University of Toronto
CSC**343**

Finding all keys

Example

Consider a relation schema R with attributes $ABCDEF$ with functional dependencies S:

$$S = \{AB \to C, \quad BF \to E, \quad C \to BE, \quad AC \to F\}$$

Suppose we need to find all keys for this relation. What does it mean to be a key? Now that we know about functional dependencies, we know that if a set of attributes, X is a key for a relation, it must functionally determine all attributes of the relation, that is, it must be a superkey. And of course it must be minimal in the sense that no subset of X may have that property.

How to do it

How do we find all the keys? A brute force approach would be to consider every possible subset of the attributes, starting from the singleton subsets. For each, take the closure; if it yields every attribute, we have found a key. We remove all supersets from consideration, because they will clearly be superkeys but they will not be minimal. Then we move on to the subsets consisting of two attributes, and so on.

This process is painful, because there are a lot of subsets. If we have n attributes, there are 2^n possible subsets. In this case, that's 64 — ouch! We know from the outside that the empty set cannot be a key; and if there are any non-trivial dependencies, the set of all attributes cannot be a key. But that only spares us 2 of the 2^n closures we would need to make.

Does it have to be that hard?

If we're smart about it, we can save ourselves much more work. If you are reading this before working on it in class, take a couple of minutes to try to discover the potential for big speed-ups. Start looking for keys and see what you can notice about the attributes. Are there any that you know will never be in any key? Any that must be in every key? When you've played with it a bit, move on to the next page.

Major speed-ups

You probably noticed that D is not anywhere in the FDs. That means it has to be in every key — there is no other way to get it. In fact, even if an attribute appears in an FD, if it never appears on a RHS, it will have to be in every key. A is an example of this.

If, on the other hand, an attribute appears only on the RHS of FDs, never the left, it is of no help to us in computing closures. It cannot be in any key, because you could always remove it and still get the same closure.

To summarize, here is what we've learned:

	Appears on		
Attribute	LHS	RHS	Conclusion
D	–	–	must be in every key
A	✓	–	must be in every key
E	–	✓	is not in any key
B, C, F	✓	✓	must chek

In our example, this means that we only have to consider all combinations of B, C, and F For each, we must add in D and A, since they are in every key.

- $DAB^+ = DABCEF$. So DAB is a key.
- $DAC^+ = DACBEF$. So DAC is a key also.
- $DAF^+ = DAF$. This is not a key.
- All other possibilities include DAB or DAC, so we're done!

Instead of 64 closures, we only had to compute 3. Phew!