BELGIUM CAMPUS iTversity

It's the way we're wired

# DATABASE DEVELOPMENT

ARE YOU READY?

DBD371/381

# DISTRIBUTED DATABASES

## Query Decomposition and Data Localization

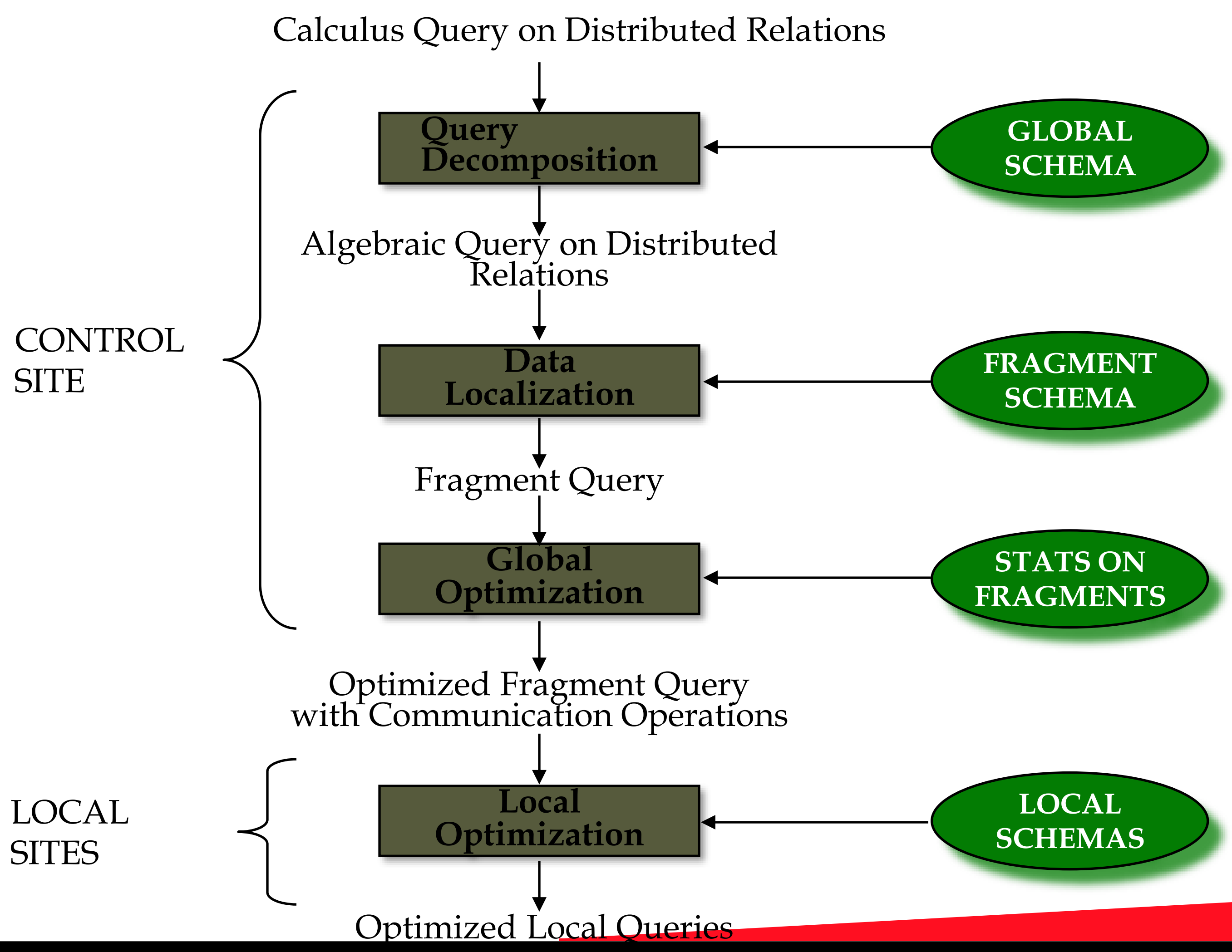
# LEARNING OBJECTIVES

BELGIUM CAMPUS iTversity
It's the way we're wired

- Techniques for query decomposition and data localization
- Query Decomposition
- Generating operator tree
- Equivalence Rules
- Introduction to Relational Algebra

3

# DISTRIBUTED QUERY PROCESSING METHODOLOGY



# TECHNIQUES FOR QUERY DECOMPOSITION AND DATA LOCALIZATION

- Query decomposition maps a distributed calculus query into an *algebraic query* on global relations.
- Data localization takes as input the decomposed query on global relations and applies data distribution information to the query in order to localize its data.
- Remember that to increase the locality of reference and/or parallel execution, relations are fragmented and then stored in disjoint subsets, called fragments, each being placed at a different site.
- Data localization determines which fragments are involved in the query and thereby transforms the distributed query into a *fragment query*.

# TECHNIQUES FOR QUERY DECOMPOSITION AND DATA LOCALIZATION.

- In a multidatabase systems a query is expressed in the global query language against the global schema
- Since the global query may need data from various local DBMSs, it is necessary to decompose the global query into subqueries such that data needed by each subquery are available from one local DBMS
- When a global query is issued to the system it is sent to the query decomposer
- The global query decomposition process is highly dependent on the schema integration information

# EXAMPLE

select e.name, e.age, d.deptname
from employee e, department d
where e.salary >3000
and e.deptno =  d.deptno
and d.address =  'ODTU/Ankara'

```
origin      db1:emp e1
origin      db1:mgr_emp m1
origin      db2:employee e2
origin      db3:employee e3
origin      db2:division d2
```

**Oracle**

emp (emp_id, age, ename, salary, dno)
dept (dno, dname)
mgr_emp (mgr_id, emp_id)

**Sybase**

employee (ssno, name, salary, dname, status)
division (dno, dname, location)
student (sno, name, sex, department, gpa, age)

**Adabas D**

employee (eno, ename, salary, deptno)
department (dno, deptname, address)

# EXAMPLE

- In the second step  projection list is processed

q1 →select e.ename, e.name, e.ename

from e1 ,e2 ,e3

projection list of q1 is used rather than the global attribute

q2→ select e1.salary, e2.salary, e3.salary

from e1, e2, e3;

# EXAMPLE

Final three subqueries generated

```
•   select e1.emp_id, e1.ename, e1.age,
           e1.salary, d1.dname
    from   emp e1, dept d1
    where  e1.salary > 3000 and
           e1.dno = d1.dno;

•   select e2.ssno, e2.name, e2.salary,
           d2.dname, d2.location
    from   employee e2, division d2
    where  e2.salary > 3000 and
           d2.location = 'ODTU/Ankara' and
           e2.dname = d2.dname;

•   select e3.eno, e3.ename, e3.salary,
           d3.deptname, d3.address
    from   employee e3, department d3
    where  e3.salary > 3000 and
           e3.deptno = d3.dno and
           d3.address = 'ODTU/Ankara';
```

# QUERY DECOMPOSITION

- Query decomposition is the first phase of query processing that transforms a relational calculus query into a relational algebra query.
- Both input and output queries refer to global relations, without knowledge of the distribution of data.
- Query decomposition is the same for centralized and distributed systems.
- Assumption: query is syntactically correct
- When this phase is completed successfully the output query is semantically correct and good in the sense that redundant work is avoided.
- successive steps of query decomposition:
  - ❑(1) normalization,
  - ❑(2) analysis,
  - ❑(3) elimination of redundancy,
  - ❑ (4) rewriting

# Normalization

- The input query may be complex, depending on the facilities provided by the language.
- It is the goal of normalization to transform the query to a normalized form to facilitate further processing.
- Most important transformation is that of the query qualification (the WHERE clause), which may be complex, quantifier-free predicate, preceded by all necessary quantifiers ($\forall$ or $\exists$).

# Normalization

- There are two possible normal forms for the predicate, one giving precedence to the **AND (^)** and the other to the OR (V).
- The conjunctive normal form is a conjunction (^ predicate) of disjunctions (V. predicates) as follows:

$$(p_{11} \vee p_{12} \vee \cdots \vee p_{1n}) \wedge \cdots \wedge (p_{m1} \vee p_{m2} \vee \cdots \vee p_{mn})$$

- A qualification in disjunctive normal form, on the other hand, is as follows:

$$(p_{11} \wedge p_{12} \wedge \cdots \wedge p_{1n}) \vee \cdots \vee (p_{m1} \wedge p_{m2} \wedge \cdots \wedge p_{mn})$$

# Normalization

- The transformation of the quantifier-free predicate using the well-known equivalence rules for logical operations:

1. $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$
2. $p_1 \vee p_2 \Leftrightarrow p_2 \vee p_1$
3. $p_1 \wedge (p_2 \wedge p_3) \Leftrightarrow (p_1 \wedge p_2) \wedge p_3$
4. $p_1 \vee (p_2 \vee p_3) \Leftrightarrow (p_1 \vee p_2) \vee p_3$
5. $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$
6. $p_1 \vee (p_2 \wedge p_3) \Leftrightarrow (p_1 \vee p_2) \wedge (p_1 \vee p_3)$
7. $\neg(p_1 \wedge p_2) \Leftrightarrow \neg p_1 \vee \neg p_2$
8. $\neg(p_1 \vee p_2) \Leftrightarrow \neg p_1 \wedge \neg p_2$
9. $\neg(\neg p) \Leftrightarrow p$

# DISJUNCTIVE NORMAL FORM

- The query can be processed as independent conjunctive subqueries linked by unions (disjunctions) although this form may lead to replicated join and select predicates
- The reason is that predicates are very often linked with the other predicates by AND
- The use of rule 5 mentioned above, with p1 as a join or select predicate, would result in replicating p1

# EXAMPLE

- “Find the names of employees who have been working on project P1 for 12 or 24 months”
- Query expressed in SQL

```
SELECT ENAME
FROM EMP, ASG
WHERE EMP.ENO = ASG.ENO
AND ASG.PNO = "P1"
AND DUR = 12 OR DUR = 24
```

Similar to

```
SELECT ENAME FROM EMP
INNER JOIN, ASG
ON EMP.ENO = ASG.ENO
WHERE ASG.PNO = "P1"
AND DUR = 12 OR DUR = 24
```

1. Qualification in conjunctive normal form

   **EMP.ENO = ASG.ENO ^ ASG.PNO = “P1” ^ (DUR = 12 V DUR = 24)**

2. Qualification in disjunctive normal form

**(EMP.ENO = ASG.ENO ^ ASG.PNO = “P1” ^ DUR = 12) V (EMP.ENO = ASG.ENO ^ ASG.PNO = “P1” ^ DUR = 24)**

**Idea is to eliminate common subexpressions →treating the two conjunctions independently may lead to redundant work**

# CONJUNCTIVE NORMAL FORM

- The conjunctive normal form is more practical since query qualifications typically include more AND than OR predicates.

**EMP.ENO = ASG.ENO ^ ASG.PNO = “P1” ^ (DUR = 12  V  DUR = 24)**

# ANALYSIS

- Query analysis enables rejection of normalized queries for which further processing is either impossible or unnecessary
- The main reasons for rejection are that the query is type incorrect or semantically incorrect.
- When one of these cases is detected, the query is simply returned to the user with an explanation.
- A query is type incorrect if :
  - ➢any of its attribute or relation names are not defined in the global schema,
  - ➢operations are being applied to attributes of the wrong type.

# EXAMPLE

- Type incorrect query

```
SELECT E#
FROM EMP
WHERE ENAME > 200
```

**Reason**

1. Attribute E# is not declared in the schema.
2. The operation “>200” is incompatible with the type string of ename.

# EXAMPLE

- A query is semantically incorrect if:
  - its components do not contribute in any way to the generation of the result.
  - ❑ **Find the names and responsibilities of programmers who have been working on the CAD/CAM project for more than 3 years.”**
- **Semantically incorrect Query**

```
SELECT ENAME, RESP
FROM EMP, ASG, PROJ
WHERE EMP.ENO = ASG.ENO
AND PNAME = "CAD/CAM"
AND DUR >= 36
AND TITLE = "Programmer"
```

- **Reason**:
  - ➢ The query is disconnected
- Solution?

❑ Three solutions to the problem are:

(1) Reject the query,

(2) Assume that there is an implicit cartesian product between relations ASG and PROJ,

(3) Infer (using the schema) the missing join predicate ASG.Pno = proj.PNO which transforms the query

# ELIMINATION OF REDUNDANCY

- User queries may be enriched with several predicates
- The enriched query qualification may then contain redundant predicates.
- Redundancy may be eliminated by simplifying the qualification with the well-known **idempotency rules:**

# IDEMPOTENCY RULES

1. $p \wedge p \Leftrightarrow p$
2. $p \vee p \Leftrightarrow p$
3. $p \wedge true \Leftrightarrow p$
4. $p \vee false \Leftrightarrow p$
5. $p \wedge false \Leftrightarrow false$
6. $p \vee true \Leftrightarrow true$
7. $p \wedge \neg p \Leftrightarrow false$
8. $p \vee \neg p \Leftrightarrow true$
9. $p_1 \wedge (p_1 \vee p_2) \Leftrightarrow p_1$
10. $p_1 \vee (p_1 \wedge p_2) \Leftrightarrow p_1$

# EXAMPLE

SELECT TITLE

FROM EMP

WHERE (NOT (TITLE = "Programmer")

AND (TITLE = "Programmer“ OR TITLE = "Elect. Eng.")

AND NOT (TITLE = "Elect. Eng."))

OR ENAME = "J. Doe“

can be simplified using the previous rules to become

SELECT TITLE
FROM EMP
WHERE ENAME = "J. Doe"

# SIMPLIFICATION

- Let p1 be TITLE = “Programmer”,
- p2 be TITLE = “Elect. Eng.”, and p3 be ENAME = “J. Doe”.

(NOT (TITLE = "Programmer")AND (TITLE = "Programmer“ OR TITLE = "Elect. Eng.")
AND NOT (TITLE = "Elect. Eng."))O R ENAME = "J. Doe“ $(\neg p_1 \wedge (p_1 \vee p_2) \wedge \neg p_2) \vee p_3$

The disjunctive normal form for this qualification is obtained by applying rule

which yields $(\neg p_1 \wedge ((p_1 \wedge \neg p_2) \vee (p_2 \wedge \neg p_2))) \vee p_3$ which yields

$(\neg p_1 \wedge p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2 \wedge \neg p_2) \vee p_3$ $p \wedge false \Leftrightarrow false$

By applying rule $p \wedge \neg p \Leftrightarrow false$ we obtain

$(false \wedge \neg p_2) \vee (\neg p_1 \wedge false) \vee p_3$

By applying the same rule, we get

$false \vee false \vee p_3$ $p \vee false \Leftrightarrow p$

which is equivalent to p3 by rule

# Rewriting

- The last step of query decomposition rewrites the query in relational algebra and use transformation rules according to six most useful equivalence rules, which concern the basic relational algebra operators.
- 1. **Commutativity of binary operators.** The Cartesian product of two relations R and S is commutative:

$$R \times S \Leftrightarrow S \times R$$

# OPERATOR TREE

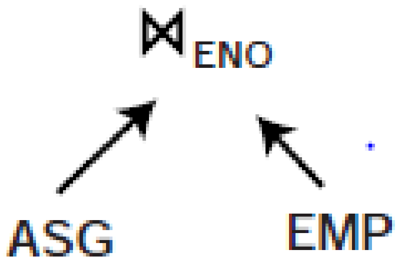
- “Find the names of employees other than J. Doe who worked on the CAD/CAM project for either one or two years”
- SQL expression

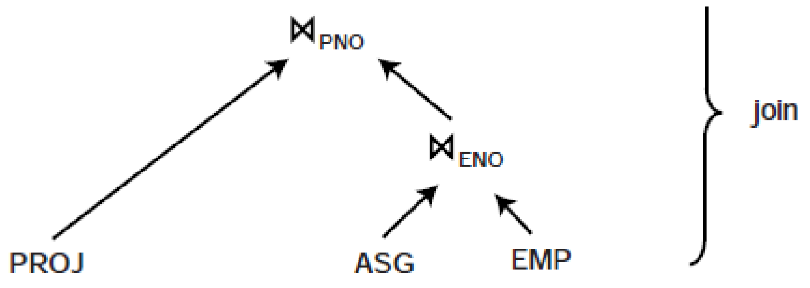
```
SELECT ENAME
FROM PROJ, ASG, EMP
WHERE ASG.ENO = EMP.ENO
AND ASG.PNO = PROJ.PNO
AND ENAME != "J. Doe"
AND PROJ.PNAME = "CAD/CAM"
AND (DUR = 12 OR DUR = 24)
```

# GENERATING OPERATOR TREE

- A different leaf (available in the FROM clause. ) is created for each different tuple variable (corresponding to a relation).

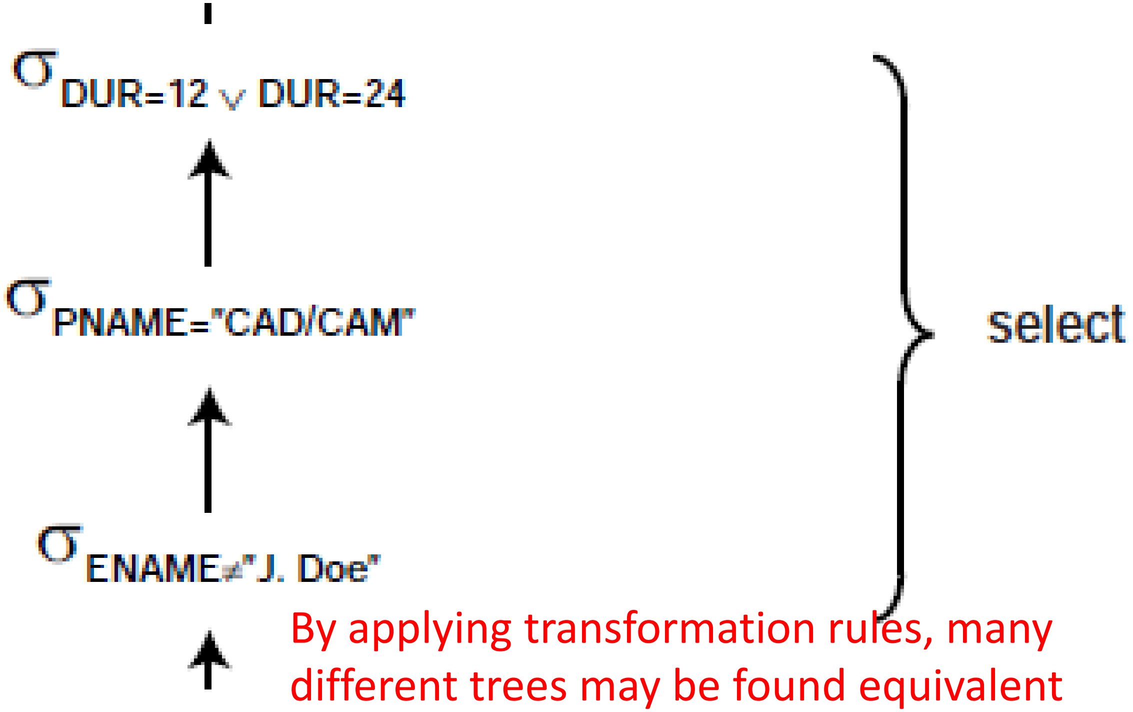

- the root node is created as a projection (found in the SELECT clause) operation involving the result attributes



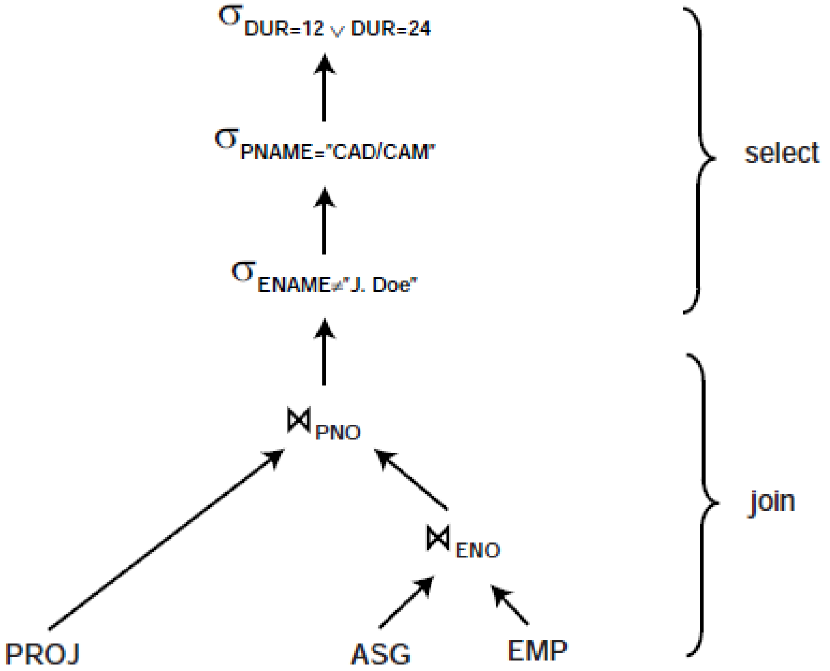



# GENERATING OPERATOR TREE

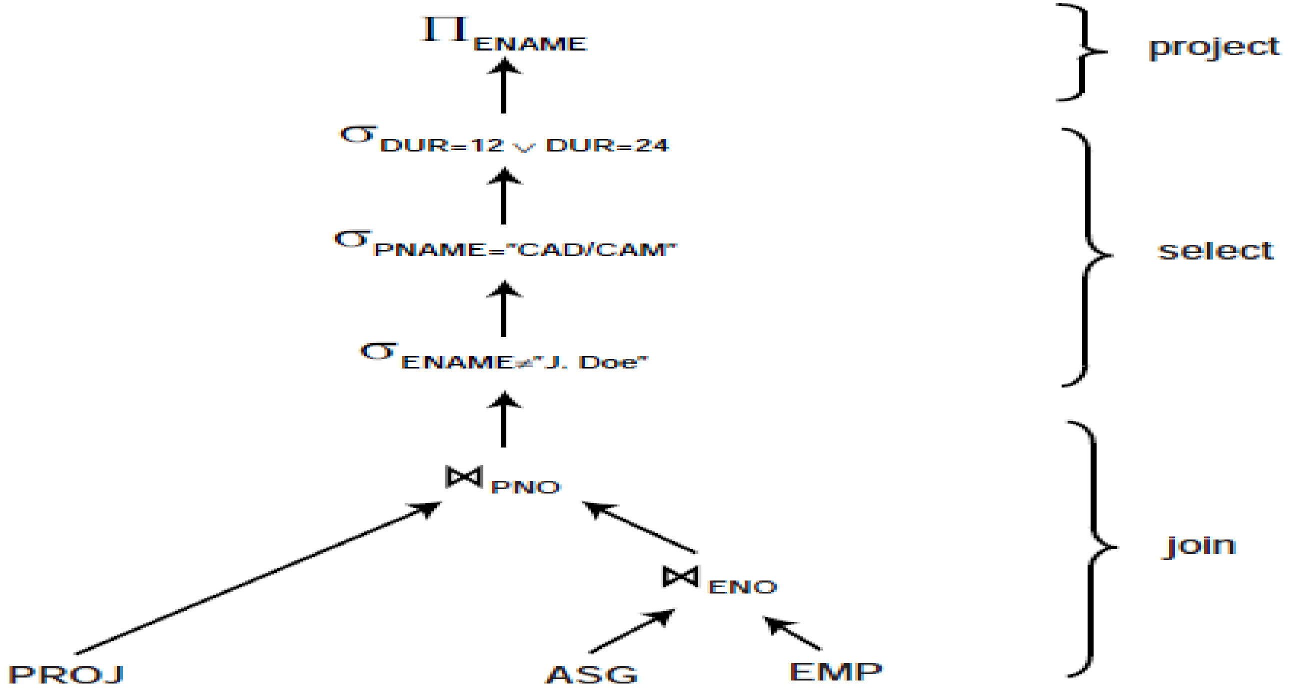
- the qualification (WHERE clause) is translated into the appropriate sequence of relational operations (select, join, union, etc.) going from the leaves to the root.
  - ➢given directly by the order of appearance of the predicates and operators.




By applying transformation rules, many different trees may be found equivalent



# GENERATING OPERATOR TREE



## Equivalence Rules

- By applying transformation rules, many different trees may be found equivalent
- **Equivalence Rules, Which Concern The Basic Relational Algebra Operators.**

1. **Commutativity of binary operators**. The Cartesian product of two relations R and S is commutative: or the join of two relations is commutative

$$R \times S \Leftrightarrow S \times R$$

$$R \bowtie S \Leftrightarrow S \bowtie R$$

## Equivalence Rules

- 2. **Associativity of binary operators.** The Cartesian product and the join are associative operators:

$$(R \times S) \times T \Leftrightarrow R \times (S \times T)$$
$$(R \bowtie S) \bowtie T \Leftrightarrow R \bowtie (S \bowtie T)$$

## Equivalence Rules

- 3. **Idempotence of unary operators.** Several subsequent projections on the same relation may be grouped.
- A single projection on several attributes may be separated into several subsequent projections.
- If R is defined over the attribute set A $A' \subseteq A, A'' \subseteq A$, and $A' \subseteq A''$, then

$$\Pi_{A'}(\Pi_{A''}(R)) \Leftrightarrow \Pi_{A'}(R)$$

## Equivalence Rules

- Several subsequent selec $\sigma_{p_i(A_i)}$ on the same relation, where $p_i$ is a predicate applied to attribute $A_i$, may be grouped as

$$\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) = \sigma_{p_1(A_1)\wedge p_2(A_2)}(R)$$

- 4. **Commuting selection with projection.** Selection and projection on the same relation can be commuted as follows:

$$\Pi_{A_1,\ldots,A_n}(\sigma_{p(A_p)}(R)) \Leftrightarrow \Pi_{A_1,\ldots,A_n}(\sigma_{p(A_p)}(\Pi_{A_1,\ldots,A_n,A_p}(R)))$$

**if Ap is already a member of fA1; : : : ;Ang, the last projection on [A1; : : : ;An] on the right-hand side of the equality is useless.**

## Equivalence Rules

- 5. Commuting selection with binary operators.
- Selection and Cartesian product can be commuted using the rule (remember that attribute $A_i$ belongs to relation R):

$$\sigma_{p(A_i)}(R \times S) \Leftrightarrow (\sigma_{p(A_i)}(R)) \times S$$

$$\sigma_{p(A_i)}(R \bowtie_{p(A_j,B_k)} S) \Leftrightarrow \sigma_{p(A_i)}(R) \bowtie_{p(A_j,B_k)} S$$

**Selection and union can be commuted if R and T are union compatible (have the same schema):**

$$\sigma_{p(A_i)}(R \cup T) \Leftrightarrow \sigma_{p(A_i)}(R) \cup \sigma_{p(A_i)}(T)$$

# Equivalence Rules

- 6. **Commuting projection with binary operators.** Projection and Cartesian product can be commuted.

If $C = A' \cup B'$, where $A' \subseteq A$, $B' \subseteq B$, and A and B are the sets of attributes over which relations R and S, respectively, are defined then

$$\Pi_C(R \times S) \Leftrightarrow \Pi_{A'}(R) \times \Pi_{B'}(S)$$

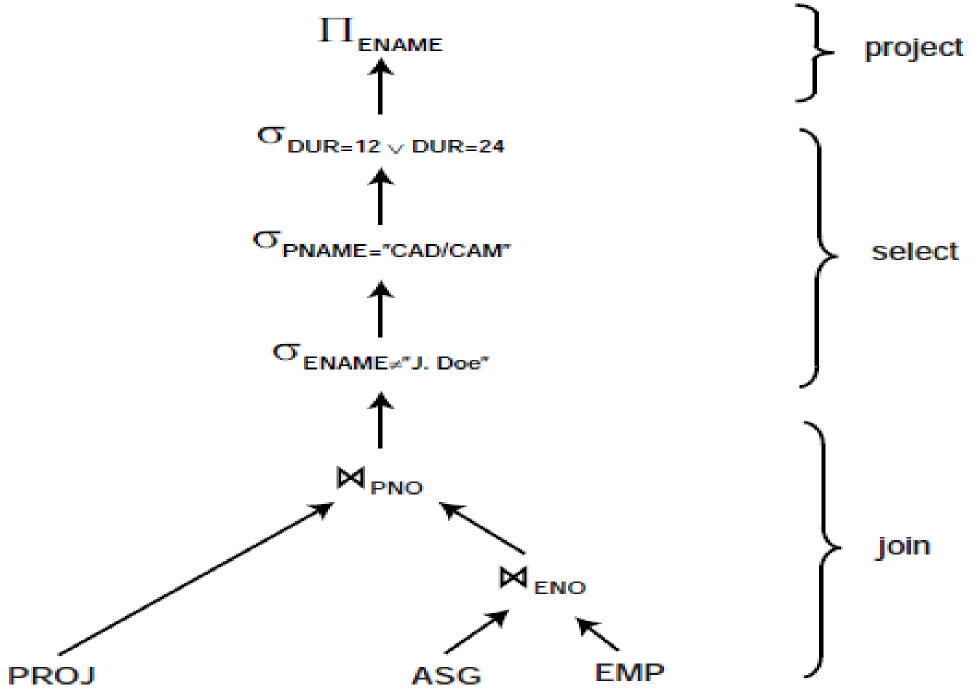
Projection and join can also be commuted.

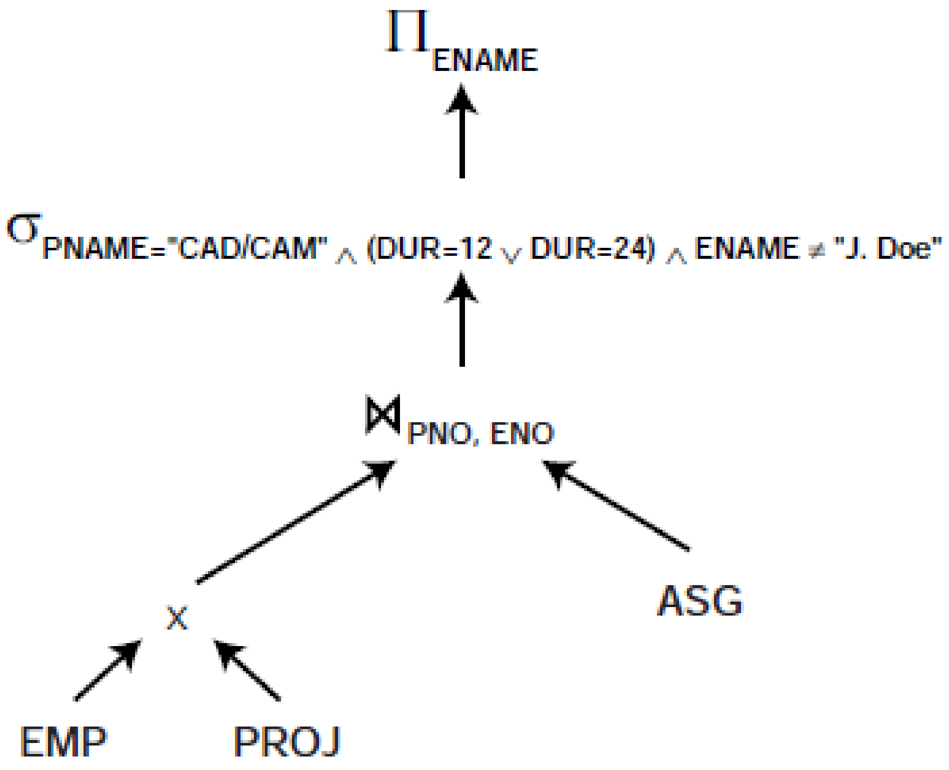
$$\Pi_C(R \bowtie_{p(A_i,B_j)} S) \Leftrightarrow \Pi_{A'}(R) \bowtie_{p(A_i,B_j)} \Pi_{B'}(S)$$

$$\Pi_C(R \cup S) \Leftrightarrow \Pi_C(R) \cup \Pi_C(S)$$

## Equivalence Rules

- The application of these six rules enables the generation of many equivalent trees.



# INTRODUCTION TO RELATIONAL ALGEBRA

# RELATIONAL ALGEBRA

- A formal query language for asking questions
- A query is composed of a collection of operators called **relational operators**
- Unary operators: selection, projection, renaming
- Binary operators: union, intersect, difference, cartesian product,join
- Relations are closed under relational operators
- Operators can be composed to form **relational algebra expressions**

# SELECTION CONDITION

- **Selection condition** is a boolean combination of terms
- A **term** is one of the following forms:

1. attribute **op** constant — **op** $\in \{=, =, <, \leq, >, \geq\}$
2. $\text{attribute}_1$ **op** $\text{attribute}_2$
3. $\text{term}_1 \wedge \text{term}_2$
4. $\text{term}_1 \vee \text{term}_2$
5. $\neg\ \text{term}_1$
6. $(\text{term}_1)$

- Operator precedence: (), op, $\neg$, $\wedge$, $\vee$

# SET OPERATIONS

- **Union**: R ∪ S returns a relation containing all tuples that occur in R or S (or both)
- **Intersection**: R ∩ S returns a relation containing all tuples that occur in both R and S
- **Set-difference**: R − S returns a relation containing all tuples in R but not in S
- Two relations are **union compatible** if
  - they have the same arity, and
  - the corresponding attributes have same domains
- union (∪), intersection (∩), and set-difference (−) operators require input relations to be union compatible

# SET OPERATIONS (CONT.)

- Consider R(A,B,C) and S(X, Y )
- **Cross-product**: R × S returns a relation with attribute list (A,B,C,X, Y ) defined as follows:

$$R \times S = \{(a, b, c, x, y) \mid (a, b, c) \in R, (x, y) \in S\}$$

- Cross-product operation is also known as **cartesian product**

# Join

- Combines cross-product, selection, and projection
- Join operator is more useful than the plain cross-product operator
- Three types of join:
  - Condition join
  - Equijoin
  - Natural join

# Equijoin: $R \bowtie_c S$

- Where

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

- – $c$ is a conjunction of equality conditions of the form $R.A_i = S.A_j$
- – $L$ is a sequence of attributes consisting of $L_1$ followed by $L_2$
- – $L_1$ is a sequence of attributes in schema of $R$
- – $L_2$ is a sequence of attributes in schema of $S$ that are not referenced in c

# Example Database

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

Actors

| actor | ayear |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

Acts

| actor | title |
|---|---|
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

Directors

| director | dyear |
|---|---|
| Coen | 1954 |
| Hanson | 1945 |
| Raimi | 1959 |

- Find movies made after 1997
- Find movies made by Hanson after 1997
- Find all movies and their ratings
- Find all actors and directors
- Find Coen's movies with McDormand
- Find movies with Maguire but not McDormand
- Find actors who have acted in some Coen's movie
- Find (director, actor) pairs where the director is younger than the actor
- Find actors who have acted in all of Coen's movies

# Selection: σ



- $\sigma_c(R)$ selects rows from relation R that satisfy selection condition c
- **Example:** Find movies made after 1997

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$$\sigma_{myear > 1997}(\text{Movies})$$

| title | director | myear | rating |
|---|---|---|---|
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

# Selection Condition (cont.)

- **Example:** Find movies made by Hanson after 1997

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$$\downarrow \quad \sigma_{myear>1997\ \wedge\ director=\text{'}Hanson\text{'}}(\text{Movies})$$

| title | director | myear | rating |
|---|---|---|---|
| Wonder Boys | Hanson | 2000 | 7.6 |

# Projection: π

- $\pi_L(R)$ projects columns given by list **L** from relation R
- **Example**: Find all movies and their ratings

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$\downarrow \quad \pi_{title,\ rating}(\text{Movies})$

| title | rating |
|---|---|
| Fargo | 8.2 |
| Raising Arizona | 7.6 |
| Spiderman | 7.4 |
| Wonder Boys | 7.6 |

# Renaming: $\rho$

- Given relation R(A,B,C), $\rho_{S(X,Y,Z)}(R)$ renames it to S(X, Y,Z)

Actors

| actor | ayear |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

$$\rho_{Stars(name,yob)}(\text{Actors})$$

Stars

| name | yob |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

# FIND ALL ACTORS AND DIRECTORS

Actors

| actor | ayear |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

Directors

| director | dyear |
|---|---|
| Coen | 1954 |
| Hanson | 1945 |
| Raimi | 1959 |

$\downarrow \pi_{actor}(\text{Actors})$

| actor |
|---|
| Cage |
| Hanks |
| Maguire |
| McDormand |

$\downarrow \pi_{director}(\text{Directors})$

| director |
|---|
| Coen |
| Raimi |
| Hanson |

$\pi_{actor}(Actors) \cup \pi_{director}(Directors)$

| actor |
|---|
| Cage |
| Hanks |
| Maguire |
| McDormand |
| Coen |
| Raimi |
| Hanson |

**Union Example:**

Find all actors & directors

$$\pi_{actor}(Actors) \cup \pi_{director}(Directors)$$

# • FIND COEN'S MOVIES WITH MCDORMAND

BELGIUM CAMPUS iTversity
It's the way we're wired

**Acts**

| actor | title |
|---|---|
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

**Movies**

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$e_1$

| title |
|---|
| Fargo |
| Raising Arizona |
| Wonder Boys |

$e_2$

| title |
|---|
| Fargo |
| Raising Arizona |

$e_1 \cap e_2$

| title |
|---|
| Fargo |
| Raising Arizona |

Intersection Example:

Find Coen's movies with McDormand

$$e_1 = \pi_{title}(\sigma_{actor=\text{'McDormand'}}(Acts))$$

$$e_2 = \pi_{title}(\sigma_{director=\text{'Coen'}}(Movies))$$

$$\text{result} = e_1 \cap e_2$$

# FIND MOVIES WITH MAGUIRE BUT NOT MCDORMAND

**Acts**

| actor | title |
|---|---|
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

$\sigma_{actor=\text{'}McDormand\text{'}}(\textbf{Acts})$

$\sigma_{actor=\text{'}Maguire\text{'}}(\textbf{Acts})$

| title |
|---|
| Spiderman |
| Wonder Boys |

| title |
|---|
| Fargo |
| Raising Arizona |
| Wonder Boys |

$\sigma_{actor=\text{'}Maguire\text{'}}(\textbf{Acts})$ - $\sigma_{actor=\text{'}McDormand\text{'}}(\textbf{Acts})$

| title |
|---|
| Spiderman |

Set-difference Example:

Find movies with Maguire but not McDormand

$\sigma_{actor=\text{'}Maguire\text{'}}(\textbf{Acts})$ - $\sigma_{actor=\text{'}McDormand\text{'}}(\textbf{Acts})$

# Cross-product Example

- Find actors who have acted in some Coen's movies

$$e_1 = \rho_{T(title2)}(\pi_{title}(\sigma_{director='Coen'}(Movies)))$$

Movies

| title | director | myear | rating |
| --- | --- | --- | --- |
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$\xrightarrow{e_1}$

T

| title2 |
| --- |
| Fargo |
| Raising Arizona |

# Cross-product Example (cont.)

Acts

| actor | title |
|---|---|
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

$\times$

T

| title2 |
|---|
| Fargo |
| Raising Arizona |

| actor | title | title2 |
|---|---|---|
| Cage | Raising Arizona | Fargo |
| Cage | Raising Arizona | Raising Arizona |
| Maguire | Spiderman | Fargo |
| Maguire | Spiderman | Raising Arizona |
| Maguire | Wonder Boys | Fargo |
| Maguire | Wonder Boys | Raising Arizona |
| McDormand | Fargo | Fargo |
| McDormand | Fargo | Raising Arizona |
| McDormand | Raising Arizona | Fargo |
| McDormand | Raising Arizona | Raising Arizona |
| McDormand | Wonder Boys | Fargo |
| McDormand | Wonder Boys | Raising Arizona |

# Cross-product Example (cont.)

| actor | title | title2 |
|---|---|---|
| Cage | Raising Arizona | Fargo |
| Cage | Raising Arizona | Raising Arizona |
| Maguire | Spiderman | Fargo |
| Maguire | Spiderman | Raising Arizona |
| Maguire | Wonder Boys | Fargo |
| Maguire | Wonder Boys | Raising Arizona |
| McDormand | Fargo | Fargo |
| McDormand | Fargo | Raising Arizona |
| McDormand | Raising Arizona | Fargo |
| McDormand | Raising Arizona | Raising Arizona |
| McDormand | Wonder Boys | Fargo |
| McDormand | Wonder Boys | Raising Arizona |

$$\downarrow e_3 = \sigma_{title=title2}(e_2)$$

| actor | title | title2 |
|---|---|---|
| Cage | Raising Arizona | Raising Arizona |
| McDormand | Fargo | Fargo |
| McDormand | Raising Arizona | Raising Arizona |

$$\xrightarrow{\pi_{actor}(e_3)}$$

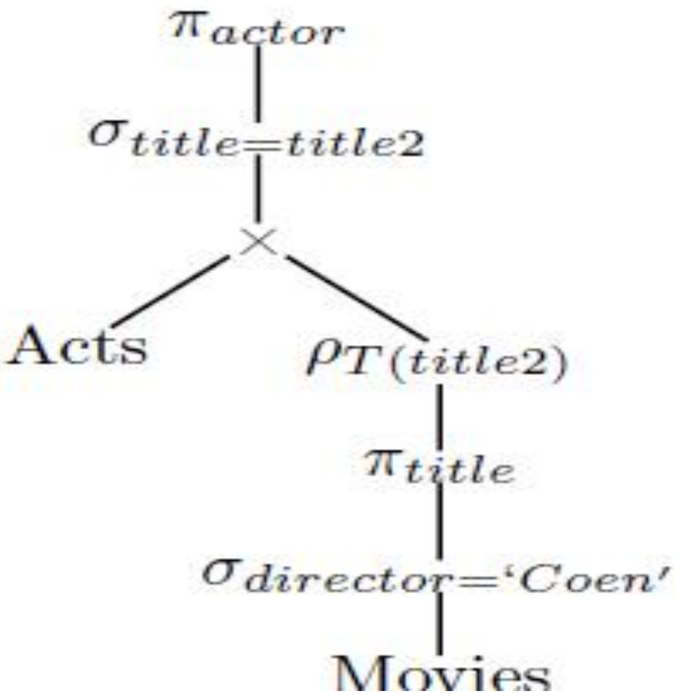
| actor |
|---|
| Cage |
| McDormand |

# Cross-product Example (cont.)

BELGIUM CAMPUS iTversity
It's the way we're wired

- **Query**: Find actors who have acted in some Coen's movie

Answer: $\pi_{actor}(\ \sigma_{title=title2}\ (\ \text{Acts} \times \rho_{T(title2)}(\ \pi_{title}(\ \sigma_{director='Coen'}(\text{Movies})\ )\ )\ )\ )$



# Condition Join: R$\bowtie_c$S



**Condition join** = Cross-product followed by selection

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

- **Example**: Find (director,actor) pairs where the director is younger than the actor

- Answer: $\pi_{director,actor}(\text{ Directors } \bowtie_{dyear>ayear} \text{ Actors})$

Directors

| director | dyear |
|---|---|
| Coen | 1954 |
| Hanson | 1945 |
| Raimi | 1959 |

Actors

| actor | ayear |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

$e_1 = \text{Directors} \bowtie_{dyear>ayear} \text{Actors}$

| director | dyear | actor | ayear |
|---|---|---|---|
| Raimi | 1959 | Hanks | 1956 |
| Raimi | 1959 | McDormand | 1957 |

$\pi_{director,actor}(e_1)$

| director | actor |
|---|---|
| Raimi | Hanks |
| Raimi | McDormand |

# Equijoin (cont.)



- **Example:** Find actors who have acted in some Coen's movie

$$\pi_{actor}(\ \sigma_{director=\text{'}Coen\text{'}}\ (\ \text{Acts} \bowtie_{Acts.title\ =\ Movies.title} \text{Movies}\ )\ )$$

$$e_1 = \text{Acts} \bowtie_{Acts.title\ =\ Movies.title} \text{Movies}$$

| actor | title | director | myear | rating |
|---|---|---|---|---|
| Cage | Raising Arizona | Coen | 1987 | 7.6 |
| Maguire | Spiderman | Raimi | 2002 | 7.4 |
| Maguire | Wonder Boys | Hanson | 2000 | 7.6 |
| McDormand | Fargo | Coen | 1996 | 8.2 |
| McDormand | Raising Arizona | Coen | 1987 | 7.6 |
| McDormand | Wonder Boys | Hanson | 2000 | 7.6 |

$$\pi_{actor}(\sigma_{director=\text{'}Coen\text{'}}((e_1))$$

| actor |
|---|
| Cage |
| McDormand |

# Natural Join: $R \bowtie S$

- **Natural join** = Equijoin of the form

$$R \bowtie S = R \bowtie_c S$$

- where c is specified for all attributes having the same name in R and S
- **Example**: Find actors who have acted in some Coen's movie

$$\pi_{actor}(\ \sigma_{director='Coen'}\ (\ \text{Acts} \bowtie \text{Movies}\ )\ )$$

- **Example**: Find the name and the year of birth of all actors who
- were in some Coen's movie

$$\pi_{actor,ayear}(\ \sigma_{director='Coen'}\ (\text{Movies}) \bowtie \text{Acts} \bowtie \text{Actors}\ )$$

# Example: Condition, Equi-, Natural Joins



R

| A | B | X |
|---|---|---|
| 0 | 6 | $x_1$ |
| 1 | 9 | $x_2$ |
| 2 | 7 | $x_3$ |

S

| A | B | Y |
|---|---|---|
| 0 | 8 | $y_1$ |
| 1 | 5 | $y_2$ |
| 2 | 7 | $y_3$ |

$R \bowtie_{A=A' \wedge B<B'} \rho_{S'(A',B',Y)}(S)$

| A | B | X | A' | B' | Y |
|---|---|---|---|---|---|
| 0 | 6 | $x_1$ | 0 | 8 | $y_1$ |

$R \bowtie_{A=A'} \rho_{S'(A',B',Y)}(S)$

| A | B | X | B' | Y |
|---|---|---|---|---|
| 0 | 6 | $x_1$ | 8 | $y_1$ |
| 1 | 9 | $x_2$ | 5 | $y_2$ |
| 2 | 7 | $x_3$ | 7 | $y_3$ |

$R \bowtie S$

| A | B | X | Y |
|---|---|---|---|
| 2 | 7 | $x_3$ | $y_3$ |

# QUIZ

- **Query**: Find actors who have acted in all Coen’s movies

$$CMovies = \pi_{title}(\sigma_{director='Coen'}(Movies))$$

Movies

| title | director | myear | rating |
| --- | --- | --- | --- |
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$\longrightarrow$

CMovies

| title |
| --- |
| Fargo |
| Raising Arizona |

Acts

| actor | title |
| --- | --- |
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

# Rewriting

- 2. **Associativity of binary operators**. The Cartesian product and the join are associative operators:

$$(R \times S) \times T \Leftrightarrow R \times (S \times T)$$

- 3. **Idempotence of unary ope**nt projections on the same relation may be grouped.

- 4. **Commuting selection with projection**. Selection and projection on the same relation can be commuted as follows:

$$\sigma_{p_1(A_1)}(\sigma_{p_2(A_2)}(R)) = \sigma_{p_1(A_1) \wedge p_2(A_2)}(R)$$

- if Ap is already a member of fA1; : : : ;An, the last projection on
- [A1; : : : ;An] on the right-hand side of the equality is useless.

$$\Pi_{A_1,\ldots,A_n}(\sigma_{p(A_p)}(R)) \Leftrightarrow \Pi_{A_1,\ldots,A_n}(\sigma_{p(A_p)}(\Pi_{A_1,\ldots,A_n,A_p}(R)))$$

# Rewriting

- 5. **Commuting selection with binary operators.** Selection and Cartesian product can be commuted using the following rule

$$\sigma_{p(A_i)}(R \times S) \Leftrightarrow (\sigma_{p(A_i)}(R)) \times S$$

6. **Commuting projection with binary operators.** Projection and Cartesian product can be commuted. If $C = A' \cup B'$, where $A' \subseteq A$, $B' \subseteq B$, and $A$ and $B$ are the sets of attributes over which relations $R$ and $S$, respectively, are defined, we have

$$\Pi_C(R \times S) \Leftrightarrow \Pi_{A'}(R) \times \Pi_{B'}(S)$$

# Localization of Distributed Data

- These global techniques discussed  apply to both centralized and distributed DBMSs and do not take into account the distribution of data. This is the role of the localization layer.
- The localization layer translates an algebraic query on global relations into an algebraic query expressed on physical fragments.
- Localization uses information stored in the fragment schema which is defined through fragmentation rules, which can then be expressed as relational queries.
- The global relation is reconstructed by applying the reconstruction (or reverse fragmentation) rules and deriving a relational algebra program whose operands are the fragments (localization program)
- In short We do not consider the fact that data fragments may be replicated,
- To localize a distributed query is to generate a query where each global relation is substituted by its localization program
- the query obtained this way  is called the localized query.

# QUESTIONS

This Photo by Unknown Author is licensed under CC BY

# REVISION QUESTIONS

- Explain the difference between a type incorrect or semantically incorrect query?
- What solutions do you propose to semantically incorrect queries?
- Simplify the following query, expressed in SQL, using idempotency rules:

```
SELECT ENO
FROM ASG
WHERE RESP = "Analyst"
AND NOT(PNO="P2" OR DUR=12)
AND PNO != "P2"
AND DUR=12
```

- Give the query graph of the above query

# REVISION QUESTIONS

- Consider the following query on our Engineering database:

```
SELECT ENAME,SAL
FROM EMP,PROJ,ASG,PAY
WHERE EMP.ENO = ASG.ENO
AND EMP.TITLE = PAY.TITLE
AND (BUDGET>200000 OR DUR>24)
AND ASG.PNO = PROJ.PNO
AND (DUR>24 OR PNAME = "CAD/CAM")
```

- Compose the selection predicate corresponding to the WHERE clause and transform it, using the idempotency rules, into the simplest equivalent form.