DOI: 10.19884/j.1672-5220.202504018

Recognizing Prime Attributes Based on *F*-based Derivation Tree

YU Xiaoxue, LIU Guohua*, XU Yulu, LIU Changqi, ZHANG Limeng, ZHU Dongyan, HE Songda
School of Information and Intelligent Science, Donghua University, Shanghai 201620, China

Abstract: Although the prime attribute problem is nondeterministic polynomial-time complete (NP-complete), the difficulty of determining whether an attribute is prime is related to how the attribute appears in the set of functional dependencies (FDs). The prime attribute problem is extensively studied based on the occurrence of an attribute in the set of FDs. First, in a relation scheme $R(U, F)$, the attribute set U is partitioned into four distinct subsets according to where attributes in U appear in F: U_1 (attributes appearing exclusively on the left-hand sides of FDs in F), U_2 (attributes appearing exclusively on the right-hand sides of FDs in F), U_3 (attributes appearing on both sides of FDs in F), and U_4 (attributes not appearing in any FD in F). Second, the occurrence patterns of attributes in the set of FDs are mapped to an *F*-based derivation tree (*F*-based DT) forest, and the features of attributes in U_1, U_2, U_3 or U_4 are shown in the *F*-based DT forest. Then, an algorithm for recognizing prime attributes based on *F*-based DT is proposed. Finally, the following conclusions are obtained: when an attribute belongs to U_3, the prime attribute problem is NP-complete; when an attribute belongs to U_1, U_2, or U_4, the problem is in polynomial-time complexity class (P-class), meaning it can be efficiently solved in polynomial time. Compared with traditional methods, the proposed algorithm can simplify the process for recognizing prime attributes significantly, and improve database normalization.

Keywords: *F*-based derivation tree; prime attribute; NP-complete; normalization
CLC number: TP311.13 **Document code**: A
Article ID: 1672-5220(2026)04-0092-10

0 Introduction

Prime attributes play an important role in the process for deciding normal forms. However, since the prime attribute problem is nondeterministic polynomial-time complete[1] (NP-complete), developing efficient methods has always been a key goal in the field of database normalization. Since keys are carriers of prime attributes, many algorithms for finding keys have emerged over the years[2-16]. Existing studies can be roughly classified into several categories, including tableaux-based methods[2-3,12], graph-based methods[5,7,11,15], attribute classification methods[8,11,16], and candidate-key construction methods[6,9,14]. Among these methods, graph-based methods have shown advantages in terms of feasibility. Feng[7] proposed a graph-based algorithm that only included one attribute on the left-hand side of each functional dependency (FD). Similarly, Saiedian et al.[11] utilized directed graphs to compute candidate keys in polynomial time. Furthermore, they represented the FDs among attributes in a relation scheme by using a graph and introduced an algorithm to find all minimal keys in the same work. Bahmani et al.[17] suggested an automatic method based on graph theory for generating keys. The candidate graph, as proposed by Bordoloi et al.[18], serves as an approach to finding all minimal keys. Additionally, Fernandez et al.[5] defined port graph rewriting rules and a strategy for determining minimal candidate keys in a relation scheme, along with a method based on the strategic port graph rewriting to find all minimal keys.

Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. Based on their appearance in F, the attributes in U can be partitioned into four distinct subsets: U_1, U_2, U_3, and U_4. Specifically, U_1 consists of attributes that appear exclusively on the left-hand sides of FDs in F, U_2 consists of attributes that appear exclusively on the right-hand sides of FDs in F, U_3 consists of attributes that appear on both the left-hand sides and the right-hand sides of FDs in F, and U_4 consists of attributes that do not appear in any FD in F. Let X and V be subsets of U, while A and B are attributes in U. The set of all FDs that can be derived from F by repeatedly applying the inference rules, including the FDs in F itself, is called the closure of F and is denoted by F^+. The feature of the prime attribute is shown in Ref. [19], namely, an attribute A is a prime attribute if and only if for any FD $X \rightarrow A \in F^+$, there must be an FD $V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$. Thus, it is an important step to find pairs of FDs $X \rightarrow A$ and $V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$. The derivation of an FD plays a crucial role in the finding process.

Received date: 2025-04-23
* Correspondence should be addressed to LIU Guohua, email: ghliu@dhu.edu.cn
Citation: Yu X X, Liu G H, Xu Y L, et al. Recognizing prime attributes based on *F*-based derivation tree [J]. *Journal of Donghua University (English Edition)*, 2026, 43(4): 92-101.

To model derivations, Beeri et al.[20] introduced a graph model known as *F*-based derivation tree (*F*-based DT). It has practical value for finding pairs of FDs $X \rightarrow A$ and $V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$. In contrast to the aforementioned work, our focus is on addressing the prime attribute problem. The *F*-based DT serves as an effective tool for describing the derivation of an FD and revealing features of prime attributes. Based on this model, we propose a recognition algorithm for prime attributes.

We aim to investigate the identification of prime attributes based on *F*-based DT and make the following contributions.

1) We prove that the prime attribute problem is in polynomial-time complexity class (P-class) when the attribute is in U_1, U_2, and U_4, but it is NP-complete when the attribute is in U_3.

2) We propose an algorithm for recognizing prime attributes based on the *F*-based DT, which can simplify the process for recognizing prime attributes significantly, and improve the database normalization.

Section 1 provides an introduction to relational model preliminaries. Section 2 first discusses the representation of prime and nonprime attribute features in *F*-based DTs, followed by the presentation of an algorithm for recognizing prime attributes based on *F*-based DTs. Section 3 analyzes the time efficiency of Algorithm 3. Section 4 concludes the paper.

1 Preliminaries

1.1 Relation schemes and relations

A relational database is composed of a collection of tuple sets. Each tuple set in this collection is referred to as the state or extension of the associated relation scheme, and it can be viewed as a relation in the mathematical sense[21]. Therefore, the state or extension of a relation scheme is also known as a relation. A relation scheme is used to describe a single relation and consists of the relation name, attributes, and a set of data dependencies. The relation scheme is denoted as $R(U, \Gamma)$, where R represents the relation name, $U = \{A_1, A_2, \cdots, A_n\}$ is the set of attributes appearing in the scheme, and each attribute has an associated domain consisting of a set of possible values. It is essential to recognize that different attributes may have the same domain. The set of data dependencies is represented by Γ. The primary data dependency discussed in this paper is the FD, which will be defined precisely in Subsection 1.2.

In practical terms, a relation is typically depicted as a table, with each column representing a distinct attribute and each row representing a distinct entity. An <entity, attribute> entry in the relation denotes a value linked to the entity, chosen from the attributes domain. The set of tuples that comprise a relation usually changes over time as entities are inserted, deleted, or modified, while the schema of the relation tends to remain relatively static throughout the lifecycle of the relation.

1.2 FDs

FDs are a central concept in the database, significantly influencing both the design of database schemas and the optimization of queries[22]. FDs, serving as fundamental integrity constraints, formally characterize the interrelationships between attributes and constitute a specialized instrument within relation schemes.

Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. Let A and B be two arbitrary attributes in U, which are denoted by $A, B \in U$. If A functionally determines B, or equivalently, B is functionally dependent on A, there exists at most one corresponding value of $b \in \mathrm{DOM}(B)$ for each value of $a \in \mathrm{DOM}(A)$, where $\mathrm{DOM}(A)$ and $\mathrm{DOM}(B)$ represent the domains of A and B, respectively. It is common practice to omit the DOM and use the notation $f: A \rightarrow B$ to indicate the functional dependence of B on A. This relationship is referred to as an FD.

By applying Armstrong's axioms[9-10], it is possible to infer additional FDs beyond those explicitly provided in F. The closure of F, denoted as F^+, consists of all FDs that can be derived from F, including both the dependencies originally in F and those generated through repeated applications of these axioms.

1.3 Prime attributes and keys

Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. A subset K of U in relation R is a superkey if every attribute in U is functionally determined by K. A subset K of U is considered a candidate key of R if it is a minimal superkey, meaning that no proper subset of K can serve as a superkey of R. An attribute A is classified as a prime attribute if it belongs to any candidate key of R; otherwise, it is considered as a nonprime attribute.

Lemma 1[19] Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs.

1) Attributes in U_1 or U_4 are prime attributes, and $U_1 \cup U_4$ is included in any key of R.

2) Attributes in U_2 are nonprime attributes.

3) K is the unique key of R if and only if $K \cap U_3 = \emptyset$.

Lemma 2[19] Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. If attribute A belongs to U_3, attribute A is a prime attribute if and only if for any $X \rightarrow A \in F^+$, there must exist an FD $V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$.

Lemma 3[19] Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. Attribute A is a prime attribute if and only if for any $X \rightarrow A \in F^+$, there must exist an FD $V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$.

1.4 *F*-based DT

To model derivations, Beeri et al.[20] introduced a graph model, called *F*-based DTs. Let *F* be a set of FDs. The set of *F*-based DTs is defined as follows.

Rule 1 If A is an attribute, then a node labeled with A is an *F*-based DT.

Rule 2 If T is an *F*-based DT with a leaf node labeled with A and the FD $B_1B_2\cdots B_m \rightarrow A$ is in F, then the tree constructed from T by adding $B_1B_2\cdots B_m$ as children of the leaf labeled with A is also an *F*-based DT.

Rule 3 A labeled tree is an *F*-based DT only if it is obtained by a finite number of applications of Rules 1 and 2.

If the set of leaves' labels in a DT is contained in the set of attributes X, then we call it an X-DT. An X-DT with root labeled by A is called a DT for the FD $X \rightarrow A$.

Lemma 4[20] The FD $X \rightarrow Y$ is in F^+, where X and Y are sets of attributes, if and only if for each attribute A in Y, there exists an *F*-based X-DT with root labeled A.

Lemma 5[20] Let T be an *F*-based DT. Y is a nonempty set of nodes' labels of T and X is the set of labels for all leaves in T. Then, the FD $X \rightarrow Y$ is in F^+.

2 Recognizing Prime Attributes Based on *F*-based DT

The features of prime and nonprime attributes have been presented, and the necessary and sufficient condition for an attribute to be classified as a prime attribute has been formally proven in early research in Ref. [19].

According to Lemma 3, in order to determine whether attribute A is a prime attribute, it is essential to identify each pair of FDs $X \rightarrow A, V \rightarrow B \in F^+$, where $A \in V$ and $B \in X$. Therefore, the derivation of an FD holds significant importance in the decision-making process. The *F*-based DT serves as an effective tool for deriving FDs. In this section, we will present a method for recognizing prime attributes based on the *F*-based DT.

Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. For simplicity, we assume that all FDs have single attributes on their right-hand sides, unless explicitly stated otherwise. According to the definition of an *F*-based DT, there exists at least one *F*-based DT with a root labeled as A for any attribute in U.

We define an *F*-based DT as maximal if the label of each leaf is identical to the label of only one ancestor, or the label is in U_1. The maximal *F*-based DT is denoted as an MF-based DT. For the FD $X \rightarrow A$, if there exists an *F*-based DT with a root labeled as A, and the set of labels of the leaves is a subset of the attributes in X, we refer to it as an (X, A)-DT. Furthermore, we define a set of *F*-based DTs as an *F*-based DT forest.

Example 1 Consider a relation scheme $R(U, F)$, where $U = \{A, B, C, D, E, G, H\}$ is the set of attributes, and $F = \{A \rightarrow B, A \rightarrow C, BC \rightarrow A, BCD \rightarrow E, E \rightarrow G, G \rightarrow E\}$ is the set of FDs.

The MF-based DT forest for Example 1 is illustrated in Fig. 1.

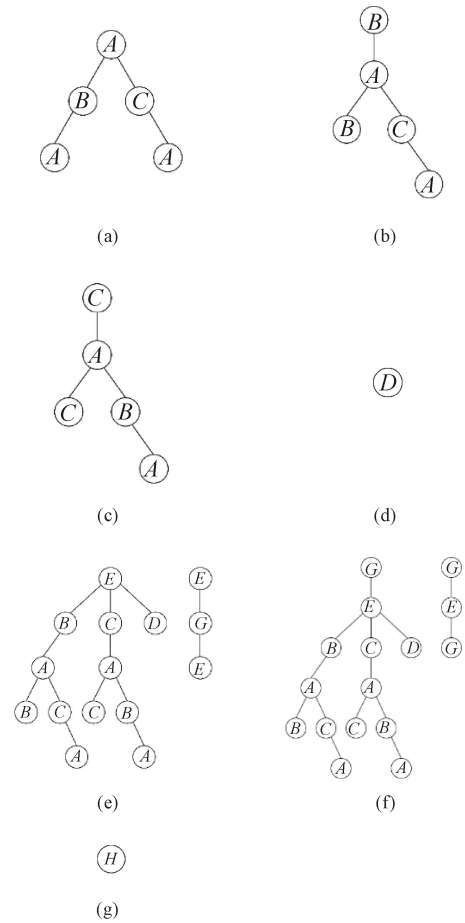


Fig. 1 MF-based DT forest whose roots are labeled with different attributes: (a) A; (b) B; (c) C; (d) D; (e) E; (f) G; (g) H

Lemma 6 Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. Attributes in U_1 or U_4 correspond to an *F*-based DT forest in which every *F*-based DT contains only one node.

Proof If A is in U_1 or U_4, it does not appear on the right-hand side of any FD in F. According to the definition of an *F*-based DT, the *F*-based DT corresponding to A consists of only one node labeled with A. Therefore, A is in U_1 or U_4, corresponding to an *F*-based DT forest in which every *F*-based DT consists of only one node.

In Example 1, $U_1 = \{D\}$, $U_4 = \{H\}$, Figs. 1 (d) and 1 (g) represent the *F*-based DTs corresponding to U_1 and U_4, respectively, and they both consist of only one node.

Lemma 7 Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. In

an *F*-based DT forest, if an *F*-based DT consists of only one node, the attribute labeled on that node belongs to either U_1 or U_4.

Proof According to the definition of an *F*-based DT, if an *F*-based DT consists of only one node, the attribute labeled on that node does not appear on the right-hand sides of any FDs in *F*. Therefore, the attribute may only appear on the left-hand sides of FDs in *F* or may not appear in *F* at all, indicating that the attribute belongs to either U_1 or U_4.

In Example 1, the *F*-based DT forest comprises nine MF-based DTs. Figures 1 (d) and 1 (g) represent the *F*-based DTs, each consisting of only one node labeled as *D* and *H*, respectively. Based on the attribute appearance in *F*, we can deduce that attribute *D* belongs to U_1 and attribute *H* belongs to U_4.

Theorem 1 Let $R(U, F)$ be a relation scheme, where *U* is the set of attributes, *F* is the set of FDs and the right-hand side of each FD in *F* contains only one attribute. *T* is the *F*-based DT forest corresponding to attributes in *U*. If an *F*-based DT in *T* consists of only one node, the attribute labeled on the node is a prime attribute.

Proof If an *F*-based DT in *T* consists of only one node, according to Lemma 7, the attribute labeled on that node belongs to either U_1 or U_4. By Lemma 1, attributes in U_1 or U_4 are prime attributes. Therefore, the attribute labeled on the node is also a prime attribute.

Lemma 8 Let $R(U, F)$ be a relation scheme, where *U* is the set of attributes, *F* is the set of FDs and the right-hand side of each FD in *F* contains only one attribute. For an *F*-based DT forest in which the root of a tree is labeled with an attribute in *U*, if a subtree consists only of the root and its children, then the subtree represents an FD in *F*.

Proof Let (X,A)-DT be the subtree consisting only of the root and its children. Consequently, there exists an *F*-based *X*-DT with root labeled as *A*. By Lemma 4, an FD $X \rightarrow A$ is in F^+. According to the definition of an *F*-based DT, for an FD $B_1B_2\cdots B_m \rightarrow A$ in *F*, the nodes labeled as $B_1B_2\cdots B_m$ are the children of the root labeled as *A*. By the definition of (X, A)-DT, $B_1B_2\cdots B_m$ are included in the set of attributes *X*. Thus, the subtree (X,A)-DT can represent the FD $B_1B_2\cdots B_m \rightarrow A$.

In Example 1, (BC, A)-DT is a subtree that consists only of the root and its children. From *F*, we can deduce that $BC \rightarrow A \in F$.

Lemma 9 Let $R(U, F)$ be a relation scheme, where *U* is the set of attributes, *F* is the set of FDs and the right-hand side of each FD in *F* contains only one attribute. For an *F*-based DT forest in which the root of a tree is labeled with an attribute in *U*, if an attribute is never labeled on a child of a root in the *F*-based DT forest, then the attribute must be in U_2.

Proof According to Lemma 8, a subtree consisting of a root and its children represents an FD in *F*. If an attribute is never labeled on a child of a root in the *F*-based DT forest, it implies that the attribute does not appear exclusively on the left-hand sides of FDs in *F*. Therefore, the attribute must be in U_2.

Lemma 10 Let $R(U,F)$ be a relation scheme, where *U* is the set of attributes, *F* is the set of FDs and the right-hand side of each FD in *F* contains only one attribute. For an *F*-based DT forest in which the root of a tree is labeled with an attribute in *U*, if an attribute labeled on a root is also labeled on a child of another root in the *F*-based DT forest, the attribute must be in U_3.

Proof According to Lemma 8, a subtree consisting of a root and its children represents an FD in *F*. If an attribute labeled on a root is also labeled on a child of another root in the *F*-based DT forest, then it indicates that the attribute appears on both sides of FDs in *F*. Therefore, the attribute must be in U_3.

In Example 1, *A*, *B*, *C*, *E* and *G* are the labels of roots of MF-based DTs in Figs. 1 (a), 1 (b), 1 (c), 1 (e) and 1 (f), respectively, *A* is the label of a child of the roots of (AB, B)-DT and (AC, C)-DT, *B* is the label of a child of the root of (A, A)-DT, *C* is the label of a child of the root of (A, A)-DT, *E* is the label of a child of the root of (G, G)-DT, and *G* is the label of a child of the root of (E, E)-DT. Based on the appearance of attributes in *F*, we can deduce that *A*, *B*, *C*, *E* and *G* are in U_3.

Theorem 2 Let $R(U, F)$ be a relation scheme, where *U* is the set of attributes, *F* is the set of FDs and each FD in *F* only contains a single attribute on the right-hand side. In an *F*-based DT forest, where the root of each tree is labeled with an attribute from *U*, if an attribute never appears as a child of any root in an *F*-based DT forest, then it must be a nonprime attribute.

Proof By Lemma 9, if an attribute is never labeled on a child of a root in the *F*-based DT forest, it must be in U_2. According to Lemma 1, all attributes in U_2 are nonprime attributes. Therefore, if an attribute is never labeled on a child of a root in the *F*-based DT forest, it must be a nonprime attribute.

In our preliminary research, we have established the relationship between an attribute's possibility of becoming a prime attribute and the set $U_i(1 \leqslant i \leqslant 4)$ it belongs to. Algorithm 1 can determine $U_i(1 \leqslant i \leqslant 4)$ that an attribute is in.

Algorithm 1 Recognizing attributes in U_1, U_2, U_3, or U_4 based on F-based DT

INPUT: A relation scheme $R(U, F)$, where U is the set of attributes, F is the set of FDs and an attribute $A \in U$

OUTPUT: If A is in U_1, then return "1"; else if A is in U_2, then return "2"; else if A is in U_3, then return "3"; else if A is in U_4, then return "4"

Begin

1: $T_0 \leftarrow \emptyset$; $T_1 \leftarrow \emptyset$

2: for $i = 1$ to $|U|$ do {

3: $t_i \leftarrow$ a tree consisting of only one node labeled with the ith attribute in U;

4: $T_0 \leftarrow T_0 \cup \{t_i\}$;

} //Construct the F-based DT forest with each F-based DT consisting of only one node

5: for $i = 1$ to $|T_0|$ do {

6: $k \leftarrow 1$;

7: for $j = 1$ to $|F|$ do {

8: If the attribute labeled on the root of t_i appears on the right-hand side of the jth FD in F, then construct an F-based DT r_k with the root labeled as the attribute on the root of t_i and the leaves labeled with the attributes appearing on the left-hand side of the jth FD in F;

}

9: $T_1 \leftarrow T_1 \cup \{r_k\}$;

10: $k \leftarrow k + 1$;} //Construct the F-based DT forest in which the leaves of the root of every F-based DT are the children of the root

11: If attribute A is the label on the root of an F-based DT consisting of only one node and is also only the label on the leaf of another F-based DT in T_1, then return "1";

else if attribute A is the label on the root of an F-based DT consisting of only one node, then return "4";

else if attribute A is not the label on the leaf of an F-based DT in T_1, then return "2";

else return "3";

End

Let $R(U, F)$ be a relation scheme, where U and F are the same as those in Example 1. We will determine the set U_i $(1 \leqslant i \leqslant 4)$ to which each attribute in U belongs by using Algorithm 1. In Algorithm 1, T_0 is the F-based DT forest in which every F-based DT consists of only one node, and T_1 is the F-based DT forest in which the height of every F-based DT is 1. The initial values of T_0 and T_1 are assigned as empty sets in Step 1. Step 2 to Step 4 construct the F-based DT forest in which every F-based DT consists of only one node, as described in Fig. 2.

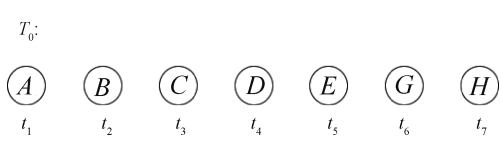


Fig. 2 F-based DT forest with each F-based DT consisting of only one node

Step 5 to Step 10 construct the F-based DT forest in which the height of every F-based DT is 1. For t_1, the attribute labeled on the root appears on the right-hand side of $BC \rightarrow A$. Therefore, the root of r_1 is labeled with A, and the leaves are labeled with B and C. For t_2, the attribute labeled on the root appears on the right-hand side of $A \rightarrow B$. Therefore, the root of r_2 is labeled with B, and the leaf is labeled with A. For t_3, the attribute labeled on the root appears on the right-hand side of $A \rightarrow C$. Therefore, the root of r_3 is labeled with C, and the leaf is labeled with A. For t_4, the attribute labeled on the root does not appear on the right-hand side of any FD in F. Therefore, no F-based DT is constructed. For t_5, the attribute labeled on the root appears on the right-hand sides of $BCD \rightarrow E$ and $G \rightarrow E$. Therefore, the root of r_4 is labeled with E, and the leaves are labeled with B, C and D. The root of r_5 is labeled with E, and the leaf is labeled with G. For t_6, the attribute labeled on the root appears on the right-hand side of $E \rightarrow G$. Therefore, the root of r_6 is labeled with G, and the leaf is labeled with E. For t_7, the attribute labeled on the root does not appear in F, so, no F-based DT is constructed.

The resulting F-based DT forest, where the height of each F-based DT is 1, is shown in Fig. 3.

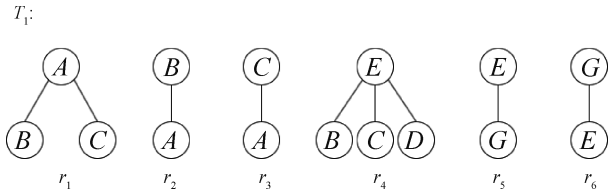


Fig. 3 *F*-based DT forest with height of each *F*-based DT being 1

Now, we can determine the set U_i $(1 \leqslant i \leqslant 4)$ to which each attribute in U belongs by using Step 11 in Algorithm 1. For D, it is labeled on the root of an F-based DT consisting of only one node and is also the label on the leaf of some other F-based DT in T_1. Therefore, D is in U_1.

For A, it is the label on the root of an F-based DT consisting of only one node and is not only the label on the leaf of some F-based DT in T, but also the label on the root of some F-based DT in T. Therefore, A is in U_3. B, C, E and G have the same characteristics as A. For H, it is the label on the root of an F-based DT consisting of only one node. Therefore, H is in U_4.

Theorem 3 Algorithm 1 can recognize the attribute which is in U_1, U_2, U_3 or U_4 exactly.

Proof In Algorithm 1, Step 8 utilizes Lemma 7 to identify attributes belonging to either U_1 or U_4, and Lemma 9 to identify attributes belonging to U_2. Since an attribute can only belong to U_1, U_2, U_3 or U_4, if an attribute does not belong to U_1, U_2 or U_4, it must belong to U_3. Therefore, in Step 8, the final decision is that the attribute belongs to U_3.

Theorem 4 The time complexity of Algorithm 1 is $O(|U| \times |F|)$, where $|U|$ and $|F|$ denote the cardinality of U and F, respectively.

Proof In Algorithm 1, Step 5 is repeated $O(|T_0|)$ times, where $|T_0| = |U|$. Step 7 is repeated $O(|F|)$ times, and since Step 7 is in the loop body of Step 5, the longest step among all the steps is Step 8. Therefore, the time complexity of Algorithm 1 is $O(|U| \times |F|)$.

Corollary 1 When an attribute belongs to U_1, U_2 or U_4, the prime attribute problem can be solved in polynomial time.

Proof According to Lemma 1, an attribute in U_1 or U_4 is a prime attribute, while an attribute in U_2 is a nonprime attribute. By utilizing Algorithm 1, we can identify whether an attribute belongs to U_1, U_2 or U_4 in polynomial time. Hence, when an attribute belongs to U_1, U_2 or U_4, the prime attribute problem can be solved in polynomial time.

Lemma 11 Let $R(U, F)$ be a relation scheme, where U is the set of attributes, F is the set of FDs and the right-hand side of each FD in F contains only one attribute. For an attribute A, if $X \rightarrow A \in F^+$, then there exists an MF-based DT with its root labeled as A and its leaves labeled with attributes in X, or there exists a subtree of an MF-based DT with its root labeled as A and its leaves labeled with attributes in X.

Proof If $X \rightarrow A \in F^+$, according to Lemma 4, there exists an F-based DT with its root labeled as A and its leaves labeled with attributes in X. Considering the definition of an F-based DT forest, all F-based DTs with their roots labeled as A are contained within the MF-based DT with its root labeled as A. Thus, the F-based DT with its root labeled as A and its leaves labeled with attributes in X can be considered as an MF-based DT or a subtree of an MF-based DT.

Theorem 5 Let $R(U, F)$ be a relation scheme, where U is the set of attributes, F is the set of FDs and the right-hand side of each FD in F contains only a single attribute. $A \in U_3$ is a prime attribute if and only if A is labeled on a node of an MF-based DT, and some attribute in X of an (X, A) -DT is labeled on the root of the MF-based DT.

Proof To prove the sufficiency of the asserted condition, we consider the definition of an (X, A) -DT. There exists an F-based DT with its root labeled as A and the set of labels of the leaves contained in the set of attributes X. By Lemma 4, for any $X \rightarrow A \in F^+$, if A is labeled on a node of an MF-based DT whose root is labeled with some attribute $B \in X$, then there exists a (V, B) -DT and $A \in V$. By Lemma 4, $V \rightarrow B \in F^+$. This shows that for $X \rightarrow A \in F^+$, there must exist an FD $V \rightarrow B \in F^+$ where $A \in V$ and $B \in X$. According to Lemma 2, A is a prime attribute.

To prove the necessity of the asserted condition, suppose that $A \in U_3$ is a prime attribute. According to Lemma 2, for $X \rightarrow A \in F^+$, there must exist an FD $V \rightarrow B \in F^+$ where $A \in V$ and $B \in X$. By Lemma 4, there must exist an (X, A) -DT and a (V, B)-DT for $X \rightarrow A$ and $V \rightarrow B$. According to the definition of an MF-based DT, the (V, B) -DT must be a subtree of an MF-based DT with its root labeled as B. Since $A \in V$ and $B \in X$, A is labeled on a node of an MF-based DT, and some attribute in X of the (X, A) -DT is labeled on the root of the MF-based DT.

Based on Theorem 5, we proposed Algorithm 2 to recognize prime attributes from U_3 by F-based DT.

Algorithm 2 Recognizing prime attributes from U_3 by F-based DT

INPUT: A relation scheme $R(U, F)$, where U is the set of attributes, F is the set of FDs and an attribute $A \in U_3$

OUTPUT: If A is a prime attribute, return "True"; otherwise return "False"

Begin

1: Construct the F-based DT forest T in which each F-based DT is maximal;
2: $f \leftarrow$ "False";
3: $i \leftarrow 1$;
4: while $i \leqslant |T|$ do {
5: if A is the label of some node in the ith MF-based DT m then
6: $j \leftarrow 1$
7: while j is the number of subtrees whose roots are labeled with A in m do {//Test subtrees whose roots are labeled with A in m one by one
8: Select the jth subtree (X, A) -DT;
9: $k \leftarrow 1$;
10: While $k \leqslant |X|$ do {
if A is the label of the node in the MF-based DT whose root is labeled with the jth attribute of X then $k \leftarrow |X| + 2$;
else $k \leftarrow k + 1$; } //If it is found that A is labeled on a node of an MF-based DT whose root is labeled with the jth attribute of X, stop the loop. Otherwise, continue until every attribute in X has been checked
11: if $k = |X| + 2$ then $j \leftarrow j + 1$
else $j \leftarrow 2+$ the number of subtrees whose roots are labeled with A in m; }//If it is found that A is not the label of the node in the MF-based DT whose root is labeled with an attribute of X, stop the loop. Otherwise, continue until every subtree whose root is labeled with A in m has been checked
12: if $j = 2 +$ the number of subtrees whose roots are labeled with A in m then $i \leftarrow |T| + 2$
else $i \leftarrow i + 1$ } //If it is found that A is not the label of the node in the MF-based DT whose root is labeled with an attribute of X, stop the loop. Otherwise, continue until every MF-based DT whose root is labeled with A in T has been checked
13: if $i = |T| + 1$ then $f \leftarrow$ "True";
14: return f;
15: **End**

Let $R(U, F)$ be a relation scheme, where U and F are the same as those in Example 1. We will use Algorithm 2 to determine whether A and G in U_3 are prime attributes.

Step 1 constructs the F-based DT forest T, where every F-based DT is maximal as shown in Fig. 1 to Fig. 3 and $|T| = 9$. When A is input into Algorithm 2, Step 5 selects the first MF-based DT containing a node labeled with A as shown in Fig. 1 (a). In Fig. 1 (a), there are four subtrees whose roots are labeled with A: (AA, A) -DT (the MF-based DT itself), (BC, A) -DT, (A, A)-DT (left) and (A, A)-DT (right). Step 7 checks them one by one. For (AA, A) -DT, Step 10 decides that A is the label of the node in the MF-based DT whose root is labeled with A, then $k \leftarrow |AA| + 2$. Next, in Step 11, $j \leftarrow 1 + 1$, which means that the second subtree (BC, A) -DT is selected. For (BC, A) -DT, Step 10 decides that A is the label of the node in the MF-based DT whose root is labeled with B, then $k \leftarrow |BC| + 2$, and $j \leftarrow 2 + 1$ in Step 11, which means that the third subtree (A, A)-DT is selected. For (A, A)-DT, Step 10 decides that A is the label of the node in the MF-based DT whose root is labeled with A, then $k \leftarrow |A| + 2$, $j \leftarrow 3 + 1$ in Step 11, which means that the fourth subtree (A, A)-DT is selected. For (A, A)-DT, Step 10 decides that A is the label of the node in the MF-based DT whose root is labeled with A, then $k \leftarrow |A| + 2$, $j \leftarrow 4 + 1$ in Step 11, since $j = 4 + 1 = 5 > 4$, Step 7 stops the while loop and proceeds to Step 12. In Step 12, since $j \leftarrow 4 + 1$, $i \leftarrow 1 + 1$, which means that the second F-based DT in T is selected, and the process repeats from Step 5 to Step 12. In fact, A is the label of the node in the MF-based DT whose root is labeled with the jth attribute of X in the subsequent (X, A) -DTs. When Step 4 stops the while loop, $i = |T| + 1 = 9 + 1$, then $f \leftarrow$ "True" in Step 13. Therefore, Algorithm 2 returns "True" in Step 14, indicating that A is a prime attribute.

For G, when Step 5 selects the seventh MF-based DT in which some node is labeled with G as shown in Fig. 1 (f), there is a subtree (BCD, G) -DT. Step 10 shows that none of the attributes B, C and D is the label of the root of an MF-based DT in which A is the label of some node. Therefore, in Step 11, $j \leftarrow 2+$ the number of subtrees whose roots are labeled with G in Fig. 1 (f). In Step 12, $i \leftarrow |T| + 2$. Since $i = |T| + 2$, the value of f remains unchanged in Step 13, namely, f is set to "False". Therefore, Algorithm 2 returns "False" in Step 14, indicating that G is a nonprime attribute.

Theorem 6 Algorithm 2 can recognize the prime attributes from U_3 exactly.

Proof In Algorithm 2, Step 10 determines whether A is the label of the node in the MF-based DT whose root is labeled with an attribute of X. Step 7 checks every subtree whose root is labeled with A in an MF-based DT. Step 4 checks every MF-based DT whose root is labeled with A. When Step 4 is completed, there are two situations: $i = |T| + 1$ or $i = |T| + 2$. If $i = |T| + 1$, then in the MF-based DT forest T, for every subtree t whose root is labeled with A, A is the label of the node which is in the MF-based DT whose root is labeled with the attribute labeled on the leaf of the subtree t. According to Theorem 5, A is a prime attribute. If $i = |T| + 2$, then A is not the label of the node in the MF-based DT whose root is labeled with an attribute of X. According to Theorem 5, A is a nonprime attribute.

Theorem 7 The time complexity of Algorithm 2 is $O(|U|^2 \times 2^{|U|})$, where $|U|$ and $2^{|U|}$ are the cardinality of U and the power set of U, respectively.

Proof In Algorithm 2, Step 4 repeats $O(|T|)$ times and $|T| = |U|$. Since the maximum number of sets of attributes that can functionally determine A is $2^{|U|}$, Step 7 is in the loop body of Step 4 and repeats $O(|U| \times 2^{|U|})$ times. Step 10 is in the loop body of Step 7 and repeats $O(|U|^2 \times 2^{|U|})$ times. Step 10 is the longest in all steps, and thus the time complexity of Algorithm 2 is $O(|U|^2 \times 2^{|U|})$.

Theorem 8 Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. When an attribute is in U_3, the prime attribute problem is NP-complete.

Proof To prove that the problem is NP-complete, we must show that it is in NP and that all NP problems are polynomial time reducible to it. First, we show that the problem is in NP. Let (X, A)-DT be a subtree whose root is labeled with A in the MF-based DT forest T. We select the set of attributes S which consists of the label of every node in the MF-based DT whose root is labeled with an attribute of X as a certificate, and test whether A belongs to S. If A does not belong to S, then A is not a prime attribute. It shows that when an attribute is in U_3, the prime attribute problem is polynomially verifiable, namely, it is in NP. To prove the second part, we show that Lucchesi and Osborn's prime attribute problem is polynomial time reducible to our prime attribute problem. By utilizing Algorithm 1, we can determine whether A belongs to U_1, U_2, U_3 or U_4 in polynomial time. As stated in Corollary 1, the prime attribute problem can be solved in polynomial time when an attribute belongs to U_1, U_2 or U_4, so we can convert Lucchesi and Osborne's prime attribute problem into our prime attribute problem by excluding the cases where A belongs to U_1, U_2 or U_4 in polynomial time.

Combining Corollary 1 and Theorem 8, we can deduce that attributes in U_3 pose the most challenging aspect of the prime attribute problem. Hence, we can determine whether an attribute is prime by considering the scenarios where the attribute is in U_1, U_2, U_3 or U_4. Drawing upon Algorithm 1 and Algorithm 2, we present Algorithm 3 for recognizing prime attributes based on the F-based DT.

Algorithm 3 Recognizing prime attributes based on F-based DT

INPUT: A relation scheme $R(U, F)$, where U is the set of attributes, F is the set of FDs and an attribute $A \in U$

OUTPUT: If A is a prime attribute, return "True"; otherwise return "False"

Begin

1: $T_0 \leftarrow \emptyset$; $T' \leftarrow \emptyset$;

2: for $i = 1$ to $|U|$ do {

3: $t_i \leftarrow$ a tree consisting of a single node labeled with the ith attribute in U;

4: $T \leftarrow T \cup \{t_i\}$; } //Construct the F-based DT forest with each F-based DT having a single node

5: for $i = 1$ to $|T|$ do {

6: $k \leftarrow 1$; for $j = 1$ to $|F|$ do {

7: if the attribute labeled on the root of t_i appears on the right-hand side of the jth FD in F,
then construct an F-based DT r_k with the root labeled as the attribute on the root of t_i and the leaves labeled with the attributes appearing on the left-hand side of the jth FD in F;

8: $T \leftarrow T' \cup \{r_k\}$; }

9: $k \leftarrow k + 1$; } //Construct the F-based DT forest, with the leaves of each F-based DT's root consisting of the children of the root

10: if A is the label on the root of an F-based DT consisting of a single node and is also the label on the leaf of some other F-based DT in T', then return "True";
else if A is the label on the root of an F-based DT consisting of a single node, then return "True";
else if A is not the label on the leaf of any F-based DT in T', then return "False";
else call Algorithm 2;

11: **End**

3 Time Efficiency Analysis

Let $R(U, F)$ be a relation scheme, where U is the set of attributes and F is the set of FDs. The prime attribute problem is NP-complete. Traditional algorithms, particularly those based on key enumeration[1,14], typically require computing all candidate keys prior to determining prime attributes. These methods incur a worst-case time complexity of $O(2^{|U|} \times |F|)$:

$$O(2^{|U|} \times |F|) = 2^{3|U|}.$$

To address this inefficiency, Kundu[9] proposed an improved method that avoided full key enumeration, thereby reducing the time complexity to $O(2^{|U|-1} \times |F|)$:

$$O(2^{|U|-1} \times |F|) = 2^{3|U|-1}.$$

Our research is based on mapping the features of prime attributes to the F-based DT forest, which offers the potential for further optimization of time complexity. When an attribute belongs to U_1, U_2 or U_4, the problem is in the P-class.

As stated in Theorem 4 and Theorem 7, the time complexity of Algorithm 1 is $O(|U| \times |F|)$, while Algorithm 2 incurs a higher time complexity of $O(|U|^2 \times 2^{|U|})$. In Algorithm 3, Step 1 to Step 9 and the first three conditional branches in Step 10 reuse the structure and logic of Algorithm 1. Therefore, when an attribute belongs to U_1, U_2 or U_4, the time complexity of Algorithm 3 is $O(|U| \times |F|)$. However, when an attribute belongs to U_3, Algorithm 2 is invoked, resulting in a time complexity of $O(|U|^2 \times 2^{|U|})$. Consequently, the worst-case time complexity of Algorithm 3 is $O(|U|^2 \times 2^{|U|})$:

$$O(|U|^2 \times 2^{|U|}) = 2^{|U|-2} \times |U|^2.$$

For a prime attribute problem, Algorithm 3 proposed in this study can solve the problem in polynomial time when an attribute is in U_1, U_2 or U_4. When an attribute is in U_3, the algorithm still achieves a lower time complexity compared to conventional methods. Experimental results demonstrate that the proposed algorithm significantly improves the efficiency of prime attribute recognition.

4 Conclusions

The features of prime and nonprime attributes are presented[19]. To effectively recognize prime attributes, we propose an algorithm for recognizing prime attributes based on F-based DT. Through F-based DT, the occurrence of attributes in the FDs is mapped to an F-based DT forest and the features of attributes in U_1, U_2, U_3 or U_4 are represented in the F-based DT forest. By the construction of F-based DT, Algorithm 3 can recognize prime attributes among attributes in both the complex subset (U_3) and the straightforward subsets (U_1, U_2 or U_4). The algorithm not only significantly reduces the exponential time complexity of traditional methods but also improves the efficiency and reliability in prime attribute recognition. When the attribute belongs to U_3, determining whether it is a prime attribute becomes NP-complete.

In future work, we will explore the application of this method to determine normal forms and address key-related problems. The prime attribute problem has a negative impact on the process of determining normal forms. However, with our algorithm, membership between an attribute and one of U_1, U_2, U_3 and U_4 can be easily determined, and it is possible to determine whether an attribute is a prime attribute when it is in U_1, U_2 or U_4 in polynomial time. Therefore, we will investigate the relationship between attributes in U_3 and normal forms.

References

[1] Lucchesi C L, Osborn S L. Candidate keys for relations[J]. *Journal of Computer and System Sciences*, 1978, 17(2): 270-279.

[2] Cordero P, Enciso M, Mora A. Automated reasoning to infer all minimal keys[C]//IJCAI. Palo Alto: IJCAI, 2013: 817-823.

[3] Demba M. KeyFinder: an efficient minimal keys finding algorithm for relational databases[J]. *Inteligencia Artificial*, 2021, 24(68): 37-52.

[4] Demetrovics J, Thi V D. Relations and minimal keys[J]. *Acta Cybernetica*, 1988, 8(3): 279-285.

[5] Fernandez M, Varga J. Finding candidate keys and 3NF via strategic port graph rewriting[C]// Proceedings of the 22nd International Symposium on Principles and Practice of Declarative Programming. New York: ACM, 2020: 1-14.

[6] Fadous R, Forsyth J. Finding candidate keys for relational data bases[C]//Proceedings of the 1975 ACM SIGMOD International Conference on Management of Data. New York: ACM, 1975: 203-210.

[7] Feng Y C. Algorithm for solving candidate key by graph theory[J]. *Chinese Journal of Computers*, 1988, 11(9): 556-558. (in Chinese)

[8] Hao Z X, Liu G H. An algorithm to find out all candidate keys of relation scheme[J]. *Chinese Journal of Computers*, 1991, 14(4): 300-307. (in Chinese)

[9] Kundu S. An improved algorithm for finding a key of a relation[C]//Proceedings of the Fourth ACM SIGACT-SIGMOD Symposium on Principles of Database Systems. New York: ACM, 1985: 189-192.

[10] Mannila H, Raiha K J. Practical algorithms for finding prime attributes and testing normal forms [C]//Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. New York: ACM, 1989: 128-133.
[11] Saiedian H, Spencer T. An efficient algorithm to compute the candidate keys of a relational database scheme [J]. *The Computer Journal*, 1996, 39(2): 124-132.
[12] Wastl R. Linear derivations for keys of a database relation scheme [J]. *Journal of Universal Computer Science*, 1998, 4(11): 883-897.
[13] Yu C T, Johnson D T. On the complexity of finding the set of candidate keys for a given set of functional dependencies [J]. *Information Processing Letters*, 1976, 5(4): 100-101.
[14] Liu G H, Hao Z X, Chen Z J. A quick replacing algorithm for finding all candidate keys of a relation scheme [J]. *Chinese Journal of Computers*, 1998, 21(10): 890-895. (in Chinese)
[15] Hao Z X, Guo J F. A hypergraph-based method for finding out all candidate keywords in relational schemes [J]. *Chinese Journal of Computers*, 1992, 15(4): 264-270. (in Chinese)
[16] Hao Z X, Liu G H, Liu C L. A method for finding all candidate keys of relational schema based on attributes relative table [J]. *Journal of Computer Research and Development*, 1994, 31(6): 6-13. (in Chinese)
[17] Bahmani A H, Naghibzadeh M, Bahmani B. Automatic database normalization and primary key generation [C]//Canadian Conference on Electrical and Computer Engineering (CCECE). Piscataway, NJ: IEEE, 2008: 11-16.
[18] Bordoloi S, Kalita B. Designing graph database models from existing relational databases [J]. *International Journal of Computer Applications*, 2013, 74(1): 25-31.
[19] Xu Y L, Liu G H, Yu X X, et al. Features of prime attributes in a relation scheme[J]. *Journal of Donghua University (English Edition)*, 2025, 42(6): 689-698.
[20] Beeri C, Bernstein P A. Computational problems related to the design of normal form relational schemas [J]. *ACM Transactions on Database Systems*, 1979, 4(1): 30-59.
[21] Hodel R E. An introduction to mathematical logic [M]. New York: Dover Publications, 2013.
[22] Nakos V, Ngo H Q, Tsourakakis C E. Targeted least cardinality candidate key for relational databases [PP/OL]. arXiv (2024-08-24) [2025-03-21]. https://arxiv.org/abs/2408.13540.

基于 *F*-based 推导树的主属性识别算法

于晓雪，刘国华*，徐雨露，刘长奇，张丽萌，朱东燕，何松达
东华大学 信息与智能科学学院，上海 201620

摘　要：尽管主属性问题是非确定性多项式时间完全（nondeterministic polynomial-time，NP-complete）问题，但判断一个属性是否为主属性的难度与该属性在函数依赖集中的出现方式密切相关。本文根据属性在函数依赖集中出现的特征对主属性问题进行深入研究。首先，在关系模式 $R(U, F)$ 中，根据属性在函数依赖集 F 中的出现情况，将属性集 U 划分为四个子集：U_1（仅出现在 F 左部的属性）、U_2（仅出现在 F 右部的属性）、U_3（同时出现在 F 左右部的属性）和 U_4（在 F 中未出现的属性）。其次，将属性在函数依赖集中的出现情况映射到基于函数依赖的推导树（*F*-based derivation tree，*F*-based DT）森林，在 *F*-based DT 森林展示出在 U_1、U_2、U_3 或 U_4 中属性的特征。再次，提出了一种基于 *F*-based DT 的主属性识别算法。最后，得出以下结论：当属性属于 U_3 时，主属性判定问题是 NP 完全；当属性属于 U_1、U_2 或 U_4 时，该问题属于多项式时间类（polynomial-time complexity class，P-class），即可以在多项式时间内高效求解。相较于传统方法，本文提出的算法能够显著地简化主属性的识别过程，并促进数据库规范化。

关键词：*F*-based 推导树；主属性；NP 完全；规范化