VirginiaTech

CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #6: Extended Operators in Relational Algebra

Announcements

- Homework 1 due today
- Project Assignment 2 released
  - Due on Friday Feb 22nd, beginning of the class
- NO class and office hours on Wednesday Feb 20th

Specific Project Guidelines

- We will create an account and a database for each student.
- A database for each project group will be created.
  - The name of the database is the name of your group name.
  - Only the members of each project will be able to access the database for their project.
- A webpage detailing how you can access the database is maintained.
- You can create as many tables within a database as you want.

General Project Guidelines

- The database schema is not something that should change often.
  - Think long and hard about your schema.
  - DROP may be better than ALTER TABLE.
- Do not delete the files containing raw data.
- Read documentation for the RDBMS you are using.

Bags

- A *bag* (or *multi-set*) is like a set, but an element may appear more than once.
- Example: {1,2,1,3} is a bag.
- Example: {1,2,3} is also a bag that happens to be a set.

Why Bags?

- So far, we said RA and SQL operate on sets
- Real RDBMSs treat relations as bags of tuples.
  - SQL, is actually a bag language.
- Performance is one of the main reasons; duplicate elimination is expensive since it requires sorting.
  - Some operations, like projection, are much more efficient on bags than sets.
- If we use bag semantics, we may have to redefine the meaning of each relation algebra operator.

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.

Bag Semantics: Projection and Selection

- Project: process each tuple independently; a tuple might occur multiple times
- Selection: process each tuple independently…

R

A	B	C
1	2	3
1	2	4
2	3	4
2	3	4

$\pi_{A,B}(R)$

A	B
1	2
1	2
2	3
2	3

$\sigma_{C \geq 3}(R)$

A	B	C
1	2	3
1	2	4
2	3	4
2	3	4

Bag Union

- An element appears in the union of two bags the sum of the number of times it appears in each bag.
- R **U** S: if tuple t appears k times in R and l times in S, t appears in R **U** S k + l times.

R

A	B
1	2
1	2
2	3
2	3

S

A	B
1	2
1	2
1	2
2	3
2	4

$R \cup S$

A	B
1	2
1	2
1	2
1	2
1	2
2	3
2	3
2	3
2	4

Bag Intersection

- An element appears in the intersection of two bags the minimum of the number of times it appears in either.
- R ∩ S: if tuple t appears k times in R and l times in S, t appears min {k, l} times in R ∩ S

R	
A	B
1	2
1	2
2	3
2	3

S	
A	B
1	2
1	2
1	2
2	3
2	4

R ∩ S	
A	B
1	2
1	2
2	3

Bag Difference

- An element appears in the difference R - S of bags as many times as it appears in *R*, minus the number of times it appears in *S*.
  - But never less than 0 times.
- R –S: if tuple t appears k times in R and l times in S, t appears in R – S max{0, k – l} times.

R

A	B
1	2
1	2
2	3
2	3

S

A	B
1	2
1	2
1	2
2	3
2	4

R – S

A	B
2	3

Bag Semantics: Products and Joins

- Product (×): If a tuple r appears k times in a relation R and tuple s appears l times in a relation S, then the tuple <r, s> appears kl times in R × S.

- Theta-join and Natural join (⋈): Since both can be expressed as applying a selection followed by a projection to a product, use the semantics of selection, projection, and the product.

Extended Operators

- Powerful operators based on basic relational operators and bag semantics.
- Sorting: convert a relation into a list of tuples.
- Duplicate elimination: turn a bag into a set by eliminating duplicate tuples.
- Grouping: partition the tuples of a relation into groups, based on their values among specified attributes.
- Aggregation: used by the grouping operator and to manipulate/combine attributes.
- Extended projections: projection on steroids.
- Outerjoin: extension of joins that make sure every tuple is in the output.

Sorting

RA $\tau_{A_1,A_2,\ldots}(R)$.

SQL `SELECT ... FROM ...WHERE ... ORDER BY` $A_1, A_2, \ldots$.

- The result is a list of tuples in R but with the tuples sorted by their values in attributes A_1, A_2, ...
- In SQL, use `DESC` after an attribute to specify sorting in descending order; `ASC` is the default.
- If you use the result in another query, sorted order is lost.

Example: Sorting

R = (

A	B
1	2
3	4
5	2

)

τ_B (R) = [(5,2), (1,2), (3,4)]

Duplicate Elimination

RA $\delta(R)$ is the relation containing exactly one copy of each tuple in R.

SQL `SELECT DISTINCT ...`

- Duplicate elimination is *expensive*, since tuples must be sorted or partitioned.
- Set operations in SQL (`UNION`, `INTERSECT`, and `EXCEPT`) operate on *sets* of tuples, i.e., they first eliminate duplicates.
- To make these operators treat relations as bags, follow the operation with the keyword `ALL`.

Example: Duplicate Elimination

R = (

A	B
1	2
3	4
1	2

)

σ(R) =

A	B
1	2
3	4

Extended Projection

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

Example: Extended Projection

R = (

A	B
1	2
3	4

)

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

A+B	A1	A2
3	1	1
7	3	3

Aggregation Operators

- Operators that summarize or aggregate the values in a single attribute of a relation.
- Operators are the same in relational algebra and SQL.
- All operators treat a relation as a bag of tuples.
- SUM: computes the sum of a column with numerical values.
- AVG: computes the average of a column with numerical values.
- MIN and MAX:
  - for a column with numerical values, computes the smallest or larges value, respectively.
  - for a column with string or character values, computes the lexicographically smallest or largest values, respectively.
- COUNT: computes the number of tuples in a column.
- In SQL, can use COUNT (*) to count the number of tuples in a relation.

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

- How do we answer the query "Count the number of classes and the total enrollment of the classes each department teaches"?
- Can we answer the query using the operators discussed so far?
- We need to group the tuples of Teach by DeptName and then aggregate within each group.
- Use the grouping operator.

Applying $\gamma_L(R)$

- How do we answer the query "Count the number of classes and total enrollment of the classes each department teaches"?
  1. Group `Courses` by `DeptName`.
  2. For each group, create a new attribute that stores the number of classes taught by the department.
  3. For each group, create a new attribute that stores the total enrollment of the classes taught by the department.
- γ_L(`Courses`), where L is a list containing three elements:
  1. `DeptName`: the *grouping attribute*,
  2. `COUNT(Number)` $\rightarrow$ `NumCourses`: an *aggregated attribute* computing the count of the `Number` attribute in each group and naming the new attribute `NumCourses`, and
  3. `SUM(Enrollment)` $\rightarrow$ `TotalEnrollment`: an aggregated attribute computing the total of the `Enrollment` attribute and naming the new attribute `TotalEnrollment`.

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* by *A* and *B* :

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 $R \infty_C S$.
- 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