Minimal Covers Revisited: Correct and Efficient Algorithms

Jim Diederich

Math Dept., UC Davis 95616

In [1] Nummenmaa and Tanisch show that the algorithm in [2] for computing minimal covers is incorrect even though it purports to correct the algorithms in [3-6]. As they illustrate with F = { A B -> C}, the algorithm in [2] allows B to be eliminated as an extraneous attribute since the dependency AB -> C is implied by A -> C using augmentation. Thus F is replaced by F' = {A -> C}, which is clearly not equivalent to F. The problematic step of the algorithm in [2] that allows this to occur is

> Consider each dependency X -> A in some order. If Z is a subset of X such that F is contained in the closure of $(F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$, then immediately replace X -> A by Z -> A in F. This step continues until no left side of any dependency in F can be reduced

Nummenmaa and Tanisch propose a solution in which one not only tests to show that F is contained in the closure of $(F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ but also tests that $(F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\}$ is contained in the closure of F. One problem with this proposed solution is it makes algorithms that are already highly inefficient even more so. In [7] we proposed new efficient methods for computing minimal covers. The purpose of this note is to illustrate these methods, make them more accessible, and in particular, to show that they already avoid the pitfalls discussed by Nummenmaa and Tanisch, so that we get correct and efficient algorithms.

The dependencies F in Fig. 1 will be used to illustrate first our techniques for eliminating redundant dependencies. After that we will use another set to illustrate removing extraneous attributes. Let F(W) denote the set of dependencies with lhs, left hand side, W. For example, F(emp#) is the set of six dependencies in the left hand column of Fig.1. From [7] **the r-closure of X with respect to a set of dependencies G,** denoted $X_G^{\wedge}$, is defined as the set of all attributes A such that Y -> A is in G and Y is in standard closure of X with respect to G, denoted X_G^{+}. For example, with G = F and X = {emp#} the standard closure of X is

$\{emp\#\}_G^{+}$ = {emp#, emp-name, address, title, salary, dept#, manager, dept-name, building#, office#}

while the r-closure is

$\{emp\#\}_G^{\wedge}$ = {emp-name, address, title, salary, dept#, manager, dept-name, building#, office#}.

The attribute emp# is not in $\{emp\#\}_G^{\wedge}$ because there is no dependency Y -> emp# in G with Y in $\{emp\#\}_G^{+}$. The attributes of X are not automatically included in the r-closure $X_G^{\wedge}$, while they are always included in the standard closure X_G^{+}. As with the example F, we assume that the set of dependencies has no trivial or duplicate dependencies since they can be eliminated easily without computing closures.

F:		
emp# -> emp-name	dept# -> dept-name
emp# -> address	dept# -> manager
emp# -> title	dept# -> building#
emp# -> salary	dept# -> office#
emp# -> dept#	
emp# -> manager	dept-name -> dept#

Figure 1

A simple modification of standard closure algorithms can be used to generate the r-closure of X. The only difference is that we start with $X^{\wedge}$ empty and X^{+} = X and each time a dependency is examined that has its lhs in X^{+}, we add its rhs, right hand side, to $X^{\wedge}$ and X^{+}, if it is not currently in either or both of them.

The power of r-closures lies in the way they are used to reduce the number of closures computed in eliminating extraneous attributes and redundant dependencies. Computing closures is typically the costliest part of the processes.

We first illustrate r-closures in eliminating redundant dependencies. With F in Fig. 1, take G = F - F(emp#), that is, G is F less the set of dependencies in F with common lhs emp#. We take X = {emp#, dept#}, i.e., the emp# is included in X because it is the lhs of the dependencies in F(emp#), and the dept# is included because it occurs on the rhs of some dependency in F(emp#) and occurs on the lhs of some dependency in the set G. No other attributes in the dependencies in F(emp#) qualify. Computing the r-closure we get

$\{emp\#, dept\#\}_G^{\wedge}$ = { dept-name, manager, building#, office#, dept#}.

From [7] a dependency in F(emp#) whose rhs does not appear in { emp#, dept#}$_G^\wedge$ is not redundant in F. Thus the first 4 dependencies in F(emp#) are not redundant. Likewise any dependency in F(emp#) whose rhs appears in { emp#, dept#}$_G^\wedge$ and not in X must be redundant. Thus the sixth dependency in F(emp#) is redundant. The only dependency remaining in F(emp#) is the fifth one. It must be checked using standard methods since its rhs is in the r-closure and also in X. It is not redundant though. So we used 1 r-closure and 1 standard closure instead of 6 standard closures to eliminate redundant dependencies in F(emp#). Setting E equal to F less the redundant dependency detected, we take G = E - E(dept#). In this case X = {dept#, dept-name}, no other attributes qualify, and the r-closure with respect to G is {dept#}. Consequently, no dependency in E(dept#) is redundant. The case for G = E - E(dept-name) is similar with the r-closure of {dept-name, dept#} equal to {dept-name} and no dependency in E(dept-name) is redundant. Thus we have used 4 closures total, 3 r-closures and 1 standard, instead of 11 standard closures to eliminate the redundant dependencies in F.

To illustrate eliminating extraneous attributes with r-closures we use the dependencies H in Fig. 2.

H:	a b -> d	b -> a
	a b -> e	b -> h
	a b -> f	b -> g
	a b -> g	
	a b c -> d	d -> a
	a b c -> j	
	a b c -> k	b n -> h

Figure 2

We start with G = H - H(a b) and X = {a, b, d}. As before the attributes {a, b} are included in X since they form the lhs of the dependencies in H(a b) and d is included because it appears on the rhs of some dependency in H(a b) and the lhs of some dependency in G. The r-closure is { a, b, d}$_G^\wedge$ = { a, h, g}.

The results in [7] establish that among the attributes {a, b}, which form the lhs of the dependencies H(a b), only those that appear in the r-closure need to be checked as possibly extraneous, so only the attribute 'a' needs to be checked. This is done by computing the standard closure of {b}, i.e.,{a,b} - {a}, with respect to the current full set H. Since 'a' is in this closure we can eliminate it from the lhs of all of the dependencies in H(a b) because it is implied by the other attributes, i.e., {b}. We don't have to check each dependency and we don't have to check every attribute!

The dependencies in H(a b c) are handled in the same manner and only 'a' is tested and eliminated as extraneous. Dependencies with only a single attribute on the lhs do not need to be checked. That leaves the final dependency to be checked, i.e., b n -> h. Computing the r-closure with this dependency removed to form G yields {b, n}$_G^\wedge$ = {d, e, f, g, a , h}. Since neither b nor n are in the r-closure we don't check either as extraneous. Yet, clearly n is extraneous since b -> h is a dependency in H. The reason we don't check this is because such dependencies, where the extraneous attributes are not implied by the other attributes on the lhs, will be removed as redundant when the dependencies are checked for redundancy, so we can ignore them when checking for extraneous attributes.

The total number of closures in eliminating extraneous attributes from H using r-closures is 5 versus 19 standard closures for the standard methods. If in addition the check suggested by Nummenmaa and Tanisch is added, the total climbs from 19 to 27. Also, the example given by Nummenmaa and Tanisch does not cause any problems for our methods because the r-closure of {A, B} will be empty since there is only one dependency and it is removed from the set to form G and thus neither of {A, B} needs to be checked. Finally we point out that r-closures do not handle the simple cases of trivial dependencies like A -> A or duplicate dependencies, that is, two copies of the same dependency in a set. But these can be more efficiently handled without closures anyway. For further details and for some benchmarks for r-closures in creating minimal covers see [7].

References

[1] Nummenmaa, J., and P. Thanisch. Yet Another Note on Minimal Covers, Sigmod Record, 19, No. 3, Sept. 1990, p. 30.

[2] Atkins, J. A Note on Minimal Covers, Sigmod Record, 17, No. 4, Dec. 1988, pp. 16-21.

[3] Salzberg, B. Third Normal Form Made Easy, Sigmod Record, 15, No. 4, Dec. 1986, pp. 2-18.

[4] Stout, F., and A. Woodworth. Relational Databases, The Amer. Math. Monthly, 90, 1983, pp. 101-118.

[5] Ullman, J. Principles of Database Systems, Computer Science Press, 1982.

[6] Yang, C. Relational Databases, Prentice-Hall, 1988.

[7] Diederich, J., and J. Milton, New Methods and Fast Algorithms for Database Normalization, TODS, 13, No. 3, Sept. 1988, pp. 339-365.