# *Relational Algebra*


Northern Illinois University
September 2018

# Data Manipulation Language

- Data Updating
  - Insert
  - Delete
  - Modify

# Data Manipulation Language

- Data Retrieval Language (Query)
  - Relational Algebra
    - Procedural
  - Relational Calculus
    - Tuple Oriented (QUEL)
    - Domain Oriented (QBE)
  - Structured Query Language (SQL)
    - non-procedural
    - between Rel Algebra and Rel Calculus

# Relational Algebra
# Basic Operators

- Projection
- Selection
- Cartesian Product
- Union
- Set Difference

# Relational Algebra
# Basic Operators

- The result of any relational algebra operation is another relation.

- The empty relation is considered a relation.

# Projection

- Gives a vertical subset of a relation

R

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

$$\pi_{A}(R) =$$

| A |
|---|
| a1 |
| a2 |

Duplicate tuples are eliminated

# Projection

- Can project more than one column

R

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a2 | b2 | c2 | d2 |

$$\pi_{B,C}(R) =$$

| B | C |
|---|---|
| b1 | c1 |
| b2 | c2 |

Duplicate tuples are eliminated

# Selection

- Gives a horizontal subset of a relation

R

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a2 | b2 | c2 | d2 |

$\sigma_{(B=b2)}(R) =$

| A | B | C | D |
|---|---|---|---|
| a2 | b2 | c2 | d2 |

# *Selection*

- Operators that may be used in the select clause

  = ~= > >= < <=

- Operands may be of any data type
- Clauses can be combined using

And $\wedge$ OR $\vee$ NOT ~

# Selection

R

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a2 | b2 | c2 | d2 |
| a3 | b3 | c3 | d3 |
| a4 | b4 | c4 | d4 |

$$\sigma_{(B=b2)\vee(C=c3)}(R) =$$

| A | B | C | D |
|---|---|---|---|
| a2 | b2 | c2 | d2 |
| a3 | b3 | c3 | d3 |

# Cartesian Product

- Set of all tuples [r  s] where r tuples belong to the first relation R and s tuples belong to the second relation S.
- Concatenation of every tuple in R with every tuple in S.

# Cartesian Product

R

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

S

| C | D |
|---|---|
| c1 | d1 |
| c2 | d2 |
| c3 | d3 |

R X S =

| A | B | C | D |
|---|---|---|---|
| a1 | b1 | c1 | d1 |
| a1 | b1 | c2 | d2 |
| a1 | b1 | c3 | d3 |
| a2 | b2 | c1 | d1 |
| a2 | b2 | c2 | d2 |
| a2 | b2 | c3 | d3 |

# *Union*

- Union operator can only be used when the degree of both relations are exactly the same and the corresponding attributes have the same domain called **union compatible** relations.
- Concatenate two relations R and S with r and s tuples respectively into one relation with a maximum of r+s tuples.
- Duplicates tuples are eliminated

## *Union*

R

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

S

| B | D |
|---|---|
| b1 | d1 |
| b2 | d2 |
| b3 | d3 |

$$\pi_{B}(R) \quad \cup \quad \pi_{B}(S)$$

$$\|$$

| B |
|---|
| b1 |
| b2 |
| b3 |

# Set Difference

- The set of tuples in first relation but not in second relation.  (R-S is the set of tuples in R but not in S)
- The two relations MUST be union-compatible

# Set Difference

R

| B | D |
|---|---|
| b1 | d1 |
| b2 | d2 |
| b3 | d3 |

S

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

$$\pi_B(R) - \pi_B(S)$$

||

| B |
|---|
| b3 |

# Set Difference

R

| A | B |
|---|---|
| A1 | b1 |
| a2 | b2 |
| a3 | b3 |

S

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

R - S =

| A | B |
|---|---|
| a3 | b3 |

# Relational Algebra
# Additional Derived Operators

- Intersection
- Division
- Join
  - theta-join
  - equi-join
  - natural join
  - outer join
  - semi-join

# *Intersection*

- Set of tuples that are in both R and S
- R and S MUST be union-compatible

$$R \cap S = R - (R - S)$$

# *Intersection*

R

| A | B |
|---|---|
| A1 | b1 |
| a2 | b2 |
| a3 | b3 |

S

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

$R \cap S =$

| A | B |
|---|---|
| a1 | b1 |
| a2 | b2 |

# *Division*

- Degree of R is r
- Degree of S is s

$R \div S$ is the set of ( r - s ) tuples t such that for s-tuples u in S, the tuple t exists in R

- Equivalent to $T = \pi_{1,2,\ldots\ r\text{-}s}(R)$

$$R \div S = T - \pi_{1,2,\ldots\ r\text{-}s}((T X S) - R)$$

R

| A | B | C | D |
|---|---|---|---|
| a | b | c | d |
| a | b | e | f |
| b | c | e | f |
| e | d | c | d |
| e | d | e | f |
| a | b | d | e |

# *Division*

S

| C | D |
|---|---|
| c | d |
| e | f |

$$R \div S =$$

| A | B |
|---|---|
| a | b |
| e | d |

# Division

R

| A | B | C | D |
|---|---|---|---|
| a | b | c | d |
| a | b | e | f |
| b | c | e | f |
| e | d | c | d |
| e | d | e | f |
| a | b | d | e |

S

| C | D |
|---|---|
| c | d |
| e | f |

$$R \div S =$$

| A | B |
|---|---|
| a | b |
| e | d |

R

| A | B | C | D |
|---|---|---|---|
| a | b | c | d |
| a | b | e | d |
| b | c | e | f |
| e | d | c | d |
| e | d | e | f |
| a | b | d | e |

# *Division*

S

| C |
|---|
| c |
| e |

$R \div S =$

| A | B | D |
|---|---|---|
| a | b | d |

a3 b3 d3

# *Theta-Join*

- The theta-join creates a relation that contains tuples from the Cartesian product of R and S and that satisfy a predicate P.
- $R \bowtie_p S = \sigma_{(p)} (R \times S)$
- P can contain any of the operators of <, <=, >, >=, =, ~=

# Theta-Join

R

| B | A |
|---|---|
| b1 | a1 |
| b2 | a2 |
| b3 | a3 |

S

| X | Y |
|---|---|
| a1 | y1 |
| a2 | y2 |

$$R \bowtie_{(R.A > S.X)} S =$$

| A | B | X | Y |
|---|---|---|---|
| a2 | b2 | a1 | y1 |
| a3 | b3 | a1 | y1 |
| a3 | b3 | a2 | y2 |

## *Equi-Join*

- An equi-join is a theta-join with the predicate P containing only the equal (=) operator.

# *Equi-Join*

R

| B | A |
|---|---|
| b1 | a1 |
| b2 | a2 |
| b3 | a3 |

S

| X | Y |
|---|---|
| x1 | b1 |
| x2 | b2 |

$R \bowtie_{(B = Y)} S =$

| A | B | X | Y |
|---|---|---|---|
| a1 | b1 | x1 | b1 |
| a2 | b2 | x2 | b2 |

# *Natural Join*

- Natural join is an equi-join the relations R and S have common attributes. One of the occurrences of each of the common attributes is eliminated in the resulting relation.

# *Natural Join*

R

| B | A |
|---|---|
| b1 | a1 |
| b2 | a2 |
| b3 | a3 |

S

| X | B |
|---|---|
| x1 | b1 |
| x2 | b2 |

R * S =

| A | B | X |
|---|---|---|
| a1 | b1 | x1 |
| a2 | b2 | x2 |

| B |
|---|
| b1 |
| b2 |

Eliminated in the natural join.
Would NOT be eliminated
in an equi-join.

# *Outer Join*

- Used when even the unmatching tuples from one of the relations used in the join are also wanted in the resulting relations.
- Left outer join: R ⟕ S   a join where tuples from R (left hand relation) that do not match on the common columns with tuples of S are also included in the resulting relation.

# Outer Join

- Right outer join: R⟖S  a join where tuples from S (right hand relation) that do not match on the common columns with tuples of R are also included in the resulting relation.
- Full outer join: R⟗S is a join where tuples from both relations are kept, padding tuples with nulls where no match is found

# Outer Join

R

| B | A |
|---|---|
| b1 | a1 |
| b2 | a2 |
| b3 | a3 |

S

| X | B |
|---|---|
| x1 | b1 |
| x2 | b2 |

## Left outer join

$R ⟕ S$ =

| A | B | X |
|---|---|---|
| a1 | b1 | x1 |
| a2 | b2 | x2 |
| a3 | b3 | null |

# Relations to Use in Example Relational Algebra Exercises

**S**

| S# | SNAME | STATUS | CITY |
|---|---|---|---|
| S1 | Smith | 20 | London |
| S2 | Jones | 10 | Paris |
| S3 | Blake | 30 | Paris |
| S4 | Clark | 20 | London |
| S5 | Adams | 30 | Athens |

# Relations to Use in Example Relational Algebra Exercises

**P**

| P# | PNAME | COLOR | WEIGHT |
|---|---|---|---|
| P1 | Nut | Red | 12 |
| P2 | Bolt | Green | 17 |
| P3 | Screw | Blue | 17 |
| P4 | Screw | Red | 14 |
| P5 | Cam | Blue | 12 |
| P6 | Cog | Red | 19 |

# *Relations to Use in Example Relational Algebra Exercises*

**SP**

| S# | P# | QTY |
|---|---|---|
| S1 | P1 | 300 |
| S1 | P2 | 200 |
| S1 | P3 | 400 |
| S1 | P4 | 200 |
| S1 | P5 | 100 |
| S1 | P6 | 100 |
| S2 | P1 | 300 |
| S2 | P2 | 400 |
| S3 | P2 | 200 |
| S4 | P2 | 200 |
| S4 | P4 | 300 |
| S4 | P5 | 400 |

# Relational Algebra Examples

- Ex1: List the part numbers for all parts supplied.

# Relational Algebra Examples

- Ex2: List the full details of all suppliers.

# Relational Algebra Examples

- Ex3: List supplier numbers for all suppliers in Paris with a status greater than 20.

# Relational Algebra Examples

- Ex4: List the supplier numbers and status for suppliers in Paris in descending order of status.

# Relational Algebra Examples

- Ex5: For each part supplied, get the part number and names of all the cities supplying the part.

# Relational Algebra Examples

- Ex6: List the supplier numbers for all pairs of suppliers such that two suppliers are located in the same city.

# Relational Algebra Examples

- Ex7: List the supplier names for suppliers who supply part P2.

# Relational Algebra Examples

- Ex8: List the supplier numbers for suppliers with status less than the current maximum status value in the S table.

# Relational Algebra Examples

- Ex9: List the supplier names for suppliers who supply at least one red part.

# Relational Algebra Examples

- Ex10: List the supplier numbers for suppliers who supply at least one part also supplied by S2.

# Relational Algebra Examples

- Ex11: List the part numbers for all parts supplied by more than one supplier.

# Relational Algebra Examples

- Ex12: List all the names of suppliers who do not supply part P2.

# Relational Algebra Examples

- Ex13: List the supplier numbers for suppliers who are located in the same city as supplier S1.

# Relational Algebra Examples

- Ex14: List the supplier names for suppliers who supply all the parts.

# Relational Algebra Examples

- Ex15: Get supplier numbers for all suppliers who supply at least all those parts supplied by S2.

# Relational Algebra Examples

- Ex16: Get the part numbers for all parts that either weigh more than 18 pounds or are currently supplied by supplier S2.