Relations

Part 4

1

Cross Products

Databases use this…

2

Cross Products

- Sets can be multiplied, which will result in a set of tuples
- Well, a set of ordered pairs, to be more specific
- Cross products are important in databases and counting *(to name a few)*

3

Products

- A *cross product* is a set of ordered pairs
- Note: Unlike multiplication, the order of the operands is important

$$A \times B = \{ (x, y) \mid x \in A \text{ and } y \in B \}$$

4

Example Product

```
Given:
  A = {small, big}
  B = {cat, dog}

A × B = { (small, cat), (small, dog),
          (big, cat), (big, dog) }
```


5

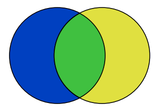
Binary Relations

How Stuff Compares to Stuff

6

Relations

- A *binary relation* is a stated fact between on two objects
- A "fact" is called a *predicate*
- Evaluates to true or false
- These are the foundation of most programming tasks



7

Example Relations

- "x is bigger than y"
- "x lives less than 50 miles from y"
- "x ≤ y"
- "x and y are siblings"
- "x has a y"

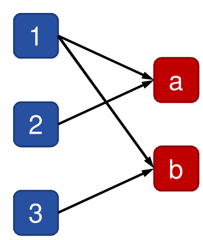
8

Relations

- A *binary relation* from A to B is a subset of the cross-product A × B
- A relation from A to B is a set of ordered pairs (a, b) where a ∈ A and b ∈ B
- We can use the shorthand notation of: a R b to denote that (a, b) ∈ R

9

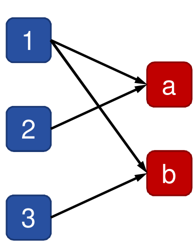
Relationship Chart



	a	b
1	✓	✓
2	✓	
3		✓

10

Relationship Chart




```
{ (1, a), (1, b),
  (2, a), (3, b) }
```


11

Example: Capitols

- A is a set of all cities in the World
- B is a set of all states in the World
- The relation *a* R *b* specifies that *a* is the capitol of *b*

12

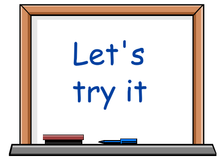
Example: Capitol Members

- (London, Britain)
- (Sacramento, California)
- (Madrid, Spain)
- (Tokyo, Japan)
- (New Delhi, India)
- (Albany, New York)

13

Let's Draw One!

- Let's draw a relationship graph
- It will be between students and shows/movies they enjoy



14

Relation Domain

- The *domain* of a relation is a set of all the first elements of each tuple
- So, it is the elements that the make up the left-hand side of *a* R *b*

$$\{ a \mid (a, b) \in R \text{ for some } b \}$$

15

Relation Range

- The *range* of a relation is a set of all second elements from each tuple
- So, it is the elements that the make up the right-hand side of *a* R *b*

$$\{ b \mid (a, b) \in R \text{ for some } a \}$$

16

Example

```
R = { (1,1), (1,2), (1,3),
      (1,4), (2,4), (4,4) }

Domain of R = { 1, 2, 4 }

Range of R = { 1, 2, 3, 4 }
```


17

Inverse Relation

- The *inverse* of a relation swaps first and last element for each tuple
- The number of elements are the same, but the range and domain are reversed

$$R^{-1} = \{ (b, a) \mid (a, b) \in R \}$$

18

Inverse Example

```
R = { (1,1), (1,2), (1,3),
      (1,4), (2,4), (4,4) }

R⁻¹ = { (1,1), (2,1), (3,1),
        (4,1), (4,2), (4,4) }
```


19

Relations can be infinitely large

- On a finite set, relations are quite simple…
- For a set with n elements, the maximum number of relations is simply $n \times n = n^2$
- However, many relations are defined over an infinite set – e.g. all integers, reals, etc…

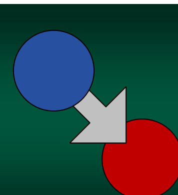
20

Representing Relations

- We can represent a relation using set notation
- However, some are required to be denoted using set builder notation

```
R1 = { (a, b) | a is bigger than b }
R2 = { (a, b) | a ≤ b }
```


21



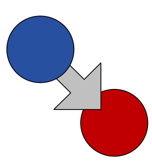
Types of Relations

How a Set Sees Itself

22

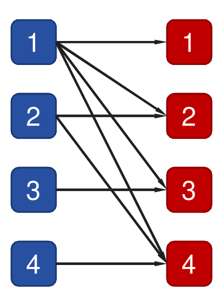
"Relation On"

- Some relations of a set A are upon itself
- In other words, each object in the related to the same "type" of object
- This is called a *relation on* A
- …and it is a important to examine its properties



23

Example Relationship Chart



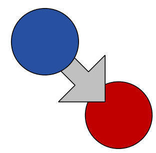
	1	2	3	4
1	✓	✓	✓	✓
2		✓		✓
3			✓	
4				✓

24

Example Relation Chart

- The previous chart represents when *a* divides *b*
- In other words, *a* times some integer equals the value *b*
- So, R = { (1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4) }

Reflexive Relations



- A *reflexive* relationship means that there is *a*R*a* for every *a*
- Basically, everything has to be related to itself
- *Every element, in the domain, must be related to itself*

To Determine Reflexive…

- Look for some *a* ∈ A where there isn't a *a*R*a*
- If found, not reflexive
- Otherwise reflexive

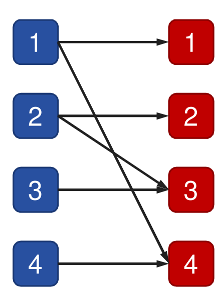
Reflexive Example

```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,4), (2,2),
      (2,3), (3,3), (4,4) }
```


Reflexive Example

```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,4), (2,2),
      (2,3), (3,3), (4,4) }
```


Reflexive Example



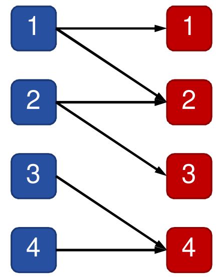
	1	2	3	4
1	✓			✓
2		✓	✓	
3			✓	
4				✓

Nonreflexive Example

```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,4), (2,2),
      (2,3), (3,1), (4,4) }
```


31

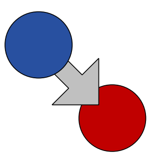
Nonreflexive Example



	1	2	3	4
1	✓	✓		
2		✓	✓	
3				✓
4				✓

32

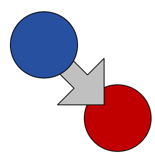
Symmetric Relations



- A *symmetric* relationship means that for every *a*R*b* there is a *b*R*a*
- So, if (a, b) exists in the relation, so must (b, a)

33

Symmetric Relations

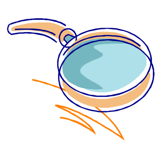


- Note: Unlike the definition for reflexive, not every element in the domain needs to exist
- If a relation is not symmetric, it is *antisymmetric*

34

To Determine Symmetric…

- Look for an *a* and *b* where there is a *a*R*b* but no *b*R*a*
- If found, nonsymmetric
- Otherwise symmetric



35

Symmetric Example

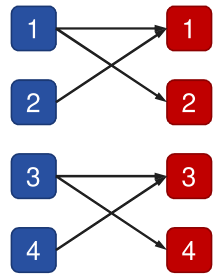
```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,2), (2,1)
      (2,4), (3,3), (4,2) }
```


36

Symmetric Example

```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,2), (2,1)
      (2,4), (3,3), (4,2) }
```


Symmetric Example

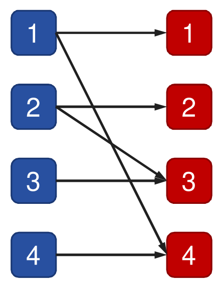


	1	2	3	4
1	✓	✓		
2	✓			✓
3			✓	
4		✓		

Nonsymmetric Example

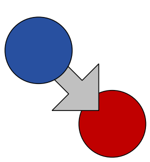
```
Relation on set {1, 2, 3, 4}

R = { (1,1), (1,4), (2,2),
      (2,3), (3,3), (4,4) }
```


Nonsymmetric Example



	1	2	3	4
1	✓			✓
2		✓	✓	
3			✓	
4				✓

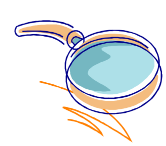
Transitive Relations



- A *transitive* relationship means that for every *a*R*b* and *b*R*c* that also *a*R*c*
- So, if (a, b) and (b, c) exists in the relation, so must (a, c)

To Determine Transitive…

- Look for an *a*, *b*, *c* where there is a *a*R*b* and *b*R*c* but no *a*R*c*
- If found, non transitive
- Otherwise transitive



Transitive Example

```
Relation on set {1, 2, 3, 4}

R = { (2,1), (3,1), (3,2),
      (4,1), (4,2), (4,3) }
```


43

Transitive Example

```
Relation on set {1, 2, 3, 4}

R = { (2,1), (3,1), (3,2),
      (4,1), (4,2), (4,3) }
```

Starting with (4,3)

44

Transitive Example

```
Relation on set {1, 2, 3, 4}

R = { (2,1), (3,1), (3,2),
      (4,1), (4,2), (4,3) }
```

Starting with (3,2)

45

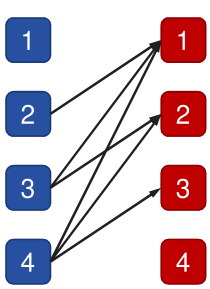
Transitive Example

```
Relation on set {1, 2, 3, 4}

R = { (2,1), (3,1), (3,2),
      (4,1), (4,2), (4,3) }
```

Starting (again) with (4,3)

46

Transitive Example

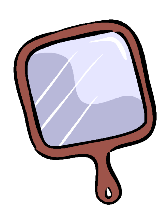


	1	2	3	4
1				
2	✓			
3	✓	✓		
4	✓	✓	✓	

47

Equivalence Relations

- If a relation has all three properties:
  - reflexive
  - symmetric
  - transitive
- Then, and only then, it is an *equivalence*



48

Example Table

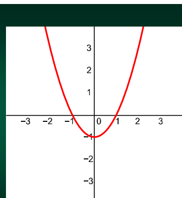
Relation	Reflexive	Symmetric	Transitive	Equivalent
$\leq$				
$\subset$				
Perpendicular lines				
Parallel lines				

49

Example Table

Relation	Reflexive	Symmetric	Transitive	Equivalent
$\leq$	✓	✗	✓	✗
$\subset$	✗	✗	✓	✗
Perpendicular lines	✗	✓	✗	✗
Parallel lines	✓	✓	✓	✓

50



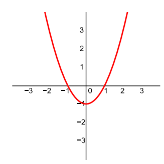
Functions

Math Friendly Relations

51

Functions

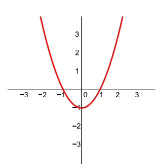
- We have all seen functions – which take inputs and produce output
- Example: $f(x) = x^2$
  - $f(1) = 1$
  - $f(2) = 4$
  - $f(3) = 9$ ...



52

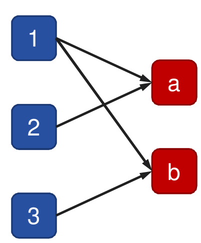
Functions

- Sets give a way to document "types" in mathematical functions
- A *function* from set X to set Y is a mapping from each element in set X to elements in set Y



53

Relationship Review




```
{ (1, a), (1, b),
  (2, a), (3, b) }
```


54

Function Attributes

- Function Rules:
  - must be defined for *every element in domain*
  - each value in domain *maps to one element*
- Notice that a function defines a set of ordered pairs: e.g. (1,1) (2,4) (3,9) ...
- We can therefore think of a function as a special kind of relation.

55

Relations vs. Functions

- Each domain element, in a relation, can specify *many* relationships
- While, each element in a function domain only specifies *one* relationship
- So….
  - every function is a relation
  - but not every relation is a function

56

Definition of a Function

Let f be a relation from A → B

f is a function if and only if:

each a ∈ A appears exactly once in an ordered pair (a, b) ∈ f for some b

57

Function Signature

- We will restrict a functions inputs and outputs by giving a "signature" for it
- f is the function name

f: N → N

58

Function Signature

- The first N is the function domain
- The second N is the function range (codomain)

f: N → N

59

Domain and Range Definitions

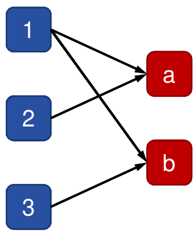
- The domain and range of f is defined exactly as we saw for relations
- Which is not surprising given what a function really is

domain(f) = { x | (x, y) ∈ f for some y }

range(f) = { y | (x, y) ∈ f for some x }

60

Let's Look At This Again….




```
{ (1, a), (1, b),
  (2, a), (3, b) }
```

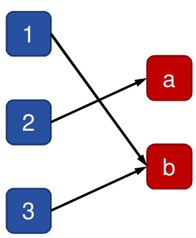

61

Relations vs. Functions

- Not that in the example (with 1,2,3 and a, b) that some elements in A had multiple values in B
- In a function, each member in A maps to exactly one value in B
- So, that relation was not a function!

62

Function Example




```
{ (1, b), (2, a),
  (3, b) }
```


63

Examples

- For the following examples, let each example be defined as a relation from A to B
- Domain and range (codomain) are defined as:

```
A = {1, 2, 3}
B = {x, y, z}
```


64

Is This a Function?

```
Let f = { (1,x), (2,y) }
```

No, the domain value 3 is missing as a first ordered-pair element

65

Is This a Function?

```
Let g = { (1,x), (2,y),
          (3,z), (1,y) }
```

No, the domain element 1 is listed twice.

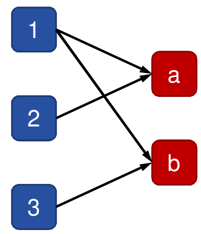
66

Is This a Function?

```
Let h = { (1,x), (2,y), (3,x) }

Yes, each domain element of A is
the first element once.

(There is no restriction on
 the second element)
```


67

Is This a Function?

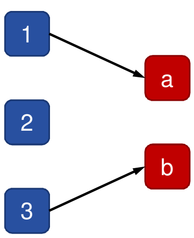


No!

(1,a) and (1,b)
both exist

68

Is This a Function?

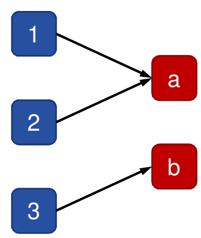


No!

The domain element
2 is not mapped to
a or b

69

Is This a Function?



Yes!

70

Function Definitions

The Mapping of Sets

71

Function Definitions

- Functions are usually defined using a formula
- You should be able to tell that these match a Java method definition – header and body

```
f: Z → Z
f(x) = x * x
```


72

Function Definitions

- First part tells us that f maps every integer to an integer
- Second part tells us f(x) and x^2 are the same thing

```
f: Z → Z
f(x) = x * x
```


73

Example

- In the following, is *g* a valid function?
- R is a set of reals
- √ is the square root function

```
g: R → R
g(x) = √x
```


74

Example

- **No.**
- Not every element of R maps to something in R
- For example, g(-1) $\notin$ R

```
g: R → R
g(x) = √x
```


75

{ } Manipulating Relations

How a Set Sees Itself

76

Manipulating Relations

- Because relations are representable as sets, we can use set notation to define them
- We can also use set notation to manipulate them

{ }

77

Example

```
A = { (1,1), (2,2), (3,3) }
B = { (1,1), (2,4), (3,9) }
```


78

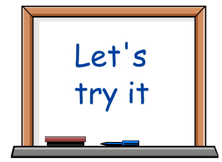
Example

```
A ∪ B =   { (1,1), (2,2), (2,4),
            (3,3), (3,9) }

A ∩ B =   { (1,1) }

A \ B =   { (2,2), (3,3) }

B \ A =   { (2,4), (3,9) }
```


79

Let's Examine Some…

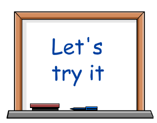
- Let's use students to create two relations
- Classes you plan to take
- Classes you might/did enjoy



80

Let's Examine Some…

- Let's examine two relations over a simple set {1, 2, 3} using set operators
- We can check if it is:
  - reflexive
  - symmetric
  - transitive



81

Let's Examine Some…

```
A = {1,2,3}: R, S relations.

R = { (1,1), (1,2), (2,2),
      (2,3), (3,1), (3,3)}

S = { (1,1), (1,2), (1,3),
      (2,1), (2,3), (3,2) }
```


82

Closures

Making a relation "complete"

83

Closure

- *Closure* of relation R is the smallest set (when unioned) gives R the desired property
- So, the closure of R is R ∪ C, where C is the smallest set giving R ∪ C the desired property

84

Some Examples

- For the following examples, the relation is over the set {1, 2, 3, 4}
- The slides will show how to make the closures for reflexive, symmetric, and (the hard one) transitive

85

Example Reflexive Closure

```
R = { (1,2), (2,3), (3,4) }

C = { (1,1), (2,2), (3,3), (4,4) }
```

Missing (1,1) (2,2), (3,3) and (4,4)

86

Example Reflexive Closure

```
R U C = { (1,2), (2,3), (3,4),
          (1,1), (2,2), (3,3),
          (4,4) }
```


87

Example Symmetric Closure

```
R = { (1,2), (2,3), (3,4) }

C = { (2,1), (3,2), (4,3) }
```


88

Example Symmetric Closure

```
R U C = { (1,2), (2,3), (3,4),
          (2,1), (3,2), (4,3) }
```


89

Example Transitive Closure (1 of 3)

```
R = { (1,2), (2,3), (3,4) }

C = { (1,3)               }
```

Added due to (1,2) and (2,3)

90

Example Transitive Closure (2 of 3)

```
R = { (1,2), (2,3), (3,4) }

C = { (1,3), (2,4)        }
```

Added due to (2,3) and (3,4)

Example Transitive Closure (3 of 3)

```
R = { (1,2), (2,3), (3,4) }

C = { (1,3), (2,4), (1,4) }
```

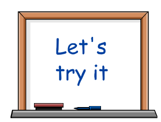
Had to add after we added (2,4) since R contains (1,2)

Example Transitive Closure

```
R ∪ C = { (1,2), (2,3), (3,4),
          (1,3), (2,4), (1,4) }
```


Let's Try Some

- Set is over {1, 2, 3, 4, 5}
- R = { (1,2), (2,3), (3,5), (4,1) }
- What are the closures for this relation?



Composition

The Inception of Functions

Composition

- *Composition* of two functions means the output of one function is used as the input as another
- This is very common in programming – you use the result of one expression as input to another

Notation

- Notation for composition is straight forward – it simply consists of a empty circle operator
- Sometimes the (x) is put in front of the first function, but this is not always the case

$$f \circ g(x) \equiv f(g(x))$$

Composition Example

$$f(x) = x + 4$$
$$g(x) = x^2$$

$$f \circ g(z) = f(g(z)) = f(z^2) = z^2 + 4$$

Composition Example 2

$$f(x) = x + 4$$
$$g(x) = x^2$$

$$g \circ f(z) = g(f(z)) = g(z + 4) = z^2 + 8z + 16$$

Composite Example

R = { (1,2), (3,1), (5,3) }
S = { (2,3), (2,6), (3,9) }

R ∘ S = { (1,3), (1,6), (5,9) }