# Databases 1

## Extended Relational Algebra

# Relational Algebra

- What is an "Algebra"?
- Mathematical system consisting of:
  - *Operands* --- variables or values from which new values can be constructed.
  - *Operators* --- symbols denoting procedures that construct new values from given values.

# Core Relational Algebra

- Union, intersection, and difference.
  - Usual set operations, but require both operands have the same relation schema.
- Selection: picking certain rows.
- Projection: picking certain columns.
- Products and joins: compositions of relations.
- Renaming of relations and attributes.

# Relational Algebra on Bags

- A *bag* is like a set, but an element may appear more than once.
  - *Multiset* is another name for "bag."

- Example: {1,2,1,3} is a bag. {1,2,3} is also a bag that happens to be a set.

- Bags also resemble lists, but order in a bag is unimportant.
  - Example: {1,2,1} = {1,1,2} as bags, but [1,2,1] != [1,1,2] as lists.

# Why Bags?

- SQL, the most important query language for relational databases is actually a bag language.
  - SQL will eliminate duplicates, but usually only if you ask it to do so explicitly.

- Some operations, like projection, are much more efficient on bags than sets.

# Operations on Bags

- Selection applies to each tuple, so its effect on bags is like its effect on sets.
- Projection also applies to each tuple, but as a bag operator, we **do not eliminate duplicates.**
- Products and joins are done on each pair of tuples, so duplicates in bags have no effect on how we operate.

# Example: Bag Selection

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

S

| B | C |
|---|---|
| 3 | 4 |
| 7 | 8 |

$\text{SELECT}_{A+B<5}$ (R) =

| A | B |
|---|---|
| 1 | 2 |
| 1 | 2 |

# Example: Bag Projection

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

S

| B | C |
|---|---|
| 3 | 4 |
| 7 | 8 |

$PROJECT_A(R)$ =

| A |
|---|
| 1 |
| 5 |
| 1 |

# Example: Bag Product

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

S

| B | C |
|---|---|
| 3 | 4 |
| 7 | 8 |

R * S =

| A | R.B | S.B | C |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |
| 5 | 6 | 3 | 4 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |

# Example: Bag Theta-Join

R

| A | B |
|---|---|
| 1 | 2 |
| 5 | 6 |
| 1 | 2 |

S

| B | C |
|---|---|
| 3 | 4 |
| 7 | 8 |

R JOIN $_{R.B<S.B}$ S =

| A | R.B | S.B | C |
|---|---|---|---|
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 7 | 8 |

# Bag Union

- Union, intersection, and difference need new definitions for bags.
- An element appears in the union of two bags the sum of the number of times it appears in each bag.
- Example: {1,2,1} UNION {1,1,2,3,1} = {1,1,1,1,1,2,2,3}

# Bag Intersection

- An element appears in the intersection of two bags the minimum of the number of times it appears in either.
- Example: {1,1,2,1} INTER {1,1,2,3} = {1,1,2}.

# Bag Difference

- An element appears in the difference $A - B$ of bags as many times as it appears in *A*, minus the number of times it appears in *B*.
  - But never less than 0 times.
- Example: {1,2,1} – {1,2,3} = {1}.

# Beware: Bag Laws <> Set Laws

- Not all algebraic laws that hold for sets also hold for bags.
- For one example, the commutative law for union (*R* UNION *S* = *S* UNION *R* ) *does* hold for bags.
  - Since addition is commutative, adding the number of times *x* appears in *R* and *S* doesn't depend on the order of *R* and *S*.

# An Example of Inequivalence

- Set union is *idempotent*, meaning that
  *S* UNION *S* = *S*.
- However, for bags, if *x* appears *n* times in *S*, then it appears 2*n* times in *S* UNION *S*.
- Thus *S* UNION *S* <> *S* in general.

# The Extended Algebra

1. DELTA = eliminate duplicates from bags.
2. TAU = sort tuples.
3. *Extended projection* : arithmetic, duplication of columns.
4. GAMMA = grouping and aggregation.
5. OUTERJOIN: avoids “dangling tuples” = tuples that do not join with anything.

# Duplicate Elimination

- R1 := DELTA(R2).
- R1 consists of one copy of each tuple that appears in R2 one or more times.

# Example: Duplicate Elimination

R =

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 1 | 2 |

DELTA(R) =

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |

# Sorting

- R1 := $TAU_L$ (R2).
  - *L* is a list of some of the attributes of R2.
- R1 is the list of tuples of R2 sorted first on the value of the first attribute on *L*, then on the second attribute of *L*, and so on.
  - Break ties arbitrarily.
- TAU is the only operator whose result is neither a set nor a bag.
- ORDER BY in SQL

# Example: Sorting

R =

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 2 |

$TAU_B$ (R) = [(5,2), (1,2), (3,4)]

# Extended Projection

- Using the same $PROJ_L$ operator, we allow the list L to contain arbitrary expressions involving attributes, for example:
  1. Arithmetic on attributes, e.g., *A*+*B*.
  2. Duplicate occurrences of the same attribute.

# Example: Extended Projection

R =

| A | B |
|---|---|
| 1 | 2 |
| 3 | 4 |

$\text{PROJ}_{A+B,A,A}$ (R) =

| A+B | A1 | A2 |
|---|---|---|
| 3 | 1 | 1 |
| 7 | 3 | 3 |

# Aggregation Operators

- Aggregation operators are not operators of relational algebra.
- Rather, they apply to entire columns of a table and produce a single result.
- The most important examples: SUM, AVG, COUNT, MIN, and MAX.

# Example: Aggregation

R =

| A | B |
|---|---|
| 1 | 3 |
| 3 | 4 |
| 3 | 2 |

SUM(A) = 7
COUNT(A) = 3
MAX(B) = 4
AVG(B) = 3

# Grouping Operator

- R1 := $GAMMA_L$ (R2). *L* is a list of elements that are either:
  1. Individual (*grouping* ) attributes.
  2. AGG(*A* ), where AGG is one of the aggregation operators and *A* is an attribute.

# Applying GAMMA$_L$(R)

- Group *R* according to all the grouping attributes on list *L*.
  - That is, form one group for each distinct list of values for those attributes in *R*.
- Within each group, compute AGG(*A* ) for each aggregation on list *L*.
- Result has grouping attributes and aggregations as attributes. One tuple for each list of values for the grouping attributes and their group's aggregations.

# Example: Grouping/Aggregation

R =

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

$\text{GAMMA}_{A,B,\text{AVG}(C)}$ (R) = ??

First, group *R* :

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

Then, average *C* within groups:

| A | B | AVG(C) |
|---|---|---|
| 1 | 2 | 4 |
| 4 | 5 | 6 |

# Outerjoin

- Suppose we join $R$ JOIN$_C$ $S$.
- A tuple of $R$ that has no tuple of $S$ with which it joins is said to be *dangling*.
  - Similarly for a tuple of $S$.
- Outerjoin preserves dangling tuples by padding them with a special NULL symbol in the result.

# Example: Outerjoin

R =

| A | B |
|---|---|
| 1 | 2 |
| 4 | 5 |

S =

| B | C |
|---|---|
| 2 | 3 |
| 6 | 7 |

(1,2) joins with (2,3), but the other two tuples are dangling.

R OUTERJOIN S =

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | NULL |
| NULL | 6 | 7 |