A New Characterization of Independence

Peter Honeyman

Bell Laboratories
Murray Hill, New Jersey 07974

Edward Sciore†

SUNY Stony Brook
Long Island, New York 11794

ABSTRACT

We introduce a restriction on the structure of a database scheme, called the *primary key condition*, and show that this condition characterizes independent database schemes when constraints are presented as keys. The primary key condition provides added insight into the structure of independent schemes, and leads to a general design methodology. We describe a linear-time algorithm for recognizing independent schemes.

1. Introduction.

Database theory has been able to identify certain database schemes as having desirable properties. Among these properties is the *independence* condition; this provides the ability to check and maintain integrity constraints given by functional dependencies in an efficient way‡. The *uniqueness condition* [S], the only known characterization of independent database schemes, has two drawbacks. First, the uniqueness condition stresses what the scheme should not have, instead of what it *ought* to have, thus it adds little insight to guide a designer when deriving an independent scheme. Second, it is difficult to understand the structure of an independent scheme using the uniqueness condition.

In this paper, we present a new characterization of independence, called the *primary key condition*, one that is radically different from the uniqueness condition. It states that each relation scheme must have a designated primary key, and that these keys must interact in a specific way. In Section 3, we show that the primary key condition holds exactly when a scheme is independent.

The existence of a designated primary key provides new insight into the structure of independent schemes. In Section 4, we consider the problem of recognizing independent schemes; we show how an algorithm of Beeri and Bernstein [BB] can be used to obtain a simple linear-time recognition algorithm. In Section 5, we discuss semantic implications of the primary key condition. We show that the existence of non-primary keys has no effect on the independence of a scheme; thus one can add as many other keys as one likes to an

† E. Sciore's work was supported in part by NSF grant IST81-09824.

‡ We assume all functional dependencies are given as candidate keys for relation schemes.

independent scheme without changing its independence. We also discuss how selecting primary keys can guide a database design methodology, and how the methodology relates to the "classical" entity-association approach.

2. Preliminaries.

We assume that the reader understands the relational concepts *attribute, relation, database, key,* and *functional dependency* (fd). A *relation scheme* is a set of attributes with an candidate set of keys. Let R be a set of attributes with associated keys $\{K_1, \ldots, K_n\}$; we write the relation scheme $\langle R, \{K_1, \ldots, K_n\}\rangle$. When the keys are not relevant, we write the relation scheme simply as R. A *database scheme* is a finite set of relation schemes.

A relation scheme $\langle R, \{K_1, \ldots, K_m\}\rangle$ is said to *embody* the fd's $F_R=\{K_1 \rightarrow R, \ldots, K_m \rightarrow R\}$. The database scheme $\{R_1, \ldots, R_n\}$ embodies the fd's $F=\bigcup F_{R_i}$. The *closure* of R_i, written $CLOSURE(R_i, F)$, is the set of attributes that are implied by R_i using F. We also write the closure of R_i as R_i^+. R_i is said to be a *database key* if $R_j \subseteq R_i^+$ for each j.

A database is *locally satisfying* if each relation satisfies its embodied keys. A locally satisfying database may have global consistency problems. We say that a database $r_1, \ldots, r_n$ is *globally satisfying* if there is a relation I defined over all attributes such that I satisfies all embodied fd's, and each r_i is contained in the projection of I on R_i. We shall not attempt to justify this formalism here, which relies on the *universal relation scheme assumption* [FMU]; the interested reader should consult [HI, S].

A database scheme is *independent* if every locally satisfying database is also globally satisfying. Thus, changes to the database can occur to relations individually, and no global constraint checking need be done; this makes independence a powerful and important property of a database scheme. Sagiv characterized independence using the *uniqueness condition* [S]. A database scheme $\{R_1, \ldots, R_n\}$ is said to satisfy the uniqueness condition if for all R_i and R_j there do not exist attributes KA in R_j such that K is a key of R_j, $A \notin K$, and $KA \subseteq CLOSURE(R_i, F-F_j)$.

It has been shown that a database scheme is independent iff it satisfies the uniqueness condition [S]. In the next section we provide a different characterization of independence.

3. The Primary Key Condition.

We begin by assuming that our database scheme contains a database key. Later in this section we shall consider the general case.

Definition. Let $\{R_1, \ldots, R_n\}$ be a database scheme containing a database key. Then $\{R_1, \ldots, R_n\}$ satisfies the *primary key condition* if there exists a key K_i of each R_i and a total ordering ("<") of $\{R_1, \ldots, R_n\}$ such that if $R_i < R_j$ then $R_i \cap R_j \subseteq K_j$. K_i is called the *primary key* of R_i. □

Example. Consider the relation schemes $R_1=\langle ACD, ACD\rangle$, $R_2=\langle BCD, \{CD, BD\}\rangle$, and $R_3=\langle ABC, ABC\rangle$. The database scheme $\{R_1, R_2, R_3\}$ satisfies the primary key condition; choose CD as the primary key of R_2, and let $R_1 < R_2 < R_3$. □

Example. Consider the relation schemes $R_1=\langle ABC, AB\rangle$, $R_2=\langle CDE, CD\rangle$, $R_3=\langle ADE, DE\rangle$, and $R_4=\langle BDEF, BDE\rangle$. The database scheme $\{R_1, R_2, R_3, R_4\}$ does not satisfy the primary key condition. There is no way to arrange R_1, R_2, and R_3 properly in a total order. □

We note that the first example satisfies the uniqueness condition, whereas the second example does not. In general, we can show that if a database scheme is independent, then it satisfies the primary key condition. This result follows from the following theorem.

Theorem 1. If $\{R_1, \ldots, R_n\}$ contains a database key and satisfies the uniqueness condition, then it satisfies the primary key condition.

Proof. Suppose $\{R_1, \ldots, R_n\}$ satisfies the uniqueness condition and contains a database key. The proof that $\{R_1, \ldots, R_n\}$ satisfies the primary key condition is by induction on n. We will order $\{R_1, \ldots, R_n\}$ so that $R_1 < R_2 < \cdots < R_n$. Let R_1 be the database key.

Basis: $n = 2$. Let us denote by I the intersection of R_1 and R_2. Since R_1 is a database key, I must functionally determine R_2. It follows, then that I contains a key for R_2. We designate this key as K_2, the primary key

for R_2. All that remains is to show that I is contained in some key of R_2, i.e., that I is identical to K_2. Suppose otherwise and let A be an attribute in I but not in K_2. Since K_2A is in $CLOSURE(R_1, F_2)$, K_2A violates the uniqueness condition, violating our initial assumption. Note that we may pick the primary key for R_1 arbitrarily. This completes the basis step.

Induction: Suppose the implication is true for database schemes with fewer than n relation schemes, and let $\{R_1, \ldots, R_n\}$ be a database scheme that satisfies the uniqueness condition. Consider the process of calculating $CLOSURE(R_1)$. (Recall that R_1 is a database key.) Let the first dependency used be from some key K; call the scheme for which K is the key R_2 and let K be the designated primary key for R_2. As in the basis step, R_1 and R_2 satisfy the requirements of the uniqueness condition. Consider now the scheme $\{R_{1,2}, R_3, \ldots, R_n\}$ in which we replace R_1 and R_2 by their union, denoted $R_{1,2}$.

We claim that $\{R_{1,2}, R_3, \ldots, R_n\}$ satisfies the uniqueness condition, for suppose it does not. Then some KA is contained in some R_j such that K is a key of R_j, $A \notin K$, and $KA \subseteq CLOSURE(R_i, F-F_j)$ for some i. We assumed that $\{R_1, \ldots, R_n\}$ satisfies the uniqueness condition, so either R_i or R_j must be $R_{1,2}$.

Suppose $R_{1,2}$ plays the role of R_i. Since $R_{1,2} \subseteq CLOSURE(R_1, F-F_j)$, KA and R_1 violate the uniqueness condition. On the other hand, if $R_{1,2}$ plays the role of R_j, then K must also be a key for either R_1 or R_2, and again the uniqueness condition is violated in $\{R_1, \ldots, R_n\}$. Thus, $\{R_{1,2}, R_3, \ldots, R_n\}$ satisfies the uniqueness condition and, by the induction hypothesis, the primary key condition.

To complete the proof, we see from the above argument that relations $R_3, \ldots, R_n$ satisfy the primary key condition. Suppose, then, that the intersection of R_1 and some R_i is not contained in K_i. Then $R_{1,2}$ and R_i must also violate the primary key condition, contradicting the proof that $\{R_{1,2}, R_3, \ldots, R_n\}$ satisfies the primary key condition. Since the same holds for R_2, we have shown that $\{R_1, \ldots, R_n\}$ satisfies the primary key condition. □

The converse to this theorem also holds.

Theorem 2. If $\{R_1, \ldots, R_n\}$ satisfies the primary key condition, then it satisfies the uniqueness condition.

Proof. Suppose that $\{R_1, \ldots, R_n\}$ is ordered according to the primary key condition, and that it violates the uniqueness condition. Then there exists R_i and R_j and attributes KA in R_j such that K is a key of R_j and $KA \subseteq CLOSURE(R_i, F-F_j)$. Choose j such it is the largest of all such (i,j) pairs. Now there must exist some attribute B that is in KA but not in K_j. Let l be the relation scheme that adds B to the closure of R_i. Since B is not in K_j, $i<l$ by the primary key condition. Since B is added to the closure of R_i, there must exist a key K' in R_l such that the key dependency $K' \rightarrow B$ is used to compute the closure of R_i. Since all dependencies in F_l are key dependencies, we know that only one of these dependencies can be used in calculating the closure of R_i; so it must be $K' \rightarrow B$, and $K'B \subseteq CLOSURE(R_i, F-F_l)$. Thus R_l and R_i violate the uniqueness condition. Since $l>j$, we have a contradiction. □

Theorems 1 and 2 establish the equivalence of the uniqueness condition and the primary key condition whenever the database scheme has a database key. We now show how to extend the primary key condition to remove this restriction.

Definition. Let $\{R_1, \ldots, R_n\}$ be a database scheme. Then for each $i \leq n$ we define $S_i = \{R_j | R_j \subseteq R_i\}$. □

Thus for each i, S_i is the database scheme consisting of all relation schemes in the closure of R_i. R_i is a database key for S_i.

Definition. $\{R_1, \ldots, R_n\}$ satisfies the *extended primary key condition* if for every i, S_i satisfies the primary key condition. □

The above definition implies that the selection of primary keys and choice of ordering may vary for each S_i. Also note that we need to consider only "maximal" S_i, since if $R_j \subseteq R_i$ and S_i satisfies the primary key condition, then the same keys and ordering for S_i will work for S_j.

Example. Consider the relation schemes $R_1 = \langle ABD, AD \rangle$, $R_2 = \langle BCE, CE \rangle$ and $R_3 = \langle ABC, \{AB, BC\} \rangle$. $S_1 = \{R_1, R_3\}$ and $S_2 = \{R_2, R_3\}$. For S_1 we must choose $K_3 = AB$; for S_2 K_3 must be BC. □

Theorem 3. The database scheme $\{R_1, \ldots, R_n\}$ satisfies the extended primary key condition iff it satisfies the uniqueness condition.

Proof. Suppose that $\{R_1, \ldots, R_n\}$ violates the uniqueness condition. Then there exist relation schemes R_i and R_j as in the proof to Theorem 2. Since $R_j \subseteq R_i^+$, both schemes are in S_i. From Theorem 2, we know that S_i cannot satisfy the primary key condition; thus $\{R_1, \ldots, R_n\}$ cannot satisfy the extended primary key condition.

Now suppose that $\{R_1, \ldots, R_n\}$ violates the extended primary key condition. Then there is some S_i that violates the primary key condition; by Theorem 1, the uniqueness condition is violated for S_i and thus for $\{R_1, \ldots, R_n\}$. □

Since the extended primary key condition is an extension of the (regular) primary key condition, and Theorem 3 is a similar extension to Theorems 1 and 2, we shall ignore the distinctions, and refer to both conditions as the *primary key condition*.

4. Recognizing Independent Schemes.

We begin by considering schemes for which some scheme is a database key, say R_1. The test for independence relies on the computing the closure of R_1. We will use a naive algorithm to describe the test.

The straightforward way to compute the closure is to begin with R_1 and iteratively check whether the left side of some dependency $X \rightarrow Y$ is contained in the closure so far; if so, the Y is added to the closure. When no more attributes can be added, the result is $CLOSURE(R_1, F)$.

Consider the first dependency $X \rightarrow Y$ that allows us to add attributes to the closure. We argued in the proof of Theorem 1 that this dependency arises from the existence of a key for some scheme, which we call K_2 and R_2 respectively, and furthermore that the intersection of R_1 and R_2 must be identical to K_2. The next step in computing the closure is to find another dependency whose left side is contained in $R_1 \cup R_2$ and add the attributes of the left side. To continue to maintain the requirements of the induction step of the proof, the left side of the dependency used must be equal to the intersection of $R_1 \cup R_2$ and the scheme from which the dependency was taken.

Thus to determine whether $\{R_1, \ldots, R_n\}$ is independent, we compute the closure of R_1, checking that each time a dependency is used, the left side is equal to the intersection of the scheme from which the dependency is derived and the closure of R_1 so far. In addition, we require that every relation scheme be used in calculating the closure. The following algorithm implements the independence test.

```
closure ← database key;
remaining ← {R1, . . . ,Rn} − {database key};
while remaining is not empty
    for each R in remaining
        if closure∩R is equal to a key of R
            add R to closure;
            remove {R} from remaining;
        else if closure∩R contains a key of R
            reject {R1, . . . ,Rn};
            halt;
```

If there is no database key, we apply to the test to each maximal S_i, as defined in Section 3. Since we can compute closures in linear time [BB], the time complexity for testing independence is linear for each maximal S_i.

5. Semantic Issues.

The primary key condition has two aspects: each relation scheme must have a primary key declared for it, and the relation schemes must have an appropriate total order. Since this order depends only on the primary keys, it follows that the existence of any other keys is irrelevant. That is, once a scheme is determined to be independent, the addition of other secondary keys will not change things. Thus secondary keys do not provide any structural information; their only purpose is to serve as integrity constraints. This result mirrors the sentiment behind [FMU], in which functional dependencies are classed as "structural" and "incidental".

When the primary key condition holds, every attribute in a relation scheme has a specific role. The attributes in the primary key serve to identify the relation scheme; all references to the scheme will be through these attributes. Any attribute not in the primary key of a relation scheme either appears nowhere else or refers to the primary key of

another ("higher") relation scheme. Thus there are three possible kinds of attributes in a database scheme: identifying (primary key) attributes, property (unique) attributes, and foreign key (referencing) attributes.

In the context of the entity/association design methodology, this classification appears as follows. An entity relation contains only identifying and property attributes; attributes in entity relations are disjoint from one another. Attributes are related via associations. A (many-many) association is a relation containing primary keys of entities, and perhaps some property attributes. The primary key of the association is the union of the component entity keys.

Every database scheme composed of entities and many-many associations trivially satisfies the primary key condition, since common attributes will always appear in primary keys. The construct that complicates things is the many-one association. A many-one association differs from a many-many association in that some of the component keys are not in the primary key of the association. Thus some non-primary key attributes are foreign keys, and appear in other relations. Consider one such attribute A in relation R. The primary key condition requires that A appear as a non-primary key attribute only in R; moreover, there must exist an ordering on the relations such that R is the first relation containing A. As long as we design schemes with this constraint in mind, we are guaranteed independence.

A similar approach appears in the methodology of RM/T [C]. Central to RM/T is the notion of a *surrogate*, which is a system-defined entity identifier that serves as the primary key of every relation. In our approach, it is simple enough to require that all entities and associations contain surrogates. However, in order for the primary key condition to hold, we cannot assume that a surrogate will be the primary key of the relation. This separation of surrogates as entity identifiers from their role as primary keys appears to be the only crucial distinction between independent schemes and RM/T schemes. Of course, the primary key condition (and independence) is based on a universal relation assumption, whereas RM/T is not. Nevertheless, it is interesting to see how close the methodologies are.

The notion of a primary key is not new. The argument for primary keys has always centered on implementation concerns; a primary key was necessary as an identifier that could not contain null values. Theoretically, it never has been clear how necessary primary keys are. The results of this paper show that there is no loss of generality in assuming the existence of a primary key; more to the point, schemes without them are poorly designed.

References

[BB] Beeri, C. and P. Bernstein, "Computational Problems Related to the Design of Normal Form Relation Schemes." ACM TODS (4:1) March 1979, pp. 241–259.

[C] Codd, E. "Extending the Data Base Relational Model to Capture More Meaning." ACM TODS (4:4) Dec. 1979, pp. 397–434.

[FMU] Fagin, R., A. Mendelzon and J. Ullman, "A Simplified Universal Relation Assumption and Its Properties", ACM TODS (7:3) Sept. 1982, pp. 343–360.

[H1] Honeyman, P., "Testing Satisfaction of Functional Dependencies", J. ACM (29:3) July 1982, pp. 668–677.

[H2] Honeyman, P., "Extension Joins", Proc. VLDB 1980, pp. 239–244.

[S] Sagiv, Y., "A Characterization of Globally Consistent Databases and their Correct Access Paths", Tech. Rep., U. Illinois, 1981.