

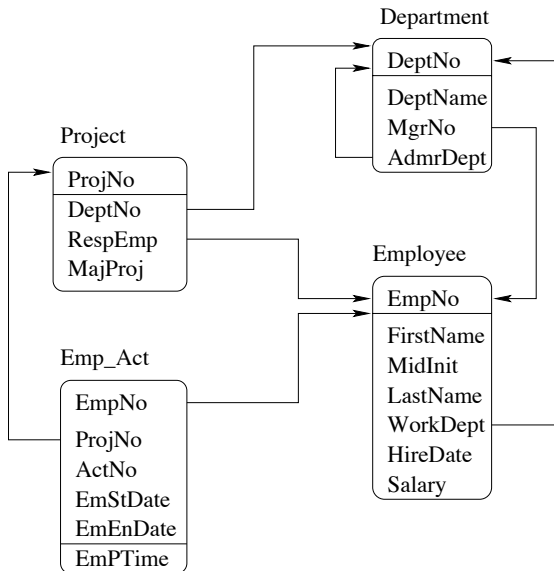
# Relational Algebra

Grant Weddell

Cheriton School of Computer Science  
University of Waterloo

CS 348  
Introduction to Database Management  
Winter 2017

# Database Schema Used in Examples



- the relational algebra consists of a set of *operators*
- relational algebra is *closed*
  - each operator takes as input zero or more relations
  - each operator defines a single output relation in terms of its input relation(s)
  - relational operators can be composed to form expressions that define new relations in terms of existing relations.
- *Notation:*  
 $R$  is a relation name;  $E$  is a relational algebra expression

# Primary Relational Operators

- Relation Name:  $R$

# Primary Relational Operators

- Relation Name:  $R$
- Selection:  $\sigma_{condition}(E)$ 
  - result schema is the same as  $E$ 's
  - result instance includes the subset of the tuples of  $E$  that each satisfies the condition

# Primary Relational Operators

- Relation Name:  $R$
- Selection:  $\sigma_{condition}(E)$ 
  - result schema is the same as  $E$ 's
  - result instance includes the subset of the tuples of  $E$  that each satisfies the condition
- Projection:  $\pi_{attributes}(E)$ 
  - result schema includes only the specified attributes
  - result instance could have as many tuples as  $E$ , except that duplicates are eliminated

# Primary Relational Operators (cont'd)

- Rename:  $\rho(R(\overline{F}), E)$ 
  - $\overline{F}$  is a list of terms of the form *oldname*  $\rightarrow$  *newname*
  - returns the result of  $E$  with columns renamed according to  $\overline{F}$ .
  - remembers the result as  $R$  for future expressions

# Primary Relational Operators (cont'd)

- Rename:  $\rho(R(\overline{F}), E)$ 
  - $\overline{F}$  is a list of terms of the form *oldname*  $\rightarrow$  *newname*
  - returns the result of  $E$  with columns renamed according to  $\overline{F}$ .
  - remembers the result as  $R$  for future expressions
- Product:  $E_1 \times E_2$ 
  - result schema has all of the attributes of  $E_1$  and all of the attributes of  $E_2$
  - result instance includes one tuple for every pair of tuples (one from each expression result) in  $E_1$  and  $E_2$
  - sometimes called cross-product or Cartesian product
  - renaming is needed when  $E_1$  and  $E_2$  have common attributes



# Cross Product Example

$R$

$AAA$	$BBB$
$a_1$	$b_1$
$a_2$	$b_2$
$a_3$	$b_3$

$S$

$CCC$	$DDD$
$c_1$	$d_1$
$c_2$	$d_2$

$R \times S$

$AAA$	$BBB$	$CCC$	$DDD$
$a_1$	$b_1$	$c_1$	$d_1$
$a_2$	$b_2$	$c_1$	$d_1$
$a_3$	$b_3$	$c_1$	$d_1$
$a_1$	$b_1$	$c_2$	$d_2$
$a_2$	$b_2$	$c_2$	$d_2$
$a_3$	$b_3$	$c_2$	$d_2$

# Select,Project,Product Examples

- Note: Use *Emp* to mean the Employee relation, *Proj* the project relation
- Find the last names and hire dates of employees who make more than \$100000.

# Select,Project,Product Examples

- Note: Use *Emp* to mean the Employee relation, *Proj* the project relation
- Find the last names and hire dates of employees who make more than \$100000.

$$\pi_{LastName,HireDate}(\sigma_{Salary>100000}(Emp))$$

# Select,Project,Product Examples

- Note: Use *Emp* to mean the Employee relation, *Proj* the project relation
- Find the last names and hire dates of employees who make more than \$100000.

$$\pi_{LastName,HireDate}(\sigma_{Salary>100000}(Emp))$$

- For each project for which department E21 is responsible, find the name of the employee in charge of that project.

# Select,Project,Product Examples

- Note: Use  $Emp$  to mean the Employee relation,  $Proj$  the project relation
- Find the last names and hire dates of employees who make more than \$100000.

$$\pi_{LastName, HireDate}(\sigma_{Salary > 100000}(Emp))$$

- For each project for which department E21 is responsible, find the name of the employee in charge of that project.

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo = E21}(\sigma_{RespEmp = EmpNo}(Emp \times Proj)))$$

- Conditional join:  $E_1 \bowtie_{condition} E_2$ 
  - equivalent to  $\sigma_{condition}(E_1 \times E_2)$
  - special case: *equijoin*

$$Proj \bowtie_{(RespEmp=EmpNo)} Emp$$

- Conditional join:  $E_1 \bowtie_{condition} E_2$ 
  - equivalent to  $\sigma_{condition}(E_1 \times E_2)$
  - special case: *equijoin*

$$Proj \bowtie_{(RespEmp=EmpNo)} Emp$$

- Natural join ( $E_1 \bowtie E_2$ )
  - The result of  $E_1 \bowtie E_2$  can be formed by the following steps
    - 1 form the cross-product of  $E_1$  and  $E_2$  (renaming duplicate attributes)
    - 2 eliminate from the cross product any tuples that do not have matching values for *all pairs* of attributes common to schemas  $E_1$  and  $E_2$
    - 3 project out duplicate attributes
  - if no attributes in common, this is just a product

# Example: Natural Join

- Consider the natural join of the Project and Department tables, which have attribute DeptNo in common
  - the schema of the result will include attributes ProjName, DeptNo, RespEmp, MajProj, DeptName, MgrNo, and AdmrDept
  - the resulting relation will include one tuple for each tuple in the Project relation (why?)



# Set-Based Relational Operators

- Union ( $R \cup S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear either in  $R$  or in  $S$  or in both
  
- Union Compatible:
  - Same number of fields.
  - 'Corresponding' fields have the same type

# Set-Based Relational Operators

- Union ( $R \cup S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear either in  $R$  or in  $S$  or in both
- Difference ( $R - S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear in  $R$  and that do not appear in  $S$
  
- Union Compatible:
  - Same number of fields.
  - 'Corresponding' fields have the same type

# Set-Based Relational Operators

- Union ( $R \cup S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear either in  $R$  or in  $S$  or in both
- Difference ( $R - S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear in  $R$  and that do not appear in  $S$
- Intersection ( $R \cap S$ ):
  - schemas of  $R$  and  $S$  must be “union compatible”
  - result includes all tuples that appear in both  $R$  and  $S$
- Union Compatible:
  - Same number of fields.
  - ‘Corresponding’ fields have the same type

# Relational Division

$A$	$B$	$C$
$a_1$	$b_1$	$c_1$
$a_1$	$b_1$	$c_2$
$a_1$	$b_2$	$c_2$
$a_2$	$b_1$	$c_1$
$a_2$	$b_1$	$c_2$
$a_2$	$b_2$	$c_2$
$a_2$	$b_3$	$c_3$
$a_3$	$b_1$	$c_1$

$B$	$C$
$b_1$	$c_1$
$b_1$	$c_2$
$b_2$	$c_2$

$A$
$a_1$
$a_2$

# Division is the Inverse of Product

$R$		$S$		$R \times S$			$(R \times S)/S$
$A$		$B$	$C$	$A$	$B$	$C$	$A$
$a_1$		$b_1$	$c_1$	$a_1$	$b_1$	$c_1$	$a_1$
$a_2$		$b_1$	$c_2$	$a_1$	$b_2$	$c_2$	
		$b_2$	$c_1$	$a_2$	$b_1$	$c_1$	
			$c_2$	$a_2$	$b_1$	$c_2$	
				$a_2$	$b_2$	$c_2$	

# Summary of Relational Operators

$$\begin{array}{l} E ::= R \\ | \sigma_{condition}(E) \\ | \pi_{attributes}(E) \\ | \rho(R(\overline{F}), E) \\ | E_1 \times E_2 \\ | E_1 \bowtie_{condition} E_2 \\ | E_1 \bowtie E_2 \\ | E_1 \cup E_2 \\ | E_1 \cap E_2 \\ | E_1 - E_2 \\ | E_1 / E_2 \end{array}$$

# Algebraic Equivalences

- This:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

# Algebraic Equivalences

- This:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(E \bowtie_{RespEmp=EmpNo} P))$$



# Algebraic Equivalences

- This:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(E \bowtie_{RespEmp=EmpNo} P))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(E \bowtie_{RespEmp=EmpNo} \sigma_{DeptNo=E21}(P))$$

# Algebraic Equivalences

- This:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(E \bowtie_{RespEmp=EmpNo} P))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(E \bowtie_{RespEmp=EmpNo} \sigma_{DeptNo=E21}(P))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\left( \begin{array}{l} \pi_{LastName, EmpNo}(E) \\ \pi_{ProjNo, RespEmp}(\sigma_{DeptNo=E21}(P)) \end{array} \right) \bowtie_{RespEmp=EmpNo})$$

# Algebraic Equivalences

- This:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(\sigma_{RespEmp=EmpNo}(E \times P)))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\sigma_{DeptNo=E21}(E \bowtie_{RespEmp=EmpNo} P))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(E \bowtie_{RespEmp=EmpNo} \sigma_{DeptNo=E21}(P))$$

- is equivalent to this:

$$\pi_{ProjNo, LastName}(\left( \pi_{LastName, EmpNo}(E) \right) \bowtie_{RespEmp=EmpNo} \left( \pi_{ProjNo, RespEmp}(\sigma_{DeptNo=E21}(P)) \right))$$

- More on this topic later when we discuss database tuning...

# Relational Completeness

## Definition (Relationally Complete)

A query language that is at least as expressive as relational algebra is said to be *relationally complete*.

## Definition (Relationally Complete)

A query language that is at least as expressive as relational algebra is said to be *relationally complete*.

- The following languages are all relationally complete:
  - safe relational calculus
  - relational algebra
  - SQL

## Definition (Relationally Complete)

A query language that is at least as expressive as relational algebra is said to be *relationally complete*.

- The following languages are all relationally complete:
  - safe relational calculus
  - relational algebra
  - SQL
- SQL has additional expressive power because it captures duplicate tuples, unknown values, aggregation, ordering, ...