The Relational Algebra

Textbook ch. 6.5-6.7

Overview

- Operations on whole relations
  - Inputs are sets, output is a set
- Can nest arbitrarily complex expressions
- SELECT σ, PROJECT Π
- Set union ∪, intersection ∩, difference –, on compatible relations
- Cartesian product ⊗ and various flavors of JOIN
  - Division ÷ sort of inverse of product
- Aggregate functions (unofficial)

Select

- Unary operation
- Select a subset of tuples from a relation, based upon a condition
  - use AND, OR, NOT for compound conditions
- Result: a table with same attributes as original: a proper subset
  - may be given a name (temporary)
- Notation:

$$\sigma_{\textbf{condition}}(\textbf{\textit{relationname}})$$

Project

- Unary operation
- Select a subset of columns
- Result: a table with same number of rows as original
  - not actually a subset of the original (unlike σ)
  - may be given a name (temporary)
- Notation:

$$\Pi_{\textbf{col-list}}(\textbf{\textit{relationname}})$$

Join

- A binary operation on relations
- Result is a whole relation
- General description: a ⊗ followed by a σ.
  - The σ condition equates or otherwise relates common attributes between the two relations
  - Often a superfluous common attribute is removed
- Notation (these slides):

  $R1_{JN\ \text{join-condition}}\ R2$

"Equi" and "Natural" join

- Common attributes are compared for equality
  - no need to specify a join condition
  - could join on more than one attribute
  - need to list attributes if names are not the same
- "Natural join": Superfluous columns are removed automatically
- Notation (our text): $R1 *_{\text{attr-lists}} R2$

Division: R1 ÷ R2

- Sort of the reverse of Cartesian product
- Like integer division in that any "remainder" is discarded
- Main idea: find all the tuples in R1 which are joined to all the values in R2
  - the R2 attributes are discarded
- Same thing can be accomplished with combination of Π, ⊗, –

10/12/97 F-7

Division Details of R = R1÷R2

- R1 (dividend): attribute set X ∪ Y, |R1| rows
- R2 (divisor): attribute set Y, |R2| rows
- R (quotient):
  - attribute set X, i.e., the attributes of R1 not in R2
  - at most |R1|/|R2| rows
  - A row is in the answer (R) if that row (X attributes) occurs in R1 with each combination of the rows (Y attributes) of R2.

10/12/97 F-8

Division Examples

- Who would have lots to talk about with Bessie? "Find (all) customers who have rented (all) the same movies as Bessie has."
- What airlines compete with Horizon Air? "Find the airlines which serve a city also served by Horizon" (not a division query).
- Which airline is best positioned to put Horizon Air out of business? "Find the airlines which fly to (all) the cities served by Horizon" (a division query).

10/12/97 F-9

Aggregate Functions

- Technically, not part of R.A.
- Actual query languages will implement many of these
- (Usually) unary operators, take a whole relation and compute a value
- COUNT, AVERAGE, MAX, MIN
- Result is returned as a relation with one row and one column
  - i.e., not as a scalar number

10/12/97 F-10

Grouping and Aggregates

- Rows may be grouped based on attribute values
  - Think of it as a sort on those attributes
- Aggregate functions can be applied to the grouped relation
  - Computes a value for each group
- Result returned as a relation with one row for each group, one column for each aggregate function

10/12/97 F-11

Grouping Notation and Example

- <grouping attributes> $\Im$ <agg. function list> (relation)
- "List number of employees and average salary for each department"

DNAME	COUNT (SSN)	AVERAGE (SALARY)
SW Support	54	$30,301
HW Support	18	$72,600
Grounds	5	$89,600

10/12/97 F-12

Looking Ahead

- Order of operations affects efficiency
- Example: σ(R1) *σ(R2) probably much faster than σ(R1 * R2)
- Large joins can be particularly taxing
- Ideally, we do <u>not</u> let this affect how we write queries!
- Smart DBMSs do "query optimization"
  - automatically reorder operations for efficiency