Algebraic and Logical Query Languages

Spring 2011

Instructor: Hassan Khosravi

Relational Operations on Bags

Extended Operators of Relational Algebra

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.
- eliminate duplicates, but usually only if you ask it to do so explicitly.
- SQL will
- Some operations, like **projection** or **union**, are much more efficient on bags than sets.
  - Why?
    - Union of two relations in bags: copy one relation and add the other to it
    - Projection: in sets you need to compare all the rows in the new relation to make sure they are unique. In bags, you don't need to do anything extra

Operations on Bags

- **Selection** applies to each tuple, so its effect on bags is like its effect on sets.

R

A	B
1	2
5	6
1	2

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

A	B
1	2
1	2

- **Projection** also applies to each tuple, but as a bag operator, we do not eliminate duplicates.

R

A	B
1	2
5	6
1	2

π_A (R)

A
1
5
1

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

Operations on Bags

- **Products** and **joins** are done on each pair of tuples, so duplicates in bags have no effect on how we operate.

R

A	B
1	2
5	6
1	2

S

B	C
3	4
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.

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

Bag Union, Intersection, Difference

- **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} ∪ {1,1,2,3,1} = {1,1,1,1,1,2,2,3}

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

    {1,2,1} ∩ {1,2,3} = {1,2}.

- 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

  $$S \cup S = S.$$

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

- Thus $S \cup S \;!\!= S$ in general.

The Extended Algebra

1. δ: eliminate duplicates from bags.
2. Aggregation operators such as sum and average
3. γ: grouping of tuples according to their value in some attributes
4. Extended projection: arithmetic, duplication of columns.
5. τ: sort tuples according to one or more attributes.
6. **OUTERJOIN**: avoids "dangling tuples" = tuples that do not join with anything.

Example: Duplicate Elimination

- R_1 consists of one copy of each tuple that appears in R_2 one or more times.
  - $R_1 := \delta(R_2)$

(R)

A	B
1	2
5	6
1	2

δ(R)

A	B
1	2
5	6

Aggregation Operators

- They apply to entire columns of a table and produce a single result.
- The most important examples:
  - SUM
  - AVG
  - COUNT
  - MIN
  - MAX

Aggregation Operators

- Sum(B) = 2 +4+2+2 =10
- AVG(A) = (1+3+1+1) / 4 = 1.5
- MIN(A) = 1
- MAX(A)=4
- COUNT(A)=4

A	B
1	2
3	4
1	2
1	2

Grouping Operator

Sometimes we like to use the aggregate functions over a group of tuples and not all of them.

For example we want to compute the total number of minutes of movies produced by each studio.

Studio name	Sum of Lengths
Disney	12345
MGM	54321

$R_1 := \gamma_L (R_2)$

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.

γ_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, starName**)**

- **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.

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

Example: Grouping/Aggregation

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

R

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

First, group *R* :

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

Then, average *C* within groups:

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

Example: Extended Projection

- In extended projection operator, lists can have the following kind of elements
  - A Single attribute of R
  - An expression x→y, where x and y are names for attributes. Take attribute x of R and rename it to y.
  - An expression E→z, where E is an expression involving attributes of R, constants, arithmetic operators, and string operators, and z is a new name.
    - a +b =x
    - c || d = y

R

A	B
1	2
5	6
1	2

$\pi_{A,\ A+B \rightarrow X}$ (R)

A	X
1	3
5	11
1	3

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.

- τ is the only operator whose result is neither a set nor a bag.

Outerjoin

Motivation

- Suppose we join $R \bowtie S$.
- A tuple of *R* which doesn't join with any tuple of *S* is said to be *dangling*.
  - Similarly for a tuple of *S*.
  - **Problem**: We loose dangling tuples.

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.

$$R \overset{\circ}{\bowtie} S$$

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

Example: Left Outerjoin

R

A	B
1	2
4	5

S

B	C
2	3
6	7

(The left Outerjoin: Only pad dangling tuples from the left table

$R \overset{\circ}{\bowtie} {}_{L}S$

A	B	C
1	2	3
4	5	NULL

Example: RightOuterjoin

R

A	B
1	2
4	5

S

B	C
2	3
6	7

(The left Outerjoin: Only pad dangling tuples from the left table

$$R \overset{\circ}{\bowtie}_{R} S$$

A	B	C
1	2	3
NULL	6	7

Theta Outerjoin

$U \overset{\circ}{\bowtie}_{C} V$

U

A	B	C
1	2	3
4	5	6
7	8	9

V

B	C	D
2	3	10
2	3	11
6	7	12

$U \overset{\circ}{\bowtie}_{A>V.C} V$

A	U.B	U.C	V.B	V.C	D
4	5	6	2	3	10
4	5	6	2	3	11
7	8	9	2	3	10
7	8	9	2	3	11
1	2	3	NULL	NULL	NULL
NULL	NULL	NULL	6	7	12