CMPT 321
Fall 2017

Relational algebra of bags

Lecture 02.03

By *Marina Barsky*

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 is actually a bag language.
- SQL will eliminate duplicates, but usually only if you ask it to do so explicitly.

- Some operations, like **projection** or **union**, are much more efficient on bags than sets.
  - Why?

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)

A	B
1	2
5	6
1	2

S(B C)

B	C
3	4
7	8

$\sigma_{A+B<5}$ (R) =

A	B
1	2
1	2

Example: Bag Projection

R(A B)

A	B
1	2
5	6
1	2

S(B C)

B	C
3	4
7	8

π_A (R) =

A
1
5
1

Bag projection yields always the same number of tuples as the original relation.

Example: Bag Product

R(A B)

A	B
1	2
5	6
1	2

S(B C)

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

- *Each copy* of the tuple **(1,2)** of **R** is being paired with each tuple of **S**.
- So, the duplicates do not have an effect on the way we compute the product.

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:

  $\{1,2,1\} \cup \{1,1,2,3,1\}$

  $= \{1,1,1,1,1,2,2,3\}$

Bag Intersection

- An element appears in the **intersection** of two bags the **minimum** of the number of times it appears in either.

- Example:

  $\{1,2,1\} \cap \{1,2,3\}$

  $= \{1,2\}$.

Bag Difference

- An element appears in **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.

Example

- Set union is *idempotent*, meaning that

  $\boldsymbol{S} \cup \boldsymbol{S} = \boldsymbol{S}$.

- However, for bags, if $\boldsymbol{x}$ appears $\boldsymbol{n}$ times in $\boldsymbol{S}$, then it appears $2n$ times in $S \cup S$.

- Thus $\boldsymbol{S} \cup \boldsymbol{S}$ **!=** $\boldsymbol{S}$ in general.

Extended Algebra (for bags)

1. δ: eliminate duplicates from bags.
2. τ: sort tuples.
3. γ: grouping and aggregation.
4. **Extended projection**: arithmetic, duplication of columns.

Example: Duplicate Elimination

$R_1 := \delta(R_2)$

R_1 consists of one copy of each tuple that appears in R_2 one or more times.

R =

A	B
1	2
3	4
1	2

δ(R) =

A	B
1	2
3	4

Sorting

$R_1 := \tau_L (R_2)$

- *L* is a list of some of the attributes of R_2.

R_1 is the list of tuples of R_2 sorted first on the value of the first attribute on *L*, then on the second attribute of *L*, and so on.

Aggregation Operators AGG

- They apply to entire columns of a table and produce a single result.

- The most important examples:
  - SUM
  - AVG
  - COUNT
  - MIN
  - MAX

Example: Aggregation

R =

A	B
1	3
3	4
3	2

SUM(A) = 7
COUNT(A) = 3
MAX(B) = 4
MIN(B) = 2
AVG(B) = 3

Grouping Operator

$\mathbf{R_1 := \gamma_{\mathit{L}} (R_2)}$

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.

Example: Grouping/Aggregation

R =

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

$\gamma_{A,B,\text{AVG}(C)}$ (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

$\gamma_L(R)$ - Formally

- 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

StarsIn(title, year, star_name**)**

- For each star who has appeared in at least three movies give the earliest year in which he or she appeared.
  - First we group, using *starName* as a grouping attribute.
  - Then, we compute the MIN(*year*) for each group.
  - Also, we need to compute the COUNT(*title*) aggregate for each group, for filtering out those stars with less than three movies.

- $\pi_{star\_name,minYear}(\sigma_{ctTitle \geq 3}(\gamma_{starName,MIN(year) \rightarrow minYear,COUNT(title) \rightarrow ctTitle}(StarsIn)))$

Example: Extended Projection

Using the same π_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.

R =

A	B
1	2
3	4

$\pi_{A+B \rightarrow C, A \rightarrow A1, A \rightarrow A2}$ (R) =

C	A1	A2
3	1	1
7	3	3