Creating Aggregate Objects from Normalised Data We'll get some nulls. (An additional note)

David J. Greaves

(with grateful thanks to Timothy G. Griffin)

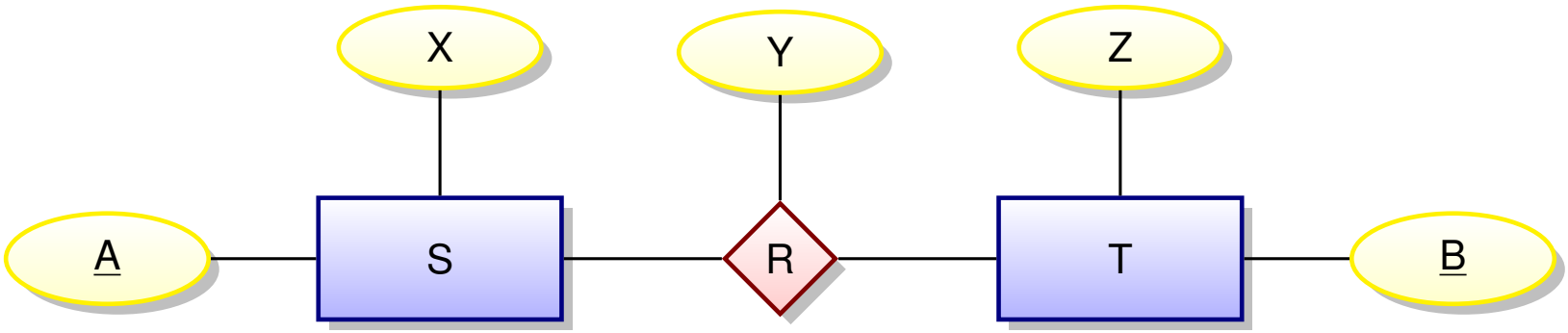
Computer Laboratory
University of Cambridge, UK

Michaelmas Term, 2022-23

TGG's Thought Experiment

- Let's do a **thought experiment**.
- In the next few slides imagine that we intend to use a relational database to store read-optimised tables generated from a set of write-optimised tables (that is, having little redundancy).
- We will encounter some problems that can be solved by representing our data as aggregate/semi-structured objects.

Start with a simple relationship ...



S is related to T by relation R:

S

A	X
a1	x1
a2	x2
a3	x3

R

A	B	Y
a1	b1	y1
a1	b2	y2
a1	b3	y3
a2	b1	y4
a2	b3	y5

T

B	Z
b1	z1
b2	z2
b3	z3
b4	z4

Imagine that our read-oriented applications can't afford to do joins!

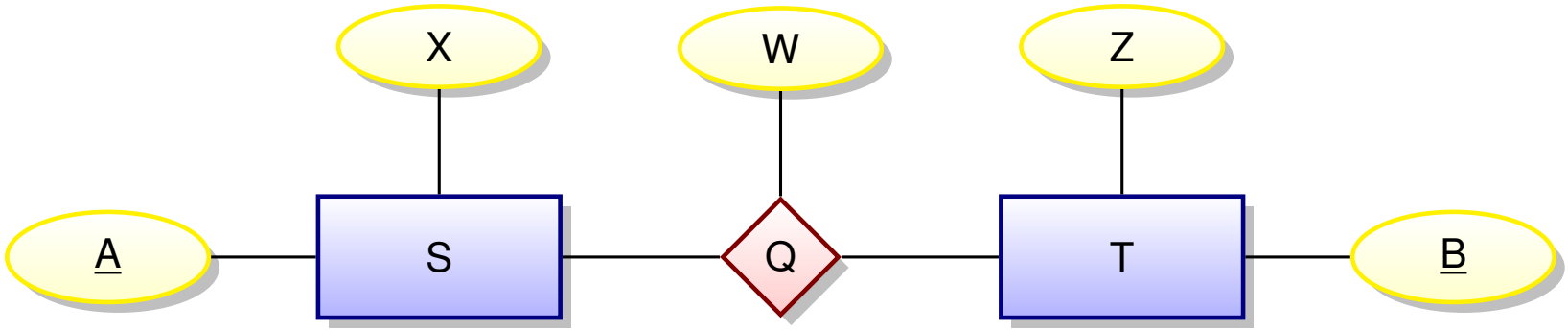
Implement the relationship as one big table?

BigTableOne: An outer join of *S*, *R*, and *T*

A	X	B	Z	Y
a1	x1	b1	z1	y1
a1	x1	b2	z2	y2
a1	x1	b3	z3	y3
a2	x2	b1	z1	y4
a2	x2	b3	z3	y5
a3	x3			
		b4	z4	

- Since we don't update this data, there will be no problems associated with redundancy.
- Notice the null values starting to pop up.

However, there may be many further relationships ...



S is further related to T by relation Q:

S

A	X
a1	x1
a2	x2
a3	x3

Q

A	B	W
a1	b4	w1
a3	b2	w2
a3	b3	w3

T

B	Z
b1	z1
b2	z2
b3	z3
b4	z4

Implement with another big table?

BigTableTwo: An outer join of *S*, *Q*, and *T*

A	X	B	Z	W
a1	x1	b4	z4	w1
a3	x3	b2	z2	w2
a3	x3	b3	z3	w3
a2	x2			
		b1	z1	

Having two tables makes reading a bit more difficult!

Combine into one big table?

BigTable: Derived from *S*, *R*, *Q*, and *T*

A	X	B	Z	Y	W
a1	x1	b1	z1	y1	
a1	x1	b2	z2	y2	
a1	x1	b3	z3	y3	
a1	x1	b4	z4		w1
a2	x2	b1	z1	y4	
a2	x2	b3	z3	y5	
a3	x3	b2	z2		w2
a3	x3	b3	z3		w3

Note that our big tables use a **composite key**.

Problems with BigTable

- We could store BigTable and speed up some queries.
- Note: if we'd also stored further attributes of A in S, that data would be replicated $|B|$ times.
- But suppose that our applications typically access data using either S's key or T's key.
- Creating indices on the A and B columns could speed things up, but our applications may still be forced to gather information from many rows in order to collect all information related to a given key of S or a given key of T.
- It would be better to access all data associated with a given key of S or a given key of T using only **a single database lookup**.

Potential Solution

- Represent the data using aggregate forms.
- This example was highly structured, but the aggregate forms also support semi-structured objects.