Relations and Relational Algebra

CS1F

IM Lecture 8

Craig Macdonald

Where are we now…….

- **Set Theory** (Tuesday)
- **Relations and Relational Algebra** (today)

- **SQL and querying a database** (lecture 9)
- **Populate and Query your database** (Next weeks Lab (week 5))
- Database Assessment (week 6)
  - Verbally in labs with queries
  - Report (intro, description, rationale, conclusion)
  - TOTAL: 24 marks

Recap on Sets

3

- What are sets? {1,3,5}
- Set builder notation for making sets; comparing sets
- Operators: making new sets from other sets
  - ∪ **union**
  - ∩ **intersection**
  - **– difference**
  - $\overline{A}$ **complement**
  - ⊕ **symmetric difference**
- And… the Cartesian Product…

Cartesian Product

4

Let A and B be sets

- The ***cartesian product*** of A and B (A X B) is
  - the set of all ***ordered pairs*** *(i.e. tuples)*

<a,b> where a ∈ A and b ∈ B

A={0,1}, B={a,b,c}

Set A has elements of the domain {0,1}
Set B has elements of the domain {a,b,c}

A X B =

1st element of each tuple has domain {0,1}
2nd element of each tuple has domain {a,b,c}

{<0,a>,<0,b>,<0,c>, <1,a>, <1,b> ,<1,c>}

How this helps with databases?

Cartesian Products & Relations

5

- Consider a relation, where the two attributes have the same domain restrictions: R(A,B)
  - R({0,1}, {a,b,c})
- All possible rows in R:
  - Same as {0,1} x {a,b,c}
- If we had less rows in our relation R, it would be a **subset** of {0,1} x {a,b,c}

A ={0,1}	B ={a,b,c}
0	a
0	b
0	c
1	a
1	b
1	c

Relations

6

- Relationships between elements of sets are represented using a structure called a ***relation***
- *"A relation R is a subset of the Cartesian product of the domains that define R"*
  - *i.e. of the domains of its attributes*
- Relations are the fundamental data structure used to store information in databases

Relations and their properties

7

A and B are sets

a **binary relation R** from A to B
is a subset of the Cartesian product A X B

- That is – a set R of ordered pairs where the first element of each ordered pair comes from A and the second element comes from B

- Notation
  - a R b denotes <a,b> ∈ R
  - a is related to b by R

An example relation

8

- Let A = {3,4,5} and B = {x,y}
- Cartesian Product A x B =
  {<3,x>, <3,y>, <4,x>, <4,y>, <5,x>, <5,y>}
- A Relation R between A and B is a **subset** of this Cartesian Product
- E.g. R= {<3,x>, <4,x>, <5,y>} is a relation from A to B
  - 3 R x True
  - 3 R y False

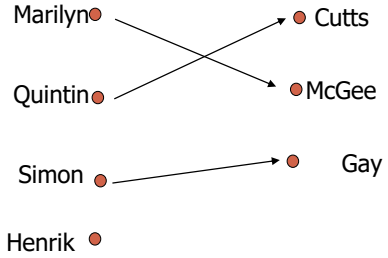
Representing a relation

9

Forename = {Marilyn, Quintin, Simon, Henrik}
Surname = {McGee, Cutts, Gay}

Domains

Names = {<Marilyn,McGee>,<Quintin,Cutts>, <Simon, Gay>}

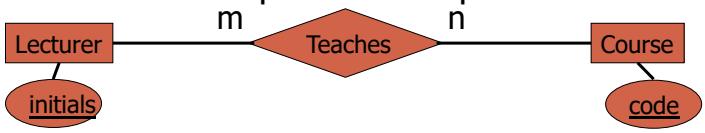


R	Cutts	McGee	Gay
Marilyn		x	
Quintin	x		
Simon			x
Henrik			

Foreign Keys as Domains

10

- Foreign Keys **further** restrict an attribute's domain
- Consider the example from the previous lecture



  - Teaches(Lecturer : VARCHAR(10), Course: VARCHAR(5))
  - As stated, the domain of the Course attribute is all possible strings of 0..5 in length: "", "a", "b", … "zzzzz", …
- But the many-to-many rule tells us that Teaches.Course will have a foreign key constraint with Course.Code
  - Therefore any Teaches.Course value must also occur in the primary key of Course.Code

RELATIONAL ALGEBRA

Relational Algebra

12

- Sets and relations help us understand how the underlying data elements are structured in a database
- Relational Algebra helps us understand how the DBMS **extracts** the required data from the DB

Querying a Database

13

A query…

…is performed when we wish to extract from the database a *subset* of the information that answers some question:

- "What are the department names?"
- "Tell me all the data held about male employees."
- "What are the names of the employees in the R&D department?“

Querying a Database

14

- Extraction of results consists of ‘programs’ built out of
  - retrieving data as a subset of some relation
  - combining two relations together in a meaningful way

Querying Languages

15

Two ways of querying a database:

- procedural (relational algebra, Pandas)
  - sequence of operations
  - the output of each operation is the input to the next operation

result ← F4 (F2 (F1(tableA), tableB), F3(tableC))

based on set theory

- declarative (SQL)
  - describes the desired results (in terms of conditions)
  - the DBMS works out the operations

result ← CONDITIONS (tableA, tableB, tableC)

This is internally implemented as RA operations

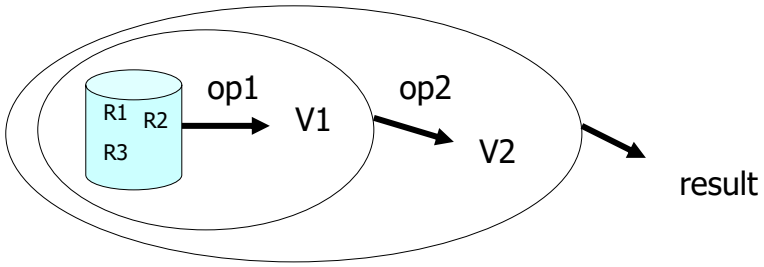
RA is key to understanding SQL query processing!

Querying a Relational Database - **Procedural**

16

Procedural approach - describe the sequence of operations

- e.g., using ***relational algebra***
  - a formal language which allows the description of queries as *a sequence of operations*
  - abstractly describes the query



Preliminaries

17

- A query is a sequence of operations applied to relation instances, and the result of a query is also a relation instance
  - Schema of relations are fixed (c.f. previous lectures)
  - The query will then execute over any valid relation instance…
  - … and return a relation
  - The schema of the result can also be determined

Relational Algebra

18

- Relational algebra is a set of operations which can be combined to provide the result of a query in the form of a relation
- The algebra consists of:
  - a collection of operations which fall into two categories:
    - special relational operations
    - traditional set operations
  - a form of "relational assignment" statement so that partial results can be assigned a name and passed to subsequent operations

Relational Assignment

19

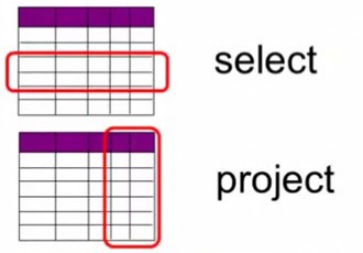
- A query is made up of a sequence of operations of the form:
- Format:
  - newRelation := UnaryOperation $_{parameter}$ (inputRelation)
    - Parameters can include column names or conditions

  Or (for a binary operator)
  - newRelation := $inputRelation_1$ Operator $_{parameter}$ $inputRelation_2$

Relation Operations

20

- Principal ***relation operations*** are:
  - **Select** - pick rows from a relation by some condition
  - **Project** - pick columns by name



  - **Join** - connect two *relations* (usually by a Foreign Key)

Set Operations

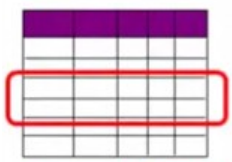
21

- ***Set operations*** include:
  - **union** - make a relation containing all the rows of two relations;
  - **intersection** - pick the tuples which are common to two relations;
  - **difference** - pick the tuples which are in one relation but not another;
  - **Cartesian Product** - pair off each of the tuples in one relation with those in another - creating a double sized row for each pair.

Selection (σ)

22

Extract the tuples (rows) of a relation (table)
which satisfy some condition on the values of their rows
and return these as a relation (table view)



select

Selection (σ)

Syntax

$\sigma_{\text{Condition}}$ **(RelationName)**

Locals := $\sigma_{\text{city = "Glasgow"}}$ (Employee)

- where the condition can contain:
  - Literals, e.g. “Glasgow”
  - column names, e.g. city
  - comparison operators (=, >, etc.)
  - boolean operators (and, not, or)

Selection

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

“*Who are the employees that live in Glasgow?*”

Locals := $\sigma_{\text{city = "Glasgow"}}$ (Employee)

Locals

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow

Selection

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

"Who are the employees that live in Glasgow?"

$$\text{Locals} := \sigma_{\text{city = "Glasgow"}} (\text{Employee})$$

Locals

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow

25

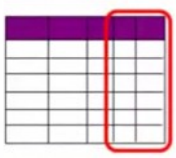
Projection (Π)

26

- Extracts some of the columns from a relation, given the names
  - E.g.

    GenderSalary := $\Pi_{(\text{gender, salary})}$ (Employee)

  - no attribute may occur more than once
  - duplicates will be removed



project

26

Projection

Employee

name	NI	Dept	gender	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

$$\text{GenderSalary} := \Pi_{(\text{Gender, salary})}(\text{Employee})$$

GenderSalary

gender	salary
F	34
M	23
M	17
F	28

27

Projection (Π)

28

- Projection and selection can be combined

$$\Pi_{(\text{house, street})}(\sigma_{\text{city = "Glasgow"}}(\text{Employee}))$$

28

Projection (Π)

29

- Projection and selection can be combined

$$\Pi_{(house, street, city)} (\sigma_{city = "Glasgow"} (Employee))$$

- does a selection

Projection (Π)

30

- Projection and selection can be combined

$$\Pi_{(house, street, city)} (\sigma_{city = "Glasgow"} (Employee))$$

- does a selection followed by a projection
- For this query, the following RA is equivalent

$$\sigma_{city = "Glasgow"} (\Pi_{(house, street, city)} (Employee))$$

- A DBMS can re-organise RA expressions for faster retrieval

Union (∪)

31

- Produces a relation which combines two relations into a new relation containing **all** of the tuples from each (removing duplicates)
- The two relations must be "**union compatible**" i.e. have the same number of attributes drawn from the same domain
- E.g.

  GlasgowOrStirling :=

  $\sigma_{\text{city = "Glasgow"}}$ (Employee) ∪ $\sigma_{\text{city = "Stirling"}}$ (Employee)

31

Union

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

GlasgowOrStirling : = $\sigma_{\text{city = "Glasgow"}}$ (Employee) ∪
$\sigma_{\text{city = "Stirling"}}$ (Employee)

GlasgowOrStirling

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling

32

Union

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

$$GlasgowOrStirling := \sigma_{city = "Glasgow"}(Employee) \cup \sigma_{city = "Stirling"}(Employee)$$

GlasgowOrStirling

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling

33

Union

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

$$GlasgowOrStirling := \sigma_{city = "Glasgow"}(Employee) \cup \sigma_{city = "Stirling"}(Employee)$$

GlasgowOrStirling

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling

34

Union Compatibility

35

- E.g. S1∪S2 is valid, but S1∪R1 is not

R1

sid	bid	day
22	101	101001
99	103	111201

S1

sid	sname	rating	age
11	Sue	7	26
22	Tim	8	26
33	Bob	9	28
55	Kim	10	28

S2

sid	sname	rating	age
10	Myleene	6	23
22	Tim	8	26
99	Julia	100	20
88	Gavin	100	21

Intersection (∩)

36

Similar to union but returns tuples that are in **both** relations

E.g.

```
FemalesInGlasgow :=
```

$\sigma_{city = "Glasgow"}$ (Employee) ∩ $\sigma_{gender = "F"}$ (Employee)

36

Intersection

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

```
FemalesInGlasgow :=
σ city = "Glasgow"(Employee) ∩
σ gender = "F" (Employee)
```

FemalesInGlasgow

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow

37

Intersection

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

```
FemalesInGlasgow :=
σ city = "Glasgow"(Employee) ∩
σ gender = "F" (Employee)
```

FemalesInGlasgow

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow

38

Intersection

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

```
FemalesInGlasgow :=
σ city = "Glasgow"(Employee) ∩
σ gender = "F" (Employee)
```

FemalesInGlasgow

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow

39

Difference (-)

40

- Similar to union but returns tuples that are in the first relation but not the second

E.g.

NonLocals := Employee – Locals

- Intersection and difference both require union compatibility
- Both use column names from the first relation

More Examples of Union compatible operations

Customers

FN	LN
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

Employees

FN	LN
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johns
Ramesh	Shah

Customers ∩ Employees

FN	LN
Susan	Yao
Ramesh	Shah

Employees-Customers

FN	LN
John	Smith
Ricardo	Browne
Francis	Johns

Customers ∪ Employees

FN	LN
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johns

42

Operations on any two **relations**

43

- Relations may have different schema
- Conditions indicate how to join the relations around **common attributes**

Cartesian Product of Relations (X)

44

- Cartesian Product A X B of two relations A and B, which have attributes A1 ... Am and B1 ... Bn……
  - ….is the relation with m + n attributes containing a row for every pair of rows, one from A and one from B
- Thus if A has **a** tuples and B has **b** tuples then the result has **a** x **b** tuples

Cartesian Product

45

Employee

name	NI#	dept
PDG	123	5
QC	456	5

Dependent

ENI#	name	sex
123	BG	F
123	DG	M
456	FC	F

Employee X Dependent

name	NI#	dept	ENI#	name	sex
PDG	123	5	123	BG	F
PDG	123	5	123	DG	M
PDG	123	5	456	FC	F
QC	456	5	123	GC	F
QC	456	5	123	DG	M
QC	456	5	456	FC	F

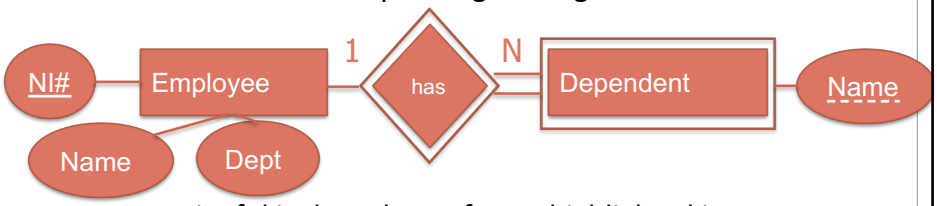
Degree: 3+3 = 6

Cardinality 2x3=6

Cartesian Product

46

- Cartesian Product - fairly meaningless & rarely used on its own
- What would be the corresponding ER diagram for the relations?



- More meaningful is the subset of rows highlighted in green
  - this could be created with the following selection:
    - $\sigma_{NI\# = ENI\#}$ (Employee x Dependent)
  - the selection essentially makes use of the **foreign key**
  - It tells us the Employees and their corresponding Dependents

46

A Join…

47

$\sigma_{NI\# = ENI\#}$ (Employee x Dependent)

- Cartesian Product followed by this kind of selection is called a **join** (⋈) because it joins together two relations

47

Join = Cartesian Product followed by Selection

48

Employee

name	NI#	dept
PDG	123	5
QC	456	5

Dependent

ENI#	name	sex
123	BG	F
123	DG	M
456	FC	F

Deps := $\sigma_{\text{NI\# = ENI\#}}$ (Employee x Dependent)

Deps

name	NI#	dept	ENI#	name	sex
PDG	123	5	123	BG	F
PDG	123	5	123	DG	M
QC	456	5	456	FC	F

48

Join operators

49

- Join operators takes two relations and makes a new one, based on the pairing of rows between those two relations
- There are a wide variety of join operators
- This particular form is called an **equi-join:**

$$\text{Relation}_1 \bowtie_{A = B} \text{Relation}_2$$

$$\Leftrightarrow$$

$$\sigma_{A = B} (\text{Relation}_1 \times \text{Relation}_2)$$

49

Join = Cartesian Product followed by Selection

50

Deps := $\sigma_{NI\# = ENI\#}$ (Employee x Dependent)

Deps

name	NI#	dept	ENI#	name	sex
PDG	123	5	123	BG	F
PDG	123	5	123	DG	M
QC	456	5	456	FC	F

$\Leftrightarrow$

Deps := Employee $\bowtie_{NI\# = ENI\#}$ Dependent

50

Natural Join ($\bowtie$)

51

- In its simplest form, the join of relations A and B pairs off the tuples of A and B so that **identically named** attributes from the relations have the same value
- We now have two columns holding the same value, so we eliminate the duplicated common attributes to form the **natural join or inner join**
- **Natural Join** is simply written as

  Relation1 $\bowtie$ Relation2

51

Example

52

R1

sid	bid	day
22	101	101001
99	103	111201

B1

bid	colour
101	red
102	blue
103	green

R1 ⋈ B1

sid	R1.bid	day	colour
22	101	101001	red
99	103	111201	green

Join Summary

53

- Cartesian product A X B : every row in A with every row in B
- Join ⋈: Cartesian product followed by a selection
  - Equi-join $\bowtie_{X=Y}$: Selection is on some attributes being equal
  - Natural join ⋈: Selection is on attributes with the same name being equal, and duplicate attributes in the resulting schema removed

Summary of Relational Algebra Operators

54

Three basic types of Relational Algebra operators:

1. **Applying to one relation**
   projection ∏ *(attributes)*, selection **σ** *(conditions)*
2. **Applying to two relations of identical structure**
   union ∪, intersection ∩, difference - *(no conditions)*
3. **Applying to two relations of different structure**
   (Cartesian) product × *(no conditions)*
   joins ⋈ *(conditions)*

Example

55

Employee and Department are two tables in a relational database.

- Employee (Name, NI-Number, Email, Phone-No, Works_In)
- Department (DeptName, Code, BuildingName)

The attribute **Works_In** is a foreign key in Employee relating to the primary key **Code** in Department.

Express in relational algebra:

(a) the names of all departments based in the Main building

DeptsInMain := $\pi_{\text{DeptName}}(\sigma_{\text{BuildingName} = \text{“Main”}}(\text{Departments}))$

Example

56

Employee and Department are two tables in a relational database.

- Employee (Name, NI-Number, Email, Phone-No, Works_In)
- Department (DeptName, Code, BuildingName)

Schema:
AllEmployeeDepts(Name NI-Number Email Phone-

yee relating to the

Schema:
AccountsEmployeesNames (Name)

(b) the nar the employees of the "Accounts" department.

AllEm eDepts := Employee $\bowtie_{Works\_In = Code}$ Department

Account Employees := $\sigma_{DeptName = \text{"Accounts"}}$ (AllEmployeeDepts)

AccountsEmployeesNames := π_{Name} (AccountsEmployees)

1. **How** much of this RA material (as a percentage) do you understand?
2. (If the answer to Q1 is less than 100%), **what** are you going to do to raise this percentage?
3. **When** (exactly) are you going to do it?

Where to go for more info...

59

- Rosen, Discrete Mathematics and Its Applications
  - sets - sections 1.4 & 1.5
  - relations - sections 6.1 & 6.2
  - http://www.mhhe.com/math/advmath/rosen/
- OR
- Garcia Molina, Chapter 5, Section 1 (pgs 199-205)
- OR
- https://www.tutorialspoint.com/dbms/relational_algebra.htm
- OR
- more worked examples in the handouts at end of these slides…!
- OR
- Relational Algebra quiz on Moodle

On Moodle…


IM Lectures


IM Labs and Tutorials


Relational Algebra Quiz

The solutions will be placed online once enough students have viewed the quiz!

Some helpful external resources for Information Management


what is a database?


What is a relational database?


what is an ER diagram?


drawing ER diagrams


tutorial on relational algebra

60

Relational Algebra Queries – Worked Examples

- Consider the following relations:

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

ResDept

Dnum	Dname	manager
5	R&D	456

Relational Algebra Queries: Example 1

- Give the name and addresses of all employees who work for the R&D department
- ResDept := $\sigma_{\text{Dname = "R\&D"}}$(Department)
  - this should return just one tuple
- ResDeptEmps := ResDept $\bowtie_{\text{Dnumber = Dept\#}}$ Employee
  - this picks out the employees in that department.
- Result := $\Pi_{\text{(Name, Address)}}$ ResDeptEmps
- Notes:
  - the order could be changed – e.g. to make it faster.
  - the sub-steps could be merged into one:

    $\Pi_{\text{(Name, Address)}}$ (($\sigma_{\text{Dname = "R\&D"}}$ (Department) $\bowtie_{\text{Dnumber = Dept\#}}$ Employee
  - but this is not so readable

Example 1: Breakdown (1/3)

63

Give name and addresses of employees who work for the R&D dept

Department

Dnum	Dname	manager
5	R&D	456
6	Production	111
7	Admin	345

$$ResDept := \sigma_{Dname = "R\&D"} (Department)$$

ResDept

Dnum	Dname	manager
5	R&D	456

Example 1: Breakdown (2/3)

64

Employee

name	NI	Dept	sex	age	salary	house	street	city
Smith	123	7	F	52	34	5	High St	Glasgow
Jones	456	5	M	35	23	21	Lo St	Glasgow
McSomething	789	6	M	24	17	75	My St	Stirling
Black	541	5	F	44	28	77	Ur St	Edinburgh

ResDept

Dnum	Dname	manager
5	R&D	456

$$ResDeptEmps := ResDept \bowtie_{Dnum = Dept} Employee$$

ResDeptEmps

Dnum	Dname	Manager	name	NI	sex	age	salary	house	street	city
5	R&D	456	Jones	456	M	35	23	21	Lo St	Glasgow
5	R&D	456	Black	541	F	44	28	77	Ur St	Edinburgh

Example 1: Breakdown (3/3)

ResDeptEmps

Dnum	Dname	Manager	name	NI	Dept	sex	age	salary	house	street	city
5	R&D	456	Jones	456	5	M	35	23	21	Lo St	Glasgow
5	R&D	456	Black	541	5	F	44	28	77	Ur St	Edinburgh

$\Pi_{(Name,\ House,\ Street,\ City)}$ ResDeptEmps

Result

name	house	street	city
Jones	21	Lo St	Glasgow
Black	77	Ur St	Edinburgh

Example 2

- List the name, controlling department name, the department manager's name, address and date of birth for every project in Stafford
- StaffordProjs := $\sigma_{PLocation = "Stafford"}$ (Project)
  - *it is common to start by restricting to an area of interest*
- StaffordProjDepts:= StaffordProjs $\bowtie_{Dnum=Department}$ Department
  - *this brings together the department information*
- StaffordProjDeptMngrs:= StaffordProjDepts $\bowtie_{MgrNI\#=NI\#}$ Employee
  - *this brings in the manager information*
- Result := $\Pi_{(Pname,\ Dname,\ Name,\ Address,\ DateOfBirth)}$ StaffordProjDeptMngrs