

QUERY OPTIMIZATION

AUTOMATIC QUERY PERFORMANCE IMPROVEMENT

IMPROVEMENT IS NOT GUARANTEED TO BE OPTIMAL

**ONLY POSSIBLE BECAUSE OF THE HIGH SEMANTIC
LEVEL OF SQL QUERIES**

STAGES OF THE OPTIMIZATION PROCESS

- **CHOOSE AN INTERNAL REPRESENTATION**
- **APPLY TRANSFORMATIONS TO IMPROVE
EFFICIENCY**
- **CHOOSE CANDIDATE LOW-LEVEL DATA ACCESS
PROCEDURES**
- **GENERATE QUERY PLANS AND ASSIGN COSTS**
- **CHOOSE THE CHEAPEST**

QUERY TREES

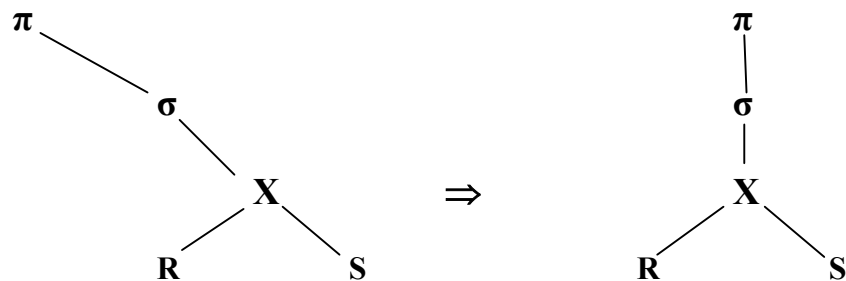
TREE REPRESENTATION OF A RELATIONAL ALGEBRA EXPRESSION WHERE:

- **INORDER TRAVERSAL OF THE TREE PRODUCES THE ORIGINAL EXPRESSION**
- **BINARY OPERATORS ARE \Join, \times, \cup**
- **UNARY OPERATORS ARE $\sigma(S), \pi(P)$**

(OPERANDS OF UNARY OPERATORS APPEAR AS RIGHT CHILDREN)

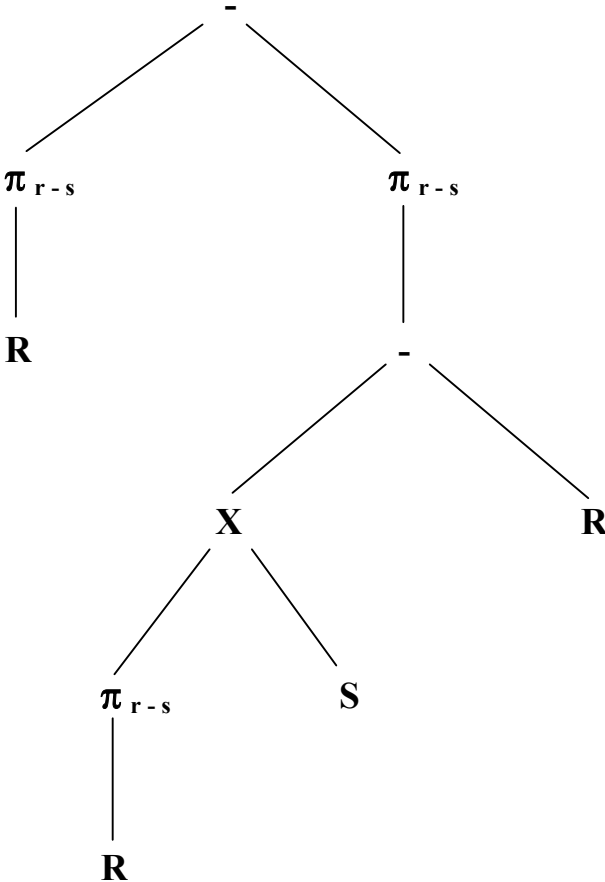
- **ALL LEAVES IN THE TREE ARE BASE RELATIONS**

EXAMPLE1: $R \Join X S \equiv \pi(\sigma(R \times X S))$



EXAMPLE2:

$$R \div S \equiv \pi_{r-s}(R) - \pi_{r-s}((\pi_{r-s}(R) \times S) - R)$$



TRANSFORMATIONS

ALGEBRAIC AND SEMANTIC MANIPULATIONS DESIGNED TO AVOID EXPENSIVE OPERATIONS

ALGEBRAIC MANIPULATIONS

GIVEN RELATIONAL EXPRESSIONS E1, E2 & E3

THE FOLLOWING TRANSFORMATION LAWS ARE
SHOWN FOR CROSS PRODUCTS BUT ALSO APPLY
TO OTHER TYPES OF JOIN OPERATIONS

1) COMMUTATIVE LAW

$$E1 \times E2 \equiv E2 \times E1$$

2) ASSOCIATIVE LAW

$$(E1 \times E2) \times E3 \equiv E1 \times (E2 \times E3)$$

3) CASCADE OF PROJECTIONS

$$\pi_a (\pi_{a,b} (E1)) \equiv \pi_a (E1)$$

4) CASCADE OF SELECTIONS

$$\sigma_{F1} (\sigma_{F2} (E)) \equiv \sigma_{F1 \text{ and } F2} (E)$$

$$\sigma_{F1} (\sigma_{F2} (E)) \equiv \sigma_{F2} (\sigma_{F1} (E))$$

5) COMMUTING SELECTS & PROJECTS

IF CONDITION F INVOLVES ONLY PROJECTED ATTRIBUTES THEN

$$\pi_a(\sigma_F(E)) \equiv \sigma_F(\pi_a(E))$$

6) COMMUTING SELECTS WITH CROSS PRODUCTS

IF CONDITION F INVOLVES ONLY ATTRIBUTES OF E1 THEN

$$\sigma_F(E1 \times E2) \equiv \sigma_F(E1) \times E2$$

IF CONDITION F = F1 AND F2

**WHERE F1 PERTAINS TO E1 AND
F2 PERTAINS TO E2**

THEN

$$\sigma_F(E1 \times E2) \equiv \sigma_{F1}(E1) \times \sigma_{F2}(E2)$$

7) COMMUTING SELECT WITH UNION

$$\sigma_F(E1 \cup E2) \equiv \sigma_F(E1) \cup \sigma_F(E2)$$

8) COMMUTING SELECT WITH DIFFERENCE

$$\sigma_F(E1 - E2) \equiv \sigma_F(E1) - \sigma_F(E2)$$

9) COMMUTING PROJECT WITH CROSS PRODUCT

$$\pi_{a,b}(E1 \times E2) \equiv \pi_a(E1) \times \pi_b(E2)$$

10) COMMUTING PROJECT WITH UNION

$$\pi_{a,b} (E1 \cup E2) \equiv \pi_a (E1) \cup \pi_b (E2)$$

NOTE: PROJECT AND DIFFERENCE ARE NOT COMMUTATIVE

QUERY OPTIMIZATION ALGORITHM

- **SEPARATE SELECTIONS INTO CASCADES OF SELECTIONS**
- **MOVE EACH SELECTION AS FAR DOWN THE QUERY TREE AS POSSIBLE**
- **FOR EACH PROJECTION - EITHER ELIMINATE IT OR MOVE IT AS FAR DOWN THE TREE AS POSSIBLE**
- **COMBINE CASCADES OF SELECTIONS INTO A SINGLE SELECTION**
- **COMBINE CASCADES OF PROJECTIONS INTO A SINGLE PROJECTION**

CHOOSE CANDIDATE LOW-LEVEL PROCEDURES

THE TRANSFORMED QUERY TREE REPRESENTS A SERIES OF LOW-LEVEL OPERATIONS

THE OPTIMIZER HAS A SET OF PREDEFINED LOW-LEVEL IMPLEMENTATION PROCEDURES FOR EACH OPERATION

THE OPTIMIZER USES SYSTEM CATALOG INFORMATION (INDEXES, CARDINALITIES, ETC.) TO ASSOCIATE A COST MEASURE WITH EACH CANDIDATE PROCEDURE

THIS PROCESS IS CALLED ACCESS PATH SELECTION

GENERATE QUERY PLANS AND CHOOSE ONE

THE OPTIMIZER GENERATES A SET OF QUERY PLANS BY COMBINING CANDIDATE LOW-LEVEL PROCEDURES

SOME HEURISTIC IS USED TO LIMIT THE NUMBER OF QUERY PLANS GENERATED

A COST (USUALLY IN DISK I/Os) IS ASSIGNED TO EACH PLAN

THE CHEAPEST PLAN WINS !!!

(ACCURATE COST ESTIMATION IS DIFFICULT BECAUSE NON-TRIVIAL QUERIES REQUIRE THE GENERATION OF INTERMEDIATE RESULTS WHOSE SIZE IS HIGHLY DEPENDENT ON ACTUAL DATA VALUES)