

60–100 Lecture 9

Databases and Relational Algebra

60-100 INDIVIDUAL ASSIGNMENT #4 (Fall 2009) – Worth twice the marks of the other assignments. (Hand-in a typed copy of your program on Tuesday 20th October. Demonstrate your program at the group lab on Thursday evening 22nd October.)
 You must end your answer sheet with "I declare that this is my own work", followed by your signature
 A) Create a file called assignment4.m and add the following definitions (OR better, copy the file from the following URL: http://cs.uwindsor.ca/~richard/60_100/indiv_assignments/indiv_4.m)

```
male = ["david", "john", "jeff", "paul", "simon", "albert", "bill"]
female = ["angela", "sue", "mary", "beth"]
parent_of = [{"david", "paul"}, {"david", "sue"}, {"jeff", "john"}, {"jeff", "angela"}, {"simon", "sue"}, {"simon", "mary"}, {"john", "beth"}, {"angela", "beth"}]
married = [{"sue", "john"}, {"john", "sue"}]
```

```
project_first_of_2 r = mkset [(x) | (x,y) <- r]
project_second_of_2 r = mkset [(y) | (x,y) <- r]
project_first_and_third_of_3 r = mkset [(x, z) | (x, y, z) <- r]
select_first_of_2 key r = [(x,y) | (x,y) <- r; x = key]
```

```
join_second_of_2_with_first_of_two r s
= [(x,y,z) | (x,y) <- r; (a,z) <- s; a = y]
union s t = s ++ (t -- s)
intersection s t = s -- (s -- t)
```

```
make_transitive [] = []
make_transitive (x,y:ps)
= new_direct_paths ++
  [(a,c) | (a,b) <- mtps; (d,c) <- new_direct_paths; b = d] ++ mtps
where
  mtps = make_transitive ps
  new_direct_paths = (x,y):[(x,b) | (n,b) <- mtps; n = y]
```

B) Now define the following relations in terms of (using) the relations male, female, and parent_of and the relational operators project_first_of_2 etc. (plus additional operators which you can define in the same way as the examples above)

```
grandparent (hint - use project_first_of_2 and parent)
ancestor_of (hint - use make_transitive and parent)
brother_or_sister_of (hint - join parent with itself then project)
sister_of
aunt_or_uncle_of
aunt_or_uncle_of_mary (hint use the aunt_or_uncle_of relation and the
select_second_of_2 operator)
niece_of
cousin_of
daughter_in_law_of
grandparents_of_bill
```

Note that you can define your own new relational algebra operators. Also, there is a built-in program called mkset which takes a list as input and which removes duplicate entries. Another useful program is the following:
 remove_reflexives r = [(x, y) | (x, y) <- r; x == y]

Relational Databases

- Most modern database systems store data as relations
- For example:
 - `employ_rel:: { (company, employee, age)`
 - `company_rel:: { (company, location) }`
- Relations often depicted as tables. eg:

Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28

Comp	location
IBM	Toronto
Nortel	Ottawa
Mitel	Toronto

Advantages of storing data as relations

- Relations correspond to simple real relationships
 - Relations can be designed to reduce redundancy.
- For example, `employ_rel` and `company_rel` are not redundant, whereas a file of records of type `(employee, company, location)` contains redundant data.

emp	comp	location
ahmed	IBM	Toronto
anis	Nortel	Ottawa
bob	IBM	Toronto
feng	Mitel	Toronto

- Low redundancy improves consistency.
- Relational structure can be easily changed without changing programs that access the relations.
- Relations are easy to manipulate using relational algebra

Relational Algebra

- An algebra is a set S , called a carrier, of type $\{*\}$, together with a collection of operators such that each operator is of type $* - > * - > * \dots - > *$ and takes operands from the set and returns a value that is a member of the set.
- Arithmetic is an algebra, where the set is the set of numbers and the operators include $+$, $-$, etc.
- In relational algebra, the set is the set of relations, and the operators are operators which take relations as argument and return relations as result.

Operators of relational algebra

- The set operators: union, intersection and difference are operators in relational algebra.
- The operators: select, project and join are operators in relational algebra.
- Remember that the relational algebraic operators take relations as input and return relations as result.

Union, intersection and difference

These operators take two relations which must be of the same type and return a relation of the same type. For example, union of two relations:

Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32



Comp	Emp	Age
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28



Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28

Select

- Select allows you to create a new relation by selecting rows from an existing relation which meet a given criteria. For example:

	employ_rel	
Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28

23

select_third_from_3_employ_rel 23 =>

	new_rel	
Comp	Emp	Age
IBM	ahmed	23
IBM	bob	23

Project

- Project creates a new relation by “projecting” some of the columns from an existing relation. Project removes any duplicate tuples. For example:

	employ_rel	
Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28


`project_third_of_3 employ_rel =>`

new_rel
Age
23
32
28


Join

Join allows you to create a “fatter” relation by joining together two relations which have common values in given columns. For example:

	employ_rel	
Comp	Emp	Age
IBM	ahmed	23
Nortel	anis	32
IBM	bob	23
Mitel	Feng	28



company_rel	
Comp	Location
IBM	Toronto
Nortel	Ottawa
Mitel	Toronto



```
join_first_of_3_with_first_of_2
      employ_rel
      company_rel
```

=>

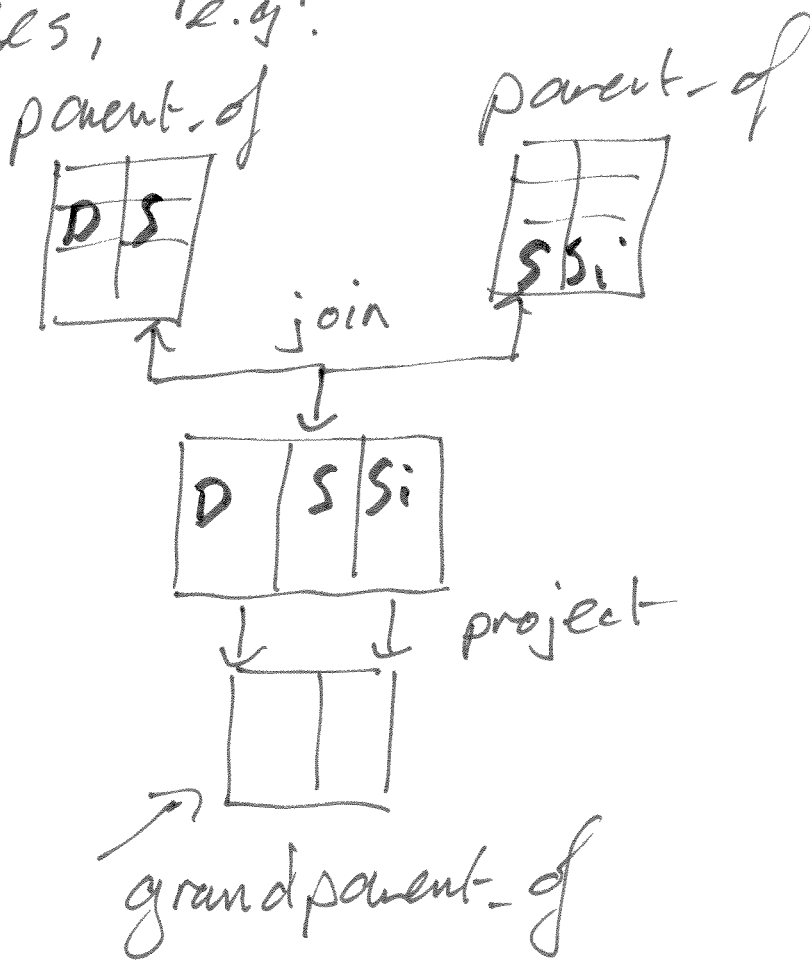
employ_rel			
Comp	Emp	Age	Location
IBM	ahmed	23	Toronto
Nortel	anis	32	Ottawa
IBM	bob	23	Toronto
Mitel	Feng	28	Toronto

Notes for assign. #4 - I

Begin by drawing all lists of data as tables, e.g.

male	female	parent_of
D		D P
J		D S
J ₀		J J ₀
P		J A
S _i		etc.
AI		
B		

Next, think of operations to create new tables, e.g.



Notes for Assign. #4

II

parent of

parent of

D	P
D	S
J	Jo
J	A
S	Si
S	m
J	Si
J	m
m	B
A	Be

D	P
D	S
J	Jo
J	A
S	Si
S	m
J	Si
J	m
m	B
A	Be

join

join - second of 2 with first of 2

D	S	Si
D	S	m
J	A	Be
S	m	B
J	m	B

project - first and third of - 3

D	Si
D	m
J	Be
S	B
J	B

This is the grandparent relation

Think of the exact joins, projects and selects.

Notes for assign #4 - III

Next convert diagram to a program, e.g.

grandparent-of

= project-first-and-third-of-3

(join-~~first~~second-of-2-with-first-of-2

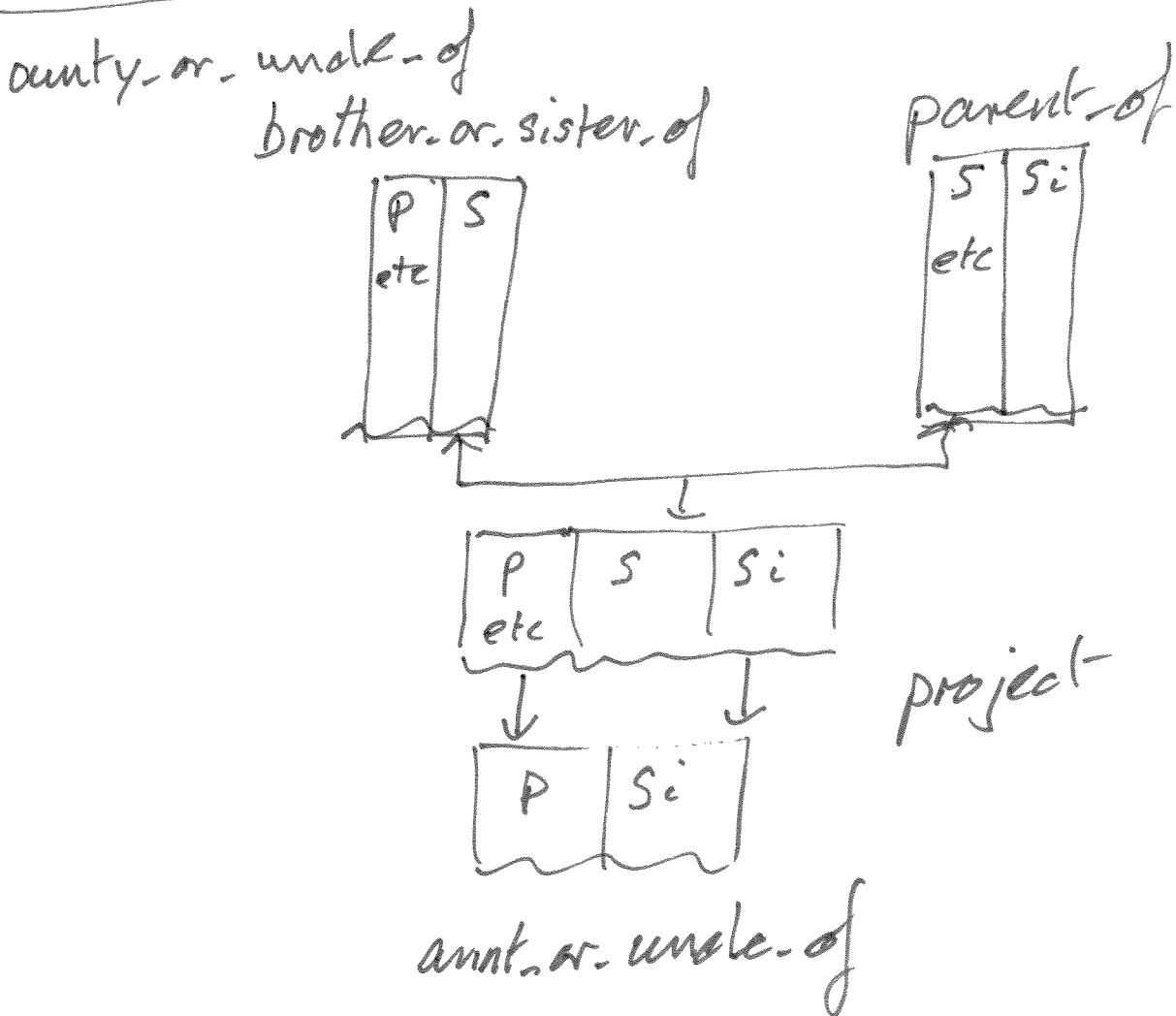
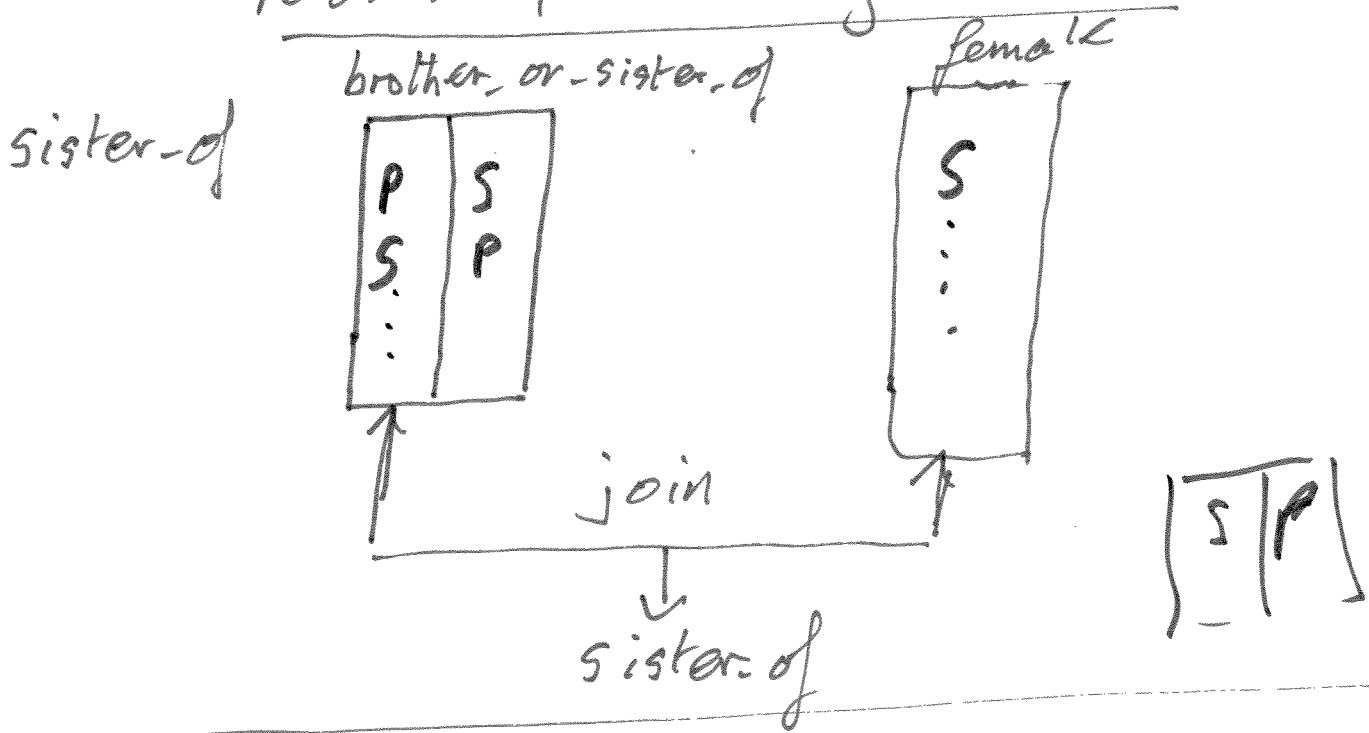
parent-of parent-of)

You also need to be able to calculate the result of a relational algebra expression.

See next page

$$sq(n-3)$$

Notes for assignment #4 - ~~IV~~ V




```
project_first_of_2 r
= mkset [(x) | (x, y) <- r]
```

```
project_second_of_2 r
= mkset [(y) | (x, y) <- r]
```

```
project_first_and_third_of_3 r
= mkset [(x, z) | (x, y, z) <- r]
```

```
project_second_and_third_of_3 r
= mkset [(y, z) | (x, y, z) <- r]
```

```
project_second_and_fourth_of_5 r
= mkset [(b, d) | (a, b, c, d, e) <- r]
```

select second of 2 ~~key~~ \mathcal{r}
= [(x, y) | (x, y) \leftarrow \mathcal{r} ; y = key]

select first of 2 ~~key~~ \mathcal{r}
= [(x, y) | (x, y) \leftarrow \mathcal{r} ; x = key]

select second of 3 ~~key~~ \mathcal{r}
= [(a, b, c) | (a, b, c) \leftarrow \mathcal{r} ; b = key]

```

join_first_of_2_with_first_of_2 r s
= [(x,y,z) | (x, y) <- r;
    (a, z) <- s;
    x = a]

```

```

join_first_of_1_with_first_of_2 r s
= [(x,y) | (x) <- r;
    a (a,y) <- s; x = a]

```

```

join_second_of_2_with_first_of_2 r s
= [(x,y,z) | (x, y) <- r;
    (a, z) <- s;
    y = a]

```

```
remove_reflexives r
= [(a, b) | (a, b) <- r; a ~ = b]
swap
= map switch
  where switch (a, b) = (b, a)
union s t = s ++ (t --s)
intersection s t = s -- (s -- t)
```

```

make_transitive [] = []
make_transitive ((x, y):ps)
  = new_direct_paths ++
    [(a, c) | (a, b) <- mtps;
           (d, c) <- new_direct_paths;
           b = d] ++ mtps
  where
mtps = make_transitive ps
new_direct_paths
  = (x, y) : [(x, b) | (n, b) <- mtps;
                 n = y]

```