Lecture Notes for Lecture 3 of CS 5200 (Database Management Systems) for the Summer 1, 2019 session at the Northeastern University Silicon Valley Campus.

Relational Algebra

Philip Gust,
Clinical Instructor
Department of Computer Science

Lecture 2 Review

- A database management system (DBMS) stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.
- The design depends on its architecture. It can be centralized or decentralized or hierarchical
- An *n-tier architecture* divides the whole system into related but independent **n** modules, including the DBMS.
- Data models define how the logical structure of a database is modeled. ER Model is best used for the conceptual design of a database.
- In an RDBMS, entities and relationships are implemented as tables with entity tuples and attribute columns.

Relational Algebra

- In this lecture, we will look at the underlying mathematical model on which Edgar Codd based the design his relational database.
- We will learn about the history and motivation of Codd's relational algebra, and then look in more detail at the individual operations.
- Finally, we will use an online tool that can execute relational algebra queries against pre-defined data bases to help prepare us for studying relational database systems.

Relational Algebra

What is Relational Algebra?

- Relational algebra is a family of algebras with a well-founded semantics used for modelling the data stored in relational databases, and defining queries on it.
- It was first created by Edgar F. Codd while at the IBM Research Center in San Jose, California.
- Relational algebra provides a theoretical foundation for relational databases, particularly query languages for such databases, chief among which is SQL.
- Relational algebra received little attention outside of pure mathematics until the publication of E.F. Codd's landmark paper in 1970.

Relational Algebra

What Relational Algebra Do?

- Relational algebra takes instances of relations as input and yields instance of relations as output.
- Unary or binary operators accept relations as input and yield relations as output.
- Operations are performed recursively on a relation and intermediate results are also relations.
- Relations in relational algebra are sets of tuples, so includes basic set operations, such as subset, superset, union, intersection, difference, and cartesian product.

Relational Algebra

Relation

- A query and its resulting relation can be thought of in one of two ways: imperatively or declaratively.
- The query is an *imperative command* that the relational calculator executes to produce a relational result.
- The query is a *declarative statement* that describes the desired relational result, without stating how to achieve it.
- A relational expression represents an *intentional relation*, while an *extensional relation* is one whose values are explicitly given. The two are equivalent in relational algebra.

Relational Algebra

Fundamental Operations of Relational Algebra

- Select (σ) : select tuples that satisfy predicate
- Project ($\prod$) : project columns that satisfy predicate
- Union ($\cup$) : binary union between relations
- Set difference ($-$) : tuples in one relation but not other
- Rename (ρ): name output relation
- Order (τ): order output relation by attribute
- Group (γ): aggregate attributes into groups
- Various join operations

Relational Algebra

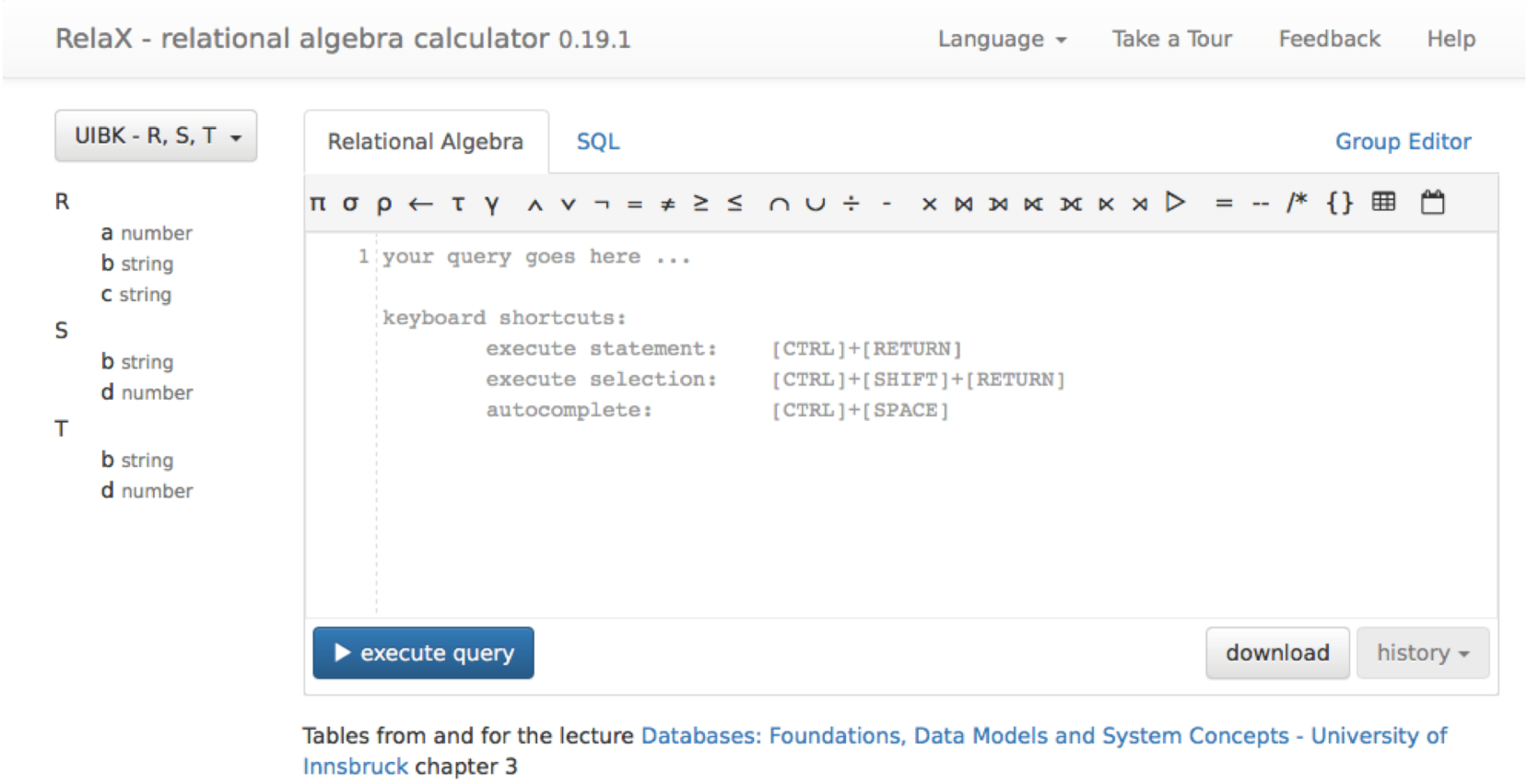
Demonstration of Relational Operators Using the *RelaX* Relational Algebra Calculator:

- http://dbis-uibk.github.io/relax/calc.htm

- Online tool for evaluating both relational algebra expressions.
- Also evaluates SQL expressions and shows equivalent relational algebra expressions
- Includes a number of datasets with the ability to create and load new ones

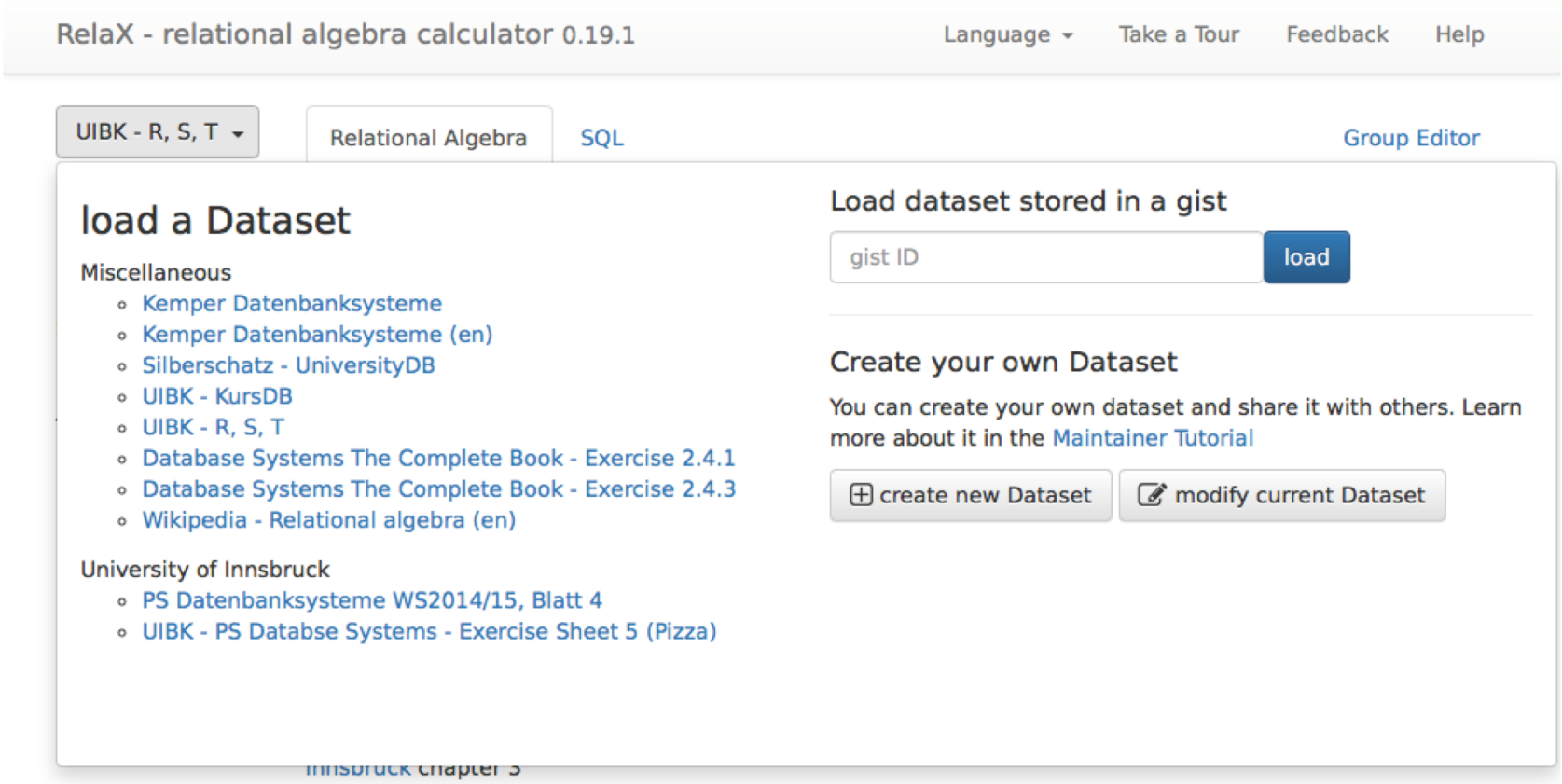
Relational Algebra

RelaX Relational Algebra Calculator



Relational Algebra

RelaX Relational Algebra Calculator



Relational Algebra

Relation

- The Wikipedia relational algebra database provides the following relations:

Employee
- Name (string)
- Empid (number)
- DeptName ()

Dept
- DeptName (string)
- Manager (string)

Completed
- Student (string)
- Task (string)

DBProject
- Task (string)

Car
- CarModel (string)
- CarPrice (number)

Boat
- BoatModel (string)
- BoatPrice (number)

Relational Algebra

Relation

- Evaluating the relation in the calculator displays its content.
  Employee

Employee

Employee.Name	Employee.EmpId	Employee.DeptName
Harry	3415	Finance
Sally	2241	Sales
George	3401	Finance
Harriet	2202	Sales
Tim	1123	Executive

Relational Algebra

Relation

- Evaluating the relation in the calculator displays its content.
  Dept

Dept

Dept.DeptName	Dept.Manager
Sales	Harriet
Production	Charles

Relational Algebra

Select Operation (σ)

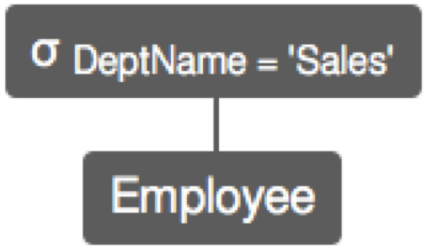
- Selects tuple that satisfies given predicate from relation
- Notation: $\sigma_p(r)$
- σ (sigma) stands for selection predicate (sigma and select both start with ‘s’)
- *r* stands for relation.
- *p* is prepositional logic formula which may use connectors like and, or, and not. These terms may use relational operators like − =, ≠, ≥, < , >, ≤

Relational Algebra

Select Operation (σ)

- Example:

  σ DeptName = 'Sales' Employee



σ DeptName = 'Sales' (Employee)

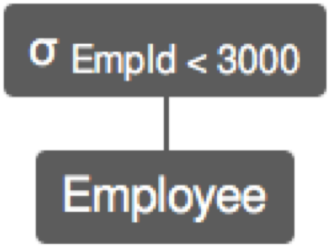
Employee.Name	Employee.EmpId	Employee.DeptName
Sally	2241	Sales
Harriet	2202	Sales

Relational Algebra

Select Operation (σ)

- Example:

  σ EmpId < 3000 Employee



σ EmpId < 3000 (Employee)

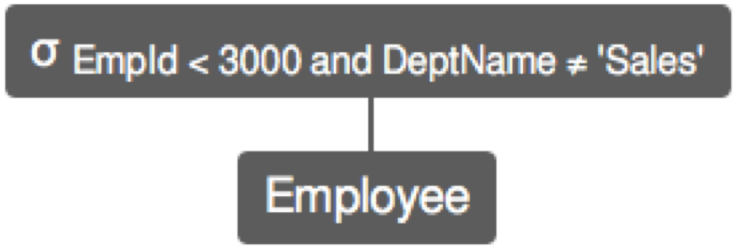
Employee.Name	Employee.EmpId	Employee.DeptName
Sally	2241	Sales
Harriet	2202	Sales
Tim	1123	Executive

Relational Algebra

Select Operation (σ)

- Example:

  σ EmpId < 3000 ∧ DeptName ≠ 'Sales' Employee



σ EmpId < 3000 and DeptName ≠ 'Sales' (Employee)

Employee.Name	Employee.EmpId	Employee.DeptName
Tim	1123	Executive

Relational Algebra

Project Operation (∏)

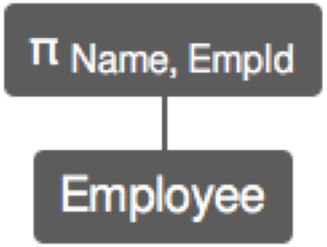
- Projects column(s) that satisfy a given predicate.
- Notation: $\prod_{A1, A2, An} (r)$
- ∏ (pi) stands for project predicate (pi and project both start with ‘p’)
- A_1, A_2 , A_n are attribute names of relation **r**. Duplicate rows are automatically eliminated, as relation is a set.

Relational Algebra

Project Operation (∏)

- Example:

  π Name, EmpId Employee



π Name, EmpId (Employee)

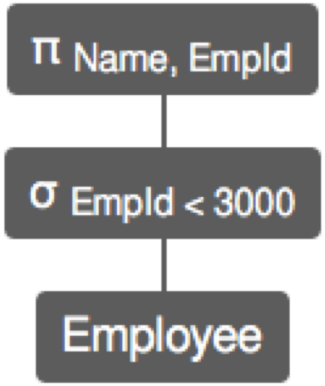
Employee.Name	Employee.EmpId
Harry	3415
Sally	2241
George	3401
Harriet	2202
Tim	1123

Relational Algebra

Project Operation (∏)

- Example:

  π Name, EmpId (σ EmpId < 3000 Employee)



π Name, EmpId (σ EmpId < 3000 Employee)

Employee.Name	Employee.EmpId
Sally	2241
Harriet	2202
Tim	1123

Relational Algebra

Union Operation (∪)

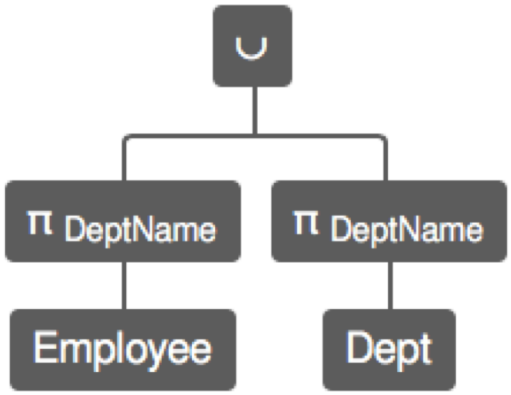
- Projects column(s) that satisfy a given predicate
- Notation: r ∪ s
- ∪ stands for the union operator (same as set union)
- **r** and **s** are either database relations or relation result set (temporary relation).
- For a union operation to be valid, the following conditions must hold
  - **r**, and **s** must have the same number of attributes.
  - Attribute domains must be compatible.
  - Duplicate tuples are automatically eliminated.

Relational Algebra

Union Operation (∪)

- Example:

  (π DeptName Employee) ∪ (π DeptName Dept)



π DeptName Employee ∪ π DeptName Dept

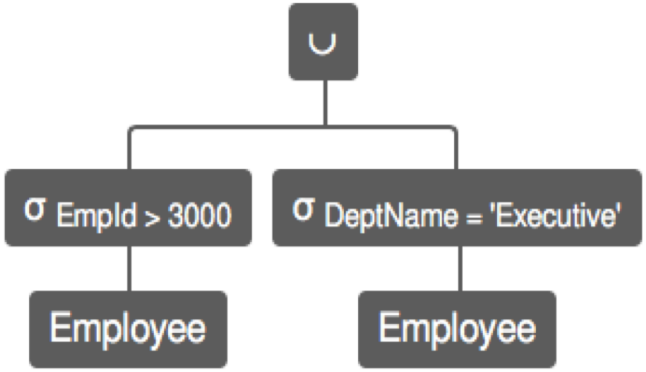
Employee.DeptName
Finance
Sales
Executive
Production

Relational Algebra

Union Operation (∪)

- Example:

  (σ EmpId > 3000 Employee) ∪ (σ DeptName = 'Executive' Employee)



σ EmpId > 3000 Employee ∪ σ DeptName = 'Executive' Employee

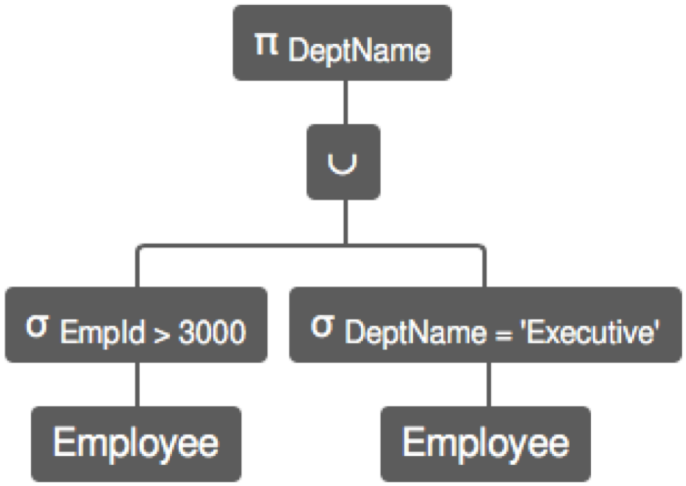
Employee.Name	Employee.EmpId	Employee.DeptName
Harry	3415	Finance
George	3401	Finance
Tim	1123	Executive

Relational Algebra

Union Operation (∪)

- Example:

  π DeptName ((σ EmpId > 3000 Employee) ∪ (σ DeptName = 'Executive' Employee))



π DeptName ((σ EmpId > 3000 Employee) ∪ (σ DeptName = 'Executive' Employee))

Employee.DeptName
Finance
Executive

Relational Algebra

Intersect Operation (∩)

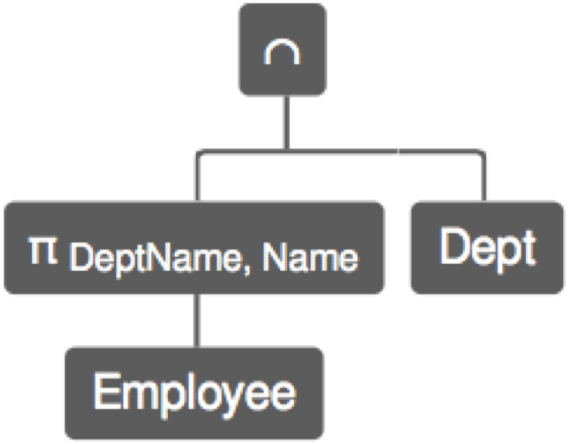
- Result of set intersect is tuples that are in both relations.
- Notation: r ∩ s
- ∩ stands for the intersect operator (same as set union)
- **r** and **s** are either database relations or relation result set (temporary relation).
- For a intersect operation to be valid, the following conditions must hold
  - **r**, and **s** must have the same number of attributes.
  - Attribute domains must be compatible.
  - Duplicate tuples are automatically eliminated.

Relational Algebra

Intersect Operation (∩)

- Example:

  (π DeptName, Name Employee) ∩ Dept



(π DeptName, Name Employee) ∩ Dept

Employee.DeptName	Employee.Name
Sales	Harriet

Relational Algebra

Difference Operation (−)

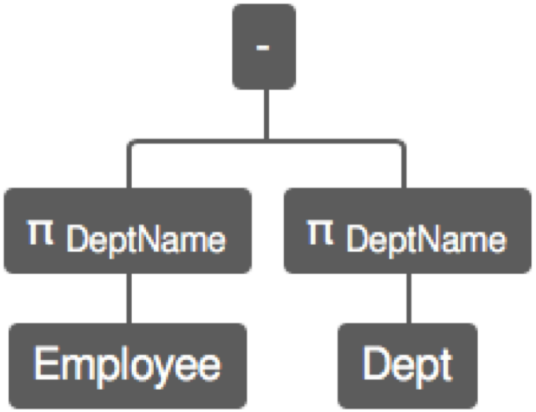
- Result of set difference operation is tuples, which are present in one relation but are not in the second relation.
- Notation: **r** − **s**
- - stands for the difference operator (same as set difference)
- **r** and **s** are relations
- Finds all the tuples that are present in **r** but not in **s**

Relational Algebra

Difference Operation (–)

- Example:

  π DeptName Employee - π DeptName Dept



π DeptName Employee - π DeptName Dept

Employee.DeptName
Finance
Executive

Relational Algebra

Sort Operation (τ)

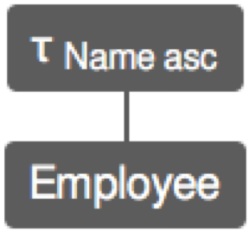
- The sort operation orders the tuples in the output relation by the value of an attribute. ’sort' operation is denoted with small Greek letter tau τ.
- Notation: $\tau_{a1\ x,\ a2\ x\ldots}(E)$
- τ is the sort operator (tau)
- ai is the attribute to sort on
- x is the sort order (asc, desc)
- E is the relation to sort
- Sorts tuples first by a_1, then by a_2, etc.

Relational Algebra

Sort Operation (τ)

- Example:

  τ Name asc Employee



τ Name asc Employee

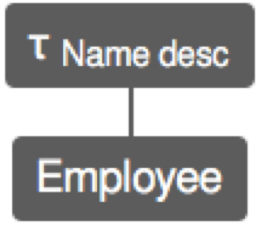
Employee.Name	Employee.EmpId	Employee.DeptName
George	3401	Finance
Harriet	2202	Sales
Harry	3415	Finance
Sally	2241	Sales
Tim	1123	Executive

Relational Algebra

Sort Operation (τ)

- Example:

  τ Name desc Employee



τ Name desc Employee

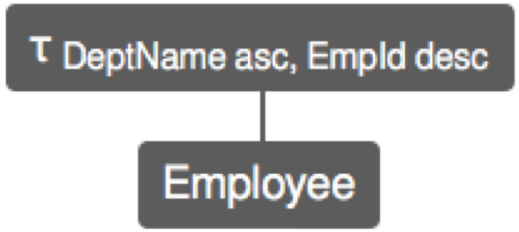
Employee.Name	Employee.EmpId	Employee.DeptName
Tim	1123	Executive
Sally	2241	Sales
Harry	3415	Finance
Harriet	2202	Sales
George	3401	Finance

Relational Algebra

Sort Operation (τ)

- Example:

  τ DeptName asc, EmpId desc Employee



τ DeptName asc, EmpId desc Employee

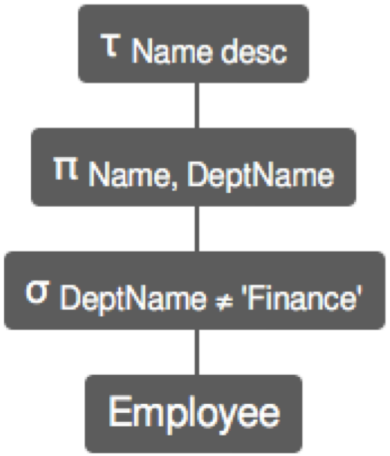
Employee.Name	Employee.EmpId	Employee.DeptName
Tim	1123	Executive
Harry	3415	Finance
George	3401	Finance
Sally	2241	Sales
Harriet	2202	Sales

Relational Algebra

Sort Operation (τ)

- Example:

  τ Name desc (π Name, DeptName (σ DeptName ≠ 'Finance' Employee))



τ Name desc (π Name, DeptName (σ DeptName ≠ 'Finance' Employee))

Employee.Name	Employee.DeptName
Tim	Executive
Sally	Sales
Harriet	Sales

Relational Algebra

Group Operation (γ)

- The group operation aggregates tuples in the output relation by a group of attributes. ‘group' operation is denoted with small Greek letter gamma γ.
- Notation - $\gamma_{x}(E)$
- γ (gamma) is the group operator (gamma and group both start with ‘g’)
- x is the group operation
- E is the relationship
- The output expression **E** is an aggregation of one or more attributes according to the operation **x**.

Relational Algebra

Group Operation (γ)

- Group operators include:

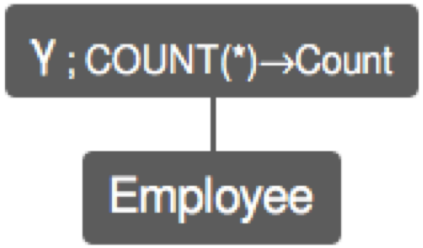
operator	number	string	date
count(*)	yes	yes	yes
count(column)	yes	yes	yes
min(column)	yes	yes	yes
max(column)	yes	yes	yes
sum(column)	yes	no	no
avg(column)	yes	no	no

Relational Algebra

Group Operation (γ)

- Example:

  γ count(*)->Count Employee



Y ; COUNT(*)→Count Employee

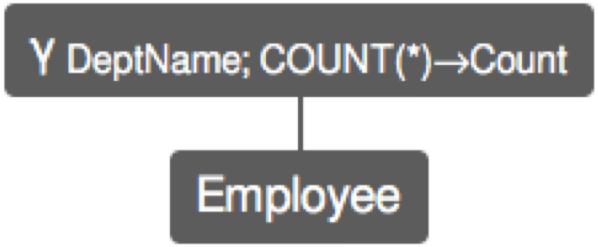
Count
5

Relational Algebra

Group Operation (γ)

- Example:

  γ DeptName;count(*)->Count Employee



γ DeptName; COUNT(*)→Count Employee

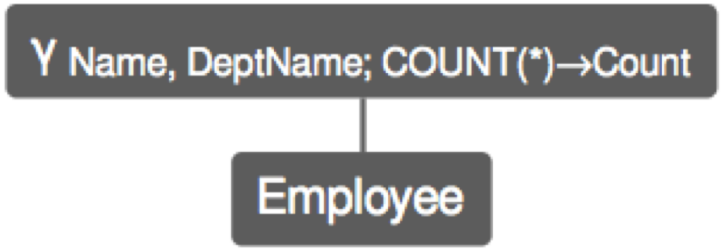
Employee.DeptName	Count
Finance	2
Sales	2
Executive	1

Relational Algebra

Group Operation (γ)

- Example:

  γ Name; count(*)->Count Employee



γ Name, DeptName; COUNT(*)→Count Employee

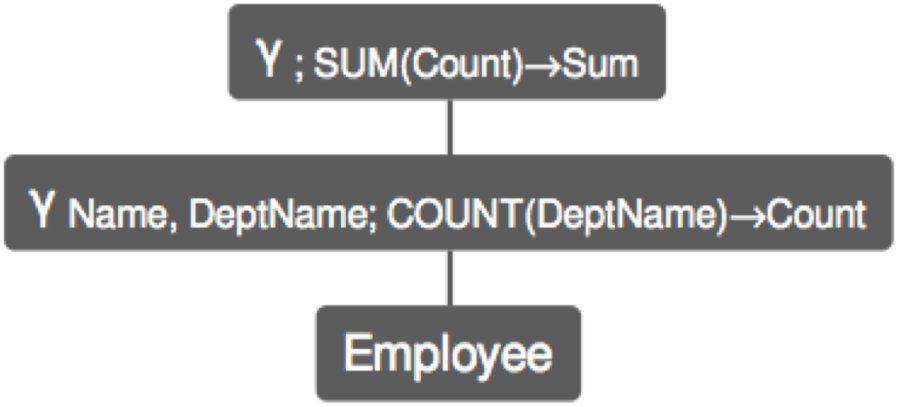
Employee.Name	Employee.DeptName	Count
Harry	Finance	1
Sally	Sales	1
George	Finance	1
Harriet	Sales	1
Tim	Executive	1

Relational Algebra

Group Operation (γ)

- Example:

  γ sum(Count)->Sum (γ Name; count(*)->Count Employee)



Y ; SUM(Count)→Sum (Y Name, DeptName; COUNT(DeptName)→Count Employee)

Sum
5

Relational Algebra

Rename Operation (ρ)

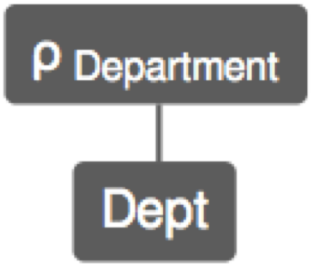
- The rename operation renames the output relation or attributes in the output relation. ‘rename' operation is denoted with small Greek letter rho ρ.
- Notation - $\rho_{E'}(E)$ or $\rho_{a1'<-a1,\ a2'<-a2}(E)$
- ρ (rho) is the rename operator (rho and rename both start with ‘r’)
- E’ is the new relationship name or a1’ and a2’ are the new attribute names
- E is the relationship
- The output expression **E** is a relation that has been renamed or whose attributes have been renamed.

Relational Algebra

Group Operation (γ)

- Example:

  ρ Department Dept



ρ Department Dept

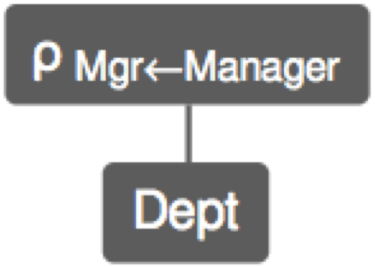
Department.DeptName	Department.Manager
Sales	Harriet
Production	Charles

Relational Algebra

Group Operation (γ)

- Example:

  ρ Mgr←Manager Dept



ρ Mgr←Manager Dept

Dept.DeptName	Dept.Mgr
Sales	Harriet
Production	Charles

Relational Algebra

Joins are binary operations that combine two relations in different ways.

- Cross Join (X)
- Natural join (⋈)
- Left Outer Join (⟕)
- Right Outer Join (⟖)
- Full Outer Join (⟗)
- Left Semijoin (⋉)
- Right Semijoin (⋊)
- Antijoin (▷)

Relational Algebra

Joins are binary operations that combine two relations in different ways.

Join	Description
Cross Join ($r \times s$)	all possible combinations of tuples from r and s
Natural join ($r \bowtie s$)	combines tuples from r and s with matching attributes
Left Outer Join ($r \mathbin{⟕} s$)	tuples from r combined with matching tuples from s or null
Right Outer Join ($r \mathbin{⟖} s$)	tuples from s combined with matching tuples from r or null
Full Outer Join ($r \mathbin{⟗} s$)	combines results from left and right outer join
Left Semijoin ($r \ltimes s$)	selects tuples from the r with matching tuples in s
Right Semijoin ($r \rtimes s$)	selects tuples from s with matching tuples in r
Antijoin ($r \triangleright s$)	selects tuples from r with no matching tuples in s

Relational Algebra

Cross Join (X)

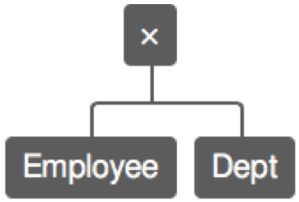
- Combines information of two different relations into one by creating all possible combinations of tuples from each relation.
- Notation: **r** X **s**
- X is the cross join operator (same as cross product for set)
- **Also known as Cartesian Product**

Relational Algebra

Cross Join (X)

- Example:

  Employee × Dept



Employee × Dept

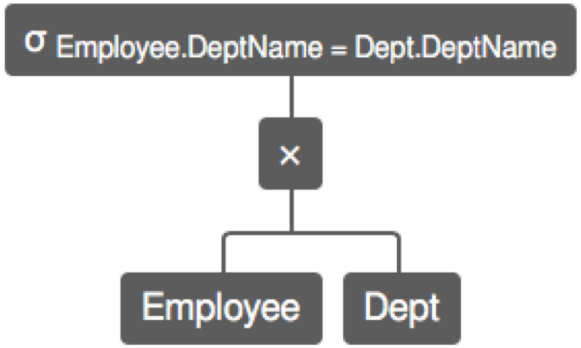
Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Harry	3415	Finance	Sales	Harriet
Harry	3415	Finance	Production	Charles
Sally	2241	Sales	Sales	Harriet
Sally	2241	Sales	Production	Charles
George	3401	Finance	Sales	Harriet
George	3401	Finance	Production	Charles
Harriet	2202	Sales	Sales	Harriet
Harriet	2202	Sales	Production	Charles
Tim	1123	Executive	Sales	Harriet
Tim	1123	Executive	Production	Charles

Relational Algebra

Cross Join (X)

- Example:

  σ Employee.DeptName = Dept.DeptName (Employee × Dept)



σ Employee.DeptName = Dept.DeptName (Employee × Dept)

Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Sally	2241	Sales	Sales	Harriet
Harriet	2202	Sales	Sales	Harriet

Relational Algebra

Natural Join (⋈)

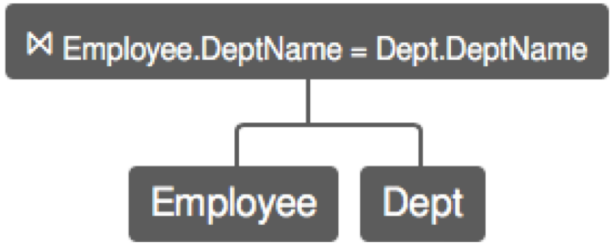
- Combines information of two different relations into one by combining tuples from each that have matching attributes
- Notation: **r** $\bowtie_x$ **s**
- ⋈ is natural join operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is a new type with tuples that match the relation

Relational Algebra

Natural Join (⋈)

- Example:

  Employee ⋈ Employee.DeptName = Dept.DeptName Dept



Employee ⋈ Employee.DeptName = Dept.DeptName Dept

Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Sally	2241	Sales	Sales	Harriet
Harriet	2202	Sales	Sales	Harriet

Relational Algebra

Left Outer Join (⟕)

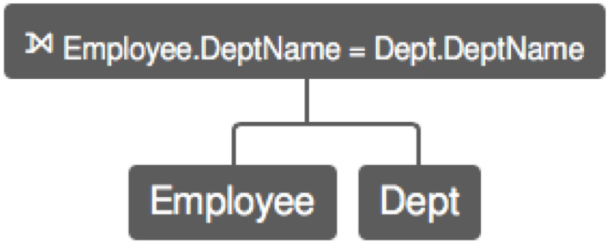
- Combines information of two different relations into one with tuples from the left relation combined with matching tuples from the right relation or null of no match
- Notation: **r** $⟕_x$ **s**
- ⟕ is left outer join operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is a new type with tuples from r combined with matching tuples from s or NULL attributes if no match

Relational Algebra

Left Outer Join (⟕)

- Example:

  Employee ⟕ Employee.DeptName = Dept.DeptName Dept



Employee ⟕ Employee.DeptName = Dept.DeptName Dept

Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Harry	3415	Finance	*null*	*null*
Sally	2241	Sales	Sales	Harriet
George	3401	Finance	*null*	*null*
Harriet	2202	Sales	Sales	Harriet
Tim	1123	Executive	*null*	*null*

Relational Algebra

Right Outer Join (⟖)

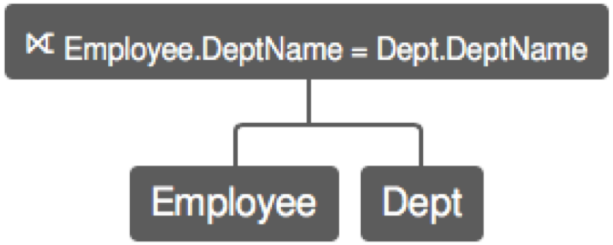
- Combines information of two different relations into one with tuples from the right relation combined with matching tuples from the left relation or null of no match
- Notation: **r** $\rtimes_x$ **s**
- ⟖ is right outer join operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is a new type with tuples from s combined with matching tuples from r or NULL attributes if no match.

Relational Algebra

Right Outer Join (⟖)

- Example:

  Employee ⟖ Employee.DeptName = Dept.DeptName Dept



Employee ⟖ Employee.DeptName = Dept.DeptName Dept

Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Sally	2241	Sales	Sales	Harriet
Harriet	2202	Sales	Sales	Harriet
null	*null*	*null*	Production	Charles

Relational Algebra

Full Outer Join (⟗)

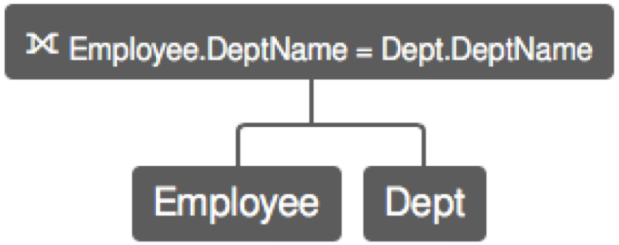
- Combines information of two different relations into one with tuples from the one relation combined with matching tuples from the other relation or null if no match
- Notation – **r** ⟗ $_{x}$ **s**
- ⟗ is full outer join operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is a new type with tuples from s combined with tuples from r, or NULL attributes if no match.

Relational Algebra

Full Outer Join (⟗)

- Example:

  Employee ⟗ Employee.DeptName = Dept.DeptName Dept



Employee ⟗ Employee.DeptName = Dept.DeptName Dept

Employee.Name	Employee.EmpId	Employee.DeptName	Dept.DeptName	Dept.Manager
Harry	3415	Finance	*null*	*null*
Sally	2241	Sales	Sales	Harriet
George	3401	Finance	*null*	*null*
Harriet	2202	Sales	Sales	Harriet
Tim	1123	Executive	*null*	*null*
null	*null*	*null*	Production	Charles

Relational Algebra

Left Semijoin (⋉)

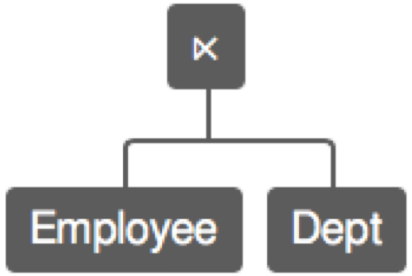
- Combines information of two different relations by selecting tuples from the left relation with matching tuples in the right relation
- Notation: $\mathbf{r} \ltimes_{x} \mathbf{s}$
- ⋉ is full left semijoin operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is tuples from r where there is a matching attribute in s.

Relational Algebra

Left Semijoin (⋉)

- Example:

  Employee ⋉ Dept



Employee ⋉ Dept

Employee.Name	Employee.EmpId	Employee.DeptName
Sally	2241	Sales
Harriet	2202	Sales

Relational Algebra

Right Semijoin (⋊)

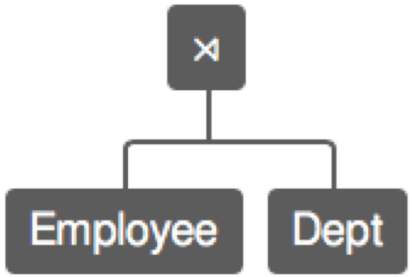
- Combines information of two different relations by selecting tuples from the right relation with matching tuples in the left relation
- Notation: $\mathbf{r} \rtimes_{x} \mathbf{s}$
- ⋊ is right semijoin operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is tuples from s where there is a matching attribute in r.

Relational Algebra

Right Semijoin (⋊)

- Example:

  Employee ⋊ Dept



Employee ⋊ Dept

Dept.DeptName	Dept.Manager
Sales	Harriet

Relational Algebra

Antijoin (▷)

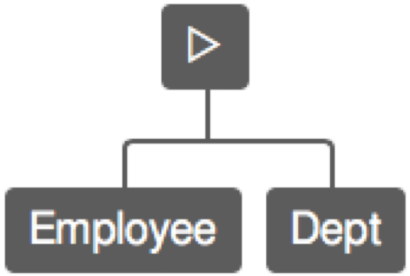
- Combines information of two different relations by selecting tuples from the left relation with no matching tuples in the right relation
- Notation: **r** ▷ **s**
- ▷ is right semijoin operator
- r is the first relation
- x is is a selection on the two the relations
- s is the second relation
- The result is tuples from r that have no matching attributes in s.
- Equivalent to r – (r ⋉ s)

Relational Algebra

Antijoin (▷)

- Example:

  Employee ▷ Dept



Employee ▷ Dept

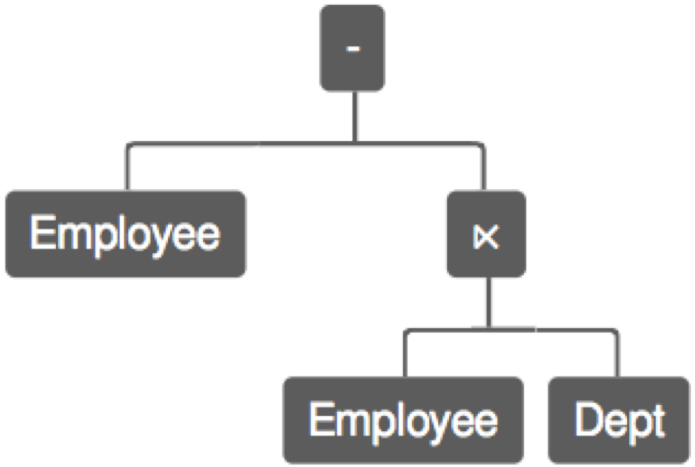
Employee.Name	Employee.EmpId	Employee.DeptName
Harry	3415	Finance
George	3401	Finance
Tim	1123	Executive

Relational Algebra

Antijoin (▷)

- Example:

  Equivalent to Employee - (Employee ⋉ Dept)



Employee - (Employee ⋉ Dept)

Employee.Name	Employee.EmpId	Employee.DeptName
Harry	3415	Finance
George	3401	Finance
Tim	1123	Executive