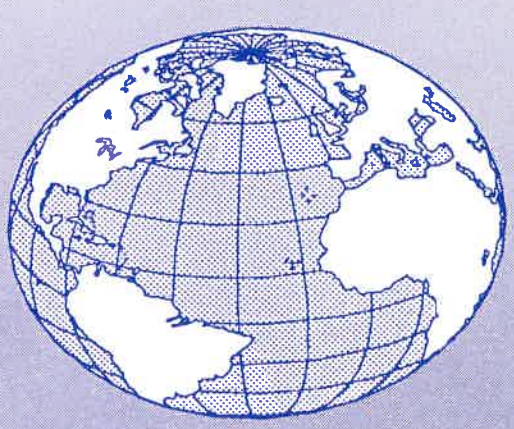
JOURNAL OF COMPUTER INFORMATION SYSTEMS

official journal of

INTERNATIONAL ASSOCIATION FOR COMPUTER INFORMATION SYSTEMS



VOLUME XXXIII
NUMBER 4
SUMMER 1993

JOURNAL OF COMPUTER INFORMATION SYSTEMS

Volume XXXIII, Number 4 Summer 1993

CONTENTS

KEY FINDING THROUGH EXAMINATION OF ATTRIBUTES IN THE FUNCTIONAL DEPENDENCIES

H. SAIEDIAN
University of Nebraska
Omaha, Nebraska 68182

INTRODUCTION

An important activity during the design of a relational database is to find the candidate keys of the individual relation schemes. Traditionally (as described in most database textbooks), a key of relation scheme *R* is determined by computing the closure of attributes in *R* (unless the key was trivial to find). Computing the closure of various combination of attributes to find all keys found could be a difficult and time consuming process. We provide an algorithm that attempts to simplify the key finding process by analyzing the position of attributes in the given functional dependencies and attempting to first identify those attributes that *must* be part of the key, as well as those attributes that may never be part of the key before computing the closures. A number of examples are provided to illustrate the effectiveness of the algorithm.

BACKGROUND

Given a relation R with attributes A_1, A_2, ...,A_n, and a set of FDs F, $K \subseteq R$, is a key of R if:[1]

- $K \rightarrow A_1A_2...A_n \in F^+$ and
- For no K', $K' \subset K$, is $K' \rightarrow A_1A_2...A_n \in F^+$.

Given a universal relation scheme R and a set of attributes F, it is essential to correctly determine all candidate keys of R. Most database textbooks (e.g., 3,2), however, suffice to provide a definition for the key but no algorithm for computing it. Others (e.g., 4) provide algorithm for computing the closure of a set of attributes (or a set of FDs) but calculation of a key is left to the students using the closure algorithms. Determining the candidate keys for a small relation with a small set of FDs may be trivial but if a relation has a relatively large number of attributes and/or FDs then determining the keys may not be a trivial process.

Using the closure algorithms while not considering the "arrangement" of attributes in F may be time consuming and inefficient. Some attributes may never participate in a key. Consideration of other attributes may lead to superkeys while late consideration of certain other attributes may simply prolong the process or lead to incorrect answers. We believe that certain "categorization" of attributes (based on their appearance on the left-hand side and right-hand side of FDs) could expedite the process substantially and lead to correct solutions perhaps early in the process. This categorization includes identifying

1. those attributes that must be part of a key,
2. those attributes that may or may not be part of a key, and
3. those attributes that will not be part of any key.

The last set of attributes can be ignored since they cannot be part of any key. The second set should be considered only if the first set does not produce all the candidate keys.

The process is described below. We have successfully applied it to many "academic" problems. In addition, the algorithm has been implemented (in both Pascal and C) and applied to several large relational database schemas. The results have been satisfactory.

KEY FINDING ALGORITHM

The following four steps lead to calculation of the candidate keys of a relation scheme. We assume a relation scheme R and a set of FDs F. Furthermore, we assume that the set of FDs is minimal.[2]

Step 1: Determining LHS, LRS and RHS

Given a relation R and a set of FDs F, divide the attributes of R into three distinct sets LHS, RHS, and LRS. The set LHS contains those attributes of R that occur only on the left-hand side of some FDs in F. Similarly, the set RHS represents those attributes that occur only on the right-hand side of some FDs in F, while LRS is the set representing those attributes that occur on both sides of some FDs in F. Observe that $LHS \cap RHS = \emptyset$ and $LHS \cap LRS = \emptyset$ and $RHS \cap LRS = \emptyset$. Furthermore, $LHS \cup RHS \cup LRS = R$.

Step 2: Considering LHS

Consider the set LHS. If LHS is not empty, then all attributes participating in LHS are prime attributes. (It can be proved that for every attribute $A \in R$, if $A \in LHS$, then A must be part of a candidate key of R.)[3]

Thus, we begin our process of computing the closure of attributes in LHS. (It can also be proved that if $LHS^+ \models R$ then LHS is the only key of R.) If a key is found, stop. Otherwise proceed to Step 3.

[1] We assume reader's familiarity with basic database concepts such as functional dependencies, candidate and superkeys, closures of a set of FDs (shown as F^+), Armstrong inference rules (1), logical implication of FDs (shown as $\models$), etc. All these concepts are explained in detail in (3).

[2] Generally speaking, a set of FDs is minimal if it contains no redundant FDs and there are no extraneous attributes on the left-hand side of FDs. Algorithm for computing a minimal cover is given in (3, 4).

[3] The proof of this and other similar statements in this paper as well as theorems about a more general heuristic approach to key finding using attribute graphs has been submitted for publication to the **Information Processing Letters.**

Step 3: Considering LRS

If the set LHS does not produce a key for R in Step 2, then begin adding attributes, one by one, from the set denoted by LRS to attributes of LHS and compute their closure. Attributes should be added to LHS in turn to ensure that all candidate keys of R are found. (If LHS is empty, then begin by computing the closure of attributes in the set LRS.)

It is most likely that all candidate keys of R are found in this step. Proceed to Step 4 if no key is found.

Step 4: Considering RHS

If no key is found in Steps 2 and 3, then consider adding all attributes of the remaining set, i.e., RHS, to attributes of LHS and LRS and compute their closure. This step rarely occurs. It needs to be followed only when computing the key of a relation that has resulted from the decomposition of the universal relation. If we are only interested in calculating the candidate keys of a universal relation, the attributes of RHS need not be even considered. (It can be proved that for every attribute A ∈ R, if A ∈ RHS, then A may not be a part of any candidate key of R.)

EXAMPLES

In this section, we use the above algorithm to find the candidate keys of the given relation schemes. For each example, a universal relation scheme R with a set of FDs F are given.[4]

Example 1. Given schema (R(CSZ) and F = {CS –> Z, Z –> C}, find all candidate keys of R.[5]

Solution. Based on the above, LHS = {S}, LRS = {ZC}, RHS = {}. Thus S must be in every key of R. Compute closure of S:

$LHS^+ = S^+ = S$.

LHS does not contain a key. Add attributes from LRS:

$\{SC\}^+$ = SCZ. SC is a key.

$\{SZ\}^+$ = SZC. SZ is a key.

Thus {SC} and {SZ} are the candidate keys of R. We need not consider combination of ZC because S must be part of every key. Furthermore, we need not consider SCZ because that would imply a superkey. °

Example 2. Given schema R(ABCDE) and F = {AB –> CDE,AC –> BDE,B –> C,C –> D,B –>E}, find all candidate keys of R.

Solution. LHS = {A}, LRS = {BC},RHS = {DE}. Thus A must be part of every key of R while D and E may not participate in any key. We begin by computing of LHS^+:

$LHS^+ = A^+ = A$. A is not a key.

A is not a key. Next we consider combining attributes from LRS with A and compute their closure:

$\{AB\}^+$ = ABCDE. AB is a key of R.

$\{AC\}^+$ = ACD. AC is not a key of R.

According to the algorithm, AB is the only key of R. Note that we need not consider BC because A must be part of any key. We need not consider any combination of D or E because these two attributes may not be part of any. Other combinations would yield a superkey. °

Example 3. Given schema R(ABCDEFG) and F = {A –> B, B –> C, AG –> D, BG –>E, CD –> AF}, find all candidate keys of R.

Solution. LHS = {G}, LRS = {ABCD}, RHS = {EF}. Thus G must be part of any key while E and F may not participate in any key:

$LHS^+ = G^+ = G$. G is not a key by itself.

Therefore we add attributes from LRS to G and compute their closure:

$\{GA\}^+$ = GABCDEF ... GA is a key.

$\{GB\}^+$ = GBEC. GB is not a key.

$\{GC\}^+$ = GC. GC is not a key.

$\{GD\}^+$ = GD. GC is not a key.

We now consider potential three-attribute keys (note that we need not consider GBC because it cannot be a key as shown above):

$\{GBD\}^+$ = GBDCDA {GBD} is another key.

$\{GCD\}^+$ = GCDAF ... {GCD} is also another key.

The candidate keys of R are {GA}, {GBD} and {GCD}. Any other combination is either a superkey or not a key. °

Example 4. Given schema R(ABCDEFG) and F = {A –> B, CD –> A, CB –> E, AF –> G, CF –> D}, find all candidate keys of R.

Solution. LHS = {CF}, LRS = {ABD}, RHS = {EG}. Thus CF must be part of any key while E and G may not participate in any key. Note that according to the algorithm, (a) we need not consider computing the closure of any single attribute because LHS contains two attributes, (b) nor need we to consider attributes E and G because they appear in RHS, and (c) we should consider attributes of LHS before considering attributes A, B, or D. We begin by computing the closure of LHS:

$LHS^+ = \{CF\}^+$ = CFDABG ...

Thus {CF} is a key.

According to the algorithm, {CF} is the only key of R. We need not consider any other combination of attributes.

Note how the key finding process has been simplified by the above algorithm. Computing the closures of every potential collection of attributes to find all keys of even a small relation like the above example would have been too time consuming.

CONCLUSIONS

In the above examples, we considered a situation where the set LHS (and/or RHS) had at least one element and we showed how the key finding process was simplified once these two sets were identified. If LHS = $\emptyset$ and RHS ≠ $\emptyset$ then it implies that except attributes of RHS, every other attribute has a fair chance of being part of a candidate key. If LHS = $\emptyset$ and RHS = $\emptyset$ (that is, LRS contains all attributes), then it implies that every attribute might be a potential component of a key. This is the case when the general solution of computing the closure of different combination of attributes would be the only alternative.

REFERENCES

1. Armstrong, W. "Dependency Structures of Database Relationships," **IFIP Congress**, Geneva, 1974, pp. 580-583.
2. Date, C.J. **An Introduction to Database Systems**, Volume I, Fifth Edition, Addison-Wesley, 1990.
3. Elmasri, R. and S. Navathe. **Fundamentals of Database Systems**, Addison-Wesley, 1989.
4. Ullman, J. **Principles of Database and Knowledge-based Systems**, Volume I, Computer Science Press, 1988.

[4] Note that our objective is to find the keys not the superkeys of R.

[5] This example is from (4), page 384. Attributes C, S, and Z stand for City, State and Zip Code, respectively.