

1. **Cascade of σ:** A **conjunctive selection** condition can be broken up **into a cascade** (sequence) of individual s operations:

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

2. **Commutativity of σ:** The σ operation is commutative:

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

3. **Cascade of π:** In a cascade (sequence) of π operations, all but the last one can be ignored:

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

4. **Commuting σ with π:** If the selection condition c involves only the attributes A1, ..., An in the projection list, the two operations can be commuted:

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

Using Heuristics in Query Optimization (2)

General Transformation Rules for Relational Algebra

5. **Commutativity of $\bowtie$ (and x):** The $\bowtie$ operation is commutative as is the x operation:

   $R \bowtie_C S = S \bowtie_C R; \qquad R \times S = S \times R$

6. **Commuting σ with $\bowtie$ (or x):** If all the attributes in the selection condition c involve only the attributes of one of the relations being joined—say, R—the two operations can be commuted as follows ($_{c0}$ is join condition) :

   $\sigma_c (R \bowtie_{c0} S) = (\sigma_c (R)) \bowtie_{c0} S$

   Alternatively, if the selection condition c can be written as (c1 and c2), where condition c1 involves only the attributes of R and condition c2 involves only the attributes of S, the operations commute as follows:

   $\sigma_c (R \bowtie_{c0} S) = (\sigma_{c1}(R)) \bowtie_{c0} (\sigma_{c2} (S))$

Using Heuristics in Query Optimization (3)

General Transformation Rules for Relational Algebra Operations (cont.):

7. **Commuting π with $\bowtie$ (or x):** Suppose that the projection list is L = {A1, ..., An, B1, ..., Bm}, where A1, ..., An are attributes of R and B1, ..., Bm are attributes of S. If the join condition c involves only attributes in L, the two operations can be commuted as follows:

$$\pi_L (R \bowtie_C S) = (\pi_{A1, ..., An} (R)) \bowtie_C (\pi_{B1, ..., Bm} (S))$$

If the join condition c contains additional attributes not in L, these must be added to the projection list, and a final π operation is needed.

Using Heuristics in Query Optimization (4)

General Transformation Rules for Relational Algebra Operations (cont.):

8. **Commutativity of set operations:** The set operations ∪ and ∩ are commutative but – is not.

   $$R \cup S = S \cup R ; \qquad R \cap S = S \cap R$$

8. **Associativity of ⋈ , x, ∪, and ∩ :** These four operations are individually associative; that is, if θ stands for any one of these four operations (throughout the expression), we have $(R\ \theta\ S)\ \theta\ T = R\ \theta\ (S\ \theta\ T)$

9. **Commuting σ with set operations:** The σ operation commutes with ∪ , ∩ , and –. If θ stands for any one of these three operations, we have

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

Using Heuristics in Query Optimization (5)

General Transformation Rules for Relational Algebra Operations (cont.):

11. **The π operation commutes with U.**

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

12. **Converting a (σ, x) sequence into $\bowtie$:** If the condition c of a σ that follows a x Corresponds to a join condition, convert the (σ, x) sequence into a $\bowtie$ as follows:

$$(\sigma_C (R \times S)) = (R \bowtie_C S)$$

Using Heuristics in Query Optimization (6)

Outline of a Heuristic Algebraic Optimization Algorithm:

1. Using rule 1, break up any select operations with conjunctive conditions into a cascade of select ops.
2. Using rules 2, 4, 6, and 10 concerning the commutativity of select with other operations, move each select operation as far down the query tree as is permitted by the attributes involved in the select condition.
3. Using rule 9 concerning associativity of binary operations, rearrange the leaf nodes of the tree so that the leaf node relations with the most restrictive select operations are executed first in the query tree rep.

Using Heuristics in Query Optimization (7)

Outline of a Heuristic Algebraic Optimization Algorithm (cont.)

4. Using Rule 12, combine a Cartesian product operation with a subsequent select operation in the tree into a join operation.
5. Using rules 3, 4, 7, and 11 concerning the cascading of project and the commuting of project with other operations, break down and **move lists of projection attributes down the tree as far as possible** by creating new project operations as needed.

Using Heuristics in Query Optimization (8)

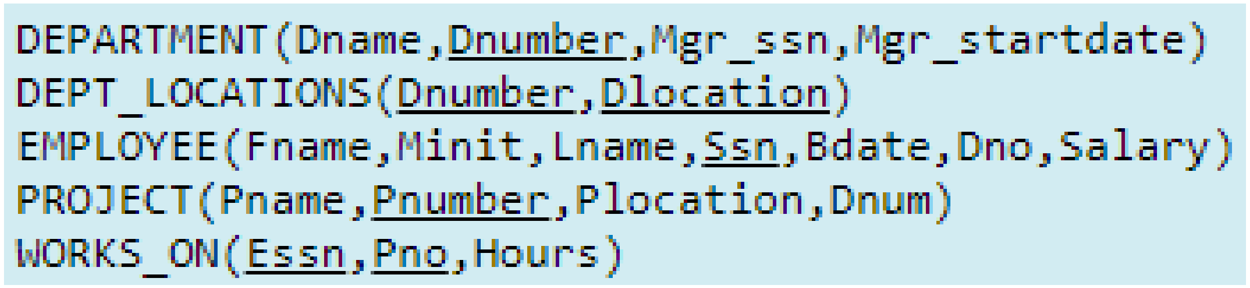
Summary of Heuristics for Algebraic Optimization:

1. The main heuristic is to **apply first the operations** that reduce the size of intermediate results.
2. **Perform select operations as early as possible** to reduce the number of tuples. **perform project operations as early as possible** to reduce the number of attributes. (**This is done by moving select and project operations as far down the tree as possible**.)
3. The **select and join operations that are most restrictive** should be executed before other similar operations. (**This is done by reordering the leaf nodes of the tree among themselves and adjusting the rest of the tree appropriately**.)

Transformation Algorithm Outline

- Transform a query represented in relational algebra to an equivalent one (generates the same result.)
- **Step 1:** Design the initial canonical tree (initial query tree) of the query
- **Step 2:** Decompose SELECT(σ) operations and move the SELECT operation down the query tree
- **Step 3:** Apply more restrictive SELECT (σ) operation first. If the two relations are residing at same site, they will be handled first.
- **Step 4:** Replace CARTESIAN PRODUCT(x) and SELECT(σ) with THETA JOIN (⋈) operation
- **Step 5:** Move PROJECT (π) operations down the query tree

Example 1:

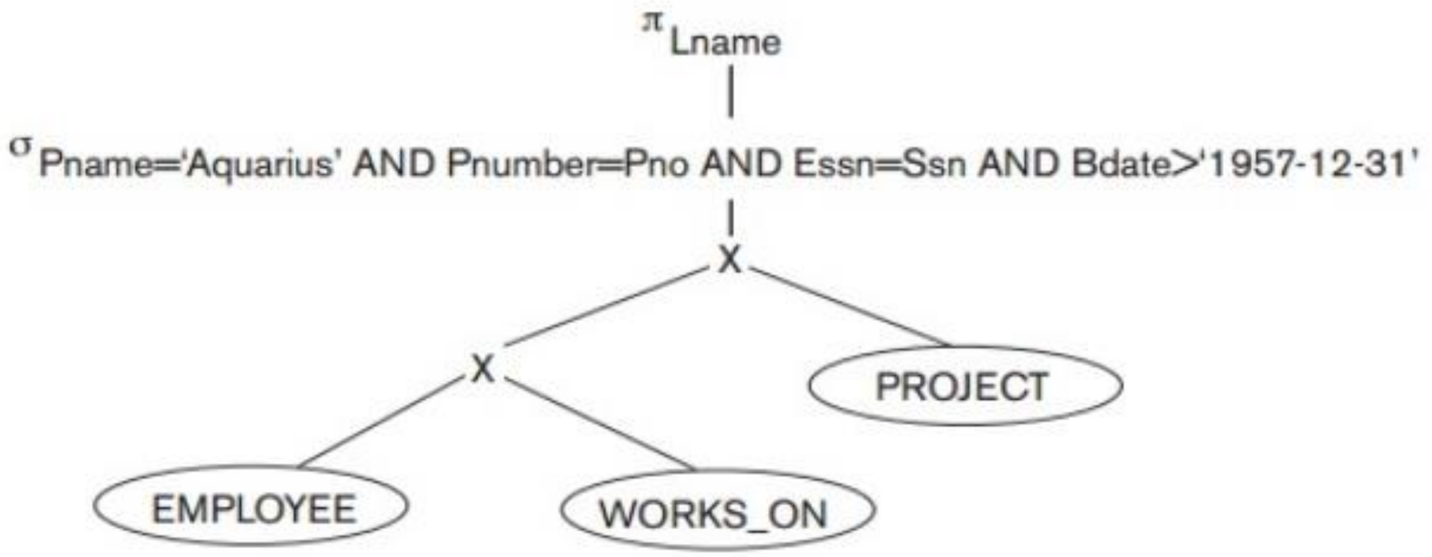


```
DEPARTMENT(Dname,Dnumber,Mgr_ssn,Mgr_startdate)
DEPT_LOCATIONS(Dnumber,Dlocation)
EMPLOYEE(Fname,Minit,Lname,Ssn,Bdate,Dno,Salary)
PROJECT(Pname,Pnumber,Plocation,Dnum)
WORKS_ON(Essn,Pno,Hours)
```

- SELECT Lname
- FROM EMPLOYEE, WORKS_ON, PROJECT
- WHERE Pname=‘Aquarius’
  AND Pnumber=Pno AND
  Essn=Ssn AND
  Bdate > ‘1957-12-31’;

Heuristic optimization process (1)

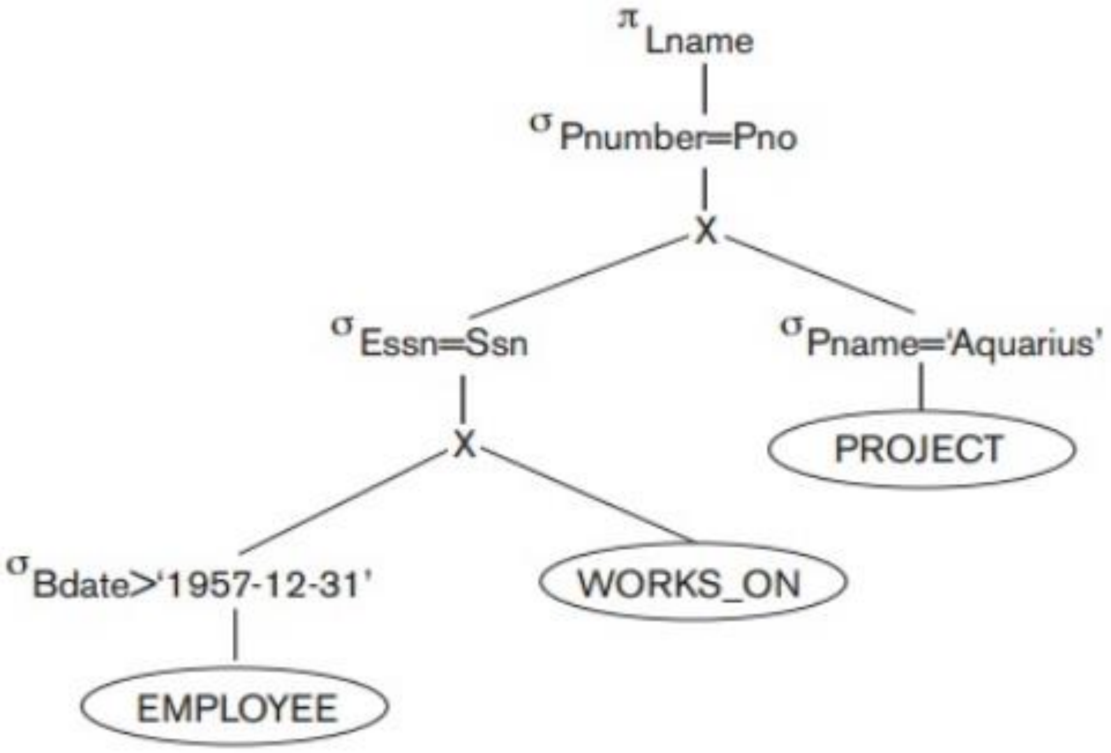
- Step 1. Initialization canonical query tree



π_{Lname}
$\sigma_{\text{Pname='Aquarius' AND Pnumber=Pno AND Essn=Ssn AND Bdate>'1957-12-31'}}$
X
X
PROJECT
EMPLOYEE
WORKS_ON

Heuristic optimization process (2)

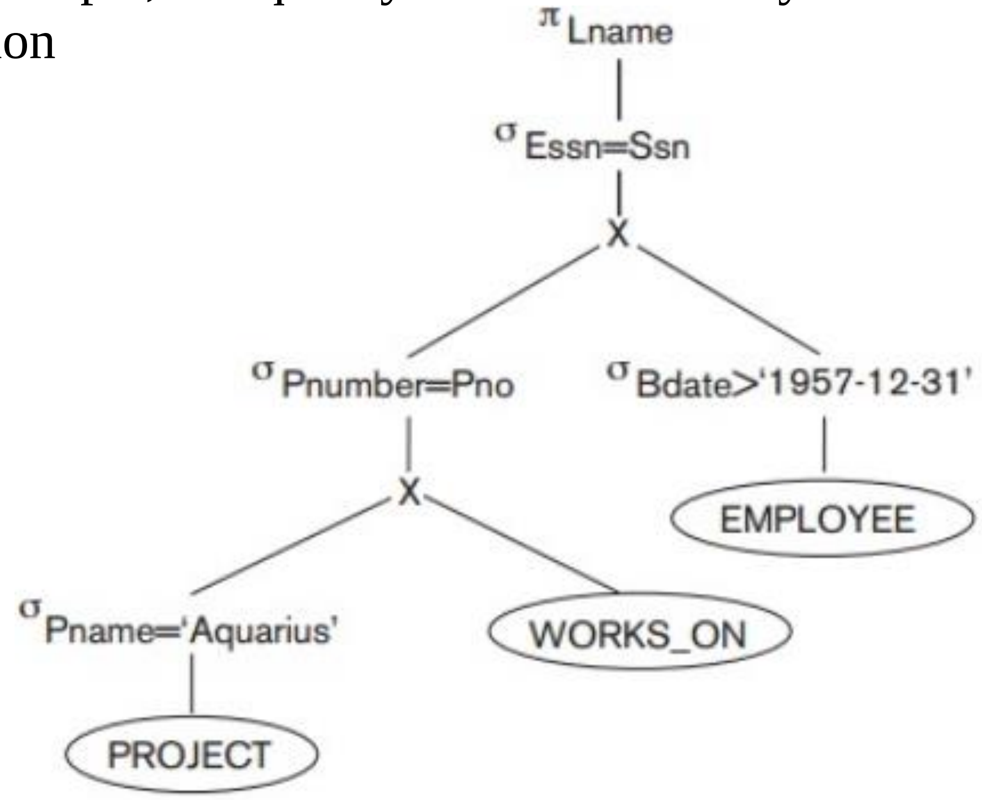
- Step 2. Decompose SELECT(σ) operations and move the SELECT operation down the query tree



π_{Lname}
$\sigma_{\text{Pnumber=Pno}}$
X
$\sigma_{\text{Essn=Ssn}}$
$\sigma_{\text{Pname='Aquarius'}}$
X
PROJECT
$\sigma_{\text{Bdate>'1957-12-31'}}$
WORKS_ON
EMPLOYEE

Heuristic optimization process (3)

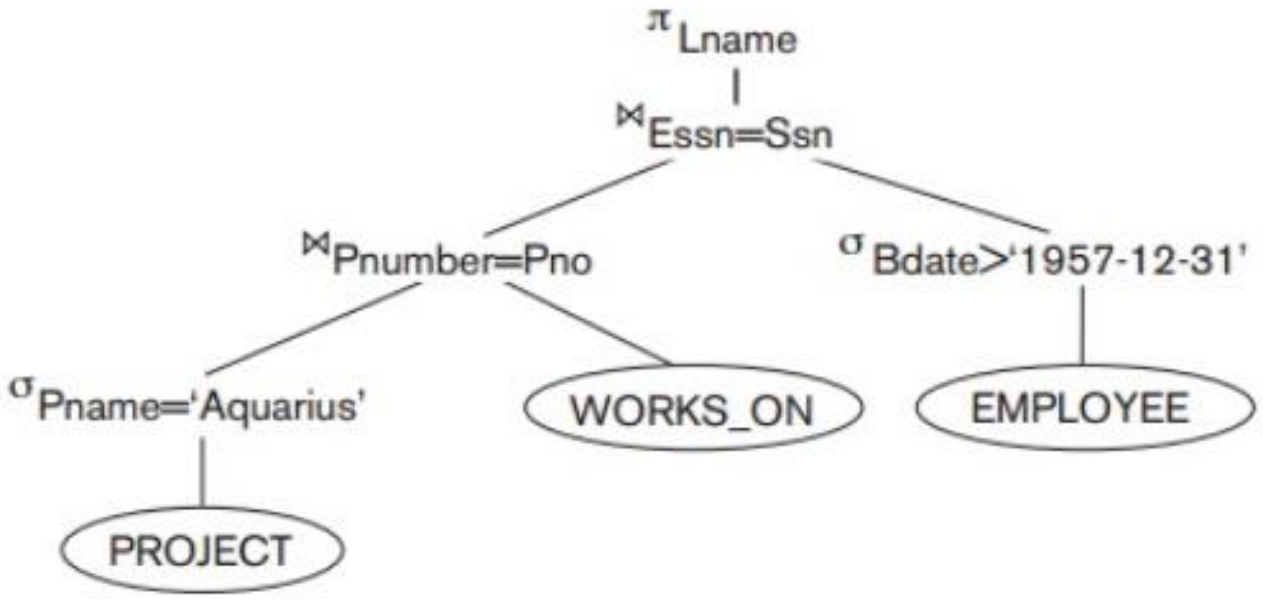
- **Step 3. Apply more restrictive SELECT operation first**
  - **most restrictive**—that is, result in relations with the fewest tuples or with the smallest absolute size—should be executed before other similar operations.
  - Using the associativity rules, rearrange the leaf nodes so that the most restrictive selection conditions are executed first. And if the two relations are residing at same site, they will be handled first.
  - For example, an equality condition is likely more restrictive than an inequality condition



Pnumber is a key attribute and with = operator will retrieve a single record only.

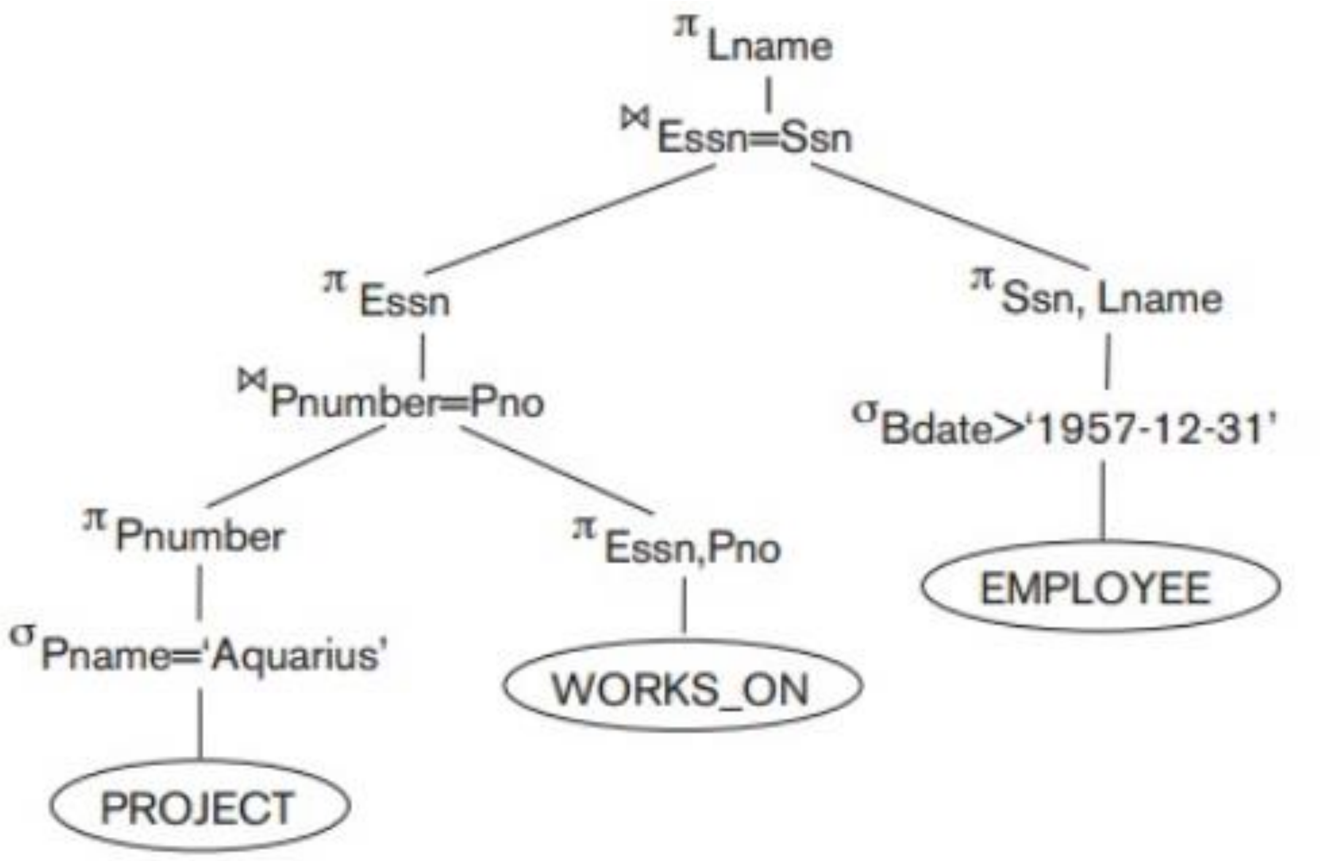
Heuristic optimization process (4)

- Step 4. Replace CARTESIAN PRODUCT and SELECT with THETA JOIN operation



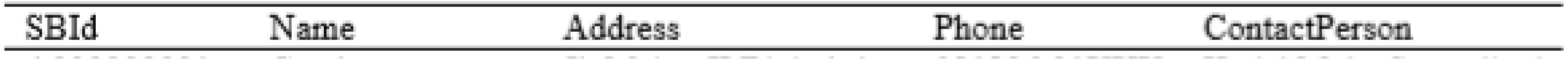
Heuristic optimization process (5)

- Step 5. Move PROJECT operations down the query tree



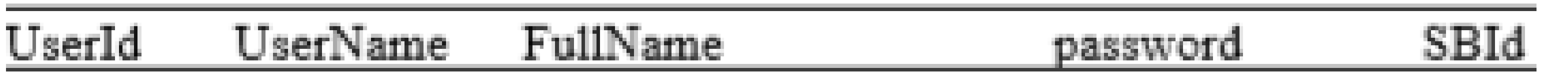
Example 2:

- Four tables:
- Table1: SmallBusiness



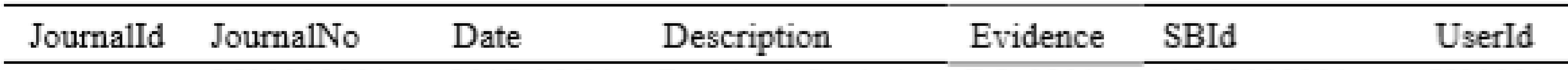
SBId	Name	Address	Phone	ContactPerson

- Table2: User



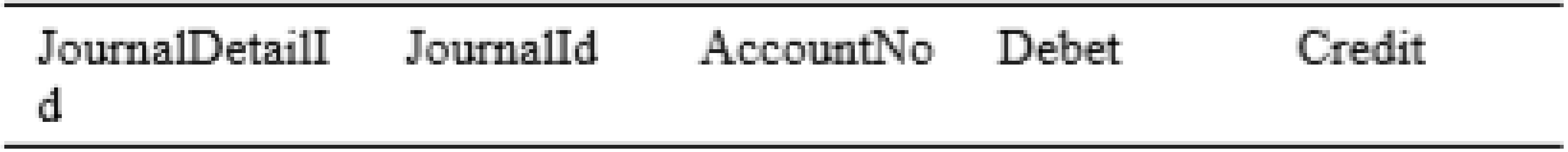
UserId	UserName	FullName	password	SBId

- Table3: Journal



JournalId	JournalNo	Date	Description	Evidence	SBId	UserId

- Table 4: JournalDetail



JournalDetailId	JournalId	AccountNo	Debet	Credit

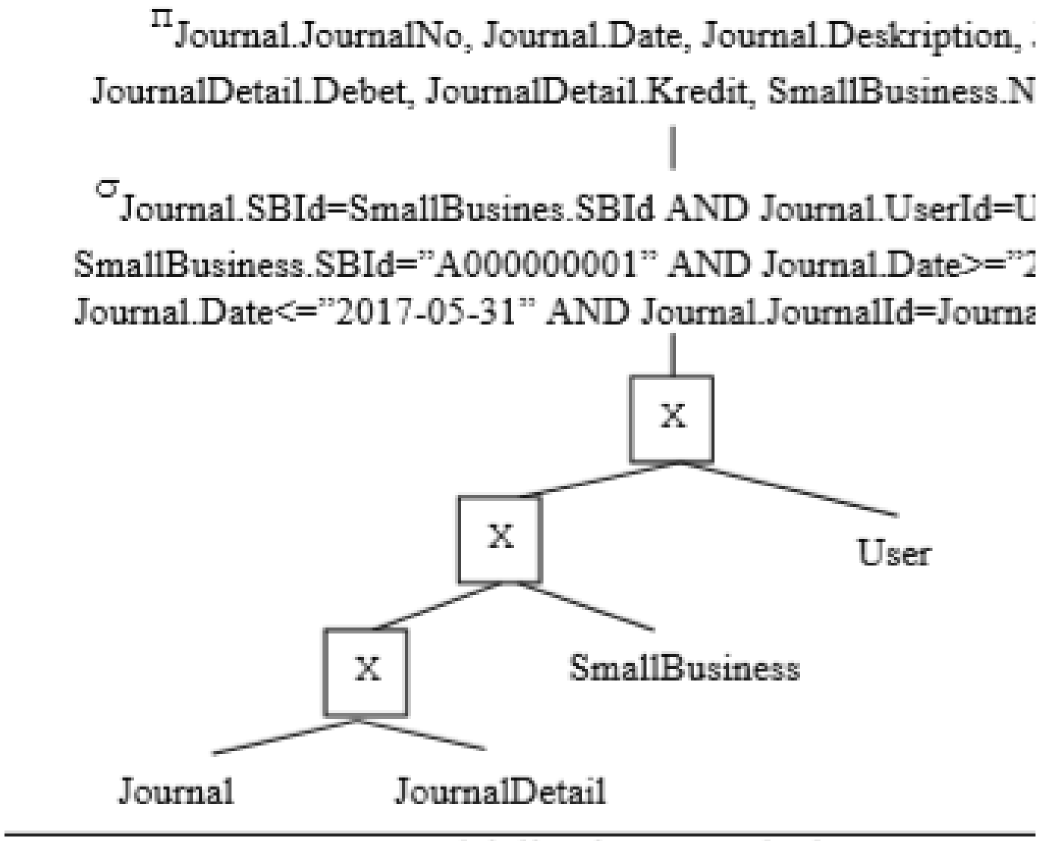
- SELECT Journal.JournalNo, Journal.Date, Journal.Description, Journal.Evidence, JournalDetail.Debet, JournalDetail.Credit, SmallBusiness.Name, User.FullName

- FROM Journal, JournalDetail, SmallBusiness, User

- WHERE Journal.SBId = SmallBusiness.SBId AND Journal.UserId = User.UserId AND SmallBusiness.SBId = ”A000000001” AND Journal.Date >= ”2019-05-01” AND
  Journal.Date <= ”2019-05-31” AND
  Journal.JournalNo = JournalDetail.JournalNo

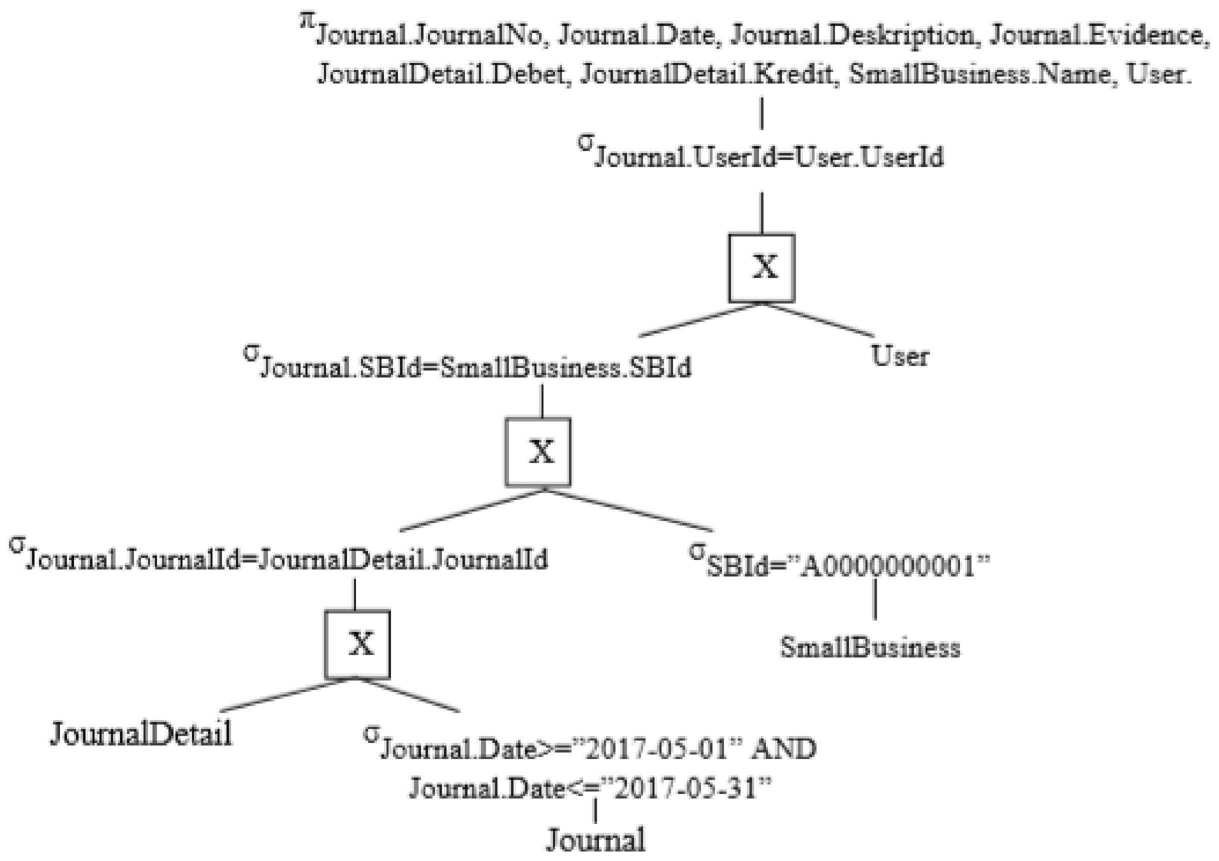
Heuristic optimization process (1)

- Step 1. Initialization canonical query tree


π Journal.JournalNo, Journal.Date, Journal.Deskription,
JournalDetail.Debet, JournalDetail.Kredit, SmallBusiness.N
σ Journal.SBId=SmallBusines.SBId AND Journal.UserId=U
SmallBusiness.SBId="A000000001" AND Journal.Date>="
Journal.Date<="2017-05-31" AND Journal.JournalId=Journa
X
X
User
X
SmallBusiness
Journal
JournalDetail

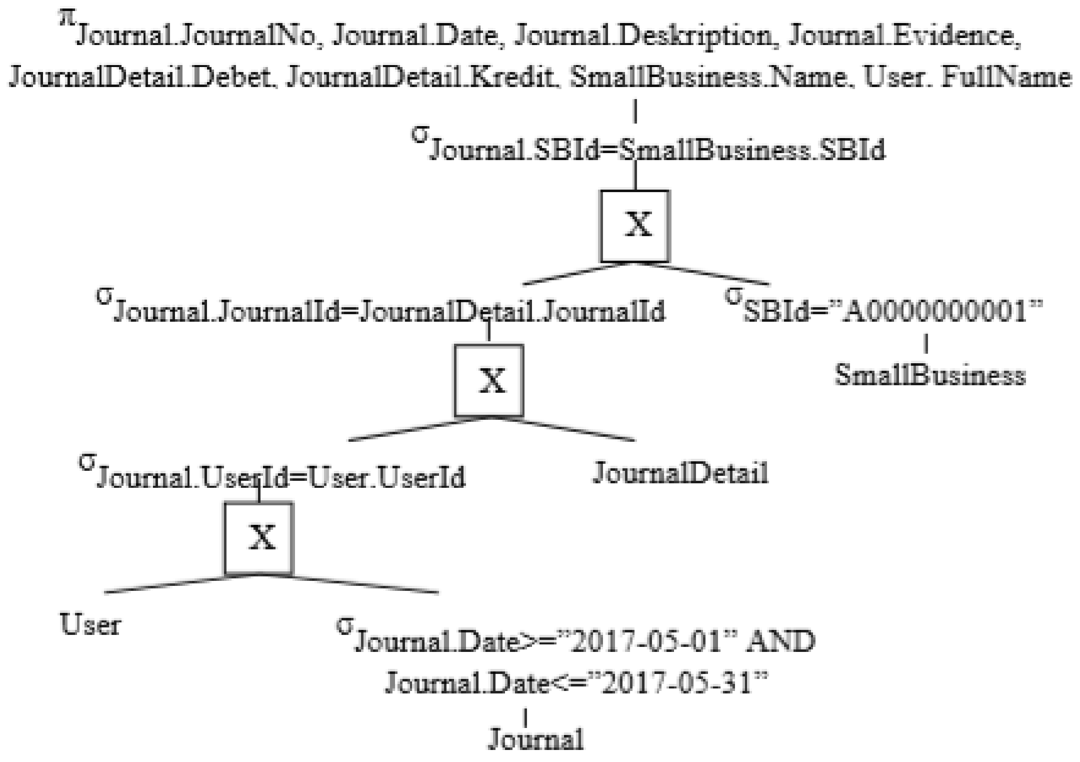
Heuristic optimization process (2)

- Step 2. Decompose SELECT(σ) operations and move the SELECT operation down the query tree


π Journal.JournalNo, Journal.Date, Journal.Deskription, Journal.Evidence, JournalDetail.Debet, JournalDetail.Kredit, SmallBusiness.Name, User.
σ Journal.UserId=User.UserId
X
σ Journal.SBId=SmallBusiness.SBId
User
X
σ Journal.JournalId=JournalDetail.JournalId
σ SBId="A0000000001"
SmallBusiness
X
JournalDetail
σ Journal.Date>="2017-05-01" AND Journal.Date<="2017-05-31"
Journal

Heuristic optimization process (3)

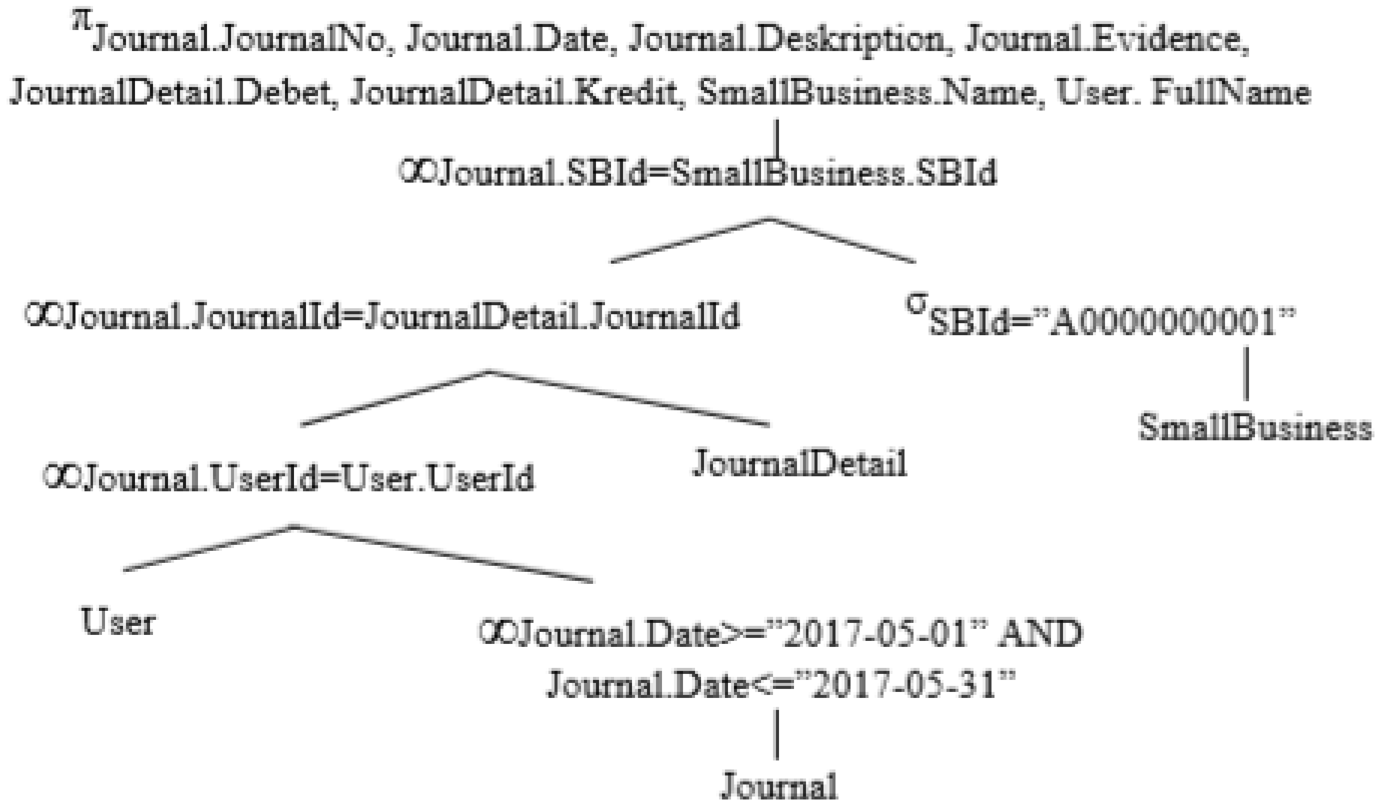
- Step 3. Apply more restrictive SELECT operation first



π Journal.JournalNo, Journal.Date, Journal.Deskription, Journal.Evidence, JournalDetail.Debet, JournalDetail.Kredit, SmallBusiness.Name, User. FullName
σ Journal.SBId=SmallBusiness.SBId
X
σ Journal.JournalId=JournalDetail.JournalId
σ SBId="A0000000001"
SmallBusiness
X
σ Journal.UserId=User.UserId
JournalDetail
X
User
σ Journal.Date>="2017-05-01" AND Journal.Date<="2017-05-31"
Journal

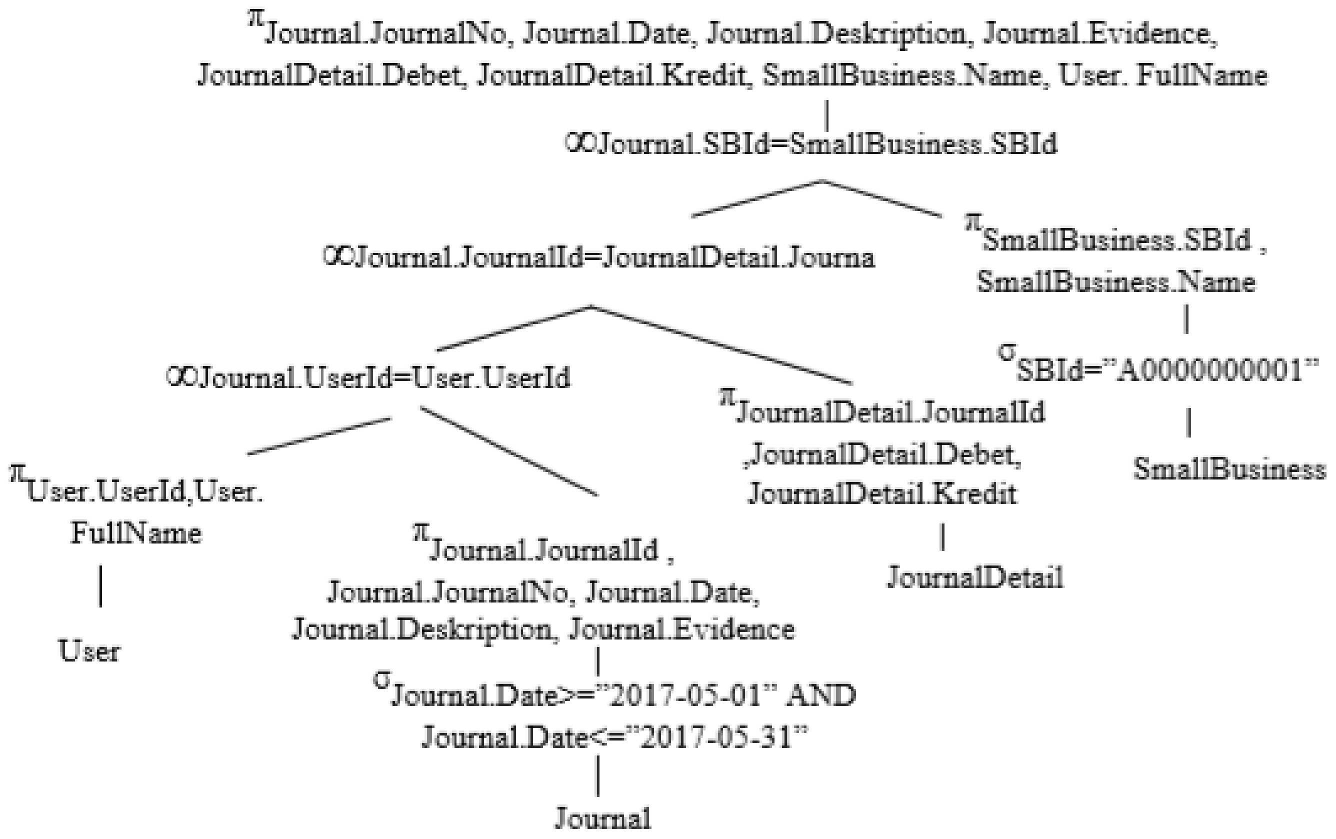
Heuristic optimization process (4)

- Step 4. Replace CARTESIAN PRODUCT and SELECT with JOIN operation


π Journal.JournalNo, Journal.Date, Journal.Deskription, Journal.Evidence, JournalDetail.Debet, JournalDetail.Kredit, SmallBusiness.Name, User. FullName
⋈ Journal.SBId=SmallBusiness.SBId
⋈ Journal.JournalId=JournalDetail.JournalId
σ SBId="A0000000001"
SmallBusiness
⋈ Journal.UserId=User.UserId
JournalDetail
User
⋈ Journal.Date>="2017-05-01" AND Journal.Date<="2017-05-31"
Journal

Heuristic optimization process (5)

- Step 5. Move PROJECT operations down the query tree

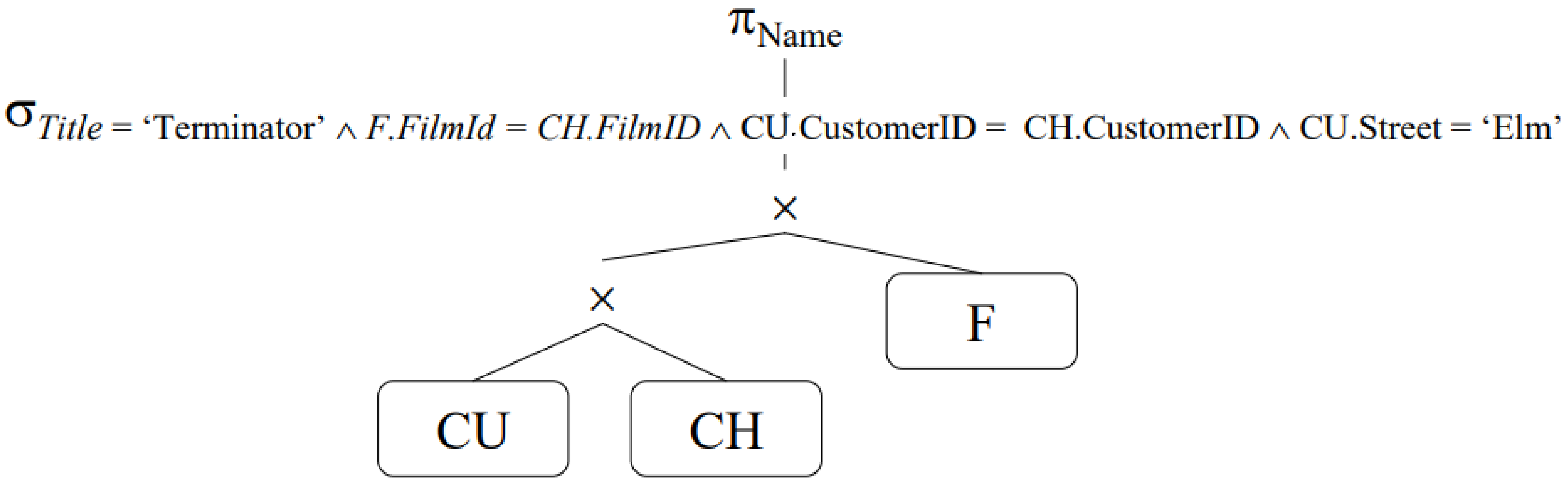

π Journal.JournalNo, Journal.Date, Journal.Deskription, Journal.Evidence, JournalDetail.Debet, JournalDetail.Kredit, SmallBusiness.Name, User. FullName
∞Journal.SBId=SmallBusiness.SBId
∞Journal.JournalId=JournalDetail.Journa
π SmallBusiness.SBId , SmallBusiness.Name
σ SBId="A0000000001"
SmallBusiness
∞Journal.UserId=User.UserId
π JournalDetail.JournalId ,JournalDetail.Debet, JournalDetail.Kredit
JournalDetail
π User.UserId,User. FullName
User
π Journal.JournalId , Journal.JournalNo, Journal.Date, Journal.Deskription, Journal.Evidence
σ Journal.Date>="2017-05-01" AND Journal.Date<="2017-05-31"
Journal

Query Tree Optimization Example 3

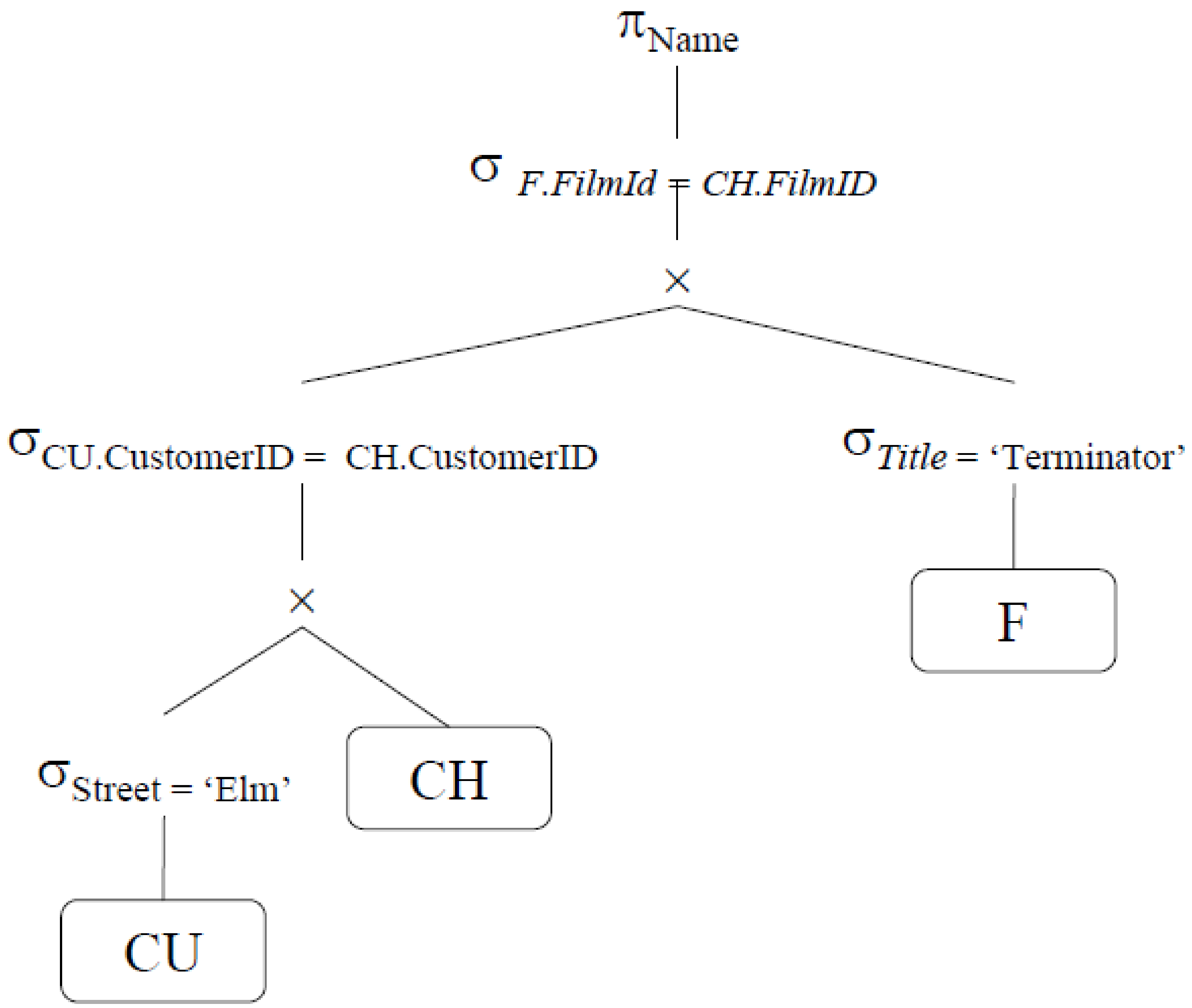
- What are the names of customers living on Elm Street who have checked out “Terminator”?
- SQL query:

```
SELECT Name
FROM   Customer CU, CheckedOut CH, Film F
WHERE  Title = ’Terminator’ AND F.FilmId = CH.FilmID
AND CU.CustomerID = CH.CustomerID AND CU.Street = ‘Elm’
```

Step 1. Initialization canonical query tree

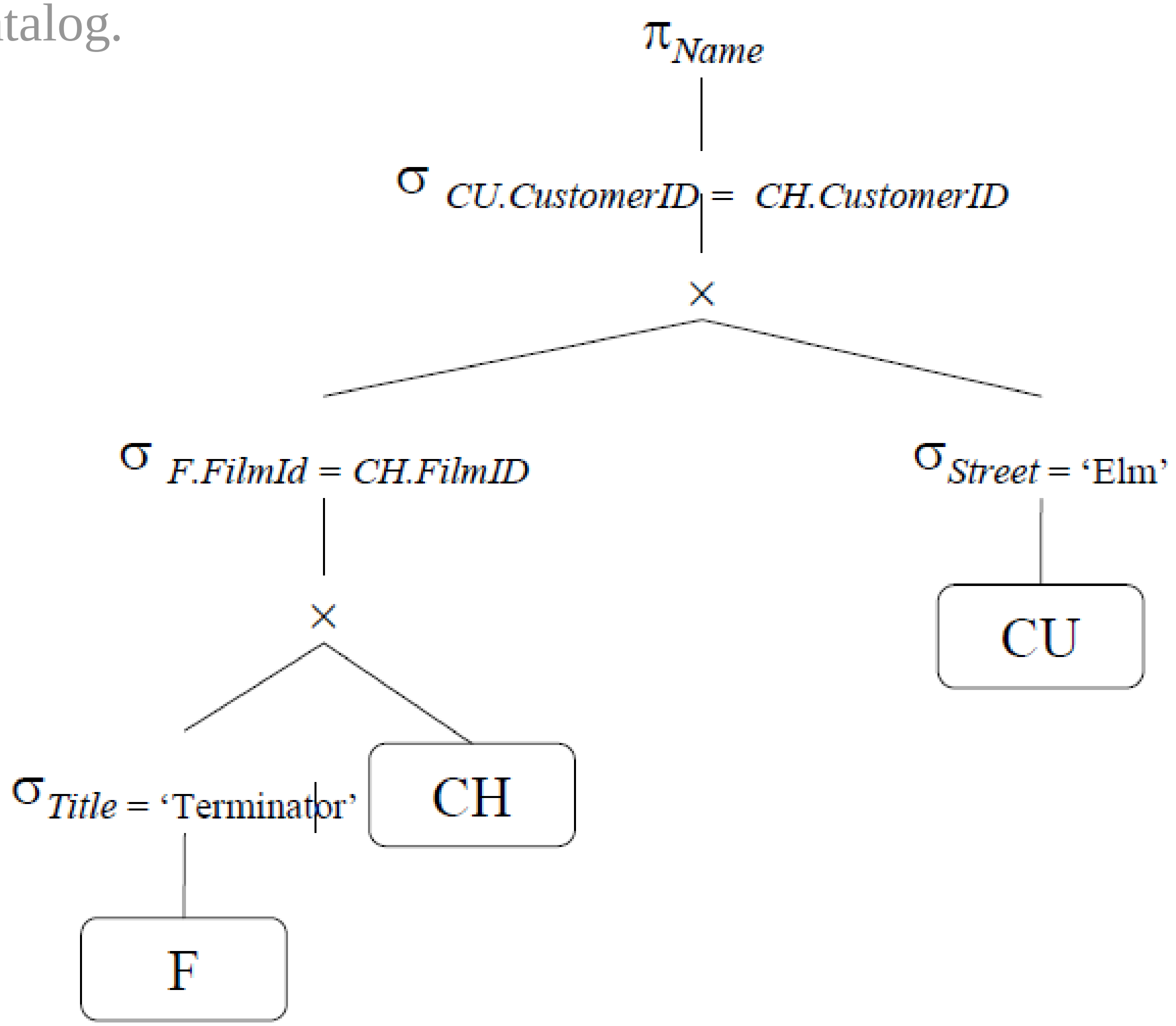

π_{Name}
$\sigma_{Title = \text{'Terminator'} \wedge F.FilmId = CH.FilmID \wedge CU.CustomerID = CH.CustomerID \wedge CU.Street = \text{'Elm'}}$
×
×
F
CU
CH

Step2: Apply Selections Early


π_{Name}
$\sigma_{F.FilmId = CH.FilmID}$
$\times$
$\sigma_{CU.CustomerID = CH.CustomerID}$
$\sigma_{Title = \text{'Terminator'}}$
$\times$
F
$\sigma_{Street = \text{'Elm'}}$
CH
CU

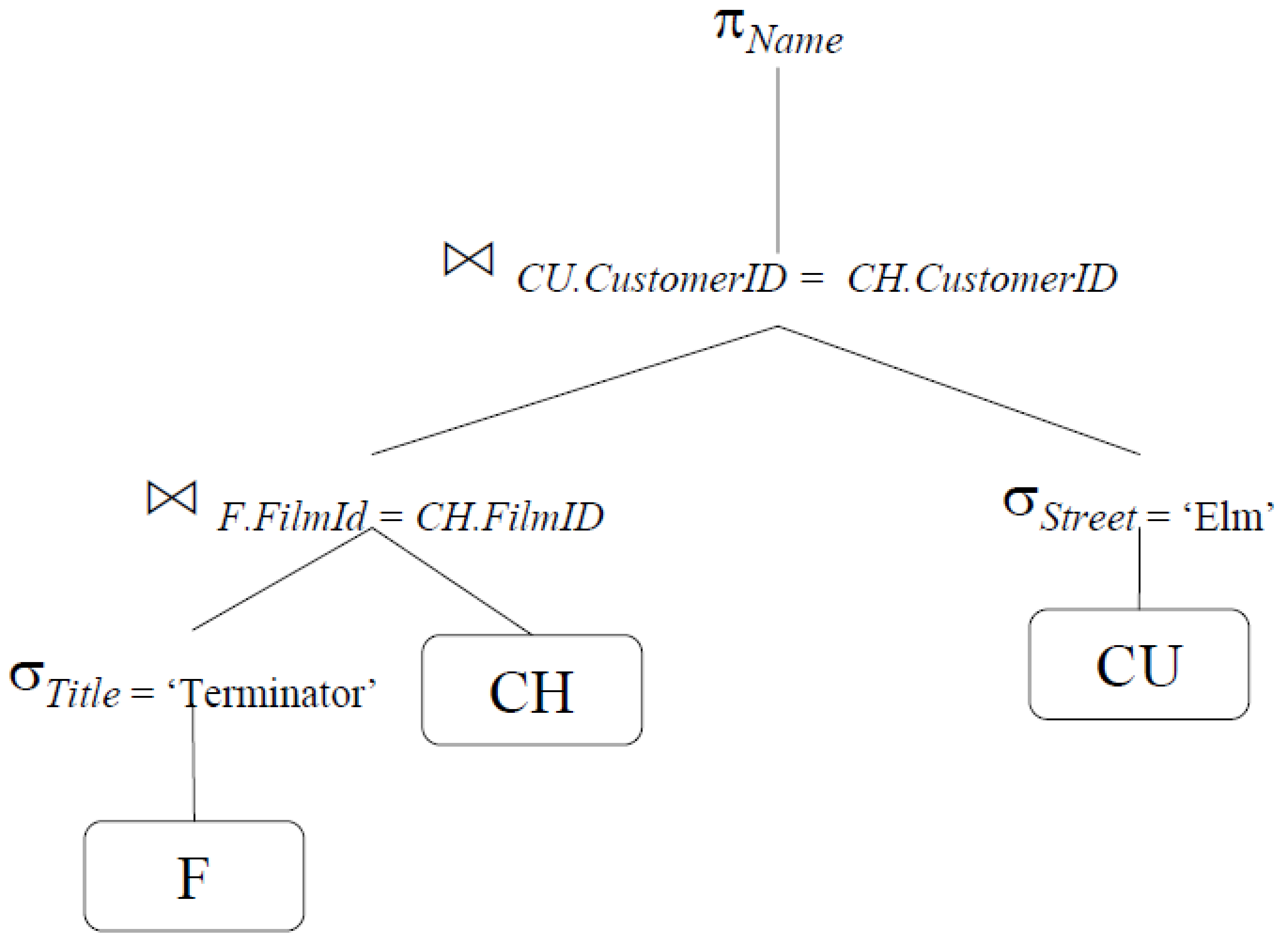
Step3: Apply More Restrictive Selections Early

The definition of *most restrictive* SELECT can mean either the ones that produce a relation with the fewest tuples or with the smallest absolute size. this is more practical because estimates of selectivity are often available in the DBMS catalog.

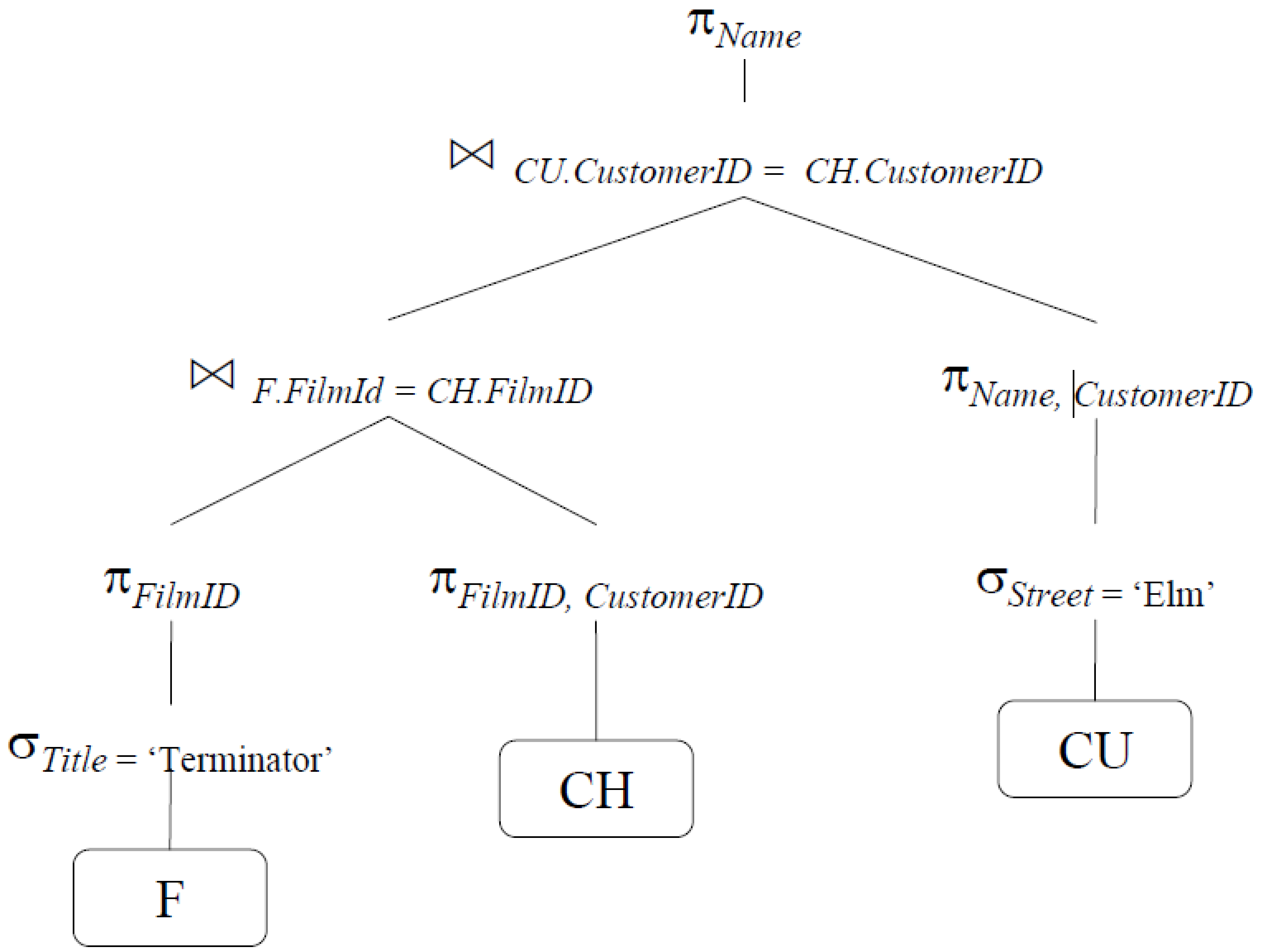


π_{Name}
$\sigma_{CU.CustomerID = CH.CustomerID}$
×
$\sigma_{F.FilmId = CH.FilmID}$
$\sigma_{Street = \text{'Elm'}}$
×
CU
$\sigma_{Title = \text{'Terminator'}}$
CH
F

Step 4: Form Joins


π_{Name}
$\bowtie_{CU.CustomerID = CH.CustomerID}$
$\bowtie_{F.FilmId = CH.FilmID}$
$\sigma_{Street = \text{'Elm'}}$
$\sigma_{Title = \text{'Terminator'}}$
CH
CU
F

Step 5: Apply Projections Early


π_{Name}
$\bowtie_{CU.CustomerID\ =\ CH.CustomerID}$
$\bowtie_{F.FilmId\ =\ CH.FilmID}$
$\pi_{Name,\ CustomerID}$
π_{FilmID}
$\pi_{FilmID,\ CustomerID}$
$\sigma_{Street\ =\ \text{'Elm'}}$
$\sigma_{Title\ =\ \text{'Terminator'}}$
CH
CU
F