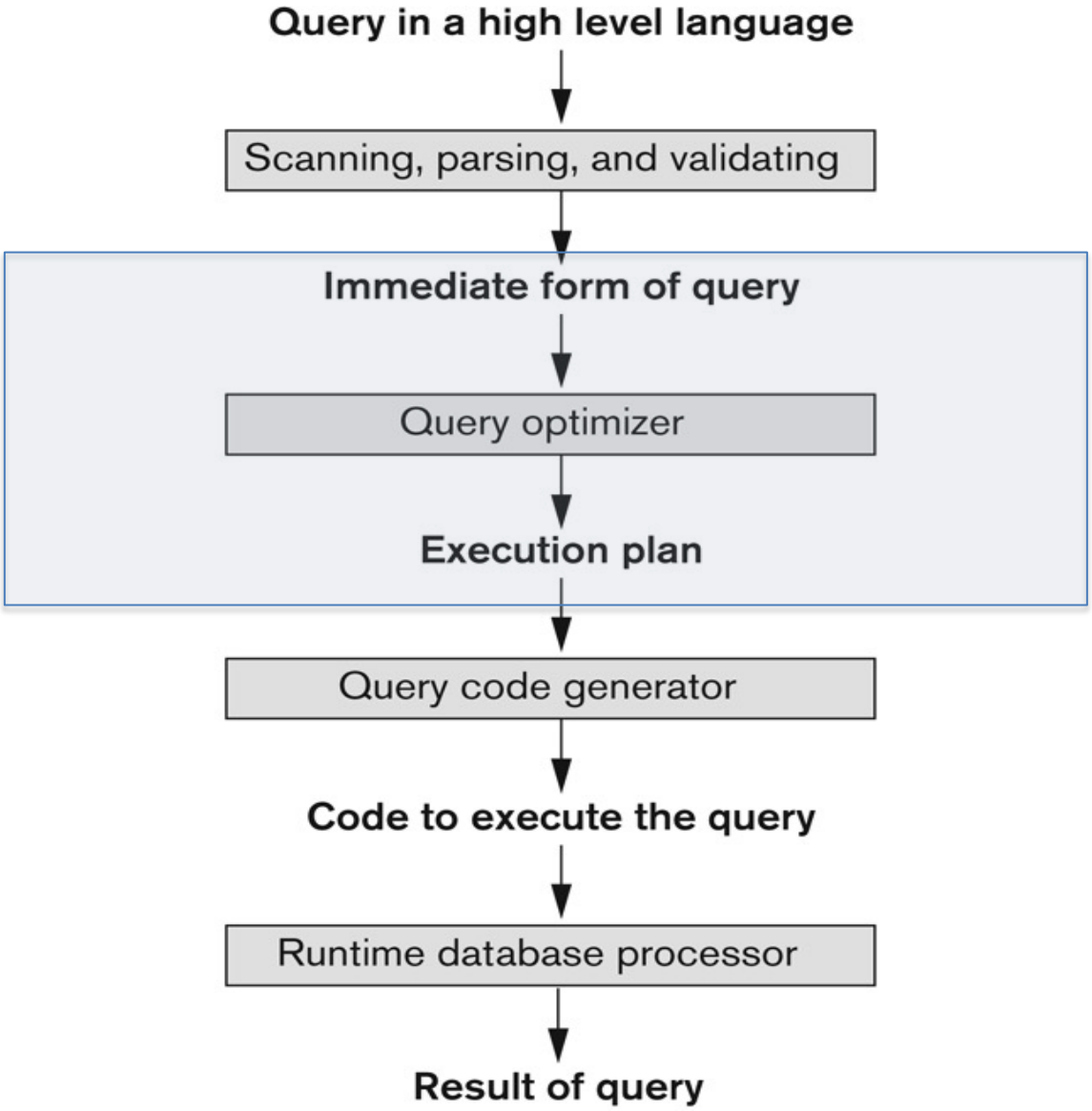
INFS7901
Database Principles

Relational Algebra and Query Processing and Optimization

Rocky Chen

Introduction to Query Processing



Conceptual Procedural Evaluation Strategy

1. Compute the cross-product of *relation-list*.
2. Discard resulting tuples if they fail *qualifications*.
3. Delete attributes that are not in *target-list*.
4. If DISTINCT is specified, eliminate duplicate rows.

Example of Conceptual Procedural Evaluation

SELECT Name

FROM MovieStar M, StarsIn S

WHERE S.StarID = M.StarID AND MovieID = 276

join selection

MovieStar **X** StarsIn

(StarID)	Name	Gender	MovieID	(StarID)	Character
1273	Nathalie Portman	Female	272	1269	Leigh Anne Touhy
1273	Nathalie Portman	Female	273	1270	Mary
1273	Nathalie Portman	Female	274	1271	King George VI
1273	Nathalie Portman	Female	276	1273	Nina Sayers
…	…	…	…	…	…

Query Optimization

- This strategy is probably the least efficient way to compute a query!
- A query typically has many possible execution strategies. The process of choosing a suitable one for processing a query is known as **query optimization**

The term optimization is actually inaccurate because in some cases the chosen execution plan is not the optimal (best) strategy. It is just a reasonably efficient strategy for executing the query.

To perform query optimization, we must first translate SQL queries to relational algebra.

Basic Relational Algebra Operations

Advanced Relational Algebra Operations

Implementation of SELECT and Join Operations

From Queries to Optimization

Relational Query Languages

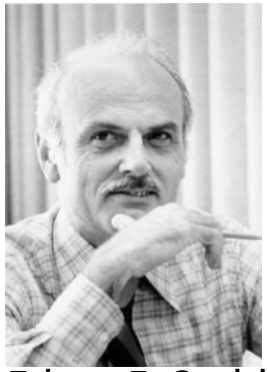
- Allow data manipulation and retrieval from a DB.

- Query Languages != Programming Languages
  - QLs provide easy access to large datasets.
  - Users do not need to know how to navigate through complicated data structures.

- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic
  - Allows for much optimization via *query optimizer*

Relational Algebra: A formal relational query language that forms the mathematical foundations for SQL.

Relational Algebra (RA)


Edgar F. Codd

- All operators take one or two relations as inputs and give. a new relation as a result
- operations:
  - *Selection* (σ): Selects a subset of rows from relation.
  - *Projection* (π): Deletes unwanted columns from relation.
  - *Set operations*(∪, ∩ , -): Use Union, intersection, and set difference to select a subset of rows from two relations.
  - *Cartesian-product* (x): Combines two relations.
  - *Rename* (ρ): Assigns a (another) name to a relation or a column.
  - *Join* (⋈):Combines two relations with some constraints.
  - *Assignment*(←): Assigns the result of an expression to a temporary relation.
  - *Division* (/): Allows for expressing queries that include a "**for all**" or "**for every**" phrase.

Example Movies Database

Movie(MovieID, Title, Year)

StarsIn(MovieID, StarID, Character)

MovieStar(StarID, Name, Gender)

Example Instances

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Gone with the Wind	1939
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

StarsIn:

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

MovieStar:

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

Selection (σ (sigma))

- Notation: $\sigma_{\mathbf{p}}(r)$
- Defined as:

$$\sigma_p(r) = \{t \mid t \in r \textbf{ and } p(t)\}$$

Set of tuples of r satisfying p

- **p** is called the selection predicate defining the selection condition in propositional logic.
- **The Result:** Selects rows that satisfy selection condition
- **Schema:** Identical to schema of input relation.

Selection Example

Movie:

MovieID	Title	Year
1	Star Wars	1977
2	Gone with the Wind	1939
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

$\sigma_{year > 1940}(Movie)$

MovieID	Title	Year
1	Star Wars	1977
4	Indiana Jones and the Raiders of the Lost Ark	1981

Selection Example #2

Find all male stars

Movie (MovieID, Title, Year)
StarsIn (MovieID, StarID, Role)
MovieStar (StarID, Name, Gender)

$$\sigma_{\text{Gender} = \text{'male'}}(\text{MovieStar})$$

StarID	Name	Gender
1	Harrison Ford	Male

Projection (π (pi))

- Notation: $\boldsymbol{\pi_{A1, A2, \ldots, Ak}}$ **(*r*)**

  where *A1, …,Ak* are attributes (the projection list) and *r* is a relation.

- **The result:** Deletes attributes that are not in projection list.
- **Schema:** Exactly the fields in the projection list, with the same names they had.

- Duplicate rows removed from result (relations are sets)

Result relation can be the input for another relational algebra operation!

Projection Examples

Movie

MovieID	Title	Year
1	Star Wars	1977
2	Gone with the Wind	1939
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

$\pi_{\text{Title, Year}}$ (Movie)

Title	Year
Star Wars	1977
Gone with the Wind	1939
The Wizard of Oz	1939
Indiana Jones and the Raiders of the Lost Ark	1981

π_{Year}(Movie)

Year
1977
1939
1981

What is $\pi_{\text{Title,Year}}(\sigma_{\text{year} > 1940}(\text{Movie}))$?

Title	Year
Star Wars	1977
Indiana Jones and the Raiders of the Lost Ark	1981

Projection Example #2

- Find the IDs of actors who have starred in movies.

Movie (MovieID, Title, Year)
StarsIn (MovieID, StarID, Role)
MovieStar (StarID, Name, Gender)

$\pi_{StarID}(StarsIn)$

StarID
1
2
3

Clicker Projection Example

- Suppose relation R(A,B,C) has the tuples:

A	B	C
1	2	3
4	2	3
4	5	6
2	5	3
1	2	6

- Compute the projection $\pi_{C,B}(R)$, and identify one of its tuples from the list below.

A. (2,3)
B. (4,2,3)
C. (6,4)
D. (6,5)
E. None of the above

Clicker Projection Example

- Suppose relation R(A,B,C) has the tuples:

A	B	C
1	2	3
4	2	3
4	5	6
2	5	3
1	2	6

- Compute the projection $\pi_{C,B}(R)$, and identify one of its tuples from the list below.

A. (2,3) Wrong order
B. (4,2,3) Not projected
C. (6,4) Wrong attributes
D. (6,5) right
E. None of the above

Selection and Projection Example

- Find the ids of movies made prior to 1950

MovieID	Title	Year
1	Star Wars	1977
2	Gone with the Wind	1939
3	The Wizard of Oz	1939
4	Indiana Jones and the Raiders of the Lost Ark	1981

$$\pi_{\text{MovieID}} (\sigma_{\text{year} < 1950} (\text{Movie}))$$

MovieID
2
3

Union, Intersection, Set-Difference

- Notation: $r \cup s$ $r \cap s$ $r - s$
- Defined as:

$$r \cup s = \{t \mid t \in r \textbf{ or } t \in s\}$$
$$r \cap s = \{t \mid t \in r \textbf{ and } t \in s\}$$
$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

- For these operations to be well-defined:

  1. *r, s* must have the *same arity* (same number of attributes)

  2. The attribute domains must be *compatible* (e.g., 2nd column of *r* has same domain of values as the 2nd column of *s*)

- **The result:** Union, intersection, or difference of the inputs.
- **Schema:** Identical to schema of the first input relation.

Union, Intersection, and Set Difference Examples

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

Singer

StarID	SName	Gender
3	Judy Garland	Female
4	Christine Lavin	Female

MovieStar ∪ Singer

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female
4	Christine Lavin	Female

Attributes compatible!

MovieStar ∩ Singer

StarID	Name	Gender
3	Judy Garland	Female

MovieStar – Singer

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female

Set Operator Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

Singer

StarID	Name	Gender
3	Judy Garland	Female
4	Christine Lavin	Female

Find the names of stars that are Singers but not MovieStars

$$\pi_{\text{Name}}(\text{Singers} - \text{MovieStars})$$

Name
Christine Lavin

Basic Relational Algebra Operations

Advanced Relational Algebra Operations

Implementation of SELECT and Join Operations

From Queries to Optimization

Cartesian (or Cross)-Product

- Notation: ***r* x *s***
- Defined as:

$$r \times s = \{ t \; q \mid t \in r \textbf{ and } q \in s \}$$

- **The results:** Each row of r is paired with row of s
- **Schema:** All of the attributes of r and all of the attributes of s with attribute names inherited if possible.

It is possible for r and s to have attributes with the same name, which creates a naming conflict. In this case, the attributes are referred to solely by position.

Cartesian Product Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

MovieStar x StarsIn

1	Name	Gender	MovieID	5	Character
1	Harrison Ford	Male	1	1	Han Solo
2	Vivian Leigh	Female	1	1	Han Solo
3	Judy Garland	Female	1	1	Han Solo
1	Harrison Ford	Male	4	1	Indiana Jones
2	Vivian Leigh	Female	4	1	Indiana Jones
3	Judy Garland	Female	4	1	Indiana Jones
…	…	…	…	…	…

Rename (ρ (rho))

- Allows us to name results of relational-algebra expressions.
- Notation: $\rho\,(E \rightarrow X)$

- **The result:** returns the expression *E* under the name *X*

- We can rename part of an expression, e.g.,

$\rho((\text{StarID} \rightarrow \text{ID}), \pi_{\text{StarID,Name}}(\text{MovieStar}))$

- We can also refer to positions of attributes, e.g.,

$\rho((1 \rightarrow \text{ID})), \pi_{\text{StarID,Name}}(\text{MovieStar})$

Cartesian Product Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

MovieStar x StarsIn

StarID1	Name	Gender	MovieID	StarID2	Character
1	Harrison Ford	Male	1	1	Han Solo
2	Vivian Leigh	Female	1	1	Han Solo
3	Judy Garland	Female	1	1	Han Solo
1	Harrison Ford	Male	4	1	Indiana Jones
2	Vivian Leigh	Female	4	1	Indiana Jones
3	Judy Garland	Female	4	1	Indiana Jones
…	…	…	…	…	…

$\rho((1 \rightarrow StarID1, 5 \rightarrow StarID2), MovieStar \times StarsIn)$

Joins ($\bowtie$)

- Theta Join is defined as

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

- **The result:** A selection on top of the cross-product.
- **Schema:** Same as cross-product.

The reference to an attribute of a relation R can be by position (R.i) or by name (R.name).

Theta Join Example

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

MovieStar $\bowtie_{\text{MovieStar.StarID} < \text{StarsIn.StarID}}$ StarsIn

1	Name	Gender	MovieID	5	Character
1	Harrison Ford	Male	2	2	Scarlett O'Hara
1	Harrison Ford	Male	3	3	Dorothy Gale
2	Vivian Leigh	Female	3	3	Dorothy Gale

Condition Join Clicker Example

- Compute R $\bowtie_{R.A < S.C \text{ and } R.B < S.D}$ S where:

R(A,B):

A	B
1	2
3	4
5	6

S(B,C,D):

B	C	D
2	4	6
4	6	8
4	7	9

Assume the schema of the result is (A, R.B, S.B, C, D). Which tuple is in the result?

A. (1,2,2,6,8)

B. (1,2,4,4,6)

C. (5,6,2,4,6)

D. All are valid

E. None are valid

Condition Join Clicker Example

- Compute R $\bowtie_{R.A < S.C \text{ and } R.B < S.D}$ S where:

R(A,B):

A	B
1	2
3	4
5	6

S(B,C,D):

B	C	D
2	4	6
4	6	8
4	7	9

Assume the schema of the result is (A, R.B, S.B, C, D). Which tuple is in the result?

A. (1,2,2,6,8) — (2,6,8) would have to be in S

B. (1,2,4,4,6) — (4,4,6) would have to be in S

C. (5,6,2,4,6) — Violates R.A < SC & R.B < S.D (5 > 2, and 6 = 6)

D. All are valid

E. None are valid — Correct

Equi-Join & Natural Join

- *Equi-Join*: A special case of Theta join where condition contains only ***equalities***
  - **Schema:** similar to cross-product, but contains only one copy of fields for which equality is specified.

- *Natural Join*: Equijoin on *all* common attributes
  - **Schema:** similar equi-join, but has only one copy of each common attribute.
  - No need to show the condition.
  - If the two attributes have no common attributes, this would be the same as cross product.

Equi and Natural Join Examples

MovieStar

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

StarsIn

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

$MovieStar \bowtie_{MovieStar.StarID = StarsIn.StarID} StarsIn$ *or*

$MovieStar \bowtie StarsIn$

StarID	Name	Gender	MovieID	Character
1	Harrison Ford	Male	1	Han Solo
1	Harrison Ford	Male	4	Indiana Jones
3	Judy Garland	Female	3	Dorothy Gale
2	Vivian Leigh	Female	2	Scarlett O'Hara

Join Example

- Find the names of all Movie Stars who were in any Movie.

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, role)
MovieStar(StarID, Name, Gender)

$$\pi_{name}(\text{MovieStar} \bowtie \text{StarsIn})$$

Name
Harrison Ford
Vivian Leigh
Judy Garland

Join Example

- Find names of actors who have starred in "Indiana Jones"

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, role)
MovieStar(<u>StarID</u>, Name, Gender)

$(\sigma_{\text{Title = "Indiana Jones ..."}}\ \text{Movie})$

MovieID	Title	Year
4	Indiana Jones and the Raiders of the Lost Ark	1981

$((\sigma_{\text{Title = "Indiana Jones ..."}}\ (\text{Movie})) \bowtie \text{StarsIn})$

MovieID	Title	Year	StarID	Character
4	Indiana Jones and the Raiders of the Lost Ark	1981	1	Indiana Jones

$(\pi_{\text{Name}}((\sigma_{\text{Title = "Indiana Jones ..."}}\ \text{Movie}) \bowtie \text{StarsIn} \bowtie \text{MovieStar}))$

Name
Harrison Ford

Clicker Exercise

Find the names of actors who have been in a movie with the same title as the actor's name

Which of the following does *not* do that correctly:

A. $\pi_{\text{Name}}((\text{Movie} \bowtie \text{StarsIn}) \bowtie_{\text{title = name } \wedge \text{ StarID = MovieStar.StarID}} \text{MovieStar})$

B. $\pi_{\text{Name}}(\text{MovieStar} \bowtie_{\text{Name = title } \wedge \text{ MovieStar.StarID = StarID}} (\text{StarsIn} \bowtie \text{Movie}))$

C. $\pi_{\text{Name}}((\text{StarsIn} \bowtie (\pi_{\text{StarID,Name}} \text{MovieStar})) \bowtie_{\text{MovieID = Movie.MovieID } \wedge \text{ title = name}} \text{Movie})$

D. All are correct

E. None are correct

Clicker Exercise

Find the names of actors who have been in a movie with the same title as the actor's name

Which of the following does *not* do that correctly:

A. $\pi_{\text{Name}}((\text{Movie} \bowtie \text{StarsIn}) \bowtie_{\text{title = name } \wedge \text{ StarID = MovieStar.StarID}} \text{MovieStar})$

B. $\pi_{\text{Name}}(\text{MovieStar} \bowtie_{\text{Name = title } \wedge \text{ MovieStar.StarID = StarID}} (\text{StarsIn} \bowtie \text{Movie}))$

C. $\pi_{\text{Name}}((\text{StarsIn} \bowtie (\pi_{\text{StarID,Name}} \text{MovieStar})) \bowtie_{\text{MovieID = Movie.MovieID } \wedge \text{ title = name}} \text{Movie})$

D. All are correct

E. None are correct

All are correct (D)

Assignment Operation

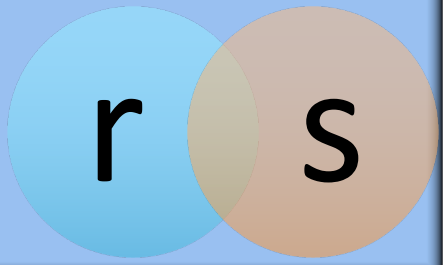
- Notation: $t \leftarrow E$

- **The result:** assigns the result of expression E to a temporary relation t.
- Used to break complex queries to small steps.
- Assignment is always made to a temporary relation variable.

Example: Write $r \cap s$ in terms of $\cup$ and –

$temp1 \leftarrow r - s$

$result \leftarrow r - temp1$



Assignment Example

- Find names of movie stars who have been in "Indiana Jones" and "Star Wars"

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, role)
MovieStar(StarID, Name, Gender)

1. Find id of movie stars in "Indiana Jones"

$$Indy \leftarrow \pi_{starID}((\sigma_{Title = \text{“Indiana Jones …”}} (Movie)) \bowtie StarsIn)$$

2. Find id of movie stars in "Star Wars"

$$StarWars \leftarrow \pi_{starID}((\sigma_{Title = \text{“Star Wars”}} Movie) \bowtie StarsIn)$$

3. Find ids of movie stars in both

$$Indywars \leftarrow Indy \cap StarWars$$

4. Find name of movie stars in both

$$\pi_{name}(Indywars \bowtie MovieStar)$$

Follow-up Example

- Find names of actors who have been in "Indiana Jones" or "Star Wars".

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, role)
MovieStar(StarID, Name, Gender)

$(\sigma_{\text{Title} = \text{“Indiana Jones …”} \vee \text{title} = \text{“Star Wars”}} (\text{Movie}))$

MovieID	Title	Year
1	Star Wars	1977
4	Indiana Jones and the Raiders of the Lost Ark	1981

$(\pi_{\text{Name}}((\sigma_{\text{Title} = \text{“Indiana Jones…”} \vee \text{title} = \text{“Star Wars”}} \text{Movie}) \bowtie \text{StarsIn} \bowtie \text{MovieStar})$

Name
Harrison Ford

Division

- Notation: r / s *or* $r \div s$
- Defined as

Capital π, removes duplicates

$$r / s = \{ t \mid t \in \prod_{r-s}(r) \wedge \forall u \in s (tu \in r) \}$$

Concatenation of t and u

- Useful for expressing queries that include a "**for all**" or "**for every**" phrase, e.g., *Find movie stars who were in **all** movies*.

- **Results:** identifies the attribute values from r that are found to be paired with all of the values from s.

- **Schema:** r(attributes) – s(attributes)
  - r = (A_1, …, A_m, B_1, …, B_n) and s = (B_1, …, B_n)
  - Schema r / s = (A_1, …, A_m)

Examples of Division A/B

A

sno	pno
s1	p1
s1	p2
s1	p3
s1	p4
s2	p1
s2	p2
s3	p2
s4	p2
s4	p4

B1

pno
p2

A/B1

sno
s1
s2
s3
s4

B2

pno
p2
p4

A/B2

sno
s1
s4

B3

pno
p1
p2
p4

A/B3

sno
s1

Examples of Division

- Find the names of actors who have been in all movies after 1950.

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, role)
MovieStar(StarID, Name, Gender)

1. Find ids of movies after 1950

$$\text{LateMovieIds} \leftarrow \pi_{\text{MovieID}}(\sigma_{\text{year} > 1950}(\text{Movie}))$$

2. Find ids of actors that were in all movies after 1950

$$\text{InAll} \leftarrow (\pi_{\text{StarID, MovieID}}(\text{StarsIn}) / \text{LateMovieIds})$$

3. Find names of actors that were in all movies after 1950

$$\pi_{\text{Name}}(\text{InAll} \bowtie \text{MovieStar})$$

Division Clicker Question

R

C	D	E
1	2	1
2	2	1
3	2	1

S

A	B
1	2
2	2
3	2
1	1

T

???
2

Which of the following is a possible expression for creating T?

A. $X(D) \leftarrow \pi_A S$
   $\pi_{C, D}(R)/X$
B. $Y(A) \leftarrow \pi_C R$
   $\pi_{B, A}(S)/Y$
C. $Z(C) \leftarrow \pi_A S$
   $\pi_{E, C}(R)/Z$
D. All of the above
E. None of the above

Answer A exposed

R

C	D	E
1	2	1
2	2	1
3	2	1

S

A	B
1	2
2	2
3	2
1	1

T

???
2

Which of the following is a possible expression for creating T?

A. $X(D) \leftarrow \pi_A S$

$\pi_{C, D}(R)/X$

C	D
1	2
2	2
3	2

D
1
2
3

Answer B exposed

R

C	D	E
1	2	1
2	2	1
3	2	1

S

A	B
1	2
2	2
3	2
1	1

T

???
2

Which of the following is a possible expression for creating T?

A. $X(D) \leftarrow \pi_A S$
$\pi_{C, D}(R)/X$

B. $Y(A) \leftarrow \pi_C R$
$\pi_{B, A}(S)/Y$

B	A
2	1
2	2
2	3
1	1

A
1
2
3

Answer C Exposed

R

C	D	E
1	2	1
2	2	1
3	2	1

S

A	B
1	2
2	2
3	2
1	1

T

???
2

Which of the following is a possible expression for creating T?

A. $X(D) \leftarrow \pi_A S$
$\pi_{C, D}(R)/X$

B. $Y(A) \leftarrow \pi_C R$
$\pi_{B, A}(S)/Y$

C. $Z(C) \leftarrow \pi_A S$
$\pi_{E, C}(R)/Z$

E	C
1	1
1	2
1	3

C
1
2
3

Division Clicker Question

R

C	D	E
1	2	1
2	2	1
3	2	1

S

A	B
1	2
2	2
3	2
1	1

T

???
2

Which of the following is a possible expression for creating T?

A. $X(D) \leftarrow \pi_A S$
$\pi_{C, D}(R)/X$

nothing

B. $Y(A) \leftarrow \pi_C R$
$\pi_{B, A}(S)/Y$

right

C. $Z(C) \leftarrow \pi_A S$
$\pi_{E, C}(R)/Z$

No, 1

D. All of the above

E. None of the above

Translating SQL Queries into Relational Algebra

- **Query block**: a single SELECT-FROM-WHERE expression, as well as GROUP BY and HAVING clause
- **Nested queries** are identified as separate query blocks.

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

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

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

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

$\mathcal{F}_{\text{MAX SALARY}} (\sigma_{\text{DNO=5}} (\text{EMPLOYEE}))$

Basic Relational Algebra Operations

Advance Relational Algebra Operations

Implementation of SELECT and Join Operations

From Queries to Optimization

Algorithms for SELECT Operations

Student(sID, sName, GPA, age)
College(cName, state, enrollment)
Apply(sID, cName, major, decision)

- Implementing the SELECT Operation

- Examples:
  - (OP1): $\sigma_{\text{cName='Stanford'}}$(College)
  - (OP2): $\sigma_{\text{sID}> 100}$ (Student)
  - (OP3): $\sigma_{\text{GPA} =3.2}$(Student)
  - (OP4): $\sigma_{\text{GPA} =4 \text{ AND age}<21}$ (Student)
  - (OP5): $\sigma_{\text{GPA} =4 \text{ OR age}<21}$ (Student)

Algorithms for SELECT Operations

1. Point query on non-key, unsorted file
   - $\sigma_{GPA=3.2}(Student)$

Student(sID, sName, GPA, age)
College(cName, state, enrollment)
Apply(sID, cName, major, decision)

- Linear Search
  - Retrieve every record in the file, and
  - Test whether its attribute values satisfy the selection condition.

Algorithms for SELECT Operations

2. Point query on non-key, sorted file
   - $\sigma_{age=21}$(Student)

Student(<u>sID</u>, sName, GPA, age)
College(<u>cName</u>, state, enrollment)
Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

- Binary search
  - Find the first record that satisfies the condition, then scan until the condition is no longer satisfied.

Algorithms for SELECT Operations

3. Range query on non-key, sorted file
   - $\sigma_{age>21}$(Student)

Student(<u>sID</u>, sName, GPA, age)
College(<u>cName</u>, state, enrollment)
Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

- Binary search
  - Find the first record that satisfies the condition, then scan until the condition is no longer satisfied.

Algorithms for SELECT Operations

4. Point query on index attribute
   - $\sigma_{cName='Stanford'}(College)$

Student(sID, sName, GPA, age)
College(cName, state, enrollment)
Apply(sID, cName, major, decision)

- Hash-based functions
  - Use the associated hash function to find the corresponding records.

5. Range query on index attribute
   - $\sigma_{sID> 100}(Student)$

- Tree structured index
  - Use B+ trees to find the corresponding records.

Putting it together

- For a single condition: $\sigma_{\text{R.attr op value}}(R)$
  1. Use the index on the condition attribute if available, else
  2. Use Binary search if the files is sorted on the condition attribute, else
  3. Use the "brute force" linear search approach

Algorithms for SELECT Operations

6. Conjunctions

- $\sigma_{\text{GPA} = 3.2 \text{ AND age} < 21}(\text{Student})$

Student(<u>sID</u>, sName, GPA, age)
College(<u>cName</u>, state, enrollment)
Apply(<u>sID</u>, <u>cName</u>, <u>major</u>, decision)

- Conjunctive selection using an individual index
  - use that index to retrieve the records
  - for each retrieved record, check if it satisfies the remaining conditions in the conjunction
- Conjunctive selection using a composite index
  - Use the composite index to retrieve the records

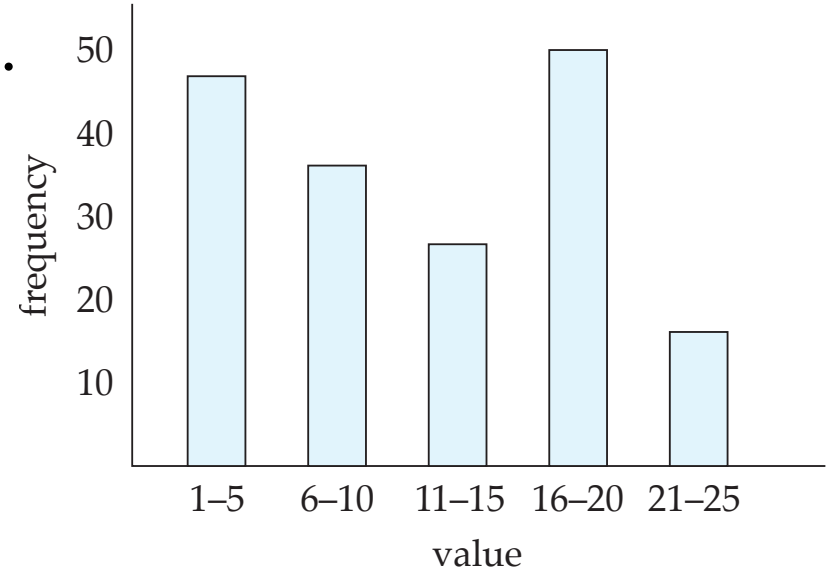
Conjunctive Selection Conditions

- Whenever more than one of the attributes involved in the conditions have an access path, query optimization should choose the index that retrieves the fewest records

  - Example: $\sigma_{\text{GPA =4 AND age<21}}$(Student)
    - Total students: 10,000
    - Those with GPA=4 : 100
    - Those that are younger than 21: 6000
  - The optimizer should:
    1. Retrieve all students with GPA=4 first, and then
    2. Check which of the 100 students is younger than 21.

Selectivity

- **Selectivity (S)**: The ratio of the number of records (tuples) that satisfy the condition to the total number of records (tuples) in the file (relation)
  - S = 0: no records satisfy the condition
  - S = 1: all records satisfy the condition
- Estimates of selectivities are often kept in the DBMS **Catalog** in form of histograms.



Algorithms for SELECT Operations

7. Disjunction
   - $\sigma_{\text{GPA =3.2 OR age<21}}(\text{Student})$

Student(sID, sName, GPA, age)
College(cName, state, enrollment)
Apply(sID, cName, major, decision)

- Little optimization can be done ☹
  - If none of the conditions have an access path, we have to use linear search.
  - If all conditions have access paths, retrieve each separately and then apply the union operator.

Algorithms for JOIN Operations

$$R \bowtie_{A=B} S$$

1. **Nested-loop join:** For each record t in R (outer loop), retrieve every record s from S (inner loop) and test whether the two records satisfy the join condition t[A] = s[B].
2. **Single-loop join:** If a hash key exists for B of S retrieve each record t in R, one at a time (single loop), and then use the hash function to retrieve directly all matching records s from S that satisfy s[B] =t[A].
3. **Sort-merge join:** If the records of R and S are physically sorted (ordered) by value of the join attributes A and B then use merging algorithm. Secondary indexes can also be used for sorting.

Basic Relational Algebra Operations

Advance Relational Algebra Operations

Implementation of SELECT and Join Operations

From Queries to Optimization

From Relational Algebra to Query Tree

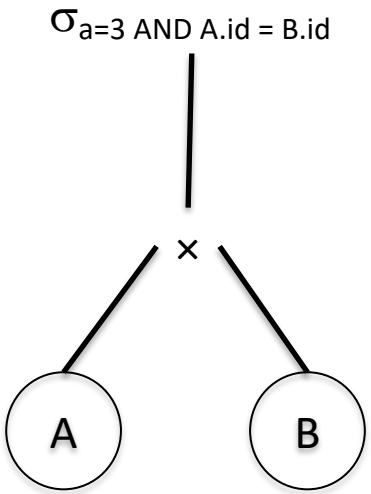
- A **query tree** is a tree data structure that corresponds to a relational algebra expression:
  - Input relations of the query as **leaf nodes**
  - Relational algebra operations as **internal nodes**

An **execution** of the query tree consists of:

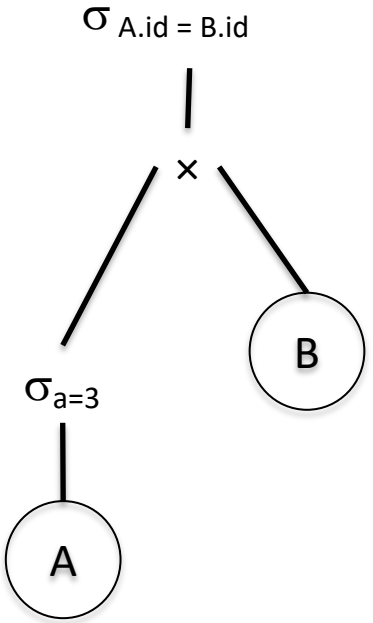
1. Executing an internal node operation whenever its operands are available.
2. Replacing that internal node by the relation that results from executing the operation
3. Executing the root node to produce the final results

Query Tree Examples

$\sigma_{a=3 \text{ AND } A.id = B.id}(A \times B)$



$\sigma_{A.id = B.id}((\sigma_{a=3} A) \times B)$



1. Two query trees are **equivalent** if the relational algebra expressions they express are equivalent.

2. Each query tree represents a partial order over the operations that need to be executed.

Initial Query Tree

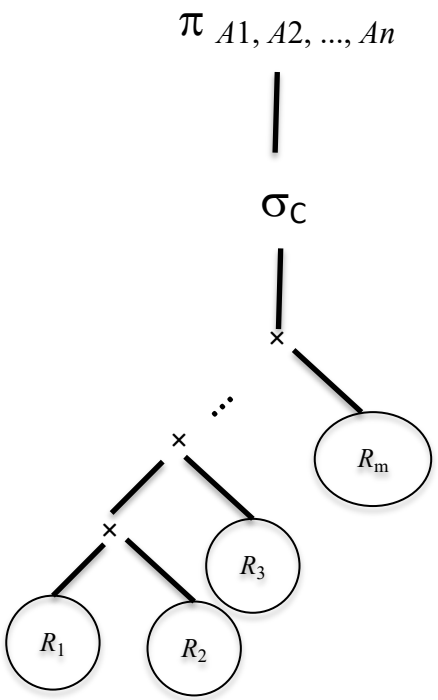
SELECT A1, A2, ..., An
FROM R1, R2, ..., Rm
WHERE Condition

1. Apply Cartesian Product of relations in FROM
2. Selection and Join conditions of WHERE is applied
3. Project on attributes in SELECT

- Equivalent to the following query in relational algebra.

$$\pi_{A1, A2, ..., An} (\sigma_C (R_1 \text{ X } R_2, \ldots \text{ X } R_m))$$

- Equivalent to the query tree on the right-hand side



Using Heuristics in Query Optimization

1. Receive input in form of a high-level query such as SQL
2. The parser of a high-level query generates an initial query tree based on the input.
3. Apply heuristics rules to transform the initial query tree into a final query tree that is efficient to execute.
4. A query execution plan is generated to execute groups of operations based on the access paths available on the files involved in the query.

- The main heuristic is to first apply the operations that reduce the size of intermediate results.
  - E.g., Apply SELECT and PROJECT operations before applying the JOIN or other binary operations.

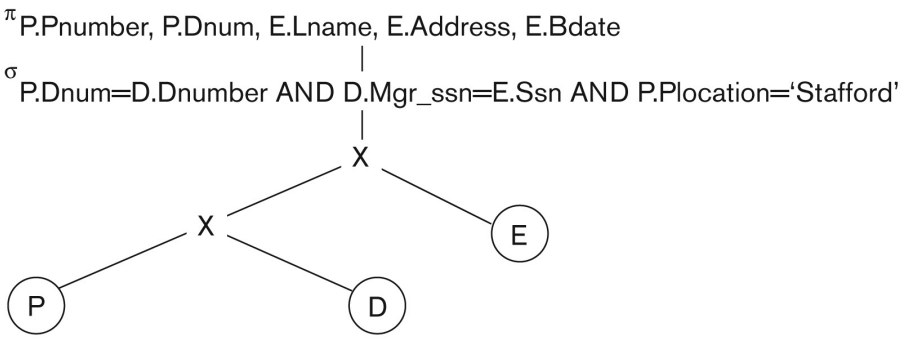
Optimization Example (1)

EMPLOYEE(SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT(DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT(PNUMBER, PNAME, PLOCATION, DNUM)

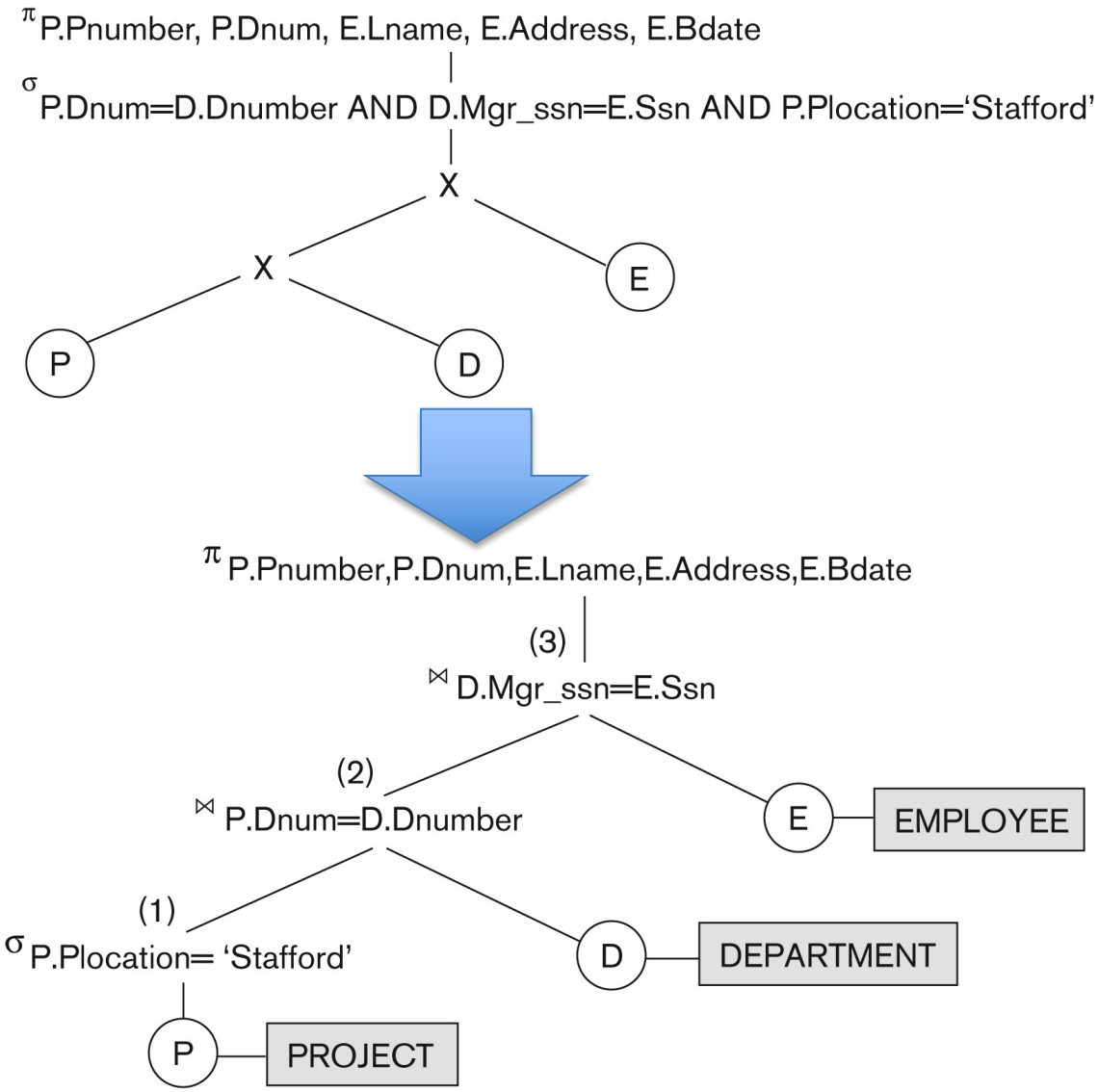
- **EXAMPLE:** For every project located in 'Stafford', retrieve the project number, the controlling department number and the department manager's last name, address and birthdate.

```
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';
```

$\pi_{\text{PNUMBER, DNUM, LNAME, ADDRESS, BDATE}}(\sigma_{\text{PLOCATION='STAFFORD' AND DNUM=DNUMBER AND MGRSSN=SSN}}(\text{EMPLOYEE} \times \text{DEPARTMENT} \times \text{PROJECT}))$



Optimization Example (1)



General Transformation Rules

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

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

2. **Commutativity of σ**: (Execute the one with the fewest records first)

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

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

$$\pi_{list1}(\pi_{list2}(\ldots(\pi_{listN}(R))\ \ldots\ )) = \pi_{list1}(R)$$

4. **Commuting σ with π:** If c involves only attributes in A1, A2, …, An, then the two operations can be commuted.

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

General Transformation Rules

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

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

6. **Commuting σ with ⋈ (or x):** If all the attributes in the selection condition c involve R—the two operations can be commuted as follows:

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

- 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 then:

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

General Transformation Rules

7. **Commuting π with ⋈ (or x):** Suppose $L = \{A_1, ..., A_n, B_1, ..., B_m\}$, where $A_1, ..., A_n$ are attributes of R and $B_1, ..., B_m$ are attributes of S.

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

8. **Associativity of ⋈, x, ∪, and ∩:** if θ stands for any one of these four operations

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

9. **And more…**

Outline of algebraic optimization

1. Break up selections (with conjunctive conditions) into a cascade of selection operators.
2. Push selection operators as far down in the tree as possible.
3. Convert Cartesian products into joins
4. Rearrange leaf nodes so that to:
   - Execute first the most restrictive select operators
5. Move projections as far down as possible

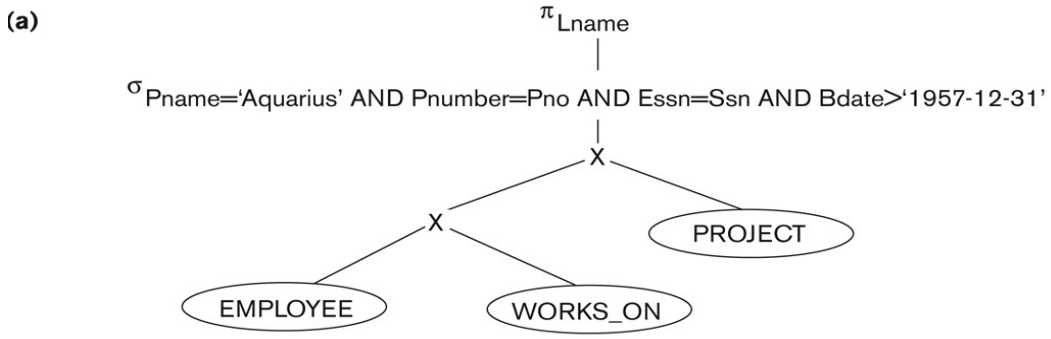
Optimization Example (2)

EMPLOYEE (SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT (PNUMBER, PNAME, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)

- **EXAMPLE:** Find the last names of employees born after 1957 who work on a project named 'Aquarius'.

```
SELECT LNAME
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE PNAME='AQUARIUS' AND PNUMBER=PNO AND ESSN=SSN AND BDATE > '1957-12-31';
```

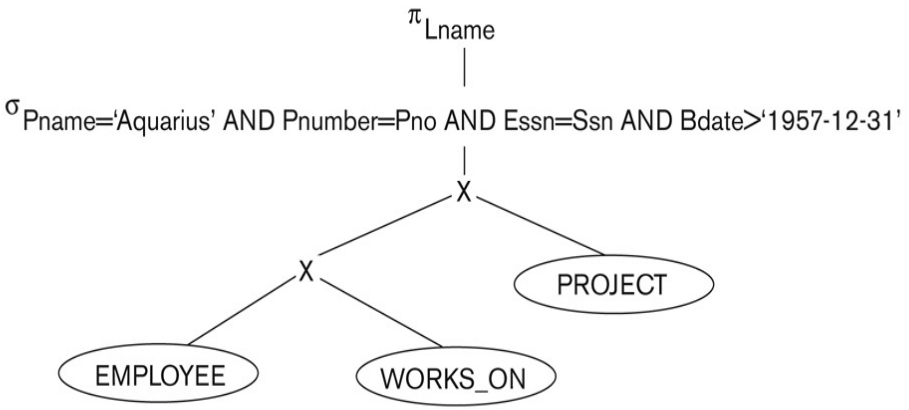
π_{LNAME} ($\sigma_{PNAME='Aquarius' \text{ AND } PNUMBER=PNO \text{ AND } ESSN=SSN \text{ AND } BDATE>'1957\cdot12\cdot31'}$ (EMPLOYEE × WORKS_ON × PROJECT))



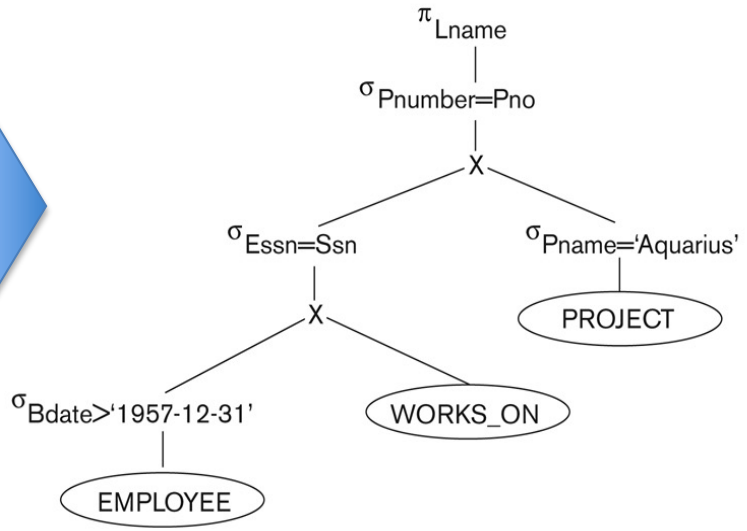
Optimization Example (2)

EMPLOYEE (SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT (PNUMBER, PNAME, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)

(1) Initial query tree



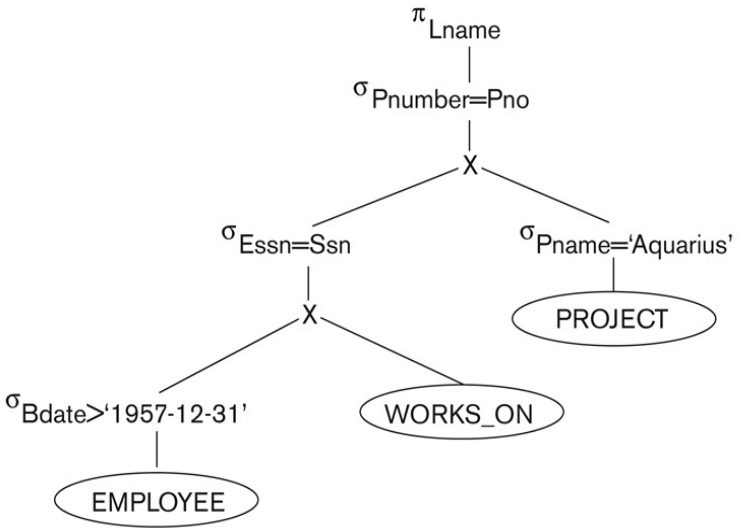
(2) Breaking and Moving SELECT operations down the query tree



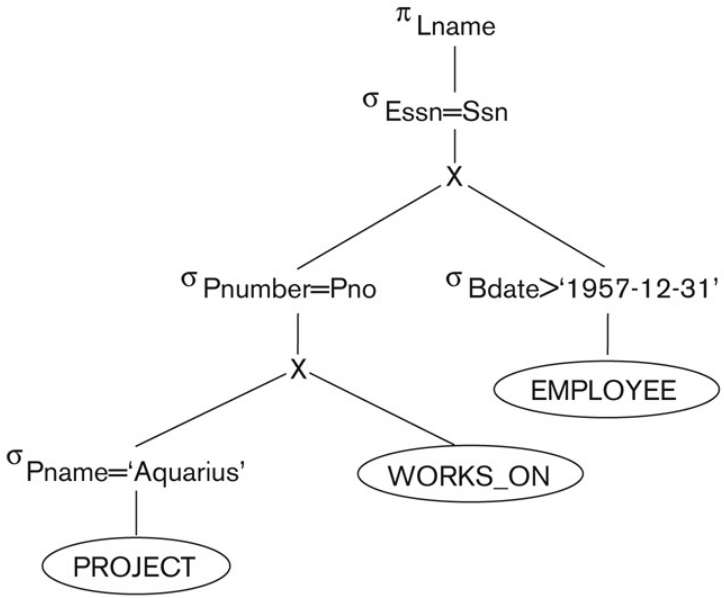
Optimization Example (2)

EMPLOYEE (SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT (PNUMBER, PNAME, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)

(2) Breaking and Moving SELECT operations down the query tree



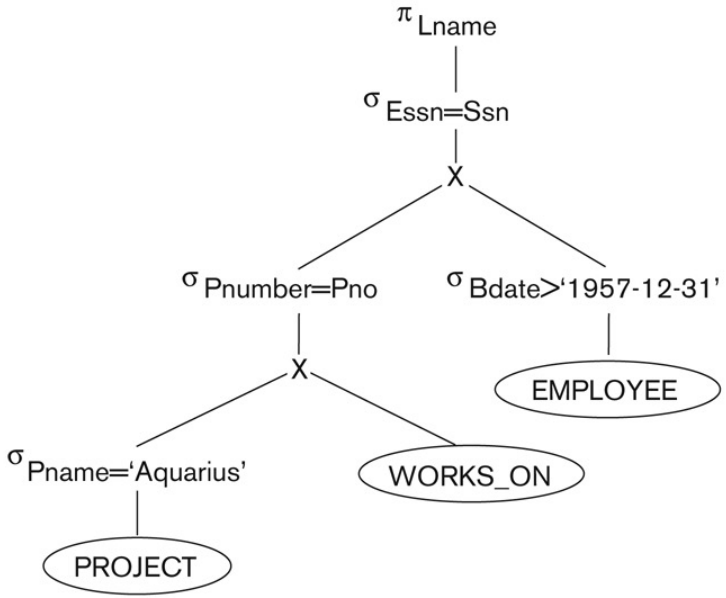
(3) Applying the more restrictive SELECT operation first.



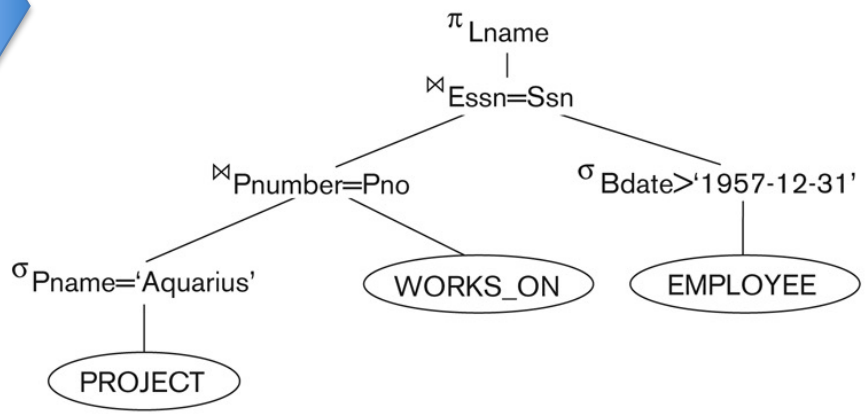
Optimization Example (2)

EMPLOYEE (SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT (PNUMBER, PNAME, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)

(3) Applying the more restrictive SELECT operation first.



(4) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.

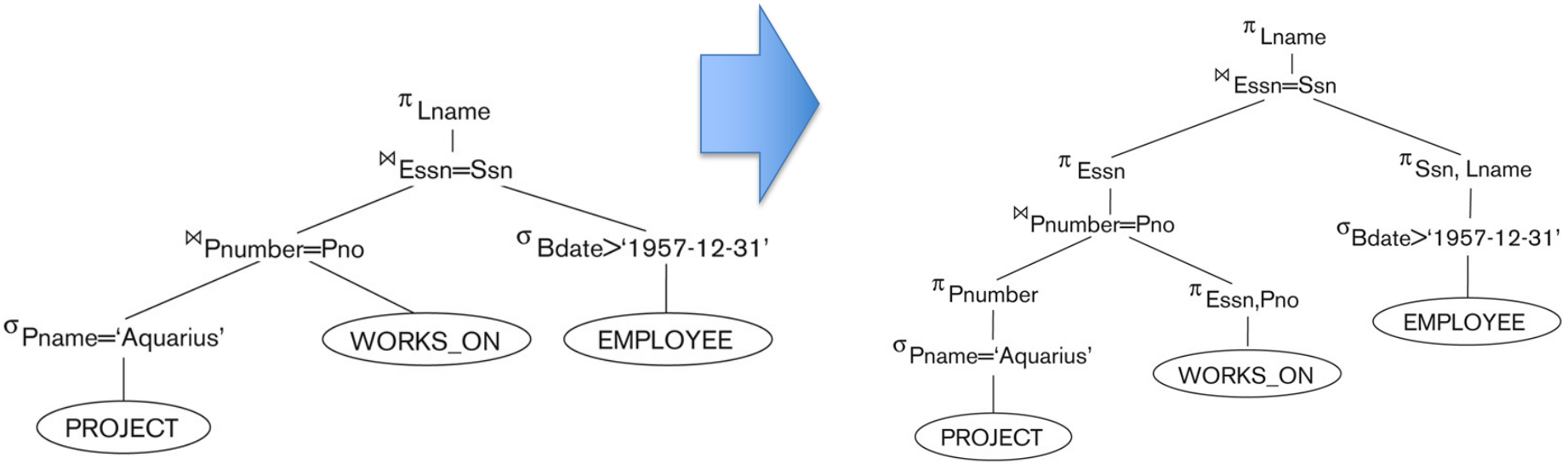


Optimization Example (2)

EMPLOYEE (SSN, FNAME, MINIT, LNAME, BDATE, ADDRESS, DNO)
DEPARTMENT (DNUMBER, DNAME, MGRSSN, MGRSTARTDATE)
PROJECT (PNUMBER, PNAME, PLOCATION, DNUM)
WORKS_ON (ESSN, PNO, HOURS)

(4) Replacing CARTESIAN PRODUCT and SELECT with JOIN operations.

(5) Moving PROJECT operations down the query tree.



One Step Further: Query Execution Plans

A complete query execution plan needs:

1. The **query tree**
2. The **access methods** to be used for each relation,
3. The **algorithms** to be used for relational operators stored in tree, e.g., nested/single loop join