ECE356: Database Systems Winter 2018

Lecture 10 — Decomposition: Functional-Dependency Theory

Jeff Zarnett

Functional-Dependency Theory

Earlier we talked about the closure of a set of functional dependencies. Recall from earlier that the closure contains all the functional dependencies that are explicitly satisfied as well as those that are logically implied. That is, if $A \rightarrow B$ and $B \rightarrow C$ then it is implied that $A \rightarrow C$ and this is true no matter how many steps are in between. because this is transitive. More formally, if the functional dependencies is $A \rightarrow B$ and $B \rightarrow C$ then we know that for two tuples t_1 and t_2 if $t_1[A] = t_2[A]$ then $t_1[B] = t_2[B]$ and $t_1[C] = t_2[C]$. The notation to show the closure of a set F of functional dependencies is F^+ as previously discussed.

It is unlikely, however, that we want to compute F^+ from the definition of functional dependencies; if F is large there are many rules and many implied rules and we have to construct every logically implied element of F^+ from first principles. Instead, we would like to use *axioms*, handy rules of inference, that allow us to reason about the dependencies in a simpler way. The first three axions are simple enough and are called *Armstrong's Axioms*, named after the person who came up with them [**?**]:

- **Reflexivity**: If α is a set of attributes and β is contained within α, then $\alpha \rightarrow \beta$ holds.
- **Augmentation**: If $\alpha \rightarrow \beta$ holds and γ is a set of attributes, then $\gamma\alpha \rightarrow \gamma\beta$ holds.
- **Transitivity**: If $\alpha \rightarrow \beta$ holds and $\beta \rightarrow \gamma$ holds, then $\alpha \rightarrow \gamma$ holds.

These axioms are considered both sound and complete, because they do not produce any errors and they allow generation of F^+ given F. There are proofs of these properties, but we prefer to omit those and just focus on using them. This is a minimal set, though, and there are some rules that we can derive some more from Armstrong's rules that would be convenient shortcuts [**?**]:

- **Union**: If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.
- **Decomposition**: If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (reverse of previous rule).
- **Pseudotransitivity**: If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds.

We'll work on the example in the textbook([**?**]) which says that the relation r has the attributes (A, B, C, G, H, I) and the functional dependencies are: (1) $A \rightarrow B$, (2) $A \rightarrow C$, (3) $CG \rightarrow H$, (4) $CH \rightarrow I$, (5) $B \rightarrow H$.

Based on the rules that we have, what logically implied functional dependencies can we observe?

There are three [**?**]:

- $A \rightarrow H$ which is found by transitivity ($A \rightarrow B$ and $B \rightarrow A$).
- $CG \rightarrow HI$ which is found by the union rule ($CG \rightarrow H$ and $CG \rightarrow I$).
- $AG \rightarrow I$ which is found by pseudotransitivity ($A \rightarrow C$ and $CG \rightarrow I$).

The full algorithm for building up the closure just uses Armstrong's rules and is as below. The algorithm terminates when there is nothing left to add, which is certain to happen because the worst case scenario is that everything is functionally dependent on everything else which would mean if there are n attributes, there are $2^n \times 2^n$ possible functional dependencies. Now, the algorithm [**?**]:

1. The initial condition is that F^+ begins as F.
2. For each functional dependency f in F+ F^+:
   (a) apply the transitivity rule to f and add it to F^+
   (b) apply the augmentation rule to f and add it to F^+
3. For each pair of functional dependencies f_1 and f_2, if they can be combined using transitivity, add the newly created combination to F^+.
4. If anything was added in steps 2 or 3, go back to step 2; otherwise the algorithm terminates.

Closure of Attribute Sets. Suppose we have some attribute(s) α and we wish to determine if it is a superkey. The strategy we will use requires us to compute the set of attributes that are *functionally determined* by α. An attribute B is functionally determined by α if $\alpha \rightarrow B$. Then the simple algorithm requires us to compute F^+ as above and then take all functional dependencies where α is the left hand side and the union of the right hand side of all dependencies, but this is difficult and expensive because F^+ may be large. Instead, a more efficient algorithm follows [**?**].

1. The initial condition is that α^+ begins as α.
2. For each functional dependency $\beta \rightarrow \gamma$ in F, if β is contained in α^+, then γ is added to α^+
3. If anything was added in step 2, repeat step 2; if nothing was added, the algorithm terminates.

Much like computing the closure of F, here we compute the closure of α.

Going back to the earlier example with the five functional dependencies, we would like to evaluate something to see if it is a superkey. Given the rules, is A a superkey? Let's go through the steps.

The initial condition is that A is in α^+ (the result). Based on the rules, we can look at the rule $A \rightarrow B$ and see that we can add to the result B, which is now AB. The next rule that says $A \rightarrow C$ allows us to add C to the result so ABC. The rules that require CG on the left hand side cannot be added, at least not yet, because CG does not appear in the result (C does, but not G). We can look at the rule for $B \rightarrow H$ and add H to the result meaning the result is $ABCH$. This is not the full set ($ABCGHI$) and therefore we conclude that A is not a superkey.

This procedure can be repeated for any individual attribute and we will quickly find that no single attribute is a superkey for this relation.