HALMSTAD UNIVERSITY

DS4001 Databases (7.5 credits)

Lecture 11 – Query Optimization

Yuantao Fan

yuantao.fan@hh.se

Halmstad University

Introduction

- SQL queries are declarative…
  - You wrote want you want
  - Not how to process data in tables to acquire the result
  - Differece in performance depending on the methods/plans

- First query optimizer – IBM system R
  - Human can write better queries compared to DBMS

- Query optimization
  - Heuristics - improve the query by rewriting inefficient operations, without evaluating the cost
  - Cost-based search
    - Generate a set of equivalent plans, compute their cost and select the most efficient one

Example – Predicate Pushdown

Teacher(tid, full_name, age, nationality)

tid	full_name	age	nationality
11	John Smith	42	America
22	Jens Jonathon	31	Sweden
33	Stefan Miller	39	Sweden
44	Kayle Persson	33	UK

```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid
        AND tt.hours > 50
```

Teaches(tid, cid, hours)

tid	cid	hours
11	1	80
11	2	100
22	4	50
33	4	50
44	3	100

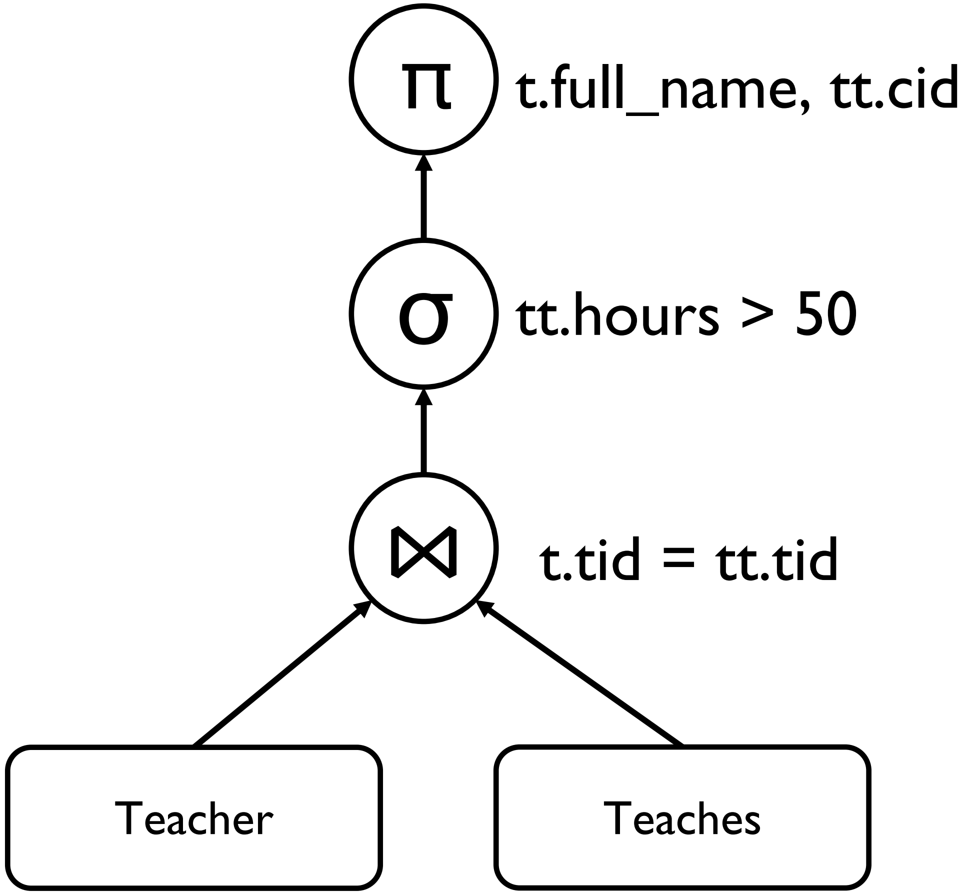
$$\pi_{\text{full\_name, cid}}(\sigma_{\text{hours}>50}(\text{Teacher} \bowtie \text{Teaches}))$$

Query Plan

```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid
        AND tt.hours > 50
```

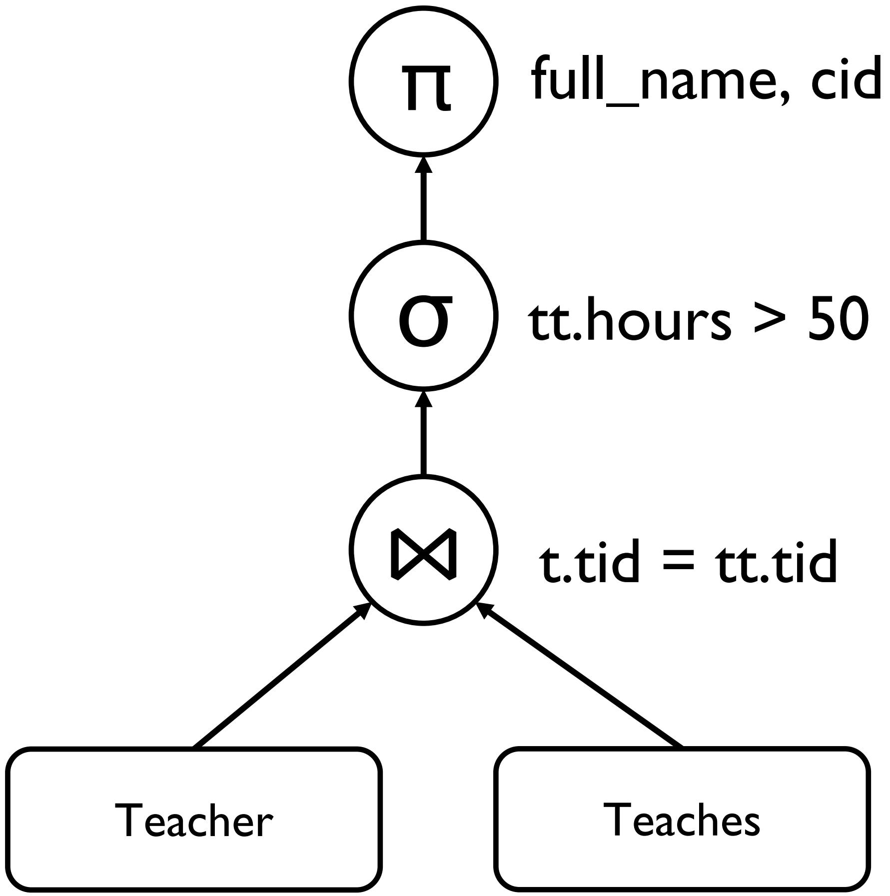
- A list of instructions that the database needs to follow in order to execute a query on the data
- Query tree is a tree data structure that corresponds to an extended relational algebra expression
- The operations are arranged in a tree and the input data (i.e. tables/relations) flow from the bottom (leaves) to the top (root)

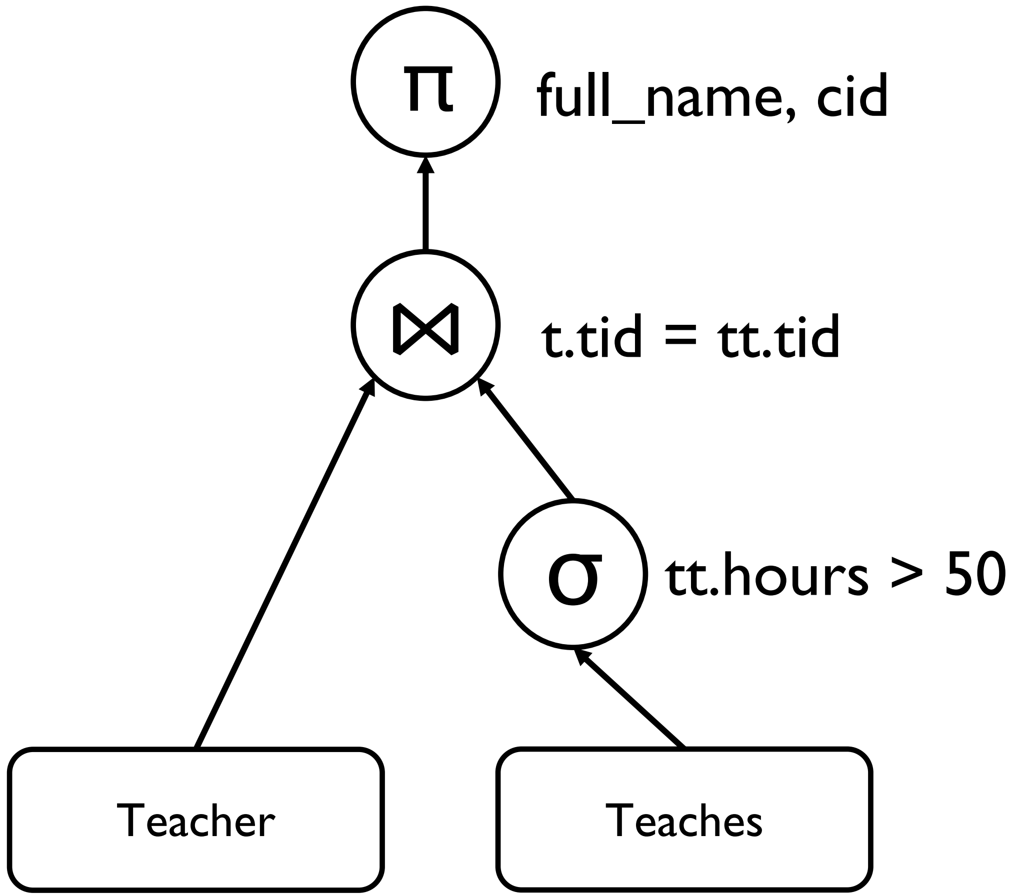
$$\pi_{\text{full\_name, cid}}(\sigma_{\text{hours>50}}(\text{Teacher} \bowtie \text{Teaches}))$$



Example – Predicate Pushdown

```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid
        AND tt.hours > 50
```

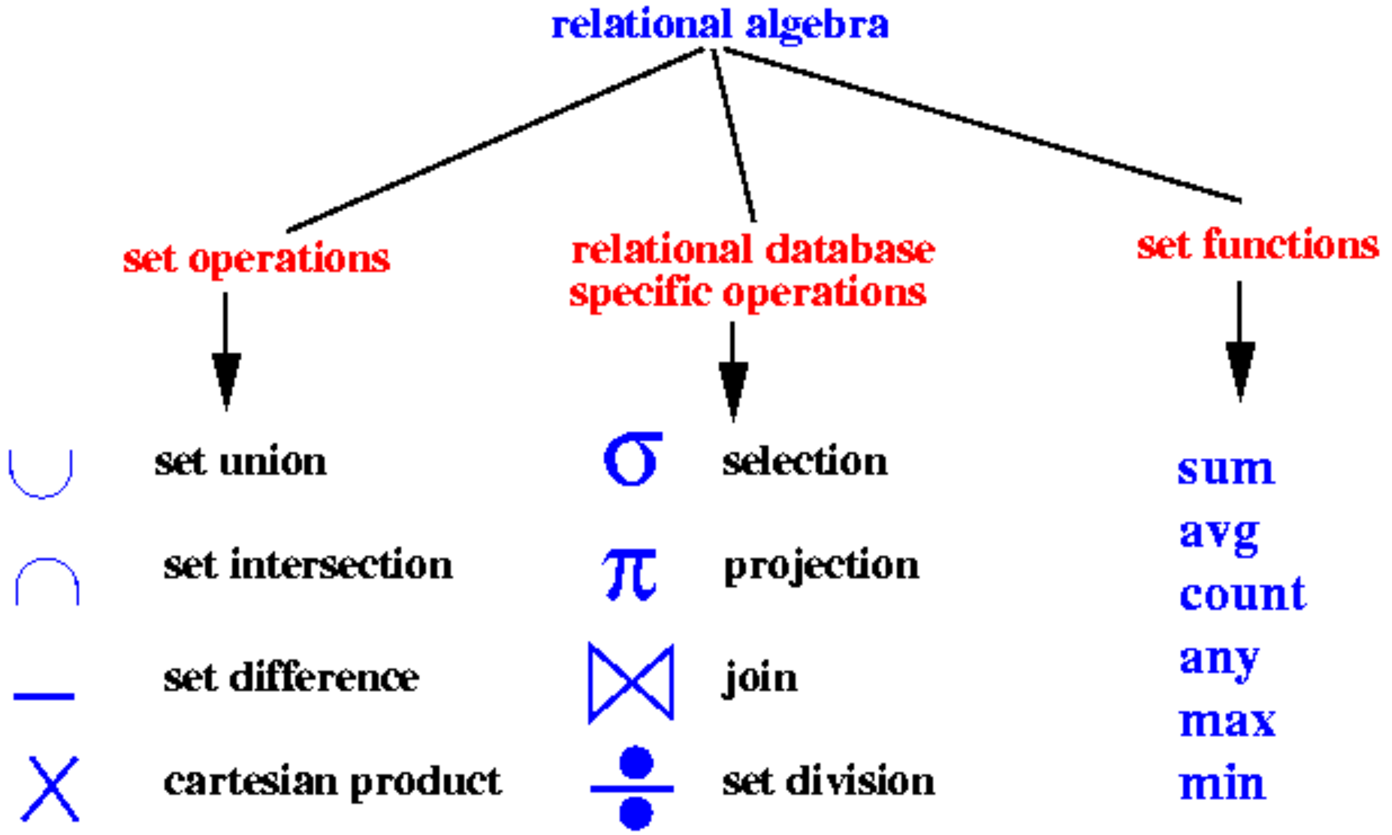




$$\pi_{\text{full\_name, cid}}(\sigma_{\text{hours>50}}(\text{Teacher} \bowtie \text{Teaches})) \quad = \quad \pi_{\text{full\_name, cid}}(\text{Teacher} \bowtie \sigma_{\text{hours>50}}(\text{Teaches}))$$

Recall Relational Algebra

- Fundamental operations to retrieve and manipulate tupels in a relation



Source: https://medium.com/@sahirnambiar/relational-algebra-the-underpinnings-of-sql-74959481231a

Relational Algebra - Selection

- Syntax
  - $\sigma_{predicate}(R)$
- Choose a subset of the tuples from a relation that satisfies a selection predicate
  - Filtering using predicates
  - Combine multiple predicates

T(Tid, Cid)

Tid	Cid
a1	10
a2	11
a3	12
a3	13

$\sigma_{\text{Tid='a1'}}(\text{T})$

Tid	Cid
a1	10

SELECT *
FROM T
WHERE Tid='a1';

$\sigma_{\text{Tid='a3'} \wedge \text{Cid='12'}}(\text{T})$

Tid	Cid
a3	12

SELECT *
FROM T
WHERE Tid='a3' AND Cid='12';

Relational Algebra - Projection

- Syntax
  - $\pi_{A1,A2,\ldots An}(R)$

- Generate a relation with tuples that contains attributes specified
  - Can manipulate the values
  - Can rearrange the order

T(Tid, Cid)

Tid	Cid
a1	10
a2	11
a3	12
a3	13

$\pi_{\text{Tid,Cid*2}}\ (\sigma_{\text{Tid='a2'}}(\text{T}))$

Tid	Cid
a2	24

```
SELECT Tid, Cid*2
FROM T
WHERE Tid='a2';
```

Relational Algebra - Union

- Syntax
  - (A ∪ B)

- Collect all tuples in both relations

A(Tid, Cid)

Tid	Cid
a1	10
a2	11

B(Tid, Cid)

Tid	Cid
a2	11
a3	13

(A ∪ B)

Tid	Cid
a1	10
a2	11
a2	11
a3	13

```
(SELECT * FROM A)
UNION ALL
(SELECT * FROM B);
```

Relational Algebra - Intersection

- Syntax
  - (A ∩ B)

- Collect tuples that appear in both relations

A(Tid, Cid)

Tid	Cid
a1	10
a2	11

B(Tid, Cid)

Tid	Cid
a2	11
a3	13

(A ∩ B)

Tid	Cid
a2	11

```
(SELECT * FROM A)
INTERSECT
(SELECT * FROM B);
```

Relational Algebra - Product

- Syntax
  - (A x B)

- Generate a relation that contains all possible combinations of the tuples from both relations

SELECT * FROM A, B;

SELECT * FROM A CROSS JOIN B;

A(Tid, Cid)

Tid	Cid
a1	10
a2	11

B(Tid, Cid)

Tid	Cid
a2	11
a3	13

(A x B)

A.Tid	A.Cid	B.Tid	B.Cid
a1	10	a2	11
a1	10	a3	13
a2	11	a2	11
a2	11	a3	13

Relational Algebra - Join

- Syntax
  - (A ⋈ B)

- $R \bowtie_{\text{<join condition>}} S$
- Generate a relation that contains all tuples with a common value(s) of one (or more) attrubute(s)

A(Tid, Cid)

Tid	Cid
a1	10
a2	11

B(Tid, Cid)

Tid	Cid
a2	11
a3	13

(A ⋈ B)

Tid	Cid
a2	11

SELECT * FROM A NATURAL JOIN B;

The Division operation ÷

- Syntax
  - R ÷ S

- Produces a relation R(X) that includes all tuples t[X] in S(Z) that appear in R in combination with every tuple from S(Y), where Z = X ∪ Y

- "An example is Retrieve the names of employees who work on all the projects that 'John Smith' works on."

Figure 8.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

T

B
b1
b4

Textbook p.256

Aggregate Functions and Grouping

- Syntax
  - <grouping attribute> $\Im$ <function list>(R)

Where

- <grouping attributes> is a list of attributes of the relation specified in R; and
- <function list> is a list of (<function> <attribute>) pairs. In each such pair;
- <function> is one of the allowed functions—such as SUM, AVERAGE, MAXIMUM, MINIMUM, COUNT—and <attribute> is an attribute of the relation specified by R.

$\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(_{\text{Dno}}\ \Im\ _{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$

(a) R

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Figure 8.10
The aggregate function operation.

a. $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}(_{\text{Dno}}\ \Im\ _{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$.

b. $_{\text{Dno}}\ \Im\ _{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

c. $\Im\ _{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

Relational Algebra and operations

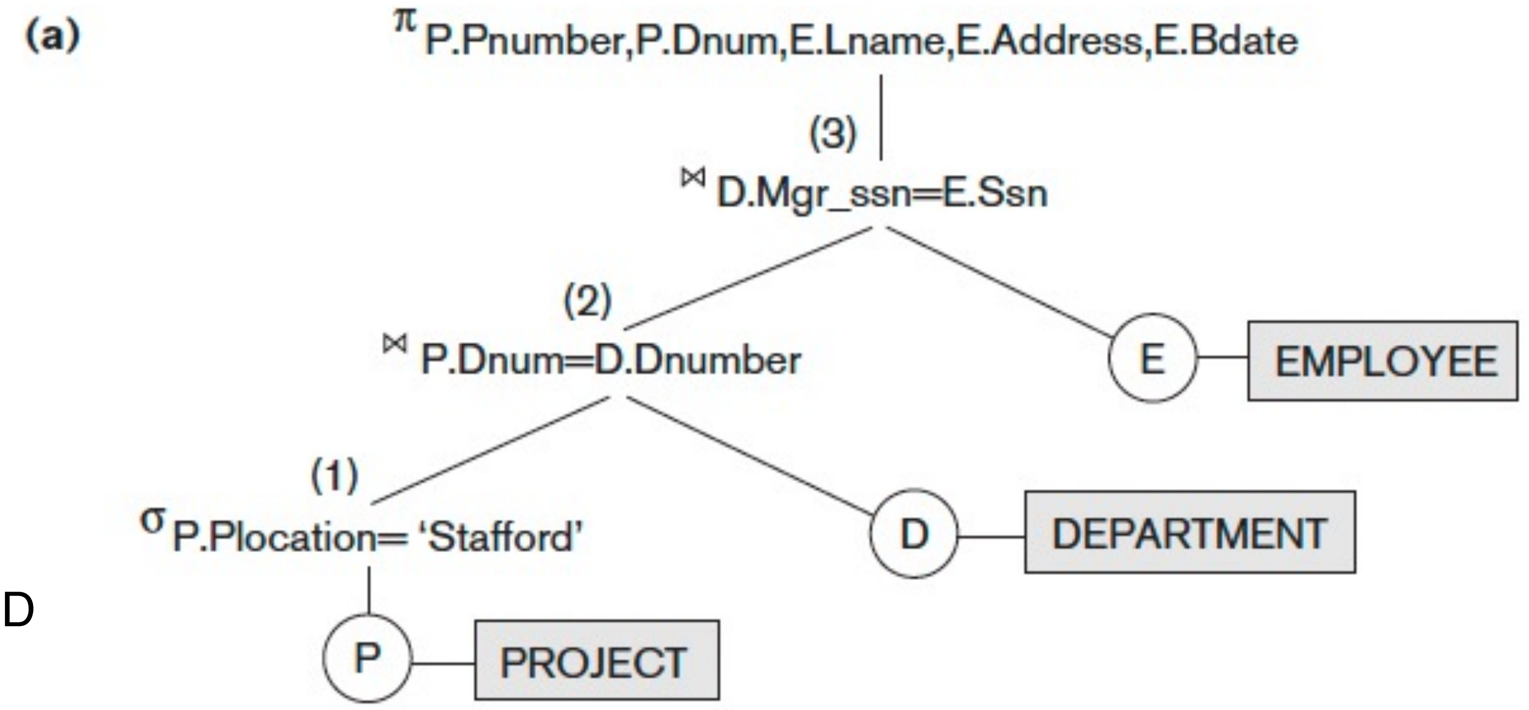
- The complete set of relational algebra operations {σ, π, ∪, ρ, –, x}
  - Select operation σ
  - Project operation π
  - Union operation ∪
  - Rename operation ρ
  - Set difference –
  - Cartesian Product x

- *The division operation ÷

Query Tree Examples

```
SELECT P.Pnumber,
       P.Dnum,
       E.Lname,
       E.Address,
       E.Bdate
FROM   PROJECT P,
       DEPARTMENT D,
       EMPLOYEE E
WHERE  P.Dnum=D.Dnumber AND
       D.Mgr_ssn=E.Ssn AND
       P.Plocation= 'Stafford';
```

(a) Query tree corresponding to the relational algebra expression



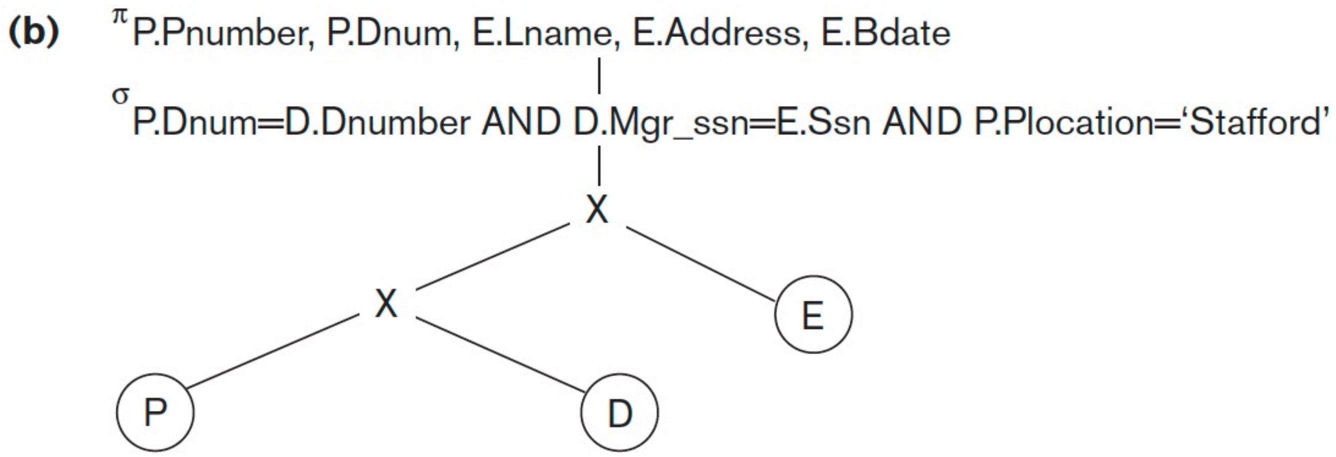
$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$

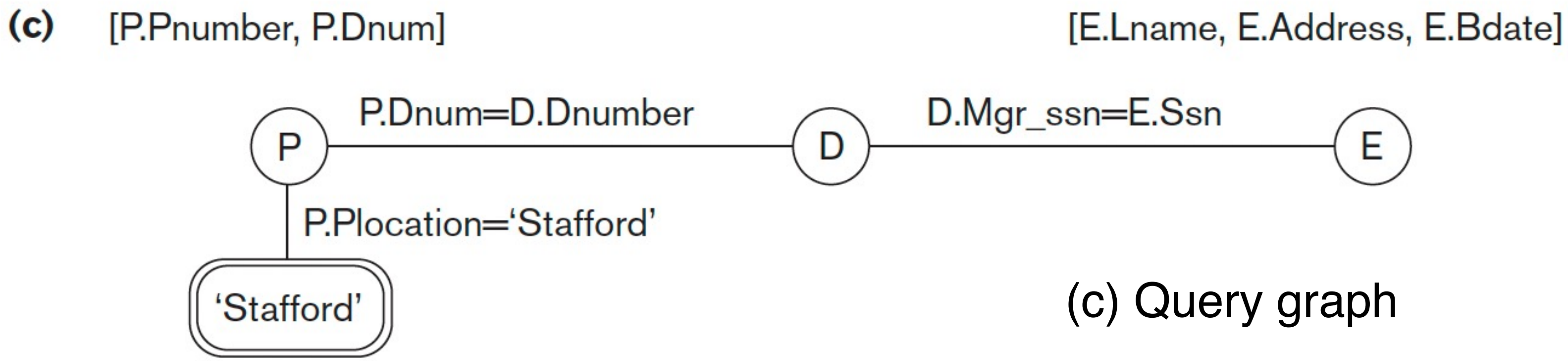
Textbook p.693

Query Tree Examples

```sql
SELECT P.Pnumber,
       P.Dnum,
       E.Lname,
       E.Address,
       E.Bdate
FROM   PROJECT P,
       DEPARTMENT D,
       EMPLOYEE E
WHERE  P.Dnum=D.Dnumber AND
       D.Mgr_ssn=E.Ssn AND
       P.Plocation= 'Stafford';
```

(b) Initial (canonical) query tree for the SQL query





(c) Query graph

$$\pi_{\text{Pnumber, Dnum, Lname, Address, Bdate}}(((\sigma_{\text{Plocation='Stafford'}}(\text{PROJECT})) \bowtie_{\text{Dnum=Dnumber}}(\text{DEPARTMENT})) \bowtie_{\text{Mgr\_ssn=Ssn}}(\text{EMPLOYEE}))$$

Textbook p.693

General Transformation Rules for Relational Algebra Operations

- Cascade of σ. A conjunctive selection condition can be broken up into a cascade (that is, a sequence) of individual σ operations:

$$\sigma_{c_1 \text{ AND } c_2 \text{ AND } \ldots \text{ AND } c_n}(R) \equiv \sigma_{c_1}(\sigma_{c_2}(\ldots(\sigma_{c_n}(R))\ldots))$$

- Perform filtering as early as possible
- Break a complex predicate and push them down

General Transformation Rules for Relational Algebra Operations

- Joins $\bowtie$
- Commutative and associative

$$R \bowtie_c S \equiv S \bowtie_c R$$

$$(R \,\theta\, S) \,\theta\, T \equiv R \,\theta\, (S \,\theta\, T) \qquad \theta \in \{\cup, \cap, \times, \bowtie\}$$

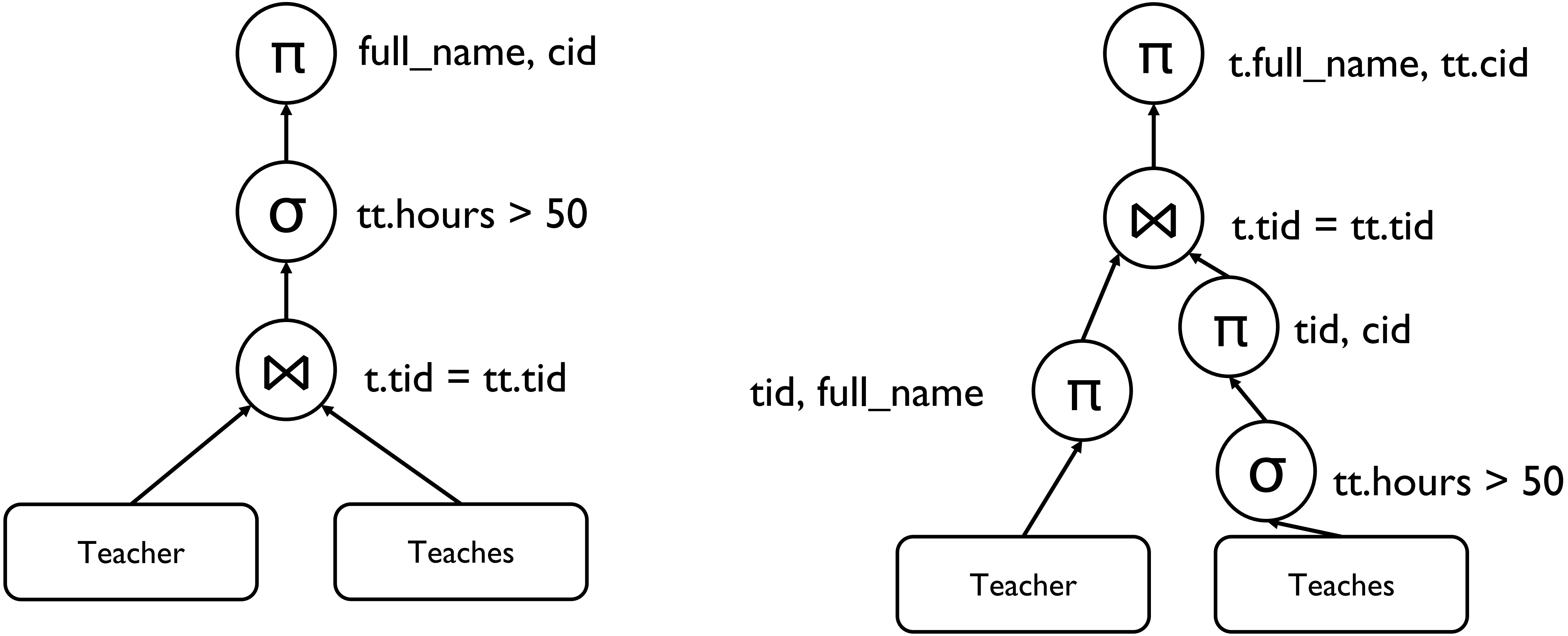
- The number of joins orderings for an n-way join is 4^n
  - Exhaustively enumerating all data is too slow…

General Transformation Rules for Relational Algebra Operations

- Projections
  - Perform projects early to create smaller tuples and reduce intermediate results
  - Only takes the attributes that are required (e.g. joining keys) or requested by the query

Example – Projection Pushdown

```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid
        AND tt.hours > 50
```



$\pi_{\text{full\_name, cid}}(\sigma_{\text{hours>50}}(\text{Teacher} \bowtie \text{Teaches}))$

General Transformation Rules for Relational Algebra Operations

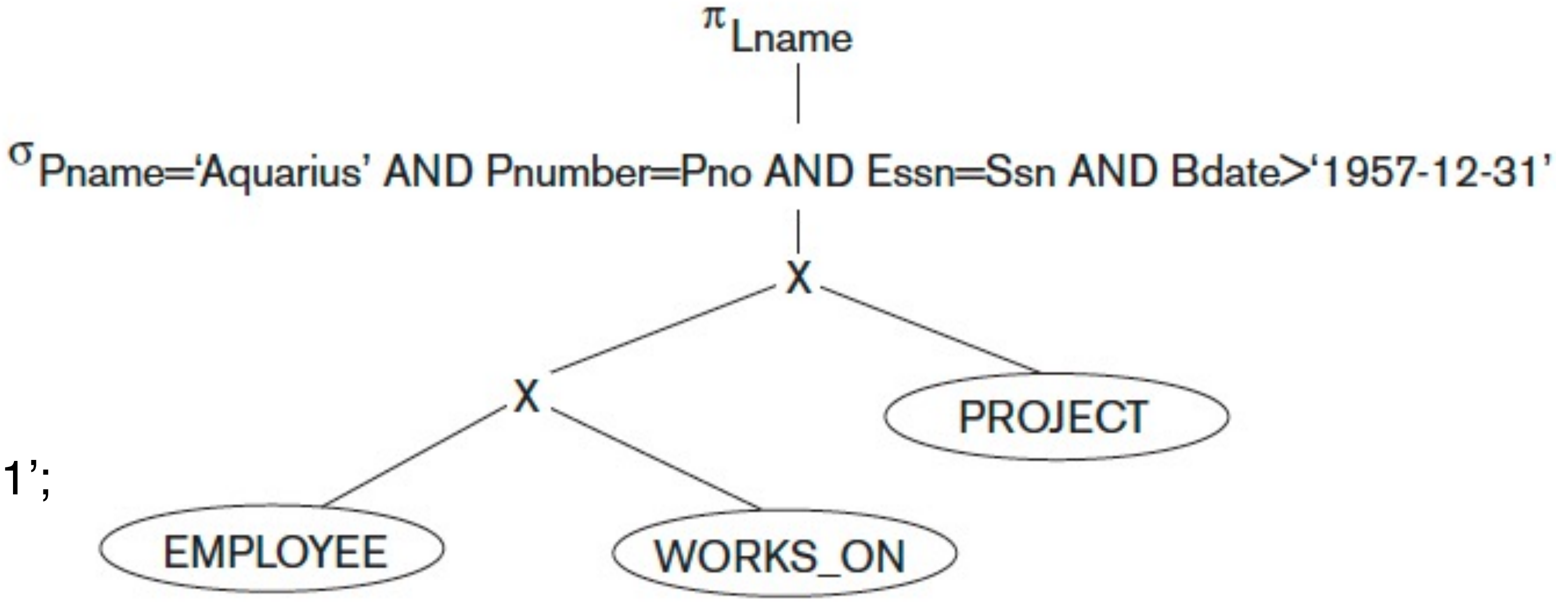
- Textbook page 698 – 699
- Cascade of selection and projection
- Commutativity of Selection
- Commuting selection with projection
- Commutativity of joins
- Commuting selection with joins
- Commuting projection with joins
- Commutativity of set operations
- Associativity of joins, x, union and intersect
- …

General Transformation Rules for Relational Algebra Operations

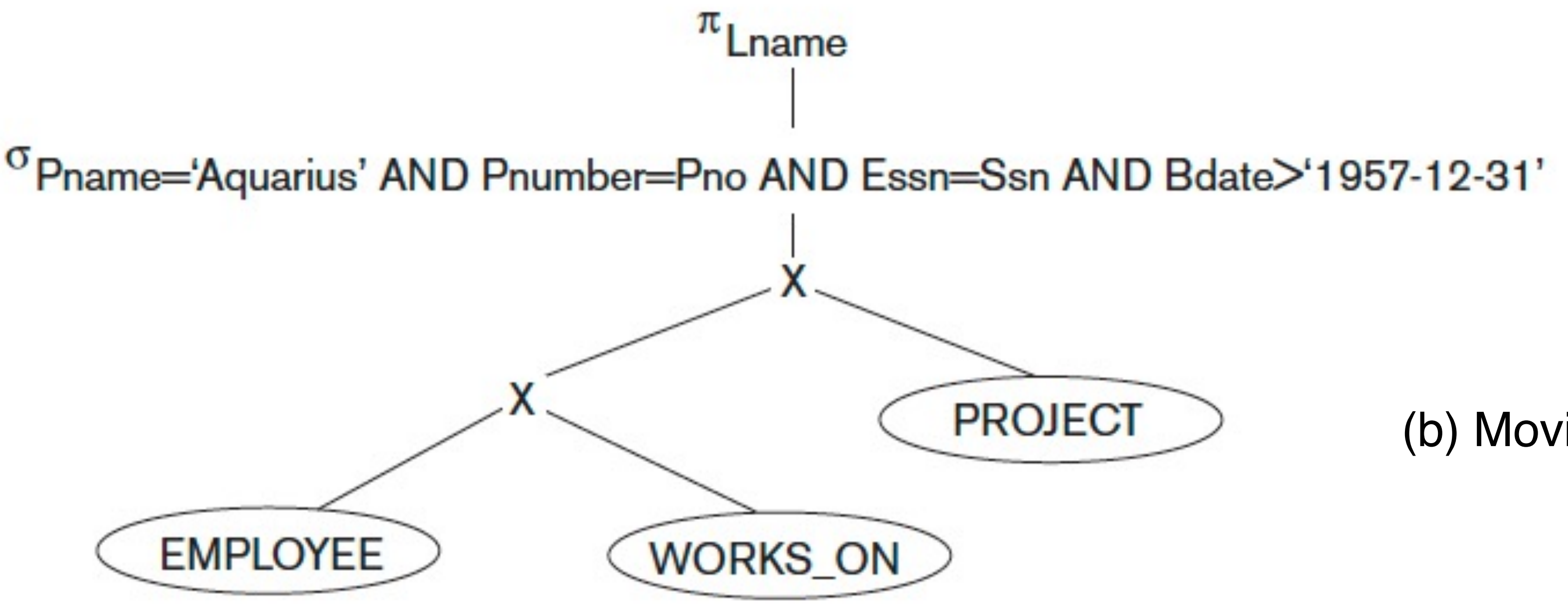
- Outline of a Heuristic Algebraic Optimization Algorithm
  - Using Rule 1, break up any SELECT operations with conjunctive conditions into a cascade of SELECT operations.
  - Using Rules 2, 4, 6, and 10, 13, 14 concerning the commutativity of SELECT with other operations, move each SELECT operation as far down the query tree as is permitted by the attributes involved in the select condition.
  - Using Rules 5 and 9 concerning commutativity and associativity of binary operations, rearrange the leaf nodes of the tree using the following criteria.
  - Using Rule 12, combine a CARTESIAN PRODUCT operation with a subsequent SELECT operation in the tree into a JOIN operation, if the condition represents a join condition.
  - 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.

Example transforming a query

```
SELECT E.Lname
FROM   EMPLOYEE E,
       WORKS_ON W,
       PROJECT P
WHERE  P.Pname='Aquarius'
       AND P.Pnumber=W.Pno
       AND E.Essn=W.Ssn
       AND E.Bdate > '1957-12-31';
```

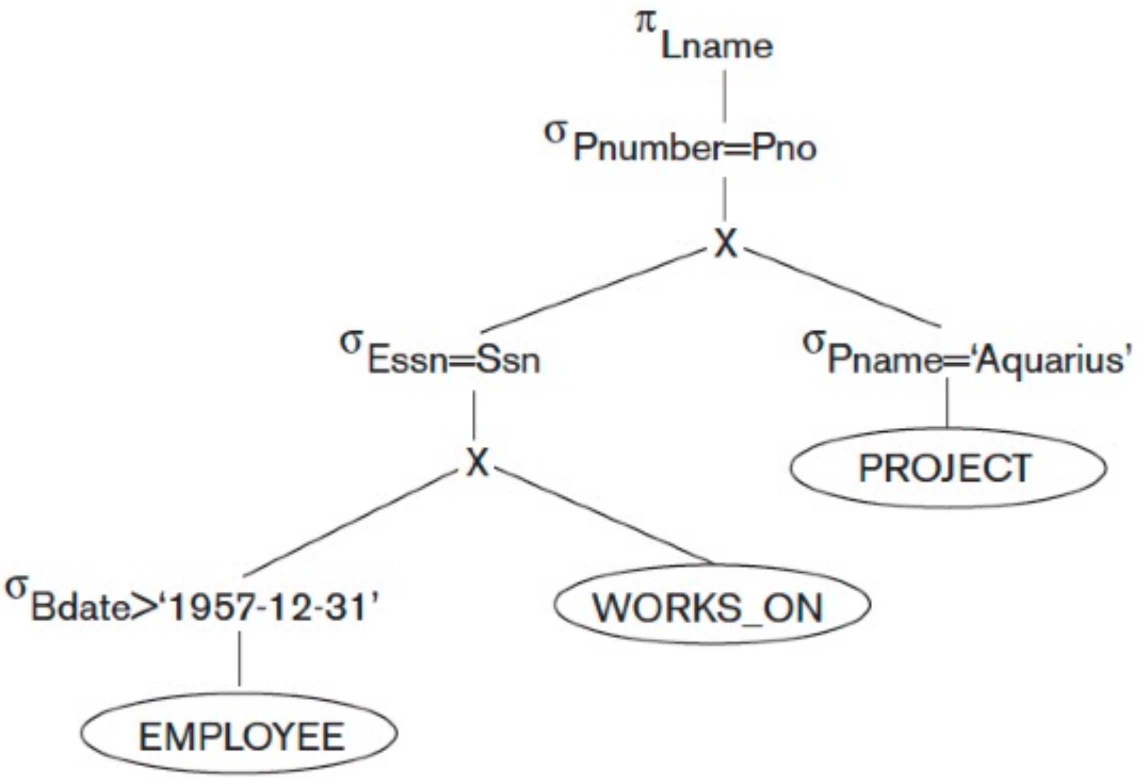


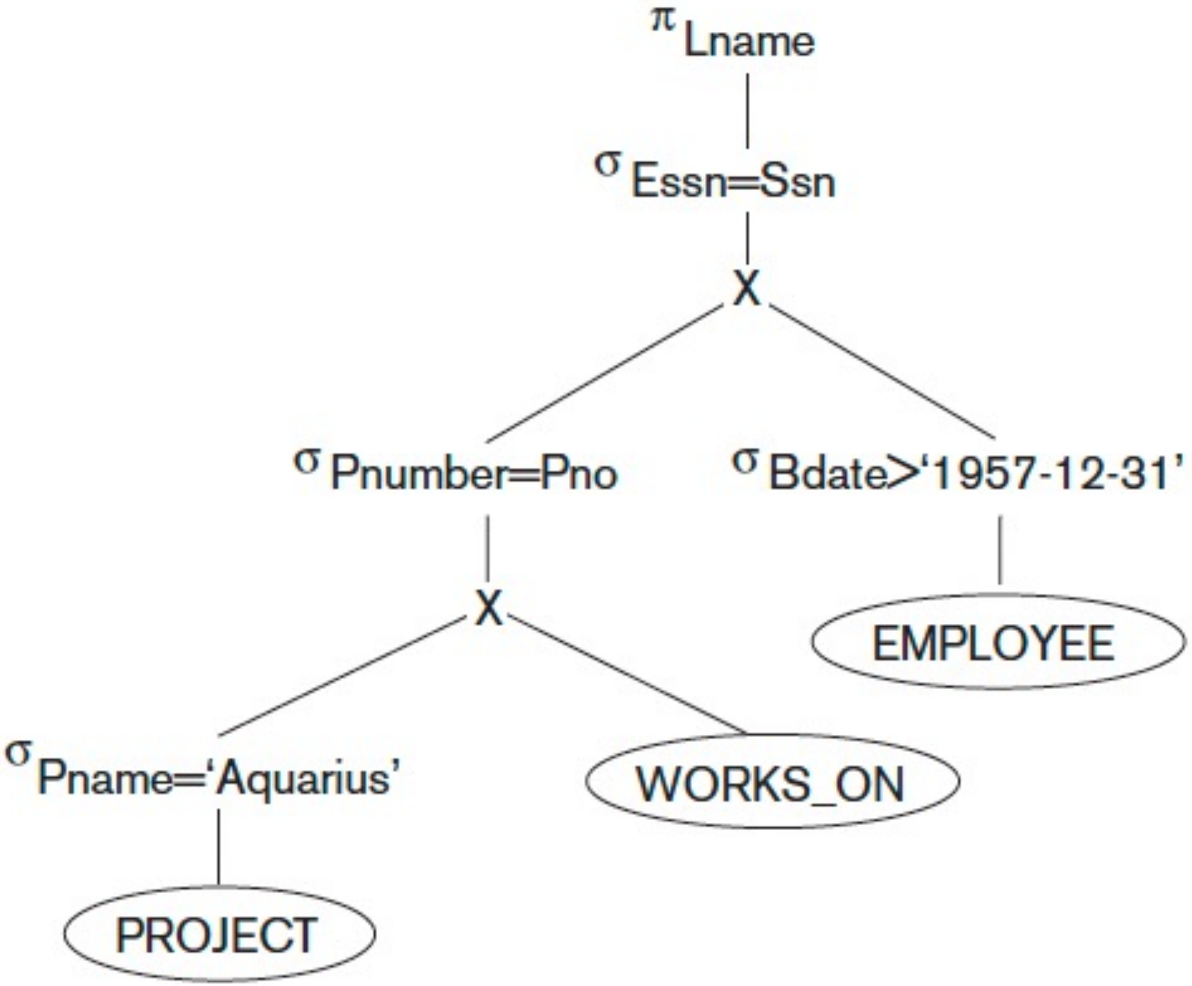
(a) Initial (canonical) query tree for SQL query



(a) Initial (canonical) query tree for SQL query

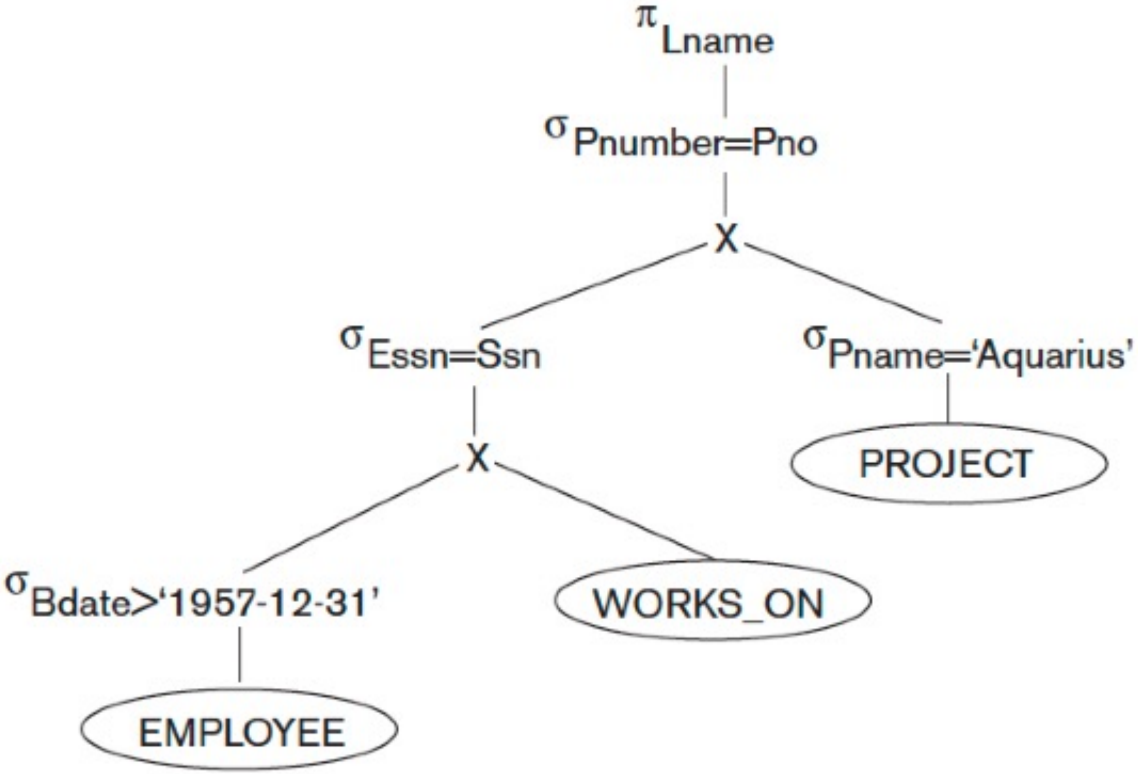
(b) Moving SELECT operations down the query tree.



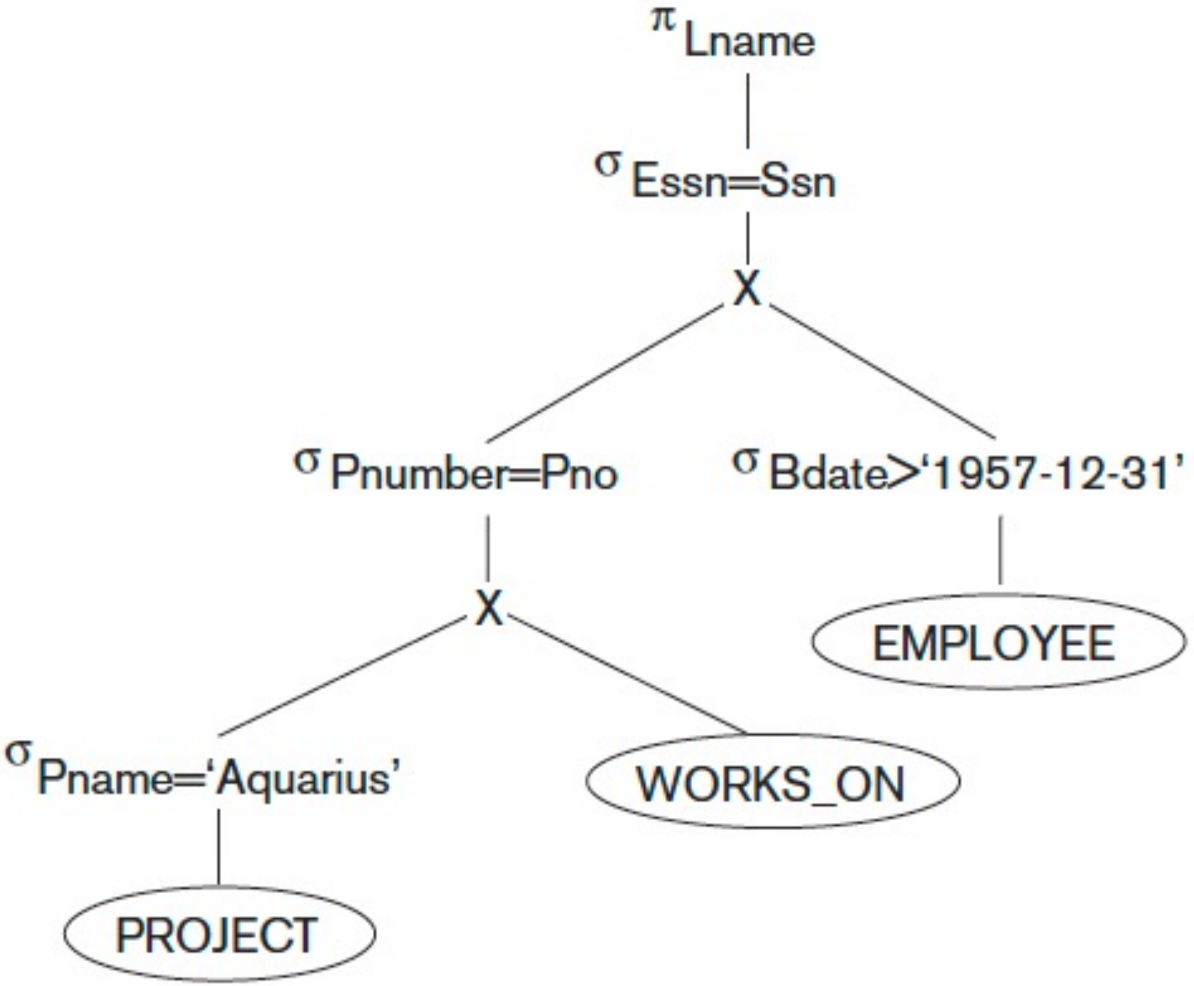


(c) Applying the more restrictive SELECT operation first

(b) Moving SELECT operations down the query tree.

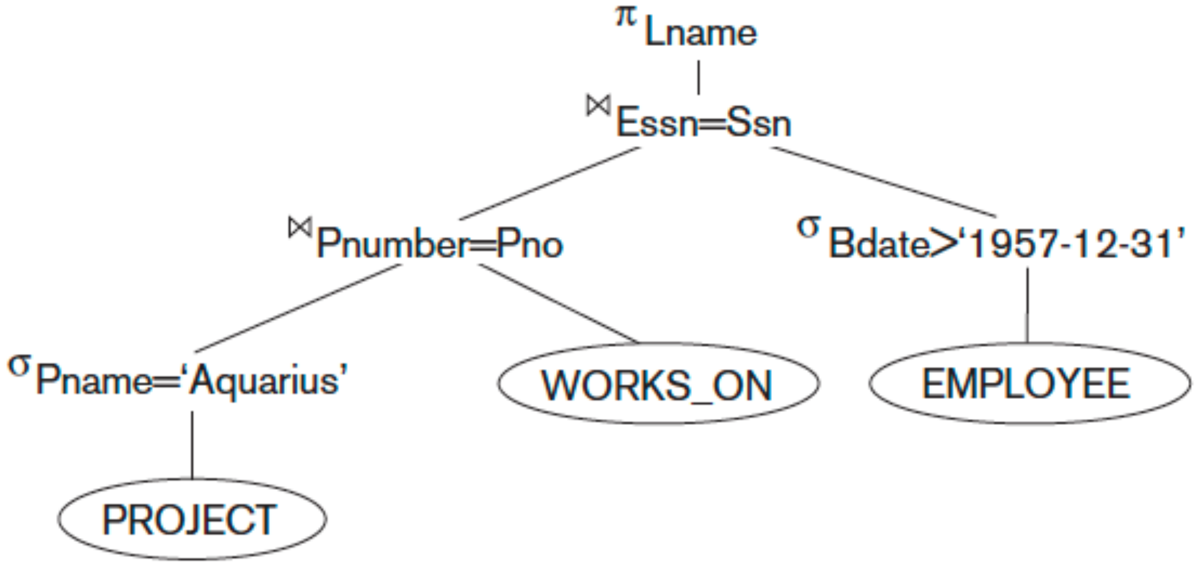


Textbook p.696



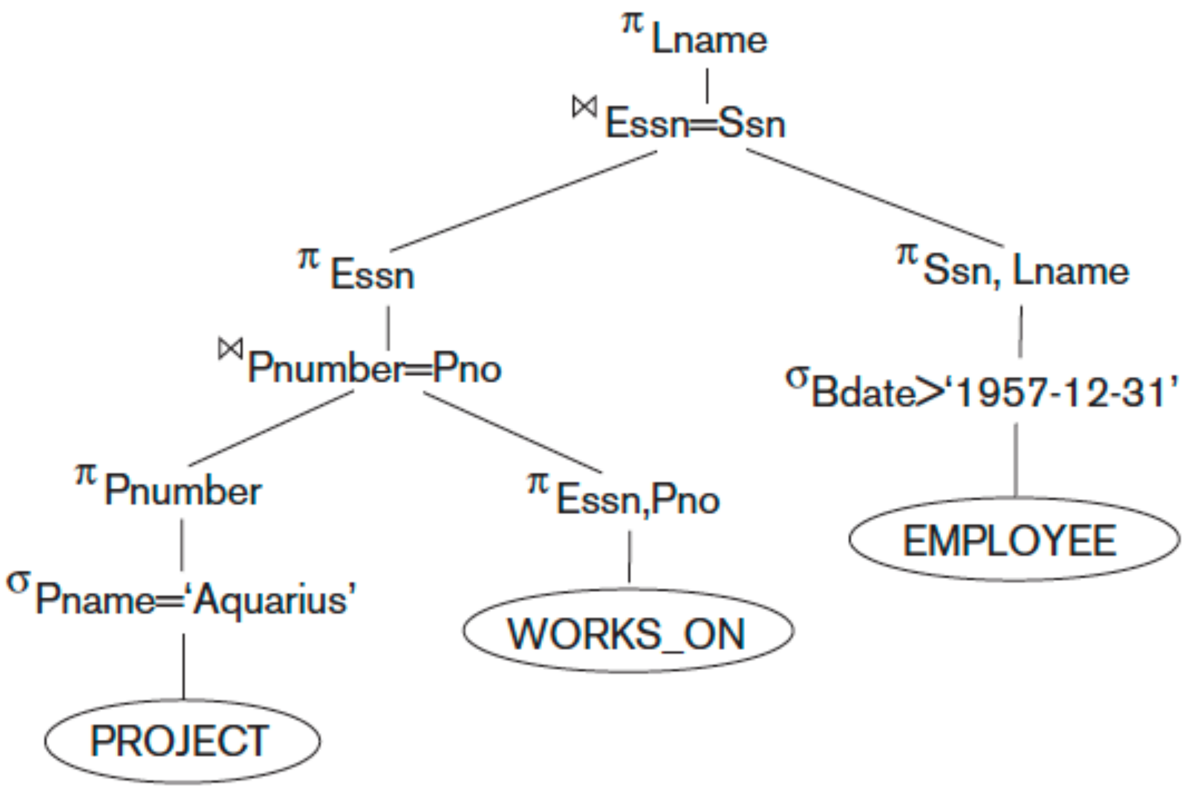
(c) Applying the more restrictive SELECT operation first

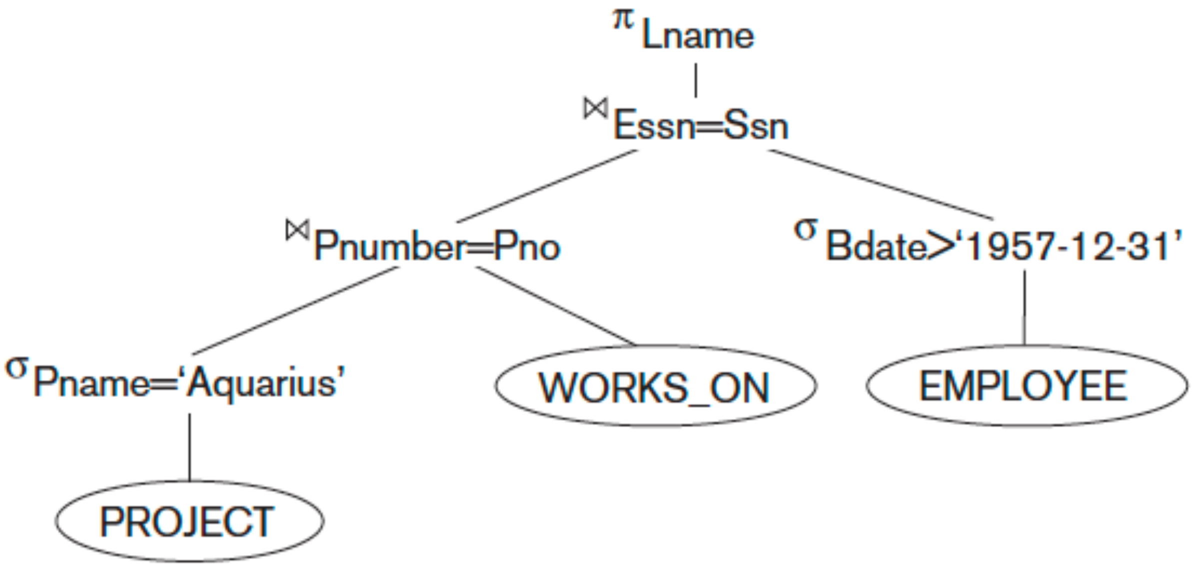
(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations



Textbook p.696

(d) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations

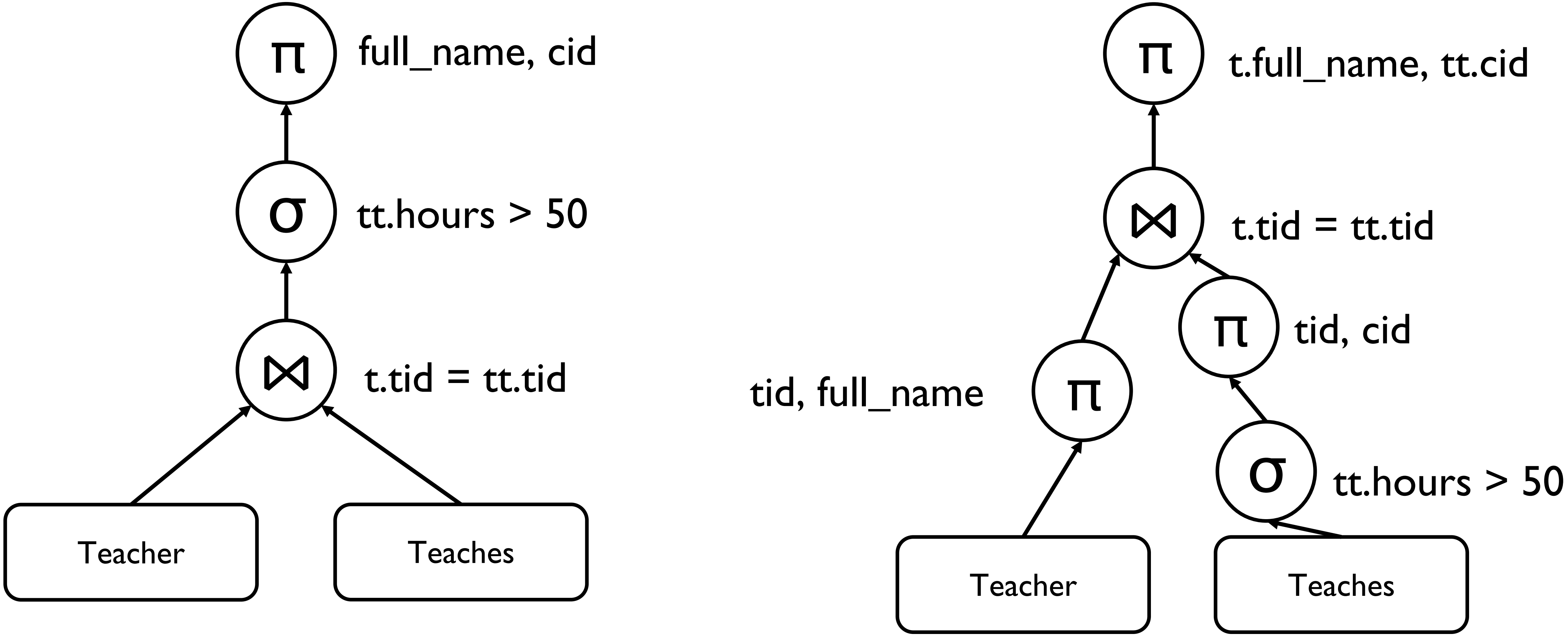




(e) Moving PROJECT operations down the query tree.

Example – Projection Pushdown

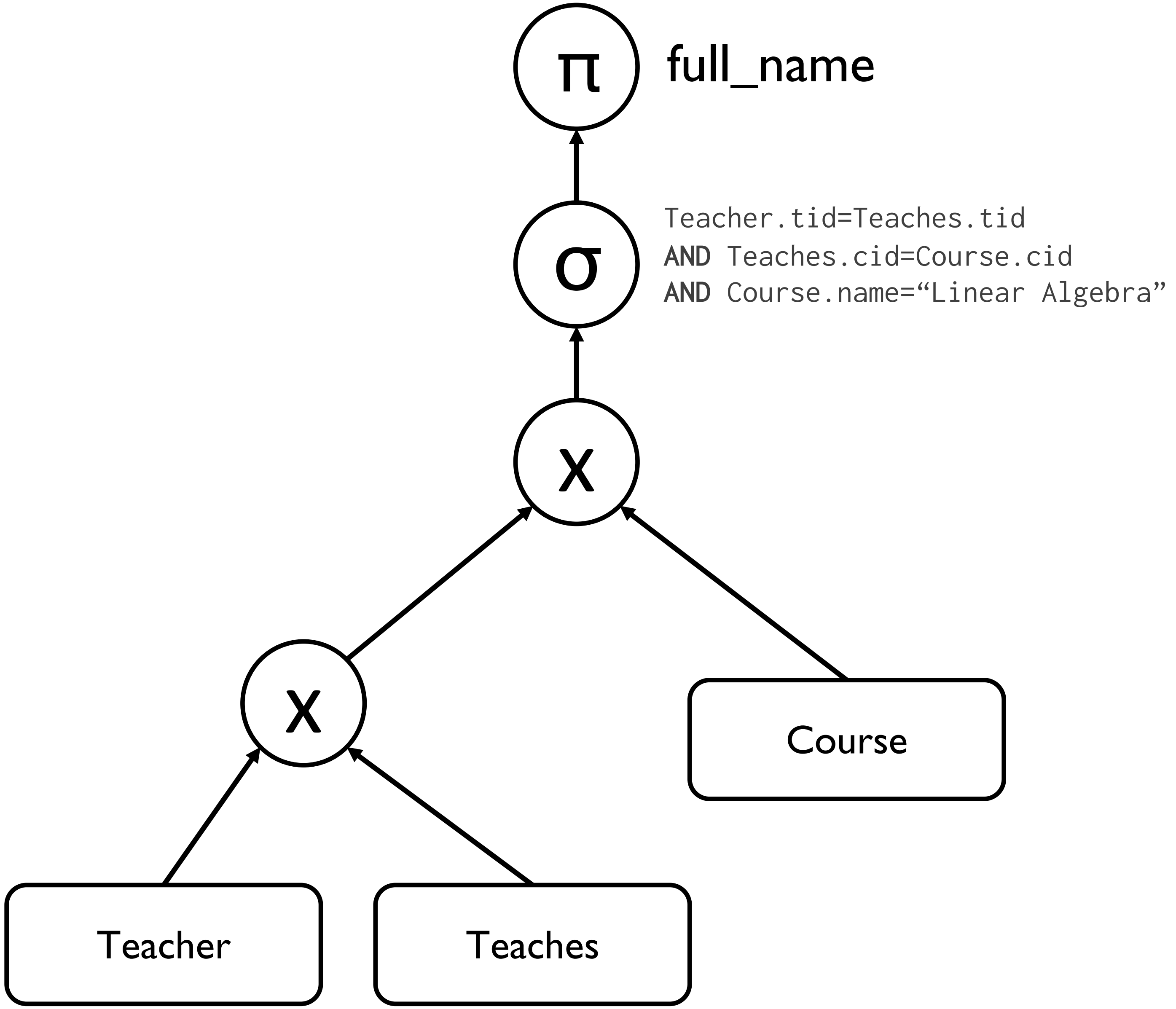
```
SELECT t.full_name, tt.cid
FROM Teacher AS t, Teaches AS tt
WHERE t.tid = tt.tid
        AND tt.hours > 50
```



$$\pi_{\text{full\_name, cid}}(\sigma_{\text{hours}>50}(\text{Teacher} \bowtie \text{Teaches}))$$

Example

```
SELECT Teacher.full_name
FROM  Teacher, Teaches, Course
WHERE Teacher.tid=Teaches.tid
      AND Teaches.cid=Course.cid
      AND Course.name="Linear Algebra"
```

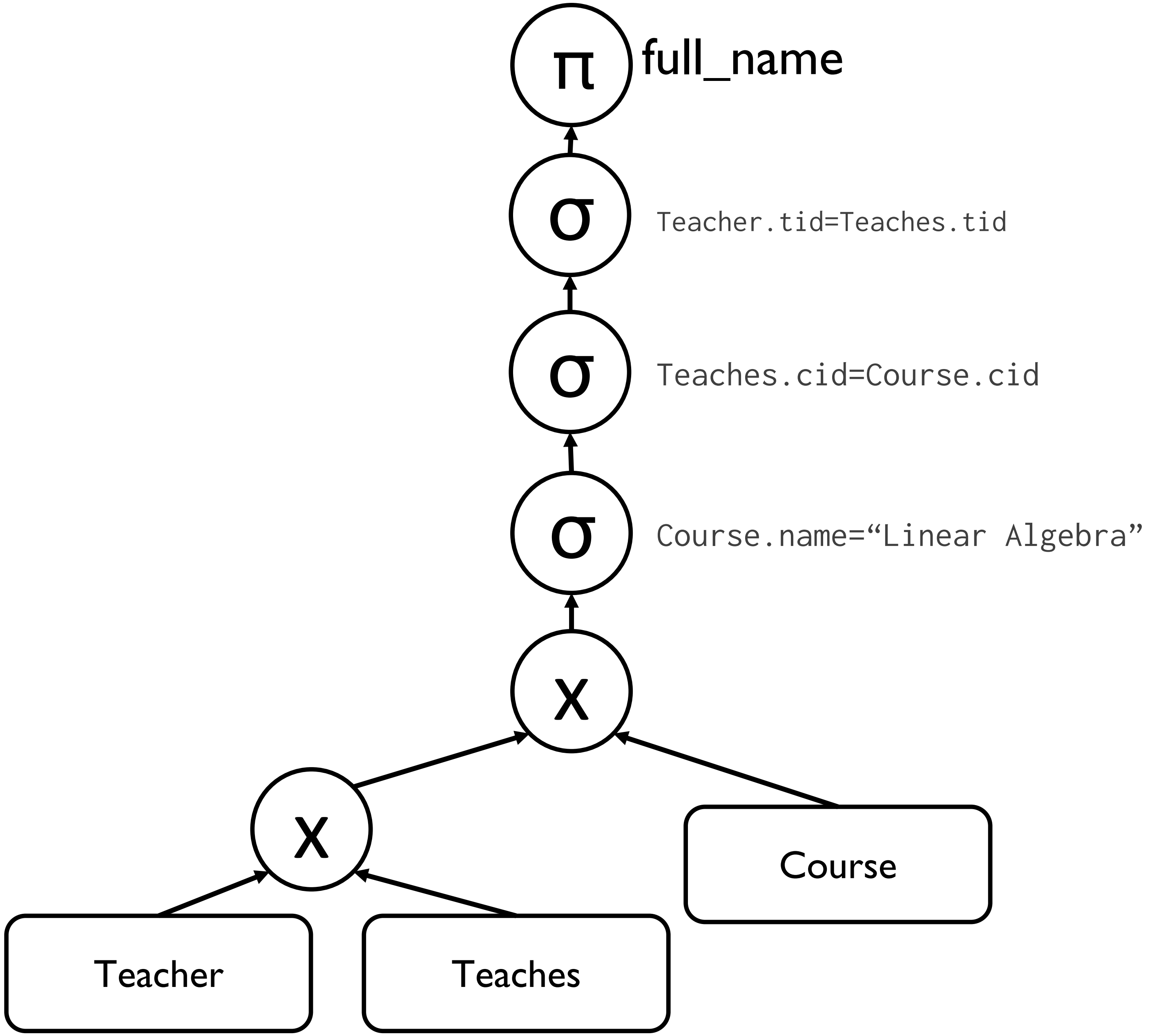


$$\pi_{\text{full\_name}}(\sigma_{\text{t.tid=tt.tid AND t.cid=c.cid AND c.name="Linear Algebra"}}(\text{Teacher} \bowtie \text{Teaches} \bowtie \text{Course}))$$

Example

```
SELECT Teacher.full_name
FROM  Teacher, Teaches, Course
WHERE Teacher.tid=Teaches.tid
      AND Teaches.cid=Course.cid
      AND Course.name="Linear Algebra"
```

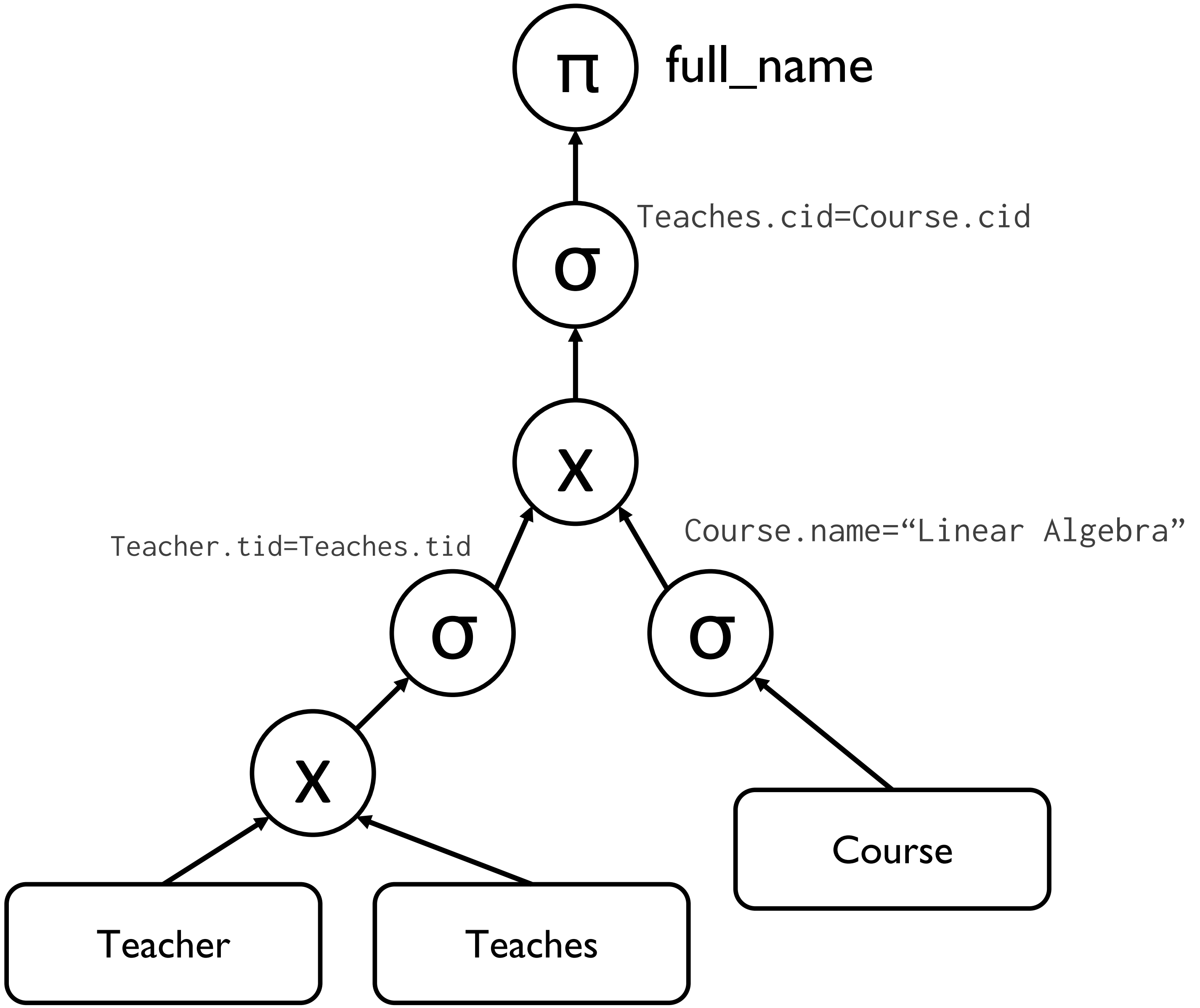
Decompose predicates into their simplest forms to make it easier for the optimizer to move them around.



Example

```
SELECT Teacher.full_name
FROM  Teacher, Teaches, Course
WHERE Teacher.tid=Teaches.tid
      AND Teaches.cid=Course.cid
      AND Course.name="Linear Algebra"
```

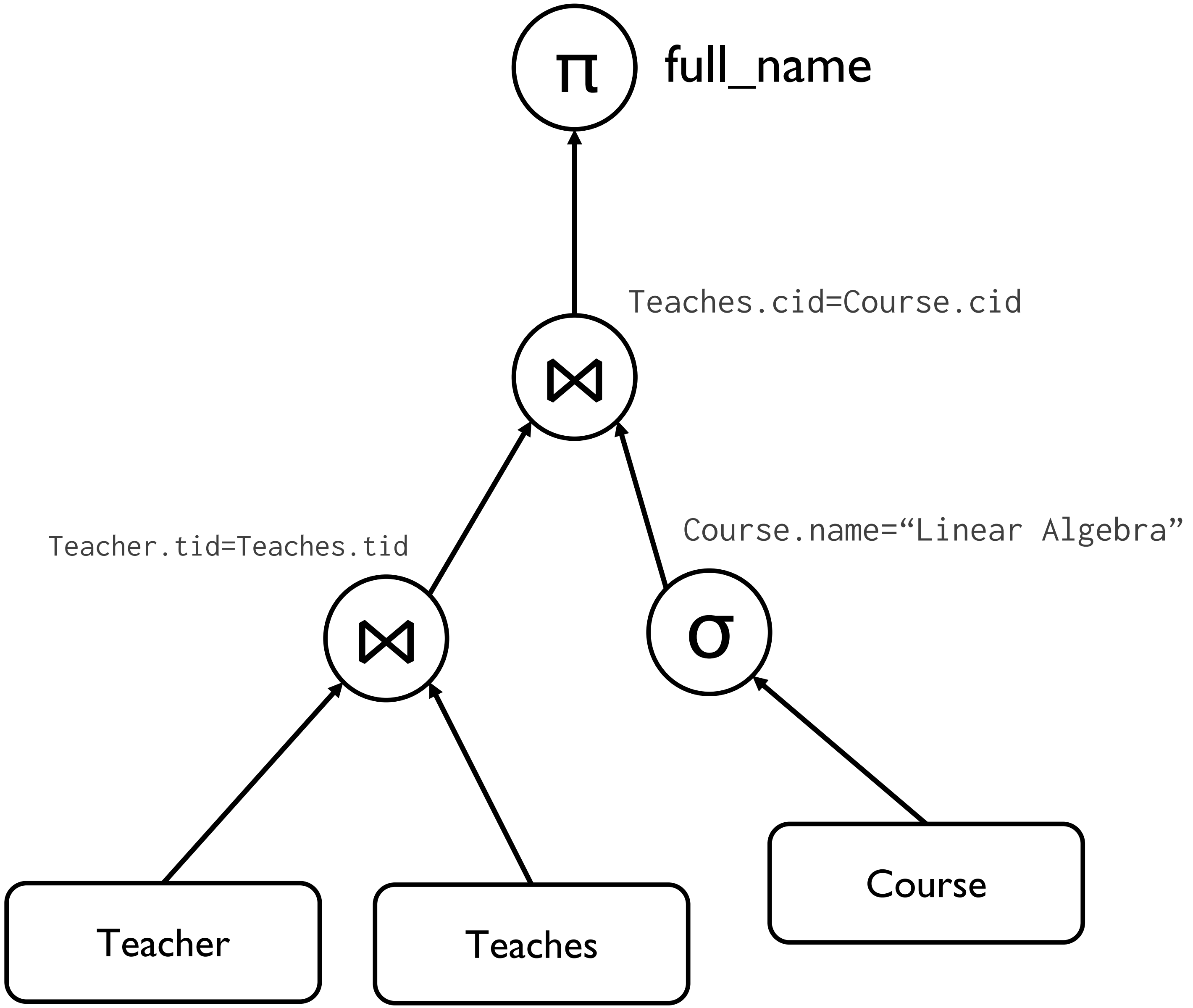
Move the predicate to the lowest point in the plan after Cartesian products.



Example

```
SELECT Teacher.full_name
FROM  Teacher, Teaches, Course
WHERE Teacher.tid=Teaches.tid
      AND Teaches.cid=Course.cid
      AND Course.name="Linear Algebra"
```

Replace all Cartesian Products with inner joins using the join predicates.



Example

```
SELECT Teacher.full_name
FROM  Teacher, Teaches, Course
WHERE Teacher.tid=Teaches.tid
      AND Teaches.cid=Course.cid
      AND Course.name="Linear Algebra"
```

Eliminate redundant attributes before pipeline breakers to reduce materialization cost.